

C-mulator

Design and development of an educational web
application for teaching C language



Lucía Uguina Gadella

Tutorship: Iria Manuela Estévez Ayres

**Bachelor's Degree in Telecommunication Technologies
Engineering School**

March 2016

"To finish a work?

To finish a picture?

What nonsense!

To finish it means to be through with it, to kill it, to rid it of its soul, to give it its final blow, the coup de grace for the painter as well as for the picture."

Pablo Picasso

Summary

In the Bachelor's Final Project named *C-mulator: Design and development of an educational web application for teaching C language* the analysis, design, development and evaluation process of an educational web application is described.

C-mulator is a tool created with the aim of helping students understand concepts related to Systems Architecture subject.

It is conceived as a Java application that simulates C files generating machine states which change with the code statements. Its main characteristic is the capability of showing the memory condition for each machine state. *C-mulator* also shows the C code and the output of the program. In order to facilitate the use, this Java application was embedded in a web application.

This web application was designed in order to facilitate the use of the *C-mulator* tool. A Client-Server model with a three tiers architecture has been implemented for this project.

After doing a thorough research about the possible technologies that could have been used in the application development, the selected ones were: Apache-Tomcat as the web server, JSON as the client-server communication language, AJAX as the client web technique and MySQL as database management system.

As it has been said before, one of the major functionalities of *C-mulator* are the possibility of simulate C programs in a web based architecture. But *C-mulator* has other characteristics such as having several C files stored in a database that sorted them by chapter and the administrator capability of releasing chapters for specific students' groups.

C-mulator helps the understanding of medium-level abstract concepts that are related with C programming and the Systems Architecture subject.

Acknowledgement

Gracias a mis padres, por apoyarme siempre y haber estado animándome cada vez que perdía las ganas de continuar. Sin vosotros todo este camino no habría sido posible.

Gracias a mi tío y a mi tía, por hacerme sonreír en los momentos más arduos y por acompañarme siempre, sin importar lo duras que fueran las circunstancias.

Por supuesto, gracias a mi tutora, Iria. Que me ha brindado una inestimable ayuda durante todo el desarrollo de este trabajo y ha hecho que me supere día a día.

Agradecer a mis amigos toda su paciencia y sus ganas de aguantarme. Gracias, Carlos, por haberme soportado estos tres años, práctica tras práctica. Gracias, Elena, por esas tardes de risas. Gracias, Christian, Artai, Sandra, Carlota..., todos habéis sido un apoyo inestimable. Sobre todo, gracias Jose, por llevarme de la mano en todo este recorrido.

También brindo este trabajo a todos los que en algún momento de mi vida han hecho que continúe con mis sueños. Sin todas esas personas, yo no habría llegado hasta aquí.

Pero sobre todo, gracias *Yayi*.

Contents

1	Introduction	1
1.1	Context and Motivation	1
1.2	Objectives	3
1.3	Regulatory Framework	3
1.4	Report structure	4
2	State of the Art	6
2.1	Introduction	6
2.2	MVC	7
2.3	Server	7
2.3.1	Database Management System	8
2.3.1.1	Oracle	9
2.3.1.2	Microsoft SQL Server	9
2.3.1.3	MySQL	10
2.3.1.4	IBM DB2	11
2.3.1.5	Microsoft Access	12
2.3.1.6	Comparison	12
2.3.2	Middleware	14
2.3.2.1	Application Server	15
2.3.2.2	Web Server	15
	Apache	16
	Nginx	16
	Tomcat	17
	Comparison	17
2.4	Client	18
2.4.1	HTML/XHTML	18
2.4.2	JavaScript	19
2.4.3	Shockwave Flash	20
2.4.4	AJAX	20

2.4.5	Comparison	20
2.5	Client-Server Communication	22
2.5.1	XML	22
2.5.2	YAML	22
2.5.3	JSON	22
2.5.4	Comparison	23
2.6	Related Work	23
2.6.1	A crowd of little man computers: visual computer simulator teaching tools	23
2.6.2	DCMSim: Didactic cache memory simulator	25
2.6.3	DRAMSim2: A cycle accurate memory system simulator	25
2.6.4	Teaching computer architecture/organisation using simulators	25
2.6.5	Design and Evaluation of a Cache Memory Simulation Program	26
2.6.6	Conclusion	26
3	Project Description	27
3.1	Introduction	27
3.2	Requirements	27
3.2.1	C Language Subset	28
3.2.2	Application Requirements	29
3.3	Design	30
3.4	Use cases	30
3.4.1	Simulation	32
3.4.2	Database	32
3.4.3	Server	33
3.4.3.1	Servlet	33
3.4.4	Client Design	34
3.4.5	Client-Server Communication Design	34
3.4.6	Summarize	34
4	Implementation	37
4.1	Introduction	37
4.2	Selected Technologies	37
4.2.1	Server	37
4.2.2	Client	39
4.2.3	Client-Server Communication	39
4.2.4	Conclusion	39
4.3	Simulation part	39

4.3.1	Introduction	40
4.3.2	Program behaviour	40
4.3.2.1	Serialization	42
4.3.2.2	ALU	43
4.3.2.3	IO	44
4.3.2.4	BUS	44
4.3.2.5	RAM	45
4.3.2.6	CPU	46
4.4	Data Model	51
4.5	Server	52
4.5.1	Access System	52
4.5.2	Servlet	53
4.6	Client	54
4.6.1	Access System	54
4.6.2	File selection system	56
4.6.3	Simulation part	57
4.6.4	Disconnect	60
4.6.5	Information pages	60
5	Validation	61
5.1	Introduction	61
5.2	Obtained results	61
5.2.1	Test 1	61
5.2.2	Test 2	62
5.2.3	Test 3	64
5.2.4	Test 4	64
5.2.5	Test 5	65
5.2.6	Test Comparison	65
5.3	Conclusion	67
6	Conclusions and Future Work	68
6.1	Introduction	68
6.2	Conclusions	68
6.3	Future Work	69
7	Planning and Budget	70
7.1	Planning	70
7.1.1	Gantt Chart	72
7.2	Budget	72

Appendix A Raspberry Configuration	77
1 Installing Raspbian	77
2 Installing and Configuring Apache-Tomcat	77
3 Installing and Configuring MySQL and phpMyAdmin	79
4 Configuring Hypertext Preprocessor (PHP) in Tomcat	82
5 TLS Configuration	83
6 Application Deployment	84

Bibliography

List of Figures

1.1	<i>Percentages of student respondents (n = 64) who preferred to use visual, aural, read-write, kinaesthetic, and multiple sensory modalities when learning information [3]</i>	2
2.1	<i>Three tiers architecture</i>	6
2.2	<i>Model View Controller (MVC) view [12]</i>	7
2.3	<i>Average Execution Time</i>	13
2.4	<i>Average Central Processing Unit (CPU) Utilization</i>	13
2.5	<i>Average Memory Usage</i>	13
2.6	<i>Basic presentation of an application server and its environment. Figure from [23]</i>	15
2.7	<i>How Asynchronous JavaScript And XML (AJAX) works</i>	21
3.1	<i>Application Scheme</i>	30
3.2	<i>Use Cases</i>	31
3.3	<i>Simulator tool behaviour</i>	32
3.4	<i>Database Structure</i>	33
3.5	<i>Server Sections</i>	34
3.6	<i>Web Page Views</i>	35
3.7	<i>Design Scheme</i>	35
4.1	<i>Raspberry Pi comparative</i>	38
4.2	<i>Class Diagram</i>	41
4.3	<i>Serialization Flowchart</i>	42
4.4	<i>CPU flow diagram</i>	47
4.5	<i>deleteSpaces flowchart</i>	48
4.6	<i>tagger flow diagram</i>	49
4.7	<i>Access System</i>	55
4.8	<i>Check login behaviour</i>	55
4.9	<i>File selection system</i>	57
4.10	<i>Simulation behaviour</i>	58

5.1	<i>Login</i>	61
5.2	<i>Invalid Username or Password</i>	62
5.3	<i>Not Allowed User</i>	62
5.4	<i>Navigation bar</i>	62
5.5	<i>Chapter Selection</i>	63
5.6	<i>File Selection</i>	63
5.7	<i>Chapters for Students</i>	63
5.8	<i>Simulation Page</i>	64
5.9	<i>Contact Information</i>	64
5.10	<i>C-mulator in class</i>	67
5.11	<i>C-mulator in class</i>	67
7.1	<i>Gantt Chart</i>	73
A.1	<i>Tomcat Welcome Page</i>	78
A.2	<i>MySQL Configuration</i>	80
A.3	<i>phpMyAdmin Configuration</i>	80
A.4	<i>phpMyAdmin Configuration</i>	81
A.5	<i>phpMyAdmin Configuration</i>	81
A.6	<i>phpMyAdmin Configuration</i>	81
A.7	<i>phpMyAdmin Configuration</i>	82

List of Tables

2.1	Server Types [13]	8
2.2	Relational Database Management Systems (RDBMS)s comparison	14
2.3	RDBMSs comparison	14
2.4	Web Servers comparison	17
2.5	Web Servers comparison	18
2.6	Client Technologies Comparison	21
2.7	Client-Server Communication Technologies Comparison	23
2.8	Related Work Comparison	24
4.1	Selected Technologies	40
4.2	LDAP code	53
4.3	HyperText Markup Language (HTML) header	54
4.4	Selection Query	56
4.5	Logging User Activity	60
5.1	Generating Log File	65
5.2	Tests Comparison	66
7.1	Previous Study Duration	70
7.2	Simulation Block Duration	71
7.3	Client Block Duration	71
7.4	Database and Server Block Duration	71
7.5	Test Phase Block Duration	72
7.6	Report Duration	72
7.7	Project Duration	74
7.8	Staff Cost	75
7.9	Equipment Cost	75
7.10	Amortization	75
7.11	Total cost	76
7.12	Total cost with taxes	76

A.1	Tomcat Installation	78
A.2	Java SE Development Kit (JDK) Installation	78
A.3	Update System	79
A.4	MySQL Installation	79
A.5	PHP and phpMyAdmin Installation	79
A.6	JavaBridge Configuration	83
A.7	Keystore Creation	84
A.8	Keystore Creation	84

Glossary

ADO Automatic Data Optimization.

AEPD Agencia Española de Protección de Datos.

AJAX Asynchronous JavaScript And XML.

ALU Arithmetic Logic Unit.

API Application Programming Interface.

ASM Automatic Storage Management.

CPU Central Processing Unit.

CSS Cascading Style Sheets.

DBMS Database Management System.

EJB Enterprise JavaBean.

FIFO First In First Out.

GUI Graphic User Interface.

HDMI High-Definition Multimedia Interface.

HTML HyperText Markup Language.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IIS Internet Information Services.

IO Input/Output.

Java EE Java Platform Enterprise Edition.

JavaCC Java Compiler-Compiler.

JDK Java SE Development Kit.

JSON JavaScript Object Notation.

JSP JavaServer Pages.

LDAP Lightweight Directory Access Protocol.

LMC Little Man Computer.

LOPD Ley Orgánica 15/1999 de Protección de Datos.

LRU Least Recently Used.

MVC Model View Controller.

ODBC Open Database Connectivity.

PDF Portable Document Format.

PHP Hypertext Preprocessor.

RAM Random Access Memory.

RDBMS Relational Database Management Systems.

RGPD Registro General de Protección de Datos.

RPC Remote Procedure Call.

SD Secure Digital.

SQL Structured Query Language.

SWF Shockwave Flash.

TLS Transport Layer Security.

URL Uniform Resource Locator.

W3C World Wide Web Consortium.

XHTML eXtensible Hypertext Markup Language.

XML eXtensible Markup Language.

XPS XML Paper Specification.

YAML YAML Ain't Markup Language.

Chapter 1

Introduction

This chapter introduces the objectives and motivations of this project. It will also give a brief description of the project structure.

1.1 Context and Motivation

During the last century, the educational research field has developed different theories trying to bring light on how humans learn and how to make the learning process more effective . For example, in 1956, Bloom [1] proposed their taxonomy of educational objectives, where the authors try to convey and classify educational objectives into levels of complexity and mastery. More specifically, in the cognitive domain this taxonomy recognises the following ordered categories: knowledge, comprehension, application, analysis, synthesis and evaluation. Although there are controversies around how the higher levels are ordered and several other authors [1] reviewed the original taxonomy, there is consensus about the complexity of the higher levels. More specifically when it came to the learning of critical and logical thinking.

Additionally to this inherent difficulty, when the teachers have to explain some logical thoughts there could be some students that get it right away and some of them that could get lost. This is due to the fact that people do not learn in same ways.

A *learning style model* classifies students according to where they fit on a number of scales pertaining to the ways they receive and process information [2].

Depending on the student, its way of learning will be different from the others. Because of that, the appropriate solution for getting the best of the students is giving them the right tools for any of their learning styles. While some of the students need the out-loud explanation of the teacher, others learn better if they can see what the

teacher is explaining on the blackboard as a picture. Some others just need a written explanation. And, finally, there are students that learn by themselves, just trying the exercises and practising.

Nevertheless, there are some students that have a *multi-modal* way of learning. It has been researched by other scientists that more than a half of the students have a multiple sensory modality [3] as it is shown in Figure 1.1.

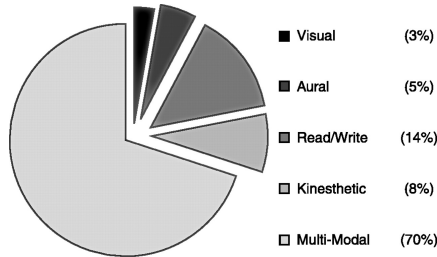


Figure 1.1: *Percentages of student respondents (n = 64) who preferred to use visual, aural, read-write, kinaesthetic, and multiple sensory modalities when learning information [3]*

Felder exposed that there are several types of learning styles. There is a *perception* dimension which could be sensory or intuitive. *Input* dimension is divided in visual or auditory. *Organization* could be inductive or deductive. *Processing* could be done actively or reflectively. And finally the *understanding* dimension is divided in sequential or global [2].

Due to the disclosed before, the strategies and tools used in education have changed in time. The teachers ways of speaking and implication with their students are different now than thirty years ago. Since the technology introduction in classrooms, the way of teaching and learning has changed from bottom to top.

Some studies have found a correlation between the use of technology in the classroom and an improvement in the students achievement from the lessons [4].

Even though, this does not mean that learning a computer language is less demanding. With a language close to the machine one, it is more difficult to understand widely what is happening in the computer while the program is running. This is due to the high abstract thinking that is required in programming subjects [5].

