UNIVERSITY OF PATRAS
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
ELECTRONICS AND COMPUTERS FIELD

DESIGN AND DEVELOPMENT OF A LOCATION AWARE EDUCATIONAL GAME FOR A MUSEUM BASED ON THE ANDROID PLATFORM

CITYSCRABBLE

DIPLOMA THESIS OF

DIEGO RODRÍGUEZ PUERTO

STUDENT OF THE UNIVERSIDAD DE VALLADOLID

SUPERVISOR: PROF. N. AVOURIS

DIPLOMA THESIS NUMBER
PATRAS MAY 2011

CERTIFICATION

It is certified that Diploma Thesis with the title:

DESIGN AND DEVELOPMENT OF A LOCATION AWARE EDUCATIONAL GAME FOR A MUSEUM BASED ON THE ANDROID PLATFORM

CityScrabble

Of the student of the University of Valladolid

Rodríguez Puerto	Diego
(Surnames)	(Name)

was presented in public at the Department of Electrical and Computer Engineering of the University of Patras, Greece on May 31, 2011.

The supervisor	The head of the Electronics and Computers division
N. AVOURIS	E.CHOUSOS

Abstract

In the present project I propose the design and implementation of a real time network multi-player game, for mobile devices based on the Android platform, using the programming language Java Android

For the development I used the toolkits of the official *Software Development Kit* (*SDK*), in the development environment Eclipse. Later, I tested it on a real device which uses *Android* as native operating system. They all are free distribution programs, without being necessary the obtaining license.

The application developed is a kind of Scrabble game in which several players can play in a real location. This game is a kind of game called location-based. The location of the game can be diverse, it can be from museums to entire cities (a tool is been implemented to do this feature). Each player will be able to choose a topic about which he plays in every moment, and about this topic he must answer a group of hints. If the player answers correctly a hint, it will add the score value of the hint to his score and all the players will be informed about it in real time.

In the way that the application is network based, it has been necessary the design of a communication protocol in a Server-Client architecture. I chose to implement a Web service based on REST architecture to realize this function. In order to implement the REST Web Service, it uses the framework Restlet for Java platform. Therefore it uses the Wi-Fi connection of the mobile to consume the Web Service.

The UI implemented has all the necessary features needed for game development. It has been designed trying to be as robust, intuitive and friendly as possible. It also has taken particular care that the load times are as small as possible. Due to the limitations of mobile devices, it has tried to optimize resource utilization of both processor and memory.

Key words: Mobile Game application, *wireless* technology, mobile devices, *Web Services, QR* Codes, *Android, Restlet*.

Acknowledgments

I would like to begin this report by thanking all the people who oneway or another contributed to the success of this work.

First, and most especially I would like to thank my supervisor *Christos Sintoris*, I am very much Indebted to him, for his support, motivation, encouragement and guide during these eight months. It was under his supervision I became interested in *Java*, *Android* and mobile devices development

I would like to thank the other members of the *HCI Group* for allowing me to work in their laboratory and making me feel very comfortable: *Ioannis Ioannidis*, who was all the days in the laboratory with me and for providing me with everything I needed; *Irene Chounta*, who was everyday in a pleasant mood; *Filio Vogiantzi*, who helped me when I had to make administrative papers. I want also to thank Professor *Nikolaos Avouris* (my *Erasmus* coordinator in Patra), *Dimitrios Raptis* and *Eleftherios Papachristos* for their good reception when I came to Patra. I would like thank *Francisco de Borja Sanchez Sancho*, who began this experience with me last September and helped me a lot at the beginning and *Samuel Herrero García*, who is finishing this experience with me, both of them Erasmus students of University of Valladolid.

There aren`t enough paper to thank to my several Erasmus friends, who made me have the greatest time of my life, I will not be able to forget you.

Finally I would like thank my family, especially to my parents, *Severiano*, who has been very sick the last months, and *Mercedes*, because without your support, love and sacrifices I cannot never achieve this.

Contents

Figure 1. Activated Android Devices

Chapter 1

Introduction

The year was 1958 when the American physicist William Higinbotham created Tennis for Two [1], one the first video games in the history. Since then there have been a lot of technological improvements which sooner or later have been used to create video games. And that is the entertainment industry is one of the great industries of present time.

The technological revolution in recent years has become possible with the wireless mobile devices. Therefore also this technology is used for creating video games. It must take into account the special characteristics of this technology in the design of applications for these devices

The aim of the thesis is the design and implementation of a network multiplayer game, for mobile devices based on the Android platform, using the programming language Java Android

The idea to develop this game begins in a previous game [2] that was developed by a member of HCI group of the Patras University. The game was implemented as a real time networked multi-player application. A central game server coordinates the handheld devices. The handhelds were pen-operated PDAs running Windows Mobile 5 with a VGA touch-screen, to which RFID readers were attached.

Now it wants to give a new dimension to the game. So the main goals of this project are: Design a new and modern UI for the game that takes advantages of the possibilities of the new technologies; Implement all the features that the game has in the Android platform; create a Web Service that coordinates the handheld devices; and extend the game scenario along entire cities using new tool that will be implemented in the future.

This game is a location-based game (or location-enabled game), that is one in which the game play somehow evolves and progresses via a player's location. Thus, location-based games almost always support some kind of localization technology, for example by using satellite positioning like GPS. "Urban gaming" or "Street Games" are typically multi-player location-based games played out on city streets and built up urban environments.

In order to extend the scenarios of the game to entire cities, it will use a new tool to create the different environments or scenarios of the game.

To implement the Web Service that coordinates the mobile devices, we will take the REST architecture using the Restlet framework.

To fulfill the other two goals of the game it needs known the features of Android platform. Due to this, it was necessary a previous work in order to familiarize with this technology. After seeing the fast growing of this platform in recent months, due among other factors to be an open source platform, the future looks promising for applications based on this platform. Also the technological improvement of mobile devices in recent years has been very significant. The size and the resolution of the new devices is much bigger, also the new touch screen gives to the developer new possibilities that before it didn´t have, to improve the playability of the games on these devices at the same time become it more attractive.

In addition, these devices have the possibility to connect to new data networks via Wi-Fi, this will provide new features that can be used to create applications.
The way to do this is developing and evaluating new Android applications on the Android emulator from the official Software Development Kit (SDK) in the IDE Eclipse. Later test it on a real device which uses Android as native operating system.

Chapter 2

Theory and Analysis

2.1 Introduction

In this chapter we go to analyze some of the tools that are used to develop and implement the application.
First it goes to discuss the convenience or not to use the android platform, and it will show the advantages and disadvantages of its use.
Then it will describe the mobile device which will be used to implement the application

2.2 Why Android

Since October of 2008 T-Mobile released the G1 by HTC, the first commercially available mobile device running on the Android operating system, the growth of Android has been unstoppable. For consumers, this marked the first opportunity to use a new kind of handset born of the Open Handset Alliance, a Google-led collaboration of now more than 79 member companies who pledged to support open standards for mobile devices. With OEMs such as HTC, Motorola, Samsung and LG all adopting the Android OS, it expects a great success of Android.

According to a study realized by the company of consultancy Flurry [4] a through its analytics service, it calculates that the amount of mobile devices based in Android O.S that have been activated between 2009 and 2010, it increased dramatically from 5.9 million to 53 million in only one year, which means that in one year is multiplied by 10 the number of devices.

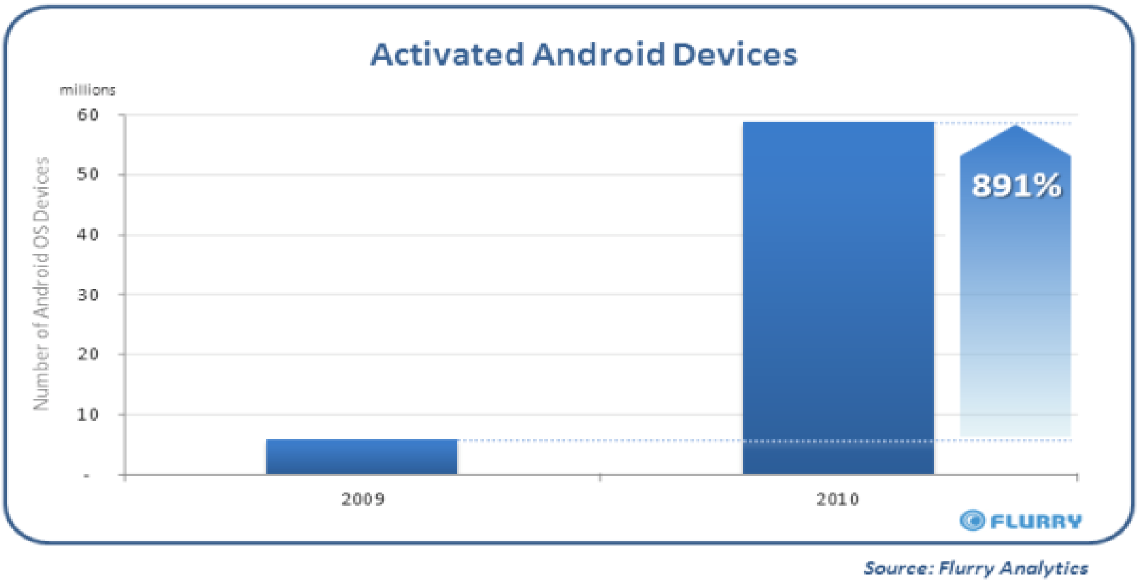


Figure 1. Activated Android Devices

The main advantages of Android are:

- Android can run on every mobile device and not for the mobile device itself
- OS designed to develop Mobile Applications
- Strategy: reach as much people as possible. Provide an affordable OS for everybody.
- It’s free. Android SDK: Open Source Platform
- Based on Linux Kernel
- Increase of stability, performance and security
- Programming based on Java

Due to this spectacular growth, which ensures a large market for the applications for these devices, and the features that provides android, this project has been realized with Android.

2.3 HTC Desire

The mobile device used to test the application has been the **HTC Desire** (codenamed Bravo) that is a smartphone developed by the HTC Corporation, announced on 16 February 2010 and released in Europe and Australia in the second quarter of the same year. The HTC Desire runs the Android operating system, version 2.2 "Froyo". Android, version 2.3 "Gingerbread" update coming in May or June 2011. This is a variant of *Google's Nexus One*

Figure 2. Mobile HTC Desire

The smartphone used was provided by the HCI Group of the University of Patras. Below are shown the characteristics of this device.

Dimensions, weight and display

HTC Desire weighs 135gr with the included battery and his dimensions are 119 x 60 x 11,9 mm. Desire comes with a tactile screen AMOLED of 3,7 inches that offers a maximum of 16M colours in a resolution WVGA of 480 x 800 pixels.

Camera

HTC Desire comes with a camera of 5 megapixels with characteristics as auto focus, flash LED, geo-labelled of photography and detection of smile and face. This camera is compatible with recording video WVGA to 15fps and with the HTC Desire we can reproduce video in MPEG4, H.263, H.264 and WMV9 formats and also to record sequences of video.

Audio

HTC Desire has an integrated music player that supports WAV, MP3, AAC +, and WMA9 formats and takes a service of integrated e-mail that supports push e-mail and there comes with a service of messages multimedia (MMS) and a service of message of text (SMS).

Connectivity

It works in GSM and HSDPA networks and provides an excellent selection of Internet function as Google's search and compatibility with YouTube. It is necessary to say that The HTC Desire comes with connectivity Wi-Fi, a function of navigation GPS with support A-GPS with Google Map’s support, a digital built-in compass and is compatible with GPRS and EDGE. We can transfer information by means of a wireless connection Bluetooth or a connection of cable micro USB.

Memory

As for the memory, HTC Desire comes with 512MB of ROM memory and 576MB of RAM memory being extendable up to 32GB for micro SD.

Battery

The battery supplied with the HTC Desire is standard Li-ion with a capacity of 1400 mAh that give us a waiting time of up to 340h (2G or 360h (3G and one time in conversation of up to 6,40h (2G or 6,30h(3G).

Sensors

The device has the following sensors that you can use also to develop applications: G-Sensor, Digital compass, Proximity sensor, Ambient light sensor

Network

Dual-band (800 and 1900 MHz) CDMA2000 1xRTT/1xEVDO/1xEVDO rev. A and IS-95A/B voice or data with up to 1.8 Mbps uplink and 3.1 Mbps downlink speeds.(Band frequency and data speed are operator dependent.)

Chapter 3

Design

3.1 Introduction

The way to design an application for mobile devices isn´t the same that for desktop devices. It should think the special features of these devices, and the way for design them is different. Mark Dunlop and Stephen Brewster[3] collect some of these design standards:

- Designing for mobility: as users are mobile they will not have many of the props around them to support work (e.g. notes on desks), will need to work with small devices, are likely to have a far from ideal working environment and this environment will change drastically as the user moves;
- Designing for a widespread population: users will not normally have any formal training in their technologies and consider them as devices to be used rather than computers to be maintained;
- Designing for limited input/output facilities. Screen sizes will improve in resolution in terms of color support and pixels per cm, but will always be small due to the need for portability. Sound output quality is often very poor with restricted voice recognition on input. Keyboards are limited in size and number of keys and other pointing devices are often hard to use when on the move.
- Designing for (incomplete and varying) context information: through various sensors and networks, mobile devices can be made aware of their context (e.g. current location through the Global Positioning System). This gives new information to the systems but brings problems of implying task and user level activities from sensor information and unreliable or patchy sensor coverage. Work on position aware tourism guides, for example, highlight many of these problems.
- Designing for users multitasking at levels unfamiliar to most desktop users: multitasking and support for task interruption is one of the keys to successful desktop design with mobile devices, the opportunities for, and frequency of, interruption are likely to be much higher, given the environments in which the devices will be used.

3.2 Designing the new application GUI

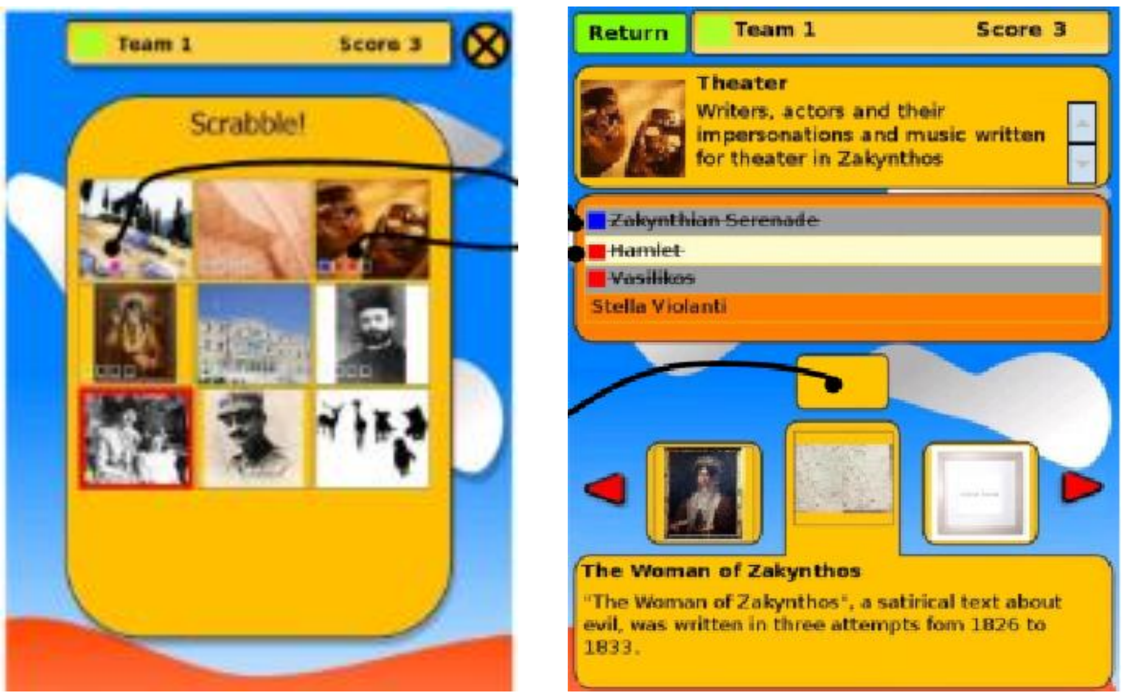
As mentioned previously, the idea of this game starts from another game [2]. The previous game was not tested before the execution of this project. We were available only the screenshots of the game and a document explaining how it worked. Therefore the first game was used to take an idea about the game, but it didn´t help to develop the code.

The previous game was designed to be played in the Museum of Science and Technology of the University of Patras, Greece. This museum is linked with Museum Solomos and Eminent Zakynthias, placed in the island of Zakynthos, Greece.

When the screenshots of the game were analyzed, one of the first things that I wanted to improve was the UI. Another objective will be extending the scenario of the game to entire cities.

Now it wants to take advantage of the new features of the Android platform and the features of the new mobile devices.

Below it shows the UI of the previous game:



Writers, actors and their impersonations and music written for theater in Zakynthos

"The Woman of Zakynthos", a satirical text about evil, was written in three attempts fom 1826 to 1833.

Figure 3. First and second screens in the game MuseumScrabble

3.2.1 First Design

At the beginning the game was designed with similar elements to the previous game.

3.2.1.1 Scrabble screen

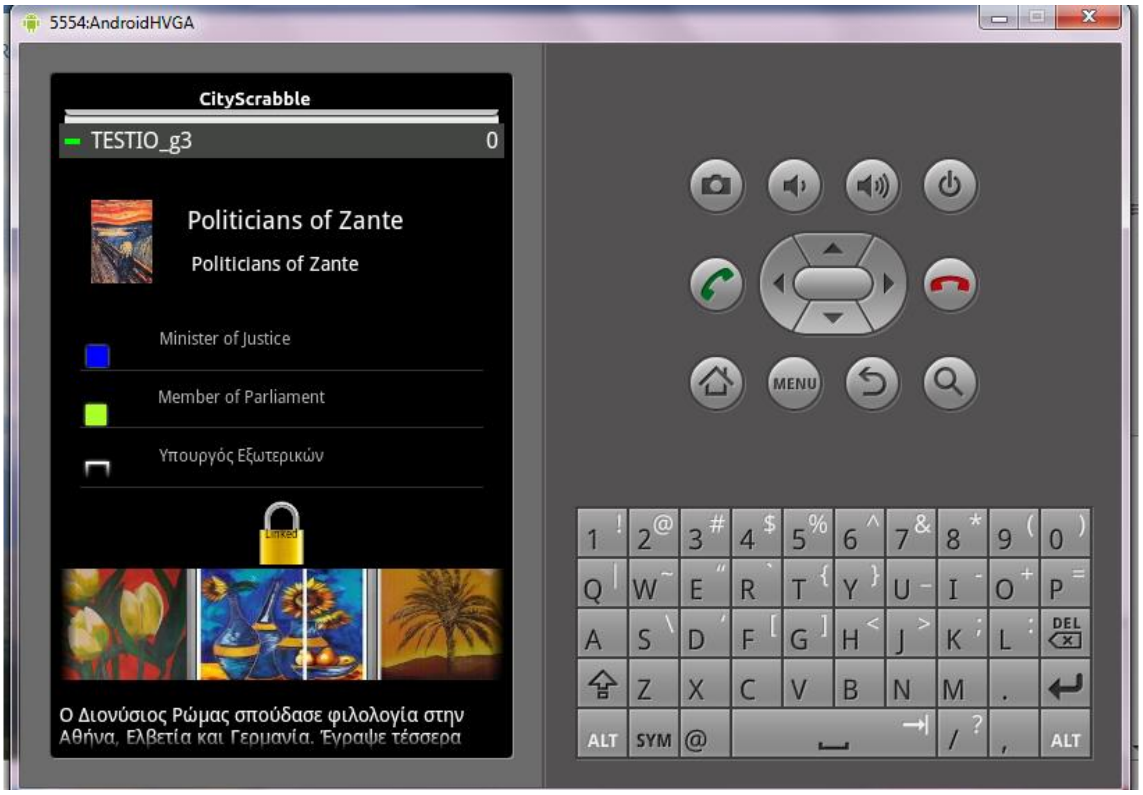
The first design for the Topic screen contained the following elements:

- A ScoreBar that shows the color of the player, the name of the player and the score
- Nine image buttons customized that represented the nine topic of the game
- Each image button had four flags to show the state of the hint

3.2.1.2 Topic screen

The first design for the Topic screen contained the following elements:

- A ScoreBar that shows the color of the player, the name of the player and the score
- A small thumbnail of the topic with a title and the description of the topic
- A ListView that shows the hints and the flags or boxes which inform about the status of the hint
- A Toggle button that is used to link the hints with the exhibits
- A Gallery that shows the exhibits
- A text that describes the exhibit selected



Ο Διονύσιος Ρώμας σπούδασε φιλολογία στην Αθήνα, Ελβετία και Γερμανία. Έγραψε τέσσερα

Figure 4. First design of Topic screen

3.2.2 Final Design

After an initial design based on a previous game [2], we proceeded to a second design more functional, robust and modern that takes advantage of the features offered by Android, and that makes it more attractive for the user.

The principal improvements were made in order to improve the speed of the game, loading times, the organization of widgets on the screen, the gameplay and provide to the player with more feedback to receive more information from the status of the game.

In order to re-design the application, it proceeded first with the Scrabble screen and then with the Topic screen.

3.2.2.1 Scrabble screen

The Scrabble screen design has been the least change has experienced. Only little details were introduced in order to improve the load times, the feedback provided to the player and other features.

The first task done was the change of the nine images buttons of the topics, which were on the Relative Layout, for a GridView that contains these images. This task is done on the class TopicTileSet. Due to this improvement it got a better screen structure and a better load time. In addition, the images now are joined in the same widget, getting a better management.

In a similar task, the linear layout that grouped the different components of the state informing to the player of the game was replaced with for a Scorebar. Scorebar is a class that extends the View class and will enable many more design options due to implementation of the OnDraw method.

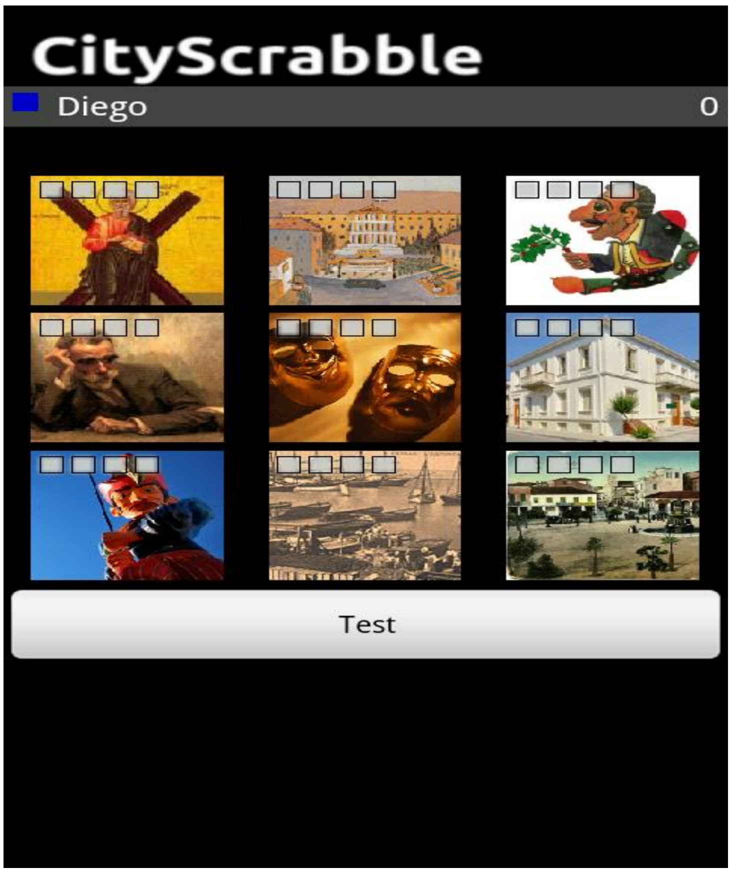
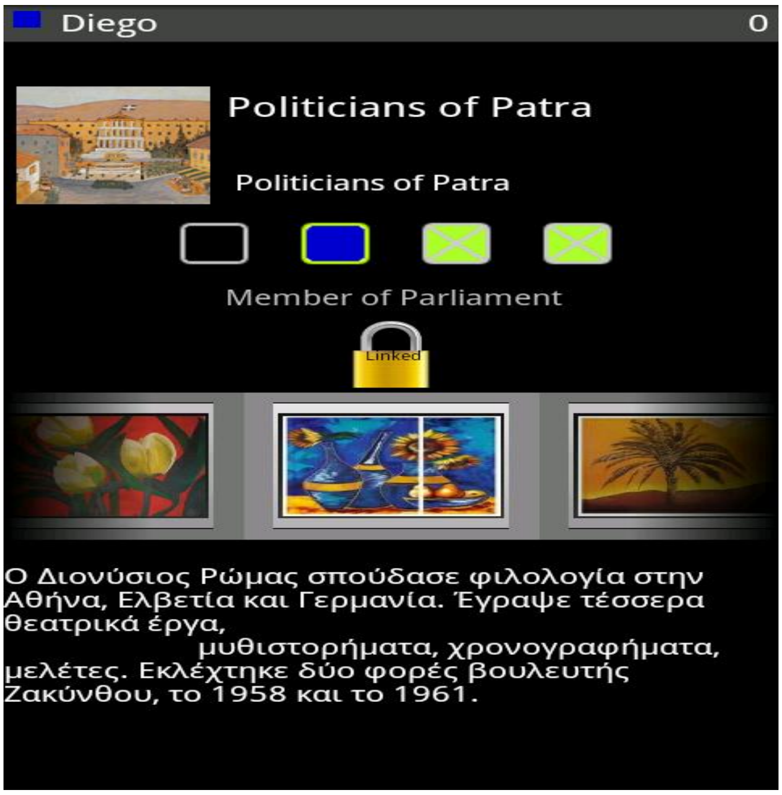


Figure 5. Final design of Scrabble screen

3.2.2.2 Topic screen

For the second screen we have designed a new UI after to implement all the necessary features of the game, such as communications, management and updating of the game status…To do this, some of the widget were re-designed, getting a new UI.



Ο Διονύσιος Ρώμας σπούδασε φιλολογία στην Αθήνα, Ελβετία και Γερμανία. Έγραψε τέσσερα θεατρικά έργα,
μυθιστορήματα, χρονογραφήματα, μελέτες. Εκλέχτηκε δύο φορές βουλευτής Ζακύνθου, το 1958 και το 1961.

Figure 6. Final design of Topic screen

The main change realized was to replace the ListView with a new customize control that consists in a Radio group of radio buttons that is explained in the section 4.2.7.
The reasons to change the first design were because the first design had various problems like the size of the control that took up a lot of space on the screen, or on the other hand, if the size of the ListView was limited with one scrollbar, there were hints and flags hidden on the screen.

Finally the design of the topic screen contains the following elements:

- A ScoreBar that shows the color of the player, the name of the player and the score
- A small thumbnail of the topic with a title and the description of the topic
- A radio group of radio buttons that represents the hints
- The text of the hint that has been selected on the radio group
- A Toggle button that is used to link the hints with the exhibits
- A Gallery that shows the exhibits
- A text that describes the exhibit selected

When the game starts and the player has not decoded yet the exhibits with the camera of the mobile using the QR codes, the exhibits will appear empties and the button that let to use the camera, will be available and visible.

Now we get three objectives:

- We get more space free on the screen
- Only text of one hint is visible now at the same time. Therefore the player has clear which hint is selected
- The player can see all the flags of the hints.

Finally the gallery has been re-designed, spacing more the exhibits between themselves. Also other improvements have been introduced to improve the feedback of the player

3.3 Communications

The image below shows the general architecture of the communication in the application. In the following sections all of these communications will be explained.

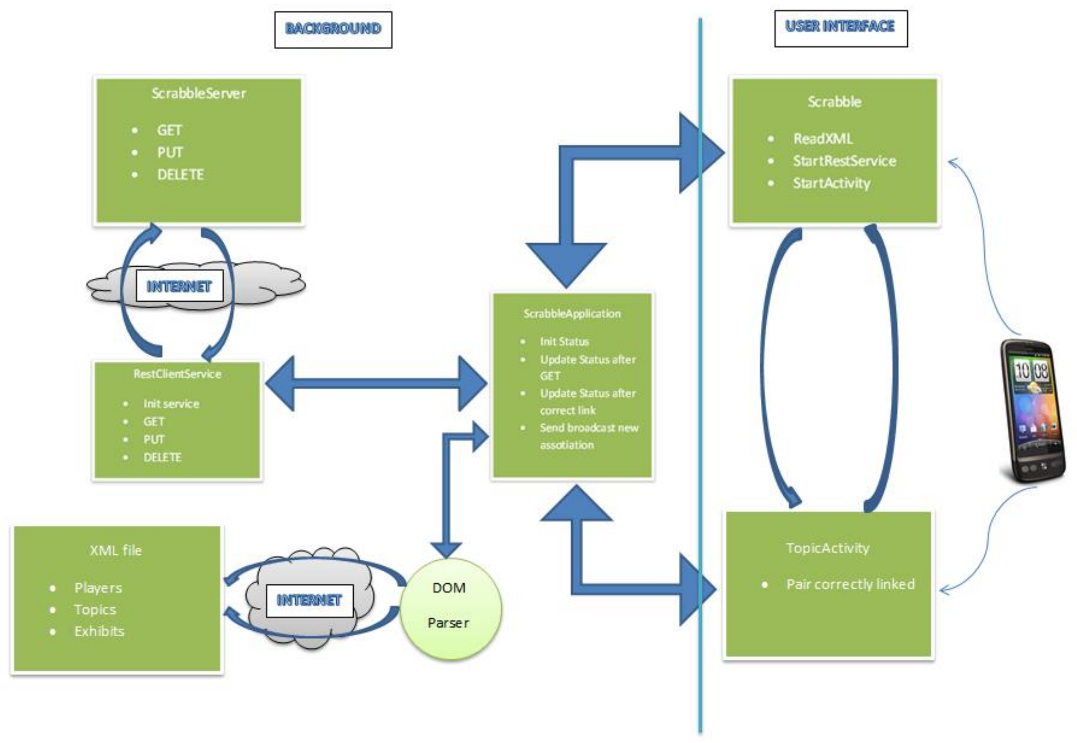


Figure 7. General scheme of the application

3.3.1 Introduction

One of the most complex parts of the application is the communication. The application needs to know the different stages of the classes and variables to manage itself, so communication is needed between classes and resources. There are different types of communications that relate to the different requirements of the application and which information is needed. There are communications as varied as Intents, requests client-server using a Restlet Web Service, requests URL to download a file from a server...

