DELIVERABLE

Project Acronym:	**CitySDK**
Grant Agreement number:	**297220**
Project Title:	**Smart City Service Development Kit and its Application Pilots**

Tourism Pilot Application and its SDK Components

Revision: 2

Authors:

Jacqueline Jardim (IST)
Pedro Sousa (IST)
Ricardo Lopes Pereira (IST)
André Oliveira (ISA)
Valter Ferreira (CML)
Nuno Xavier (CML)
Jean Barroca (Alfamicro)

With contributions from:

Maarten Groen (HvA – Amsterdam University of Applied Sciences)
Wouter Meys (HvA – Amsterdam University of Applied Sciences)
Susanna Ollila (Forum Virium Helsinki)
George Koukas (Gnosis)
Giovanni Calcerano (Rome)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	
C	Confidential, only for members of the consortium and the Commission Services	X

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Revision	Date	Author	Organisation	Description
1	03/12/2012	Ricardo Lopes Pereira	IST	Initial draft
2	24/12/2012	Ricardo Lopes Pereira	IST	Included contribution from pilots

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of Contents

1 Introduction

Tourism is an expensive activity, both financially and in terms of time. Tourists want to make the most of their time and money. This was always so, fostering tourism related businesses like travel books and guided tours. Aiding tourists in planning their travel is valuable for the tourists themselves, but also for the cities and touristic venues, which can expect higher returns and fidelization.

Cities have much information which is relevant to tourists. They make it available through city guides, tourism offices, websites and even mobile apps. This information pertains mostly about Points of Interest (POI) and events. Mobility is also an important issue as tourists are unfamiliar with the city, being covered by a distinct Work Package within CitySDK.

Cities realize that the return on investment for applications may be low. It is very expensive to develop and advertise applications for the most significant platforms (for mobile alone, there are several), so the number of effective users of each application may be low. By opening their data, cities open the door for entrepreneurs to create new, appealing apps, which facilitate the lives of tourists, while costing nothing to the cities.

Entrepreneurs are more dynamic, agile and less constrained than cities, being able to keep up with the constant technological evolution. Furthermore, they can integrate data from several sources to make their applications more valuable. However, few cities make their data available for others to freely use (what is known as OpenData). Furthermore, in the absence of standards, each city which embraces OpenData publishes their data on its own format. This complicates developers' lives, effectively limiting the geographic scope of each application.

By adopting a single format, a standard, for publishing their data, cities will enable the creation of applications which are portable among cities, thus increasing their potential market. This will enable entrepreneurs to make higher investments and more complex applications. Applications will become better and both cities and tourists will benefit.

1.1 The CitySDK Tourism API

In this document, we present an initial version of the CitySDK Tourism Service API, which will be tested in the Lisbon pilot. The learnings from this pilot will be reflected upon before the replication pilots.

The proposed API is based on two concepts: POI and events. There is a third concept, the route, which is simply a sequence of POIs together with a description and other data.

The POI encompasses every tourist attraction. It can be the city itself, a street, a statue, a store, an hotel, a museum, a painting inside a museum, a concert hall, etc. Each POI will have information associated with it. This information may be provided by a single source or it may be provided by several sources, where some will only have partial information and direct to other sources for the full information.

An event is an occurrence that takes place during a limited period of time. Examples of events include a concert, a theater play, a fireworks display, a book reading or a charity sale. Events take place in a POI.

A route is a sequence of POIs which are related according to some theme. An example would be a route of medieval monuments, where a sequence of POIs is proposed for the tourist to visit. The route itself has a description and other data.

The CitySDK Tourism service provides access to POIs, events and routes. The service builds upon the W3C POI recommendation, which is the base for the message format. To this we add the search functionality which enables data to be searched and retrieved.

1.2 Roles

Municipalities are not the only ones with valuable data for tourists. Businesses have this data, museums have this data, cultural organizations have this data. As such, we envision that CitySDK services may be provided not only by municipalities but also by other entities, either public or private. The following participants will, thus, take part in the CitySDK tourism ecosystem:

- **Service provider**

The service provider is an entity which has data to share with tourists. It implements the CitySDK Tourism service API, making this service available to developers. A municipality will be a service provider, but within that city there may be other services providers. The amount of information a city possesses about its POIs is broad but it is probably not very deep. An example would be a museum: the city knows about it but does not have detailed information about its collections. The museum would be a POI in the city's CitySDK Tourism service, but the museum could also provide its own CitySDK Tourism service, with information about its collections. A tourist, using a mobile application, would learn of the museum's location using the city's service, but, once within, he/she would use the museum's service to find more information about a painting identified by a bar code.

- **Application developer**

An application developer is an entity, individual or company, which creates applications making use of CitySDK Tourism services. These applications will retrieve data from CitySDK Tourism service providers and present it to users in ways that help them accomplish some tasks, or somehow are of value. The applications may combine this data with data from other sources in order to increase their value. Applications may be free, sponsored (e.g. by advertisements) or paid. This can have a local, national or international score. They may work online with up-to-date data or use a local database with previously downloaded data. Applications may access data from the service providers directly or use a service provided by the developer which has performed some processing to the CitySDK data. It is up to the developers to devise ways to make the most out of the CitySDK data in order to make their applications enticing and profitable.

- **Tourist**

The tourist is the main beneficiary of the CitySDK Tourism service. He/she will be the one using the applications which consume the data made available by the service providers.

1.3 Document Structure

This document presents the CitySDK Tourism Service API and the rationale for its design. As such, **Section 2** describes the current situation at each of the partners taking part in the tourism domain. A significant effort has been made by most cities to provide information to their tourists. Much of this information is currently being provided in the form of application. There are some efforts regarding OpenData, but the lack of a common representation format makes porting applications difficult.

Section 3 presents the requirements that were gathered during the project so far. They are the motivation for all the functionalities presented later.

Section 4 presents the general architecture of the CitySDK Tourism Service, analyzing how it can be implemented, both in terms of software and hardware requirements.

POIs are a concept which is not new. They are used in many applications and standardization efforts have emerged. In **Section 5** we provide an overview of the W3C POI recommendation and supporting standards: vCard and iCalendar. They are the basis for the message formats used in the CitySDK Tourism API, which is presented in **Section 6**.

In **Section 7** we detail how the CitySDK Tourism Service will be implemented in the Lisbon Lead Pilot. We indicate which data sources will be integrated to populate the CitySDK Tourism Service and which applications will be created to validate the service.

Section 8 discusses the CitySDK Tourism Service implementation, this time taking place in each of the Replication Pilots. Contributions from Amsterdam, Helsinki, Lamia and Rome are included, in regard to the selected data sources, architecture, applications that shall make use of the service and initiatives undertaken to engage the developer community.

2 Current Situation

This section describes the pilot situation in a pre-CitySDK point. It describes the most relevant tourism related applications found in each one of the pilots, the usage limitations of the applications and identifies the found portability issues.

2.1 Current Data Sets and Applications

Currently, the pilot cities present a set of already deployed stand-alone touristic services that can be potentially upgraded to the CitySDK platform during the next project's phases. This section results from a deep analysis performed to the pilot cities' current offering and depicts the existing relevant tourist related services among those cities.

2.1.1 City of Lisbon

City of Lisbon presents a significant set of open-data raw datasets delivered in the form of files in various formats, such as comma separated values (CSV), excel files or XML files that can be downloaded from the city's open-data web portal. Along with the available data files, for some of the datasets, the web portal exposes Web Services using REST and SOAP architectures. The complete list of available datasets contains about 100 different categories. The following list exemplifies some of datasets that are relevant for building tourism related applications:

- List of public art location with some characterization;
- List of restaurants location and contacts;
- List of hotels and residences with location and contacts;
- List of public gardens (with location information);
- List of public parks (with location information).

Currently all the data presented in the open-data web portal is provided and maintained by the Lisbon Municipality and can be used freely by using its site[1].

There is also a considerable amount of alternative non open-data sources of information available for different platforms such as websites, iPhone/iPad or Android. Some of these sources are currently under the responsibility and maintained by the Municipality itself and others belong to third parties. Among these non open-data sources, we can find information about the events that occur in the city, in the form of a standard static web page. Due to the form of presentation of this data, building applications which integrate events information is very difficult and inefficient at the present moment.

[1] http://www.lisboaparticipa.pt//pages/apresentacaoDados.php

The following list shows some currently available application/webpages examples which are under the responsibility of the Municipality:

- Website *AgendaLX*[2] (Lisbon Schedule) - Site with information regarding cultural events in the city;
- Website *Casa Fernando Pessoa*[3] (Fernando Pessoa House) - Online access to the work of Fernando Pessoa;
- Website *Lisboa Património Cultural*[4] (Lisbon Cultural Heritage) - Online access to information and pictures of statues, sculptures and tiles;
- Website *CatalogoLX*[5] (Lisbon catalogue) - Online access to a searchable catalog of all available libraries;
- Website *Museu da Cidade*[6] (City Museum) - Online access to information following the timeline from pre-historic findings to XX century.
- Website *Fonoteca*[7] - Online access to music database searchable by author and to the database of available books in Lisbon City Hall libraries.

As previously mentioned, other entities also provide some relevant applications/webpages. The following list shows some examples:

- Website *VisitLisboa*[8] - Provides detailed information about Lisbon, Lisbon Region, Culture, Where to go, Accommodation, Gastronomy, Publications and others;
- Mobile application *Lisboa*[9] - Provides a travel guide for the city of Lisbon;
- Mobile application *YouGo Lisboa*[10] - Provides a travel guide for the city of Lisbon;
- Mobile application *I Love Bairro Alto*[11] - Provides a travel guide for the city of Lisbon;
- Mobile application *Lisboa 360*[12] - Provides information about Lisbon's touristic POIs;
- Mobile application *EatOut Lisboa*[13] - Provides information about Lisbon's restaurants;
- Web site *Sapo Timeout*[14] - Provides events agenda/information;
- Web site *Cine Cartaz*[15] - Provides information about cinemas, including current movies and schedules;
- Mobile application *Fundação Calouste Gulbenkian*[16] [17] - Provides information about the agenda for *Fundação Calouste Gulbenkian*'s events.

[2] http://www.agendalx.pt
[3] http://casafernandopessoa.cm-lisboa.pt
[4] http://www.lisboapatrimoniocultural.pt
[5] http://catalogolx.cm-lisboa.pt
[6] http://www.museudacidade.pt
[7] http://fonoteca.cm-lisboa.pt
[8] http://www.visitlisboa.com/
[9] https://itunes.apple.com/pt/app/lisboa/id386148314?mt=8
[10] http://www.yougoplanet.com/Guide.aspx?ID=4&City=Lisboa
[11] https://play.google.com/store/apps/details?id=com.ilovebairroalto.mobile&hl=pt_PT
[12] http://itunes.apple.com/pt/app/lisboa-360/id478272384?l=pt&ls=1&mt=8
[13] http://eatoutapp.com/index.pt.html
[14] http://timeout.sapo.pt/
[15] http://cinecartaz.publico.pt/
[16] http://itunes.apple.com/pt/app/id433222164?mt=8
[17] https://market.android.com/details?id=com.minsight.gulbenkian

2.1.2 City of Helsinki

The Helsinki city currently has an events and venues directory service containing information about the city's events and the locations where the events occur (the venues). This service is open and publicly available through a XML document located in a specific URL[18].

The city also has a directory service for touristic points of interest (POIs) containing location and contact data for several points such as museums, churches, statues, hospitals among others. This service is available through an open and publicly available webservice (using REST technology), located at the URL[19].

At this moment, the city's Municipality also offers some applications and websites for its users, presenting events and venues information (presentation of the data available in the open-data service).

A Mobile guide involving POI-based physical path with digital signage (NFC and QR codes) has been tested in the *Walk and Feel Helsinki* pilot. It involved a mobile application, browsed-based version and physical signage.

The *Tramstop Wall* (Pysäkkiseinä) pilot aims to bring tram passengers together on an electronic message wall. This new media channel, alongside with the sale of tickets using SMS (mobile SMS ticket), are being tested on the canopied tram stops in Helsinki. NFC tags in the stops facilitate the buying of the tickets as well as access to the web-based service.

The City of Helsinki's tourism and travel bureau[20] also offers a list of third party applications dedicated to tourism, such the *Tassa.fi* website. As such, the City of Helsinki doesn't have its own tourism-related applications.

2.1.3 Province of Rome

Currently, Province of Rome, in its open-data web portal, provides publicly available data containing information about touristic environment. Specifically, Province of Rome has made available, in open-data format (xml, csv, tsv), tourism files regarding:

- List of public libraries;
- List of public museums;
- List of *ProvinciaWifi*'s[21] public hotspots for free internet connection;
- Daily events from *Roma&più*[22] - the portal guide to best in leisure activities around Rome;
- List of archaeological sites in the province.

Regarding other non open-data services, Rome provides the previously mentioned portal-guide to the best in leisure activities named *Rome&più*, which can also be browsed using a specific mobile app[23]. The Province also provides a website where users can display and navigate the map of its *ProvinciaWifi*'s public hotspots for free internet connection.

[18] http://www.hri.fi/fi/data/helsingin-kaupunki-tietoa-matkailuun-liittyvista-kohteista-ja-tapahtumista
[19] http://www.hel.fi/palvelukarttaws/rest/ver2_en.html
[20] http://www.visithelsinki.fi/en/come/tourist-information/helsinki-mobile-apps
[21] http://85.18.173.117/mappe/index.php
[22] http://www.romaepiu.it/en
[23] https://itunes.apple.com/it/app/id465619487?mt=8

Furthermore, Province of Rome provides an augmented reality application, the *Augmented Appia Antica*[24] (AAA), which aims to give users an augmented reality experience using a digital archive of the monuments of the park, employing many different technologies for 3D representation; currently this application is available free of costs and presents a layer of 11 POIs which deliver rich contents to its users. Some other audio guides, rich mobile webpages and 3D models of ancient buildings are also available as non open-data services for the city's users.

2.1.4 City of Lamia

At present, Lamia provides a rich set of services, interfaces and APIs that are currently used in existing tourism related applications or that can potentially be used for creating new touristic services and applications. For instance, Lamia provides a service[25] containing open data about the city's POIs (such as city monuments, recreation parks, museums, and others) and city specific resources (such as roads, fiber optics, water, etc). The city also provides a similar service[26] more oriented to the touristic guides providing the same information about the city POIs and routing services. Along with the read access to the mentioned data services about the city's POIs, the city also provides the possibility of an authenticated user, who has interest in updating the public database, to perform it by executing an update-check-post procedure.

New Web Site[27] of Municipality of Lamia also offers a list of events[28] occurring in the city. A visitor can find information about date and time, organizer, location, price, registration, etc. All the information about POIs[29] and events[30] are exposed through a Web service using REST/JSON/XML technology.

An API for transportation services, public transportation lines and routes is also provided by the city.

Finally, Lamia provides online information (website, csv file) though its weather information system that can be used by applications that involve weather and road conditions information. This system measures, collects and displays data in various channels.

2.1.5 City of Amsterdam

The city of Amsterdam currently presents open-data initiatives, which provide data openly regarding events and POIs, and non open-data services/applications to its users.

The *Amsterdam open data* initiative[31] makes available ongoing events around Amsterdam in a CSV file format, as well as databases containing collections from the Rijksmuseum and Amsterdam museum, and all monuments in The Netherlands. Additionally, a searchable API containing photos of places in Amsterdam is also provided.

[24] http://www.appia.itabc.cnr.it/
[25] http://www.lamia.gr/?q=tracker_aksiotheata_perioxis
[26] http://guide.lamia-city.gr
[27] Since Web Site is still in development mode credentials should be applied to access web pages. Username: citysdk Password: citysdk at link : http://www.lamia.gr/?q=user/login
[28] http://www.lamia.gr/?q=calendar_ekd/table
[29] http://www.lamia.gr/?q=tax-ypiresies-2/term/720
[30] http://www.lamia.gr/?q=tax-ypiresies-2/term/721
[31] amsterdamopendata.nl

Arts Holland[32] is an open-data project that intends to gather data from several Dutch cultural organizations, regarding various forms of Dutch art, events and POI's, mainly within the Randstad region of the Netherlands (Amsterdam, Rotterdam, Den Haag and Utrecht); provide a platform that supports all gathered data; and, ultimately, advertise what treasures the Randstad region holds dear, with special focus on tourists.

Among the city's provided services/applications, there is an offline mobile city guides (roaming free), named *I amsterdam*[33] and *I amstermam to go*[34] mobile city guide, which already count more than 150.000 downloads and enable their users to easily find internationally renowned museums and attractions, explore local markets and quirky boutiques, or sample the local delicacies at Amsterdam's best restaurants. These applications also provide top cultural events to its users.

Another application to highlight should be the *I amsterdam QR spots*[35] application, which enables the city users to discover the stories about several (150 in total) places in Amsterdam by scanning a QR code placed near the place.

Most of the applications built for the Amsterdam city by third parties are mainly city guides, travel guides and (interactive) city maps.

2.1.6 Summary of available data

The following table (Table 1) summarizes the current scenario among the pilot cities, describing, for each one of the touristic data categories (POIs list, City Events and other services) the available datasets and its openness.

City	POIs List	City Events	Other Services
Lisbon	Open/REST,XLS,XML	Closed	Provides open-data for several other data categories.
Helsinki	Open/REST	Open/XML	Open data for other categories as well. NFC/QR based applications.
Rome	Open/XML,CSV,TSV	Open/XML,CSV,TSV	Wi-Fi AP Map. Mobile App for daily Touristic Events (Roma&più) Closed AR applications.
Lamia	Open/REST,XLS	Open/REST	Open transportation data. Open weather data.
Amsterdam	Open	Open/CSV	Provides closed applications.

Table 1 – Summary of available data in the pilot cities

[32] http://www.artsholland.com/
[33] https://itunes.apple.com/pt/app/i-amsterdam/id338997290?mt=8
[34] https://itunes.apple.com/pt/app/i-amsterdam-to-go/id451112655?mt=8
[35] https://itunes.apple.com/pt/app/i-amsterdam-qr-spots/id493226373?mt=8

2.2 Usage Limitations

From the previous sub-section describing the currently available touristic applications and existing datasets for each one of the pilot cities, we find that it is very difficult to find a complete application containing all the data a tourist needs for its journey in the city. As example, among the applications for the majority of the pilot cities, there is a shortage of open-data information about the activities occurring in the cities (events information).

Another constraint to denote in the pre-CitySDK scenario is that because some of the applications are based on non-OpenData sources but provided by the company that develops the application or by third-party companies, there is a high probability of providing low quality (outdated or very limited in terms of volume or quality) data to the users. Even when using OpenData sources, in order to provide a rich set of information to the end user, application developers have to deal with multiple sources of information, such as REST webservices, SOAP webservices, XML files, XLS files, CSV files or even by parsing data from websites (web scraping). This multitude of data sources and high complexity of integration (some of them lack proper documentation) for each one of them represents a barrier to the application development process, resulting in the focus of the efforts in the data interfacing mechanisms instead of the end-user interface and application usability, resulting in poorer and less attractive applications.

2.3 Portability Issues

Currently, despite of the needed touristic data being similar among the different pilot cities (information about touristic points of interest and information about events occurring in the city), every one of the cities provides data in distinct forms and uses different data formats. Even for similar types of applications and using open-data provided by the cities, the differences in the characteristics of the available data makes it unfeasible to use the same application in distinct cities. In order to port an application from a city to another, it would be necessary to re-design its entire data layer, engine and user interface to accommodate the different data sources and specification (e.g. the data format and fields) so that little could be reused from the original version.

Typically in the porting process for a present touristic application, only the major functionalities and look and feel could be maintained with minor changes. For this reason, the non-existence of a CitySDK common layer has a huge impact on the portability of the touristic applications among the pilot cities, turning the task of porting the applications a heavy effort-demanding task. CitySDK aims to abolish this significant barrier, enabling appliances to be ported effortlessly among the participant cities, as the data interfaces and data formats are equal. This way, when a developer builds an application for CitySDK, he will have access to a well specified and document data source and also to data from all the participant cities without the need of any additional effort. This feature provided by the CitySDK represents a very strong incentive for the developers to build new innovative tourism related apps.

3 CitySDK Tourism API Requirements

The CitySDK Tourism API defined in this document answers the requirements gathered in the earlier months of the project, mostly in the context of WP2. By analysis of current applications deployed on the CitySDK participating cities, datasets available at the participating cities, talks with developers, workshops, meetings with different stakeholders and other initiatives, it was possible to identify the requirements for the Tourism Service.

The requirements for a CitySDK Tourism service are fulfilled by providing information on Points of Interest, Routes and Events and by providing access to that data.

3.1 Point of Interest

The Point of Interest (POI) is the basic piece of information for a tourist visiting a city. Tourists visit POIs and therefore, in order to plan and execute their trip, need to have access to basic information about the POI. Basic information about the POI is made up of the following content relative to each POI:

- **Name** - Each POI is identified by a name. This allows tourists to identify the POI.
- **Description** - POI should have a description which provides information about the POI. This information could cover its history, purpose or any other information useful to the tourist.
- **Location** - The POI’s location, given in coordinates, is essential for tourists to be able to locate the POI. It is also a crucial piece of information for integration into navigation applications.
- **Address** - An address should be provided as an alternative means of locating the POI. It’s also useful for contact by mail.
- **Telephone** and **other contacts** - A phone number is a basic form of contact and should be provided for POIs which have it (a statue will not have one). Other contemporary forms of contact include fax number, website, email address, skype handle. This data should be provided in a flexible way as new technologies will certainly appear in the future.
- **Opening hours** - Many tourist attractions or businesses aren’t open 24 hours a day. In order to plan their visits, tourists require information about the schedule of these POIs.
- **Price** - The price of an attraction is taken into consideration by a tourist when deciding what to visit.

This is the basic set of information to be made available for a POI, even though not all fields will be required to characterize every POI. As the final target audience is tourists, they are not all expected to speak the native language. As such, there should be the possibility of **providing data in more than one language**.

Applications are usually visual and tourism applications are expected to be even more so. POIs can thus be enriched with multimedia content such as **Photos** and **Videos**. Some municipalities, such as Lisbon, have multimedia archives which will enable some POIs to be enriched with photos, videos and even **sounds** or **music**. Other cities, such as Rome, have access to **3D models** of monuments. This will allow the tourist to perform “virtual visits”. The **shape** of the area occupied by a POI may also be useful for map applications.

However, just providing information about each POI within a city is not sufficient. Tourists visit cities and therefore, require **information on the city itself**. Information about the city can, thus, be provided in the form of a POI, representing the city. Other POIs would be within the city, thus, establishing a **relationship among POIs**, which would help tourist identify and browse POIs. E.g. a store would be within a shopping mall, which itself would be in a borough, which in turn would be within a city.

When visiting the city or a POI, a tourist would like to learn about applications developed specifically for or related to the city or POI. **Links to POI specific applications** would enable tourists to access these applications.

Classification into categories is a common way to help users browse through large amounts of data. POIs can be **categorized** using a classification scheme. Complementary to the categories, a POI could also have **tags** associated, in a way transversal to categories. E.g., both an hotel and a restaurant (possible categories) could be targeted at “young people”, a possible tag.

Technologies such as RFID, NFC or barcodes, allow rapid, automatic identification of objects. These could be used to identify POIs, enabling tourist to use, for instance, their mobile phones’ cameras to identify POIs in order to retrieve information about them. There should also be complementary ways to **identify a POI** using these technologies.

There are very popular attractions where tourists find themselves waiting in line for their turn to enter. This is a displeasing waste of time, which tourists would like to avoid. Some of these attractions may have access control mechanisms which enable wait time estimates to be provided, enabling tourists to avoid the long peak hour queues and plan their visit for the off-peak hours, which is both advantageous to the tourist and the POI managers. When these technologies are available, **current wait time to enter** should be provided. For other attractions, the **current capacity occupation** (i.e., percentage of the total capacity is being used) could be a better metric.

3.2 Route

Routes are a sequence of POI, grouped according to some logic. These would enable applications to guide tourist through a circuit. The thematic routes would group POIs targeted at a certain demographic of other type of group, in order to maximize their visits and increase the appeal of the attractions.