This situation does not motivate students in subjects related to Computer Science [5]. The logical difficulty is high and, to our knowledge, there are no tools that facilitate the part of understanding the processes running in the computer. Furthermore, the most difficult part of understanding a programming language is when it is related to pointers and memory handling, as they are abstract concepts [5].

The students' performance could be enhanced with computer simulations if it is compared with a lack of them. Researchers strongly recommend the use of simulations as a supplementary tool to the traditional lectures, in order to help students defy its barriers in cognitive learning [6].

As a solution, computer simulations go along with the lines exposed before, *C-mulator* comes up to help students with its difficulties learning the C programming language within the scope of a second-year programming course of an engineering bachelor degree, namely the Systems Architecture subject in the Bachelor's Degree in Telecommunications Technologies.

1.2 Objectives

This project consists in developing the simulation tool *C-mulator*, which is embedded in a web application that simplifies the usage of this tool. It solves the problems defined above, trying at the same time to improve the students' performance in the subject.

This application needs to cover the simulation of several files stored in the system but organized in a database. It will also implement a log-in system in order to monitor with a log file the user that is testing the application.

The preliminary objectives of this project are:

- Study of the technologies that can be applied in order to achieve the final objective of the project.
- Analysis, design and development of the application system.
- Carrying on the necessary tests on the application to prove that it works according to the specifications.
- Distribution of the tool to an audience and study of its benefits in a classroom.
- Establishment of project future lines in order to continue and improve the tool.

1.3 Regulatory Framework

This project will manage some data about the users of the application. Due to this it is important to talk about the Ley Orgánica 15/1999 de Protección de Datos (LOPD). The organization that is in charge of data protection compliance in a national level

is the Agencia Española de Protección de Datos (AEPD). The LOPD objective is to guarantee and protect, in what is related with the personal data treatment, people's public freedoms and fundamental rights, and specially their honour and personal and familiar privacy [7].

When someone is treating with files containing sensitive data, those files have to be registered in the Registro General de Protección de Datos (RGPD) [7].

Besides, this law establishes three levels of security: low, medium and high. These depend on the nature of the files and if they contain sensitive information and how much information related to personal data is stored [8]:

- **HIGH LEVEL:** files or data that contain information about several topics such as health, ideology, sexual behaviour or religion.
- **MEDIUM LEVEL:** files or data that contain information about administrative infractions related with the 29th article of the LOPD, economic data, fiscal administrations, etc.
- **LOW LEVEL:** any other data or file that contain any personal information, such as phone, address, age or name.

The level of security of *C-mulator* is the lowest one, as it only stores the university username (a number). Moreover, all the login process is done through the university servers, implying that *C-mulator* does not store any sensitive password. *C-mulator* generates and stores user logs that will be used for research purposes and it runs in an university server within the University facilities.

Due to the fact that this data is generated by the university and it is below the protection of the LOPD, the files do not need to be registered [9], as they are already gathered by the university.

1.4 Report structure

This report is divided in several chapters:

1. This chapter, *Introduction*, explained the motivation and social context of this project, its objectives and the regulatory framework that surrounds it.
2. The *State of the Art* chapter reviews the different possible technologies that can be used to implement the project, along with a revision of the related work.

3. The *Project Description* chapter presents the requirements and the design of the application.
4. The *Implementation* chapter uses the previous study performed in chapter 2 to choose the specific technologies to be used. Moreover, all the application operation is explained here step by step.
5. The *Validation* shows how the tool works and it proves that the requirements are fulfilled.
6. In *Conclusions and Future Work* the obtained conclusions of developing the project are described along with the future lines that can be carried out.
7. The *Planning and Budget* chapter describes the time planning of the project and its budget.

Chapter 2

State of the Art

2.1 Introduction

This chapter presents the carried-out study about the different technologies that can be selected to implement this project.

Although the core of the *C-mulator* tool is a simulator of the execution of a program over a specific architecture, one of the main requirements of the project is the development of such simulator as a web application. This constraint conditions the set of technologies to be studied and also the structure of the developed application.

Thus, in this project, the Client-Server model will be used along with a three tiers architecture (see Figure 2.1).

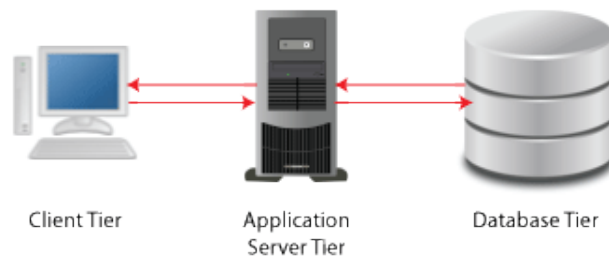


Figure 2.1: *Three tiers architecture*

In a Client-Server architecture, the client, a local computer, sends a request to the server, an application that offers a service to internet users. This request is processed by the server which returns the results to the client.

The three tiers architecture, as shown in Figure 2.1, includes a database server

which stores structured information. This database can be accessed by the server in order to extract data. In this architecture, Remote Procedure Call (RPC) calls from presentation client to middle-tier server provide greater overall system flexibility than the Structured Query Language (SQL) calls made by clients in the two-tier architecture. Therefore, the client presentation does not need to 'speak' SQL [10].

The three tiers architecture also provides for more-flexible resource allocation. Middle-tier functionality servers are highly portable and can be dynamically allocated and shifted as the needs of the application change [10].

Apart from the architecture, it is also remarkable to study this technologies in a MVC pattern because this system makes maintenance and testing simple and easier [11]. This simplicity is due to the separation of the HTML from the rest of the application.

2.2 MVC

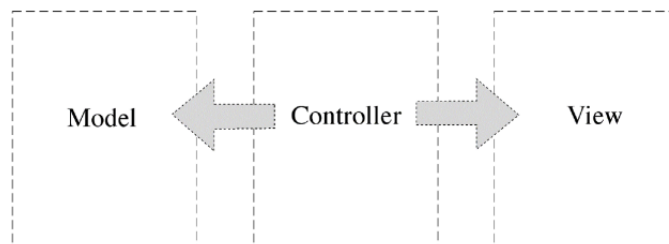


Figure 2.2: *MVC view* [12]

Usually, if an application follows the MVC pattern, it means that it will be split in three different parts [11], as shown in 2.2:

- *Models*: which contain or represent the data that the user works with.
- *Views*: which are used to render some part of the model as a user interface.
- *Controllers*: which process incoming requests, perform operations on the model, and select views to render to the user.

2.3 Server

There are several types of servers, the relation is shown in Table 2.1.

Server Type	Definition
Mail Server	A server that stores, send and receive mails from and to its clients.
Proxy Server	This server is located between a client and a conventional server. It processes the request and send it like an anonymous user, so the final server does not know who has send the request.
Web Server	Stores and sends to clients several types of information, such as HTML files, images, videos...
Database Server	Allows database storing and management to its clients.
Image Server	Supports a high number of images without consuming lots of resources of a web server.
Dedicated Server	Shared server.
Cluster	Group of servers that stores large amounts of information, foreseeing a lost of it by creating backups in other servers.

Table 2.1: Server Types [13]

As explained above, the server in this project will be a Web Server. In addition to the web server that is going to give the user access to the application, this project also consists of a database management server which will allow the access to organized C files related to the tool.

2.3.1 Database Management System

First of all, a database can be defined as a collection of related data from which users can efficiently retrieve the desired information [14]. This access is mainly done by a Database Management System (DBMS) which is an integrated set of programs used to create and maintain a database. The main objective of a DBMS is to provide a convenient and effective method of defining, storing, retrieving and manipulating the data contained in the database [14].

There exist several types of DBMS but in this report only RDBMS will be studied as the data to be stored in the *C-mulator* databases is related and susceptible of being stored in bi-dimensional tables.

The next section presents a review of the most popular RDBMS (Oracle, Microsoft SQL, MySQL, IBM DB2 and Microsoft Access). Notice that all of them use SQL for managing data.

2.3.1.1 Oracle

Oracle has not only a RDBMS but it also has its own database. The Oracle Database comprises the database and, at least, an instance of the application. An instance comprises a set of operating system processes and memory structures that interact with the storage [15].

This database has also a system of online redo logs, which are two or more files that store the database changes, that can be turned into archive logs which will allow a data recovery or data replication, depending on the situation. The data storage is done logically with table-spaces and physically with data files [15].

Its main characteristics are [16]:

- Client/Server model.
- Multi-tenant architecture which is intended for working with multiple and separated Pluggable Databases.
- Possibility of migration from IBM DB2 to Oracle.
- Interval and reference partitioning.
- Statistics automatically generated.
- Automatic Data Optimization (ADO) which provides the ability to automate compression and movement of data.
- Privilege analysis along with Real Application Security, these monitor the users privilege as well as the assignation of those privileges.
- Multi-user Concurrency with very large groups of users without noticing contention.
- Highly availability thanks to features like Automatic Storage Management (ASM) Servers, Data Guard or Point-in-time Recovery.
- High scalability.

2.3.1.2 Microsoft SQL Server

Microsoft SQL Server is a relational database management system that supports the implementation of clusters and mirroring.

A cluster is a set of SQL Servers working in parallel and identically distributed, that enables to perform load balancing.

Apart from supporting data mirroring, it also includes a data partitioning system for distributed databases [15].

Its language is not SQL as such, it is Transaction-SQL, which is an implementation of the standard SQL. SQL Server also supports atomic, durable, consisted and isolated transactions [15].

Its main characteristics are [17]:

- Handle data in sets.
- Partitioned tables segments data across multiple file-groups.
- It allows both clustering and mirroring. But it has only one copy of the database, that can be accessed by two or three servers at the same time [18].
- Scheduled backups with logs [18].
- SQL Replication allows granular data at a table level [18].
- It has access control and data protection with tools such as integrated cryptography and nested roles.
- Concurrency and Cloud Concurrency.
- Client/Server model.

2.3.1.3 MySQL

MySQL is an open-source and free RDBMS. It has a client/server architecture, as the majority of database systems, which allows it to have the database server in one computer and the management system in another, just connected through network. It is compatible with various databases, though its primary language is standard SQL [19].

This management system is quite popular, with more than ten million installations. It is also multi-user and multi-threaded [15].

Its main characteristics are [19]:

- Easy to use as it is popular and there are several GUIs (Graphical User Interface) for MySQL.
- Multi-platform support.
- Client/Server architecture.

- Several Application Programming Interface (API) for different languages.
- Standard SQL as its native language.
- Stored Procedures for simplification and triggers.
- Full-text search.
- Replication of data and transactions, this means that several operations are executed as a block.
- Foreign key constraints.
- Supports Open Database Connectivity (ODBC).

2.3.1.4 IBM DB2

This RDBMS can be accessed through a terminal or a Graphic User Interface (GUI). The GUI is a Java client that eases the learning of the tool to novice administrators, as the command-line interface requires a greater knowledge of the product. However, the command-line interface is more powerful, allowing the implementation of scripts and automated processes [15].

IBM DB2 can be run in Linux, Unix and Windows systems. It can be integrated in Eclipse or Visual Studio .NET and stores eXtensible Markup Language (XML) data with XQuery natively. This database system also provides the programmer with an error processing tool for SQL statements [15].

Its main characteristics are [20]:

- Client/Server architecture.
- Database-aware clustering solution.
- Portability from other databases.
- Transparent automated fail-over, automatic workload balancing and automatic client reroute capabilities.
- Multi-platform.
- Shared-storage formatter for a shared-everything architecture.
- Built-in capabilities for formatting, adding or deleting disks.

2.3.1.5 Microsoft Access

Microsoft Access is a relational database management system which combines the Microsoft Jet Database Engine with a graphical user interface. This RDBMS does not supports triggers or stored procedures. It is mainly intended for application development prototypes as this RDBMS is easy to use and fast for programming development [15]. Moreover, Microsoft Access is a *file-server* system not only a client/server system as the others, which has a poor performance in a network scenario but a higher one if they are installed (both database and the system) in the same machine [19].

Its main characteristics are [21]:

- File-server architecture, not Client/Server one.
- Easy to use due to its native GUI.
- If the security requisites are high, it is not recommended to use Microsoft Access.
- Reduced scalability.
- Access from web apps. It is able to modify the database from the browser.
- Database templates.
- Export to Portable Document Format (PDF) and XML Paper Specification (XPS) files database reports.

2.3.1.6 Comparison

The following Figures, 2.3, 2.4, 2.5; are extracted from [15] and compare the performance of the RDBMS for several queries.

The Tables 2.2 and 2.3 shows the comparison between RDBMSs. In the tables, ✓ means that that RDBMS has that characteristic, if the following symbol is shown, ✓✓, it means that the characteristic is fully fulfilled.

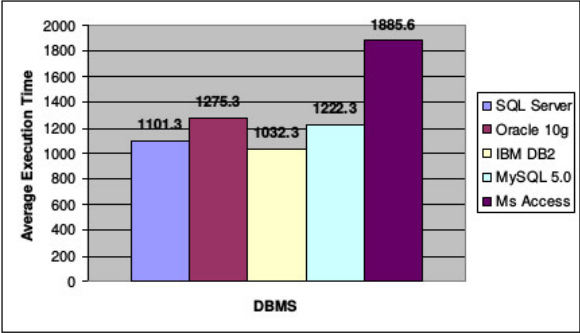


Figure 2.3: Average Execution Time

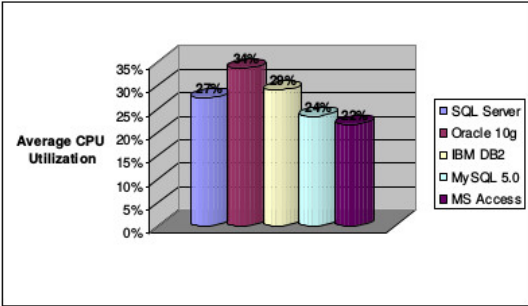


Figure 2.4: Average CPU Utilization

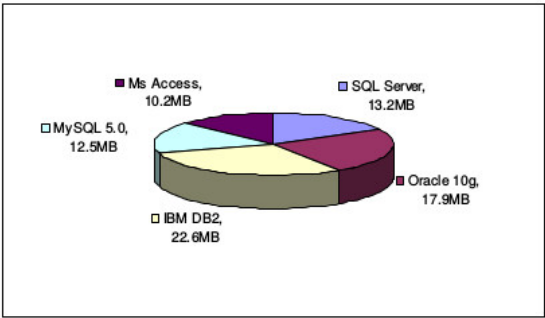


Figure 2.5: Average Memory Usage

RDBMS	Velocity	Strength	Capacity	Security	Reduced Memory Usage
<i>Oracle</i>		✓✓	✓✓	✓✓	
<i>SQL Server</i>	✓		✓	✓	
<i>MySQL</i>	✓	✓✓	✓✓		✓
<i>IBM DB2</i>	✓✓	✓	✓	✓✓	
<i>Microsoft Access</i>		✓			✓✓

Table 2.2: RDBMSs comparison

RDBMS	Multi-platform	Scalability	Transactions Support	Function Definition Support
<i>Oracle</i>	✓✓	✓✓	✓	✓✓
<i>SQL Server</i>		✓	✓	✓
<i>MySQL</i>	✓✓	✓	✓	
<i>IBM DB2</i>	✓	✓	✓✓	✓✓
<i>Microsoft Access</i>			✓	✓

Table 2.3: RDBMSs comparison

As it is stated in both tables and in charts, the *Oracle* database is the most powerful one, with great security and multiple tools and extensions. But this database is not open-source nor free to use. The cost of this database is really high as well as the *SQL Server* RDBMS cost.

