MASTER THESIS

TITLE: Real Time Commercial on TV Detector

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Antonio Ibáñez Luján

DIRECTOR: Roc Meseguer Pallares

DATE: March 30th 2011
Títol: Real Time Commercial on TV Detector

Autor: Antonio Ibáñez Luján

Director: Roc Meseguer Pallares

Data: 30 de Març de 2011

Resum

El propòsit d’aquesta memòria es tractar els principals aspectes dels sistemes de comunicació d’un prototip que s’ha desenvolupat amb l’objectiu de detectar la finalització d’emissió d’anuncis publicitaris a la televisió, avisant als usuaris interessats en aquests esdeveniments.

Aquest ha estat realitzat pels desenvolupadors des de zero, motiu pel qual en els primers apartats del document s’abordaran tòpics relacionats amb anàlisi, requeriments i disseny del projecte.

Per tal de poder desenvolupar la part de comunicació, complimentant els requisits que s’exposaran en l’apartat corresponent, s’ha escollit i adaptat una extensió del protocol XMPP (Extensible Messaging and Presence Protocol). Aquest permet poder dur a terme connexions de tipus push entre client i servidor.

Amb l’objectiu de testejar el correcte funcionament del sistema, i garantir als usuaris una interfície amena, s’ha desenvolupat una aplicació client en java per a ordinadors personals.

Finalment, s’ha realitzat una descripció de la posada a punt del prototip, i de les configuracions de les diferents màquines que es van tenir que dur a terme, principalment servidors OpenFire, per tractar el protocol XMPP, a més d’Apache Tomcat per tal de realitzar connectors. Cal destacar que tots els components de software que s’han fet servir son de tipus Open Source.

En aquesta memòria no es troba cap tipus de referència a qüestions relacionades amb tractament o processament d’imatges, així com tampoc conceptes sobre DVB-T. Aquests es poden trobar dins la memòria, amb mateix nom, redactada per Jordi Coscolla Álvarez a la FIB (Facultat d’Informàtica de Barcelona).
Overview

The aim of this report is to deal with the main important topics of the communication system of a prototype developed in order to detect when commercials on television finishes, and warn users interested in this kind of events.

This project has been totally assembled de novo by developers, and for this reason the first sections of this report are about analysis, requirements and design of the system.

In order to develop the communication part filling all the requirements, which are going to be explained, an extension of XMPP (Extensible Messaging and Presence Protocol) protocol was chosen and adapted. It would provide us push-type connections between client and server side.

With the aim of testing the proper functioning of the system, and ensure the first pleasant interface to users, a visual client application in Java has been also developed.

Finally, there is a description of the tuning process of the prototype: configurations of different machines that have been set up, mainly OpenFire (XMPP server) and Apache Tomcat (used to create plugings and connectors). Note that all software components used are Open Source type.

Inside this report you will not find any reference to topics related to image treatment or processing, neither DVB-T concepts. They can be found in a report with the same name, written by Jordi Coscolla Álvarez, at FIB (Facultat Informàtica Barcelona).
INDEX

CHAPTER 1. PROJECT OUTLINE................................................................................. 1
  1.1. Introduction .................................................................................................................. 1
  1.2. Personal motivations ..................................................................................................... 1
  1.3. Has been it developed yet? .......................................................................................... 2
  1.4. Is it the right moment to starting this project? ............................................................ 3
  1.5. Technical goals ............................................................................................................. 4
  1.6. Timing and project organization .................................................................................. 5

CHAPTER 2. TECHNOLOGIES............................................................................. 7
  2.1. XMPP ............................................................................................................................ 8
    2.1.1. XMPP extensions .................................................................................................. 9
    2.1.2. XMPP Server: OpenFire .................................................................................... 10
  2.2. Layer connectors: Java Servlets .................................................................................. 11
  2.3. Data base: PostgreSQL .............................................................................................. 13
  2.4. Desktop program: language and XMPP library .......................................................... 14

CHAPTER 3. SYSTEM DESIGN AND ARCHITECTURE ................................ 15
  3.1. System requirements .................................................................................................. 15
    3.1.1. System functional requirements ........................................................................ 15
    3.1.2. System non-functional requirements .................................................................. 16
  3.2. Architecture definition ............................................................................................... 17
    3.2.1. Architecture layers ............................................................................................ 17
    3.2.2. Client Layer ....................................................................................................... 18
    3.2.3. Communication Layer ..................................................................................... 19
    3.2.4. Detection Layer .................................................................................................. 20

CHAPTER 4. MESSAGE SERVICE DEPLOYMENT ...................................... 21
  4.1. Openfire Server .......................................................................................................... 21
    4.1.1. Room user's configuration ................................................................................. 21
    4.1.2. Room configuration ............................................................................................ 23
  4.2. System messages ........................................................................................................ 24
    4.2.1. TV program / commercial broadcasting detected ............................................... 24
    4.2.2. User register and login ...................................................................................... 26
    4.2.3. User request ....................................................................................................... 27
    4.2.4. User error reports ............................................................................................. 27
    4.2.5. Kicking users ..................................................................................................... 28
4.3. Database System ................................................................. 29
  4.3.1. User information ............................................................... 29
  4.3.2. Channel information ...................................................... 30

CHAPTER 5. DESKTOP CLIENT DEPLOYMENT ......................... 31
  5.1. Functional schema ............................................................... 31
  5.2. Sequence schema ............................................................... 32
  5.3. Developing technologies ................................................... 33
    5.3.1. Libraries ................................................................. 33
    5.3.2. Installations platforms ................................................ 34
  5.4. The result ................................................................. 35

CHAPTER 6. PROTOTYPE INTEGRATION ENVIRONMENT ............... 37
  6.1. Environment description .................................................. 37
  6.2. Configuration integration environment ............................... 37
  6.3. Evaluating integration environment ................................... 39
    6.3.1. Response time test ................................................... 39
    6.3.2. User feeling test ....................................................... 41
    6.3.3. Are we really pushing? .............................................. 42

CHAPTER 7. CONCLUSIONS AND FUTURE LINES ...................... 45
  7.1. Technical conclusions ...................................................... 45
  7.2. Personal conclusions ....................................................... 46
  7.3. Project feedback ............................................................. 47
  7.4. Environmental project impact ............................................ 48
  7.5. Future lines ............................................................... 49

BIBLIOGRAPHY ................................................................. 51
CHAPTER 1. PROJECT OUTLINE

In this chapter we are introducing and explaining the main project motivations which have prove us to develop this thesis, as well as the main objectives and goals that we want to achieve.

1.1. Introduction

The 1\textsuperscript{st} of July of 1941, the first television advertisement was broadcasted [1]. Bulova, an enterprise that carried out the assembly of clocks and watches, paid 4 US dollars for 10 seconds of a spot. The commercial was reproduced before a baseball game between the Brooklyn Dodgers and Philadelphia Phillies. This spot showed a simple image: a superimposed clock was displayed on a United States of America map. This event, in spite of the rudimentary way it was developed, supposed an important landmark.

Nowadays, in the capitalism society where we are living, TV advertisements are considered as the most important and effective format in order to reach high volumes of future customers. This fact has revolutionized television broadcasting, as well as the costs of emitting a commercial: per 30 seconds of TV spot, advertisers have paid around 2.7 million of North American dollars [2], without taking into account the costs of assembling it. Special effects technology, the most popular personalities like actors, politicians or sportsmen, for example, have been invested in a large number of advertisements, with a result of a high-quality spots.

But, although the important amounts of money that enterprises are spending in inventing new advertisements and the popularity reached for some of them, there is an annoyance feeling among viewers caused by different factors: Some spots set volume higher as broadcasting levels, others are emitted loads and loads of times, the duration of some advertisement intervals are too large, and other objections that most of common people could comment us.

Here is where our project starts: What about skipping automatically advertisements on television? Is it possible to fight against this commercial phenomenon? The answer is, definitively, yes. In the report, which we are dealing with, the communication part of a Real Time Commercial on TV Detector system is going to be developed, as well as other aspects directly related with it.

1.2. Personal motivations

I strongly believe in the project that we have developed in order to get this final thesis ready. Before starting it, I thank about others ideas, but finally I decided to start this new adventure with Jordi Coscolla, a recent Computer Science Engineer and, above all, a one of my best friends since, almost, ten years.
Project outline

The first reason is that last year, as Jordi as me, were at the same academic point: both with all the subjects of our degree finished and pending to the development of our final thesis. One evening we started to talk about doing something together and, after a beer meeting (with brainstorming included) we decided to develop a Real Time Commercial on TV Detector in order to warn users about when commercials are over.

“Everybody needs one!” we repeated again and again. It seemed to us a useful idea and, after a year, we even thing it. Develop and assembly something with direct application on common life was essential for us.

Another important consideration is that this idea seems to us a good personal challenge, and we start to think about it. When explaining this occurrence to other people, they answered to us that it was impossible: no way, if it is not invented means that it is impossible. Nowadays, and after having the project running in a prototype version, there are some of them than maintains this position. For us it was an extra motivation point, but not necessary: we did not want to demonstrate nothing to others, we wanted to demonstrate our learned engineer abilities to us.

After a previous analysis of the project, we started to think about the fact of combining two different engineering sciences and the results that in a near future we would obtain: Is it a good idea to mesh computer and telecommunications sciences. Yes, in fact it has result in a perfect combination to deploy this thesis. Our personal and technical knowledge was different, but complementary. We had the possibility of sharing concepts and discussing new ideas from different points of view. The critic thought was ensured.

These are the main motivations of the project. I could write further than the ones that we have mention in the last paragraphs, but in fact, I am able to summing up it in one: Jordi Coscolla is one of the most important persons that I have meet in my personal and academic life, in spite studying different degrees. I admire him as for his spectacular engineer abilities as well as his personal manner of treat people. I am sure that, with the essential good luck that is needed and the way of working he has acquired, in few years he is going to be a reference to follow. I have the chance of learn with him, and I’m not going to trash it.

1.3. Has been it developed yet?

Before starting this project, we have also done an intensive research (Annex A) about similar initiatives. It was a very interesting point because for us was very important do something different, and feel the innovation spirit inside. In this subsection we are going to enumerate some related projects that differ on the one we are dealing with. It is important to mention that this topic (commercial on TV detection) has been deeply and widely studied.

In “RealTime Commercial Detection Using MPEG Features” [15] paper, the authors present different algorithms that achieve a precision of 95% when
detecting this kind of program changes. They have applied this developed technology in order to obtain a PVR (*Personal Video Recorder*) skipping commercials.

The same matter with “A fast method for animated TV logo detection” [16]. In this paper authors try also to detect the status of a TV channel in real-time mode. They have done it in order to get the same application than the last paper: PVRs.

“A Confidence Based Recognition System for TV Commercial Extraction” [17] paper show a different vision: the authors are not interested on the status of a channel, they are not interested whether a TV station is broadcasting commercials or not. What they want is to recognize some patterns of well-known commercials and list the number of times that it is repeated. In other words, they are enterprise oriented in order to ensure that the SLA of commercials within advisers and TV stations is accomplished. The same idea is presented by “Local Color Analysis for Scene Break Detection” [18].

Summing up, we have to point out that there are a huge quantity of papers and literature about how to detect when commercials are broadcasted on TV and when they finish, defining intervals in order to erase them from PVR or getting a number of times that a certain commercial has appeared. But it is important to mention that we do not have found any system that sends different traps and warnings to users in order to permit skipping them in real-time. For this reason, we have to mention that the project that we have developed would be different to the rest.

**1.4. Is it the right moment to starting this project?**

After gathering together all the information of the previous research done in Annex A, we have to mention that we strongly believe that is the right moment to start to develop this project. Next, the most important points that we have summarized are showed:

- **Consumption of television** has a high incidence in Spanish homes, and is decreasing very slowly. This year is expected, as consequence of DTT, an increment of viewers.

- Although the difference between the UE zone, Internet is becoming, gradually, an important piece inside Spanish media. This difference is going to be avoided in the next years.

- Different researches shows that exists an interesting kind of Multi-tasking users related with the media consumption. This users watches television while surfing internet within personal computers and others devices, such as mobile telephones.
• We have found some similar detection projects done, but not exactly the same: Neither of them has a system in order to warn users about the status of a channel in real time.

• Digital Terrestrial Television has been introduced over all the Spanish country. It is an important point because, it is going to be an advantage in order to process broadcasted images when detecting if is commercial time or not.

• In addition, the desired system would be ready to be used in more European countries, not only in Spain, because of the DVB-T standard incidence on the European Union.

1.5. Technical goals

In this section the technical goals that want to be reached with this thesis are going to be detailed. It is important to mention that at the end of this document they are going to be evaluated, in order to get a feedback and define if them has been reached successfully, or not.

• Detect system requirements and specify the ones that are going to be critical for the right operation of the communication system.

• Study and design the best manner of broadcasting the changes of the channel status. For the right development of this point, it is going to be essential take into account scalability features.

• Perform a multiplatform development, in order to achieve a portable system and not have any dependence of the operating system or hardware features.

• Develop the first client software piece for personal computers, in order to test the system and prepare the future public application. It must to be compatible the highest number operating systems and platforms.

• Set a prototype of the system in an integration scenario, and set it running in order to evaluate the continuous executing mode.

• Use open source software when possible. It has been done in order to avoid the payment of licenses, and because the fact that we strongly believe in this manner of developing software. It is important to mention that free not means free-of-charge in all cases, as well open source do not involves free.

