Geocon: A Middleware for Location-aware Ubiquitous Applications

Loris Belcastro[1,(✉)⋆], Giulio Di Lieto[1], Marco Lackovic[2], Fabrizio Marozzo[1], and Paolo Trunfio[1]

[1] DIMES, University of Calabria, Italy,
[lbelcastro, fmarozzo, trunfio]@dimes.unical.it
[2] Helmes AS, Estonia,
marco.lackovic@helmes.ee

Abstract. A core functionality of any location-aware ubiquitous system is storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as users, places, events and other resources. The goal of this work is to design and provide the prototype of a service-oriented middleware, called Geocon, which can be used by mobile application developers to implement such functionality. In order to represent information about users, places, events and resources of mobile location-aware applications, Geocon defines a basic metadata model that can be extended to match most application requirements. The middleware includes a *geocon-service* for storing, searching and selecting metadata about users, resources, events and places of interest, and a *geocon-client* library that allows mobile applications to interact with the service through the invocation of local methods. The paper describes the metadata model and the components of the Geocon middleware. A prototype of Geocon is available at https://github.com/SCAlabUnical/Geocon.

1 Introduction

With the widespread diffusion of mobile technologies and location-based services, it is possible to provide ubiquitous access to context-aware information (e.g., interesting attractions or events in a given place being visited). A core functionality of any location-aware ubiquitous system is storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as (mobile) users, places, events and other resources (e.g., photos, media, comments). The goal of this work is to design and provide the prototype of a service-oriented middleware, called Geocon, which can be used by mobile application developers to implement such functionality. Geocon can be used to discover location-aware content, to share context-related information, and to facilitate interaction among users of mobile apps. Examples of services that can be

⋆ This work has been partially supported by Project PON04a2_D DICET-INMOTO-ORCHESTRA funded by MIUR.

implemented in a mobile app using Geocon are: *i*) discovery of cultural places to be visited during a trip; *ii*) publication of user reviews about hotels and restaurants; *iii*) sharing of real-time information about events, traffic, and so on.

A key benefit for developers using Geocon is the possibility to focus on the front-end functionality provided by their mobile application, without the need of implementing by scratch back-end components for data storing, indexing and searching, since they are provided by the middleware. In order to represent information about users, places, events and resources of mobile location-aware applications, Geocon defines a basic metadata model that can be extended to match most application requirements. The widely-used *JavaScript Object Notation* (JSON) format is employed to represent such metadata. The architecture of the middleware includes a *geocon-service* that exposes methods for storing, searching and selecting metadata about users, resources, events and places of interest, and a *geocon-client* library that allows mobile applications to interact with the service through the invocation of local methods. The interaction between service and client is based on the REST model.

Given the huge number of users, places, events and resources that may be involved in location-aware ubiquitous applications, scalability plays a fundamental role [8]. Geocon was designed to ensure scalability through the use of a NoSQL indexing and search engine, Elasticsearch, that can scale horizontally on a very large number of nodes as the system load increases. Elasticsearch is used in combination with an external NoSQL database, MongoDB, which is more focused on constraints, correctness and robustness. Data stored in MongoDB can be asynchronously pushed to Elasticsearch, making it possible to persist in Elasticsearch a subset of the data stored in the external database.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the metadata model. Section 4 describes the middleware architecture and components. Finally, Section 5 concludes the paper.

2 Related Work

The European research project CRUMPET [5] (*Creation of User-Friendly Mobile Services Personalised for Tourism*) was developed in the early 2000s, before the mass diffusion of smartphones, for addressing issues related to the mobility of tourists. Taking into account different user's travel purposes (e.g. business, leisure, entertainment, education), CRUMPET aims to provide information services meeting the different tourists' needs. It exploits information about users' personal interests and their geographical position to filter the content available to them. One of the primary goals of CRUMPET was to implement and improve FIPA[3](*Foundation for Intelligent Physical Agents*) specifications for mobile applications. The project used the explicit/implicit feedback concepts and the GML (*Geography Markup Language*) standard for storing geographic data.

[3] http://www.fipa.org

Yu and Chang[10] extended the CRUMPET project ideas by seeking new intelligent solutions for overcoming the limitations of handheld devices in terms of reduced screen size for displaying information and limited bandwidth for transmitting data over a mobile network. Schmidt-Belz and Poslad[7] presented another study connected to the CRUMPET project, which aims to assess the quality of CRUMPET usability from the end user's point of view. The study takes into account four European locations (i.e., Heidelberg, Helsinki, London and Aveiro) and makes use of the standard questionnaire SUMI[4] (*Software Usability Measurement Inventory - ISO/IEC 9126*) to evaluate quality of software usability. This questionnaire was replaced in 2011 by SQuaRE[5](*Systems and software quality Requirements and evaluation - ISO/IEC 25010*).

