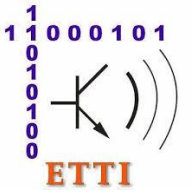
UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF ELECTRONICS, TELECOMMUNCATIONS AND INFORMATION TECHNOLOGY
TECHNOLOGY AND COMMUNICATION SYSTEMS DEPARTMENT



DIPLOMA PROJECT
Multimedia Storage and Streaming Web-service

Thesis supervisor:
Prof. PhD. Eng. Corneliu Burileanu
PhD. Eng. Horia Cucu

Author:
Bogdan Alexandru Zlate

BUCHAREST

2015

Annex 1bis

University "Politehnica" of Bucharest
Faculty of Electronics, Telefcommunications and Information Technology
Department: Electronics and Telecommunications / Technology and Communications Systems

Department Director's Approval :

Prof. Dr. Ing. Silviu Ciochină

DIPLOMA THESIS
of student Zlate C. Bogdan Alexandru, 441G

1. Thesis title: **Multimedia Storage and Streaming Web-service**

2. The student's original contribution will consist of (not including the documentation part): *describe in 15..20 rows:*

Design and implementation of a software program performing:

- *client-server media transmission over the internet, or at least over an intranet.*
- *client can access and read any media file in a specified directory located on server (play music, watch a movie, look at some photos).*
- *communication between client and server over HTTP*
- *media transmission over RTP*

Implementing in Java a set of frameworks to facilitate the communcation between server and client. These frameworks facilitate the communication between server and client:

- *JAX framework for directories and files structure on the server. They are transmitted as an XML over HTTP to the client.*
- *Java Media Framework to be able to manipulate some media file formats, including RTP media transmission*
- *JVLC API so the client is able to play the media files*
- *Jersey API to facilitate the communcation over HTTP between client and server*

3. The project is based on knowledge mainly from the following 3-4 courses:
Object Oriented Programming, Computer Programming and Data Structures and Algorithms

4. The Intellectual Property upon the project belongs to: *UPB and student*

5. The research is performed at the following location: *UPB*

6. The thesis project was issued at the date: 12.10.2014

Thesis Advisor: *Ș.l. Dr. Ing. Horia Cucu*

STUDENT: *Bogdan Zlate*

Annex 5 bis

Statement of Academic Honesty

I hereby declare that the thesis "Multimedia streaming and storage web-service", submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of Engineer in the domain of Electronics and Telecommunications, study program *Technology and Communications Systems* is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited "as is" or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations and measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations and measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, July, 03, 2015

Bogdan Alexandru Zlate

(student's signature)

TABLE OF CONTENTS

LIST of ACRONYMS

A

ABAP – Advanced Business Application Programming
API – Application Programming Interface
AWT – Abstract Window Toolkit

C

CDT – C/C++ Development Tooling
COBOL – Common Business Oriented Language
CSS – Cascading Style Sheets

D

DBMS – Database Management System
DCCP – Datagram Congestion Control Protocol
DFS – Depth-First Search
DOM – Document Object Model

G

GIF – Graphics Interchange Format
GNU – GNU Not Unix
GUI – Graphic User Interface

H

HTML – HyperText Markup Language
HTTP – HyperText Transfer Protocol

I

IDE – Integrated Development Environment

J

J2SE – Java Standard Edition
JAR – Java Archive
JAXB – Java Architecture for XML Binding
JDBC – Java Database Connectivity
JDT – Java Development Tools
JFC – Java Foundation Classes
JIT – Just in Time
JPEG – Joint Photographic Experts Group
JSON – JavaScript Object Notation
JVM – Java Virtual Machine

M

MMU – Memory Management Unit
MVC – Model View Controller

O

OOP – Orientet Object Programming
OSI – Open Systems Interconnection

P

PDT – PHP Development Tools
PHP – Php: Hypertext Preprocessor

R

REST – Representational State Transfer
RTCP – Real-time Transport Control Protocol
RTP - Real-time Transport Protocol
RTSP – Real Time Streaming Protocol

S

SDK – Standard Development Kit
STP – Set-Top-Box (project)
SOAP – Simple Object Access Protocol
SQL – Structured Query Language
SVG – Scalable Vector Graphics

U

UDP – User Datagram Protocol
UI – User Interface
URI – Uniform Resource Identifier
URL – Uniform Resource Locator

V

VLC – VideoLan
VLCJ – VideoLan for Java
VM – Virtual Machine

X

XHTML – Extensible HyperText Markup Language
XML – Extensible Markup Language
XUL – XML User Interface Language

W

W3C – World Wide Web Consortium
WAR – Web Archive
WWW – Wolrd Wide Web

LIST of FIGURES

1. INTRODUCTION

The data transfer is a problem which interested the human kind since the antiquity and even before. Nowadays, the data transfer rate is so high that simple information, as text, overpasses thousands of times the human being understanding rate. Users can transfer in a single second more than they could read in an entire day. This is why information interchange evolved to span a wide range of platforms and formats.

Today, the visual information interchange is a problem with a solution in development. The ideal case would be to transfer video information, with a higher clarity than the eye perceives, between any two end points all over the world and even outside it.

The main advantage of our day to day technology is that we have a tremendous data transfer speed. This high information flow available for everyone decreases the amount of stored information. If one knows that he can get something very easy, he will have not the interest of keeping that thing for a future use too. This is also available when we speak about information. If the data speed increases, the interest in storing data decreases.

This thesis approaches the idea of high speed and low interest in storing, as it is nowadays.

The video data is the most complex current method of information interchange. It started once with the analog television, but this technology has the drawback that the users can only receive the information transmitted by an expensive source, with designated bandwidths. Now, with the help of high speed internet, the video data evolved as video streaming and any point can be a source, so the users can transmit too.

The video streaming is, in fact, compressed content sent over the internet and displayed to the user in real time. With this service, of media streaming, a user is not compelled to download a file, as total video content, and then play locally. Everything that the user needs is a player that can decompress the data, and play it. In this action there is not needed any kind of data storage.

The streaming, real time applications are probably unlimited. Some of the most common are:

- video-conference, when the physical presence is not compelled because each member has a device that films and sends it to all the others.
- security, there are designated video filming systems that capture every moving object in a well defined area and then stream this information to a server
- video-calling which let two or more users to not only speak to each other, but see them in real time too.
- screen capture, which let two users see the personal computer one to the other and eventually help each other with different computer related tasks.
- on demand live stream, when a user can access a media file and play it, in real time, without storing anything on his PC.

The last live streaming application is the object of the current thesis. The most important product that provides to a common user this service is YouTube. There, each user can upload a media file and play it whenever he wants, his own or the file of another user. The drawback of this well-known web platform is when a user want to see a high definition movie of two hours, or when a user want to listen his own music playlist, without any video content.

Multimedia storage and streaming web-service suppresses these both disadvantages. The user may choose any video file and he will receive video streaming, with the minimum necessary performances, without having a web browser which consumes resources. Also, when the user may want to listen to music, the streaming will provide only audio content, so the service will need a smaller data rate on both ends.

This thesis is organized in three chapters and two more for introduction and conclusions, which will be briefly described as follows:

- Introduction: presents general information about streaming, evolution, applications. It

also provides a brief comparison between this project and the most important existing product.

- Chapter 1: describes the main technologies and programming languages needed in order to develop the current project.
- Chapter 2: presents the development tools needed in
- Chapter 3: describes the application functioning and its architecture
- Conclusions: summaries the current thesis, presents disadvantages, propose improvements and future development.

2. USED TECHNOLOGY

Multimedia storage and streaming web-service is an application based on server – client architecture. The chosen programming language for its development is Java, for both, the client and the server. In order to have a reliable communication, it was used a set of already existing languages, frameworks and standards which will be presented in details in the next sections. Having in mind that the server has complex communication mechanism, using a set of standards, such as XML, HTTP and RTP it must use existing stable frameworks. Also, the need of a web interface made the set of needed technology to increase. So, in the following sections, it will be provided details about the most significant used technology.

2.1. Java

Java is a programming language created by enterprise Sun Microsystems at the beginning of the 90's. It is an object-oriented language, which means it is oriented to something well determined, like molecules, a circle, a triangle, a square etc. These objects have certain characteristics that are known in the programming domain as attributes. An object can create operations known as methods[1].

Java is a high level, very powerful and versatile language that can be executed over any operating system, from Solaris, Linux, Windows to others and it is used to create phone applications, games in general. Java is also allowing games not to take up a lot of memory from cellular devices that lead to compatible and faster applications. In general this programming language is updating every month and on Sun's website on the internet there is an API (application programming interface) where developers from all over the world may consult how to use certain methods or how to realize operations according to their requirements.

Java is an international successful programming language that had solved a lot of programmer's problems, achieving its main **objective**[2]. It has been created to improve the productivity, to be practical and to bring a lot of benefits to programmers. For example, Java is based on libraries that allow programs to run faster. A programmer may use only the libraries that contain the necessary functions to speed up the process.

2.1.1. History of Java language

Java was designed by James Gosling from Sun Microsystems in 1990 as electronic devices software, as calculator, microwave etc.

The three fundamentals reasons of Java creation were:

- It increased the necessity of a more user friendly and intuitive interface comparing with the existing windows systems that were not achieving the expectation at that moment.
- To ease the development and code reliability. The Java creator, James Gosling observed a lot of characteristics from other programming languages, as C or C++ that were having a high testing and development cost. He focused on a programming language that could solve others languages issues.
- Electronic controller's diversity. The electronic devices are using cheap microprocessors, which are frequently changed and used with instruction sets. Java

[1]Thomas Stamford Raffles, *The History of Java*, J. Murray publisher, vol. 2, 2005, pp. 23-39.
[2]http://mathbits.com/MathBits/Java/Introduction/BriefHistory.htm

permits to write a common code for all the devices[3].

Java's technology has been created as a programming tool to be used in the set-top-box project: STB. This project has been used in the denominated operation "The Green Project" in Sun Microsystems in 1991. The Green team was formed by 3 people and coordinated by James Gosling, who worked 18 months at his development.

Figure 2.1 - Sun Microsystems Logo

Source: www.sun.com

The language from this project was named Oak, but for being a trademark it has been denominated Green and at the end it was renamed Java.

Java's term has been chosen from a cafeteria where Green team used to spend time. It is not clear if it is an acronym or not, even if it can be one from the names of the creators: James Gosling, Arthur Van Hoff and Andy Bachtolsheim.

Figure 2.2 - Java's logo

Source: www.java.com

Gosling's objective was to implement a virtual machine and a language with structure and syntaxes similar to C++. Between June and July 1994, after a three days brainstorming session, James Gosling, Joy Naughton, Wayne Rosing and Eric Schmidt reoriented the platform to the web. They felt that the release of the Web Mosaic browser would make the Internet an interactive environment. Then, Naughton created a prototype browser WebRunner, later known as HotJava.

Java 1.0a could have been downloaded for the first time in 1994, but Java and HotJava had to wait till 23th of May 1995 to be officially released. Gosling first promise was to Write Once, Run Anywhere had been reflected into the platform Java Virtual Machine.

2.1.2. Java language characteristics

In Java, the systems are easier to express and understand. When you write a code, you write in a direct way a solution to your problem. In Java the errors are managed by using exceptions. These exceptions are notifying the errors from the code and by using them it become easier to control complex programs.

Comparing this new language with previous languages that were handling long codes in a complex way and were less effective, thanks to Java these limits had disappeared. When somebody wants to write a program that can show on a screen the following text: "I like chocolate", there is no need to use OOP (Object-oriented programming). The characteristics will always remain the same when the programmer will require[4].

Despite of being practical, Java allows programmers to skip the meticulous and boring task

[3] http://www.freejavaguide.com/history.html

[4] Herbert Schildt , *Java: A Beginner's Guide*, Mcgraw-Hill Osborne Media publisher, 6 edition, 2014, pp. 56-80.

to rewrite an existing code in the program, by generating objects to interact with. In java classes are created, and based on them, it is possible to create objects. When we create a new class and merge it with another one that we already had, we can create another one and this can happen successively, permitting us to make changes to the derivate classes, known as child classes or to make changes to the creator classes, known as parent classes. In Java this is similar to a genealogical tree and the process is known as inheritance.

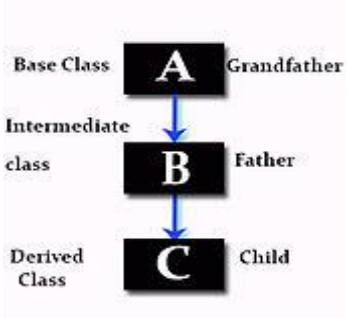


Figure 2.3 - Multilevel Inheritance

Source: https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

Java is multithreading, that means it can execute a lot of processes at the same time, thus, while a thread is in charge of the communication process, another one can interact with the user, and at the same time, another thread is presenting an animation on the screen and another is realizing the mathematical computations.

Java can generate two types of programs: applets and independent applications that are acting as any other program. Applets are little programs that appear on the web pages, like graphics or texts are, but they have the capacity to execute complex actions, like animating images, establishing web connections, presenting menus and chat windows.

To sum up, the most important characteristics of Java languages are the below:

- Java is simple
- Java is object-oriented
- Java is distributed and interpreted
- Java is secure and dynamic
- Java is architectural-neutral
- Java is portable and multithreading[5]

2.1.3. Java Virtual Machine

A Java Virtual Machine is a native process virtual machine that means it is executable in a specific platform and it is able to read and execute a special binary code that is generated by the Java's compiler.

Java's binary code is not a high level language, quite the contrary, it is a real low level coding machine, viable, even as an entry language for a physical microprocessor. Like all the java's puzzle pieces, it has been developed by Sun[6].

