Point-of-Interest Based Navigation System

Mauro Rafael Pereira Teixeira
Instituto Superior Técnico
mauro.teixeira@tecnico.ulisboa.pt

Abstract—In the recent years many tourism-oriented apps and websites have made the life of thousands of tourists easier by providing access to information about multiple “Points of Interest” around the world. However, there isn’t a standardized way to do so – the result is the existence of multiple disconnected initiatives, all using different data models and not offering many options for itinerary creation and management. This document proposes a data model and architecture for an application that allows aggregation of POI information sources and the definition of relationships. It allows users to search multiple sources at once and create their own POIs and relationships between these venues. This system addresses the problem of creating a centralized platform that can serve as basis for the creation of apps, implementable by any company or entity, public or private. This system aggregates information from multiple sources and allows content-creators to actively translate their knowledge into reusable content distributable to several independent platforms.

Index Terms—Point of Interest, venues, tourism, apps, standardization, e-tourism

I. INTRODUCTION

According to Emmanouilidis et al. [1], when talking about tourism-oriented applications (apps) (either for the web or for mobile devices) there is a basic system architecture that is usually common for most of them. This high-level basic architecture, represented in figure 1, maps the three essential players required to build a successful tourism app: the traveler, the tourism-industry professional and the developer.

![Figure 1 - Common High-Level Architecture for tourism services](image)

The realm of the traveler is, usually, mobile apps – nowadays they are a powerful tool that helps with travel planning for thousands of travelers around the world. But in order for these apps to be of any use, they must already contain usable information, and that’s where industry-professionals become important in the process of building a successful service. However, developers are the ones who gather all the requirements and build the service itself, so they become an important part of this process.

A. The traveler’s perspective

According to the Portuguese Statistics Institute, almost 50% of travelers visit other places for the purpose of having fun [2].

Thus, “What to do?” is often the question the tourist tries to answer before embarking on any journey.

Fortunately, nowadays there are a number of different sources where these travelers can gather information on the city and on the points of interest a city has to offer. On the Internet, for example, there are even some websites (for example tripadvisor.com) whose sole purpose is to rank these places according to their quality. In fact, thanks to the Internet and these websites, the “traveler’s process” has become more easy and less time consuming since, sometimes, there are even suggested itineraries the traveler may take.

These places are what in this work are referred as “Points of Interest”, or POIs. From a technical standpoint, a point of interest is a “location for which information is available” [3]. In practice, a point of interest can be anything from a zoo or an aquarium, to a museum, an archeological site or even a round-about or avenue. There are hundreds of websites, mobile apps and initiatives that offer information about POIs.

Even though these tools provide different information for points of interest, most of them (if not all) are not able to provide context on how these POIs relate with each other or with the region. No smartphone nowadays can replace a human in understanding how the POIs relate, and these tourism apps do not provide verifiable information about these relationships. What some of these platforms provide is, at most, “educated guesses” about what the tourist may be interested to see. Among the available research, there have been published works analyzing what places tourists usually visit [4] and papers defining algorithms to calculate these suggestions [5]. Often these tools take advantage of crowdsourcing methods to calculate suggestions of places the user may be interested in visiting based on profile preferences. The OnDroad Planner [6] is an example of such a tool: by using travelling oriented social networks (like Minube) and taking in the user’s preferences it is able to automatically generate itineraries. But because it focuses on user reviews and not on verifiable facts, there is no verifiable direct relationships between the suggested places to visit. Thus, even though all this information exists...
and despite the availability of these tools, there are still tourists who would rather prefer hiring a professional guiding service than use these technological solutions. Every trip should have a well-defined itinerary. A randomly-selected set of POIs is not an itinerary: this set should consider the relationships between each POI and the tourist’s interest. Thus, from a traveler’s perspective, it is fundamental that a tourism app serves as a guide by providing good itineraries, preferably made by experienced professionals who know the city’s history.