At conclusion chapter (Chapter 7) apart from these technical goals, personal ones are going to be shown. They are as important as the technical, but in some cases very difficult, or impossible, to define before starting to work.
1.6. **Timing and project organization**

Time is precious, and money. When starting this project, Jordi Coscolla Alvarez and I were working at different enterprises. For this reason, the fact of organize tasks and events was very important. Preparing meetings and coordinating our thesis development was one of the most difficult challenges.

One of the first things to do, that we totally agree, was preparing an individual Gantt diagram with the task to execute. When it was finished, we gather together and set some common points to evaluate the project development, within meetings and testing proofs in order to ensure that the individual tasks were running properly. In this section, a list of the most important phases of the project, with the most relevant tasks, is going to be shown. It involves only the part that I have developed, but takes into account the meetings and tasks done with Jordi Coscolla [3]. Next they are going to be briefly explained. The Gantt diagram has been located in Annex B.

- **Project definition phase**: Inside this temporal phase, we were standing up the system. A brainstorming with different ideas, reflection times and the first definition of what objectives we want to reach were the more relevant task that we developed.

- **Previous research phase**: Before starting with the design and deployment of the whole system, we were very interested in quantify the possibilities of success of the project. We known that many times it is impossible to known if an idea would result useful or not, but detecting the weakness of it would help us to make some adaptations. This phase includes tasks as research of social behaviors, similar projects and, after these two points, the definition of system requirements.

- **Design phase**: In this part, Jordi Coscolla and I were defining essential aspects about the operation of the project. For this reason, our actions were developed closely. Tasks as define the logical business, project layers, architecture, technologies and message between different modules were discussed here.

- **Deployment and first testing phase**: In this phase is when the project started to be assembled, using all the premises explained in the “Previous research phase”. It is the part that has been developed with more independence, but it does not mean with less coordination. In order to demonstrate that the project was running properly, different previous tests were carried out to evaluate it.

- **Integration phase**: Once the development phase was completed, we started to setting up all the software developed and configurations in a server that we bought for keeping running the most time as possible the system. Unfortunately, and due to the budget limitations, a production environment has not been reproduced.
- **Reporting phase:** Finally, this phase consists on compositing this report and the slides that will be shown in the thesis defense. It is important to mention that it has been done little by little, with the data collected in previous phases.

The duration of this project has been estimated in 9 months, approximately, without taking into account vacation periods.

Finally, other important consideration to mention in the project organization is the necessity of setting a private and personal server in order to make easier sharing files and communicate each with other. A good communication and a fast flow of contents have helped us a lot to achieve our objectives.
CHAPTER 2. TECHNOLOGIES

In this chapter we are going to explain the different technologies that we have used in order to develop the communication part of this Real Time Commercial on TV Detector project. Different protocols, frameworks, languages and libraries that we have decided as the most appropriated to assemble it are described.

After starting the definition of this chapter, we are going to explain briefly the different parts, or layers, in what our project is going to be divided. A further description is done in next chapter (Chapter 3), but we have considered important to introduce it here in order to explain why these technologies have been chosen.

In Fig. 2.1 the different system layers are shown. As we can see, the Communication layer is going to swap information between the Detection and the Client layer. These layers are going to give and request information about the status of TV channels, respectively. Client layer defines, in other words, the software pieces that user is going to use, while Detection refers to one that is going to monitorize in real-time all TV channels available in the system, deciding when commercials are over.

![Fig. 2.1 Modular architecture representation of the system](image)

It is important to mention that the chosen technology in order to develop the communication part of this project would be defined by the behavior and size of Detection and Client Layer. The Detection one would be predictable and reduced, because it would be done and manage by a controlled number of developed programs done by Jordi Coscolla, while the Client one would increase with the number of users, and may be no predictable and cause big loads of transactions. Anyway, both (Client-Communication and Detection-Communication messaging) must be done in real time.
Taking into account these considerations, now we are going to choose the most appropriated technologies in order to developing the communication part of this real time commercial on TV detector project.

2.1. XMPP

For developing the communication part of this project between clients and the whole system, we have decided to use XMPP (Extensible Messaging and Presence Protocol) technology. This protocol is an open technology for real-time interaction, totally biased on Jabber [19] [20]. In fact, some books and resources define “Jabber” and “XMPP” as words interchangeable, because in 2002 the IETF (Internet Engineering Task Force) chose the name XMPP to distinguish the developer community to the protocol. Next we are going to point out the main advantages that it is going to provide:

- **Scalable and real time:** It uses the “push” model of information in order to transfer data. It solves a serious scaling problem associated with traditional HTTP-based pull approaches.

Briefly, it is important to explain that Pull technology is network communication paradigm where the initial request for data is originated from the client, and then server responses. This kind of request is the most classical of network computing, where many clients request data from a centralized server. HTTP pages are a Pull extensively example.

On the other hand, and it is the model that we are going to use, we can find Push technology. It describes a style of network communication where the request for information transaction is initiated by the server, and not the client. This technology is further used in chat messages, user clients emails where new information is pushed by server and go on. Gmail and Gtalk [21] are perfect example, because they are biased in Push paradigm, achieving real-time communications, or near real-time.

In Fig 2.2 we can see the difference between Pull (polling) paradigm and Push. It is important to note that push clients receives before (in most of cases) early information than pull one.

![Fig. 2.2 Polling versus Push paradigm](image-url)
• **Standard**: The core aspects of this protocol have undergone rigorous public review within the IETF. The extensions to XMPP are published in an open, developer-oriented standards process run by the XMPP Standard Foundations (XSF). This approach has resulted in strong technologies that can be freely implemented under any licensing terms, from open source to shareware to proprietary code.

• **Proven**: XMPP has been developed over 10 years. It is considered stable, widely deployed, seriously tested, Internet-scale technology, with dozens of interoperable codebases, tens thousands of deployed services, and millions of end users.

• **Decentralized**: XMPP technologies are deployed in decentralized client-server architecture with unlimited number of servers. Any person or organization can run their own XMPP server and connect it to the rest of the network using standard Internet infrastructure such as DNS, and certificates are freely available through the XSF.

• **Secure**: It provides built-in support for channel encryption and strong authentication, inherent resistance to many forms of malware, a diverse ecosystem of implementations, a decentralized network without a single point of failure, and significant deployment at some of the most security-conscious financial organization and government agencies worldwide.

Over the years, the developer community defined a large number of extensions to core protocols. These extensions are developed through an open, collaborative standards process and published in the XSF’s Extension Protocols (XEP). This have been done because XMPP has been used in order to build a large range of applications, including content syndication, alerts and notifications, web services, multimedia sessions negotiation, intelligent workflows geolocation and more.

We have also to mention that different servers have been developed in order to implement the XMPP communications, mostly free and open source.

![XMPP logo](image)

**Fig. 2.3** XMPP logo

Next we are going to choose the most interesting XEP for our project, as well as the most appropriated XMPP server.

### 2.1.1. XMPP extensions

It is important not to forget the global aim of the communication part of this project: warn to the interested users when our detection server realizes that a
TV station has passed from broadcasting commercials to television programs, and vice versa.

In a first term, and after reading the different models, we started to think about implementing Publish/Subscribe (Pub/Sub) model. This communication model has an extension inside the XMPP protocol: XEP-0060.

Pub/Sub is a messaging pattern where publishers (or senders of messages) are not programmed to send their messages to specific subscribers (or receivers). Published messages are characterized into classes.

Subscribers express interest in one or more classes, and only receive messages that are of interest. This message paradigm can allow for greater scalability, and a more dynamic network topology.

Although all this benefits, and the fact that it was the better solution to our communication layer, the Pub/Sub model was rejected. It was done because there were not available XEP-0060 libraries for implementing clients in Flex, and for us reach this client was considered a priority [22].

In order to solve this problem, we though in a different solution based on MUC (Multi-User Chat). The XMPP extension that corresponds to this technology is XEP-0045.

The basic idea behind MUC is that people can join a room and send messages that are delivered to all the other participants. Thus the room acts as a kind of message “reflector” or “multiplier” (one incoming message is multiplied into many outgoing messages).

XEP-0045 has libraries available for all the client platforms in which we were interested, not such as XEP-0060. The unique considerations that we have to take into account is that chat rooms are going to be treat as Pub/Sub classes, as well as the configuration clients roles: users will not be able to transmit information, only receiving. Only system controlled MUC participants would insert messages in the communication system. These premises are totally compatible with this protocol extension, as we are going to see in next chapters.

Summing up, due to the reasons explained before, our objective was to implement a Pub/Sub communication model adapting XEP-0045, and we got it.

2.1.2. XMPP Server: OpenFire

Nowadays, we have different options in order to implement XEP-0045 through different XMPP servers: Ejabberd, Jabberd14, Jabberd2, Openfire or Tigase are some well-known examples. In order to make a decision, we have done two comparison tables. They can be found in Annex C: Inside the first one (Table C.1) we have represented the most important aspects and characteristics of them, while in the second (Table C.2) the supported XMPP extensions are described for each one.
We have rejected automatically the option of implementing the service through Jabber2, because it does not have implemented XEP-0045 yet (Table C.2) [23] [24]. We have also discarded Jabberd14, jabber2 and Tigase. This decision has been made because, in a future, we would be interested in implementing clients via Pub/Sub (XEP-0060), and these servers are not ready to run this extension.

It means that we only have two XMPP servers candidates: ejabberd and Openfire.

Ejabberd is a server based on the first that was created over the XMPP protocol. It has been done in Erlang language, and is under GPL license: is free and open code. It is multi-platform, modular (you can add new modules in order to reach more functionality), fault tolerant and let clustering.

Openfire is a XMPP server developed for JiveSoftware, and in the past it was called Wildfire. It has been developed over Java language, and consequently it is multiplatform. The license under it has been developed is GNU and is open code. It is modular and based on a plugin system. It also permits doing server clustering.

After studying these two servers, we have decided to implement the project communication XMPP service over Openfire [25]. The main reasons taken into account are:

- It is easy to install and use. It has a friendly and complete web interface in order to manage different parameters.
- It uses java language, the same that we have considered to develop our desktop clients.
- The library that we are going to use (Smack) for the client has been developed also for JiveSoftware. This fact is going to ensure the compatibility.
- Permit more XMPP extensions (XEPs) than ejabberd.

Fig. 2.4 Openfire logo

In this project, we have used the Openfire version 3.6.4. [26]

2.2. Layer connectors: Java Servlets

In order to connect, mainly, the Detection layer and the Communication one, we have developed some connectors that are going to catch messages and
process them. In this subsection we are explaining the essential concepts of Java Servlets (the one we have chosen) as well as the reasons why we have selected it. In Annex D it is further information and a comparative between CGI [27] and Fast-CGI [28] technologies.

Java Servlet [29] is a small, pluggable extension to a server that enhances the server's functionality. This technology allows developers to extend and customize any Java-enabled server such as web servers, mail servers, application servers and others.

Servlets can be loaded dynamically, to expand the functionality of a server. They run inside a Java Virtual Machine (JVM) on the server (Fig. 2.5), so it is safe and portable. Servlets operate solely within the domain of the server: they do not require support for Java in the web browser.

![Fig. 2.5 Servlet life cycle](image)

Servlets are all handled by separate threads within the web server process, unlike CGI and FastCGI. This means that servlets are also efficient and scalable. It is also important to mention that because servlets run within the web server, they can interact very closely with the server to do things that are not possible with CGI scripts.

Another advantage of this technology is that they are portable across operating systems and also web servers: all of the major web servers support servlets. Although servlets are most commonly used as a replacement for CGI scripts on a web server, they can extend any sort of server.

Finally our decision has been to use Java Servlets in order to develop the connection between different layers. The reasons are going to be listed next:

- **Portability**: Servlets are written in Java. They are highly portable across operating systems and across server implementations.

- **Efficiency and Endurance**: It's invocation is highly efficient. Once a servlet is loaded, it generally remains in the server's memory as a single object instance. Due to this fact, it can hold on to external resources, such as database connections, that may otherwise take several seconds to establish.

- **Integration**: Servlets are tightly integrated with the server, allowing cooperation with the server in ways that other programs cannot.
• **Extensibility:** The Servlet API is designed to be easily extensible. It includes classes that are optimized, but at a later date, it could be extended and optimized for another type of servlets.

• **Power:** Servlets can harness the full power of the core Java APIs: networking and URL access, multithreading, image manipulation, data compression, database connectivity, internationalization, remote method invocation (RMI), CORBA connectivity, and object serialization, among others.

• **Safety:** This technology supports safe programming practices on a number of levels. Because they are written in Java, servlets inherit the strong type safety of the Java language. In addition, the Servlet API is implemented to be type-safe.

All Java Servlets of the whole system have been deployed over Apache Tomcat [30] in order to develop the connection between different layers. This servlet content has been used because it was not in the scope of the project discovering the most appropriated and we had worked before with Tomcat.

### 2.3. **Data base: PostgreSQL**

In the scope of this project, it is not considered the evaluation of the most appropriated data base technology for assemble the prototype. In spite of this fact, we have done a previous research in order to do a little approximation to the one that would fit better. The technologies that we have deal with are Oracle8, MySQL and PostgreSQL.

Oracle8, developed by Oracle Corporation, was rejected because it was developed under proprietary licenses, and was incompatible with the philosophy of the entire project. This fact reduces our decision to PostgreSQL and MySQL.

MySQL technology was rejected for the same reason. In spite of being an Open Source solution, it is owned by Sun Microsystems, company which was acquired by Oracle Corporation in 2010. In other words, the product is Open Source, but not the project [31] [32].