JVM is a fundamental part of Java's platform. Basically it is situated on a high level than the system hardware that executes the application and it acts like a bridge that understands the bytecode. When an application is written in a Java application, it will be executed by a virtual machine that will be the last instance that converts the bytecode into a native code for the final device. The main advantage of this machine is that it brings portability to the language and because of this, Sun Microsystems created different virtual machines for different architectures. This way, a

[5] http://www.javatpoint.com/features-of-java
[6] http://java-virtual-machine.net/download.html

.class program written in Windows will be able to be read in a Linux environment. It exists a Java famous axiom that describes this process: “Write once, run anywhere”.

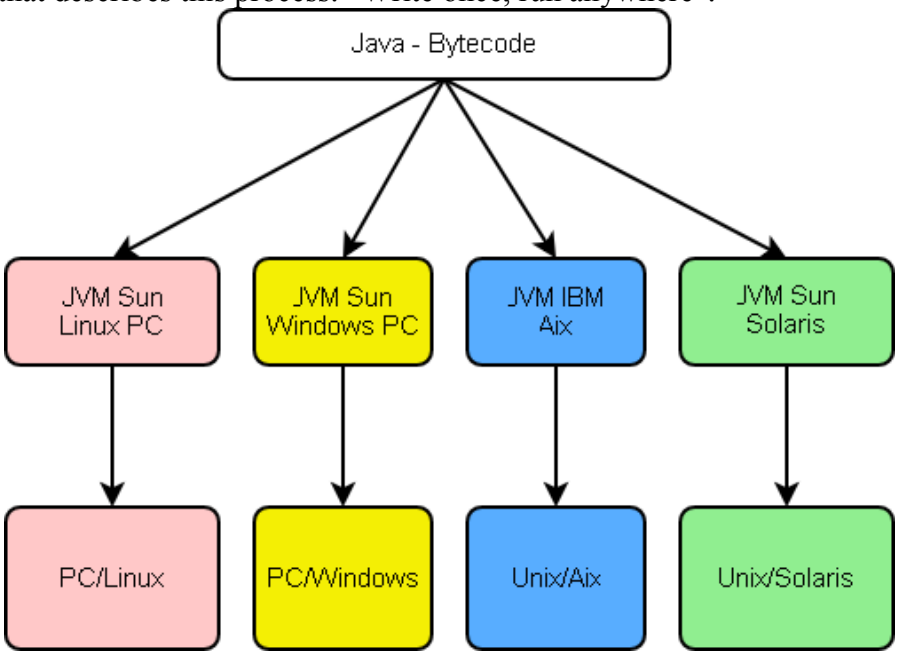


Figure 2.4 - General Architecture of a running program

Source: http://videos.web-03.net/artigos/Allan_Romanato/JavaVirtualMachine/JavaVirtualMachine2.jpg

In order to execute an application in a Java Virtual Machine, the code program should be compiled in according with a standard binary format, normally as a .class file. A program can be formed by multiple classes and each one should have its own archive .class. To facilitate the application distribution, the classes archive can be packed together in a .jar archive. A package can avoid overloading and can close the connections for each necessary fragment.

The resulted code from the compilation it is executed by JVM that carries out the emulation of the set of instructions, usually using a JIT (Just in time) compiler, like HotSpot from Sun. This last option converts the bytecode into a native code and increases the execution speed. The inconvenient is the initial necessary time for the compilation[7]. JVM verifies the entire bytecode before to execute it. It means that only a limited quantity of sequences of bytecode can form valid programs. For example, a JUMP instruction can point only a statement within the same function.

Code verification ensures that arbitrary bit patterns cannot be used as addresses. Memory protection is achieved without requiring a memory management unit (MMU). Thus, JVM is an efficient way to obtain protection memory chips even it does not have MMU.

Java Virtual Machine has instructions for the following groups of tasks:

- Loading and saving
- Arithmetic
- Conversion rates
- Creating and manipulating objects
- Battery Management (push and pop)
- Transfers control (branching)
- Invocation and return to methods
- Exceptions[8]

[7] http://www.javaworld.com/article/2077184/core-java/the-lean--mean--virtual-machine.html
[8] https://www.jcp.org/en/jsr/detail?id=45

The key is the binary compatibility. Each operating system of a particular host needs its own implementation of JVM and runtime. These JVM are interpreting the bytecode in the same way, but the actual implementation may vary. It is more complicated because only the emulation of bytecode is compatible and efficient implementation of the Java API, which must be mapped to each host operating system.

Virtual machine architecture allows fine-grained control over the actions that the code can do in the machine. This is designed to allow safe execution of untrusted code from remote sources; a famous used model is: Java Applets. Applets are executed in a virtual machine incorporated in a user's browser, executing code downloaded from a remote HTTP server. The remote code runs in a "sandbox" highly restricted, which is designed to protect the wearer from erroneous or malicious code.

The J2SE Edition has two implementations of the virtual machine:

- Java HotSpot Client VM: The default virtual machine, ready for maximum performance in applications running in the client environment, for example, minimizing startup time of a Java application.
- Java HotSpot Server VM: Ready for peak performance in the execution of applications on the server[9].

2.2. Java Swing

The Java Swing Library is a part of **JFC** (Java Foundation Classes) which offers the necessary components used to design modern graphics interfaces. Swing brings up a lot of improvements to the old AWT (Abstract Window Toolkit) package. As an example, in the old AWT for Java 1.0, the programmer could use only four fonts and a limited number of graphical components, unfortunately insufficient for designing complex interfaces. Even though, the Swing packages are not completely separated from the AWT packages, but it is constructed on top of it. AWT is now used as a middle-ware between Swing and the Operating System which runs the Java platform.

Through the improvements brought by Swing, despite the new added components, we can enumerate the behavior and presentation mode for a platform independent interface, the possibility to have rectangular form components etc. This package was thought so that for simple things there is a little code needed and for complex things there is needed much more code. Also, the naming of classes and methods are self-descriptive and logical.

[9] http://www.oracle.com/technetwork/java/index.html

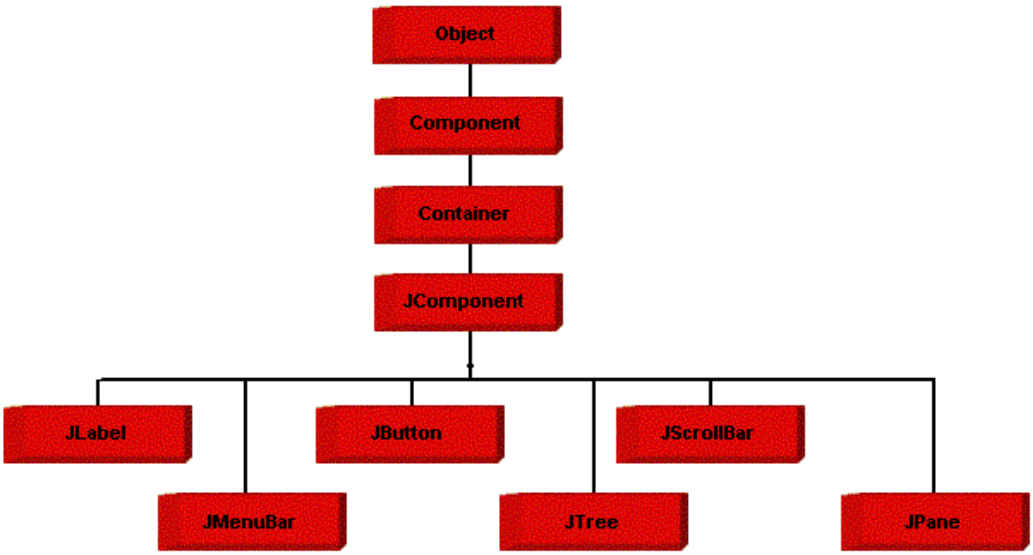


Figure 2.5 - Java Swing hierarchy

Source: http://www.wideskills.com/java-tutorial/java-swing-tutorial

2.2.1. Graphic interfaces

The User Interface (UI), represent the whole set of mechanisms which permits the interactions between the user and the application. A particularity of user interface is represented by the Graphic User Interface (GUI), which means strictly the visual communication between the user and the application.

Usually, the applications are seen by the final users only through their interfaces, so a poor interface may compromise the entire application. In this case, a problem in the developing process of a graphic interface is the possibility that the user couldn't understand how to use it.

The application developers must foresee the user's conceptual models of application functionalities and they must incorporate these models in the graphic interface. The method used in this process is the concretized analogy which represents the conventions between the developers and the users. For example, the deletion function can be easily kept in mind as an object thrown in a bin. So, in order to delete an item, the user pulls that item over the bin and then he releases it.

The Sun Company, which describes the specifications of the Java language, suggests the using of a standard set of icons, grouped in an archive at the address http://developer.java.sun.com/developer/techDocs/bi/repository/.

From the point of view of programming engineering, the moment in which the problem of user interfaces arises is present in the step of architectural designing of the application. As there is created a blueprint of the product, it is recommended to be designed a schema of user interface.

2.2.2. Swing library components

Almost all Swing components extend a single base class named JComponent, which inherits the Container class from AWT. This extension provides these components the possibility to have panels, tooltips, and a configurable look-and-feel. Also, by this extension there are provided the methods through which each component may be sized and positioned.

A graphic component represents an object with a well-defined graphical representation which can be displayed on the screen so that the user may interact with it. Graphical representations correspond to any instances of the classes which inherit javax.awt.Component or javax.swing.JComponent.

2.2.3. Events and listeners

The Java graphic user interface interacts with the user through different actions which are simply reduced to the so called events and listeners. Any command performed by the user in form of clicks or keys pressed are seen by Java as an event and the element that should handle a corresponding event is called a listener. In this way, any event triggered is passed to its corresponding listener which performs the programmed tasks. This type user interaction handling is applied for Swing and AWT too. For both main user interface java packages all events are subclasses of EventObject, so all events are derived from this main class.

The events are handled by an event-listener method which has as a single argument, the respective event. An important aspect that must be kept in mind is that an event listener method must be as fast as possible because a repainting operation may seem slow or unresponsive. Considering this, one must not have a complicated not optimized code in the listener method. If that is mandatory, then the program should start a new thread or at least, call a new thread to perform some of the instructions so the final user is not deranged at all by the application. More about Java threads will be discussed in the next section.

The events can be split in two categories: low-level and semantic events. The first type consists in mouse and key events which are exclusively user inputs. The second type represents the cases when a Java graphic component is designed to receive certain user inputs such as button clicks, key pressed in text fields and others. It is recommended to listen, whenever possible, the second type of events because this increases the application portability and the efficiency in terms of time.

2.2.4. Java threads

Execution threads, multithreading or concurrency are an important aspect in any programming language. These so called threads are related to multitasking. We know that all final users use to multitask. They open as many applications as they want at the operating system level, but also, in a single application, they may want to do multiple things at a time. Multitasking is only an implication of multithreading, for example, in the case of a player, there is a small number of user commands, but the application perform the media signal processing, playing it and can wait for user inputs too, at the same time.

In programming, the concurrency is split in two basic concepts: processes and threads. Java is mostly concerned with the second ones. They are sometimes called lightweight processes. Any process has at least one thread. Processes share their resources, memory and open files with threads in order to increase its efficiency, but also the number of problems which may appear in their communication.

Multithreading is an essential Java feature. Each application needs at least a thread to run, without counting the system threads which take care of memory management or signal handling.

In Swing, there are three kinds of threads:

- Initial threads, which represent the starting point of an application. In standard applications there is only a single initial thread, which is the one that invokes the main method. In applets, initial threads invoke the init and start methods.
- Event dispatch, which has the job of event handling. Also, in this thread are run most of the Swing methods.
- Worker threads, which run long-running tasks. They are also known as background threads.

One of the main potential problems that may occur when using threads is the memory consistency error. This happens when two threads try to access the same part of memory, in form of

an object or a primitive. In order to avoid this, the programmer should always know when a variable may be accessed by a thread or by another and he must order the thread execution in a convenient way. If the order of variable access is uncertain, then a synchronized block must be used.

2.3. Representational State Transfer

REST is the acronym of REpresentational State Transfer and represents an architectural model for web services creation. It is an analytical (post hoc) description of the existing web architecture. REST describes a resource oriented architecture. This idea was presented for the first time in Roy's Fielding's PhD thesis. The motivation of this architecture is started from the fact that the RPC services and the others SOAP based bring up with them a wide complexity. The WEB simplicity represented the base idea of REST web-services. The second one uses all the components which made the web a world-wide success. REST applies the web architecture over web services. Even if REST is not a standard, it uses standards:

- HTTP (HyperText Transfer Protocol)
- URI (Uniform Resource Identifier)
- XML/HTML/GIF/JPEG

In other words, REST represents the building of a web service, using HTTP, XML and URI exactly as the web was built.

Here, there is Roy Fielding's explanation regarding the meaning of Representational State Transfer:

"Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."[10]

In REST, the main idea is that anything is a resource. Each resource has an address, known as Uniform Resource Identifier (URI). In order to access a resource, one must use a HTTP method. A huge advantage of this architecture is that the first line of a HTTP request contains all the necessary information to access a resource ("GET /example/index HTTP/1.1").

2.3.1. REST Characteristics

- **Resources** – an important idea of REST is the existence of resources. Usually, a resource is an entity which may be stored on a computer and can be represented with a string of bits. Here, each resource has an globally identifier, known as URI. In order to manipulate the resources, the network components (clients and servers) communicate through a standardized interface, protocols, in this case HTTP and they change resources representations.
- **URI** – each resource must have at least one URI, but two resources cannot be accessed through the same identifier. URI is the name and the address of a resource. If there is information which does not have an URI, then it is not a resource. It is recommended that the uniform resource identifiers to be self-descriptive. Also, their structure must vary in a predictable way because they are accessed by users and this will improve the speed of their resource finding.
- **Addressability** – an application is addressable if it exposes at least an URI for the information that might be useful. Usually, there will be a wide number of identifiers. The addressability is one of the best characteristics of a web application. This property

[10] "Architectural Styles and the Design of Network- based Software Architectures" by Roy Fielding

let the clients to use the applications in more complex ways than the developers thought.