The game consists in two screens that correspond with the two activities of the application. These two activities have to communicate between themselves and also with other different modules of the application. The majority of the communications can see on the figure 7.

The communication between activities will do through intents. The intents allow to change the activity; to pass from the first screen to the second the player has to press a topic; and to pass from the topic activity to scrabble activity the player has to press the return button of the mobile device. This is possible because the screens in Android are overlapped on each other like a stack. In the section 4.2.2.1 are explained the intents

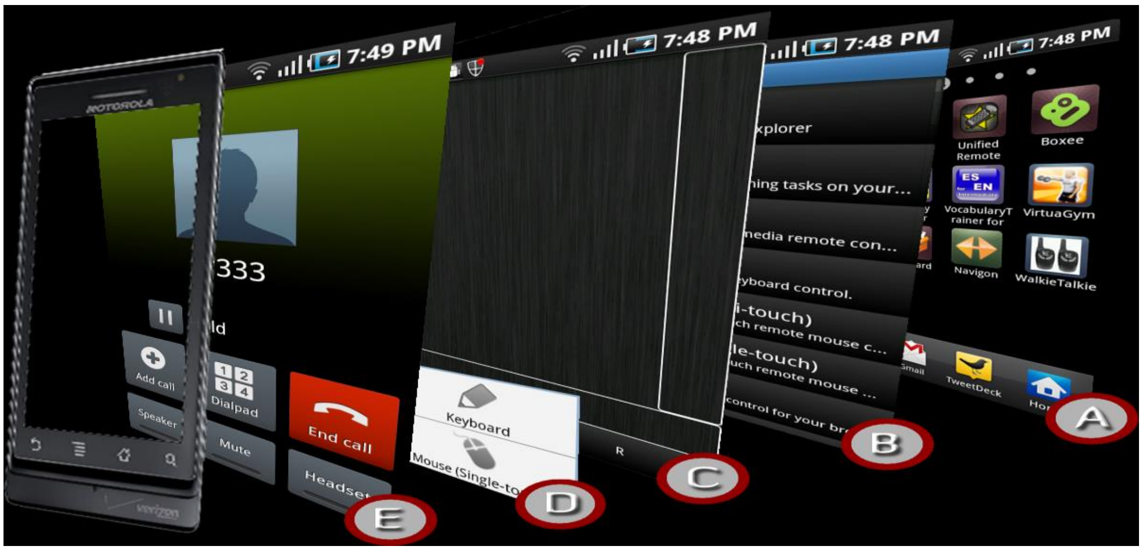


Figure 8. Intents in the applications

3.3.2 Web Services

3.3.2.1 Introduction

A web service is a method of communication between two electronic devices over a network.

The W3C defines a "web service" as "a software system designed to support interoperable machine-to-machine interaction over a network. Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks.

A Web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided. For example you can implement a Web service using one agent written in one language or another agent in a different language, but if the functionality is the same, the Web service remains the same.

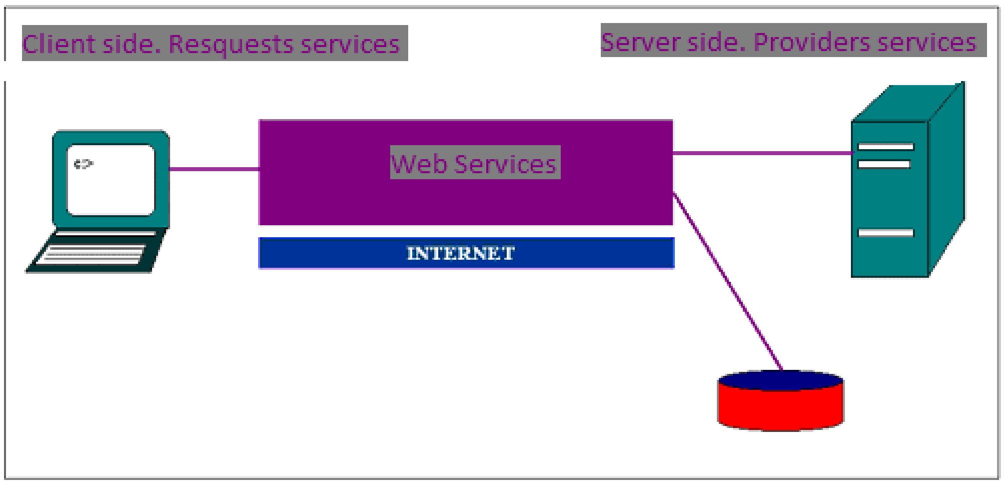

Client side. Resquests services
Server side. Providers services
Web Services
INTERNET

Figure 9. Web Services

3.3.2.2 Using Web Services

In this project, players must know details of other players, like for example, the hints that have been answered, and this must be known at the real time. At the same time the game has to know the status of all players for their management.

There are different ways of doing this as databases or web services. In this project we have decided to use Web services for this.

Therefore we will use Web services for the application to know at all times that hints have been or are being answered and the players that have answered.
To do this when a player correctly answers a hint to the corresponding exhibit, the application sends information to the web service that hint has been answered and the ID of the player who has responded to disable other players for this hint. On the other hand, the application will ask the Web services that hints have been answered and that the players have responded to disable and therefore players may only respond to the hints available.

3.3.2.3 Different philosophies: soap restful…

The definition of Web Services proposal relate to many different types of systems, but have in common the use of clients and servers that communicate between them. The world of Web Services has been practically a monopoly of SOAP Web Services architecture; most Web Services were designed using a SOAP framework.

In recent years it has become popular software architectural style known as REST (Representational State Transfer). This new style has brought a new style option of using Web Services. Listed below are the three most common types of uses:

- Remote procedure calls (RPC)

RPC Web services present a distributed function (or method) call interface that is familiar to many developers. Typically, the basic unit of RPC Web services is the WSDL operation.

The first Web services tools were focused on RPC, and as a result this style is widely deployed and supported. However, it is sometimes criticized for not being loosely coupled, because it was often implemented by mapping services directly to language-specific functions or method calls. Many vendors felt this approach to be a dead end, and pushed for RPC to be disallowed in the WS-I Basic Profile.

- Service-oriented architecture (SOA)

Web services can also be used to implement architecture according to service-oriented architecture (SOA) concepts, where the basic unit of communication is a message, rather than an operation. This is often referred to as "message-oriented" services.

SOA Web services are supported by most major software vendors and industry analysts. Unlike RPC Web services, loose coupling is more likely, because the focus is on the "contract" that WSDL provides, rather than the underlying implementation details

- Representational state transfer (REST)

REST attempts to describe architectures that use HTTP or similar protocols by constraining the interface to a set of well-known, standard operations (like GET, POST, PUT, DELETE for HTTP). Here, the focus is on interacting with stateful resources, rather than messages or operations.

3.3.2.4 Using Rest

One of the reasons for not working with RPC is because is it not as loosely coupled style and be a style more and more obsolete and that developers are advised to go away from using. The RPC is more appropriate for isolated environments, where the environment is perfectly known. Developments in this type of system is simple, each user is modified to meet the new requirements.

A different situation will be when you have many users who consume the Web service, this will require using a different strategy. It needs to propose an explicit mechanism for the interoperability of systems that do not have the same API. The protocols that follow the strategy RCP are not suitable for this type of system, since it is difficult to change its interface.

REST and SOAP are often termed "Web services", but they are totally different approaches. REST is an architectural style for building client-server applications. SOAP is a protocol specification for exchanging data between two endpoints. Each one has its advantages and disadvantages. Since the REST architecture proposed, it was considered as an alternative to SOAP, having a fierce debate between proponents and detractors of either philosophy.

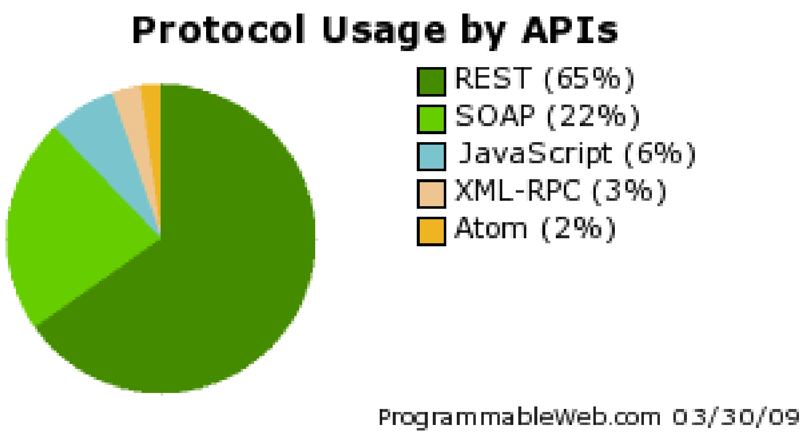


Figure 10. Protocol usage by APIs

On the images below are showed the main features of both philosophies:

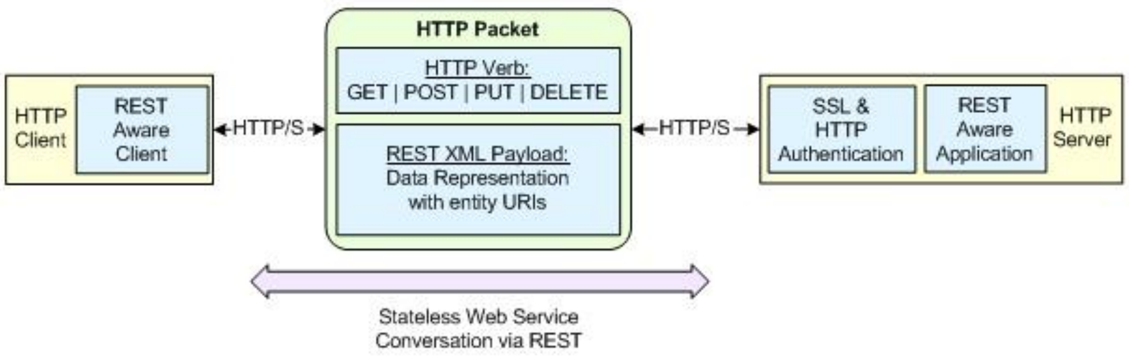

HTTP Client
REST Aware Client
HTTP/S
HTTP Packet
HTTP Verb:
GET | POST | PUT | DELETE
REST XML Payload:
Data Representation with entity URIs
HTTP/S
SSL & HTTP Authentication
REST Aware Application
HTTP Server
Stateless Web Service Conversation via REST

Figure 11. REST architecture

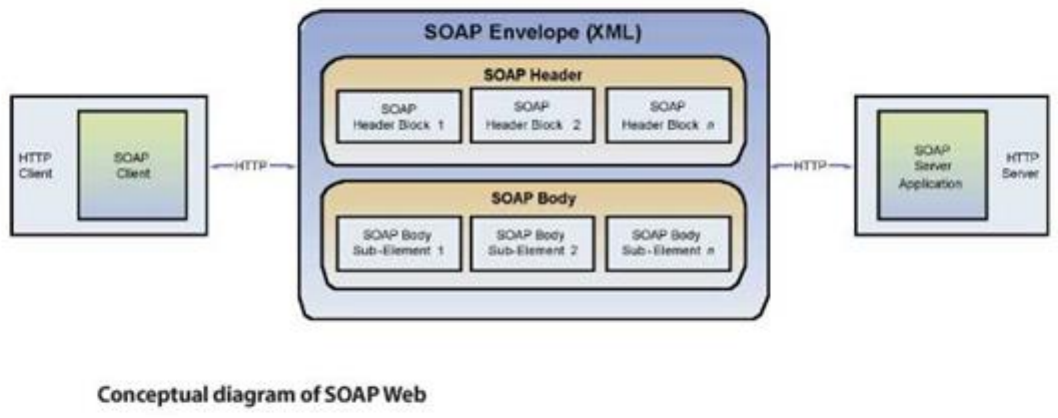


Figure 12. SOAP architecture

The main differences in the functioning of both are:

- SOAP is an architectural style-oriented RPC (Remote Procedure Call), ie, a style based on remote procedure calls, while for REST HTTP only exists methods and is focused on resources.

- REST does not allow the strict use of "session" since by definition is stateless, while for SOAP, being oriented RPC, Web services create sessions to invoke the remote method.

- SOAP uses HTTP as a tunnel that passes messages, uses XML to encapsulate data and functions in the messages. If we draw a protocol stack, SOAP would be just over HTTP, while REST proposes HTTP like the application level

One of the main advantages of SOAP is to be tightly coupled, allowing to be tested and debugged before starting the application. On the other hand, the advantages of REST-based deployment are the potential scalability of such systems, as well as access to low consumption of resources to its operations due to the limited number of operations and unified addressing scheme.

The application will be used in mobile devices where the consumption of resources is a critical point, and also the bandwidth is important and needs to be limited. REST is especially useful on devices with limited resources, where the overhead of the headers and additional layers of SOAP elements should be restricted. So it is implemented using the REST style. The REST architecture is simpler, that is precisely one of its main attractions, as the developer never worked with Web services and their use is only part of the application.

3.3.3 Rest

Roy Thomas Fielding first coined the term REST as a concept in his PhD dissertation [5]. He was one of the people who worked on the specification that drives most of the Internet today: Hypertext Transfer Protocol (HTTP).

Arguably, the Web can be regarded as the largest, most scalable, and most popular distributed application of all time. The constraints of REST are based on the same underlying principles that govern the Web. Those principles are:

- User agents interact with resources, and resources are anything that can be named and represented. Each resource can be addressed via a unique Uniform Resource Identifier (URI).

- Interaction with resources (located through their unique URIs) is accomplished using a uniform interface of the HTTP standard verbs (GET, POST, PUT, and DELETE). Also important in the interaction is the declaration of the resource's media type, which is designated using the HTTP Content-Type header. (XHTML, XML, JPG, PNG, and JSON are some well-known media types.)

- Resources are self-descriptive. All the information necessary to process a request on a resource is contained inside the request itself (which allows services to be stateless).

- Resources contain links to other resources (hyper-media).

3.3.3.1 Framework restlet

Restlet is a comprehensive yet lightweight RESTful web framework for Java that lets you embrace the architecture style of the Web (REST) and benefit from its simplicity and scalability.

Restlet has a light core but thanks to its pluggable extension, it is also a comprehensive REST framework for Java. It supports all REST concepts (Resource, Representation, Connector, Component, etc.) and is suitable for both client and server Web applications. It supports major Web standards like HTTP, SMTP, XML, JSON, WADL, and Atom. Many extensions are also available to integrate with Servlet, Spring, Jetty, Grizzly, Simple, JAXB, JAX-RS, JiBX, Velocity, FreeMarker, XStream, Jackson, SLF4J, ROME, Netty.

Several editions for GWT, GAE, Android, Java SE and Java EE are also available and keep synchronized with an automated and unique porting process
Restlet is a very flexible framework that is suitable for most Web application developments.

Restlet can be used for both client-side and server-side applications and of course for applications that act as both clients and servers. With the availability of a port for the GWT platform, it can also works in a Web browser.

3.3.3.2 Architecture

The Restlet Framework is composed of two main parts. First, there is the Restlet API, a neutral API supporting the concepts of REST and HTTP, facilitating the handling of calls for both client-side and server-side applications. This API is backed by the Restlet Engine and both are now shipped in a single JAR ("org.restlet.jar").

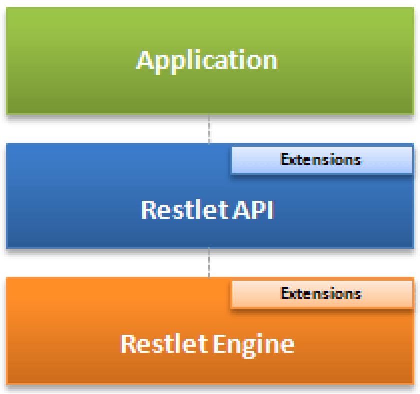


Figure 13. Restlet architecture

Restlet framework supports the standard REST software, and also it provides a set of classes that simplify the hosting of multiple applications within a single JVM(Jave Virtual Machine). The advantage is to provide a RESTful, portable and more flexible alternative to the existing Servlet API. In the diagram below, we can see the three types of Restlets that are provided in order to manage these complex cases. Components can manage several Virtual Hosts and Applications.

Virtual Hosts support flexible configuration where, for example, the same IP address is shared by several domain names, or where the same domain name is load-balanced across several IP addresses. Finally, it uses Applications to manage a set of related Restlets, Resources and Representations. In addition, Applications are ensured to be portable and reconfigurable over different Restlet implementations and with different virtual hosts. In addition, they provide important services like access logging, automatic decoding of request entities, configurable status page setting and more.

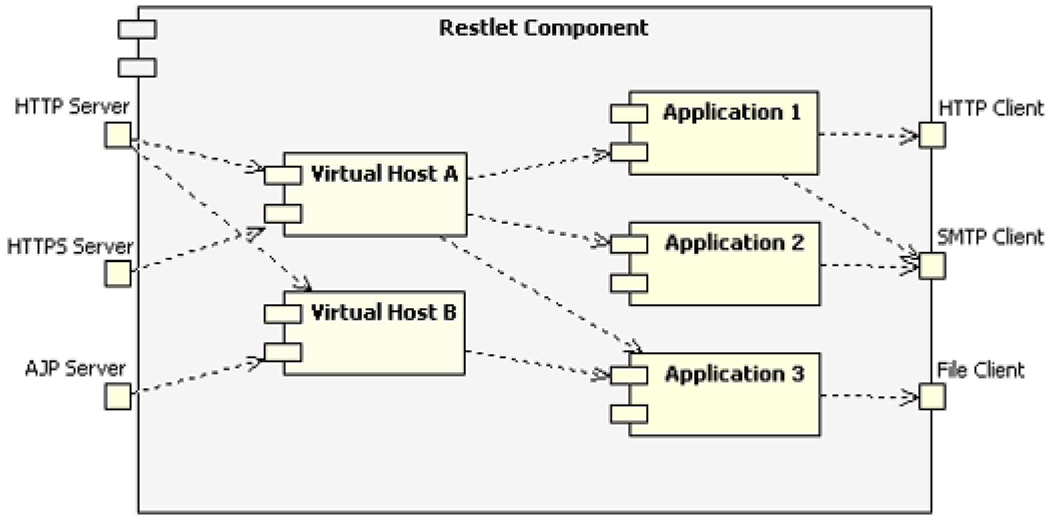


Figure 14. Restlet components

3.3.3.3 Design client-server Restlet

Below it describes the steps followed in order to design the Web Service that is used in the application. It consists in a client and a server. The server stores the necessary data to manage the application and the client makes requests to the server in order to retrieve and store data. The client also can send a request to delete data stored in the server.

The first step is to define the URIs of the application. The resources of the application must be identified, and then they go to be associated with their URIs

Two resources have been defined:

Se han definido dos recursos:

- GameMessage>> it consists in an object that has the variables hintId, exhibitId and playerId. With these data it can identify totally what Hint has been answered and the player who has answered it.

- GameMessageList>>it consists in a list of GameMessage resources

It mustn´t work directly with the resources if it follows strictly the REST architecture, but it must work with representations of the resources, in formats like XML, HTML or JSON.

In this application it works directly with the resources, not with representations of them. This doesn´t fulfill the principles of REST architecture, but for simplicity it will work in this way.
GameMessage resource:

```
public class GameMessage implements Serializable{
    private static final long serialVersionUID = 1L;

      private int hintId;
      private int playerId = -1;
      private int exhibitId = -1;
      private int isActive = 0; // instead of delete. if isActive==0,
then link is// deleted, otherwise active

      public GameMessage() {
      }

      public GameMessage(int playerId, int HintId,int exhibitId) {
            super();
            this.playerId = playerId;
            this.hintId = HintId;
            this.exhibitId=exhibitId;
      }
// getters and setters omitted for brevity
      …
public void setActive(boolean active) {
            if (active)
                  isActive = 1;
            else
                  isActive = 0;
      }

      public Boolean isActive() {
            if (isActive == 1)
                  return true;
            else
                  return false;
      }
public boolean isEqual(GameMessage message) {
            if ((message.hintId == this.hintId) &&
                 (message.playerId == this.playerId) &&
                 (message.exhibitId == this.exhibitId)) {
                  return true;
            }
            return false;
      }
```

Listing 1. GameMessageResource

List of GameMessage resources:

```
public static ArrayList<GameMessage> gameMessageList = new
ArrayList<GameMessage>();
```

Listing 2. List of GameMessage resources

Now it has to define how the resources go to be referenced. With the URIs and with the methods supported, the access will be possible. IT possible to make the access by a lot of ways, retrieving the representation of a resource(GET or PUT), adding or modifying a representation(POST or PUT) and removing some or all representations(DELETE).

The list of methods allowed on each Resource is:

- The "GameMessageResource" responds to GET requests with an ArrayList of GameMessages that lists the currently registered game messages.
- PUT request update and store a new GemaeMessage in the ArrayList of GameMessages.
- DELETE request remove a GameMessage of the ArrayList of GameMessages

It is necessary explains better how PUT and DELETE methods work. At the first time the GameMessage was stored and removed directly in the ArrayList, but in this way there were a lot of problems with the threads of the application. To resolve this is done following:

- If it wants to be stored GameMessage by the first time, it stores in the ArrayList and the variable isActive, which is part of the GameMessage, is updated to 1.
- If it wants to delete a GameMessage, which is in the ArrayList of the server, we don´t go delete it, only we go to update its variable isActive to 0.
- When it wants to store a GameMessage which has been stored before and after this it has been deleted, only it will be necessary to update its isActive variable to 1; we don´t need store the GameMessage again

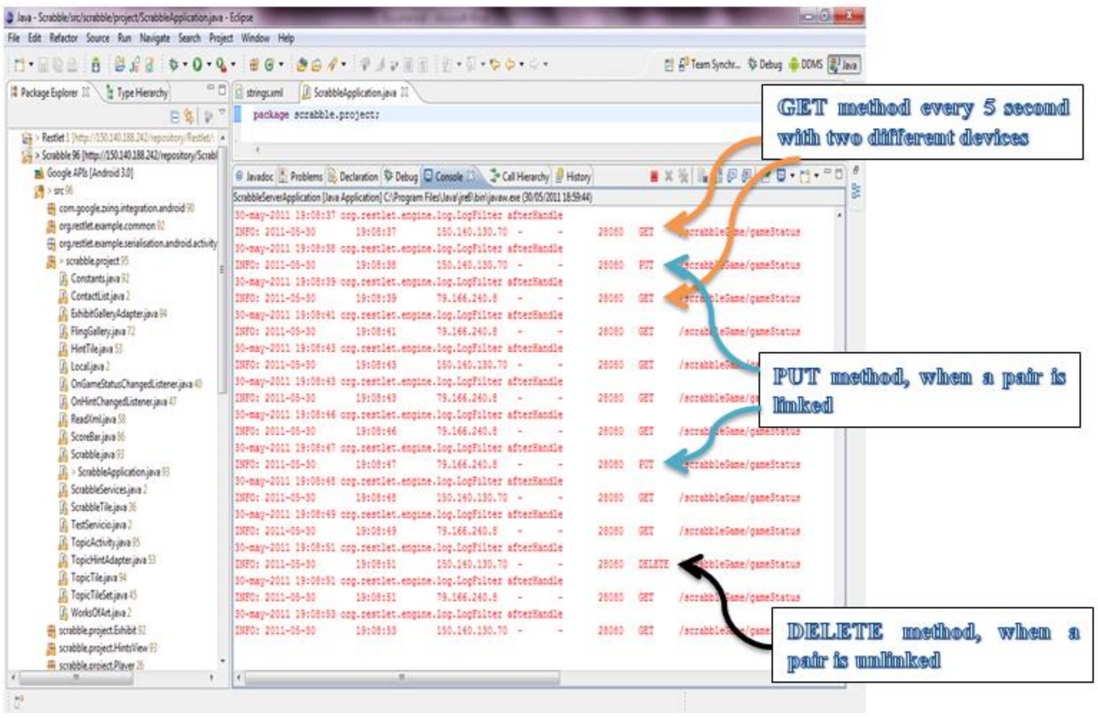
There are two resources; therefore two URIs should be defined. This will be not necessary due to for the methods implemented which fulfill with the requirements of the application, only they need to work with GameMessage resource.

GET>>return Arraylist<GameMessage>
PUT>>store GameMessage in Arraylist<GameMessage>
DELETE>>remove GameMessage in Arraylist<GameMessage>

For example, if it wants to retrieve a unique GameMessage it should define another GET method which returns it, and would be necessary to implement another URI.
Below it shows the URI that identify the resource:

SERVER_URL + "/scrabbleGame/gameStatus""

In the image below it`s possible to check the communications between the Rest client on the application with the REST server through request Web Service


GET method every 5 second with two different devices
PUT method, when a pair is linked
DELETE method, when a pair is unlinked

```
package scrabble.project;
```

```
ScrabbleServerApplication [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (30/05/2011 18:59:44)
30-may-2011 19:08:37 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:37	150.140.130.70	-	-	28080	GET	/scrabbleGame/gameStatus
30-may-2011 19:08:38 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:38	150.140.130.70	-	-	28080	PUT	/scrabbleGame/gameStatus
30-may-2011 19:08:39 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:39	79.166.240.8	-	-	28080	GET	/scrabbleGame/gameStatus
30-may-2011 19:08:41 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:41	79.166.240.8	-	-	28080	GET	/scrabbleGame/gameStatus
30-may-2011 19:08:43 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:43	150.140.130.70	-	-	28080	GET	/scrabbleGame/gameStatus
30-may-2011 19:08:43 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:43	79.166.240.8	-	-	28080	GET	/scrabbleGame/gameStatus
30-may-2011 19:08:46 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:46	79.166.240.8	-	-	28080	GET	/scrabbleGame/gameStatus
30-may-2011 19:08:47 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:47	79.166.240.8	-	-	28080	PUT	/scrabbleGame/gameStatus
30-may-2011 19:08:48 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:48	150.140.130.70	-	-	28080	GET	/scrabbleGame/gameStatus
30-may-2011 19:08:49 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:49	79.166.240.8	-	-	28080	GET	/scrabbleGame/gameStatus
30-may-2011 19:08:51 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:51	150.140.130.70	-	-	28080	DELETE	/scrabbleGame/gameStatus
30-may-2011 19:08:51 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:51	79.166.240.8	-	-	28080	GET	/scrabbleGame/gameStatus
30-may-2011 19:08:53 org.restlet.engine.log.LogFilter afterHandle
INFO: 2011-05-30	19:08:53	150.140.130.70	-	-	28080	GET	/scrabbleGame/gameStatus
```

Figure 15. Web Services communications

3.4 How to play

The aim of the ScrabbleCity application is to engage groups of people to solve the hints of scrabble game along a location or scenario, the location can be from a city to a museum, interacting through mobile devices. The people thus need to be divided in different groups. Each group receives one name and one color that identify them.

The Scrabble game has two main screens that correspond to the two activities of the application, Scrabble and TopicActivity. Scrabble activity shows nine images with the topics about which the game will develop. In the second one shows all information about the chosen topic, as well as the different hints and exhibits to answer and the button to link them.

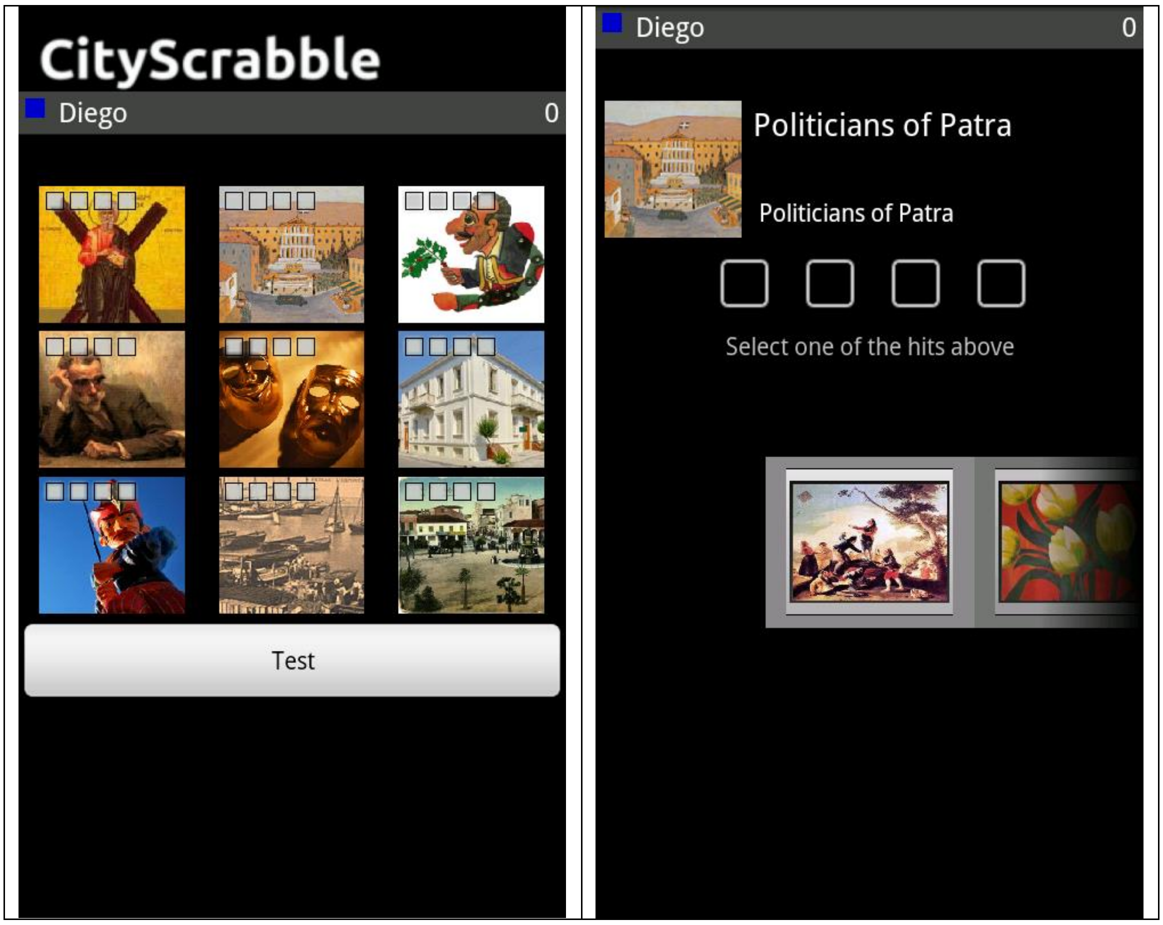


Figure 16. Scrabble and Topic screen at the beginning of the game

When the game starts, the player sees the topic selection screen. Each topic icon has four flags signaling availability of its hints. Here the game has just begun and the flags are empty; that means the topics are available for scoring points. Later this flags will show the hints which have been answered and the team which has answered them. When the player knows about which topic he wants to play, he has to push over the chosen image topic and the topic screen will be displayed.