B. The Industry Professional’s perspective

Even though there are quite a few services that specialize in points of interest all over the world, there seems to be no services that centralize all this information in one format and that allows the definition of relationships between these points of interest in an easy-to-access way. The result is the existence of several websites with scattered and unstructured information, or with structured information in many different formats. Limited as they are, these sources don’t offer any information about how these POIs connect.

One example of a site that offers these itineraries is the website of the “Direcção-Geral do Património Cultural” (http://www.patrimonionacional.pt/), the Portuguese entity that manages cultural heritage: they offer a few itineraries, each one of those containing several points of interest related to the general theme of the itinerary (which can be, for example, “Historical Bridges in Alentejo”). Some of those itineraries offer detailed information about each POI, others end up not offering more than a single text description, so it’s not only hard to relate POIs in the same itinerary but also impossible to deduce a relationship between POIs from different itineraries. Thus, tourism-industry professionals plan their itineraries and trips using tools like Excel to organize information and Google Maps to search locations on a map. These tools are not tourism-specific and lack the necessary resources to allow an industry professional to properly create and manage an itinerary. It is also not possible for tourism-industry professionals and public entities to collaborate in the creation of itineraries and sharing of information: even though there are many specialists who’ve made itineraries and the proper research, there simply isn’t any platform that allows to do so in a centralized way, and since these professionals and entities are not IT specialists they don’t have the resources or the know-how to define and develop this infrastructure. By not allowing a crowdsourcing effort, the current situation is actually obstructing creativity and innovation [7].

The tourism-industry urgently needs a platform that allows professionals to aggregate POI information from multiple sources and enables the creation and management of relationships between them “locally” and in a private or collaborative way.

C. The developer’s perspective

The ever increasing use of information technologies has opened up opportunities for developers and companies to expand their business, launching websites and mobile apps that can do almost anything. But creating all these apps from scratch can become tiresome: the developer will be creating something that has been done hundreds of times before. In order to help their mission, developers create tools to help them do their work. Software development kits (SDKs) and application programming interfaces (APIs) are some of these tools that can really save them time and, hence, money when developing an app.

However, some apps have very specific needs like, for example, the “Museum of London: Streetmuseum” mobile app (http://www.museumoflondon.org.uk/Resources/app/you-are-here-app/home.html) which marks several points of interest in London and where each marker can be used to view an image of how the city used to look like in the past.

There are many others tourism apps of several kinds just as specific [8], and for these the developers have no choice but to start from scratch. There is a lot of wasted effort, even more so if we consider that the technical design for different apps is very similar (see figure 1). Also, by not using a common platform to manage information and allowing users to create their content, most of these apps can’t be considered part of the smart tourism concept that many tourism-related venues are turning towards [9].

Hence, there is a need to define a generic, open and expansible platform and data model that can be used by developers to fast-track their deployments when building tourism apps. This platform can be used and re-used for several scenarios and can serve as a base for most tourism applications.

D. Proposed Work

This document analyzes what tools and services are currently available that can support the creation of a tourism platform that meets the requirements of the three stakeholders. In order to define this platform, a few use cases for this tool are presented according to the problems addressed. These use cases can then be translated into requirements for the definition of the platform. The next step is translating these requirements into a data-model and an architecture that can support a platform for the collaborative content creation and management of POIs and its relationships. The architecture is defined in a way that it can serve as basis for most tourism applications available. To prove that the proposed architecture is implementable, web technologies were chosen and a proof-of-concept was developed in the form of a website for content creation. Being based on the concepts of Crowdsourcing [16] and Open Knowledge [17], it allows users all over the world to upload their knowledge about POIs and their relationships into the platform and share this information with other users.

II. RELATED WORK

In order to propose a viable solution, there is a need to study existing systems and technologies. This overview will make clear what tools can be used to achieve this goal (like APIs and data models) and where the existing platforms fail in providing an answer to the identified needs.