- **Stateless** – this means that each HTTP request is completely isolated from the others. When the server catches a request, it has all the necessary information to respond, in that request. The server will never send responses based on precedent requests. In other words, the client must not modify the server state in order to receive a correct response.
- **Representations** – when an application is spitted in resources, the client-server interaction area is expanded. The users may construct a URI to access the data they need. But the resources are not data. Shortly, a resource is a representations source and a representation is composed of data which describes the current state of a resource.
- **Uniform Interface** – there is a limited number of actions that may be used to manipulate a resource. HTTP offers a set of most common operations: GET, in order to return a representation of a resource, POST, to create a new resource, PUT, to modify an existing resource, DELETE, to delete a resource, HEAD, to return a representation formed only with metadata, OPTIONS, to verify the what kind of methods does a resource support. A uniform interface provides two services the advantage to be similar one with the other as two sites are similar one with the other.

2.4. Real-time Transport Protocol

Real-time Transport Protocol (RTP) is a protocol used for media information (sounds and images) transmission in a communication network.

In internet, like in other networks, it is possible to lose or to change the order of the transmitted packages. The multimedia applications have strict conditions over transmission service. In order for such a service to work over the internet, the RTP was created. This protocol is based on Klark and Tenenhauzen's ideas and has the purpose to transmit data in real time (e.g.: audio or video information). Usually, applications implement RTP over UDP to use the possibility to multiplex or checksum control. But RTP can be implemented over any OSI layer 4 protocol.

It must be considered that RTP doesn't guarantee the package transmission in time and does not guarantee the transmitted data integrity. The information transmission correctitude can be assured by the receiver using the package order numbers. This possibility is important when RTP is used for image transmission.

In practice, RTP is not independent of RTCP (Real-time Transport Control Protocol). The second one monitories the transmitted information about the exchanging data users.

Considering an audio-conference, each member must own one address and two ports: a port for audio information transfer (RTP packages) and a port for data control information (RTCP packages). Each conference member must know all the other members parameters. During the conference, each member sends small audio packages once each 20 milliseconds. All these packages are enclosed in RTP data fields. RTP also is integrated in UDP. The RTP packages header determines what kind of audio coding is used, if a new user connected to the conference or if the transmission speed should be increased or decreased.

During the audio transmission, an important part is the interaction between time coded fragments. To clarify this problem, the RTP header contains both the time and the order information. The order number helps not only to regenerate the fragments order, but also to find out how many packages were lost during the transmission.

If, during the conference, there are transmitted audio and video signals, they are completely independent one on the other. They are transmitted over different flows incorporated in UDP protocol. Also, the RTCP packages are sent independently for each type of signal.

RTP package structure

- Ver (version - 2 bits) – RTP protocol version. Currently most used version is 2
- P (padding - 1 bit) - indicates if there is supplementary information at the end of RTP

package

- X (extension – 1 bit) – indicates if there are used protocol extensions in the package
- CC (CSRC Count - 4 bits) – contains the CSRC identifier number
- M (marker – 1 bit) – information used at the application level
- PT (Payload Type – 7 bits) – indicates the format of the payload
- SN (Sequence Number – 16 bits) – used to count number of the lost packages

2.5. WEB

2.5.1. HyperText Markup Language

HyperText Markup Language (HTML) is a markup language used for web pages development which can be displayed in a browser. The HTML purpose is information presentation (paragraphs, fonts, tables etc.) and not document semantic description.

HTML is a form of oriented marking dedicated to present text documents on o single page using a specialized software, called HTML user agent, present in any web browser.

HTML provides the ways through which the document content can be annotated with different types of metadata and display indications. The display indications may vary from minor text decorations, such as underlining a word, to sophisticated scripts, image maps and forms. The metadata may include information about the document title and author, structure information about document segmentation, paragraphs, lists etc. but also essential information about document links to other documents, which are called a hyperlinks.

HTML is a text format designed to enable editing and reading using a simple text editor. However, in order to write and modify the pages in this way one needs solid HTML knowledge, but also it is time consuming. There are graphic editors which provide to user the ability to treat the web pages as Word documents. The advantage of such editors is that the code is written very fast, but the HTML code is of a poor quality.

HTML pages are composed of etiquettes or tags and have the extension "html" or "htm". The majority of these tags compose a pair, one used for opening <tag> and one in order to close </tag>. There are also cases in which one must use an unclosing tag <tag/>. The main tag, which if missing the HTML page cannot be displayed, is <html> and </html>. This etiquette must be present as the first respective the last tag of a document. Everything outside these tags will not be interpreted and displayed. Many tags can have properties or attributes which provide a wide variety of visual or logical functionalities.

The web browser interprets the etiquettes displaying the result on the screen. In order to alter HTML tags, or their properties, JavaScript scripting language can be used. Also, there is another language that can help HTML during the page display. This one is called CSS (Cascade Style Sheets) and has the purpose of modularizing the application in terms of content and display, where the content is provided by HTML and the display by CSS. These topics will be discussed later.

HTML specifications are maintained by the World Wide Web Consortium (W3C).

2.5.2. CSS

Cascading Style Sheets (CSS) is a standard used for HTML document elements formatting.

It is commonly used in HTML or XHTML, but it can be applied to any XML document, including plain XML, SVG and XUL. Along with HTML and JavaScript, CSS became a mandatory technology for every web application development.

The primary usability of CSS is the separation of document content from document presentation, including the layout, colors and fonts. Also, CSS enables multiple web pages to share

the same format without duplicating the written code, when it is externalized in another ".css" file. For each matching HTML element, it provides a list of formatting instructions. For example, a CSS rule might specify that "all titles should be blue" leaving pure semantic HTML markup without formatting code such as a <color...> attribute indicating how such text should be displayed.

Another useful feature of CSS externalization is that an user can use his own ".css" file to view the markup documented, overriding the author one. Also, an important advantage of CSS is that aesthetic changes to the graphic design of a document, or hundreds of documents, can be applied quickly and easily, by editing a few lines in one file, rather than doing the same repetitive work for each page.

The CSS specifications are maintained by the World Wide Web Consortium (W3C).

2.5.3. JavaScript

The Netscape Company offers once with the Netscape 2.0 browser a new, complete, object oriented environment dedicated to WWW pages developers without a wide programming experience. This new environment is JavaScript. Motivated by the Java technology of Sun, Mark Andreessen, of Netscape, buys a Java license and produce a scripting programming language based on this one, named LiveScript. In December 1996, Sun and Netscape merged their efforts in order to improve this new language, which got the name JavaScript, version 1.0. That was the point in time when JavaScript began to suffer continuous exhaustive improvements.

As in the other programming languages, in JavaScript one can define different scalar types or compound variables. Also one may use attribution, test, cycling or control instructions, or may use objects which contain predefined or programmer written methods.

The main JavaScript characteristics are described below:

- Variables*:* In JavaScript, the variables identifiers may be composed of letters, digits or underline (symbol "_"), with local or global scope. They are also case-sensitive, as in C or Java.
- Data types: The predefined types include: integer numbers (base 10, 8 or 16) and rational (written in decimal or scientific form), Boolean type, character strings, Object, Null and Undefined type. A variable that doesn't have any associated value is an Undefined type.
- Operators: JavaScript include usual arithmetic, relational, logic, special, or object manipulation operators. In JavaScript 1.2 regular expression are permitted.
- Instructions: There can be used attribution, block, test (if-else, switch), repetitive (for, do-while, while), control (break, continue), object manipulation (for-in, with) instruction. Also, the user may create a function in order to use it later in the Web page.
- Objects: The programmers may define their own objects, but the main language advantage is to complete predefined object hierarchy, respecting the Document Object Model (DOM).

JavaScript programming language is exclusively dedicated to Web pages development. The source code is interpreted by the browser when the HTML document is loaded. With its help, the site user can have an enjoyable experience while navigating. Through JavaScript, the site responds instantly to some common user requests, without the need of accessing the server.

In other words, JavaScript represents a way to manipulate HTML content depending on specific user actions.

3. DEVELOPMENT TOOLS

Software applications evolved fast in their existance. They become complex and complex, year after year so they must be handled by stable solid tools. In order to manipulate a wide number of technology layers of an application, the development tool must be as complicated as the application itself, or even more complicated.
In the case of "Multimedia streaming and storage web-service" application, developed in Java programming language, Eclipse Integrated Development Environment (IDE) was chosen and a series of Java frameworks.

3.1.Eclipse IDE

"In computer programming, Eclispe is an integrated development environment. It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostnly in Java, Eclipse may also be used to develop applications in other programming languages: Ada, ABAP, C, C++, COBOL, Fortan, Haskell, JavaScript, Lasso, Lua, Natural, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework). It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala., Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others."[11]

Firstly, the Eclipse project started as an extension of IBM VisualAge. This IDE is released under Eclipse Public License, Eclipse SDK being a free open source software, incompatible with GNU General Public License.

Eclipse has the name inspired from the Company which develops Java, Sun Microsystems. The name was not meant to shadow the usability of Sun's programming language, but quite the contrary, to increase its popularity providing a powerful IDE.

Eclipse project released over 10 final products with names inspired by astral corpses. Some of them are sattelites or planets names such as: Europa, Galileo, Juno, Kepler, Luna, Mars and others. Also, there are announced releases and the project suffers an unexhausted development even today.

3.1.1. Application development using Eclipse

Eclipse provides a set of useful features helping in the development and debugging stage of an application. Using this IDE, one can create any type of project he may need, but the most common is simple Java Project.

In the first phase when Eclipse is opened, one must choose or create a workspace. This self-descriptive term is a folder where the source code, configuration files or any other type of files will be placed. It is, in fact, the place where all projects are.

After the workspace is set, then a project must be created. The "Multimedia streaming and storage web-service" application uses two types of projects: Java Project and Dynamic Web Project. The first one is used for the client application, which, in the final stage will become a Java Archive (JAR) and the second one will be a Web Archive (WAR).

For both project types, Eclipse provides the template by default. For example, in the case of

[11] https://en.wikipedia.org/wiki/Eclipse_%28software%29

a Java Project, Eclipse will create the *src* folder, where the programmer sources is placed, the *classpath* file and the *bin* folder, where the compilated sources are.

During the development process, Eclipse facilitates the utilization of frameworks. They can be easily added within the project using simple "file choosing" dialogs. Using this IDE, the programmer must not add manually the framework path in the classpath, it will be automatically added.

Eclipse owns hundreds of features, options and shortcuts. Some of the most used will be enumerated and shortly described below:

- auto-complete class parameters: in the moment of creation, for a class can be chosen the interfaces which it will implement, if it will contain the main method and others. The IDE will auto-complete all these details, incluauding the package of the corresponding class.
- fast creation of setters and getters: when the fields of a class are defiened, with a single click one can create the setter and getter for each one.
- auto-complete or list the related words: while writing the code, Eclipse will show you one or more related words to what you mean to write. Using this feature, simple, compile errors are excluded and the code will be written faster.
- permanent compilation: after each written word, the IDE will compile the actual code and will show the existing compile errors. In this way, the classical code writing is totally forgotten. For example, in others old IDEs, after a part of the code is written and you want to test it, you will compile it at least three times to eliminate all the errors. It must be specified that this feature is available for all the programming languages supported by Eclipse including HTML, CSS, Javacscript and XML.

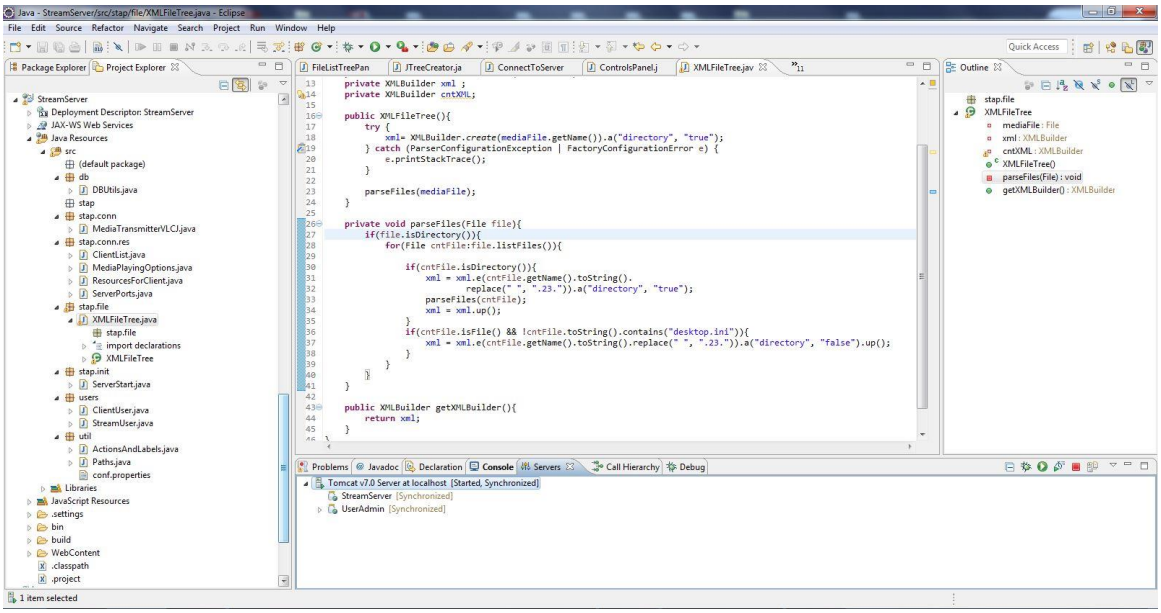


Figure 3.1 - Eclipse IDE

It can be observed a detailed directory structure of the containing files in the project, in the left part of the image. In center, there is the actual code written by the programmer, with other opened files as tabs, in the upper side of the image. In the right part there is the structure of the current opened class with type of methods and fields. On the bottom, there are the most used parts in the development phase of a project: the Console,Problems, Javadoc, Servers and Debug. It must

be mentioned that each of these views can interchange their position: from right to bottom, from left to right etc. Also, there are more than fifty other views available for the programmer to add and configure in his workspace.