The next option should be *IBM DB2*. Since *DB2* has also a great cost is discharged too.

Due to the exposed before, the options are MySQL or Microsoft Access. Access is visual and easy to use, mainly intended for novice programmers or for a small enterprise [21]. On the other part, MySQL is intended for systems in which the security is not a real matter and capacity and strength are highly required [19].

2.3.2 Middleware

Middleware is sometimes described as the software layer between the application and the operating system [22]. So, in this section, the project web and application servers

are going to be analysed.

2.3.2.1 Application Server

An application server provides the infrastructure for executing applications that run a project or business [23]. It serves as a platform for developing web services and usually refers to a Java Platform Enterprise Edition (Java EE) application server [23].

The application server acts as a middleware between back-end systems and clients, as can be seen in Figure 2.6. It provides a programming model, an infrastructure framework, and a set of standards for a consistent design link between them [23].

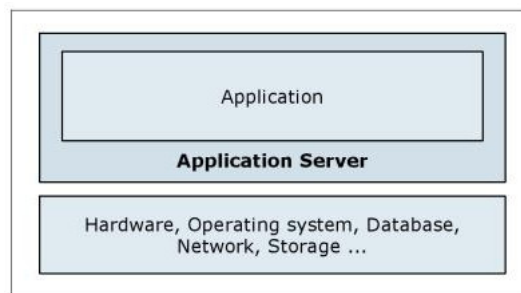


Figure 2.6: *Basic presentation of an application server and its environment. Figure from [23]*

Some of these application servers are WebSphere from IBM, WebLogic from Oracle, JBoss AS from JBoss, Geronimo from Apache and some others.

Unless Tomcat may seem an Application Server, it is only a Web Application Server Container. Tomcat is designed for running Servlets and JavaServer Pages (JSP) but not Enterprise JavaBean (EJB) [24]. Due to this, it will be analysed in 2.3.2.2.

2.3.2.2 Web Server

A web server main function is translating an Uniform Resource Locator (URL) either into a program name, run it and throw back its output, or into a file name, and send the file back to the client [25].

This web server *listens* to a port, the Hypertext Transfer Protocol (HTTP) one is 80, waiting for any connection to arrive. Then, when this connection reaches the server, it fulfils the request and throws back the output [25].

In this report, the web servers that are going to be analysed are: *Apache*, *lighttpd*, *Internet Information Services (IIS)* and *Nginx*.

Apache

Apache is the most popular web server, almost the 50 percent of active websites use Apache [26]. It is open-source, secure and stable [25]. Apache is a quite reliable web server as it is open-source, that is, the source code can be examined by anyone who wish to take a look at it, so if it has any error or bug, there would be thousands of users looking for the bug in order to correct it [25].

Its main characteristics are [25]:

- Open-source and free to use.
- Secure and stable due it is open-source.
- It is flexible, it is intended for small sites as well as large ones.
- Popular and easy to use. There are lots of pages with tutorials related to Apache.
- Compatible with Windows, Linux and Mac.

Nginx

Nginx is an open-source high-performance web server [27]. Nginx is not the web server leader, as it is Apache, but it has gained some popularity in some of the most visited sites, such as Facebook, Netflix or WordPress [28].

Nginx has been chose as a load-balancing proxy server by lots of smaller Web 2.0 companies due to its modular architecture and small footprint [27].

Nginx main characteristics are [28]:

- Efficient and lightweight.
- It has asynchronous sockets.
- Ease of use.
- Modularity.
- Compatible with Windows and Linux.

Tomcat

Tomcat is a Java servlet container and a web server developed by the Apache Software Foundation. It provides both Java servlet and JSP technologies in addition to serving traditional static web pages [29].

Tomcat is a good choice if the objective is developing a web application or using a JSP or Java servlet engine. It is free and open-source and can be also used in conjunction with other web servers such Apache [29].

Tomcat main characteristics are [29]:

- Java servlet container and JSP.
- Open-source and free.
- Clustering.
- Fewer web server features than Apache.
- Compatible with Windows, Linux and Mac.

Comparison

Tables 2.4 and 2.5 show the comparison between the three web servers, attending to their speed, Hypertext Transfer Protocol Secure (HTTPS) support, authentication availability, Java Servlets technologies and Server-Side JavaScript in the first table. In the second one the multi-platform, scalability, popularity and development language features will be compared.

In these tables ✓ means that the characteristic is present in the web server. A ✓✓ means that the characteristic is fulfilled by far in that web server.

Web Server	Speed	JavaScript	HTTPS	Auth.	Java Servlets
<i>Apache</i>	✓	✓✓	✓✓	✓✓	
<i>Nginx</i>	✓✓	✓	✓✓	✓	
<i>Tomcat</i>	✓	✓✓	✓✓	✓	✓✓

Table 2.4: Web Servers comparison

Even though, Nginx is a powerful and fast web server, it does not have a large amount of users and developers behind it. So the options will be Apache or Tomcat depending on the architecture of the application. If this application is going to be

RDBMS	Multi-platform	Scalability	Popularity	Language
<i>Apache</i>	✓✓	✓✓	✓✓	C
<i>Nginx</i>	✓	✓	✓	C
<i>Tomcat</i>	✓✓	✓	✓	Java

Table 2.5: Web Servers comparison

developed with Servlets, Tomcat will be the chosen option, if not, Apache will be the web server.

2.4 Client

In this section the technologies related with the client part will be analysed.

As in the tree tiers architecture the client is completely separated from the server, it has to be analysed in a different way.

The client presentation is a set of programs that send requests and receive responses to and from the server [10]. The responses can be static documents (HTML), multimedia content (Flash), interactive content (AJAX), dynamic documents (JavaScript)...

As these technologies allow the client to execute some processes, this alleviates the server load processing.

2.4.1 HTML/XHTML

HTML is a collection of standard tags and rules for identifying web content in a way that enables web browsers to render web pages properly. The **hyper** in *HyperText* refers to the fact that web pages typically contain interactive links that connect to other content on the same or different sites [30].

There have been several versions of HTML since the Web began, and the development of the language is overseen by an organisation called the World Wide Web Consortium (W3C) [31].

One of the major versions of HTML was the 4.01 (December 2009). In January 2000, some stricter rules were added to HTML 4.01, creating what is known as eXtensible Hypertext Markup Language (XHTML) [31].

Nevertheless, HTML has a new major version HTML 5. This version introduces several improvements over earlier standards, providing better ways to structure doc-

uments and make them more animated and interactive [30]. Some of these improvements are:

- **Simple document type declaration (DTD)**: in HTML5 it can be declared as `<!DOCTYPE html>`
- **`<audio>` and `<video>` tags** : These tags simplify the process of placing audio and video content on sites and enable the function of adding text, such as the transcript of a video for the visually impaired.
- **New interactive elements** :
 - `<details>` : it is used for adding information about an element that can be displayed when a mouse pointer is over it.
 - `<data grid>` : creates an interactive table.
 - `<menu>` : used for creating toolbars or context menus.
- **New `async` attribute** : this attribute tells the browser not to wait until a particular script loads before displaying other elements on the page, thus eliminating delays often caused by scripts.

2.4.2 JavaScript

JavaScript is an interpreted language used for creating dynamic effects in web pages, such as interacting with users, getting information from them, and validating their actions.

JavaScript is not the only scripting language; there are others such as VBScript and Perl.

The main reason for choosing JavaScript is its widespread use and availability. Both of the most commonly used browsers, IE and Firefox, support JavaScript, as do almost all of the less commonly used browsers.

JavaScript is not the script version of the Java language. In fact, although they share the same name, that's virtually all they do share. Particularly good news is that JavaScript is much, much easier to learn and use than Java. In fact, languages like JavaScript are the easiest of all languages to learn, but they are still surprisingly powerful [32].

2.4.3 Shockwave Flash

Shockwave Flash (SWF) is a proprietary file format developed by Adobe to deliver multimedia and vector graphics to the Web. A SWF file contains the video, audio, animations, interactive scripts, program controls and other features related with user interactions.

SWF is less intuitive than JavaScript and XML/HTML because it requires several steps between the coding process and the implementation in a web page.

With the new HTML standard, HTML5, SWF could get extinct as their functionalities are covered in HTML5 [33].

2.4.4 AJAX

AJAX is a group of interrelated web-programming technologies that can send and retrieve data in the background, without having to reload the page [34].

AJAX offers a technique to make background server calls via JavaScript and retrieve additional data as needed, updating portions of the page without causing full page reloads [35].

Some of the benefits of AJAX are [35]:

- It makes it possible to create responsive and intuitive web applications.
- It encourages the development of patterns and frameworks that reduce the development time of common tasks.
- It makes use of the existing technologies and features that are already supported by all modern web browsers.
- It makes use of many existing developer skills.

The AJAX operation is described in Figure 2.7.

2.4.5 Comparison

HTML/XHTML is intended for static web pages, although HTML5 provides several improvements. JavaScript is used for creating dynamic web pages as well as SWF, but this last one is quite more complex than JavaScript.

Finally, AJAX is a mixture of several technologies, including JavaScript, HTML and XML.

In the Table 2.6, the described technologies are compared in a more visual way. As before, a ✓ means that the characteristic is fulfilled by the technology.

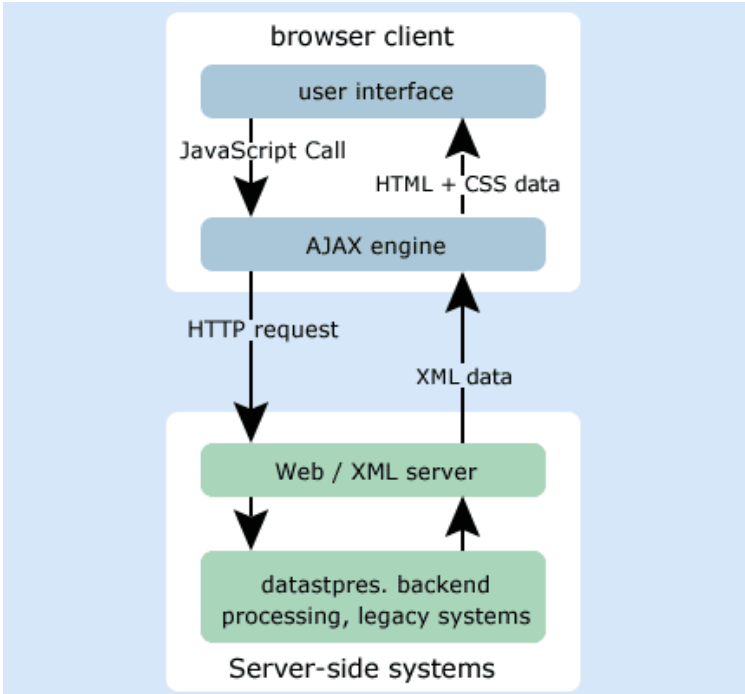


Figure 2.7: How AJAX works

	Ease of Use	Dinamic	Secure	Multimedia	Interactive
<i>HTML</i>	✓	✓		✓(HTML5)	
<i>JavaScript</i>	✓	✓	✓		✓
<i>Shockwave Flash</i>				✓	
<i>AJAX</i>	✓	✓	✓		✓

Table 2.6: Client Technologies Comparison

2.5 Client-Server Communication

As in this project is desired to exchange data between server and client, it is important to choose the information format. In this section some of the most used formats are going to be compared.

2.5.1 XML

XML describes a class of data objects called XML documents and partially describes the behaviour of programs which process them.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, which form character data or markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.

XML processor is used to read XML documents. The application describes the required behaviour of an XML processor.

It is used in complex communication systems and when the data is relevant [36].

2.5.2 YAML

YAML Ain't Markup Language (YAML) is a data serialization language designed to be human-friendly and work well with modern programming languages for common everyday task.

YAML was designed from the start to be useful and friendly to people working with data. It uses Unicode printable characters, some of which provide structural information and the rest containing the data itself. YAML achieves a unique cleanness by minimising the amount of structural characters and allowing the data to show itself in a natural and meaningful way. For example, indentation may be used for structure, colons separate key: value pairs, and dashes are used to create "bullet" lists [37].

2.5.3 JSON

JavaScript Object Notation (JSON) is a data format derived from the literals of the JavaScript programming language. This means that JSON is a subset of the JavaScript language. Even though, JSON is a subset of a programming language, it is not a programming language but, in fact, a data interchange format [38].

JSON is today's standard in data formatting for the web. It implies that it can be used as the data format wherever the exchange of data occurs [38].

2.5.4 Comparison

In Table 2.7 the mentioned technologies for Client-Server Communication are compared.

	Legible	Ordered	Simple	Development Tools
<i>XML</i>	✓	✓	✓	✓✓
<i>YAML</i>	✓	✓		✓
<i>JSON</i>	✓✓	✓✓	✓✓	✓

Table 2.7: Client-Server Communication Technologies Comparison

As it can be seen in the table 2.7, the XML language has the best support and a huge variety of tools, but JSON is more lineal, simple and legible.

2.6 Related Work

This section explains some technologies that inspired this work or have similar procedures to this project.

The table 2.8 shows the comparison of several researches about this topic.

2.6.1 A crowd of little man computers: visual computer simulator teaching tools

The usage of the LMC paradigm in order to create a simulation tool for teaching computer architecture is described in [39].

Several simulation tools that are based in the LMC paradigm are described. One of this tools is the Postroom Computer which is a tool that can simulate its own assembly language.

Another tool is the web based LMC mentioned in [39]. It allows students to visualize simultaneous events happening during the execution of their LMC assembly language programs. It is done through a Java applet.

