

A WEB BASED GIS MASHUP FOR ARCHAEOLOGY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BÜLENT ÖZTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
GEODETIC AND GEOGRAPHICAL INFORMATION TECHNOLOGIES

APRIL 2010

Approval of the thesis:

A WEB BASED GIS MASHUP FOR ARCHAEOLOGY

submitted by **BÜLENT ÖZTÜRK** in partial fulfillment of the requirements for the degree of **Master of Science in Geodetic and Geographical Information Technologies Department, Middle East Technical University** by,

Prof.Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences**

Assoc.Prof.Dr. Mahmut Onur Karşlıoğlu
Head of Department, **Geodetic and Geographical Inf. Tech., METU**

Assoc.Prof.Dr. Ahmet Coşar
Supervisor, **Computer Engineering Dept., METU**

Prof.Dr. Adnan Yazıcı
Co-supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Assoc.Prof.Dr. Zuhale Akyürek
Civil Engineering Dept., METU

Prof.Dr. Adnan Yazıcı
Computer Engineering Dept., METU

Assoc.Prof.Dr. Ahmet Coşar
Computer Engineering Dept., METU

Assoc.Prof.Dr. Burcu Erciyas
Settlement Archaeology, METU

Dr. Nigar Şen Köktaş
Space and Defense Technologies, METU-Technopolis

Date:

29 April 2010

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: BÜLENT ÖZTÜRK

Signature :

ABSTRACT

A WEB BASED GIS MASHUP FOR ARCHAEOLOGY

Öztürk, Bülent

M.S., Department of Geodetic and Geographical Information Technologies

Supervisor : Assoc.Prof.Dr. Ahmet Coşar

Co-Supervisor : Prof.Dr. Adnan Yazıcı

April 2010, 96 pages

Information technologies have achieved an important role in archaeology. The management, research and exchange of the large amount of data gathered from archeological sites needs the tools of information technologies. The web based GIS combines the advantages of both GIS and the Internet technologies. This system can be used as a tool that helps to support the management of information for archaeological sites and provides support functions for specialists. This web based system can hold many types of archaeological site data from small excavation campaigns to large sites. The system consists of a relational database, a web server, and a GIS mapping server. Google Maps Server is used as a GIS mapping server in this study. The client computers require only the availability of a proper browser that supports javascript and ajax technologies. With this system, Google Maps can be used as an archaeological research tool for everybody interested in archaeology. In a general, everybody can reach the system using their web browsers in order to search and retrieve information regarding archaeological sites. This system also enables specialists to upload, search and share archaeological data. The aim of this study is to provide easy and simple access to the

GIS related archaeological site information in Turkey, using a web based and user friendly interface and share the information the information with specialists all over the world.

Keywords: WEB , GIS, Google Map, Archaeology, Mashup

ÖZ

ARKEOLOJİ İÇİN AĞ TABANLI CBS MELEZ UYGULAMASI

Öztürk, Bülent

Yüksek Lisans, Jeodezi ve Coğrafi Bilgi Teknolojileri Bölümü

Tez Yöneticisi : Doç.Dr. Ahmet Coşar

Ortak Tez Yöneticisi : Prof.Dr. Adnan Yazıcı

Nisan 2010, 96 sayfa

Bilgi teknolojileri arkeoloji’de önemli bir rol elde etmiştir. Arkeolojik alanlardan toplanan büyük miktarlardaki verinin yönetimi, araştırılması ve alışverişi bilgi teknolojileri araçları gerektiriyor. Web tabanlı CBS, CBS ve internet teknolojisinin avantajlarını birleştirir. Bu sistem, arkeolojik alanların bilgi yönetimine yardımcı destek ve uzmanlar için destek fonksiyonlarını sağlayan bir araç olarak kullanılabilir. Bu web tabanlı sistem örneğin, küçük kazı kamplarından büyük alanlarına kadar çok çeşitli arkeolojik alan verilerini tutabilir. Sistem, bir ilişkisel veritabanı, bir web sunucusu ve bir CBS harita sunucusundan oluşur. Bu çalışmada CBS harita sunucusu olarak Google Maps kullanılmıştır. Kullanıcı bilgisayarlarında sadece Javascript ve Ajax teknolojilerinin kullanılabilirliğini destekleyen uygun bir tarayıcı gerektirir. Bu sistem ile, Google Maps, arkeoloji ile ilgilenen herkes için bir arkeolojik araştırma aracı olarak kullanılabilir. Genel bir şekilde, herkes sisteme arkeolojik alanlara ilişkin bilgi aramak ve almak için web tarayıcılarını kullanarak ulaşabilir. Bu sistem aynı zamanda uzmanlara arkeolojik verileri yüklemeyi, aramayı ve paylaşmayı etkin kılar. Bu çalışmanın

amacı, Türkiye deki CBS ile ilgili olan arkeolojik alan bilgilerine, web tabanlı ve kullanıcı dostu arayüzü kullanarak, kolay ve basit erişim sağlamak ve bu bilgileri dünyanın her tarafındaki uzmanlarla paylaşmak.

Anahtar Kelimeler: WEB , CBS, Google Map, Arkeoloji, Mashup

to my twin brother

ACKNOWLEDGMENTS

I would like to express my appreciation to my supervisor and co-supervisor, Assoc.Prof.Dr. Ahmet Coşar and Prof.Dr. Adnan Yazıcı for their remarkable patience, guidance, suggestions and evaluation during the preparation of this study.

I am grateful to my friend Hüseyin Kalyoncu for his assistance and support.

I am also thankful to,

Özgür Gencel, who has encouraged me to resume my MA degree after 3 years;

Vedat Toprak, for his guidance in finding thesis advisors during my registration period;

Sibel Gülnar, who has supported me in all chancellery work and related subjects;

Nazife Baykal, who gave permission and support for using the computer at METU Inf. Inst. as a host;

Burcu Erciyas, for generating the database of my thesis;

Özlem Albayrak, for her invaluable ideas;

Ümit Kızıloğlu and Sinan Kaan Yerli, for their support in writing the thesis;

Geomatics Ltd. Şti, for providing GPS devices to do some of the measurements;

Volkan Evrin, Yusuf Arla, Mustafa Vural and Mithat Onur for all their support;

Rüştü Öztürk, for his support during the thesis (taaaaaammaaaaaammmmm...);

and finally to my mother and my father, whom I owe everything.

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ	vi
ACKNOWLEDGMENTS	ix
TABLE OF CONTENTS	x
LIST OF TABLES	xiv
LIST OF FIGURES	xvi
LIST OF ABBREVIATIONS	xix
CHAPTER	
1 INTRODUCTION	1
1.1 STATEMENT OF THE PROBLEM	2
1.2 OBJECTIVE OF THE STUDY	2
1.3 STRUCTURE OF THE THESIS	3
2 BACKGROUND	4
2.1 WHAT IS A MASHUP	5
2.1.1 Mashup Archaeology	5
2.1.2 Portable Antiquities Scheme	6
2.1.3 Online Archaeology	7
2.1.4 Heritage Gateway	8
2.1.5 TAY Project	10
2.1.6 Google Maps, Web-Based Map Service	11
2.1.7 Timeline, Web Widget for Temporal Data	14
2.2 RETRIEVING INFORMATION	15
2.2.1 Topological Queries	18

	2.2.1.1	Point In Polygon (Containment Query)	18
	2.2.1.2	Line In Polygon (Containment Query)	19
	2.2.1.3	Polygon Overlay	20
	2.2.2	Distance Queries	20
	2.2.2.1	Distance From a Center	21
	2.2.2.2	Distance From a Geographic Object	21
	2.2.3	Attribute Queries	21
	2.2.4	Relational Queries	22
	2.2.5	Semantic Queries	23
2.3	WEB SERVICES		23
3	SYSTEM DESIGN AND DEVELOPMENT		26
3.1	CLIENT SERVER ARCHITECTURE		26
	3.1.1	Network Side	29
	3.1.2	Server Side (Remote - Map Server)	29
	3.1.3	Server Side (Local)	29
	3.1.3.1	Web and Database Service	30
	3.1.4	Client Side	34
3.2	DATABASE DESIGN		35
3.3	SOFTWARE DESIGN IN SERVER SIDE		37
	3.3.1	Web Server Directory Structure	37
	3.3.2	PHP Class Structure	38
3.4	SOFTWARE DESIGN ON CLIENT SIDE		41
	3.4.1	JavaScript and AJAX Frameworks	41
	3.4.1.1	Sardalya Library	42
	3.4.1.2	ExtJS Library	42
	3.4.1.3	Timeline	42
	3.4.1.4	Projected Overlay Library	43
	3.4.1.5	Google Maps API	43
3.5	GRAPHICAL USER INTERFACE DESIGN		44
	3.5.1	Title of the Mashup	45

3.5.2	Map Section of the Mashup	45
3.5.3	Mini Map Section	45
3.5.4	Timeline Section	46
3.5.5	Query Criteria Section	46
3.5.6	Result Area of the Mashup	47
3.5.7	Layer Area	47
3.5.8	Controls of the Mashup	47
3.5.9	Information Panels of the Mashup	48
3.6	SYSTEM DEVELOPMENT	48
3.6.1	Developer Side Development	48
3.6.2	Client Side Development	52
3.6.3	Server Side Development	53
4	TEST CASE AND IMPLEMENTATION	56
4.1	SYSTEM OVERVIEW	56
4.2	USAGE OF THE SYSTEM	56
4.3	TEST CASES QUERIES	58
4.3.1	Spatial Queries	58
4.3.1.1	Distance (Buffer) From a Center	59
4.3.1.2	Distance (Buffer) From an Object	59
4.3.1.3	Object Proximity	61
4.3.1.4	Point in Polygon (Rectangular)	62
4.3.1.5	Point in Polygon (Province)	63
4.3.2	Attribute, Relational and Semantic Queries	63
4.3.3	Result & Controls of Queries for Hattusa	66
4.3.3.1	Map Controls	67
4.3.3.2	Timeline Controls	69
4.3.3.3	Accordion Type Information Windows	71
4.4	SUMMARY	72
5	CONCLUSION	74
5.1	ADVANTAGES	75

5.2	DISADVANTAGES	75
5.3	RESULTS	76
5.4	FUTURE WORKS	76
	REFERENCES	78
APPENDICES		
A	JSON FORMAT	82
B	ENCODED POLYLINE ALGORITHM FORMAT	86
C	DATABASE TABLES	88
D	SOME CODE SAMPLES	94
D.1	GOOGLE MAPS API CODE SAPMLE	94
D.2	EXTJS (AJAX) CODE SAMPLE	95
D.3	PHP CODE SAMPLE	96

LIST OF TABLES

TABLES

Table 2.1	Database comparison	17
Table 3.1	HTML sample	31
Table 3.2	PHP sample	31
Table 3.3	SQL connection and query	32
Table 3.4	Some database tables	36
Table 3.5	PHP class definition	39
Table 3.6	PHP class structure	40
Table 4.1	Map controls	68
Table A.1	Simple JSON example	85
Table C.1	Table: excavation	88
Table C.2	Table: eventimeline	88
Table C.3	Table: excavation_general_info	89
Table C.4	Table: excavation_periods	89
Table C.5	Table: excavation_references	89
Table C.6	Table: excavation_session	90
Table C.7	Table: excavation_session_supporters	90
Table C.8	Table: images	90
Table C.9	Table: people	90
Table C.10	Table: permission	91
Table C.11	Table: previous_works	91

Table C.12	Table: publications	91
Table C.13	Table: stratigraphic_sequence	91
Table C.14	Table: supporters	92
Table C.15	Table: team	92
Table C.16	Table: team_session	92
Table C.17	Table: team_session_people	93
Table C.18	Table: visual	93
Table D.1	Initilazing "Google Maps API"	94
Table D.2	ExtJS (Ajax) grid panel code	95
Table D.3	PHP sql query code sample	96

LIST OF FIGURES

FIGURES

Figure 2.1 "Mashup Archaeology Aggregating Museum Archaeology & Archaeological Heritage" web page	6
Figure 2.2 "Portable Antiquities Scheme" web page	7
Figure 2.3 "Online Archaeology" web page	8
Figure 2.4 "Heritage Gateway" web page	9
Figure 2.5 TAY web page	10
Figure 2.6 "Google Maps" web page	12
Figure 2.7 Timeline example	14
Figure 2.8 Point in polygon	19
Figure 2.9 Line in polygon	19
Figure 2.10 Polygon overlay	20
Figure 2.11 Distance from a center	21
Figure 2.12 Distance from a geographic object	22
Figure 2.13 Web service	24
Figure 2.14 "Google Maps" service	25
Figure 3.1 Simple structure of client-server architecture.	26
Figure 3.2 System architecture	28
Figure 3.3 Server side components	30
Figure 3.4 PostgreSQL sql query 1	33
Figure 3.5 PostgreSQL sql query 2	33
Figure 3.6 Database architecture	35
Figure 3.7 Ajax working structure	41

Figure 3.8 GUI Design	44
Figure 3.9 Title of the Mashup	45
Figure 3.10 Map section of the Mashup	45
Figure 3.11 Mini map section	46
Figure 3.12 Timeline section	46
Figure 3.13 Area query criteria	46
Figure 3.14 Other query criteria	46
Figure 3.15 Result area of the Mashup	47
Figure 3.16 Layer area	47
Figure 3.17 Controls of the Mashup	47
Figure 3.18 Information panels	48
Figure 3.19 "Quanta Plus Web Development" tool	49
Figure 3.20 pgAdmin	50
Figure 3.21 phpPgAdmin	51
Figure 3.22 PostgreSQL console	51
Figure 3.23 Firebug plugin for Firefox.	52
Figure 4.1 Screen shot of Mashup	57
Figure 4.2 Flow chart of the Mashup	58
Figure 4.3 Distance from a center	59
Figure 4.4 Distance from a river (in 35km)	60
Figure 4.5 Distance from a river (very-near)	61
Figure 4.6 Proximity (in 5km)	62
Figure 4.7 Proximity (very near)	62
Figure 4.8 Points in a rectangular area	63
Figure 4.9 Points in a province	64
Figure 4.10 Distance from a center belonging to the Neolithic period	65
Figure 4.11 Points in a rectangular area belonging to the Hittite period	65
Figure 4.12 Points in a province with Phrygian period	66

Figure 4.13 Distance from a river belonging to the Bronze age period	66
Figure 4.14 Proximity with Neolithic period	67
Figure 4.15 Zoom-out Hattusa	68
Figure 4.16 Zoom-in Hattusa	69
Figure 4.17 Pdf exports of Hattusa	70
Figure 4.18 Timeline and layer of Hattusa	70
Figure 4.19 Photo panels	71
Figure 4.20 Info panel 1	72
Figure 4.21 Info panel 2	72
Figure A.1 Object in JSON format	82
Figure A.2 Array in JSON format	83
Figure A.3 Value in JSON format	83
Figure A.4 String in JSON format	84
Figure A.5 Number in JSON format	84

LIST OF ABBREVIATIONS

Adodb	Database Abstraction Library for PHP
ADSL	Asymmetric Digital Subscriber Line
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BSD	Berkeley Software Distribution
DOM	Document Object Model
GIS	Geographic Information Systems
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
OS	Operating System
PHP	Personal Home Page
RSS	Really Simple Syndication
SIMILE	Semantic Interoperability of Metadata and Information in unLike Environments
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
WSDL	Web Services Description Language
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

Turkey has a very rich historical and cultural heritage. Therefore, many important archaeological studies in our country. There are many archaeological sites and cultural heritage areas, which are spread all over the country. Most of these sites have been or are being excavated. Others are waiting for excavation or discovery. Although there are many, it is not easy to access information such as locations, histories, works in progress, the discovered remains and related information. Information sources for these are not stored in easily accessible media and in some cases require some authorized permissions in order to access them.

Accessing the information sources is easy through the Internet. Therefore, desired information about archaeological sites can also be reached by ordinary people, if they are placed on the Internet in an accessible way.

This system was designed and implemented for easy access to information about archaeological sites and related findings. Accessing information is not sufficient by itself. Retrieving selectively only the required information is also important in order to avoid information pollution. To achieve this, some search methods using various criteria have been designed. One of the most important criteria is to be able to make spatial queries. In this way, it is possible to perform spatial search by specifying criteria such as geographical boundaries, distance to reference geographical locations, proximity to a class of geographical object (such as rivers, mountains, hills, etc.).

Prior to conducting this study, it was realized that there are no web based archaeological systems that contain information and allow filtering or that have search capabilities. Besides, it was also realized that there is no web based system that stores archaeological features that

can be queried. Existing systems/databases usually store only simple textual attributes about archaeological sites and researchers working on that site. The lack of such a system causes specialists to perform their research unaware of other works or studies.

1.1 STATEMENT OF THE PROBLEM

Maps have always been important to humanity. They show where we are living, where our natural resources are located, how to go to places, where our neighbours are, where our borders are, etc.

For a specialist who may be an archaeologist it is necessary to locate archaeological sites on the map and archaeological background information on those sites. A database that contains all the textual and graphical information and is linked to maps that show the locations of archaeological sites would be of great value to especially archaeologists, but also to experts studying man made artifacts.

Most of the archaeological excavations have their own web sites and hold all related information in web pages in non-standard or non-uniform formats. It is necessary that all sort of information are stored in a standard format. Otherwise it may be impossible or extremely difficult for researchers to find the data they need. This system is designed and planned to overcome such problems.

1.2 OBJECTIVE OF THE STUDY

The objective of this study is to design and implement a web-based map mashup for archaeological sites. It is intended to add spatial information to the database so that it is possible to issue spatial queries on the maps using this system. Combining maps, archaeological information and timeline, it will be possible to create a mashup application and publish it on the Internet. In order to create such a system, we must design a suitable GUI (Graphical User Interface) that is able to use and make this system reachable with technologies such as Ajax, PHP, PostGIS, and integrate it with external web-based systems such as Google Maps API. By designing such a system, the main goal of easily sharing archaeological site information will have been achieved.

1.3 STRUCTURE OF THE THESIS

This paragraph briefly describes the structure of the thesis. Chapter 2 gives the background of similar archaeological web sites and technologies which are commonly used in this type of systems. In this chapter, some examples are given in order to define and have a better understanding of mashup based systems and the capabilities of existing systems. Chapter 3 introduces the design and development of our system. In the design section, the parts of the system are explained, and in the development section, the tools and technologies that have been used in building the system are introduced. Chapter 4 describes some case statements and implementations of the system. Some supported search queries and results are also shown in this chapter in order to better describe capabilities of our system design and to point out the differences with the existing ones. In Chapter 5, the conclusions and advantages/disadvantages of the system are discussed, achieved results and possible future works are mentioned.

CHAPTER 2

BACKGROUND

Geographic Information Systems (GIS) have rapidly spread out in scientific area and daily life. It is used in almost all geoscientific applications, and as a result has increasingly entered our daily life. Meanwhile, the development of the Internet and related technologies introduce new applications in the field of GIS, "web based GIS".

Internet is a useful and powerful tool for gathering and manipulating information. Most of the information known to the humanity can be reached through the Internet. This is also possible for geographic information (Using GIS in Public Policy Analysis in North Carolina, Watershed Education for Communities and Officials NC State University). Today, many people can easily search, find, reach and access all kinds of information by taking advantage of the web, which was not easy before the Internet. Therefore, Web-based GIS has become an important spatial information source for data entry, sharing, searching, analysis and easy access. By the help of the web, even an ordinary user can benefit from GIS through a computer which is connected to the Internet. This was not possible before the Internet. It is not a privilege of specialists to use GIS. Web based access by the public makes the use of web based GIS by people who are interested in the subject, easy.

In this study, more than one system and architecture were used in order to build such a web-based GIS system. In this chapter, brief information about the background of the study, the tools and some definitions are given.

2.1 WHAT IS A MASHUP

Mashup is defined by Paul J. D. and Harvey (Paul J. D., Harvey M. D., 2008) as a "Combined content or functionality from existing web services, websites and/or RSS feeds to serve a new one". It also is described as "more than one sources and services are "mashed up" or combined or mixed to make a hybrid or to create a new one or adding valuable something which is known" (Ajax-Powered Google Maps Mashup Tutorial, ORACLE Technology Network). A set of common technologies are used for making a Mashup application. JavaScript, Ajax, JSON, php, Postgresql and PostGIS add-on and finally, what is important for our mashup, Google Maps API technologies are brought together for a Mashup.

In the same manner, in a GIS mashup, at least one of the contents is connected with geographic information or technologies in nature and it can typically be displayed on a map (GIS Mashups for Geospatial Professionals, GEOG 863). In this study, the geographical side of this Mashup is connected to the mapping data supplied by Google Maps with a data service supplied by PostGIS and its archaeological spatial data (Ajax-Powered Google Maps Mashup Tutorial, ORACLE Technology Network).

According to "Farallon Geographics Inc."s definition of "Enterprise Mashup" as stated on their web pages, mashups have in common two main threads. The first one is using location and maps as a theme, the second one is providing a decision support by using a simple visual web interface (Enterprise Mashup, Farallon Geographics). In the following pages, a few archaeological mashup applications for the web will be introduced. However, some of them lack GIS and / or mapping technologies on their web pages. These will be represented first, then web-GIS based ones will be shown. So the differences between Web based GIS mashup and the others, which are not GIS based applications, can be compared.

2.1.1 Mashup Archaeology

"Mashup Archaeology, Aggregating Museum Archaeology & Archaeological Heritage" web page is first example.

In Figure 2.1, "A Mashup Archaeology, Aggregates Web Content on Museum Archaeology and Archaeological Heritage" is shown. The web site takes advantage of web feeds (such

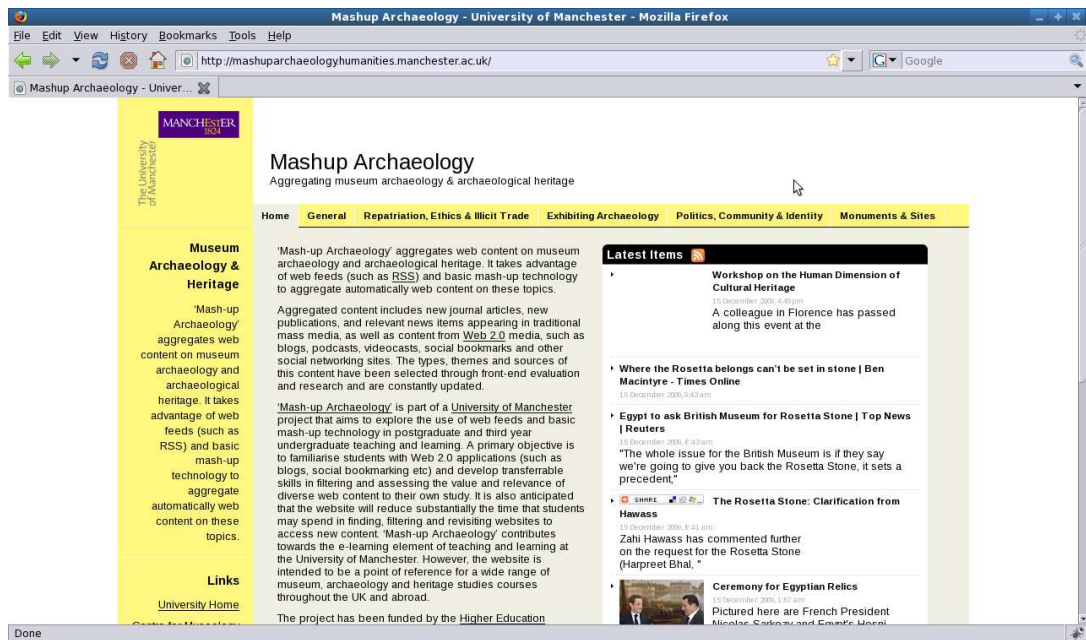


Figure 2.1: "Mashup Archaeology Aggregating Museum Archaeology & Archaeological Heritage" web page

as RSS) and basic mashup technology. Its contents include journal articles, publications, and news which appear in traditional media, as well as blogs, pod-casts, video-casts, social bookmarks and other social sites appearing in web 2.0 media.