When the code is written and needs to be smoke tested, Eclipse provides a helpful debugging service. In user to use the debug, the programmer must provide one or more key points where he suspects logical errors would appear. These key points are called *breakpoints* and when the application flow reaches them, the execution will freeze. Now the programmer has all the time to see and to understand the current stack and variables state. Also, Eclipse provides a very useful information contracted debugging view.

3.2. Database

"A database is an organized collection of data. It is the collection of tables, queries, reports, views and other objects. The data is typically organized to model aspects of reality in a way that supports processes reuiring information, such as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies."[12]

The practical and usable database is in fact the Database Management System (DBMS).

3.2.1. Database Management System

A database management system (DBMS) is a software which stores and processes high amounts of data. In order to be fast, these systems are optimized for standard low processing operations. In literature, the term *database* means the data to process and *management* refers to storage and modification actions.

The DBMS perfoms four standard functions:

- definition – to create new data types and data structures, usually this means new tables and new sequences
- update – to modify the existing data or to add a new set o data in an existing defined structure
- retrieval – to perform search or counting in the whole existing database
- administration – to add new database users, to improve security, to monitor the data and others

In order to perform these actions, the DBMS take commands in a standard name Structured Query Language (SQL). Any interraction with the database must be clearly defined in SQL.

For "Multimedia streaming and storage web-service" application was chosen PostgreSQL.

This is an open source object-relational database system. The project has more than 15 years and is designed to be supported by all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) and Windows. It also supports foreing keys, joins, views, triggers and stored procedures. As datatypes, it allows declaration of INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL and TIMESTAMP.

In terms of maximum limits, PostgreSQL provides unlimited database size, 32 TB table size, 1.6 TB row size. 1 GB field size, unlimited rows per table, 1600 columns and unlimited indexes per table.

It must be mentioned that this PostgreSQL has a documentation of over 3000 pages, so there

[12] https://en.wikipedia.org/wiki/Database

is a high probability that the programmer will find there anything he may ever need regarding this DBMS.

PostgreSQL won a series of industry recognition prizes, including the best DBMS for a couple times.

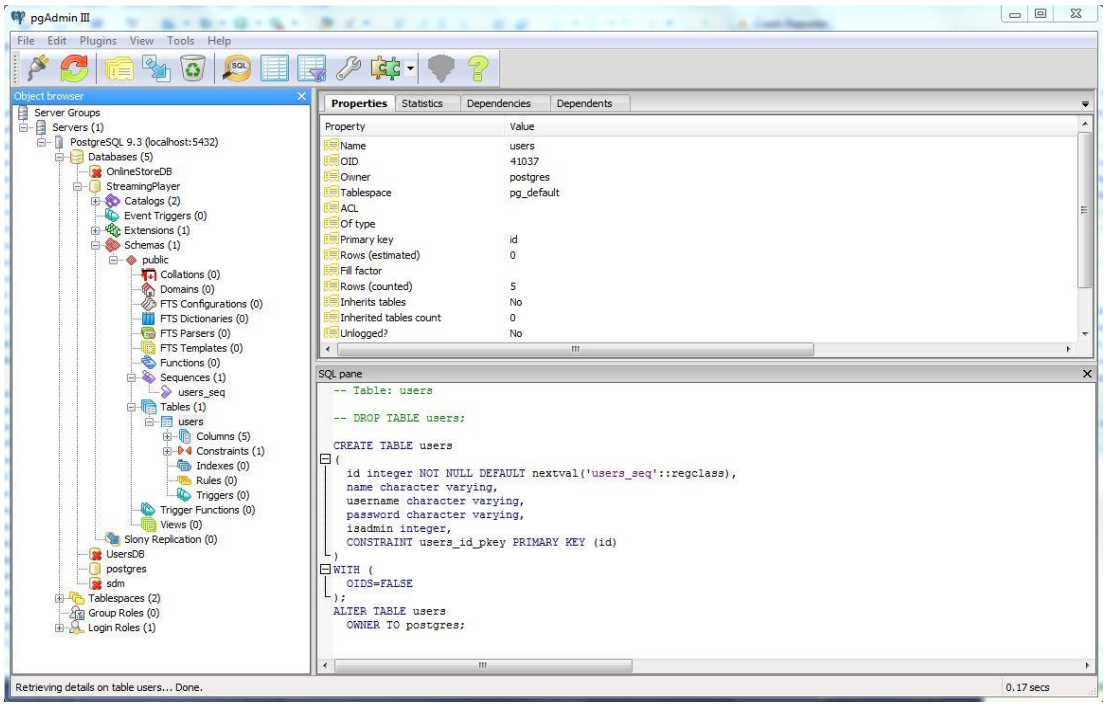


Figure 3.2 - PostgreSQL

This picture shows the general structure of a database inside PostgreSQL. One may observe that all databases belongs to a server. Each database contains one or more schemas, each of these having one or more tables. A table can be created with SQL code or using the UI. In this case, UI was used and PostgreSQLshows the actual code it generated automatically. This DBMS will have the same behavior in case of sequences or other elements contained by a schema. Also, the content of a table can be altered using the user interface.[13]

3.3. Java frameworks

When developing an application that may use high level already existing services, the use of Application Programming Interface (API) or frameworks is recommanded. The frameworks and APIs are already implemented and stable libraries that provides a series of complicated services in just a couple of lines of code. For example, in REST frameworks, such as Jersey and Spring, there is a class that catches all the incoming HTTP requests. Imagine how complicated it would be to implement this very important feature for the back-end of each web-application. The wheel would be reinvented for each site.

In the following sections, there will be briefly described and explained the frameworks used for the development of "Multimedia streaming and storage web-service".

3.3.1. Jersey

Jersey is an open source Java framework specialized in REST services. This API provides

[13] http://www.postgresql.org/about/

the ability to create a REST client and also to create a series of server important features.

Figure 3.3 - Jersey Framework

Source: http://javasolutionsonline.blogspot.co.uk/2014/02/creating-simple-web-service-with-jersey.html#.VZFNM2M4em4

The REST client is simply to implement. All that is needed is a new client from ClientBuilder class and then, over this to add the URL to connect to, the type of HTTP method and the document format of the request: XML, text and JSON. After the request is done, the object will contain the actual response of its request. In this result there are the HTTP response codes and other protocol related details, but also the data response of the server.

On the other side, the server is much more complicated to implement. First of all there is needed something to catch all the incoming HTTP requests on server URL. This catcher is called Dispatcher and can be configured to listen to any URL. Then, in application, there must be a so controller. This has the function to handle each request (the Dispatcher catches the request and passes to the controller). Also, the controller sends the answer back to the client.

3.3.2. Spring

Spring is an open source Java framework that provides support for Java applications. This is a powerful API that can be used even in enterprise applications development. It provides support for:

- MVC design
- JDBC template
- resource management
- web applications
- testing
- marshalling and unmarshalling
- data access
- security

Figure 3.4 - Spring framework

Source: http://zeroturnaround.com/blog/jrebel-5-2-released-updates-for-apache-camel-spring-framework-jdeveloper-ide-plugins/

One can observe that Spring suits for any kind of Java application. The main advantage of this framework is that it can be configured in any decision point of an application flow. The configuration is provided through one or many XML documents where are described all the attributes of the above mentioned features that are used in the application. For example, in the case of an web application developed with MVC design pattern, Spring links the controller, model and view by itself. All the programmer must do is to describe this in the configuration file.

3.3.3. VLCJ

VLCJ stands for VLC Java framework.

VLC is a free open source multimedia player. It can play all the media main formats, such as: MPEG (ES, PS, TS, PVA, MP3), AVI, ASF, WMV, WMA, MP4, MOV, 3GP, OGG, OGM, WAV, Raw DV, Raw Audio, MXF, FLAC, FLV, Nut, MIDI, SMF, MKV and Real. In other words, it can play almost everything. VLC also supports streaming over different protocols: RTP, RTSP, UDP, DCCP, HTTP and MMSH. VLC is also supported by all main operating systems. In other words, VLC is probably the most powerful player.

VLCJ is an open source framework which is in fact an API over VLC. This way, the framework inherits everything media player can do. VLCJ can be used only if the VLC is installed on the machine it runs on. All its services are, in fact, the ones of the player. The framwork accesses the player's library in order to work. For example, when creating a player in Java using VLCJ, the application will not play if it cannot find the VLC main library.

This also happens in the case of streaming. A VLC library must be found on the server, and also another on the client. On server, the coding of packets is done by the VLC library and on the client, the decoding takes place, also using the same kind of library.

The main idea is that, using VLCJ, a Java application can stream over specialized protocols almost any type of media files.

3.3.4. JAXB

XML, Extensible Markup Language is a meta-language recommanded by W3C used to create other markup languages, such as: XHTML, RDF, RSS, MathML, SVG, OWL and others. These languages form the family of XML languages.

Its advantages are:

- extensibility – one can easily define new indexes.
- validity – data structure correctitude is verified.
- application independent – the users can represent their data independent of application.

- simple and accessible – a XML document is in fact a text file used to structure, store and transfer information.
- editing – can be easily edited and modified using a simple text editor.

Due to the above mentioned advantages, the data exchange between applications is usually done through XML document format.

As XML emerged as the standard for data transfer, Java technology developers provided a framework which helps the others application developers. This API is called JAXB, which is an abbreviation of Java Architecture for XML Bindings.

JAXB provides helpful tools for both, marshalling and unmarshalling. The first term is explained in computer science literature as the process of transforming the memory representation of an object to data format suitable for storage or transmission, while the second one is exactly the inverse process.

Using JAXB, the programmer can create a XMLBuilder object which provides all the necessary methods to manipulate this format. He can add the root, elements, sub-elements and also, attributes for them. The programmer describes the logical format of XML thorugh Java code, and the framework will create the physical standard format with angular brackets, tabs, quotation marks and everything else.[14]

[14] http://www.oracle.com/technetwork/articles/javase/index-140168.html#introjb

4. APPLICATION DESCRIPTION

The final application is composed of three projects:

- Database Administration System. An administration website that can be used to create/modify/delete users for the Multimedia Streaming Service.
- End User Streaming Client. A standalone Java application, used by the final user to access and play the resources provided by the Multimedia Streaming Services.
- Multimedia Streaming Service. The service that manages and provides access to (streams) the multimedia resources.

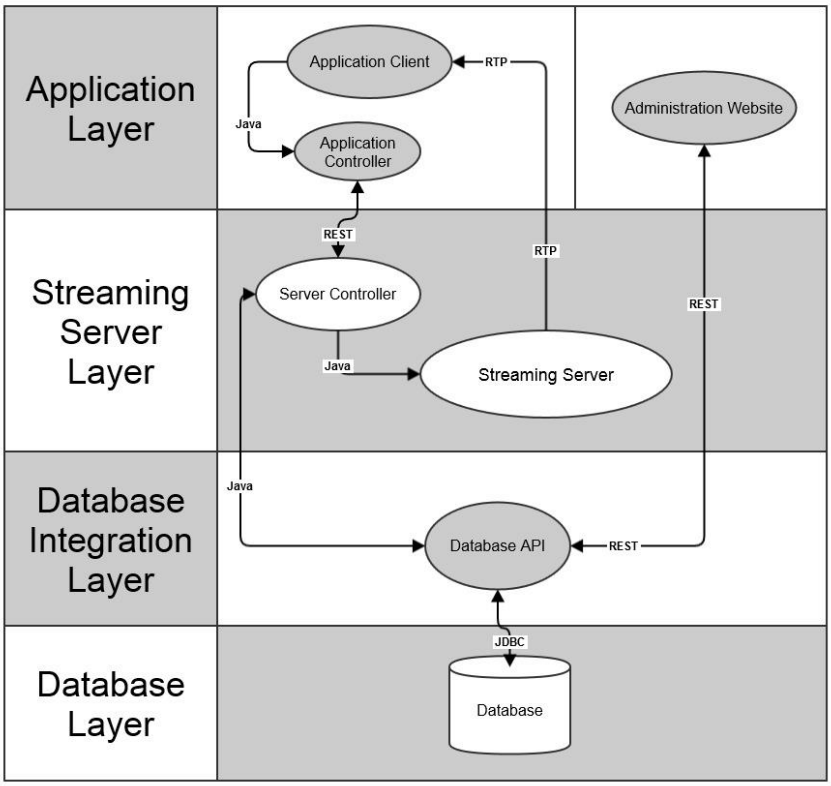


Figure 4.1 – Project architecture diagram

In the diagram it is described the projects connection and flow.

One can observe the client application, connecting to server controller through application controller, because it is designed using the MVC pattern. The server controller redirects the requests depending on its type:

- for login, it redirects them to the database
- for streaming commands, it redirects them to the streaming server

Also, we can observe that the client commands are transmitted through REST services, over HTTP and then the server process each request and may respond through REST too, or through RTP in the case that streaming packets are requested.

