2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing

A Mobile Application Framework For Rapid Integration of Ubiquitous Web Services

Meng-Yen Hsieh[1] Hua-Yi Lin[2] Ching-Hung Yeh[3] Kuan-Ching Li[1] Bo-Shiung Wu[1]
[1]Dept. of Computer Science and Information Engineering, Providence University, Taichung, Taiwan
mengyen@pu.edu.tw, kuancli@pu.edu.tw, andy200783@gmail.com

[2]Dept. of Information Management, China University of Technology, Taipei, Taiwan
calvan.lin@cute.edu.tw

[3]Department of Computer Science and Information Engineering, Far East University, Tainan, Taiwan
chyeh@cc.feu.edu.tw

Abstract — **This paper describes a mobile application framework with multimodal dialogues to integrate ubiquitous web services. We implement a tourism widget based on Android platform with dynamic HTTP-based REST service on Internet by following the proposed framework. Generally, to obtain linked data or resource from external HTTP-based service could be as simple as the establishment of remote services with an internal data repository supervised by producers. We discuss the complexity and usability of integration between the widget and internal or external web services in the framework. The widget provides not only the tour-based inquiry services by web services but also the maintenance of individual tour schemes with a number of dialogue interfaces. Three categories of components are constructed to process user inquiry, then to obtain responses through the usability of web services or from local backup dataset. As experience results, data exchange with different formats between the widget and server-side web services are emulated.**

Keywords; ubiquitous; HTTP; REST; mobile

I. Introduction

Over the recent few years, web services in SOA (Service-Oriented Architecture) and ROA (Resource-Oriented Architecture) have been applied to mobile, ubiquitous, or Internet-of-Things applications. The techniques integrating with wireless communication such as WiFi, NFC, WiMAX or 3G are presented in various mobile applications. Most mobile applications need to establish server-side database to handle a great deal of individual data, business data, or application structure information. Client-side applications such as mobile widget must develop HTTP connection components to request server-side data using web services.

However, to construct a server-side data repository with rich information is a complicated procedure where data must be gathered from several and diverse ways, such as data collection on Internet, predetermined rules with business data, or user input data. Generally, multimodal dialogue components [1] are designed between client-side applications and backend-side services for data query. The mashup mechanisms and linked data sharing in web applications become more active, so that mobile applications can clearly integrate with web services or capture their resource through open APIs or REST-based URIs. In this study, it is developed a framework easy to be adopted in mobile applications, by including only a few components to combine with external web services on Internet.

Due to the limitations of mobile device and wireless bandwidth in the past, developing mobile applications always needs the support of customized backend services to gain data in database or from Internet. Developers must spend the cost of designing and implementing mobile client and custom web services. Currently, identifying service or resource through different methods of web services such as REST or SOAP is easy to be applied in developing mobile or ubiquitous applications. Recent advances in smart phones or devices combined with large number of released public web services on Internet could offer developers to implement applications without backend servers [2].

A number of alternative formats for data exchange between mobile client and web services are applied such as XML or JSON. When transferring the same data, XML format is always used with the total amount of bytes higher than other transfer methods. Moreover, data with complicated structures are organized into a XML document easy to be understood by humans. The experience in our measurement reports the transmission performance corresponding to the different data format, while our mobile client requests REST-based web services. Most of mobile applications always connect to Web services over wireless connections.

However, the focus of this paper is to provide a mobile framework supporting internal or external web services for improving the usability of mobile applications. The applications based on our framework can be carried out, adapted to unstable wireless connections.

An evaluation of the internal REST-based web service of Taiwanese tourism backend server has been implemented in order to integrate with the complex external web services and to determine how to enable efficient access with different format of data exchange. We developed interactive semantic mediators for requesting external web service as particular HTTP-based components in mobile or ubiquitous clients, that also could reduce the development complexity and implementation cost. In this study, our proposed mobile framework is able to be carried out with or without backend servers. An individual repository in mobile client is required to resist those disabled HTTP connections.

