Pereira *et al. Journal of Internet Services and Applications* (2015) 6:24
DOI 10.1186/s13174-015-0039-z

Journal of Internet Services and Applications
a SpringerOpen Journal

EXPERIENCE REPORT Open Access

CitySDK Tourism API - building value around open data

Ricardo Lopes Pereira[1,2*], Pedro Cruz Sousa[1], Ricardo Barata[1], André Oliveira[3] and Geert Monsieur[4]

Abstract

Tourism is a major social and cultural activity with relevant economic impact. In an effort to promote their attractions with tourists, some cities have adopted the open-data model, publishing touristic data for programmers to use in their own applications. Unfortunately, each city publishes touristic information in its own way.

A common Application Programming Interface (API) for accessing this information would enable applications to seamlessly use data from several cities, increasing their potential market while reducing the development costs. This would help developers in making cross-city applications, lowering the overhead of supporting new cities and providing them with increased exposure. Finally, tourists will also benefit from better and cheaper applications due to the boosted competition.

This paper provides an overview of the design, deployment and utilization of the CitySDK Tourism API, which aims to provide access to information about Points of Interest, Events and Itineraries. It was designed in order to be used by municipalities, regional or national governments as well as other public or private entities interested in publishing touristic information. The API comprehends a delegation model, allowing applications to access worldwide information by only knowing a single API endpoint.

The API was created and validated in the context of the CitySDK project, through which a server reference implementation, client libraries and a set of demonstration applications have also been made available. The API is currently available for the cities of Amsterdam, Helsinki, Lamia, Lisbon and Rome. Several companies have developed mobile applications that use this API.

Keywords: Smart cities, Tourism, API, Open data

1 Introduction

1.1 Motivation

Tourism is a very important social, cultural and economic activity. According to the World Tourism Organization, in 2014 tourism was responsible for 9 % of the world's Gross Domestic Product (GDP) and for 1 in every 11 jobs[1]. Tourism generated over 1.5 trillion US$ in exports and accounts for 6 % of the world trade and 30 % of the services exports. The number of international tourists reached 1.135 billion in 2014 and is expected to continue growing at an average rate of 3.3 % a year until 2030 [1]. More than half of international tourists elect Europe as their destination.

As vacation time is limited and tourism is a costly activity, tourists wish to make the most of their stay. There is an industry around travel guides, maps and advice. This business is also being explored on the Internet and is now making its way to the ubiquitous smartphone, where it can take advantage of interactivity, positioning systems, wireless Internet access, augmented reality, social networks and crowd-sourcing. However, often the foundation for tourism applications continues to be accurate, high quality, reliable information from authoritative sources.

National, regional and city authorities compile large amounts of information to use in their internal processes. Municipalities understand the value of these data and many have gone through a multi-step process to share it with tourists in order to improve their experience and attract them to their cities. First, municipalities created

*Correspondence: ricardo.pereira@inesc-id.pt
[1] Instituto Superior Técnico, Universidade de Lisboa, Avenida Professor Cavaco Silva, 2744-016 Porto Salvo, Portugal
[2] INESC-ID, Avenida Professor Cavaco Silva, Edifício IST, 2744-016 Porto Salvo, Portugal
Full list of author information is available at the end of the article

applications (web or mobile) for sharing data with the tourists. This approach is costly and unsuitable for most cities: the number of visitors is not large enough to recuperate the investment; municipalities are not software houses, being unable to keep up with the pace of innovation. Furthermore, municipalities are limited in the types of applications they can provide: e.g. publishing negative opinions written by users about an attraction could expose them to liability.

Making data available to third parties is often a better investment and many entities have followed the open-data path in a second step. With access to data, programmers bear the costs and risk, but are free to integrate data from several sources to create novel applications. Market forces should drive innovation, creating the applications tourists want. By publishing data openly, municipalities may further contribute to local economic growth by aiding in the creation of local businesses exploiting this data [2]. Still, the open-data model is not without flaws as each entity publishes different datasets using different data formats. For instance, the municipality of Lisbon publishes open data sets under several formats: Excel spreadsheets, CSV files (each with its own structure and semantics), Webservices (each with its own Application Programming Interface (API))[2]. Additionally, these are very difficult to merge, as even location information is different with some sources using street addresses while others rely on coordinates. The city of Helsinki also does the same [3], as do many others. Programmers are forced to invest into dealing with the particularities of each data representation format, thus limiting the number of data sources that can be included into an application. These factors will limit the breadth of data and the number of cities covered by each application and thus its potential market, limiting the number and size of the investments. Also, the local nature of the applications will make them difficult for tourists to find, as they must discover the particular application for each city visited.

The path taken by municipalities has also been walked by other entities related to tourism, such as national and regional governments, museums, concert halls or cultural events organizers.

1.2 CitySDK

If the same touristic data was made available in a single format by several entities, programmers would be able to reach larger audiences with a smaller investment. This would increase competition and tourists would benefit from a wider choice of applications.

Smart City Service Development Kit and its application Pilots (CitySDK) was an European European Information and Communication Technologies (ICT) Policy Support Programme (PSP) project involving 29 partners from 9 countries, running from January 2012 to October 2014. One of the most important goals of CitySDK was to create an ecosystem in which the work of an application developer is facilitated by having unified and open data interfaces available across different cities in Europe. This means that it should be relatively easy for developers to make use of touristic data coming from multiple European cities, because in such an ecosystem data access is open and unified. Developers will be able to use this information and create useful, innovative applications that use it in new ways and combine it with other sources of information.

In the scope of CitySDK three APIs were designed and deployed: one for participation services (e.g. FixMyStreet), one for mobility data (e.g. public transport data) and one for touristic information. In this paper, we present the *CitySDK Tourism API*, show how it addresses the problems that municipalities and developers face, present the dissemination and bootstrapping efforts and discuss the lessons learned in the process. The CitySDK Tourism API enables access to information about Points of Interest (POIs), events and thematic itineraries. It can be implemented by municipalities (the main focus in the scope of the CitySDK project), other government levels and other private or public organisations such as museums or concert halls. Endpoints for the CitySDK Tourism API are currently available for the cities of Amsterdam, Helsinki, Lamia, Lisbon and Rome.

Tourists stand to gain the most from the availability of the CitySDK Tourism API, even though they will never need to interact directly with it, or even know it exists. They will be able to choose the applications that best suites their needs and use it in different locations. Applications can provide functionality to replace or complement most information providing artifacts used by tourist today, such as travel guide or audio-guides.