Besides the **sequence of POIs**, the route itself would have most of the same fields of a POI, enabling users to search, browse and learn about the route. These fields would include: **name, description, multimedia content, identification methods, shape, categories, tags** and **links to applications.**

3.3 Event

An event is something that only takes place within a certain period of time. It is different from opening hours, e.g., a museum would be a POI, but a temporary exhibit would be an event.

Some events are very important for tourists and can act as a source of attraction of tourists. It is therefore important to have a way to convey these events to the proper audience. Events are characterized by the same information as POIs. However, instead of a location, these take **place at a POI**. The **time of the event** is crucial to the characterization of the event.

3.4 Services to be provided

The main purpose of the CitySDK Tourism service should be to provide access to the POI, event and route information. For some developers it will make sense to **download all the information** relevant to their application for offline use or for processing and combination with other data. Other developers might prefer to developed online applications which **fetch up-to-date data** to present to the user whenever needed.

The later applications require efficient and flexible search methods to find the required information among all that will be made available by a service instance. Search methods should allow POIs, routes and events to be **searched according to their name, description, location, categories, tags, relation to other entities and time at which they take place (events).**

In order to search for categories or tags, users need to know what they are. As such, there should be a way to **retrieve the categories and tags** being used at each service instance.

For applications which keep a local copy of data, it is important to have a method for **detecting changes to objects** in order to refresh their local database.

It is conceivable that a single entity will be unable to create and maintain current a database big enough to cover all relevant information for a tourist, even if the scope is limited. For instance, a city would be willing to create an entry (POI) for a museum but would probably be unable to keep information about events or even about the art pieces on display up to date. This would even be more complicated if the museum wanted to provide real-time wait time information. The CitySDK Tourism service could, thus, be implemented by several participants in the tourism area, with each one specializing in providing detailed information on a limited area. A good example could be the aforementioned museum. The municipality would maintain a simple POI about the museum and that POI would include a link to the CitySDK Tourism Service server of the museum. This **delegation** would allow the museum to provide detailed information about the museum and its events and collections using its own server. Inside the museum, the tourist could use barcodes to identify art pieces and learn more about them (they would be POIs). The multimedia content could provide audio-guides.

4 CitySDK Tourism Service

This section contains the specified WP5 architecture and its software components to be developed during the next CitySDK project's phase. Based on the defined architecture and the expected demand retrieved from the CitySDK indicators for tourism, found in the project's DoW, the hardware requirements were also estimated and included in this section.

4.1 Architecture

This sub-section describes the proposed architecture for the Tourism CitySDK platform and its composing components.

As shown in Figure 1, in the Tourism CitySDK platform's architectural scenario the city acts as a data provider to the platform, providing one or more distinct touristic information datasets. As the CitySDK platform is built in a modular form by nature, it can support one or more Data Adapters, used to access the city's data providers, using its specific access form (web service, file, etc…) and format (xml, json, csv, etc…). This particularity of the platform ensures that it can be extended to support any possible touristic data source during its lifetime. For each one of the pilot sites, specific Data Adapters should be developed and configured by the pilot responsible to enable the platform to retrieve the available data from the sources.

Data retrieved through the Data Adapters is processed by the CitySDK Tourism Engine and stored in its own internal database using an optimal format and structure to maximize the readiness and throughput of the applications requests coming from the CitySDK Tourism Service Provider component. The CitySDK Tourism engine is responsible for periodically polling the available city's data sources and for performing updates to its internal database with the updated data elements.

The Touristic CitySDK data is exposed to its end-users (e.g., mobile or web applications) through the CitySDK Tourism Service Provider component. This component represents the front-end of the framework, exposing the specified methods and hiding the internal framework's details to the exterior, and is responsible for attending to the received data requests. For each received request, it queries the CitySDK Tourism Engine which retrieves and pipelines the data present on the CitySDK database to fulfil the request. As this component represents the front-end of the framework, special care should be taken to assure the framework's integrity during eventual received attacks such as DoS or security breach exploitations. There is also the possibility that the POIs provided by the CitySDK contain references (e.g., links to pages, images, videos or other resources/services) provided by external data providers; in this case, applications are responsible to access the data referred by the retrieved POI.

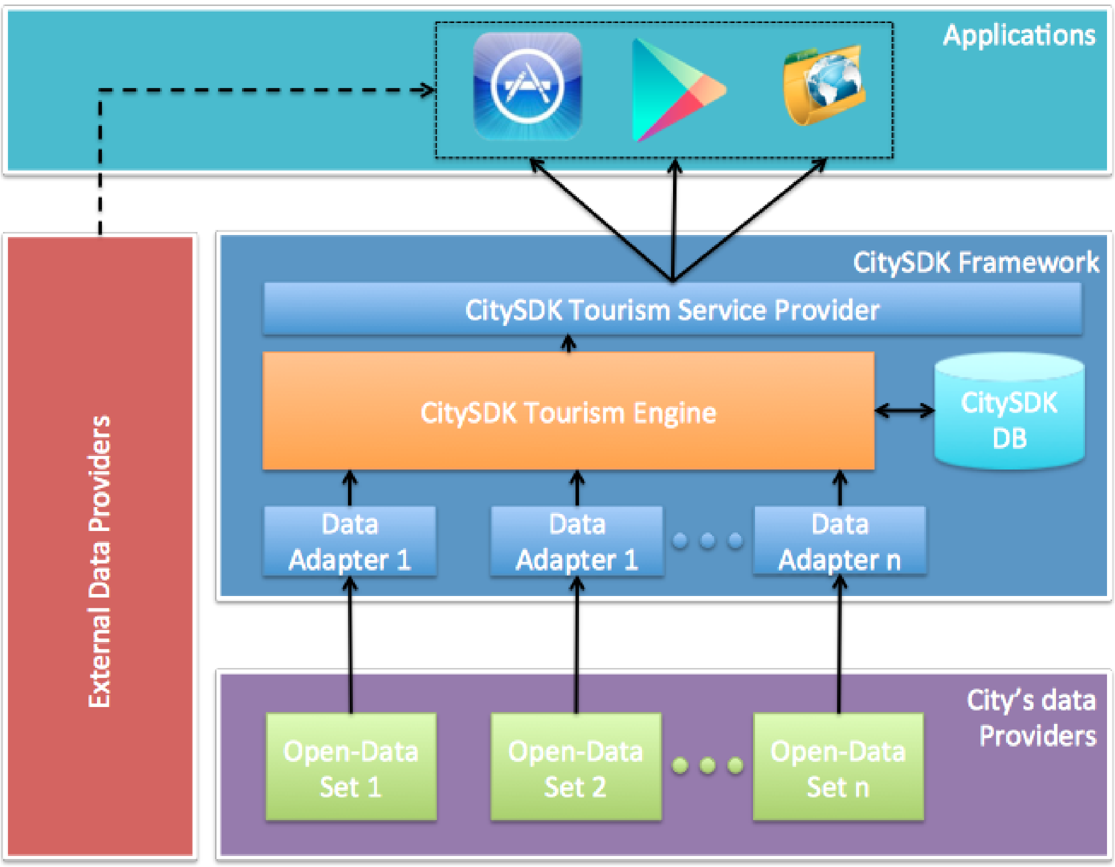


Figure 1 – CitySDK Tourism platform architecture

4.2 Software Components

The service is based on a main software component, the application server, aided by a platform's database, as shown in Figure 2. The application server contains all the software modules such as the Data Adapters, the CitySDK Tourism engine and the CitySDK Tourism Service Provider. The platform's database represents a relational database that contains all the platform's touristic data, ready for serving the requests originated from the applications.

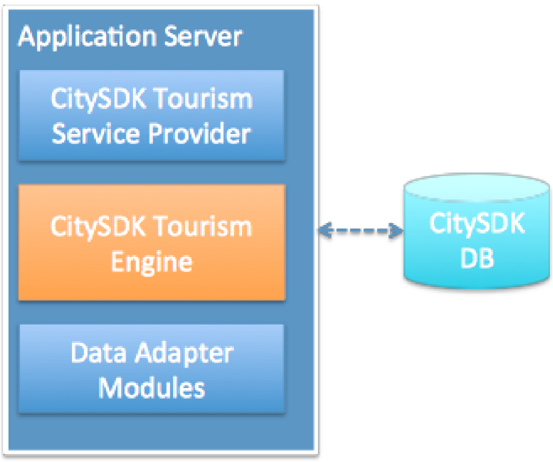


Figure 2 – Application server and database components

The application server is a software component that provides software applications with several facilities such as security, data services, transaction support, load balancing, and management of large distributed systems. The application server term was originally used for web servers that support the Java Platform, Enterprise Edition, however, currently its use isn't restricted to Java technologies.

For the CitySDK's application server, a wide set suitable of technologies are available. The most plausible alternatives for the development and deployment of the platform are:

- **Java based platform**

The JEE (Java Platform Enterprise Edition), formerly known as J2EE, defines a core set of interfaces and features of Java based application servers. In order to be JEE certified, application servers must meet all the requirements defined by the JEE standard. Currently, there are several commercial (i.e. non-open source) and open source application servers available in the market. In the commercial application servers we can find dominant solutions such as Oracle's WebLogic Application Server or IBM's WebSphere Application server. Among the open source application servers, we can also find several popular solutions such as JBoss from Red Hat, GlassFish from Oracle or Tomcat from Apache Foundation.

- **.Net Framework based platform**

The .Net Framework is a technology developed by Microsoft that runs mainly on Microsoft Windows systems and can also be used to implement the CitySDK's application server in the Microsoft's Internet Information Services (IIS) server.

Despite of the Microsoft's solution being non-open source and limited to the deployment on Microsoft Windows platforms, we can also find alternative third-party implementations of the .Net Framework on the market. The most popular .Net based alternative to Microsoft's implementation is the open source Mono project, owned by Novell Inc and licensed under GPL.

- **Other platforms**

Despite of being less popular that the later two technologies, there are other alternatives available that could be also fit to implement the CitySDK services.

For instance, for PHP technology there are also application servers available in the market. The most popular solution for PHP application server is the Zend Server, built by Zend Technologies. Other less popular application servers (or frameworks) based on Perl, such as Zope from Zope Inc or Django from Divio AG, and Python, such as Catalyst from Catalyst Foundation, could also be used for the implementation of the needed services.

In terms of databases, there are currently several suitable options regarding relational databases available in the market to be used in the CitySDK platform, such as:

- **Oracle**

Oracle is a very popular commercial (non-open source) relational database system owned by Oracle Inc., widely used in the idustrial/enterprise environment and known by its high performance under heavy loads. It can be deployed in the most enterprise environments such as Linux, BSD and Microsoft based operating systems.

- **Microsoft SQL Server**

SQL Server is the Microsoft's commercial (non-open source) relational database system. As it only can be deployed on Microsoft based environments, it is very popular on Microsoft based industries and enterprises.

- **Postgres**

PostgreSQL claims to be the world's most advanced open source relational database system. Just like MySQL, PostgreSQL is a very time-proven solution that strongly competes with commercial database software. PostgreSQL presents a densely featured database system which is sometimes presented as an open-source version of the commercial Oracle database system.

- **MySQL**

MySQL is by far the most popular open source database in the world. It is well known by its performance and by its extensively proven reliability. The MySQL database system is widely used by most of the web sites to store their data. Despite of being started as an independent open source project and maintaining the open source status, it was recently owned by the Oracle Corporation.

In the next phase of the project, an extensive evaluation will be made in order to choose one of the presented technologies. There are several factors to be taken into account in determining the optimal technology such as: technology openness - open source technologies should be privileged; costs of deployment - technologies that present lower costs of deployment should be preferred as they will provide benefits in terms of acceptance by future CitySDK cities; prior team experience - development team's most experienced and most dominated technologies should have higher preference.

4.3 Hardware Requirements

The hardware requirements found in this sub-section represent only a recommendation and were estimated from the expected demand extrapolated from the CitySDK indicators for the WP5 component of the project.

The relevant indicators needed for the assessment of the required capacity for pilot site's hardware will be the total number of distinct users of the tourism platform for the pilot cities. Assuming that typically, in the worst-case scenario at peak, 1% of the total pilot's users are actively using their touristic CitySDK based application (i.e., are actively connected to the CitySDK platform performing requests) at a specific time, and that each one of the users generates in mean one request to the platform per minute (the time needed for the tourist user to consume the retrieved contents from the platform), we can estimate the number of request/sec that the platform should assure and, consequently, the required hardware to power the platform. Despite of the software platform implementation details having some influence on the needed hardware power, a typical

SoA platform was considered on the estimation of the needed hardware.

Table 2 represents the estimation of the number of Open-Data end-users at the end of the project, present on the project's DoW indicators.

City	End-users at M30
Lisbon	3.000.000
Helsinki	40.000
Barcelona	50.000
Lamia	7.500
Rome	50.000

Table 2 – Number of expected end-users at M30

The mean users per pilot calculated from Table 2 is almost 630.000, consequently the estimated worst-case scenario of active users that CitySDK platform should support is 6.300, thus the recommended hardware platform for each pilot should be able to enable the platform to process 105 requests per second.

From the estimated number of requests per second needed to serve by a typical SoA platform, such as the CitySDK platform, the recommended hardware to fulfill the required estimated load for a pilot is represented on the next table. The table also contains a possible hardware solution for the pilots from the Dell hardware manufacturer/provider.

Hardware Component	Recommendation	Possible Hardware Solution
Chassis	Rack compatible chassis for installation in the cities' data servers room	PowerEdge M610x Blade Server
Processor	Single server grade processor with 4 cores, 12MB of cache and 2.40GHz or better	Intel Xeon E5620, 4C, 2.40GHz, 12M Cache, 5.86GT/s, 80W TDP, Turbo, HT, DDR3-1066MHz
Memory	16 GB of 1066MHz memory or better	16GB Memory for 1 CPU, DDR3, 1066MHz (4x4GB Dual Ranked LV RDIMMs), Using 1333MHz DIMMs
Hard Drive	600GB SAS hard drives (2 similar hard drives for RAID1 configuration)	2x 600GB, SAS 6Gbps, 2.5-in, 10K RPM Hard Drive (Hot Plug)
Hard Drive Controller	RAID1 capable hard drive controller with 1GB of cache memory	C3 R1 with PERC H200M, Exactly 2 SATA/SAS or SSD Drive
Power supply	Power supply compatible with the chassis, redundant if possible	Chassis power supply
Network Connectivity	Dual gigabit ethernet network interface card	Embedded Dual Port Gigabit Ethernet Controller with 2P TOE for Fabric A

Table 3 – Recommended hardware

5 Standards for Tourism

As already mentioned, a barrier to the tourism application portability among different cities is the diversity and multitude of proprietary data formats and transport mechanisms presently used to exchange data regarding POIs and events. Standardization is often seen as the ideal solution to eliminate this kind of barriers.

As an effort to overcome this heterogeneity in data representation and exchange, two standard initiatives emerged: the International Press Telecommunications Council’s EventsML-G2 standard (IPTC EventsML-G2) and the World Wide Web Consortium’s Point of Interest Working Group (W3C POI WG), the last one being launched in 2010. The former aimed at formalizing a standard for conveying event information in a news industry environment, whilst the latter aimed at developing technical specifications for the representation of POI information on the Web. Specifications of both EventsML-G2 and W3C POI are presented in this section.

Two other standards are also presented later in this section - vCard and iCalendar -, since both can be used in W3C POI to represent certain attributes in a more structured and detailed way, and iCalendar’s Event component can very well be compared to EventsML-G2, as the first standard’s features are virtually covered by the second [1].

5.1 EventsML-G2

EventsML-G2 is a member of the family of IPTC G2-Standards, which is built on a common structural and function framework called the IPTC News Architecture (NAR). The EventsML-G2 specifications build on the NewsML-G2 structural specifications and add a well defined functionality for conveying events.

Additionally, EventsML-G2 makes use of industry standards, since its syntax is built on W3C’s XML, furthermore, EventsML-G2 makes use of W3C XML Schema and complies with the basic notion of the Semantic Web, the Resource Description Framework (RDF).

The specification [1] defines several elements, element and attribute groups, and datatypes for the NewsML-G2 specification. For the sake of simplicity, we will only present the most relevant elements/properties in the event context.

5.1.1 Properties

Table 4 presents the generic properties which characterize a given event, whilst Table 5 enters into more detail, presenting event-specific properties.

Property	Description	Value Type
Name (*name*)	Name of the event. Should be rather concise and expressed in different languages.	IntlStringType
Definition (*definition*)	More extensive than the name, can also be expressed in different languages.	BlockType

Note (*note*)	Explanation of details or additional information regarding the definition. May also be expressed in different languages.	BlockType
Relationship	Used to define a relationship between this event and another one. **sameAs** – This event is equivalent of another. **broader** – This event is a sub-event of another. **narrower** – An identifier to a more specific event or sub-event. **related**	FlexPropType *or* FlexProp2Type

Table 4 – Generic event properties

Property	**Description**	**Value Type**
Dates (*dates*)	Sub-structure to express the start and end date, or duration of the event. If this event is recurring, it can be expressed by means of recurrence properties which align to equivalent properties of the **iCalendar** standard [2].	-
Occurrence Status (*occurStatus*)	Indicates the certainty of the occurrence of the event - if this is an unplanned or planned event, and if it is planned how likely it is to occur.	QCodePropType
Registration (*registration*)	Information which may be used to define how and when persons have to register for the event. May also include information about cost.	BlockType
Access Status (*accessSatus*)	Indicates the accessibility of the event.	QCodePropType
Participation Requirement (*participationRequirement*)	Represents the requirements for participating in an event. May be used, e.g., to express age limits (e.g., required parental guidance for movies) or for formal requirements for training course events.	FlexPropType
Subject (*subject*)	Expresses what the event is about, the event’s content.	FlexPropType
Location (*location*)	Location (geographical area or POI) where the event is taking place. However, it should be noted that, e.g., festivals could have more than one location.	FlexPropType
Participant (*participant*)	Lists all kinds of parties (persons or organizations) appearing in different roles at the event.	FlexPropType
Organiser (*organiser*)	Lists all parties (persons or organizations) involved in organizing the event	FlexPropType
Contact Information	The *location*, the *participant* and the *organiser* properties may contain *contactInfo* structures, but they pertain only to	-

(*contactInfo*)	this particular property, while this *contactInfo* is to be used for the event as a whole. Child elements defined are the following: **email –** An email address. **im** – An instant message system address. **phone** – An international phone number. **fax** – An international fax number. **web** – A Web address. **address** – A line of address information. **note** – Additional natural language information. **extension point** – any set of provider-defined properties.	
Language (*language*)	Reflects all languages that will be spoken at the event.	-

Table 5 – Event-specific properties wrapped by the **eventDetails** property

The following properties, presented in Table 6, consist in elements which may be included in the event information, in order to specify its location, either further venue/POI details or geopolitical area where it will take place.

Property	Description	Value Type
POI Details (*POIDetails*)	A group of properties specific to a POI, such as: **position** – The geographic coordinates of the location. **openHours** – Opening hours of the place, in natural language. **capacity** – Total capacity of the place, in natural language. **access** – Ways to access the place, including directions. **details** – Location details. **contactInfo –** Contact information of the POI. **created –** Date of creation of the POI. **ceasedToExist** – Date at which the POI ceased to exist. **extension point** – Set of provider-defined properties.	-
Geopolitical Area Details (*geoAreaDetails*)	A set of properties specific for a geopolitical area. Child elements that may be included in this property are the following: **position** – The geographic coordinates of the location. **line** – Defines a line as a geographic area by listing two or more points. **circle** – Definition of a circular geometry as a geographic area. **polygon** – Defines a polygon as geographic area by a	-

	listing of three or more points. **founded** – Date of Foundation of Geopolitical Area. **dissolved** – Date of Dissolution of Geopolitical Area. **extension point** – Set of provider-defined properties.	
Postal Address (*address*)	A registered address of an entity (person or organization). Children elements which describe a postal address are the following: **line –** A line of address information, in the format expected by a recipient postal service. **locality –** A city, town, village, etc. **area –** A subdivision of a country. **country –** The country. **postalCode** – A postal code.	-

Table 6 – Additional entities relevant in the POI/Event context

5.1.2 Use Case

The following example represents the conveying of two events which are related to one another. The first event consists of the 3 day duration *IPTC Autumn Meeting 2012*, and the second a sub-event of a *broader* event - the first event -, which consists of a 1-hour meeting scheduled on the first day of this 3-day meeting.

```
<?xml version=“1.0” encoding=“UTF-8”?>
<knowledgeItem
  xmlns="http://iptc.org/std/nar/2006-10-01/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://iptc.org/std/nar/2006-10-01/ XSD/NewsML-G2_2.12-spec-All-Power.xsd"
  guid="urn:newsml:iptc.org:20101019:qqwpiruuew4712"
  version="1"
  standard="NewsML-G2"
  standardversion="2.12"
  conformance="power"
  xml:lang="en">
      <catalogRef href="http://www.iptc.org/std/catalog/IPTC-G2-standards_19.xml" />
      <catalogRef href="http://www.example.com/events/event-catalog.xml" />
     <itemMeta>
         <itemClass qcode="cinat:concept" />
         <provider literal="IPTC" />
         <versionCreated>2010-10-26T12:00:00Z </versionCreated>
         <pubStatus qcode="stat:usable" />
    </itemMeta>
    <contentMeta>
        <urgency>5</urgency>
        <contentCreated>2012-10-16T12:15:00Z</contentCreated>
        <contentModified>2012-10-16T14:35:00Z</contentModified>
        <subject qcode="subj:04010000">
            <name>media</name>
        </subject>
        <subject qcode="subj:04010004">
             <name>news agency</name>
        </subject>
    <subject qcode="subj:13022000">
       <name>IT/computer sciences</name>
    </subject>
  </contentMeta>
  <conceptSet>
     <concept>
<!-- FIRST EVENT! -->
<!-- x -->
      <conceptId created="2012-10-16T12:15:00Z" qcode="event:1234567" />
      <name>IPTC Autumn Meeting 2012</name>
      <eventDetails>
        <dates>
          <start>2012-10-22T09:00:00-04:00</start>
          <duration>P3D</duration>
        </dates>
        <registration> Registration with the IPTC office is required.
          A <a href="http://www.example.com/register"> web form </a> may be used until 02 October 2012
        </registration>
        <participationRequirement>
          <name>Membership</name>
          <definition>Only members of the IPTC and their invited guests may attend.</definition>
        </participationRequirement>
        <accessStatus qcode="accst:easy" />
        <language tag="en" />
        <organiser literal="IPTC" role="orgrole:mainOrganiser">
          <name>International Press Telecommunications Council</name>
          <organisationDetails>
            <founded>1965</founded>
          </organisationDetails>
        </organiser>
        <contactInfo>
          <email>mdirector@ipct.org</email>
          <note>Michael Steidl, Managing Director</note>
          <web>http://www.iptc.org</web>
        </contactInfo>
        <location literal="RiaNovosti">
          <name>Ria Novosti, Zubovsky Boulevard 4, Moscow, Russia</name>
          <related rel="frel:venuetype" qcode="ventyp:office" />
          <POIDetails>
            <position latitude="55.737126" longitude="37.591542" />
            <contactInfo>
              <web>http://en.rian.ru/</web>
            </contactInfo>
          </POIDetails>
        </location>
        <participant literal="StephaneGuerillot">
          <name>Stéphane Guérillot</name>
          <definition role="drol:jobtitle">IPTC Chairman</definition>
        </participant>
        <participant literal="MichaelSteidl">
          <name>Michael Steidl</name>
          <definition role="drol:jobtitle">Managing Director</definition>
        </participant>
      </eventDetails>
    </concept>
    <concept>
```

```
<!-- SECOND EVENT! -->
<!-- x -->
      <conceptId created="2012-09-30T12:00:00+00:00" qcode="event:91011123" />
      <name>Annomarket text analytics EU project</name>
      <broader type="cpnat:event" qcode="event:1234567">
        <name>IPTC Autumn Meeting</name>
      </broader>
      <eventDetails>
        <dates>
          <start>2012-10-22T10:00:00-04:00</start>
          <duration>PT1H</duration>
        </dates>
        <participationRequirement>
          <name>Membership</name>
          <definition>Only members of the IPTC and their invited guests may attend </definition>
        </participationRequirement>
        <accessStatus qcode="accst:easy" />
        <language tag="en" />
        <participant literal="JaredMcGinnis">
          <name>Jared McGinnis</name>
          <definition role="drol:sessionrole">Presenter</definition>
        </participant>
        <participant literal="MichaelSteidl">
          <name>Michael Steidl</name>
          <definition role="drol:sessionrole">Moderator</definition>
        </participant>
      </eventDetails>
    </concept>
  </conceptSet>
</knowledgeItem>
```

Figure 3 – Conveying of two related events information, based on the example in **[3]**

5.2 W3C POI

On May 2011, the first working draft of the W3C POI Core Recommendation was publicly released. The last working draft dates back to March 2012 [4]. We begin by presenting this last working draft, due to the absence of a final recommendation version.