The paper is organized as follows: we describe literatures related to our contribution in Section 2. Section 3 introduces the application framework, followed by the detailed application scenario for managing a travelling schedule with few tour days. The application prototype and experiment results of analyzing web services are presented and discussed in Section 4, and finally, Section 5 concludes this paper.

II. Related Work

A number of mobile application frameworks [1, 3-5] have been developed for various purposes, while web services using standard protocols [6-10] can provide outright distributed resource over wireless networks and Internet. This section only introduces parts of the literatures related to the goals of our study.

Daniel *et al.* [1] described a multimodal mobile application framework supporting HTTP-based REST services. The framework consists of a three-tier querying architecture, distributed into two practical parts, iPhone or Car Dashboard, and GAPI or Own HTTP/Rest API. The first layer is the application tier where mobile users submit a keyword to query data through a dialogue manager GUI. The query with a keyword is translated to be a URL together with a particular parameter. The translation is implemented as a query builder in the Query Model/Semantic Search layer. Meta Web Services in the third layer provide remote resource to respond to the URL. Two custom HTTP/REST meta web services, Google Maps Local Search and GeoNames [11] are used in the experience results. The mobile application built according to the two previous layers is implemented as a DHTML-based rich internet application. Authors do not take care of the problems introduced by unstable wireless networks, as also the user-friendly difficulty of using browsers in mobile devices.

The mobile framework proposed by [3] tries to help the development over different mobile OS platforms such as iOS, Android, and BlackBerry. However, this kind of framework is hard to be used in creating gorgeous interfaces always containing multiple dialogues between user and application. In addition, the framework often has incomplete libraries and the lack of the APIs connecting to external services.

F. AlShahwan *et al.* [4] described how to use adaptive mobile web service to aid the resource-limited mobile devices hosting complex web services. The Mobile Host Web Service framework, denoted as MHWF, is developed for building, providing, and executing distributed mobile web services. Offloading modules in MHWF facilitate the provision of adaptive services by distributing the execution tasks of web services over multiple hosts. The authors also compare the REST-based with the SOAP-based MHWF modules. As experience results, the SOAP modules always have higher number of overheads more than the REST modules in terms of memory, offloading process, and total response time.

At present, to distribute resources on Internet is tending to ubiquitous evolution through the development of SOAP and REST web services [6]. The REST and SOAP standard protocols completely support HTTP connections with simple data formats such as XML and JSON.

S. Hahmann *et al*. [10] evaluated the features between linked data and database with multiple representations. Each of them has its own advantages. For example, the dataset in database is persistent and consistent, supervised by producers. Oppositely, the contents of a linked dataset cannot be guaranteed by web-linked connections. However, the linked data is better than the database in terms of data representation, ontological description, ontology matching, and data distribution, among others. Authors also compare the dataset in GeoNames with those in LinkedGeoData. GeoNames is a community-driven linked data, including place names and points of interest, which quality of data is better than those in LinkedGeoData. For example, the total number of residential area features in LinkedGeoData is by 12.5% less than in GeoNames dataset. In addition, the populated places in GeoNames are set to points with the integration support of REST web services, that are appropriate to our proposed mobile framework.

III. System Framework

The Figure 1 depicts the system architecture with two layers, mobile client and remote service or resource. The mobile client includes the three main categories of components and an individual repository to provide users with interactive mobile applications in smart phones. Remote data on Internet responding to the requests of mobile client is extracted from external or internal web services.