Some works have been devoted to the development of context aware-mobile applications that use context to provide information and/or services relevant to the user, in which the relevance depends on the user's intentions [1]. An example is COMPASS [9] (*Context-aware Mobile Personal Assistant*), which provides services and information based on user's interests and position. For selecting relevant services, COMPASS uses two types of criteria: *i*) *strict criteria*, which are used for discarding irrelevant results; and *ii*) *soft criteria*, for sorting results and assigning a relevance score to each service/information. The application is based on the WASP platform [4] that provides general support services, such as context manager and indexing services. WASP can be integrated with other services and can be easily applied to other domains, such as taxi reservation or dwelling search.

3 Metadata Model

We defined a metadata model for representing information about users, places, events and resources of mobile location-aware applications. The model identifies a number of categories for indexing items in the domain of interest, which are generic enough to satisfy most of the application contexts. In particular, the metadata model is divided into four categories:

- *User*: defines basic information about a user (e.g., name, surname, e-mail).
- *Place*: describes a place of interest (e.g., square, restaurant, airport), including its geographical coordinates.
- *Event*: describes an event (e.g., concert, exhibition, conference), with information about time and location.
- *Resource*: defines a resource (e.g., photo, video, web site, web service) associated to a given place and/or event, including its Uniform Resource Identifier (URI).

Tables 1-4 present the basic metadata fields for each of the four categories listed above. Metadata are meant to be extensible, i.e., it is possible to include

[4] http://sumi.ucc.ie/
[5] http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733

additional fields based on the specific application. For example, the user schema may be extended to include birth date, city, linked social network accounts, and so on.

Table 1. Basic User metadata.

Name	Type	Description
id	String	Unique user identifier
name	String	Given name
surname	String	Family name
email	String	E-mail
token	String	Authentication token

Table 2. Basic Place metadata.

Name	Type	Description
id	String	Unique place identifier
name	String	Name of the place
description	String	Textual description of the place
latitude	Real	Latitude of the place
longitude	Real	Longitude of the place
address	String	Full address of the place
user_id	String	Id of the user who created the place

Table 3. Basic Event metadata.

Name	Type	Description
id	String	Unique event identifier
name	String	Name of the event
description	String	Textual description of the event
start_date	String	Date and time when the event begins
end_date	String	Date and time when the event ends
place_id	String	Id of the place where the event is held
user_id	String	Id of the user who created the event

To represent metadata, the *JavaScript Object Notation* (JSON) is used. JSON is a widely-used text format for the serialization of structured data that is derived from the object literals of JavaScript [2]. Fig. 1 shows an example of JSON metadata describing a User. Beyond the basic metadata (id, name, etc.), it includes some additional fields (city, linked accounts and food preferences).

Fig. 2 shows an example of Place metadata, regarding the "Kabuki" restaurant in Washington, DC, USA, which is tagged as a Japanese and sushi specialties restaurant using an additional "tags" field.

Table 4. Basic Resource metadata.

Name	Type	Description
id	String	Unique resource identifier
name	String	Name of the resource
description	String	Textual description of the resource
URI	String	Link to the resource
place_id	String	Id of the place to which the resource is associated
event_id	String	Id of the event to which the resource is associated
user_id	String	Id of the user who created the resource

```
{
"id": "jdoe",
"name": "John",
"surname": "Doe",
"email": "john.doe@example.com",
"token": "19800308",
"city": "New York, NY, USA",
"linked-accounts": [
{"name":"facebook", "token":"424911363"},
{"name":"google", "key":"23467223454"}
],
"food-preferences": ["sushi", "pizza"],
"date-created": "2016-03-27T08:05:43.511Z"
}
```

Fig. 1. Example of User metadata in JSON.

```
{
  "id": "534",
  "name": "Kabuki",
  "description": "Japanese Restaurant",
  "latidude": "38.897683",
  "longitude": "-77.006081",
  "address": "Union Station 50, Washington, DC, USA",
  "user_id": "jdoe",
  "tags": ["Japanese", "sushi"]
}
```

Fig. 2. Example of Place metadata in JSON.

4 Middleware

Fig. 3 describes the architecture of the middleware, which includes two main components:

– *geocon-service*, which contains a central registry for indexing users, resources, events and places of interest; it exposes methods for storing, searching and selecting metadata about these entities.