This mashup application is part of a project that is carried out by the 'University of Manchester'. Their project aims to teach the use of web feeds and basic mashup technology to postgraduate students, third year undergraduate students and people related to archaeology (Dr K. Arvanitis & Dr S. Jones, Mashup Archaeology).

Although it uses some web technologies and archaeological sources with data, it has no geographical applications or mapping interface related with this mashup.

2.1.2 Portable Antiquities Scheme

Another example is the "Portable Antiquities Scheme" web site.

This site (Figure 2.2) records the archaeological objects which are found by metal-detector users and ordinary people who are discovering these objects when walking, gardening or going somewhere in daily life in England and Wales. The site provides news, articles, event

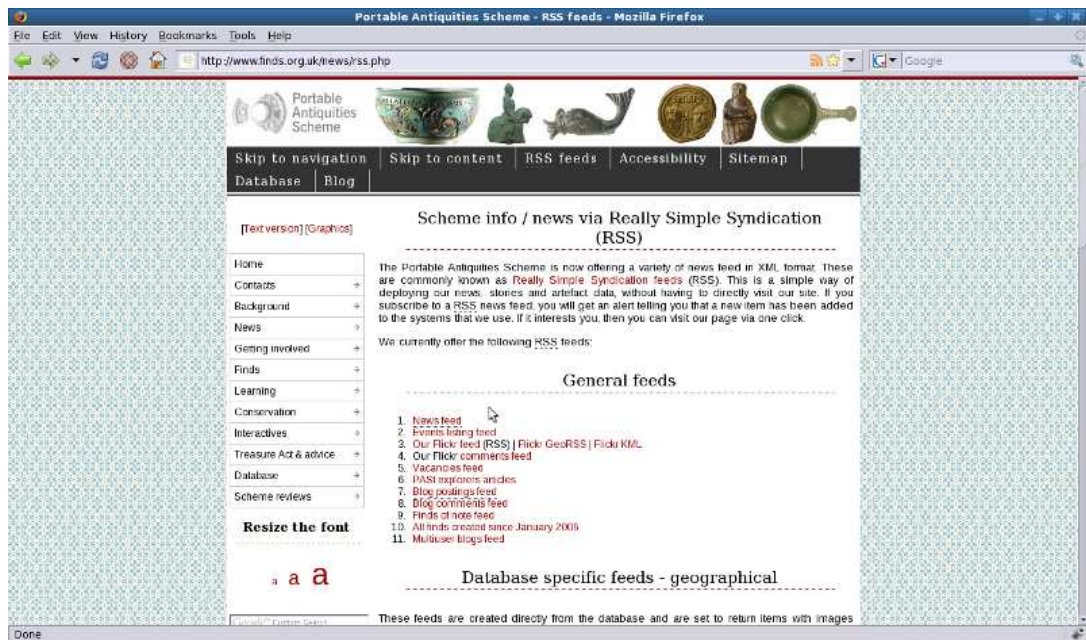


Figure 2.2: "Portable Antiquities Scheme" web page

listings and access to its own database of objects and images. It also gives some valuable information about archaeology and related subjects. Basically, the mashup side of this site consists of web RSS feed and collected data about archaeological remains. These feeds are divided into three main categories. General feeds consist of news, articles, photos from 'flickr', blogs, event listing, and notes. Database specific feeds show the latest finds with names of location. Period feeds show the finds with their periods that extend from paleolithic to post-medieval. Basically, the site was structured on data entries which come from people who share these data with RSS feed technologies. However, this site does not include any GIS based web application.

2.1.3 Online Archaeology

"Online Archaeology - UK Archaeology Map Resource" (Steve W., Online Archaeology) is one of the examples that has geographic information and maps together in a mashup application.

The web site (Figure 2.3) consists of two main parts. The first part is holding text based information where all archaeological things are discussed in a forum stored by this site. This

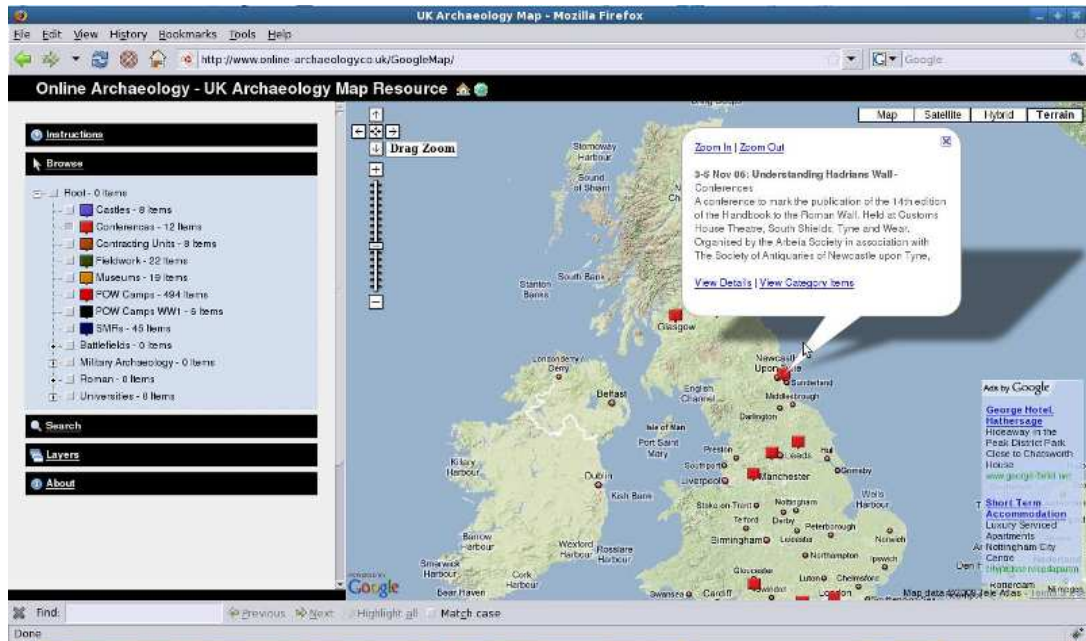


Figure 2.3: "Online Archaeology" web page

part also includes books, news, blogs, articles, links, movies and photos. The second part holds the maps and GIS information about the archaeological sites in UK. For the second part, web technologies, Google Maps and archaeological data with locations are used in order to assemble a mashup application. On the web site, the archaeological sites are shown on a map when selected from the list which is located on the left side of the map. Each row on the list shows locations of selected items. The search section is simple and can only be searched by layer name. The map is created by Google Maps API and all items are displayed on it. Although it combines archaeological data, simple search, and web based maps, this second part lacks GIS based database and spatial queries. This site can still be defined as an archaeological mashup.

2.1.4 Heritage Gateway

The fourth example of mashup application on archaeology is "Heritage Gateway".

The web site (Figure 2.4) provides people the ability to search historical sites and buildings in England (The Historic Buildings and Monuments Commission for England, Heritage Gateway). The search part of it contains where, what, when, who, and resources sections. So, any



Figure 2.4: "Heritage Gateway" web page

combination of these sections; location (where), type (what), period (when) and person (who) can be used as search criteria together. The resources can also be selected. In the "Where Section", the search is made depending on text based search and the result is signed and shown on the map. After that, a selection is made for "search within" the selected location. The "search within" choice creates a rectangle on the map which determines the borders of the search parameters within a combo-box. Depending on the selection made, a message like "5 km of selected location" or "100 meters of selected location" etc appears. The results are shown on a new page in a row based notation. In the "What Section", the search parameters consist of Building and Site type. It has a tree based notation. Each category is divided into sub categories like tree branches. The "What Section" search corresponds to the "Where Section". Search results are coming from both sections' combined queries. The "When Section" determines periods parameters of the search queries The "Who Section" holds the person who is associated or who is the architect and the "Resources Section" contains all data. The site is a complete mashup example with its sources, construction, mapping application and its archaeological data. Although it is a good example of a mashup applications, it lacks complex spatial queries and spatial database. The search results are not shown on the map either. It also does not support the layer application with timeline.

2.1.5 TAY Project

The last example (Figure 2.5) of an archaeological mashup is TAY Project (Türkiye Arkeolojik Yerleşmeleri- Turkey Archaeological Settlements).

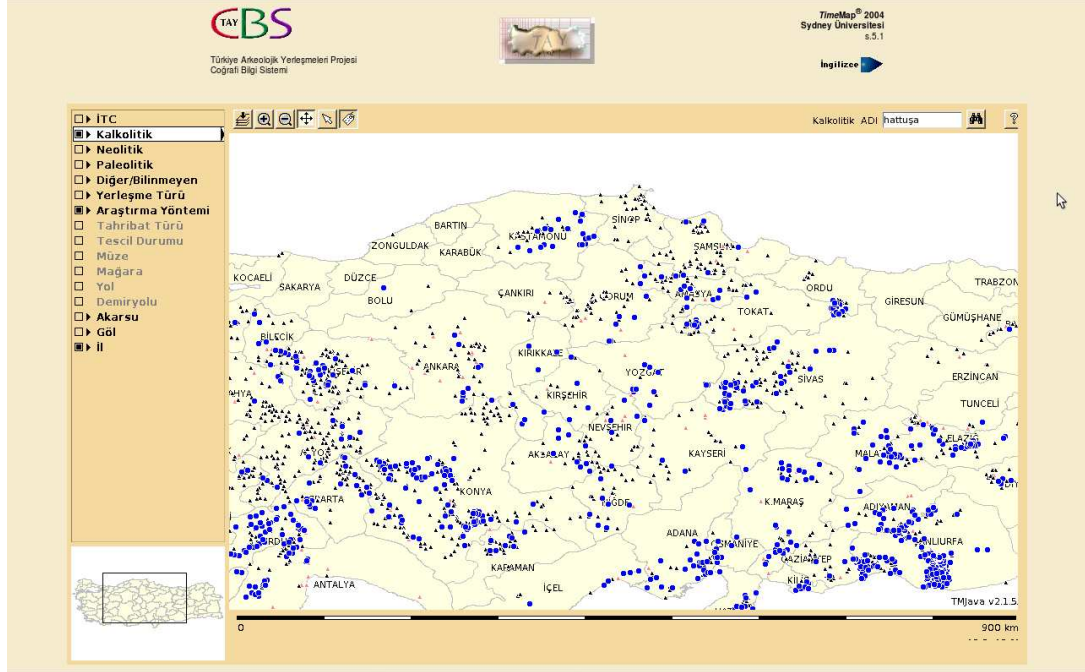


Figure 2.5: TAY web page

This project is a good example of mashup. It contains a set of web technologies and valuable information about archaeological data. The aim of this project is to publish, validate and monitor archaeological data of Turkey, as well as to create and share chronological inventories on an international platform by the help of web technologies. The web site also gives some information about archaeology.

The most important part of the site is the TAY-CBS (GIS) section. It shows the locations of the settlements and some simple text based searches can be made in this section. Java Applet technology was used in building this site. In order to make use of it, the web browser either has to support java applet or suitable java programs have to be installed on the computer.

A simple map of Turkey is located on the center of the page. Some selection criteria options are shown on the left side of the page (Figure 2.5). These selections are historical chronology, settlement types, research methods, registration status, museums, cave, railway, river or lake.

When one of them is selected, matching points, appear on the maps. A text based search can also be made using the textbox in the upper right corner, together with the selection made in the left of the map. The search results are shown on the map and a pop-up window.

Although it uses mixed web technologies and archaeological data, it lacks spatial database and spatial queries such as proximity, area search etc. It also lacks capability to show satellite image and layered mapping. However it is the most important contributor to archaeological science as a source of collected information.

2.1.6 Google Maps, Web-Based Map Service

There are many web-based mapping services and their APIs are ready to be used by the general public. Two of the open sources are GeoServer and MapServer. The Commercial softwares include ObjectFX Web Mapping Tools, ArcGIS Server, ArcIMS, GeoWebPublisher, GeoMedia, Oracle Map Viewer and LizardTech's Express Server. Although there are many similar applications and sites, the most useful and widespread web-based mapping service is Google Maps. It uses the Mercator projection for mapping (Google Maps, Wikipedia), hence the angles between latitude and longitude are preserved. Although it has some distortion when in a zoom-out view, the distortion is not noticeable in a close-up view (Google Maps Help Forum, Google).

Google Maps (Figure 2.6) is an advanced web mapping service which contains the latest web technology, up-to-date information of maps, and satellite images. It also contains some locational information such as driving directions, street views and POI (Point of Interest) search such as, buildings, areas, schools, squares, national parks, shopping centers, parking areas, etc. All of these are provided by Google's databases.

In addition to all these features of Google Maps, it gives users the opportunity to benefit from the information that is stored. It also gives the user the ability to create his or her own customized maps. In order to achieve this, Google Maps software called API, which is available for free.

Google Maps API is a collection of application programs objects which were created by Google and allows developers to integrate Google Maps into their websites with their own data by using JavaScript functions. It is possible to embed or re-write the full Google Maps

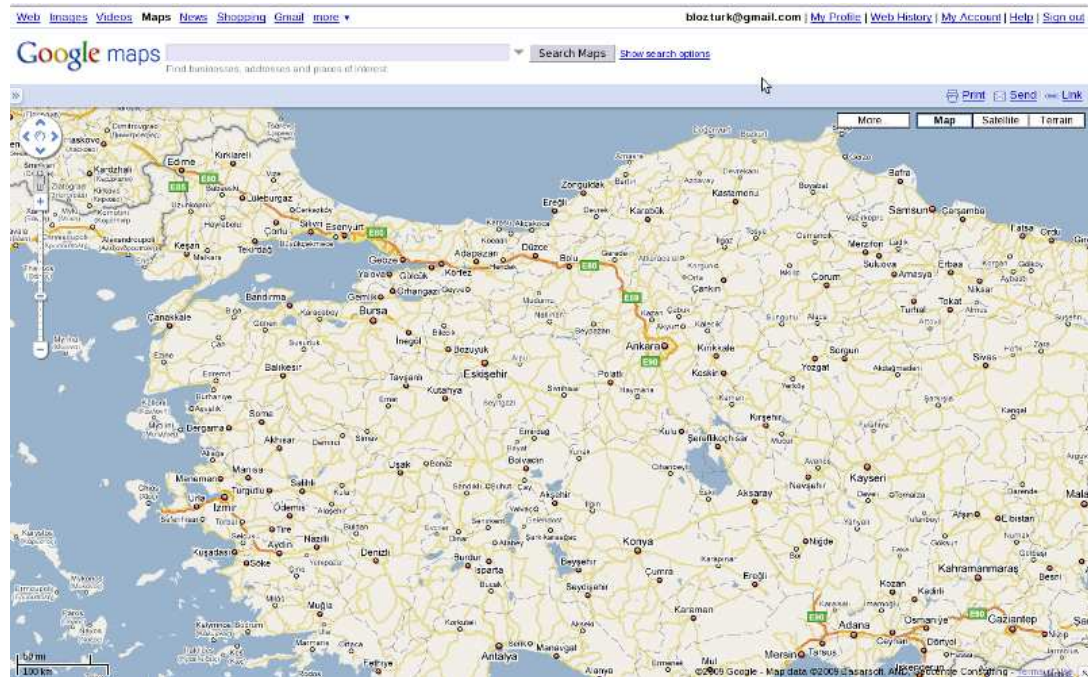


Figure 2.6: "Google Maps" web page

site tailored to personal requirement.

The API files are loaded from Google when loading the web page. The API contains many javascript codes and functions. After loading the web page, API functions become ready to use. These codes, functions, or variables are called by the client's side programs that are also composed of JavaScripts. Therefore, these are run on the client side as a piece of the program.

The reasons for choosing Google Maps in this study are:

- simplicity

The end user, who is using the maps, does neither have to use complex or expensive software nor does he have to learn new and complex using guides. This gives the opportunity to save considerable time and money in building a web site coupled to Google Maps if it is programmed in a right way. According to G. Nicoara "Users simply want to see results" (G. Nicoara, GITA) and that's true. Google Maps API provides this, if it is programmed in a right way.

- free software

Obeying the license agreement for non-commercial use, it can be used everywhere where maps are part of a mashup application. Therefore, it is preferred by many users and as a result, there are many new applications, codes and mashups developed using it.

- easy to learn

There are many various web-based application mashups etc. that can be used as examples. Besides, there are also many online resources for learning and example codes can be found from forum sites where the codes are discussed in detail.

In addition to all these, JavaScript and AJAX support make Google Maps easy and applicable in every way.

The other advantages of Google Maps API are listed below;

- Quick development time
- Light weight application on client side
- Easy user interface
- Fast loading
- No complex display for selected area and layer
- Focused only on mapping.

Although it has many advantages, it also has some disadvantages that are listed below;

- More limited functionality than commercial products
- Difficult to overlay complex GIS data
- Can not read GIS database directly
- Fewer people are interested in its development

Despite all these disadvantages, Google Maps and its API are one of the most desired, selected and used web based GIS applications and mapping services.

2.1.7 Timeline, Web Widget for Temporal Data

Presentation or displaying of chronological sequence of events or objects, that are related to time can be defined as a timeline (What is Timeline, Midmarket CIO Definitions). Graphical presentation makes it easier for the user to understand the temporal data. It is used in many branches of science in order to view periods of development, chronology, or changes in time. Briefly, it shows events that happened. There are some web-based tools for visualizing temporal data, information or objects, related with time. Their purpose is to show the temporal data using web based technologies such as Flash or Ajax. One of them is Simile-Widgets project's Timeline.

Timeline is a JavaScript code based API component. It allows web site creators to assemble an interactive timeline and display its data on their sites. JavaScript supported web browsers are sufficient to run it. Simile-Widget's Timeline is often used on web pages that have Google Maps API (Simile-Widgets, MIT). When both of them are used, they show the temporal data on a map and create a simple mashup.



Figure 2.7: Timeline example

The Timeline (Figure 2.7) contains one or more bands. Each band is scaled with date and contains events. It also can be panned by the mouse pointer, mouse scroll or arrow keys. If there are more than one band, they can be synchronized with each other with different time scales (Simile-Widgets, MIT).

The time data which are shown on the bands are in JSON format. These are static on the

web page and client side or dynamically produced on the server side. Each event data can be programmed by using JavaScript and assigned a function, which is activated when clicked. So the Timeline and its events make the web page dynamic.

2.2 RETRIEVING INFORMATION

Before describing information, we have to define data and relations between information. In a simple way, data can be defined as raw facts. They are facts such as phone numbers, addresses, coordinates, location names, areas etc. All of these do not form meaningful things by themselves. They must be organized and brought together in order to create a meaningful content. Information is the organization of these "raw facts" in a meaningful manner (The Information, eCheat.com). Although well organized, reported or tabulated, it is not always meaningful to people.

There are many ways to arrange data to make them meaningful. The type of data also affects the information. If data is put together inappropriately with respect to type, the information cannot become meaningful.

Of course, all these data must be stored and retrieved when needed. The storage media or systems are called database. For different types of data, there are different types of databases. The data, stored on database, could be queried and retrieved to become information which is ordered and made meaningful. Therefore the information is created from the data, which are stored in a database by querying in a correct way. For geographical data, a different type of database is needed so that the relations between geographical data can be queried.

Spatial database system is a kind of database system, that proposes principal database technology for Geographic Information Systems. According to traditional databases, spatial database deals with managing, storing and querying data related to space such as geometric, geographic or spatial data objects in space, including points, lines and polygons (R.H. Güting, 1994).

Traditional databases deal with the objects attributes and their features, quantities, etc.. Therefore, there is no connection with the location of an object (e.g. an archaeological site) and its other attributes, such as date range, remains, finds, structural evidence etc.. As a result, the space related queries can not be answered in traditional databases (D. Wheatley, M. Gillings,

2002).

- What are the shapes of artifacts which are found on "abc" excavation site ?
- What is the quantity of pottery in the "xyz" museum ?

are examples of traditional database queries (PostGIS, Using PostGIS)

However, spatial databases give answers to queries that relate to space-location queries that traditional ones can not handle.

- What is the total length of all silk roads, expressed in kilometers ?
- What is the distance between two archaeological sites ?
- What are the locations of archaeological sites within a certain distance to a river "xyz" ?
- What are the positions of archaeological objects that belong to the Iron Age period and are within 15km to the "xyz" river ?
- Are there any excavation sites in the selected area which are supported and managed by the government ?

are examples of spatial database queries (PostGIS, Using PostGIS).

Today, there are many spatial databases or traditional (relational) databases with spatial extensions. MySQL with spatial extension, PostgreSQL with PostGIS extension, Firebird, Ingres with spatial object library, MaxDB, IBM DB2 Spatial Extender, Oracle Spatial, Informix Spatial DataBlade Microsoft SQL Server are examples of both commercial and open source spatial databases (G. Brent Hall, Michael G. Leahy, 2008).

Some popular open source and commercial spatial databases are compared in Table 2.1.

PostGIS is a spatial extension for PostgreSQL object relational database. It was written in C. So, it is a C-based spatial engine for PostgreSQL. It enables spatial related queries and retrieval support for geographic information systems in PostgreSQL. The PostGIS's power comes from being a standard back-end for most applications and tools as a "Free and Open

Table 2.1: Database comparison

Database Comparison			
	MySQL	PostGIS	Oracle Spatial
OGC compliant	SQL with Geometry types. Compliant with the exception of precise spatial operations	SFSQL-TF 1.1 Certified	SFS1; GML 2.0; OLS 1.1; SRS; WMS 1.1
Spatial data-types (vector-oriented) 2D,3D	2D only, Rtree keys	As specified in OGC SF-SQL: Point, Linestring, Polygon, Multipoint, Multilinestring, Multipolygon, Geomtrycollection	All types supported in SFS1 + circles, arcs, combination of arcs and lines and rectangles; Support 3D storing of lines / points / polygons
Spatial data-types (raster-oriented)	N	CHIP datatype to store rasters in PostgreSQL	Grid-based data and image data are supported using GeoRaster data type
Support of spatio / temporal models	N	User maintenance	Versioned tables with Workspace Manager
Exchange formats XML, GML, CityGML, X3D, KML output	N	All FME Formats GML, SVG KML	GML 2.0 and much of GML 3.0. OGCs Open Location Services. Forthcoming WFS-T, Web Catalog Service, OpenLS

Source Software for Geospatial” . It is also used heavily by applications and libraries in the Java development language (G. Brent Hall, Michael G. Leahy, 2008). Therefore it is commonly used in a wide area and many different application platforms.