A. Existing Platforms

In order to address the needs of tourists, tourism-industry professionals and developers, several platforms have been
developed:

CitySDK (http://www.citysdk.eu/) is an initiative that self-defines as a “City-service development kit”, co-founded by the European Union, that provides tools for developers and cities themselves to be able to develop new “Smart City” scalable services [10]. It currently offers three APIs, being one of them a Tourism API that provides a centralized way of accessing information about Points of Interest that participating cities publish.

CitySDK’s primary goal is to offer APIs for developers, but a mobile application using the tourism platform’s resources was also created under the name of the platform [11].

Mosey (https://www.mosey.com/) is a platform that allows collaborative creation of ordered itineraries and the addition of notes for each POI inside an itinerary. Even though it also allows the creation of custom POIs inside an itinerary, these POIs are not made available for other users to re-use. All POI data seems to be extracted from Foursquare exclusively. Once published, an itinerary is always made public, with no privacy controls. It offers a web interface and an iOS mobile app.

VisitEurope (http://www.visiteurope.com/en/) is a European initiative with the aim to market Europe’s potential for tourism. The content in the platform is not editable and it doesn’t allow any kind of collaboration or re-use of content. The initiative offers a mobile app, but only for iOS devices.

Minube (http://www.minube.com/) is a tourism-oriented social network. Besides offering other capabilities like hotel and flight search, it is also a collaborative tool that allows users to create itineraries and make them available to the community. It feature privacy settings so that the user can choose whether to publish an itinerary or keep it private, as well as sharing options. However, it does not allow the addition of notes for individual POIs in an itinerary and it doesn’t allow the creation of custom POIs. It offers mobile apps for iOS and Android devices.

Like the previous platforms, other tourism apps offer similar functionalities but continue to be limited in what they can offer to the user or developer when it comes to the content management and information aggregation.

B. Data Models

Data Modeling is the basis of every application. In the case of a tourism application focused on points of interest, the basis should be the data models of these points of interest themselves.

The W3C (World Wide Web Consortium) launched, on September 13th 2010, the “Points of Interest Working Group” with a mission to “develop technical specifications for the representation of Points of Interest information on the Web” [12]. Their work was later absorbed by the OGC (Open Geospatial Consortium) which incorporated W3Cs work in their other working groups’ work.

They define a POI as having a label (name) and a description, and can be categorize according to categories that are not defined – W3C mentions that this category can be anything from the type of venue (museum, restaurant, etc.) to a popular rating or a security classification. The POI can also have a “time” that can represent a start date or an end date for that place, and a location that can be defined in many ways, from geographic coordinates to civic addresses or bounding boxes. There are other proprieties for the POI that are used for POI management, like an ID, date of creation, date of modification, date of deletion, author, URL, and others. Perhaps the most interesting one if the “Relationship” data entity, but a closer analysis shows this relationship is a spatial one: it’s only used to say whether a POI in overlapping with another target POI, or if the POIs are next to each other.

This W3C/OGC project seems to be one of the few that attempt to standardize a Data Model for representation of POIs. Even though it is a good basis, it also is, however, insufficient for purposes of this work due to the fact that relationships between POIs are not defined to satisfaction.

C. Data Sources

Most tourism-oriented apps usually have a map interface where points of interest are shown in certain places. One of the challenges to address is where to find information about the several Points of Interest scattered around the world – this information is, also, scattered in multiple sources. Some companies and initiatives tried to address this issue and created centralized sources of information. These sources of Information provide information either on a regional/local scale (like a city, or state) or in a global scale. A few of these services are summarized in table 1.

<table>
<thead>
<tr>
<th>Global Services</th>
<th>Regional Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps / Google Places</td>
<td>CitySDK</td>
</tr>
<tr>
<td>OpenStreetMap</td>
<td>Direcção Geral do Património Cultural</td>
</tr>
<tr>
<td>TripAdvisor</td>
<td></td>
</tr>
<tr>
<td>Foursquare</td>
<td></td>
</tr>
<tr>
<td>Facebook</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 - Examples of POI Information Sources

CitySDK’s Tourism API (http://www.citysdk.eu/) provides a centralized way of accessing information about Points of Interest that participating cities publish. It only features POI information from participating cities, so it will probably have more accurate information on the local points of interest than other tools.