By contrast, PostgreSQL relies on a global community of developers and companies that are developing it: it is not controlled by any single company. This is the main reason that we have used in order to choose this option. It is important to repeat that choose a data base technology is not inside the scope of the project, and this decision is as good as other would be possible.

![PostgreSQL logo](image)
In order to administrate our database we have used pgAdmin in its third version [33]. pgAdmin is a free and open source graphical front-end administration tool for PostgreSQL, which is supported on most popular computer platforms, included Linux. This software solution is easy and fast to configure, as well as in order to administrate when installed.

2.4. Desktop program: language and XMPP library

The program language that we have used in this project to develop the user client software is Java. This decision has been made taking into account different factors, but the most important is that learn a new program language is not in the scope of this thesis.

We have also to point out that programming in Java is going to provide us portability. It means that our written Java program would run similarly on any supported hardware/operating-system platform. In addition, pieces of programs written inside Java Servlets would be reused in order to create the desktop client.

In order to implement XMPP clients and system servlets, we have studied different XMPP Java libraries. In Annex C the comparison between them can be found.

After studying the different options, we have also decided to choose the Smack library. This decision has been made because it has a complete API documentations, is fully Java developed and it is compatible with Openfire server (JiveSoftware has developed Openfire and Smack library).

Smack is an Open Source library that has been assembled in order to develop clients of Instant Messaging (IM) and manage the presence of them, as well as to add new functionalities using new extensions. It has been built under Java, and the main advantage is that is very easy and let do loads of operations in few code lines. It license is under Apache [34] [35].

We have also to mention that Smack has an extension library: Smackx [36]. This library contains more extensions of the XMPP protocol, as well as a debugger in order to monitoring the communication between different entities.

It is important to point out that aSmack [37] is the library that Google Android uses in order to implement the XMPP protocol. This factor has to be taken into account because Jordi Coscolla has to develop also a client for this kind of devices. The last available version of Smack library is the 3.1.0 [38], and has been also developed by Jivesoftware.
CHAPTER 3. SYSTEM DESIGN AND ARCHITECTURE

In this chapter we are going to explain the functionalities and requirements of the Real Time Commercial on TV Detector system, taking into account what users are expecting. Next, and also inside this section, the chosen architecture that we have defined to carry out the entire project is going to be shown and justified.

Defining functionalities, requirements and the architecture before developing the project has helped us to thinking in modular way. It has been done in order to divide tasks and personal responsibilities. It does not mean that each module is totally independent to each other, but they were enough defined in order to be done at the same time.

3.1. System requirements

In this section we are going to evaluate the requirements of the whole system. In order make it easier, we have divided it in two different parts: functional requirements and non-functional requirements.

3.1.1. System functional requirements

A functional requirement is going to define a function of our software system and its component. This requirement can refer to different aspects as technical details, data manipulation and others specific that define what our system is supposed to accomplish. Next, system functional requirements are going to be detailed.

- **User identification:** The identification and differentiation of users is going to be also an interesting requirement of the system. It would help us to collect information of the user’s habits and be ready to process it later, discovering behavior patterns.

- **Log and reporting system:** Receiving information from users about when the system is not working properly would help us in order to detect what is producing a failure inside the system, and try to correct it the faster as possible. These user reports would be translate into a simple log system.

- **Warnings:** When commercials on the channel that the user was interested in are over, the application has to notify it. In order to warn the user, that may be doing other tasks, it should to emit different kinds of signals. Apart from the visual ones, sound and vibration (if possible in user device) would be also played.
3.1.2. System non-functional requirements

A non-functional requirement is a requirement that is going to specify criterias that can be used to judge the operation of a system. They are often called qualities of a system.

- **Response time**: The designed software has to present less delay as possible when a commercial on television channel is over. It is one of the most important requirements, because the user will feel annoyed and disappointed if the application does not warn him at this moment.

- **Simplicity**: This program has to be a simple tool. In maximum three steps, or clicks, the user’s application has to reach a listening status where the client software is waiting for events (of the channel that user is watching).

- **Multiplatform**: When developing the Desktop client application, it would be very desirable to make it compatible with the most important platforms that common users have installed at home. Linux, Windows and MacOSX clients should be deployed to gain the majority of personal computers users.

- **Graphical design**: Develop user-friendly graphical software has been also a requirement. In fact, and taking into account that this project is in development stage, we have to mention that is going to be the less restrictive and important.

- **System scalability**: It is one of most important requirements that we have considered. Scalability refers to the propriety that should have our system to increase its total load. For us it is essential because we expect to reach a big number of users and channels to process in a near future.

- **System availability**: This requirement, as scalability, is also very important. It means that we want to achieve the system the maximum portion of time in functioning condition, designing (if necessary) recovery modes in case of a failure of a device or piece of software.

- **Future changes and platform compatibility**: Implementing future changes, or change the platform that system is using without altering others parts is a very common issue of informatics and telecommunications systems. In this case we are not going to make an exception.

This is interesting because, frequently, bugs and other kind of errors are detected, and it is important to solve them easily and fast as possible. Be ready for implementing futures technologies is going to be very useful for the maintenance of the system.
• **Free software licenses:** Use this kind of software will help us in order to keep money in the deployment and production stage, as well as avoid problem in a next future with laws and products copyright. It has been also a personal decision that we have taken, according to our ideals and manner of working.

• **Environmental impact:** Minimize the environmental impact while developing and thinking in when it would be in production stage is an important requirement. Nowadays, IT enterprises and products are also responsible of the total global warming. As IT engineers, mitigate this effects is our responsibility.

### 3.2. Architecture definition

When we gathered together in order talk about the architecture under which the project had to be included, there was an interesting discussion. In order to continue, we decided to take into account some considerations of system requirements as objectives. In this section different modules defined are going to be briefly explained.

### 3.2.1. Architecture layers

Taking into account that we are going to use the Client-Server paradigm and the objectives that we have define in the previous section, we have divided the architecture in three layers or modules: Detection, Communication and Client.

In Fig. 3.1 these three layers are shown inside different color boxes. The red encloses the Client one, while the blue and the oranges refer to the Communication and the Detection respectively.

![Modular architecture representation of the system](image)

Although they will be completely different in technological terms, the connection between them is going to be shared and highly defined. In Communication
chapter (Chapter 4) this aspect is going to be explained. In any case, these layers are going to be defined separately in the next section. Inside these layers we have located different servers, which are going to develop different functions. In the next image (Fig 3.2) they are shown.

**Fig. 3.2 System architecture**

### 3.2.2. Client Layer

It is the agent that is going to interact with the final user. We have developed different interfaces in order to achieve the higher number of platforms and users as possible. Android, Flash, Facebook and Java desktop programs have been ensembled.

In this thesis document the unique software that is going to be explained is the Java desktop one, because the others clients have been developed by Jordi Coscolla and are defined in his project report [3].

It is also important to mention that this piece of software is going to help us to collect information of users. In order to achieve it, it will keep a unique identification number, which is going to be sent with all the request of the user. With this information, we would be able to study behaviours of viewers in future project extensions.

Summing up, this layer is the one that is going to interact with the user. It has to show the available channels to the user (which ones are monitorized), collect the information about him (what program he wants to see) and send it to the server, as well as listen to a response that warns about a change in the status of the channel. This Java desktop application will be explained better in Chapter 5.
3.2.3. Communication Layer

This layer consists on a combination of servers that are going to implement the logical business part related with communication features. The main function of this module is to broadcast changes of each channel (pass from commercials to broadcasting, and vice versa) only to the users that are watching them.

It is important to mention that, apart from managing these changes, it is going to manage users, collect data about behavior of them, register incidences reported by users about a failure when detecting or receiving this information in order to be able later to debug where and when the error was located.

In next diagram (Fig. 3.3) we can see the implemented servers inside this layer. Now they are going to be briefly described. We have to point out that all these servers have been deployed over Ubuntu Server 8.04 LTS [39].

![Fig. 3.3 Communication Layer servers](image)

- **Login User Server**: This server implements a Java Servlet that receive user's request for connecting to the system. It searches if the credentials which the users pretend to connect are right.

  In order to see if their credentials are correct, it does some queries to the OpenFire database. If the user name already exists and is not connected, it let the connection with these credentials. Otherwise, if they exist or are already connected, new random credentials are generated and returned to the client.

  The messages between client and this Servlet are explained in Chapter 4. It runs over an Apache Tomcat 6.0.

- **XMPP Server**: This server implements the OpenFire server on its version 3.6.4 [40]. As we have mention in technologies (Chapter 2), it is the responsible of managing the XMPP communications between users and the system.
• **Gateway Detection Message Server:** This module is the one that is going to receive data from Detection Layer. It means that when a change is detected in a channel, it is going to be the first entity that will receive a notification.

It has been implemented over a Servlet, which is going to receive an XML message generated by the Detection layer, and later retransmit it to the XMPP Openfire server, to advertise users that a change has taken place. It has been deployed over Apache Tomcat 6.0.

• **DB Server:** Inside this system, and as we have mention in technologies (Chapter 2), the data base system that we have chosen is the PostGreSql 8.3 [41].

In this server two data base are going to be managed: one related to OpenFire functions, and other that is going to store all changes that took place on the system, as well as the possible failures of it and user behavior information.

• **User Valoration Server:** Finally, and in order to collect the user’s error perceptions, we have develop a Servlet. It is going to receive the channel that users where watching, storing the data and time of the failure.

It has been done in order to debug application failures, as well as detect if is some error on the detection algorithm. This module runs over an Apache Tomcat 6.0.

In Chapter 4 there is a detailed explanation about the manner of taking decisions of this layer, the messages between the different agents and the implementation of the whole communication system.

In order to achieve the maximum scalability and availability, we have to mention that all the technologies used in all this modules can be set in cluster mode. Also, it is important to mention that Apache Tomcat, Openfire and PostGreSQL are able to run over UNIX and Windows platforms.

### 3.2.4. Detection Layer

It is the layer that is going to do the image processing tasks, and will decide if a television channel is broadcasting commercial, or not. When a change took place, a XML message is going to be sent to Gateway Detection Message Server.

This part is explained inside Jordi Coscolla thesis [3], but the messages between this layer and the Gateway are explained in Chapter 4, because the implication that it has in the communication part.
CHAPTER 4. MESSAGE SERVICE DEPLOYMENT

Inside this chapter we are going to explain how we have implemented the message service of the whole system. Messages transmitted between different agents, actions and decisions taken by them and data manipulations are going some aspects that are going to be described.

4.1. Openfire Server

In order to use Openfire server to exchange messages, and as we have mention in technologies chapter (Chapter 2), we are going to use and adapt the XEP-0045 protocol. This protocol implements a group chat, also called MUC (Multi-User Chat).

Our basic idea is going to create one room per channel processed. When users join the system, they will select and join a room, and they are going to receive different events send by detection server. One room is going to represent only one channel or TV station.

To achieve this kind of communication, different parameters of the rooms and users permissions have been defined. Next we are detailing this configuration. Firstly we are going to show the user’s configuration inside a room, and later the whole configuration of the room.

4.1.1. Room user’s configuration

Room user’s configuration is going to be an essential part of the communication layer. In email list, web forum or chat rooms some people, unfortunate, misbehave. For this reason, enable crowd control in MUC technology within defining leaders (typically called moderators or admins) who are going to be able to kick out or ban users is very important.

In the communication part of this project, enable this kind of control would be further important than in a conventional MUC system. We are appointing out it because if a user is able to write inside the room, and its messages are reflected to all users, it can put our system at risk (creating false alarms or flooding the room with non-controlled messages).

Luckily, XMPP group chat is no exception. MUC technology contains various systems in order to enable crowd control, including kick out and ban users, and the ability to limit who can talk in a MUC room. Next we are going to define the different affiliations, or roles, that have been defined in XMPP MUC [19] [42]:

- **Outcast**: Someone who cannot even join the room.
- **Visitor:** Someone who can join the room and listen to the conversation, but who cannot speak.

- **Participant:** Someone who can both: listen and speak inside the room.

- **Member:** Someone who can listen, speak and join the room if it is members-only.

- **Moderator:** Someone who can listen, speak, kick participants and visitors, and toggle others’ ability to speak.

- **Admin:** Someone who can listen, speak, kick participants and visitors, toggle others’ ability to speak, see the real JIDs (Jabber ID) of occupants, name new moderators and members, and reconfigure some room options.

- **Owner:** Someone who can listen, speak, kick participants and visitors, toggle others’ ability to speak, see the real JIDs of occupants, name new moderators and members, and reconfigure all room options, name new admins, and destroy the room.

For this prototype version, the unique tools that we have considered is the kick out and talk limitation. For this reason the roles that are going to be used for common users (clients) is visitors, while the ones of the system are going to be moderators (tvadmin and kickadmin). It is important to mention that inside the system we have defined a controlled visitor user (tvreg) in order to debug and simulate common users.

For this prototype version, the unique tools that we have considered is the kick out and talk limitation. For this reason the roles that are going to be used for common users (clients) is visitors, while the ones of the system are going to be moderators (tvadmin and kickadmin). It is important to mention that inside the system we have defined a controlled visitor user (tvreg) in order to debug and simulate common users.

