MEE08:06

Mobile Service For the Olympic Games 2008

Silva Ruwan Lakmal

Xin Yu

This thesis is presented as part of Degree of

Master of Science in Electrical Engineering

Blekinge Institute of Technology

December 2007

Blekinge Institute of Technology

School of Engineering

Department of Telecommunication Systems
Supervisor: Anders Larsson
Examiner: Markus Fiedler

Abstract

Mobile tourism service applications can be developed and successfully deployed with the rapid developments in mobile phones, communication technologies and tourism.

In many situations, when travelers visit a country for a special event or just as travelers, it is obvious that they want to utilize the time and resources efficiently to enjoy the events and visit many places as possible. To achieve this, a traveler needs variety of information to make decisions and travel plans. Traditionally, this means purchasing tour guide books about a county or a city that refer them.

At the same time, the 29th Olympic game is coming to the oldest civilized country China in 2008. The goal of our mobile service is to provide needed tourist information of the host country as well as event specific information. In the case of traveling for a special event, it is crucial to be informed about the last minute changes in the schedules. Our implementation tries to address these issues as much as possible.

Key words: mobile, tourism, Olympic, Agile programming, Ruby on Rails

Acknowledgment

Firstly, we would like express our sincere gratitude to our research supervisor Markus Fiedler for recommending us to implement the project idea which emerged as a result of the course mobile services, as a thesis topic at WIP, and accepting us for supervision. And also we thank Anders Larsson for providing us this unique opportunity to work on our thesis work at WIP and guiding us throughout this thesis work. We thank all the colleagues at WIP for enlightening us with advises and knowledge, and encouraging us all the time with your innovative work.

Secondly, we would like to thank BTH for giving us the opportunity to study at a well established scientific environment and providing the background knowledge to work on the thesis.

Last, but not least, we thank our partner each other for well collaboration and giving much encouragement during the whole time.

Thank you all!!

Lakmal

Xin

Contents

Chapter 1 Introduction 1

1.1 Introduction 1

1.2 Scope 2

1.3 Organization of the Thesis 2

Chapter 2 Service plan 4

2.1 Short description of the service 4

2.2 Related work and Competitors 5

Chapter 3 Technological Background 6

3.1 Technological Aspects 6

3.2 Mobile applications and services 7

3.3 Mobile networks 8

3.4 Different types of mobile handsets 14

3.5 XHTML vs. WAP 15

3.6 Issues related to Mobile services provisioning 16

Chapter 4 Design of mobile services 20

4.1 Top level design 20

4.2 Interface design 22

Chapter 5 Implementation 33

5.1 Agile Software Development 33

5.2 Xplanner 34

5.3 Model-view-controller (MVC) 35

5.4 Velocity Template Engine 36

5.5 Ruby on Rails (RoR) 39

5.6 Implementation 41

5.7 Image Scaling 42

5.8 Implementation of weather service 46

5.9 Currency converter implementation 54

5.10 Database Design 56

Chapter 6 Conclusion 61

References 62

List of Figures

Figure3.1: User - Service Interaction [1] 7

Figure3.2: Structure of a GSM Network (key elements) [2] 10

Figure 4.1: Top level design and message flows 21

Figure4. 2: Start page of the application 22

Figure4. 3: Main menu 23

Figure 4.4: Main page of Map part 24

Figure 4.5: Venue part 25

Figure 4.6: Hotel part 26

Figure 4.7: Restaurant part 27

Figure 4.8: Main page of Schedule part 28

Figure 4.9: Attraction part 30

Figure4.10: Main page of Tips part 31

Figure5.1: Xplanner 35

Figure 5. 2: Components in a template-based transformation process 37

Figure 5.3: Interface based on existing Velocity system 38

Figure 5.4: Rail's model-view-controller flow [17] 40

Figure 5.5: Top level Architecture 41

Figure 5.6: Mobilis™ Platform [21] 44

Figure5. 7: XML feed for weather information in Beijing city 50

Figure 5.8: Message passing for fetching information from Yahoo weather API 51

Figure 5.9: Weather application internals 52

Figure 5.10: weather page 53

Figure 5.11 Resulting weather information 54

Figure 5.12: Currency form page 55

Figure 5.13: Resulting currency information..... 56

Figure 5.14: Database design..... 57

Figure 5.15: Site-admin login interface 58

Figure 5.16: Designing a page for a mobile device with site-admin 58

Chapter 1 Introduction

1.1 Introduction

In many parts of the world, mobile phones have become such an important part of people's daily lives, one wonders how we ever managed without them. At the same time, according to the improvements of the living standards, more and more people prefer traveling around the world. For these travelers, mobile phones can offer some incredible benefits. In some situations, a mobile phone may simply be the only way to get online and connected to the information super highway. This also offers us the extensive service scope.

In general, all travelers expect the most convenient tour tools to guide their traveling. Therefore, mobile services have a favorable market.

In this thesis we try to implement a service which was initially proposed in the course Mobile Services, targeting the tourists going for the Olympics 2008 in China. The aim of this proposed service is to address the above mentioned issues and to provide travelers with ultimate freedom when traveling around China, providing crucial information such as schedule information of the 29th Olympic Games, and alerts due to last minute changes in the schedules. This service can also be considered as a replacement for tour guide books, providing more updated information than books.

1.2 Scope

We try to implement a guide which is easy to manage and access via a web interface. The total development is performed based on HTTP client server architecture, hence there is no special client developed on the mobile device. Any standard web browser is capable of retrieving the guides' information. This design philosophy is adapted since it supports for

providing up to date information of events for the users as well as simplify the design. Currently most of the mobile devices shipped with an in built web browser. Hence, this enables us to cater to a broader customer base than if it was a special application.

Designing on a web services architecture has its own advantages. Most importantly, it is simple to develop an application which retrieves information from third party web services. To illustrate this concept, this thesis implements a weather service and a currency conversion service based on third party APIs and web services.

Though it is possible to fetch web pages from content providers and then use text processing tools to extract information, the efforts of this thesis are limited to extracting data based on web standards and APIs such as XML, RSS and Web services.

1.3 Organization of the Thesis

The rest of the report is organized as follows.

Chapter 2 gives a brief description of the service proposed and related work.

The design issues and technical selection methodologies are described in Chapter 3, which ranges from networking technologies to mobile web technologies. Further it discusses the technical issues such as service architecture, technologies that would be used and details of the server side implementation.

Chapter 4 gives insight details about the design of the guide. It provides information about the top level design overview of the overall system and gives more information of the page designs.

Chapter 5 is the implementation part, which details the information related to server side development, which is the main development area. It also discusses the tools used in the development, the Velocity-based implementation and the RubyOnRails implementation.

The report concludes with Chapter 6 which is the conclusion.

Chapter 2 Service plan

2.1 Short description of the service

In our design, we try to implement an easy-to-use informative solution which provides the users with more freedom when traveling around Beijing and the other six Olympic cities, by providing services such as maps, attractions, currency conversion, weather, and transportation information like flight, train schedules and subway routes among other possible services. Our presentation of the prototype service is focused on Beijing which is the main host city of the 29th Olympics Games in 2008. However it should be noted that this service can be easily extended to any other event, as well as a general traveler guide. Especially, efforts are made to provide different information related to each Olympic city.

One of the main design concepts is to develop a mobile application which is user friendly and easy to use. This can be considered as a portal where users can select the appropriate service they require, for instance checking the places of interests or checking the Olympic schedule in a particular city or routes, and transportation information.

From this design, we hope people would enjoy the journey around the old civilized country of China to a greater extent while following the Olympic Games at the same time.

This proposed mobile service is well suited for and targeting travelers who will be visiting China especially for the 29th Olympic Games. Additionally it is thought that this will attract techno geeks or even locals in the area. Hence, this number of the customers should be considerable.

2.2 Related work and Competitors

There are several other organizations which tried to implement similar services targeting Olympics with different technologies. Most of them provide information for traveling around a single city and with static content [19]. Comparing them to our service, the latter has its own merits, such as, providing information for the seven Olympic cities although it focuses mainly on the host city Beijing, and introducing dynamic content such as weather forecast and currency conversion applications, which are quite important features for travelers in general.

Along with the development of the mobile services, there are different kinds of the services based on the different operating platform (Windows Mobile Smartphone, Symbian OS and etc.). Some service providers build applications specifically targeting a particular type of mobile platform, and may be some high end models [20]. However our approach is to design a service which is accessible by most of the phones available today, hence the web architecture model.

Chapter 3 Technological Background

Designing services for mobile devices involves the integration of different types of technologies, ranging from the types of networks, types of mobile devices, to types of servers on which the service would operate on. Hence it is vital for the successful operation of the service that appropriate technologies have been chosen. This chapter discusses the available technologies and justifies for selecting a particular technology for this thesis work.

3.1 Technological Aspects

Originally when mobile services were introduced, they were implemented as logic installed in the mobile handset and in the mobile network. Hence, to improve mobile services, technology enablers both in the mobile handset and the mobile network are required. In addition, to enable the introduction of innovative services, there should also be technology enablers that bridge the mobile network to the Internet and to intranets. Wireless Independent Provider (WIP) can be considered as one of those technology enablers. Most of the definitions given below are defined in [1].

3.2 Mobile applications and services

According to Mobile Services Reference Model, a mobile application service can be defined as a computational entity that uses 1...N mobile application services. A mobile application service is a computational entity that uses at least one reference model functional component over specified interfaces. Application services may also use other application services.

The users of a mobile service interact with mobile applications. Mobile applications implemented by service developers could be for a special purpose or can be for generic user agents pre-installed on mobile devices. Figure 3.1 illustrates how users interact with the mobile applications.

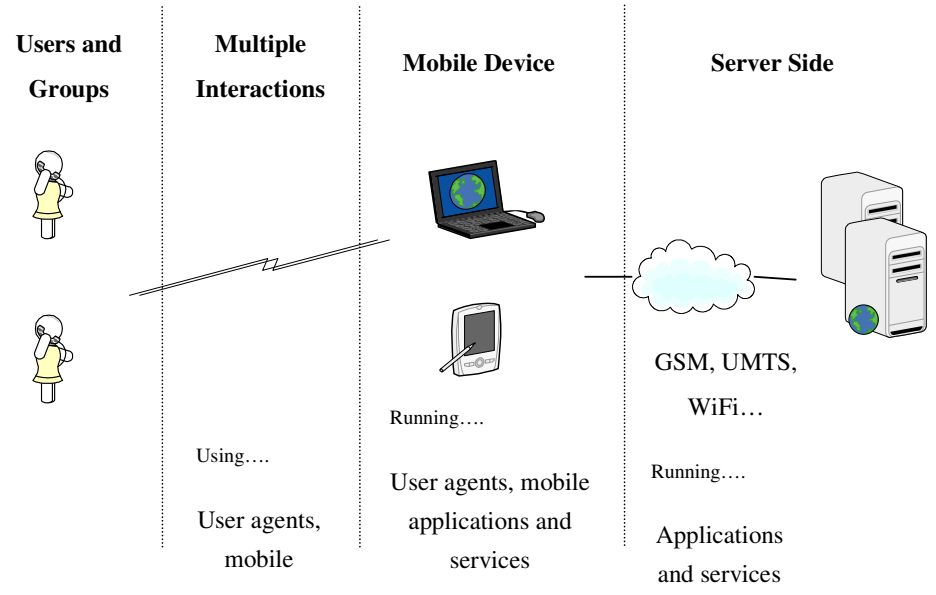


Figure3.1: User - Service Interaction [1]

Users can use multiple ways to interact with mobile applications. This can be done through user agents, which could be either general user agents delivered with the devices such as a web browser or specific applications such as Java based applications [1].