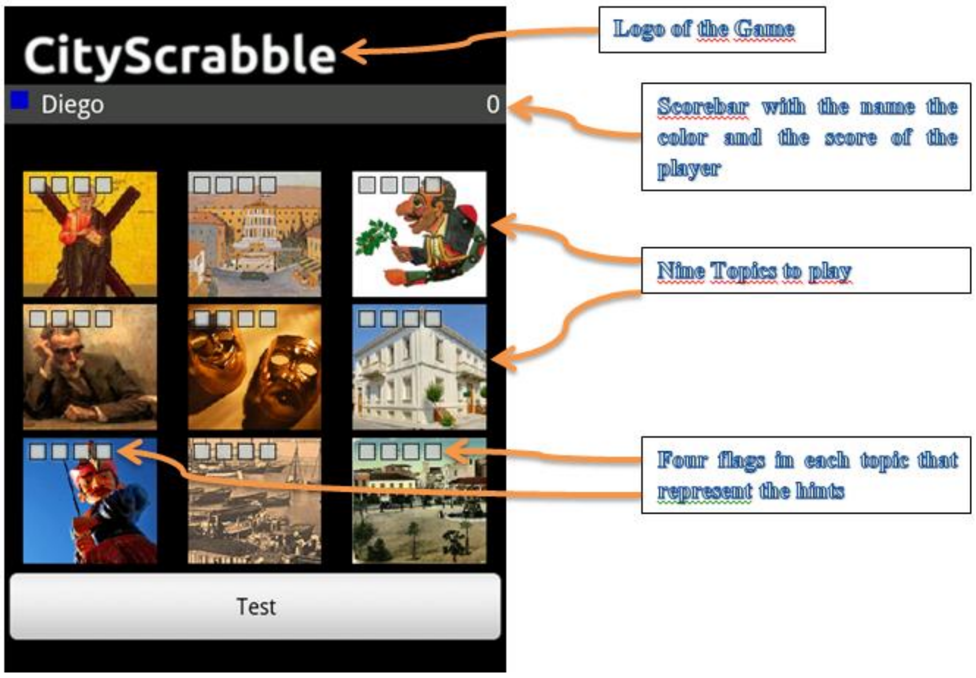

CityScrabble
Diego
0
Test
Logo of the Game
Scorebar with the name the color and the score of the player
Nine Topics to play
Four flags in each topic that represent the hints

Figure 17. Widgets of Scrabble screen

The topic screen displays the following elements.

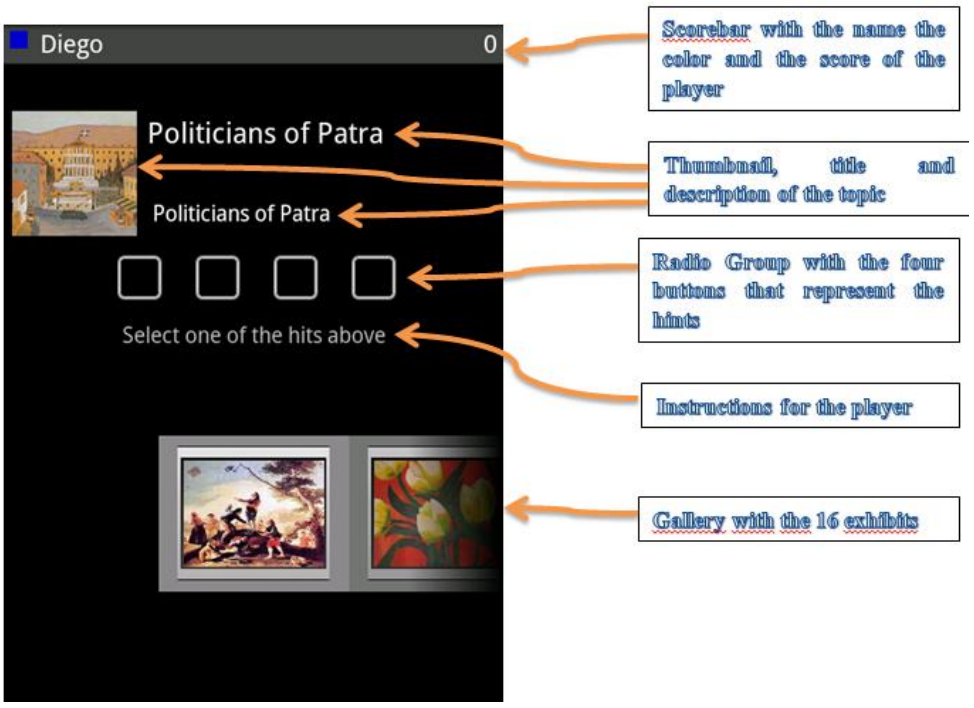

Diego
0
Politicians of Patra
Politicians of Patra
Select one of the hits above
Scorebar with the name the color and the score of the player
Thumbnail, title and description of the topic
Radio Group with the four buttons that represent the hints
Instructions for the player
Gallery with the 16 exhibits

Figure 18. Widgets of Topic screen

First it shows the image topic close to the title and to the description that is a short text describing the concept of the topic and beneath it displays a group of four buttons that represents the four hints. The description is used to put the players in context and help them to find the exhibits around the city.

The topic screen displays a group of four different hints that the player can answer. When the player presses over one of these buttons the text of the hints appears below.

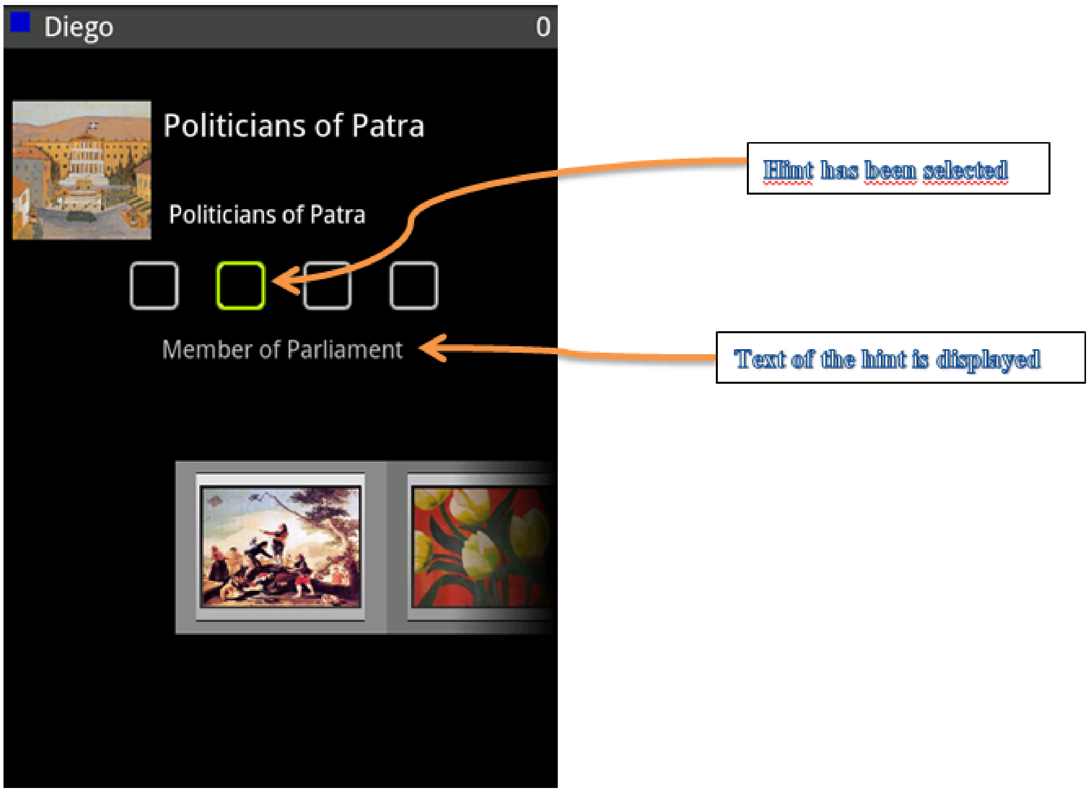
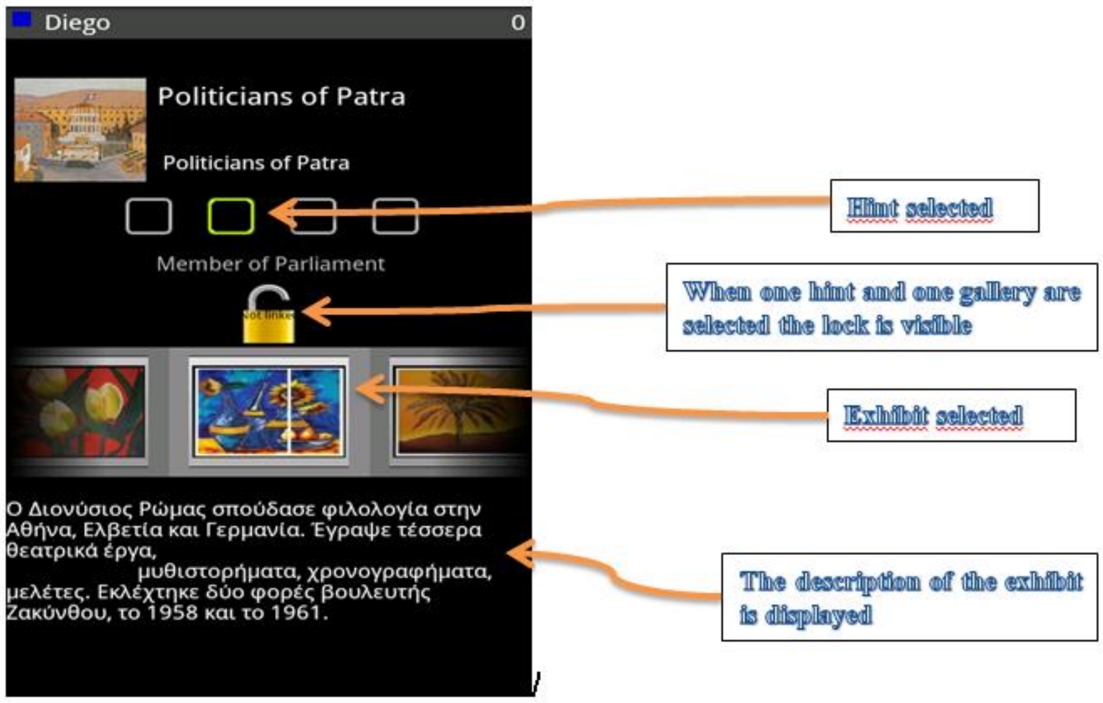


Figure 19. Select a hint in the Topic screen

In the bottom part of the screen will appear the gallery. The player now has to look around in the scenario of the game and try to locate exhibits that might be relevant to the chosen topic and the available hints. The way to do this is to search in the scenario of the game for exhibits and scan their QR code with the mobile. The scanned exhibit is then included in one of available slot of the gallery. If the player pushes on the thumbnail picture of the exhibit, he selects it and a short description will appear on the screen.

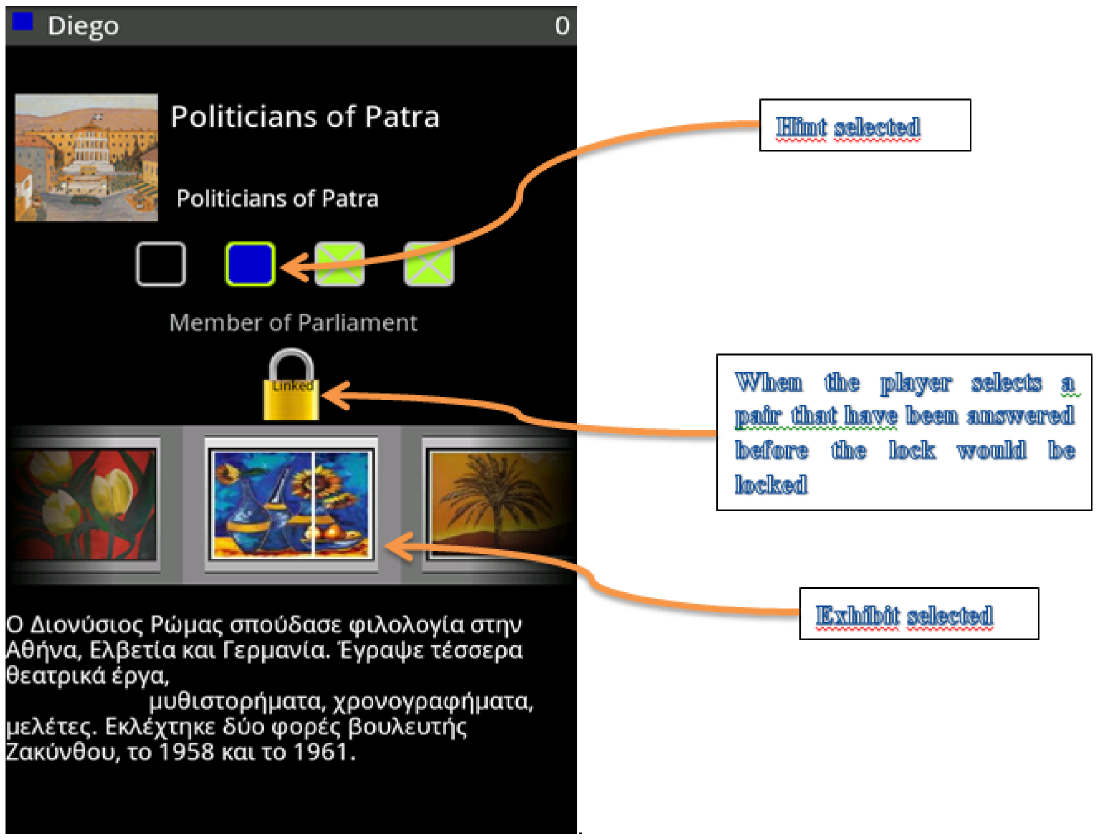

Diego
0
Politicians of Patra
Politicians of Patra
Member of Parliament
Not linked
Hint selected
When one hint and one gallery are selected the lock is visible
Exhibit selected
The description of the exhibit is displayed

Ο Διονύσιος Ρώμας σπούδασε φιλολογία στην Αθήνα, Ελβετία και Γερμανία. Έγραψε τέσσερα θεατρικά έργα, μυθιστορήματα, χρονογραφήματα, μελέτες. Εκλέχτηκε δύο φορές βουλευτής Ζακύνθου, το 1958 και το 1961.

Figure 20. Select a hint and an exhibit in Topic screen

Between the list of hints and the gallery of exhibits appears the link button. This button is used to link the hint with the exhibit when the player thinks that the join is correct. The player can also strategically join a pair hint-exhibit, doing that the pair is not available for other players, but with this the player will not get score. There will be a maximum number of incorrect answers that the player can have linked at the same time.

The player will be able to unlink his pairs. To do this, he has to select the hint, the exhibit and press on the link button. If the pair that the player has unlinked was correct, the value of this hint will be subtracted of the score. But if the pair unlinked was incorrect, nothing is subtracted.



When the player selects a pair that have been answered before the lock would be locked

Ο Διονύσιος Ρώμας σπούδασε φιλολογία στην Αθήνα, Ελβετία και Γερμανία. Έγραψε τέσσερα θεατρικά έργα,
μυθιστορήματα, χρονογραφήματα, μελέτες. Εκλέχτηκε δύο φορές βουλευτής Ζακύνθου, το 1958 και το 1961.

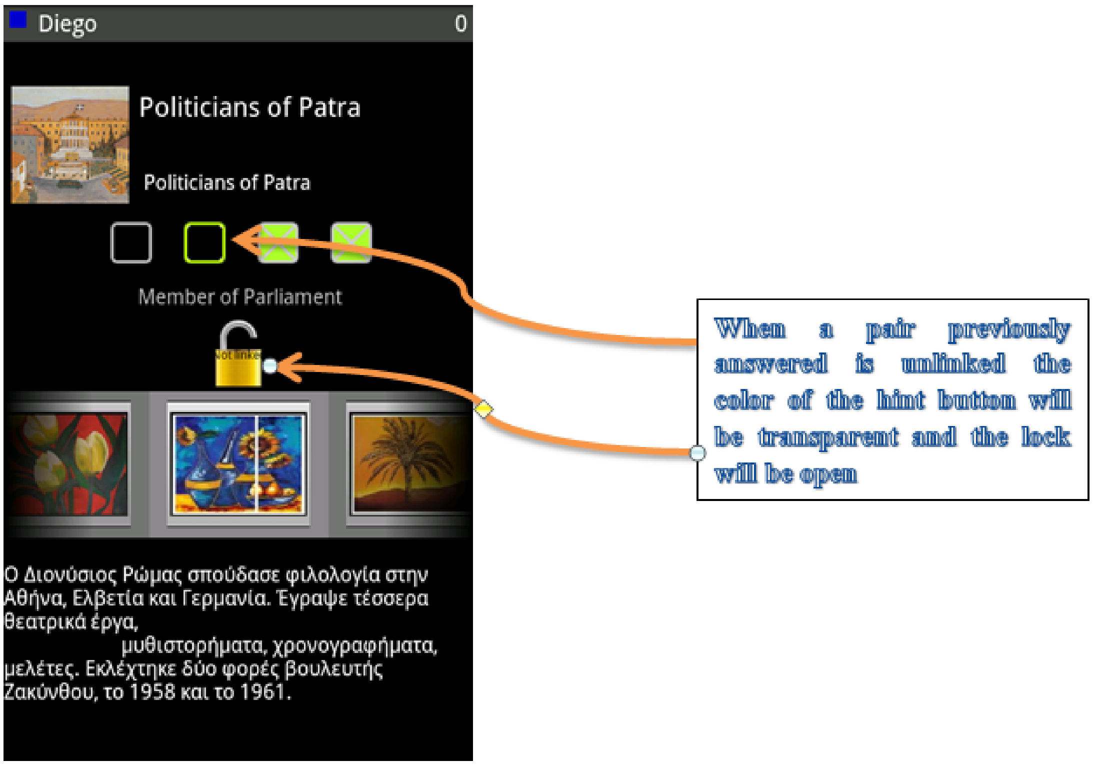
Figure 21. Topic screen with hints answered

The link button will have four states: invisible, disabled, open and locked. In the beginning the button is always in invisible state.
Below it shows all the states:

1. There aren't selected one hint and one exhibit, in the case the state of the button is invisible
2. One hint and one exhibit have been selected. The button to link is set visible and in open state.
3. The player locks the locked, and four options can occur:
    - The join is correct and the pair hasn`t been answered before. We lock the lock for the player, and the lock must be disabled for the other players. The flag is drawn with the color of the player, and finally the score is added.
    - The join is correct but the pair has been answered before for the player. The lock will be open, the score will be subtracted and the color of the flag will be transparent. Also the pair will be available for the other player. Now the state is open.
    - The join is incorrect and the hint hasn`t been answered before for the player. The lock must be closed and disabled for the other players. Also the flag will be drawn.
    - The join is incorrect and the hint has been answered before for the player. The lock must be open and the color of the flag will be

transparent. Also the pair will be available for the other player. Now the state is open.

4. When one hint has been answered for other player, automatically the flag of the hint will be drawn with the color of the player and the link button will be disabled for the player.



Ο Διονύσιος Ρώμας σπούδασε φιλολογία στην Αθήνα, Ελβετία και Γερμανία. Έγραψε τέσσερα θεατρικά έργα,
μυθιστορήματα, χρονογραφήματα, μελέτες. Εκλέχτηκε δύο φορές βουλευτής Ζακύνθου, το 1958 και το 1961.

When a pair previously answered is unlinked the color of the hint button will be transparent and the lock will be open

Figure 22. Topic screen unlinking hint

The player can continue looking for exhibit matching at the current topic or at any time return to the topic selection screen.

When the player returns to the topic screen, it is visible that the other players have been also busy securing hints. This gives an overview of the other player´s activity, but doesn´t provide any information about their score. They might as well have created meaningless associations earning no points at all.

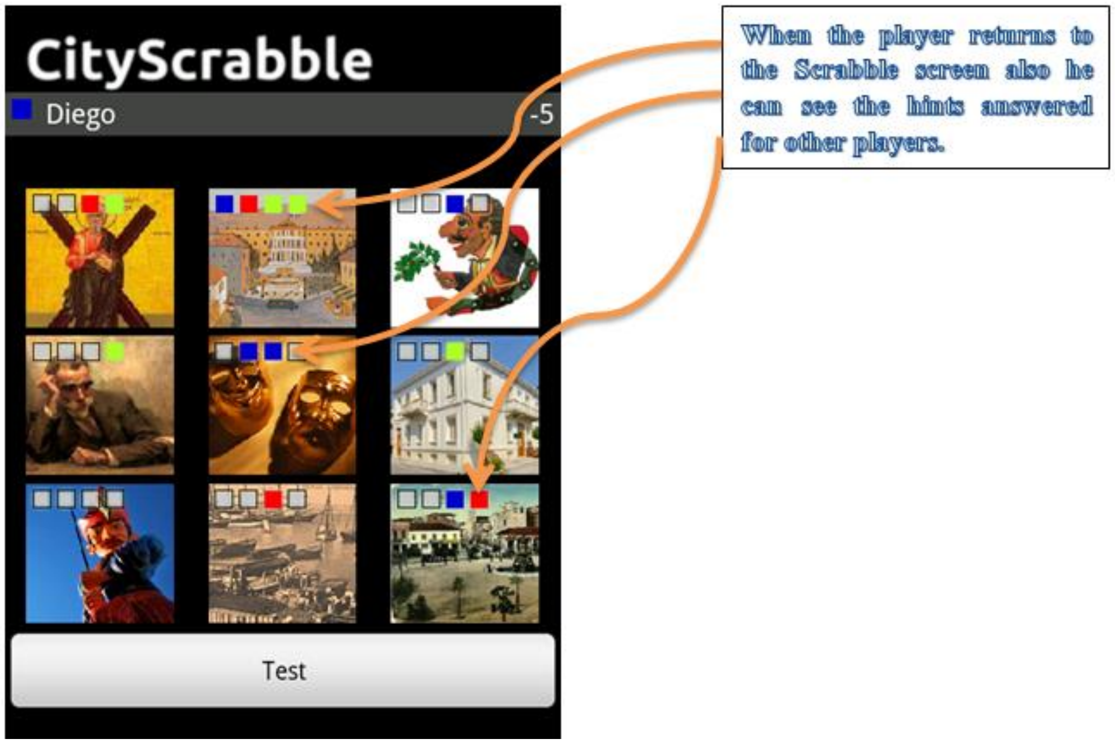

CityScrabble
Diego
-5
Test
When the player returns to the Scrabble screen also he can see the hints answered for other players.

Figure 23. Scrabble screen playing

The flags of the topic selection screen are updated instantly when an association has been made. So, one can watch the screen and get an idea about the level of activity in each topic as the time elapses.

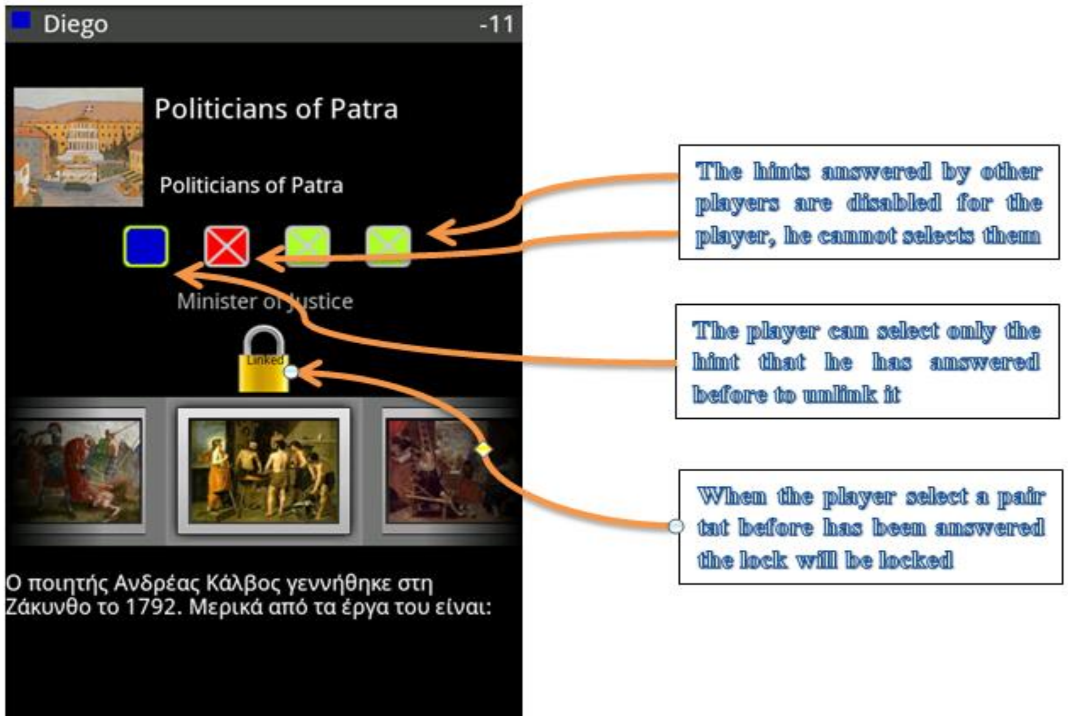

Diego
-11
Politicians of Patra
Politicians of Patra
Minister of Justice
Linked
Ο ποιητής Ανδρέας Κάλβος γεννήθηκε στη Ζάκυνθο το 1792. Μερικά από τα έργα του είναι:
The hints answered by other players are disabled for the player, he cannot selects them
The player can select only the hint that he has answered before to unlink it
When the player select a pair tat before has been answered the lock will be locked

Figure 24. Topic screen playing

Finally the problem that can occur if two players are answering the same hint at more or less the same time has been resolved.

To do this, all the players are allowed to answer the hint, but only the first player that store in the server the pair is the owner of the pair. Therefore at the beginning when a player answers a hint this is drawn with white color, and only when the data is persistent in the server the hint will be drawn with the color of the player.

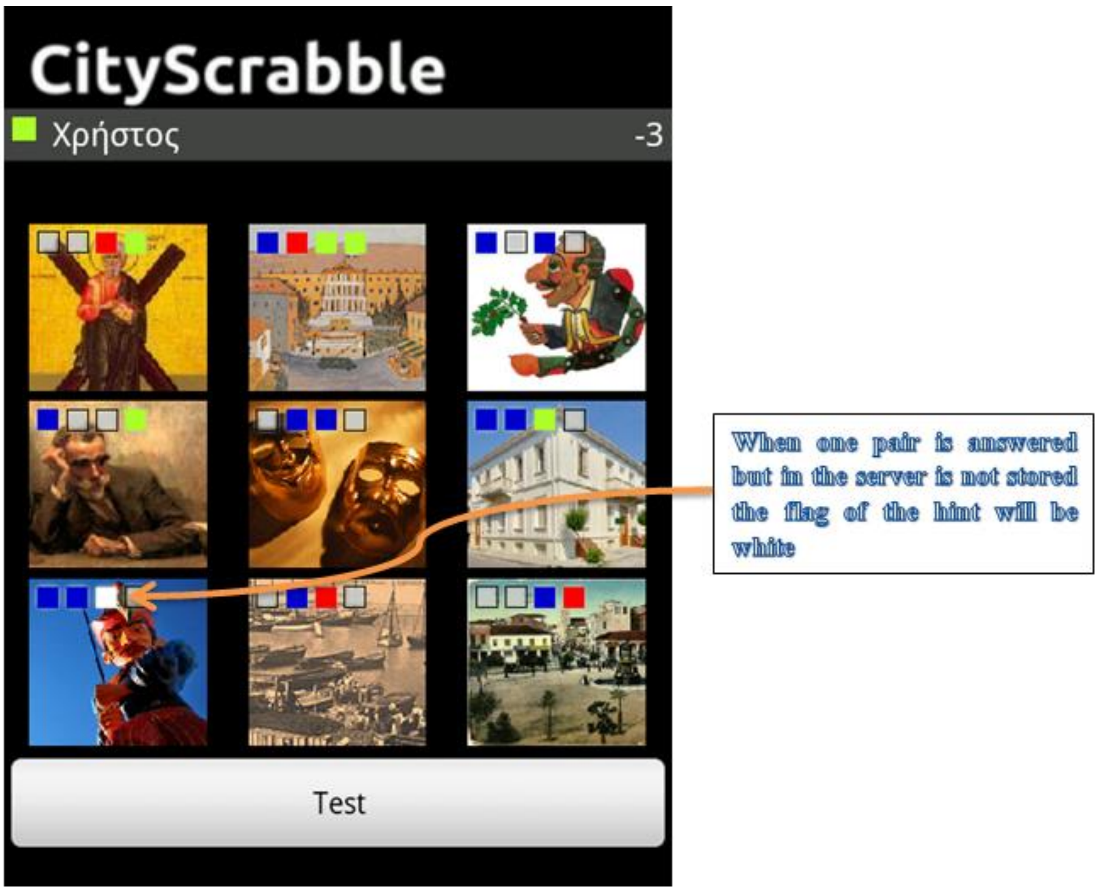

CityScrabble
Χρήστος
-3
Test
When one pair is answered but in the server is not stored the flag of the hint will be white

Figure 25. Waiting answer from the server

Chapter 4

Development and Implementation

4.1 Introduction

This project has been focused basically in the development and implementation od the application. Most of the time spent in the project has been used for this task. Due to this, it has been decided to present a detailed report of the features that have been designed and implemented in the application

4.2 Features implemented

All of the next features have been implemented in the application. Below they are explained

4.2.1 Parsers

4.2.1.1 Introduction

We want to build Android applications that can work with XML from the Internet. Android applications are written in the Java programming language, so experience that Java technology has about to work with XML files can be used.

It is not that working with XML is so interesting; it is working with the things that it enables. XML is commonly used as a data format on the Internet. If you want to access data from the Internet, chances are that the data will be in the form of XML. If you want to send data to a Web service, you might also need to send XML. In short, if the Android application will leverage the Internet, then it will probably need to work with XML. Luckily, it has a lot of options available for working with XML on Android.