The W3C POI [4] aimed at defining a generic, flexible, lightweight and extensible POI data model, and one normative syntax for the data model based on XML. Nevertheless, the possibility of mapping the data model to other formats, such as JSON, GML, RDF, GeoRSS or HTML, was also considered.

We shall start by presenting the data model defined in W3C POI and, next, a use case in both formats, XML and JSON.

5.2.1 W3C POI Core Data Model

According to the illustrated in Figure 4, the core data model comprises five main entities: **POIBaseType**, **POIType**, **POIS**, **POI**, **Location** and **POITermType**.

The *POIType* entity derives and has child entities derived from an abstract *POIBaseType* entity. Other child entities are constructed from primitive data types or *POITermType* entities. The common *POIBaseType* allows the authorship, modification tracking (creation, update and deletion) and source of POIs to be attributed at multiple levels of granularity within the data model, namely at the POI level, at the child entity level, and at a group level through the POIS grouping entity.

All entities are presented in more detail next.

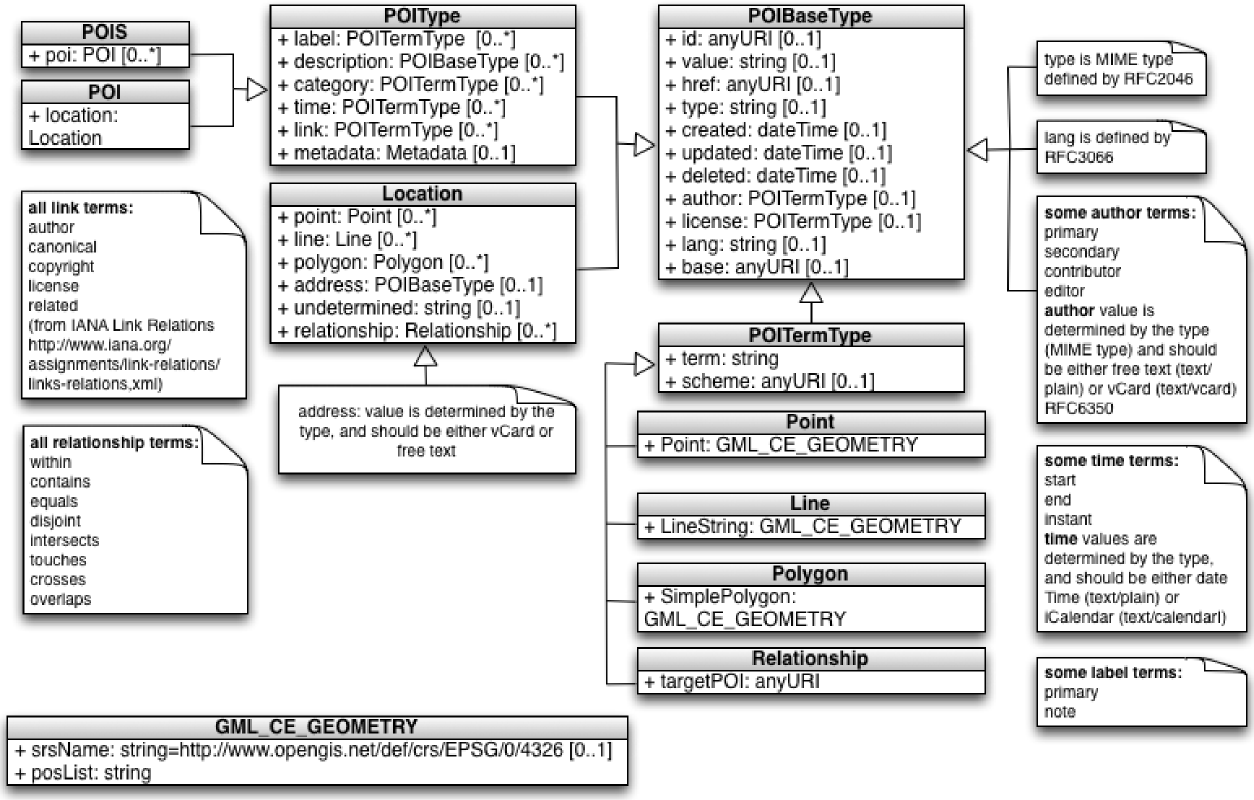

POIS
+ poi: POI [0..*]
POI
+ location: Location
POIType
+ label: POITermType [0..*]
+ description: POIBaseType [0..*]
+ category: POITermType [0..*]
+ time: POITermType [0..*]
+ link: POITermType [0..*]
+ metadata: Metadata [0..1]
Location
+ point: Point [0..*]
+ line: Line [0..*]
+ polygon: Polygon [0..*]
+ address: POIBaseType [0..1]
+ undetermined: string [0..1]
+ relationship: Relationship [0..*]
POIBaseType
+ id: anyURI [0..1]
+ value: string [0..1]
+ href: anyURI [0..1]
+ type: string [0..1]
+ created: dateTime [0..1]
+ updated: dateTime [0..1]
+ deleted: dateTime [0..1]
+ author: POITermType [0..1]
+ license: POITermType [0..1]
+ lang: string [0..1]
+ base: anyURI [0..1]
POITermType
+ term: string
+ scheme: anyURI [0..1]
Point
+ Point: GML_CE_GEOMETRY
Line
+ LineString: GML_CE_GEOMETRY
Polygon
+ SimplePolygon: GML_CE_GEOMETRY
Relationship
+ targetPOI: anyURI
GML_CE_GEOMETRY
+ srsName: string=http://www.opengis.net/def/crs/EPSG/0/4326 [0..1]
+ posList: string
all link terms: author canonical copyright license related (from IANA Link Relations http://www.iana.org/assignments/link-relations/links-relations.xml)
all relationship terms: within contains equals disjoint intersects touches crosses overlaps
address: value is determined by the type, and should be either vCard or free text
type is MIME type defined by RFC2046
lang is defined by RFC3066
some author terms: primary secondary contributor editor author value is determined by the type (MIME type) and should be either free text (text/plain) or vCard (text/vcard) RFC6350
some time terms: start end instant time values are determined by the type, and should be either date Time (text/plain) or iCalendar (text/calendar)
some label terms: primary note

Figure 4 – W3C POI core data model **[4]**

POIBaseType entity

The *POIBaseType* is the common entity from which the majority of W3C POI entities are derived and to which the properties presented in Table 7 are provided. This base type allows the *POI* entity, *POIS* grouping entity and most child entities to carry distinct information about their provenience, source and history.

Property	Multiplicity	Description	Value Type
id	[0..1]	A unique identifier for the POI. It is recommended the use of a unique URL or a resolvable combination of *id* and *base* URI.	URI
href	[0..1]	An absolute reference to the POI content, when the *id* and *base* do not combine to form a resolvable URL.	URI
value	[0..1]	The POI information content. It is also the recommended location for arbitrary	string

		strings for some child entities/human readable categorical descriptions, e.g., label, description.	
base	[0..1]	An XML base (**xml:base**) URI when *id* is not absolute, but instead a URI fragment.	URI
type	[0..1]	MIME type [5].	string
lang	[0..1]	Language type [6].	string
updated	[0..1]	Time at which the POI information was last modified.	dateTime
created	[0..1]	Time at which the POI information was created.	dateTime
deleted	[0..1]	Time at which the POI information was deleted.	dateTime
author	[0..1]	Author of the POI information. The *value* property specifies the actual author and is determined by the *type* property (MIME type). When the mime *type* property is undetermined or text/plain, the content of the *value* property is assumed to be free text. When the mime *type* property is text/vcard, the *value* content is expected in a vCard format [7].	POITermType
license	[0..1]	License restrictions on the POI information. The language the license is in may be indicated using the *lang* property. The first license encountered per language is considered the primary license for that language.	POITermType

Table 7 – *POIBaseType* entity properties

POIType entity

The *POIType* entity is an abstract entity derived from *POIBaseType* that adds entities for describing, labeling, categorizing and indicating the time span of a POI or group of POIs, as well as child entities for linking to other POIs, external Web resources or metadata.

Property	Multiplicity	Description	Value Type
label	[0..*]	A human-readable name for the POI. The label’s language may be indicated using the *lang* property, where the first label encountered per language is considered the primary label for that language.	POITermType
description	[0..*]	A human-readable POI description. The description language may be indicated using the *lang* property. The first description encountered per language is considered the primary description for	POIBaseType

		that language.	
category	[0..*]	A categorical classification of the POI, using keywords as a value or tag, such as city, restaurant, museum, etc. It is adopted from the **atom:category** object [8]. Multiple categories are allowed to accommodate the fact that POIs may be more than one thing (e.g., a casino might be a gambling hall, a restaurant and a concert venue). The *href* property may be used to provide a resolvable link to a more comprehensive definition of the category.	POITermType
time	[0..*]	The POI may have a known time when it came into being, i.e., a start date, or a known point in time when it might cease to exist (in the future or in the past), thus, having an end date. Also, a POI can exist on a regularly scheduled sequence of times. The content *value* is assumed to be in the **dateTime** format [9] when the MIME *type* attribute is undetermined or text/plain, or in the iCalendar [2] format when the MIME *type* attribute is text/calendar.	POITermType
link	[0..*]	A link to another POI or Web resource. It was adopted from the **atom:link** object [8].	POITermType
metadata	[0..1]	Where formal metadata (Dublin Core or [10]) is attached to the POI via reference or inline.	Metadata

Table 8 – *POIType* entity properties

POIS entity

Derived from the *POIType* entity, the *POIS* entity can have one or group several children entities of type *POI*.

POI entity

The *POI* entity inherits properties for capturing descriptive information from the *POIType* entity, and adds the *Location* entity which describes the POI’s location.

Location Entity

The *Location* entity is a required POI sub-entity, which provides a flexible description of a given POI’s location. It contains child entities which describe a *geometry* (e.g., the geodetic coordinates for the center of the POI, a line, bounding box or a polygon), *address* or an *undetermined* entity. If a *Location* neither includes a *geometry* entity nor an *address* entity, then it must include the *undetermined* entity.

Property	Multiplicity	Description	Value Type
point	[0..*]	A single coordinate.	GML_CE_Geometry
line	[0..*]	Two or more coordinates.	GML_CE_Geometry
polygon	[0..*]	Three or more coordinates.	GML_CE_Geometry
address	[0..*]	A civic address.	POIBaseType
undetermined	[0..*]	Indicates that the location is undetermined.	None
relationship	[0..1]	A geo-spatial relationship to one or more POI.	Relationship

Table 9 – *Location* entity properties

GML_CE_Geometry

The base entity from which all geometry entities are derived. It combines a coordinate reference system description with a set of coordinates, as presented in Table 10.

Property	Multiplicity	Description	Value Type
srsName	[0..1]	Coordinate reference system (CRS) URI being used. The World Geodetic System 84 (WGS84)[36] in 2 dimensions - latitude and longitude - is used by default.	string
PosList	1	Contains coordinate set.	string

Table 10 – *Geometry* entity properties

Point

A single *Point* entity is added, describing a single coordinate set, the interpretation of which is influenced by the coordinate reference system (CRS) defined in the *srsName* attribute.

The Point entity locates the centroid of the POI and is the most common way of specifying a location. For most places, such as cities, businesses, tourist sites or events, a center location can be useful for many types of software applications where additional detail is unnecessary, such as driving directions or computing rough distances. Therefore, even if the POI is also specified with a polygon or a line, it is good practice to include a center point.

Line

A single *LineString* entity is added, describing a list of two or more coordinate sets, the interpretation of which is influenced by the CRS defined in the *srsName* attribute.

Polygon

A single *SimplePolygon* entity is added, describing a list of three or more coordinate sets, the

[36] http://www.opengis.net/def/crs/EPSG/0/4326

interpretation of which is influenced by the CRS defined in the *srsName* attribute.

Address

This entity describes a civic address such as a mailing or street address. Its value is determined by the *type* property, which should be either a vCard (text/vcard) or free text (text/plain).

Undetermined

This entity represents a location that is yet to be determined, hence, it can be used to describe a POI prior to the final location being set. No properties were specified for this entity.

Relationship

Establishes a 1-to-1 or 1-to-many relationships between POIs, more specifically, a geo-spatial relationship between the POI and another indicated by the target *POI* property, in terms of the dimension of intersections of their boundaries, interiors and exteriors. The SQL/MM Spatial standard [11] is used for this purpose.

POITermType

The *POITermType*, presented in Table 11**,** is an abstract entity that derives from the *POIBaseType* entity and adds properties for the management of categorical descriptions.

Property	Multiplicity	Description	Value Type
term	[0..1]	A machine-readable character string to designate any number of discrete choices.	string
scheme	[0..1]	An absolute reference to the schema where the discrete choices in the *term* property are enumerated.	URI

Table 11 – POITermType entity properties

Many of the properties from the previously presented entities are instances of *POITermType*, as they require additional categorical information, namely:

- **Author** - A single author, whose authorship role may be discriminated using the *term* property. The recommended terms are primary, secondary, contributor, editor and publisher. Multiple authors can be specified by nesting *author* properties;
- **License** - A single license, however, multiple licenses may be discriminated with the *term* property (e.g., common, opensource, etc.);
- **Label** - Any number of human-readable labels, which may be discriminated with the *term* property, assuming either the primary or note recommended terms;
- **Category** - The *term* property is the classification scheme to which the category value belongs, while the *scheme* property may specify a dictionary of terms or a Web resource containing definitions of the terms and values;
- **Time** - The *term* property discriminates between different time entity categories, namely

the recommended:

  - start - Time when the POI came into being;
  - end - Time when the POI ceased to exist;
  - instant - A single time when an event happened;
  - open - A recurring time when a POI is open.

- **Link** - A number of best practices is proposed for defining links, namely the key relations from the IANA registry, among which:
  - alternate - An identical POI. Often used as a permalink;
  - canonical - The preferred version among a set of POIs with highly similar content;
  - copyright - A copyright statement that applies to the link’s context;
  - describedby - Link which provides more information about this POI;
  - edit - A resource that can be used to edit the POI's context;
  - enclosure - A related resource that is potentially large and might require special handling;
  - icon - Refers to an icon representing the link's context;
  - latest-version - Points to a resource containing the latest version;
  - license - Refers to a license associated with this POI;
  - related - Identifies a related resource;
  - search - A resource that can be used to search through the link's context and related resources;

  Additional relations were defined only within this specification’s context, namely:

  - parent - A parent POI, often the enclosing *geography* entity, or the entity this POI in under the domain of;
  - child - A child POI, often a *geography* entity enclosed or under the domain of this POI;
  - historic - Links to a POI or other Web resource that describes this place at a previous point in time;
  - future - Links to a POI or other Web resource that describes this place at a later point in time.
- **Point** - The *term* property is required to discriminate between different *Point* entity categories, namely:
  - center - The centroid of the POI location;
  - navigation point - Generic navigation point;
  - entrance – Navigation point to the entrance.
- **Relationship** - Each *Relationship* entity must be assigned one of 8 terms that describe the geo-spatial relationship to the POI indicated by the target *POI* property:
  - equals - If two POI geometries are spatially equal;
  - disjoint - If two POI geometries have no points in common, i.e., don’t intersect;
  - intersects - If two POI geometries have points in common;
  - crosses - If both interiors intersect with a dimension less that the larger of the dimension of one POI geometry of the other, and the intersection of the interior of

the crossing geometry with the exterior of the crossed one is not null;

- overlaps - If parts of the two POI geometries' respective interiors intersect, and if this intersection is the same dimension as the interiors of the original geometries;
- within - A POI is located inside the other, e.g., a POI describing a store may state that it is contained within a shopping mall;
- contains - A POI contains the other, i.e., a POI describing a mall may state that it contains POIs for each store that is within the mall;
- touches - One POI's boundary intersects with the other's interior or boundary, e.g. a POI representing a store within a mall may state that it is next door to another POI which represents the store next door.

5.2.2 Use Case

In this section, we present an example based on the one found in the recommendation[37], which describes one of UNESCO's World Heritage sites - the Taj Mahal. Figure 5 presents such example using XML.

```
<poi base="http://www.rajsingh.org/pois/" id="25245435" lang="EN-US" created="20111004T13:33:21-5:00">
  <author href="http://www.rajsingh.org/me.rdf" type="application/rdf+xml">
    <label>
      <term>primary</term>
      <value>Taj Mahal</value>
    </label>
    <description updated="20110928T09:09:00-5:00">
      <value>An immense mausoleum of white marble, built in Agra between 1631 and 1648 by order of the
Mughal emperor Shah Jahan in memory of his favourite wife, the Taj Mahal is the jewel of Muslim art in India and
one of the universally admired masterpieces of the world's heritage.</value>
      <author id="http://whc.unesco.org" href="http://whc.unesco.org/en/list/252" type="text/html">
        <value>UNESCO</value>
      </author>
    </description>
    <description lang="FR">
      <value>Immense mausolée funéraire de marbre blanc édifiée entre 1631 et 1648 à Agra sur l'ordre de
l'empereur moghol Shah Jahan pour perpétuer le souvenir de son épouse favorite, le Taj Mahal, joyau le plus
parfait de l'art musulman en Inde, est l'un des chefs-d'œuvre universellement admirés du patrimoine de l'huma-
nité.</value>
      <author id="http://whc.unesco.org" href="http://whc.unesco.org/en/list/252" type="text/html">
        <value>UNESCO</value>
      </author>
    </description>
    <category scheme="http://whc.unesco.org/en/list/">
      <term>World Heritage Site</term>
    </category>
    <category scheme="http://geonames.usgs.gov/pls/gnispublic/f?p=gnispq:8:4433905879145750">
      <term>Park</term>
    </category>
    <category scheme="http://geonames.usgs.gov/pls/gnispublic/f?p=gnispq:8:4433905879145750">
      <term>Cemetery</term>
    </category>
```

[37] http://www.w3.org/2010/POI/wiki/A_UNESCO_World_Heritage_site

```
    <link href="http://www.rajsingh.org/pois/25245435.xml">
      <term>canonical</term>
    </link>
    <link href="http://whc.unesco.org/en/list/252" type="text/html">
      <term>source</term>
    </link>
    <link href="http://whc.unesco.org/uploads/sites/gallery/original/site_0252_0001.jpg" type="image/jpeg">
      <license href="http://whc.unesco.org/en/disclaimer/"/>
      <term>related</term>
    </link>
    <link href="http://whc.unesco.org/uploads/sites/gallery/original/site_0252_0010.jpg" type="image/jpeg">
      <license href="http://whc.unesco.org/en/disclaimer/"/>
      <term>related</term>
    </link>
    <location>
      <point>
        <term>entrance</term>
        <Point srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
          <posList>27.171014 78.043694</posList>
        </Point>
      </point>
    </location>
</poi>
```

Figure 5 – Taj Mahal XML format example

In the example illustrated in Figure 5, the POI is described in terms of the base URI where it is located, its ID, language, the date and time at which it was created, as well as who added this POI (the author).

The definition of several descriptions is possible, or the same description may coexist in several languages (illustrated in both English (default) and French). Since the POI description itself is an instance of *POIBaseType*, it may also include information regarding its author, creation/deletion/update times, among other information, besides the content within the *value* property and the *language* property.

Many categories for the same POI may also coexist, according to different Web resources, or yet if the POI comprises more than one purpose, i.e., referring to our use case, besides a World Heritage site, the Taj Mahal is as much of a park as it is a cemetery.

Four links that reference this POI are presented, two of which correspond to images of the POI.

Finally, the POI's location is described in terms of the geographical coordinates of its entrance point.

Figure 6 illustrates the same example as Figure 5, however, this time mapped into the JSON format.

```
{
  "base":"http:\/\/www.rajsingh.org\/pois\/",
  "id":"25245435",
  "lang":"EN-US",
  "created":"20111004T13:33:21-5:00",
  "author":
```

```
  {
    "href":"http:\/\/www.rajsingh.org\/me.rdf",
    "type":"application\/rdf+xml"
  },
  "label":
  {
    "term":"primary",
    "value":"Taj Mahal"
  },
  "description":
  [
    {
      "value":"An immense mausoleum of white marble, built in Agra between 1631 and 1648 by order of the
Mughal emperor Shah Jahan in memory of his favourite wife, the Taj Mahal is the jewel of Muslim art in India and
one of the universally admired masterpieces of the world's heritage.",
      "updated":"20110928T09:09:00-5:00",
      "author":
      {
        "id":"http:\/\/whc.unesco.org",
        "href":"http:\/\/whc.unesco.org\/en\/list\/252",
        "type":"text\/html",
        "value":"UNESCO"
      }
    },
    {
      "lang":"FR",
      "value":"Immense mausol\u00e9e fun\u00e9raire de marbre blanc \u00e9difi\u00e9e entre 1631 et 1648
\u00e0 Agra sur l'ordre de l'empereur moghol Shah Jahan pour perp\u00e9tuer le souvenir de son \u00e9pouse
favorite, le Taj Mahal, joyau le plus parfait de l'art musulman en Inde, est l'un des chefs-d'\u0153uvre universelle-
ment admir\u00e9s du patrimoine de l'humanit\u00e9.",
      "author":
      {
        "id":"http:\/\/whc.unesco.org",
        "href":"http:\/\/whc.unesco.org\/en\/list\/252",
        "type":"text\/html",
        "value":"UNESCO"
      }
    }
  ],
  "category":
  [
    {
      "term":"World Heritage Site",
      "scheme":"http:\/\/whc.unesco.org\/en\/list\/"
    },
    {
      "term":"Park",
      "scheme":"http:\/\/geonames.usgs.gov\/pls\/gnispublic\/f?p=gnispq:8:4433905879145750"
    },
    {
      "term":"Cemetery",
      "scheme":"http:\/\/geonames.usgs.gov\/pls\/gnispublic\/f?p=gnispq:8:4433905879145750"
    }
  ],
  "link":
  [
    {
            "href":"http:\/\/www.rajsingh.org\/pois\/25245435.xml",
            "term":"canonical"
        },
        {
            "href":"http:\/\/whc.unesco.org\/en\/list\/252",
            "type":"text\/html",
            "term":"source"
        },
        {
            "href":"http:\/\/whc.unesco.org\/uploads\/sites\/gallery\/original\/site_0252_0001.jpg",
            "type":"image\/jpeg",
            "license":
            {
                "href":"http:\/\/whc.unesco.org\/en\/disclaimer\/"
            },
            "term":"related"
        },
        {
            "href":"http:\/\/whc.unesco.org\/uploads\/sites\/gallery\/original\/site_0252_0010.jpg",
            "type":"image\/jpeg",
            "license":
            {
                "href":"http:\/\/whc.unesco.org\/en\/disclaimer\/"
            },
            "term":"related"
        }
    ],
    "location":
    {
        "point":
        {
            "term":"entrance",
            "Point":
            {
                "srsName":"http:\/\/www.opengis.net\/def\/crs\/EPSG\/0\/4326",
                "posList":"27.171014 78.043694"
            }
        }
    }
}
```

Figure 6 – Taj Mahal JSON format example

5.3 vCard

vCARD (text/vcard MIME content type) is an IETF (Internet Engineering Task Force) Internet Standards Track document [7] that specifies a data format, resembling an electronic format of a business card, for representing and exchanging a variety of information about individuals and other entities. This information comprises, e.g., formatted and structured name and delivery addresses, email address, multiple telephone numbers, photograph, logo, audio, clips, etc. It is may be used in the W3C POI recommendation to represent authors or addresses.

vCard allows individuals and entities to be described by means of its properties, which, in turn, can be characterized by means of property parameters. All properties and parameters are presented next.