The administration website is another project, designed as a web client which access only the database of the server. We can observe that the communication is proceeded only through REST services. Each HTTP request is caught by a Java REST Servlet dispatcher and then redirected to the Database API. The Database returns the interrogated data and then the web-client pages are populated with it.

4.1. Administrative WEB Interface

The administrative web site appeared as a must in the development phase of the streaming application client. It represents the beginning of managing user rights in the case of application expansion. In this moment it is used only to create/delete or update users details such as name, username, password or admin rights and provides a more comfortable way than managing users and rights manually in the database. The only existing right is admin right which represents the ability to login this web-site and perform write actions over the database.

In time, the administration site will expand and will offer automatic rights management services and will provide a more detailed and complicated rights model.

The administration web interface is developed using the MVC design pattern.

The user sends requests to the server in the moment when he presses one of the buttons or links in the site. Each request is caught by a dispatcher servlet and is sent to the Controller. Then the controller take different actions depending on the URL or method type (GET or POST). In the case of GET requests (when the user presses links on the site), the controller simply asks the model to load the view and then sends back the response with the actual requested page. In case of a POST request, the dispatcher sends the request to the controller, and the controller will ask the model to access the database and then load the content in the view. The last one will be responded and displayed to the user by the browser.

4.1.1. Users administration

The client application users are stored in the database. Users details are managed through the website by a super user which has the admin rights defined in the corresponding database table. In this case, the super user is defined directly in the database and his rights can be altered only accessing directly the database without any API. For example, the super user cannot alter his rights using the administration website.

The website may be accessed through the following address: http://localhost:80/UserAdmin/rest/ in the case it is accessed from the local machine which hosts the server.

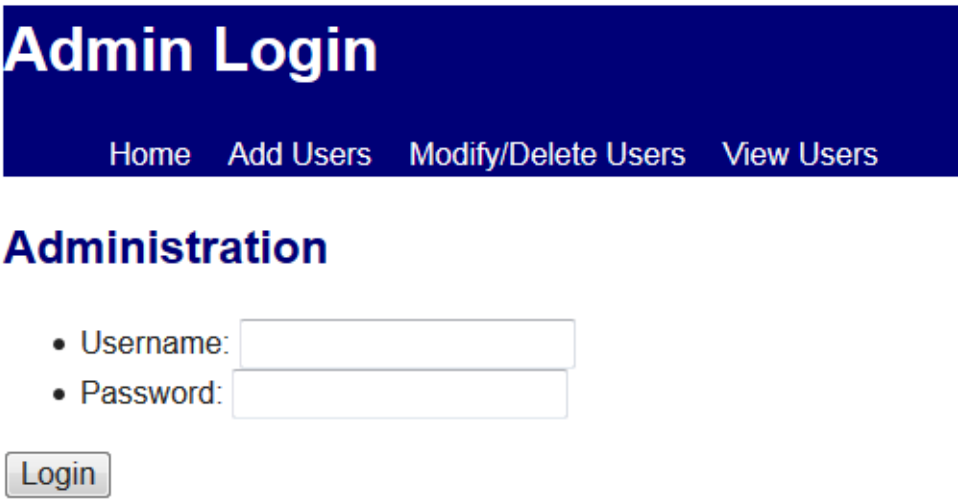


Figure 4.2 - Amin Login Page

Source: http://localhost:80/UserAdmin/rest/adminLogin.jsp

When the site is accessed, the Administration login page is prompted.

Here, only the user which has the admin rights provided in the database can proceed further with the correct credentials.

After the admin is logged in, he can add, modify, delete or view the users which can access the client streaming application.

4.1.2. Add Users

In this page, the super user can add new users in the database so they may use the client application. This page contains a HTML Form with three fields and a button which is named Insert Button. When this button is pressed, a JavaScript field validation is triggered and if it does not fail, the data is sent to the server and then inserted in the database.

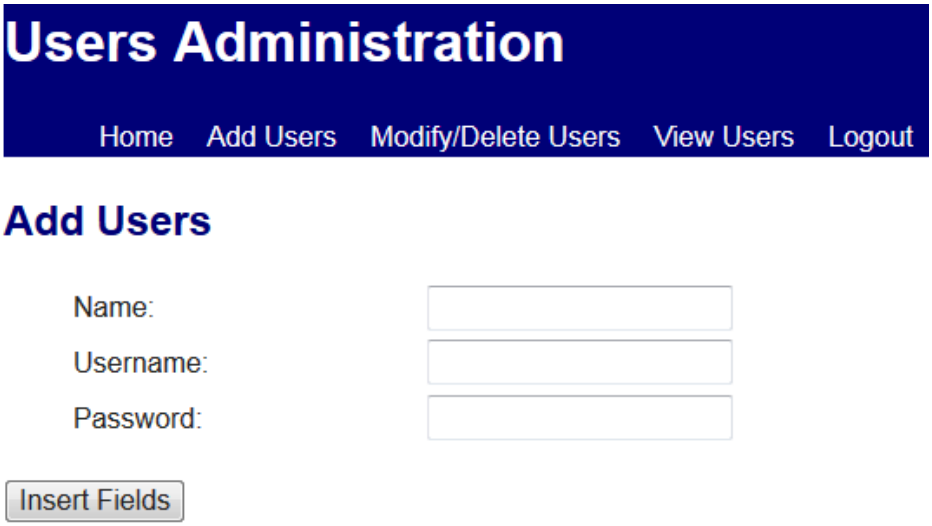


Figure 4.3 - Add Users Page

Source: http://localhost:80/UserAdmin/rest/insertUsers.jsp

To create a new user, a Name, Username and a Password must be provided. If all, or a single one of these fields is not provided, then the JavaScript validation won't pass and an alert with the corresponding uncompleted field will be prompted.

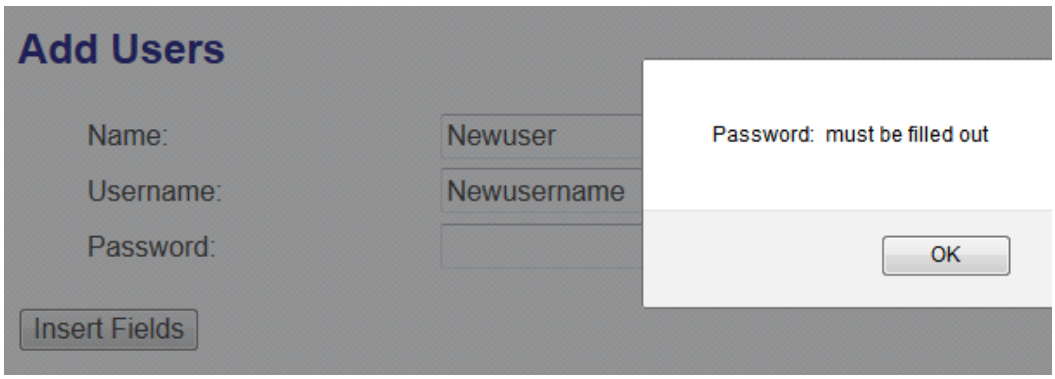


Figure 4.4 - Add Users Page

Source: http://localhost:80/UserAdmin/rest/inserts.jsp

4.1.3. Modify/Delete Users

This page contains five HTML Lists, five HTML Text fields and two Buttons. These elements are used to provide the details for all users already existing in the database. A list contains their table ids which cannot be altered because these ids are generated using database sequences. Other three lists contain the names, usernames and passwords of the users. The last list contains an integer which is used as a Boolean to see if the corresponding user is admin or not, 1 means the user has admin privileges and 0 means the user has not these rights and cannot login in this website.

When this page is accessed, the text fields are empty. When any of the elements of a list is clicked, then each text field is completed with the corresponding information using JavaScript. Then, each of the text fields may be modified with any value.

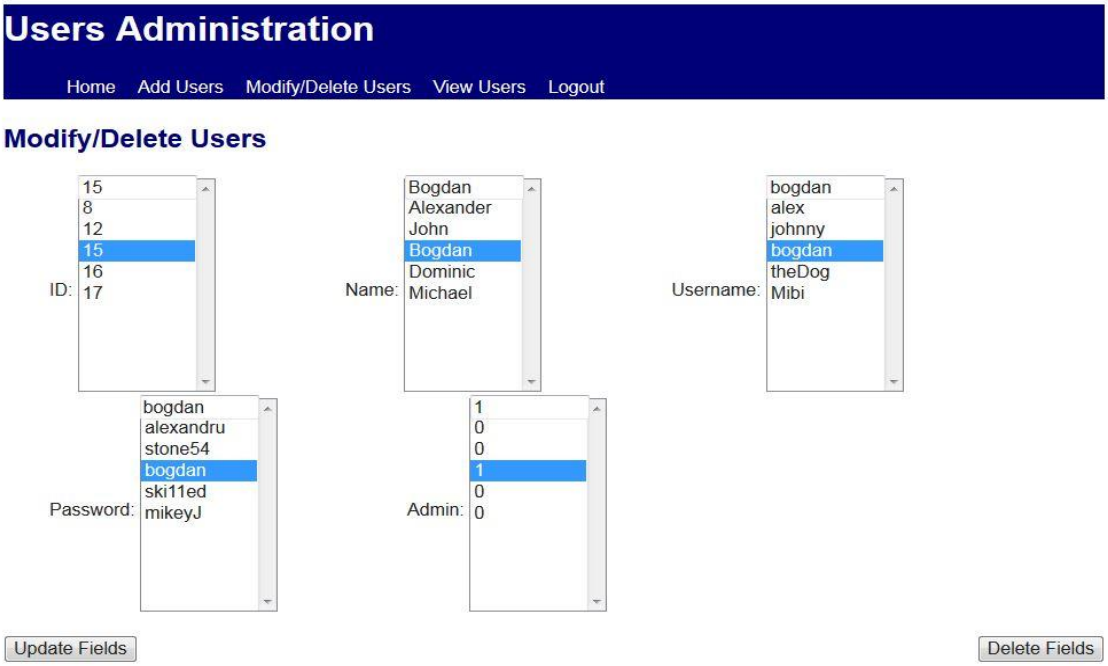


Figure 4.5 - Modify/Delete Page

Source: http://localhost:80/UserAdmin/rest/updateDelete.jsp

When Update Fields button is pressed, the data contained by the text fields will be sent to the server and the server will update the corresponding field in the database. The database query uses the ID value of the data to find and modify the fields, so modifying the ID may perform an update fail. Also, the Admin rights cannot be altered in the database table even if the HTML text field is modified.

When Delete Fields button is pressed, the data contained by the text fields will be sent tot the server and the server will delete the corresponding filed in the database. This delete service uses the ID value of the field to find and then delete the row in the database, so an ID alteration may perform a delete fail.

4.1.4. View Users

The name of this page is self-descriptive, so it is used to visualize all the users contained by the database including their details.

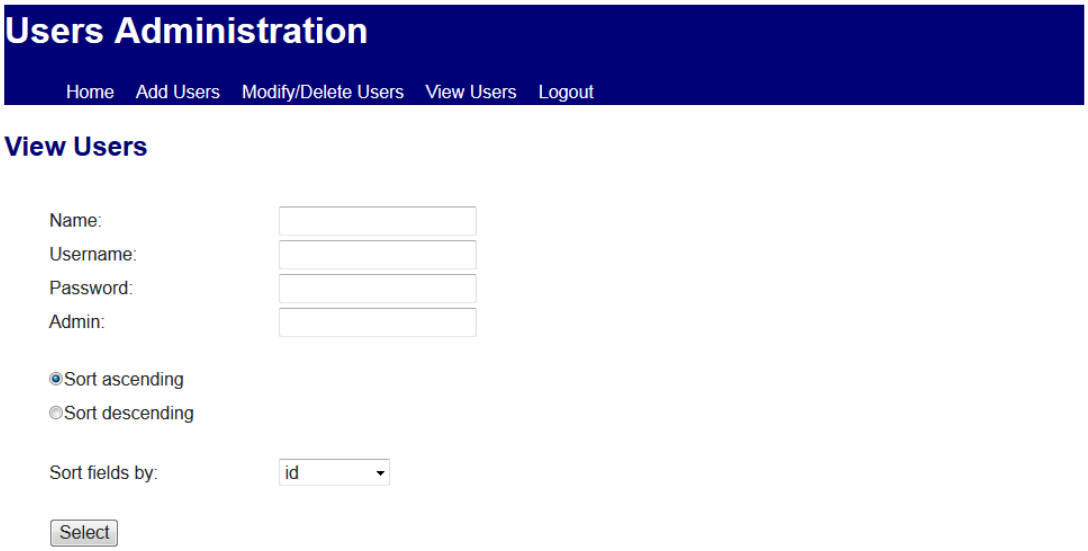


Figure 4.6 - View Users Page

Source: http://localhost:80/UserAdmin/rest/select.jsp

This page has features that allow the administrator to retrieve and sort the users corresponding to the completed fields. The element which make these features possible is a form composed by four input fields, with the corresponding labels: Name, Username, Password and Admin. Below these fields there are two radio buttons which allow the administration to sort the results ascending or descending by the fields selected in the dropdown. The dropdown contains all the above fields: name, username, password, admin and also the id. By default, all the fields are empty and the sort will be performed ascending by the id.

In the moment the Select button is pressed, the whole form is sent to the server as a HTTP POST request, then the controller of the web application will manipulate the request, redirecting it to the model which access the database. The database performs the SELECT query and passes the result back to the web application server which responds to the client.

The result returned by the server is printed in browser, as a table with four columns, in the right side of the page.