### Ocupantes en el cuarto

<table>
<thead>
<tr>
<th>ID de la Role</th>
<th>Usuario</th>
<th>Correo</th>
<th>Último modificación</th>
</tr>
</thead>
<tbody>
<tr>
<td>n0</td>
<td>4/j0</td>
<td>Oct 26, 2010 11:46 PM</td>
<td>Oct 26, 2010 11:46 PM</td>
</tr>
</tbody>
</table>

### Fig. 4.1 Different users inside TV3 channel room

It is also important to denote the difference between MUC roles and affiliations. As we can see in Fig. 4.1 roles and affiliations are not the same. Typically roles are temporary (they last only as long as a user is inside the room) whereas affiliations are permanent (they last across group chat sessions). However, MUC services are allowed to cache roles across sessions, so in practice the distinction is not clear-cut. For this reason, in this report we are going to refer to roles, and not affiliations.

Inside system messages subsection (4.4. System messages), the messages generated by room’s moderators would be defined.
In the next table (Table 4.1.) we have sump up the actions that users can carry out depending on the role that they have.

**Table 4.1. Privileges associated with roles**

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Outcast</th>
<th>None</th>
<th>Member</th>
<th>Admin</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter Open Room</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Register with Open Room</td>
<td>No</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Retrieve Member List</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Enter Members-Only Room</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Ban Members and Unaffiliated Users</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Member List</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Moderator List</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Admin List</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Edit Owner List</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Change Room Definition</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Destroy Room</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 4.1.2. Room configuration

In order to configure the Openfire service properly and fast, we have develop a command program which is able to create and erase automatically all the rooms with the properties tuned as we have considered convenient. In Annex E, we are going to explain different rooms properties and the values that we have assigned to them in order to fit them the best to our project requirements, as well as the routine that is going to implement these room creations.

When creating rooms within Openfire administrator website, there exist a form (Fig 4.2) that has to be filled with the values that administrator of this service believes as appropriated to it.

![Openfire configuration room form](image)

Fig. 4.2 Openfire configuration room form

Luckily, Smack Java library has some objects and methods in order to fill it automatically. Thanks to it, we have created the program that can create and configure them automatically. In the same annex (Annex E) we can find how we
have implemented it and the main values that we have assigned to these rooms.

We have to mention that, in order to create and configure rooms automatically, we have developed a program. This program is able to read an XML file where channels and description are introduced, as well creating and destroying system rooms.

```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<channels>
  <room id="TV1" bc="all" desc="Estat del TV1"/>
  <room id="A3" bc="all" desc="Estat de Antena 3"/>
  <room id="TS" bc="all" desc="Estat de Tele Cinco"/>
</channels>
```

**Fig. 4.3** XML file where channels are described

Finally, when executing this program with the XML file (Fig. 4.3), the result inside Openfire administrator website is the one that we can see in the next image (Fig. 4.4).

**Fig. 4.4** Conference rooms

### 4.2. System messages

Next, messages that different layers and systems agents swap are going to be detailed. It is important to mention that only the messages designed for the application are going to be explained, ignoring the ones that only depend to the XMPP protocol, but not the body or content that is introduced inside them intentionally to transmit useful information to users.

#### 4.2.1. TV program / commercial broadcasting detected

When commercials have finish in a channel, it is time to warn user about this phenomenon. In order to do it, different XML messages are going to be switched between different servers.

The first message that is send is the one generated by the Detection Server. As we have mention, this server contains the television detection algorithm. This XML message is sent to a Java Servlet of the Listener Server who will add the field “notify” and package it inside the body of XMPP message addressed to the
Openfire server (Fig. 4.4). It is important to mention that this Servlet is going to be connected to each room of Openfire server, and will send it fastly.

Finally, OpenFire will broadcast this message to all the users that are connected to this room, advising them about this event. In order to differentiate the messages that are sent via XMPP to the rest, they are painted in blue in next figures.

![Fig. 4.4 Message switched when TV program broadcasting detected](image)

We have also to mention that, in order to storage these changes, Listener Server is going to update the status of the channels inside the table “CHANNEL” of the database system, as well as insert this change into “CHANNEL_HIST” table of the same. These tables are defined in Annex F.

On the other hand, we have the case that involves when a commercial is detected in television and we have to report to the user that the TV program in which he is interested is not broadcasting. The exchange of message is exactly the same, but as we can see in Fig. 4.5 the values assigned to fields of the messages are different.

![Fig. 4.5 Message switched when commercial broadcasting detected](image)
The unique field that changes in the change XML message (between Detection server and Listener server) is \textit{ad}: in this case it is true. About the \textit{status} XML message that is send via XMPP, the changes response to fields \textit{ad} and \textit{notify}, that are true and false respectively.

4.2.2. User register and login

After a user install the application in its device, the first step in order to start using this application will be get registered in the system. It is done automatically for the user, but behind this process, there are different messages that have been exchanged between the user client software and the system. Firstly we are going to describe the messages when registering, or what is the same, the first time that a client is executing the Java desktop client that we have designed.

When a user is registering for first time, his software client sends a message to a Servlet that is inside the Register Server (Fig 4.6). This message contains a user name per default and his GPS position, if it has GPS module. When Register Server receives this message, it tries to look for it inside a database. This default user name does not exist, and consequently, it generates a new UID. When it is generated, it is included inside the Openfire and system databases. Finally, the assigned UID for this client, as well as the list with the available channels of the system, are sent.

It is important to mention that, when this stage finishes, the client will save this UID inside an XML document permanently. Then, the system will be waiting for a message in which the channel that user wants to be informed is defined. For avoiding and detecting cheaters the IP address of users would be storage in the database system.

![Fig. 4.6 Messages switched when registering](image-url)
If the user was already registered, the unique difference, as you can see in Fig. 4.7, is that data is not storage in database, neither inside the software client, and no XML is created inside the client device.

![Fig. 4.7 Messages switched when registering](image)

### 4.2.3. User request

When the user has the list of channels available loaded in his client software, he has to choose one. When this election is done, this software would send an XML message to the Selection Server Servlet. This servlet is going to ensure that the client has the same IP than in the registering stage, in order to avoid cheating, and if they match, it would response with the requested channel status (commercials or broadcasting). In next figure (Fig 4.8) the messages switched in this action are shown, as well as the database updates and actions.

![Fig. 4.8 User channel request messages](image)

### 4.2.4. User error reports

We have also designed a Servlet in order to catch incidences fired by users. They have a button in its Java desktop application in order to report when a detection or a communication failure has taken place.
This Servlet is going to receive a simple XML (Fig 4.9), in which the username of the client, as well as, the channel he was watching when the error was detected are included. This data will storage in a table called USER_REPORT, inside our database system. It is not going to response to users, is done in unidirectional way.

It is important to mention that, in each register inserted into this table, a time stamp field is going to be filled with the local ti me of the server. Thanks to this time stamp, it will be easier to detect when the error took place, and differentiate if a report has been done by further than one user.

![Fig. 4.9 Messages between user client and valuation server](image)

### 4.2.5. Kicking users

Finally, and in order to finish with the messages exchange between different agents, we have to mention that a kick user program has been designed. This program is going to emulate different users that are watching all channels, or in other words, this program is going to have a user inside each channel room.

The name of the user that is going to be inside these rooms is kickadmin, and will have administration permissions. It means that, among other actions, this user is able to kick users from a room.

For this reason, and in order to empty the rooms when the program broadcasting has started on a TV station, this kick user is going to send an XMPP presence message, kicking users when it starts. To avoiding errors, when kickadmin receives a XML message via XMPP MUC reporting that commercials have finished, it is going to wait 30 seconds, and later make a request to database of the system in order to confirm that commercials have finished. In case that the status does not corresponds to the expected, kick message for users will not be sent. In the next figure (Fig. 4.10) the messages swap are shown (when the detection has been done successfully). It means that the result of the second 30 seconds returns the expected result and the XMPP message for kicking users (status code="307") is broadcasted.
4.3. Database System

In order to manage information about connections and users, as well as keep information about TV stations broadcasting status, a Data Base system has been created and deployed. It is an important piece of the project because we have considered that, in a near future, it is going to be an added value to manage the data of user's behavior, as well as channel historical changes.

Next, the different tables created inside this data base are going to be explained, as well as the benefits that we are going to reach with the usage of them. In Annex F the SQL definition of each table is shown, in order to clarify their definition.

4.3.1. User information

In order to keep a user status and their historical actions, detect errors reported, by users, as well as to avoiding cheaters and attacks, we are going to define three database tables: USER_INFO, USER_HIST and USER_REPORT.

- USER_INFO Table: This table is going to contain the main information of users. Its UID (or USERNAME), the last IP through they connected to the system and, if possible, it GPS position are going to be collected every time the start the client application.

This table, apart from helping us to detect the place where users are watching television (geolocation via GPS when possible, and via IP address if not available GPS data) is going to help us in order to ensure that the different petitions that a user is doing are generated with identical IP addresses.
The detection of different addresses to the same user inside a detection process, or a high number of users UID behind an IP address will warn us about the fact that somebody is cheating or attacking our system.

- **USER_HIST Table**: In order to storage an historical about what users have been watching, we have designed a simple table where the USERNAME (or user UID), the channel and the time are going to be keep in the same register.

  In future applications, it is going to let us count the number of people that where watching a TV program, within mapping this data with a television's schedule grid.

- **USER_REPORT Table**: When developing the application, we thought that to get a feedback from users would be interesting. For this reason we have enable a function in order to get reports when the system, or application, does not run properly. We have also created a table in order to storage this user reports with the user UID, the channel they were watching and the moment (or time) when it was reported.

  The storage of the username is done to detect when we have to take seriously a user. Cheaters or non-criteria users would not been taking into account when processing these reports.

### 4.3.2. Channel information

In this subsection we are defining the tables that we have been designed in order to keep information about current channel status and their historical changes.

- **CHANNEL Table**: This table is going to keep the current status of each channel that the system is processing. It would be very useful because it is going to set a recovery point in case of system failure.

  As we can see, in Annex F, the unique fields that it contains is NAME of the channel, ID of the channel (It has to be unique, because is the primary key of the table) and AD, that represent the status. If AD is equals to true, it means that at this moment commercials are broadcasting in this channel. Otherwise, the program would be in broadcasting status.

- **CHANNEL_HIST Table**: Every time that is received a change from detection layer, it is going to be storage a register. It is done in order to be able to process possible detection errors, as well as, extracting information about how many time commercials have been broadcasted, and now how much time they last. It is the historical channel status.

  For further information about these tables, Annex F contains the whole SQL definition of them.
CHAPTER 5. DESKTOP CLIENT DEPLOYMENT

In this chapter, we are going to explain briefly how we have developed the first Java desktop client. In order to make it easier and understandable, we are going to present two simple UMLs. Used technologies and the obtained results of this software are going to be also showed.

5.1. Functional schema

UML (Unified Modeling Language) is the standardized general-purpose modeling language that we have chosen in order to create a visual model of our software. It offers a standard way to visualize a system’s architectural projects, including elements as activities, actors, processes and components.

The first diagram that we have chosen to represent the functionalities of the developed application is the use case diagram. It is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. It belongs to the dynamic UML view model diagram group. As we can see in the picture that represents the UML use case diagram of the system (Fig. 5.1), in our case, this kind of interaction diagram define seven different uses cases, two actors and a system boundary box.

![Fig. 5.1 UML use case diagram of the system](image)

Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behavior, perhaps the system or use case boundaries should be re-examined. Alternatively, interaction among actors can be part of the assumptions used in the use case.
• **Actors:** Inside the UML diagram we can see two different actors: User and System. An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

Client actor represents the software that we have developed for the usage of a person that download our program from an internet website, and install it in his personal computer, while Communication System Layer represent all the programs and routines that developers have designed in order to achieve the functionalities of the project.

• **Use cases:** Inside the developed diagram (Fig 5.1) it is shown different use cases. They describe the group of actions that provide our client, and which can be measurable. They have been drawn inside “Real Time Commercial on TV” rectangle as horizontal ellipses. Start the program, show available TV channels, select one, report an incidence or warn users when advertisements have finished (as well as quit the application) are the actions that our piece of software is going to implement.

• **System boundary boxes:** The rectangle is drawn around the use cases, is called the system boundary box. It indicates the scope of our system. In this case, all the use cases are inside this box, what means that all this functionalities are treated or considered of the project. If they were outside it, they will not belong to our scope.

### 5.2. Sequence schema

In order to understand how we have designed the client software, and represent the different layers and classes that has been programmed, we have developed an UML sequence diagram.

This diagram is a kind of schema that shows how processes, programmed classes and objects operate one with another, and in what order these actions take place. In Fig. 5.2 the UML sequence diagram of the client is shown. It is important to mention that not all the developed classes and objects have been represented in the diagram. Only the most important are there, and all have been simplified.

The first object that is available, when user starts the application, is “Welcome Window”. When he pushes an “ok” button, it is hidden and “Select TV channel Window” object is created and becomes visible. This object, when created, do a request to a servlet of the system who response with all the available TV channels. These channels are represented inside “Select TV channel Window”, and the client program waits for a user action.
When the user decides the TV station in which he is interested, and pushes the “start” button, a “Status TV channel Window” object is created, while hiding “Select TV channel Window”. This new object is going to create a XMPP connection in order to log and join the MUC room of the selected channel, as well request to a system servlet information about the status of this TV channel. This window object is going to show the status of the channel, and when a change took place in the selected TV station, XMPP connection is going to warn it, and the window would alert the user about this event.

5.3. Developing technologies

In order to achieve our first desktop client, we have used different Java libraries. It is important remember that the objective of the whole project is to get ready a prototype, but not a final product. For this reason the application is not as attractive as we would expect.

5.3.1. Libraries

In order to explain briefly the different libraries that we have used, the explanation has been divided in the topics they were related with.