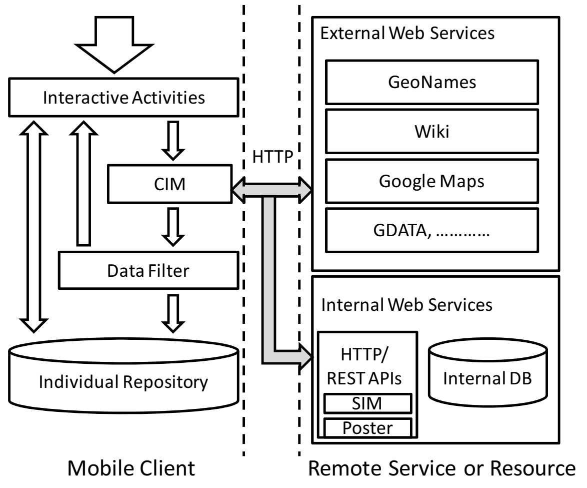


Figure 1. A mobile application framework. (CIM: Client Interactive Mediator; SIM: Server Interactive Mediator)

A. *Mobile Client*

The components in the mobile client are to interact with users to achieve the expected activities. The first component denoted as interactive activity not only provides dialogue interfaces to users, but also searches the required data for user queries. In case no data is found in the individual repository, the activity transfers user query to an interactive mediator as the second category of components. Interactive mediator in a mobile client a.k.a. CIM with limit resource is not adequate to be developed to simultaneously connect various styles web services. Hence, CIM supports the APIs of HTTP connection, accommodates the REST-based web services replacing other methods such as SOAP or RPC, and queries appropriate web services on Internet based on the predetermined URIs corresponding to user queries. The CIM component in our mobile client implemented in this study will aggregate a number of HTTP/REST-based remote service or resource suitable for tourism. Any response data that CIM received from web services will be forwarded to the last component, denoted as DataFilter, instead of immediately being returned to interactive activities that claimed the request.

The DataFilter is implemented with two objectives, data refinement and data transformation. The refinement function filters the response data of CIM to capture the meaningful information corresponding to user queries. Moreover, the information is converted with the steady formats required in the repository. For example, the JSON-style information will be transformed into XML files or datasheet in database.

Consequently, if the contents in some remote resources or services have almost no change, such as city names in a country, activities will search the individual repository first, instead of immediately sending user queries to CIM components. A number of query rules defined in interactive activities determines what user queries should be sent to CIM immediately, or to search data in the repository in advance. The repository must support only few storage formats to reduce the complexity on the design of DataFilter.

B. *Internal Remote Service or Resource*

Web services on Internet always supply various sharing purposes for composition processes such as file sharing, social communication, file management, allowable extensions of open systems built by users, and others. To reduce the composition complexity and to focus on the development of mobile framework, we only suggests that the REST-based web services supplying XML-based or JSON-based data suitable for application tasks are adapted to the development of the mobile framework. The framework through those particular data APIs can search for external resources, retrieve standard feeds, and notice the related contents. For the above reasons, we must concern on the type of tour and geographic data is important to our widget in the framework.

Meta web services could be customized by internal developers to communicate with the external web services. However, the software developers must grasp and reach server-side programming skills such as Servlet, PHP, or PERL, and establish the environment of backend servers such web and database servers. Besides, backend servers are always developed with abounding MVC-based server frameworks and libraries. However, the custom internal web services can easily observe the communication protocols required by mobile framework. This study provides two kinds of components, server-side interactive mediator and data poster in our custom web services.

Server-side interactive mediator a.k.a. SIM has three possible functionality requirements. The basic functionality is to collect data from an appointed database, and to reply to HTTP requests from CIM in mobile client. External web services could release free APIs to publish particular integration mechanisms for mobile clients, such as open authentication and data exchange protocols. To adapt similar APIs of external web services to SIM based on the requirement of mobile client will reduce the complexity of developing internal web services, and improve the integration performance. The integration mechanism is the second functionality. The final functionality is to parse the content of the external web pages, to retrieve the demand data, and to restore it in the internal database for reuse. Nevertheless, developers need to figure out that the schema of the web pages as HTML or XML files what SIM ties to integrate should have infrequent or no change in the future. The SIM is embedded in the REST-based APIs of internal web services to respond to the HTTP requests from mobile client.