If we have data, database and queries, there has to be a retrieval system. In order to talk about an information retrieval system, we simply have to have a storage or database, data and queries. It depends on the relevant information that comes from the database by reading all stored documents. In practice, the user is never interested in the selection of the relevant

information, because, the user has no time to read and process the entire document collection. Instead, computers read the entire document collection in order to extract the relevant ones by using the appropriate set of queries (Information Retrieval, The Information Retrieval Group, Computing Science University of Glasgow).

As a geographic information, a retrieval system can be shortly defined as searching for any kind of geographic related information or meta-data on a spatial databases in response to a spatially related queries (M. van Kreveld, I. Reinbacher, A. Arampatzis, R. van Zwol, 2004).

Spatial querying can be defined as a special type of database queries of data types such as points, lines and polygons and that these consider the spatial relationships between their geometries (Spatial Query, Wikipedia).

There are primarily three types of queries used for retrieving geographical information. These are, phenomenal or attribute queries, topological queries, and distance queries (J. Conolly, M. Lake, 2006).

Spatial queries require that examination of the spatial properties of the objects will provide the selection set. They can be divided into two types: Topological and Distance (Buffering) queries (J. Conolly, M. Lake, 2006).

2.2.1 Topological Queries

Topological queries are concerned with the geometric configuration of an object or relationship between objects. For example, "*select all sites within Smith Country*" (J. Conolly, M. Lake, 2006).

Assume that there are two map layers. One of them holds points which represent archaeological sites and the other one holds polygons which represent state boundaries. The question "*select all archaeological sites in Maine*" is thus a topological query as it depends on establishing whether a point lies within the boundaries of a polygon.

2.2.1.1 Point In Polygon (Containment Query)

To determine whether a point is inside a polygon or not.

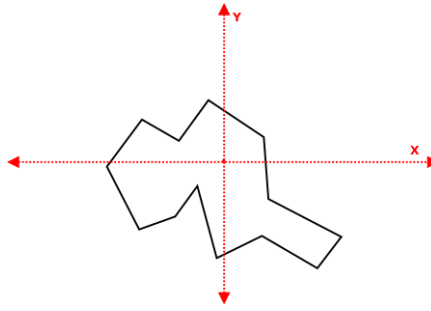


Figure 2.8: Point in polygon

Illustrated in Figure 2.8, which essentially asks the question "*What do we have at this X,Y point in the current coordinate system ?*" (R.R. Larso, Berkeley).

It gives the answer to the questions (J. Conolly, M. Lake, 2006) ;

- What proportion of the bronze age barrows in Wiltshire is located on Chakland ?
- Select all artifacts that come from building N in settlement S.
- How many sites were found in the last survey zone ?
- Are there any Late Iroquoian villages in Simcoe Country ?

2.2.1.2 Line In Polygon (Containment Query)

To determine whether a line crosses through a polygon boundary or not.

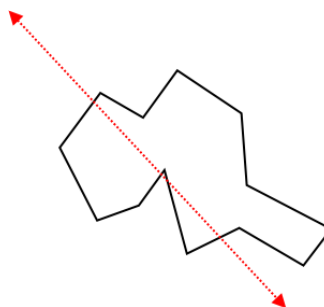


Figure 2.9: Line in polygon

It gives the answer to the questions (J. Conolly, M. Lake, 2006);

- Which counties does Offa's Dyke pass through ?
- How many kilometers of Roman road are found in Surrey ?
- How many public footpaths are there in the Avebury World Heritage Site area ?

2.2.1.3 Polygon Overlay

To determine whether a polygon intersects a polygon or not.

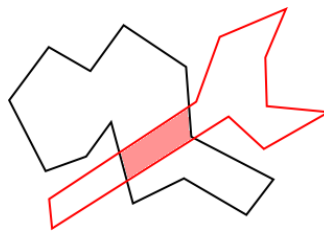


Figure 2.10: Polygon overlay

Polygon overlay queries can give answers like (J. Conolly, M. Lake, 2006);

- How many hectares of the proposed new urban zone falls in archaeological sensitive areas ?
- What is the amount of arable land that lies within a 5-km radius of the site ?

2.2.2 Distance Queries

Distance or buffer queries are concerned with the spatial location of objects. The distance and buffer zone query asks the question "*what do we have within some fixed distance of this object (point, line or polygon)*" (J. Conolly, M. Lake, 2006).

Distance queries can answer questions like;

- What proportion of sites falls within 1 km of the coast ?
- What is the change in density of sherds (fragment of pottery) moving away from the center of site k in 100-m intervals ?

- What is the difference in the average amount of high-grade arable land falling within 5 km of sites of type A versus sites of type B.
- What proportion of all scrapers are found within 2 m of hearth features ?

2.2.2.1 Distance From a Center

To determine whether there is a point object or not inside a circle with a certain radius.

For example, *”select all sites within 100 km of an obsidian source”* (J. Conolly, M. Lake, 2006).

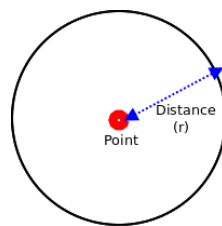


Figure 2.11: Distance from a center

Figure 2.11 shows the result of a buffer zone of a distance n around a point with a circle of radius n .

2.2.2.2 Distance From a Geographic Object

To determine whether there is a point object or not inside the buffers of a polygon or linestring.

For example, *”select all sites within 4 km of a river Green”*

Figure 2.12 shows the results of a buffer zone of a given distance, n , around a polygon.

2.2.3 Attribute Queries

Attribute queries are concerned with the non-spatial data tables of spatial objects. For example, *”select all sites that have obsidian artifacts”* (J. Conolly, M. Lake, 2006). Here, there is no need to make a geographical or locational queries.

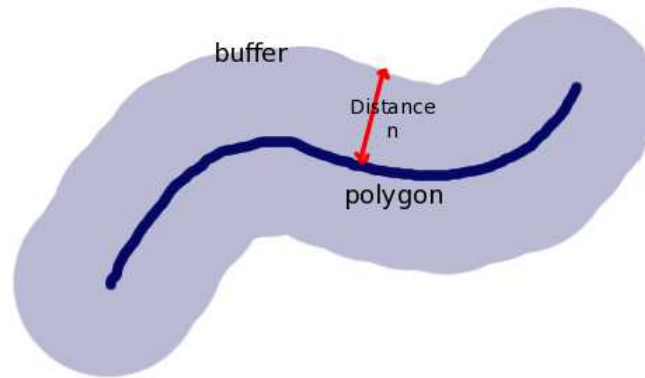


Figure 2.12: Distance from a geographic object

2.2.4 Relational Queries

The advantage of relational database model is that sets of data can be retrieved from across two or more tables (relations) by specifying the primary and foreign keys (J. Conolly, M. Lake, 2006).

If we want to find the total amount of archaeological sites which belong to the bronze age and their excavations supported by the government, the data have to be retrieved from two or more different database tables.

Assume that the first table is the archaeological site table that holds archaeological information, geographical type locations, and as a primary key, site IDs. The second table is the chronological table that holds the archaeological chronologies of the site with the foreign key, site IDs. The third one holds the supporters and also uses a foreign key, site ID. "site ID" provides the relation between the tables, as both primary and foreign key.

Therefore, it is possible to make a query between tables which are related with each other. The relation between two or more tables gives the resulting set of the amount of archaeological sites.

2.2.5 Semantic Queries

Semantic queries enable retrieval of both explicitly and implicitly derived information based on syntactic and semantic information contained in the database.

They also attempt to help a user to obtain or manipulate data in a database without knowing its detailed syntactic structure - unlike - syntactic queries which only support retrieval of explicit data based on syntactic information (KRLAB, CSIMP).

In contrast to simple queries against full-text indices, semantic queries allow queries like *"show me all excavation projects we have done in Turkey in the last 15 years"* (Semantic Web Company, Portal).

The other example, in a semantic is the query: *Show the rounded shape artifacts in certain defined location where iron age level*".

2.3 WEB SERVICES

A web service is defined as a software component stored on one computer that can be accessed via method calls by an application (or other software component) on another computer over a network. The communication between them is made possible by technologies such as XML and HTTP (Paul J. D., Harvey M. D., 2008).

In a web service, the client sends the request parameters to the server. The server processes the request and responds to the client in a particular structure. So it can be said that the web service structure is simple and composed of client, request input, server, process, and output response.

The simple web service, client-server architecture and HTTP protocols between them are shown in Figure 2.13 (Web Service, Wikipedia).

For example, assume that, archaeological sites data are kept in a database. These data are provided by using a web-based application when a request is received. So this system represents a simple web service.

In that case, web service is a structure that can convert our application to web-based ap-

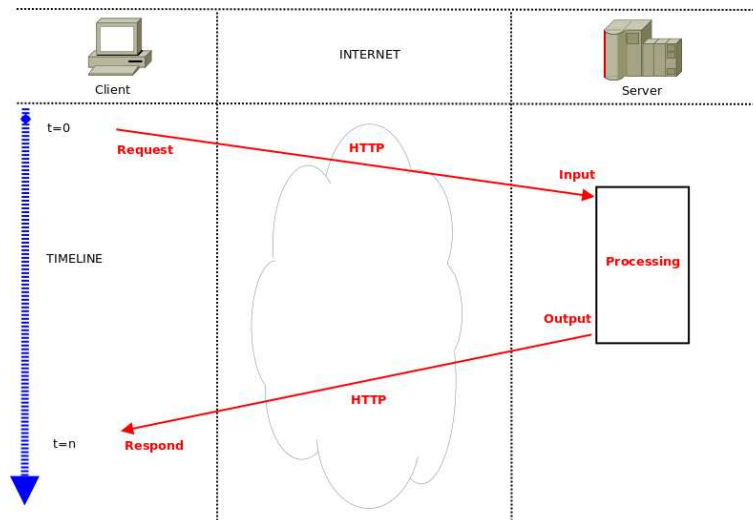


Figure 2.13: Web service

plication. Generally, web services are using WSDL between each other. WSDL includes information in an XML based language.

The other web service protocol is SOAP. SOAP is also XML-based and lets application exchange information over HTTP protocol.

Other protocols are also used. One of them is JSON. It communicates over HTTP protocol and is native JavaScript language notation. It is simpler than SOAP which uses verbose XML. It is easy to parse by any language and libraries and therefore also easy to read or write with any tool (Developer Network, Yahoo). Currently, web services are requesting the remote service using API or web API infrastructure.

In previous definitions, all mashup applications can be defined as a web service. One of the most important examples of the web services is Google Maps API.

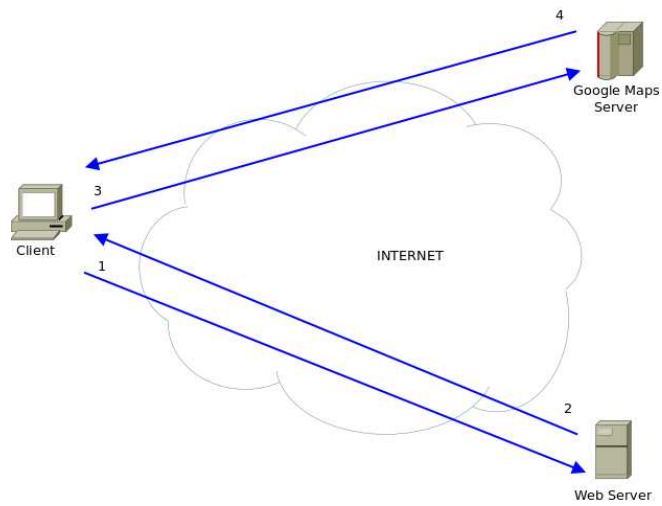


Figure 2.14: "Google Maps" service

Figure 2.14, simply shows how a Google Maps Service works. In this figure, the client requests the web page from the web server which keeps the required information about map and data. The web server replies to the request in step 2. So the client can build the web site with HTML structure and Google Maps API. In step 3, the client requests the API from the Google Maps Server. After the map API is requested by JavaScript and loaded, the other request which is related to the maps, coordinates etc. is built by Google Maps API on the client web browser.

CHAPTER 3

SYSTEM DESIGN AND DEVELOPMENT

3.1 CLIENT SERVER ARCHITECTURE

The basic structure of the system was thought of and designed as a client-server architecture. This architecture model is the fundamental of the Internet. Today, the web servers that we are using are based on this model. There are three main components in this model. As shown in Figure 3.1, these are the client, the server and the network as a communication medium.

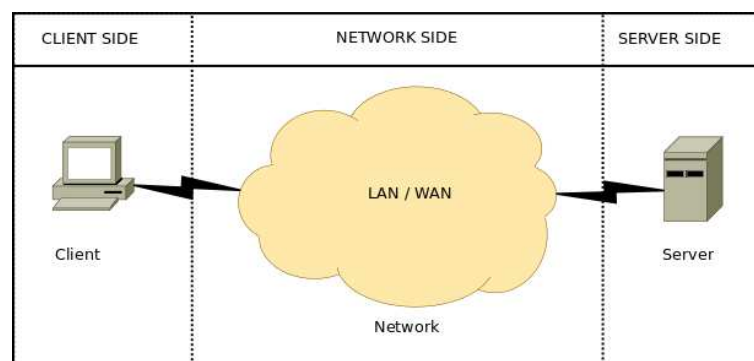


Figure 3.1: Simple structure of client-server architecture.

The client, in a simple way, usually requests data from the remote server's service. To connect to the remote server, it uses some type of network, hardware and software protocols. It provides an interface for the front end of the user interaction. With the help of this interface, it requests any kind of information or data. Therefore, it can show the results set which comes from the server. Servers are indispensable for centralized systems on which all information is kept.

The server side usually gets requests from the remote clients and provides data or information to them. This is the major purpose of the server. It usually manages the data or information in its database or file systems.

A server is capable of communicating with more than one client at the same time. When it gets a request from the client, it processes the requests and provides the data from the database. The server is like a meeting point of clients who come to buy something. Both of them are talking with each other by the help of a connection media like the Internet.

Without any connection media between the client and server, there cannot be such an architecture. The connection media is shown in Figure 3.1.

The media between the client and the server computer can be a network modem, a simple local area network, wide area network or the Internet. Communication rules, standards or any hardware equipment are not important at this time. The only important thing is that it provides communication between them (Pros and cons of client/server computing, Exfoesys Inc.).

In a client-server architecture, all resources are centralized. This means, all data, files etc. are stored in one location, the server. This provides the advantage of easily backing-up files/data and easily finding these. The server software and hardware are optimized for multiple users/clients who can reach or access this resource simultaneously.

On the server side, security control is much more advanced than on the client side. The server controls all access to the data and secures it. It is easy to add new resources to the server (Internet FAQ Archives, ...advantages and disadvantages of Client/Server).

Although it has more advantages, if the server or its network connection goes down, all clients who depend on this system are disabled. The installation of client-server architecture may be expensive on the server side. It always needs to be maintained by an expert IT staff (Internet FAQ Archives, ...advantages and disadvantages of Client/Server).

The system is generally working as a client-server architecture. In Figure 3.2, the system architecture and the connections of more than one clients to the web server over the Internet is shown.

On the client side, each one has a web browser. They may be in different locations and more than one of them can be connected to the server simultaneously.

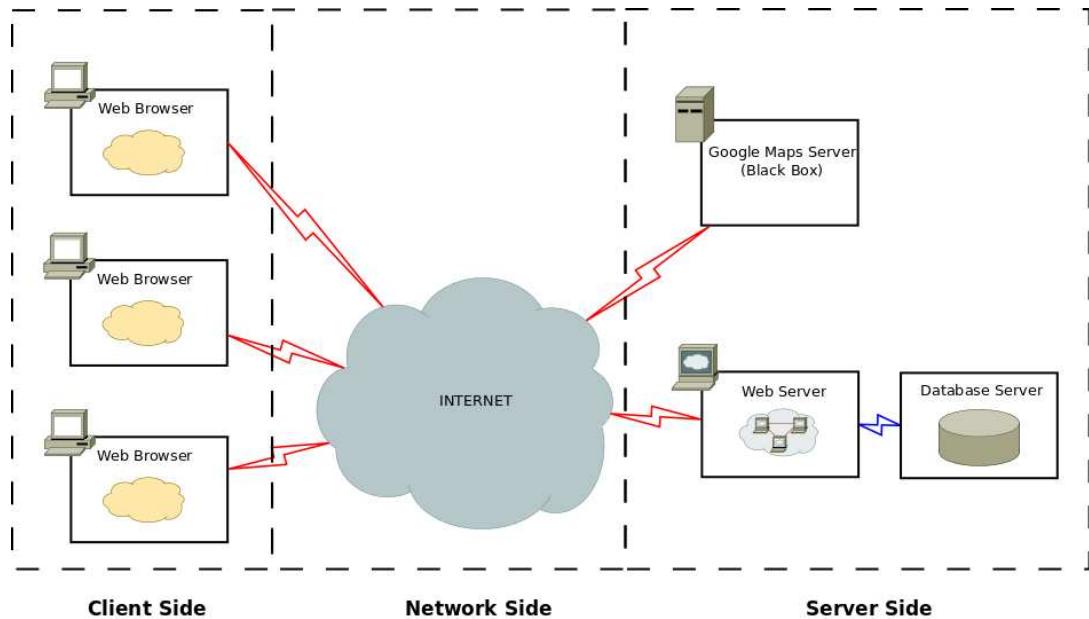


Figure 3.2: System architecture

The client can use any kind of network architecture to reach the servers. These can be adsl, metronet, satellite connection, etc. but are simply called "Internet". The server is a computer that a web service is running on. It replies to incoming web requests. Depending on the requests, it connects to the database server, executes a query, gets the results and sends them back to the client who requested the information.

In this study, clients also connect to the map server (Google Maps), request and get some required data for mapping features. So, the client can show the mapping information to the end user by the help of the web browser and its user interface.

The user, on the client side, uses JavaScript and DOM supported web browser. On the server side, there are two main servers. One of them is the local server that holds the web service, database service and archaeological data. The other one is the remote server that holds the mapping service. The client gets the required component, a special code library used when requesting information from the local web server. These components are in HTML files, JavaScript codes, address and keys for getting Google Maps API. These are run on the client computer after receiving the requested data from the server.

After that, the client also connects to the Google Maps Server and gets other necessary infor-

mation, codes, files etc. in order to create and display the map on the browser.

More detailed explanations of the installed components on the client and the server sides will be given later. Details of the network side are not included in this study, because the network itself is beyond the scope of this study. For this study it is enough to know that the network is a tool that provides communication - connection between the client and the server. The hardware and software are of no concern here.

3.1.1 Network Side

The network, can be considered as a hardware and software architecture that provides communication between all servers and client components. In this respect it is sufficient for it to be able to supply the desired communication.

3.1.2 Server Side (Remote - Map Server)

In this system, the map server is the second server that provides maps. The Google Maps Server is used as a map server in this study. Its inner architecture hardware and software are unknown.

In order to connect to and use Google Maps, some APIs are used. These are programs that run on the client side and provide the connection between the client browser and Google Maps Server using their functions. These APIs were downloaded from Google's server to the client.

It is sufficient for the programmer or user to know the API reference and commands. According to the request, the used API connects to and gets the necessary information from the Google Maps Server. Therefore, the browser can show the maps and related information coming from Google.

3.1.3 Server Side (Local)

On the server side, the operating system is the Linux based "Debian". This OS has multi-tasking and client-server computing capacities. It can handle a large number of requests and processes.

As a hardware, the server is an ordinary desktop level computer with Debian OS. Although it is a desktop level computer, it does not mean that it is a low level computer. When necessary, the hardware capacities can be upgraded. The server is located in METU Informatics Institute and benefits from the Institute’s network infrastructure. The services on that server are web and database. Detailed information about these are below;

3.1.3.1 Web and Database Service

Web and database servers are running on the server computer. Although both are working on the same computer and hardware, they could also have been working on separate systems like two servers.

Due to the easy management of the systems, they are brought together in the same server used for this study.

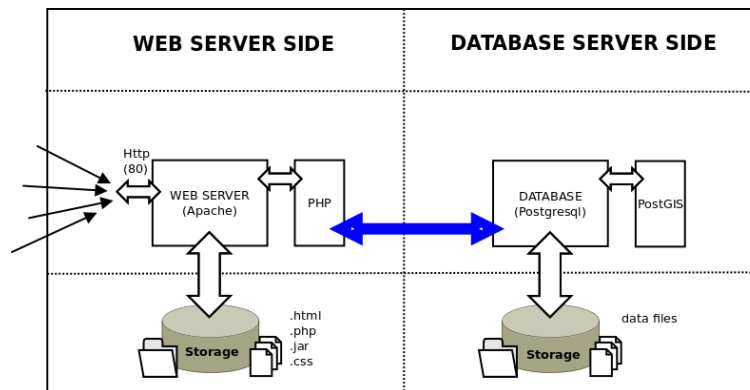


Figure 3.3: Server side components

In Figure 3.3, the interaction of both systems with each other is illustrated. The incoming requests to port 80 of the server, get processed by an Apache Web Server. According to the request, it is either processed by PHP engine and replied to or the requested file is sent to the client without any processing.

If incoming requests refer to ".php" suffix files, the Apache Web Server gets the file from the storage and sends it to the PHP Hypertext to be preprocessed.

The PHP Hypertext Preprocessor interprets on the fly and sends the results back to the Apache.

After that, Apache only sends this interpreted result to the client. In the meantime, if required, the PHP Preprocessor connects to the database, sends the query and gets the result.

Some HTML and PHP codes are shown in Table 3.1 and Table 3.2 below.

Table 3.1: HTML sample

A.html (Server Side)
<pre><HTML> <BODY> The content of the body element is displayed
 in your browser. </BODY> </HTML></pre>
Result (On Browser)
<pre>The content of the body element is displayed in your browser.</pre>

Table 3.2: PHP sample

A.php (Server Side)
<pre><HTML> <BODY> <?PHP echo "Hello"; echo "
"; echo Date("1 F d, Y"); ?> </BODY> </HTML></pre>
Result (On Browser)
<pre>Hello 1 January 03, 2010</pre>

Establishing the connection to the database is done by the PHP Preprocessor. This connection can be either a socket type, or if the security of the server permits, a TCP/IP connection from

the outside Internet.

A sample connection string to the database in the PHP file is shown in Table 3.3 below; In this sample, Adodb class is used.

Table 3.3: SQL connection and query

connection-class.php (Server Side)
<pre>\$this->odb->PConnect(\$this->dbHost,\$this->dbUserId,\$this->dbPassword,\$this->dbName); \$sql = "select * from A-tablosu"; \$query = \$this->odb->Execute(\$sql); \$result = \$query->GetRows(); print_r(\$result);</pre>

PostgreSQL database, which is running on the server, is listening to port 5432 in order to respond incoming requests to that port. To be able to connect to this port, the connection conditions must satisfy the security and safety rules. These rules are stored by the database itself.