Other ways of representing contact information were defined, namely the xCard [12], which uses an XML representation, and still a JSON representation [13]. Both formats will be presented using a common example, which shall render an easier understanding of the mapping from the vCard plain syntax to XML and JSON.

5.3.1 vCard Format Specification

vCard Properties

vCard properties are grouped into **General**, **Identification**, **Delivery Addressing**, **Communication**, **Geographical**, **Organizational**, **Explanatory**, **Security** and **Calendar** properties.

General Properties

These properties, presented in Table 12, define the vCard object, as a whole.

Property	Multiplicity	Description	Value Type
BEGIN	1	The use of this property is mandatory at the beginning of a syntactic entity within a text/vcard content-type, with a *value* of "VCARD". It is used in conjunction with the END property to delimit the current entity containing a related set of properties.	Text
END	1	Denotes the end of a syntactic entity within a text/vcard content-type.	Text
SOURCE	[0..*]	Identifies the source of directory information contained in the content type.	URI
KIND	[0..1]	Specifies the kind of object the vCard represents. Allowed *values*: **individual** (default) - a single person or entity. **group** - a group of persons or entities. **org** - an organization. **location** - a named geographical place. **x-name** - private or experimental values **iana-token** - for additional values that may be registered with IANA.	A single text value
XML (optional)	[0..*]	To include extended XML-encoded vCard data in a plain vCard.	A single text value

Table 12 – vCard general properties

Identification Properties

The properties presented in the following table (Table 13) are used to hold information regarding the personal identification and naming of the person/entity associated with the vCard.

Property	Multiplicity	Description	Value Type
FN	[1..*]	Formatted text corresponding to the name of the object the vCard represents.	A single text value
N	[0..1]	Components of the name of the object the vCard represents. The structured property value corresponds, in sequence, to the following text components: **Family Names** (a.k.a. surnames); **Given Names**; **Additional Names**; **Honorific Prefixes**; **Honorific Suffixes**. Each component can have multiple values separated by a comma character.	A single structured text value
NICKNAME	[0..*]	Nickname of the object the vCard represents. One or more text values separated by a comma character are allowed.	Text
PHOTO	[0..*]	Image or photograph information that annotates some aspect of the object the vCard represents.	A single URI
BDAY	[0..1]	The birth date of the object the vCard represents.	A single date-and-or-time value
ANNIVERSARY	[0..1]	Date of marriage, or equivalent, of the object the vCard represents.	A single date-and-or-time value
GENDER	[0..1]	Components of the sex and gender identity of the object the vCard represents. Two components, in sequence, are (optionally) defined in the structured text value: **Sex** (biological), with one of the following *values*: **M** - male **F** - female **O** - other **N** - none or not applicable **U** - unknown **Gender identity**, assuming a free form	A single structured text value

		text.	

Table 13 – vCard identification properties

Delivery Addressing Properties

The property presented in Table 14 refers to the delivery addressing information for the vCard object.

Property	Multiplicity	Description	Value Type
ADR	[0..*]	Specifies components of the delivery address for the vCard object. The structured text value type consists of a sequence of the following address components, separated by a semi-colon: **Post office box**; **Extended address** (e.g., apartment or suite number); **Street address**; **Locality** (e.g., city); **Region** (e.g., state or province); **Postal code**; **Country name**. When a component value is missing, the associated component separator must still be specified.	A single structured text value

Table 14 – vCard delivery addressing properties

Communication Properties

These properties (in Table 15) describe information about how to communicate with the object the vCard represents.

Property	Multiplicity	Description	Value Type
TEL	[0..*]	The telephone number for telephony communication with the object the vCard represents. The intended use for the telephone number is specified by its *type*: **text** - Support for text messages (SMS). **voice** (default) - A voice telephone number. **fax** - A facsimile telephone number. **cell** - A cellular or mobile telephone number. **video** - A video conferencing telephone number. **pager** - A paging device telephone number. **textphone** - A telecommunication device for people with hearing or speech difficulties.	A single free-form text value (default) *or* URI - “tel” scheme [14] (recommended)

EMAIL	[0..*]	The electronic mail address of the object the vCard represents.	A single text value
IMPP	[0..*]	The URI for instant messaging and presence protocol communications with the object the vCard represents.	A single URI
LANG	[0..*]	The language(s) that may be used for contacting the entity associated with the vCard.	A single language-tag value

Table 15 – vCard communication properties

Geographical Properties

The vCard properties presented in Table 16 are meant to specify global positioning information of the object the vCard represents.

Property	Multiplicity	Description	Value Type
TZ	[0..*]	Information related to the time zone of the object the vCard represents. Expected use of names from the public-domain Olson database [15] or, alternatively, from the IANA time zone database [16].	A single text value (default) *or* A single URI value
GEO	[0..*]	Information related to the global positioning of the object the vCard represents.	A single URI value (e.g., "geo" URI scheme [17])

Table 16 – vCard geographical properties

Organizational Properties

Organizational properties described in Table 17 describe the characteristics of the organization/organizational units of the object that the vCard represents.

Property	Multiplicity	Description	Value Type
TITLE	[0..*]	The position/job of the object the vCard represents, based on the X.520 Title attribute [18] .	A single text value
ROLE	[0..*]	The function/part played in a particular situation by the object the vCard represents, based on the X.520 Business Category explanatory attribute [18].	A single text value
LOGO	[0..*]	A graphic image of a logo associated with the object the vCard represents.	A single URI

ORG	[0..*]	The organizational name and units associated with the vCard, based on the X.520 Organization Name and Organization Unit attributes [18]. The property value is a structured type consisting of the following components: **Organization name**; **Level of organizational unit name** (0 or more).	A single structured text value
MEMBER	[0..*]	To include a member in the group this vCard represents by means of, e.g., a vCard object or an email distribution list. This property should only be present if the KIND property value is **group**.	A single URI
RELATED	[0..*]	Relationship between another entity and the entity represented by this vCard. The related entity may also be characterized according to its *type*, e.g.: **contact**, **acquaintance**, **friend**, **co-worker**, **colleague**, **co-resident**, **neighbor**, **spouse**, **agent**, **emergency**, among other values defined in [19].	A single URI *or* Text

Table 17 – vCard organizational properties

Explanatory Properties

These properties concern the need to specify additional explanatory information, e.g., notes or revisions specific to the vCard.

Property	Multiplicity	Description	Value Type
CATEGORIES	[0..*]	Specifies the application category information about the vCard, a.k.a. “tags”. One or more values may be specified, separated by a comma character.	Text
NOTE	[0..*]	Additional information or a comment that is associated with the vCard.	A single text value
PRODID	[0..1]	A unique identifier for the product that created	A single text

		the vCard object.	value
REV	[0..1]	Revision information about the current vCard.	A single timestamp value
SOUND	[0..*]	A digital sound content information that annotates some aspect of the vCard, e.g., to record the proper pronunciation of the name property value of the vCard.	A single URI
UID	[0..1]	A value that represents a globally unique identifier corresponding to the entity associated with the vCard.	A single URI (e.g., "uuid" URN namespace [20])
URL	[0..*]	A uniform resource locator (URL) associated with the object to which the vCard refers. E.g., for an individual, personal websites, blogs and social networking site identifiers.	A single URI
VERSION	1	The version of the vCard specification used to format this vCard.	A single text value

Table 18 – vCard explanatory properties

Security Properties

Security properties concern the security of communication pathways or access to the vCard.

Property	Multiplicity	Description	Value Type
KEY	[0..*]	A public key or authentication certificate associated with the object that the vCard represents.	A single URI *or* Text

Table 19 – vCard security properties

Calendar Properties

The properties in Table 20 provide information regarding the calendar of the object that the vCard represents.

Property	Multiplicity	Description	Value Type
FBURL	[0..*]	The URI for the busy time associated with the object that the vCard represents.	A single URI
CALADRURI	[0..*]	The calendar user address to which a scheduling request should be sent for the object represented by the vCard.	A single URI
CALURI	[0..*]	The URI for a calendar associated with the ob-	A single URI

		ject represented by the vCard. The property should contain a URI pointing to an iCalendar [2] object associated with a snapshot of the user's calendar store.	

Table 20 – vCard calendar properties

vCard Property Parameters

Parameters contain meta-information regarding a vCard property or the property value. In some cases, the property parameter can be multi-valued in which case its value elements are separated by a comma character.

Parameter	Description
LANGUAGE	Identifies data in multiple languages.
VALUE (optional)	Identifies the value type (data type) and format of the value, namely: **text**, **uri**, **date**, **time**, **date-time**, **date-and-or-time,** **timestamp**, **boolean**, **integer**, **float**, **utc-offset**, **language-tag**, **x-name**, **iana-token**.
PREF (optional)	Indicates that the corresponding instance of a property is preferred by the vCard author. Its value consists of an integer ranging from 1 to 100, being 1 the most preferred.
ALTID	Used to "tag" property instances as being alternative representations of the same logical property. For example, translations of a property in multiple languages generates multiple property instances having different LANGUAGE parameter that are tagged with the same ALTID value.
PID	Identifies a specific property among multiple instances. It plays a role analogous to the UID property on a per-property instead of per-vCard basis.
TYPE	Specifies class characteristics of the associated property. It may assume the following values:

	work **home** **type-param-tel** **type-param-related** **x-name** **iana-token**
MEDIATYPE (optional)	Used with properties whose value is a URI, in order provide a hint to the vCard consumer application about the media type of the resource identified by the URI (e.g., ftp).
CALSCALE	Defines the calendar system in which a date or date-time value is expressed. One of the following values are defined: **gregorian**, **x-name**, **iana-token**.
SORT-AS	Specifies the string to be used for national-language-specific sorting. When specified, then the given strings are used for sorting the vCard.
GEO	Indicates global positioning information that is specific to an address. Its value is the same as that of the GEO property.
TZ	Indicates time zone information that is specific to an address. Its value is the same as that of the TZ property.

Table 21 – vCard property parameters

5.3.2 Use case

Next, the vCard recommendation author's information is presented using the vCard format.

```
BEGIN:VCARD
VERSION:4.0
FN:Simon Perreault
N:Perreault;Simon;;;ing. jr,M.Sc.
BDAY:--0203
ANNIVERSARY:20090808T1430-0500
GENDER:M
LANG;PREF=1:fr
LANG;PREF=2:en
ORG;TYPE=work:Viagenie
ADR;TYPE=work::;Suite D2-630;2875 Laurier;Quebec;QC;G1V 2M2;Canada
TEL;VALUE=uri;TYPE="work,voice";PREF=1:tel:+1-418-656-9254;ext=102
TEL;VALUE=uri;TYPE="work,cell,voice,video,text":tel:+1-418-262-6501
EMAIL;TYPE=work:simon.perreault@viagenie.ca
GEO;TYPE=work:geo:46.772673,-71.282945
KEY;TYPE=work;VALUE=uri:http://www.viagenie.ca/simon.perreault/simon.asc
TZ:-0500
URL;TYPE=home:http://nomis80.org
```

```
END:VCARD
```

Figure 7 – The author’s (Simon Perreault) plain vCard example

By observing the example, we are able to retrieve information about the author, by inspecting each property, regarding his name, honorific titles, birthday, marriage anniversary, gender, preferred languages (French first, English second), organization name, address and geographic location, as well as work telephone and email contacts, and information security key. Additionally, the author’s homepage is also made available.

The following examples illustrate the author’s vCard example being mapped into an XML (Figure 8) and JSON (Figure 9) formats.

```
<?xml version="1.0" encoding="UTF-8"?>
<vcards xmlns="urn:ietf:params:xml:ns:vcard-4.0">
 <vcard>
  <fn><text>Simon Perreault</text></fn>
  <n>
    <surname>Perreault</surname>
    <given>Simon</given>
    <additional/>
    <prefix/>
    <suffix>ing. jr</suffix>
    <suffix>M.Sc.</suffix>
  </n>
  <bday><date>--0203</date></bday>
  <anniversary>
    <date-time>20090808T1430-0500</date-time>
  </anniversary>
  <gender><sex>M</sex></gender>
  <lang>
    <parameters><pref><integer>1</integer></pref></parameters>
    <language-tag>fr</language-tag>
  </lang>
  <lang>
    <parameters><pref><integer>2</integer></pref></parameters>
    <language-tag>en</language-tag>
  </lang>
  <org>
    <parameters><type><text>work</text></type></parameters>
    <text>Viagenie</text>
  </org>
  <adr>
    <parameters>
      <type><text>work</text></type>
    </parameters>
    <pobox/>
    <ext/>
    <street>2875 boul. Laurier, suite D2-630</street>
    <locality>Quebec</locality>
    <region>QC</region>
    <code>G1V 2M2</code>
    <country>Canada</country>
  </adr>
  <tel>
    <parameters>
```

```
        <type>
          <text>work</text>
          <text>voice</text>
        </type>
      </parameters>
      <uri>tel:+1-418-656-9254;ext=102</uri>
    </tel>
    <tel>
      <parameters>
        <type>
          <text>work</text>
          <text>text</text>
          <text>voice</text>
          <text>cell</text>
          <text>video</text>
        </type>
      </parameters>
      <uri>tel:+1-418-262-6501</uri>
    </tel>
    <email>
      <parameters><type><text>work</text></type></parameters>
      <text>simon.perreault@viagenie.ca</text>
    </email>
    <geo>
      <parameters><type><text>work</text></type></parameters>
      <uri>geo:46.766336,-71.28955</uri>
    </geo>
    <key>
      <parameters><type><text>work</text></type></parameters>
      <uri>http://www.viagenie.ca/simon.perreault/simon.asc</uri>
    </key>
    <tz><text>America/Montreal</text></tz>
    <url>
      <parameters><type><text>home</text></type></parameters>
      <uri>http://nomis80.org</uri>
    </url>
  </vcard>
</vcards>
```

Figure 8 – The author's (Simon Perreault) XML vCard example

From, it is possible to observe how vCard properties, parameters and value types are mapped into XML elements. Nevertheless, it is visible that the effort in maintaining the underlying data structure is accomplished.

```
{
    "version":"4.0",
    "fn":  "Simon Perreault",
    "n": {
        "surname": "Simon",
        "given": "Perreault",
        "suffix": [ "ing. jr", "M.Sc."]
    },
    "bday": { "date": "--0203" },
    "anniversary": { "date-time": "20090808T1430-0500" },
    "gender": { "sex" : "M" },
  "lang": [ {
    "pref": 1,
    "language-tag": "fr",
  },
  {
    "pref": 2,
    "language-tag": "en",
  },
  ],
  "org": {
    "type": "work",
    "text": "Viagenie"
  },
  "adr": {
    "type": "work",
    "street": "2875 boul. Laurier, suite D2-630",
    "locality": "Quebec",
    "region": "QC",
    "code": "G1V 2M2",
    "country": "CA"
  },
  "tel": [ {
    "type":  ["work", "voice"],
    "uri": "tel:+1-418-656-9254;ext=102"
  },
  {
    "type": ["work", "text", "voice", "cell", "video"],
    "uri": "tel:+1-418-262-6501"
  }
  ],
  "email": {
  "type": "work",
  "text": "simon.perreault@viagenie.ca"
  },
  "geo": {
    "type": "work",
    "uri": "geo:46.766336,-71.28955"
  },
  "key": {
    "type": "work",
    "uri": "http://www.viagenie.ca/simon.perreault/simon.asc"
  },
  "tz": "America/Montreal",
  "url": {
    "type": "home",
    "uri": "http://nomis80.org"
  }
}
```

Figure 9 – The author’s (Simon Perreault) JSON vCard example

Similarly to the XML example, the underlying data structure is identical to the plain vCard when using the JSON representation.

5.4 iCalendar

iCalendar (text/calendar MIME media type) is an IETF Internet Standards Track document [2] which aims at providing the definition of a common format for openly exchanging electronic calendaring and scheduling information across the Internet. It may be used in the W3C POI recommendation to express a POI's context in time, either in respect to its existence (start/end), opening/closing (e.g., museum) or recurring (e.g., fairs) dates and times, or yet event schedules (e.g., a concert).

This MIME media type provides a standard content type for capturing calendar event, to-do and journal entry information. It also can be used to convey free/busy time information. In this sense, the recommendation defines the format for specifying iCalendar object properties, components and parameters that enable the conveying of requests to schedule an event, reply to an event request, send a cancellation notice for an event, modify or replace the definition of an event, provide a counter proposal for an original event request, delegate an event request to another individual, request free or busy time, reply to a free or busy time request, or provide similar scheduling messages for a to-do or journal entry calendar component. The calendar properties, components and parameters are presented next.

Lastly, an example using this specification is presented.

5.4.1 iCalendar Object Specification

Calendar Properties

Calendar properties are attributes that apply to the iCalendar object, as a whole. They are required to be specified after the "BEGIN:VCALENDAR" property and prior to any calendar component.

Property	Description	Value Type
CALSCALE	Calendar scale used for the calendar information specified in the iCalendar object. Its value can either be: **GREGORIAN** (default), **iana-token**.	Text
METHOD	iCalendar object method associated with the calendar object. The method value is of type **iana-token**.	Text
PRODID	Identifier for the product that created the iCalendar object.	Text
VERSION	Identifier corresponding to the highest version number or the minimum and maximum range of the iCalendar specification that is required in order to interpret the iCalendar object.	Text

Table 22 – iCalendar properties

Calendar Property Parameters

Property parameters, shown in Table 23, contain meta-information about a given property they are associated with, or the property value.

Parameter	Description
ALTREP	An alternate text representation for the property value.
CN	The common name to be associated with the calendar user specified by the property.
CUTYPE	The type of calendar user specified by the property. It may be one of the following: **INDIVIDUAL**, **GROUP**, **RESOURCE**, **ROOM**, **UNKOWN**, **x-name**, **iana-token**.
DELEGATED-FROM	The calendar users that have delegated their participation to the calendar user specified by the property. The value is required to be a MAILTO URI.
DELEGATED-TO	The calendar users to whom the calendar user specified by the property has delegated participation. The value is required to be a MAILTO URI.
DIR	Reference to a directory entry associated with the calendar user specified by the property. Its value is required to be an URI specified in a quoted-string.
ENCODING	An alternate inline encoding for the property value. Inline encoding values are: **8-bit**, **BASE64**, **iana-token**, **x-name**.
FMTTYPE	The content type of a referenced object. Its value is required to be the text value for either: **iana-token**, **x-name**.
FBTIME	Free or busy time type. Possible values for this property parameter are: **FREE -** The time interval is free for scheduling. **BUSY -** the time interval is busy because one or more events have been scheduled for that interval. **BUSY-UNAVAILABLE -** the time interval is busy and that the interval cannot be scheduled. **BUSY-TENTATIVE -** the time interval is busy because one or more events have been tentatively scheduled for that interval. **iana-token**, **x-name**.

LANGUAGE	The language for text values in a property or property parameter, as defined in [6].
MEMBER	The group or list membership of the calendar user specified by the property.
PARTSTAT	The participation status for the calendar user specified by the property. The following values are allowed for a "VEVENT" component: **NEEDS-ACTION**, **ACCEPTED**, **DECLINED**, **TENTATIVE**, **DELEGATED**, **iana-token**, **x-name**. Values were also defined for "VTODO" and "VCALENDAR" components.
RANGE	The effective range of recurrence instances from the instance specified by the recurrence identifier specified by the property. The value can be: **THISANDPRIOR** - All instances prior to the recurrence identifier. **THISANDFUTURE** - The instance specified by the recurrence identifier and all subsequent recurrence instances.
RELATED	The relationship of the alarm trigger with respect to the start or end of the calendar component. The specified value cane be: **START** – Trigger off of start, **END** – Trigger off of end.
RELTYPE	The type of hierarchical relationship associated with the calendar component specified by the property. Its value may be either: **PARENT**, **CHILD**, **SIBLING**, **iana-token**, **x-name**.
ROLE	The participation role for the calendar user specified by the property. One of the following roles may be specified: **CHAIR**, **REQ-PARTICIPANT** (default), **OPT-PARTICIPANT**, **NON-PARTICIPANT**, **x-name**, **iana-token**.
RSVP	Specifies whether there is an expectation of a favor of a reply from the calendar user specified by the property value. One of the following values is allowed: **TRUE**, **FALSE** (default).
SENT-BY	The calendar user that is acting on behalf of the calendar user specified by

	the property. The value is required to be a MAILTO URI.
TZID	The identifier for the time zone definition for a time component in the property value. Is required to be specified on the "DTSTART", "DTEND", "DUE", "EXDATE" and "RDATE" properties when either a DATE-TIME or TIME value type is specified.
VALUE	Explicitly specifies the data type format for a property value. May be one of the following: **BINARY**, **BOOLEAN**, **CAL-ADDRESS**, **DATE**, **DATE-TIME**, **DURATION**, **FLOAT**, **INTEGER**, **PERIOD**, **RECUR**, **TEXT**, **TIME**, **URI**, **UTC-OFFSET**, **x-name**, **iana-token**.

Table 23 – iCalendar property parameters

Calendar Components

Calendar components, shown in Table 24, are collections of properties that express a particular calendar semantic, since they can specify an event, a to-do, a journal entry, time zone information, or free/busy time information, or an alarm.

Component	Name Description
VEVENT	Provide a grouping of component properties that describe an event.
VTODO	Provides a grouping of calendar properties that describe a to-do.
VJOURNAL	Provides a grouping of component properties that describe a journal entry.
VFREEBUSY	Provides a grouping of component properties that describe either a request for or a response to a request for free/busy time or yet a published set of busy time.
VTIMEZONE	Provides a grouping of component properties that defines time zone.
VALARM	Provides a grouping of component properties that defines an alarm.

Table 24 – iCalendar components

Calendar Component Properties

The component properties presented in Table 25 can appear within calendar components, as specified by each component property definition, and are grouped into **Descriptive**, **Date and Time**, **Time Zone**, **Relationship**, **Recurrence**, **Alarm**, **Change Management** and **Miscellaneous** component properties.

Descriptive Component Properties

The following properties specify descriptive information about calendar components.

Property	Description	Value Type
ATTACH	Provides the capability to associate a document object with a calendar component.	URI
CATEGORIES	Defines the categories for a calendar component.	TEXT
CLASS	Defines the access classification for a calendar component. The following values are defined: **PUBLIC** (default), **PRIVATE**, **CONFIDENTIAL**, **iana-token**, **x-name**	TEXT
COMMENT	Non-processing information intended to provide a comment to the calendar user.	TEXT
DESCRIPTION	Provides a more complete description of the calendar component, in contrast to the SUMMARY property.	TEXT
GEO	Specifies information related to the global position for the activity specified by a calendar component. The value consists in two float values - latitude (LAT) and longitude (LON) - separated by a semi-colon.	FLOAT
LOCATION	Defines the intended venue for the activity defined by a calendar component.	TEXT
PERCENT-COMPLETE	Used by an assignee or delegatee of a to-do to convey the percent completion of a to-do to the Organizer. The property value is a positive integer between 0 (to-do task has not yet begun) and 100 (to-do task completed).	INTEGER
PRIORITY	Defines the relative priority for a calendar component. The priority is specified as an integer in the range 0 (undefined priority) to 9 (lowest priority). 1 is the highest priority.	INTEGER
RESOURCES	Defines the equipment or resources anticipated for an ac-	TEXT

	tivity specified by a calendar entity.	
STATUS	Defines the overall status or confirmation for the calendar component. The following values for a VEVENT component are defined: **TENTATIVE**, **CONFIRMED**, **CANCELLED**. Values for VTODO and VJOURNAL are also defined.	TEXT
SUMMARY	Defines a short summary or subject for the calendar component.	Text

Table 25 – iCalendar descriptive component properties

Date and Time Component Properties

The following properties specify information regarding the date and time in calendar components.