In addition, most users always wish their revealable information distributed to social-based web services on Internet. The internal web service could include Post components to publish open individual data from mobile clients to external web services. To develop Poster components in SIM has the integration effects better than those in CIM, since the developers must figure out the ways of communicate to multiple web servers. In addition, various authentication protocols could be implemented in the Poster.

C. *External Remote Service or Resource*

Mobile client communicates with simple REST-based web services appropriate for the unstable wireless connections and the low communication overheads. To use completely different protocols in requesting external web services will cause development problems in CIM.

The Google data APIs denoted as GDATA [12] release a simple, standard protocol for reading and writing remote data in XML-based syndication formats on the Web. The CIM as a composition module can interact to the most Google web services with minimal changes, since the Google releases an atom publishing scheme uniform to the data format and transmission protocols of GDATA. Nowadays, Google data modules include a number of access services of Google Base, Blogger, Google Calendar, Google Spreadsheets, Picasa Web Albums, Google Maps, YouTube, etc. CIM can obtain the required remote service or resource from GDATA only by following the single Google's extension schemes in term of data model, query, optimistic concurrency, and security and authentication. AuthSub and ClientLogin are Google's proprietary authorization APIs, available as an alternative to OAuth to most Google APIs. For this reason, CIM will successfully access GDATA services after gaining and using a valid Auth Token from the Google authentication service.

GeoNames aggregating over hundred different data sources is a community-driven dataset that contains over ten million geographic place names and points classified into a large amount of unique features. Beyond names of the places in the dataset, each of data entities could provide its latitude, longitude, elevation, population, administrative subdivision and postal codes, hence GeoNames services are fitting in tour applications.

Besides, the GeoNames's dataset is an official public resource where people can improve the data quality by adding or correcting names, move existing features, and new features through wiki interfaces. In the overview of GeoNames web services, each of service entries identified by a stable URL links to web resources including these features. All the web services can respond the corresponding data with XML or JSON format, while any application with HTTP connection requests the GeoNames about an identified URL with few parameters. Therefore, the CIM only implements the usability of HTTP GET methods for GeoNames to capture the geographic features related to user's queries in interactive activities of mobile client. The authentication scheme for the usage of GeoNames's Web services is to pass a parameter "username" to each URL query. In addition, the username's account was registered in advance in GeoNames. The following four web services in GeoNames are adopted in our tour widget, as depicted in Fig. 2.

1) children: The service returns admin divisions and populated places for the geoname id given in a URL query. For example, the toponym and its corresponding data of Taiwan will be returned for Taiwan geonameid.

2) findNearby: Give a query with a lat/lng pair with a radius parameter, the service returns the closest toponym in the calculated range.

3) findNearByWeather: The service replying to an query returns the weather station with the most recent weather closest to a location based on the given lat/lng pair or ICAO (International Civil Aviation Organization) code.

4) search: the service can return all attributes of a place for an query string with a point data such as place name, country name, continent, and others.

5) get: the service returns the attribute of a GeoNames feature based on a predetermined geonameid in the query.

6) wikipediaSearch: the service returns the wikipedia entries found for a search keyword.

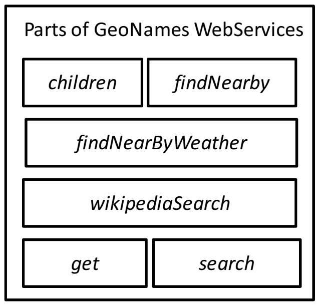


Figure 2. Parts of GeoNames's Web services used in the implementation of the mobile framework.

IV. Application Scenario

In our prototype, the interactive activities perform multiple dialogues. The widget is designed as a Android application in smart phones. Users can insert, edit, and delete travel schedules in the mobile client. To plan a travel schedule, user must determine a number of destinations and tours in the duration of few days. For example, Figure 3 shows the three travel schedules with the different number of tour days in Taiwan. We will introduce a tour scenario defined by a user to describe the operations of the widget based on the mobile framework. In addition, our internal web services will replace the external web services, while the stability and availability of the external services is not always trusted due to wireless networks.