PHP add-on of Apache web server makes connection to the database server using this port. Therefore, it is not necessary for the web and database servers to be on the same server. Both of them can be separate in different systems as individual servers (Figure 3.3).

Here is an example of queries which are incoming to the database.

SELECT < selection criteria > FROM < table of selection > WHERE < condition >

SELECT * FROM excavation;

The query and result are shown below in Figure 3.4

The database supports both, relational and spatial queries with PostGIS plug-in. PostgreSQL database is generated with the spatial feature "enabled" when it is created. So, spatial queries can be made on the "geometry" type column of the table.

Let's do a simple query in spatial terms as shown in Figure 3.4. In this table, "location" field is geometry type.

SELECT sub_id, name, type, site_type, astext location , start_date, height, nation, end_date

```

bulent@bulent:~$ psql -U postgres -d ArchaeoSiteMap
psql (9.3.8)
You are now connected to database "ArchaeoSiteMap" as user "postgres".
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=# select sub_id, name, type, site_type, astext(location), start_date, height, nation, end_date from excavation ;
-----
sub_id | name | type | site_type | astext(location) | start_date | height | nation | end_date
-----
      4 | Aphrodissia | | | POINT(37,708984 28,722332) | | | | |
      5 | Ispendos | | | POINT(36,83889 31,172222) | | | | |
      7 | Thuatira | | | POINT(38,920833 27,841667) | | | | |
      9 | Kasalhouk | | | POINT(37,883472 32,821956) | | | | |
     10 | Hasankeuf | | | POINT(37,714167 41,413086) | | | | |
     11 | Hattusa | | | POINT(40,019722 34,615278) | | | | |
     12 | Rusahinli | | | POINT(39,820856 43,403899) | | | | |
     13 | Gordion | Research | Open | POINT(39,685 31,994167) | 1900-01-01 | | | | US
(8 rows)

ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=#

```

Figure 3.4: PostgreSQL sql query 1

FROM excavation;

```

bulent@bulent:~$ psql -U postgres -d ArchaeoSiteMap
psql (9.3.8)
You are now connected to database "ArchaeoSiteMap" as user "postgres".
ArchaeoSiteMap=#
ArchaeoSiteMap=#
ArchaeoSiteMap=# select * from excavation ;
-----
id | sub_id | name | type | site_type | location | start_date | height | nation | end_date
-----
  4 |      4 | Aphrodissia | | | | 01010000007EC9C6932D7942400796B1B01E2C3F40 | | | | |
  5 |      5 | Ispendos | | | | 0101000000EE09140E00734340801F977077073B40 | | | | |
  7 |      7 | Thuatira | | | | 0101000000EE09140E00734340801F977077073B40 | | | | |
  9 |      9 | Kasalhouk | | | | 0101000000F470c2036908424082B5D00d4FB44440 | | | | |
 10 |     10 | Hasankeuf | | | | 01010000008091240086021440623F9760114E1140 | | | | |
 11 |     11 | Hattusa | | | | 010100000048DE5939401424340E91E77F72B2E345140 | | | | |
 12 |     12 | Rusahinli | | | | 010100000048DE5939401424340E91E77F72B2E345140 | | | | |
 13 |     13 | Gordion | Research | Open | | 010100000048703D00d7D343402FC37FB81BE3F40 | 1900-01-01 | | | | US
(8 rows)

ArchaeoSiteMap=#

```

Figure 3.5: PostgreSQL sql query 2

Thus, in the "location" field, (Figure 3.5), the data which were meaningless to us before, turn into point data which indicate coordinates. In more complex queries, it can be answered whether given points are in a polygon or not and their proximity to each other can be compared.

More detailed information about the database section will be given.

3.1.4 Client Side

The computer on the client side may use any operating system. Because today's operating systems are graphical-based, the programs which are running on these systems also depend on graphical interface. So, it is sufficient for the web-browser to have any of the operating systems.

This is mainly because web standards and high maturing technologies are better supporting them than other browsers (Web browser standards supports, Web Devout).

Apart from this, the client side does not need anything except being able to access the Internet. The web browser which is working on the client side and its structure will be mentioned in this chapter's software design section.

3.2 DATABASE DESIGN

A spatial database is used for this study. A PostgreSQL database was installed. After the database installation, the spatial add-on of PostgreSQL, PostGIS, was installed. Therefore, the database gained spatial properties through the support of PostGIS. The installation details can be found in PostGIS manual from the Internet. In this study, some database tables and relations are shown in Figure 3.6.

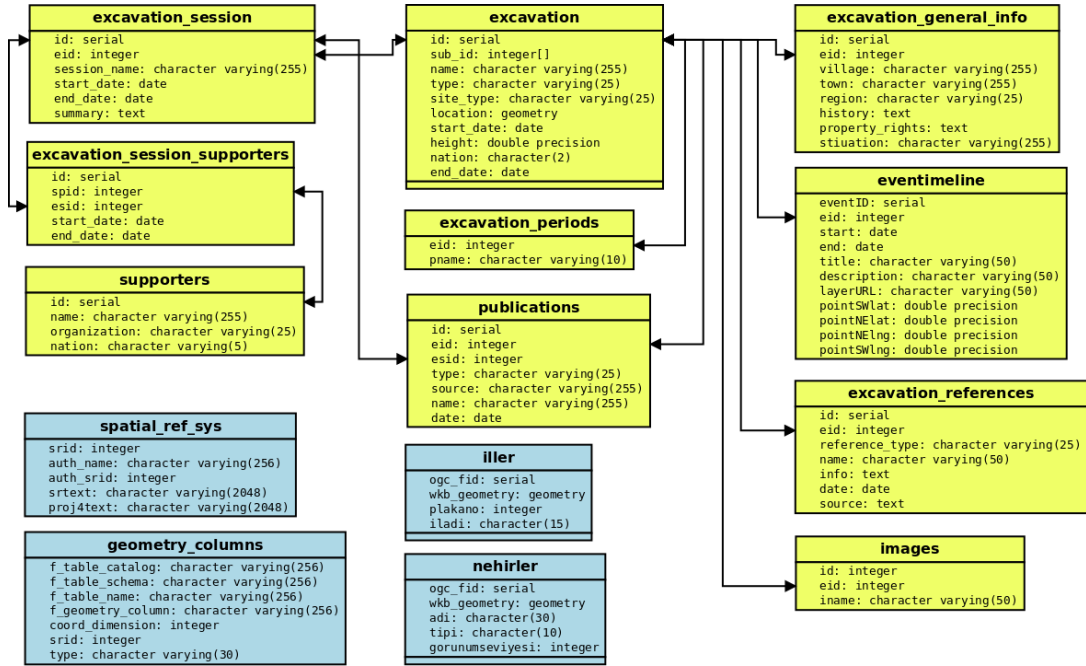


Figure 3.6: Database architecture

The database contains 18 related and unrelated tables. Details of all database tables are shown in Appendix C. In Table 3.4, some of the tables of database are shown;

Table 3.4: Some database tables

Table excavation		
Name	Type	Description
id	serial	PRIMARY KEY
sub_id	integer[]	
name	character varying(255)	
type	character varying(25)	
site_type	character varying(25)	
location	geometry	
start_date	date	
height	double precision	
nation	character(2)	
end_date	date	
Table timeline		
Name	Type	Description
eventID	serial	PRIMARY KEY
eid	integer	
start	serial	FOREIGN KEY
end	serial	
title	serial	
description	serial	
layerURL	character varying(50)	
pointSWlat	double precision	
pointNElat	double precision	
pointNElng	double precision	
pointSWlng	double precision	

3.3 SOFTWARE DESIGN IN SERVER SIDE

3.3.1 Web Server Directory Structure

The Apache Web server and its files form the main part of the system. These files include all information about;

- how the client connects to the map server
- which data is sent by Ajax
- how the interface will be
- and how the system works and is displayed

These files determine the "real" system and each of them is a part of a mashup program.

Files are located in Apache "DOCUMENT_ROOT" directory and usually are found in "/var/www/" directory in many Linux systems.

Here, these are categorized by their functions with sectioning folders. This makes them easy to maintain, add, remove or upgrade. Adding new functions or files at a later stage, can be done easily in a suitable location of folders.

Below, the addresses of the folders and files are shown;

- DOCUMENT_ROOT/ArchaeoSiteMap/var
- DOCUMENT_ROOT/ArchaeoSiteMap/roots.php
- DOCUMENT_ROOT/ArchaeoSiteMap/lib
- DOCUMENT_ROOT/ArchaeoSiteMap/photos
- DOCUMENT_ROOT/ArchaeoSiteMap/class
- DOCUMENT_ROOT/ArchaeoSiteMap/test
- DOCUMENT_ROOT/ArchaeoSiteMap/img

The main folder is, "DOCUMENT_ROOT/ArchaeoSiteMap". All other folders are in upper levels.

- **"var"** folder contains the variable files. In this folder, there is a file named "global.php". Variables are held in this file. This file is included in other ".php" files and thus they get the same variables from one place "global.php" files.
- **"lib"** folder contains some framework files. These are "Sardalya", "Timeline", "Adodb", "ExtJS" and a few ".js" file. In codes, some of them provide connection to database, some of them provide GUI, some of them support Ajax and javascript extensions etc. Each "library" or frameworks are included in codes.
- **"photos"** folder holds the photos, pictures, or drawings.
- **"class"** folder holds the class files of PHP. These class files also contain "packets" which are used in codes in order to make them simple, avoid repeating codes and use to make the same structure everywhere in the programs.
- **"test"** folder contains test codes and examples of the system. Before publishing the real codes, all are tested in this folder and debugged.
- **"img"** folder contains some images which are used in GUI.

This structure is the familiar Linux file structure and is easy to maintain for anybody who knows Linux.

3.3.2 PHP Class Structure

Php classes are used in many sections of the codes. Here, they are used as an object-oriented programming tool. Briefly, we can say that, object-oriented programming consists of classes, methods, and objects. Objects can be defined as data structures which contain a set of routines. These are called methods. So a class contains a collection of these methods and objects (PHP Classes, Spoono).

The benefit of using classes is that superior organizations with less repetitive code can be established.

Table 3.5: PHP class definition

Simple Class definition
<pre><?php class SimpleClass { // property declaration public \$var = 'a default value'; // method declaration public function displayVar() { echo \$this->var; } } ?></pre>
Creating an instance
<pre>\$instance = new SimpleClass(); // This can also be done with a variable: \$class_name = 'Foo'; \$instance = new \$class_name(); // Foo() ?></pre>

Here is a simple class example in Table 3.5 (Class and Objects, PHP Manual);

In our study, an example of class is in Table 3.6;

Table 3.6: PHP class structure

class.php
<pre>class asm{ function asm(){ /// This section is executed when the instance created.. /// mainly prepering the database connection is here. ... }function asm // \$parameter is query criteria. function Cquery(\$parameter){ /// query text \$sql = "select * from TABLE_A where TCOLONE_A = '\$parameter'"; /// execute the query \$query = \$this->odb->Execute(\$sql); /// return the results return \$query->GetRows(); }function Cquery }class asm</pre>
index.php
<pre>// class file is included. So we can reach it. // include("class.php"); // create the instance... \$query = new asm; // make the query using class.. \$query = "837"; \$result = \$asm->Cquery(\$query); // print the results print_r(\$result)</pre>

In this example (Table 3.6), we use another class in order to make a database connection. In PHP, ADOdb classes are used. ADOdb is a database abstraction library for PHP. It currently supports MySQL, Oracle, Microsoft SQL Server, Sybase, Sybase SQL Anywhere, Informix, PostgreSQL, FrontBase, Interbase (Firebird and Borland variants), Foxpro, Access, ADO and ODBC (J. Lim, ADOdb).

3.4 SOFTWARE DESIGN ON CLIENT SIDE

3.4.1 JavaScript and AJAX Frameworks

On the client side, most of the operations are performed in the web browser. These are controlled by JavaScript codes or Ajax frameworks. Actually Ajax is a kind of JavaScript, which is supported by today's high end web browsers. It provides the web browser with data sending and receiving capabilities without reloading entire web pages. Therefore, if it is programmed suitably, the web browser acts as a GUI based software.

The differences between classic web model and Ajax working principles are shown in Figure 3.7 (AJAX - Asynchronous JavaScript And XML, UVic Dept. of Comp. Sci.)

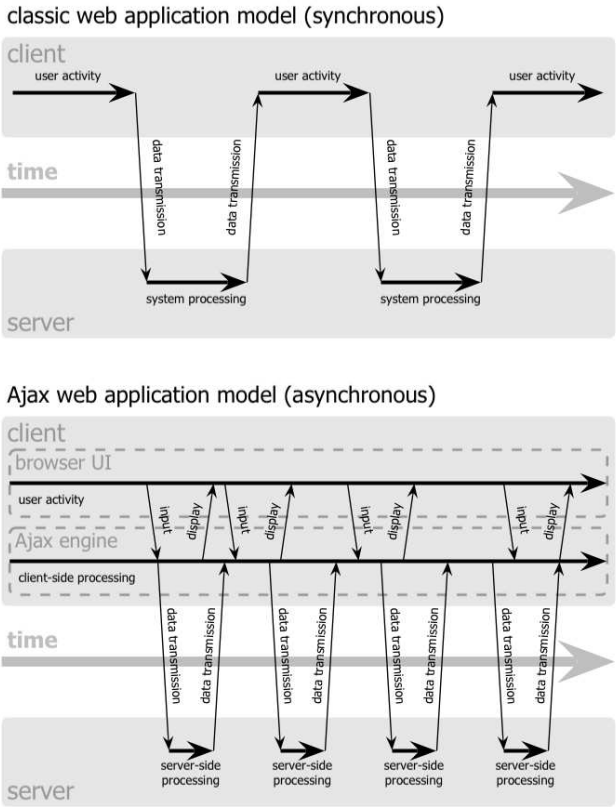


Figure 3.7: Ajax working structure

Although there are many Ajax frameworks, some of them were selected and used on the client side scripting. The selection was made based on an advise of some users who have been using them. By using Ajax frameworks, developing codes on the client side, became

easier than developing the server side PHPs codes. Here are the Ajax frameworks which are used in the client side in codes;

3.4.1.1 Sardalya Library

Sardalya is one of the open source "Ajax library"s, designed to work with today's cross-browsers which are supporting DOM. This library was created by Volkan OZCELIK (<http://www.sarmal.com/sardalya/>). The aim is to develop DHTML programming easily. It is independent from the OS. Although, it was designed for Ajax, it is also used for some other JavaScript functions.

3.4.1.2 ExtJS Library

The second is "ExtJS" Framework. The main purpose of this framework is to create user friendly interface widgets. With this UI, Ajax power is combined for getting a powerfull framework. It enables the programmer to create "amazing web applications within the web standards" with only a few lines of codes. ExtJS is also used for an Ajax property. Communication between the server and the client is also provided by this framework. This communication is made possible by data which is in JSON format. In appendix A, JSON format is given with an example.

3.4.1.3 Timeline

The other application on the client side, after Google Maps, is timeline. It is a kind of dhtml ajax widget for visualizing temporal information. It has been developed as part of the SIMILE Project of MIT(<http://simile.mit.edu/>) with the BSD licence. It enables the programmer to make an interactive timeline. In this study, timeline is located on Google Maps and contains layer related date. These layers represent the situation of the stated location.

3.4.1.4 Projected Overlay Library

This library was used as part of timeline widget. With the help of this library, it is possible to show projected images on the map and change the transparency of the images. It was developed for this purpose by John D. Coryat in 2008. This library is provided free of charge with GNU General Public License.

3.4.1.5 Google Maps API

Of course, the main structure depends on The Google Maps API . Creating a map and other operations related with this map are provided by the help of this API. So, this map can be specialized for our purpose and we can use Google Maps on our web site (Google Code Help, Google). Google Maps API provides map, satellite image and DEM image with some tools.

3.5 GRAPHICAL USER INTERFACE DESIGN

Here, in this section, Graphical User Interface (GUI) is explained. GUI mostly takes advantage of the ExtJS library. ExtJS library provides instant widgets for this study. In Figure 3.8, this part of the interface and locations are shown.

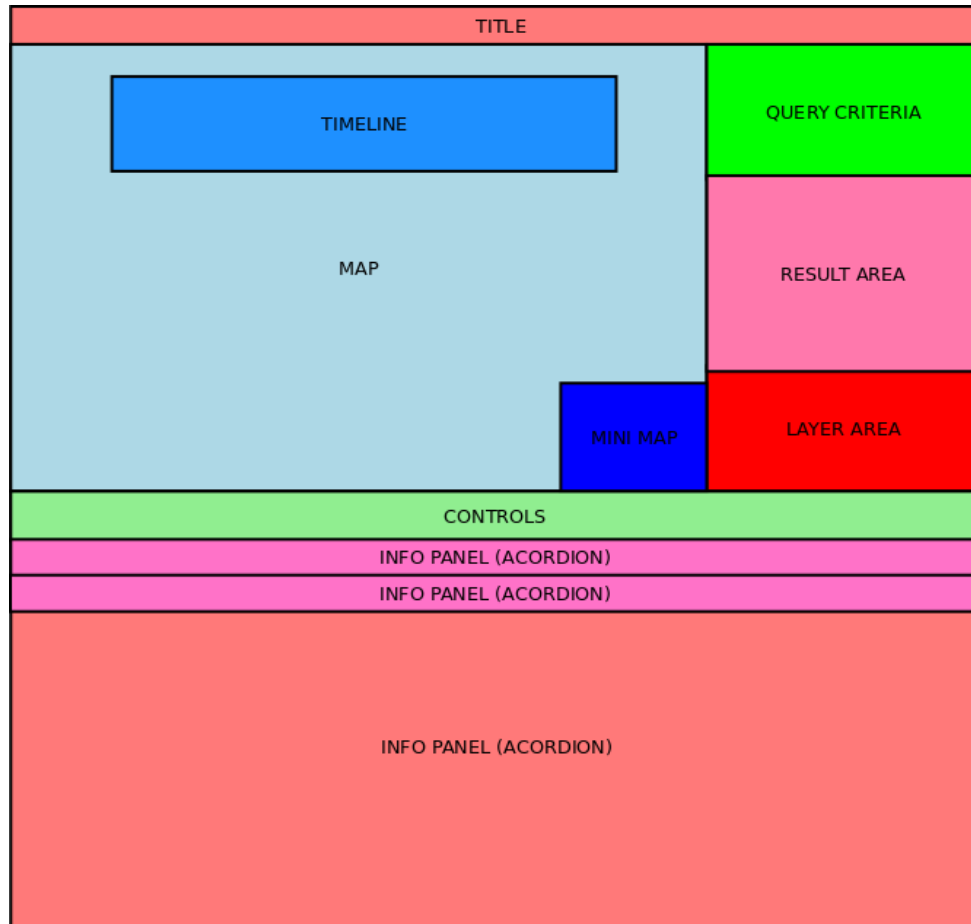


Figure 3.8: GUI Design

3.5.1 Title of the Mashup

The title of the web page is here (Figure 3.9). It was constructed with native HTML.

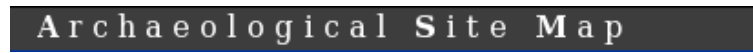


Figure 3.9: Title of the Mashup

3.5.2 Map Section of the Mashup

Google Maps API (Figure 3.10) is located in this section. The map of the selected area is also shown. When the page is loaded, it shows the map of Turkey with selections like zoom in, zoom out or sliding, the page can show different map areas.



Figure 3.10: Map section of the Mashup

3.5.3 Mini Map Section

This layer is also a part of Google Maps API. It shows (Figure 3.11) the mini look of the whole map.



Figure 3.11: Mini map section

3.5.4 Timeline Section

Timeline is located on the map section as a transparent visibility (Figure 3.12). It is built using SIMILE Timeline Library. It is used for showing the layer which depends on time on the map.



Figure 3.12: Timeline section

3.5.5 Query Criteria Section

The query area is divided into two section tabs. One of them is for geographic attributes in Figure 3.13; the other is for related, non-spatial attributes in Figure 3.14.

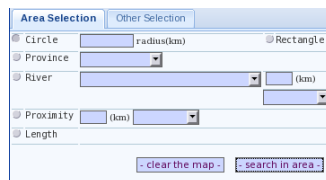


Figure 3.13: Area query criteria



Figure 3.14: Other query criteria

3.5.6 Result Area of the Mashup

The query results are shown in Figure 3.15.

Search Result			
id	name	distance (km)	geoObject
12	Hasankeyf	0	Dicle Nehri
24	Kultepe	1	Sarmisakli Cayi
9	Aspendos	1	Koprucay
7	Gordion	1	Sakarya Nehri
29	Sapinuwa	1	Salur Deresi

Figure 3.15: Result area of the Mashup

3.5.7 Layer Area

Figure 3.16, with the use of timeline, shows the layer. The layer transparency can be controlled on the map.

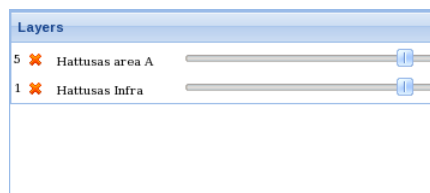


Figure 3.16: Layer area

3.5.8 Controls of the Mashup

Some map based controls such as zoom-in, zoom-out, switch off on the timeline, and export to pdf file are shown in Figure 3.17. In system development there will be more controls.



Figure 3.17: Controls of the Mashup

3.5.9 Information Panels of the Mashup

Accordion type info panels contain information of selected result. ExtJS accordion structure is used. Top panel contains photos of selected result areas. The other panels contain general information on excavation or archaeological site in Figure 3.18. Detail of the panels will be given in Section 4.3.3.3.

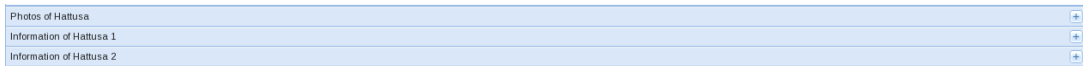


Figure 3.18: Information panels

3.6 SYSTEM DEVELOPMENT

The system design was presented in the previous section. In this section, development of the system will be explained. As known, the system architecture is a client server type archaeology mashup. The local server side of the system has Linux based OS. The client side can be any OS supporting web browser.

There is also a section which is not mentioned before, the developer section. The Developer side is also a part of this system. Without the developer side, of course, the system can not be built and can not be improved.

In this study all development of the system was made with open source software. During the development of the client side, the server side and also the developer side, open source codes or operating systems (OS) have been used.

In this chapter, firstly, infrastructure and software, that was used during the development will be explained. Afterwards, the server and client side development will be described.

3.6.1 Developer Side Development

On the developer side, Linux based Fedora 10 was selected as the OS. The selection was made because of similarities to the server side operating system. Therefore, the developer became

much more prone to the system and developed the codes without difficulty.