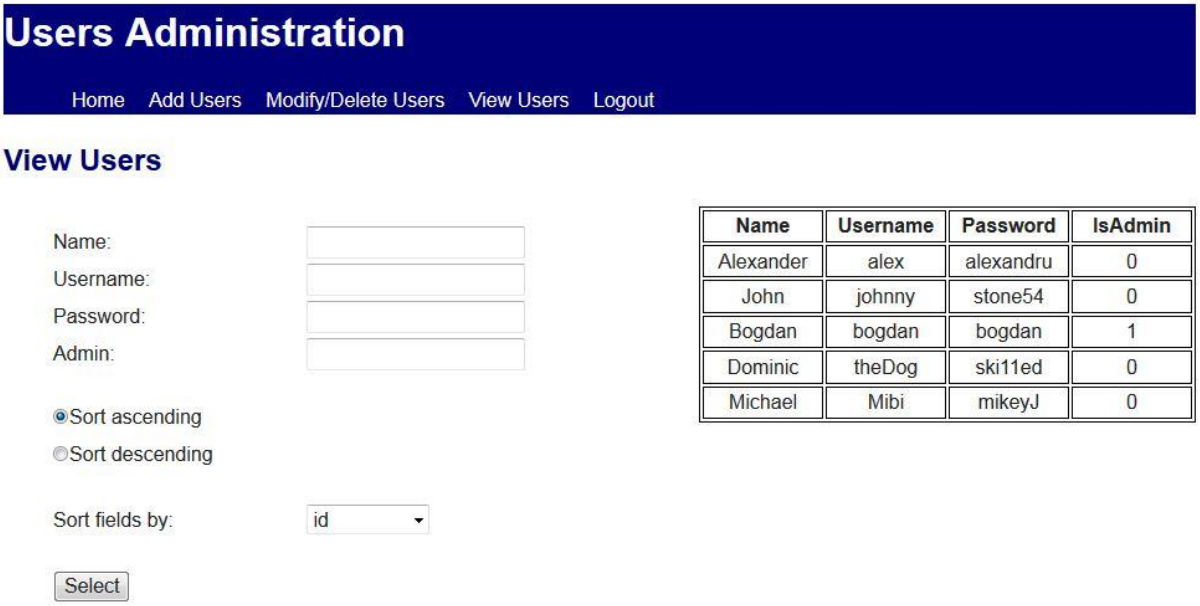


Figure 4.7 View Users Result

Source: http://localhost:80/UserAdmin/rest/select.jsp

4.2. Streaming application description

The streaming application is based on a server-client architecture. In order to work properly, the server must be started and the client application must connect to it.

The project architecture assumed a platform independency between the server and the client application. This means that the server architecture is completely independent on the client one, they can be built even in different programming languages and still have good performances.

This independency between client and server implies some important advantages:

- the client applications may be written in any programming languages. The client may only access the services offered by the server. As long as it can process the information requested, the client may be written in any programming languages.
- the server architecture and programming language may be changed without impacting any service offered to the client. The server must keep only the same services paths and types, but everything behind this can be changed.
- easier maintanance and bug free implementation in terms of the communcation between server and client. During the implementation, it is easier to spot bugs in the communcation of server and client if they are completely independent.

4.2.1. Server – client communication

The server and the client communicate over HTTP and RTP. Through HTTP the client proceed requests to which the server responds through HTTP or through HTTP and RTP.

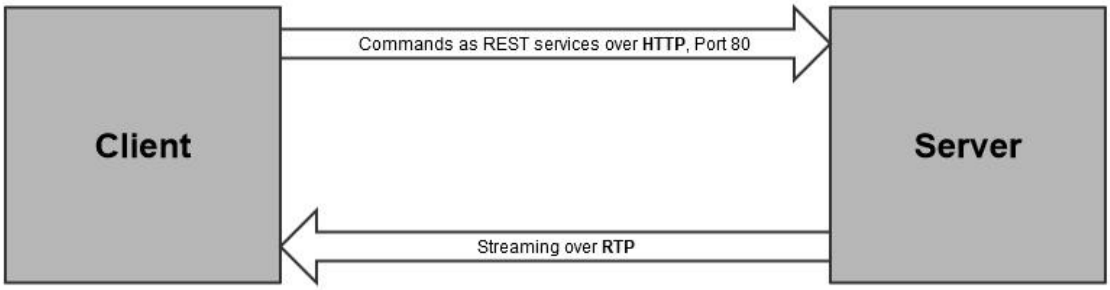


Figure 4.8 - Server - client communication

In the diagram is exposed the fact that the client use HTTP for requests as: login request, file to be played, port request and others which will be explained in details below.

Also, the server responds to these requests using HTTP responses as: login successful, port for client to connect to and others which will be explained more detailed. Also, for a file to play request, the server begins the live streaming over RTP .

It must be pointed out that a communication through Java Sockets would have been easier considering the development, maintenance and stability, but REST services were chosen due to the arguments described above: to have a portable application, to facilitate the freedom of client application development. The client may be written in any programming language and it may still be able to proceed requests to server only if it supports standard communication protocols as HTTP and RTP.

4.2.2. Server architecture

The chosen programming language for the server side is Java. The main arguments for this choice are the following:

- Java is a programming language with a high popularity
- Java has well documented libraries with many already implemented elements useful for almost any software application
- there are many open source or closed source projects which create APIs that extend the usability of Java.
- Java has IDEs with useful functionalities for server-client development

As a small conclusion of these arguments, Java make any application development easy and fast, probably easier and faster than its competitors.

In terms of project type, the server is a WAR (Web Archive) project which needs a container in order to run, in this case a Tomcat container.

In order to facilitate the communication over HTTP, Jersey API is used. Jersey is a useful framework for REST services manipulation.

For streaming, VLCJ framework is chosen. This framework is a great interface and implentation of all VLC services. This means that one have access and may use all the services exposed in VLC application and may change them.

Also, in order to access the database, the server contains a JDBC for Postgresql. With this jar help, the server has access to a configured database and may perform queries over its tables.

4.2.3. Client architecture

The client application is also written in Java programming language. The agruments of Java

choice for server can be applied for client too: high popularity, many useful already implement elements, high number of APIs and useful IDEs.

In terms of main libraries used on the client, it can be enumerated:

- Jersey, used to create a client that can proceed HTTP requests to server
- VLCJ, used to capture the live streaming over RTP
- Swing, used to create and expose the user interface of the client application
- JAXB, needed in order to process the XML received from the server

In the client application it was adopted the Model-View-Controller (MVC) design. As it is described above, the MVC decouples the data access and business logic. In this way, the application becomes more modularized and easy to maintain and develop.

4.2.4. Server - Client flow

As it was specified earlier, the server and the client communicate through REST services over HTTP. The communication begins when the client tries to access the server on the login page. As it can be observed, the user must provide the server IP, the server port to connect to and login credentials. The chosen port will usually be port 80, as it is by default the HTTP communication port which is usually not blocked by a firewall or other security services or applications.

Each time the application is closed; all text fields values are stored in a properties file. Also, when the application is started, this properties file will load the data corresponding to each field and auto-complete it. This feature facilitates the user friendly part of the application.

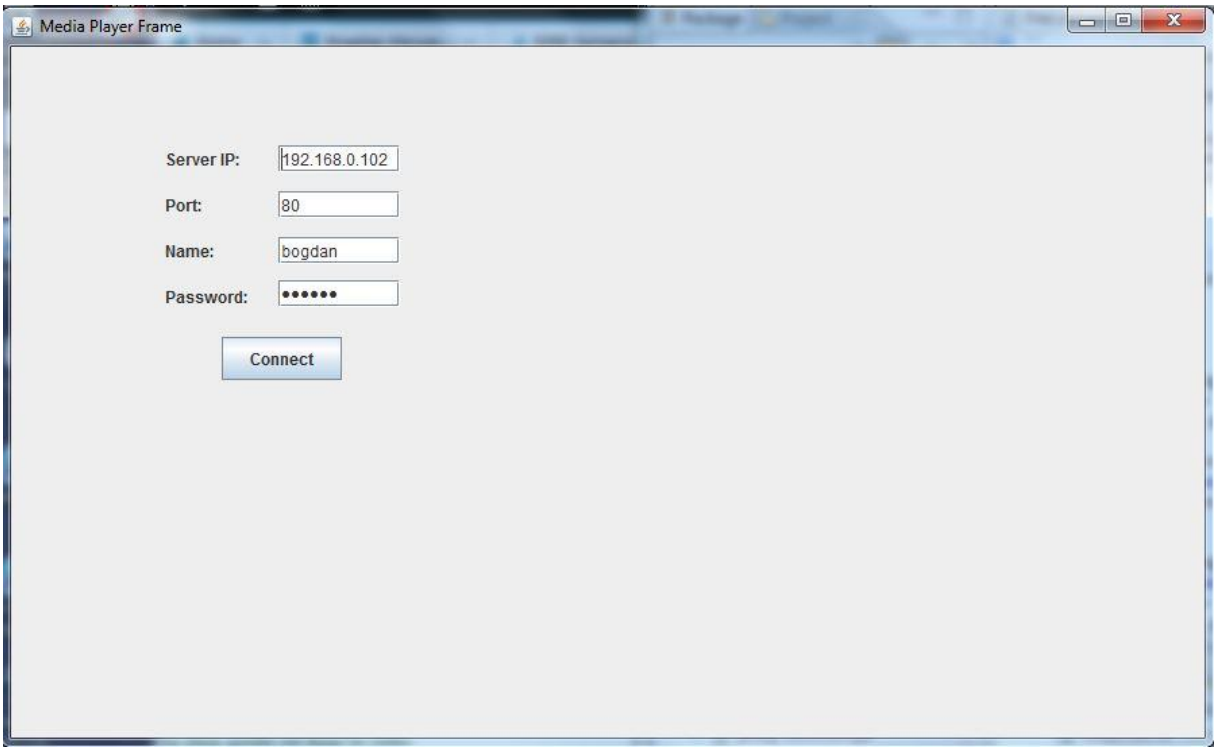


Figure 4.9 - Client login

In the momment the Connect button is pressed, a HTTP POST request is sent to the server on an URL corresponding to login service. This request contains the username and the password. Once it reaches the server, the server controller will process the request, meaning that it will read the username and password and will interrogate the database to see if the accessing user is one which has the correct credentials. Depending on the database response, the server will respond to

the client:

- wrong login or password
- sending a filelist

In the case of a wrong login, the server responds the request with a text constant which will be interpreted by the client application as a wrong login. As a result, the client will pop-up anpanel which will alert the user about the wrong credentials. Then, the user can press OK and try again with another username or password.

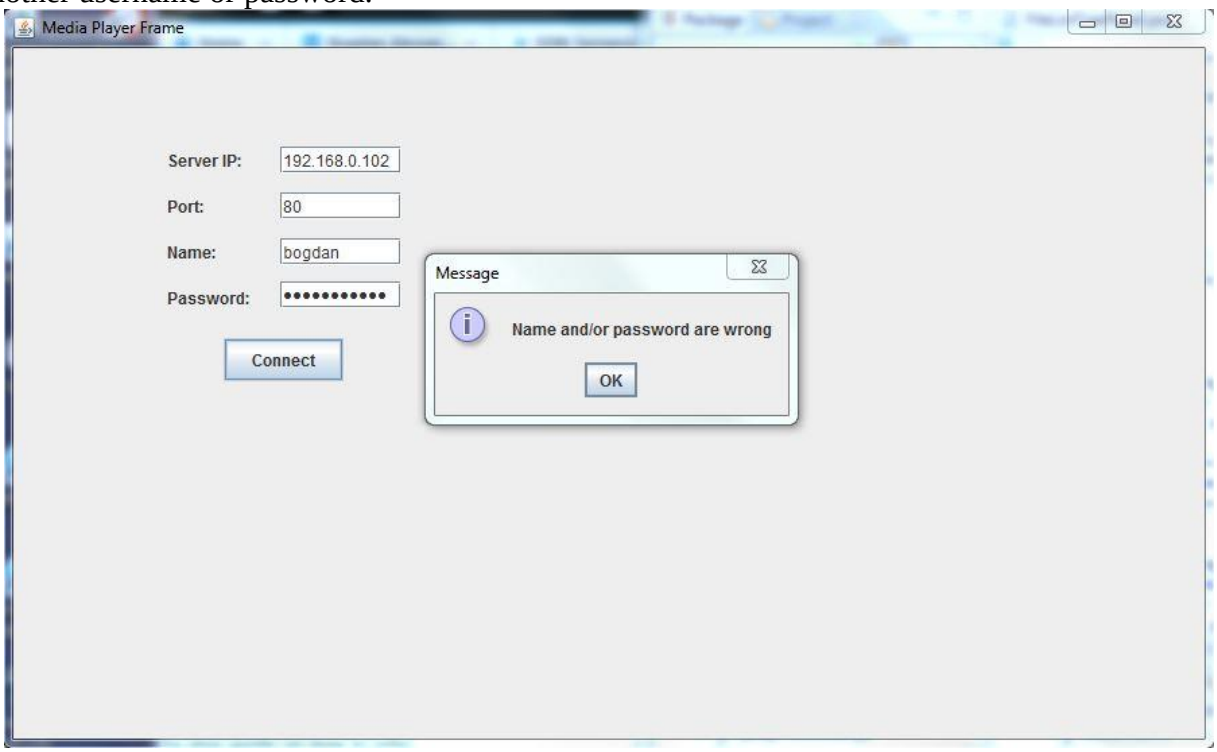


Figure 4.10 - Client login error

In the other scenario, when the credentials are correct and found in the database, the server will respond with an XML file list.

This filelist is an important component of the streaming application. It is created by the server once the client successfully logins. It contains, in fact, the absolute address of all the media files existing on the server.