1.3 Requirements

The CitySDK Tourisms API strives to fulfill the needs of tourists, developers and potential data providers. Tourist need state of the art applications that provide access to the tourist information they need, together with integration with popular applications such as social networks. Data providers (such as business and municipalities) need a standard way to provide tourist information, capable to express their current data sets as well as those foreseen in the near future. Furthermore, the API needs to be flexible and expansible enough to evolve to support future, unforeseen needs. Developers require an API that is easy to use for application of any scale, local or global.

Furthermore, the API should fulfill the following requirements: be based on standards in order to facilitate acceptance; be distributed, so to limit the scope of each participant; be scalable; provide for both static and fast updating data; and enable offline use by caching the data.

The CitySDK project also was intent on creating the conditions for a development ecosystem to emerge around tourism information. The API should provide access to quality data without hindering the business opportunities for developers. Furthermore, data providers should not be exposed to liability by being responsible for the combination of their data with crowd sourced information, a task left for developers as data providers will have limited resources, preventing them from curating crowd-source data.

1.4 Document structure

This document is organised into seven sections. In the next section, an overview of other efforts for providing normalised access to city data is provided. Section 3 details the CitySDK Tourism API. The roll-out of the API endpoints in the cities is presented in Section 4. Section 5 describes the efforts carried out for promoting and facilitating the adoption of the API. Opportunities and adoption examples of the API are presented in Section 6, which also describes the lessons learned. Finally, Section 7 presents the conclusions and describes future work.

2 Related work

The CitySDK Tourism API should be based on existing standards, making use of best practices and the experience of others, in order to increase its chances of adoption. The need for a common data representation or APIs for tourism data has been present for some time. In this section we present two major normalization efforts in this field: EventsML-G2 and World Wide Web Consortium (W3C) POI Working Group (WG). We then analyze other work aiming to provide access to touristic information.

2.1 EventsML-G2

EventsML-G2 is a standard for collecting and distributing structured event information. It is aimed at conveying event information in a news industry environment [3]. This standard is a member of the family of the International Press Telecommunications Council (IPTC) G2-Standards, built on a structural and function framework called the IPTC News Architecture (NAR), and shares many of its components with the other standards of this family. Additionally, the EventsML-G2 makes use of well-known industry standards, since its syntax is built on W3C's XML Schema and fully complies with the basic notion of the Semantic Web, the Resource Description Framework (RDF). One of the main features of the EventsML-G2 standard is its ability to transmit information (i.e. facts) about a specific event. Its comprehensiveness and extensibility makes the standard suitable for covering a large magnitude of event types and cover multiple facts about a specific event either by literal text (i.e. free text) or by codes from specified vocabularies.

Although EventsML-G2 can convey information about a POI where an event takes place, that is not its focus. Moreover, as it strives to represent all types of events, EventsML-G2 is a complex standard, making it difficult to implement and use.

2.2 W3C Point of Interest WG

The W3C set up the POI WG with the mission to develop technical specifications for the representation of POI information on the web [4]. Its Core Recommendation draft defines a generic, flexible, lightweight and extensible POI data model, and one normative syntax for the data model based on Extensible Markup Language (XML). Although XML is the primary model for this specification, other formats are also possible, such as JavaScript Object Notation (JSON).

The data model is shown in Fig. 1. It comprises six entities:

- **POIBaseType** is the common entity from which the majority of POI entities are derived. It provides basic properties related with its authorship, licensing, modification dates and identification allowing each element to carry distinct information;
- **POITermType** is an abstract entity derived from POIBaseType and adds properties for the management of categorical descriptions (such as the ones seen in category), link, label, author, license and time properties of POIType;
- **POIType** is an abstract entity derived from POIBaseType and adds entities for describing, labeling, categorizing and indicating the time span of a POI or group of POI. This entity also includes linking elements to other POIs, external web resources or metadata;
- **Location** is an entity that inherits from POIBaseType and provides a flexible description of the location of a POI. A Location can be represented using geodetic coordinates for the center of the POI, line, polygon, civic address, undetermined (representing unresolved locations) or bounding box (relationship element);
- **POI** inherits from POIType and adds the Location entity for describing the location of the POI;
- Finally, **POIS** also derives from POIType and can have one or more children entities of type POI.

This model is flexible and extensible enough to be used within CitySDK Tourism API to model the various types of data (POI, events and thematic itineraries) required. Its use is described in Section 3.1.

2.3 Tourism open data efforts

Citadel on the Move was an EU funded project which aimed to facilitate the creation of innovative mobile application that use open data[4]. Citadel focused on the creation