Direcção Geral do Património Cultural (www.patrimoniocultural.pt/) is the public entity that manages cultural heritage in Portugal. In fact, when talking about places of general interest in the country, Portugal is a perfect example on how difficult it is to manage this kind of information. DGPC offers information about relevant Portuguese points of interest and compiles a few itineraries and groups that include these POIs. Also, DGPC manages other subpages where it is also possible to find more POI-content (like http://arqueologia.patrimoniocultural.pt/). However, this entity does not provide webservces to access this information: instead, another entity designated SIPA (“Sistema de Informação para o Património Arqueológico” - www.monumentos.pt/) manages these services using DGCP content [13]. In order to centralize this and other resources, yet another website named iGEO – Informação Geográfica was created to bring together
several available web services. Even though there has been as effort to centralize the information, there is still inconsistencies due to the fact that information is spread through multiple places, and a few occurrences exist when the same POI has different information in different public platforms.

While Google Maps offers a map service with geocoding capabilities, Google Places [14] is a service for retrieving Points of Interest Information [14]. According to Google, it currently stores over 100 million points of interest through the world. Google Places is, essentially, a web service developers can use to consume information about POIs.

OpenStreetMap is a platform developed by a community of volunteer mappers that contribute and maintain POI and map data. It is run by the community and follows the Open Data concept where data is available to everyone for free. Several other services, like Foursquare and MapQuest, run on top of OpenStreetMap map data. Being completely free, it depends on partner entities to support the necessary infrastructure and the API is still under development.

TripAdvisor is another popular site when it comes to points of interest. However, TripAdvisor only offers an API to selected partners and not to the public, not even for purposes of academic research [15]. Without an API, developers cannot retrieve information from the service and so it is excluded from further analysis.

Foursquare is a Venue-Oriented Social Network – users can login from their Mobile Devices and share their locations with other users and get suggestions of places to visit; in a way, it has become the app users use to obtain “bragging rights” about being in certain places. Foursquare constitutes a possibly good source of POI information.

Facebook, being the largest Social Network in the world right now [16], provides a very complete set of information about Points of Interest as well.

Some of these services also provide map services that the developer can use to embed a map in a page in order to show, in a graphical way, several locations. From the considered services, only Google Maps and OpenStreetMap provide a map service and Geocoding capabilities.

Another important factor in the choice of services to integrate is to look at its API’s usage limits (compared in table 2.4 according to information from each service’s API documentation). Since the goal of this document is to define and implement a platform that could serve as basis for most tourism apps, table 2 only compares usage limits for global services. TripAdvisor does not provide an API for external use, so is not considered in the analysis.

D. Web service APIs

An API is “a set of programming instructions and standards for accessing a Web-based software application or Web tool” [17]. An API works on the principle that the developer asks for something and it gives an answer – the programmer doesn’t have to concern himself about how the API got the answer; it just did. Hence, an API provides a layer of abstraction and allows the consumption of resources from information sources.

When it comes to tourism apps, the most commons types of APIs are:

a) Restful APIs [19], which is an architecture style for designing network applications where HTTP calls are made between machines

b) WFS [20], a standard created by the Open Geospatial Consortium (OGC) for creating, modifying and exchanging geographic information on the Internet.

1) Data Exchange Formats

APIs have one or more ways to communicate its results to the user: it can either be in a structured plaintext file, or an excel sheet, or even other popular formats like XML and JSON. In fact XML (eXtensible Markup Language) [21] has been around for almost twenty-years now (the first working draft was published in November 1996) and, for a while, software designers turned to XML as the most obvious answer to create interfaces and allow easy communication with other software tools.