The server contains all the media files in a directory on the operating system. Also, we observe that the folders over an operating system have a tree structure. Considering these two aspects we can correlate the main directory with the tree root, each folder with tree nodes and the files can be seen as leafs. So, the optimum idea to parse all the files in the main directory is to use tree algorithms. In this case, the most appropriate is Depth-First Search (DFS). Moreover, during the parsing, the method implementing the DFS adds in an XMLBuilder the elements one by one, directory in directory and so on, putting an attribute as a flag to underline if an element is a directory or a media file. These flags are used on the client, when the XML will be processed.

```xml
<zMedia directory="true">
    <Movies directory="true">
        <Aggregation_Daniela.wmv directory="false"/>
        <DerUntergang directory="true">
            <DerUntergang.mp4 directory="false"/>
        </DerUntergang>
        <Gone.Girl.1080p.mp4 directory="false"/>
        <The.Hundred.Foot.Journey.1080p.mp4 directory="false"/>
    </Movies>
    <Music directory="true">
        <acoustic.wav directory="false"/>
        <Adiós.mp3 directory="false"/>
        <chord.wav directory="false"/>
        <flourish.mid directory="false"/>
        <Kalimba.mp3 directory="false"/>
        <Lágrimas.23.Perdidas.mp3 directory="false"/>
        <MaidwiththeFlaxenHair.mp3 directory="false"/>
        <New.23.folder directory="true">
            <New.23.folder directory="true">
                <ddd.txt directory="false"/>
            </New.23.folder>
        </New.23.folder>
        <Olvidar.mp3 directory="false"/>
        <Opus_Nigrum-Requiem.mp3 directory="false"/>
        <Opus_Nigrum-Requiem.wav directory="false"/>
        <Puedes.23.Ver.23.Pero.23.No.23.Tocar.mp3 directory="false"/>
        <SleepAway.mp3 directory="false"/>
        <Yo.23.Vivo.23.Por.23.Ti.mp3 directory="false"/>
    </Music>
    <Text directory="true">
        <ControllerFile.1.log directory="false"/>
        <ControllerFile.log directory="false"/>
        <test.txt directory="false"/>
    </Text>
</zMedia>
```

Figure 4.11 - XML sent by server

The reasons that lead the filelist structure to be incapsulated in an XML format are that the XML is the most appropriate to a tree structure and it is also a standard used by HTTP and REST Services.

We can observe that the XML contains ".23." instead of " " (space). This rendering was chosen because it was tried to avoid the white spaces in an XML.

When the client receives the file list, it will parse it and will add the XML elements in a Jtree maintaining the tree structure. In order to eliminate the white space substitute, a ComponentCellRenderer implementing TreeCellRenderer is used. Finally, the Jtree is added in it's own panel and loaded in the client view.

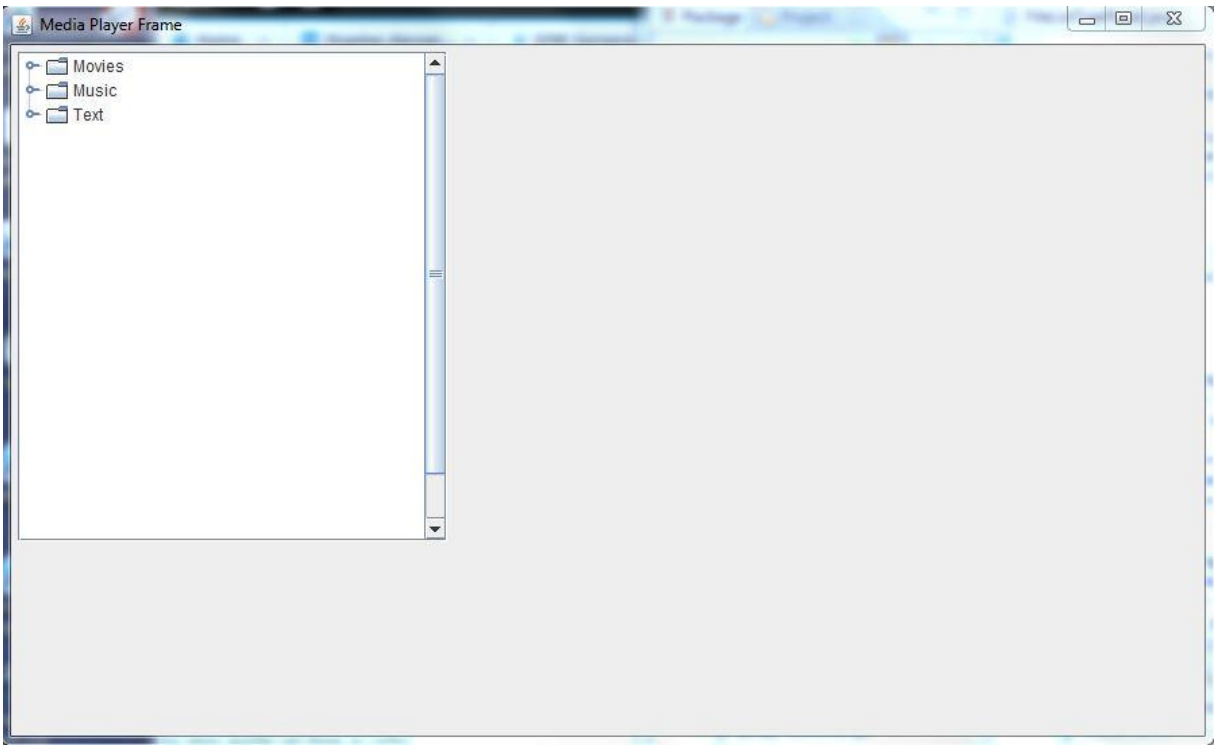


Figure 4.12 - Applcation after login

Immediately after the client receives the XML file list, the client sends another request to obtain the streaming port. In order to do this, the client proceeds with a HTTP POST request with the constant message which will ask the server to provide a port for the client to connect to. On the other side, the server has a properties file with a list of ports. Currently there are 20 ports available for the server to use. Once the server is started, the ports are read from the file and loaded into list of a singleton class. Each time a port is requested, or a connection is closed, a port is marked as occupied, respective free. Also, the server contains another singleton class that contains a list with all already connected users and their corresponding ports.

In this moment, with the file list and the port available, the application waits for user to expand the folders and choose a file to play. It can be observed that the Jtree has a JscrollPane attached. Once the folders and files tree is expanded, the scroll line will become smaller and smaller to enable for user the ability to choose any file he wants.

In the right side of the Jtree, there is left a blank space because there will be the media video panel. Also, at the bottom of the frame there is left some free space because there will be brought the control panel of the media files.

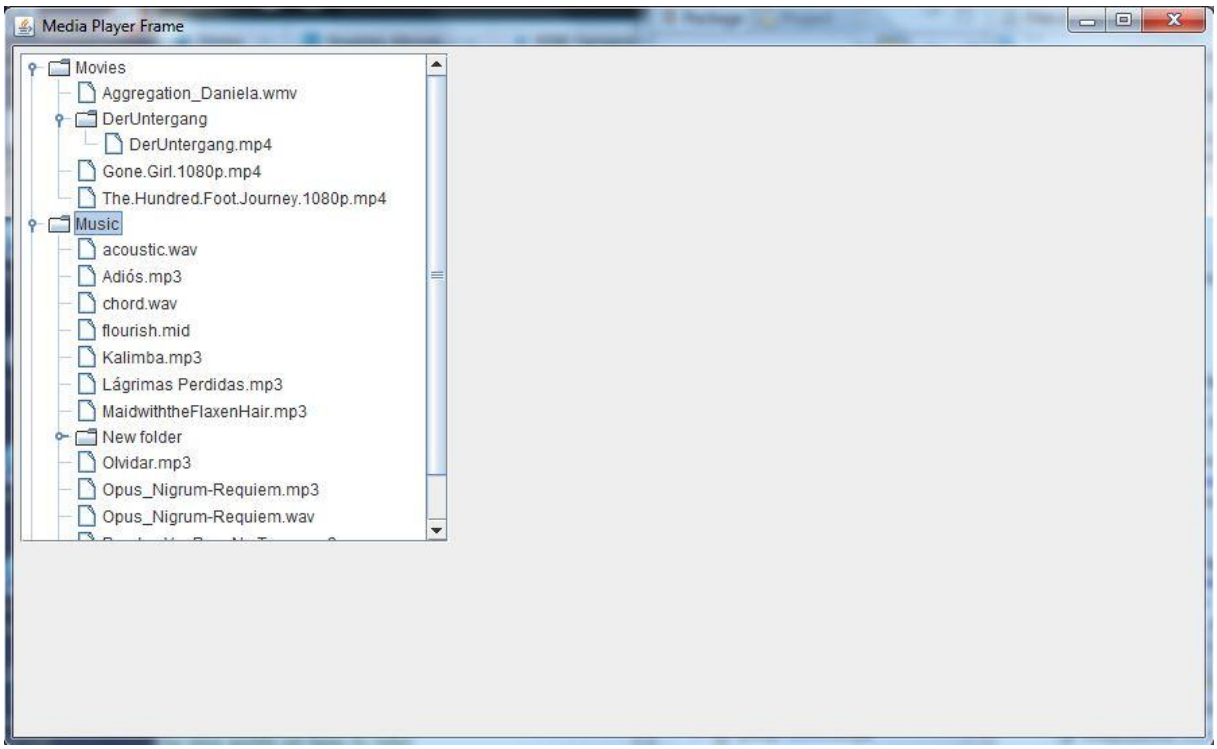


Figure 4.13 - Expanded file list tree

In the picture above can be observed how an expanded tree will look like. Also, one must remark the backward conversion from ".23." to " " in the files and folders names.

Now, the only thing the user must do is to double-click a media file at his choice.

In the moment he double-click it, a complex service will be started. First of all, the client sends a start streaming request with string message containing the address relative to the directory root.

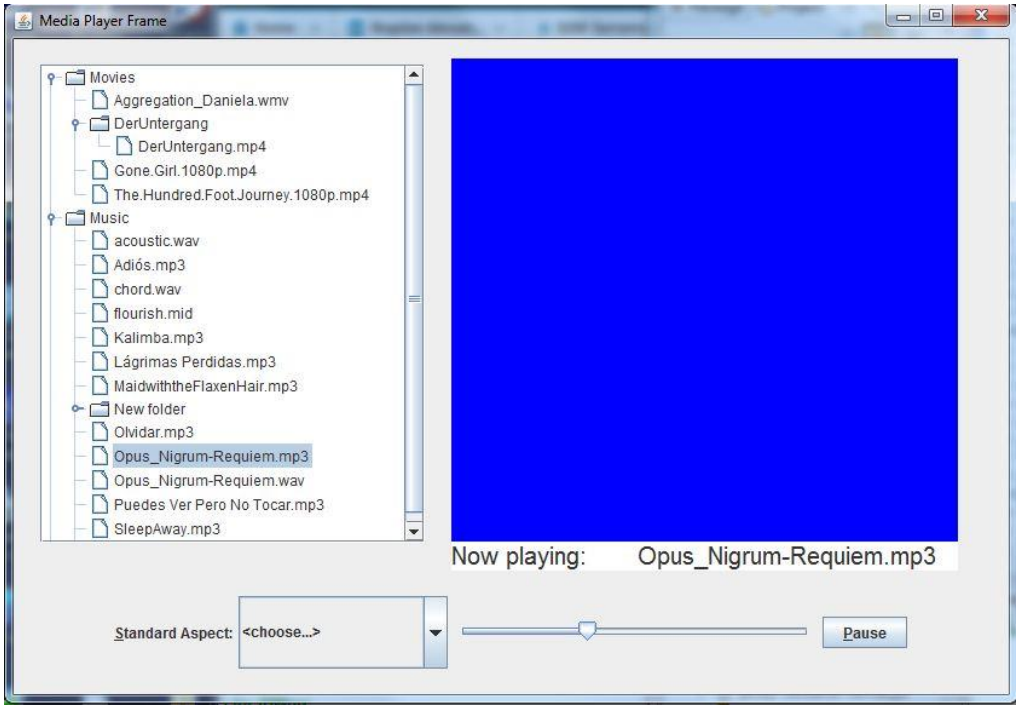


Figure 4.14 - Playing audio stream

In this case, the message will be "Music\Opus_Nigrum-Requiem.mp3". This message will be a part

of the HTTP POST request proceeded by the client. Immediately after the message is sent, the client starts a new thread with a VLCJ streaming listener on the port provided by the server. This listener looks like this, "rtp://serverIp@localhost:port". It must be mentioned that starting a new thread is necessary because the current thread must be free to send commands to server. Also, after the streaming player is created, there are loaded the video panel, in case of video streaming, a Jlabel with the currently playing file name, the control panel, the same for any kind of streaming, with an aspect ratio combobox, a Jslider to choose the moment to jump to and a Pause/Play button.

On the server side, the request is caught by the server controller which will read the message and then will create a new streaming transmitter object for the corresponding client IP. It must be mentioned that this created object is also a thread due to the above enumerated reasons. Once everything is created, the transmitter object is added in the client list too. The transmission thread is started and we have a working live streaming from server to client.

As long as the client listens and captures the streaming, nothing is initiated by the server, but the RTP package transmission.