4.2.1.2 XML Parsers

One of the greatest strengths of the Android platform is that it leverages the Java programming language. The Android SDK does not quite offer everything available to your standard Java Runtime Environment (JRE,) but it supports a very significant fraction of it. The Java platform has supported many different ways to work with XML for quite some time, and most of Java's XML-related APIs are fully supported on Android. For example, Java's Simple API for XML (SAX) and the Document Object Model (DOM) are both available on Android. Both of these APIs have been part of Java technology for many years.

4.2.1.3 SAX

In a Java environment, it can often use the SAX API when it wants a fast parser and wants to minimize the memory footprint of your application. That makes it very well suited for a mobile device running Android. It can use the SAX API as-is from the Java environment, with no special modifications needed to run on Android

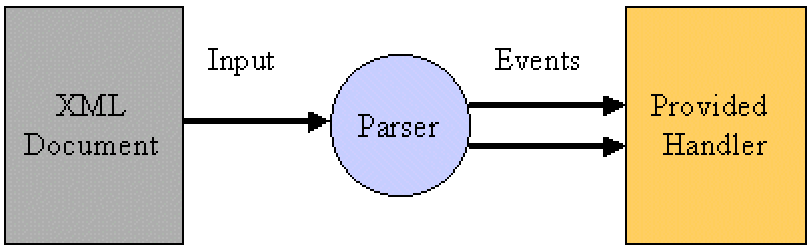


Figure 26. Sax parser

4.2.1.4 DOM

The DOM parser reads the entire XML document into memory and then allows to use the DOM APIs to transverse the XML tree, retrieving the data that you want.

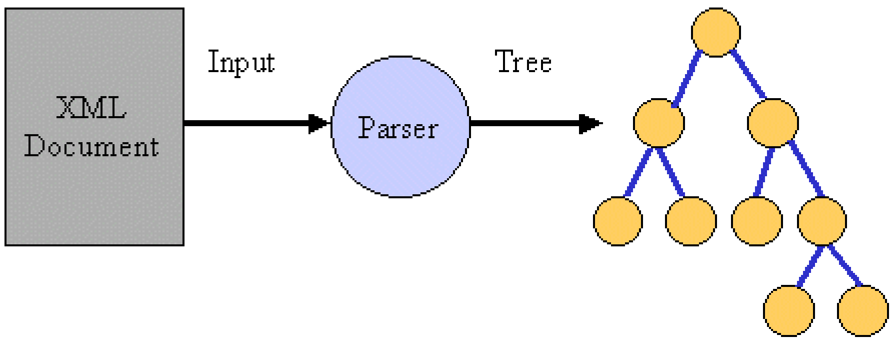


Figure 27. DOM parser

This is very straightforward code, and, in some ways, simpler than the SAX-based implementations. However, DOM generally consumes more memory as everything is read into memory first. This can be a problem on mobile devices that run Android, but it can be satisfactory in certain use cases where the size of the XML document will never be very large. As in the application it reads only one time the XML file, it goes to use this parser, because it is simpler than SAX and after read it at the first time never is read.

4.2.1.5 Using DOM

According to W3C, the Document Object Model (DOM) is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term "document" is used in the broad sense - increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model, with a few exceptions - in particular, the DOM interfaces for the XML internal and external subsets have not yet been specified.

In the DOM, documents have a logical structure which is very much like a tree. A graphical representation of the DOM of the Scrabble XML file is:

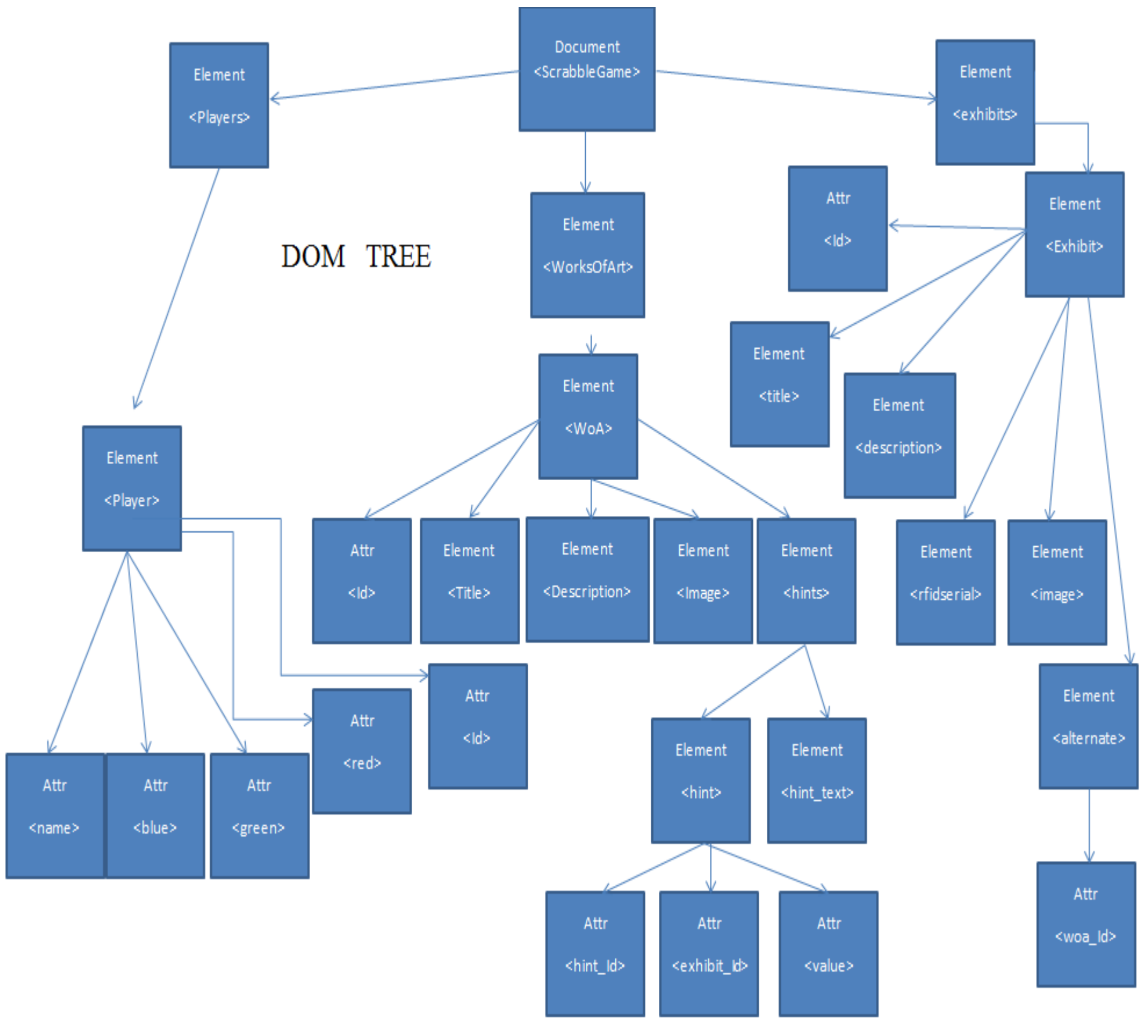


Figure 28. DOM tree of the project

The class ReadXML will take an XML file from a server and parse it into three lists of simple Java objects: players, exhibits and topics. Also inside each topic object, there is a list of hint objects. These objects can be used to back the data needs for displaying the information and the thumbnail of the topics, hints, exhibits, and for managing a lot of states.

In order to show how the DOM parser works, below be displayed as a Topic object is parsed. Player and exhibit objects are parsed in the same way therefore they are not included in this report.

4.2.1.6 XML file structure

In order to treat an *XML* file, it is necessary to know its structure. For brevity, below shows a small extract from the XML file, there are two roots WorkOfArt with the elements and attributes that are inside.

```
WoA id="36">
tle>Religion</title>
escription>Priests and Holy persons of Zakynthos</description>
nage>Papa Josifi Papamihali1.jpg</image>
ints>
int hint_id="68" exhibit_id="129" value="3">
int_text>Saints</hint_text>
</hint>
int hint_id="68" exhibit_id="61" value="3">
int_text>Saints</hint_text>
</hint>
int hint_id="69" exhibit_id="270" value="3">
int_text>Priests</hint_text>
</hint>
int hint_id="70" exhibit_id="82" value="3">
int_text>Archbishop</hint_text>
</hint>
int hint_id="71" exhibit_id="130" value="3">
int_text>Sacred Mysteries</hint_text>
</hint>
</hints>
</WoA>
VoA id="35">
tle>Politicians of Zante</title>
escription>Politicians of Zante</description>
nage>bouli.jpg</image>
ints>
int hint_id="64" exhibit_id="110" value="3">
int_text>Minister of Justice</hint_text>
</hint>
int hint_id="65" exhibit_id="126" value="3">
int_text>Member of Parliament</hint_text>
</hint>
int hint_id="66" exhibit_id="108" value="3">
int_text>Υπουργός Εξωτερικών</hint_text>
</hint>
int hint_id="67" exhibit_id="272" value="3">
int_text>Υπουργός Παιδείας</hint_text>
</hint>
</hints>
</WoA>
```

Listing 3. XML file with the data of the game

4.2.1.7 POJO and Data Structures

All of this information is stored in objects. These kinds of objects are called *Plain Old Java Object* POJO and represent the data structure of the players, topics and exhibits. In the next Listing a Topic POJO object is shown.

```
public class Topic implements Parcelable {
       private int topicId;
       private String topicTitle;
       private String topicDescription;
```

```
      private Bitmap topicThumbnail;
      private HintList topicHints;

      public Topic(int topicId, String topicTitle, String
topicDescription, Bitmap topicThumbnail, HintList topicHints) {
            this.topicId = topicId;
            this.topicTitle = topicTitle;
            this.topicDescription = topicDescription;
            this.topicThumbnail = topicThumbnail;
            this.topicHints = topicHints;
      }
      // getters and setters omitted for brevity
      …
}
```

Listing 4. POJO object

4.2.1.8 Read the data and store it

For brevity will explain only the parsing of a topic object, which is the most complex of the three. The other two objects are parsed in a similar way.

- To create a Java DOM XML parser is used the javax.xml.parsers.DocumentBuilderFactory class. The DocumentBuilder instance is the DOM parser. Using this DOM parser, XML files are parsed into DOM objects.

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

            try {
                  DocumentBuilder db = dbf.newDocumentBuilder();

            } catch (ParserConfigurationException pce) {
                  pce.printStackTrace();
            }
```

Listing 5. Creating DOM parser

- Get the document from the XML file, that before has been got from the server via URL request.

```
String URL_Test = "http://dl.dropbox.com/u/15673850/scrabble.xml";
      url = new URL(URL_Test);
      Document dom = db.parse(new InputSource(url.openStream()));
```

Listing 6. Opening the URL of the XML file

- The DOM Document object represents an XML document. When you parse an XML file using a Java DOM parser, you get back a Document object. The diagram below presents a visual representation of an extract all the code to get information out of a Document:

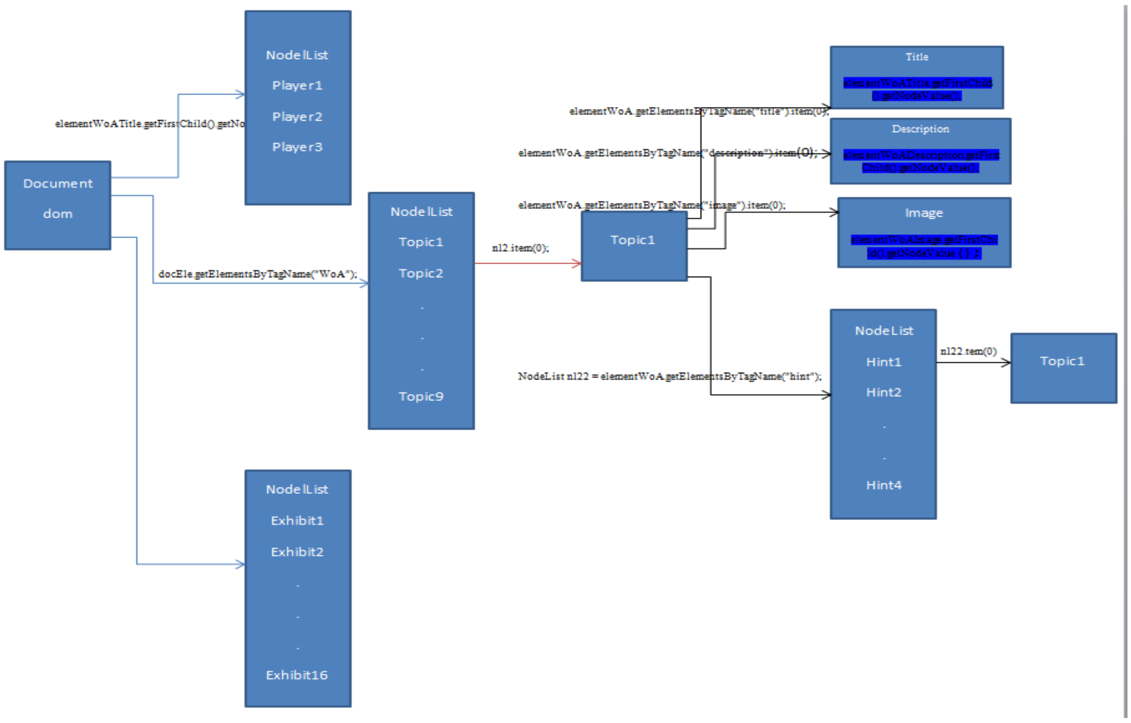

NodelList
Player1
Player2
Player3
elementWoATitle.getFirstChild().getNo
Document
dom
docEle.getElementsByTagName("WoA");
NodelList
Topic1
Topic2
.
.
.
Topic9
nl2.item(0);
Topic1
elementWoA.getElementsByTagName("title").item(0);
elementWoA.getElementsByTagName("description").item(0);
elementWoA.getElementsByTagName("image").item(0);
Title
elementWoATitle.getFirstChild().getNodeValue();
Description
elementWoADescription.getFirstChild().getNodeValue();
Image
elementWoAImage.getFirstChild().getNodeValue ( ) ;
NodeList nl22 = elementWoA.getElementsByTagName("hint");
NodeList
Hint1
Hint2
.
.
Hint4
nl22.tem(0)
Topic1
NodelList
Exhibit1
Exhibit2
.
.
.
Exhibit16

Figure 29. How work DOM parser

- In order to extract information from the Document object. First it gets the root Element of the method parseTopic(). Then it asks the document object for all the Element objects that have the tag name "WoA". This should return all the Element objects that are WoAs; all the Element objects with this tag name are returned in a NodeList object. We can use the getLength() method on this NodeList to determine how many WoA elements are in the NodeList. Here is some code to do this:

```
    private TopicList parseTopics() {
private TopicList parseTopics() {
// get the root elememt
Element docEle = dom.getDocumentElement();
// get a nodelist of <WorksOfArts> elements
NodeList nl2 = docEle.getElementsByTagName("WoA");
```

Listing 7. Getting the element and the nodelist

- Now that the application has the NodeList object containing all the WoA Elements (which are also Nodes), it can iterate it to extract information from each WoA Element (Node). The method item (int index) in NodeList returns a Node object.

```
        if (nl2 != null && nl2.getLength() > 0) {
             for (int i = 0; i < nl2.getLength(); i++) {
                                 Node topicNode = nl2.item(i);
```

Listing 8. Nodes

- Now it has the reference of every element WoA each time runs through the loop "for", which it can use to find out the Title, Description, Image, Id and the list if

Hints information of this WoA element. Since the elementWoA is an Element, it can use getElementsByTagName(String) again to get the Title, Description and Image elements in it. Here is the code to do get the Title, Description and Image elements of the elementWoA:

```
Element elementWoA = (Element) nl2.item(i);
Element elementWoATitle =
(Element)elementWoA.getElementsByTagName("title").item(0);
Element elementWoADescription =
(Element) elementWoA.getElementsByTagName("description").item(0);
Element elementWoAImage =
(Element) elementWoA.getElementsByTagName("image").item(0);
```

Listing 9. Getting the Title, Description and Image elements of the elementWoA

The next step is get the value of the elements and the id attribute:

```
int topicId = Integer.parseInt(elementWoA.getAttribute("id"));
String topicTitle = elementWoATitle.getFirstChild().getNodeValue();
String topicDescription =
elementWoADescription.getFirstChild().getNodeValue();
String topicImageLink =
elementWoAImage.getFirstChild().getNodeValue();
```

Listing 10. Getting the value of the elements and the id attribute

Finally, it repeats the procedure to get the hints, i.e., a NodeList is created in the Element elementWoA looking for all the Element objects That have the tag name "hint". It enters again in a loop through all the hints and it reads the values in the same manner as above.

```
HintList hintList = new HintList();
NodeList nl22 = elementWoA.getElementsByTagName("hint");
int numbernodos = nl22.getLength();
if (nl22 != null && nl22.getLength() > 0) {
      for (int k = 0; k < numbernodos; k++) {
            Element elementHint = (Element) nl22.item(k);
            Element elementWoAHint = (Element)
elementHint.getElementsByTagName("hint").item(0);
            Element elementWoAHint_text = (Element)
elementHint.getElementsByTagName("hint_text").item(0);
            String hint_text =
elementWoAHint_text.getFirstChild().getNodeValue();
            int hint_id =
Integer.parseInt(elementWoAHint.getAttribute("hint_id"));
            int exhibit_id =
Integer.parseInt(elementWoAHint.getAttribute("exhibit_id"));
            int value =
Integer.parseInt(elementWoAHint.getAttribute("value"));
            listelementhint_hint_id[k] = hint_id;
            listelementhint_exhibit_id[k] = exhibit_id;
            listelementhint_value[k] = value;
            listelementhint_text.add(hint_text);
      Hint newHint = new Hint(hint_id, exhibit_id, value, hint_text);
            hintList.add(newHint);
```

Listing 11. Getting hints

- Finally all the objects are stored

```
Topic newTopic = new Topic(topicId, topicTitle, topicDescription,
topicImage, hintList);
listScrabbeTopics.add(newTopic);
```

Listing 12. Storing the objects

4.2.2 Communications

In this section it goes to describe some of the communications that are necessary to implement the application. It will explain the different intends and how to check the Internet connection and how to download and XML file from Internet.
The Web service communication will be described in the Restlet chapter.

4.2.2.1 Intents

Three of the core components of an application — activities, services, and broadcast receivers — are activated through intents. Intent messaging is a facility for late run-time binding between components in the same or different applications. The intent itself, an Intent object, is a passive data structure holding an abstract description of an operation to be performed — or, often in the case of broadcasts, a description of something that has happened and is being announced. There are separate mechanisms for delivering intents to each type of component:

- An Intent object is passed to Context.startActivity() or Activity.startActivityForResult() to launch an activity or get an existing activity to do something new. (It can also be passed to Activity.setResult() to return information to the activity that called startActivityForResult().)
- An Intent object is passed to Context.startService() to initiate a service or deliver new instructions to an ongoing service. Similarly, an intent can be passed to Context.bindService() to establish a connection between the calling component and a target service. It can optionally initiate the service if it's not already running.
- Intent objects passed to any of the broadcast methods (such as Context.sendBroadcast(), Context.sendOrderedBroadcast(), or Context.sendStickyBroadcast()) are delivered to all interested broadcast receivers. Many kinds of broadcasts originate in system code.

In each case, the Android system finds the appropriate activity, service, or set of broadcast receivers to respond to the intent, instantiating them if necessary. There is no overlap within these messaging systems: Broadcast intents are delivered only to broadcast receivers, never to activities or services. An intent passed to startActivity() is delivered only to an activity, never to a service or broadcast receiver, and so on.

4.2.2.2 Intent to launch an activity

When the application starts it shows us the first screen that contains the nine topics about the player goes to play.

One of the tasks that the application has to make is that when the player click-on one of the topics, it changes to a new activity with its corresponding new screen (topicactivity).

Using this intent, the activity(topicActivity) is loaded, showing to the player the new screen

```
Intent intent = new Intent();
                        intent.putExtra("topicId", topicId);
                        intent.setClass(context, targetClass);
                        context.startActivity(intent);
```

Listing 13. Launch an intent

The first line creates the intent, and then the id of the topic is associated to the intent to be sent, with intent.putExtra("topicId", topicId). TargetClass indicates the class that it wants to initiate. TargetClass is an abstract class that has like parameter the topicActivity class. This is the class that the application really wants to initiate. *Context* indicates the current context, to know in which package the application has to look for this class.

Finally the new activity starts calling startActicvity() method.
After this in the new activity the id of the topic is got by a Bundle object in the onCreate of the activity.

```
// get the topic transmitted from the main screen
            Bundle extras = getIntent().getExtras();
            if (extras != null) {
                  currentTopicId = extras.getInt("topicId");
                        }
```

Listing 14. Get the data from a bundle

4.2.2.3 Intent to initiate a service

The application needs a service that asks periodically to the server which questions have been answered and the player that has responded them. To start this service an intent is needed.

```
serviceIntent = new Intent(this, RestClientService.class);
            serviceIntent.putExtra("myPlayerId", app.MyPlayerId);
            startService(serviceIntent);
```

Listing 15. Launch a service

With this code the service starts running in background. The service is a Reslet client that consumes a Web Service. There are two main methods. First one is a synchronous method that gets from the server a list with the Id of the hints and exhibits answered and also the player that has answered each pair. Second one is an asynchronous method that stores the information in the server if a player answers correctly.

Basically this code creates a new intent, adds it the current context and the player id, and starts the service written in the RestClientService class.

4.2.2.4 Intent to deliver to all interested broadcast receivers

The last way of communication that the application has by intent is through broadcast communications. This kind of intents broadcast messages between components with the sendBroadcast() method, which is used when a hint-exhibit pair has been answered correctly. Its function is to communicate to the classes that need to know this information to update their status.

To do this, it follows the next steps:

- Send the broadcast message when a hint has been answered correctly from the ScrabbleApplication class

```
Intent intentNewAssociation =new Intent(NEW_ASSOCIATION_INTENT);
intentNewAssociation.putExtra("hintId",hintId);
intentNewAssociation.putExtra("exhibitId",exhibitId);
intentNewAssociation.putExtra("playerId",MyPlayerId);

sendBroadcast(intentNewAssociation);
```

Listing 16. Send the broadcast message

- Set the broadcast Receiver in the RestClientService class to listen the broadcast intents. The receiver must be in the onReceive of the class necessary.

```
//The broadcast Receiver and the action that it has to do
public class NewAssociationFromPlayerReceiver extends
BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
int hintId = intent.getIntExtra("hintId", -1);
int exhibitId = intent.getIntExtra("exhibitId", -1);
int newOwnerPlayerId = intent.getIntExtra("playerId", -1);
```

Listing 17. Broadcast receiver

- The broadcast receiver must be registered either code or within the app manifest. It use Intent filters to specify which intents it is listening for. In this application the register is via code

```
//Register the broadcast receiver
   IntentFilter newAssociationIntentFilter = new
   IntentFilter(ScrabbleApplication.NEW_ASSOCIATION_INTENT);
   try {
        unregisterReceiver(newAssociationFromPlayerReceiver);
        } catch (IllegalArgumentException e) {
        }
   newAssociationFromPlayerReceiver =
   new=NewAssociationFromPlayerReceiver();
   registerReceiver(newAssociationFromPlayerReceiver,
   newAssociationIntentFilter);
```

Listing 18. Register the Broadcast

4.2.2.5 Checking Internet connection

One of the first tasks that the application does is check whether there are Internet connections available. This is important for several reasons. The first is that the game resources are hosted on a server, both the XML file that contains the game data to be parsed analyzed, and the images used in the game must be downloaded from a server, and for that you need have an Internet connection. The second reason is that the application has to connect to the Restlet server that has the game state, ie, exhibit-hint pairs and the player that has answered them, and that should be known in real time to inform players.

The incredible growth of Internet services and mobile devices has made access to the Internet an increasingly common feature on mobile phones.

The speed, reliability and cost of Internet connection depends on network technology being used (Wi-Fi, GPRS, 3G), enabling applications to handle these connections to ensure they run efficiently.

Android provides access to the network status through the "broadcasting Intents" to notify the application of changes in network connectivity and providing control over network settings and connections.
Android networking is handled primarily through ConnectivityManager API, a service that allows you to monitor connection status, configuring the preferred network connection, and manage connectivity.

The following steps are done to check the Internet connection:

- Provide Internet permissions in the application AndroidManifest.xml (In addition to this method they are necessary for other tasks of the application.) To do this the following instructions are introduced:

```
<uses-permission
android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE" />
```

Listing 19. Internet permission in Android Manifest

- The ConnectivityManager represents the connectivity service network. It is used to monitor the status of network connections and switch settings. In order to access the ConnectivityManager, getSystemService method is called, through Context.CONNECTIVITY_SERVICE parameter that represents the name of the service, as shown below:

```
ConnectivityManager connec = (ConnectivityManager)
ctx.getSystemService(Context.CONNECTIVITY_SERVICE);
```

Listing 20. ConnectivityManager

- Get all the available networks in the device (Wi-Fi, gprs...):

```
NetworkInfo[] networks = connec.getAllNetworkInfo();
```

Listing 21. Networks available

- Finally check the connection:

```
for (int i = 0; i < 2; i++) {
                  if (networks[i].getState() ==
NetworkInfo.State.CONNECTED) {
                        AvaliableConnection = true;
                  }
            }
```

Listing 22. Checking the connection

4.2.2.6 Connect to download a XML file

The necessary data to play aren´t in the mobile device but they are in a server in Internet. Therefore, when the game starts the data have to be downloaded.

The data are in a server like XML file, which through an URL connection is downloaded to the device. Then it has to transform the XML file to a format that can be used by the application classes. To do this, the XML file is parsed with a DOM parser, that reads all XML file and stores the data in objects, that after they are associated in lists of objects.

After these steps it provides the necessary data to run the application

In order to connect to an URL address to download the XML file and sending it to the parser, it makes the following:

```
String URL_Test = "http://dl.dropbox.com/u/15673850/scrabble.xml";
url = new URL(URL_Test);
dom = db.parse(new InputSource(url.openStream()));
```

Listing 23. Open an URL

Remember that it needs to give permissions to access the Internet at the android manifest as it is explained in the previous section.

4.2.3 Application Class

One of the most important classes of the application is the ScrabbleApplication class. Extending the Application class in the implementation enables to do three things:

- Maintain application state
- Transfer objects between application components
- Manage and maintain resources used by several application components

When Application implementation is registered in the manifest, it will be instantiated when application process is created. As a result the Application implementation is by nature a singleton and should be implemented as such to provide access to its methods and member variables.

Due to this class, the application status is centralized and its variables are not duplicated. It also, obtains the classes that has to access to all application resources, without the need to send arguments between some classes and other

When the XML is parsed to obtain the application resources, these are stored in lists of objects within application class, in order to be available from any class. Therefore some lists are stored containing the XML information, like a list of player, a list of topics and a list of exhibits to manage the application; in addition another necessary variables are stored. Below there is a listing with the resources stored:

```
private static ScrabbleApplication singleton;
public static final int MyPlayerId = 84;
public Player myPlayer;
static public int Score = 0;
static public ArrayList<String> topicHints = new ArrayList<String>();
static public int checkedItems = 0;
static public int checkedItemsGallery = 1;
static public int[] DisableArray = new int[30];
static public boolean configuartionNotRead = true;
PlayerList playerList;
public TopicList topicsList;
ExhibitList exhibitList;
static public HintList hintList;
```

Listing 24. Persistent data in Scrabble application

To create this class it follows the indications of[6].
First The ScrabbleApplication class extends Application class and implementing it as a singleton:

```
public class ScrabbleApplication extends Application {

      private static ScrabbleApplication singleton;
      public static ScrabbleApplication getInstance() {
            return singleton;
      }
     @Override
      public final void onCreate() {
            super.onCreate();
            initialize();
            singleton = this;
}
```

Listing 25. Singleton

OnCreate is called when the application is created. This method is overrided to initialize the application singleton and create and initialize any application state variables or shared resources. To do this, initialize method is called within onCreate.
In the initialize method, a broadcast receiver is registered to listen messages from activities. It use Intent filters to specify which intents it is listening for.
Once created, it must register the new Application class in the Android Manifest:

```
       <application
       android:icon="@drawable/icon"
       android:label="@string/app_name"
       android:name="ScrabbleApplication"
       android:debuggable="true">
```

Listing 26. Registered ScrabbleApplication in Android Manifest

Finally the application class checks the link between a hint and an exhibit when the player uses the toggle button to link them. If the linked is correct the application class will send a broadcast to inform it to other classes.