For writing the codes on the developer side, Quanta Plus Development editor was selected. It is said that, "Quanta Plus is a stable and feature rich web development environment" (Quanta Plus, KDE Web Dev Team). The use of Quanta is like other web developer programs. It has a text area and control buttons on top of it. It highlights, corrects and completes the codes while writing.

It has also many other helping properties for web development. Currently, it is running on Linux based systems. Figure 3.19 is a screen shot of Quanta Plus;

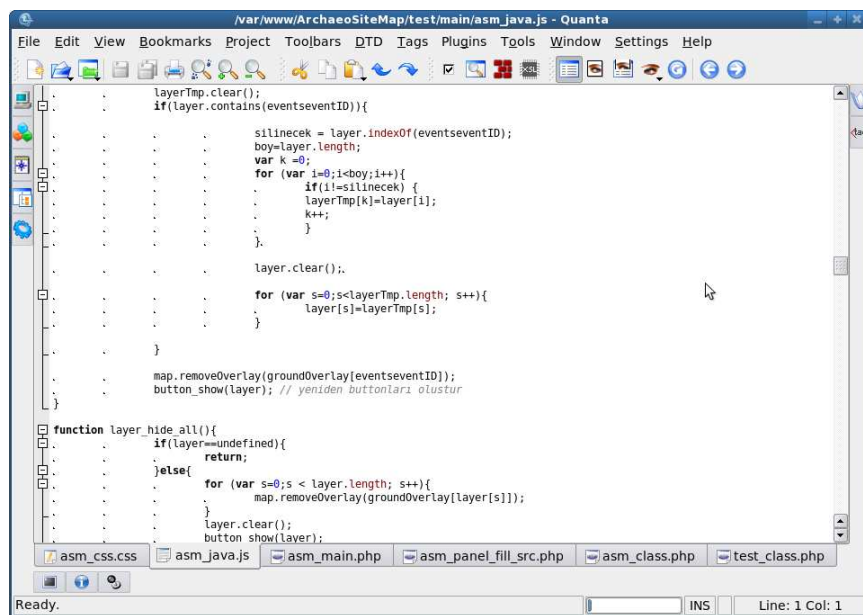


Figure 3.19: "Quanta Plus Web Development" tool

The codes, which were developed, were sent to the server with quanta ssh properties. So, development of codes were actually made on the developer side and saved to the remote system or server on the fly with ssh functions of quanta.

Similarly, database development was made similar to the codes. Although creating and storing of the database was made on the server side; development, creation of tables and assigning attributes were made on the developer side. Therefore, to reach the databases, two programs or tools were used. One of them is open source administration and development platform for PostgreSQL, pgAdmin (pgAdmin, PostgreSQL).

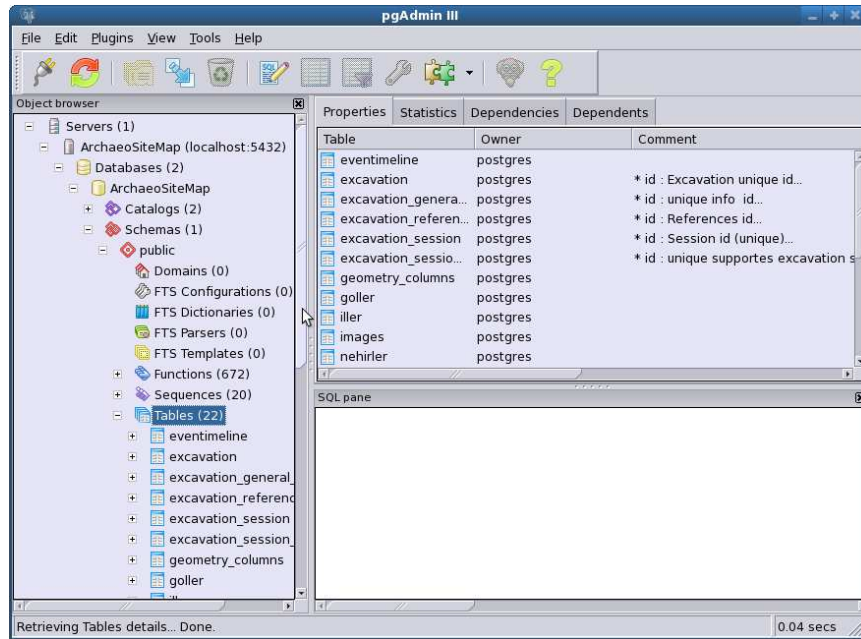


Figure 3.20: pgAdmin

In Figure 3.20, the screen shot of pgAdmin is shown. pgAdmin supports connection and management of PostgreSQL database. In the development phase, this tool was used many times for database processes.

The other tool is also an open source software, with the only difference that it is a web based database management system for PostgreSQL, phpPgAdmin (Figure 3.21)

All database development processes were managed by these two tools. Sometimes, when direct access to the database was needed, connection to the server was established and the database was intervened. Figure 3.22 represents this direct connection and intervening.

In the development phase, written codes for the client-side were debugged with Firebug. Firebug is an add-on extension for Mozilla Firefox. It allows the debugging, editing, and monitoring of the website's CSS, HTML, and JavaScript codes (Firebug, GetFirebug Web).

Firebug helped developing JavaScript codes as well as Ajax codes. Firebug screen shot is shown in Figure 3.23.

On the developer side, Dell 1501 Laptop was used. It has 2 GB ram, 100 GB hard-disk and network connection, both cable and wireless. Its operating system and other programs are

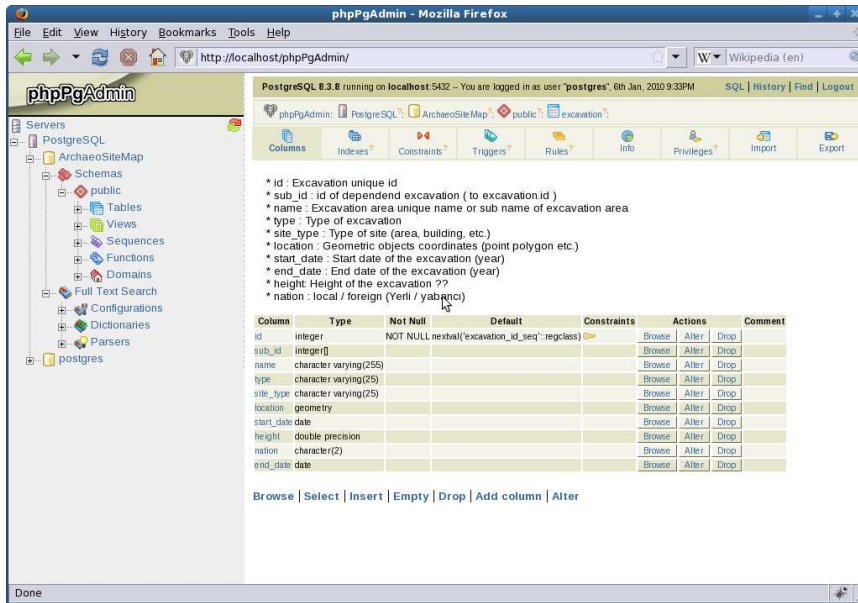


Figure 3.21: phpPgAdmin

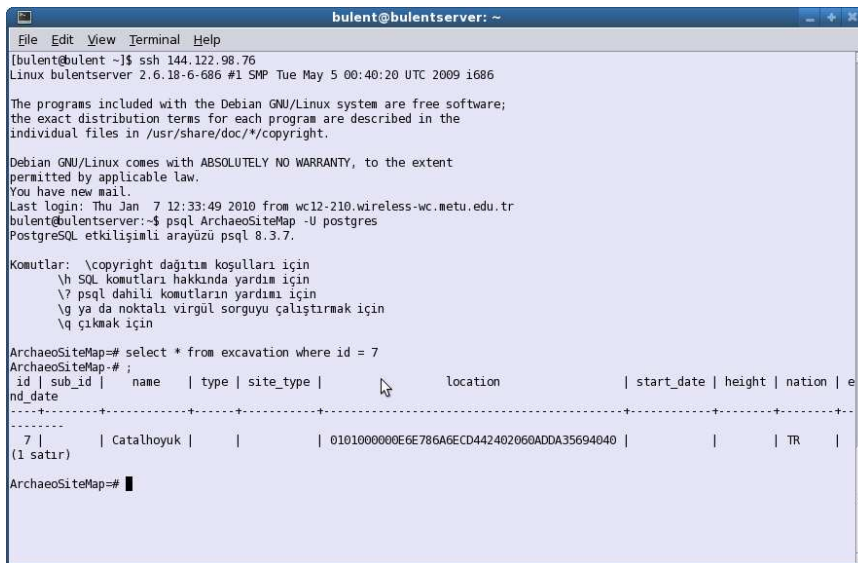


Figure 3.22: PostgreSQL console

working on this configuration efficiently.

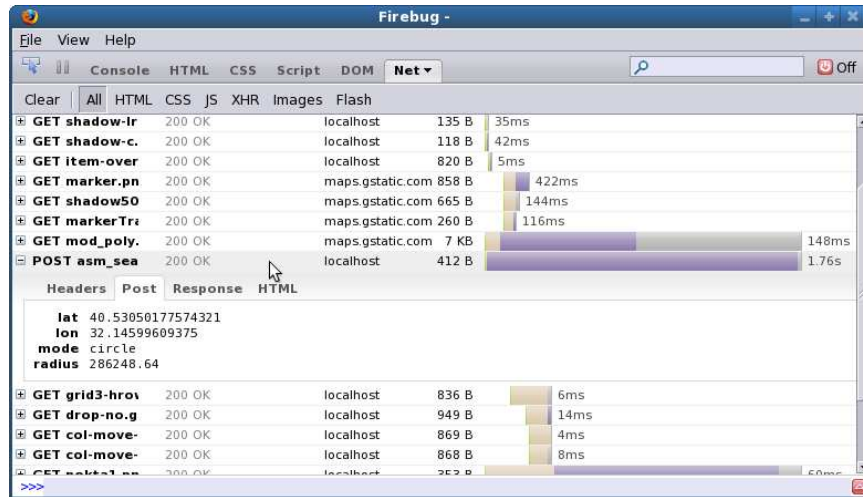


Figure 3.23: Firebug plugin for Firefox.

3.6.2 Client Side Development

On the client side, any operating system that has capability to run a web browser that supports web 2.0 standard is sufficient. However, due to the support of web standards (Web Browser Standards Supports, Web Devout), Firefox is preferred. For using Google Maps API, Google announced system requirements for supporting browsers (Google Code, Google). These are below;

- IE 6.0+
- Firefox 2.0+
- Safari 3.1+
- Google Chrome

As a hardware on the client side, a wide and full color support screen resolution (1440x900) is preferable. In addition to this, any connection components or hardware is a must for reaching servers over the Internet. Today's computers are supporting these properties. Of course, much more ram and disk capacities are also preferable.

3.6.3 Server Side Development

The server is hosted in the Informatics Institute, METU. It is benefiting from the infrastructures of Informatics Institute and METU networks. As a hardware, it has a Pentium based processor with 1.3GHz processor speed, 512 MB memory, 80 GB harddisk and network interface card.

The operating system it is working on, is Linux based Debian version 5.0. Therefore, it supports multitasking jobs and heavy requests incoming from clients. There are two important services that are working as a daemon on this server. One of them is a web-server, the other one is a relational and spatial database server.

As a web server, Apache has been installed. Apache has been developed and maintained by an open community of developers under the auspices of the Apache Software Foundation (Apache Web Server, Wikipedia).

Apache Web Server is working with a PHP module. PHP, here, undertake pre-processing for incoming requests to the Apache. Php codes which are embedded in HTML document are interpreted by the Apache web server with PHP module in order to produce dynamic web pages. Therefore, it produces web page documents on the fly. It is also used for connecting to the database, executing queries and getting the results. Sending the results of queries to the clients as a web page document is also done with PHP.

Web server installation is very simple, because when an operating system is installed, it is installed with it by default. However, if it was ignored during the installation of an OS, it is possible to install it later. In Linux, although there are a few differences in the installation, many of them are alike.

When running "apt-get install apache" command as a root, the Apache Web Server will be installed and is ready to reply to incoming requests. With Apache, php package may also be installed. PHP is installed with a similar command "apt-get install php". So, Apache and its PHP module are ready to work.

The other service is the database server. PostgreSQL was installed on the server as the database server. It has also spatial feature capability. In order to get spatial feature, PostGIS, a spatial extension for PostgreSQL database was installed as an add-on. Thus, our PostgreSQL

database became a spatial relational database. In the previous chapter, spatial database was explained in detail.

The installation of a database is similar to the installation of a web server. When the OS is being installed, it is possible to install a database server without much effort. However it is also possible to install it after the installation of an OS. This can be done with a similar command. As a root, "apt-get install postgresql" command does the installation.

In the same way, installation of PostGIS extension is done by a command line, like "apt-get install PostGIS". However, this does not mean that the created database is able to work with PostGIS. In order to enable PostGIS, spatial properties, in the database, the following steps have to be taken (PostGis-1.5 OSVN Manual, Refrations Research - Internal). All these are applied with a command line on the server side.

Create a database on PostgreSQL.

```
# createdb [yourdatabase]
```

Enable the PL/pgSQL language in the database.

```
# createlang plpgsql [yourdatabase]
```

Load the PostGIS object and function definitions into the database by loading the postgis.sql definitions file.

```
# psql -d [yourdatabase] -f postgis.sql
```

For a complete set of EPSG coordinate system definition identifiers, also load the spatial_ref_sys.sql definitions file.

```
# psql -d [yourdatabase] -f spatial_ref_sys.sql.
```

Load comments.sql into the database adding comments to the PostGIS functions is desired.

```
# psql -d [yourdatabase] -f postgis_comments.sql
```

After all these steps, the database is ready to be used with spatial capabilities.

Finally, in order to get remote access and maintain the server, ssh daemon is working in the server. Ssh daemon provides remote access with secure connection. The developer reaches the server with this ssh connection. In Linux, "ssh" command, in windows some ssh programs are used to initiate the ssh daemon. This remote access provides the direct intervention to the

system, database or web server daemon. Such an access is shown in the previous Figure 3.22.

For the backup operation of the files, *Rsync* program was used. *Rsync* is scheduled with crontab and executed at specified times. So, the backup process is executed automatically at desired times.

CHAPTER 4

TEST CASE AND IMPLEMENTATION

4.1 SYSTEM OVERVIEW

The main purpose of the mashup's design is to provide easy usage with a graphical user interface (GUI). 'Extjs' framework widgets library was used to achieve this. It supports a graphical background interface for the mashup. All the interface was prepared with this rule or purpose. The GUI is divided into some sections in itself. These sections and their relative locations are explained in here;

The map section is located at the top left side of the page. The search section is located on the right side of the map. Also, search results and layer controls are located on the right side of the map, below the search section. In the previous Chapter 3, the locations of the sections were shown. The web page is shown in Figure 4.1;

4.2 USAGE OF THE SYSTEM

In order to use the system, one has to be familiar with web based GUI and Google Maps. In addition, basic computer knowledge and being able to use the Internet are requirements.

The system rules could be kept simple. It contains these basic parts: search, select, look and examine. The flowchart of the system is shown in Figure 4.2. The user follows the levels from 1 to 2 and progresses down in the figure from T1 to T2. These levels, which represent the user's next choice, depend on the previous level and "T" represents time.

As shown in Figure 4.2, within Level 1, the processes are independent from each other. The

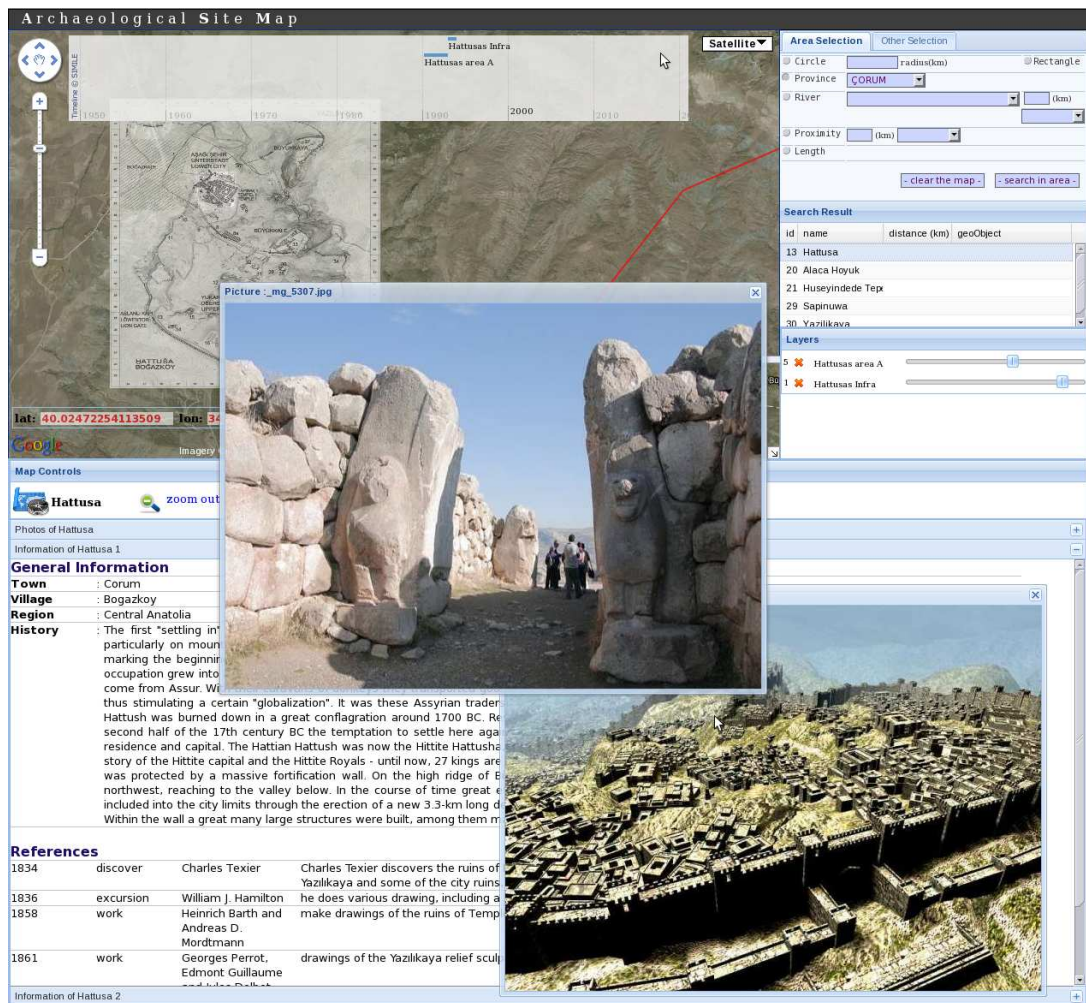


Figure 4.1: Screen shot of Mashup

user can search, view and select the results in Level 1 any time. Level 2 processes, however, are dependent on Level 1. After the selection of the results, Level 2 processes appears. In Level 2 processes, the user can make a choice about the result's detail. He can either select the close view or export the result as a pdf file. The "Zoom out" selection, in this level, is working after "zoom in", i.e. "zoom out" is ineffective, before "zoom in". Level 3 depends on Level 2 timeline objects. It is shown after the selection of timeline objects. After the selection, the layer control appears on the control panel and the layer, which is controlled by the user, appears on the map in the specific location. Here, all dependencies are related to time.

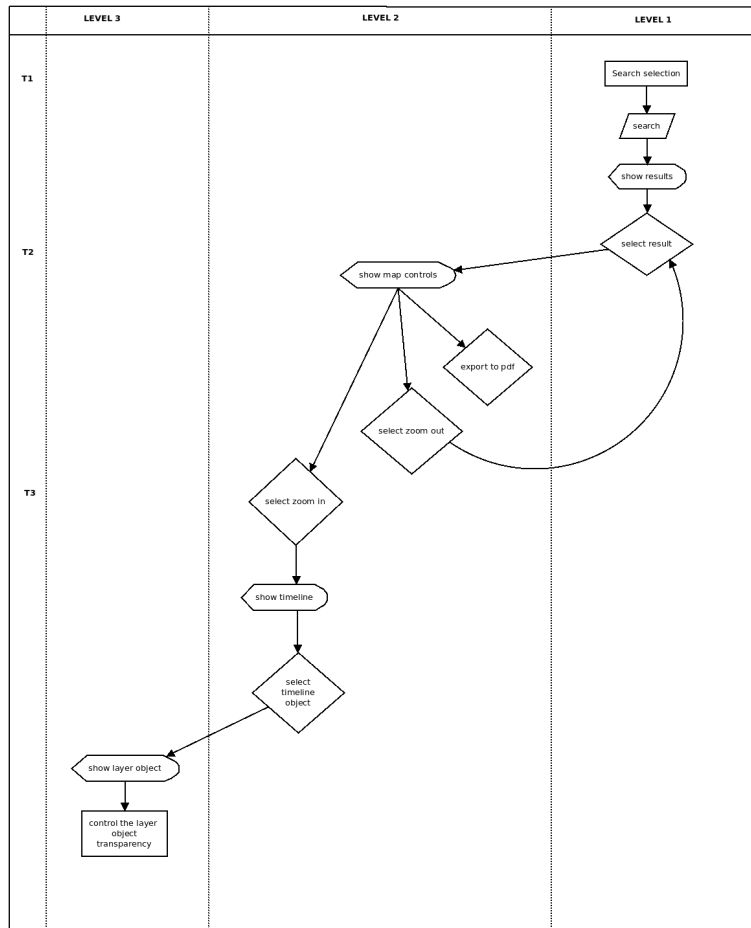


Figure 4.2: Flow chart of the Mashup

4.3 TEST CASES QUERIES

The system has the capability of making several queries and showing these. The most important of them are spatial queries. The others are phenomenal, relational, and semantic queries that are related with each other.

4.3.1 Spatial Queries

The system, owing to its structure, supports spatial queries. These queries are replied to by the PostGIS database. These are mainly distance, proximity and topological queries.

4.3.1.1 Distance (Buffer) From a Center

Distance queries deal with the object's proximity to the selected feature. They check if the object is within a certain distance to the selected feature. There are several distance queries in this system.

On the map, a point is marked with specific radius which is a buffer area around it. This radius is either selected with a second marked point or entered in a text-box before any point marking. The circle that is created by selecting the points appears on the map. The search is done within this circle. Any point in the circle is exposed by making a spatial query.