Paper/Tool	Developing Language	Simulation	Description
<i>A crowd of little man computers 2.6.1</i>	Not mentioned	The Postroom Computer simulates its own assembly language, and the web-based Little Man Computer (LMC) simulate basic commands	The LMC paradigm is intended for explaining memory usage, and only the Postroom Computer is capable of simulating a simple high level abstraction language.
<i>DCMSim 2.6.2</i>	Not mentioned	DCMSim is a development of Cache Memory Systems simulator	The system can be configured and then DCMSim can read memory trace files. However, it cannot simulate high level abstraction languages.
<i>DRAMSim2 2.6.3</i>	C++	It uses a DDR2/3 memory system for trace-based simulations	It can be combined with a cycle-accurate simulator in order to perform full system simulations. But it can only read memory commands.
<i>Teaching computer... 2.6.4</i>	C++	The three tools described in this paper simulate memory cache usage	The objective of these tools is improve the students' knowledge about pipelines.
<i>Design and Evaluation of a Cache Memory Simulation Program 2.6.5</i>	Not mentioned	This tool simulates the cache memory	It is web-based and allows the user to configure the memory parameters before doing the simulation. However, it does not allow a program simulation.

Table 2.8: Related Work Comparison

The source also presents a second version of the web based LMC. This version displays the calculator, location counter, input/output boxes and memory in a simple and intuitive interface.

Although this paper describes some interesting simulators, they are not intended for simulating a programming language such as C. These tools are intended for showing how the memory management works with its own simple assembly language.

2.6.2 DCMSim: Didactic cache memory simulator

A didactic cache memory simulator is presented in [40]. This tool simulates real cache memory system. It builds the constructive blocks such as the main memory, the cache comparator and the cache memory blocks and structure.

Users can change cache memory configurations and techniques and also verify the results and effectiveness of these changes.

This tool can read input memory trace files that will be executed and simulated.

DCMSim is a powerful memory simulator, and, even though it can read input files, these files only contain instructions related to memory moves and changes, not taking into account CPU, Arithmetic Logic Unit (ALU), BUS, etc. So, it cannot simulate a program that involves complex statements.

2.6.3 DRAMSim2: A cycle accurate memory system simulator

DRAMSim2 [41] is a cycle accurate memory system simulator. It is implemented in C++ as an object oriented model of a DDR2/3 memory system. This system includes a detailed, cycle accurate model of a memory controller that issues commands to a set of DRAM devices attached to a standard memory bus.

This tool can be used in two modes: standalone binary or shared library. In the standalone one, DRAMSim2 reads commands from a memory trace and simulates them. In the shared library mode, DRAMSim2 creates a MemorySystem object and add requests to it.

DRAMSim2 is pretty similar to the DCMSim tool analysed before. And as DCM-Sim, it cannot simulate the behaviour of program that involves complex statements. DRAMSim2 simulates the cycle of a memory controller from instructions related to memory moves and changes.

2.6.4 Teaching computer architecture/organisation using simulators

Three different pipelined processor simulators are described in [42].

The first one is the WinDLX simulator. It is based on Hennessy/Pattersons DLX architecture. It gives very little processor-internal information as it is modelled at the architecture level.

The second one is MIPSim. It is modelled at the computer organization level, MIPSim displays content and dynamic behaviour of register file, pipeline registers

and multiplexers. It is based on Hennessy/Pattersons MIPS processor.

The third simulator tool is M10kSim. It is based on the MIPS R10000 architecture. It simulates memory instructions through a pipeline. It is useful for explaining register renaming, branch resume buffer or branch history table concepts.

These simulators are pretty useful for showing how the memory operations are performed. However, they cannot simulate programs with complex statements, but just memory instructions related to memory moves.

2.6.5 Design and Evaluation of a Cache Memory Simulation Program

A memory Web-based simulator is explained in [43]. It is used for generating a complete memory cache system that can execute commands (read/write) in a CPU.

This tool can be entirely configured. The user can select the organization type of the cache memory, the block size of the cache memory, the replacement policy (Random, Least Recently Used (LRU) or First In First Out (FIFO)), the write policy in case of hit and the write policy in case of miss.

Even though this tool is capable of displaying a CPU with its ALU, it does not allow a step by step program simulation, apart from reading and writing memory operations.

2.6.6 Conclusion

Although there are several memory simulators and some of them are Web-based and are used for educational purposes, only two of the presented tools can simulate a low level abstraction language similar to assembler. These tools are the Postroom Computer and the LMC web-based simulators.

Nevertheless, these simulators cannot reproduce the processing steps done in a computer while a C, Java, Python or any other medium or high level abstraction language code is being executed. Because of this, the necessity of having a C programming language simulator gave the *C-mulator* tool as a result.

Chapter 3

Project Description

3.1 Introduction

This chapter presents the requirements of this project. It also analyses the functionality to be offered by the *C-mulator* tool.

As said before, the purpose of this project is to develop an educational web application that simulates the execution of programs written in C language over a specific architecture in order to help students to understand better concepts related to the Systems Architecture subject.

The simulation tool will not implement the execution of any program written in C language. So, a set of the C language standard was chosen as will be explained in Section 3.2.1. It is important to notice that the selection of the language subset does not affect the application requirements (see Section 3.2.2), as it should be implemented in a modular and extensible fashion to allow further development in order to support a broader subset of the language and the simulation of other functionalities, such as multiple file programs or the simulation of external libraries.

3.2 Requirements

This section will present both the language subset to be implemented and the application requirements.

3.2.1 C Language Subset

The language subset defined for *C-mulator* is based in the Systems Architecture scheme, and it includes the following features [44]:

- **Lexical Elements:** tokens that make up C source code after preprocessing. *C-mulator* implements all the lexical elements present in the C language. These elements are: identifiers, keywords, constants (comments, integers, floats and strings), operators, separators and white spaces.
- **Data types:** there are different data types in C language. *C-mulator* implements `char`, `int`, `short`, `long`, `float`, `double`, enumeration, union, structure, array and pointer.
- **Expressions and Operations:** an expression consists of at least one operand and zero or more operators. An operator specifies an operation to be performed on its operand(s). *C-mulator* implements expressions, assignment operators, incrementing and decrementing, arithmetic operators, comparison operators, pointer operators, array subscripts, function calls as expressions, operator precedence and order of evaluation.
- **Statements:** the statements are written to cause actions and to control flow within the programs. *C-mulator* implements expression statements, if statement, for statement, blocks, null statement and return statement.
- **Functions:** functions are written in order to separate parts of the program in distinct subprocedures. *C-mulator* allows function declarations, calling functions, function parameters, main function and nested functions.
- **Scope:** it refers to what parts of the program can 'see' a declared object. The scope is implemented in *C-mulator*.

The features that are not implemented in this C subset are: complex number types, incomplete types, type qualifiers, storage class specifiers, renaming types, complex conjugation, bit shifting, bitwise logical operators, sizeof operator, type casts, comma operator, member access expressions, statement and declaration in expressions, labels, switch statement, while statement, do statement, goto statement, break statement, continue statement, typedef statement, function definition, variable length parameter lists, calling functions through function pointers, recursive functions, static functions and program structure.

3.2.2 Application Requirements

In this section the functions that this application has to fulfil will be explained more specifically.

1. Access System

- (a) (all) To log user activity by storing log files in the system, for research purposes.
- (b) (all) To access with username and password using the Lightweight Directory Access Protocol (LDAP) server of the Telematics Department.
- (c) (teacher/admin) To restrict the access only to a set of users.
- (d) (all) To make possible the log out of the user at any moment recording her/his activity until that moment.
- (e) To not allow directory traversal attacks.

2. File System

- (a) (teacher/admin) To visualize every C file that is in the application database.
- (b) (all) To go to the selection page whenever the user wants to.
- (c) (all) To select which C code the user wants to simulate.
- (d) (admin) To manage files in the database.
- (e) (all) To select different files classified by themes.
- (f) (admin) To hide/unhide chapters or files to students.

3. Simulation System (all)

- (a) To support the simulation of a code written in a given a subset of the C language.
- (b) To automatically select the file for simulation after selecting it from a set.
- (c) To show the simulated Random Access Memory (RAM) in each of the program stages.
- (d) To display the output of the selected C code in a separated section from the code itself.
- (e) To show which line of code the user is simulating by highlighting it.
- (f) To show the heap and stack pointers.

(g) To stop the simulation and to select another file at any moment.

4. Contact information

- (a) To be able for the teaching staff to contact the administrator for releasing lessons as well as granting students permissions.
- (b) To be able for the user to contact either the professor or the system administrator in order to report an error.

3.3 Design

In order to do so it is necessary to implement a database management system along with its database, a server that runs the simulation program and attends the requests and the client web page that is going to provide access to the students. This relationship is shown in Figure 3.1



Figure 3.1: *Application Scheme*

Although the final project is a web page with a three tiers architecture and a MVC pattern, the design is done from bottom to top. That is, the simulation tool is conceived as a separated part that can be run in a machine without any web application.

3.4 Use cases

This section shows the application behaviour depending on the different possible scenarios.

To begin with, different actors will play distinct roles in this application:

- The administrator will be able to test the application functionality as well as manage the different C files that the web has to simulate. This person is also responsible for granting the teaching staff permissions. Moreover, the administrator has to handle possible errors and add or modify some functionalities.

- The teacher will be able to see all of the files in the applications classified by lessons. Additionally, he or she can contact the administrator in order to incrementally hide/unhide the lessons to his students.
- The student will have a reduced view of the application, as they have access only to a subset of the files.

As a consequence, the application will only need a view, but different queries to the database depending on the user.

This section can be summarized in the following diagram¹ that is showed in Figure 3.2.

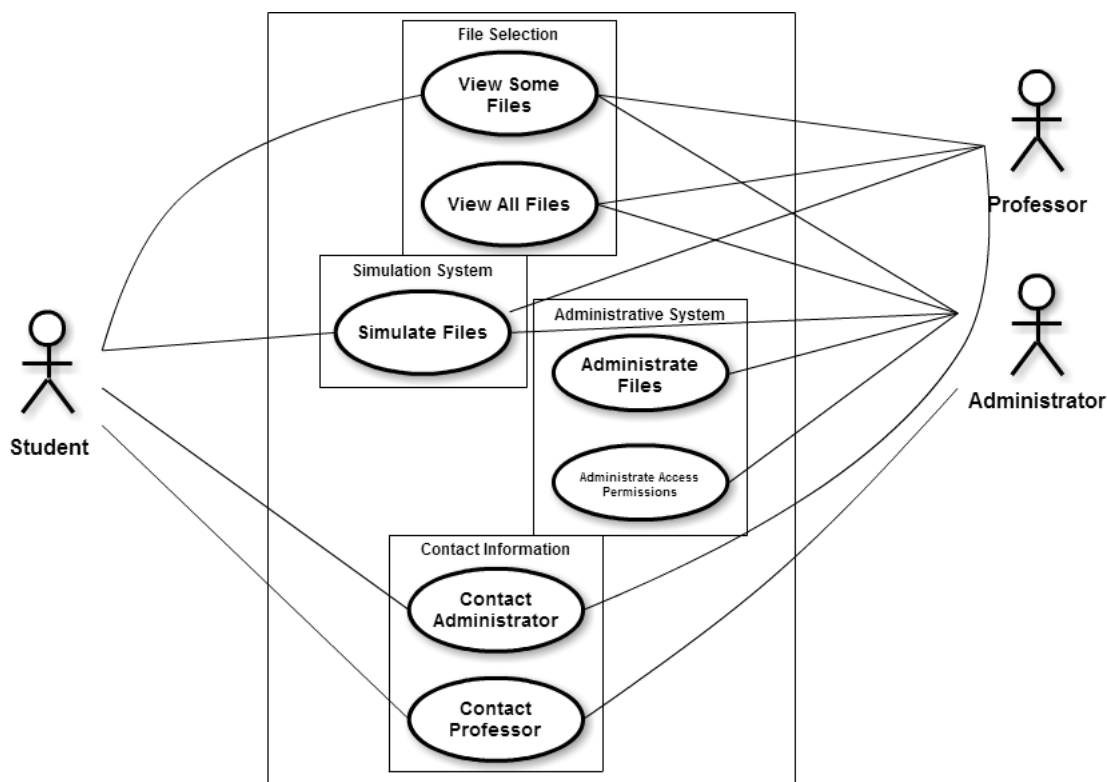


Figure 3.2: *Use Cases*

¹In this diagram, if the actor can view all of the files, it is also able to view some of them.

3.4.1 Simulation

The simulation tool will be implemented as a black-box in which a file enters in the program and after some processes, that will be explained in chapter 4, a machine state with all computer components initialized for simulating the file.

On the other hand, if it is a machine state what enters in the program, it will generate the next state, updating each component and returning a fully new file that represents that machine state.

In order to clarify this, Figure 3.3 is attached.

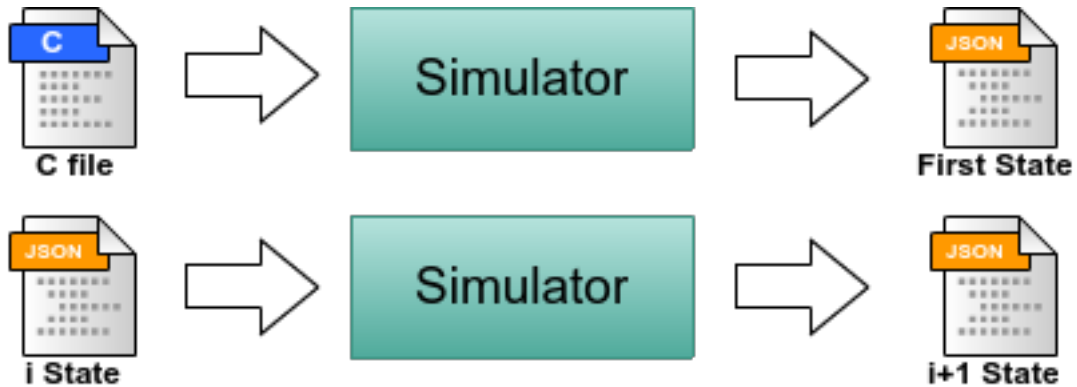


Figure 3.3: *Simulator tool behaviour*

3.4.2 Database

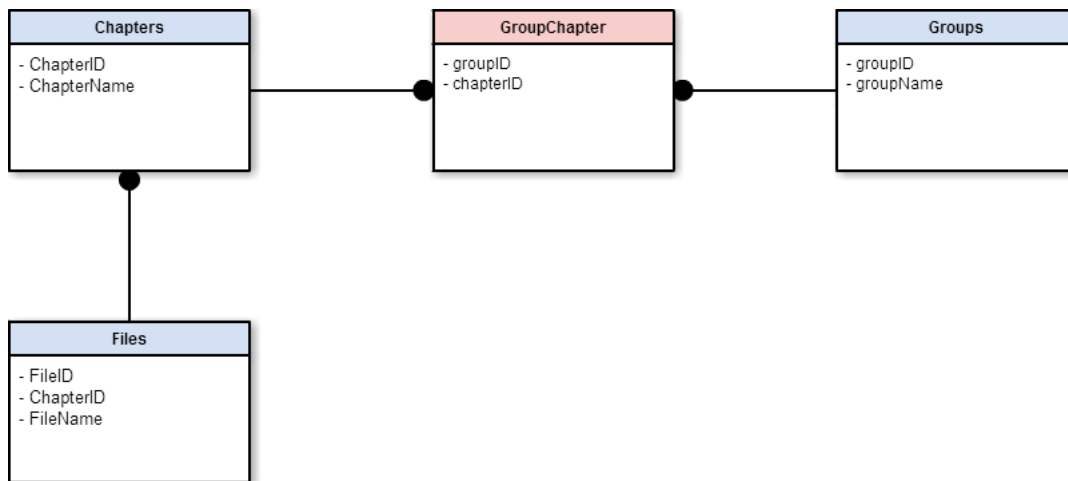
Once the simulation tool is designed, the second step is designing a database that will organize the files introduced in the simulation program.