Property	Description	Value Type
COMPLETED	The date and time that a to-do was actually completed.	DATE-TIME
DTEND	The date and time that a calendar component ends.	DATE-TIME *or* DATE
DUE	The date and time that a to-do is expected to be completed.	DATE-TIME *or* DATE
DTSTART	Specifies when the calendar component begins.	DATE-TIME *or* DATE
DURATION	Specifies a positive duration of time.	DURATION
FREEBUSY	Defines one or more free or busy time intervals.	PERIOD
TRANSP	Defines whether an event appears to consume time on a calendar, allowing it to be detected (transparent) or not by free-busy time searches. The following values are defined: **OPAQUE** (default) - Blocks or opaque on busy time searches. **TRANSPARENT** - Transparent on busy time searches.	TEXT

Table 26 – iCalendar date and time component properties

Time Zone Component Properties

These properties specify time zone information in calendar components.

Property	Description	Value Type
TZID	Specifies the text value that uniquely identifies the VTIMEZONE calendar component.	TEXT
TZNAME	Specifies the customary designation for a time zone description.	TEXT
TZOFFSET FROM	Specifies the offset which is in use prior to this time zone observance.	UTC-OFFSET
TZOFFSETTO	Specifies the offset which is in use in this time zone observance.	UTC-OFFSET
TZURL	Reference to a network location that can be used to retrieve an up-to-date version of a VTIMEZONE component.	URI

Table 27 – iCalendar time zone component properties

Relationship Component Properties

The following properties concern relationship information in calendar components.

Property	Description	Value Type
ATTENDEE	An attendee within a calendar component.	CAL-ADDRESS
CONTACT	Represents contact information or alternately a reference to contact information associated with the calendar component, e.g., a URI pointing to an alternate form, such as a vCard.	TEXT *or* URI
ORGANIZER	Defines the organizer for a calendar component.	CAL-ADDRESS
RECURREN-CE-ID	Used in conjunction with the UID and SEQUENCE properties to identify a specific instance of a recurring VEVENT, VTODO or VJOURNAL calendar component.	DATE-TIME *or* DATE
RELATED-TO	Represents a relationship or reference between one calendar component and another.	TEXT
URL	Defines a Uniform Resource Locator (URL) associated with the iCalendar object.	URI
UID	Defines the persistent, globally unique identifier for the calendar component.	TEXT

Table 28 – iCalendar relationship component properties

Recurrence Component Properties

The following properties specify recurrence information in calendar components.

Property	Description	Value Type
EXDATE	Defines the list of date/time exceptions for a recurring calendar component.	DATE-TIME *or* DATE
EXRULE	Defines a rule or repeating pattern for an exception to a recurrence set.	RECUR
RDATE	Defines the list of date/times for a recurrence set.	DATE-TIME *or* DATE *or* PERIOD
RRULE	Defines a rule or repeating pattern for recurring events, to-dos, or time zone definitions.	RECUR

Table 29 – iCalendar recurrence component properties

Alarm Component Properties

The following properties specify alarm information in calendar components.

Property	Detail	Value Type
ACTION	Defines the action to be invoked when an alarm is triggered.	TEXT
REPEAT	Defines the number of times the alarm should be repeated, after the initial trigger. The default number is 0.	INTEGER
TRIGGER	Specifies when an alarm will trigger.	DURATION (default) *or* DATE-TIME

Table 30 – iCalendar alarm component properties

Change Management Component Properties

These properties specify change management information in calendar components.

Property	Description	Value Type
CREATED	Specifies the date and time that the calendar information was created by the its user agent in the calendar store.	DATE-TIME
DTSTAMP	Indicates the date/time that the instance of the iCalendar object was created.	DATE-TIME
LAST-MODIFIED	Specifies the date and time that the calendar component information was last revised in the calendar store.	DATE-TIME
SEQUENCE	Defines the revision sequence number of the calendar	INTEGER

	component within a sequence of revisions. When a calendar component is created, its sequence number is 0 and it is monotonically incremented each time the Organizer makes a significant revision to the calendar component.	

Table 31 – iCalendar change management component properties

Miscellaneous Component Properties

Miscellaneous properties specify information about a number of miscellaneous features of calendar components.

Property	Description	Value Type
Any property name with a "X-" prefix	Provides a framework for defining non-standard properties.	TEXT
REQUEST-STATUS	Defines the status code returned for a scheduling request.	TEXT

Table 32 – iCalendar miscellaneous component properties

5.4.2 Use Case

An example for the use of iCalendar is the scheduling of an event taking place at a given POI. In the following example, information regarding the “Michael Jackson: The Immortal World Tour by Cirque du Soleil” event is being described.

```
BEGIN:VCALENDAR
CALSCALE:GREGORIAN
PRODID:-//Example Inc.//Example Calendar//EN
VERSION:2.0
BEGIN:VEVENT
DTSTART:20130411T213000Z
DTEND:20130414
ORGANIZER;CN=Everything is New:MAILTO:geral@everythingisnew.pt
STATUS:CONFIRMED
CATEGORIES:Entertainment,Music,Dance
SUMMARY:Michael Jackson: The Immortal World Tour by Cirque du Soleil, in Portugal.
UID:4088E990AD89CB3DBB484909
END:VEVENT
END:VCALENDAR
```

Figure 10 – Event information in plain iCalendar format example

Information such as the start and end dates, the event organizer, status, categories which best describe this event, and a summary are provided. As mentioned before, this format may be used by the W3C POI recommendation to specify time and date information, together with the remaining information such as the event’s venue location, which itself may be a POI.

The following examples represent the mapping of the previous example into the XML (Figure

11) and JSON (Figure 12) formats, where, in the former one, iCalendar properties, components, parameters and value types are mapped into XML elements.

```
<?xml version="1.0" encoding="utf-8"?>
<icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0">
<vcalendar>
 <properties>
  <calscale>
    <text>GREGORIAN</text>
  </calscale>
  <prodid>
   <text>-//Example Inc.//Example Calendar//EN</text>
  </prodid>
  <version>
    <text>2.0</text>
  </version>
 </properties>
 <components>
  <vevent>
   <properties>
    <dtstart>
      <date-time>2013-04-11T20:00:00Z</date-time>
    </dtstart>
    <dtend>
      <date>2013-04-14</date>
    </dtend>
    <organizer>
      <cal-address>mailto:geral@everythingisnew.pt</cal-address>
      <parameters>
        <cn><text>Everything is New</text></cn>
      </parameters>
    </organizer>
    <category>
      <text>Entertainment,Music,Dance</text>
    </category>
    <status>
      <text>CONFIRMED</text>
    </status>
    <summary>
     <text>Michael Jackson: The Immortal World Tour by Cirque du Soleil, in Portugal</text>
    </summary>
    <uid>
     <text>4088E990AD89CB3DBB484909</text>
    </uid>
   </properties>
  </vevent>
 </components>
</vcalendar>
</icalendar>
```

Figure 11 – Event information in xCal format example

```
["vcalendar",
 [
   ["calscale", {}, "text", "GREGORIAN"],
   ["prodid", {}, "text", "-//Example Inc.//Example Calendar//EN"],
   ["version", {}, "text", "2.0"]
 ],
 [
   ["vevent",
     [
       ["dtstart", {}, "date-time", "2013-04-11T20\:00\:00Z"],
       ["dtend", {}, "date", "2013-04-14"],
       [“organizer”,
       {
          "cn": "Everything is New",
        },
        "cal-address",
        "mailto\:geral@everythingisnew.pt",
       ]
       [“category”, {}, “text”, “Entertainment”, “Music”, “Dance”],
       [“status”, {}, “text”, “CONFIRMED”],
       ["summary", {}, "text", "Michael Jackson\: The Immortal World Tour by Cirque du Soleil, in Portugal"],
       ["uid", {}, "text", "4088E990AD89CB3DBB484909"],
     ],
     []
   ]
 ]
]
```

Figure 12 – Event information in jCal format example

5.5 Standards Summary

Two main standards have been presented within this section – IPTC’s EventsML-G2 and W3C POI. As seen previously, the EventsML-G2 is a well documented standard for event representation and is already in use. As for W3C POI, its main focus is on the POI representation. However, according to the working group’s wiki[38], the W3C POI WG has closed as of September 2012 and no changes to the model presented are foreseen. Nevertheless, a *“Places” Community Group*[39] was created with the aim of continuing this work, focusing on the POI representation in other formats like RDF and JSON. Additionally, the *Open Geospatial Consortium*[40] is in the process of creating a standards working group to standardize the POI conceptual data model and XML encoding.

Besides including events, persons and organizations, EventsML-G2 provides a way to represent POI’s, mostly using its *POIDetails* property, which determines the place where the event is occurring. W3C POI provides, essentially, a way to represent POI or group of POI’s, e.g., for tourism routes. Since events are likely to present very similar properties to the POI entity, we chose to extend W3C POI in order to support the event message format. W3C POI was also chosen due to its flexibility at various granularity levels of the data model, extensibility and ability to easily map into other message formats such as JSON. The EventsML-G2 standard, in spite of being a very complete standard, not only is more complex, but it covers features which are already well established in the widely used iCalendar standard which can be used in the W3C POI to address

[38] http://www.w3.org/2010/POI/wiki/Main_Page
[39] http://www.w3.org/community/places/
[40] http://www.opengeospatial.org/

its *time* property and, eventually, the event extension using iCalendar's Event component. Ultimately, persons and organizations can also be represented using vCard, another widely used standard.

5.6 POI Categorization

POIs may be categorized in many different ways, e.g., using ratings or simply keywords, being the latter the most frequent approach.

Currently, the open-data Web portal provided by the city of Lisbon[41] groups the data collection mainly into accommodation, restaurants, culture, heritage, tourism, maps, environment, leisure, shopping, urbanism, services, education, Government and Parliamentary affairs, health, sports, business or commerce, public safety, emergency and transportation categories.

Other categories can be added such as petrol stations, WiFi hotspots and recreation[42], or yet categories such as culture, tourism, arts, entertainment and heritage can be further characterized by including, e.g., architecture, dance, design, fashion, film, landmarks, music, photography, theatre, festivals and events, museums and galleries, which can be found at the Arts Holland website[43]. Other categories are also provided, in terms of accommodation for visitors, namely, apartments, BB (Bed & Breakfast), bungalow parks, camping, congress accommodations, groups accommodations, holiday rentals hotels, hostels, and yet others such as tourist offices, restaurants, golf clubs and attractions.

Several categories have been defined[44] and many, many more exist, however, to the best of our knowledge, there is no universal nomenclature or standard for category terms. In this sense, a base version will be developed by the lead pilot, which may be adopted and/or adjusted by each city pilot, later on.

[41] http://www.lisboaparticipa.pt/pages/apresentacaoDados.php
[42] http://poi.gps-data-team.com/portugal/
[43] http://www.artsholland.com/
[44] http://www.poi-factory.com/poifiles

6 CitySDK Tourism API Specification

Providing a uniform interface to consult information regarding POIs or events is necessary so to facilitate the development of tourism-related applications. We have chosen to follow a REST architectural design to represent resources and their manipulation.

In this section, we will start by describing how the W3C POI recommendation is used to represent data in the replies. Then, we present our approach into developing a RESTful API through the use of hypermedia, as well as the resources that need to be represented.

6.1 Messages Formats

Discussing message formats is of utmost importance, since it is necessary for developers to understand - and consequently their applications - what each field represents and what values they can take. We derive each message from the UML model presented in Figure 13 - taken from the section of W3C POI. All of the messages will be encoded in UTF-8 format, since it is the most preferred encoding format of JSON.

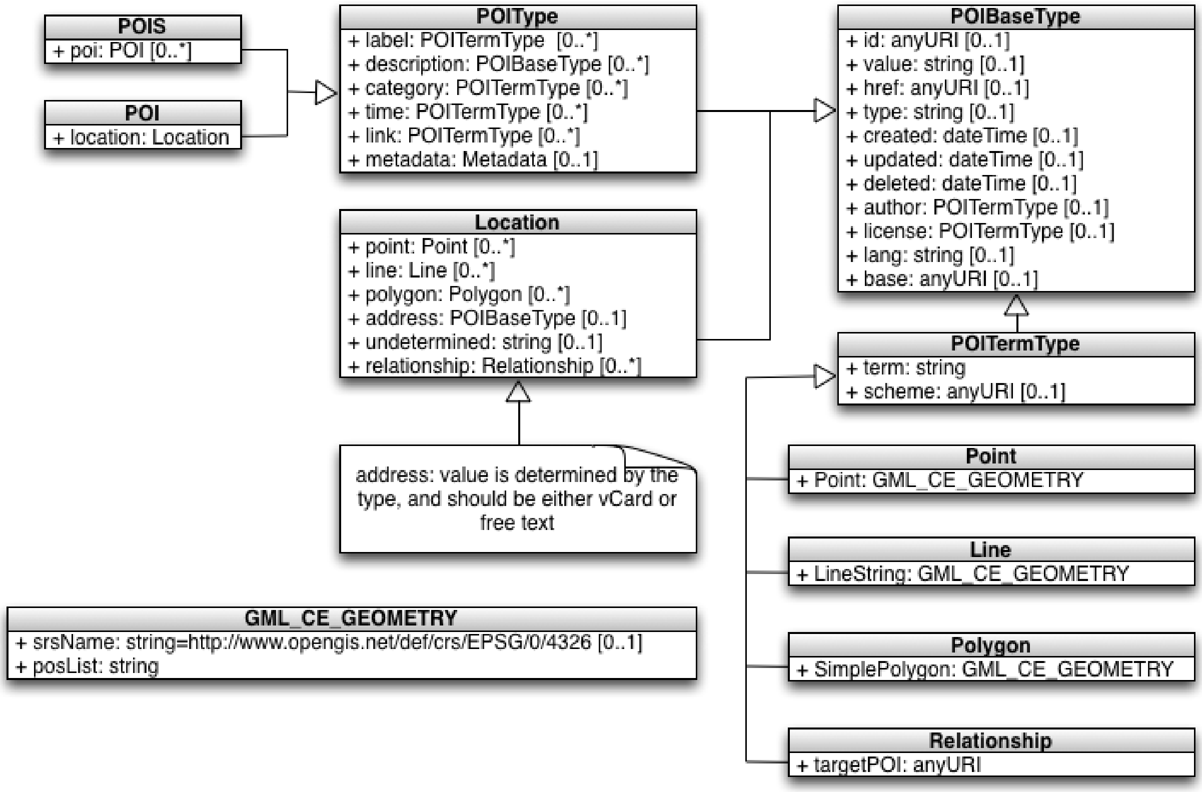

POIS
+ poi: POI [0..*]
POI
+ location: Location
POIType
+ label: POITermType [0..*]
+ description: POIBaseType [0..*]
+ category: POITermType [0..*]
+ time: POITermType [0..*]
+ link: POITermType [0..*]
+ metadata: Metadata [0..1]
POIBaseType
+ id: anyURI [0..1]
+ value: string [0..1]
+ href: anyURI [0..1]
+ type: string [0..1]
+ created: dateTime [0..1]
+ updated: dateTime [0..1]
+ deleted: dateTime [0..1]
+ author: POITermType [0..1]
+ license: POITermType [0..1]
+ lang: string [0..1]
+ base: anyURI [0..1]
Location
+ point: Point [0..*]
+ line: Line [0..*]
+ polygon: Polygon [0..*]
+ address: POIBaseType [0..1]
+ undetermined: string [0..1]
+ relationship: Relationship [0..*]
POITermType
+ term: string
+ scheme: anyURI [0..1]
address: value is determined by the type, and should be either vCard or free text
Point
+ Point: GML_CE_GEOMETRY
Line
+ LineString: GML_CE_GEOMETRY
GML_CE_GEOMETRY
+ srsName: string=http://www.opengis.net/def/crs/EPSG/0/4326 [0..1]
+ posList: string
Polygon
+ SimplePolygon: GML_CE_GEOMETRY
Relationship
+ targetPOI: anyURI

Figure 13 – W3C POI UML model

From the UML model we intend to represent three important services: **Points of Interest**, **Events** and **Routes**.

6.1.1 Point of Interest (POI) Format

Points of Interest can be represented through the following fields:

1. **IDs** and **Alternative IDs** - the unique identifier of the POI and other ways to identify the same POI, through the use of RFID, NFC, QR codes or Barcodes;
2. **Parent POI IDs** - the relationship of a given POI to another, e.g., a restaurant inside a given building;
3. **Name** - name of the POI;
4. **Description** - the description of the POI, which can be multi-lingual and yet include information regarding accessibility;
5. **Category** and **Tags** - classifications and keywords that enable characterization and search;
6. **Contacts** - any address, phone numbers, websites, social networks or any other form of contact that the POI might have;
7. **Opening hours**, **Prices** and **Location**.

These fields are the basic information in order to provide the users enough data to identify or search for a given POI. Additionally, multimedia data and related applications can be provided. Also, optional fields - like queue's waiting time and POI occupation - can be useful so to extend the information of the POI.

Taking into account the aforementioned topics, we have derived the presented UML model into the one shown below in Figure 14.

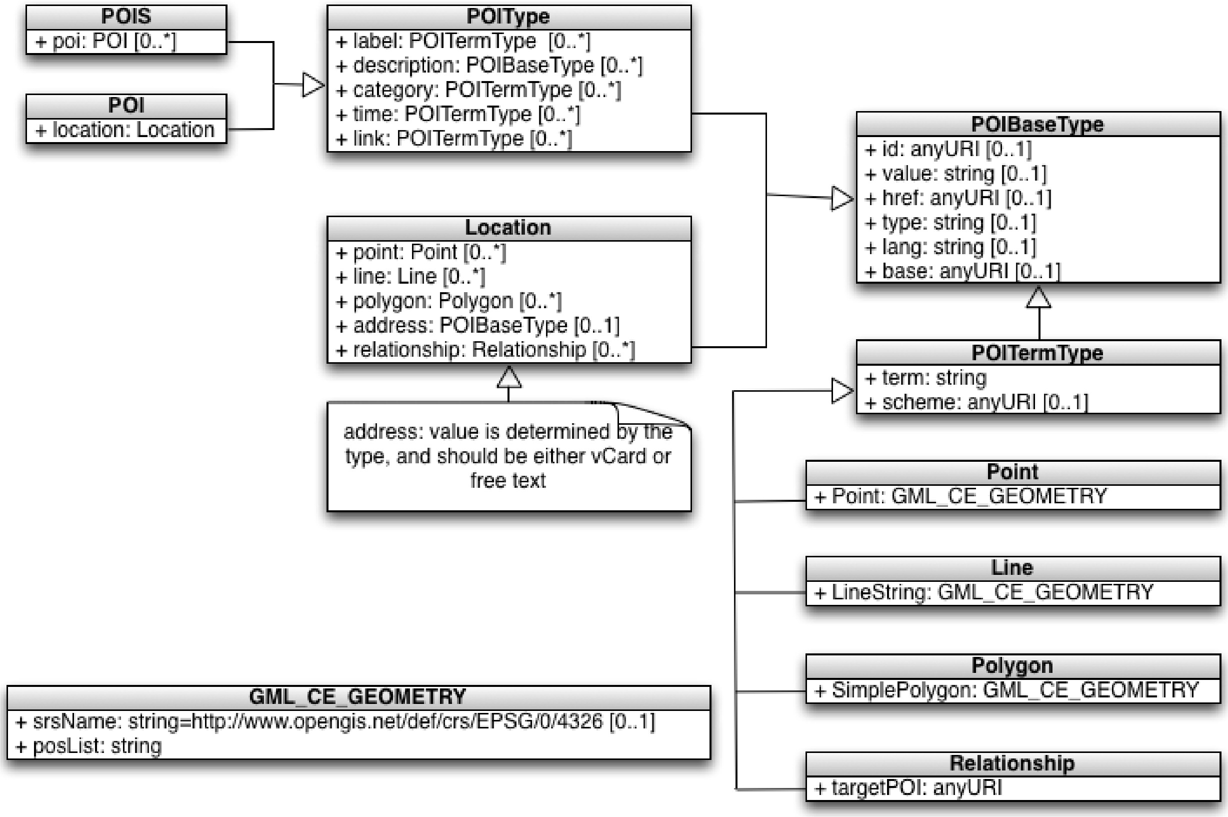

POIS
+ poi: POI [0..*]
POI
+ location: Location
POIType
+ label: POITermType [0..*]
+ description: POIBaseType [0..*]
+ category: POITermType [0..*]
+ time: POITermType [0..*]
+ link: POITermType [0..*]
POIBaseType
+ id: anyURI [0..1]
+ value: string [0..1]
+ href: anyURI [0..1]
+ type: string [0..1]
+ lang: string [0..1]
+ base: anyURI [0..1]
Location
+ point: Point [0..*]
+ line: Line [0..*]
+ polygon: Polygon [0..*]
+ address: POIBaseType [0..1]
+ relationship: Relationship [0..*]
address: value is determined by the type, and should be either vCard or free text
POITermType
+ term: string
+ scheme: anyURI [0..1]
Point
+ Point: GML_CE_GEOMETRY
Line
+ LineString: GML_CE_GEOMETRY
Polygon
+ SimplePolygon: GML_CE_GEOMETRY
GML_CE_GEOMETRY
+ srsName: string=http://www.opengis.net/def/crs/EPSG/0/4326 [0..1]
+ posList: string
Relationship
+ targetPOI: anyURI

Figure 14 – POI UML representation

Table 33 shows how each of the presented fields is mapped into a given attribute in the UML. Also, Figure 15 shows an example for a POI.

POI Field	Description	UML Attribute
POI ID	Unique identifier of the POI	POIBaseType::id
Alternative IDs	Alternative ways to represent the POI: RFID, NFC, QR codes and Barcode	POIType::link
Parent POI	Relationship from a POI to another	POIType::link Location::relationship
Name	Name of the POI	POIType::label
Description	Description of the POI.	POIType::description
Category	Classification of the POI	POIType::category
Tags	Keywords to characterize the POI	POIType::category
Contacts	Contact information of the POI: address, phone numbers, networks, etc.	Location::address
Opening Hours	Time at which a given POI is open or closed	POIType::time
Prices	Price to enter a POI	POIType::description
Queue’s waiting time	Time - in seconds - for a given user to enter the POI (only available if necessary)	POIType::description
POI occupation	Occupation percentage - integer from 0 to 100 - of the POI (only available if necessary)	POIType::description
Accessibility	Accessibility information regarding the POI. This information is mainly used by disabled people, parents with pushchairs, elderly and other who wish to see if the POI accommodates their needs.	POIType::description
Location	Where the POI is located	Location::point Location::line Location::polygon
Multimedia Data	Images, Videos or Links	POIType::link
Related Applications	Applications related to the POI	POIType::link
Specialized Service	The POI may include a link for an alternative	POIType::link

	server, which may provide a more detailed description of the POI and enable searching for POIs and events within that POI.	