The query is replied to by PostGIS database. After that, the results are shown on the map and appear in the result section. In Figure 4.3, a query and its results are shown;

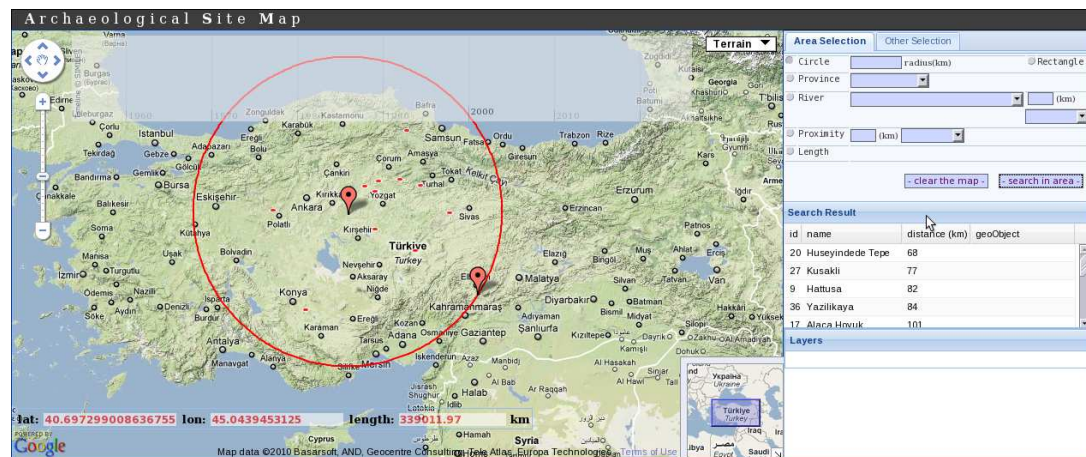


Figure 4.3: Distance from a center

In Figure 4.3, the center point is marked between Ankara and Yozgat. The radius point is marked near Kahramanmaraş. Therefore, the buffer size is a radius from the center point to the second marked point. When submitting the search button, the archaeological sites (points) and Hattusa appear near Yozgat within the buffer.

4.3.1.2 Distance (Buffer) From an Object

In this type query, rivers were selected as geometry objects. Similar to "Distance From a Center" it can be thought as if rivers are points. The remaining thought is the same as "Distance

(Buffer) From a Center”

In query ”Area” section, almost all rivers are defined and selectable by combo-box. After the selection, the selected river appears on the map. The river data comes from the database as a (multi)polygon or (multi)line-string and is shown on the map. In order to show these, Google Maps API provides an easy method. They are converted encoded structures. Therefore the client browser can easily show these without consuming memory resources. Only necessary points are shown on the polygon or drawing at a suitable zoom level. Unnecessary points are not shown or drawn to save memory of the browser. When zoom level decreases, i.e. zoomed-in, the detail of the polygon or the points are shown. The details of these feature are explained in Appendix B .

Again, after the selection of the river, the distance or buffered region are asked. This is entered in the text-box next to the river combo-box. The number entered for the distance between the river and the point has to be in km. The search is made within this buffered area, along the river within this buffered region. If there are any points/objects in this region, the results are shown on the map and in the result section as a list. In the same way, instead of entering the buffered size, there is a combo-box for the selection of some buffered region. Selections such as ”between 0 to 5 km” as a so close, ”4 to 15 km” as a close, or ”13 to 50 km” as a near can be made. These also represent the buffered area region with a specific distance from the river.



Figure 4.4: Distance from a river (in 35km)

Some examples and results are shown in Figure 4.4 and 4.5. In Figure 4.4, the region search within 35 km of ”Kızılırmak” river is shown. In Figure 4.5, the same river with a buffered

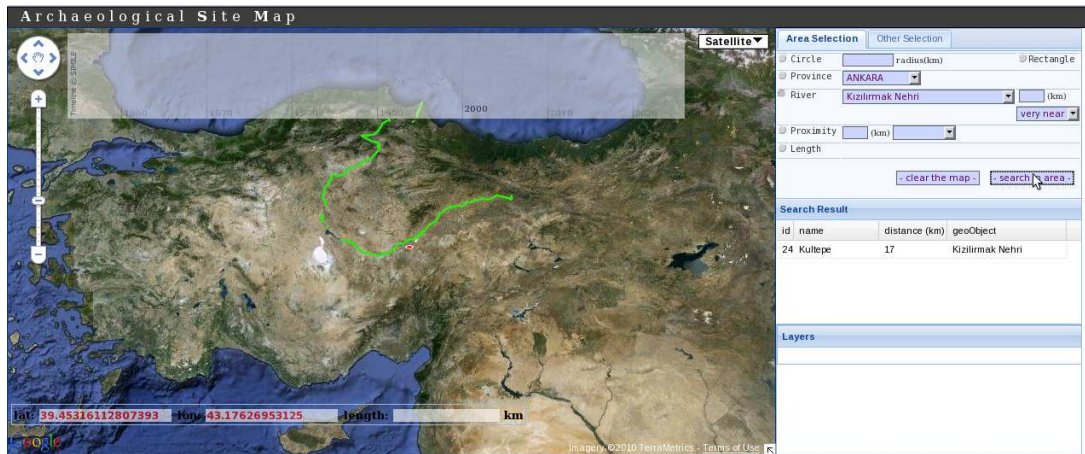


Figure 4.5: Distance from a river (very-near)

region (very near) results are shown.

4.3.1.3 Object Proximity

The last distance query type is proximity. Again, the same structure was used for this query type. However, in this query, relations between the geometry objects (polygons and points) were examined. For example; "search and bring the results of the points which are close to the given river within 1 km". In this query, all archaeological sites (points) are compared with the rivers. The query criteria are searched and if there are any matches, they are shown on the map.

Figure 4.6 shows the example of such a query and its results. In the result list, according to the selected criteria, the point of an archaeological site, the distance of it from the river and river names appear.

In the same way, proximity of any point to any river can be searched by selecting from the combo-box section. Likewise, the archaeological locations which are "very-near" can be searched for as seen in Figure 4.7.

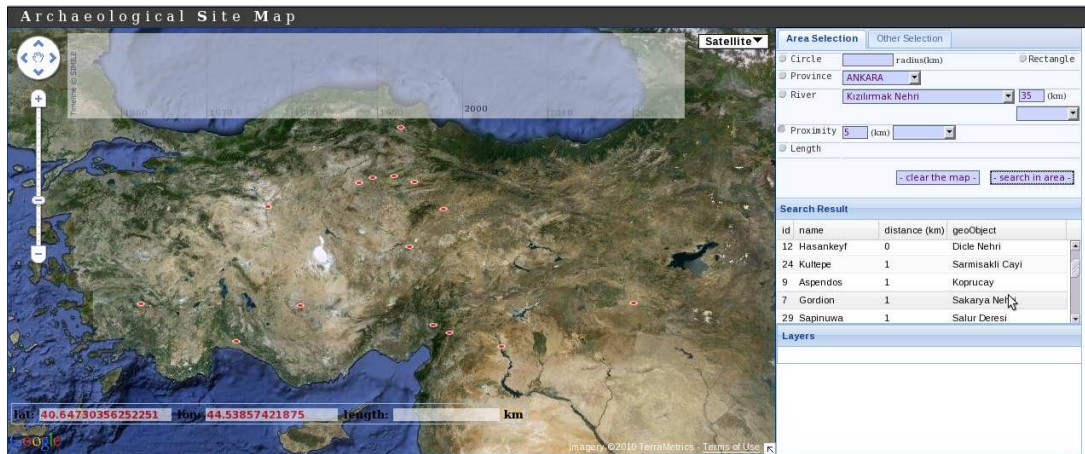


Figure 4.6: Proximity (in 5km)

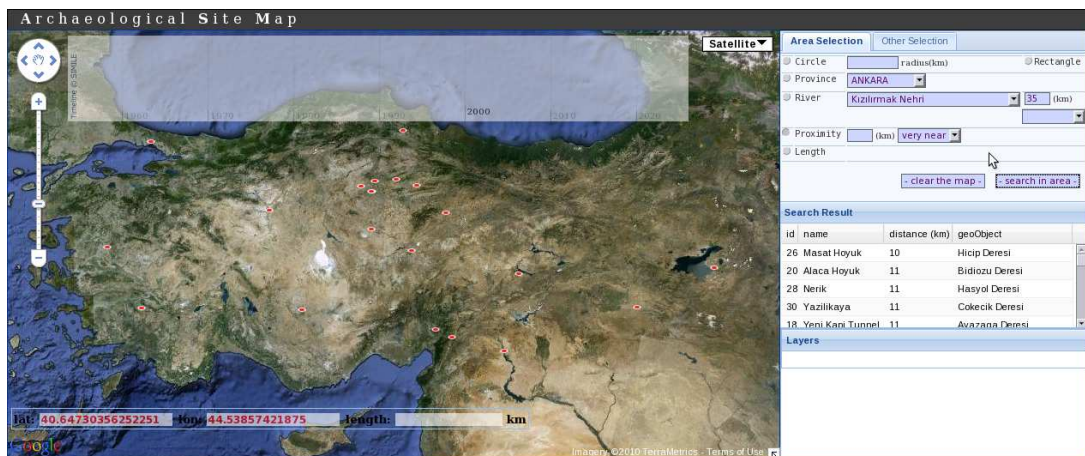


Figure 4.7: Proximity (very near)

4.3.1.4 Point in Polygon (Rectangular)

Topological queries are concerned with the objects' locations and relationship among them. They provides answer to questions like; "are there any points (or point) in the selected area or polygon". Here, a few examples of topological queries in this system are illustrated.

The first topological query is to find a point in a rectangular area. The rectangular area is selected by marking two points, which are the north-west and south-east corners of the rectangle. After marking these points, the area appears on the map.

When the search is submitted, the system seeks the archaeological sites (a point or points) in the marked area and results appear on the map and in the result list section.

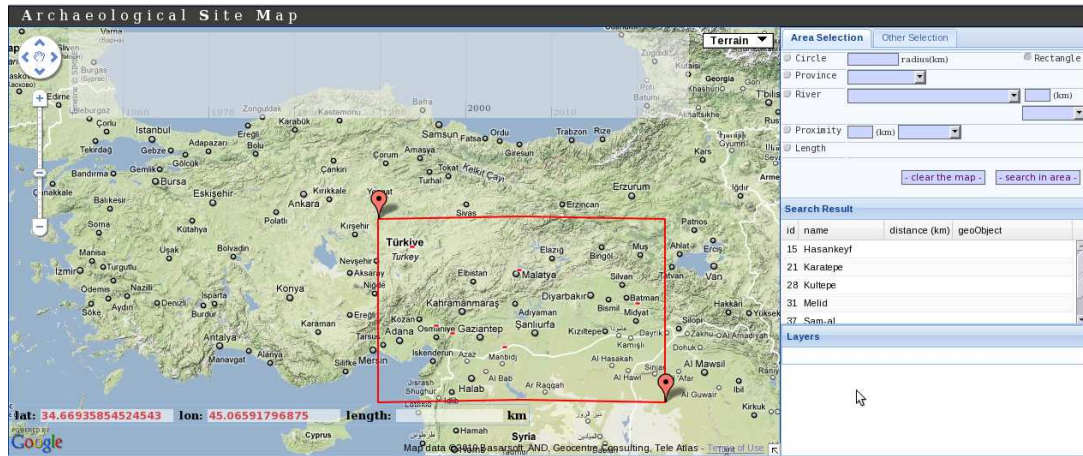


Figure 4.8: Points in a rectangular area

In Figure 4.8, the rectangular polygon search and results are shown.

4.3.1.5 Point in Polygon (Province)

The second example of topological query is to find a point in a complex polygon. Here, the polygons are the borders of the provinces of Turkey. In a query section, a province is selected in the combo-box. When the province is selected, the borders of it appears on the map. River polygons and province polygons use the same decoding algorithm, explained in Appendix B. After the province border appears, any archaeological site can be searched inside it by submitting the query. The query executed on the spatial database, PostGIS, and the results are shown on the map and in the result list section.

Figure 4.9 shows an example of this. The selected province is "Çorum" and the results appear inside Çorum's borders and in the result section list.

4.3.2 Attribute, Relational and Semantic Queries

The last type of queries can be collected under the same title. These are related to each other. Each of them will be explained individually;

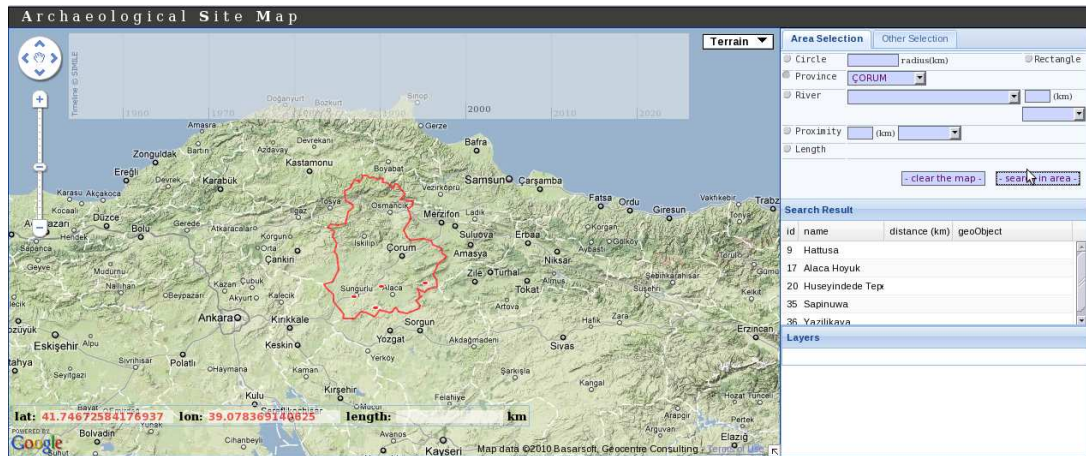


Figure 4.9: Points in a province

Attribute queries are only concerned with non-spatial data such as name queries of provinces etc. No geographical objects are queried in these.

Relational queries are concerned with more than one table. These tables can include both spatial or non-spatial data. In this type of queries, a table may contain the geographic locations, another table may contain the period of these locations. Query criteria must satisfy both tables. The tables are connected to each other with primary key and foreign keys. The table connections are shown in the Appendix C.

Similarly, semantic queries are composed of both; attribute and relational queries. However, these are a bit more complex and aim to move specific targets or answers. These type of queries are executed on more than one table and results give answers for a predetermined detailed questions.

In this study, attribute and relational queries can be made with "Other Selection" tabs, in the query criteria section. Here, the periods of the archaeological sites can be selected. These selection tabs are related to the first "Area Selection" tabs. When an area is selected in the "Area Selection" criteria, the selected periods of archaeological sites are searched within this selected area. So, the area and periods are used together in the query. Both of them are executed on two different tables that are related with primary and foreign keys. This is an example of relational queries and can be applied to each area query.

Some queries and results are shown in figures;

Figure 4.10 shows an example for a point with a distance (circle) and belonging to the Neolithic period.

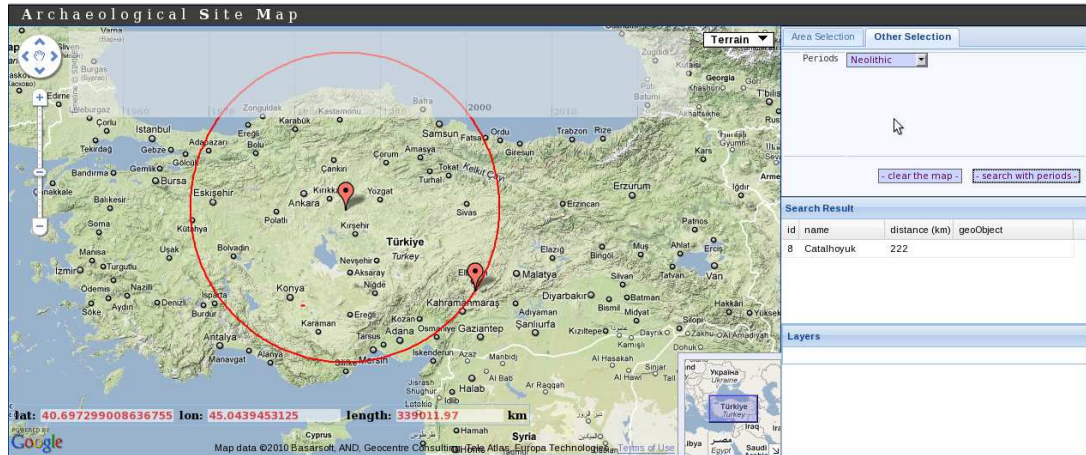


Figure 4.10: Distance from a center belonging to the Neolithic period

Figure 4.11 shows an example for a point in rectangular polygon belonging to the Hittite period.

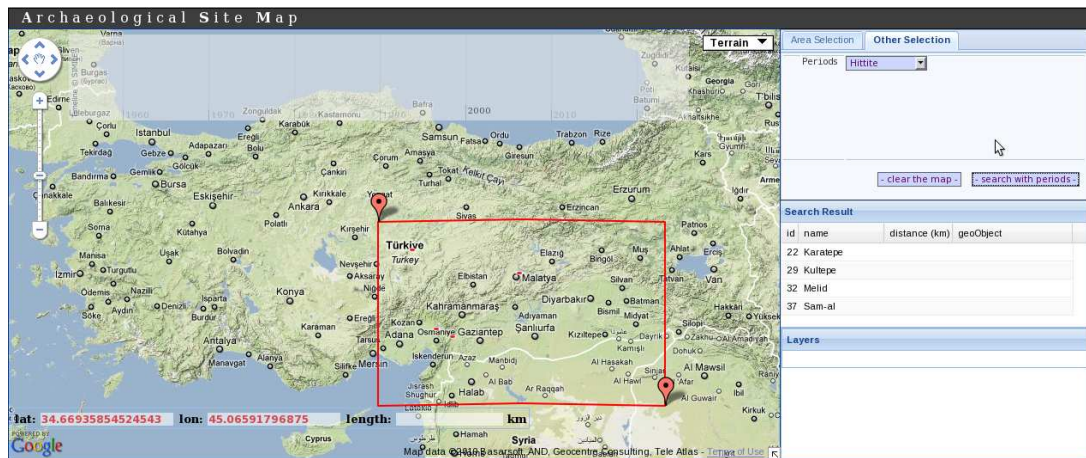


Figure 4.11: Points in a rectangular area belonging to the Hittite period

Figure 4.12 shows an example for a point within the borders (polygon) of a province in belonging to the Phrygian period.

Figure 4.13 shows an example for an object with distance (rivers) and belonging to the Bronze age period.

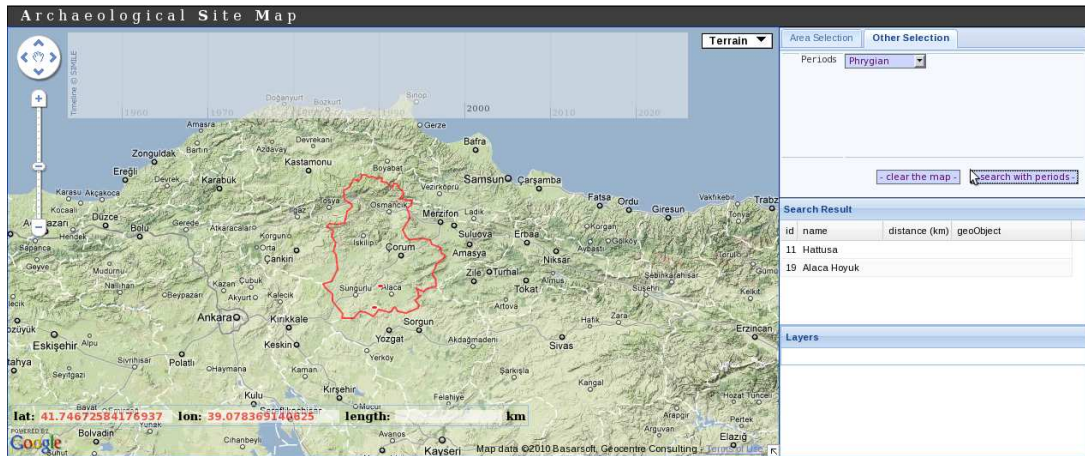


Figure 4.12: Points in a province with Phrygian period

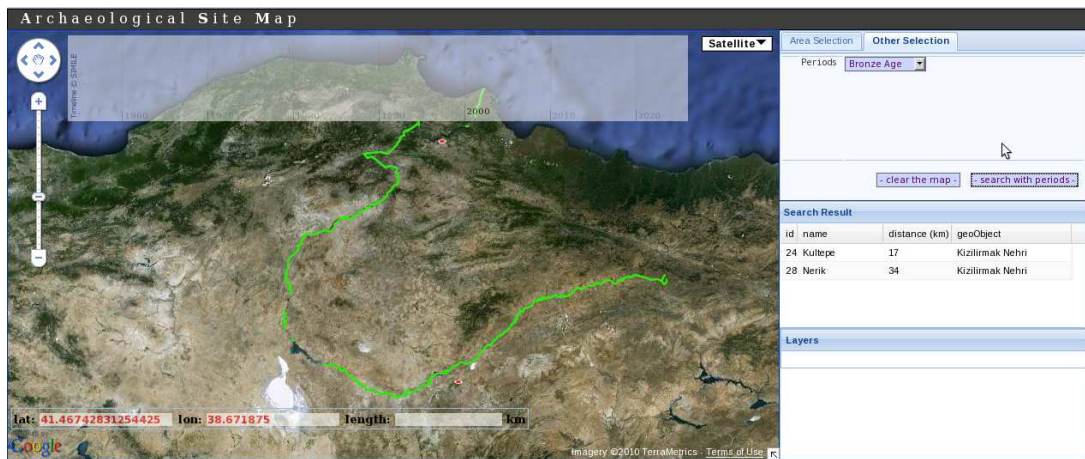


Figure 4.13: Distance from a river belonging to the Bronze age period

Figure 4.14 shows an example for a proximity query belonging to the Neolithic period

4.3.3 Result & Controls of Queries for Hattusa

The result of the queries are shown on the map. The other information or controls which are related to the result are shown in suitable places on the mashup. These are map controls, timeline controls, layer controls and info panels which include photos and documents.

Photos, information or map controls are shown below the map. Map controls are located

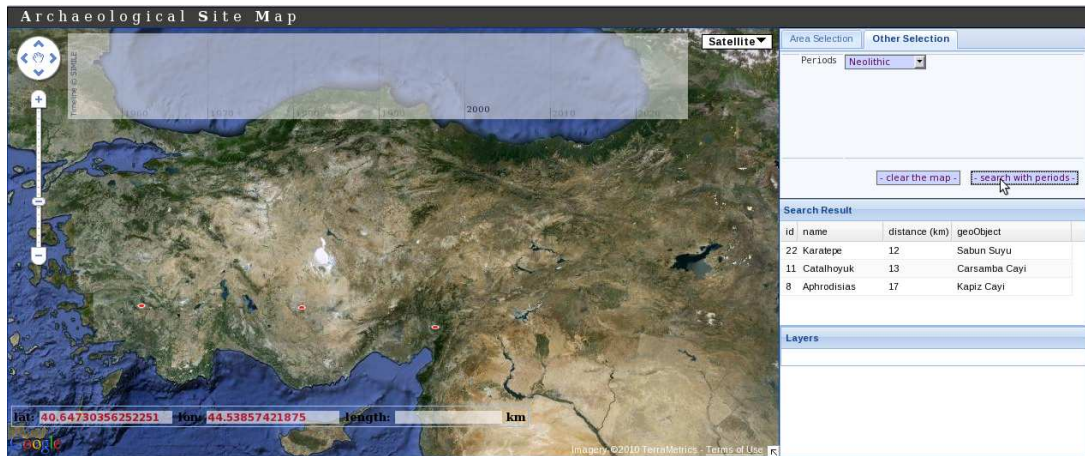







Figure 4.14: Proximity with Neolithic period

below the map, timeline control is located on top of the map and timeline related layer control is located below the search results side.