```
public void associateHintToExhibit(int currentTopicId, int exhibitId,
int hintId) {
            // TODO Auto-generated method stub
            Topic TmpTopic = null;
            try {
      TmpTopic = topicsList.getTopicByTopicId(currentTopicId);
            } catch (TopicNotFoundException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      Hint hint = TmpTopic.getTopicHints().getHintByHintId(hintId);
            int hintValue = hint.getHintValue();
           Score = (Score) + hintValue;
         // Notify the change to the application class: pair disabled
           hint.setOwnerPlayerId(MyPlayerId,
playerList.getPlayerByPlayerId(MyPlayerId).getColor());
           sendAssociationToServer(hintId, exhibitId);
      }
private void sendAssociationToServer(int hintId, int exhibitId) {
Intent intentNewAssociation = new Intent(NEW_ASSOCIATION_INTENT);
            intentNewAssociation.putExtra("hintId", hintId);
            intentNewAssociation.putExtra("exhibitId", exhibitId);
            intentNewAssociation.putExtra("playerId", MyPlayerId);
            sendBroadcast(intentNewAssociation);}
```

Listng 27. Checking linked pairs

4.2.4 Hints

One of the key points in the application design was the way of displaying the player a list of hints that are in each topic. Remember that hints should join the exhibit to win the game.

It ought to show to the player the information in a simple, intuitive and clear way, in order to provide that if the player just has a quick look at the screen, he can obtains all the necessary information.

In the first design was implemented a ListView that on each row contained the text of the hint and a flag or box that showed the status of the hint via colors. As it has related before on the section 3, this design had various problems like the size of the control that took up a lot of space on the screen, or on the other hand, if the size of the ListView was limited with one scrollbar, there were hints and flags hidden on the screen.

To resolve these problems it is designed a new control for the UI that shows the hints. In this new control for the UI there will be two LinearLayout, in the first one there will be a RadioGroup customized with four RadioButton customized too, that represent the four hints. These buttons are customized showing the flags or boxes that inform about the owner of the hint, i.e., the button is drawn with the color of the player who answer the hint.

At the beginning of the game in the second LinearLayout there will be some instructions about the game, but when the player selects one button, The hint will appear in the TextView the hint.
In this way it will get save a lot of space, and also it will get show all the information during all the time.

In the following sections first be shown how to implement the control via ListView and then how to implement the control via HintView.

4.2.4.1 Listview

One of the widgets that there are on the TopicActivity screen is a ListView. This ListView consists in four hints which should be answered by the players. In each row there are a text which explains the hint and a box or flag that shows the color of the player who has answered it or it is empty if it hasn´t been answered. The box has to be drawn every time that a player responds correctly a hint, when this happens; the box has to be updated with the color of the player.

ListView is a control which can be used for creating list of scrollable items. The data to the ListView will be provided by ListAdapter.

An Adapter object acts as a bridge between an AdapterView and the underlying data for that view. The Adapter provides access to the data items. The Adapter is also responsible for making a View for each item in the data set.

A ListView works as follows:

- ListView asks adapter “give me a view” (getView) for each item of the list
- A new View is returned and displayed

To create a ListView it must necessarily include the tags in the XML layout of the screen:

```
<ListView
            android:id="@+id/LstOpciones"
            android:layout_below="@id/RelativeLayout2"
            android:padding="15dp"
            android:cacheColorHint="#00000000"
            android:layout_width="wrap_content"
            android:layout_height="150dip"
            android:textStyle="bold"
android:background="@android:drawable/screen_background_dark_transpare
nt" />
```

Listing 28. ListView layout

A ListView, by default, only allows the developer to add lines of text. Note that if the number of items is higher that the size of the screen assigned to the control; ListView by itself is the responsible to set a scrollbar, so it is not necessary to create a Scroller, because the control is automatic.

To allow select only one hint each time, we set the parameter of the ListView `setChoiceMode like` CHOICE_MODE_SINGLE

```
lstTopicHints.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
```

Listing 29. Setting single mode

Hint class has all the data necessary to show in the ListView, therefore it´s not necessary to create a new class with this data.

If it wants to add more texts or icons to each item, it is necessary create a new XML layout with the desired structure for each line in the ListView. Also it necessary creates an ArrayAdapter instead of generic ListAdapter.

For this application it is created a new XML layout that adds an icon to each item:

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:padding="1px">
      <scrabble.project.HintTile
            android:id="@+id/hinticon"
            android:layout_width="40px"
            android:layout_height="40px"
            android:layout_alignParentTop="true"
            android:layout_alignParentBottom="true"
            android:layout_marginRight="6dip" />
      <TextView
            android:id="@+id/hinttext"
            android:layout_width="fill_parent"
            android:layout_height="40px"
            android:cacheColorHint="#00000000"
            android:padding="10dp"
            android:textSize="12sp"
            android:layout_toRightOf="@id/hinticon"
            android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
            android:singleLine="true"
            android:ellipsize="marquee">
      </TextView>
</RelativeLayout>
```

Listing 30. Elements of a row

The next steps are:

- Create a customized ArrayAdapter for displaying the elements on the ListView.
- Implement a method that will draw in the icon the color depending whether the player answers correctly the hint.

After this it has to implement the functionality of the ListView. To do this it goes to implement a listener.

```
lstTopicHints.setOnItemClickListener(itemClickListener);
```

Listing 31. Setting listener

When a row of the LisView is pressed the listener is called. Then The background of the row changes, the status of the row changes to pressed, and the method chekLinkButton() is called to do some functions.

```
private OnItemClickListener itemClickListener = new
OnItemClickListener() {
            @Override
public void onItemClick(AdapterView<?> arg0, View arg1, int position,
long arg3) {
   arg1.setBackgroundResource(android.R.drawable.menuitem_background);
   arg1.setPressed(true);
   checkLinkButton(position);
            }
      };
```

Listing 32. Listview listener

Finally each time that the data of the hints change is necessary to notify to the adapter these changes. To do this the method notifyDataSetChanged() is called.

```
topicHintAdapter.notifyDataSetChanged();
```

Listing 32. Notify the changes in the ListView

4.2.4.2 TopicHintAdapter

According to exposed in the previous section, it wants to show in each row of the ListView more information than simple text. In this case it cannot be possible to work with a simple ListAdapter but it will need to implement a class which extends ArrayAdapter class.

Below it shows the TopicHintAdapter class which extends ArrayAdapter<Hint>:

```
public class TopicHintAdapter extends ArrayAdapter<Hint> {
        public TopicHintAdapter(Context context, int _resource,
HintList topicHints) {
            super(context, _resource, topicHints);
            resource = _resource;
            topicHintsLocal = topicHints;
            myDrawable1 =
context.getResources().getDrawable(R.drawable.cuadrado_transparente);
      }

      public View getView(int position, View convertView, ViewGroup
parent) {
            Hint hint = getItem(position);
            View row = convertView;
            if (row == null) {
                  row = new LinearLayout(getContext());
                  String inflater = Context.LAYOUT_INFLATER_SERVICE;
                  LayoutInflater vi = (LayoutInflater)
getContext().getSystemService(inflater);
```

```
                    row = vi.inflate(R.layout.listviewicon, parent,
false);
              }
       hintIcon = (HintTile) row.findViewById(R.id.hinticon);
       hintIcon.setIconColor(hint.getOwnerPlayerId(),
hint.getOwnerPlayerColor(), myDrawable1);
       TextView hintText = (TextView) row.findViewById(R.id.hinttext);
       hintText.setText(hint.getHintText());
             return (row);
      }
}
```

Listing 33. TopicHintAdapter

The first thing found is the constructor to the adapter. The context (which is the activity from the adapter is created), the resources and a list with the Hints of the current Topic are passed to the adapter.

Subsequently, it defines the method responsible for generating and filling with the data all necessary controls of the graphical interface of each element of the list. This method is getView ().

GetView() method is called when it need to display a list item. First the XML layout created must be inflated. This consists in to look up the XML layout and create and initialize the equivalent java objects structure. To do this, a new LayoutInflater object is created and the objects structure is generated by the method inflate ().

After this, it gets the data of the TextView for each row, looking for the data of the hint that corresponds to each row. The id of the row is the position parameter.

The icon is obtained in the same way. Then the method setIconColor() is called, this method must draw the icon with the matching color according to the information stored in the hint.

After creating the adapter will be assigned to Listview with setAdapter method () in the class which has initialized the ListWiew (TopicActivity class).

```
             lstTopicHints.setAdapter(this.topicHintAdapter);
```

Listing 34. Setting TopicHintAdapter

4.2.4.3 New Hint UI

In the next sections it relates how is implemented the new design that replaces the ListView of hints, in order to show the hints of the game.
To implement this widget, the next three classes are used:

- The class HintTileButton that extends RadioButton. This class implements each one of the radio button that are grouped in the RadioGroup
- The class HintsGroup that extends RadioGroup. This class groups the RadioButtons
- The class HintsView that extends LinearLayout. This class contains the RadioGroup in a LinearLayout and shows the text of the hints

4.2.4.4 HintTileButton

This class creates a new kind of button that inherits the features of the RadioButtons, and then these buttons will be used on the widget of the hints.

```
public class HintTileButton extends RadioButton {
      Matrix matrix;

      public HintTileButton(Context context, AttributeSet attrs) {
            super(context, attrs);
            this.setButtonDrawable(R.drawable.hintradio);
            this.setPadding(10, 0, 40, 0);
      }

      public void setIconColor(int color) {
            this.setBackgroundColor(color);
            invalidate();
      }

      public void setAvailable(boolean b) {
            this.setEnabled(b);
      }
}
```

Listing 35. HintTileButton

4.2.4.5 HintsGroup

This class will be responsible for creating the group of HintTileButtons. To do this, the next steps have been done:

- Create a group of four HintTileButton buttons on an object HintsGroup.
- Assign an identifier to if each HintTileButton
- Check if each button has been answered previously
- Update the status of the buttons with the corresponding color
- Set the listener for the buttons
- Set the button state after each event

```
      public void setHintList(HintList hintList) {
            if (hintList != null) {
                  this.hintList = hintList;
                  createRadioButton(CREATE_REAL_HINTBUTTONS);
            } else {
                  createRadioButton(CREATE_DUMMY_HINTBUTTONS);
            }
      }

      private final Boolean CREATE_DUMMY_HINTBUTTONS = true;
      private final Boolean CREATE_REAL_HINTBUTTONS = false;

      private void createRadioButton(Boolean createDummyHintButtons) {
            RadioButton rb;
            HintTileButton htb;
            if (createDummyHintButtons) {
                   for (int i = 0; i < 4; i++) {
                         rb = new RadioButton(ctx);
                         this.addView(rb);
                         rb.setText("Dummy");
                   }

            } else {
                   this.setOrientation(RadioGroup.HORIZONTAL);
                   this.setHapticFeedbackEnabled(true);
                   LinearLayout.LayoutParams layoutParams = new
RadioGroup.LayoutParams(RadioGroup.LayoutParams.WRAP_CONTENT,
                               RadioGroup.LayoutParams.WRAP_CONTENT);
                   for (int i = 0; i < 4; i++) {
                         Hint hint = hintList.get(i);
                         htb = new HintTileButton(ctx, attrs);
                         htb.setId(hint.getHintId());
                         if (hint.isAssociated()) { // if hint is
associated, then color
                               // it with owner color
                               htb.setBackgroundColor(hint.getColor());
                               if (!hint.isSelectable()) {// if owner
is not MyPlayer, then
                                     // make it not selectable
                                     htb.setEnabled(false);
                               }
                         }
                         htb.setGravity(Gravity.CENTER);
                         hintTileButtons[i] = htb;
                         this.addView(htb, layoutParams);
                   }
            }
            this.setOnCheckedChangeListener(new
OnCheckedChangeListener() {
                   @Override
                   public void onCheckedChanged(RadioGroup group, int
checkedId) {
                   }
            });
      }
```

Listing 36. HintsGroup

- Another task of this class is updated the status of the buttons when the others players link hints in order to show to the player the necessary information.

```
      public void updateHintList(HintList hintList) {
            HintTileButton htb;
            for (int j = 0; j < this.getChildCount(); j++) {
                  htb = (HintTileButton) this.getChildAt(j);
                  if (hintList.get(j).isAssociated()) { // if hint is
associated, then color
                        // it with owner color

      htb.setBackgroundColor(hintList.get(j).getColor());
                        if (!hintList.get(j).isSelectable()) {// if
owner is not MyPlayer, then
                              // make it not selectable
                              htb.setEnabled(false);
                        }
                    }
                }
            }
```

Listing 37. Updating state

- Set the HintsGroup style:

```
      private void init(AttributeSet attrs) {
            TypedArray a = getContext().obtainStyledAttributes(attrs,
R.styleable.ScrabbleTileSet);
            // Don't forget this
            a.recycle();
      }
```

Listing 38. Setting style

Also it is set the corresponding listeners that will be waiting for the changes in the buttons. Similar listeners have been explained in previous sections, and for brevity here aren´t explained.

4.2.4.6 HintsView

The HintsView class will be responsible to create the widget that will contain the hints. This widget subsequently will be added to the layout of the second screen of the application. This class extends LinearLayout to inherit its methods.

HintsView will undertake the following tasks:

- Create a LinearLayout that contains a HintsGroup
- Create another LinearLayout that contains a text below the first one. Before selecting any button, it will show the instructions to follow, and then when it clicks on a button, it will show the hint corresponding to the selection

```
public void setHintList(HintList hintList) {
            setOrientation(LinearLayout.VERTICAL);
            LinearLayout.LayoutParams lp = new
LinearLayout.LayoutParams(
                        LinearLayout.LayoutParams.FILL_PARENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT);
            lp.setMargins(5, 5, 5, 5);
            this.hintList = hintList;
            hintsGroup = new HintsGroup(context, attrs, hintList);

      hintsGroup.setOnCheckedChangeListener(onHintGroupCheckedChangeLi
stener);
            hintsGroup.setGravity(Gravity.CENTER);
            addView(hintsGroup, lp);
            hintText = new TextView(context);
            hintText.setText("Select one of the hits above");
            lp = new LinearLayout.LayoutParams(
                        LinearLayout.LayoutParams.FILL_PARENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT);
            lp.setMargins(5, 5, 5, 5);
            hintText.setGravity(Gravity.CENTER);
            addView(hintText, lp);
        }
```

Listing 39. HintsGroup

- Set a listener that when you press on a hint button, below it shows the text of the hint corresponding

```
private OnCheckedChangeListener onHintGroupCheckedChangeListener = new
OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int
checkedId) {
                  checkedHintId=checkedId;

      hintText.setText(hintList.getHintByHintId(checkedId).getHintText
());
                  if (onCheckedChangeListener!=null) {
                        onCheckedChangeListener.onCheckedChanged(group,
checkedId);
                  }
            }
      };
```

Listing 40. Listener

- Call to the method updateHintList of the class HintsGroup to update the button state

```
      public void updateHintList(HintList hintList) {
                  hintsGroup.updateHintList(hintList);
            }
```

Listing 41. Updating the state

4.2.5 Gallery

Another widget used in the design of the application is a Gallery that shows the collection of exhibits that can link with the hints displayed in the ListView

To implement the gallery, it goes to follow the same steps than in the Listview implementation:

- Set the Gallery tags in the layout with the required characteristics

```
    <Gallery
     android:layout_below="@id/linkButton"
     android:id="@+id/exhibitgallery"
     android:layout_width="fill_parent"
     android:layout_height="80dip" />
```

Listing 42. Gallery layout

- Initialize the Gallery in TopicActivity activity and obtain the resources

```
    exhibitGallery = new Gallery(this);
    exhibitGallery = (Gallery) findViewById(R.id.exhibitgallery);
```

Listing 43. New gallery

- Create a customized BaseAdapter for displaying the elements on the Gallery. It is explained in the next chapter.
- Set the adapter to the Gallery.

```
exhibitGallery.setAdapter(new ExhibitGalleryAdapter(this,
app.exhibitList));
```

Listing 44. Set the adapter

- The method setGallery() is called in order to implement the listener of the Gallery, and it shows at bottom the Gallery the description of the selected exhibit. After this CheckLinButton() method is called to do its task.

```
       private void setGallery() {

exhibitGallery.setOnItemClickListener(new OnItemClickListener() {
public void onItemClick(AdapterView<?> parent, View v, int position,
long id) {
      exhibitDescription.setText(app.exhibitList.get(position).getDesc
ription());
      checkLinkButton(position);
       }
       });

      }
```

Listing 45. Set the gallery

4.2.5.1 ExhibitGalleryAdapter

ExhibitGalleryAdapter class extends BaseAdapter class in order to inherit its methods. First the adapter receives the context of the class which invokes it (TopicActivity). Then it obtains the resources (list of exhibits) which go to be displayed in the Gallery. Also some design styles are defined.

Below the method getView() returns the view that has to be displayed in each position of the Gallery

```
// ---returns an ImageView view---
    public View getView(int position, View convertView, ViewGroup
parent) {
        Exhibit exhibit = exhibits.get(position);
        ImageView imageView = new ImageView(context);
        if (convertView == null) { // if it's not recycled, initialize
        imageView.setLayoutParams(new Gallery.LayoutParams(150, 120));
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);
        imageView.setBackgroundResource(itemBackground);
        }else {
            imageView = (ImageView) convertView;
        }
        imageView.setImageBitmap(exhibit.getImage());
        return imageView;
```

Listing 46. Views of the gallery

4.2.6 Drawing

4.2.6.1 Introduction

The View class provides a Canvas object with a series of draw methods and Paint classes. Use them to create a visual interface with bitmaps and raster graphics.

The base View class presents a distinctly empty 100-pixel-by-100-pixel square. To change the size of the control and display a more compelling visual interface, it needs to override the onMeasure and onDraw methods.

Within onMeasure the new View will calculate the height and width it will occupy given a set of boundary conditions. The onDraw method is where you draw on the Canvas.

Next listing shows the skeleton code for a new View class, which will be the base for the next sections [6].

```
public class MyView extends View {
// Constructor required for in-code creation
public MyView(Context context) {
super(context);
}
// Constructor required for inflation from resource file
public MyView (Context context, AttributeSet ats, int defaultStyle) {
super(context, ats, defaultStyle );
}
//Constructor required for inflation from resource file
public MyView (Context context, AttributeSet attrs) {
super(context, attrs);
}
@Override
protected void onMeasure(int wMeasureSpec, int hMeasureSpec) {
int measuredHeight = measureHeight(hMeasureSpec);
int measuredWidth = measureWidth(wMeasureSpec);
// MUST make this call to setMeasuredDimension
// or you will cause a runtime exception when
// the control is laid out.
setMeasuredDimension(measuredHeight, measuredWidth);
}
private int measureHeight(int measureSpec) {
int specMode = MeasureSpec.getMode(measureSpec);
int specSize = MeasureSpec.getSize(measureSpec);
[ ... Calculate the view height ... ]
return specSize;
}
private int measureWidth(int measureSpec) {
int specMode = MeasureSpec.getMode(measureSpec);
int specSize = MeasureSpec.getSize(measureSpec);
[ ... Calculate the view width ... ]
return specSize;
}
@Override
protected void onDraw(Canvas canvas) {
[ ... Draw your visual interface ... ]
}
```

Listing 47. skeleton code for a new View class

The Canvas parameter in the onDraw method is the surface it'll use to draw the designed items. Android provides a variety of tools to help draw the design on the Canvas using various Paint objects.

The Canvas class includes helper methods for drawing primitive 2D objects including circles, lines, rectangles, text, and Drawables (images). It also supports transformations that let rotate, translate (move), and scale (resize) the Canvas while it draws on it.
When these tools are used in combination with Drawables and the Paint class (which offer a variety of customizable fills and pens), the complexity and detail that the control can render are limited only by the size of the screen and the power of the processor rendering it.

Unless it conveniently requires a control that always occupies a space 100 pixels square, it will also need to override onMeasure.

The onMeasure method is called when the control's parent is laying out its child controls. It asks the question ''How much space will you use?'' and passes in two parameters: widthMeasureSpec and heightMeasureSpec. They specify the space available for the control and some metadata describing that space.

Rather than return a result, it passes the View's height and width into the setMeasuredDimension method.

4.2.6.2 Drawing hint tiles

In this application it goes to customize the icons which are in each row of the ListView to show to the player the useful information about how the game is developed. For this if one player answers correctly a hint, the icon of the hint will draw with the color of the player´s team in the screens of all the players. So each player will have a global point of view about how the game is running and they won´t try to respond unavailable hints.
To implement these tasks it creates a new class called HintTile that extends View class to inherit from it the methods which goes to use.

```
public class HintTile extends View
```

Listing 48 HintTile extending View

Within onMeasure the new View will calculate the height and width it will occupy given a set of boundary conditions.

```
      @Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)
{
            int measureWidth = measureWidth(widthMeasureSpec);
            int measureHeight = measureHeight(heightMeasureSpec);
            setMeasuredDimension(measureWidth, measureHeight);
      }

      private int measureWidth(int measureSpec) {
            return MeasureSpec.getSize(measureSpec);
      }

      private int measureHeight(int measureSpec) {
            return 20;
      }
}
```

Listing 49. OnMeasure method

It includes a stub for overriding the onDraw method, and implement constructors that call a new init method stub.

```
	public HintTile(Context context, AttributeSet attrs) {
		super(context, attrs);
		init(context);}
```

Listing 50. Constructor

The next step is to create new private instance variables to store the Paint objects that it will use to draw the rectangle.

Then fill in the init method to get instances of the resources you created in the last two steps, and create the Paint objects.

```
private void init(Context ctx) {
		rectPaint = new Paint();
		rectPaint.setStyle(Paint.Style.FILL_AND_STROKE);
		rectPaint.setShadowLayer(2, 0, 0, Color.LTGRAY);
		rectPaint.setColor(hintTileColor);

		rectEmptyPaint = new Paint();
		rectEmptyPaint.setStyle(Paint.Style.STROKE);
		rectEmptyPaint.setShadowLayer(2, 0, 0, Color.LTGRAY);
		rectEmptyPaint.setStrokeWidth(2f);
		rectEmptyPaint.setColor(Color.WHITE);
	}
```

Listing 51. Creating paints

To draw the rectangle, it overrides onDraw and draws the rectangle using the Paint objects created. Once it has drawn, it calls the superclass's onDraw method and let it draw the rectangle as usual.

In the setIconColor() method, it obtains the player that has answered the hint and the team´s color. The rectangle will draw on the ondraw method with a white color if nobody has answered its hint or with the team´s color of the player who has answered the hint.

```
public void setIconColor(int playerId, int color, Drawable drawable) {
		this.icon = ((BitmapDrawable) drawable).getBitmap();
		this.playerId = playerId;
		hintTileColor = color;
		rectPaint.setColor(hintTileColor);
		invalidate();
	}
	@Override
	/**
	 * This method is responsible for all the drawing
	 */
	public void onDraw(Canvas canvas) {
		int canvasWidth = canvas.getWidth();
		int canvasHeight = measureHeight(MeasureSpec.AT_MOST);
		int x1 = WIDTH_PADDING;
		 rect = new Rect(x1, HEIGHT_PADDING, x1 + rectangleWidth,
rectangleWidth + HEIGHT_PADDING);
		if (playerId == -1)
			canvas.drawRect(rect, rectEmptyPaint);
		else
```

```
                    canvas.drawRect(rect, rectPaint);
       }
```

Listing 52. Ondraw method

4.2.6.3 Drawing topic tiles

TopicTile class has to draw the boxes or flags on the main screen used to inform the player about the hints that have been answered for each topic. The first screen has nine buttons with images like background that represents the nine game topics. Each image of them has four boxes or flags that identify the four hints of each topic.

When a player answers correctly a hint of a topic, it draws its box with the color of the player. Being the first box on the left which represents the first hint and so on.

The way to implement it is exactly the same as the way explained on the previous chapter.

The only new is that a broadcast receiver is implemented. It will be listening when a player makes a new association. Therefore, when this occurs the status of the hint is updated adding the new Id of the player who has responded and the color that identifies this player.

```
public void setOnHintChangedListener(OnHintChangedListener listener) {
             onHintChangedListener = listener;
       }
@Override
      public void onHintChanged(TopicTile tile, int hintId, int
newOwnerPlayerId, int newColor) {

      this.topic.getTopicHints().getHintByHintId(hintId).setOwnerPlaye
rId(newOwnerPlayerId, newColor);
            this.invalidate();
       }
```

Listing 53. Listener

4.2.6.4 Scorebar

This class is in charge of implementing the score bar which appears in all the screens to inform about the player name, the player color and the score of the player who is playing with the mobile device.

The way to do this is through the ondraw and onMesure methods like it was explained before in the chapter "using ondraw".

4.2.7 Customize Toggle Buttom

4.2.7.1 Introduction

One of the most complex implementations that have been made in the application has been the button that serves to join a hint with the appropriate exhibit that you want to respond. For it has been designed a toggle button that changes its image and status according to the situation where the player is, allowing only doing certain actions depending on the state that the player is.

4.2.7.2 Specifications

This button must fulfill a variety of specifications to implement its functionality.
The button has to have four states: disabled, invisible locked and open.
These states have been explained in the section 3.5(How to play)

```
private void setLinkButtonState(LinkButtonStates newState) {
            switch (newState) {
            case LOCKED:
                  linkButton.setEnabled(true);
                  linkButton.setChecked(true);
                  linkButton.setVisibility(View.VISIBLE);
                  break;
            case DISABLED:
                  linkButton.setEnabled(false);
                  linkButton.setChecked(true);
                  linkButton.setVisibility(View.VISIBLE);
                  break;
            case OPEN:
                  linkButton.setEnabled(true);
                  linkButton.setChecked(false);
                  linkButton.setVisibility(View.VISIBLE);
                  break;
            case INVISIBLE:
                  linkButton.setVisibility(View.INVISIBLE);
                  break;
            default:
                  break;
            }
```

Listing 54. Link button states

In addition, it must show a different image for each of the states in order to give the player as much information as possible.

4.2.7.3 Implementation

The toggle button is associated with listener to it that executes a variety of tasks whenever it is clicked. This method checks if the link between the hint and the exhibit is successful or not, and if the link has been done before or not. After check these values the method informs to the application class.

```
private OnClickListener linkButtonClickListener = new
OnClickListener() {
            boolean isCorrectPair = false;
            @Override
            public void onClick(View v) {
                  // we get the value of the hint to add to the score
                  // If the id of the exhibit = id the exhibit in the
hint the
                  // link is correct
                  if (exhibitId == hintExhibitId) {
                        isCorrectPair = true;
                  } else {
                        isCorrectPair = false;
                  }
                  // if the next if is true is because myplayer
answered it before
                  // unlink
```

```
                    if (currentHint.getOwnerPlayerId() ==
app.MyPlayerId) {
                          // we break the association if the owner is my
player
                          linkButton.setChecked(false);
                          revokeAssociateHintToExhibit(currentTopicId,
exhibitId, currentHint.getHintId(), isCorrectPair);
                    }
                    // link
                    else {
                          linkButton.setChecked(true);
                          associateHintToExhibit(currentTopicId,
exhibitId, currentHint.getHintId(), isCorrectPair);
                    }
                    //update currentHint

              }

              private void associateHintToExhibit(int currentTopicId,
int exhibitId, int hintId, boolean iscorrectpair) {
                    app.associateHintToExhibit(currentTopicId,
exhibitId, hintId, iscorrectpair);
              }

              private void revokeAssociateHintToExhibit(int
currentTopicId, int exhibitId, int hintId, boolean iscorrectpair) {
                    app.revokeAssociation(currentTopicId, exhibitId,
hintId, iscorrectpair);
              }
        };
```

Listing 55. Logic of the button to get the state after an event

The next method is called either when a hint is pressed or when an exhibit is selected. It checks if the id is from the hint or from the exhibit. Also it checks if the player have selected a hint or an exhibit and it saves the position getting the hint or the exhibit used.

```
public void checkLinkButton(int id, int source) {
            if (source == FROM_EXHIBIT_GALLERY) {
                  exhibitId = id;
            }

            int checkedHintId = hintsView.getCheckedHintId();

            if ((checkedHintId > 0) && (exhibitId > 0)) {
                  // check to see if hint is available:
// hint and exhibit selected is possible to link them, we put
// visible the button
                  linkButton.setVisibility(View.VISIBLE);

                  try {
currentHint =app.topicsList.getTopicByTopicId(currentTopicId)
.getTopicHints().getHintByHintId(checkedHintId);
if (currentHint.isSelectable()) {
      if (currentHint.isAssociated()) {
            // we break the association if the owner is my player
            if (currentHint.getOwnerPlayerId() == app.MyPlayerId) {
                  setLinkButtonState(LinkButtonStates.LOCKED);
            }
// If the owner is another should be disabled for the player
            else {
```

```
setLinkButtonState(LinkButtonStates.DISABLED);
            }
      } else {
            // If nobody have answered it

      setLinkButtonState(LinkButtonStates.OPEN);
            }
}
// if the pair is not selectable because another player is the
                        // owner
      else {

      setLinkButtonState(LinkButtonStates.DISABLED);
                        }

                  } catch (TopicNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                  }
            }

            return;
      }
```

Listing 56. Logic of the button

The code above works with the follow algorithm:

- Check if the call was from the gallery o from the radio group of buttons. And get the id of the hint or of the exhibit corresponding
- Check if there are one hint and one exhibit selected
- If there are a hint and an exhibit selected we put the lock visible, if not we go out of the method
- Check if the hint selected is selectable. If we have an affirmative answer we go to check if the pair is associated. Be selectable means that the hint is free or is property of the player. If the hint is not selectable, we put the lock disabled.
- Check if the hint is associated. Be associated means that the hint has been answered before. If the answer is affirmative, we go to check who is the player that answered it. If the hint is not associated, we go to put the state of the lock open.
- Check if the player is the owner of the hint. If the player is the owner, we put the state of the lock locked. If the player is not the owner the state of the lock will be disable

Therefore one of the states is just to disable the button that is not in use and this is done with the method:

```
linkButton.setEnabled(false);
```

Listing 57. Set the button enable

To set the state visible we use the next code:

```
linkButton.setVisibility(View.VISIBLE);
```

Listing 58. Set the button visible

Finally to change the image displayed on the screen to the other two states is done using an XML file.

The first step is to create an XML file (toggle.xml) and save it inside the folder drawable:

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

<selector xmlns:android="http://schemas.android.com/apk/res/android">

  <item android:state_checked="true" android:state_pressed="true"
      android:drawable="@drawable/stock_lock" />

  <item android:state_checked="true" android:state_focused="true"
      android:drawable="@drawable/stock_lock" />

  <item android:state_checked="true" android:state_focused="false"
      android:drawable="@drawable/stock_lock" />

  <item android:state_checked="false" android:state_pressed="true"
      android:drawable="@drawable/stock_lock_open" />

  <item android:state_checked="false" android:state_focused="true"
      android:drawable="@drawable/stock_lock_open" />

  <item android:state_checked="false" android:state_focused="false"
      android:drawable="@drawable/stock_lock_open" />

  <item android:drawable="@drawable/stock_lock_open" />
</selector>
```

Listing 59. Toggle.xml

The second step is to place the XML created as image in the code where the button is done:

```xml
<ToggleButton
            android:id="@+id/linkButton"
            android:src="@drawable/icon"
            android:background="@drawable/toggle"
            android:layout_height="50dip"
            android:layout_width="50dip"
            android:layout_below="@id/LstOpciones"
            android:layout_centerHorizontal="true"
            android:textSize="8sp"
            android:textOn="Linked"
            android:textOff="Not linked" />
```

Listing 60. Link button layout

This code ensures that while the toggle button is not checked or by default displays the image of the padlock open and if it answered correctly shows a locked padlock image. Images of the three states needed are obtained with these last two images, and the image of the padlock disabled

4.2.8 Reslet

4.2.8.1 Common classes

The following classes are available on the tree project. They are used by the server and the clients in order to produce the serialized representation of the GameMessage object and to deserialize incoming representations.

- GameMessage
- GameMessageList
- GameMessageResource
- GameMessageParcelable

GameMessageResource is an interface annotated with Restlet annotations:

```
public interface GameMessageResource {
    @Get
    public ArrayList<GameMessage> retrieve();

    @Put
    public void send(GameMessage gameMessage);

    @Delete
    public void delete(GameMessage gameMessage);
}
```

Listing 61. GameMessageResource

It represents the contract passed between the client and the server.

There are another three common classes. They are GameMessage, GameMessageParcelable and GameMessageList. GameMessage as shown in the previous section is a POJO class and represents the data structure of the objects exchanged, and GameMessageList is a list of objects GameMessage. And GameMessageParcelable in a class similar to GameMessage but implements parcelable.

4.2.8.2 Server part

For the most part, the bulk of a RESTful application built with the Restlet framework requires the use of two base classes: Application and Resource. Logically speaking, an Application instance maps URIs to Resource instances. Resource instances do the work of handling the basic CRUD commands, which are, of course, mapped to GET, POST, PUT, and DELETE. In this application are necessary only three of them, GET, PUT and DELETE.

It creates a starting point by extending from the framework's Application class. The first task is created a new component. After this it adds a new HTTP server listening on port 8080, and then it attaches the application to the component and starts it.