There are at least two variations of XML that are used mostly with geographic data: GML and KML. GML (Geography Markup Language) [22] was defined by the OGC to express geographical features and is usually used for data exchange in Webservices (like WFS). KML (Keyhole Markup Language) [23] is format for data exchange used particularly to display geographic data in mapping applications like Google Earth. Unlike XML, it is usually provided as a downloadable file (usually compressed in a .KMZ file) and can be imported to a mapping application. XML-, JSON- and GML-based webservice generate content on-the-fly according to the user’s request and the information in the server, but KML/KMZ files are usually not editable outside a mapping application.

In the last few years JSON (JavaScript Object Notation) [24], has been gaining popularity in the programming community. Like XML, JSON has also been the subject of some variations. Specifically for exchanging geographical data, a group of developer specified and maintains the

<table>
<thead>
<tr>
<th>Table 2 - Usage Terms for considered APIs</th>
<th>Webservices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Google Maps</td>
</tr>
<tr>
<td>Info Requests</td>
<td>Free until 150.000 requests per 24 hours</td>
</tr>
<tr>
<td>Load Request</td>
<td>Free until 25.000 map loads per day</td>
</tr>
<tr>
<td>Geocoding Requests</td>
<td>Free until 2.500 requests per day and 10 requests per second</td>
</tr>
</tbody>
</table>


definition for GeoJSON [25], “a format for encoding a variety of geographic data structures” using the JSON format.

Table 3 shows that while Map and Geocoding services still use XML as an alternative to JSON, social network APIs have discontinued XML as a response format and are now using JSON exclusively.

<table>
<thead>
<tr>
<th>WebServices</th>
<th>Response Formats</th>
<th>XML</th>
<th>GML</th>
<th>KML/KMZ</th>
<th>JSON</th>
<th>GeoJSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>CitySDK Tourism API</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DGPC/SIPA API</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Foursquare API</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Google APIs</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Facebook’s Graph API</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Bing Maps APIs</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MapQuest APIs</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

E. Cloud

Cloud Computing [26] is a concept that is revolutionizing the way people and companies use IT resources. António Miguel Ferreira, in his book “Introdução ao Cloud Computing”, defines Cloud Computing as a “model that allows access […] to a series of shared computation resources […] that can be rapidly provisioned or released”.

Using Cloud services has many advantages but also presents some challenges. The main advantages of using a cloud service comes from the fact that users (or company) don’t have to concern themselves with buying, maintaining and upgrading the IT infrastructure required to run their applications – this requires a lot of initial investment before the application starts its operation: hardware costs, electricity costs and man-power costs in order to correctly configure the system all together represent thousands of euros of initial investment, and not all organizations have the funding for it. Another great advantage is the ability to request the provisioning of more or less resources depending on the current needs. For example, if a startup is running a website and schedules an online promotional sales campaign for the duration of a weekend, they can easily reinforce the resources of their infrastructure (in the case of IaaS – Infrastructure as a Service) just for the duration of that weekend. After that, they can easily downgrade the infrastructure to their usual needs. In the end, they only pay for what was used. A tourism application can both use and adopt this operation model: by using a PaaS (Platform as a Service) service to deploy the platform, a SaaS (Software as a Service) tourism service is offered to content-creators and content-consers.

F. Evaluation

Even though existing platforms do run in the cloud and some of them provide APIs developers can use, the data-models used by those solutions are incomplete, either by just focusing on the POI as a static concept or by not allowing the creation of more complex relationships between POIs. Also, these platforms usually have proprietary apps and do not provide ways to integrate with other apps. This lack of openness doesn’t benefit tourists, tourism-industry professionals nor developers.

III. DESIGN

The users of the platform described in this section are content creators, either amateurs or professionals of the tourism-industry. They will use this platforms with multiple goals in mind: either to plan a trip by defining itineraries, to search existing content in the platform or even to organize and create itineraries for commercial use. Whatever the reason, this platform must offer the necessary operations to support these activities.