4.3.3.1 Map Controls

Table 4.1 shows the map control symbols and meanings.

Table 4.1: Map controls

 Hattusa	<p>map title, the location of control names appear in that title.</p>
 zoom in	<p>zoom in, the selected result is zooming in when it is used.</p>
 zoom out	<p>zoom out, the selected result is zooming out. This option can only be used after the zoom in control is used.</p>
 timeline	<p>timeline, the timeline module is closed and opened with this.</p>
 pdf export	<p>pdf export, using this, the information about that selected result information is exported as a pdf file.</p>

If Hattusa is selected in Figure 4.15, "zoom-in", the near satellite view appears on map section in Figure 4.16.

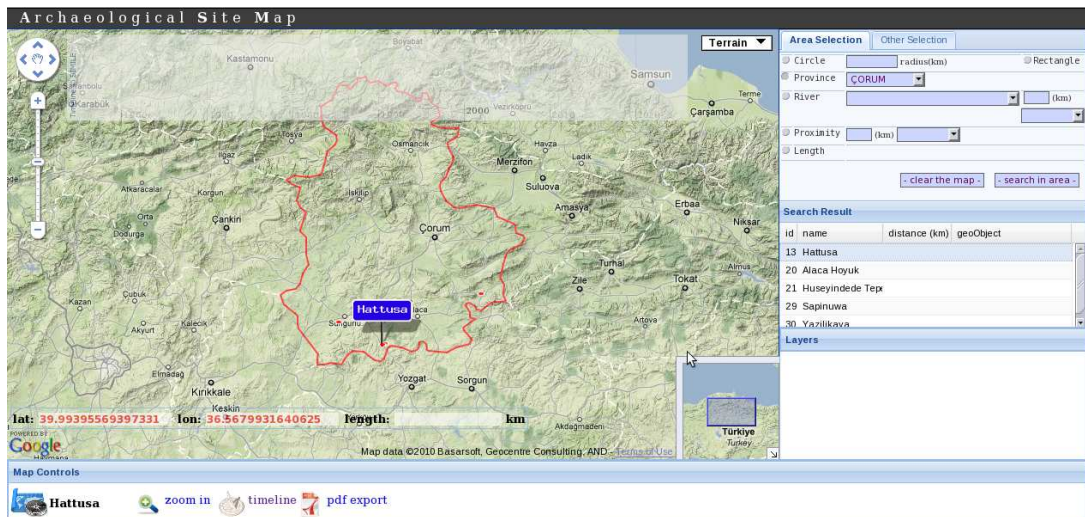


Figure 4.15: Zoom-out Hattusa

In a similar way, "zoom-out"'s duty is to restore the map's previous view during the search phase. "topological query and result" or "buffer-queries and results" figures can be seen in

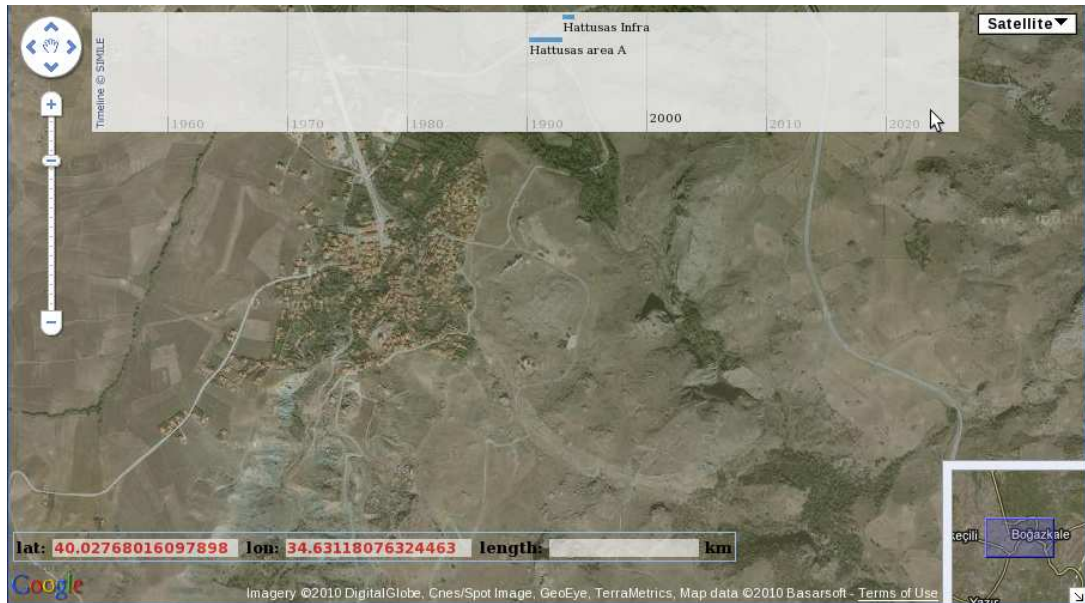


Figure 4.16: Zoom-in Hattusa

the zoom-out phase.

The control "Pdf export" exports all the information about the selected location "Hattusa" pdf file format in Figure 4.17. This is part of the web-service. It can be considered as an interactive web service. It also provides the end-user with the opportunity to easily document the result.


Finally, "timeline" control's duty is to close or open the timeline section, which is located on the top side of the map. Its only function is to provide a wide view area on the map. However, the objects on the timeline, have some functions related to the maps.

4.3.3.2 Timeline Controls

Timeline appears when one of the search results is selected. The objects (or text) on it are related to the time and layer.

In Figure 4.16, the timeline and its objects are shown on the map. The objects (or text) on the timeline are located with respect to time. Each object on it represents one layer, which is located on a specific area in time. When the object is selected, the layer appears on the

Hattusa



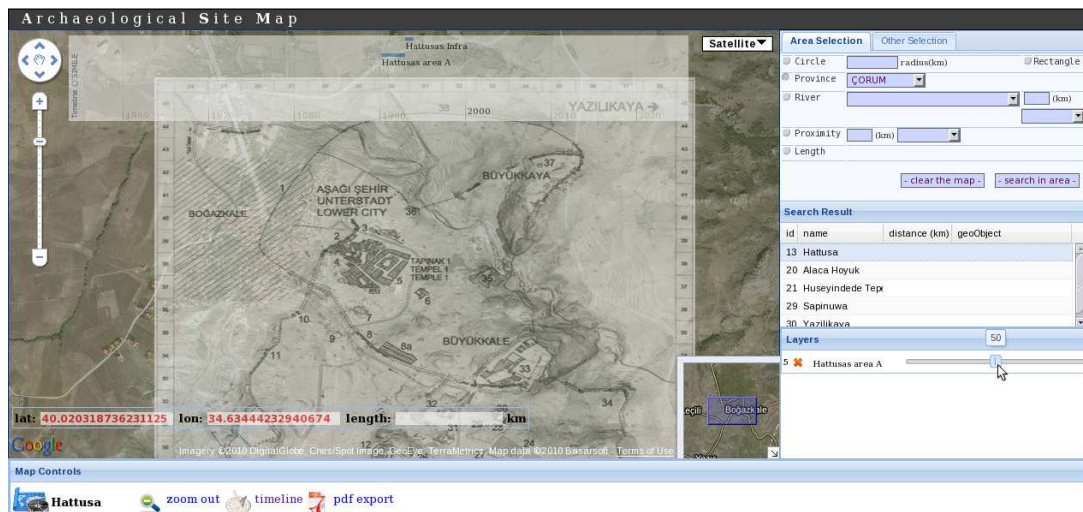
General Information

Town : Corum
Village : Boğazköy
Region : Central Anatolia
History : The first "settling in" around Boğazköy took place in the 6th millennium BC during the Chalcolithic period, when small widely scattered hamlets appeared most particularly on mountain slopes and rocky outcroppings. Late in the 3rd millennium BC, towards the end of the Early Bronze Age, a Hattian settlement developed, marking the beginning of continuous occupation at the site. The Hattians, native Anatolians, called their town Hattush. During the Middle Bronze Age the Hattian occupation grew into a city of such significance that a Karum was established here in the 19th and 18th centuries BC - a trading post of Assyrian merchants who had come from Assur. With their caravans, of donkeys they transported goods to and from Mesopotamia, and along their route they also dealt in local Anatolian products, thus stimulating a certain "globalization". It was these Assyrian traders who first introduced writing to Anatolia. The ruins excavated demonstrate that the city of Hattush was burned down in a great conflagration around 1700 BC. Responsible was King Arlita from Kussar, who also put a curse on the site. But already by the second half of the 17th century BC the temptation to settle here again had obviously become overwhelming, for a Hittite king had indeed chosen the site as his residence and capital. The Hattian Hattush was now the Hittite Hattusha, and the king took the name of Hattushili, or "one from Hattusha." This is the beginning of the story of the Hittite capital and the Hittite Royals - until now, 27 kings are known by their names. The Old Hittite city comprised an area of almost 1 square kilometer; it was protected by a massive fortification wall. On the high ridge of Büyükkale was the residence of the Great King, and the city lay on the slope below to the northwest, reaching to the valley below. In the course of time great effort was spent on the development of the Upper City. This area south of the Old City was included into the city limits through the erection of a new, 3.3-km long defense wall with several monumental gates, thus bringing the size of the city to 182 hectares. Within the wall a great many large structures were built, among them many temples - houses for "the Thousand Gods of the Hatti Land".

References			
1834	discover	Charles Texier	Charles Texier discovers the ruins of Hattusha. Believing he has found Perla, a city of the Medes, he makes drawings of reliefs at Yazılıkaya and some of the city ruins, and prepares a rough plan of the city.
1836	excursion	William J. Hamilton	he does various drawing, including a plan of Temple I. He is convinced that the ruins represent the Galatian/Roman city Tavium.
1858	work	Heinrich Barth and Andreas D. Mordmann	make drawings of the ruins of Temple I and have the reliefs in the smaller Chamber B at Yazılıkaya cleared.
1861	work	Georges Perrot, Edmond Guillaume and Jules Delbet	drawings of the Yazılıkaya relief sculpture, and publish the first photographs of Yazılıkaya, Yesicekale and the Nisanjay inscription.
1864	work	Henry J. van Lennep	drawings of Yazılıkaya
1993	excavation	Jürgen Seeher	Nearly all the remains of the Hittite Royal Citadel on Büyükkale have been cleared, and large-scale excavation has exposed wide areas of the settlement in the Lower City, the Great Temple, the temple precinct in the Upper City and its surroundings, as well as on the high spur of Büyükkaya. Excavation on a smaller scale has been carried out in various other locations within the city and in the immediate surroundings, as well as in the rock sanctuary of Yazılıkaya.

Figure 4.17: Pdf exports of Hattusa

map and the "transparency" control section appears on the right side of the map. The control mechanism becomes a slide bar. The transparency of the layer is controlled by this slide bar.



The screenshot shows a web-based archaeological site map. The main map area displays a topographic view of the Hattusa site with various landmarks labeled, including ASACI SEHIR, UNTERSTADT, LOWER CITY, BOGAZKALE, BOYUKKALE, and YAZILIKAYA. A search panel on the right side of the map is active, showing search criteria and results. The search criteria include: Area Selection (Circle, Rectangle), Province (CORUM), River, Proximity (km), and Length. The search results list several sites, with '30 Yazilikaya' selected. Below the search results, there is a 'Layers' section with a slide bar for 'Hattusas area A' set to 50% transparency. The map interface also includes a 'Map Controls' section with buttons for 'Hattusa', 'zoom out', 'timeline', and 'pdf export'.

Figure 4.18: Timeline and layer of Hattusa

In Figure 4.18, the drawing is related to time and located on Hattusa as a layer. The transparency of the figure is controlled by the "Layers" section on the right-bottom, and adjusted to 50. The timeline, map and layer control relations are defined in Figure 4.18. The working principles of these relations can be seen.

Many layers that are related to time can be shown on the map using Timeline and the transparency of each of them is controlled by the Layers section. This gives the advantage of being able to trace the changes within time and mark some areas as a layer for classifying the locations.

4.3.3.3 Accordion Type Information Windows

When a result is selected, the information related with this also comes to the accordion type windows. First part of the window shows the photos of the selected result(s). When user clicks the mini photo, a big photo opens in a pop-up window. The other part of the accordion gives some information about the archaeological site such as general information, references in time, related publications and information about session supporters.

A screen capture of that part is shown in Figures 4.19, 4.20 and 4.21.

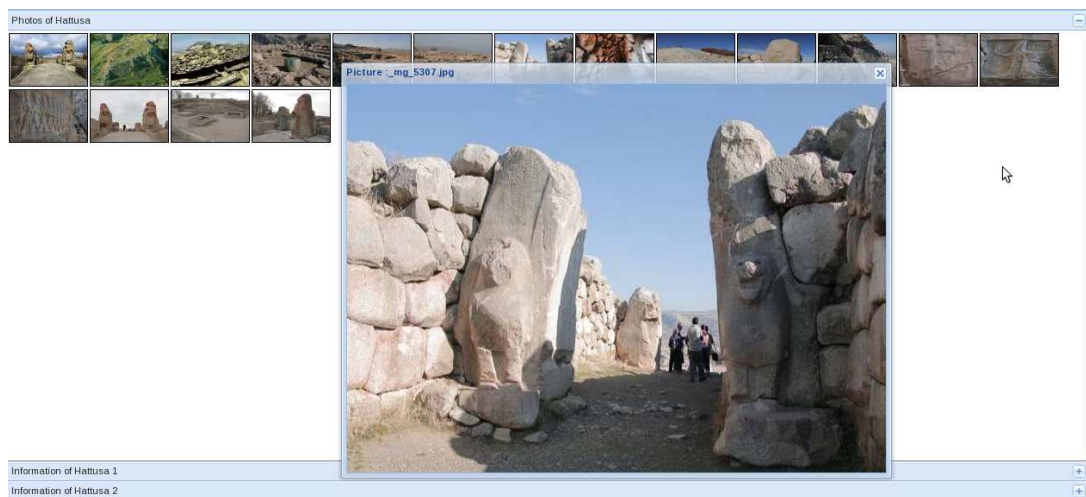


Figure 4.19: Photo panels

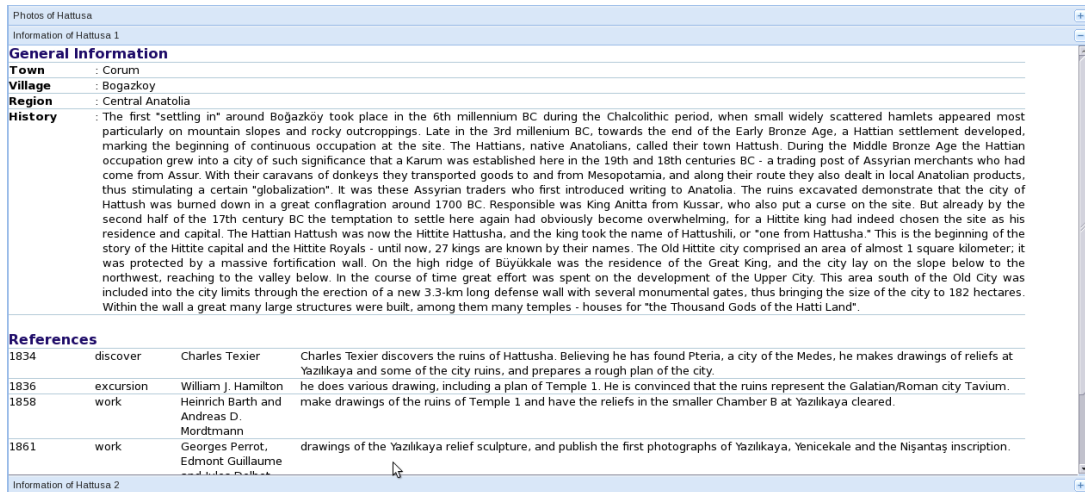


Figure 4.20: Info panel 1

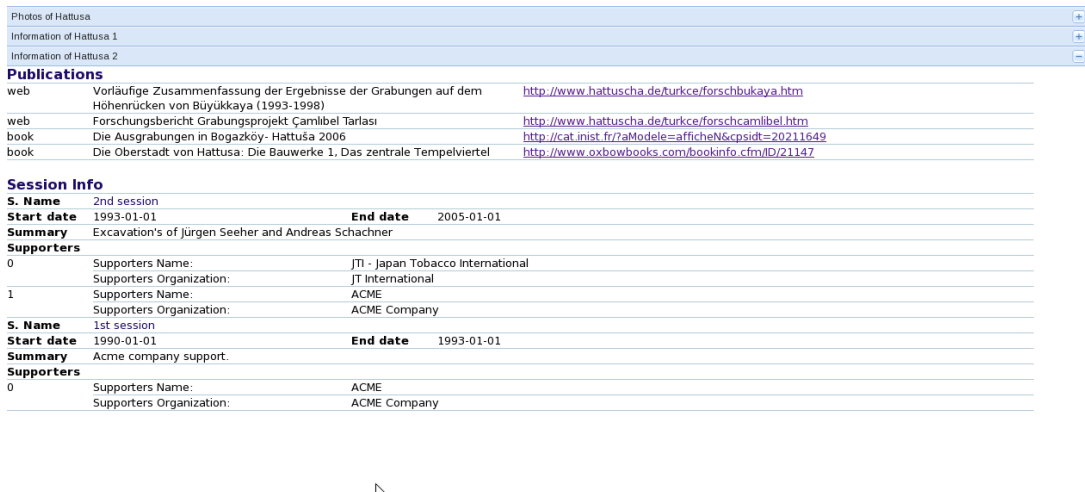


Figure 4.21: Info panel 2

4.4 SUMMARY

The mashup system works as intended. As a future work, some functions will be added. However, like any computer program, it may contain some "bug"s and therefore may need to be debugged as these are discovered. Better algorithms in codes may also be used so that the mashup's performance improves.

Computer programs never stay unchanged. They are always developed to get better than

before; so will the mashup. The performance and other requirements will become apparent during the usage of the system. As a web-based GIS, the possibilities of the program are almost infinite. The development of it will continue.

CHAPTER 5

CONCLUSION

With the capabilities that today's technologies offer, GIS is being used in many areas by government bodies and institutes. GIS has been too complex and expensive to be used by ordinary users until recently. In an addition, these applications were installed on singular systems that were not open to public for sharing. However, as a web based application, this technology has been rapidly spreading out in many areas and attracts more users. Archeology has become one of these web based applications and is already widely being used by the public.

Turkey is a real treasure in terms of archaeological sites. There are thousands of historical remains and archaeological sites spread throughout the country. In many of them, surface research and excavations work is being carried out and yet many others are waiting to be excavated. Although there are numerous publications on these excavations it is not easy to search and reach these sources or even basic information on these.

A web based GIS can be used to facilitate the search for locations of archaeological sites, get information on the work that has been done or is being done, querying these and sharing the information. In this respect, this study intends to contribute to archaeology in a scientific way.

At the same time, this study can be a base structure and an idea for similar systems that will be constructed in the future.

5.1 ADVANTAGES

The mashup that was constructed has many advantages, because it is a web based application. It does not depend on any client side platform to work. It can work on browsers which are independent of the operating systems and structures. This means that it can be used by most of today's computers. Using such an independent system which works on almost all computers is a real advantage making it accessible to researchers around the globe.

It is not necessary to install any add-on or supporting programs on a web browser or computer. Therefore, there is no need to purchase such programs. This in turn is an advantage in terms of budget and installation time for the end user.

Changing, updating or improving the structure of the mashup or the database is easier than with the non-client server structure. There is no need to do it for each client. All clients who are using this application can benefit from all kind of changes made on the server side. This is an advantage in management over the one center systems.

Web based systems have a structure that users are familiar with. Users spend the time to learn how to use the structure, which is much easier than to learn complex programs. In this respect, the system was kept quite simple. So, it makes it possible that the application is used by people of all levels.

Finally, this system is based on sharing information as a mashup. As a purpose, sharing information appears to be the most important advantage of it.

5.2 DISADVANTAGES

Although, it has many advantages, it also has some disadvantages caused by the structure of the system. The most important of them is that it is necessary to have the Internet connection. In client-server structures, there must be a connection between the components. This is provided in some way by either the Internet or another connection medium. If this connection is not established or breaks for some reason, the system becomes totally ineffective.

In the same way, there is a dependency on the server side. If there is a problem on the servers which are being used, the system collapses. The only way to overcome this is to create

more than one distribution server structure, route all connections to the other servers when one of them becomes unavailable. This mashup system does not support such a distributed architecture. However, such a structure could be built as a future work.

For the time being, the system is supported and works with only one web browser, the "Firefox". The reason for this is that Firefox is supporting the W3C standards and the system codes have been written with this standard. This disadvantage can be easily overcome by adding the standards that other browsers can interpret on the current codes. This can be another future work.

5.3 RESULTS

The mashup works as planned. In order to use GIS properties, it has a spatial database and Google Maps, which is used for supporting this database, as the map server. Google Maps is a powerful GIS tool that is used in many scientific applications.

Combination of these two important features helped creating a mashup for archaeology. A similar study has been done by TUBA. However it is not for public use and does not contain map information as in this study (TUBA, Ayşe Erguvan). It has been confirmed by the General Directorate of Cultural Heritage and Museums that there is no operational study using maps like they are used in this study (General Directorate of Cultural Heritage and Museums, P.Ç. Ermiş).

Since, it is to be the first version of its kind, it may be insufficient in some ways or may contain errors. Therefore, this system most probably will have to be modified and corrected during its usage.

Opening the service and further work phases of this study will eliminate the lack of information sharing and will provide an important contribution to archaeology.

5.4 FUTURE WORKS

Although the system has been tested and is working functionally, it needs the addition and upgrading of features. Certain user groups and authorities will be given defined permissions.

The authority system is envisaged to have several levels. At the top, an admin type user will be assigned the authority to control all the system, whereas on the bottom, an ordinary user will be given permission to only search or see information. All other levels of permissions can be assigned with respect to these.

Secondly, screens for entering information to the system will be built. One of the aims of the system is to share information. In order to do this, GUI entrance will be opened to people who are assigned authority to enter information or data. The system was designed to provide such data entering. Therefore, the addition of input screens can be easily done in the future.

Furthermore, this system will contain the suggestions of users and can be developed considering these suggestions.

In addition to these, by adding more than one server to the system, it can be converted to a distributed structure and some code corrections need to be done so that it can be used with other type browsers.

Finally, it is desired that this study will serve as an example to other studies of similar kind. From this point of view, developing this study will continue.