```
public class ScrabbleServerApplication extends Application {
      private static final int PORT_NUMBER = 8080;
    public static void main(String[] args) throws Exception {
        Component component = new Component();
        component.getClients().add(Protocol.FILE);
        component.getServers().add(Protocol.HTTP, PORT_NUMBER);
        component.getDefaultHost().attach(new
ScrabbleServerApplication());
        component.start();
    }
```

Listing 62. ScrabbleServerApplication

Now it creates a root Restlet that will receive all incoming calls. In this class, it defines `resources` that respond to URIs. This definition process is done with the framework's Router class. To the URI "/scrabbleGame/gameStatus", it needs to specify which object can handle these requests. This object is an instance of the framework's Resource type. It links objects with URIs by attaching them to a Router instance:

```
public Restlet createInboundRoot() {
        Router router = new Router(getContext());
        getConnectorService().getClientProtocols().add(Protocol.FILE);
public Restlet createInboundRoot() {
        Router router = new Router(getContext());
        getConnectorService().getClientProtocols().add(Protocol.FILE);
        router.attachDefault(dir);
        router.attach("/scrabbleGame/gameStatus",
GameServerResource.class);

        return router;
    }
```

Listing 63. Attach the resource

It is necessary also creates the file and the directory where the data will be store but this is omitted for brevity.

Now that the Application instance has been defined to handle one URI pattern, the next step is implement the Resource. Resource types in the Restlet framework are known as *Restlets*. They are the heart of any RESTful application developed with the Restlet framework. Unlike the Application type, the base Resource class is not abstract. It's more like a template with default behavior that you can override as needed.

Via the Router class in the ScrabbleServerApplication class, the URI is linked to the GameServeResource class. GameServeResource class implements the three methods that the application needs.

```
public class GameServerResource extends ServerResource implements
            GameMessageResource {
@Get
      public ArrayList<GameMessage> retrieve() {

            return gameMessageList;
      }
@Put
      public void send(GameMessage gameMessage) {
            //First check if message exists
            for (GameMessage message : gameMessageList) {
                  if (message.isEqual(gameMessage)) {
                        message.setActive(true);
                        return;
                  }
            }

            //else add message
            gameMessage.setActive(true);
            gameMessageList.add(gameMessage);
      }

@Delete
      public void delete(GameMessage gameMessage) {
            for (GameMessage message : gameMessageList) {
                  if (message.isEqual(gameMessage)) {
                        message.setActive(false);
                        break;
                  }
            }
      }
}
```

Listing 64. GameServerResource

The correct way to work with REST Web services is working with representations of the data; REST said that we cannot work directly with the data. But this application works directly with the data due to we want a simple way to work with Web Services.

4.2.8.3 Client part

The Restlet client consumes the Web Service of the server that it is just design, it is built into the application and it runs as a service running in the background since the start of the application. Thus synchronously every x second the method retrieve() is called, in order to implement the GET command. Also asynchronously each time that one question of the game is answered correctly, the method send() is called, in order to implement the PUT command.

In the client side also must be implemented the common classes that are explained in the previous server part chapter, i.e., GameMessage, GameMessageList, GameMessageParcelable and GameMessageResource.

Now it goes to ignore the parts of the code corresponding to the implementation of the RestClientService class as a service and it goes to analyze the code that is required to implement the Restlet client.

First the resource proxy is initialized with the URI that has been defined. GAE doesn't support HTTP chunked encoding, therefore serialized object can't be sent (via POST or PUT) to a GAE server. Since Restlet Framework version 2.1 M4 there are a workaround available that buffers the HTTP entity to prevent chunk encoding. To use it, it calls to the ClientResource.setRequestEntityBuffering(boolean) method with a "true" value.

After this it associates the interface GameServeResource by clientResourceGameStatus.wrap(GameMessageResource.class).

```
ClientResource clientResourceGameStatus = new
ClientResource(SERVER_URL + "/scrabbleGame/gameStatus");
public void onCreate() {
            super.onCreate();
clientResourceGameStatus.setRequestEntityBuffering(true);
```

```
            resource                                                 =
clientResourceGameStatus.wrap(GameMessageResource.class);
}
```

Listing 65. Create a client

It is planned that every certain time it makes the GET request, in order to implement this request periodically, it makes use of the java.util.TimerTask and java.util.**Timer** classes within the Java API.

The Timer class allows running a scheduled tasks, each Timer is actually a Java Thread that runs all the TimerTask to be assigned sequentially.

The TimerTask class represents the task to be executed periodically. To use it, it needs to build a class that extends TimerTask class and override the run () method, inside this method, there is the code to run when launching the task.

When the time expires during which the GET request is made, it calls the cancel () method of the TimerTask class to finish executing the task without stop the run Timer.

```
public int onStartCommand(Intent intent, int flags, int startId) {
            if ((flags & START_FLAG_RETRY) == 0) {
                  try {
                        startPeriodicUpdateTask();
                  } catch (Exception e) {
                  }
            } else {
                  startPeriodicUpdateTask();
            }
            return Service.START_STICKY;
      }
```

```
// service business logic
      private void startPeriodicUpdateTask() {
            if (!timerTaskRunning) {
                  try {
      gameUpdateTask = new GameUpdateTask();
      timerTaskRunning = true;
      timer.scheduleAtFixedRate(gameUpdateTask, 0, UPDATE_INTERVAL);
                  } catch (Exception e) {
                  }
               }
      }
```

```
      private void stopPeriodicUpdateTask() {
            if (timerTaskRunning) {
                  timerTaskRunning = false;
                  if (timer != null) {
                        gameUpdateTask.cancel();
                        timer.purge();
                  }
            }
      }
```

Listing 66. Timed the service

Finally the three methods are called. GET method is called periodically and PUT and DELETE methods are called when a pair hint-exhibit is linked or unlinked respectively.

```
private class GameUpdateTask extends TimerTask {
            public void run() {
                  _getGameUpdate();
            }
      }
private void _getGameUpdate() {
        try {
                  // Get the remote contact
                  messages = resource.retrieve();
            } catch (Exception e) {
                  Log.e(TAG, e.toString());
          }
      }
```

Listing 68. GET method

```
      private void _sendToServer(int playerId, int hintId, int
exhibitId) {
            // create resource for checking the result of the "send"
            GameMessage newMessage = new GameMessage();
            newMessage.setplayerId(playerId);
            newMessage.setHintId(hintId);
            newMessage.setExhibitId(exhibitId);
            try {
                  resource.send(newMessage);
            } catch (Exception e) {

            }
      }
```

Listing 69. PUT method

```
private void _deleteInServer(int playerId, int hintId, int exhibitId)
{
            // create resource for checking the result of the "send"
            GameMessage newMessage = new GameMessage();
            newMessage.setplayerId(playerId);
            newMessage.setHintId(hintId);
            newMessage.setExhibitId(exhibitId);
            newMessage.setActive(false);
            try {
                  resource.delete(newMessage);
            } catch (Exception e) {

            }
      }
```

Listing 70. DELETE method

4.2.8.4 How to install Restlet Framework

There are two ways to use Restlet within Eclipse IDE. The first one is to create a Java project and use the Restlet JARs as external dependencies. This is very simple and works well in most cases.

The second way is to install Restlet JARs as Eclipse/OSGi bundles. All Restlet JARs including dependencies are valid OSGi bundles, so this is very convenient if you work in an Eclipse plugRestlet in environment, such as an Eclipse RCP application. You then just need to create a plug-in project.

For the first option should take the following steps:

1. Connect to the website of Restlet to the downloads section http://www.restlet.org/downloads/

The Restlet Framework is available under three versions, tagged like Debian releases:

- *"Stable"* is the release that we recommend for applications in production. The API of this release is frozen and only bug fixes are made.
- *"Testing"* is the release that we recommend for new developments. We are looking for feed-back and contributions regarding bugs and missing features.
- *"Unstable"* is the release where active development happens. It corresponds to builds of the development trunk passing all unit tests.

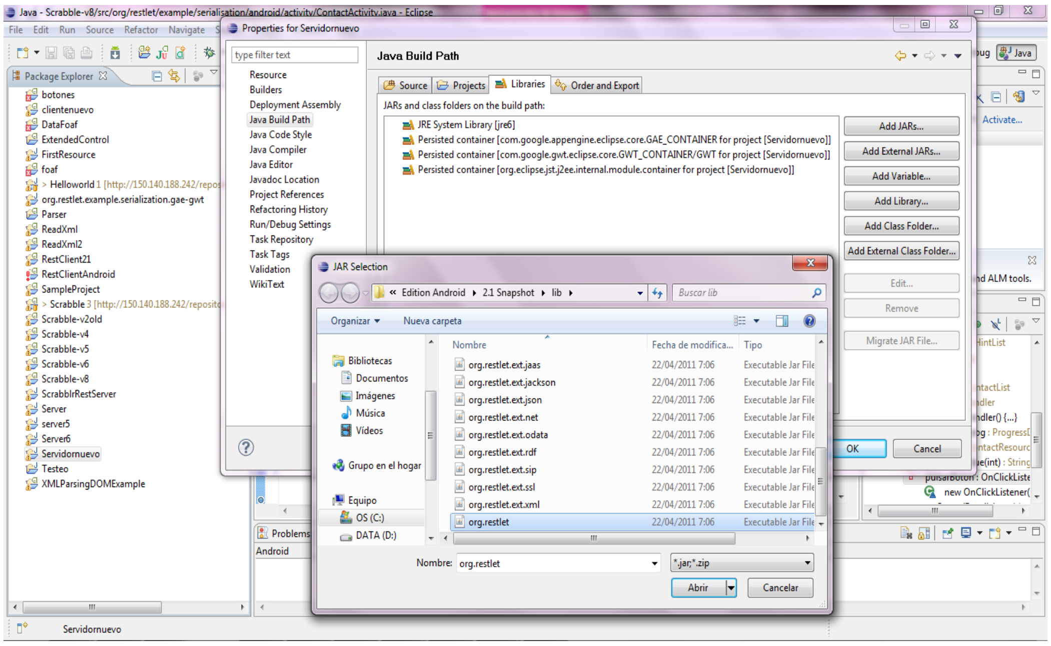
Due to several bugs that were found in previous versions that affected the implementation of this application must be installed the *Version 2.1 Milestone 4* or later versions, as though they are in testing phase have solved a number of problems that prevented work on previous versions.

2. Select available editions of the chosen version of the framework

The following shows available editions:

- Edition for Java SE
- Edition for Java EE
- Edition for Google AppEngine
- Edition for Google Web Toolkit
- Edition for Android

3. Select Android edition and download the Zip archive or run the Windows installer. Both download into the computer the necessary files to run the application.
4. In Android project and in the Java project which contain client and server respectively select with right mouse button:
5. Properties→Java Built Path→ libraries→ add external jars


Java - Scrabble-v8/src/org/restlet/example/serialisation/android/activity/ContactActivity.java - Eclipse

Properties for Servidornuevo — Java Build Path

JARs and class folders on the build path:

- JRE System Library [jre6]
- Persisted container [com.google.appengine.eclipse.core.GAE_CONTAINER for project [Servidornuevo]]
- Persisted container [com.google.gwt.eclipse.core.GWT_CONTAINER/GWT for project [Servidornuevo]]
- Persisted container [org.eclipse.jst.j2ee.internal.module.container for project [Servidornuevo]]

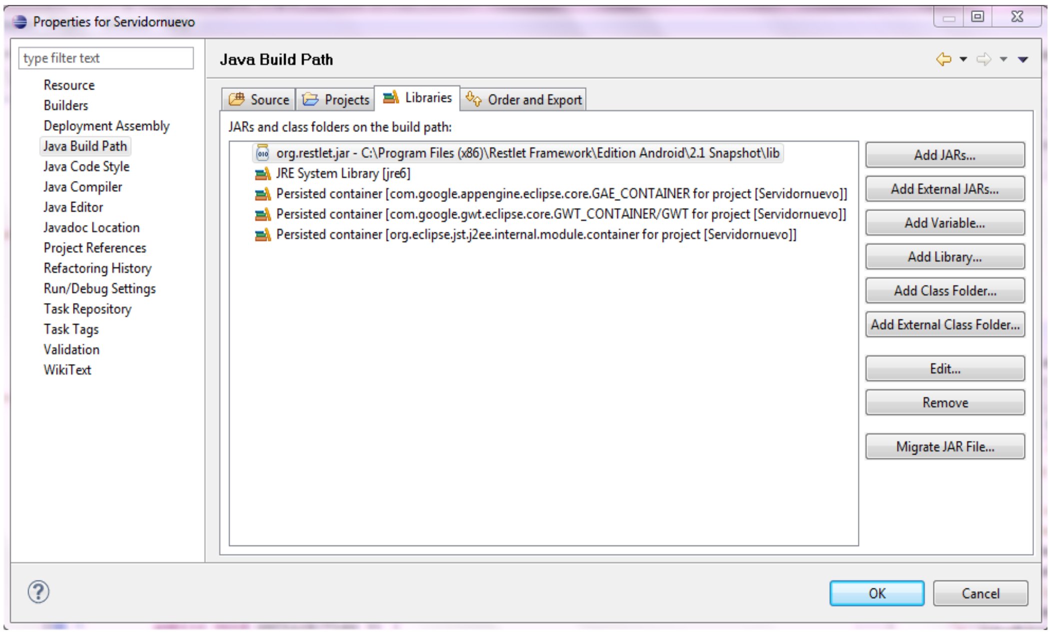
JAR Selection — « Edition Android ▸ 2.1 Snapshot ▸ lib

Nombre	Fecha de modifica...	Tipo
org.restlet.ext.jaas	22/04/2011 7:06	Executable Jar File
org.restlet.ext.jackson	22/04/2011 7:06	Executable Jar File
org.restlet.ext.json	22/04/2011 7:06	Executable Jar File
org.restlet.ext.net	22/04/2011 7:06	Executable Jar File
org.restlet.ext.odata	22/04/2011 7:06	Executable Jar File
org.restlet.ext.rdf	22/04/2011 7:06	Executable Jar File
org.restlet.ext.sip	22/04/2011 7:06	Executable Jar File
org.restlet.ext.ssl	22/04/2011 7:06	Executable Jar File
org.restlet.ext.xml	22/04/2011 7:06	Executable Jar File
org.restlet	22/04/2011 7:06	Executable Jar File

Nombre: org.restlet — *.jar;*.zip

Figure 30. How to intall Restlet framework

6. Add org.restlet.jar which is stores in the file *lib* inside the framework directory previously downloaded


Properties for Servidornuevo — Java Build Path

JARs and class folders on the build path:

- org.restlet.jar - C:\Program Files (x86)\Restlet Framework\Edition Android\2.1 Snapshot\lib
- JRE System Library [jre6]
- Persisted container [com.google.appengine.eclipse.core.GAE_CONTAINER for project [Servidornuevo]]
- Persisted container [com.google.gwt.eclipse.core.GWT_CONTAINER/GWT for project [Servidornuevo]]
- Persisted container [org.eclipse.jst.j2ee.internal.module.container for project [Servidornuevo]]

Figure 31. How to install Restlet framework

4.2.9 QR Codes

4.2.9.1 Introduction

The study of the QR codes and their implementation aren´t objectives of this project. Therefore it isn´t reported a detailed study of them. There is in the Appendix D a little report that explains the basic features of them, as well as the way to implement the tools necessary to use the camera in the application on the mobile.

The mode of operation for retrieving the information of the exhibits is using the *QR Codes*. The QR codes will be used to decode the exhibits that the players should go finding. Each exhibit has associated a QR Code. Thus in the beginning of the game the players will have all the exhibits decoded. When a player finds a QR code, he will scan the QR code obtaining the exhibit correspondent, if the application found the exhibit related to the *QR Code* scanned, it will show on the Gallery of the Topic screen the information about the exhibit.

In case the *QR Code* scanned doesn’t match with some of the stored in the database, the application will show nothing.

The way to find the QR codes around the city in the future will be through maps. This feature isn´t already implemented and it is one of the future works that might be done. The player will follow some instructions in the map, and he will try to find the locations where the QR codes will be. Then with the camera of the mobile devices will decode the QR code and the exhibit will appear in the gallery, making possible that the exhibit can be linked with one hint.

All the features are implemented already in the application except the maps feature. It isn´t necessary the change of application on the mobile to take the photo which decodes the QR code, because thanks Zxing is possible in Android an alternative method for simple access to barcode scanning, via Intents rather than direct use of project code.

4.2.9.2 ScanningViaIntent

- Manually

If the *Barcode Scanner* is installed on an *Android* device, the developer can have it scan for him and return the result, just by sending it an Intent. For example, it is possible to hook up a button to scan a *QR* code like the next Listing shows:

```
private OnClickListener qrScanButtonOnClick =new OnClickListener () {

            @Override
public void onClick(View v) {
Intent intent = new Intent("com.google.zxing.client.android.SCAN");
intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
startActivityForResult(intent, 0);
            }
      };

public void onActivityResult(int requestCode, int resultCode, Intent
intent) {
      if (requestCode == 0) {
            if (resultCode == RESULT_OK) {
            String contents = intent.getStringExtra("SCAN_RESULT");
      String format = intent.getStringExtra("SCAN_RESULT_FORMAT");
                        // Handle successful scan
                        //find if CODE exists
if (app.exhibitList.isCodeExists(contents)) {
      app.exhibitList.markExhibitAsFound(contents);
      ExhibitList tmp = app.exhibitList;
      exhibitGalleryAdapter.notifyDataSetChanged();
                        }
else {
      }
} else if (resultCode == RESULT_CANCELED) {
      // Handle cancel
                  }
            }
      }
```

Listing 71. Implement QR codes

- IntentIntegrator

  ZXing team provides a small library of classes that encapsulate some of the details of above. See IntentIntegrator for a possibly easier way to integrate. In particular this will handle the case where *Barcode Scanner* is not yet installed.
  `http://code.google.com/p/zxing/source/browse/trunk/android-integration/src/com/google/zxing/integration/android/IntentIntegrator.java`

Chapter 5

Conclusions

After the work presented in this paper it`s possible to say that the first prototype of the game is running satisfactorily. There is a lot of work to do before this game may be a commercial game, not only an academic prototype, but the bases of the game are done.

The communications between classes, the Web Service to manage the sessions of all the players and other main features are implemented working efficiently.

It has tried to use the most modern technologies available to do the application. It has used Web Services that are now one of the most popular software nowadays. And the most popular architecture between Web Services certainly is the REST architecture, like figure 10 shows.

To implement the REST Web Service it has been used the Restlet framework. Although at the beginning the Restlet frameworks caused a lot of problems finally and after understand it, it´s possible to work with it acceptably.

The idea of the game is extend the location or scenario of the game not only to cities or museums, the location will be able to anywhere. The creation of new locations should be quick and easy. The application only needs a XML file with a particular structure to adapt to a new location. One of the future works that might be realized would be the creation of a tool to do this.

Another idea to develop in the future would be the creation of one feature that informs to the player with a map where he and the other player are and where the possible locations of the exhibits are.

Also it will be necessary improve the feedback given to the player and adapt the Web Service to a totally REST architecture.

Maybe also an interface to introduce the login and the team of the player would be useful.

Finally, an experiment with real players might be necessary to test the application and find possible improvements to do.

References

[1] Nosowitz, Dan (2008-11-08). "Tennis for Two, the World's First Graphical Videogame". Retromodo. Gizmodo. http://gizmodo.com/5080541/retromodo-tennis-for-two-the-worlds-first-graphical-videogame.

[2] Sintoris C., Stoica A., Papadimitriou I., Yiannoutsou N., Komis V., Avouris N. (2010). MuseumScrabble: Design of a mobile game for children's interaction with a digitally augmented cultural space

[3] Mark Dunlop and Stephen Brewster(2002). The Challenge of Mobile Devices for Human Computer Interaction

[4] Peter Farago (2011) Android Special Report: Is Samdroid the new Wintel?. http://blog.flurry.com/bid/54035/Android-Special-Report-Is-Samdroid-the-new-Wintel

[5] Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, Ph.D. Thesis, University of California, Irvine, Irvine, California, 2000.
http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[6] Reto Meier. Professional Android 2 Application Development. Wiley Publishing, 2010.

[7] Ed Burnette. *Hello, Android Introducing Google's Mobile Development Platform*. Third Edition. The Pragmatic Bookshelf, July 2010.

Appendix

A. ANDROID

A.1 INTRODUCTION

Android is a complete software platform for different sorts of devices, including mobile devices, which were used in this thesis. This means that it is an operating system and software stack to run *Android* specific software.

It was introduced by *Google Inc*. in 2007 and is aimed at changing the way mobile applications are developed. The system is based on *Linux* and runs *Java* programmed applications on top.

The big difference from normal mobile *Java* development is that *Android* supports a large part of the *Java SE*[1] through *Apache Harmony* instead of the crippled *Java ME package*. *Android* also does not use *Sun's Java Virtual Machine* or even *Java* byte code, but instead uses its own engine named *'Dalvik'* together with its own set of byte code. This means that *Android* does not run *Java* applications, but rather *Android* applications programmed in the *Java* language with *Android* extensions [6].

A.2 THE LINUX KERNEL

Android is based on the *Linux* 2.6 kernel. Just as in any other *Linux* system, the *Android* kernel provides core system functionalities mostly linked to the hardware abstraction [6].

Even though *Android* is based on a *Linux* kernel, it is not a *GNU/Linux*, which is what is normally thought of in relation to the word *'Linux'*. *Android* does not include all standard *Linux* utilities; it has no native window manager and it lacks compatibility with the standard *C/C++* libraries of the *GNU C* library (*glibc*).

Android includes its own *C* library (*libc*) called *Bionic*. *Bionic* is optimized for embedded applications, with small and fast code paths and a custom implementation of *POSIX* threads (*pthreads*). Furthermore, it has built-in support for some *Android* specific features like system properties and logging. The reason that *Bionic* is not fully compatible with that it lacks *glibc* some *POSIX* features which are not needed in *Android*. These missing features are mostly concerning *C*++ libraries.

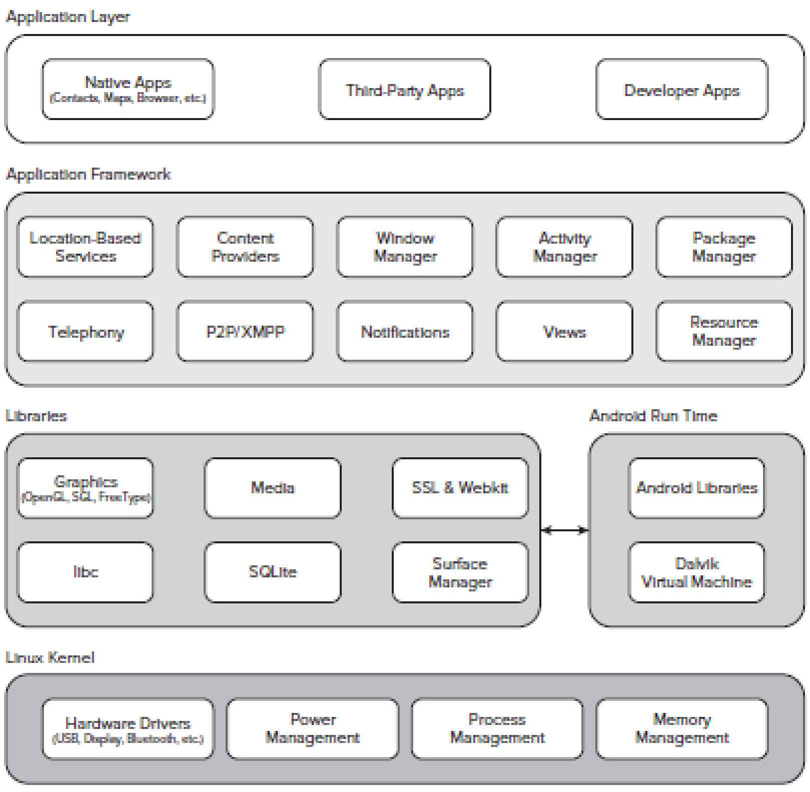


Figure 32 The Android Software Stack

A.3 The System libraries

System libraries (written in *C/C++*) are exposed through the application framework and provide graphics capabilities, hardware acceleration etc. and also includes a fully functional *SQL* server (*SQLite*), *SSL* libraries for security, an *OpenCORE* based media framework and *WebKit* for web browsing [6].

A.4 Android runtime

A.4.1 Core libraries

Android implements a core set of standard *Java* libraries through *Apache Harmony*. This means that the *Android Java library set* differs from *Java ME* in that it is not sandboxed and stripped, but is rather 'desktop *Java* on a small screen'. It is almost fully compatible with *Java SE* 1.6 except for graphical parts, but in return adds its own graphical environment and a large collection of fitting help classes and implementations to speed up *Android* development [6].

A.4.2 Dalvik Virtual Machine

Dalvik is the *Virtual Machine* (*VM*) of the *Android* system. It is actually technically not a *Java Virtual Machine* (*JVM*) since it comes with a whole different instruction set than a *JVM* would interpret. *Android* applications are written in *Java* code, but compiled to *Dalvik* byte code.

Dalvik is a register based machine, as opposed to most *JVM*s which have a stack based structure. This structure choice will normally require larger instructions and thus often larger files, but that is not true in the case of Dalvik due to its trimmed down *VM*. *Dalvik* will also reduce the number of *CPU* instructions needed per *VM* instruction due to the fact that a register machine does not read from a stack and push the answer back but rather keeps addresses in instructions. This means that the *Dalvik* engine will generally be faster and load smaller files than a *JVM* would do.

It should be noted that the *Dalvik* engine will not do some optimizations such as loop unrolling in the manner that a normal *JDK*/*JVM* would. This means that the programmer will have to do some extra optimizations and smart design choices.

A.4.3 THE CODE VERIFIER

The local system (device) *VM* that loads the compiled *Dalvik* code (*.dex*) contains a system specific verifier. This verifier will check that the loaded code is type safe, is compatible with the device and also perform local and system specific optimizations of the byte code where possible.

A.5 APPLICATION FRAMEWORK

The application framework contains the code base for hardware interaction, the specialized design patterns and other parts of the *Android* development methodology. The main parts are

- **Activity manager** handles the activities of the application.
- **Views** are the building blocks of *UI* creation and manipulation.
- **Notification manager** provides different types of signaling alternatives to make the user and device aware of changes.
- **Content providers** provide a simple way of sharing data between applications. A normal *Android* application has its own file system which cannot be manipulated from other applications. Data channels are instead opened through content providers.
- **Resource manager** handles the resources of the application. Application resources are entities such as *UI* elements, strings and other parts that can benexternalized from the code into *XML* files and fetched when needed [6].

A.6 THE STRUCTURE OF AN APPLICATION

”*All applications are created equal.*”

This means that there is no difference between built-in applications and third-party applications.

It is also possible to replace core applications such as the contact list, dialer or home screen. An application can be built by one or more parts. These parts are defined in *XML* and described in this section.

Each application defines its permissions in the application manifest. Typical permissions are *internet* access and camera usage. The user decides at install time if these needed permissions are acceptable. This means that the user will only agree to permission once, but it will be granted for as long as the application is installed.

A.6.1 ACTIVITY

An *activity* is a part of an application which displays a graphical interface of some sort and can retrieve input from the user. Most applications will be made of a collection of activities [7]. These activities contain *views* to display information. A very simple example of an *activity* can be seen in Listing 1.

```
import android.app.Activity;
import android.os.Bundle;

public class MyActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Listing 72 Basic Activity Stub

A.6.2 SERVICE

Activities can contain services. Services are background processes which can run during a long period of time. These will most often sleep when there is nothing to be done and process information when wanted. This is somewhat like the concept of ”server *daemons*” in *UNIX* [7]. An example of a service use case would be downloading weather forecast information for display in a widget.

A.6.3 BROADCAST RECEIVER

A *Broadcast Receiver* receives and may answer specific events. *Broadcast Receivers* must process very quickly and are used as small background services which can run when called. An example would be a *Broadcast Receiver* which receives an event which says that a photo has been taken and logs this to a specific file.

A.6.4 CONTENT PROVIDERS

Android applications do not share a file system beyond data saving on an *SD* card. Content providers are meant to serve applications with useful data of specific sorts. This data can for example be contacts in a phone book, bookmarks from a browser etc.

A.7 THE LIFE CYCLE OF AN APPLICATION

A.7.1 INTENTS

Applications are started from a system called ”*Intents*”. *Intents* define that an application is capable of handling a job. That job can be, for example ”viewing a web page", "calling a phone number", "pick a photo" or "open the pod bay doors". When one application wants to perform a task that another application should implement it sends out an *Intent* [7].

If there is an application that can handle the *Intent* it will be started. If there are more than one application that can handle an intent, one will get a menu to choose from or the preferred one will be selected automatically depending on the situation.

Explicit staring of an application from for example an application menu is done by sending a specific intent to start said application.

A.7.2 KNOWING THE STATE OF AN APPLICATION

Android applications need to keep track of their state at all times. Instead of explicitly starting and stopping applications, the system is designed to be able to pause applications when they are not displayed and later possibly bring them back. This means that an application has to save its state when it gets a pause command or data might be lost if *Android* decides to shut down the application in order to save memory or battery power. During its lifetime, each activity of an *Android* program can be in one of several states, as shown in Figure 32.

If an application does not correctly pause its execution when told to, it will continue to run in background until it might be killed at any time without any further warning. This behavior makes it very simple to resume a session from the previously run since paused applications will keep their memory space until some other application claims it. It can also have the effect that applications will become useless when resumed unless first explicitly killed and later started again since clicking a "back" button or similar and starting the application again will not have the expected effect of an application restart unless the applications is designed for this behavior.