Mobile application services may not directly render a user interface. They could have a specialized per-device application that renders a user interface for them or they use a general browser that invokes a server-side application that accesses the application service and transforms the output to a usable format. Mobile devices on the other hand can execute mobile application services or some functional components on the device itself or can use server-side resources, where the resources are accessed via networks such as GSM, UMTS or WiFi. Next section briefly introduces different types of networks that exist today.

3.3 Mobile networks

In today's context when considering mobile networks, there are many technologies to choose from when designing mobile based service. GPRS, 3G, WLAN, GPS are some of them. It is important to consider over which networking technology the intended mobile service is planning to operate. Factors that need to be considered are mainly the data rates supported by a particular technology. The following sections briefly introduce some of the important mobile networking technologies that exist in today's mobile networks.

3.3.1 Global System for Mobile communications (GSM)

GSM, which stands for Global System for Mobile communications, is the most popular standard for mobile phones in the world. GSM is used by over 2 billion people over more than 212 countries and territories. Its ubiquity makes international roaming very common between mobile phone operators, enabling subscribers to use their phones in many parts of the world. GSM differs from its predecessors in that both signaling and speech channels are digital call quality, and so it is considered a second generation (2G) mobile phone system. This has also meant that data communication was built into the system using the 3rd Generation Partnership Project (3GPP) standards.

One of the key features of GSM is the Subscriber Identity Module (SIM), commonly known as SIM card. The SIM is a detachable smart card containing the user's subscription information and phonebook. This allows the user to retain his or her information after switching handsets. Alternatively, the user can also change operators while retaining the handset simply by changing the SIM. Some operators will block this by allowing the phone to use only a single SIM, or only a SIM issued by them; this practice is known as SIM locking, and is illegal in some countries.

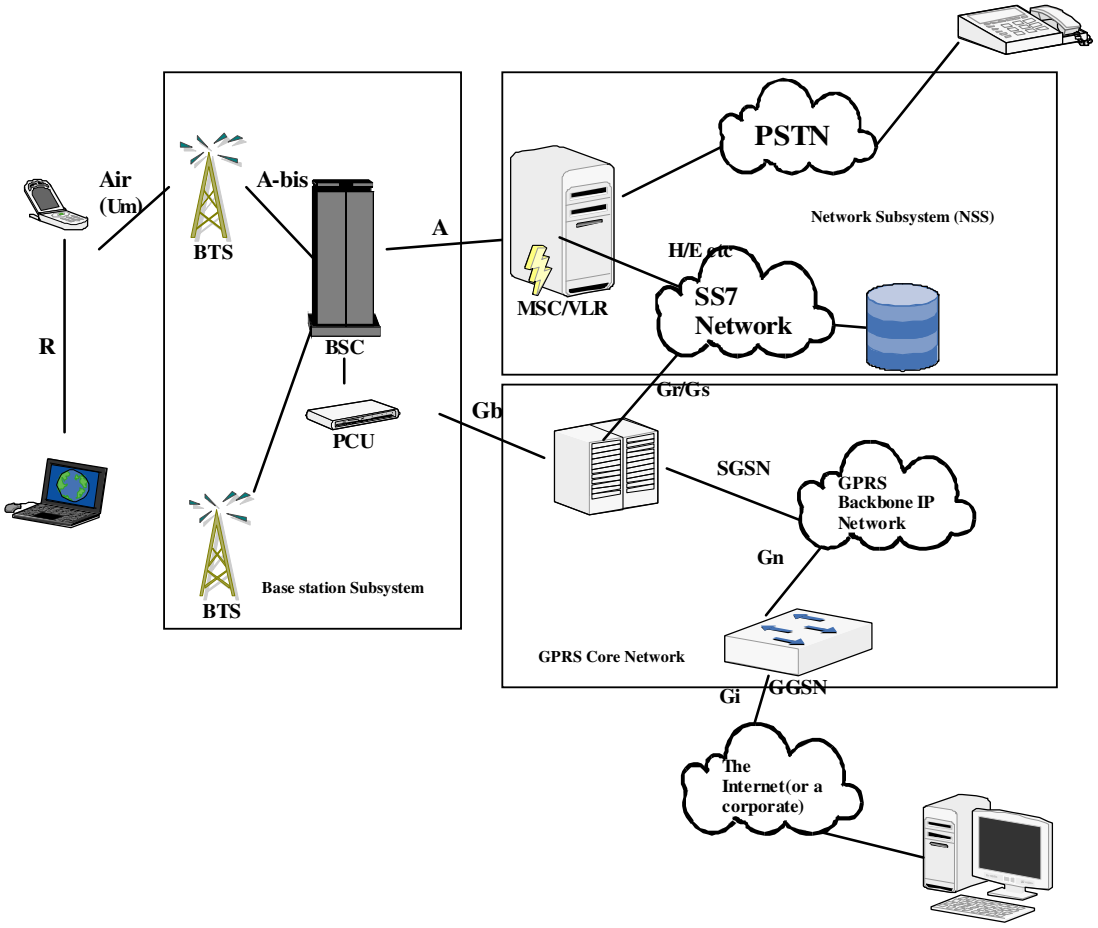


Figure3.2: Structure of a GSM Network (key elements) [2]

The key advantage of GSM systems to consumers has been better voice quality and low-cost alternatives to making calls, such as the Short Message Service (SMS). The advantage for network operators has been the ease of deploying equipment from any vendors that implement the standard. Like other cellular standards, GSM allows network operators to offer roaming services so that subscribers can use their phones on GSM networks all over the world.

GSM was designed with a moderate level of security. The system was designed to authenticate the subscriber using a pre-shared key and challenge-response.

Communications between the subscriber and the base station can be encrypted. GSM only authenticated the user to the network (and not vice versa). The security model therefore

offers confidentiality and authentication, but limited authorization capabilities, and no non-repudiation, where the development of UMTS introduces an optional USIM that uses a longer authentication key to give greater security [2].

3.3.2 General Packet Radio Service (GPRS)

General Packet Radio Service is a Mobile Data Service available to users of Global System for Mobile Communications (GSM) and IS-136 mobile phones. GPRS data transfer is typically charged per kilobyte of transferred data, while data communication via traditional circuit switching is billed per minute of connection time, independent of whether the user has actually transferred data or has been in an idle state. The latter is because even when no data are being transferred, the bandwidth is unavailable to other potential users. GPRS can be used for services such as Wireless Application Protocol (WAP) access, Short Message Service (SMS), Multimedia Messaging Service (MMS), and for Internet communication services such as email and World Wide Web access.

Normally, 2G cellular systems combined with GPRS is often described as "2.5G", that is, a technology between the second (2G) and third (3G) generations of mobile telephony. It provides moderate speed data transfer, by using unused Time Division Multiple Access (TDMA) channels in, for example, the GSM system. Originally there was some thought to extend GPRS to cover other standards, but instead those networks are being converted to use the GSM standard, so that GSM is the only kind of network where GPRS is in use. GPRS is integrated into GSM Release 97 and newer releases. It was originally standardized by European Telecommunications Standards Institute (ETSI), but now by the 3rd Generation Partnership Project (3GPP).

The multiple access methods used in GSM with GPRS are based on frequency division duplex (FDD) and FDMA. During a session, a user is assigned to one pair of up-link and down-link frequency channels. This is combined with time domain statistical multiplexing, i.e. packet mode communication, which makes it possible for several users to share the

same frequency channel. The packets have constant length, corresponding to a GSM time slot. The down-link uses first-come first-served packet scheduling, while the up-link uses a scheme very similar to reservation ALOHA. This means that slotted Aloha (S-ALOHA) is used for reservation inquiries during a contention phase, and then the actual data is transferred using dynamic TDMA with first-come first-served scheduling [4].

GPRS originally supported (in theory) Internet Protocol (IP), Point-to-Point Protocol (PPP) and X.25 connections. The last has been typically used for applications like wireless payment terminals, although it has been removed from the standard. X.25 can still be supported over PPP, or even over IP, but doing this requires either a router to perform encapsulation or intelligence built in to the UE (User Equipment), i.e. the end-device or terminal. In practice, when the mobile built-in browser is used, IPv4 is being utilized. In this mode PPP is often not supported by the mobile phone operator, while IPv6 is not yet popular. But if the mobile is used as a modem to the connected computer, PPP is used to tunnel IP to the phone. This allows DHCP to assign an IP Address and then the use of IPv4 since IP addresses used by mobile equipment tend to be dynamic [4].

3.3.3 Universal Mobile Telecommunications System (UMTS)

UMTS, which stands for Universal Mobile Telecommunications System, is one of the third-generation (3G) cell phone technologies. Currently, the most common form uses W-CDMA as the underlying air interface, is standardized by the 3GPP, and is the European answer to the ITU IMT-2000 requirements for 3G cellular radio systems.

UMTS combines the W-CDMA, TD-CDMA, or TD-SCDMA air interfaces, GSM's Mobile Application Part (MAP) core, and the GSM family of speech codecs. In the most popular cellular mobile telephone variant of UMTS, W-CDMA is currently used. Note that other wireless standards use W-CDMA as their air interface [3].

UMTS over W-CDMA uses a pair of 5 MHz channels. In contrast, the competing CDMA2000 system uses one or more arbitrary 1.25 MHz channels for each direction of

communication. UMTS and other W-CDMA systems are widely criticized for their large spectrum usage, which has delayed deployment in countries that acted relatively slowly in allocating new frequencies specifically for 3G services (such as the United States).

For existing GSM operators, it is a simple but costly migration path to UMTS: much of the infrastructure is shared with GSM, but the cost of obtaining new spectrum licenses and overlaying UMTS at existing towers can be prohibitively expensive.

To differentiate UMTS from competing network technologies, UMTS is sometimes marketed as 3GSM, emphasizing the combination of the 3G nature of the technology and the GSM standard which it was designed to succeed. A major difference of UMTS compared to GSM is the air interface forming Generic Radio Access Network (GeRAN). It can be connected to various backbone networks like the Internet, ISDN, and GSM or to a UMTS network. GeRAN includes the three lowest layers of OSI model. The network layer (OSI 3) protocols form the Radio Resource Management protocol (RRM). They manage the bearer channels between the mobile terminals and the fixed network including the handovers [3].

3.4 Different types of mobile handsets

There are a variety of mobile device manufacturers such as Sony Ericsson, Nokia, Motorola, Samsung, who design different types of mobile devices, which are running on different operating systems and architectures. Hence, when designing mobile applications, compatibility issues need to be considered, so that they would work across many types of devices. Since newer mobile devices generally come with built-in Java platforms, one solution would be to develop Java based applications that would run on top of the Java platform, which makes the applications platform-independent.

However, our approach for the development is based on pure web architecture, where any mobile device which has a embedded web browser would be able to use the service. This has the advantage of the service being available across a wider range of customers.

Even a pure web based mobile service introduces challenges in the development due to the sizes of the screens of the mobile devices, the way data is interpreted on different models. Hence this demands a dynamic architecture, i.e. the structure and behavior of software is changing at run-time as well as the location where the software is executed. A pervasive computing environment also brings new non-functional requirements related to interoperability, heterogeneity, mobility and adaptability [1]. In other words, the content displayed via a web browser need to be optimized depending on the supporting features and characteristics of the mobile device. The Mobilis™ platform which is developed by WIP handles this very well and is described later in the report.