REFERENCES

- Paul J. D., Harvey M. D., 2008, "AJAX, Rich Internet Applications, and Web Development for Programmers", Prentice Hall, United States
- Ralf Hartmut Güting, 1994, "An Introduction to Spatial Database Systems", The VLDB Journal - The International Journal on Very Large Data Bases
- David Wheatley & Mark Gillings, 2002, "Spatial Technology and Archaeology, The Archaeological Applications of GIS", Taylor & Francis, Great Britain, p23
- G. Brent Hall, Michael G. Leahy, 2008, "Open Source Approaches in Spatial Data Handling", Springer, Berlin
- M. van Kreveld, I. Reinbacher, A. Arampatzis, and R. van Zwol, 2004, "Distributed ranking methods for geographic information retrieval." In Proceedings of EWCG-04, the 20th European Workshop on Computational Geometry.
- James Conolly and Mark Lake, 2006, "Geographical Information Systems in Archaeology", Cambridge, United Kingdom
- Google Maps, Wikipedia, http://en.wikipedia.org/wiki/Google_Map, last accessed on December 24, 2009
- Spatial Query, Wikipedia, http://en.wikipedia.org/wiki/Spatial_query, last accessed on December 24, 2009
- Web Service, Wikipedia, http://en.wikipedia.org/wiki/Web_service, last accessed December 24, 2009
- Steve W., Online Archaeology, <http://www.online-archaeology.co.uk/GoogleMap/>, last accessed on December 13, 2009
- Dr Kostas Arvanitis, Dr Sian Jones, Mashup Archaeology, Aggregating museum archaeology & archaeological heritage, <http://mashuparchaeology.humanities.manchester.ac.uk/>, last accessed on December 22, 2009

Greta Nicoara, GITA, 2008, www.gita.org/chapters/texas_northcentral/ppts/02_14_08/GoogleMapsAPI.ppt, last accessed December 24, 2009

Simile-Widgets, MIT, <http://code.google.com/p/simile-widgets/wiki/Timeline>, last accessed on December 24, 2009

PostGIS, Using PostGIS, <http://postgis.refractory.net/docs/ch04.html#id2538511>, last accessed January 17, 2010

Ray R. Larso, "Geographic Information Retrieval and Spatial Browsing", https://sherlock.ischool.berkeley.edu/geo_ir/PART1.html, last accessed on December 20, 2009

KRLAB, Computer Science and Information Management Program, Asian Institute of Technology, <http://kr.cs.ait.ac.th/Home>, last accessed December 24, 2009

Semantic Web Company, <http://www.semantic-web.at/1.32.catchword.281.semantic-query.htm>, last accessed December 24, 2009

Developer Network, Yahoo, http://www.theserverside.com/news/thread.tss?thread_id=42722, last accessed December 24, 2009

Pros and cons of client/server computing, Exfoesys Inc., <http://www.exforsys.com/tutorials/programming-concepts/pros-and-cons-of-client-server-computing.html>, last accessed January 03, 2010

Internet FAQ Archives, ...advantages and disadvantages of Client/Server, <http://www.faqs.org/qa/qa-17360.html>, last accessed January 03, 2010

Web Devout, Web browser standards supports, <http://www.webdevout.net/browser-support-summary>, last accessed January 03, 2010

Class and Objects, PHP Manual, <http://www.php.net/manual/en/language.oop5.basic.php>, last accessed January 17, 2010

John Lim, ADOdb, <http://adodb.sourceforge.net/>, last accessed January 05, 2010

AJAX - Asynchronous JavaScript And XML, UVic Dept. of Comp. Sci., <https://secure.cs.uvic.ca/twiki/bin/view/Research/AJAX>, last accessed January 17, 2010

Google Code Help, Google, [http://code.google.com/support/bin/answer.py?hl=en &answer=16532](http://code.google.com/support/bin/answer.py?hl=en&answer=16532), last accessed January 03, 2010

pgAdmin, PostgreSQL Tools, <http://www.pgadmin.org/>, last accessed January 05, 2010.

Firebug, Parakey, Inc., <http://getfirebug.com/>, last accessed January 18, 2010.

Google Code, Google, <http://code.Google.com/apis/maps/>, last accessed Januray 05, 2010

PostGis-1.5 OSVN Manual, Refractions Research - Internal, <http://postgis.refractions.net/documentation/>, last accessed Januray 07, 2010

Quanta Plus, KDE Web Dev Team, <http://quanta.kdewebdev.org/>, last accessed Januray 05, 2010

Web Browser Standards Supports, Web Devout, <http://www.webdevout.net/browser-support-summary>, last accessed Januray 03, 2010

The Historic Buildings and Monuments Commission for England, Heritage Gateway, <http://www.heritagegateway.org.uk/gateway/>, last accessed on December 22, 2009

PHP Classes, Spoono, <http://www.spoon.com/php/tutorials/tutorial.php?id=27>

Apache Web Server, Wikipedia, http://en.wikipedia.org/wiki/Apache_web_server, last accessed Januray 07, 2010

sing GIS in Public Policy Analysis in North Carolina, Watershed Education for Communities and Officials NC State Universi, <http://www.ces.ncsu.edu/depts/design/research/WECO/policyGIS/why.html>, last accessed on April 9, 2010

Ajax-Powered Google Maps Mashup Tutorial, ORACLE Technology Network, <http://www.oracle.com/technology/pub/articles/dev2arch/2007/05/google-mashups.html>, last accessed on April 9, 2010

GIS Mashups for Geospatial Professionals, GEOG 863, Department of Geography, <https://courseware.e-education.psu.edu/courses/geog863/content/syllabus.html>, last accessed on April 9, 2010

Enterprise Mashup, Farallon Geographics, <http://www.fargeo.com/services/enterprise-mashup>, last accessed on April 9, 2010

Google Maps Help Forum, Google, <http://www.google.com/support/forum/p/maps/thread?tid=075eb10962e00cc5&hl=en>, last accessed on April 9, 2010

What is Timeline, Midmarket CIO Definitions, http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci214199,00.html, last accessed on April 12, 2010

The Information, eCheat.com, <http://www.echeat.com/easay.php?t=31901>, last accessed on April 12, 2010

Information Retrieval, The Information Retrieval Group, Computing Science University of Glasgow, <http://www.dcs.gla.ac.uk/Keith/Chapter.1/Ch.1.html>, last accessed on April 12, 2010

JSON, Json.org, <http://www.json.org/>, last accessed on May 9, 2010

Encoded Polyline Algorithm Format, Google, <http://code.google.com/apis/maps/documentation/polylinealgorithm.html>, last accessed on May 11, 2010

APPENDIX A

JSON FORMAT

JSON (JavaScript Object Notation) is a lightweight data-interchange format (JSON, Json.org).

JSON is built on two structures:

- Name value pairs (*object*, record, struct, dictionary, hash table, keyed list, or associative array)
- List of values (an *array*, vector, list, or sequence)

An *Object* (in Figure A.1) is an unordered set of name value pairs.

Begins with { (left brace)

Ends with } (right brace).

Each name is followed by : (colon).

The name value pairs are separated by , (comma).

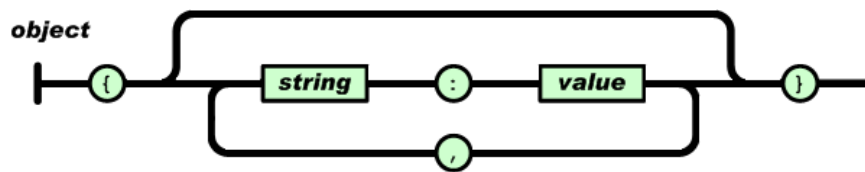


Figure A.1: Object in JSON format

An *Array* (in Figure A.2) is an ordered collection of values.

Begins with [(left bracket).

Ends with] (right bracket).

Values are separated by , (comma).

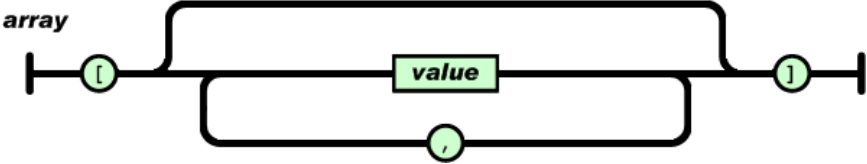


Figure A.2: Array in JSON format

A *Value* (in Figure A.3) can be a string in double quotes, or a number, or true or false or null, or an object or an array.

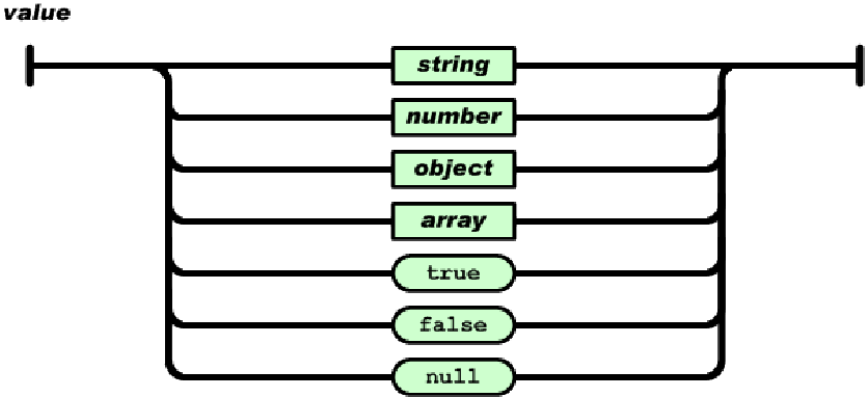


Figure A.3: Value in JSON format

A *String* (in Figure A.4) is a collection of zero or more Unicode characters, wrapped in double quotes, using backslash escapes.

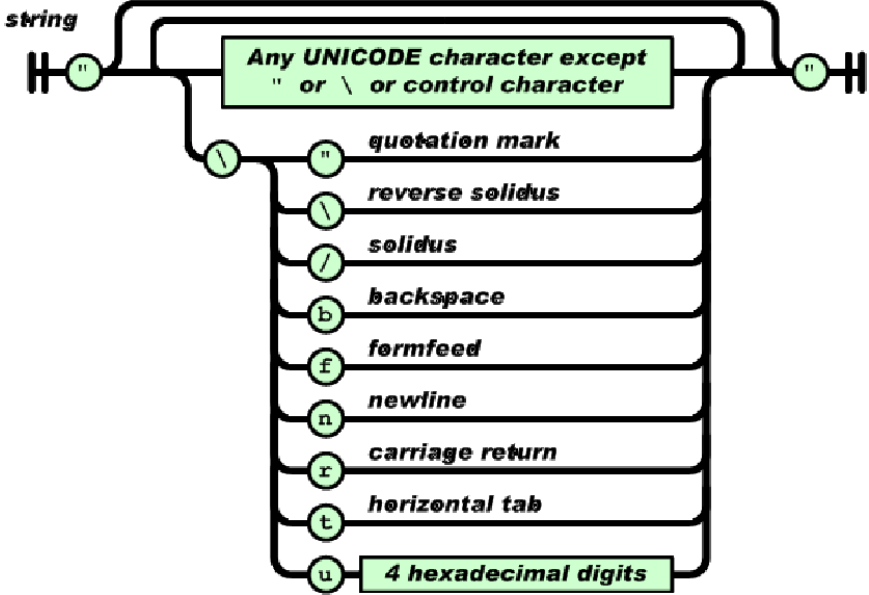


Figure A.4: String in JSON format

A *Number* (in Figure A.5) is very much like a C or Java number, except that the octal and hexadecimal formats are not used.

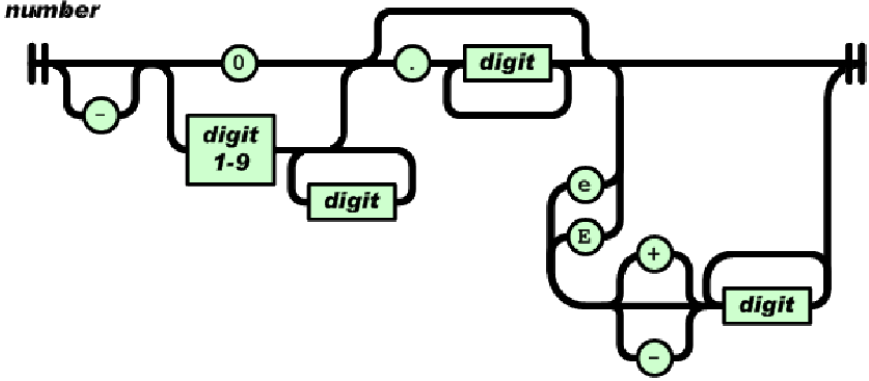


Figure A.5: Number in JSON format

Table A.1: Simple JSON example

Json data (Input)
<pre>var myObject = { 'color' : 'blue', 'animal' : {'dog' : 'friendly' } };</pre>
Accessing data in Json (Output)
<pre>document.writeln(myObject.animal.dog); // outputs friendly</pre>

In table A.1 a simple JSON example is shown.

APPENDIX B

ENCODED POLYLINE ALGORITHM FORMAT

For any set of points, encoded polylines store two types encoded information (Google Maps API, Encoded Polyline Algorithm Format, Google).

- the latitude and longitudes.
- the maximum zoom levels to display.

Levels are encoded using unsigned values, however point coordinates are encoded signed values. Therefore the encoding process is different for levels and point coordinates.

The process converts a binary value into ASCII characters codes. It uses similar to base64 encoding scheme.

The steps for encoding are specified below (Google Maps API, Encoded Polyline Algorithm Format, Google).

1 Take the initial signed value:

-179.9832104

2 Take the decimal value and multiply it by 1e5, rounding the result:

-17998321

3 Convert the decimal value to binary.

00000001 00010010 10100001 11110001

11111110 11101101 01011110 00001110

11111110 11101101 01011110 00001111

4 Left-shift the binary value one bit:

11111101 11011010 10111100 00011110

5 If the original decimal value is negative, invert this encoding:

00000010 00100101 01000011 11100001

6 Break the binary value out into 5-bit chunks (starting from the right hand side):

00001 00010 01010 10000 11111 00001

7 Place the 5-bit chunks into reverse order:

00001 11111 10000 01010 00010 00001

8 OR each value with 0x20 if another bit chunk follows:

100001 111111 110000 101010 100010 000001

9 Convert each value to decimal:

33 63 48 42 34 1

10 Add 63 to each value:

96 126 111 105 97 64

11 Convert each value to its ASCII equivalent:

'õia

Example

Points: (38.5, -120.2), (40.7, -120.95), (43.252, -126.453)

Encoded polyline: _p iF ps—U_ulLnnqC_mqNvxq'

APPENDIX C

DATABASE TABLES

Table C.1: Table: excavation

excavation Structure		
Name	Type	Description
id	serial	PRIMARY KEY
sub_id	integer[]	
name	character varying(255)	
type	character varying(25)	
site_type	character varying(25)	
location	geometry	
start_date	date	
height	double precision	
nation	character(2)	
end_date	date	

Table C.2: Table: eventimeline

eventimeline Structure		
Name	Type	Description
eventID	serial	PRIMARY KEY
eid	integer	FOREIGN KEY
start	date	
end	date	
title	character varying(50)	
description	character varying(50)	
description	character varying(50)	
layerURL	character varying(50)	
pointSWlat	double precision	
pointNElat	double precision	
pointNElng	double precision	
pointSWlng	double precision	

Table C.3: Table: excavation_general_info

excavation_general_info Structure		
Name	Type	Description
id	serial	PRIMARY KEY
eid	integer	FOREIGN KEY
village	character varying(255)	
town	character varying(255)	
region	character varying(25)	
history	text	
property_rights	text	
stituation	character varying(255)	

Table C.4: Table: excavation_periods

excavation_periods Structure		
Name	Type	Description
id	integer	PRIMARY KEY
eid	integer	FOREIGN KEY
pname	character varying(10)	

Table C.5: Table: excavation_references

excavation_references Structure		
Name	Type	Description
id	serial	PRIMARY KEY
eid	integer	FOREIGN KEY
reference_type	character varying(25)	
name	character varying(50)	
info	text	
date	date	
source	text	

Table C.6: Table: excavation_session

excavation_session Structure		
Name	Type	Description
id	serial	PRIMARY KEY
eid	integer	FOREIGN KEY
session_name	character varying(255)	
start_date	date	
end_date	date	
summary	text	

Table C.7: Table: excavation_session_supporters

excavation_session_supporters Structure		
Name	Type	Description
id	serial	PRIMARY KEY
spid	integer	FOREIGN KEY
esid	integer	
start_date	date	
end_date	date	

Table C.8: Table: images

images Structure		
Name	Type	Description
id	integer	PRIMARY KEY
eid	integer	FOREIGN KEY
iname	character varying(50)	

Table C.9: Table: people

people Structure		
Name	Type	Description
id	serial	PRIMARY KEY
name	character varying(50)	
surname	character varying(50)	
prefix	character varying(25)	
nationality	character varying(25)	
organization	character varying(50)	
role	character varying(25)	

Table C.10: Table: permission

permission Structure		
Name	Type	Description
id	serial	PRIMARY KEY
esid	integer	FOREIGN KEY
name	character varying(50)	
date	date	
auth_owner	character varying(50)	

Table C.11: Table: previous_works

previous_works Structure		
Name	Type	Description
id	serial	PRIMARY KEY
eid	integer	FOREIGN KEY
esid	integer	
name	character varying(255)	
refer	character varying(255)	
link	character varying(255)	
info	text	

Table C.12: Table: publications

publications Structure		
Name	Type	Description
id	serial	PRIMARY KEY
eid	integer	FOREIGN KEY
esid	integer	
type	character varying(25)	
source	character varying(255)	
name	character varying(255)	
date	date	

Table C.13: Table: stratigraphic_squence

stratigraphic_squence Structure		
Name	Type	Description
id	serial	PRIMARY KEY
esid	integer	
ss_phase	character varying(5)	

Table C.14: Table: supporters

supporters Structure		
Name	Type	Description
id	serial	PRIMARY KEY
name	character varying(255)	
organization	character varying(25)	
nation	character varying(5)	

Table C.15: Table: team

team Structure		
Name	Type	Description
id	serial	PRIMARY KEY
sub_id	integer	
name	character varying(25)	

Table C.16: Table: team_session

team_session Structure		
Name	Type	Description
id	serial	PRIMARY KEY
esid	integer	FOREIGN KEY
tid	integer	

Table C.17: Table: team_session_people

team_session_people Structure		
Name	Type	Description
id	serial	PRIMARY KEY
pid	integer	
tsid	integer	
start_date	date	
end_date	date	

Table C.18: Table: visual

visual Structure		
Name	Type	Description
id	serial	PRIMARY KEY
eid	integer	
esid	integer	
link	character varying(255)	
type	character varying(255)	
file_type	character varying(25)	
adder	character varying(50)	
date	date	

APPENDIX D

SOME CODE SAMPLES

D.1 GOOGLE MAPS API CODE SAPMLE

Table D.1: Initalizing "Google Maps API"

```
var centerLatitude = 39.00;
var centerLongitude = 35.2;
var description="";
var startZoom = 6;
var point;
map = new GMap2(document.getElementById("_map"));
var location = new GLatLng(centerLatitude, centerLongitude);
map.setCenter(location, startZoom);
map.addMapType(G_PHYSICAL_MAP);
map.addControl(new GMenuMapTypeControl());
map.addControl(new GLargeMapControl3D());
map.setMapType(G_PHYSICAL_MAP);
var overlayControl = new GOverviewMapControl();
map.addControl(overlayControl);
GEvent.addListener(map, 'mousemove', function(latlng) {
    document.getElementById("latbox").value=latlng.lat();
    document.getElementById("lonbox").value=latlng.lng();
});
GEvent.addListener(map, 'singlerightclick', function(latlng) {
    addMarker2(document.getElementById("latbox").
value,document.getElementById("lonbox").value);
});
}
```

D.2 EXTJS (AJAX) CODE SAMPLE

Table D.2: ExtJS (Ajax) grid panel code

```
var proxy2 = new Ext.data.HttpProxy({
url: 'asm_search.php'
});
var reader2 = new Ext.data.JsonReader({
totalRecords: '@total'
},[
{name: 'id', type: 'int'},
{name: 'sub_id', type: 'int'},
{name: 'name'},
{name: 'type'},
{name: 'distance'},
{name: 'river'},
{name: 'site_type'},
{name: 'location'},
{name: 'start_date'},
{name: 'end_date'},
{name: 'height'},
{name: 'nation'},
{name: 'location_text'},
{name: 'location_gml'},
{name: 'location_svg'}
]);
var store2 = new Ext.data.Store({
proxy:proxy2,
reader:reader2
});
store2.load();
grid2 = new Ext.grid.GridPanel({
store: store2,
columns: [
{header: 'id', id:'id', width: 20, hidden: false, dataIndex: 'id'},
{header: 'sub_id', width: 100, hidden: true,dataIndex: 'sub_id'},
{header: 'name', width: 100, sortable:true, dataIndex: 'name'},
{header: 'type', width: 100, hidden: true, dataIndex: 'type'},
{header: 'distance (km)', width: 80, dataIndex: 'distance'},
{header: 'geoObject', width: 140, sortable: true, dataIndex: 'river'},
{header: 'site_type', width: 100, hidden: true, dataIndex: 'site_type'},
{header: 'location', width: 100, hidden: true, dataIndex: 'location'},
{header: 'start_date', width: 100, hidden: true,dataIndex: 'start_date'},
{header: 'end_date', width: 100, hidden: true,dataIndex: 'end_date'},
{header: 'height', width: 100, hidden: true,dataIndex: 'height'},
{header: 'nation', width: 40,hidden: true, dataIndex: 'nation'},
{header: 'location_text', width: 150,hidden: true, dataIndex: 'location_text'},
{header: 'location_gml', width: 140,hidden: true, dataIndex: 'location_gml'},
{header: 'location_svg', width: 310,hidden: true, dataIndex: 'location_svg'}
],
stripeRows: true,
height:150,
width:360,
title:'Search Result'
});
document.getElementById('_resultDiv').innerHTML = "";
grid2.render("_resultDiv");
```

D.3 PHP CODE SAMPLE

Table D.3: PHP sql query code sample

```
$sql = "
SELECT DISTINCT ON (excavation.id) excavation.id, pname, name, astext(location) as
location_text, asgml(location) as location_gml, assvg(location) as location_svg
FROM excavation, excavation_periods
WHERE
location && SetSRID('BOX3D($lat0 $lon0, $lat1 $lon1)::box3d,32767)
AND
excavation_periods.eid = excavation.id
AND
excavation_periods.pname LIKE '$periods'
ORDER BY excavation.id
";

$query = $this->odb->Execute($sql);
$error_msg = $this->odb->errorMsg();
$error_no = $this->odb->errorNo();

if($query->RecordCount()){
    $result = $query->GetRows();
}
else{
    return null;
}
foreach($result as $number => $images){
    foreach($images as $number2 => $images2){
        if(!is_int($number2))
            $result2[$number2]=$images2;
    }
    $result3[]=$result2;
}
unset($result);
unset($result2);
return $result3;
```