– *geocon-client* is a client-side library that allows mobile applications to interact with *geocon-service* through the invocation of local methods.

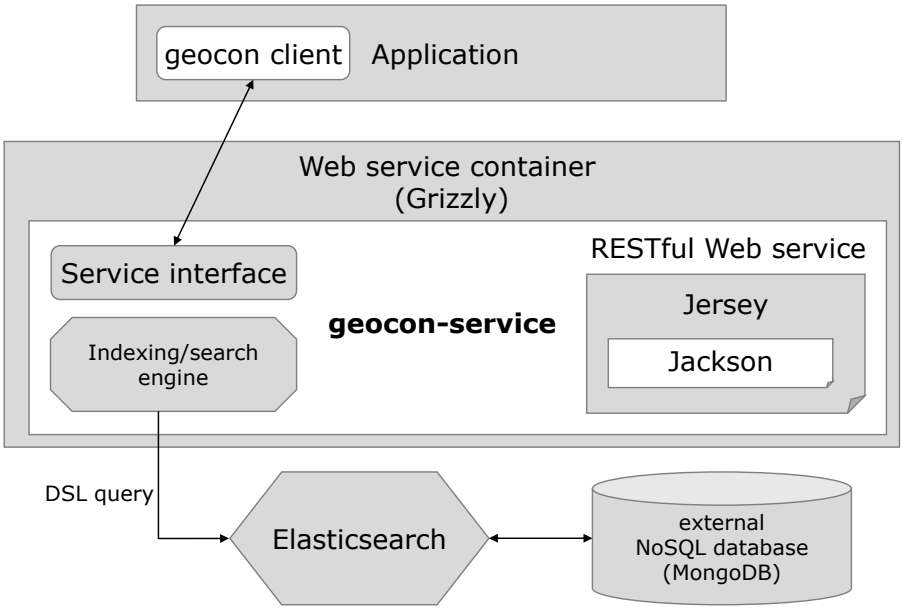

Fig. 3. Architecture of the middleware.

The interaction between service and client is based on the REST model [6]. To this end, a complete support to *CRUD* (Create, Read, Update, and Delete) operations on the metadata has been defined through Java APIs.

4.1 Geocon-service

The geocon-service has been implemented as a RESTful Web service and exposed via the Web service container Grizzly [6], which was deployed on the Microsoft Azure platform that ensures scalability, reliability, and access to external data analytics [3].

The framework used in our implementation to develop RESTful Web services is *Jersey*[7], an open source framework that implements JAX-RS (Java API for RESTful Web Services) using annotations to map a Java class to a Web resource, and natively supports JSON representations through the integrated library *Jackson*[8].

[6] https://grizzly.java.net
[7] Jersey: `http://jersey.java.net/`
[8] Jackson: `http://jackson.codehaus.org/`

The core component of *geocon-service* is the indexing and search engine, which has been implemented using Elasticsearch[9]. Elasticsearch is an open-source, distributed, scalable, and highly available search server based on Apache Lucene[10], and provides a RESTful web interface. Elasticsearch has been chosen because of several benefits, including:

- it is document-oriented, which means that entities can be structured as JSON documents;
- it is schema-free, which means it is able to detect the data structure automatically without need to specify a schema before indexing documents;
- it is horizontally scalable: if more power is needed, other nodes can be added and Elasticsearch will reconfigure itself automatically;
- it has APIs for several programming languages, including Java, which makes it easily integrable with other systems.

Geocon-service uses the query language provided by Elasticsearch, which is a full Query DSL (*Domain Specific Language*) based on JSON. Therefore, queries can be defined through the following main commands:

- *term*: returns all the documents whose specified field contains a given term. The following example returns all the documents whose field *name* contains the word "Mary":

```
{"term" : { "name" : "Mary" }}
```

- *prefix*: returns all the documents whose specified field contains a term beginning with a given prefix. The following example returns all the documents whose field *surname* begins with "Ro":

```
{"prefix" : { "surname" : "Ro" }}
```

- *bool*: returns all the documents containing a boolean combination of queries. It is built using one or more boolean clauses (i.e., *must*, *must_not*, *should*, and the parameter *minimum_should_match* that is the minimum number of clauses to be met). The following example returns all the users whose *name* is "Mary", that are not between 10 and 20 years old, and that like eating sushi or pizza:

```
{"bool" : {
    "must" : { "term" : { "name" : "Mary" } },
    "must_not" : {
      "range" : {"age" : { "from" : 10, "to" : 20 }}
    },
    "should" : [
      { "term" : { "food-preferences" : "sushi" } },
      { "term" : { "food-preferences" : "pizza" } }
    ],
    "minimum_should_match" : 1
  }}
```