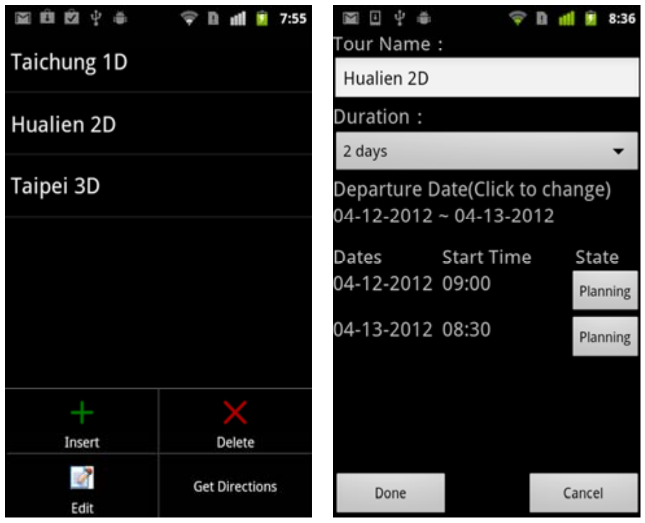
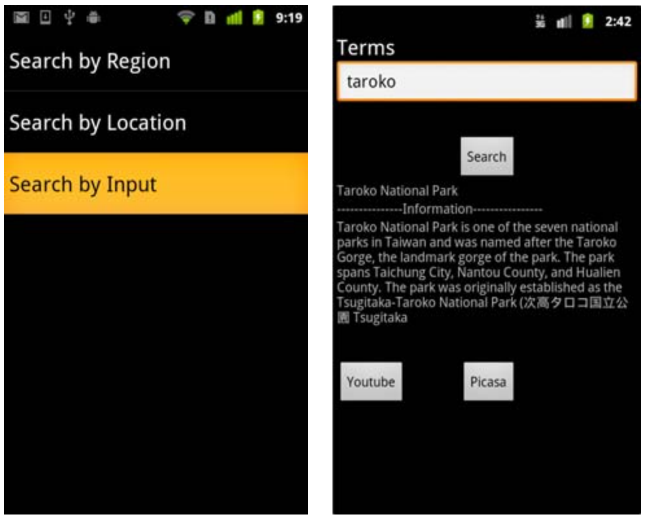


Figure 3. GUIs to maintain multiple individual travel schedule



Terms

taroko

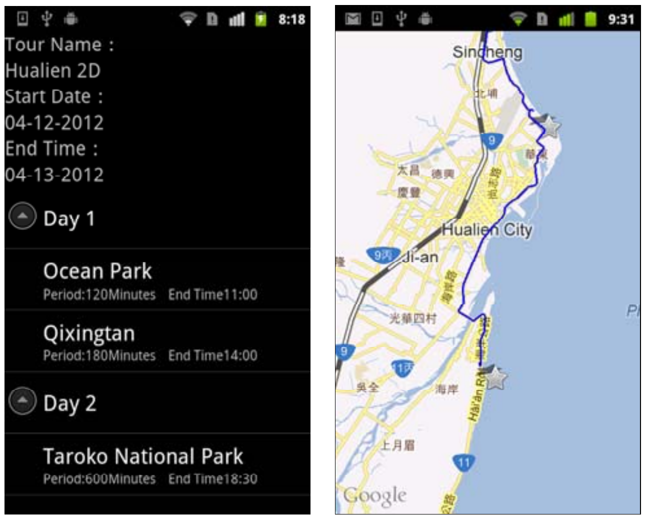
Search

Taroko National Park

---------------Information---------------

Taroko National Park is one of the seven national parks in Taiwan and was named after the Taroko Gorge, the landmark gorge of the park. The park spans Taichung City, Nantou County, and Hualien County. The park was originally established as the Tsugitaka-Taroko National Park (次高タロコ国立公園 Tsugitaka

Figure 4. GUIs to add a site in a tour through the searching methods.