In the so-called mobile world, it is envisioned that the user can access a variety of services at anytime, anywhere, using his/her personal wireless devices as well as other resources and embedded devices in the environment. It is also important to design services which are independent of the network operators.

3.5 XHTML vs. WAP

Mobile devices which are capable of web browsing have been in the market for quite a long time. Due to the nature of the limited available data rates over mobile networks and the size of the screen of mobile devices, it is not possible to browse web pages as we do in a normal PC or a laptop. Hence, there is a need to have a separate markup language to design web interfaces for mobile devices. Two of the commonly used technologies are the Extended Hyper Text Markup Language (XHTML) and the Wireless Access Protocol (WAP). The following section tries to provide a brief introduction to these technologies and discusses pros and crons of each.

3.5.1 Extensible Hyper Text Markup Language (XHTML)

Extensible Hyper Text Markup Language is a markup language that has the same depth of expression as HTML and also conforms to XML syntax.

XHTML is an application of XML and also a more restrictive subset of SGML (Standard Generalized Markup Language) even covering HTML, which is a very flexible markup language. Unlike HTML, which requires a relatively complex, lenient and generally custom parser, XHTML needs to be well-formed and allows for automated processing to be performed using standard XML tools. To some extent, XHTML can be thought as the intersection of HTML and XML in many respects, because it is a reformulation of HTML in XML.

The technology we use in our implementation is XHTML Mobile Profile (XHTML-MP) which is a subset of the XHTML family. XHTML-MP is based on XHTML Basics and targets hand phones specifically by adding mobile phone-specific elements to XHTML Basic.

3.5.2 Wireless Application Protocol (WAP)

WAP, which is the abbreviation of Wireless Application Protocol, is an open international standard for applications that use wireless communication. Its principal application enables access to the Internet from a mobile phone or PDA (Personal Digital Assistant).

The WAP browser provides all of the basic services of a computer-based web browser but simplified to operate within the restrictions of a mobile phone. By the way, there is a Japanese system named i-mode which is another major competing wireless data protocol. WAP sites are websites written in, or dynamically converted to, WML (short for Wireless Markup language) and accessed via the WAP browser.

3.5.3 XHTML over WAP

XHTML Mobile Profile (XTHML MP) is the markup language of WAP 2.0. The XHTML MP markup language along with Cascading Style Sheets (CSS) is most of what makes up WAP 2.0. It offers some advantages over the original WML, particularly in the area of content presentation, but it also borrows a great deal of what works well in WAP. For instance, scripting and push borrow heavily from earlier WAP specifications.

3.6 Issues related to Mobile services provisioning

Major hurdles in mobile service provisioning are briefly discussed in this section [1].

3.6.1 Service roaming

Roaming is a term referred to in the wireless communication field, as the extension of connectivity service in a location that is different from the home location where the service was registered. Due to today's frequent mobility of the users, it is important that users are able to access mobile services irrespective of their geographical location and current location i.e. to provide service roaming in addition to voice roaming. The services should be seamlessly available to the users and the charging and subscription should be simple and cost effective.

3.6.2 Authentication methods

Authentication is a basic need in many of the online / mobile services. This can be simple plain text authentication to more complex authentication methods, depending on the type of mobile service. To achieve this, most of the mobile service providers user authentication methods which vary from one provider to another. These methods should be secure and

reliable, as well as efficient enough to be used in mobile applications. For instance a mobile banking application would require strict authentication methods.

3.6.3 Network limitations

Network limitation is another problem which arises due to the inability of the networks and systems to provide the required service level. There are three aspects which contributed to this, i.e. connectivity limitations, network hardware infrastructure limitations and service infrastructure heterogeneity. Mobile service providers should consider these network limitations when designing mobile applications, such that they operate efficiently in the present mobile networking environments. Generally, the majority the networking improvements can be done by the network operators who own the mobile networks but sometimes, they probably need to check with their suppliers.

3.6.4 Device limitations

The resources used by the devices such as battery and computing power need to be considered. Sometimes a mobile terminal acting as a context provider (CP) must be permanently collecting raw data and sending it through the network connection which affects the battery and computing power of the device.

3.6.5 Interoperability, interworking and portability

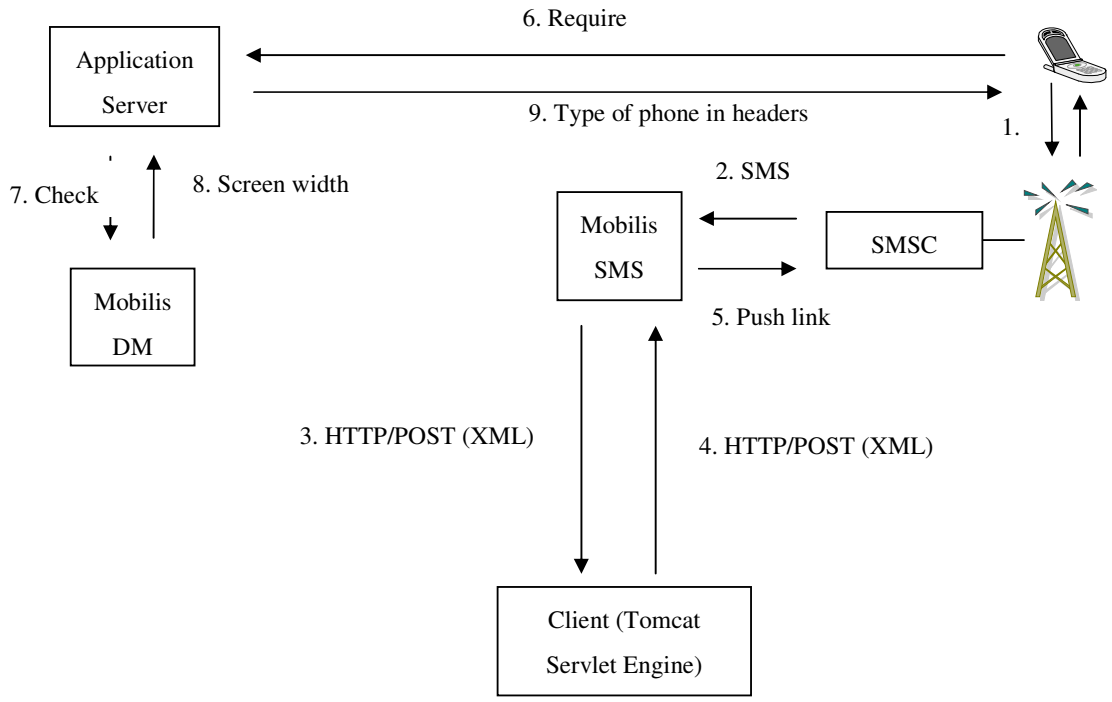
The interoperability, interworking and portability hurdle consists of three interconnected concepts. Interworking can be seen as an enabler for interoperability. Interoperability is the feature that allows the user to seamlessly access and use applications and services across different networks and terminals, independent of the specific implementation and of the specific terminal in use for the application. End-to-end interoperability can thus be seen as a requirement that stems from the user, although it has significant impact on the network,

service and application development. Portability is a feature that allows developers to easily install their software components on different platforms, including different terminals. An application that is portable may not be capable of being integrated or interoperable with other applications. Interoperability between multiple platforms, services and applications is vital for commercial success.

Chapter 4 Design of mobile services

4.1 Top level design

A top level design overview is shown in figure 4.1 below which illustrates the interaction between different units of the total communication platform.



XML file in step 3

<?xml……..>

<sms Deliver batched = “something” company=”Name”

<destaddr>SOMETHING</destaddr>

<destAddrNbr>Number</destAddrNbr>

<origAddr>your phone number</origAddr>

<message>Name</message>

Figure 4.1: Top level design and message flows

The communication process starts when a user sends a SMS with a specific message to a specific number (1). For instance with the agreement with a mobile operator, a shorter number such as 71128 could be assigned for this SMS services when accessing the service locally. When the SMS comes to this number, it is routed by the SMS center to the Mobilis™ platform (Mobilis™ SMS) (2). Then the Mobilis™ SMS server generates an HTTP/POST request with an XML-based message (3) which contains among other things, the originating phone number, and guide name. A sample of such XML based message is shown at the bottom of the figure 4.1. This XML-based message is passed to a Tomcat servlet client, which passes the information in the XML file and then generates another XML message using the HTTP/POST method. This essentially contains the URL for the selected service or the guide. Once this XML file is received by Mobilis™ SMS server, it generates a PUSH SMS and sends back to the users' mobile phone. This is used by the user to go to the actual location of the service.

Upon sending the first request to the application server, the latter then extracts the mobile phone model information which comes along with the request header. Then the application server queries the Mobilis™ Device Manager component to get the screen width information specific to that mobile model. This information is crucial since it determines how the web pages would be displayed in a uniform manner on different mobile devices. The device manager is optimized for this process and can be considered as one of the main components of the system.

When the screen width is received by the application server, it will use that information to perform scaling of the images, which would be optimized for the mobile device in concern.

4.2 Interface design

In the Olympic guide, there are six parts which are illustrated below. The screen shots of the start page and main menu of our application are displayed as Figure 4.2 and Figure 4.3, respectively.

Figure4. 2: Start page of the application

Figure4. 3: Main menu

4.2.1 Map

For the Map part, there are two modes, one which provides static information and the other which provides dynamic information where a customer can orientate its current position as soon as arriving to a city. In the event of locating places, the system will try to automatically pin-point the users' current location on the map and mark some main surroundings, making it more convenient for the user to start its journey.

In the following, we just focus on the first mode, since it is easier to implement and also much more common for existing mobile phones as compared to the second mode.

In the Map part, it provides three different items, such as the venues, the hotels and the restaurants. Each of them is displayed for the seven Olympic cities. These are shown in figures 4.4 and 4.5.

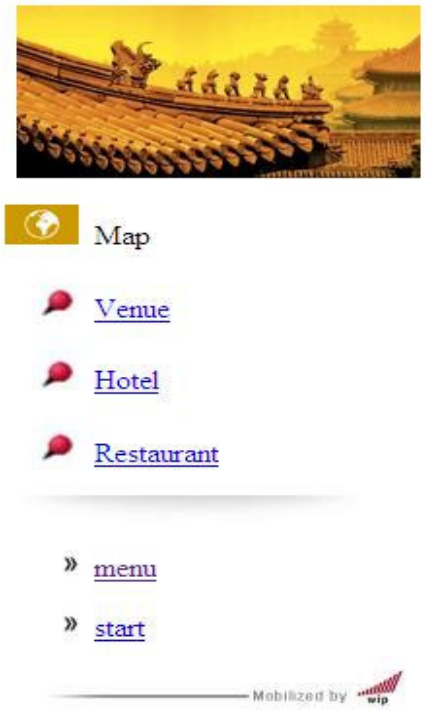


Figure 4.4: Main page of Map part

In detail, for each specific venue, the user is offered with an image of the venue which make sure the users can find the right place they want, the rough map of the venue and some sorts of the most important information of it, such as the particular address, the competitions will be hold there, and how to get there.