As shown in Figure 3.4, the database structure will include a table with the chapter names, another table with files associated with the chapters, a table of student groups, along with the administrative group, and, finally, a table associating the groups and the chapters that the students in the group are able to see.

The access to the database will be divided in *reading* information and the *insertion and modification* of that information. The first one can be fully done by the administrator and the professor, but the students will have a limited access to the chapters and files. This is because the professor can release or not the files that are in some chapters.

The *insertion and modification* of data can only be done by the administrator.

The user access registry will be modelled by creating folders in the server with

Figure 3.4: *Database Structure*

the user id and writing, inside those folders, logs file for research purposes and JSON files for communication between server and client. Therefore, tables with users' information are not needed as the *.htaccess* protocol will be used, so only the allowed users can access to the directory in which the web files are stored.

3.4.3 Server

This module is in charge of processing client requests and respond them with the data required.

This block implementation is divided in three different parts, the two explained before and the servlet, which perform distinct functions as can be seen in Figure 3.5.

3.4.3.1 Servlet

The servlet will be in charge of communicating with the client. It will pass the file selected by the user to the Serialization class, explained before. Then, when the Serialization class had written the JSON file, the servlet will deliver it to the client, in order to let the JavaScript update the RAM tables and the code and output windows.

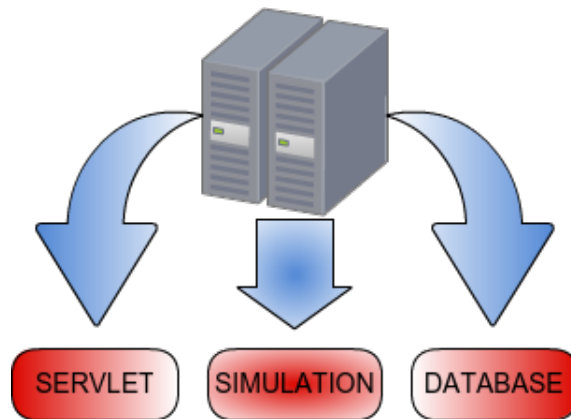


Figure 3.5: *Server Sections*

3.4.4 Client Design

Once the simulation, database and servlet parts are designed the next step is preparing the client architecture.

The client will consist of a fully designed web page with access service, log-out service and the choosing files and simulation parts. Although the application is intuitive, it is necessary to introduce a description with the required steps to simulate a file along with the professor and administrator's emails.

Figure 3.6 shows the main design of all the web page views that are in the application as well as their relationship with each others.

This is the mock-up of the web page. It is a graphical representation of how it will look like once it is fully implemented, this is explained in chapter 4. The mock-up is the best way to show how the finished web site will look like [45].

3.4.5 Client-Server Communication Design

In order to communicate the client browser with the server, JSON format will be used in AJAX requests. Then, the client can send parameters in JSON format to the servlet and this can send the responses in the same format for the JavaScript to analyse these responses.

3.4.6 Summarize

The client and server behaviour, along with the communication between them, explained before can be summarized in the diagram showed in Figure 3.7.

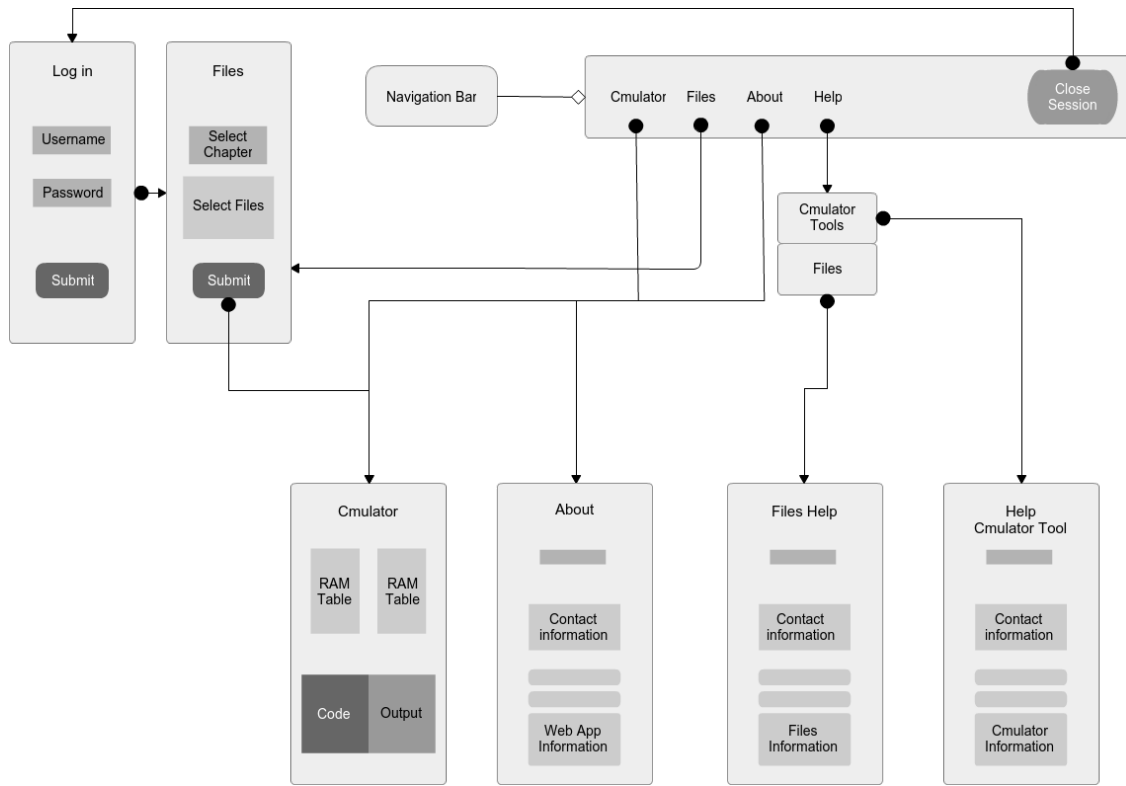


Figure 3.6: *Web Page Views*

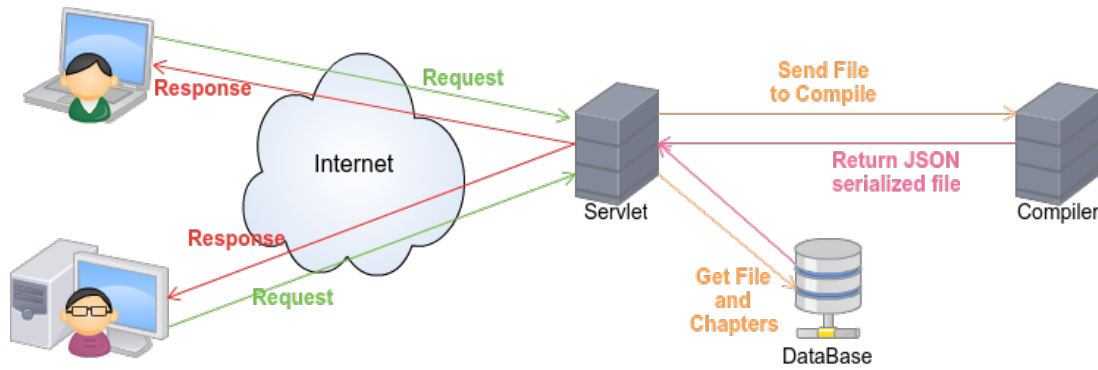


Figure 3.7: *Design Scheme*

Figure 3.7 explains how the requests are delivered through the internet to the servlet. Then, this servlet, after performing some operations, gets the file and chapter names from the database in order to show them in the web page.

Once the user selects a file, the server sends the file to the compiler, this compiles the file and returns a machine state in JSON format to the servlet. Finally, the servlet sends the JSON machine state as a response to the client.

Chapter 4

Implementation

4.1 Introduction

The first section of this chapter will explain the selected technologies that are going to be implemented in the application. The simulation tool will be fully explained and finally, the other parts of the web application will be added.

4.2 Selected Technologies

In this section, the results and conclusions of chapter 2 will be useful for deciding the right technologies that are going to be used in this project.

First of all, it is important to consider that web applications are complex compounds of different technologies that are related between them. For this reason, it is critical to make a previous study of the available technologies and compare them in a suitable way.

The technologies decision will be analysed starting from the software of the project server, the database management system and the middleware. All of them are related with the server side.

Secondly, the client language will be decided and finally, the client-server communication part will be exposed.

4.2.1 Server

The *C-mulator* application has been developed in a **Raspberry Pi 2** with a **Raspbian** operative system. This decision has been done taking into account the porta-

bility, due to its reduced size, of the **Raspberry**. Regarding the model, the second one has been decided due to the greater computational capability, comparing it with other **Raspberry Pi** models, and its larger RAM capacity (1 Gb). The image 4.1 is a comparative of these mini-computers. The configuration of the **Raspberry Pi** is explained in Appendix A.


				
Raspberry Pi:	Model A+	Model B	Model B+	2, Model B
Price:	\$19.99	\$39.99	\$29.99	\$39.99
Availability:	Add to Cart	Add to Cart	Add to Cart	Add to Cart
Quick summary:	Cheapest, smallest single board computer.	The original Raspberry Pi.	More USB and GPIO than the B. Ideal choice for schools	Newest, most advanced Raspberry Pi.
Chip:	Broadcom BCM2835			Broadcom BCM2836
Processor:	ARMv6 single core			ARMv7 quad core
Processor Speed:	700 MHz			900 MHz
Voltage and Power Draw:	600mA @ 5V			650mA @ 5V
GPU:	Dual Core VideoCore IV Multimedia Co-Processor			
Size:	65x56mm	85x56mm		
Memory:	256 MB SDRAM @ 400 MHz	512 MB SDRAM @ 400 MHz		1 GB SDRAM @ 400 MHz
Storage:	Micro SD Card (not included)	SD Card (not included)	Micro SD Card (included)	Micro SD Card (not included)
GPIO:	40	26	40	
USB 2.0:	1	2	4	
Ethernet:	None	10/100mb Ethernet RJ45 Jack		
Audio:	Multi-Channel HD Audio over HDMI, Analog Stereo from 3.5mm Headphone Jack			
Make coverage:	First look		First look	First look In-depth Windows support

Figure 4.1: *Raspberry Pi comparative*

On the other hand, regarding the Database Management System, discussed previously in subsection 2.3.1, two systems were chosen among five of them, taking into account the cost of the other three RDBMSs. Then the decision of the database was done between MySQL and Microsoft Access.

MySQL was chosen instead of Microsoft Access because of its extensive background and bibliography. Moreover, the community behind MySQL is the largest one and this had a considerable impact in the decision. Even though, MySQL characteristics such as its multi-user and multi-platform capabilities and its computational power related to large quantities of data, were another noteworthy point for taking the decision.

Finally, the middleware was limited between Apache or Tomcat. As the project was thought to be implemented with servlets, due to their simplicity, ease of use and extended response capabilities, Tomcat was the desired option.

4.2.2 Client

C-mulator is a dynamic web page, as the required use is to show the user how a simple C code works in a computer. Due to this consideration, only HTML could not be the desired choice. SWF was dismissed due to its complicity and because of the lack of multimedia content in the application.

Then the choice was reduced to AJAX or JavaScript. As the AJAX solution contains itself the JavaScript language, it was the chosen client technology for this project.

4.2.3 Client-Server Communication

As any of the data interchange formats that have been discussed in subsection 2.5 will fit the into the project structure. JSON has been chosen among the others because it is the most popular one and it has been chosen as the standard in data formatting for the web [38].

4.2.4 Conclusion

Finally, the selected technologies are structured and described together with the reason for its choice in Table 4.1.

4.3 Simulation part

This part can be explained as a separated tool in which a web application has been implemented on top of it. Because of that, this tool can be used with or without a web application.

Technology	Reason
<i>MySQL</i>	One of the most used RDBMSs. With Oracle is one of the most powerful systems but it has the advantage that it has no cost unlike Oracle.
<i>Tomcat</i>	As the project has been implemented in Java, servlets came as a suitable option for server implementation. That reason explains why Tomcat was chosen above Apache.
<i>AJAX</i>	AJAX is a compound of several technologies including JavaScript, which has a large community of users. It made a suitable solution for client implementation.
<i>JSON</i>	JSON has been chosen as the data formatting standard [38]. It has a large community of users as well and provides a clear and structured way of viewing data.

Table 4.1: Selected Technologies

4.3.1 Introduction

The simulation part will be done by a compiler made in Java. This compiler will check the code in order to find any syntax errors. If there were any, it will return an error message and will not continue with the execution.

After checking the code, the program will analyse it line by line doing the required tasks with the RAM, Input/Output (IO), BUS, ALU and CPU.

This program will be controlled by a Serialization class, that will provide the code and the number of lines that the compiler needs to simulate. Once the compiler executes the code and the execution pointer return to Serialization. This class will serialize all the other classes and will save the serialization in the user directory as a JSON file.

4.3.2 Program behaviour

In the program that does the simulation part, a file and an username are the input parameters. Once this program has a file or a machine status, it returns the first machine status of the program or the next one, respectively.