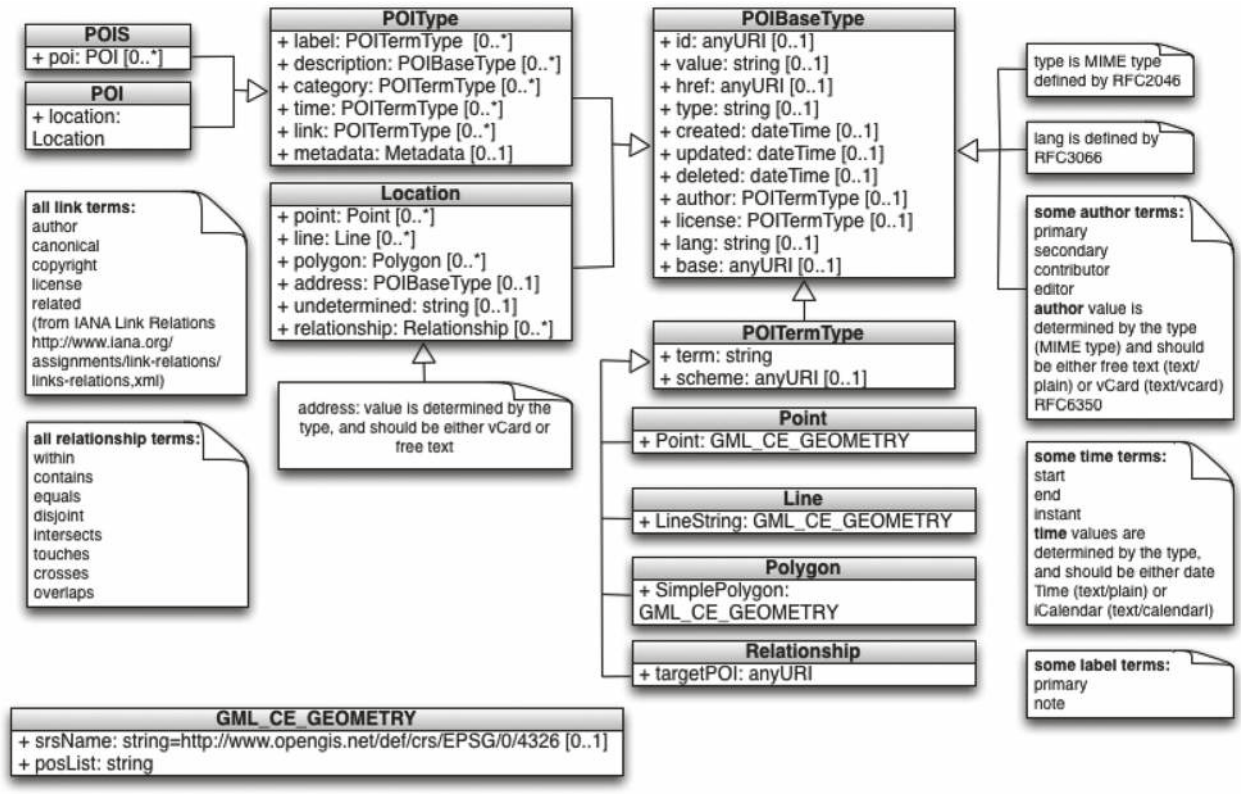


Fig. 1 W3C POI Core Data Model [4]

of templates for the creation of mobile application, making use of open data. In the tourism field, no API was defined for accessing open data. Instead, a limited scope data representation format for POIs was created for use with the template based mobile applications.

The Open Cities project has collected 23 open data set from different European cities [5], making them available to developers in hackathons. However, the goal of Open Cities was not to unify data access but to validate how to approach Open and User Driven Innovation methodologies to the Public Sector[5]. Hackathons were successfully used to engage developers.

Several efforts have used RDF to represent touristic data, albeit using different ontologies. The Swiss Linked Open Tourism Data uses RDF to represent tourism statistical data in Switzerland[6]. RDF has also been used to create a metadata model for encoding semantic tourism destination information [6]. We are not aware of the existence of any widely accepted ontology for the use of RDF to express touristic information.

3 CitySDK Tourism API

In this section, we will describe some of the key features of the CitySDK Tourism API. We will describe the message format model, how the API is designed, the features enabled by this design and how it meets the requirements expressed in Section 1.3.

We decided to design an API, as it provides both a method for providing access to the data and a data representation format. An alternative would have been to only provide a data format and have applications download the entire database. This would not scale well, forcing applications to periodically download all the data even if they did not need it. This would likely lead to applications using stale data. Using an API, applications can be light-weight and only access the information they need. Ubiquitous Internet access makes API use possible, even thought it is still possible to download the entire database if needed for offline use. The use of an API allows data providers to integrate multimedia content without having to worry about the database size as data is only fetched on demand.

In order to address the broader goals of CitySDK (see Section 1.2) a sound methodology is of critical importance to avoid the pitfalls of deploying an uncontrolled maze of APIs [7, 8]. The well-adopted Service Oriented Architecture Development Lifecycle (SDLC) [7] provides a solid foundation for service enablement in an orderly fashion so that services can be efficiently used in Service Oriented Architecture (SOA)-based smart city applications. SDLC relies on three fundamental SOA design principles:

coupling, cohesion and granularity. These design principles need to guarantee that services are self-contained and come equipped with clearly defined boundaries and interfaces to allow for service composability. Standards and reference models (e.g. W3C POI) are crucial in coping with such design principles. Typically, these standards improve SOA design by defining (sector-specific) business concepts that have a high-degree of cohesion and low-degree of coupling. The design of the tourism API is fundamentally based on W3C POI (see Section 3.1). By reusing existing standards, the chance for adoption is substantially increased. Furthermore, it reduces coupling because developers are not bound to use message formats linked to a specific service implementation [7].

Before diving into a deeper description of the CitySDK API there are three fundamental aspects that should be mentioned: the API provides four types of data models; methods to retrieve information concerning these same data models; and also methods and description fields that retrieve the relationships between each of them. Regarding the data models we provide the following:

- **POI** describes various places in a given city, ranging from monuments and museums to eating places and cultural venues;
- **Event** describes cultural events that happened or are about to happen in the city;
- **Itineraries** describe a group of POIs organized in such a way that they form an itinerary of a given topic (e.g. the life of a given person, the history of a given region or even just specific sightseeing spots);
- **Categories/Tags** describe a list of available categories and tagging terms for each of the aforementioned models.

Each model can also be grouped into a list of its own type. These four data models were judged as necessary and sufficient to express the touristic information made available by the several participating cities, one of our requirements. Furthermore, they allow to express the touristic information types gathered from interviews, focus groups and brainstorming with both developers and tourists, as well as those found by analyzing the most popular tourist applications. We purposely designed an API which is to be presented to developers as read only, leaving crowd-sourced information (such as reviews) as added value services to be provided by the applications, thus leaving space for new business opportunities and limiting the liability of the data providers.

3.1 W3C POI model in the API

The four data models are mapped using the W3C POI Model presented in Section 2.2. The **Point of Interest (POI)** is the most easily modelled element of the API. Since the W3C POI Model is specific to this type of data, we used its already specified properties to map our data model. The POIs are mapped and described by using the *POI* entities directly. It should be mentioned that, since the POI is somewhat detailed and verbose, we defined two granularities for this element: a minimal description, that only includes the key essential properties, and a complete model. The minimal model is used to map each element of a list of *POIs*. Such list is described by the *POI* entity, but it does not use the descriptive properties of *POIType*.

The **Events** are modelled the same way as POIs, but instead of having a *Location* entity completly specified, we used the *relationship* property of the same entity to link a given Event to a POI and omitted the *address* and *undetermined* properties. So, we have an Event completly described using the *POI* entity and use the *relationship* property to also specify and descibe the location of the Event. An Events list is modeled using the *POIS* entity, much like the POIs, but it does not have a different granularity and the root name is *event* instead of *poi*.

The **Itineraries** data model is somewhat more complex. It is defined by using the *POIS* entity and all of its descriptive properties. So, we have the description of the Itinerary itself by using the *POIType* descriptive properties and have the group of POIs by using the *poi* property named as *pois* (so not to confuse with the mentioned list of POIs). It should be mentioned that these POI are not the original POIs, but are described in the context of the Itinerary, though they include the relationship with its original counterpart, so to fetch the actual description. Finally and like the previous two, the Itineraries has a list associated with it. Much like the POI, it has a second granularity - a minimal version - in which only the description of each Itinerary is included and their POI are omitted.