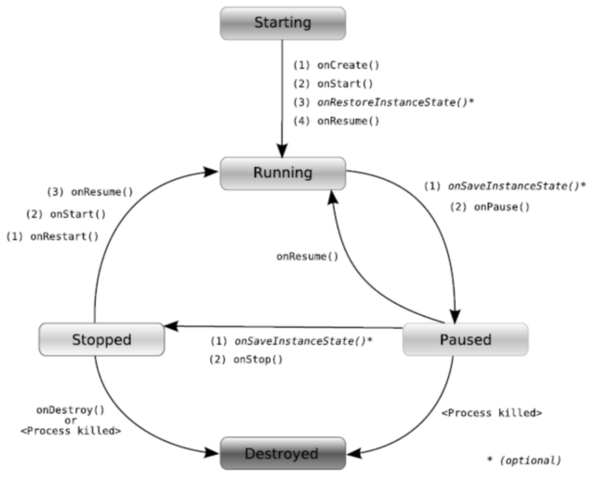


Figure 33 Life Cycle of an Android Activity

B. Subversion (SVN)

Subversion is an open source version control system. Founded in 2000 by CollabNet, Inc., the Subversion project and software have seen incredible success over the past decade. Subversion is developed as a project of the Apache Software Foundation.

B.1 Features

Apache Subversion is a full-featured version control system originally designed to be a better CVS. Subversion has since expanded beyond its original goal of replacing CVS, but its basic model, design, and interface remain heavily influenced by that goal. Even today, Subversion should still feel very familiar to CVS users.

The following list of features shows a basic understanding of what version control is and how version control systems work in general.

- **Most CVS features.**

CVS is a relatively basic version control system. For the most part, Subversion has matched or exceeded CVS's feature set where those features continue to apply in Subversion's particular design.

- **Directories are versioned.**

Subversion versions directories as first-class objects, just like files.

- **Copying, deleting, and renaming are versioned.**

Copying and deleting are versioned operations. Renaming is also a versioned operation, albeit with some quirks.

- **Free-form versioned metadata ("properties").**

Subversion allows arbitrary metadata ("properties") to be attached to any file or directory. These properties are key/value pairs, and are versioned just like the objects they are attached to. Subversion also provides a way to attach arbitrary key/value properties to a revision (that is, to a committed change set). These properties are not versioned, since they attach metadata to the version-space itself, but they can be changed at any time.

- **Atomic commits.**

No part of a commit takes effect until the entire commit has succeeded. Revision numbers are per-commit, not per-file, and commit's log message is attached to its revision, not stored redundantly in all the files affected by that commit.

- **Branching and tagging are cheap (constant time) operations.**

There is no reason for these operations to be expensive, so they aren't.

Branches and tags are both implemented in terms of an underlying "copy" operation. A copy takes up a small, constant amount of space. Any copy is a tag; and if you start committing on a copy, then it's a branch as well. (This does away with CVS's "branch-point tagging", by removing the distinction that made branch-point tags necessary in the first place.)

- **Merge tracking.**

Subversion 1.5 introduces merge tracking: automated assistance with managing the flow of changes between lines of development, and with the merging of branches back into their sources. The 1.5 release of merge tracking has basic support for common scenarios; we will be extending the feature in upcoming releases.

- **File locking.**

Subversion supports (but does not require) locking files so that users can be warned when multiple people try to edit the same file. A file can be marked as requiring a lock before being edited, in which case Subversion will present the file in read-only mode until a lock is acquired.

- **Symbolic links can be versioned.**

Unix users can place symbolic links under version control. The links are recreated in Unix working copies, but not in win32 working copies.

- **Executable flag is preserved.**

Subversion notices when a file is executable, and if that file is placed into version control, its executability will be preserved when it it checked out to other locations. (The mechanism Subversion uses to remember this is simply versioned properties, so executability can be manually edited when necessary, even from a client that does not acknowledge the file's executability, e.g., when having the wrong extension under Microsoft Windows).

- **Apache network server option, with WebDAV/DeltaV protocol.**

Subversion can use the HTTP-based WebDAV/DeltaV protocol for network communications, and the Apache web server to provide repository-side network service. This gives Subversion an advantage over CVS in interoperability, and allows certain features (such as authentication, wire compression) to be provided in a way that is already familiar to administrators

- **Standalone server option (`svnserve`).**

Subversion offers a standalone server option using a custom protocol, since not everyone wants to run an Apache HTTPD server. The standalone server can run as an inetd service or in daemon mode, and offers the same level of authentication and authorization functionality as the HTTPD-based server. The standalone server can also be tunnelled over ssh.

- **Parseable output.**

All output of the Subversion command-line client is carefully designed to be both human readable and automatically parseable; scriptability is a high priority.

- **Localized messages.**

Subversion uses gettext() to display translated error, informational, and help messages, based on current locale settings.

- **Interactive conflict resolution.**

The Subversion command-line client (`svn`) offers various ways to resolve conflicting changes, include interactive resolution prompting. This mechanism is also made available via APIs, so that other clients (such as graphical clients) can offer interactive conflict resolution appropriate to their interfaces.

- **Repository read-only mirroring.**

Subversion supplies a utility, `svnsync` for synchronizing (via either push or pull) a read-only slave repository with a master repository.

- **Write-through proxy over WebDAV.**

Subversion 1.5 introduces a write-through proxy feature that allows slave repositories (see read-only mirroring) to handle all read operations themselves while passing write operations through to the master. This feature is only available with the Apache HTTPD (WebDAV) server option.

- **Natively client/server, layered library design with clean APIs.**

Subversion is designed to be client/server from the beginning; thus avoiding some of the maintenance problems which have plagued CVS. The code is structured as a set of modules with well-defined interfaces, designed to be called by other applications.

- **Binary files handled efficiently.**

Subversion is equally efficient on binary as on text files, because it uses a binary diffing algorithm to transmit and store successive revisions.

- **Costs are proportional to change size, not data size.**

In general, the time required for a Subversion operation is proportional to the size of the *changes* resulting from that operation, not to the absolute size of the project in which the changes are taking place.

- **Bindings to programming languages.**

The Subversion APIs come with bindings for many programming languages, such as Python, Perl, Java, and Ruby. (Subversion itself is written in C.)

- **Changelists.**

Subversion 1.5 introduces changelists, which allows a user to put modified files into named groups on the client side, and then commit by specifying a particular group. For those who work on logically separate changesets simultaneously in the same directory tree, changelists can help keep things organized.

B.2 Subversive

Subversive is a totally new Eclipse plug-in, that provides you a probability to use supported SVN® clients easily directly from your workbench. Friendly user interface of Subversive makes it much more comfortable to operate repositories. All SVN® operations are supported and there are no difficulties while working with command line from now on.

B.2.1 Getting SVN Connectors

In order to work with SVN the plug-in user should also install one of SVN connectors, which are distributed from different update-site through the legacy reasons. The plug-in won't work without SVN Connectors.

B.2.2 Subversive unique features

- Realizes the full support of Repository location layouts, recommended by Subversion®.
- Provides the probability to browse revisions right in the repository location view.
- Allows to freeze svn:externals while creating tags and branches.
- Provides the automatic search of Eclipse projects in the repository.
- Provides the support of SVN® operations for working sets and for sets of project.
- Subversive plug-in is widely extensible and has well defined API.
- Bug reporting included to realize opportune support and provide the community a probability to take part in the improving cycle of the product, made for them, in the nearest time.

B.2.3 Requirements

Subversion versions 1.1, 1.2, 1.3 or 1.4
JDK 1.5 or higher
Eclipse 3.3

B.2.4 Subversive installation instructions

To install Subversive:

1. Start Eclipse and select menu item *'Help > Install New Software...'*

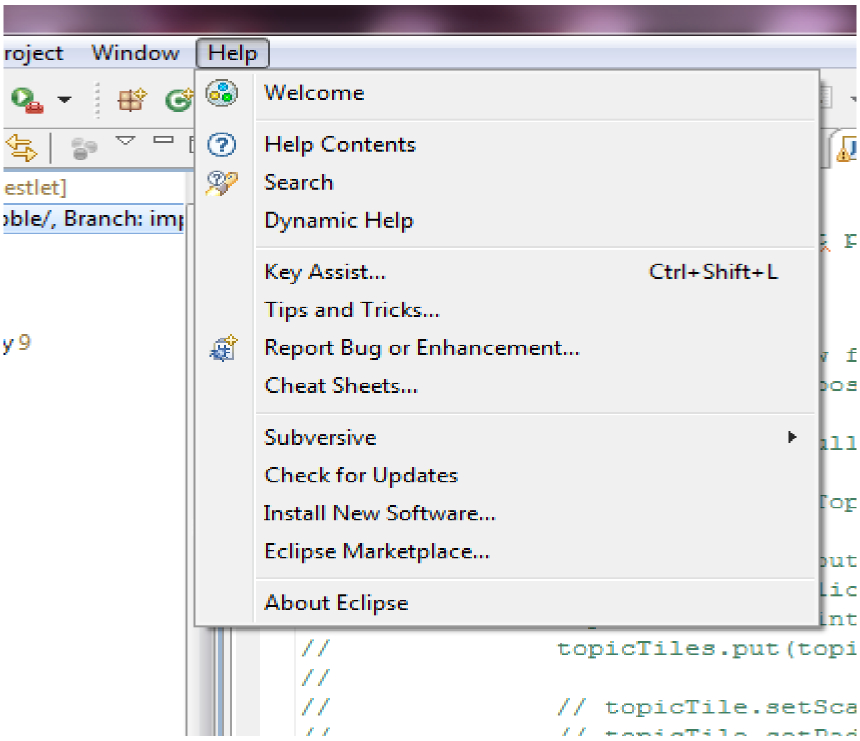
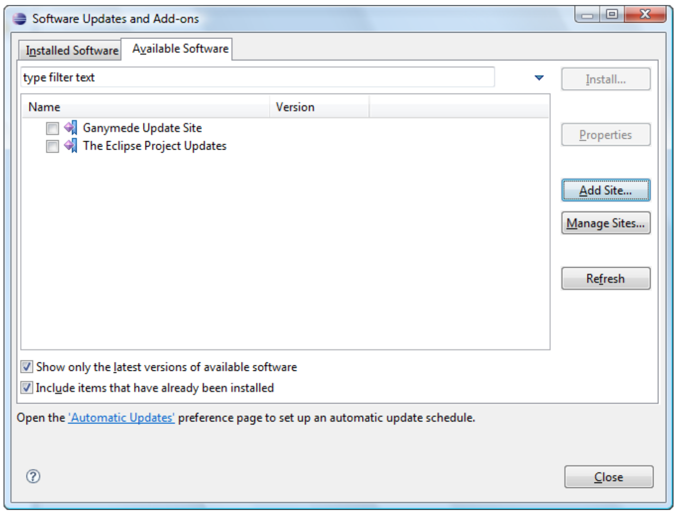


Figure 34. Installing SVN

2. Select the *'Available Software'* tab group and click the *'Add Site...'* button.



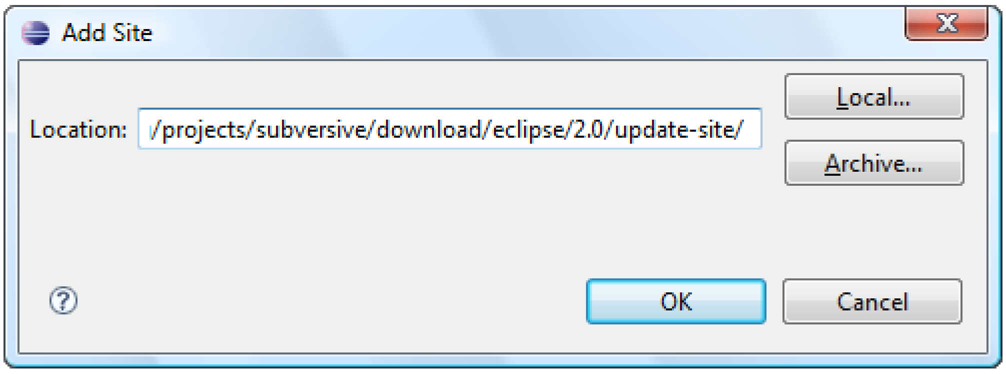
☑ Show only the latest versions of available software
☑ Include items that have already been installed

Open the 'Automatic Updates' preference page to set up an automatic update schedule.

Figure 35. Installing SVN

3. On the *'Add Site'* dialog enter the URL to SVN Connectors update site. The proper URL can be found on http://www.eclipse.org/subversive/downloads.php. Also the update-site for Subversive integration plug-ins can be added in the same way.

Click on the *'OK'* button to store update site information.



Location: /projects/subversive/download/eclipse/2.0/update-site/

Figure 36. Installing SVN

4. The Subversive update site can be found in 'Ganymede Update Site' in 'Collaboration Tools' category and provides following features list.

   Required feature, which should be installed unconditionally:

   a. **SVN Team Provider** - The Eclipse Team Provider for the Subversion version control system.

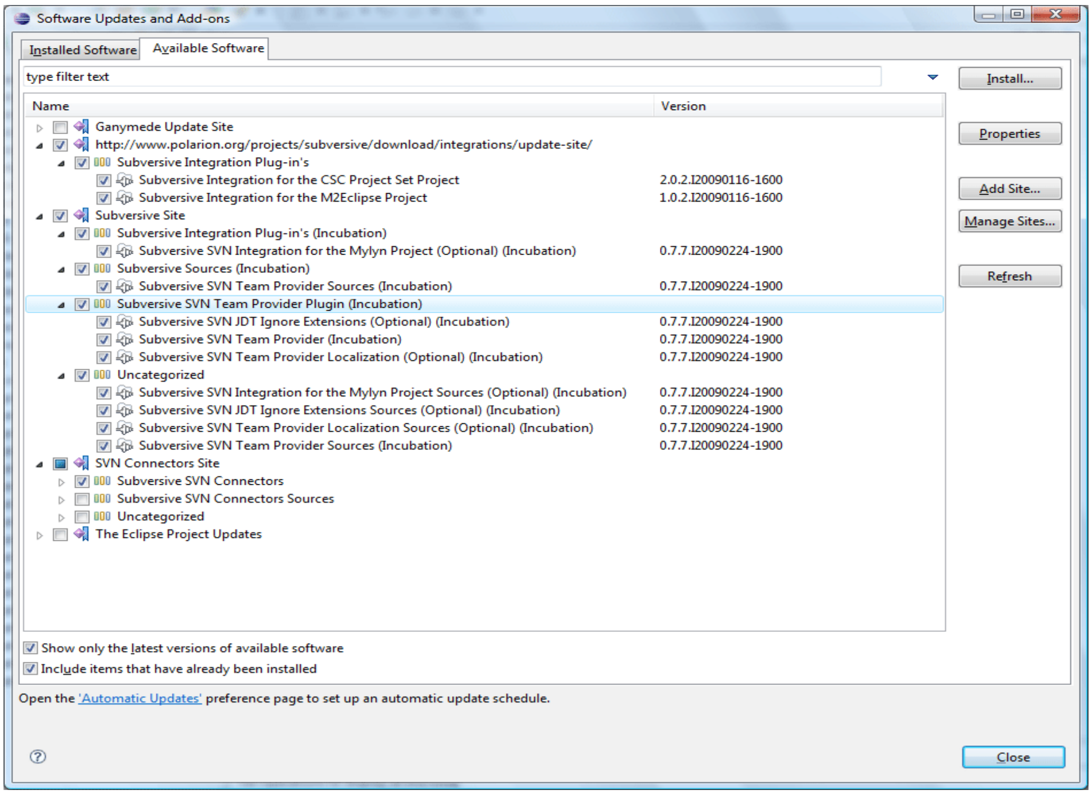
   Optional integrations with other plug-ins, which can be installed if you use these plugins and want to have them integrated with Subversive:

   b. **Subversive Integration for the Mylyn Project** - Integration with Mylyn.

   Other optional features:

   c. **JDT Ignore Extensions** - The feature is useful for Java development because it allows to interpret output folders as ignored resources automatically.
   d. **SVN Team Provider Sources** - The sources of the Eclipse Team Provider for Subversion.

Select SVN Team Provider, Subversive SVN Connectors and other features if required and click the *'Install'* button.


Software Updates and Add-ons

Installed Software | Available Software

type filter text

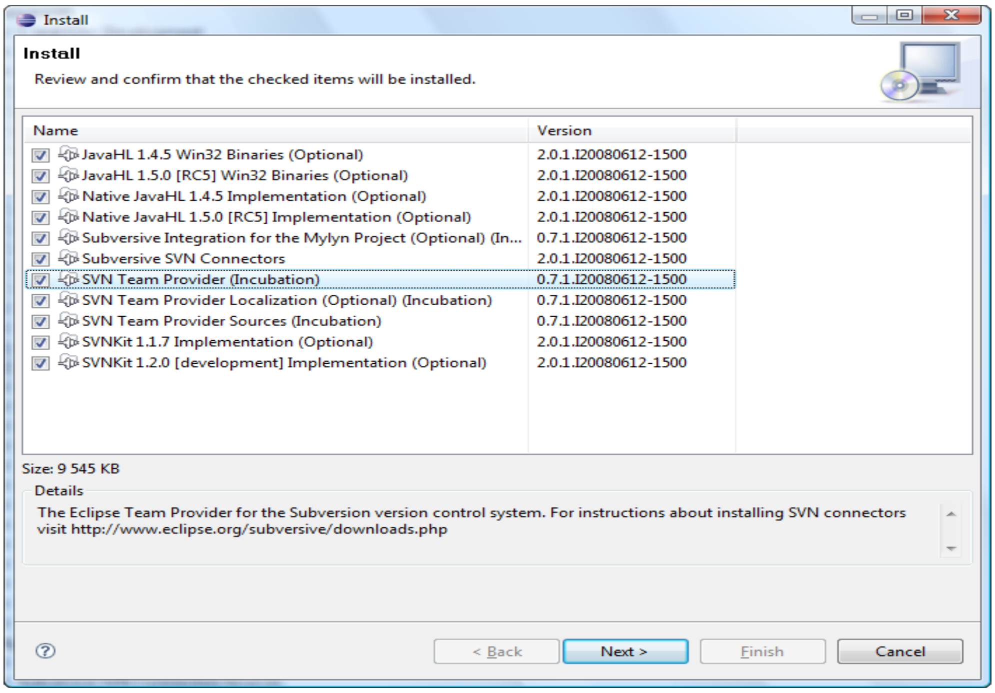
Name	Version
☐ Ganymede Update Site	
☑ http://www.polarion.org/projects/subversive/download/integrations/update-site/	
☑ Subversive Integration Plug-in's	
☑ Subversive Integration for the CSC Project Set Project	2.0.2.I20090116-1600
☑ Subversive Integration for the M2Eclipse Project	1.0.2.I20090116-1600
☑ Subversive Site	
☑ Subversive Integration Plug-in's (Incubation)	
☑ Subversive SVN Integration for the Mylyn Project (Optional) (Incubation)	0.7.7.I20090224-1900
☑ Subversive Sources (Incubation)	
☑ Subversive SVN Team Provider Sources (Incubation)	0.7.7.I20090224-1900
☑ Subversive SVN Team Provider Plugin (Incubation)	
☑ Subversive SVN JDT Ignore Extensions (Optional) (Incubation)	0.7.7.I20090224-1900
☑ Subversive SVN Team Provider (Incubation)	0.7.7.I20090224-1900
☑ Subversive SVN Team Provider Localization (Optional) (Incubation)	0.7.7.I20090224-1900
☑ Uncategorized	
☑ Subversive SVN Integration for the Mylyn Project Sources (Optional) (Incubation)	0.7.7.I20090224-1900
☑ Subversive SVN JDT Ignore Extensions Sources (Optional) (Incubation)	0.7.7.I20090224-1900
☑ Subversive SVN Team Provider Localization Sources (Optional) (Incubation)	0.7.7.I20090224-1900
☑ Subversive SVN Team Provider Sources (Incubation)	0.7.7.I20090224-1900
▣ SVN Connectors Site	
☑ Subversive SVN Connectors	
☐ Subversive SVN Connectors Sources	
☐ Uncategorized	
☐ The Eclipse Project Updates	

☑ Show only the latest versions of available software
☑ Include items that have already been installed

Open the 'Automatic Updates' preference page to set up an automatic update schedule.

Figure 37. Installing SVN

5. The update manager calculates dependencies and offers you a list of features to install. Select the needed ones and click the *'Next >'* button.


Install

Install

Review and confirm that the checked items will be installed.

Name	Version
☑ JavaHL 1.4.5 Win32 Binaries (Optional)	2.0.1.I20080612-1500
☑ JavaHL 1.5.0 [RC5] Win32 Binaries (Optional)	2.0.1.I20080612-1500
☑ Native JavaHL 1.4.5 Implementation (Optional)	2.0.1.I20080612-1500
☑ Native JavaHL 1.5.0 [RC5] Implementation (Optional)	2.0.1.I20080612-1500
☑ Subversive Integration for the Mylyn Project (Optional) (In...	0.7.1.I20080612-1500
☑ Subversive SVN Connectors	2.0.1.I20080612-1500
☑ SVN Team Provider (Incubation)	0.7.1.I20080612-1500
☑ SVN Team Provider Localization (Optional) (Incubation)	0.7.1.I20080612-1500
☑ SVN Team Provider Sources (Incubation)	0.7.1.I20080612-1500
☑ SVNKit 1.1.7 Implementation (Optional)	2.0.1.I20080612-1500
☑ SVNKit 1.2.0 [development] Implementation (Optional)	2.0.1.I20080612-1500

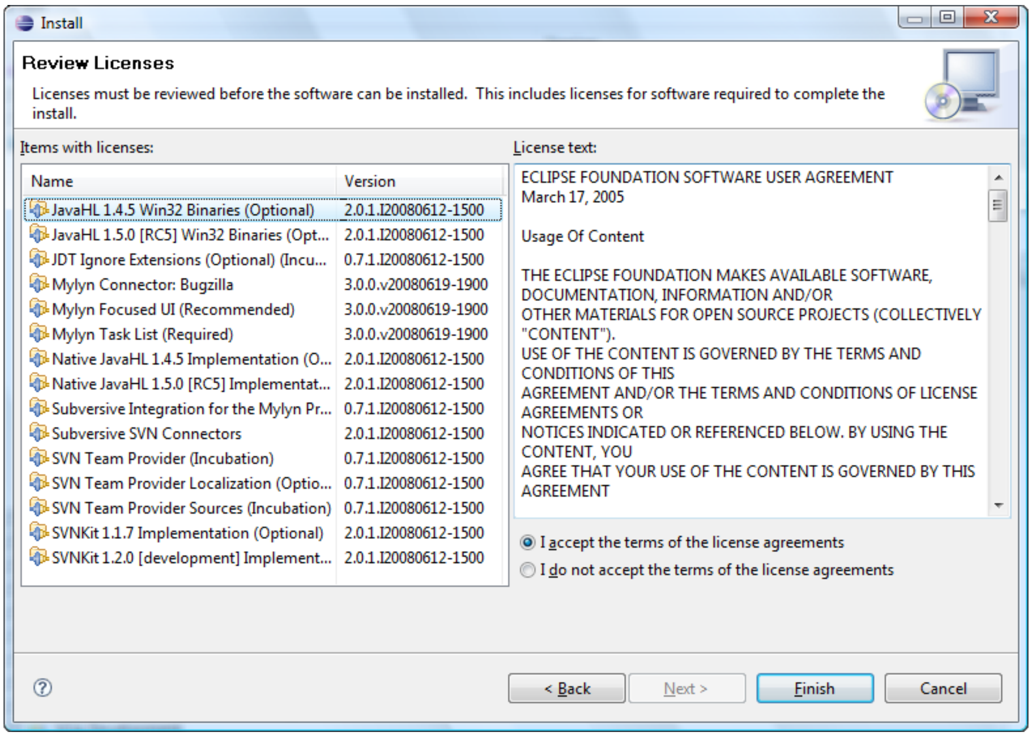
Size: 9 545 KB

Details

The Eclipse Team Provider for the Subversion version control system. For instructions about installing SVN connectors visit http://www.eclipse.org/subversive/downloads.php

Figure 38. Installing SVN

6. Accept terms of license agreement and click the *'Finish'* button in order to start the download of selected features.



Install

Review Licenses

Licenses must be reviewed before the software can be installed. This includes licenses for software required to complete the install.

Items with licenses:

Name	Version
JavaHL 1.4.5 Win32 Binaries (Optional)	2.0.1.I20080612-1500
JavaHL 1.5.0 [RC5] Win32 Binaries (Opt...	2.0.1.I20080612-1500
JDT Ignore Extensions (Optional) (Incu...	0.7.1.I20080612-1500
Mylyn Connector: Bugzilla	3.0.0.v20080619-1900
Mylyn Focused UI (Recommended)	3.0.0.v20080619-1900
Mylyn Task List (Required)	3.0.0.v20080619-1900
Native JavaHL 1.4.5 Implementation (O...	2.0.1.I20080612-1500
Native JavaHL 1.5.0 [RC5] Implementat...	2.0.1.I20080612-1500
Subversive Integration for the Mylyn Pr...	0.7.1.I20080612-1500
Subversive SVN Connectors	2.0.1.I20080612-1500
SVN Team Provider (Incubation)	0.7.1.I20080612-1500
SVN Team Provider Localization (Optio...	0.7.1.I20080612-1500
SVN Team Provider Sources (Incubation)	0.7.1.I20080612-1500
SVNKit 1.1.7 Implementation (Optional)	2.0.1.I20080612-1500
SVNKit 1.2.0 [development] Implement...	2.0.1.I20080612-1500

License text:

ECLIPSE FOUNDATION SOFTWARE USER AGREEMENT
March 17, 2005

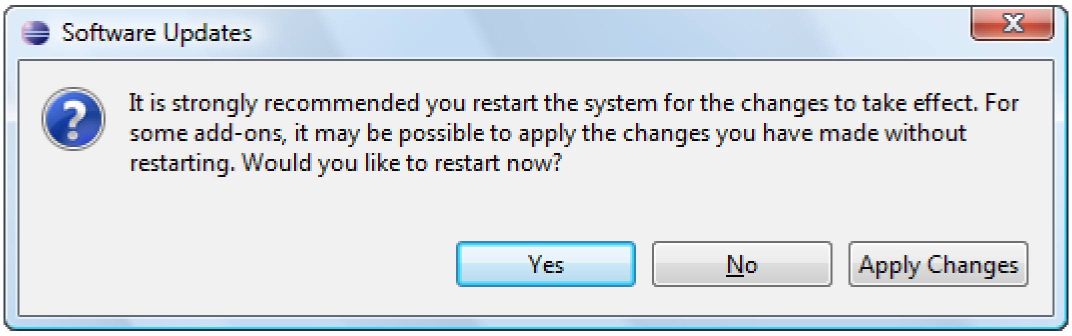
Usage Of Content

THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT

[✓ I accept the terms of the license agreements]
I do not accept the terms of the license agreements

Figure 39. Installing SVN

7. To apply installation changes and restart Eclipse click on the *'Yes'* button.



Software Updates

It is strongly recommended you restart the system for the changes to take effect. For some add-ons, it may be possible to apply the changes you have made without restarting. Would you like to restart now?

Figure 40. Installing SVN

8. After Eclipse restart you are able to start using Subversive.

9. You can find Subversive perspective and views in correspondent dialogs, activated by menu items *'Window > Open Perspective > Other...'* and *'Window > Show View > Other...'*.

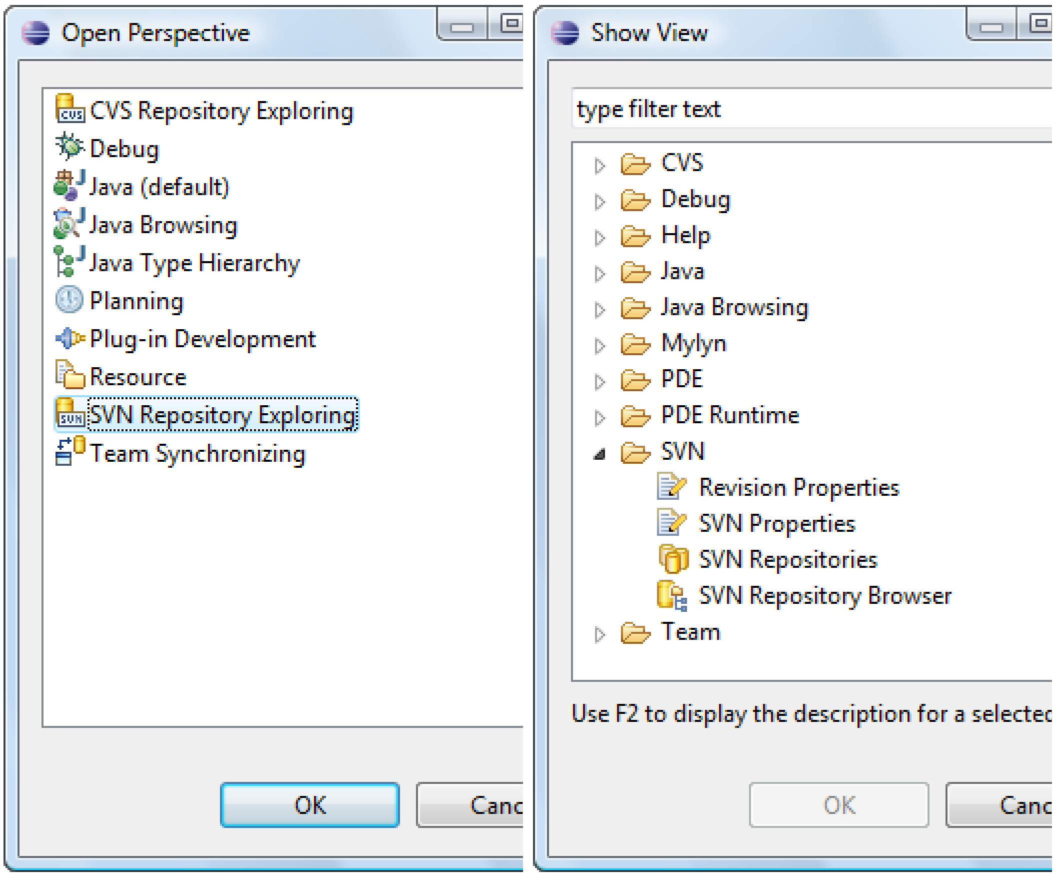


Figure 41. Installing SVN

C. QR Codes and ZXing

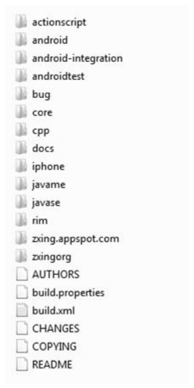
C.1 ZXing

ZXing is an open-source, multi-format 1D/2D barcode image processing library implemented in *Java*. This library focuses on using the built-in camera on mobile phones to photograph and decode barcodes on the device, without communicating with a server.