In order to get a clearer view of what is going to be explained, Figure 4.2 represents a class diagram of the six classes of this program. CPU class will be analysed

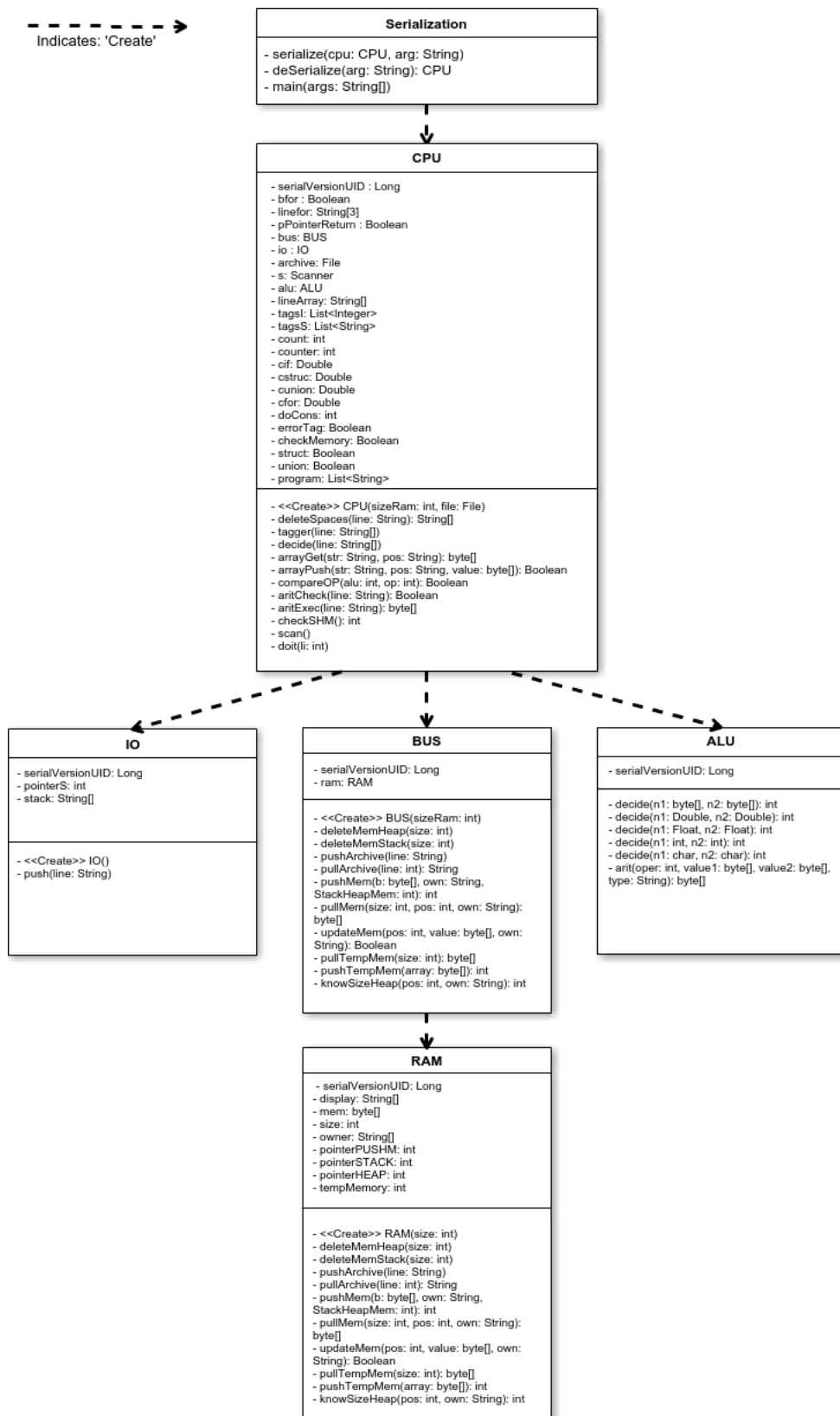


Figure 4.2: Class Diagram

at the end of this section due to its complexity and importance.

4.3.2.1 Serialization

The one that has the main method is called *Serialization*. Figure 4.3 shows the behaviour of this class.

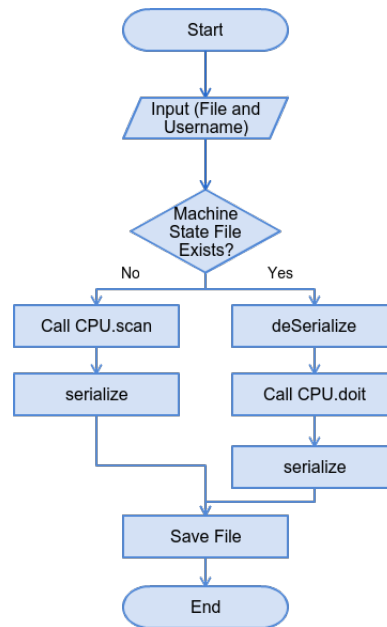


Figure 4.3: *Serialization Flowchart*

This class checks if there exists a previous serialized program inside the user directory previously made in file `check_login.php`. If it does not exist any serialized file, it means that is the 'loading' process, not the execution one. Then the file is created and class *CPU* is created, initialized and the *scan* method is called.

When the *CPU* class finishes, *Serialization* checks if there were any errors in the *scan* process. If there were, this function calls the *IO* class and introduces a "Check syntax" error message. If there were none, it calls its own method *serialize*, with the object 'cpu', instance of the class with the same name, and the username as parameters.

If there is not a serialized version of the program, the second argument of the server is an integer number, not a file name. Then the cpu object is 'de-serialized' with the method *deSerialize*, having as an argument the username. Once the cpu object is created, method *doit* from class *CPU* is invoked with the previous inte-

ger number passed as parameter. Finally, the existing serialized file is deleted and *serialize* method is invoked.

Finally, the three methods that are part of ***Serialization*** are going to be explained:

- *void serialize(CPU cpu, String arg)*: this method receives the cpu object created in the *main* method, and a string that is the student's username. This two arguments are used for creating a file inside the user's directory in the server that will store the serialization of the ***CPU*** class in JSON format. Moreover, this method will return the JSON file to the servlet.

- *CPU deSerialize(String arg)*: this method receives the student's username as an argument and it access to the JSON file stored in the student's directory in order to deserialize it and continue in the same state as the object cpu was when it was serialized.

- *static void main(String[] args)*: the main method receives as arguments the student's username, which is the first input argument, and a second string that can be a number, if the JSON file is already created in order to continue with the C code simulation, or it can be a string that indicates the file name in order to start the simulation of that C file.

If it is the first time that the main is simulating a file, second input argument is a file name, this method will create a ***CPU*** object called cpu and it will also invoke the method *scan()* of this cpu object.

If it is not the first time that the main is simulating the file, second input argument is a number, main will call the *deSerialize()* method and it will create a CPU object from the JSON file. Then, it will call the *doit()* method from cpu.

4.3.2.2 ALU

This class carries out the arithmetic operations that have to be done in any code. For example comparing numbers, characters or adding or subtracting them.

This class has several methods that are going to be explained one by one:

- *byte [] arit(int oper, byte[] value1, byte[] value2, String type)*: this method receives as arguments, the operation type as an integer, and the values of each byte array that are going to be operated. It returns the result of the operation in an array of bytes.

- If the *oper* argument is 1 or 3, the *arit* method will perform a **sum**.
- If the *oper* argument is 2 or 4, the *arit* method will perform a **subtraction**.
- If the *oper* argument is 5, the *arit* method will perform a **multiplication**.
- If the *oper* argument is 6, the *arit* method will perform a **division**.
- *int decide (**type** n1, **type** n2)*: there are five different methods with the same purpose, deciding if the values passed as parameters are equal, n1 greater than n2 or n1 smaller than n2. The types of inputs can be:
 - **char**: due to Java way of compiling, chars can be compared as normal numbers.
 - **int**.
 - **float**.
 - **double**.
 - **byte[]**: this type is compared by checking if the arrays are equal, if they are not, then the bytes are compared one by one.

4.3.2.3 IO

This class is used for displaying values or messages in the output window of the web page. It has a constructor and a method:

- *IO()*: it initializes the attributes of the class, setting the pointer to 0 and the array length for the system output to 1024 lines.
- *void push (String line)*: this method checks the input line, verifying that it is not null, and then add it to its *stack* array.

4.3.2.4 BUS

The **BUS** class is intended for not directly access the RAM. Due to this, the **BUS** class is used as a 'transport' between the operational part and the memory.

This class has several methods and a constructor:

- *BUS(int sizeRAM)*: as it is shown in Figure 4.2, the BUS creates the RAM object. In this constructor the BUS creates a RAM object with the size indicated as parameter.

- *other methods*: these methods call the namesake one in the RAM, so they will be analysed in RAM.

4.3.2.5 RAM

Here is where the memory management happens. The RAM is structured with three arrays, one for storing the program lines, another for storing the memory bytes and the last one for storing the owner of those bytes. This class also has pointers in order to maintain the data mentioned before.

The constructor and several methods that are inside this class are:

- *RAM(int size)*: the memory has a size of 2^{size} as well as the owner array, as they have to be of the same length. However, the display array which stores the code has a size of $size^2$. And the pointers starting positions are:
 - *tempMemory*: $2^{size} - 100$, then the temporary memory is 100 bytes long.
 - *pointerSTACK*: $2^{size} - 2000$.
 - *pointerHEAP*: $2^{size} - 101$.
- *void pushArchive(String line)*: this method introduces each line of the C file inside the array of the RAM that is prepared for this.
- *String pullArchive(int line)*: this one, on the contrary, pulls out the line that has the position passed as parameter from the array.
- *void deleteMemStack(int size)*: it clears from the stack the number of bytes indicated by size.
- *void deleteMemHeap(int size)*: it clears from the heap the number of bytes indicated by size.
- *int knowSizeHeap(int pos, String own)*: this method returns the number of bytes in the heap that have the same owner as the one passed by parameter from the position indicated also by parameter.
- *int pushMem(byte b[], String own, int StackHeapMem)*: this one introduces the byte array *b* passed as parameter with the owner indicated in the stack, heap or normal memory, depending on the third parameter. This method returns the first position in which the array is stored.

- *byte[] pullMem(int size, int pos, String own)*: this method returns the array of bytes stored in the position *pos* with owner *own* and size *size*. If the owner does not corresponds with *own* in any of the positions requested, it throws a **Segmentation Fault** error.
- *boolean updateMem (int pos, byte value[], String own)*: this method updates the memory stored in position *pos* overwriting it with the array of bytes named *value* and checks if the owner is the same as *own* in each position. If it is not the same owner, it throws a **Segmentation Fault** error.
- *int pushTempMem(byte b[])*: it stores the array of bytes *b* in the temporary memory, returning the first position in which the array is stored.
- *byte[] pullTempMem(int size)*: this method returns the last *size* bytes introduced in the temporary memory.

4.3.2.6 CPU

This is the main and most important class. This class has to analyse and check every line of code in order to know if it is a line in which a variable is declared or if it is a for line.

In order to get a clearer view of this class, a flow diagram of it is shown in Figure 4.4.

These two methods and the constructor of *CPU* class can be explained like this:

- *CPU(int sizeRAM, File file)*: in this constructor IO, ALU and BUS objects are created. Moreover, three *arraylists* that are going to be useful for tagging and memory positions are also created. An scanner of the file is initialized too.
- *void scan()*: this method is executed when the file is loaded in **Serialization** class, that means, when it is first selected by the user in the web page. It traverse the entire C code, storing each line in the memory and calling the methods *deleteSpaces* and *tagger* that are going to be analysed after.
- *void doit(int li)*: as it is shown in the flowchart of Figure 4.4, this method pulls code lines from the RAM, through the BUS, and calls the method *decide*. If the string returned by *decide* is null, *doit* will end its execution, if the string is an error, *doit* will call the IO object in order to push the string "Error in doit". Finally, if the returned string starts with a minus, it will mean that the program has reached and end of an if clause and will jump the final line. If none of this

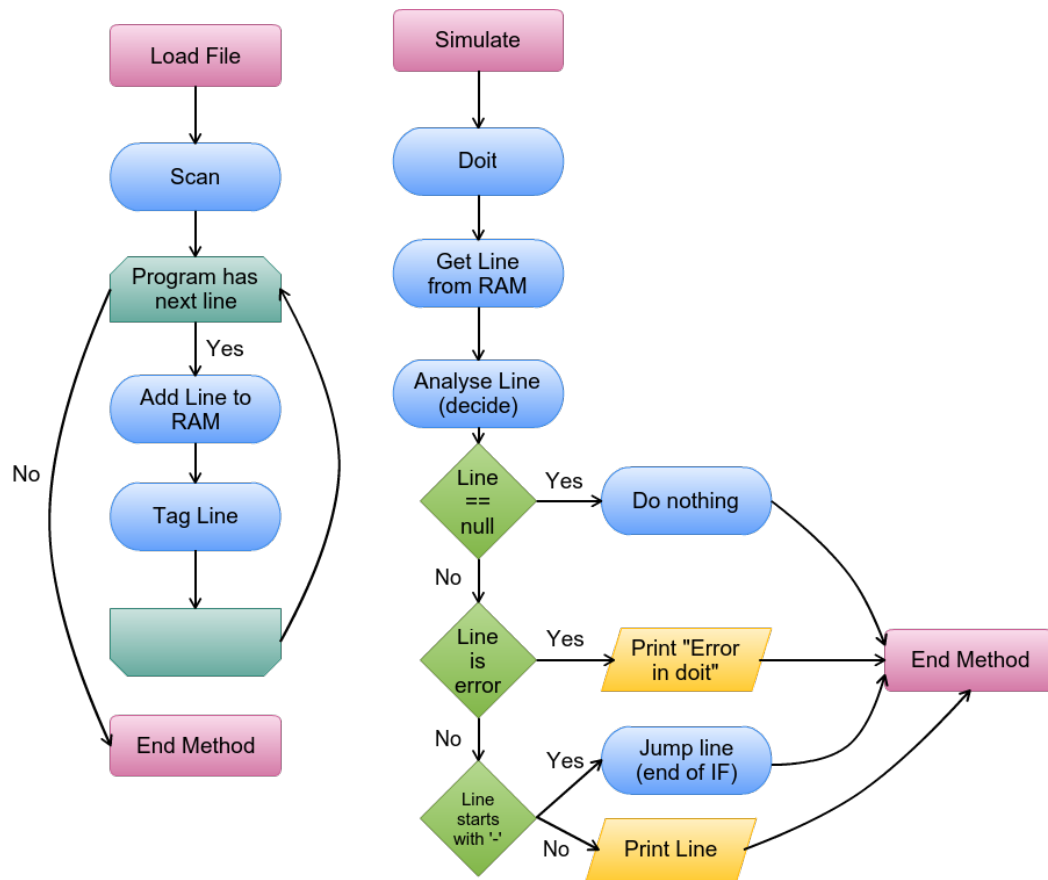


Figure 4.4: CPU flow diagram

clauses is accomplished, then the method will call the IO object and will print the line.