The creation of an itinerary starts with searching the points of interest to be part of the itinerary, so the tool must support searches from a wide variety of POI information sources. If the POI isn’t listed, the user must be able to create the POI. These POIs can then be added to multiple itineraries and notes can be attached to each POI in a group. As explained further ahead, some of these POIs inside an itinerary may be closely associated with other POIs, so the user should also be able to create this relationships. This content may be private, public and shared with other users, and created POIs can also be reused later for other itineraries.

It is important that this content is stored in an open platform that also provides management features that allows users to manage (view, edit and delete) content already created.

A. Requirement Analysis

In order to support these operations, a set of requirements are defined:

1) A POI is defined by a Name, Latitude, Longitude, Address, Contact Information and a URL to the venue Website;
2) A POI should support the storage of Multiple Photos or reference to photos;
3) A POI should be able to store references for multiple POI Information Providers.
4) POIs can be added or removed from groups;
5) The same POI can be part of several groups;
6) It should be possible to write notes and attach support materials to each POI inside a group.
7) Relationships can be defined between two points of interest;
8) Notes and attachments are associated to a relationship supporting the connection.
9) Groups and Relationships can be either private or public;
10) Groups and Relationships can be shared between users when private;
11) Created POIs can be either private or public;
12) Allowing the connection to an external POI information provider must be as simple as adding a plugin to the application;
13) A webservice should be provided, allowing external entities to consume resources and execute functions.

B. Data Model

A data model is the result of connecting multiple data entities in a way that best translates the imposed requirements in a usable and implementable way, while Data Entities are implementable units of Information. By the end of the requirement analysis phase, one understands what is required to do from a functional perspective. Transforming these requirements into a data model represents the transformation of a functional specification into a technical one.

The following data entities represent one possible way of fulfilling the requirements 1 to 13:

1) POI: Basic structure for a Point of Interest;
2) POIPhotos: Represents a reference (URL) to a photo;
3) POIProvider: Represents a POI Information Provider (like Foursquare or Google Places);
4) POIInExternalProvider: References the ID of one POI with its External Providers. A POI can exist in multiple POI Information Providers, so this structure store a reference to a certain POI in an external provider;
5) POIGroup: The basic structure for a Group;
6) POIGroupTheme: Defines a Group theme;
7) POIInGroup: Entity responsible for attaching several notes and files to a certain POI in a certain group; also defines the order of the POI in the group;
8) POIRelationship: satisfies requirements 9 and 10 and defines a relationship between two POIs;
9) POIAttachment: Represents a file and stores its information;
10) POIRelatedData: Represents the entity that stores a note and an attachment;
11) POIOwner: Represents the Owner of an entity (POI or POIGroup)

Using the data entities defined in the previous section it’s possible to define a data model that supports the implementation of one or more applications making use of these concepts.

Following a more practical perspective, one possible implementation is to design a website that makes use of this data model. In order to do so, it’s necessary to define a data model implementable via code that can be translated to a database structure to allow the app to store information.

Thus, figure 2 represents a UML class diagram for code implementation. Notice the addition of another data entity named POIExternal. This entity is derived from the POI data entity and represents the same POI structure with the addition of another property (ExternalID); this is necessary because the ID of the POIs, when fetched from external sources like Foursquare, usually have a String as an ID instead of an integer as defined in the POI data entity.

C. App Architecture

Figure 3 shows the basic generic architecture for an application supporting the proposed data model.

A user runs a client side application which in turn connects to the application server, where the server-side application receives the client’s request and sends the appropriate information via some communication protocol like HTTP or even a custom protocol over TCP or UDP. In order to process the request, the server-side application logic implements the data model. Depending on the technologies used, this logic may or may not implement a presentation layer: for example, if the client is a browser and the server a webserver, than this logic will most likely include code for the generation of dynamic webpages.