The **Categories/Tags** are equal in nature, but a Category provides a recursive format that the Tags do not. Both borrow from the W3C POI Data Model, but their format is more specific to the needs of our API, rather than following the mentioned model. So, a Category simply follows the *POIType* entity and allows recursiveness and a Tag borrows its properties from the *POIBaseType* to specify a language and value.

At last, most of the terms used in the *POITermType* are those suggested by the WG itself. However, we've added five more terms regarding price, waiting time, occupation and accessibility information for handicapped people.

The presented models are transmitted using JSON. We choose it as it is widely supported by popular programming languages, in particular those user for web and mobile development. Combined with the W3C POI standard, this also allows for easy expansion by including new attributes.

3.2 API description

The API follows the Representational State Transfer (REST) architectural design [9]. We designed a RESTful API over HTTP using JSON. Hence, for each of the presented models we designed various methods to obtain data by providing certain parameters. Many of the parameters are common between the POIs, Events and Itineraries, such as the ability to search for each one of them using a category reference, a description or using geographic boundaries(e.g. coordinates and radius or a polygon). Also, we have provided limitation parameters to allow applications to lazy load the data presented by the API. Of course, there are some parameters that are specific to the data model (e.g. if we search using a description of the POIs, one can ask for either the minimal or complete version or in the case of the Events, we can search using time spans). Furthermore, and for both POIs and Events, one can also search for the relation of a single POI/Event with other POIs/Events. E.g. a set of concerts may be children of a music festival event, or a set of parishes may be children of a city POI. One final method is the ability to search using a Quick Response (QR) Code or an one-dimensional barcode. Using a single method and providing the textual or code information, we retrieve any POIs, Events or Itinerary that matches such information. As for the categorization models, we have provided methods to retrieve the categorical information for each of the aforementioned models.

Another feature of the API is complying to the Hypermedia as the Engine of Application State (HATEOAS) constraint of Fielding's seminal work on REST APIs [10]. This constraint states that a client interacts with a network application entirely through hypermedia provided dynamically by the application servers. Therefore, it needs no prior knowledge about how to interact with any particular application or server beyond a generic understanding of hypermedia. We made use of this constraint in three ways: from the entry URL (the only URL the client needs to know) we present the API version and the resources made available by the visited server, including the allowed search parameters using URI Templates [11]; each of the Data Models has an identification (specified by a base URL and ID) that allows to fetch information about that specific model; the POIs, Events and Itineraries can be further described by using a *described-by* (in the *links* property) which indicates an entry point to another server, which can provide further data on that specific entity.

The use of the HTTP protocol, allows the API to benefit from all the well-known load balancing mechanisms applied to common web servers to ensure its scalability. Furthermore, HTTP's cache control techniques will be available for helping clients synchronize their caches with the data on the servers.

3.3 Delegation

The use of hypermedia allows the use of delegation between the various entities involved in the system. Figure 2 shows a diagram of the possible interactions between each entity. A world wide directory will allow developers to use a single endpoint regardless of the user's location. This directory will contain a POI for each city, linking to the adequate server. Within each city, further details about a POI or event may be provided by other servers. A global tourism database is thus distributed over the servers of the several data providers. Delegation is accomplished using the *described-by* property in the *links* property. This delegation mechanism can be construed as a Linked Data mechanism [12].

3.4 Interaction example

Consider a simple tourism application that only uses data accessed through the CitySDK Tourism API. More sophisticated application are expected to merge this data with other sources. Building this application, making full use of the API, requires the developers to write code to access, interpret, use and display all the types of information that can be provided by the servers. However, it does not require the developer to know about which servers are available or update his application as new ones become available. An application will work, without change, in every city that provides data using the API.

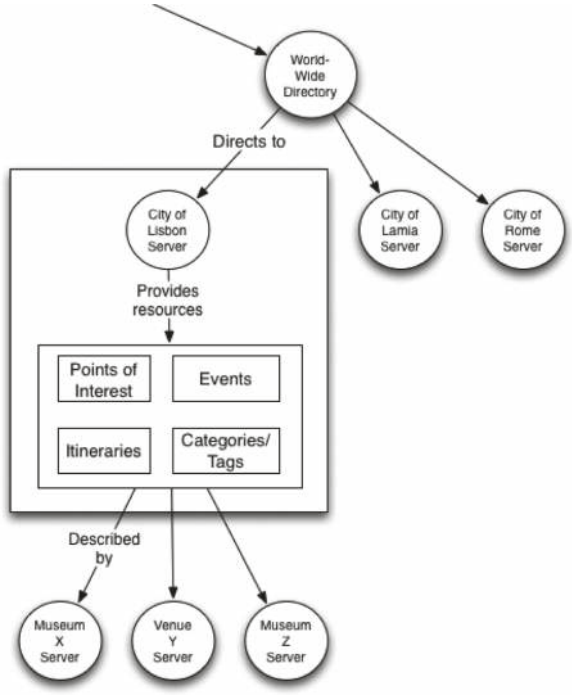


Fig. 2 Delegation model

Data providers, such as municipalities, tourism bodies and associations, private and public entities, need not worry about the applications that will use the data they choose to make available. They need not worry about what is the most popular mobile platform for their data or how to create the best application to reach the tourist. Developers will take care of the applications, creating the opportunity for each tourist to find the application that best suits his needs. Data providers are free to focus on procuring and curating high quality data, to be made available to tourists. As long as data is made available using the CitySDK Tourism API, several application will be available to make use of the data.

Consider a tourist visiting Lisbon for the first time. He needs not install a Lisbon specific application, as the one he used in other cities will also work there. Upon arriving in Lisbon, the tourist searches for a hotel. The application will use the Global Positioning System (GPS) to determine its location. As it does not know a server for that location, it will contact the world wide directory searching for POIs in that location. The directory will return the POI for Lisbon, containing a description of the city and a pointer to the municipality run server (at least, there could be more than one service).

The application then uses the provided server, querying it for hotels. It uses the "Hotel" category and the "boutique" tag, searching for matching POIs within a radius of 5 Km of the current location, according to the users preferences. The municipality server, which has more detailed information on the city of Lisbon, will be able to respond with a list of nearby hotels POIs, which the application displays to the user (see Fig. 4b).