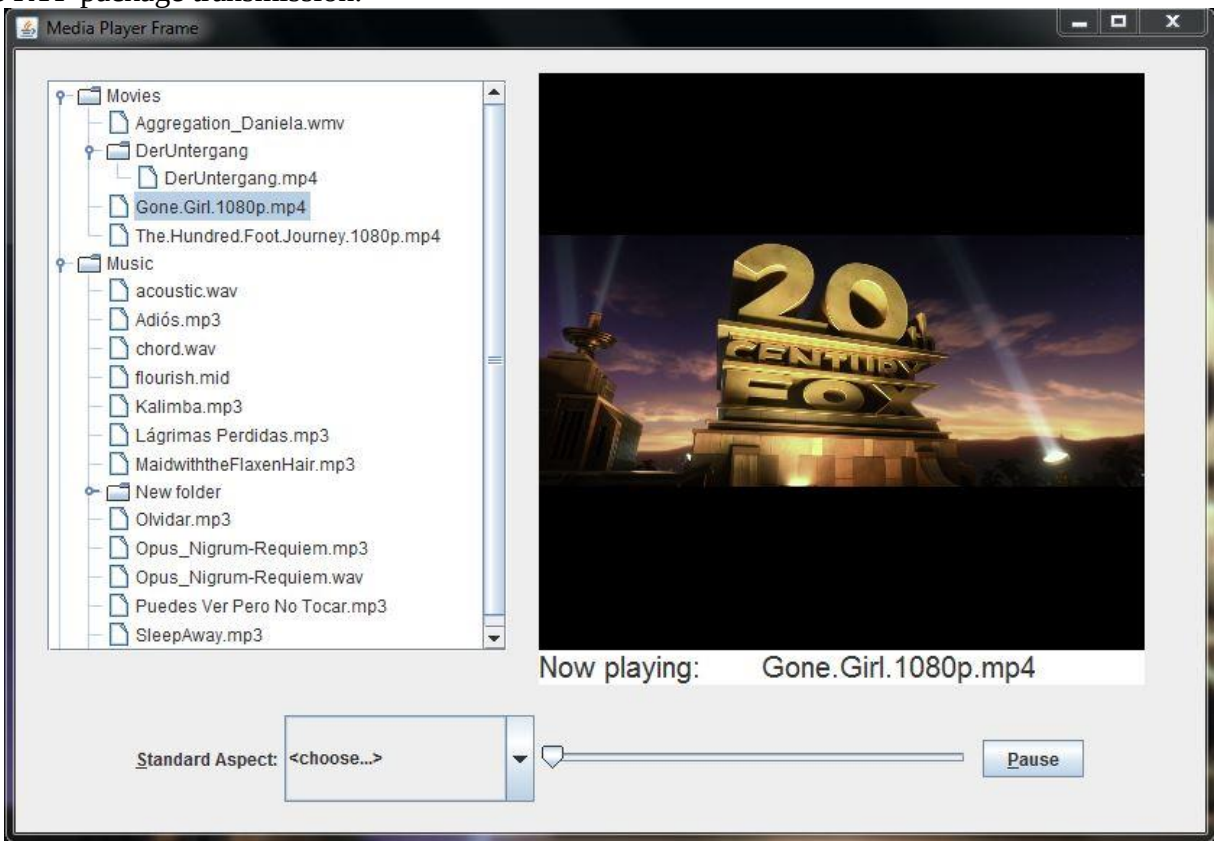


Figure 4.15 - Beginning of video play

When another file is double-clicked, a new similar service as the above one is started. The main differences are that when the mouse event is triggered, the "Now playing" label is updated and a HTTP POST request is sent. The server will read the message and will initiate the streaming cancel. Immediately after the cancel request, another HTTP POST request which contain the file to play is sent. From now on, the communication flow is the same as the one described above.

Additional commands to server are triggered by the client when the slider or Pause button is used. For the "pause" command, a HTTP POST request is sent and the server will initiate a break for media package transmission. This break will last until another request reaches the server, but it's complementary command is the "play" command which respect exactly the same flow.

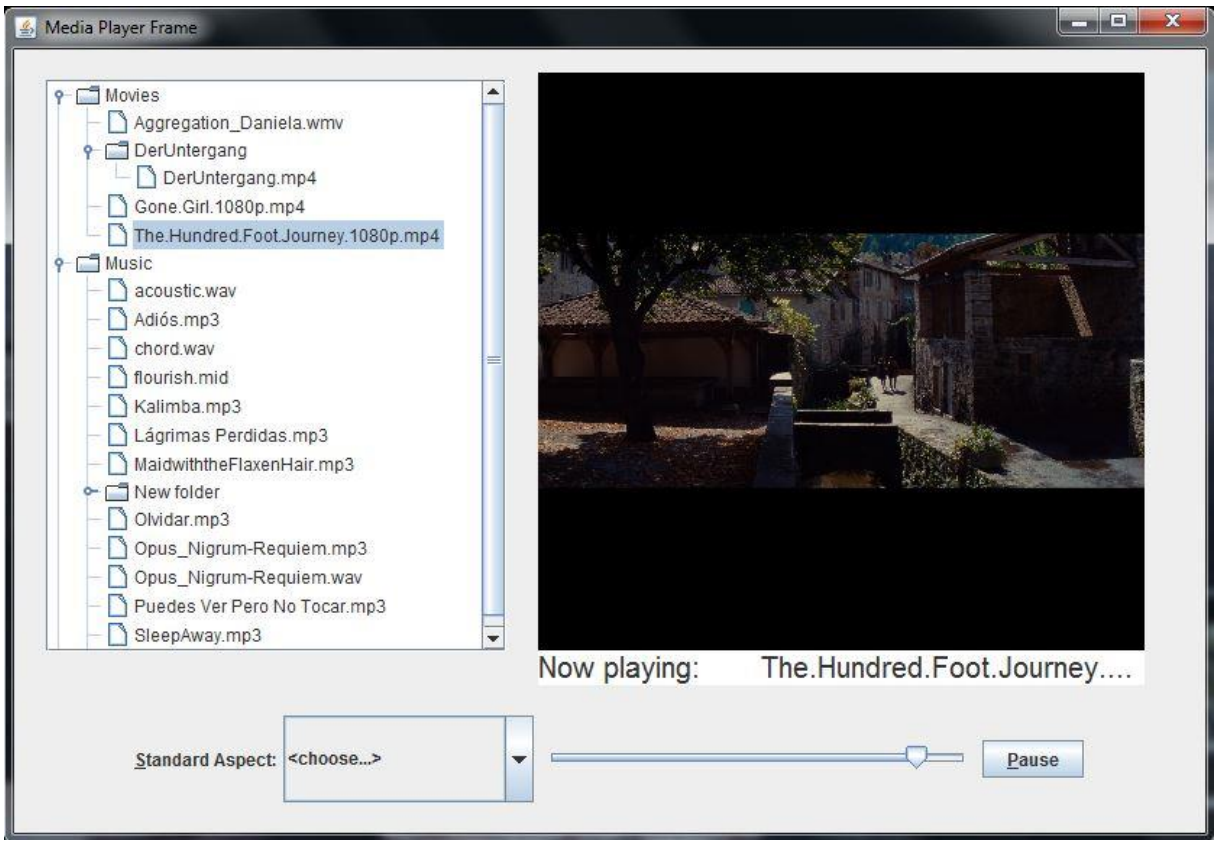


Figure 4.16 - Using JSlider during video streaming

The JSlider part is a more delicate service. When the client initiate the "start streaming" command for a file, the server takes two actions. One is the streaming over RTP, on the client allocated port, the other is a response to HTTP request. This response contains the actual length, in milliseconds, of the media file. The client gets this response and every time a slider event is triggered by the user, it computes the requested millisecond to play from. This millisecond is transmitted to server through another HTTP POST request and the second one will take the corresponding actions.

Beside slider and play/pause commands, we can observe the Standard Aspect JComboBox. Its events interact only with the client application. It is not a command to the server. The only thing this drop-down triggers is the image aspect modification of video display. It contains the standard 16:9, 4:3 and 1:1 aspects.

It must be specified the actions taken when the user closes the application. The client is composed of a single frame with different panels added over it. This frame implements the WindowListener interface which provides methods to enable window events manipulation. In this case, the "windowClosing" method does two things:

- it saves the connection details provided by the user before he presses the "Connect" button. These details are stored in the same properties file the application checks in when it is opened.
- it performs a final HTTP POST request in order to announce the server about connection loss. The server will stop the streaming, delete the user from the field in its singleton class and will also mark the corresponding port as free for future connections.

In the next section it will be presented a client-server flow diagram compacted and shortly presented. The picture correspond to a working properly connection and correct credentials sent by the client.

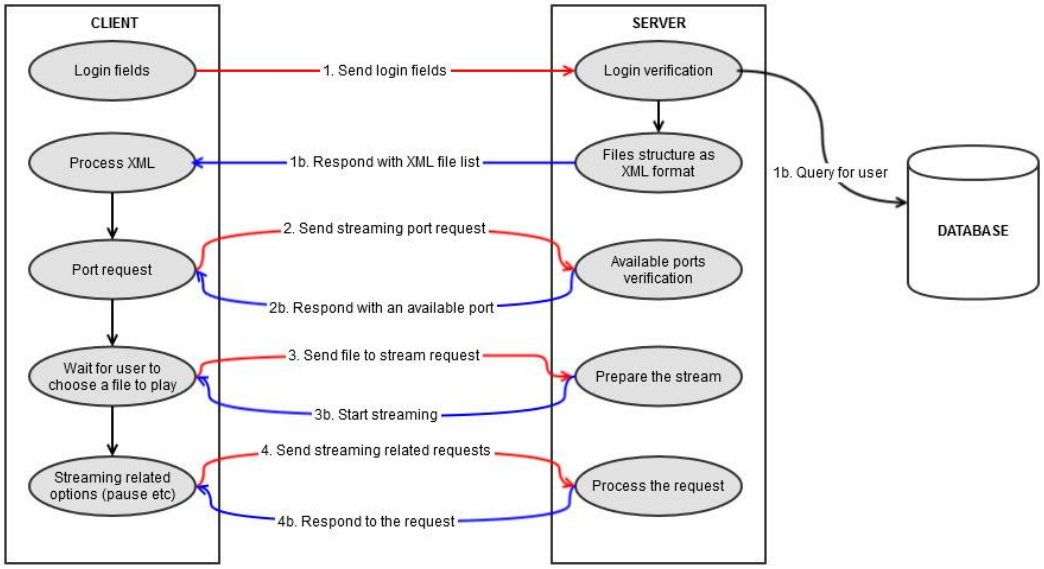


Figure 4.17 - Server - client flow

It can be observed that there are 4 primary steps, each one divided in at least 2 steps. In red are represented the HTTP requests, and in blue, the HTTP responses. Shortly, the client sends login credential to server, the second one interrogates the database, if they are found, the server sends the file list in XML format. The client parses the XML, show its structure in a JTree and send a “port request” command. The server finds a free port and response the client with its number. After this, the client waits for user “file to play” command. The wanted file name is sent to the server which begins the RTP packages transmission and also waits for other commands. The client receives the packages, process them, play them and waits for other user's requests.

5. CONCLUSIONS

The main objective of this thesis was to create a easy to use and easy to manage multimedia streaming application. The standalone application corresponds to this description, being user friendly and intuitive. It is also low performance because everything it must do is to load the VLC library and then it must only decode and display the data received on the designated streaming port. On the other hand, the management of the whole application is reduced to final user credentials manipulation. This represents only the beginning of management, any future implementation can be simply added over the actual stage.

An important advantage is represented by the fact that the implemented software uses designated standards for each specific task:

- REST services communication between server and web interface with the data contained by XML format
- REST services communication dedicated for client to server commands, with XML format when of high amount of data transfer is required.
- streaming packages is sent only over RTP protocol

The advantages described above show that the architecture of the entire application affords probably unlimited upgrades with minimum effort.

From a different perspective, using only standards for any kind of data transfer, "Multimedia streaming and storage web-service" does not restrict the programming languages of clients and servers to Java. The clients can be implemented in any other programming language as long as they use these standards. The same idea is valid for server too.

In terms of future development, as it is presented in the introduction, the current application handles a single service of streaming, that of on demand multimedia live streaming. One of the first future approach would be to implement a streaming web-client, on a separate or on the same web-interface dedicated for users administration. Another type of client can be a mobile application, compatible with any operating system. Also, with the same importance degree is a media upload and download service exposed to users. After this, a well defined table of user privileges must be implemented, for example, a user can define private and public files. The private files must be seen only by himself, while the public ones are exposed to anyone.

Finally, I intend, as future development over other technologies, that this application can be used together with some supervising cameras for home security puproses. In this case, the cameras must act like the server is acting now, sending the filmed images over RTP to the server, where the entire movie is stored and can be accessed by the user, anywhere he will be situated. He will be able to supervise his own house anytime. This would be an improvement in the internet of things concept through which the entire world is converging.

Bibliography

1. "Architectural Styles and the Design of Network- based Software Architectures" by Roy Fielding
2. Herbert Schildt , Java: A Beginner's Guide, Mcgraw-Hill Osborne Media publisher, 6 edition, 2014, pp. 56-80.
3. http://java-virtual-machine.net/download.html , accessed on: 13.04.2015
4. http://mathbits.com/MathBits/Java/Introduction/BriefHistory.htm, accessed on:14.04.2015
5. http://www.freejavaguide.com/history.html, accessed on: 14.04.2015
6. http://www.javatpoint.com/features-of-java, accessed on: 22.04.2015
7. http://www.javaworld.com/article/2077184/core-java/the-lean--mean--virtual-machine.html, accessed on: 23.04.2015
8. http://www.oracle.com/technetwork/articles/javase/index-140168.html#introjb, accessed on: 23.04.2015
9. http://www.oracle.com/technetwork/java/index.html, accessed on: 02.05.2015
10. http://www.postgresql.org/about/, accessed on: 3.05.2015
11. https://www.jcp.org/en/jsr/detail?id=45, accessed on: 20.05.2015
12. Thomas Stamford Raffles, The History of Java, J. Murray publisher, vol. 2, 2005, pp. 23-39.