Now that the core of this class has been evaluated, the next step is going deeper inside the class and study the secondary methods.

- *String[] deleteSpaces(String line)*: this method is called by *scan* and it splits the lines, in case there are more than one ';', deletes the tabs and spaces for an easier analysis and takes into account the comments that could appear in the lines. So, it returns an array of Strings that stores each line of code that has been splitted from the original one. Figure 4.5 summarize this.

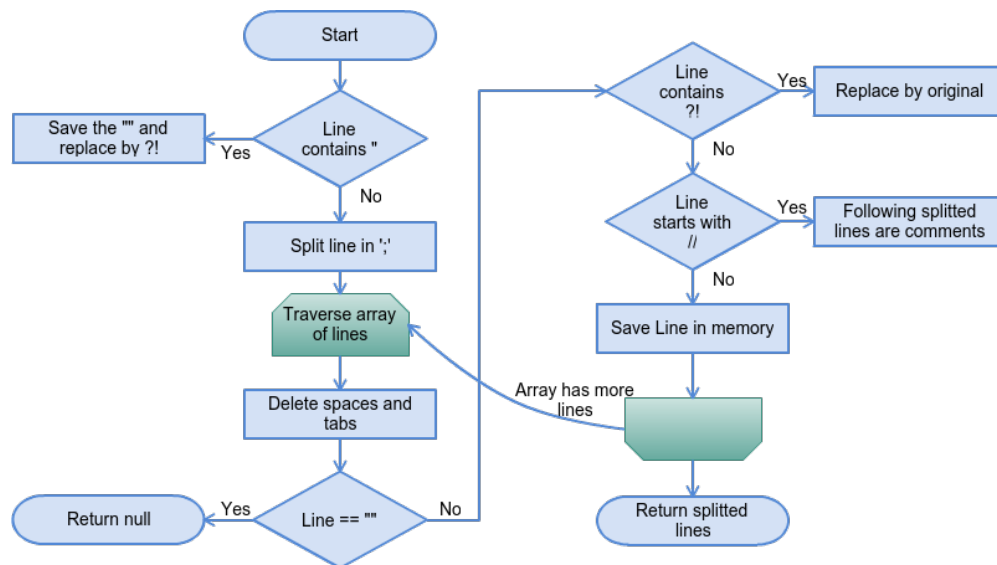


Figure 4.5: *deleteSpaces* flowchart

- *void tagger(String[] line)*: it is one of the most important methods of this class. Its function is to determine which function does each line of C code. That is, it checks whether the line is a 'printf', a variable declaration or so on. Depending on the result, this method will behave differently. In order to explain it in a clearer way, Figure 4.6 is attached.
- *String decide (String line)*: this method, along with *tagger*, is one of the most important of the entire tool. Unlike *tagger*, this method does not determine

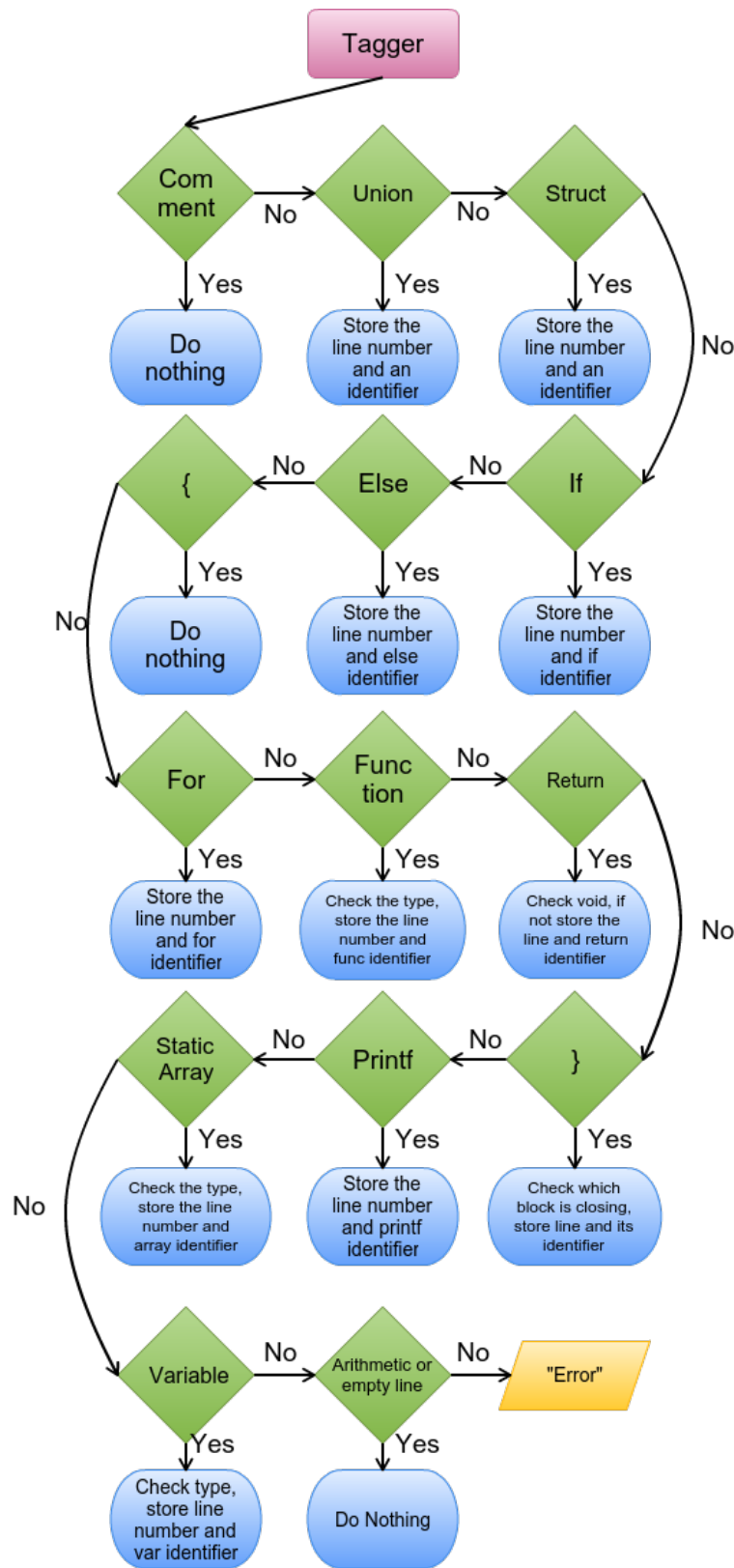


Figure 4.6: *tagger flow diagram*

the function of each line, it executes each line taking into consideration the *arraylists* that were filled in tagger. Then depending on the function of each line, this method will behave differently.

- **Jumping to main function:** if the program is starting and it tries to enter in a function that it is not the main before entering the main, *decide* will jump to the line where main is.
- **Calloc, realloc or free:** *decide* will reserve memory taking into account the input parameters of these functions. If it is a free, it will delete the heap memory that is related with the parameter pointer of this function.
- **Checking if the program is inside a for:** if the simulation reaches an end of a for loop, *decide* will check it and it will go to the next iteration if the for condition is satisfied.
- **Return of a function:** if the simulation reaches a return line, *decide* will delete the memory associated with that function (local variables) and return to the line where the function was called. Moreover, the program will store the returned value and it will check if it is the return of the main, if that is the case, the simulation will finish its process. Nevertheless
- **Calling a function:** when the code calls a function, this method stores the input parameters in the stack, creating a copy of them and then it jumps to the function called.
- **If,for or while:** if the simulation reaches a line with an if or a for, *decide* will first check if the condition is true or false. If it is false, *decide* will jump to the else, if there is any, or will not enter into the loop. If it is true and an if, it will just jump into the if block. Finally, if it is true and a for or a while, this method will store the for variable in the memory (in for case), if it is not stored before, and it will jump inside the loop.
- **Else:** if the method finds an else line it will mean that the if clause was true and the block was simulated, then it will be jumped. When the if is not true and there exists an else, the jumped is done in the previous explained situation and the simulations enters directly into the else clause.
- **Printf:** if the line is a printf, *decide* will just return the String between the ””.
- **Static Array:** this method will store the values in case that the line initializes the array. If the array is not initialized and only declared, *decide* will not store memory, but will store the name of the array in case

it is initialized afterwards. If the case is that the array has been declared before and in this line it is being initialized, *decide* will store the values in memory and will change the array declaration to an initialized one.

- **Variable:** similar to the array case. If the variable is only declared, *decide* will not store memory but will store the variable name. If it is declared and initialized, it will store the value in memory and the variable name. If the variable is being initialized after it has been declared, the definition of it will change to initialized and the value will be stored.
- *byte [] arrayGet (String str, String pos)*: this method is used for retrieving from the memory an array. 'pos' String is parsed to an integer and it will indicate the first memory position of the array. 'str' indicates the owner of the array, that is, the variable name.
- *boolean arrayPush (String str, String pos, byte[] value)*: on the other hand, this method stores an array of bytes, 'value', in the memory with a starting position 'pos' and the owner is 'str'. It will return a true if everything was fine, if not, it will return a false.
- *boolean compareOP (int alu, int op)*: this method compares the result of contrasting two variables in the ALU and the clause that an if or a loop has. If the condition is fulfilled it returns true, if not, it returns false.
- *boolean aritCheck (String line)*: this one indicates if there is an arithmetic operation in that line.
- *byte [] aritExec (String line)*: it executes the arithmetic operation detected in the previous method and returns the byte array with the result.
- *int checkSHM ()*: this method checks if the memory needs to be stored in stack, heap or global memory. Depending on the option, it returns an integer or another.

4.4 Data Model

As explained in section 3.3 the database structure is a really important part in the server implementation. This is due to the reason that the web server has to make queries to the database regularly. Because of this, the database architecture is a fundamental part of the system.

Knowing the requirements and the use cases that the application would have, the database was designed as explained in Figure 3.4 .

In this project there are only needed four tables with a relationship between them.

This tables are needed in order to have the C files structured by chapters or lessons and for releasing them incrementally or how the professor needs.

The table **chapters** stores the titles of every lesson of the System Architecture subject. In this table, the ids of every chapter are generated incrementally and automatically by the RDBMS. These ids are the primary keys for the **chapters** table.

The table **files** contains the id of every file, this id is a primary key as in the **chapters** table; the `chapter_id` which relates every file with its related lesson and, finally, the `file_name` which has the name of the C file that is stored in the server.

Finally, the table **Groups** defines the group label and their corresponding id. These ids are related with the chapters ids in table **GroupChapter**.

This tables allow the server to display all the files classified by chapter and also, it is possible to hide some entire chapters and not display them for each students' group.

4.5 Server

This part was implemented at the same time as the client side so some things that are explained here are repeated but from the client view in section 4.6.

As there are only three types of requests to the server and two of them are almost equal and the third is quite similar, in order to simplify the code and not making repetitions, the behaviour of the application will be modelled by only one servlet.

4.5.1 Access System

This system is modelled entirely by the PHP file explained in 4.6.1 that is connected to an external LDAP server. This LDAP server stores the students' Telematics accounts and also the professor's account.

The ip of this server is 'ldaps://ldap.lab.it.uc3m.es'. The necessary code in order to connect to, bind to and search in, is in Table 4.2.

```

$zonas = array (array ('type' => '1', 'server' => 'ldaps://ldap.lab.it.uc3m.es' , 'basedn' => 'dc=lab,dc
=it,dc=uc3m,dc=es'),);

for ($i = 0, $size = count($zonas); $i < $size; ++$i) {
    $server = $zonas[$i]['server'];
    $ldap_base_dn = $zonas[$i]['basedn'];
    $ldapconn = ldap_connect($server)
        or die("Could not connect to LDAPit server.");
    // Set some ldap options for talking to
    ldap_set_option($ldapconn, LDAP_OPT_PROTOCOL_VERSION, 3);
    ldap_set_option($ldapconn, LDAP_OPT_REFERRALS, 0);
    if ($ldapconn) {
        $ccid=$_POST['username'];
        $filter="(&(uid=$ccid))";
        if (!( $result = ldap_search($ldapconn, $ldap_base_dn, $filter))) {
            die("Unable to search ldap server");
        }
        if ( ldap_count_entries($ldapconn,$result) == 1 ) {
            //User found
        }else{
            //user not found
        }
    }else{
        //No connection
    }
}

```

Table 4.2: LDAP code

4.5.2 Servlet

In this servlet the three different AJAX calls are managed. As this calls are pretty similar, the decision was to made an unique servlet in order to simplify the management of the server.

The three possible scenarios are:

- **Load function:** when the user selects a file for the first time, the client sends the user id and the file selected. Then the servlet checks if the directory of this user has an older serialized state in it and delete it. When everything is prepared, it sends the file and the username to the simulation tool and waits until the machine state in form of a JSON file is returned. Once the servlet has the JSON file, it sends it back to the client.
- **Reload function:** is exactly the same as the **load function** from the point of view of the servlet.
- **Play function:** once the user clicks the 'play' button at the client, the client sends only the username. The servlet gets the previous machine state from the file stored in the user directory and pass it to the simulation tool. This returns the following state and the servlet updates the file in the user directory. Then it passes the JSON to the client.

4.6 Client

In this section the client functionalities will be explained.

The client implementation is based in a combination of HTML, Cascading Style Sheets (CSS) and JavaScript files.

There are five different HTML files, four of them have a navigation bar which allow the user to log out or change views whenever he wants. The other HTML file is the index one, in this web the user can log in the application with his telematic user and password.

Secondly, there are six CSS files that are used to model the web pages style. Five of them are related with each HTML file, and the other one is the style of the navigation bar that appears in four of the HTMLs.

Finally, there are several PHP files used for displaying the files, connecting to the LDAP server and, also, there is an .htaccess file that grants permissions to the users. Even though some of the features are managed by the server in PHP, there also exists a JavaScript which handles the entire simulation behaviour and connects the file *cmulator.html*, simulation view, with the server.

4.6.1 Access System

As it is said previously, the user has to be validated by the application in order to enter into the simulation part. Then the authentication part is needed.

For this purpose, the HTML page *index.html* was created. This file header can be seen in Table 4.3.

```
<html lang="en">
<head>
<title>Cmulator</title>
<link href="styleIndex.css" rel="stylesheet">
</head>

[...]

</html>
```

Table 4.3: HTML header

The page shows the application logo as well as a form for introducing the user password and username. These credentials are the same as for the Telematics labs, this is due to the fact that the PHP file to which the credentials are sent makes a request to the LDAP server of the Telematics Department.