On the way to the hotel, the tourist can use an augmented reality application to learn about the locations he goes by (see Fig. 4a). The application will periodically ask the municipality server for the list of POIs within the vicinity of its current location, using their geodetic coordinates to place display them.

The following day, the tourist decides to take a tour of the city. The application proposes several itineraries (obtained from the municipality server) and the user selects to follow the Fado (a local type of music) itinerary. The application queries the server for the itinerary details, receiving a textual description, links to images and a list of POIs to visit, each with its own description with information relevant in the scope of the itinerary.

During the tour, the tourist visits a concert venue. The application retrieves information about the concert venue and learns that there is a server specific to it. This server is updated daily by the venue owner. The application uses this server to get detailed information about the shows taking place there, in the form of events. Later the tourist chooses to visit a museum. The POI for the museum also indicates that there is a server providing more detailed information. The museum curator used POIs to represent the paintings on display. As GPS does not work inside, QRs codes are used to identify each painting. The same application is now able to provide information about each painting, and even display multimedia content associated to each one.

4 Implementation

The most important part in the specification of an API is the specification itself, that must be explicit enough to allow for distinct, compatible implementations. But to bootstrap adoption, we developed a reference implementation (hence forward called platform) with two goals: proving that the API was implementable and testing it; lower the implementation costs for data providers by providing a reusable server implementation. Data providers are free to use it or implement their own version.

4.1 Platform architecture

Figure 3 shows the architectural components of the implemented CitySDK platform. This platform was developed for use in the lead pilot, the city of Lisbon, and was later adopted by the other participating cities. The architecture was chosen to match the needs of the cities, that each had several different data sources using different formats.

In the platform's architecture, the city acts as a data provider which offers one or more relevant touristic information datasets. The CitySDK platform is capable of incorporating several distinct data sources by using a modular approach where each one of the data sources provides its information through a Data Adapter module. The Data Adapter module is a component developed to retrieve raw data from a data source in its native format (XML, JSON, CSV or other) using the data source's native access form (web service, file, database or other) and provide the touristic data (regarding POIs, Events and Itineraries) to be stored in the platform in the data format presented earlier. Each Data Adapter module interacts with the CitySDK platform by invoking its Authenticated data administration REST API.

Even though the CitySDK's platform has a single API endpoint, it can be divided into two distinct conceptual APIs: the public API (used by clients) and the data administration API. The data administration API requires a valid authentication to access it, as it is the interface to be used for the purpose of manipulating data elements. The CitySDK platform maintains data retrieved in its own database.

4.2 Implementation details

As shown in the architecture representation (Fig. 3), the core of the CitySDK platform is based on a main component, the application service denoted as "CitySDK Platform API". This provides the previously mentioned APIs

(public and data administration), and is aided by a platform's database.

In order to speed up the development effort and ensure high performance, the application service was implemented using a fast and lightweight open-source REST Web Services framework named ServiceStack[7]. The ServiceStack base framework is responsible for handling the protocol part of the CitySDK's platform interface, converting the received web requests into simple method calls and converting back the method return values into a web response.

For the CitySDK platform's database, the open-source high performance document database mongoDB[8] was used instead of a traditional relational SQL based database. The choice of this database was mainly related to the high performance required to handle the foreseen database request load and its ability to process geographical queries (i.e. search for elements within a polygon or within a distance from a specified geographical point), as required by the CitySDK Tourism API. This document database stores both the platform's data models and some minor administrative data (e.g. access credentials). The data models are stored in an optimal format, taking advantage of the characteristics inherited from the fact that the database in use is a document database: minimal effort for the platform's engine to adapt the retrieved data elements to the replies for the clients requests. Although the used database is not a relational database, it also provides the possibility to index any of the stored document's attributes to enable the possibility of performing quicker indexed searches.

Although Data Adapter modules are not part of the CitySDK platform's itself, they are essential, as they perform the important task of populating it with valuable data. These modules have to be CitySDK-compliant on the data output side and datasource-compliant on the data input side, transforming the datasource's data from its native format into the W3C POI format suitable for insertion into the platform. For the case of the Lisbon implementation, a single database containing aggregated data from POIs and Events (including the relation of the POI where Events occur whenever possible) was identified, so only a Data Adapter module was implemented for the two types of data. The implemented module runs as a service and, as the volatility of the data is very low (at most a few records are updated each day), the data updating process runs once a day during the night. A different Data Adapter populates the database with itineraries.

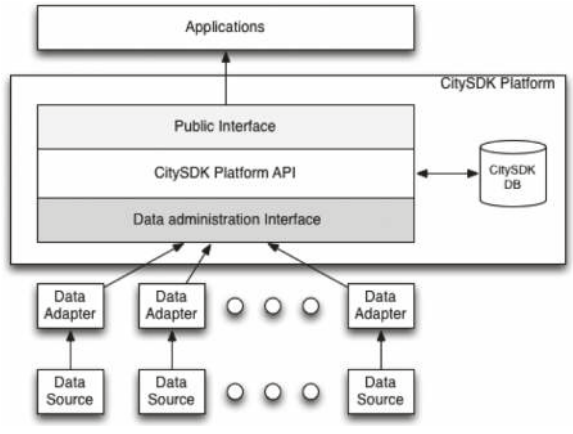


Fig. 3 CitySDK platform architecture

The reference server implementation was written in C# for the .Net framework as this was the language of choice of the available development team at the partner implementing the server. It can be run using a Microsoft Server environment or Linux using Mono framework [9]. It was deployed in Lisbon in early 2013, with Lamia, Rome and Amsterdam following that same year. Helsinki launched its endpoint in early 2014.

5 Promotion

In order for application developers to adopt an Open Data API, data must be available in quantity and with quality. Cities and other data owners are more likely to adopt an API to provide their data in an Open Data format when there already exist application using that API. They will also be more likely to invest in improving the quality of their data, e.g. by providing translations to several languages, detailed description, multimedia content and a consistent level of detail. In order to overcome this chicken and egg problem and bootstrap the API adoption we developed a strategy whose corner stones were: proving the effectiveness of the API through pilot deployments and demo applications; lowering the cost of development for application creators and data providers; providing ample documentation and contact media; presence in several dissemination fora. The pilot deployments in Amsterdam, Helsinki, Lamia, Lisbon and Rome proved that the API could be deployed and made use of the reference server implementation, which is publicly available in order to reduce adoption cost by data providers. The next sections detail the other efforts.