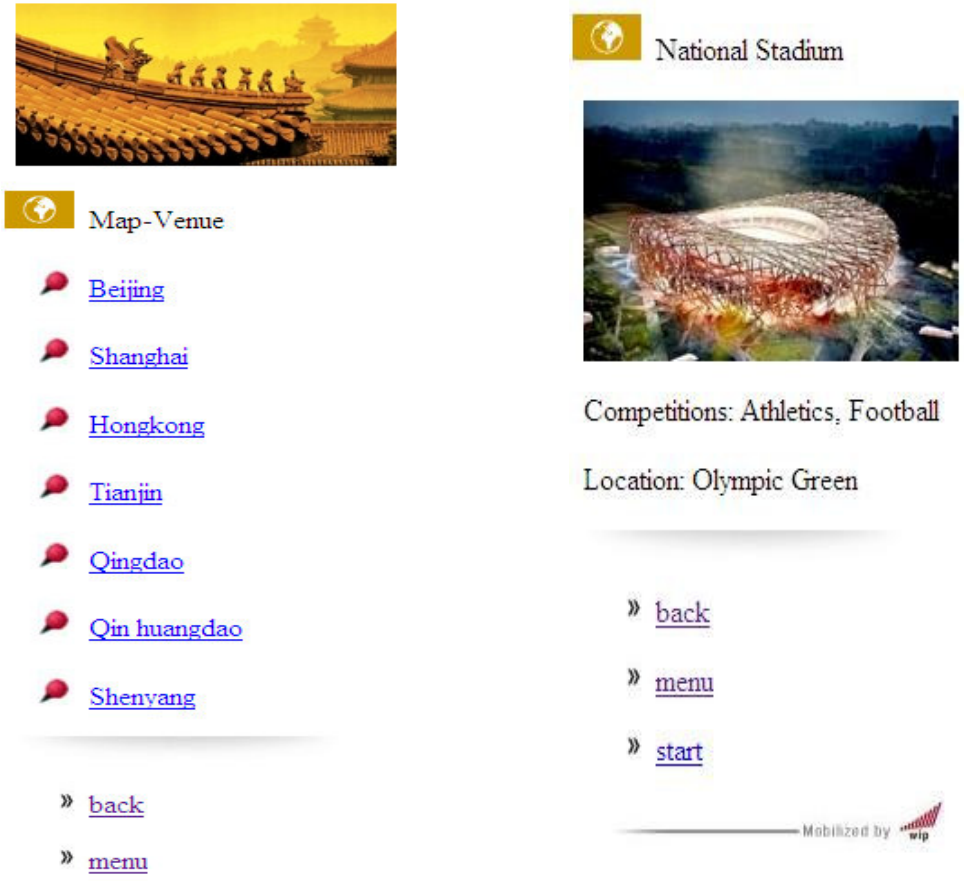


Figure 4.5: Venue part

In the hotel item, there is a small difference between Beijing and the other six cities. This is because most of the competitions are hold in Beijing, which is the famous traveling city among the seven Olympic cities. In the hotel of Beijing, there are more options in to the 5 star, the 4 star and the 3 star categories. However, for each hotel, the clients can make phone calls directly by the telephone number we provided. Following figure 4.6 shows this interface.

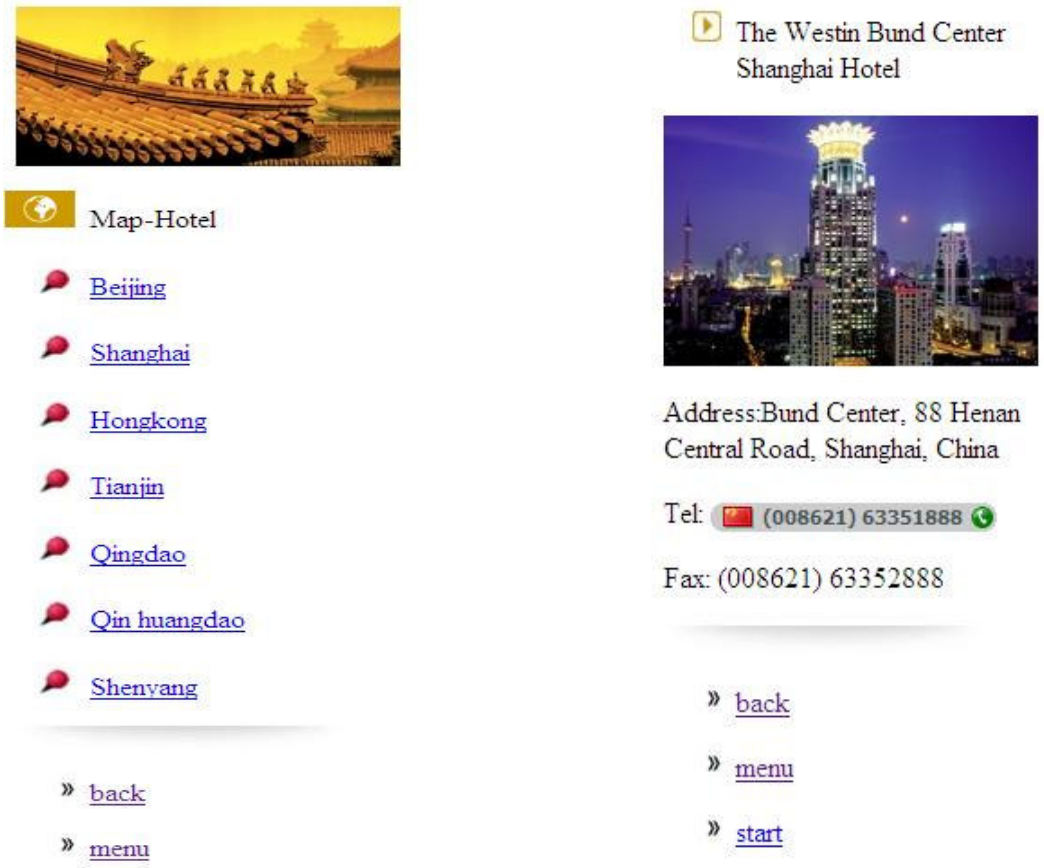


Figure 4.6: Hotel part

Similar to the hotel item, in the restaurant item, we will also provide more choices for Beijing city according to gourmet streets, long-standing restaurants and the western food. Why we do like this? It is because we hope our customers can taste as many traditional Chinese foods as possible and also can taste the type of their own food that they are used to. This interface is shown in figure 4.7.

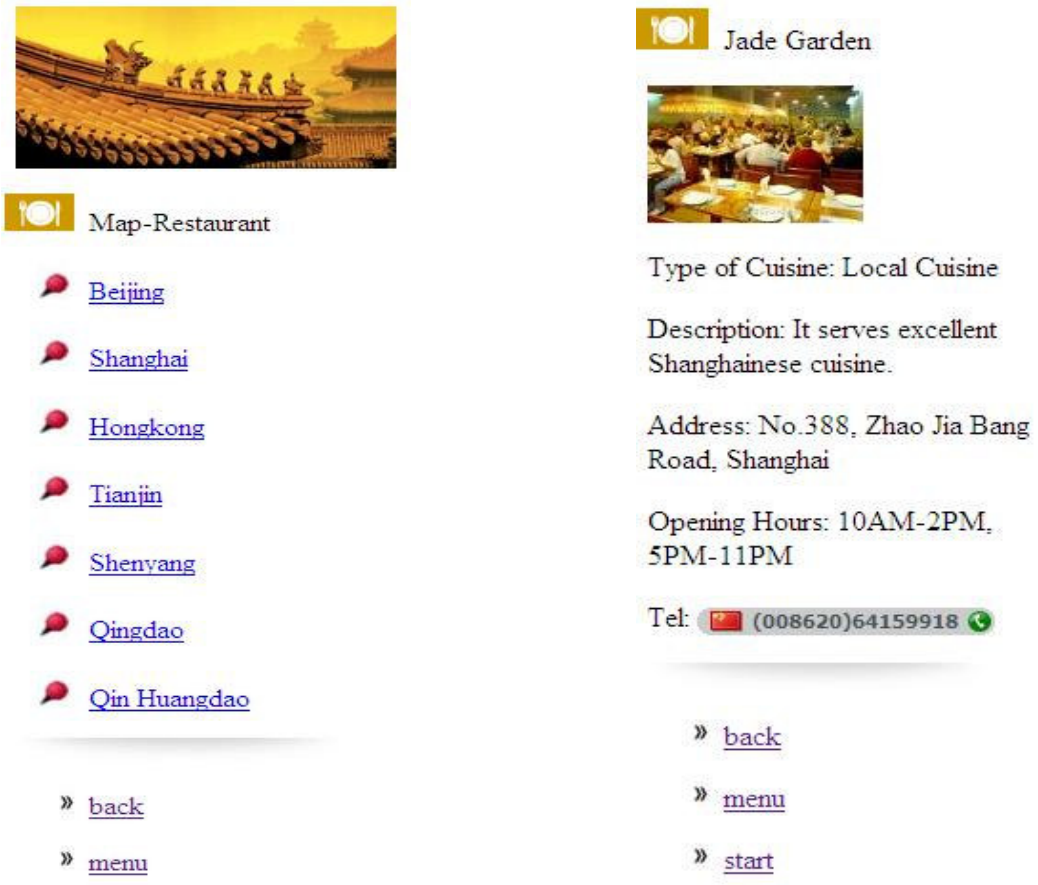


Figure 4.7: Restaurant part

4.2.2. Schedule

As shown in figure 4.8, in the Schedule part, the users can check their schedule according to the different venues and sports they are interested in. Since there are 28 different sports in the 29th Olympics Games, then in our case it is easier for the user to just get the information what they are interested in.

Of course, we will update the latest changes timely, which is crucial for watching such an important event.

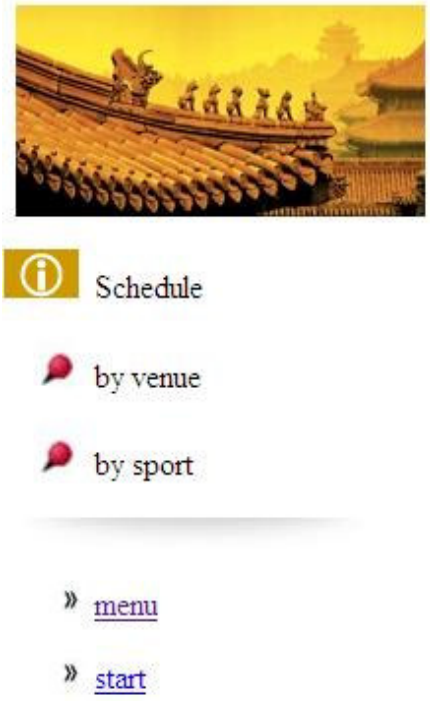


Figure 4.8: Main page of Schedule part

4.2.3. Attraction

In this section, necessary tour information is provided, followed by the different Olympics cities in order to keep the coordination of the structure. And as we mentioned in the hotel item (Map part), Beijing is the main city to held the 29th Olympic Games in, and thus, we accentuate on the Beijing city in this part also.

In the Beijing item, there are 4 options, for instance, historical interests, natural interests, shopping places, and entertainment places. For the historical interests and natural one, we just collect the top and the most famous places for the users to help them make decision easily. From the shopping places, the clients can search some traditional Chinese goods as china, silks and so on. We also give some typical suggestion about the entertainment places where the clients can taste quite different culture and have more fun during the trip time.

For the other six Olympic cities, the classification is almost the same as Beijing city, but a little briefer than regarding the Beijing attractions. But for each interesting place, it includes

the brief introduction of it and its address, permission fee, opening hours and a route how to get there. Figure 4.9 is a screenshot of the attractions that is being implemented in our service.