- **XMPP connections:** The library that we have chosen to implement the XMPP communications in the client side is Smack [35]. It has been selected as the best option because different reasons. The first one reflects that it is fully compatible with OpenFire server (it has been also developed by Jive Software), and the second one (and no least
important) is that is the one that we have chosen to implement the business logical of our communication project layer, and we known how it runs. In Chapter 2 it is available more information about this library.

- **Visual interface:** In order to implement the Graphical User Interface (GUI) of the program we have chosen Swing [43] library. This library is the primary Java GUI widget toolkit. It is part of Sun Microsystems’ Java Foundation Classes (JFC), which is an API designed for providing a GUI for Java programs.

  We have used it because it was included inside different Java IDEs and it provides a more sophisticated set of GUI components than the AWT [44] (Abstract Window Toolkit).

  Other important reason in order to make this decision was the fact that provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform.

  It is important to mention that, in spite of using Swing, to include some functionalities and properties we have also used AWT.

- **XML document process:** As we have reflected in Chapter 4, the client keeps an UID in order to log in future occasions. In the Java Desktop application this data is storage inside a simple and small XML document, which is only created once: the first time that the user tries to connect to the system.

  In order to process this XML documents we have used the DOM library [45]. It has been done because when programming its model is much more familiar than others, as well as having it loaded in memory does not affect to our performance program. If the file had other features, may be SAX’s libraries would be also considered, because it is able to process XML documents that are not available into memory, and gets better performance results. SAX is ideal when reading streams XML communications because lets serialization.

### 5.3.2. Installations platforms

When the client application was finished, we decided to create different installers for the most important and popular Operating Systems. It was done because, in spite of no belonging to the scope of the project, we considered important to simplify the installation to FFF (Friends, Family and Fools). In fact, during the reporting of this document different persons belonging to FFF group has tested this prototype and has done a feedback to us (Annex J).

The different platforms considerate as most popular were: Linux, MacOsX and Windows. After a modest research, we discover a tool prepared to generate
executable files from a .jar file for this kind of Operating Systems. It is called Install4j [46].

Install4j is a multi-platform Java installer builder that generates native installers and application launchers for Java applications. It provides Mac OS X, all 32-bit and 64-bits versions of Windows (included Vista), Unix and Linux RPM support.

The unique inconvenient of this piece of software is that is not free. It has been developed by ej-technologies, enterprise which has all the rights of this application. In fact, it is the only non-free software that we have used in the entire project. For this reason, we used a 90 days free evaluation version, with some limitations, but free of charge.

5.4. The result

Finally, we are showing the result of this first client application. In order to do it, we have considered that the best way is to show some screenshots of this piece of software, as well as describe them briefly.

When users execute the program, the first windows that is going to appear is the one that should provide information about what this software wants to achieve, the version that user has installed and where he can download it. This window corresponds to the one that is on the left of Fig 5.3, and we have named it as “Welcome” window. If user pushes the “Start!” button that is locate inside “Welcome” window, it would disappear, and another window called “Select TV channel” would become visible. The screenshot of this window is shown inside Fig 5.3, on the right of the image.

```
Fig. 5.3 Client “Welcome” window (left) and “Select TV channel” window (right)
```

“Select TV channel Window” is the one that is going to show the available channels that user would be able to choose. They are going to be shown inside a combo. Then the user has to select the one he is interested in and push “Start detection” button, which has been located in the bottom of the window.

Finally, and when the “Start detection” button has been pushed, it would appear “Channel status” window. This window is going to show if in the selected
channel they are broadcasting commercials or not, as well as if the user has become disconnected from the system (Fig. 5.4).

![Client “Channel status” windows](image)

**Fig. 5.4** Client “Channel status” windows

It is important to mention that the disconnected status is going to be shown when the system would kick the user from the room where channel status is broadcasted. As we have mention in Chapter 4, it would take place 30 seconds later of when program broadcasting starts.

User can also report an incidence when detects a failure clicking the “Report failure” button (Fig. 5.4). This button is inside “Channel status” window. When reporting it, a dialog would appear in order to show our gratitude to the client that has warned this error.
CHAPTER 6. PROTOTYPE INTEGRATION ENVIRONMENT

In this chapter we are going to explain the platform that we have used in order to create our prototype integration environment, as well as the configuration of it. With this, what we pretend is to demonstrate that the whole system runs, starting to identify different SPOFs (Single Points Of Failure) and bottlenecks.

6.1. Environment description

The first step in order to start assembling our prototype was to decide the hardware that we were going to use: our server and DVB-T PC cards. The DVB-T cards choice is explained in Jordi Coscolla’s report thesis [3] because, as we have explained, he has developed all that is relative to the processing and detection of television signal.

The server choice was directly related with the money available for the development team, or in other words, it was directly related with our own pocket. The device bought was a HP ProLiant ML115 g5 server [47]. Their main features are explained inside Table H.1 (Annex H).

Fig. 6.1 HP ProLiant ML 115 G5

This server has been located in a domestic LAN, and it is connected to internet through an ADSL. The download/upload maximum speeds achieved are 6 Mbps/1 Mbps respectively. Without considering this connection, the total cost of setting the server, with all the components in order to develop the first prototype, was 780€. It is also detailed in Annex H.

6.2. Configuration integration environment

When setting up the server, we started to think about how to deploy our prototype and the most appropriated operating system to install inside it. Different options were considered, and finally a virtualized scenario was carried out. The main benefits of for our project are:

- **Isolation:** Between different virtual servers there is an implicit isolation that is going to provide us a protection. Install different applications (or in
Our case, system functions) in different servers are going to prevent impacts when upgrades or changes are made.

- **Efficiency**: We can change the requirement of the different virtual machines taking into account the resources that they are going to use. They are not fixed, and can be shared in order to achieve a high performance.

- **Multiplatform**: Virtualization lets us to install different and multiplies operating system technologies on a single hardware platform. Linux, Windows and others operating system are able to run simultaneously.

- **Duplication**: We can develop a standard virtual server and build other only duplicating it, very fast.

Nowadays there are different software pieces available in order to achieve a virtualization scenario: Citrix XenServer [48], VMWare eSXI [49], and VMWare Server [50] are the most popular solutions for the server side.

VMware Server 2.0 [50] has been chosen because it was the unique free virtualization solution that supports USB connection devices to virtual machines when the prototype was developed, and USB support was needed because DVB-T PC card uses this communication interface to transmit information to computers.

Briefly, we have to mention that VMware Server is an entry-level virtualization-software server suite developed and promoted by VMware, Inc. It is not a hypervisor, but supervisor. This means that is a part of an operating system that controls the execution of virtual machines [51].

![Fig. 6.2 VMware Server 2.0 Infrastructure Web Access](image)

As we can see in Fig. 6.2 VMware Server 2.0 uses a web-based user-interface, the "VMware Infrastructure Web Access". This product is free license, but closed source. On January 2010, VMware Server was declared end of availability, and it general support will end on June 30, 2011.
The hardware reservation resources assigned to the different virtual machines are described in Table 6.1. As you can see them are very similar, but there exists some differences of RAM memory and hard disk capacity.

Table 6.1. Virtual machines configuration

<table>
<thead>
<tr>
<th>Server Name</th>
<th>RAM (MB)</th>
<th>CPU</th>
<th>HD (GB)</th>
<th>USB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection Server</td>
<td>1024</td>
<td>1 processor</td>
<td>32</td>
<td>Yes</td>
</tr>
<tr>
<td>Openfire Server</td>
<td>512</td>
<td>1 processor</td>
<td>8</td>
<td>No</td>
</tr>
<tr>
<td>Web Server</td>
<td>512</td>
<td>1 processor</td>
<td>8</td>
<td>No</td>
</tr>
<tr>
<td>DB Server</td>
<td>512</td>
<td>1 processor</td>
<td>12</td>
<td>No</td>
</tr>
<tr>
<td>Support Server</td>
<td>512</td>
<td>1 processor</td>
<td>12</td>
<td>No</td>
</tr>
</tbody>
</table>

6.3. Evaluating integration environment

In this subsection we are going to evaluate the final behavior of the system prototype. It has been done within two different tests. The first one consist on take objective measures of the use of the Java Desktop application, while the second is biased in doing different tests with persons in order to evaluate their feeling about the application and the whole system.

6.3.1. Response time test

In order to measure the response time of the Java desktop application and the system, we have done some measurements with Wireshark [52]. This application is going to show us the XMPP (Jabber) flow packets, as well as other that client application interchanges with the system.

This test has been done manually, or what is the same, we have simulated different users. It has been done because benchmarking separately servers or only a part of the system is not fully suitable for our case.

- **Register user servlet**: As we have mention in Chapter 4, this is the first request that a client does to the system. As we can see in Table 6.2 the average response in that case has result on 68 ms, what is a good value taking into account that the network connection where server is connected is a simple ADSL of 6 Mbps/1 Mbps of download and upload respectively.

![Fig. 6.3 Time schema of Register user request and response](image)

In order to get representative values of this request, we have repeated the same test (Table 6.2). In fact, we have done it 15 times.
**XMPP connection:** When the response of the Register User servlet arrives to the client, and the system has confirmed that he is an authorized user of the system, the XMPP connection between the client and the Openfire server starts. It is important to mention that Wireshark recognizes the XMPP protocol within Jabber name. This is because, as we have said, XMPP is a protocol that has been developed from Jabber protocol.

In the next figure (Fig. 6.4) there are different messages that the client swaps with the system. As we can see, the XMPP messages have been transmitted under a TLS protocol. This protocol provide to our communication security over the Internet.

As we can see there is a gap of 0.138 seconds between the HTTP response 200 OK of the Register User servlet (Fig. 6.3) and the first request of the XMPP communication (Fig. 6.4). This delay is due to the execution of the client application deployed.

In 3.4 seconds (average), the whole communication between the Openfire server and the Java Desktop Client has been established. In next table the register XMPP process are going to be quantified, making and average of
the different connections that we have done. These values have been extracted from the 15 times that we repeated this test.

<table>
<thead>
<tr>
<th>Max. [s]</th>
<th>Min. [s]</th>
<th>N/A [s]</th>
<th>Average [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>2.8</td>
<td>-</td>
<td>3.4</td>
</tr>
</tbody>
</table>

- **Client information servlet**: In order to discover the status channel that the user has chosen, a request to Client Information servlet is done. This process is explained in Chapter 4. As we can see in Fig 6.5 the response in that case has been of 89 ms, very similar response time to the register user servlet case.

In the next table (Table 6.4), and as we have done in XMPP connection, the values of the test are going to be shown. We have also repeated this test 15 times.

<table>
<thead>
<tr>
<th>Max. [ms]</th>
<th>Min. [ms]</th>
<th>N/A [ms]</th>
<th>Average [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>320</td>
<td>79</td>
<td>-</td>
<td>82</td>
</tr>
</tbody>
</table>

### 6.3.2. User feeling test

In order to evaluate the user feeling when using the Java Desktop client, we have done a simple acceptance test. This test has consisted on testing the operational system by internal project developers and FFF group (*Family, Friends and Fools*), before having it available and released to clients or costumers. A total of 15 users checked the developed prototype.

When they were testing the application, a questionnaire was hand over to them. It can be found in Annexes J. Now we are going to explain briefly the results obtained.

- **Client GUI**: The 90% of the testers that has used the Java Desktop application client for windows have reported that it has to improve its graphical interface. They would prefer a launch icon near to the windows clock, as applications as MS messenger or Skype.

- **Simplicity**: The 100% of testers are satisfied with the simplicity of the application. In spite of the fact that some of them do not understand
English, they have used it without problem, knowing in each moment what to do with the application.

- **Speed**: 70% of testers have reported that when pushing the “Start” button of the Welcome window presents a little delay. It is a bit annoying, but not important because they have reported also that when the program starts it runs quickly and in appropriate mode.

- **Usefulness**: All the users that have been done this test have mention that they find this application useful for their daily life. We know that this kind of comment may result subjective, because they are family and friends, but we have to reflect it in this report because it was what they answered in the questionnaire.

- **Others**: The 60% of testers that have used the Java Desktop application do not like the sound emitted when commercials finishes. They would prefer other sound less annoying. In fact, and personally, I totally agree with them.

### 6.3.3. Are we really pushing?

In this section we have evaluate the consumption of bandwidth generated by a single client to the system. In order to get some results, we have installed the desktop client application inside a personal computer which has disabled all the applications and services which connects to internet or other network. Then we started to monitorize within Wireshark software the packets received and transmitted by this computer.

It is important say that Kickuser feature was disabled from the system, because we wanted to analyze the total number of packets that XEP-0045 adapted by us was producing, and not the fact if this kick skill of the system was running properly.

In the next figure (Fig. 6.6) we have represented the total traffic generated in the system when starting a client application. The green line represents the total traffic of the network interface, the black one the input (download from server to client side) while the red the output (generated from client to server side). The y-axis represents the data transmission in bps, while the x-axis GMT time. This information was collected one Sunday (23\textsuperscript{rd} of November of 2010) between 04:44 pm and 04:59 pm. The simulated user was watching TV5.
This graphic (Fig 6.6) has been extracted directly from Wireshark, and corresponds to the phase in which the client is logging into the system. In order to summing up this information we have processed the information relative to these packets and we have design the next table (Table 6.5).