5.1 Website

A website[10] was created and promoted in order to concentrate information on the API. It is meant to be a one stop source of information for application developers and data source owners interested in using the API. The site contains a blog with news related to the CitySDK Tourism API, documentation on the API, the reference server implementation and its usage documentation, endpoints for existing deployments, demo applications and known third-party applications and contacts. Besides the

API documentation and code examples, developer support is provided through a mailing list and a Google group, embedded into the website. The server reference implementation and the several demo applications and libraries are open-source and available in GitHub[11]. The documentation on the API includes links to the live APIs, providing developers with real examples.

5.2 Client-side stubs

To facilitate the development cycle for potential developers, we provided four libraries which abstract the use of the API and the parsing of the received data. These libraries are very similar in usage, as they provide equal naming conventions for the methods to perform requests and to read the received data. The client libraries are available in Java, JavaScript/jQuery, PHP and Objective-C and are available online, as is their corresponding documentation and usage examples.

5.3 Proof of concept applications

To further test our client libraries and API, we developed a group of key applications in various programming languages and frameworks. This process also had the goal of providing code examples to developers and supply applications to be used in demonstrations to potential data providers, developers and even users.

The first application made use of the Java library. We developed an Android application which displays POIs, Events and Itineraries following an user's criteria (e.g. using categorical information and/or using coordinates). It displays the retrieved data in both list and map formats (in the case of the Itineraries, it draws the route itself). Its main goal is to use the API to its fullest and demonstrate the various possibilities for applications. The Android platform enabled us to showcase our data using the localization features available in mobile platforms.

Later, a second Android app was developed, that allows the user to see POI and Event details including associated media, such as pictures (Fig. 4c), and navigate through the CitySDK Tourism data in a map (Fig. 4b), using the data made available by the five different cities. This app has the particularity of integrating functionality from another API promoted by the CitySDK project: the participation API, which is based on Open 311 [13]. Users can use this app to submit reports about the data, e.g. request the translation of a POI to a certain language. By providing this feature, the data can now be incrementally improved.

A set of applications making use of the JavaScript/jQuery library were also developed. This set is composed by an Event's calendar widget, which makes specific use of Events and time/categories related searches, and a Map widget displaying all data types. The Map widget makes use of geometric figures (circles and polygons) drawn by the user, to restrict the searched area. These applications, which run on a browser, allow everybody visiting the website to observe the API in operation and have access to the open data. They also provide a debug mode, where developers can see and tweak the requests being made to the servers and the replies provided. This way, developers can see and test the data and all the information offered by the different city partners.

At last, an Augmented Reality application using the Layar framework[12] was also created by developing a PHP web-server, thus using the PHP library. This server enables Layar to access CitySDK Tourism data. Layar makes use of the position, camera and sensors of the user's device to display information about the surroundings and/or the building being displayed in the screen. POI information from CitySDK is overlayed on the image. It can be used with the Android (Fig. 4a) or IOS Layar App or with Google Glasses. With this application, we accomplished the goal of reaching the major mobile platforms as well as being as interactive as possible.

A basic IOS app accompanies the Objective-C library.

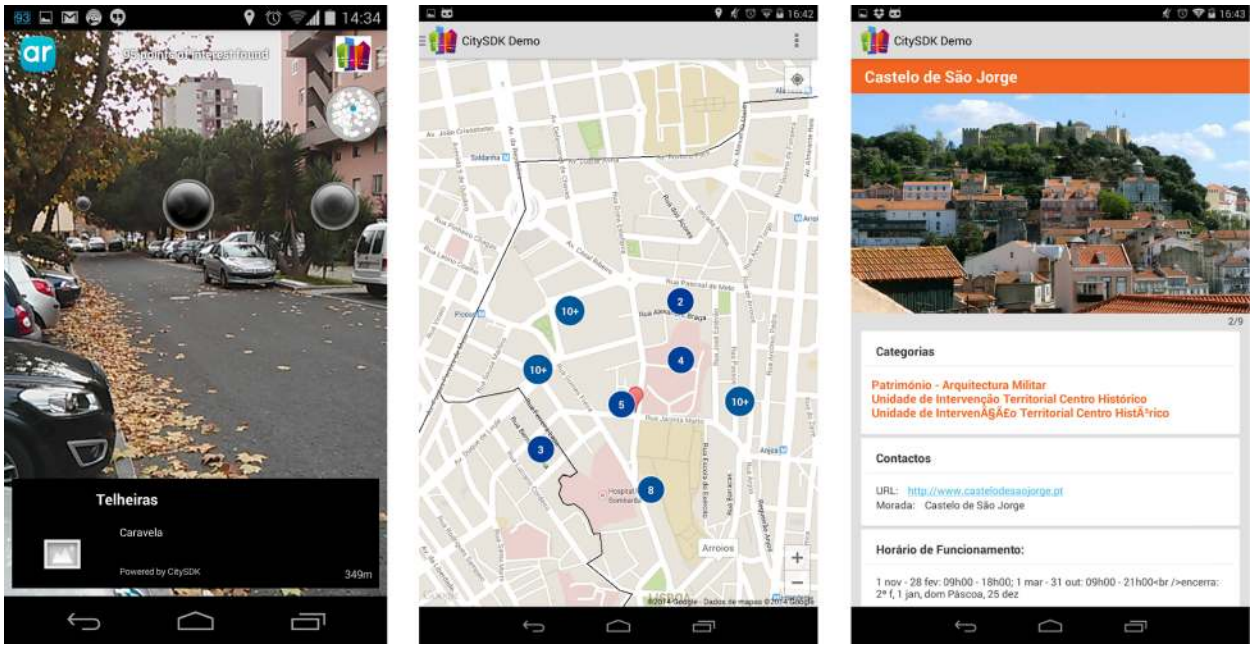
5.4 Collaboration with other projects

During the CitySDK project there was another EU funded project: Citadel On The Move. This project provided application templates with the goal of facilitating the development process and allowing entities to easily provide their users with applications to access their data. One of the applications was an HTML 5 based mobile application for accessing POI data. The used data model was simpler than the one we adopted, but enabled a subset of our data to be provided. With this similarity, we decided to build an API converter and, by doing that, we automatically enabled the possibility to build mobile applications using Citadel On The Move with CitySDK data. This enabled us to use Citadel On The Move to increase the reach of the CitySDK Tourism API and data.

5.5 Dissemination activities

Our dissemination activities mainly targeted developers and data owners. Tourists were not our target audience, as they were a much wider audience, requiring a larger budget and greater men-power, even though user driven demand could be a motivation for data owners to publish.