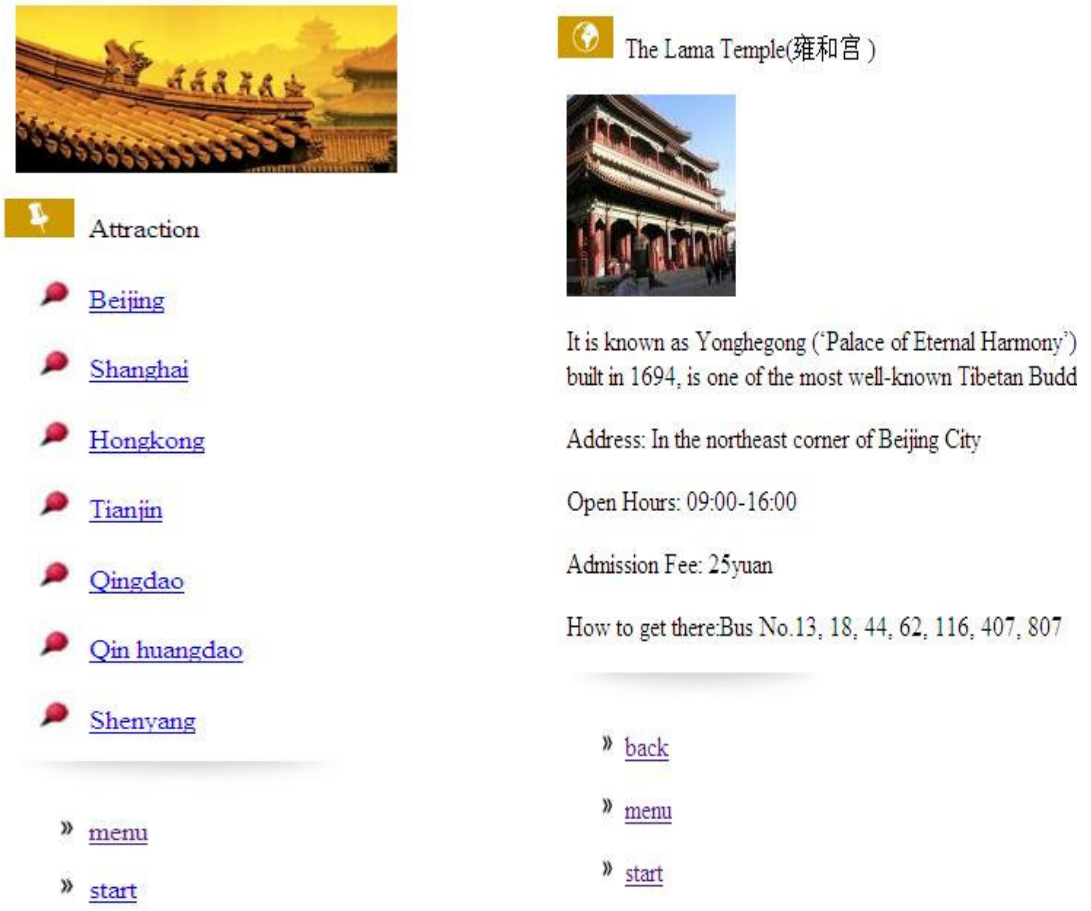


Figure 4.9: Attraction part

4.2.4. Currency

In this Currency part, initially we planned to make all potential users to convert their home currency to Chinese currency, which is called RenMinBi and abbreviated as RMB or CNY, based on the latest available converting rate. But considering the limitation of the screen of mobile phones and the vast amount of types of different currencies all over the world, finally we just picked up some (most popular) of them, such as EUR, USD and SEK.

4.2.5. Weather

In the Weather part, we always show the current weather of Olympic cities. And the user can also check the 24 hours forecast for helping them to plan their trip.

The currency and weather part, are described in more details in section 5.8 and 5.9.

4.2.6. Tips

In this small part, we just want to give some more tips for our clients during their trip such as how to convert their own currency to Chinese currency safely and easily, as well as some basic Chinese words. And we also give some information in case of the emergency cases such as the numbers of some hospitals and the embassies. This is illustrated in figure 4.10 below.

Figure4.10: Main page of Tips part

In a word, it is a thoughtful part of our application which makes our customers feeling more convenient.

Chapter 5 Implementation

The implementation of this mobile service essentially requires software development methods. This chapter briefly introduces the software development methods which have been used, and then moves into more details of the actual design of the service both on the Apache Velocity [9] platform as well as on RubyOnRails [17].

5.1 Agile Software Development

There are several design paradigms which could be used in the software development such as waterfall model, v-model, spiral model and agile development [18]. For the development of this mobile application, a concept known as agile software development, gaining popular these days, is adapted, which happens to be the style followed at WIP.

Agile software development is a conceptual framework for software engineering projects that promotes development iterations throughout the life-cycle of the project [12]. This in turn promotes extreme programming and supports rapid application development. The general idea is that iterations run through the whole process of the system/product, from the starting point to the end point, with iteration durations of two to three weeks. At the end of the each iteration some deliverables are produced, which can be tested. Due to the nature of the mobile application and the time constraints of the project, we adapted this technique.

5.2 Xplanner

XPlanner is a project planning and tracking tool for eXtreme Programming (XP) teams [6] and is used to manage the project. As an initial step, user stories are added to a backlog,

which is a place for containing all the user stories. This ensures that new ideas would not slip away.

A user story is a very high-level definition of a requirement, containing just enough information for the developers to produce a reasonable estimate of the product. This is mainly written from a customer's perspective and simple enough so that a non-technical person can understand it.

Several user stories are selected from the backlog for the current iterations, hence, the user stories in a iteration are a subset of the backlog user stories. For each iteration, developers add tasks and estimated times for each task. Once the iteration starts, it is convenient to monitor each developer's progress and the progress of the project itself. These data is used by Xplanner to provide management reports at the end of the project. Figure 5.1 shows a screen shot of the Xplanner.

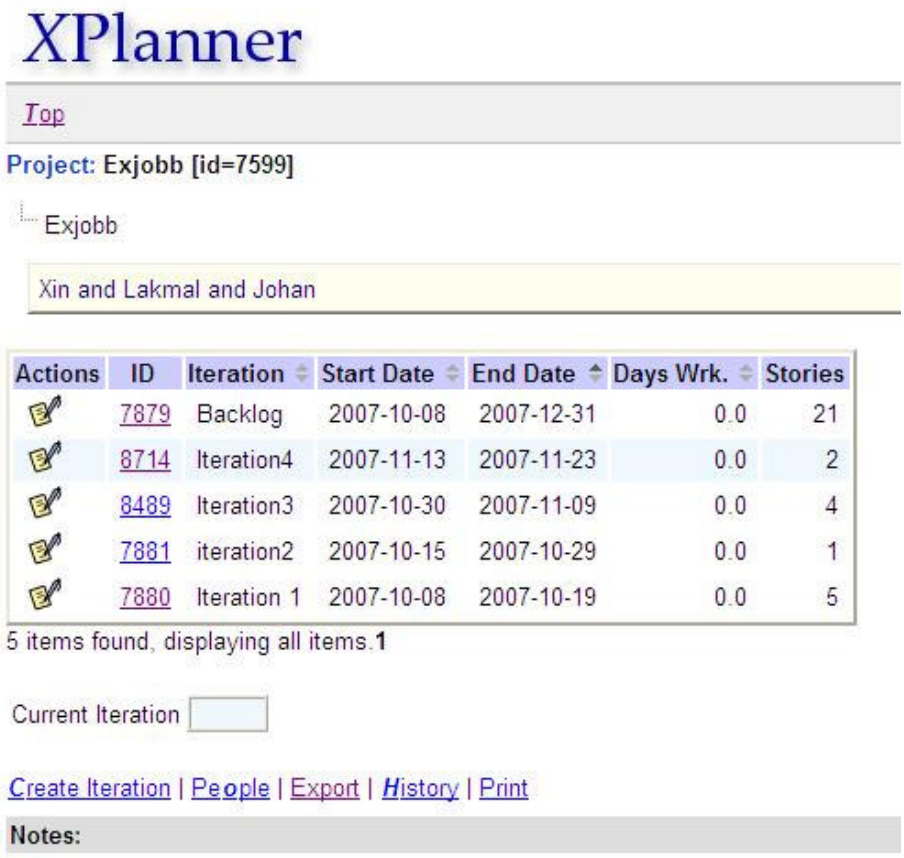


Figure5.1: Xplanner

5.3 Model-view-controller (MVC)

In the 70's, a programming language called Smalltalk defined an architecture to keep the essence of an application separated from of its interfaces, called the Model-View-Controller architecture. Since then it has become a common-place idiom within the object-oriented systems [8]. In complex computer applications that present a large amount of data to the user, a developer often wishes to separate data (model) and user interface (view) concerns, so that changes to the user interface will not affect data handling, and that the data can be reorganized without changing the user interface. "The model-view-controller solves this problem by decoupling data access and business logic from data presentation and user interaction, by introducing an intermediate component: the controller" [8].

5.4 Velocity Template Engine

There is an existing guide application at WIP, which is running on Apache Velocity. Velocity is a Java-based template engine which can be used as a simple yet powerful template language to reference objects defined in Java code [9].

When Velocity is used for web development, Web designers can work in parallel with Java programmers to develop web sites according to the Model-View-Controller (MVC) model, meaning that web page designers can focus solely on creating a site that looks good, and programmers can focus solely on writing top-notch code. Velocity separates Java code from the web pages, making the web site more maintainable over its lifespan and providing a viable alternative to Java Server Pages (JSPs) or PHP.

Velocity's capabilities reach well beyond the realm of the web; for example, it can be used to generate SQL, PostScript and XML from templates. It can be used either as a standalone utility for generating source code and reports, or as an integrated component of other systems. For instance, Velocity provides template services for the Turbine web application

framework, together resulting in a view engine facilitating development of web applications according to a true MVC model.

5.4.1 Components in a template-based transformation process

Figure 5.2 shows the four components involved in a template-based transformation process. They are:

Data Model: Contains data, organized in a specific structure, that have to be transformed.

Template: Formats the data model into the output code. It contains references to entities belonging to the data model.

Template Engine: The application that performs the transformation. It has the input data model and the template, and produces output by replacing the template internal references with real data coming from the model.