<table>
<thead>
<tr>
<th></th>
<th>Max. [kbps]</th>
<th>Min. [kbps]</th>
<th>Average [kbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>12</td>
<td>0</td>
<td>3,12</td>
</tr>
<tr>
<td>Input</td>
<td>14</td>
<td>0</td>
<td>4,32</td>
</tr>
<tr>
<td>Total</td>
<td>26</td>
<td>0</td>
<td>7,44</td>
</tr>
</tbody>
</table>

It is important to mention that when the session to the server is established (the user is waiting for a trap), the traffic is negligible. In next figure (Fig 6.7) it is shown. As we can see there are synchronous transmissions in order to maintain the connection established, in exception at (16:58:41), which is the moment when the system warn to the user about the fact that commercials on this TV channel has finished. In Table 6.6 numerical results are shown.

<table>
<thead>
<tr>
<th></th>
<th>Max. [kbps]</th>
<th>Min. [kbps]</th>
<th>Average [kbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>1,0</td>
<td>0</td>
<td>22,81</td>
</tr>
<tr>
<td>Input</td>
<td>2,2</td>
<td>0</td>
<td>18,84</td>
</tr>
<tr>
<td>Total</td>
<td>2,4</td>
<td>0</td>
<td>41,65 bps</td>
</tr>
</tbody>
</table>

Summing up, we can divide the transmission rates of the desktop client in two stages: the logging-choose channel and the waiting for event one. The logging-
choose channel is the one that more network consumption is going to require, because different information is exchanged in a short period of time (about 12 seconds), while in the waiting for an event stage, the average of transmission is negligible.

This data has to be taken into account when implementing new architectures of the current system in future deployments. This is going to be done in next project phases because, and as we have said, the unique objective of this thesis was to mount and test the first prototype.
CHAPTER 7. CONCLUSIONS AND FUTURE LINES

In this chapter we are going to explain the conclusions of the whole project. It is important to mention that as personal as technical ones are going to be reflected here. Defined objectives and environmental impact issues are going to be also analyzed in this section.

7.1. Technical conclusions

When developing this project, we have extracted some different conclusions about technologies that we have used and the way of implementing them. Next we are going to summarize the most important ones.

- **Multi User Chat XMPP emulating PubSub**: Using XMPP in order to get push connections was one of the first requirements that we see as essential to our project. PubSub paradigm would be the best choice for the system that we have designed, but no XMPP library was ready for implementing it over flash clients. For this reason we implement a XEP-0045 library (MUC over XMPP) to get the same functions as XEP-0060 (XMPP PubSub extension) through role configurations of MUC users. It was a success.

- **Openfire server, simple but complet**: When choosing XMPP server we decided to use Openfire. Now, we are finishing the project, and we are ensured that the decision was right: it is easy and fast to install. In addition, we have discovered that it contains loads of plugins in order to adapt new standards, and permit a fully configuration.

- **Apache Tomcat, secure value**: We decided to use Apache Tomcat for our project because it was one of the well-known Java Servlets that we had deal with. After doing the whole implementation of the system we have to mention that it has done exactly what we expect, and we are happy about it performance. While working with tomcat we did not have any problem, and all the Java IDEs that we have used (Eclipse and NetBeans) were fully compatible with development of tools and programs for this server.

- **Java Graphical User Interface (GUI)**: Using NetBeans in order to implement the GUI of the Java Desktop client has simplified a lot the workload. For me was one of the first times of using this IDE in order to implement this kind of application, but I can say that it has been a successful experience and I would repeat this decision: luckily there are loads of gadgets to improve the obtained result, in spite of not having use them, due to time limitations.

- **Ubuntu Server, just an Operating System**: In its download page [39] it says that this OS is thin, lean, fast and powerful. That’s correct. We only
needed an Operating System, with the less services activated as possible. While installing Ubuntu server 8.04 TLS we realize that the selection done was the correct. We have to point out that it was very easy to integrate it with the programs that we assembled, and we have to mention that it behavior has been reliably and predictably. Just perfect for us.

- **Open Source software everywhere:** All is possible with open source solutions, or almost. In spite of having use VMware server or Install4j we have to say that all this project has been develop under open source and free software. Not to waste time thinking about license problems has been reported to us much more time to implement new functionalities and think better in what was important: our project.

- **VMWare Server 2.0, an amateur virtualization system:** Personally, I was excited about using Citrix but unluckily it did not support yet USB devices mapping to virtual machines. Instead of this virtualization OS we use VMWare Server 2.0. It has loads of restrictions because it is free, but we have to say that it works really good for the function that we wanted it. Summing up, it is amateur version of one of the most popular virtualization platforms, but enough for us.

### 7.2. Personal conclusions

When finishing the project it is time to summing up the personal feelings and experiences that I have undergo. In this subsection I am going to explain the most important topics that I have learn, as technical as personal ones.

- **Improve and reinforce program concepts:** During my degree I have programmed in different languages, mainly in .Net and Java. In the course of this project, I have improved my skills as Java programmer, discovering new ways of implementing ideas and polishing up the style of the generated code.

- **Discover and learn about XMPP protocols:** During my degree and my modest career I did not work with XMPP protocol. Discovering it and implementing the communication service throughout this technology has been result very interesting. XMPP is much more present as I expected when I started to read about it: important companies and social networks, such as Facebook or Google, use it frequently to solve some communication procedures.

- **Knowledge enlargement, Database and virtualization:** For me it has been the first time that I have been working with database, and it has result in a great experience in order to manage them. It is important to mention that the deployment of this kind of service has been done basically, but now I am encourage about thinking and deploying new personal ideas. It has been a good challenge to get started.
Other technology that I did not deal with was server virtualization. When deploying the project in the reduced server that we bought, we started to install different virtualization environments such as Citrix XenServer, VMWare Server or VMWare eSXi. Installing and testing different platforms involved around two weeks, but also my first experience working inside virtualization world in the server side.

- **Work in team group:** Since last year, I have been working with Jordi Coscolla designing and deploying this project. I have to point out that it has been one of the best academic experiences that I have undergone. Doing the project together has open-minded me, and give me more experience in managing timing and task scheduling.

- **Personal satisfaction:** Finally, I have to mention that I am very proud about the result of the whole project. The desired prototype runs properly, and when showing results to friends and family they are greatly surprised about it.

### 7.3. Project feedback

In this subsection we are going to evaluate if we have reach the objectives that when starting the thesis project were defined as requirements. They are going to be briefly explained in order to justify why we believe that we have accomplished them, or not.

It is important to mention that this feedback has been done when gathering together all the pieces developed by Jordi Coscolla and me, and the prototype was ready. To evaluate these requirements, different tests were done.

- **Functional requirements:** We have to mention that all the functional requirements defined inside system design and architecture chapter (Chapter 2) have been achieved in the final prototype.

  Summing up, we have made a system that can warn users about when the commercials on TV are over, with visual and sound signals. In addition, the system is able to differentiate between users, and create a log about what channels they have been watching. Moreover, this data is storage inside a database, in order to process it in future project extensions and create reports.

- **Non-functional requirements:** We have developed our first graphical desktop client over Java, making it compatible with all the platforms that permit the installation of JVM (Java Virtual Machine). In addition, different installers have been created in order to make the launch of the program easier for the final user and permit a multiplatform usage. Taking into account the different user tests, working with the developed client software is easy and simple.
About the response time of the client-server communication, we have to say it is enough for warning users about when the commercials on TV are over. In fact, the communications of these events are done in real time.

Thanks to XMPP protocol, and the adaption that we have done of the XEP-0045, the manner of broadcasting TV status changes has become efficiently, converting our system communications between the server side and the client one scalable. Push connections has let us avoid the risk of becoming bomb by user requests, reducing network consumption. About the availability, we have to mention that all the technologies that we have used can work in clustering mode. For this reason, and in spite of not having tested it, we can say that is possible to achieve a high availability.

In addition, it is important to point out that all the main components that we have installed and use in this project are open source. Getting this piece of software has been completely free, and we have to mention that workings with these components are not hard as some people think: community support is very useful and complete.

Summing up, we have to point out that all the project requirements have been accomplished.

7.4. Environmental project impact

During the deployment of this project, we have to mention that the environmental impact has been taken into account.

In the first development stage, while we were developing our first concepts proves and previous prototypes, we only use our personal computers, and the ecological print was negligible.

Later, and when testing the final prototype, in order to reducing the cost of having different machines working, as we have mention in last chapters, virtualization environment has been implemented. It reduced the power consumption cost, because inside one computer we have installed all the needed machines to carry out it. It is also important to mention that, when finishing prototype tests the server was shuttled down in order to avoid unnecessary CO₂ emissions.

Using XMPP protocol has to be also considered as a respectful measure to the environment. In fact, and taking into account Mar Pascual thesis [53], we have to mention that unnecessary traffic in mobile networks involves high battery consumption. With the push paradigm it is reduced, as well as the network resources and energy costs.

As telecommunication engineer we have to be aware that High-tech industries are also responsible of total global warming. A studio of Wissner-Gross found
that it takes on average about 20 milligrams of CO$_2$ per second to visit a Web site [54]. Taking into account the number of requests that are done daily into different web sites and the consumption of other web biased systems, as the one that we have designed; we have to point out that a little effort in order to make more efficient communications world would imply an important reduction of CO$_2$ emissions.

7.5. Future lines

It is important to mention that as Jordi Coscolla as me strongly believe on the project that we have developed. For this reason, we have thought about some modifications and topics that are pending to develop. Next they are going to be detailed.

- **System clustering:** In spite of the fact that all the technologies that we have use permit the clustering, it is important to mention that we do not have enough time to start clustering them. For this reason, we believe that a studio of clustering the whole system would be interesting in order to refute its scalability and availability.

- **Audit server security:** We have taken some security measures in order to protect our server and service from attacks. VPN and firewalls have been set in order to achieve it, but we have to mention that this security has to be improved in order to avoid possible system attacks. For this reason, a modest security auditory is going to be developed in future, just to make it safer.

- **Attractive interface of the desktop client:** In Chapter 5 we have show the first Java desktop client. The user interface of this piece of software is rude, but functional. For this reason, a future objective would be to modify this interface, and try to get one more attractive for final users.

- **Reporting obtained data:** In order to get information of the system data bases and process them, Jordi Coscolla Alvarez has started to develop a simple dashboard. It is one of the first versions, and has to be improved. This dashboard permits follow historic about commercial broadcasting on different TV channels and the number of viewers that were using our system. Next month, we are thinking in developing it properly, and starting to work with this information in order to have it ready, and report this data.

- **Suggestions to user:** When a user is watching a TV program and decides to use our application, is moment to suggest him other programs or products to buy. When registering, and as we have mention during this report, a unique UID is storage for this user. Making a studio about the TV consumptions would help us to customize the advertisements that we can introduce inside the application in the future.
BIBLIOGRAPHY


ANNEXES

TITLE: Real Time Commercial on TV Detector

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Antonio Ibáñez Luján

DIRECTOR: Roc Meseguer Pallares

DATE: March 30th 2011
ANNEX A. SCENARIO: PREVIOUS RESEARCH

Before developing the technical part of the project, we did a little investigation about if there is any system similar to it running in our country, or abroad, and the way which common people use media nowadays. This research is an essential part of the project, because it would define if the idea would have any possibility of success and diffusion between general public, or not, as well as the way to develop it.

For this reason, in this chapter we are going to deal with topics as the social scenario where we are living today, the hypothetical relevance that this system would have for people that is around us, the incidence of different media in Spain.

A.1. Spanish media scenario

Spain is a country that is located in the south of Europe, on the Iberian Peninsula. It is the ninth largest economy in the world, and 15th highest Human Development Index [4]. It belongs to the United Nations, European Union, NATO, OECD and WTO. This data shows that if we take into account the economical and social skills of this country, it is inside one of the most important developed nations of the world. But, what about the incidence of new technologies and the media use? Are we aware about how Spanish people use media?

At the beginnings of the last century, the concept of mass media was defined. This term represent the section of the media specifically designed to reach a large audience. Nowadays, it includes media as television, radio, newspapers, magazines or Internet. [5] For us, the most important would be the ones that have a direct competence with television, in the manner of uses: the radio.

The first step that we have done, in order to evaluate the status of these media is to search in the INE [6] data base the usage of television and radio. It would be important, mainly, because what we want to develop is a system that may avoid watching the commercials on TV. The results of this research are shown in Fig. A.1.
As we can see, the consumption of television is decreasing very slowly (2.2% in six years), in spite of the new platforms that provide the TV contents via Internet. Its direct competitor, the radio, is suffering a faster declination (4.8% in the same period).

But, what about internet connections? Has this media the same impact as radio or television? The answer is, definitively, no. As we can see in Fig. A.2 we are 10 point under the Euro Zone average, in spite of growing gradually in the last six years the difference is exactly the same.

This data has been extracted from INE [7] and only shows the incidence in number of domestic connections. We have also to point out an important data relative to connection quality. In Spain only a 2.8% of this connections are equals, or faster, than 5 Mbps [8]. In our country, the download speed average is 2624Kbps, while in other as Holland and Belgium the number of total connections faster than this 5 Mbps is around 30%, achieving 39% in Sweden.

The data that we have deal with is relevant in order to demonstrate that internet is becoming, gradually, very important, as well as the incidence of television do not seem to be in a clear declination process. But, is it enough? What about the phenomenon that consists on surfing in Internet while watching television?
EIAA [9] (*European Interactive Advertising Association*) is an organisation created for sellers that uses interactive and technological media. Its objective is to evaluate and quantify the impact of these new communication paths in order to introduce a product or service, and define what the best way to invest in advertising.

This organization develop a research called EIAA Media Multi-tasking Report [10] that provides very interesting information about the convergence of the use of television and internet at the same time. This research shows that almost a quarter of Europeans citizens (22%) use TV and internet simultaneously.