C.2 Getting Started

C.2.1 Download ZXing

Grab the latest estable distribution from `http://code.google.com/p/zxing/downloads/list`. After that, extract the contents of the `.zip` file in the root directory of the hard disk.

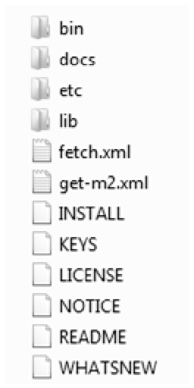


```
actionscript
android
android-integration
androidtest
bug
core
cpp
docs
iphone
javame
javase
rim
zxing.appspot.com
zxingorg
AUTHORS
build.properties
build.xml
CHANGES
COPYING
README
```

Figure 42 *Zxing* Library contents

C.2.2 Download Apache Ant

Apache Ant is a *Java* library and command-line tool which's mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of *Ant* is the build of *Java* applications. Download the last version from `http://ant.apache.org/bindownload.cgi` and extract the content.



```
bin
docs
etc
lib
fetch.xml
get-m2.xml
INSTALL
KEYS
LICENSE
NOTICE
README
WHATSNEW
```

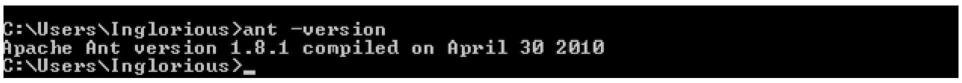
Figure 43 Apache Ant contents

To install the Apache Ant is necessary to update the environment variables. This step is similar to the step shown in B.2.5. Thus, depending on Operating System used, the developer may add the `..\apache-ant-1.8.1\bin` to an appropriate path.

To verify the environment variables are correctly updated, from a console window run the following command:

```
ant –version
```

If it displays the version information as the next figure



```
C:\Users\Inglorious>ant -version
Apache Ant version 1.8.1 compiled on April 30 2010
C:\Users\Inglorious>_
```

Figure 44 Checking the installation of Ant

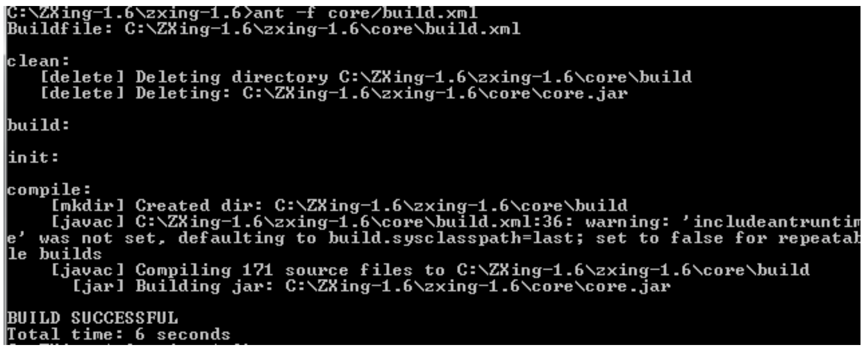
Ant is correctly configurated and installed.

C.2.3 Build

The code lives in several subdirectories like it is shown in Figure 27, corresponding to the various subcomponents, like "*core*" and "*javase*".

Within each, there is a `build.xml` *Ant* build file which controls building of that component. As *Apache*'s *`Ant`* tool is already present on the system, simply type `ant -f core/build.xml` in the *ZXing* directory to build that component.

To build everything, simply use the `build.xml` file found in the top-level directory, above all components.



```
C:\ZXing-1.6\zxing-1.6>ant -f core/build.xml
Buildfile: C:\ZXing-1.6\zxing-1.6\core\build.xml

clean:
   [delete] Deleting directory C:\ZXing-1.6\zxing-1.6\core\build
   [delete] Deleting: C:\ZXing-1.6\zxing-1.6\core\core.jar

build:

init:

compile:
    [mkdir] Created dir: C:\ZXing-1.6\zxing-1.6\core\build
    [javac] C:\ZXing-1.6\zxing-1.6\core\build.xml:36: warning: 'includeantruntim
e' was not set, defaulting to build.sysclasspath=last; set to false for repeatab
le builds
    [javac] Compiling 171 source files to C:\ZXing-1.6\zxing-1.6\core\build
      [jar] Building jar: C:\ZXing-1.6\zxing-1.6\core\core.jar

BUILD SUCCESSFUL
Total time: 6 seconds
```

Figure 45 Building ZXing

C.2.4 Integrate ZXing source lib into the Android Code project through Eclipse

In *Eclipse* right-click the Project Folder → *Properties* → *Java Build Path* → Libraries → Add External JARs →Navigate to the *ZXing* folder and open the core directory. Select *core.jar*.

Finally, it is necessary to correct a few errors in the translations and the `AndroidManifest.xml` file. After that, it is possible to compile, and the developer will have a working standalone barcode scanner application based on the *ZXing* source.

C.3 QR Codes

C.3.1 Introduction

QR code is a kind of 2D (two-dimensional) code. The origin of these codes started with the bar codes. Reading speed and precision are some of the advantages which make the bar codes so popular.
When bar codes became popular and universally recognized, the market started to demand better characteristics such as storing more information, more character types or printing in smaller spaces. The initial solution to get these requirements was the increasing of the number of bar codes digits or lying out multiple bar codes. However, these improvements caused problems such as enlarging the bar code area, complicating reading operations and increasing painting cost. 2D code emerged to solve these problems and later they also progressed from the stacked bar code method (that stack bar codes), to increase information destiny matrix method. QR code is one of these matrix codes.

C.3.2 What is a QR code

The quick response (QR) Code is 2D (two dimensional) Code developed by Japanese corporation Denso Wave (a division of Denso Corporation at the time) and released in 1994 with the primary aim of being a symbol that is easily interpreted by scanner.

QR code contains information in both the vertical and horizontal directions, whereas a bar code contains data only in one direction. QR code holds a considerably greater volume of information than a bar code.

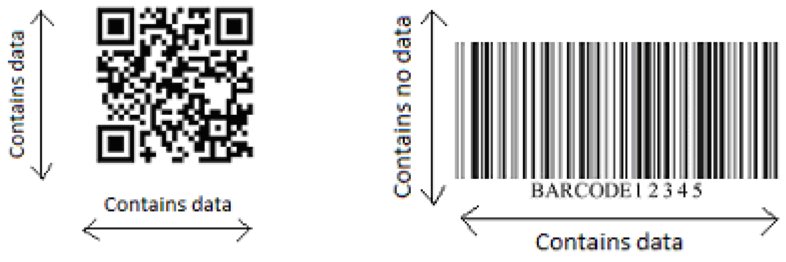


Figure 46. Bar codes

C.3.3 Properties

The following features of the QR codes show the advantages of this kind of codes compared with conventional bar codes.

- *High Capacity Encoding of Data*

QR code can store several dozen to several hundred times more information than conventional bar codes which are capable of storing a maximum of approximately 20 digits.
Besides QR code can handle all types of data, such as numeric and alphabetic characters, Kanji, Kana, Hiragana, symbols, binary and control codes. Up to 7,089 characters can be encoded in one symbol. The next table show the data capacity for the different kinds of data.

QR Code Data Capacity	
Numeric only	Max. 7,089 characters
Alphanumeric	Max. 4,296 characters
Binary (8 bits)	Max. 2,953 characters
Kanji, full width Kana	Max. 1,817 characters

Tabla 1: QR code Data Capacity

A QR code symbol with the size showed in this figure can encode 300 alphanumeric characters.

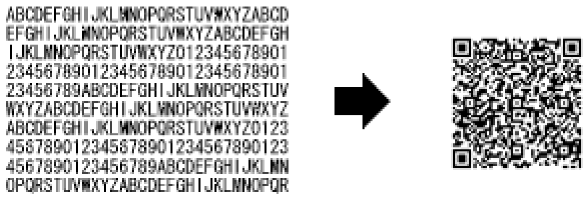

ABCDEFGHIJKLMNOPQRSTUVWXYZABCD
EFGHIJKLMNOPQRSTUVWXYZABCDEFGH
IJKLMNOPQRSTUVWXYZ012345678901
234567890123456789012345678901
23456789ABCDEFGHIJKLMNOPQRSTUV
WXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123
456789012345678901234567890123
4567890123456789ABCDEFGHIJKLMN
OPQRSTUVWXYZABCDEFGHIJKLMNOPQR

Figure 47. High capacity ancoding of data

- *Small Printout size*

While bar codes carry information only horizontally or vertically, QR codes carry information in both dimensions. QR code is capable of encoding the same amount of data in approximately one-tenth the space of the traditional bar code.

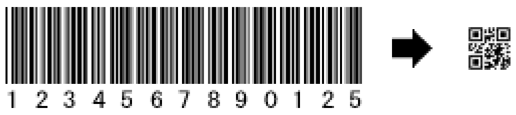


Figure 48. Small printout size

There is also Micro QR code for a smaller printout size, this kind of QR code will be explained later.

- *Kanji and Kana Capability*
As a symbology developed in Japan, QR code is capable of encoding JIS Level 1 and Level 2 Kanji character set.
In case oj Japanese, one full-with Kana or Kanji character is efficiently encoded in 13 bits, allowing QR code to hold more than 20% data than other 2D symbologies.

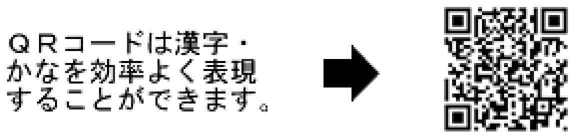


Figure 49. Kanji and Kana Capability

- *Dirt and damage resistant*

QR code has error correction capability. Data can be resolved even if the symbol is partially dirty or damaged. A maximum 30% of codewords can be restored; it depends on the amount of dirt and damage.

A codeword is the unit of data area, in case of QR code is 8 bits.

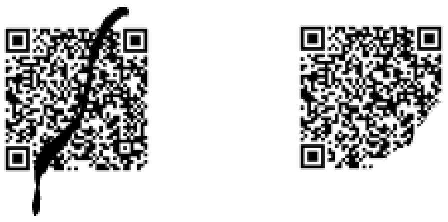


Figure 50. Damage and dirty resistant

- *Readable from any direction in 360º*

QR code uses position detection patterns located t the three corners to be able to read the code from any direction in 360 degree (omni-directional). These positions detection patterns guarantee stable high-speed reading, circumventing the negative effects of background interference.

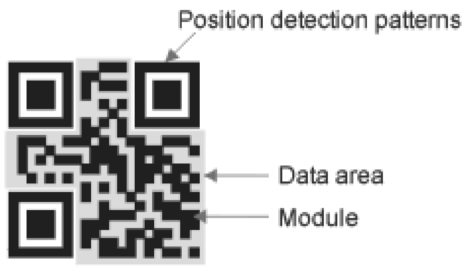


Figure 51. Detection patterns

- *Structured append feature*

QR code can be divided into multiple data areas, so information stored in multiple QR code symbols can be reconstructed as single data symbols.
One data symbol can be divided into up to 16 symbols, allowing printing in a narrow area.
The same data can be read either from the upper symbol or the lower four symbols.

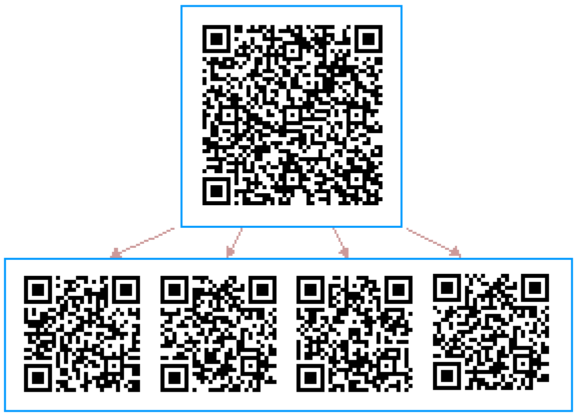


Figure 52. Structure append feature

- *Resistant to Distorted Symbols*

Symbols often get distorted when attached onto a curved surface or by the reader being tilted (angled between the CCD sensor face and the symbol face). To correct this distortion, QR Code has alignment patterns arranged with a regular interval within the range of the symbol. The variance between the centre position of the alignment pattern estimated from the outer shape of the symbol and the actual centre position of the alignment pattern will be calculated to have the mappings (for identifying the centre position of each cell) corrected. This will make the distorted linear/non-linear symbols readable.

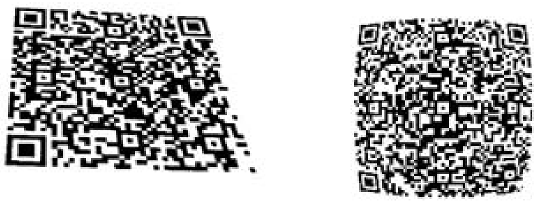


Figure 53. Distored QR codes

- *Masking Process*

By having special patterns to process masking, QR Code is enabled to have black and white cells well arranged in a balanced order. To accurately finalize the data that had been read, it is necessary to arrange the white and black cells in a well-balanced manner. To enable this, EX-OR calculation will be implemented between the data area cell and the mask pattern (template) cell when encoding the stored data and arranging it into the data area. Then, the number of unique patterns existing and the balance between the white cells and the black cells will be assessed against the data area where the calculation had been implemented. There are eight mask patterns. Assessment will be made for each mask pattern, and the mask pattern with the highest assessment result together with the EX-OR calculation result will be stored into the data area.

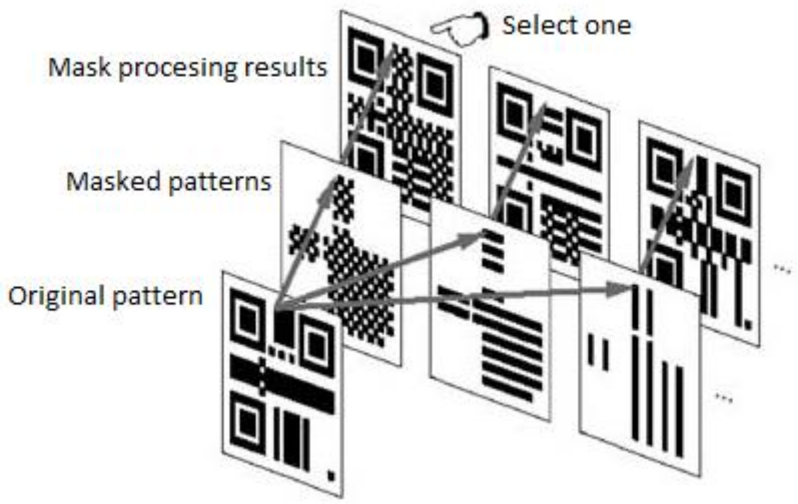


Figure 54. Masking process

- *The Confidentiality of the Code*

By making the relationship between the character type and the stored data unique for a special usage, QR Code can be easily encrypted. Unless the conversion table between the character type and the stored data is deciphered, no one will be able to read the QR Code.

- *Direct Marking*

QR Code exerts superior readability even for symbols which are directly marked using laser or dot pin markers. For directly marked symbols, the cell shape does not necessarily have to be square as shown in Figure 10. It can also be circular shape. Even if the white part (with high reflectance) and the black part (with low reflectance) are inverted due to the angle of the illuminating ray, the code can still be read in an accurate manner. It is also possible to read from the back side of the symbol when it is marked upon a transparent material such as glass, etc.

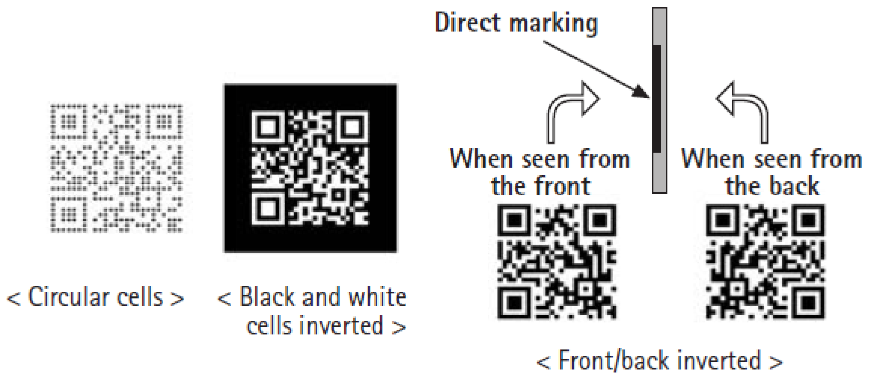


Figure 55. Direct marking

C.3.4 QR code system

A QR code printer (or QR code creation software) and QR code scanner establish the QR code system. QR code must be generated with QR code creation software and a special printer and must be read with the QR code scanner more suitable for the application (for example a cellular phone can be better reader).
The code generated must be properly sized for being read correctly, so the size (area) of the code is important. Size decision factors can be read below.

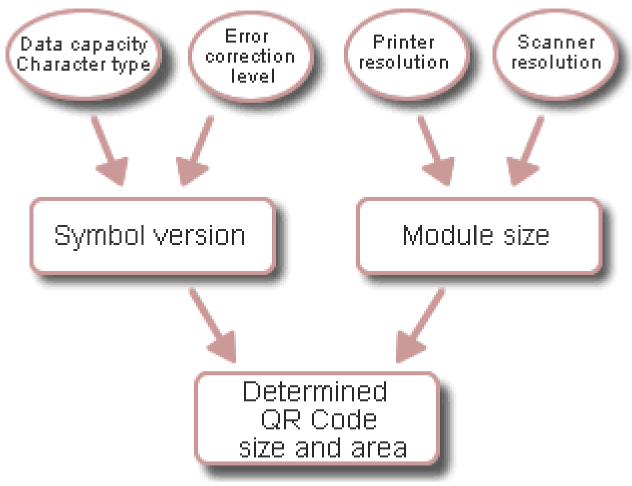


Figure 56. Size decisión factors

- *Symbol version*

The symbol versions of QR Code range from Version 1 to Version 40. Each version has a different module configuration; this is the number of modules contained in a symbol. A module is the black and white dots that make up the QR code.

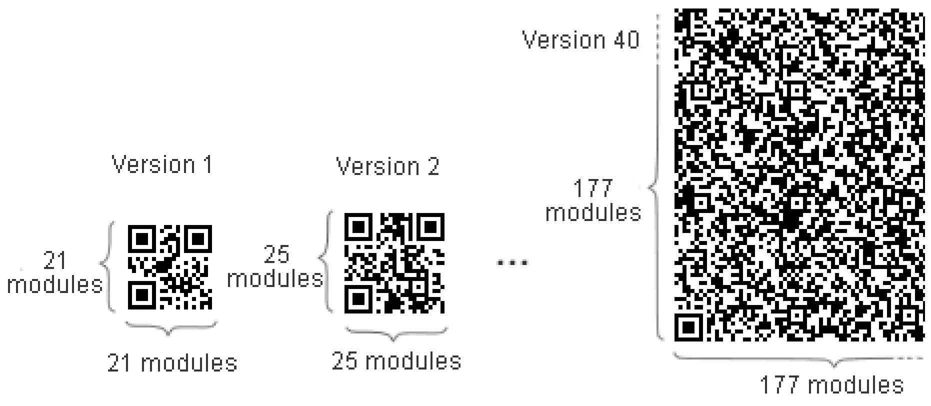


Figure 57. Number of modules in the different versions

Each QR Code symbol version has the maximum data capacity according to the amount of data, character type and error correction level.
In other words, as the amount of data increases, more modules are required to comprise QR Code, resulting in larger QR Code symbols.

There are four level of error correction for each version; these levels are showed in the next table.

QR Code Error Correction Capability	
Level L	Approx.7%
Level M	Approx. 15%
Level Q	Approx. 25%
Level H	Approx. 30%

Table 2: QR code corrections levels

Error Correction has the capability to restore data if the code is dirty or damaged. Four error correction levels are available for users to choose according to the operating environment. Raising this level improves error correction capability but also increases the amount of data QR Code size.

To select error correction level, various factors such as the operating environment and QR Code size need to be considered.
Level Q or H may be selected for factory environment where QR Code get dirty, whereas Level L may be selected for clean environment with the large amount of data. Typically, Level M (15%) is most frequently selected.

The QR Code error correction feature is implemented by adding a *Reed-Solomon Code* to the original data. QR code therefore represents its error correction rate as a ratio of the total codewords (codewords of QR code an Reed-Solomon code).

- *Module size*

Once a symbol version is determined, the actual size of the QR Code symbol depends on the millimetre size of the module (one square area comprising QR code) to be printed.

The larger the module is the more stable and easier to read with a QR code scanner it becomes. On the other hand, as the QR Code symbol size gets larger, a larger printing area is required.
It is, therefore, necessary to determine the module size of each application after considering all the relevant factors. It is recommended that QR Code symbols be printed as large as possible within the available printing area.

The figure below shows how the size of the printing area increases when the module is larger.

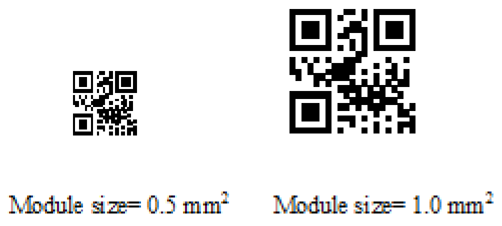


Figure 58. Versión 1 QR Code (21x21 modules)

The module size must be chosen according to the properties of the printer and the scanner.
Increasing the number of dots improves printing quality, eliminates printing width or paper feed speed fluctuations, distortion of axis, blurring, etc., and enables more stable operations.

It is recommended for stable operations that each module is made up of 4 or more dots.

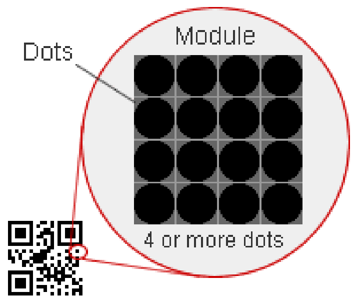


Figure 59. Dots of the module

The scanner has its own readable module size limit represented with the scanner resolution. The scanner resolution must be less than the module size.

The size of the QR code symbol is determined with the symbol version and module size, but the QR code symbol area requires a margin or “quite zone” around it to be used.

The margin is a clear area around a symbol where nothing is printed. QR code requires a four-module wide margin at all sides of a symbol.

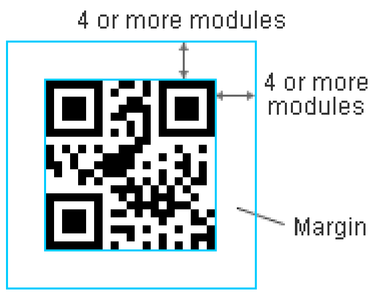


Figure 60. Margen of the QR codes

There is a very small QR code called Micro QR code, which is used in applications that require a smaller space and use smaller amounts of data, such as ID of printed circuit boards and electronic parts, etc.

The configuration of Micro QR code allows painting in areas even smaller than QR code.

The most important characteristic which increases the efficiency of data encoding is that in Micro QR code there is only one position detection pattern, compared with regular QR code that require a certain amount of area because position detection patterns are located at the corners of a symbol.

Besides, the Micro QR code requires less wide margin around a symbol, it needs only two-modules instead of four-module of wide margin required in regular QR code.

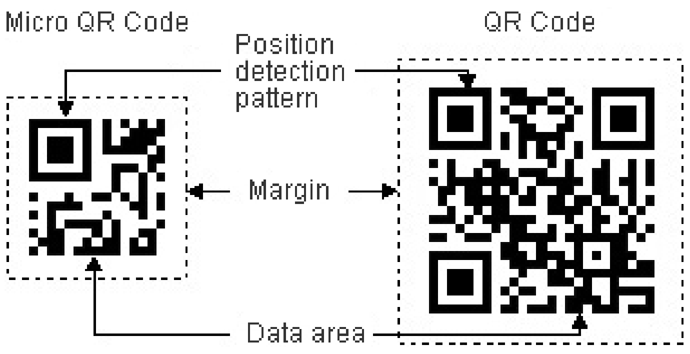


Figure 61. Micro QR code

Micro QR Code is not capable of storing much data (maximum 35 numeric characters).

However, because it can store data for each symbol size more efficiently than QR Code, the size of Micro QR Code symbols does not significantly increase, even though the amount of data is increased (see the figure below).

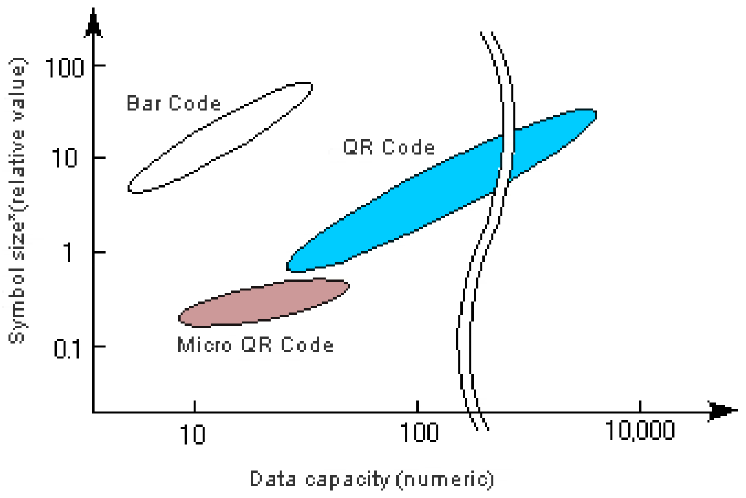
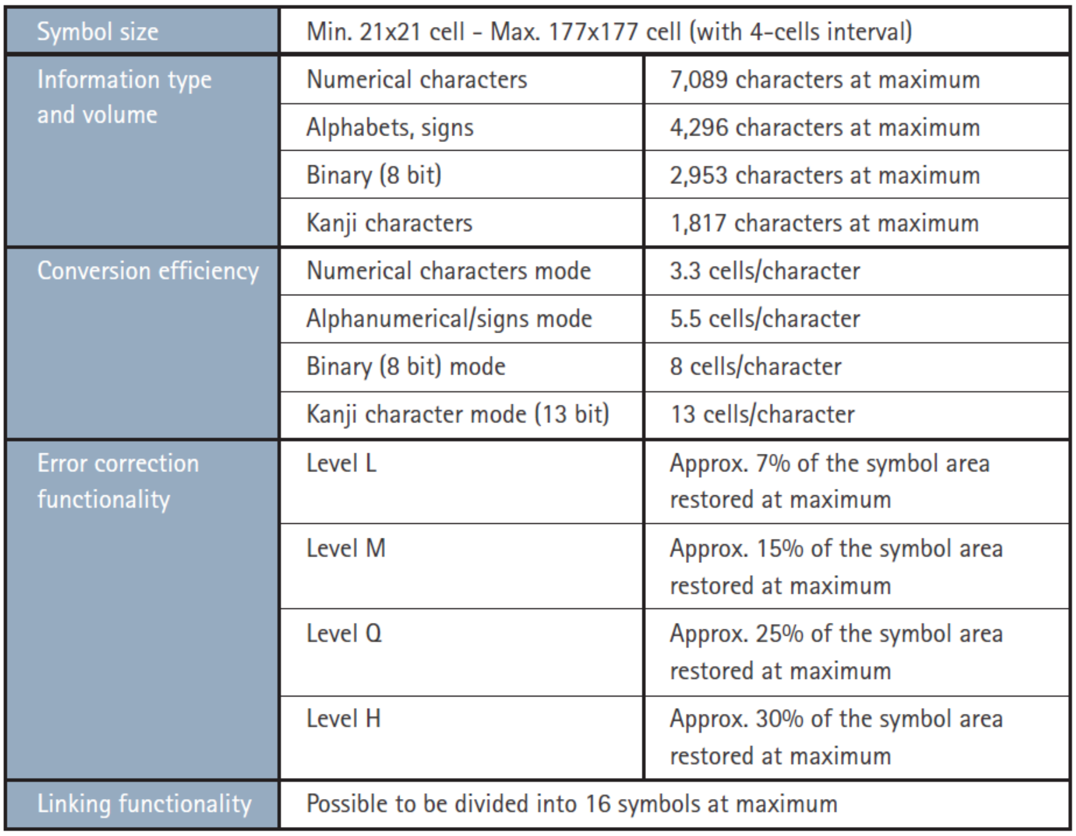


Figure 62. Maximum data capacity of Micro QR code depending on symbol version

C.2.5 The specifications of QR codes

The specifications of the QR Code are as described in Table 3.



Symbol size	Min. 21x21 cell - Max. 177x177 cell (with 4-cells interval)	
Information type and volume	Numerical characters	7,089 characters at maximum
	Alphabets, signs	4,296 characters at maximum
	Binary (8 bit)	2,953 characters at maximum
	Kanji characters	1,817 characters at maximum
Conversion efficiency	Numerical characters mode	3.3 cells/character
	Alphanumerical/signs mode	5.5 cells/character
	Binary (8 bit) mode	8 cells/character
	Kanji character mode (13 bit)	13 cells/character
Error correction functionality	Level L	Approx. 7% of the symbol area restored at maximum
	Level M	Approx. 15% of the symbol area restored at maximum
	Level Q	Approx. 25% of the symbol area restored at maximum
	Level H	Approx. 30% of the symbol area restored at maximum
Linking functionality	Possible to be divided into 16 symbols at maximum	

Table 3: Specifications of QR codes

- *Symbol Size*

QR Code can have its size freely selected according to the data volume to be stored and the reading method. The symbol size is incremented by four cells in both vertical and horizontal direction - 21x21 cells, 25x25 cells,29x29 cells..., and there are 40 size types with the maximum size set to 177x177 cells.

For example, in the case for 45x45 cells, if a single square cell is sized 0.25mm, one side of the symbol will be 45x0.25mm = 11.25mm. The quiet zone will need to be added on both sides of the symbol whose minimum size is four cells, and therefore, the space required for having this symbol printed will be a square of (4+45+4)x0.25mm which is 13.25mm.

- *Information Type and Volume*

QR Code can handle various types of data such as numerical characters, alphabets, signs, Kanji characters, Hiragana, Katakana, control signs, and images. It can basically have character sets supported by ISO/IEC 646 and ISO/IEC 10646.
These data can also coexist. The maximum available volume of the information is listed in Table 3.
- *Data Conversion Efficiency*

QR Code has four types of conversion mode numerical characters, alphanumerical/signs, binary, and Kanji characters for encoding the data. Each mode has had considerations to improve its conversion efficiency. The number of cells required for each character in each mode is listed in Table 3.

- *Error Correction Functionality*

QR Code has an error correction functionality for restoring the data. There are four different restoration levels so that you can select the level that matches with each usage environment. Each restoration capability is as listed in Table 3.