Target: The result of the transformation process. [10]

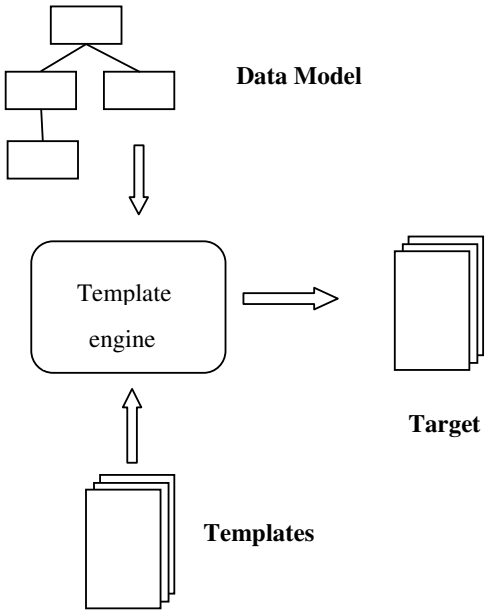


Figure 5. 2: Components in a template-based transformation process

5.4.2 The interface based on the existing Velocity system

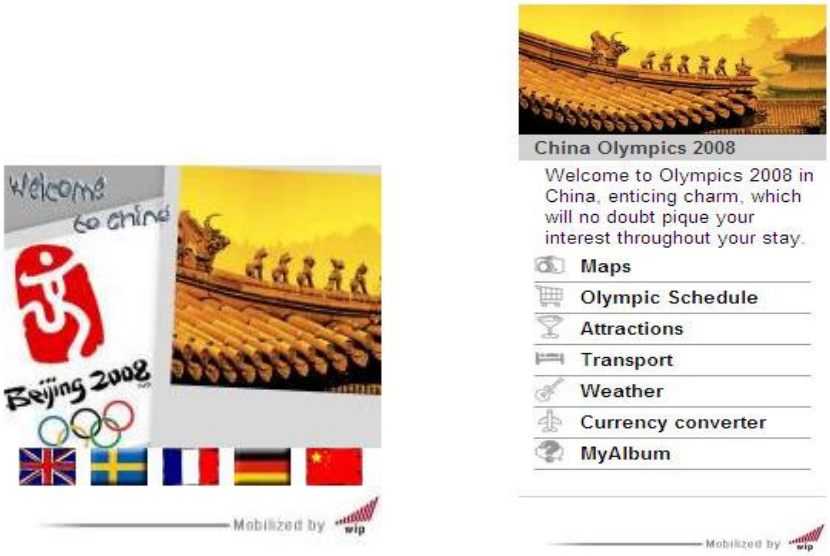


Figure 5.3: Interface based on existing Velocity system

5.4.3 Shortcomings of the Velocity-based system

There are several shortcoming of the exiting Velocity-based system, in terms of scalability and improvements. When the number of guides increases, it is hard to manage all of them in a cost effective manner. Due to the nature of the Velocity template files, the person who is responsible for adding and changing need to be technically competent enough to make modifications updates to guides. This is not very feasible and a practical method, due to the cost involved in the hiring such technically competent persons for a small job. Further, it's not practical to monitor and update the mobile guides for each and every dynamic change such as opening and closing hours of shops during winter and summer time, special deals they provide time to time. A better way of handling this would be to have distributed levels of administrations for guides. In the existing system, it is not easy to implement such levels of administration. The main reason for this problem is that the system is not a database-driven system. To overcome these shortcomings, the existing platform needs to be transformed to a database-driven architecture. Hence the new system is implemented on RubyOnRails, which is a rapid application development framework and naturally supports

database driven applications. The proceeding sections introduce the components within the RubyOnRails.

5.5 Ruby on Rails (RoR)

5.5.1 Ruby

"Ruby is an interpreted scripting language for object-oriented programming. Interpreted scripting implies that a Ruby application is run without first compiling the application". [16]

5.5.2 Rails

"Rails is a full-stack framework for developing database-backed web applications according to the Model-View-Control pattern". [17]

5.5.3 RoR basic introduction

RubyonRails is a framework which is based on the Ruby programming language and the Rails framework.

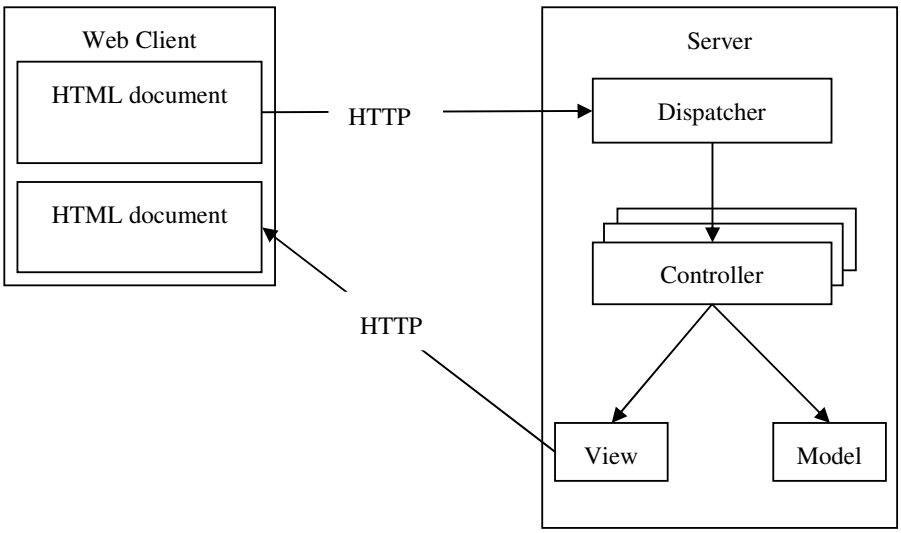


Figure 5.4: Rail's model-view-controller flow [17]

5.5.4 New architecture based on RoR

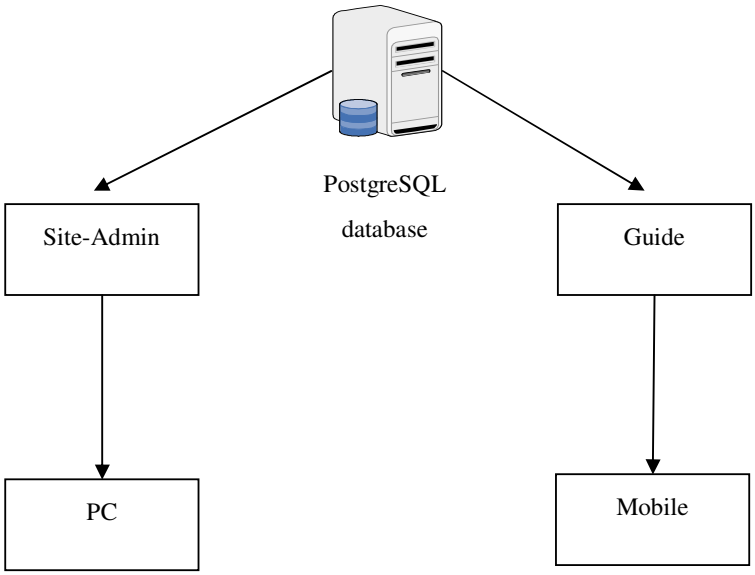


Figure 5.5: Top level Architecture

The new architecture depicted in Figure 5.5, consists of two applications running on RubyOnRails, which share a single database. The “Site-Admin” application is used to input data into the database via a GUI interface whereas the “Guide” application is responsible for providing content to the mobile device. The “site-admin” application consists of several layers of administration which is achieved with different roles in the database.

5.6 Implementation

5.6.1 Client-side implementation

There are several ways to implement the client-side. Since there are lot of dynamic content involved the Olympic guide, the client side implementation was purely implemented as a XHTML-based web application. Hence, from a client’s perspective, it's simply navigating web pages and there is no need to implement a client side application. All that is needed on the client device is the existence of a web browser.

5.6.2 Server-side implementation

The server side is implemented in Linux platform. It's an operating system which is popular today since it is open source, free and stable.

For the mobile guides to work, an http server needs to be installed. In our implementation we chose Apache http server which is one of the most commonly used web servers that exist today. For the Velocity application to work, on top of apache, tomcat servlet engine needs to be installed. Then, finally, the Velocity template engine is installed on top of these.

RubyOnRails is installed and apache is configured to use fast-cgi for processing RubyOnRalis applications.

5.7 Image Scaling

Today there exist so many varieties of phone models which vary depending on the manufacturer. They are varying regarding the screen size, the way some types content are interpreted, for instance tables. It's not a practical to design many types of web applications which would render content in same way. WIP has already built a solution for this, which is known as the Mobilis™ Platform. The following section gives a brief introduction to Mobilis™.

Mobilis™, which is developed by WIP, is a component-based product with a high level of flexibility, scalability and reliability. The combination of several mobile communication technologies gathered in one platform enables maximum usage of the multiple communication protocols within one mobile solution. The encapsulation of the complexity and variety of the mobile and wireless technologies for the developers of mobile services provides clients with faster, easier and more cost-effective access to mobility [11].

For most of the mobile web applications, the size of the images which are displayed on the screen needs to vary depending on the screen width. Hence it is important that the images in the original web application need to be scaled to an extent suitable for the mobile phones. This can be achieved by using the mobile device model information contained in the user agent header. Using these basic information of the mobile device's information, a query is made to a component in the Mobilis™ platform known as the Device manager, which contains the width of the mobile device. Based on this information it is possible to scale the images which are then optimized for a given mobile device.

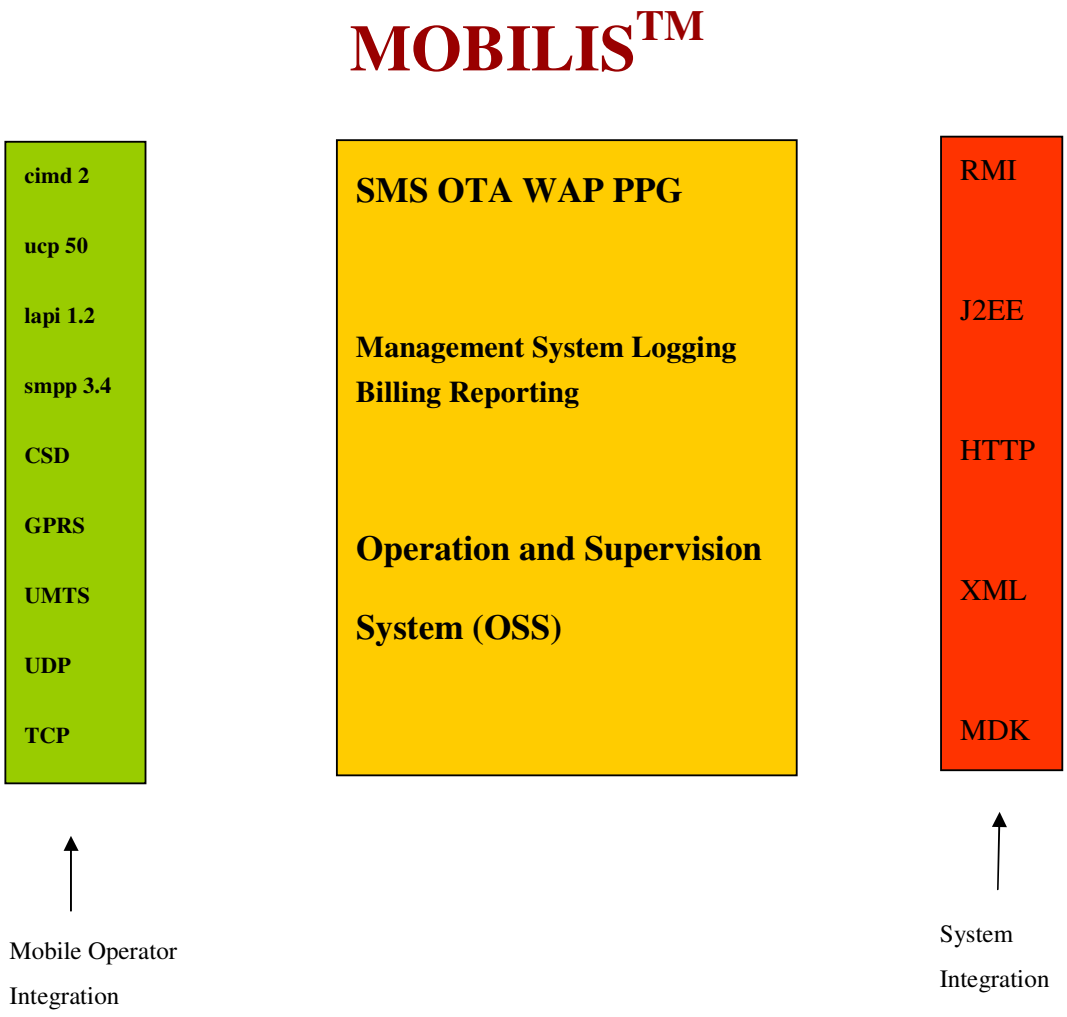


Figure 5.6: Mobilis™ Platform [21]

For testing the mobile devices effect during the development stage, the Firefox browser was used along with the 'Modify Header' and 'User Agent switcher' add-ons on the browser, which can be treated as mobile simulators, and real time testing were carried out on actual phone models Sony Ericsson K800i, Nokia 6220. In to achieve a good testing effect, we modify the header and user agent switcher in Firefox.