Tour Name :
Hualien 2D
Start Date :
04-12-2012
End Time :
04-13-2012

Day 1

Ocean Park
Period:120Minutes End Time11:00

Qixingtan
Period:180Minutes End Time14:00

Day 2

Taroko National Park
Period:600Minutes End Time18:30

Figure 5. GUIs to display the details of one tour.

A. One of tour scenarios with user dialogues

The tour, called "Hualien 2D", is a plan with two days and one night. Additionally, the departure dates as also the start time for each are also assigned by users. Based on the scenario defined, the sequences of the tour on user interaction using the widget are as follows:

1) *Insert/Edit:* when pressing the Insert or Editor button, user inputs basic information for the tour, including the tour name, the duration, the start date, and every day's starting time in the range defined. Figure 3 describes the interfaces.

2) *Planning:* when pressing each planning button for one tour dates, user can add or delete a site within the date. The figure is not shown due to the page number limitation.

3) *Search:* once user starts to add a site, three alternative search schemes can be selected for seaching the required site. User can seach the site according to the list of regions such as city geoname. The second alternative seach is based on user's GPS position to recommend the neighbor sites. In the other search, user can input terms to find the required site. Figure 4 depicts the searching interfaces.

4) *Site presentation:* the widget gives the different presentations for the required site according to the different searching schemes. We only show the site display with the search driven by user input terms. In Fig. 4, the wiki data of "taroko" as a Taiwanese scenic spot is displayed.

5) *Tour details:* Fig. 5 depicts the description of one tour information, consisting of details of all sites in each tour date, and a tour path in the electronic map.

The widget will connect REST-based web services during the period of the search scenarios. If "Search by Region" is selected, the activity requests the GeoNames services about the city geonames through the CIM at the first time in query. After this first query, the activities will query the individual repository for city information in advance. In the same way, all XML files of the sites queries by user before are stored for the same query at next time.

B. The Implementation of Internal Web Services

We create seven REST APIs using PHP techniques in internal web services, which automatically aggregate and then reorganize the tour data from the external web services, while few *cronjob* tasks are designed to periodically capture the resources. If the search activities of the widget cannot obtain data from the local repository and external web services, they will request the internal web services. Given that processing the JSON-style data in CIM shows better performance than the processing with XML-style data, each of the APIs defines the resource data with two different formats, XML and JSON. The interactive activities can determine the type of data format required in the widget. In the first API, the resource URLs with the two data styles is represented as "/xml/Get_Point/By_Cate/api_key/{para}" and "/json/Get_Point/By_Cate/api_key/{para}", individually. Generally, to query about the JSON-style data is appropriate, the data will not be copied in the repository.

TABLE I. INTERNAL WEB SERVICES FOR THE TOUR WIDGET

	XML or JSON	Examples
A.	/Get_Point/By_Cate/api_key/{para}	/Get_Point/By_Cate/api_key/398/2
B.	/Get_Plan/Show_Plan /api_key/{para}	/Get_Plan/Show_Plan /api_key/all
C.	/Get_Info/Get_Code/api_key/{para}	/Get_Info/Get_Code/api_key/Loc (or Cate)
D.	/Get_Info/Get_Weather/api_key/{para}	/Get_Info/Get_Weather/api_key/Taipei
E.	/Get_Info/Get_Point_Info /api_key/{para}	/Get_Info/Get_Point_Info /api_key/2
F.	/Get_Point/Get_Range_Point/api_key/{para}	/Get_Point/Get_Range_Point/api_key/121.502892/25.13791

		7/10
G.	/Get_Plan/Get_Point/api_key/{para}	/json/Get_Plan/Get_Point/api_key/4

Table 1 defines the URLs for these REST APIs. The usage of the URLs is detailed as follows:

- The 'A' API is to query a simple list of the sites, while the parameters, {para}, are given with the city and category codes of the sites. The API should be requested after requesting the 'C' API.
- The 'B' API is to query the tour schedules pre-stored in the internal DB. The tour schedules are pre-determined to recommend users that connect the widget to the internal web services in the first time.
- The only two values of {para} assigned in the 'C' API are to figure out the classification of tour sites. If the API is requested, the city or category codes will be responded.
- The 'D' API is to obtain the weather information of the requested position as the parameter value.
- The 'E' API is to provide the detail information of an assigned site. The parameter could be a site code retrieving from the result of requesting the 'A' API. The parameter also can be a tem to search the site.
- The GPS location of the user that manipulates the widget can be a parameter in the 'F' API. The API will respond the simple list of the sites within a square range of the position. The values of the square range are assigned within the {para} of the URL request. In the API's example of the table, the range values are 25 and 10 kilometers.
- The 'G' API is to present the detail of any pre-determined tour schedule in the internal DB, and {para} is a code of the tour schedules. The API should be requested after requesting the 'B' API.

We evaluate each of the APIs with 500 testing requests in the widget. Table 2 shows the average reply time for each of the APIs that was requested, and replied to the widget. The duration includes the API's execution time and the wireless communication time. Experiment results show that the reply duration of the APIs with the JSON-format data is shorter than these with XML-format data.

TABLE II. THE AVERAGE REPLY DURATION (SEC.)

	A	B	C	D	E	F	G
XML.	0.041	0.041	0.109	0.081	0.036	0.077	0.046
JSON	0.0392	0.040	0.107	0.0751	0.0365	0.0554	0.043

Figures 6, 7, and 8 describe the response time of the three APIs that generates the amount of data with three different lengths, large, middle, and small. In the procedure of the 'C' API, to deal with the JSON-format is not always faster than the XML-format procedure. In the procedure of the 'G' API, the procedures with XML and JSON often have the same performance.

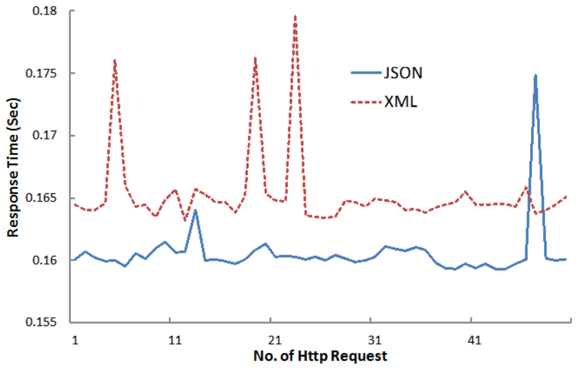


Figure 6. The response time in requesting the 'C' API.

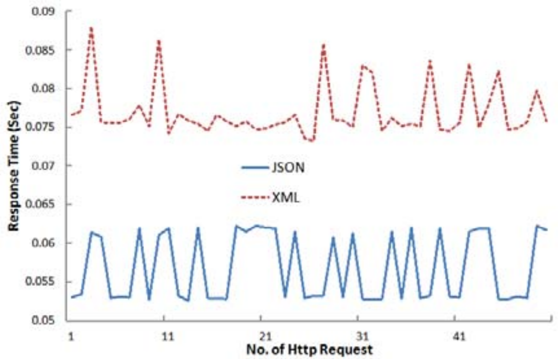


Figure 7. The response time in requesting the 'F' API.

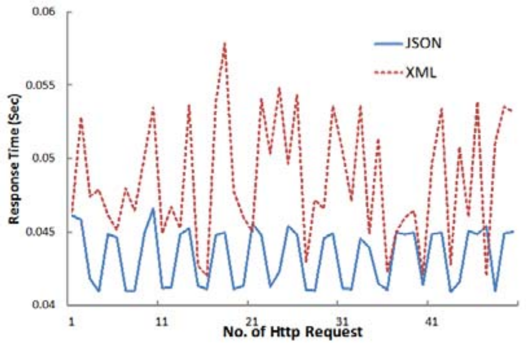


Figure 8. The response time in requesting the 'G' API.