Table 33 – POI description and representation

```
{
 “poi”:
  {
   “base”: “http:\/\/example.pt\/poi\/”,
   “id”:”1234”,
   “lang”:”EN”,
   “label”: { “term”:”primary”, “value”:”Museum Calouste Gulbenkian” },
   “description”:
   [
     { “lang”:”PT”, “value”:”Descrição” },
     { “lang”:”EN”, “value”:”Description” },
     { “type”:”X-citysdk/accessibility-textual”, “lang”:”EN”, “value”:” The doors connected to the entrance stand out clearly.
Outside the door there is sufficient room for moving e.g. with a wheelchair. The door is heavy or otherwise hard to
open.”}
     { “type”:”X-citysdk/accessibility-properties”,”id”:”ENTRANCE.01.DOOR.stands_out_clearly”,“value”:”true” }
     { “type”:”X-citysdk/accessibility-
properties”,”id”:”ENTRANCE.01.DOOR.sufficiently_room_for_wheelchair_in_front”,“value”:”true” }
     { “type”:”X-citysdk/accessibility-
properties”,”id”:”ENTRANCE.01.DOOR.opening_method”,“value”:"is_heavy_or_otherwise_hard_to_open”}
     { “type”:”X-citysdk/price”, “lang”:”EN”, “value”:”0€ per person” },
     { “type”:”X-citysdk/waiting-time”, “value”:”7200” },
     { “type”:”X-citysdk/occupation”, “value”:”30” }
   ],
   “category”:
   [
     { “term”:”category”, “value”:”Museum” },
     { “term”:”category”, “value”:”Park” },
     { “term”:”tag”, “value”:”culture”},
     { “term”:”tag”, “value”:”nature”}
   ],
   “location”:
   {
       “point”:
       {
         “term”:”entrance”,
         “Point”:
         {
            “srsName”:”http:\/\/www.opengis.net\/def\/crs\/EPSG\/0\/4326”,
            “postList”:”38.738369 -9.154962”
         },
      },
     “address”:
     {
       “type”:”text\/vcard”,
       “value”:”BEGIN\:VCARD
                VERSION\:4.0
```

```
                    N\:Gump;Forrest;;;
                    FN\: Forrest Gump
                    ORG\:Bubba Gump Shrimp Co.
                    TITLE\:Shrimp Man
                    PHOTO\:http:\/\/www.example.com\/dir_photos\/my_photo.gif
                    TEL;TYPE=\"work,voice\";VALUE=uri\:tel\:+1-111-555-1212
                    TEL;TYPE=\"home,voice\";VALUE=uri\:tel:+1-404-555-1212
                     ADR;TYPE=work;LABEL=\"42 Plantation St.\\nBaytown, LA 30314\\nUnited States of America"
                               \:;;42 Plantation St.;Baytown;LA;30314;United States of America
                    EMAIL\:forrestgump@example.com
                    REV\:20080424T195243Z
                    END\:VCARD”
          },
          “relationship”:
          {
             “term”:”within”,
             “targetPOI”:”0”
          }
   },
   “time”:
   {
      “term”:”open”,
      “type”:”text\/calendar”,
      “value”:”BEGIN\:VCALENDAR
                  VERSION\:2.0
                  PRODID\:-\/\/hacksw\/handcal\/\/NONSGML v1.0\/\/EN
                  BEGIN\:VEVENT
                  UID\:uid1@example.com
                  DTSTAMP\:19970714T170000Z
                  ORGANIZER;CN=John Doe\:MAILTO\:john.doe@example.com
                  DTSTART\:19970714T170000Z
                  DTEND\:19970715T035959Z
                  SUMMARY\:Bastille Day Party
                  END\:VEVENT
                  END\:VCALENDAR”
   },
   “link”:
   [
      { “term”:”related”,“type”:”application/vnd.android.package-archive”,
        “href”: “http:\/\/link-to-related-apps.com\/apps\/1234” },
      { “term”:”related”, “type”:”image\/jpeg”, “href”:”http:\/\/link-to-multimedia.com\/photo.jpeg” },
      { “term”:”related”, “type”:”image\/jpeg”, “rel”:”icon”, “href”:”http:\/\/link-to-multimedia.com\/thumbnail.jpeg” },
      { “term”:”describedby”, “base”:“http:\/\/museumX.org\/citysdk\/”, "id":"0" },
      { “term”:”self”, “id”:”1234”, “base”:”http:\/\/link-to-alternative-ids.com\/qrcode\/” },
      { “term”:”parent”, “id”:”2345”},
      { “term”:”child”,”id”:”0”}
   ],
 }
}
```

Figure 15 – A POI description message format example

Finally, it could be useful to have a simpler way to represent the data of a POI. In Figure 16, such representation is given, which will have the fields shown in Table 34. This minimalistic representation could be used for search results, where the full information of each returned POI is not necessary.

POI Field	Description	UML Attribute
POI ID	Unique identifier of the POI	POIBaseType::id
Name	Name of the POI	POIType::label
Description	Description of the POI	POIType::description
Category	Classification of the POI	POIType::category
Tags	Keywords to characterize the POI	POIType::category
Opening Hours	Time at which a given POI is open or closed	POIType::time
Prices	Price to enter a POI	POIType::description
Location	Where the POI is located	Location::point Location::line Location::polygon
Thumbnail	Images of the POI in thumbnail size	POIType::link

Table 34 – Minimal representation of a POI

```
{
 “poi”:
 {
  “base”: “http:\/\/example.pt\/poi\/”,
  “id”:”1234”,
  “lang”:”EN”,
  “label”: { “term”:”primary”, “value”:”Museu Calouste Gulbenkian” },
  “description”:
  [
    { “lang”:”PT”, “value”:”Descrição” },
    { “lang”:”EN”, “value”:”Description” },
    { “type”:”X-citysdk/price”, “lang”:”EN”, “value”:”0€ per person” }
  ],
  “category”:
  [
    { “term”:”category”, “value”:”Museum” },
    { “term”:”category”, “value”:”Park” },
    { “term”:”tag”, “value”:”culture”},
    { “term”:”tag”, “value”:”nature”}
  ],
  “location”:
  {
     “point”: {
       “term”:”entrance”,
        “Point”:
        {
           “srsName”:”http:\/\/www.opengis.net\/def\/crs\/EPSG\/0\/4326”,
           “postList”:”38.738369 -9.154962”
        },
      }
  },
  “time”:
  {
     “term”:”open”,
     “type”:”text\/calendar”,
     “value”:”BEGIN\:VCALENDAR
               VERSION\:2.0
               PRODID\:-\/\/hacksw\/handcal\/\/NONSGML v1.0\/\/EN
               BEGIN\:VEVENT
               UID\:uid1@example.com
               DTSTAMP\:19970714T170000Z
               ORGANIZER;CN=John Doe\:MAILTO\:john.doe@example.com
               DTSTART\:19970714T170000Z
               DTEND\:19970715T035959Z
               SUMMARY\:Bastille Day Party
               END\:VEVENT
               END\:VCALENDAR”
  },
  “link”:
  [
    { “term”:”related”, “rel”:”icon”, “type”:”image\/jpeg”, “href”:”http:\/\/link-to-multimedia.com\/thumbnail.jpeg” }
  ]
 }
}
```

Figure 16 – Example of a minimal description of a POI

The main differences between this model and the previous one reside in the following:

1. The minimal version has no contacts field;
2. The minimal version has no alternative IDs or parent POIs;
3. The minimal version has only a thumbnail and no other multimedia data;
4. The minimal version has no links to related applications;
5. The minimal version has no optional fields.

6.1.2 Event Format

Just like a POI, an event can be represented the same way just by using some of the fields. An event can be represented by the following:

1. **ID** - the identifier of the event;
2. **Name** - the name of the event;
3. **Description** - the description of the event;
4. **Category** and **Tags** - classifications and keywords that characterize the event and enable search;

5. **Date** - the date and time of the event;
6. **Location –** the POI where the event takes place;
7. **Images**;
8. **Price**.

The UML representation is shown in Figure 17 and each entity property in Table 35, along with an example in Figure 18.

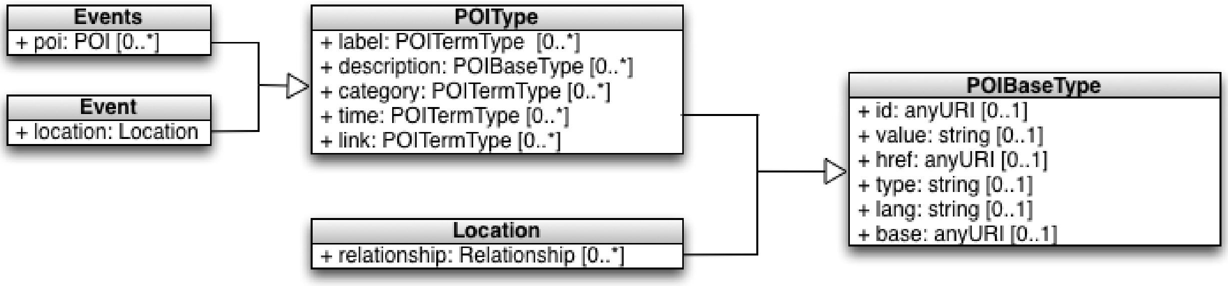


Figure 17 – Event UML representation

Event Field	Description	UML Attribute
Event ID	Primary identifier of the event	POIBaseType::id
Alternative IDs	Alternative ways to represent the POI: RFID, NFC, QR codes and Barcode	POIType::link
Name	Name of the event	POIType::label
Description	Description of the event	POIType::description
Category	Classification of the event	POIType::category
Tags	Keywords to characterize the event	POIType::category
Opening Hours	Time at which a given event is taking place	POIType::time
Prices	Price to enter at the event	POITermType
Location	Where the event is located	Location::relationship POIType::link
Image of the event	Image of the event	POIType::link

Table 35 – Representation of the event

```
{
 “event”:
 {
  “base”: “http:\/\/example.pt\/poi\/”,
  “id”:”2345”,
  “lang”:”EN”,
  “label”:
   [
     { “lang”:”PT”, “term”:”primary”, “value”:”A paisagem marítima no século XIX” },
     { “lang”:”EN”, “term”:”primary”, “value”:”The seascape in the nineteenth century” }
   ]
  “description”:
  [
    { “lang”:”PT”, “value”:”Descrição” },
    { “lang”:”EN”, “value”:”Description” },
    { “type”:”X-citysdk/price”, “lang”:”EN”, “value”:”0€ per person” }
  ],
  “category”:
  [
    { “term”:“category”, “value”:”Event” },
  ],
  “location”:
  {
    “relationship”:
    {
       “term”:”within”,
       “targetPOI”:”1234”
    },
  },
  “time”:
  {
     “term”:”open”,
     “scheme”:”http:\/\/link-to-scheme.com”,
     “type”:”text\/calendar”,
     “value”:”BEGIN\:VCALENDAR
              VERSION\:2.0
              PRODID\:-\/\/hacksw\/handcal\/\/NONSGML v1.0\/\/EN
              BEGIN\:VEVENT
              UID\:uid1@example.com
              DTSTAMP\:19970714T170000Z
              ORGANIZER;CN=John Doe\:MAILTO\:john.doe@example.com
              DTSTART\:19970714T170000Z
              DTEND\:19970715T035959Z
              SUMMARY\:Bastille Day Party
              END\:VEVENT
              END\:VCALENDAR”
  },
  “link”:
  [
    { “term”:”related”, “rel”:”icon”, “type”:”image\/jpeg”, “href”:”http:\/\/link-to-multimedia.com\/thumbnail.jpeg” },
    { “term”:”self”, “id”:”2345”, “base”:”http:\/\/link-to-alternative-ids.com\/qrcode\/” },
    { “term”:”child”, “id”:”1234”}
  ]
 }
}
```

Figure 18 – An event message format example

6.1.3 Route Format

Routes can be represented as a group of POIs, ordered in a way that a user can be guided to visit each POI in a certain matter (e.g.: chronological order). Each field of a route will be a minimal description of a given POI. Figure 19 shows the UML representing a route and Figure 20 an example of a message format.

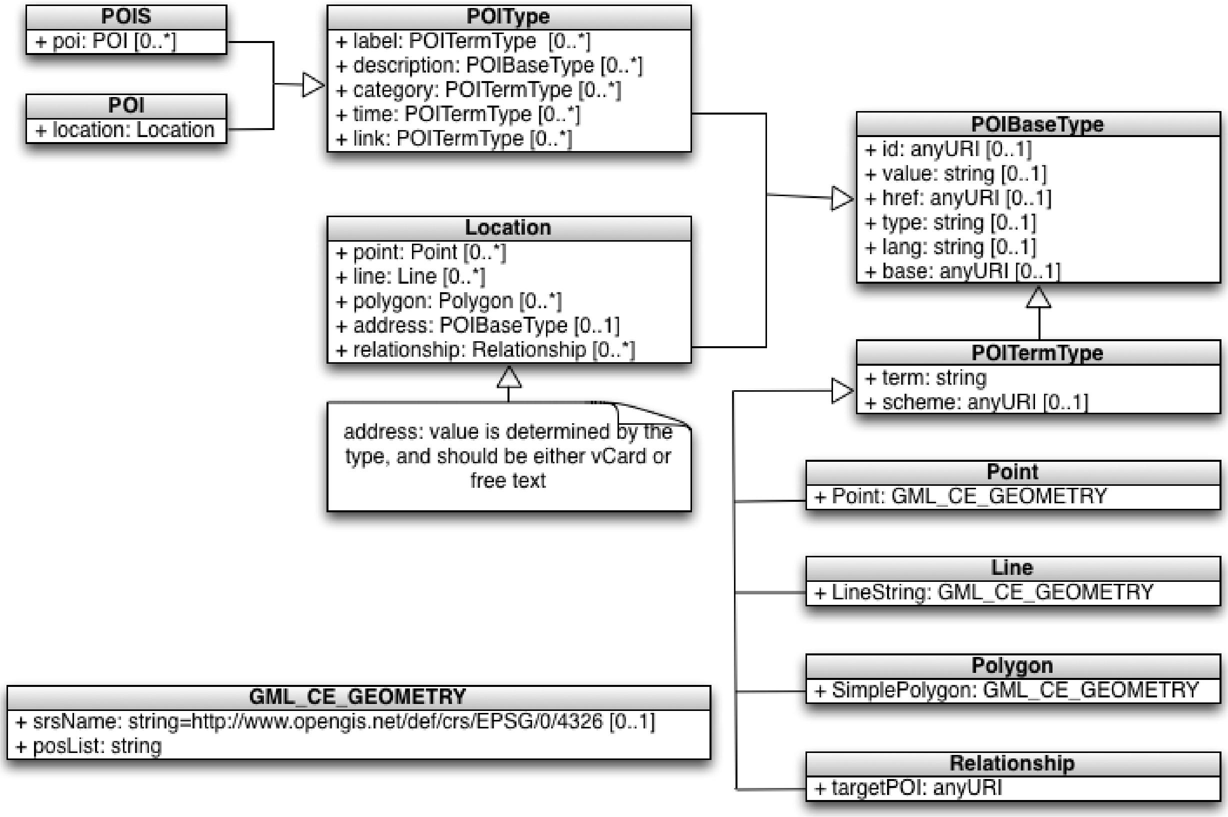

POIS
+ poi: POI [0..*]
POI
+ location: Location
POIType
+ label: POITermType [0..*]
+ description: POIBaseType [0..*]
+ category: POITermType [0..*]
+ time: POITermType [0..*]
+ link: POITermType [0..*]
POIBaseType
+ id: anyURI [0..1]
+ value: string [0..1]
+ href: anyURI [0..1]
+ type: string [0..1]
+ lang: string [0..1]
+ base: anyURI [0..1]
Location
+ point: Point [0..*]
+ line: Line [0..*]
+ polygon: Polygon [0..*]
+ address: POIBaseType [0..1]
+ relationship: Relationship [0..*]
POITermType
+ term: string
+ scheme: anyURI [0..1]
address: value is determined by the type, and should be either vCard or free text
Point
+ Point: GML_CE_GEOMETRY
Line
+ LineString: GML_CE_GEOMETRY
Polygon
+ SimplePolygon: GML_CE_GEOMETRY
GML_CE_GEOMETRY
+ srsName: string=http://www.opengis.net/def/crs/EPSG/0/4326 [0..1]
+ posList: string
Relationship
+ targetPOI: anyURI

Figure 19 – Route UML representation

```
{
  “pois”:
   {
   “base”: “http:\/\/example.pt\/route\/”,
   “id”:”45”,
   “lang”:”EN”,
   “label”:
    [
      { “lang”:”PT”, “term”:”primary”, “value”:”Rota da Amália”},
      { “lang”:”EN”, “term”:”primary”, “value”:”Amália’s Route” }
    ]
   “description”:
   [
     { “lang”:”PT”, “value”:”Descrição” },
     { “lang”:”EN”, “value”:”Description” },
   ],
  “poi”: [ “poi”:{ ... }, “poi”:{ ... }, “poi”:{ ... }, “poi”:{ ... } ]
  }
}
```

Figure 20 – A route message format example

6.2 Technical Details

REST (Representational State Transfer) is a style of software architecture for distributed systems and has emerged as a predominant Web service design model. It was proposed by Roy Fielding in his PhD dissertation [21] and it describes six constraints for the design of the REST architectural style:

1. **Client-Server** - communication is mainly made between the client and the server through the use of a uniform interface (explained in point 6);
2. **Stateless** - no state is saved in the server;
3. **Cacheable** - messages can be cacheable, explicitly or implicitly;
4. **Layered System** - clients can communicate with the server through the use of proxies or any intermediary nodes;
5. **Code on Demand** (optional) - extend the client through the transfer of executable code;
6. **Uniform Interface** - The uniform interface between clients and servers simplifies and decouples the architecture, which enables each part to evolve independently. There are four guiding principles for this interface and they are detailed below:
    a. Identification of resources;
    b. Manipulation of resources through representations;
    c. Self-descriptive messages;
    d. Hypermedia as the engine of application state (a.k.a. HATEOAS) - allows the client to discover future actions that can be taken, just by receiving hypermedia within each resource representation. This way, the client needs no prior knowledge of specific URIs (except for the root).

Using HTTP, all of the presented constraints are offered for free, except for point 6.d. By ignoring this guideline, we are unable to use the REST architectural design to its fullest. The pros and cons of using hypermedia APIs - that is, obeying the point 6.d - are detailed in Table 36.

Pros	Cons
Promotes scalability (the server decides what work can be done and takes care of load balancing or caching)	**Increased latency** (because it is the server that leads the client, there can be many messages exchanged between them)
Provides resilience to future changes (because the server and client are decoupled, the server can change without making the client inoperative)	**Bigger message payloads** (the messages need to represent the resources, but also the various states and steps that the client can go to)
Promotes decoupling and encapsulation (server and client are decoupled)	**Use lots of caching** (to decrease latency caching should be used, but with careful planning)

Reduces duplication of business logic (the server has the logic and leads the client, so the client doesn’t need to know what to do)	**Building a client requires more effort** (but in the long run, it can compensate)

Table 36 – Pros and Cons of hypermedia APIs

Bearing in mind the representations presented in previous sections, using an hypermedia API could be beneficial so to have the decoupling aspect between the client and the server.

As such, and since the presented representations already give the relationship between each POI and event, switching between states - that is, leading the client through the resources and their manipulation - is what is lacking in the API.

Since the developed applications will not edit the stored information, we only need to focus on the searching feature. This aspect needs to be added to the aforementioned representations, and there are some solutions that offer a way to do so.

Hypertext Application Language (HAL) [22] provides a set of conventions for expressing hyperlinks to, and embeddedness of, related resources - the rest of a HAL document is just plain old JSON or XML.

Collection+JSON[45] is a JSON-based read/write hypermedia-type designed to support management and querying of simple collections. It is similar to the the Atom Syndication Format [8] and the the Atom Publishing Protocol [23]. However, Collection+JSON defines both the format and the semantics in a single media type. It also includes support for Query Templates and expanded write support through the use of a Write Template.

Finally, **Siren+JSON**[46] offers structures to communicate information about entities, actions for executing state transitions, and links for client navigation. Siren is intended to be a general specification of a generic media type that can be applied to other types that are not inherently hypermedia-powered. The initial implementation is JSON Siren. Other implementations, such as XML Siren, may also be implemented using the Siren specification.

Looking into the short description of each convention, the Collection+JSON seems to be the more appropriate one to represent what we need, as it offers query templates on a set of collections - which in our case are POIs and events.

A small example of a query template using Collection+JSON is presented in Figure 21.

```
{ "collection" :
  {
    "version" : "1.0",
    "href" : "http:\/\/example.pt\/poi\/",
    "queries" : [
      {"rel" : "search", "href" : "http:\/\/example.poi\/poi\/search", "prompt" : "Search"
        "data" : [
          {"name" : "search", "value" : ""}
        ]
      }
```

[45] http://amundsen.com/media-types/collection/
[46] https://github.com/kevinswiber/siren

```
   ]
  }
 }
```

Figure 21 – Example of a query using Collection+JSON

Following the specifications, clients that support the Collection+JSON media type should be able to recognize and parse query templates found within given responses. Query templates consist of a data array associated with an href property. The queries array supports query templates.

For query templates, the name/value pairs of the data array set are appended to the URI found in the href property associated with the queries array (with a question-mark ["?"] as separator) and this new URI is sent to the processing agent. In the above example the following URI would be produced, if the user supplied Gulbenkian as its search:

```
http://example.pt/poi/search?search=Gulbenkian
```

Because Collection+JSON already formalizes certain aspects of the resources through the use of collections and items (just like W3C POI), it seems a better approach to just borrow the ideas from the specifications. So, following the W3C POI specification, we add the query functionality, similar to Collection+JSON, to our presented messages by inheriting the *POITermType* with a **query** field as shown below in Figure 22.

```
{
  “poi”:
  {
    (POI description)
    “query”:
    [
      { “term”:”poi-complete”, “base”:“http:\/\/example.pt\/poi\/search”, “id”:”complete” },
      { “term”:”poi-minimal”, ”base”:”http:\/\/example.pt\/poi\/search”, “id”:”minimal”}
    ]
 }
}
```

Figure 22 – Implementing search using W3C POI

The “poi-complete” term would allow the client to get a list of POIs matching the search with a complete description - that is, the message format that includes all the information regarding the desired POI. Likewise, the “poi-minimal” would give a minimal description of the POI. This field allows the search of specific details of a POI, like its name or the various fields of the description.

So, following the same procedure as the Collection+JSON, the produced URI would be similar to the one presented below (with “Gulbenkian” as our search):

```
http://example.pt/poi/search?minimal=Gulbenkian
```

The answer would then be a minimal description of the desired POIs that match the query. Searching through tags or categories can then be done in a similar fashion. As seen in Figure 23, the “poi-category” would search for POIs that match the specified category and the “poi-tags” would search for POIs with a given set of tags.

```
{
  “poi”: {
    (POI description)
    “query”:
    [
      { “term”:”poi-category”, “base”:“http:\/\/example.pt\/poi\/search”, “id”:”category” },
      { “term”:”poi-tags”, ”base”:”http:\/\/example.pt\/poi\/search”, “id”:”tag” },
      { “term”:”poi-complete”, “base”:“http:\/\/example.pt\/poi\/search”, ”id”:”complete” },
      { “term”:”poi-minimal”, ”base”:”http:\/\/example.pt\/poi\/search”, “id”:”minimal” }
    ]
  }
}
```

Figure 23 – Implementing tag and category search using W3C POI

In the next section, it will be shown the resources that will be used, how they relate with each other and how a given client can navigate through each of the resources.

6.3 Resources

The key resources used in the CitySDK Tourism API are focused on the presented message formats:

1. Points of Interest - which describe all the POIs available in a given city;
2. Events - description of all the events that are being or have been undertaken;
3. Routes - minimal description of a group of ordered POIs.

Following the presented requirements, we can extract three main resources: **poi**, **event** and **route**. There is one additional resource which is the **categories**, which is used - as presented - for the categorization of our key resources. The way a given client can use each of the resources or even search for them can be represented through a state machine shown in Figure 24.