5.7.1. The User-Agent Header

The User-Agent Header contains a line of text that can used to identify a user agent and client device [7]. Most of the time, we can find the device model and manufacturer from the User-Agent header. It may also include information such as the client device's OS version, browser version, Java capabilities and so on.

In general, the User-Agent header is useful in the following situations:

1) When we need to identify a specific mobile device model. For example, the User-Agent header can help us determine whether a mobile device is a Sony Ericsson K800i cell phone.
2) When we need to differentiate mobile devices or user agents made by different companies. For example, the User-Agent header can help us determine whether a cell phone is made by Nokia or Sony Ericsson.
3) When we need to determine whether a user agent is a web browser on a personal computer or a micro browser on a mobile device.

Our case belongs to the first situation. An example of a user agent header is given below for Sony Ericsson K800i phone:

user_agent = 'SonyEricssonK800i/R1CB Browser/NetFront/3.3 Profile/MIDP-2.0 Configuration/CLDC-1.1'

Each mobile device manufacturer has its own User-Agent header format. Thus, the keyword to use to identify a specific mobile device model is different for different manufacturers. You may need to check the website of the mobile device manufacturer to find its User-Agent header format. In general, the User-Agent header contains information that can help to differentiate mobile devices or user agents made by different companies. Like in our case, the word "SonyEricsson" can be found in the User-Agent header of Sony Ericsson's cell phones.

5.7.2. The x-wap-profile Header

The x-wap-profile header and the profile header are used to find the UAProf (User agent profile) document of a mobile device, which is an XML document that contains information about the features and capabilities of a mobile device [15]. Very often the URL that points to the UAProf document of a mobile device can either be found in the x-wap-profile header or the Profile header, but in some cases it is located in other HTTP headers. For above Sony Ericsson phone, the x-wap-profile is given below:

x_wap = 'http://wap.sonyericsson.com/UAprof/K800iR101.xml'

To extend the checking device capability, we define an initial filter function as check_device in the MVC model.

5.8 Implementation of weather service

A weather report consists of dynamic content, which needs to be up-to-date when a user requests for weather information. Due to this dynamic nature of data and the accuracy of data demanded for such a services, it is not an economical and a practical solution to have a database or a system which is managed by the mobile service provider itself. Though there exist a few XHTML-based weather services available today especially for mobile applications, from a mobile service providers' point of view, it is not a viable solution to place such links into the mobile guides, since users are then it is moved out of the guide in concern. This may introduce confusion to the user when redirected to many other sites which may not be of interest. Hence it is important to have a service still belonging to the mobile service provider, which at the same time obtains content from a reliable external source and provides it to the user.

The weather service in this thesis work is implemented using the Yahoo weather API, which is a free service and provides sufficient weather information. The Yahoo Weather RSS feed enables to get up-to-date weather information for a selected location. This RSS feed is a dynamically-generated feed, based on the zip code or Location ID. Most of the

information contained in this section is extracted from the official Yahoo weather API [12].

The information fetching process from Yahoo weather server is a simple process. The RSS feed request follows a simple HTTP GET syntax starting with a base URL and then adding parameters and values after a question mark (?). Multiple parameters are separated by an ampersand (&). For weather RSS, there are two parameters as following:

1) p – location parameter

2) u – unit for temperature (Fahrenheit or Celsius)

An example is given below, which makes a request for weather information of Beijing city in Celsius.

http://weather.yahooapis.com/forecastrss?p=CHXX0008 "&u=c

In this example, CHXX0008 is the location parameter value for the Beijing city. A full list of codes for cities around the world can be found in [13]. Since this information transfer happens via XML files from yahoo server to WIP, we need an XML parser on the local server to extract the information from the XML file. XML has become one of the de-facto standards for inter-application communication.

The response from the Yahoo server to the HTTP GET request conforms to RSS 2.0 and XML 1.0. A sample response is given in Figure 5.7 and Figure 5.8 depicts the message transfer between application server and the Yahoo weather application server. Some of the important elements that our XML parser extracts from the XML feed are the yweather:location, yweather:unit and yweather:astronom. Other elements can be extracted in a similar way if there is a demand exists.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<rss version="2.0" xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0"
xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">

 <channel>

 <title>Yahoo! Weather - Beijing, CH</title>

<link>http://us.rd.yahoo.com/dailynews/rss/weather/Beijing__CH/*http://weather.yahoo.com/fo
recast/CHXX0008_f.html</link>

 <description>Yahoo! Weather for Beijing, CH</description>

 <language>en-us</language>

 <lastBuildDate>Mon, 03 Dec 2007 6:00 pm CST</lastBuildDate>

 <ttl>60</ttl>

 <yweather:location city="Beijing" region="" country="CH" />

 <yweather:units temperature="F" distance="mi" pressure="in" speed="mph" />

 <yweather:wind chill="30" direction="40" speed="4" />

 <yweather:atmosphere humidity="23" visibility="999" pressure="30.53" rising="1" />

  <yweather:astronomy sunrise="7:18 am" sunset="4:50 pm" />


<link>http://us.rd.yahoo.com/dailynews/rss/weather/Beijing__CH/*http://weather.yahoo.com/fo
recast/CHXX0008_f.html</link>

 <pubDate>Mon, 03 Dec 2007 6:00 pm CST</pubDate>

 <yweather:condition text="Fair" code="33" temp="34" date="Mon, 03 Dec 2007 6:00 pm CST" />