The logic includes an interface for external communications, namely with the Database server and with External POI Information Providers. To do so, this interface references a component for the communication with the database server and a plug-in component to allow communication with External POI Providers.

In order to obey the requirement 14, the amount of external POI information providers the server communicates with depends on the installed plugins.

Also, the server would include a webservice component allowing other devices to obtain information and allowing the execution of functions by users, as asked in requirement 15. If nothing else, this webservice would allow the easy deployment of a mobile app for any mobile OS.

IV. IMPLEMENTATION

A. Used Technologies

The concept of SaaS (Software as a Service) used to define services provided over the web has been gaining popularity. Following this same idea, this prototype is designed so that anyone with a browser can access its functionalities in an easy-to-use and interactive way.

For this implementation, the following choices were made:

Webdevelopment Framework:

Being a website, the technologies used to build it must be appropriate for webdevelopment. ASP.NET was chosen to be the framework used for this website, with code-behind written in C#. For the integration with a map service and for added interactivity on the client-side, Javascript was also used.

Database Management System:

Since the chosen webdevelopment Framework was developed for Microsoft, the Database Management System used in the implementation of the Entity-Relationship Diagram is Microsoft SQL Server, for added compatibility with the Framework.

---

1 Requirements 14 and 15 affect the architecture of the app but not the base data model.
Figure 2 – UML Data Model
POI Information Providers:
The proof-of-concept is built so that the connection to POI Information Providers is managed by a plugin system – this way, a community of developers can build plugins for connection with many other POI Information Providers not considered in this prototype.

Map Services:
A search on the web reveals Google Maps to be the most popular map service used by many developers, thus community support for this service is easy to find. Hence, Google Maps is the map service implemented by using the Google Maps Javascript API to display an interactive map.

B. Implemented Architecture
The implementation choices made result in the following specifications:

1) Clients use a WebBrowser compatible with Javascript for accessing the application, so the communication protocol with the server is HTTP;
2) The server is a Webserver running the ASP.NET Framework (version 4.5);
3) The Application logic is programmed in C# and implements a communication interface comprised of two components:
   a) A Database interface, which specifies several queries for querying the database server and offers many different methods for content manipulation in the database; these are specific for the RDBMS used;
   b) A Plugin component for the integration with External POI Information Providers.
4) The Database server uses Microsoft SQL Server to manage databases;
5) The integrated External POI Information Providers are Foursquare and Google Places;

Since this is a proof-of-concept to demonstrate the use of a possible implementation of this data model, the WebService component of the architecture was not implemented. However, a possible API to export available data was defined.

In order to make it available to testers and the public, the system was deployed using a PaaS Cloud Service. Both the webservice running the ASP.NET framework and the Database Server were provided by the Microsoft Azure platform.

C. Available Features:

a) **Search POIs:** Users can search POIs by writing keywords that are matched against the app’s and the External Providers’ databases;

b) **Create new POIs:** new POIs can be created in any place, POIs can also be private or publicly shared;

c) **View POI details:** any user can view the details of an existing accessible POI, including connections to External POI Information Providers;

d) **Editing POI Details:** POI details can be changed either if it is a private POI or if no other user is using the POI in groups or relationships;

e) **Delete POIs:** POIs can be deleted if it is a private POI or if no other user is using the POI in groups or relationships;

f) **Merge POIs:** POIs from External Providers can be merged with existing application POIs;
g) **Map display options**: the user can select from which providers retrieve POIs, as a well has delete the information displayed on the map;

h) **Create new Groups**: the page offers a form for the easy registration of new groups;

i) **See list of groups**: shows a list of groups the users has ownership over;

j) **Modify Group Details**: a user can modify group details, including adding attachments to groups;

k) **Share groups with other users**: a user can share a group with other users;

l) **See/Modify Notes of POIs in groups**: The user can see the notes added for a POI in a group; if the user has ownership over the group, notes can also be modified;

m) **Create POI Relationships**: once a group is selected, a user can create relationships between two POIs (at least one of them must belong to the group); relationships are represented by lines;

n) **See/Modify Relationship details**: the user can see the notes and POIs that define the relationship. If the user has ownership over the group, modification is also possible.

o) **Delete Group**: An user can delete a group;

p) **Share groups with other users**: a user can share a group with other users;

q) **View POIs and Relationships in Groups**: when a group is selected, a list of POIs and relationships for that group is visible.