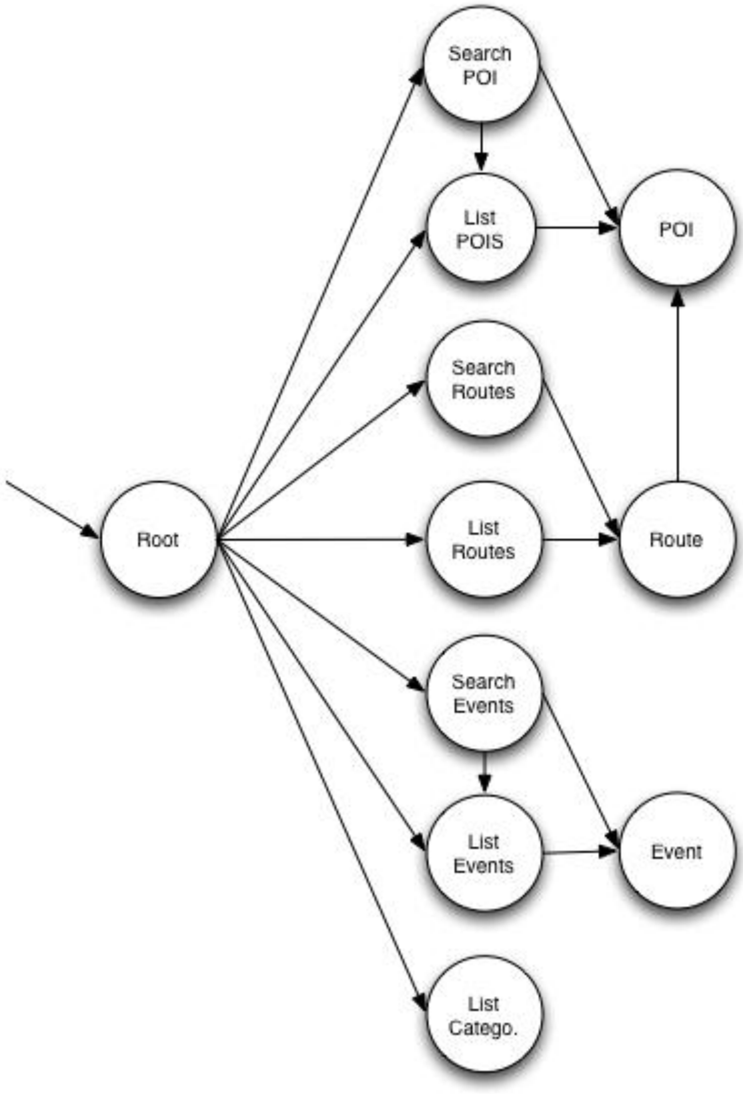


Figure 24 – State machine of available client procedures

Further explaining, a given client upon accessing the root URI - after retrieving it through a service discovery mechanism - it would receive the URIs of the searching methods, as well as the URIs giving access to a list of POIs, Events and Routes - as seen in Figure 25. Table 37 summarizes each term and each response and Table 38 exemplary URIs for each term.

```
{
  “_links”:
  {
    “query”:
    [
      { “term”:”poi-category”, “base”:“http:\/\/example.pt\/poi\/search”, “id”:”category” },
      { “term”:”poi-tags”, ”base”:”http:\/\/example.pt\/poi\/search”, “id”:”tag” },
      { “term”:”poi-complete”, “base”:“http:\/\/example.pt\/poi\/search”, “id”:”complete” },
      { “term”:”poi-minimal”, ”base”:”http:\/\/example.pt\/poi\/search”, “id”:”minimal” },
      { “term”:”poi-relation”, “base:”http:\/\/example.pt\/poi\/\/search”, “id”:”relation” },
      { “term”:”poi-coords”, “base”:”http:\/\/example.pt\/poi\/”, “id”:”coords” },
      { “term”:”event-category”, “base”:“http:\/\/example.pt\/event\/search”, “id”:”category”  },
      { “term”:”event-tags”, ”base”:”http:\/\/example.pt\/event\/search”, “id”:”tag” },
      { “term”:”event”, “base”:“http:\/\/example.pt\/event\/search”, “id”:”search” },
      { “term”:”event-time”, “base”:”http:\/\/example.pt\/event\/search”, “id”:”time” },
      { “term”:”event-coords”, “base”:”http:\/\/example.pt\/event\/search”, “id”:”coords” },

      { “term”:”route-category”, “base”:“http:\/\/example.pt\/route\/search”, “id”:”category” },
      { “term”:”route-tags”, ”base”:”http:\/\/example.pt\/event\/search”, “id”:”tag”},
      { “term”:”route”, “base”:“http:\/\/example.pt\/event\/search”, “id”:”search” },
      { “term”:”route-coords”, ”base”:”http:\/\/example.pt\/route\/”, “id”:”coords” },

      { “term”:”categories”, “base”:”http:\/\/example.pt\/category”, “id”:”list” }
    ],
  }
}
```

Figure 25 – Representation of the data given by the root

Term	Description	Response
poi-category	Search POIs using categories.	List of POIs with minimal description matching the specified categories.
poi-tags	Search POIs using tags.	List of POIs with minimal description matching the specified tags.
poi-complete	Search for a given POI. This search should also include description.	List of POIs with complete description matching the search.
poi-minimal	Search for a given POI. This search should also include description.	List of POIs with minimal description matching the search.
poi-relation	Searches for the relations of a given POI - only available when consulting a specific POI. The searches that should be used are “child” and “parent”.	List of all the POIs that have the specified relationship with the POI.
poi-coords	Get a list of all available POIs within a given set of WGS84 coordinates (latitude, longitude and a radius).	List of all the POIs within a given set of coordinates.
event-category	Search events using categories	List of events with the specified categories.
event-tags	Search events using tags	List of events with the specified tags.
event	Search for a given event. This search should also include description.	List of events matching the specified search.
event-time	Get a list of all available events for a given date or a range of dates (in YYYY-MM-DD[-HH-MM-SS] format)	List of events that are within the specified time interval.
event-coords	Get a list of all available events that are within a given set of WGS84 coordi-	List of events within the specified coordinates.

	nates.	
route-category	Search routes using categories.	List of routes matching the specified categories.
route-tags	Search routes using tags,	List of routes matching the specified tags.
route	Search for a given route. This search should also include description.	List of routes matching the search.
route-coords	Get a list of all available routes within a given set of WGS84 coordinates.	List of routes matching the specified coordinates.
categories	Get a list of all available categories.	List of all categories

Table 37 – Term explanation and responses

Term	Base	ID	Example URI
poi-category	http://example.pt/poi/search	category	http://example.pt/poi/search?category=bar
poi-tags	http://example.pt/poi/search	tag	http://example.pt/poi/search?tag=music
poi-complete	http://example.pt/poi/search	complete	http://example.pt/poi/search?complete=Aula&nbspMagna
poi-minimal	http://example.pt/poi/search	minimal	http://example.pt/poi/search?minimal=Aula&nbspMagna
poi-relation	http://example.pt/poi/1234/search	relation	http://example.pt/poi/1234/search?relation=<child> || <parent>
poi-coords	http://example.pt/poi/search	coords	Single point: http://example.pt/poi/search?coords=<lat>,<lon>,<radius> Polygon: http://example.pt/poi/search?coords=<lat>,<lon>&nbsp<lat>,<lon>
event-category	http://example.pt/event/search	category	http://example.pt/event/search?category=music
event-tags	http://example.pt/event/search	tag	http://example.pt/event/search?tag=rock
event	http://example.pt/event/search	search	http://example.pt/event/search?search=Steve&nbspVai
event-time	http://example.pt/event/search	time	http://example.pt/event/search?time=<start>&nbsp<end>

event-coords	http://example.pt/event/search	coords	Single point: http://example.pt/poi/search?coords=<lat>,<lon>,<radius> Polygon: http://example.pt/poi/search?coords=<lat>,<lon>&nbsp<lat>,<lon>
route-category	http://example.pt/route/search	category	http://example.pt/event/search?category=cultural
route-tags	http://example.pt/route/search	tag	http://example.pt/route/search?tag=fado
route	http://example.pt/route/search	search	http://example.pt/route/search?search=Rota&nbspda&nbspAmália
route-coords	http://example.pt/route/search	coords	Single point: http://example.pt/poi/search?coords=<lat>,<lon>,<radius> Polygon: http://example.pt/poi/search?coords=<lat>,<lon>&nbsp<lat>,<lon>
categories	http://example.pt/category	list	http://example.pt/category?list

Table 38 – Example URIs of each term

From the root URI - the only URI known - the client would know how to search for a given event, poi, route, as well as accessing a list of them and categories. By sending the following URI:

```
http://example.pt/poi/search?coords=38.73872,-9.154836,3
```

The client would receive a list of all the available POIs within a 3 meter radius of the specified coordinates (in minimal description), as shown in Figure 26.

```
{
"poi":
[
 {
 "poi": {
   "base": "http:\/\/example.pt\/poi\/",
   "id":"1234",
   "lang":"EN",
   "label": { "term":"primary", "value":"Calouste Gulbenkian Museum" },
   "description":
   [
     { "lang":"PT", "value":"Descrição" },
     { "lang":"EN", "value":"Description" },
     { "type":"X-citysdk/price", "lang":"EN", "value":"0€ per person" },
   ],
   "category":
```

```
  [
    { “term”:”category”, “value”:”Museum” },
    { “term”:”category”, ”value”:”Park” },
    { “term”:”tag”, “value”:”culture”},
    { “term”:”tag”, “value”:”nature”}
  ],
  “location”:
  {
      “point”: {
         “term”:”entrance”,
         “Point”:
         {
            “srsName”:”http:\/\/www.opengis.net\/def\/crs\/EPSG\/0\/4326”,
            “postList”:”38.738369 -9.154962”
         },
      }
  },
  “time”:
  {
      “term”:”open”,
      “type”:”text\/icalendar”,
      “value”:”BEGIN\:VCALENDAR
                 VERSION\:2.0
                 PRODID\:-\/\/hacksw\/handcal\/\/NONSGML v1.0\/\/EN
                 BEGIN\:VEVENT
                 UID\:uid1@example.com
                 DTSTAMP\:19970714T170000Z
                 ORGANIZER;CN=John Doe\:MAILTO\:john.doe@example.com
                 DTSTART\:19970714T170000Z
                 DTEND\:19970715T035959Z
                 SUMMARY\:Bastille Day Party
                 END\:VEVENT
                 END\:VCALENDAR”
  },
  “link”:
  [
     { “term”:”related”, “rel”:”icon”, “type”:”image\/jpeg”, “href”:”http:\/\/link-to-multimedia.com\/thumbnail.jpeg” }
  ]
 }
 },
“poi”: { ... },
“poi”: { ... }
]
}
```

Figure 26 – Representation of the POIS state

In a similar fashion, the events and routes would follow the same procedure. Finally, from this list, the client can now navigate to an actual POI and its representation would be a complete description of the selected POI. The same applies to events and routes, that is, upon selecting a given event or route, it would receive the information about the selected resource. A route could further go into detailing the information about a given POI. For more complex searches, that is, combining different parameters, if the base URI is the same, it would simply be a combination of IDs. An example of such URI could be the following one:

```
http://example.pt/event/search?category=music&time=2012-12-01&nbsp2012-12-01
```

In this case, all events matching the category “music” that would take place in the 1st of December of 2012 would be returned, since the start and the end date are the same. Another example is the use of multiple coordinates with other parameters, like the one shown below:

```
http://example.pt/poi/search?coords=<lat>,<lon>&nbsp<lat>,<lon>&nbsp<lat>,<lon>&nbsp<lat>,<lon>&category=restaur
ant&tags=”portuguese food,tejo”
```

Finally, and since it was not previously described what is the format of a category message, in Figure 27, we present the result of querying the list of available categories. There will always be a national and english version of each category and each category value must be unique.

```
{
 “categories”:
 [
  “category”:
  [
   { “lang”:”PT”, “value”:”alojamento”,
    “categories”:[“category”:{“lang”:”PT”,“value”:“hotéis”},{“lang”:”PT”, “value”:“hostel”},{“lang”:”PT”,”value”:“motel”]},

   { “lang”:”EN”, “value”:”housing”,
  “categories”:[“category”:{“lang”:”EN”,”value”:“hotels”},{“lang”:”EN”,”value”:“hostel”},{“lang”:”EN”,”value”:“motel”]},

   ],
    “category”:[...]
  ]
}
```

Figure 27 – Categories message format

7 Tourism Lead Pilot (Lisbon)

7.1 Open Data in Lead Pilot

The Lisbon Municipality has a project named “Open Data Lx”, built together with the Agency for Administrative Modernization, whose purpose is to provide to all citizens (in particular developers) datasets regarding the city of Lisbon, in several branches. The data provided can be used for academic papers or in useful applications for the city. “Open Data” projects aim to open data that was hidden for ages through reels of red tape.

Transparency, citizen participation, creative economy and innovation ecosystem increases are what should sustain Open Data initiatives, by ensuring the use or reuse of data in the creation of high potential services/products.

7.1.1 Selected Data Sets

From the datasets that the Lisbon Municipality provides, the ones related with tourism and events, mostly supplied by the Municipality itself and ATL - Lisbon Tourism Association, those which are relevant to the domain are:

- **Restaurants** - where you can find restaurants divided in subsets such as traditional food, Italian restaurants, pubs, among others;
- **Culture** - where you can find information about the cultural facilities in the city and the venues that take place;
- **Tourism** - where you can find information about sightsee and the best touristic offer of the city;
- **Sports** - where you can find information about places to practice sports or places where you can watch sport events;
- **Business** - where you can find information about traditional shops, such as handicraft and souvenirs, as well as the shopping centers and the biggest European companies.

7.2 Architecture

Figure 28 depicts the proposed CitySDK platform’s architecture for the Lisbon Pilot. The selected datasets can be obtained from the different available city’s data sources:

- LXI - *Lisboa Interactiva*[47] - POI datasets;
- Open Data LX[48] - POI datasets;
- *Agenda LX*[49] - cultural events;
- Municipal Archive[50] - multimedia content such as photos for POIs.

[47] http://lxi.cm-lisboa.pt/
[48] http://www.lisboaparticipa.pt/pages/smartlx.php/A=18___collection=cml_article_smartlx
[49] http://www.agendalx.pt/

Figure 28 depicts how these data sources will be combined into the CitySDK service to be provided by the Lisbon Municipality. Data adapters will be built for each of these data sources, enabling the CitySDK database to be fed.

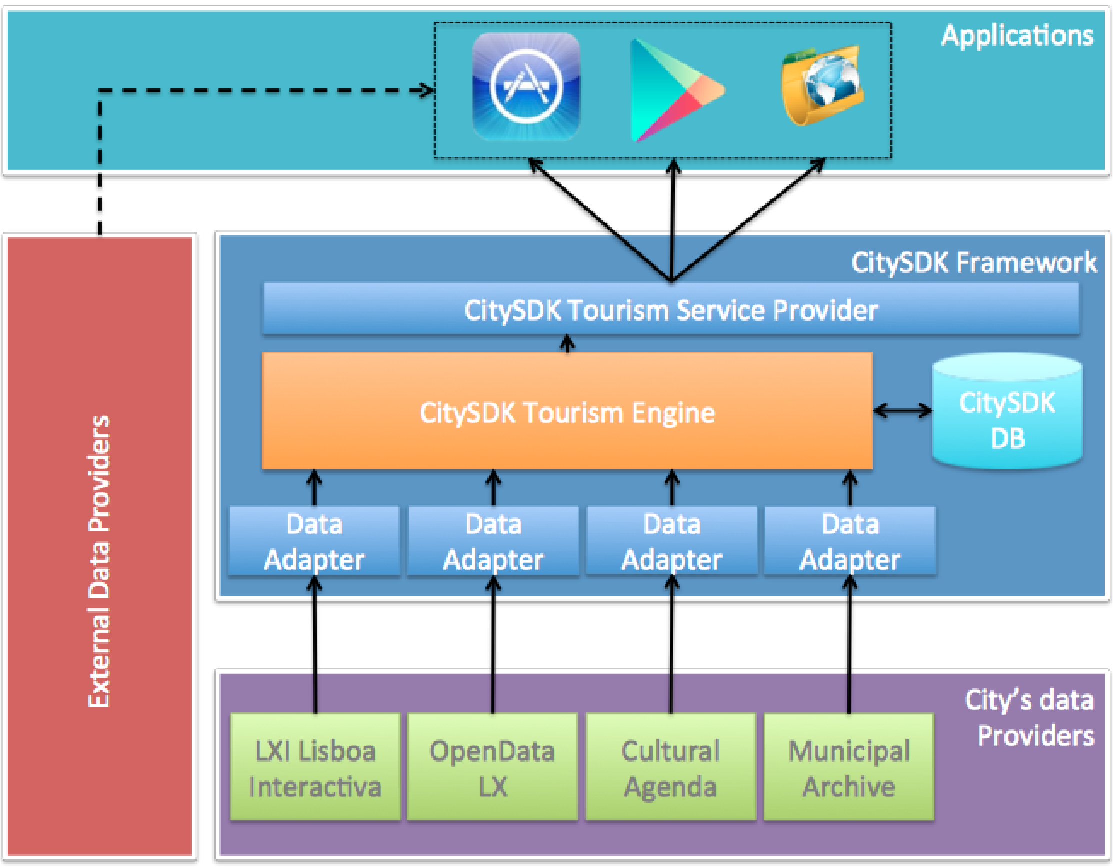


Figure 28 – CitySDK Tourism platform architecture for the Lisbon pilot

Developers are expected to combine the data provided by CitySDK together with other sources to create rich applications. The value of the applications will come not only from the CitySDK data, but from the clever integration of other data sources and social information to create new applications which are not yet available or imagined.

7.3 Applications to be developed

The ability to access publicly available tourism data opens a panoply of possibilities and applications that can be developed with the same purpose. We intend to provide a group of exemplary applications that make use of the CitySDK Tourism API in order to demonstrate the possibilities that can be achieved with the presented API.

7.3.1 Application 1 – Mobile Guide

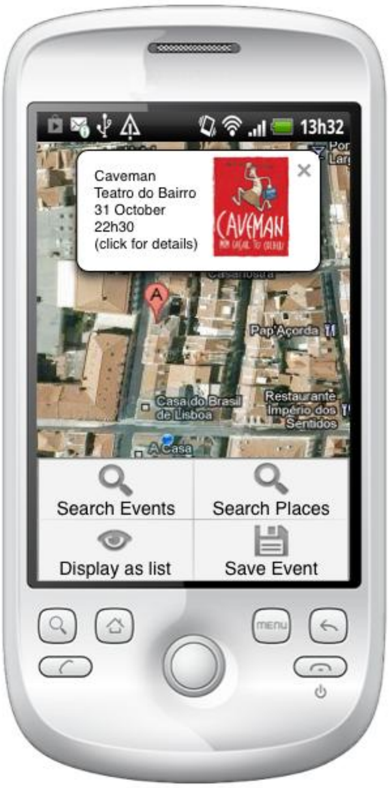
The main purpose of this application is to allow the user to search for a given POI or event. It

[50] http://arquivomunicipal.cm-lisboa.pt/default.asp?s=12079

will either provide a group of POIs/events in a list form or within a given map. The user will be allowed to visualize each POI in a more descriptive manner as an individual item or through a pop-up on the map. Also, if a user is interested in attending an event of his/her interest, he/she can save it and later on be alerted to go to that same event. An optional feature for this application is an alert mechanism that warns the user - based on the preferences/configurations that the user has specified - if he/she is near a POI or an event that will take place in a near future.

This application has more potential and it is more useful if implemented in mobile devices like Android or iPhone.

Figure 29 shows a mockup of the application.



Caveman
Teatro do Bairro
31 October
22h30
(click for details)

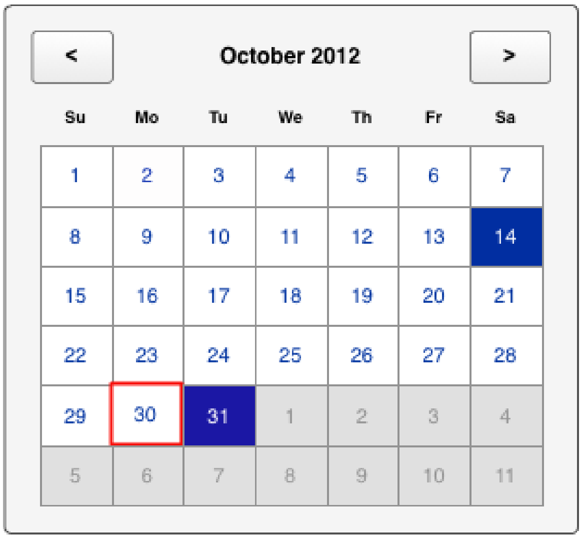
Figure 29 – Application 1 mockup

7.3.2 Application 2 – Web Widgets

Web widgets can be useful for a user to find events. They are especially useful if they are also able to provide information about other related events.

This application is meant to be a Web widget with the goal of providing the ability of event searching - using specific filters through the use of categories, tags, dates or locations -, save them and warn the user of their occurrence. Also, when the user opens his/her webpage, the saved events can be shown in a calendar or list style. This application can also be implemented in mobile devices.

Figure 30 shows the intended application.



October 2012

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

Figure 30 – Application 2 mockup

7.3.3 Application 3 – Augmented Reality POI Visualizer

This application allows the user to get routes that travel through specific POIs. The user can either get a list of routes or search for specific ones, or even provide the application enough POIs for it to suggest a given route. Such routes can be shown within a map.

This application can either be a Web application, thus, allowing the user to print the routes, or in a mobile device and, thus, allow the user to use the GPS to check his/her current position on his/her travel. It can be further complemented through the use of Layar[51], which provides the ability to use augmented reality within a given application.

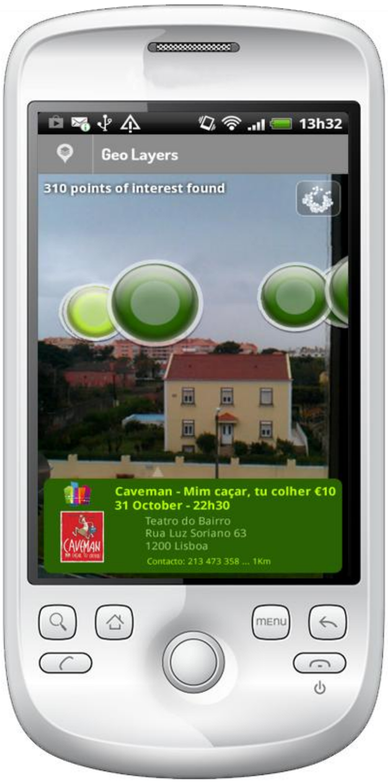
Figure 31 shows a mockup of the presented application.

7.4 Developer Engagement

Developer engagement - which is a crucial part of the Tourism domain - was launched using various tools. Firstly, a Facebook page[52] was set up to support the general awareness creation and developer engagement. The page has gradually been promoted since October.

Secondly, the Municipality is setting up an App contest with a major telecommunications company, promoting the use of Lisbon's open data as well as the CitySDK project outputs. The App contest aims at developing and strengthening a community of developers and giving them the conditions to set up business models to distribute their apps. This initiative will be an annual event that lasts approximately two months culminating in the deployment of the winner apps.

[51] http://www.layar.com/
[52] https://www.facebook.com/CitySDKLisbon



310 points of interest found

Caveman - Mim caçar, tu colher €10
31 October - 22h30

Teatro do Bairro
Rua Luz Soriano 63
1200 Lisboa

Contacto: 213 473 358 ... 1Km

Figure 31 – Application 3 mockup

Finally, in the scope of the Lisbon Participatory Budget, one of the proposals most voted aims to create an App contest directed to leverage the usage of the available datasets in the local Open Data initiative. This clearly reflected the local awareness of the importance of Open Data for emergent applications. One of the domains that will be addressed by the competition will be the one of Tourism. The budget approved for this initiative is of 100,000 EUR.

8 Replication Pilots

This section will describe the application of the CitySDK Tourism service on each of the replication pilots. For each of the pilots, the selected data sources will be presented and the particular architecture described. Still in this section, the foreseen applications to be developed using and to validate the CitySDK API are presented, as well as the effort conducted by each of the replication pilots in order to engage the developer community.

8.1 Amsterdam

8.1.1 Open Data in Pilot

The Amsterdam open data initiative[53] is a great example of how the municipality sees the importance of making data publicly available. There are datasets available in various categories such as culture, education, economy, recreation, etc.

Also, *Arts Holland*[54] from Waag (which hosts open data that is made available by third parties) shows that open data does not have to come from just the municipality. While the data available on *Arts Holland* is not yet fully complete, considering the status of open data in Amsterdam at the moment, it is a very promising platform that can be incorporated into CitySDK.

We will be using data from *Arts Holland* as primary input in the replication pilot in Amsterdam. Extra data from Amsterdam open data can be used to enrich the data from *Arts Holland*. Also, because several more datasets will be added to the Amsterdam open data platform next year, more datasets might be used to incorporate into CitySDK.