V. CONCLUSIONS

As resources supplied by individuals on Internet is continuous increasing, to integrate ubiquitous web services and to gain external linked data or resource will improve the development also the quality of mobile applications. The research provides a mobile framework with some essential components to deal with REST-based web services and to avoid the problems of unstable wireless communication. The tour widget in Android environment is implemented following the framework guide. Multiple dialogues are designed in the interactive activity components, so that user can easily establish multiple tour schedules. The establishment of each tour will connect to ubiquitous web services such as GeoNames and GDATA to get the information of the sites required by users. Application supports different search schemes, and the CIM component endures the various procedures of service connections corresponding to the searches.

To implementing the internal web services is an alternative method that the widget can gain data from ubiquitous services or resources on Internet. The performance of the internal REST APIs is evaluated with the XML and JSON formats of replying data. In order to integrate various web services to one component such as CIM or SIM, HTTP/REST schemes are better than other similar ones. However, business applications in terms of transaction using SOAP may be more flexible than these implemented with REST.

ACKNOWLEDGMENT

The paper is supported by the National Science Council (NSC), Taiwan, under grants NSC100-2221-E-126-001, NSC101-2221-E-126-006- and NSC101-2221-E-126-002-.

REFERENCES

[1] D. Sonntag, D. Porta, J. Setz, "HTTP/REST-based Meta Web Services in Mobile Application Frameworks", Proc. of the 4nd Int. Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp. 170-175, 2010.

[2] K. Hameseder, S. Fowler, and A. Peterson, "Performance analysis of ubiquitous web services", International Symposium on Performance Evaluation of Computer & Telecommunication Systems (SPECTS), June, 2011.

[3] D. Sambasivan, N. John, S. Udayakumar, R. Gupta, "Generic framework for mobile application development", In Internet (AH-ICI), the Second Asian Himalayas International Conference on, November, 2011, pp. 1-5, doi:10.1109/AHICI.2011.6113938 .

[4] F. AlShahwan, K. Moessner, and F. Carrez, "Distributing Resource Intensive Mobile Web Services", International Conference on Innovations in Information Technology, pp.41-46, Apr. 2011. DOI: 10.1109/INNOVATIONS.2011.5893861

[5] M.Y. Hsieh, H.Y. Lin, and K.C. Li, "Multimedia Recommendation Services based on Social Context Awareness in Mobile Networks", Information-An International Interdisciplinary Journal,vol. 14, no. 7, pp. 2451-2458, 2011.

[6] Malcolm Attard, "Ubiquitous Web Services", Proc. of Computer Science Annual Workshop (CSAW'03), 2003

[7] Irum Rauf, and Ivan Porres, "Designing Level 3 Behavioral RESTful Web Service Interfaces", ACM Applied Computing Review, pp.19-31, 2011.

[8] Lars Johnsrud, Dinko Hadzic, Trude Hafsoe, Frank T. Johnsen, Ketil Lund, "Efficient Web Services in Mobile Networks", on Web Services, IEEE Sixth European Conference, pp. 197-204, 2008.

[9] M.Y. Hsieh, H.Y. Lin, and K.C. Li, "A Web-based Travel System using Mashup in the RESTful", International Journal of Computational Science and Engineering, vol. 6, no. 3, pp. 185-191,2011.

[10] S. Hahmann, and D. Burghardt, "Connecting LinkedGeoData and Geonames in the Spatial Semantic Web", in Proceedings of extented abstracts, Zurich, Switzerland, 2010.

[11] GeoNames, available [online], http://www.geonames.org

[12] Google Data APIs, available [online], https://developers.google.com/gdata/