Promotion to public decision makers was performed through the presence in tens of conferences and events, populated by decision makers, namely some promoted by the World Bank. Promotion with developers was achieved through Facebook, Twitter and the presence in conferences, developer events and Hackathons. In particular, the CitySDK Tourism API was highlighted in the Lisbon Big Apps challenge, in 2014.



(a) Augmented reality (b) POI mapping (c) POI Detail

Fig. 4 Demonstration applications

6 Adoption, opportunities and lessons

In this section we present the current state of the API adoption and discuss the possible business and value opportunities that justify the use of the API by the several stakeholders. We also discuss the lessons learned.

6.1 Adoption

Currently, following the work done on the CitySDK project, the API is available in five European cities: Amsterdam, Helsinki, Lamia, Lisbon and Rome. General touristic data is available as POIs and Events, including data such as Museums, statues, monuments and concerts. Lisbon in the only city to currently provide itineraries. Rome also provides access to the location of public WiFi access points. The amount and quality of the data varies from city to city. Also, the categorization of data is not uniform among cities, as the published data already existed and continues to be made available using the previously used, proprietary formats.

We have been approached by several city officials, from the USA and Europe, regarding the adoption of the API. So far, these have not been made available.

Besides the demo applications made available by the core team behind the API creation, several other applications have been made available by the participating cities. These are in use and made available to the citizens or tourists visiting those cities. These applications are presented in our website.

From the various events and Hackathons, several third-party applications have emerged that combine tourism data with other data sources or use if for particular purposes. One such example is an application that allows tourists who can play a music instrument to discover other musicians they can play with while visiting another city. The first application we became aware of was the *Spot in* series, that was initially available for Helsinki, but today features versions for Amsterdam, Helsinki, Lamia and Lisbon, thanks to the CitySDK Tourism API. We currently know of about half-a-dozen applications and have been contacted by developers interested in creating new apps and students and professors interested in using the API in their projects or courses. The applications we are aware of are show-cased in our website, but there is no requirement for developers to notify us that they are using the API.

6.2 New opportunities

As mentioned, this API was developed under the CitySDK project which involved 29 partners from 9 countries. Ideally this API could be extended to the rest of Europe and the world, making it a very powerful tool to enable tourism related businesses and applications. CitySDK Tourism API provides business opportunities for application developers, cities, service providers and venue owners.

Application developers are the most obvious beneficiaries of the API. Since each city is integrated with the same API and data models, it is much easier and less costly to

create new applications and, as a result of our delegation model, integrating new cities has much less overhead and complexity. Competition will drive new uses for the data as well as integration with other data sources. Application development may be financed by advertisement on applications, paid applications or sponsoring. Tourists will also benefit from the broader availability of the applications and the quality of the data.

Cities, especially early adopters, will benefit from a competitive advantage that will provide some differentiation, by providing tourists with information for their visit and having that information made available through the applications created by developers. This may help promote the city as a touristic destination. When cities provide their own endpoints, they may also use API access information to help determine the most popular POIs, analyze usage trends and predict tourist activity.

The same can be said about venue owners, that may wish to provide their own CitySDK Tourism API endpoints, where detailed information about the venue (i.e. POI) and events may be provided. Thanks to the delegation model, applications will find these specific endpoints through the city's server. For instance, a museum can provide more detail information by providing information regarding pricing, the number of people that can be accommodated, queue's length and waiting time or even information on accessibility for persons with disability. This type of highly detailed data or data with a small timespan would be difficult to be managed by a municipality. The Amsterdam partners have demonstrated such a use case by integrating real-time queue length data for the Rijksmuseum, obtained from a sensor network, into Amsterdam's endpoint [14].

Also, our API provides opportunities to create new businesses that either provide more detailed information or provide services to the other players. A company could provide high quality data on a city or area through a paid API, accessible only to paying developers. An alternative approach would be to offer a two-tier level of service, with a paid service offering higher throughput or other differentiating service.

Other companies could run CitySDK Tourism API endpoints of behalf of others, lessening the investments required to publish data in this format. For instance, a bar might be interested in publishing its live music events, but be unable to run its own server. It would hire a CitySDK Tourism API hosting provider that would make the endpoint available while providing the bar owner with a simple web page for updating the events information.

Tourism is an activity that is highly related to mobility. Integration of tourism data with mobility information can help tourists make the most of their time and even save money on transportation. Some will be willing to pay for applications that provide them these benefits.

Today, crowd-sourcing is an important way to obtain information on POIs and events. In particular, a tourist may resort to comments, opinions and ratings provided by others when choosing what to visit. In the design of the API, crowd-sourced information was explicitly left out in order not to expose the participating municipalities to potential liability, by publishing negative data about a venue. However, this is an interesting business area to be explored by private enterprises.

High performance servers, or ones that better integrate with a city's particular ICT infrastructure, will be required where the reference server implementation does not fit the bill. Hosting and consulting opportunities will also be available, as specialised companies can provide a better value proposition for cities wishing to open their data using this API.

6.3 Lessons learned

During the Lisbon pilot, by supporting the creation of the other endpoints and from interaction with developers and users, several lessons have been learned. Here we present the main ones.

Multi-language support is very important for a tourism API. Support for multiple languages was included from the onset. However, not all cities were capable of providing translations for all their data in multiple languages but only on their native one. For instance, in Lisbon, thousand of POIs are available in Portuguese, but not all of them are available also in English, and none are available in other languages. It is up to the data providers to make data available in several languages, even though this process can be facilitated by machine translations.

As data made available from the several cities already existed, different ontologies were in use to classify it. A single ontology for classification would facilitate users' searches and would facilitate data presentation by the application developers. The process of defining such ontology, agreed upon by all cities, was left outside the scope of the CitySDK project but could in the future be promoted in the scope of the CitySDK association.

When evaluating an API to use, the likelihood that it will be supported in the future is an important criteria. As the API was created in the context of a EU funded project, concerns about what would happen once funding run out were expressed. The project participants addressed this issue with the creation of an association that will carry on the work after the project ends. However, the creation of this association was a late answer to the problem, that should have been planned from the beginning in order to provide potential users with an assurance about the continuity of the APIs.

We have learned about the importance of engaging the developer community in such a project. Feedback from developers is important from the requirement gathering,

through the API design, all the way to the deployment and testing of the endpoints. Relations with developers should be nourished throughout the development cycle as these were important disseminators of the API. We propose that projects with similar goals should setup the figure of community manager, as many open-source projects do.