More than a half of multi-tasking users, while surfing internet and watching TV, enter into Instant Messaging platforms (Messenger, Gtalk and go on), as well as via social networks (Facebook or Twitter). Other activity that this kind of TV audience develops is to visit well known brands and price comparison websites.

We have to point out that the majority of European media multi-taskers are under 35 years old. In fact, the 25% of those are inside the youth category, or what is the same between 16 and 24 year olds. The 29% of this population are part of a group defined as 'Golden Youth' (24-35 year olds). Finally, 13% of media Multi-taskers are aged between 45 and 54 years old. Older users do not mesh the uses of media, but it seems that in few years this data would change.

All this Multi-tasking media information deals with data extracted from UK, France, Spain, Italy, Germany, Belgium and the Netherlands. It is important to remember that the number of Spanish domestic internet connections is below the ones that are in this countries, resulting in lower percentages. Anyway, the Spanish tendency is achieve European values, and it has to be taken as a prediction of what is going to take place in our region in few years.

Finally, we have to mention that in our country, and taking into account the last report of Multimedia Corporation Consultant of Spain [11], common people is watching television more than never before, reaching new audience levels and spending 14 minutes per day more than five years ago.

This given data is more relevant as it seems, because it do not include people that watch TV contents via internet and it is taken between September and June of each year, ignoring events as Olympic games, football World cups and similar events, which have very important audiences.

This research has reported that the most important factors that may produce this increment are the global situation of unemployment that Spain is suffering and the fact that digital television has arrived setting more free channels available: Anyway, watching television is one of the cheaper entertainments.

### A.2. Incidence of DTT in Spain

DDT (*Digital Terrestrial Television*) is a group of technologies designed to the transmission of television communications. Its name is because it is biased on
land based terrestrial signals [12]. The purpose of this technology is to provide best spectrum frequency use, better-quality picture and lower operating costs, compared to the analogue one.

Inside the different standards used to the integration of DTT, all the countries that belong to the European Union use DVB-T (Digital Video Broadcasting Terrestrial) [13]. This standard was created for the DVB, and can transmit audio, video and other data through a MPEG-2 codification. As we can see in Fig. A.3 most of countries that have set this technology uses this specification.

Since the 30th March of 2010, Spain switched totally from the analogical television broadcasting to DVB-T. It is important to mention that a general feature of this system is that the channels that involve this kind of transmission can be free or under payment. In any case, the Spanish law 7/2010 named “Ley General de la Comunicación Audiovisual” typifies that at least the 50% of the DTT spectrum has to be assigned to free channels.

Finally, we have to mention that it is very soon in order to analyze and quantify the incidence of this new technology in the television habits of Spanish people. This is because weather and other facts have a lot incidence in the audience levels, and we do not have enough data of different year to evaluate it. Anyway, taking into account different articles, it seems that the consumption of television is rising. [14]
ANNEX B. PROJECT GANTT DIAGRAM

In this section we are going to show the Gantt diagram of the project that we develop in order to deploy this thesis. These images are fully explained in Chapter 1 (Section 1.4: Timing and project organization).

It is important to mention that not all the tasks were done exactly at the time showed inside these figures, but almost. The difference between the final date expected (Fig B.3) and the current has been due to personal reasons.

---

![Fig. B.1 Project definition, Previous research and Design phases](image1)

![Fig. B.2 Deployment and Integration phases](image2)

![Fig. B.3 Reporting phase](image3)
ANNEX C. XMPP SERVERS AND LIBRARIES

In this Annex, we are going to include all the comparisons that we have used in order to choose the most suitable XMPP server and libraries for our project. The references that we have visit and read in order to create these table could be found inside Chapter 2 (Section 2.1: XMPP and extensions and Section 2.4 Desktop program: language and XMPP library).

C.1. XMPP Servers

Table C.1. Comparison between XMPP servers

<table>
<thead>
<tr>
<th>Server Name</th>
<th>Characteristics</th>
<th>Language</th>
<th>License</th>
</tr>
</thead>
</table>
| Ejabberd    | Admin Interface
Authentication via LDAD
Authentication via PAM
Data storage MS SQL
Data storage MySQL
Data storage PostgreSQL
SASL DIGEST-MD5
SASL PLAIN | Erlang | GPL |
| Jabberd14   | Authentication via Certificates
Data storage MySQL
Data storage File System
SASL DIGEST-MD5
SASL PLAIN
Server statistics | C | GPL |
| Jabberd2    | Authentication via LDAD
Authentication via PAM
Data storage LDAP
Data storage MySQL
Data storage Oracle
SASL DIGEST-MD5
SASL PLAIN
SASL ANONYMOUS | C | GPL |
| Openfire    | Admin Interface
Authentication via Certificates
Authentication via LDAD
Authentication via PAM
Authentication via RADIUS
Authentication via Kerberos
Data storage LDAP
Data storage Active Directory
Data storage MySQL
Data storage PostgreSQL
Data storage Oracle
SASL DIGEST-MD5
SASL PLAIN
SASL ANONYMOUS
Server statistics | Java | GPL |
| Tigase      | Data storage MS SQL
Data storage MySQL
Data storage PostgreSQL
Data storage File System
SASL DIGEST-MD5
SASL PLAIN
Server statistics | Java | GPL |
Table C.2. Supported XMPP extensions per server

<table>
<thead>
<tr>
<th>XEP number</th>
<th>Ejabberd</th>
<th>Jabberd14</th>
<th>Jabber2</th>
<th>Openfire</th>
<th>Tigase</th>
</tr>
</thead>
<tbody>
<tr>
<td>XEP-0004</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XEP-0012</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XEP-0013</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0016</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XEP-0020</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0030</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XEP-0033</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0045</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0047</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0048</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XEP-0049</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XEP-0050</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XEP-0054</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0059</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0060</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0065</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0066</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0077</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XEP-0079</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0085</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0092</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0095</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0096</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0100</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0106</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0114</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>XEP-0115</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0124</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0138</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0145</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0153</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0163</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0191</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XEP-0199</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0202</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0203</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>XEP-0206</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>TOTAL</td>
<td>19</td>
<td>15</td>
<td>15</td>
<td>29</td>
<td>21</td>
</tr>
</tbody>
</table>
C.2. XMPP libraries

**Table C.3. Supported XMPP extensions per library**

<table>
<thead>
<tr>
<th>Library</th>
<th>Characteristics</th>
<th>XEPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smack</td>
<td>API Documentation</td>
<td>XEP-0004, XEP-0012, XEP-0013, XEP-0016, XEP-0020, XEP-0030, XEP-0033, XEP-0045, XEP-0047, XEP-0048, XEP-0049, XEP-0050, XEP-0054, XEP-0066, XEP-0071, XEP-0077, XEP-0092, XEP-0095, XEP-0096, XEP-0106, XEP-0138, XEP-0153, XEP-0202, XEP-0203</td>
</tr>
<tr>
<td>Echomine Muse</td>
<td>API Documentation</td>
<td>XEP-0009, XEP-0045, XEP-0100, XEP-0108</td>
</tr>
<tr>
<td>Echomine Feridian</td>
<td>API Documentation</td>
<td>XEP-0004, XEP-0045, XEP-0100, XEP-0106, XEP-0114</td>
</tr>
</tbody>
</table>
ANNEX D. WEB TECHNOLOGIES

In this annex we are going to show the different technologies that we have deal with in order to create our connectors between different layers. CGIs, Fast-CGIs and Servlets basics are described here. This annex refers to the topics that we can find inside Chapter 2 (Section 2.2: Layer connectors: Java Servlets).

D.1. Common Gateway Interface

CGI (Common Gateway Interface) was one of the first practical techniques for creating dynamic content. With this technology a web server can passes certain requests to an external program, which output is then sent to the client in place of a static file [27].

The most important of CGI technology is that made it possible to implement all sorts of new functionality in web pages, quickly becoming a de facto standard, implemented on a high number of web servers.

It is interesting to mention that the ability of CGI programs to create dynamic web pages is a side effect of its intended purpose: to define a standard method for an information server to talk with external applications. This explains why CGI has one of the worst life cycles of dynamic web technologies. In figure Fig. D.1 the life cycle is represented.

![Fig. D.1 CGI life cycle](image)

As we can see, when a server receives a request that accesses a CGI program, it must create a new process to run the CGI program and pass to it all the information that might be necessary to generate a response. Creating process for every request requires time and significant server resources, which limits the number of requests a server can handle concurrently.

D.2. Fast-CGI

Open Market Company developed an alternative to standard CGI named FastCGI [28]. This technology works just like CGI, but the important difference is that creates a single persistent process for each FastCGI program. The life cycle is different to CGI, as is shown in Fig. D.2.
As we can see, Fast-CGI eliminates the need to create a new process for each request. Although it is a step in the right direction, it still has a problem with process proliferation: there is at least one process for each FastCGI program. If a FastCGI program is to handle concurrent requests, it needs a pool of processes, one per request. Another problem is that it does nothing to help the FastCGI program more closely interact with the server: FastCGI approach has not been implemented by some of the more popular servers.

D.3. Java Servlets

Java Servlet [29] is a small, pluggable extension to a server that enhances the server's functionality. This technology allows developers to extend and customize any Java-enabled server such as web servers, mail servers, application servers and others.

Servlets can be loaded dynamically, to expand the functionality of a server. They run inside a Java Virtual Machine (JVM) on the server (Fig. D.3), so it is safe and portable. Servlets operate solely within the domain of the server: they do not require support for Java in the web browser.

Servlets are all handled by separate threads within the web server process, unlike CGI and FastCGI. This means that servlets are also efficient and scalable. It is also important to mention that because servlets run within the
web server, they can interact very closely with the server to do things that are not possible with CGI scripts.

Another advantage of this technology is that they are portable across operating systems and also web servers: all of the major web servers support servlets.

Although servlets are most commonly used as a replacement for CGI scripts on a web server, they can extend any sort of server.
ANNEX E. XMPP ROOM CONFIGURATION

As we have mention inside Chapter 5 (Section 5.1.2 Room Configurations) Smack Java library has some objects and methods in order to fill it automatically. Thanks to it, we are able to create a program to create and configure them automatically.

Next we are going to explain the most important properties that we have configured, as well as the values that we have assigned to them.

- **Creating Formulary object**

  ```java
  Form form = cVar.chat.getConfigurationForm();
  Form submitForm = form.createAnswerForm();
  ```

- **AllowInvites**

  This feature defines whether non-admins users are allowed to invite other people to the room. Per all the rooms defined, we have set it as false, or in other words, we have disabled this option.

  ```java
  submitForm.setAnswer("muc#roomconfig_allowinvites", false);
  ```

- **ChangeSubject**

  This option reflects if a non-admin user is able to change the room subject. Change the room subject is the same as changing the room name. For our case, we have decided not to permit this kind of options.

  ```java
  submitForm.setAnswer("muc#roomconfig_changesubject", false);
  ```

- **EnableLogging**

  enablelogging option defines if the MUC service is going to store an archive of the discussions that occur in the room, typically to an HTML file or database. In our case we have disable it, but it would be a good option in order to make a log of keeping the changes in a channel status. It has been done because we have developed other systems that carry out this feature.
submitForm.setAnswer("muc#roomconfig_enablelogging", true);

- **MaxUsers**

  This option defines the maximum number of occupants per room. If the room has reached its maximum number, the service should deny access to the room and inform the user of the restriction. In our case, we have set it to “0”, or what is the same, max users have not been defined, this option has not been limited.

  ```java
  List<String> lista = new ArrayList<String>();
  lista.add("0");
  submitForm.setAnswer("muc#roomconfig_maxusers", lista);
  ```

- **MembersOnly**

  If the created room is members-only but the user is not on the member list, the service must deny access to this user. In our project we do not needed it: all the rooms are going to be open type.

  This option has been disabled because we cannot define the members that would join a room. This fact would depend on the success of the project and it would make impossible to manage the Openfire system.

  ```java
  submitForm.setAnswer("muc#roomconfig_membersonly", false);
  ```

- **ModeratedRoom**

  This option defines whether only users with voice are allowed to post messages to the room or anyone can post. This would determine if the room is moderated or not. In our case, we have to reach fully control in messages send to the room, and we have enable this option.

  ```java
  submitForm.setAnswer("muc#roomconfig_moderatedroom", true);
  ```

- **PersistentRoom**

  Persistent room field defines if the room would be destroyed when the last occupant left it. The antonymous of persistent room is temporary room.
In our case we have enabled the persistent room because each TV channel has one room and it would be never destroyed.

```java
submitForm.setAnswer("muc#roomconfig_persistentroom", true);
```

- **PresenceBroadcast**

  This option defines if the presence of users is going to be broadcasted to users. It means that when a user joins or leaves a room, all or some of the participants of the room would receive a notification warning about a new presence.

  For our project it has been disabled, because it is not relevant to users known if someone else is watching the same TV channel as him. In fact, neither moderator users are interested in receiving these notifications. Neglecting these messages would minimize the total traffic load of the system.

  ```java
  List<String> lista = new ArrayList<String>();
  lista.add("";
  lista.add("";
  lista.add("";
  submitForm.setAnswer("muc#roomconfig_presencebroadcast", lista);
  ```

- **PublicRoom**

  When user is discovering a service in order to discover different rooms, this option defines if the room that we are dealing with is going to be public for him, or would be hidden.

  In our project, we have defined all as public, because we are interested in providing users the access to all the channels that we have configured.

  ```java
  submitForm.setAnswer("muc#roomconfig_publicroom", true);
  ```

- **RoomAdmins**

  This field specifies who is going to be a room administrator. The JIDs of the users that would have this affiliation has to be included.