 <description><![CDATA[

<img src="http://l.yimg.com/us.yimg.com/i/us/we/52/33.gif" /><br />

 <b>Current Conditions:</b><br />
```

Figure5. 7: XML feed for weather information in Beijing city

Since our main platform used for development is Ruby on Rails, it would be ideal to implement the XML parser application and the XHTML generation in Ruby.

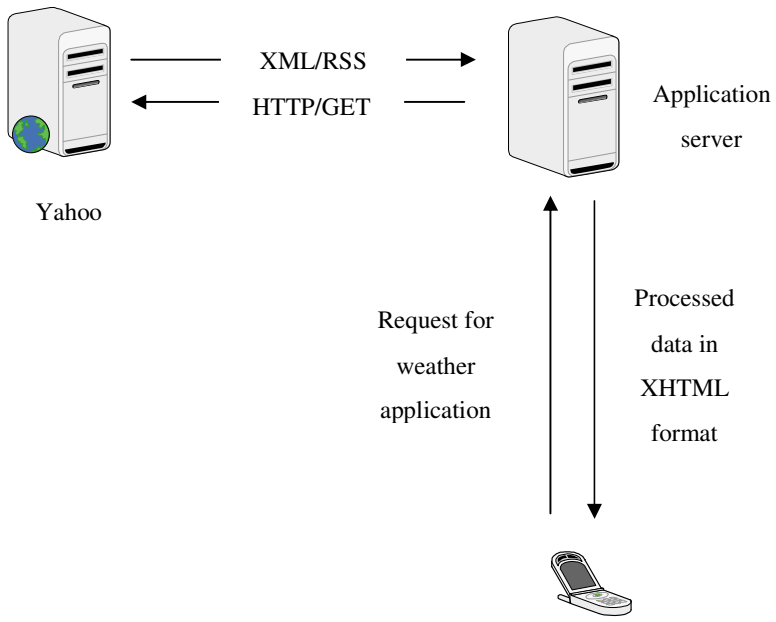


Figure 5.8: Message passing for fetching information from Yahoo weather API

Many APIs exist in Ruby for parsing and generating XML. REXML is the standard API that ships as part of the Ruby core. Other frameworks such as 'Builder' and 'Hpricot' are available as gems. Gems referred to third-party APIs or libraries for Ruby that conform to a specific format. They are easily downloadable and installable.

For this design, REXML was chosen since it comes along with Ruby by default in all the Ruby installations so that it would be compatible across different platforms and installations. Due to the dynamic nature of the weather data, the content that is fetched from the yahoo server will not be stored in the local database, but directly extract the information and generate an XHTML file to be sent to the client. The following Figure 5.9 depicts the message handling and control within RubyOnRails.

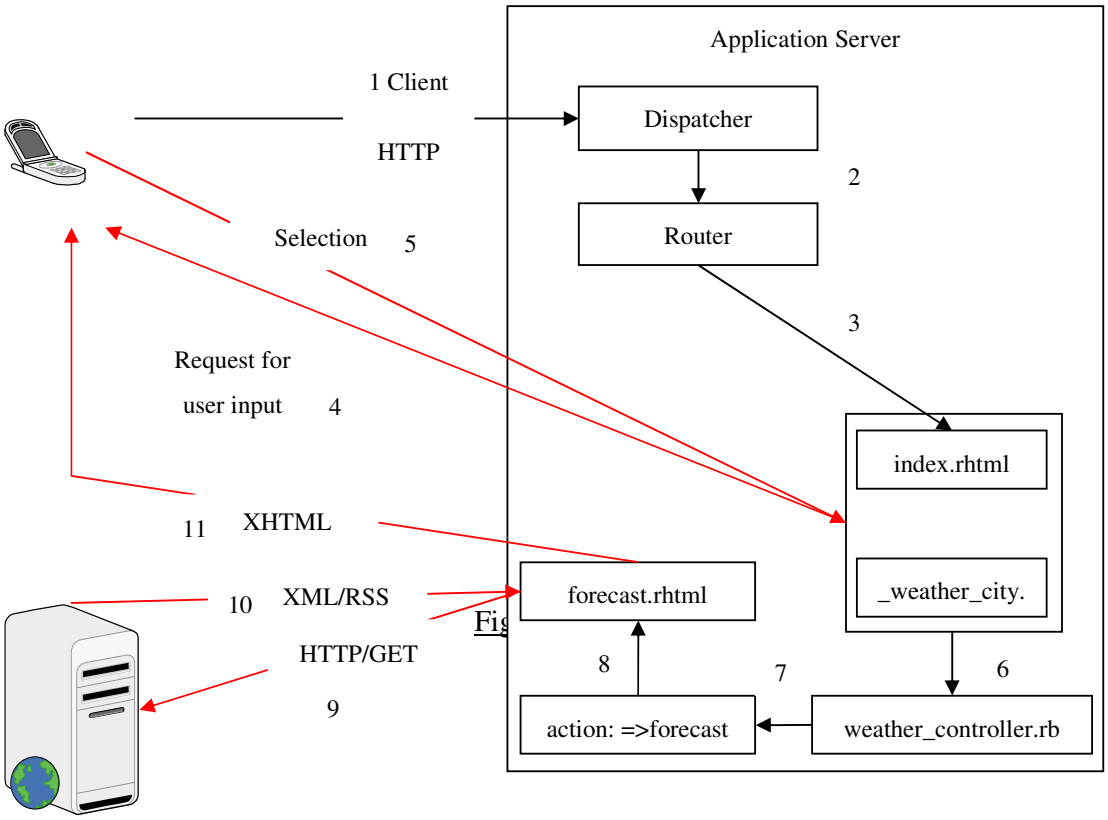


Figure 5.9: Weather application internals

Figure 5.9 shows the process of generation of the XHTML file, which is finally sent to the mobile client along with weather information. When a request arrives at the server (1), the dispatcher in Ruby on Rails passes the request down to the routing section to check for the appropriate controller which the request should be passed onto (2). As this request is made to the weather controller, the control is passed to the weather folder under 'view' of the, which contains the index.rhtml file (3). The content of this is simply a form which is presented to the user's mobile device (4). Once the selection are made and the form is submitted (5), the control is passed to weather_controller.rb (6), which in turn will trigger the 'forecast' action and store the parameters which the user has sent (7). Then, the forecast.rhtml is called with these parameters (8) and it will make the HTTP/GET request

to the Yahoo server (9). Finally the response from the Yahoo server (XML RSS feed) (10) is processed with REXML and the captured data is presented to the user in an XHTML file (11).

Figure 5.10 shows a screenshot of the form which is given to the user to select the temperature units and the city, and the resulting weather information is shown in the following figure 5.11.

Figure 5.10: weather page

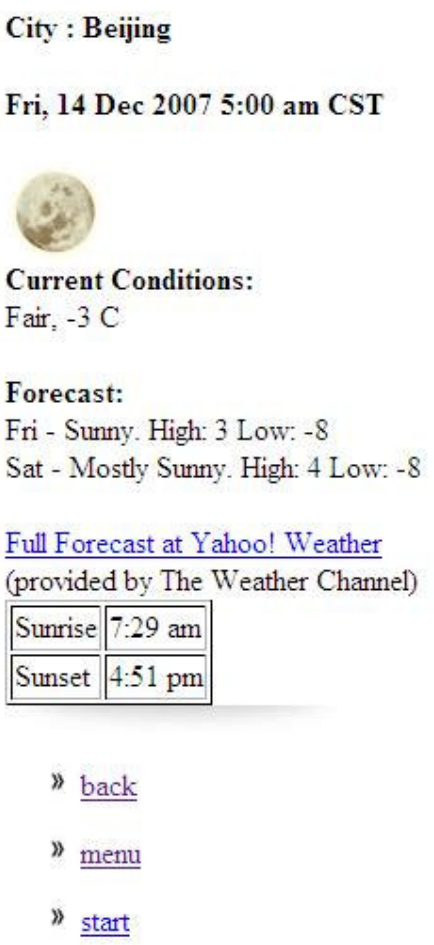


Figure 5.11 Resulting weather information

5.9 Currency converter implementation

Currency converter is implemented in a similar way to the weather application. The XML-formatted currency exchange rates are obtained from the WebservicesX.net [14]. It provides On Demand XML Web Services for Financial, Distribution, Retail, Health Care, Manufacturing, Telecom, Government and Educational Industry. With this service, conversion rate from one currency to another currency could be obtained. An example of a query is shown below, which requests for the exchange rate between Swedish Krona SEK and Chinese RMB [11].

http://www.webservicex.net/CurrencyConvertor.asmx/ConversionRate?FromCurrency=SEK&ToCurrency=CNY

Figure 5.12: Currency form page

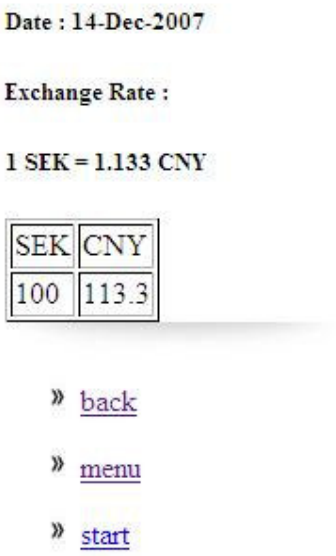


Figure 5.13: Resulting currency information

5.10 Database Design

The design of the database plays a crucial role in the new system, since web pages which are rendered based on the content that are fetched from the database. The idea behind the database is simple. It is designed considering the part types which would be included in a page such has headers, list items, images, URLs, text and tables. For each type database tables are created to store the data. An overview of the database design is given in figure 5.14.

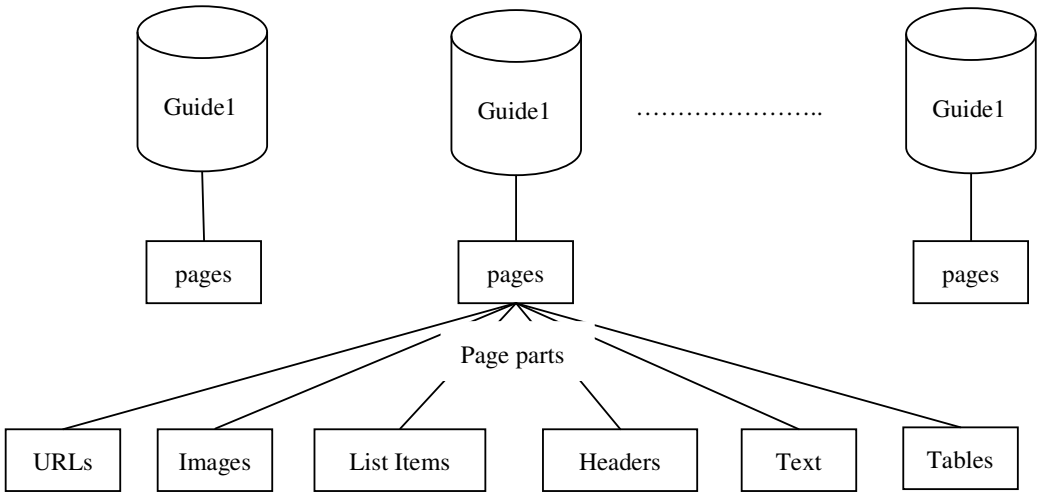


Figure 5.14: Database design

An application called ‘site-admin’ is developed to design guides and to store page data into the database. This is the same database which will then be used by the ‘guide’ application to render the requested pages. Some screen shots of the site-admin are given below in figures 5.15 and 5.16, respectively.

Figure 5.15: Site-admin login interface

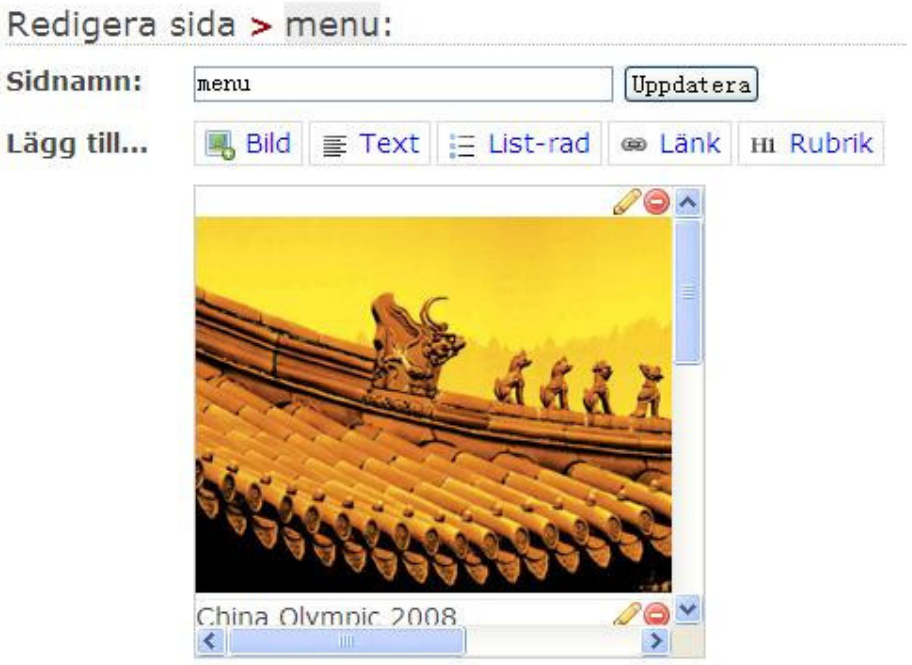


Figure 5.16: Designing a page for a mobile device with site-admin

Figure 5.16 depicts the page configuration interface of the site-admin application. The mid area represents the mobile screen so that the designers will have a feel how the actual page

will look like in a mobile device. The figure also shows the types of content that can be included in a page, for instance, images, text, lists, links etc.

For different parts of this database, different roles are assigned. These hierarchies of administration levels are listed below. However it should be noted that this functionality is not implemented with the current release of the site-admin application.

1) System administrator

A system administrator role has access to the whole database, and has full control over all the data. Tasks such as creating new guides, new users are permitted for this role. This is mainly a task which is allocated to a person at WIP.

2) Guide owner

A guide owner has access to one or several guides. He can view only the guides allocated to him, and none of the other guides' information is available to him.

3) Writers

For a guide, there can be one or several writes that writes content from via the site-admin application, but does not have permission to publish the content.

4) Authors

Authors review the content and gives consent to be published.

Chapter 6 Conclusion

We implement the whole structure of our mobile guide service and all the functionalities we designed are realized in our thesis. It enables our customers to travel to China easily and freely, especially during the time of the29th Olympic.

Unlike most menu driven services, our system is provided with the advantage of combining currency converter and weather forecast according to the latest information. In many existing applications, the features are scattered and might have to register with different providers to get different services. Our service can be considered as an all in one solution. The prototype project '2008 freedom China Traveler' is unique in a sense that it tries to provide an all-in-one service, an equivalent to a web portal for the customers without having to subscribe with many service providers. In fact, the services that are integrated into the project are not novel concepts but are customized and carefully selected to suite for normal travelers as well as who are attending special events as Olympics.

Along with the successful development of this Olympic year travel service, we can extend it to other popular events, such as the World Cup and / or as a general travel and event guide.

References

[1] Mika Klemettinen “Enabling technologies for mobile services”, The Atrium, Southern Gate, Chichester, John Wiley & Sons Ltd, West Sussex PO19 8SQ, England, 2007.

[2] http://en.wikipedia.org/wiki/GSM, Last seen on 1st February 2008.

[3] http://en.wikipedia.org/wiki/Universal_Mobile_Telecommunications_Systym, Last seen on 1st February 2008.

[4] http://en.wikipedia.org/wiki/General_Packet_Radio_Service, Last seen on 1st February 2008.

[5] http://en.wikipedia.org/wiki/Agile_software_development, Last seen on 1st February 2008.

[6] http://www.xplanner.org, Last seen on 1st February 2008.

[7] http://www.developershome.com/wap/detection/detection.asp?page=userAgentHeader, Last seen on 1st February 2008.

[8] http://en.wikipedia.org/wiki/Model-view-controller, Last seen on 1st February 2008.

[9] http://velocity.apache.org, Last seen on 1st February 2008.

[10] http://www.onjava.com/pub/a/onjava/2004/05/05/cg-vel1.html, Last seen on 1st February 2008.

[11] Mats Karlsson & Ola Nilsson, “WIP lecture_Mobile_Services”

[12] http://developer.yahoo.com/weather, Last seen on 1st February 2008.

[13] http://weather.yahoo.com, Last seen on 1st February 2008.

[14] http://www.webservicex.net, Last seen on 1st February 2008.

[15] http://www.developershome.com/wap/detection/detection.asp?page=profileHeade, Last seen on 1st February 2008.

[16] Deepak Vohra, “Ruby on Rails for PHP and Java Developers”, Springer Berlin Heidelberg Publisher, August 2007.

[17] http://www.rubyonrails.org, Last seen on 1st February 2008.

[18] Ian Sommerville, ” Software Engineering”, Addison Wesley; 6 edition, August 11, 2000.

[19] http://capinfo.com.cn, Last seen on 1st February 2008.

[20] http://wcities.net/en/mobile/78/mobile.html, Last seen on 1st February 2008.

[21] http://www.wip.se, Last seen on 1st February 2008.