8.1.2 Architecture

Figure 32 shows the data sources that will be used as input for CitySDK in the replication pilot. Data adapters will have to be created to make sure the open data from the various platforms is suitable for CitySDK. Also, the dynamic tourist data will consist of data such as queue's waiting time and POI occupation. The data this provides will be optional and can used as a proof of concept to show what is possible with open sensor data.

[53] http://www.amsterdamopendata.nl
[54] http://artsholland.com/

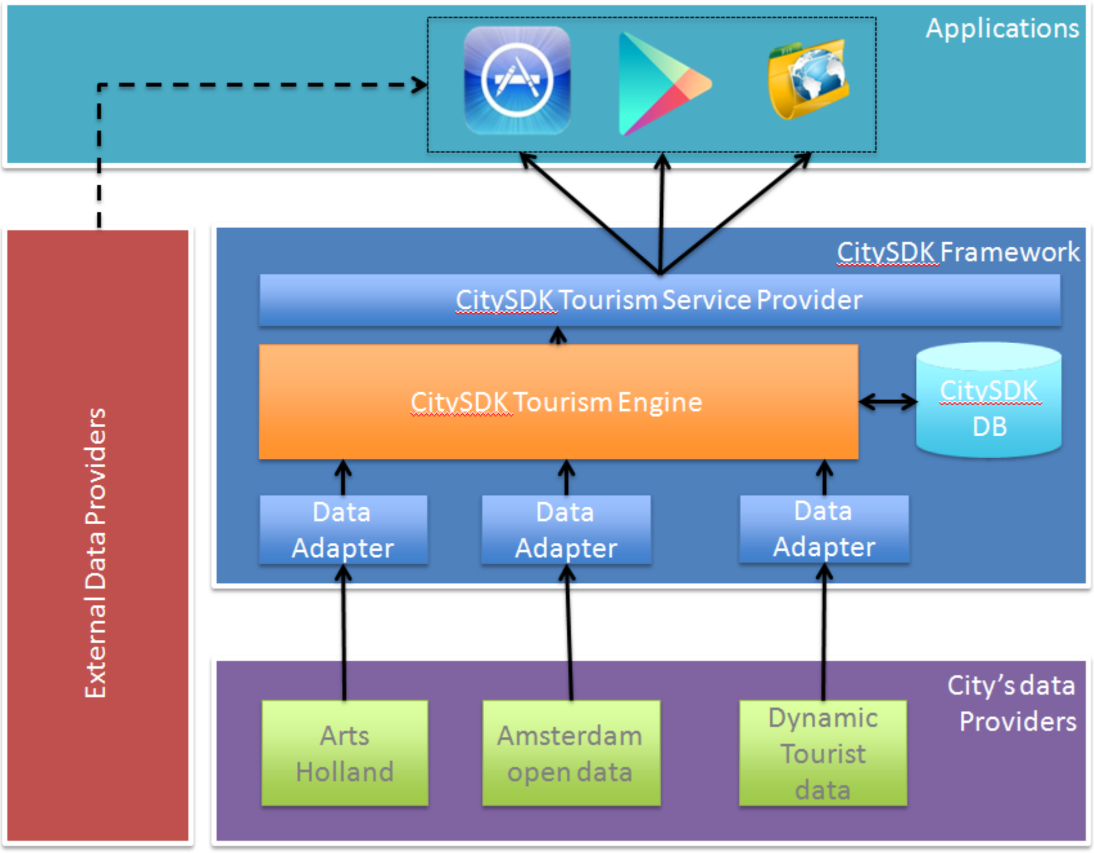


Figure 32 – CitySDK Tourism platform architecture for the Amsterdam replication pilot

8.1.3 Applications to be developed

Application 1 – Dynamic Routing

This application will guide users on a touristic route which guides the user to several POI's. This application differs from the application 3 (described in the 7.3.3) in a way that it also uses information about the POI's queue's waiting time and POI occupation (described in 6.1.1). The user can pre-define how long they are willing to stand in line before they can visit the POI. By using this information, the application will be able to compute a route through the city that will dynamically change according to the waiting time and capacity of the POI. It will also take the user preference into account by creating a route that will be customized for every user.

The following features will be implemented in the application:

- Digital tour guide with navigational function.
- Works with the CitySDK API to receive data from the city and information about POI's.
- Is able to compute a dynamic route based on user's preference, the queue's waiting time and POI capacity.

8.1.4 Developer Engagement

In Amsterdam, various initiatives have been set up to increase the developer engagement. Guest lectures have been given to Computer Science students to motivate them to look at the

CitySDK and understand the importance of the framework. From these lectures, we think they will think of implementing CitySDK when creating apps in the future.

CitySDK was also promoted in the municipality themselves, by explaining the possibilities to the different district managers in Amsterdam. They have close connections with developers that are situated in their district and who are creating apps for their district. Using CitySDK, they will be able to create richer applications of their district or the entire city.

Next year, we are planning to organize a DevCamp for students of the university, and let them create or design apps based on CitySDK. This is a great way to show the power of CitySDK, as well as finding concepts of possible apps to be created using the CitySDK framework.

8.2 Helsinki

8.2.1 Open Data in Pilot

The city of Helsinki has a wide range of tourism-related open data available. Open data in Helsinki relating to tourism include both Point of Interest-based data and Point of Event-based data. One of the incubators for opening data has been the *Helsinki Region Infoshare* project. It aims to make regional information quickly and easily accessible to all. Different sorts of data are available on distinct topics.

- **Events & Culture -** The Helsinki Travel and Tourism bureau gathers event information[55].
- **Service POI's** - The REST API for the City of Helsinki Service Map[56].
- **Traffic** - Related to the Mobility package, Helsinki Region Travel offers an API about the schedules and routes of public transport in Helsinki.

The goal is to search an optimal and coherent way of presenting the tourism-related open data and share these. Furthermore, rather than using the aforementioned distinct API's, the vision would be to have one shared API for the data. Through the CitySDK project, great possibilities exist to also achieve a true European harmonization in this.

8.2.2 Architecture

In the CitySDK's Tourism work package in Helsinki, we would like to test the compatibility of the *Linked Events* to the W3C POI format.

The *Linked Events* project[57] aims to develop the event API of the City of Helsinki. In the first phase of the project, one can search for the event data from the *Helsinki Travel & Tourism Office*. The API is a JSON based REST interface which provides information about the events, categories and places contained in this service.

The EventsML standard[58] (see Figure 33 below) is already in use and a well documented

[55] www.visithelsinki.fi
[56] http://www.hel.fi/palvelukarttaws/rest/ver2_en.html
[57] http://events.hubi.fi/
[58] http://www.iptc.org/site/News_Exchange_Formats/EventsML-G2/

standard. It models events and points of interest separately. Events can happen in many places (POI's) and a place can have many events. Both can have hierarchies. It is extensible and Semantic Web compatible.

Compared to W3C POI, EventsML also includes events, persons and organizations. However, it is more complex because it has metalevels and extensions.

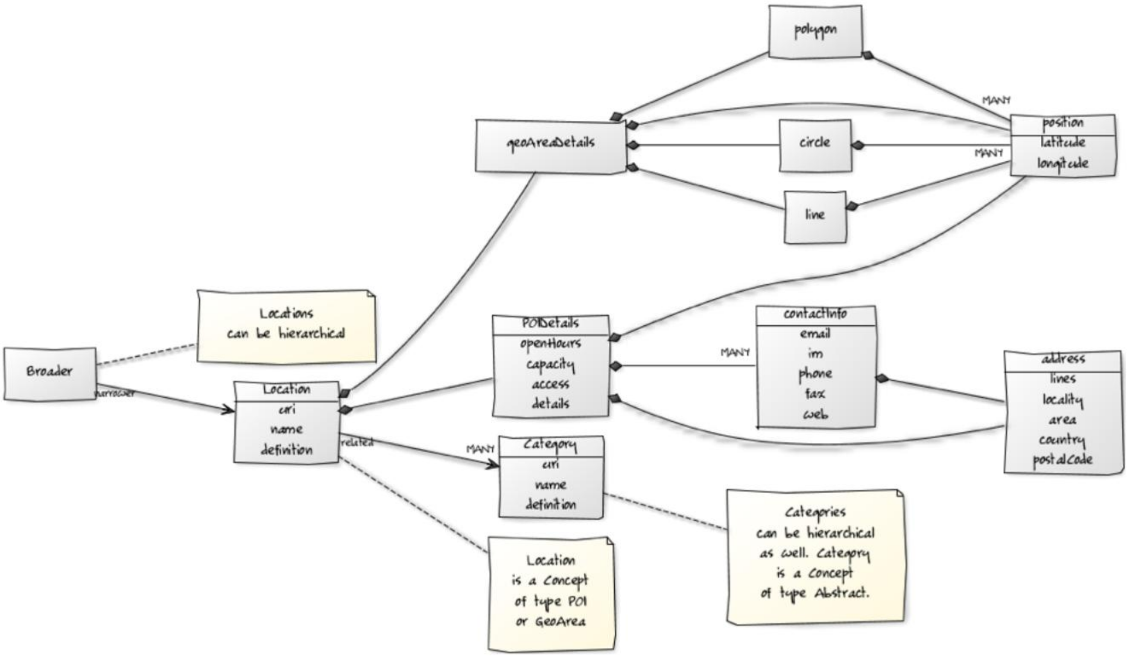


Figure 33 – EventsML data model

8.2.3 Applications to be developed

The goal of the replication pilot in Helsinki is not to develop applications as such. However, the desire is to be able to utilize the applications produced in the lead pilot of the package also locally in Helsinki. In addition, current networks with developers and other projects will be utilized in order to get the API developed in use in other applications. The goal is also to get feedback from the application developers themselves.

8.2.4 Developer Engagement

The developer engagement of the tourism domain is being promoted together with the other domains of the CitySDK project in Helsinki. In order to engage the developer community, different competitions could work as an efficient channel. For example, the *Apps4Finland* competition in Helsinki is one of the channels of fostering the use of the API's offered in the CitySDK project to developers.

Through existing networks with developers, entrepreneurs and public sector, *Forum Virium Helsinki* is taking forward the use of the developed API's. In addition, developer meetings will be arranged during the next spring to more specifically promote the Tourism-related open data in Helsinki. The hope is to achieve improved API's to developer feedback. Through the CitySDK project, we hope to also achieve European level collaboration with developers.

8.3 Lamia

8.3.1 Open Data in Pilot

The Municipality of Lamia has implemented the interfaces necessary to provide ease of access to a variety of open datasets regarding touristic information within the Municipality. The datasets themselves, as well as the publicizing interfaces, have been designed with the purpose of opening the existing rich, but strictly city-oriented, touristic information knowledge to the widest possible range of developers and as a counter effect to the widest group of citizens and visitors.

In that respect, the Municipality, through its web interfaces[59], provides a number of dedicated datasets in the form of REST Web Services. Two broad categories of provided data can be recognized:

- **Points of Interest (POIs)**, representing significant city provisions, touristic attractions and sightseeing highlights within the borders of the Lamia Municipality.
- **Points OF Events (POEs)**, reflecting locations of interest for tourists and citizens where special happenings and events take place.

Typical POIs may refer to accommodation information, restaurants, traditional product sale locations, historical and archaeological sites, activities and sports clubs and proposed sightseeing routes, while POEs may include concerts, art exhibitions, discount markets and cinema/theatre schedules or even political events and public speeches.

8.3.2 Architecture

As aforementioned, open datasets are provided in the form of REST Web Services in dedicated web views of the Municpality's website. Apart from simple listings of all documented and stored POIs and POEs, the provided Web Services offer a variety of different functionalities for the developers. In more detail, through their appropriate manipulation a developer can:

- request a list of all POI/POE basic/parent categories,
- request a list of all POI/POE subcategories (more specialized categories),
- request a list of all POI/POE subcategories belonging to specific categories,
- request a list of all POI/POE categories referring to specific regions of the Municipality,
- request a list of all POI/POE belonging to specific categories,
- request a list of all POI/POE referring to certain regions of the Municipality,
- request a list of all POI/POE with specific key-words identified in their title or description and

[59] http://109.123.109.139/~lamiagr/?q=rest/pois/poi-articles
http://109.123.109.139/~lamiagr/?q=rest/pois/poi-regions
http://109.123.109.139/~lamiagr/?q=rest/pois/poi-categories
http://109.123.109.139/~lamiagr/?q=rest/pois/poi-events
http://109.123.109.139/~lamiagr/?q=rest/pois/poe-regions
http://109.123.109.139/~lamiagr/?q=rest/pois/poe-organizers
http://109.123.109.139/~lamiagr/?q=rest/pois/poe-categories

- request a list of all POEs organized by certain organizers.

The Municipality's employees and even registered, authenticated and authorized external personnel exploit the appropriate content management interfaces for storing the available information on the SDK databases. Furthermore, the necessary data processing adapters for the transformation and adaptation of existing data sources to the CitySDK context are provided. Figure 34 illustrates a high level overview of the system's architecture.

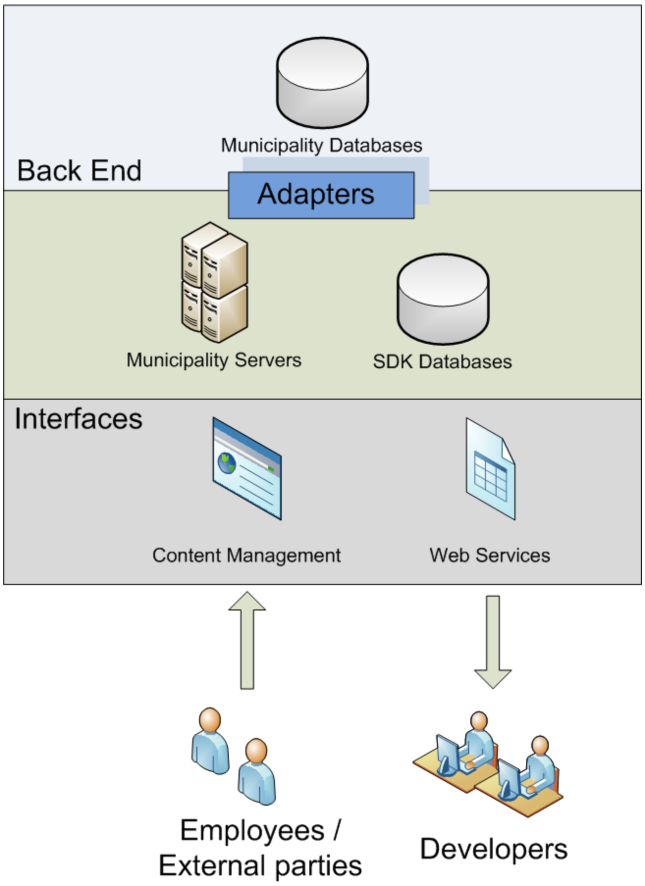


Figure 34 – CitySDK Tourism platform architecture for the Lamia replication pilot

8.3.3 Applications to be developed

In order to clearly demonstrate the possibilities for application development and further disseminate the results of the data opening process, a mobile application offering a touristic guide for the city is implemented within the Municipality. A snapshot of the application in action is illustrated in Figure 35.

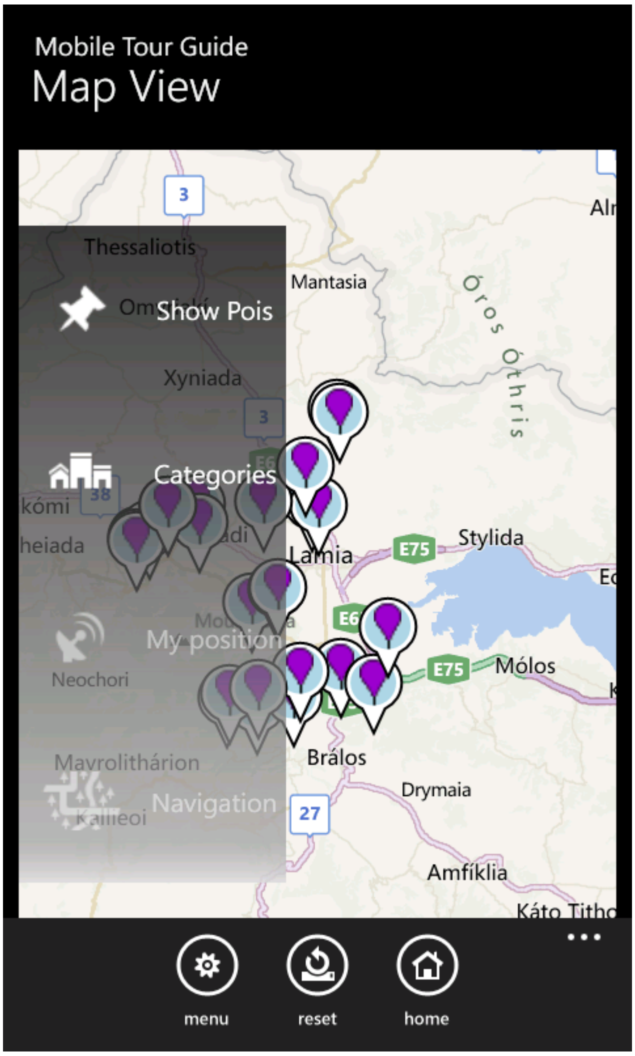


Figure 35 – Example of a developed Lamia touristic guide mobile application

Users can customize the application in terms of location acquisition consent, sounds incorporation, location marker update distance setup, nearest POIs distance setup and time period specification for POEs search. Depending on the users' provisions, different functionalities are available during the operation of the application. Simple text screens with POI and POE information are available to users through a specifically designed search page. Search filters include categories of data, text filters and certain time periods in cases of POEs search. POI and POE information pages include descriptive texts, location information, pictures, relative web links, etc. The user can be navigated to POI and POE information pages from a map view or an augmented reality environment as well. When in map context, users perceive POIs and POEs through distinguished markers located on the appropriate geo-location of each data item. When in the augmented reality mode, users are able to identify POI and POE objects in space via the device's camera view and according to the camera's pointing direction. By tapping on the presented markers, users are navigated to the selected POI and POE information pages. In the latter, users are offered the opportunity to select to be routed towards the demonstrated POI/POE starting from their current location. Both graphical as well as textual representation of the route is available. Moreover, in map view and in augmented reality mode, users can select to be automatically informed about nearby POI/POE objects based on their current position.

8.3.4 Developer Engagement

Our first approach to local developer community was the presentation we organized at Annual meeting of High School’s Computer Science Teachers. We accomplished two things. Firstly we engaged teachers (many of them are part time developers) and secondly high school students will be informed about CitySDK project.

Two events took place for local Academic audience, at Technological Education Institution of Lamia and at University of Central Greece. Audience consisted of Professors of Departments of Computer Science and students.

Furthermore, one more event was organized at Technological Chamber of Greece – Department of Lamia. Audience consisted of Software engineers.

8.4 Rome

8.4.1 Open Data in Pilot

As already mentioned, Province of Rome already supplies touristic information in open-data format. Currently, they are only available in Italian language, both for field names and for content, and they are not referable to any standardization.

Thus, Province of Rome is aiming to provide some of its touristic data in the above mentioned format, starting from two specific datasets, namely the list of daily touristic events and the list of archaeological sites in the province.

8.4.2 Architecture

The pilot project will consist in creating a REST endpoint which will supply data in JSON format. As mentioned, through the REST endpoint will be available two different dataset:

- *Roma&più* list of daily touristic events - POI datasets;
- Archaeological sites in the province - POI datasets.

The first one has to be updated daily, whilst the second one has a low refresh frequency.

8.4.3 Developer Engagement

Since next January 2013, Province of Rome will launch a series of events whose aim is, among the others, to involve developer communities in CitySDK project goals. We will start on January, 18th with *WhereCamp*, in which the most relevant technical aspects related to maps, geolocation, Geoweb services etc. will be discussed.

9 References

[1] International Press Telecommunications Council (IPTC), “NewsML-G2 Specification Version 2.12 - Core Conformance Level,” 2012.

[2] B. Desruisseaux, “Internet Calendaring and Scheduling Core Object Specification (iCalendar). Internet RFC 5545,” September 2009. [Online]. Available: URL: http://www.rfc-editor.org/rfc/rfc5545.txt.

[3] International Press Telecommunications Council (IPTC), “NewsML-G2 Implementation Guide 5.0,” 2012.

[4] A. Hill and M. Womer, “ W3C Points of Interest Core. W3C Working Draft. (Work in progress.),” 16 March 2012. [Online]. Available: URL: http://www.w3.org/2010/POI/documents/Core/core-20111216.html.

[5] N. Freed and N. Borenstein, “Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. Internet RFC 2046,” November 1996. [Online]. Available: URL: http://www.ietf.org/rfc/rfc2046.txt.

[6] H. Alvestrand, “Tags for the Identification of Languages. Internet RFC 3066.,” January 2001. [Online]. Available: URL: http://www.ietf.org/rfc/rfc3066.txt.

[7] S. Perreault, “vCard Format Specification. IETF RFC 6350,” August 2011. [Online]. Available: URL: http://tools.ietf.org/html/rfc6350.

[8] M. Nottingham and R. Sayre, “The Atom Syndication Format. IETF RFC 4287,” December 2005. [Online]. Available: URL: http://www.ietf.org/rfc/rfc4287.txt.

[9] International Organization for Standardization, “Representation of dates and times. ISO 8601:2004.,” 2004. [Online]. Available: URL: http://www.iso.org/iso/catalogue_detail?csnumber=40874.

[10] International Organization for Standardization, “Geographic information -- Metadata. ISO 19115:2003,” 2003. [Online]. Available: URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26020.

[11] International Organization for Standardization, “Information technology -- Database languages -- SQL multimedia and application packages -- Part 3: Spatial. ISO13249-3:2011,” 2011. [Online]. Available: URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?.

[12] S. Perreault, “xCard: vCard XML Representation. IETF RFC 6351,” August 2011. [Online].

Available: URL: http://tools.ietf.org/html/rfc6351.

[13] R. Bhat and P. Saint-Andre, “A JavaScript Object Notation (JSON) Representation for vCard. IETF I-D (Work in progress.),” June 2012. [Online]. Available: URL: http://tools.ietf.org/html/draft-bhat-vcarddav-json-00.

[14] H. Schulzrinne, “The tel URI for Telephone Numbers. RFC 3966,” December 2004. [Online]. Available: URL: http://tools.ietf.org/html/rfc3966.

[15] A. Olson, “Time zone code and data. <ftp://elsie.nci.nih.gov/pub/>,” [Online].

[16] E. Lear and P. Eggert, “IANA Procedures for Maintaining the Timezone Database. (Work in Progress.). I-D,” January 2011. [Online]. Available: URL: http://tools.ietf.org/html/draft-lear-iana-timezone-database-02.

[17] A. Mayrhofer and C. Spanring, “A Uniform Resource Identifier for Geographic Locations ('geo' URI). RFC 5870,” June 2010. [Online]. Available: URL: http://tools.ietf.org/html/rfc5870.

[18] International Telephone and Telegraph Consultative Committee, “Information Technology - Open Systems Interconnection - The Directory: Selected Attribute Types. CCITT Recommendation X.520,” November 1988. [Online].

[19] T. Celik, M. Mullenweg and E. Meyer, “XFN 1.1 profile,” [Online]. Available: URL: http://gmpg.org/xfn/11.

[20] P. Leach, M. Mealling and R. Salz, “A Universally Unique IDentifier (UUID) URN Namespace. Internet RFC 4122,” July 2005. [Online]. Available: URL: http://tools.ietf.org/html/rfc4122.

[21] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures (PhD),” University of California, Irvine, 2000.

[22] M. Kelly, “Hypertext Application Language: A lean hypermedia type (Work in Progress.),” October 2012. [Online]. Available: URL: http://stateless.co/hal_specification.html.

[23] J. Gregorio and B. de hOra, “The Atom Publishing Protocol. Internet RFC 5023,” October 2007. [Online]. Available: URL: http://www.ietf.org/rfc/rfc5023.txt.