  In our case, we have included also a user that is going to kick other users from rooms. Its JID is “kickadmin@kvnspncr”. 
List<String> lista = new ArrayList<String>();
lista.add("kickadmin@kvnspncr");
submitForm.setAnswer("muc#roomconfig_roomadmins", lista);

- **RoomDesc**

This is a text that is going to provide a natural-language description of the room. When a user is looking for a room, this field is going to help him to understand what topics are discussing inside it.

```java
submitForm.setAnswer("muc#roomconfig_roomname", "Here is the channel 3 messages, to known when commercials are over");
```

- **RoomName**

This field specifies the name within the room is going to be named. In our project we have named as with the same name as the channel it would be treat inside.

```java
String channel = "TV3";
submitForm.setAnswer("muc#roomconfig_roomname", channel);
```

- **RoomOwner**

This option specifies the different owners that the room is going to have. In our case only “tvadmin@kvnspncr” JID has this kind of role.

```java
ArrayList<String> lista = new ArrayList<String>();
lista.add("tvadmin@kvnspncr");
sendForm.setAnswer("muc#roomconfig_roomowners", lista);
```

- **WhoIs**

This option defines if only moderator users, or anyone, are allowed to discover the real JID of the room occupants. This determines whether the room is anonymous or semi-anonymous type.

For our project, we have enabled this option and we have assigned these privileges for only two users: the one that is going to manage the room (tvadmin) and the one that is going to kick users (kickuser) from the room.
PasswordProtectedRoom

This option defines whether a user can enter without first providing the correct password to a room, or if it has to enter its password. In our case, we have disabled security, and no room secret has been defined.

submitForm.setAnswer("muc#roomconfig_passwordprotectedroom",false);
submitForm.setAnswer("muc#roomconfig_roomsecret", "");
ANNEX F. DATABASE TABLES

In this annex we are going to define the tables that have designed in order to keep information about users and channels current status, as well as their historical changes.

Their function is described in Chapter 4 (Section 4.3.1: User information and Section 4.3.2: Channel information), when describing Database usage. For this reason, the objectives of these tables are not going to be defined in this annex.

F.1. USER_INFO Table

```sql
CREATE TABLE "USER_INFO"
(
  "USERNAME" character(25) NOT NULL,
  "IP" character(20),
  "GPS" character(20),
  CONSTRAINT "USER_INFO_pkey" PRIMARY KEY ("USERNAME")
)
```

F.2. USER_HIST Table

```sql
CREATE TABLE "USER_HIST"
(
  "USERNAME" character(25),
  "CHANNEL" character(25),
  "TIME" timestamp with time zone
)
```

F.3. USER_REPORT Table

```sql
CREATE TABLE "USER_REPORT"
(
  "USERNAME" character(25),
  "CHANNEL" character(25),
  "TIME" timestamp with time zone
)
```
F.4. CHANNEL Table

CREATE TABLE "CHANNEL"
(  
"NAME" character(25),
"AD" boolean,
"ID" character varying(30) NOT NULL,
CONSTRAINT "ClavePrimaria" PRIMARY KEY ("ID")
)

F.5. CHANNEL_HIST Table

CREATE TABLE "CHANNEL_HIST"
(  
"TIMESTAMP" timestamp with time zone,
"CHANNEL" text,
"AD" Boolean
)
ANNEX G. JAVA DESKTOP CLIENT

In order to explain better the Java Desktop Client software developed that has been done to warn the user about when commercials are over, we are going to briefly explain and show all the windows designed.

The first object that is available is “Welcome Window” (Fig. G.1). This window show to the user information about the version that he has download and install, as well as license terms. In addition, and in order to make sure that the client is going to remember the site where he is able to download latest versions, we have included the URL of the site.

When the user of the client software piece pushes the “Start!” button, “Welcome” window is hidden and “Select TV channel” (Fig. G.2) is created and becomes visible. This object, when created, do a request to a servlet of the system who response with all the available TV channels. These channels are represented inside “Select TV channel Window”, and the client program waits for a user action.
When we execute the program in order to get the screenshots that we are showing five TV stations were available: Test, T5, Cuatro, TV3 and A3 (Fig. G.2). When the user decides the TV station in which he is interested, and pushes “Start detection” button, a “Status TV channel” window (Fig. G.3) is created, while hiding “Select TV channel” one. This new object is going to create a XMPP connection in order to log and join the IM (Instant Messaging) room of the selected channel, as well request to a servlet of the system about the status of this TV channel. This window object is going to show the status of the channel, and when a change took place in the selected TV station, XMPP connection is going to warn it, who is going to alert the user about this event.

We have to mention that there are three possible status that are going to be shown inside “Channel status” window. The first one is the one that we can see in the last picture (Fig. G.3). It denotes that in the selected TV station they are broadcasting commercials and you are able to other tasks (if you want). When this commercial time is over, the designed software rings and “Channel status” windows changes it warning message (Fig. G.4).

Finally, when the system confirms that it was not a detection error and a false alarm, users becomes disconnected from the system in order to save network resources. For warning about this situation “Channel status” window is going to show “disconnected” message (Fig. G.5).
It is also important to mention that we have included a system where you can report whether an detection error has taken place. In order to do it, a button inside “Client status” window has been created. It sends information about the TV station that the user was watching and shows a dialog (Fig. G.6).

---

**Fig. G.5** Client “Channel status” windows

**Fig. G.6** Client “Channel status” windows
ANNEX H. PROTOTYPE DEPLOYMENT

In this annex we are going to detail some technical information about the prototype development that we have done. The most important details of the software and hardware used for reach this objective is going to be briefly described. This annex is directly connected to Chapter 6 (Prototype integration environment).

H.1. HP Proliant ML 115 G5

Table H.1. HP ProLiant ML 115 G5 main features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of processors</td>
<td>1</td>
</tr>
<tr>
<td>Processor</td>
<td>Quad Core AMD Opteron Model 1354 2.2 GHz, 2 MB</td>
</tr>
<tr>
<td>RAM</td>
<td>6 GB (Installed)</td>
</tr>
<tr>
<td></td>
<td>8 GB (Maximum supported)</td>
</tr>
<tr>
<td>Redundant power supply</td>
<td>Not available</td>
</tr>
<tr>
<td>USB Ports</td>
<td>8 Total</td>
</tr>
<tr>
<td></td>
<td>(4 rear, 2 front panel, 2 internal)</td>
</tr>
<tr>
<td>Networking</td>
<td>Embedded NC105i Express Gigabit Ethernet Server Adapter</td>
</tr>
<tr>
<td>Hard Disk</td>
<td>160 GB (Installed)</td>
</tr>
</tbody>
</table>

Fig. H.1 HP ProLiant ML115 G5
H.2. Server costs

Table H.2. Hardware project costs

<table>
<thead>
<tr>
<th>Item</th>
<th>Units</th>
<th>Unitary Cost</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>1</td>
<td>400 €</td>
<td>400 €</td>
</tr>
<tr>
<td>RAM 2 GB</td>
<td>3</td>
<td>50 €</td>
<td>150 €</td>
</tr>
<tr>
<td>DVB-T PC Card</td>
<td>6</td>
<td>35 €</td>
<td>210 €</td>
</tr>
<tr>
<td>TV connectors</td>
<td>-</td>
<td>20 €</td>
<td>20 €</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td></td>
<td><strong>780 €</strong></td>
</tr>
</tbody>
</table>

H.3. Virtualization environment

H.3.1. Citrix XenServer

Citrix XenServer is a free and complete, managed server virtualization platform built on the powerful Xen hypervisor. This technology is widely acknowledged as the fastest and most secure virtualization software in the industry [48].

It is important to mention that hypervisor, also called virtual machine monitor (VMM), is a virtualization technique which are installed on server hardware, and whose only task is to run guest operating systems. Non-hypervisor virtualization systems are used for similar tasks on dedicated server hardware, but also commonly on desktop, portable and even handheld computers. Hypervisor technology is also known hardware virtualization.

H.3.2. VMWare eSXi 4.0

VMware ESX and VMware ESXi are bare-metal embedded hypervisors. These pieces of software designed by VMware's enterprise software run directly on server hardware without requiring an additional underlying operating system [49].

VMware ESXi is a smaller footprint version of ESX that does not include ESX's Service Console. It is available as a free download from VMware though certain features are disabled without the purchase of a vCenter license. It is important to mention that although it is free of charge, the license of this software is proprietary.
H.3.3. VMWare Sever

VMware Server is an entry-level virtualization-software server suite developed and promoted by VMware, Inc. It is not a hypervisor, but supervisor. This means that is a part of an operating system that controls the execution of virtual machines [50].

VMware Server 2 uses a web-based user-interface, the "VMware Infrastructure Web Access". This product has a free license, but closed source.

On January 2010, VMware Server was declared end of availability, and it general support will end on June 30, 2011.
### ANNEX I. TV CONSUMPTION SURVEY

**Encuesta sobre consumo de televisión** / Enquesta sobre el consum de televisió

<table>
<thead>
<tr>
<th>Edad:</th>
<th>Sexo:</th>
</tr>
</thead>
</table>

1) **¿Cuántas horas diarias dedicas al consumo de televisión?**  
**Quantes hores diaries dediques al consum de televisió?**

- De 0 a 2 horas
- De 2 a 4 horas
- De 4 a 6 horas
- Más de 6 horas
- NS/NC

2) **¿En qué franja horaria acostumbras a realizar este consumo televisivo?**  
**A quina franja horaria acostumes a realitzar aquest consum televisiu?**

- Matí (8h a 13h)
- Mig dia (13h a 16h)
- Tarda (16h a 19h)
- Vespre (19h a 22h)
- Nit (22h a 08h)
- NS/NC

3) **¿Aprovechas las pausas publicitarias para realizar zapping o ausentarte?**  
**Aprofites les pauses publicitaries per realitzar zapping o ausentar-te?**

- Sí
- No
- NS/Nc

4) **¿Encuentras abusivos los espacios dedicados a publicidad? En caso afirmativo, indica en qué aspecto.**  
**Trobes abusius els espais dedicats a la publicitat? En cas afirmatiu, indica en quin aspecte.**

- Sí
- No
- NS/Nc

<table>
<thead>
<tr>
<th>Aspecto / Aspecte:</th>
<th>Duración</th>
<th>Frecuencia</th>
<th>Otros</th>
<th>..........................</th>
</tr>
</thead>
</table>

5) **¿Dispones de ordenador con conexión a internet en tu domicilio o vivienda?**  
**Disposes d’ordinador amb connexió a Internet al teu domicili o habitatge?**
6) ¿Eres propietario/usuario de un teléfono móvil de tercera generación con conexión a Internet?
Ets propietari/usuari d’un telèfon mòbil de tercera generació amb connexió a internet?
- Sí
- No
- Ns/Nc

7) ¿Estarías interesado en un sistema que te avisase sobre el final de periodos publicitarios en el canal que estás viendo?
Estaries interessat en un sistema que t’avises de quan han acabat els anuncis publicitaris al canal que estàs veient?
- Sí
- No
- Ns/Nc

8) ¿Realizas algún tipo de actividad con tu teléfono móvil u ordenador personal mientras ves la televisión? En caso afirmativo, indica qué dispositivo utilizas con más frecuencia.
Realitzes algun tipus d’activitat amb el teu telefon móvil o ordenador personal mentre veus a la televisió? En cas afirmatiu, indica quin dispositiu utilitzes amb més freqüència.
- Sí
- No
- Ns/Nc

Dispositivo: ........................................

9) ¿Eres miembro de alguna red social como Facebook o Tuenti? En caso afirmativo, ¿Con qué frecuencia las utilizas?
Ets membre d’alguna xarxa social com Facebook o Tuenti? En cas afirmatiu, amb quina freqüència les utilitzes?
- Diariamente
- Semanalmente
- Mensualmente
- No utilizo redes sociales
- NS/NC

Muchas gracias por tu colaboración
Moltes gràcies per la teva col·laboració
TV consumption survey responses

Fig. I.1 Response results of question 1

Fig. I.2 Response results of question 2

Fig. I.3 Response results of question 3
Fig. I.4 Response results of question 4

Fig. I.5 Response results of question 5

Fig. I.6 Response results of question 6
Fig. I.7 Response results of question 7

Fig. I.8 Response results of question 8

Fig. I.9 Response results of question 9
ANNEX J. APPLICATION USAGE SURVEY

Encuesta sobre el uso de la aplicación / Enquesta sobre la utilització de l’aplicació

Edad: Sex: 

1) ¿Qué te ha parecido la interface gráfica del programa?
   Què t’ha semblat la interfície gràfica del programa?

2) ¿Te ha parecido complicada la aplicación? ¿Te ha costado mucho utilizarla por primera vez?
   T’ha semblat complicat l’ús de l’aplicació? T’ha costat molt utilitzar-la per primera vegada?

3) ¿Te ha parecido rápida la aplicación? ¿Se te ha colgado mientras la utilizabas?
   T’ha semblat ràpida l’aplicació? Se t’ha penjat mentre la feies servir?

4) ¿Te ha parecido útil la aplicación?
   T’ha semblat útil l’aplicació?

5) Otros comentarios.
   Altres comentaris.

Muchas gracias por tu colaboración
Moltes gràcies per la teva col·laboració
Application usage survey responses

Fig. J.1 Response results of question 1

Fig. J.2 Response results of question 2

Fig. J.3 Response results of question 3
Fig. J.4 Response results of question 4