V. **Evaluation**

Previously a few tourism-oriented services were introduced that present some similarities to the tool implemented for this work. However, these tools tend to be very tourist-oriented and very tourism-driven since content is created by the tourists themselves. Thus, they don’t provide the necessary functionalities for tourism-industry professionals to create and manage their content.

Table 4 compares many features from the developed prototype with the capabilities of the platforms previously enumerated.

In fact, taking into account the capabilities of the prototype, it is possible to take, for example, one of the itineraries available at DGCP and create all the related content in the platform, thus centralizing it, making it easy to manage and allowing the addition of related content.

VI. **Conclusions**

This document started by describing what are the current issues felt by tourists, tourism-industry professionals and developers of tourism applications. The goal of this document was to produce a data model and a system architecture to support the development of a centralized platform that would address the described issues and serve as basis for future developments.

To approach this problem, a market analysis was necessary and feedback from tourism-industry professionals was received and incorporated in order to define the requirements of such application. Then, based on this requirements and the knowledge obtained on how these applications work, a data-model was defined and a generic architecture was also presented.

<table>
<thead>
<tr>
<th>Comparison Aspects</th>
<th>Other tourism services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prototype</td>
</tr>
<tr>
<td><strong>POI Edition</strong></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Itinerary Edition</strong></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Community driven</strong></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Aggregates information from many sources</strong></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Add notes to POIs in Itinerary</strong></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Privacy Management</strong></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Shareable Content</strong></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Reusable architecture</strong></td>
<td>✓</td>
</tr>
</tbody>
</table>

This architecture is flexible enough that it can either be used by a single company or entity to launch multiple applications that connect to the system through the web service to retrieve information, or used on a more global level to distribute information for hundreds of tourism-based applications in the world. Either way, the amount of time saved in the development phase of these applications is greatly increased.

The implemented application can be used both by tourism-industry professionals and tourists: tourism-industry professionals have the necessary tools to define itineraries, adding notes, attachments of their research, and, above all, keeping all this information private, while tourists can use this tool to search POIs from multiple sources of information (as many as can be implemented) and create their own groups, which they can actually share with the community in order to make it grow. Developed with the concepts of Crowdsourcing and Open Data in mind, the more people in the community, the more content will be available for everyone.

Thus, this work takes a good first step for the centralization of POI information and extending the concept of POI Groups and POI Relationships that so far lacked definition in current
market solutions. With a few modifications and a good business model, one can easily transform the prototype produced into a profitable service.

A. Future Work

Even though the goals proposed for this document were achieved, taking this solution to market would require further investigation in order to further develop and extend the concepts introduced in this document. Some of this investigation would fall on the following topics:

1) Can a business model be developed based on this data-model? Imagine tourism professionals are willing to sell their research (translated to POIs, groups and relationships on the platform) to other users. What about a user renting an itinerary to another user for a fixed amount of time? How can it be possible? Would a need to change the data model exist? What changes would need to be done? What if they are only willing to share this information for a limited amount of time: is it possible? How?

2) How about adopting a SaaS business model where users can use the application for free and access public information, but would require the payment of a monthly fee to create private itineraries and content? Is this viable from a business standpoint?

3) In this work an itinerary (representing a path from A to B) is represented by a group containing ordered POIs. Does it make sense to define a path in another away? How to define it?

4) How to allow the produced data to be integrated with other services? Via SOAP/ReST Webservices? Semantic Web?

REFERENCES