In our case, we saw significant adoption of the API through hackathons. Although these events can be expensive to organize, especially if sponsors are not found, the allure of prizes and the visibility provided to the winners helps attract talented developers and startups which often came up with innovative out-of-the-box ideas for using the API. These were then able to gain their own notoriety on which our API was able to piggyback in the media.

7 Conclusions

This paper presents the CitySDK Tourism API, which is based on the data model defined by the W3C POI WG. This API provides access to information about POIs, Events and Itineraries, enabling municipalities, regional or national governments as well as other public or private entities to publish touristic information for developers to incorporate into their applications. This open-data model is expected to increase the market for tourism applications while lowering the cost of entry. The increased competition should drive down application costs while increasing their quality, benefiting tourists. Data publishers also benefit from increased exposure, increasing the appeal and visibility of their attractions.

This paper provides an account of the API creation and promotion effort, providing a snapshot of its current usage and motivating the business case and opportunities for its adoption by the several stake holders.

The team behind the creation of the API continues to provide assistance to the developers and data owners interested in adopting or evaluating the API. The CitySDK partners, as a whole, are in the process of setting up an association that will carry forward the development and maintenance of the APIs now that the EU funding has ended. In particular, the cities running the endpoints have committed to continue to do so in the future.

Currently, a worldwide directory for CitySDK Tourism API endpoints is run by the municipality of Lisbon, enabling new cities to easily join this service. A single endpoint is crucial for enabling existing applications to take advantage of new CitySDK Tourism API deployments as they become available. However, centralized management is not scalable. In the future, we plan on designing a peer-to-peer, trust based, federation system, that enables each city endpoint to act as an entry point to all the cities and allows for new cities to join without having the contact a single central entity.

Developer keys are often used to identify developers, providing a way for server owners to throttle API usage in order to deter heavy-hitters and providing an incentive for programmers to produce efficient code. These can also be used to ban misbehaving applications from using an API. A worldwide directory would only make sense if developer keys are portable among different endpoints. In the future, we plan to propose a model for distributed developer keys issuing which does not compromise these goals.

Endnotes

[1] http://www2.unwto.org/content/why-tourism
[2] http://www.lisboaparticipa.pt/pages/newApps.php
[3] http://www.hri.fi/en/
[4] http://www.citadelonthemove.eu/en-us/about/aims.aspx
[5] http://www.opencities.net/content/project
[6] http://make.opendata.ch/wiki/project:lotd
[7] http://www.servicestack.net/
[8] http://www.mongodb.org/
[9] http://www.mono-project.com/
[10] http://tourism.citysdk.eu/
[11] https://github.com/CitySDK
[12] http://www.layar.com/

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
All authors contributed to the writing of the manuscript. All authors read and approved the final manuscript.

Acknowledgements
The authors would like to thank all the members of the CitySDK project, in particular the teams responsible for the deployment of the replication city endpoints: Stichting Hogeschool Van Amsterdam, Forum Virium, Municipality of Lamia, Provincia di Roma. A special recognition is due to the Municipality of Lisbon, for providing the data used in the lead pilot, and Alfamicro for the diligent dissemination efforts.
The CitySDK project was financed by the European Commission under the Information and Communication Technologies Policy Support Programme (ICT PSP) as part of the Competitiveness and Innovation framework Programme (CIP), through grant number 297220.
This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

Author details
[1]Instituto Superior Técnico, Universidade de Lisboa, Avenida Professor Cavaco Silva, 2744-016 Porto Salvo, Portugal. [2]INESC-ID, Avenida Professor Cavaco Silva, Edifício IST, 2744-016 Porto Salvo, Portugal. [3]ISA Energy, Rua D. Manuel I, 30, 3030-320 Coimbra, Portugal. [4]ERISS - European Research Institute in Service Science, Tilburg University, Warandelaan 2, 5037AB Tilburg, The Netherlands.

Received: 29 May 2015 Accepted: 5 November 2015
Published online: 17 November 2015

References

1. United Nations World Tourism Organization UNWTO Tourism Highlights, 2015 Edition. Booklet (2015). http://mkt.unwto.org/publication/unwto-tourism-highlights-2015-edition. Accessed 12 Nov 2015
2. Vilajosana I, Llosa J, Martinez B, Domingo-Prieto M, Angles A, Vilajosana X (2013) Bootstrapping smart cities through a self-sustainable model based on big data flows. Commun Mag IEEE 51(6):128–134
3. Holland K (2012) NewsML-G2 Implementation Guide, revision 5.0. IPTC Standards. http://www.iptc.org/site/News_Exchange_Formats/EventsML-G2/Specification/. Accessed 12 Nov 2015

4. Hill A, Womer M (2012) W3C Points of Interest Core. W3C Editor's Draft. Online, accessed 11 Jun 2013. http://www.w3.org/2010/POI/documents/Core/core-20111216.html
5. Sfairopoulou A Open Data and Open Sensor Network Challenges. Deliverable D4.4.54-D6.6.44, Open Cities Project (October 2013). http://opencities.net/sites/opencities.net/files/content-files/repository/D4.4.54%20-%20D6.6.44%20Open%20Data%20and%20Open%20Sensor%20Network%20Challenges%20-%20b.pdf
6. Kanellopoulos DN, Panagopoulos AA (2008) Exploiting tourism destinations' knowledge in an RDF-based P2P network. J Netw Comput Appl 31(2):179–200
7. Papazoglou MP, Van den Heuvel W-J (2007) Service oriented architectures: approaches, technologies and research issues. VLDB J - Int J Very Large Data Bases 16(3):415
8. Monsieur G, Snoeck M, Lemahieu W (2012) Managing data dependencies in service compositions. J Syst Softw 85(11):2604–28
9. Fielding RT, Taylor RN Principled design of the modern web architecture. ACM Trans Internet Technol 2(2):115–50
10. Fielding RT Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California: Ervine; 2000
11. Gregorio J, Fielding R, Hadley M, Nottingham M, Orchard D URI Template. IETF RFC 6570 2012. https://tools.ietf.org/html/rfc6570
12. Berners-Lee T Linked Data. W3C Design Issues, Last accessed May 10th, 2015. http://www.w3.org/DesignIssues/LinkedData.html
13. (2011) GeoReport v2. Open311 Stable Specification. Online, last accessed 20 May 2013. http://wiki.open311.org/GeoReport_v2
14. Groen M, Meys W, Veenstra M (2013) Creating smart information services for tourists by means of dynamic open data. In: Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication. UbiComp '13 Adjunct. ACM, New York, NY, USA. pp 1329–30