[9] https://www.elastic.co/
[10] https://lucene.apache.org/

Due to some limitations of Elasticsearch (e.g., absence of transaction support, possible loss of write operation during cluster reforming/splitting), we use it in combination with an external NoSQL database, MongoDB[11], which is more focused on constraints, correctness and robustness. Compared to relational databases, MongoDB provides several benefits in terms of simplicity, flexibility, and scalability. Data stored in MongoDB can be asynchronously pushed to Elasticsearch. In such way, it is possible to persist in Elasticsearch a subset of the data stored in the external database, possibly using a different data format.

4.2 Geocon-client

Geocon-client is the library used by mobile applications to interact with *geocon-service*. The library aims to facilitate communication with the *geocon-service* methods, hiding some low-level details (e.g., authentication, REST invocation, etc.) and providing users with a complete set of functions for executing CRUD operations. These functions are implemented using a set of objects and methods provided by the client library to the application layer.

Geocon-client consists of five classes: four classes are used to represent the metadata categories (*User*, *Place*, *Event* and *Resource*), while a fifth class (*SearchEngine*) is used to expose the methods for storing and searching data on *geocon-service*. For each class representing a metadata category, the *SearchEngine* class provides a set of CRUD methods: *register*, *get*, *update*, and *delete*. As an example, Table 5 shows the CRUD methods provided to register, get, update and delete Resource elements in the service.

Table 5. CRUD methods for Resource elements.

Method	Description
`register (Resource r)`	Registers a resource to the service
`get (Resource r)`	Returns the metadata of a resource
`update (Resource r)`	Updates the metadata of a resource
`delete (Resource r)`	Deletes a resource

5 Conclusions

Geocon is a service-oriented middleware designed to help mobile developers to implement location-aware ubiquitous applications. In particular, Geocon provides a service and a client library for storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as (mobile) users, places, events and other resources (e.g., photos, media, comments). A key benefit for developers using Geocon is the possibility to focus

[11] https://www.mongodb.com

on the front-end functionality provided by their mobile application, without the need of implementing by scratch back-end components for data management, which are provided by the middleware.

Geocon defines a basic metadata model to represent information about users, places, events and resources of mobile location-aware applications, which can be easily extended to match most application requirements. In order to ensure a high level of decoupling and efficient communication between client and service, the REST model has been adopted. Moreover, given the huge number of users, places, events and resources that may be involved in location-aware ubiquitous applications, Geocon uses the Elasticsearch engine that can scale horizontally on a very large number of nodes. A prototype implementation of Geocon is available at `https://github.com/SCAlabUnical/Geocon`.

References

1. Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, UK, 1999. Springer-Verlag.
2. ECMA. Ecma-262: ECMAscript Language Specification. Fifth edition. *ECMA (European Association for Standardizing Information and Communication Systems)*, 2009.
3. Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. A cloud framework for big data analytics workflows on azure. *Advances in Parallel Computing*, 23:182–191, 2013.
4. Sylvain Martin and Guy Leduc. An active platform as middleware for services and communities discovery. In *International Conference on Computational Science*, pages 237–245. Springer, 2005.
5. Stefan Poslad, Heimo Laamanen, Rainer Malaka, Achim Nick, Phil Buckle, and Alexander Zipl. CRUMPET: creation of user-friendly mobile services personalised for tourism. In *3G Mobile Communication Technologies, 2001. Second International Conference on (Conf. Publ. No. 477)*, pages 28–32, 2001.
6. Leonard Richardson and Sam Ruby. *RESTful web services*. O'Reilly Media, Inc., 2008.
7. Barbara Schmidt-Belz and Stefan Poslad. User validation of a mobile tourism service. In *Proc. Workshop on HCI in mobile guides*, pages 57–62. University of Udine, 2003.
8. Domenico Talia, Paolo Trunfio, and Fabrizio Marozzo. *Data Analysis in the Cloud*. Elsevier, October 2015.
9. Mark Van Setten, Stanislav Pokraev, and Johan Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 235–244. Springer, 2004.
10. Chien-Chih Yu and Hsiao-Ping Chang. Personalized location-based recommendation services for tour planning in mobile tourism applications. In *International Conference on Electronic Commerce and Web Technologies*, pages 38–49. Springer, 2009.