Depending on the server response, the PHP file handles several actions:

- **Not registered user or not allowed student** The application will return an error that will be displayed in the user's browser with the following message: "Invalid Username or Password".
- **Allowed user** The user will be redirected to the web page in which he has to select the file that he wants to simulate.

In order to clarify this, the diagram 4.7 is added.

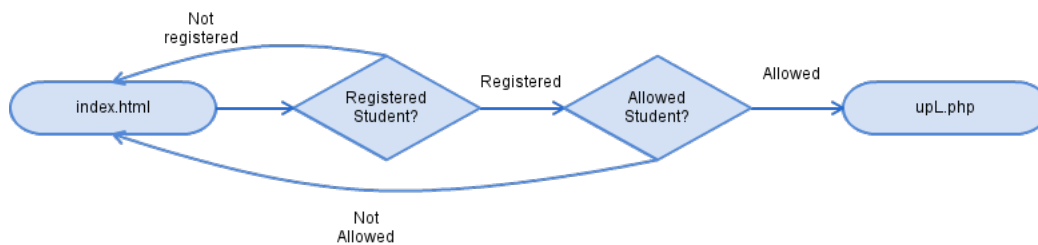


Figure 4.7: *Access System*

To conclude this section, the role of the PHP file (*check_login.php*) that makes this behaviour possible will be described as shown in Figure 4.8.

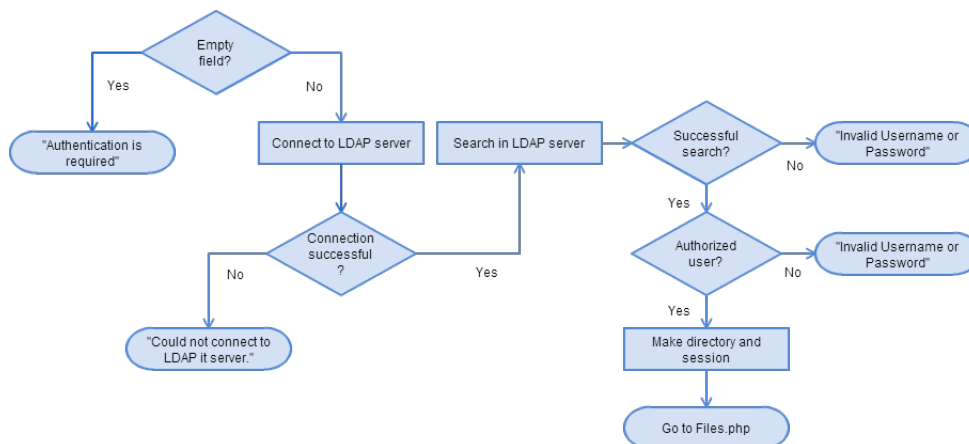


Figure 4.8: *Check login behaviour*

- **Empty field:** the file will check it and return an error if it is the case.

- **Cannot connect to LDAP server:** the file will try to establish a connection with the server, if it is not possible, it will throw an error.
- **Cannot search into LDAP server:** before doing the query, the PHP file checks if bind and search are possible. If not, it will throw an error.
- **Not registered user:** if the user introduced and the password are not in the LDAP server, the file will announce the error.
- **Registered but not allowed user:** if the user does not have privileges to enter the web page, the server will alert him and redirect to the login page.
- **Allowed user:** if the user is registered and has privileges, the application will create a session with his username and a log directory for educational and research purposes.

4.6.2 File selection system

As there is a difference between the files that students can visualize and the ones that the professor or the administrator can access to, there is a PHP file that makes the queries to the database server depending on the type of user that has opened the session.

This file that models the chapters visualization is *Files.php*. It checks if the user is the professor or the administrator, if it is not any of them, it will only show certain chapters.

This decision is done with the simple code shown in Table 4.4.

```

<?php
[... ]

if($_SESSION['group'] == 0)
{
$_SESSION['group']="admin";
}
$query1 = "SELECT Chapters.ChapterID, Chapters.ChapterName FROM Chapters INNER JOIN GroupChapter ON
GroupChapter.chapterID=Chapters.ChapterID WHERE GroupChapter.groupID=(SELECT groupID FROM
Groups WHERE groupName='{$_SESSION['group']}')";
[... ]
?>

```

Table 4.4: Selection Query

Then the chapters returned from the query are displayed as a select input in the same web page. Once the user selects a chapter and click the submit button, the management of the application is delivered to another PHP file, the file *files.php*.

This file has the same structure as *Files.php* but it displays all of the files that are inside the selected chapter with a submit button that allows the post request and stores the selected file in the PHP session.

As a summarize, the Figure 4.9 illustrate the above explained.

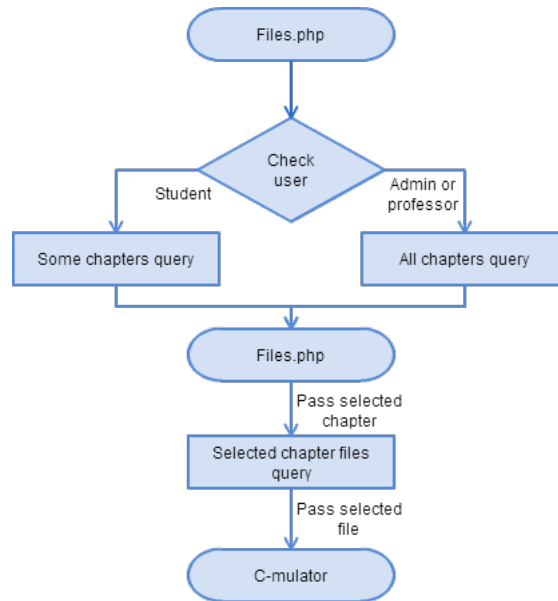


Figure 4.9: *File selection system*

4.6.3 Simulation part

As this is the client section, only the simulation part related with the HTML and JavaScript files will be explained here.

The file that is in charge of the simulation web page is *cmulator.php* which contains two tables, one that stores global variables and the other that contains the heap and stack piles. Moreover, this application view has two rectangular blocks, one for the C code and the other for the output of the program.

The behaviour of these blocks is modelled by the file *app.php*, which is a JavaScript file but with a PHP header in order to be able to get the user and file session variables.

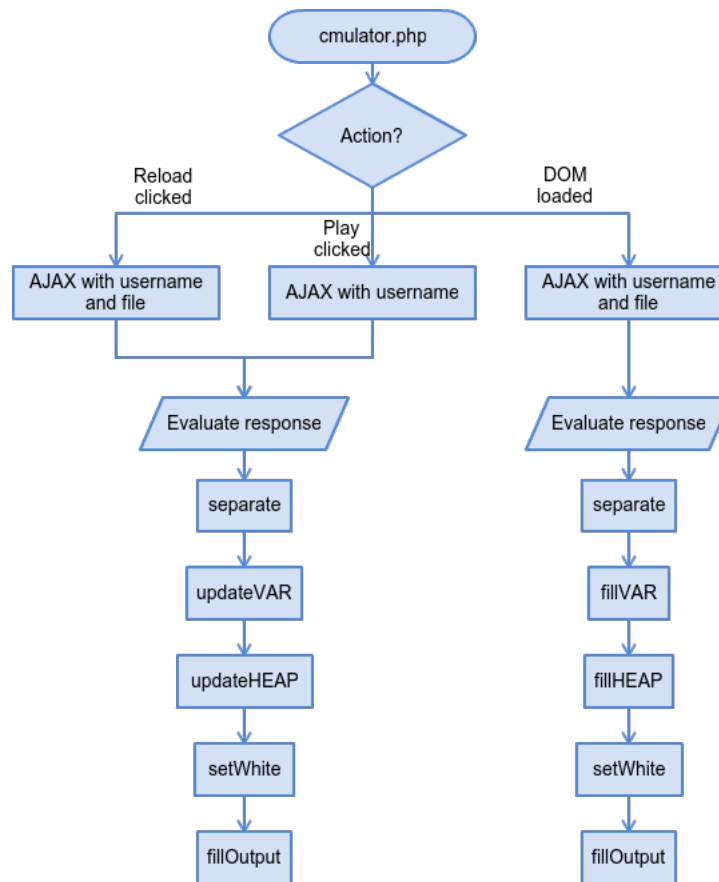
Figure 4.10: *Simulation behaviour*

Figure 4.10 shows the flow diagram of the JavaScript file.

This file has three main functions, one that is executed just after the page is loaded, other that is accomplished when the play button is clicked, and the last one is done when the reload button is hit. These functions behaviour is the following:

- **Load function:** this function sends an AJAX post request to the servlet with a string parameter that contains the username and the file selected. When the response reaches the client in a JSON format, this function calls the secondary functions *separate*, *setWhite*, *fillVAR*, *fillHEAP* and *fillOutput*. These JavaScript functions populate the RAM tables and the output window. The load function also fills the code window that is passed in a JSON format too.
- **Play function:** this function also performs an AJAX request to the servlet, but with only the username as a parameter. If the response, also in JSON format, is successful it calls the secondary functions *separate*, *setWhite*, *updateRAM*, *updateHEAP* and *fillOutput*. This function updates the information that is stored in the RAM tables and in code and output windows.
- **Reload function:** this function performs almost the same activity as the load function but updating the tables to the program very first state.

The secondary functions that are named previously behave in the following way:

- **separate(text,textT):** this function creates a table in which rows are filled with code lines and returns the full table.
- **setWhite(table):** it erases the background of the code table.
- **fillVAR(ram,own,pos):** this function fills the global memory table with the first two thousands positions of the ram that is passed as parameter. It also relates the position pointer and owner with its byte of information. The function returns the full table.
- **fillHEAP(ram,own,heap,stack):** it performs the same activity as the *fillVAR* function. Even though, the heap only has one hundred positions and it has the stack and heap pointers that are passed as parameters. It also returns the filled table.

- **updateRAM(ram,own,pos)**: this function deletes the global memory table and fills it again.
- **updateHEAP(ram,own,heap,stack)**: it performs the same function as *updateRAM* but with the heap table.
- **fillOutput(ioS,textO)**: this function erases the output window table (textO) and then fill it again with the ioS array.

4.6.4 Disconnect

The session control is done throughout the entire application with a PHP session. There is no custom time limit, but the default one is twenty-four minutes, as it is configure in the server.

As with every click in play or reload buttons the session is overwritten, it is not necessary for the user to log in every twenty-four minutes. The server will only close session automatically if the session is inactive the entire default time.

Even though there exists that time limit, if the user wants to close session at any instant, he can do it by pressing the 'Close Session' button that is in the right top corner of the web page. This button is present in all of the application views with the exception of the log-in page.

The functionality is handled by the code shown in Table 4.5.

```

<?php
session_start();
$string = date("d-m-Y_H:i:s");
$string = "{$string}_User:_{$_SESSION['username']}_closed_session.";
file_put_contents("/users/{$_SESSION['username']}/log",$string . PHP_EOL, FILE_APPEND | LOCK_EX);
session_destroy();
header('Location:../index.html');
?>

```

Table 4.5: Logging User Activity

This code writes in the log file the manual session close and then destroys the PHP session.

4.6.5 Information pages

There are three information pages. One of them is the about page, it shows the description of the application together with contact information. The others are help pages that explain the operation of the file selection and the simulation part.

Chapter 5

Validation

5.1 Introduction

In this chapter the different tests done to the application are going to be described. These tests were made in order to confirm that the different requirements of chapter 3 were fulfilled.

The equipment needed for carrying out these tests was a computer connected to the internet with a browser.

5.2 Obtained results

5.2.1 Test 1

In this test the login platform is going to be evaluated. As it can be seen in Figure 5.1 the application asks for an username and a password [Requirement: 1b].

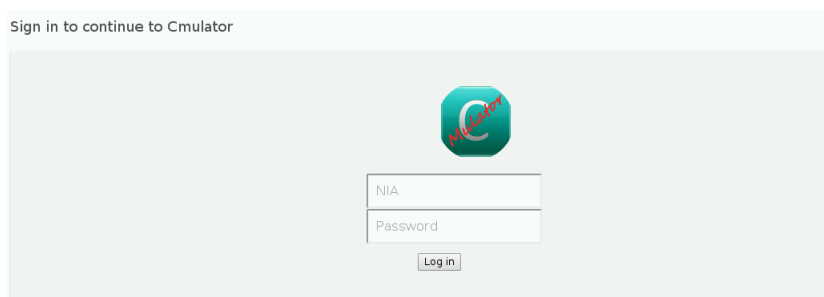


Figure 5.1: *Login*

If the wrong username or password is introduced the application shows an error message as in Figure 5.2.

Error
Invalid Username or Password

Figure 5.2: *Invalid Username or Password*

Moreover, if the user do not have permissions to access the webpage, the application will alert this and will not allow the visualization of the other pages [Requirement: 1c]. The error is shown in Figure 5.3.

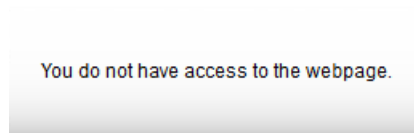


Figure 5.3: *Not Allowed User*

The navigation bar that is shown in Figure 5.4 is active in every page of the application [Requirement: 1d, 2b and 3g].

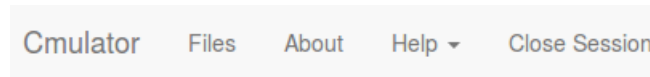


Figure 5.4: *Navigation bar*

If the user tries to access any web page that is not the login one without introducing its username and password, it will be automatically redirected to the login page [Requirement: 1e].

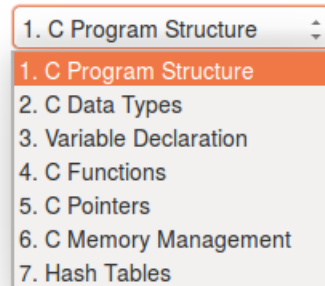
5.2.2 Test 2

In this test the chapter selection system will be checked.

As it can be seen in Figure 5.5 and in Figure 5.6 the files are ordered by chapters and by selecting one chapter the application displays all the files that are contained in that chapter [Requirements: 2a, 2c and 2e].

If the user enters and the professor has not released the chapters 5, 6 and 7, the web will look like Figure 5.7.

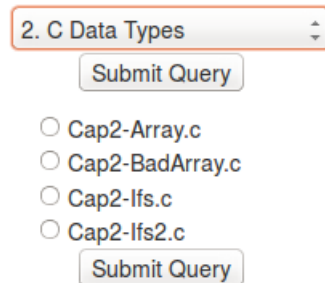
Please select a chapter to see the files:



A dropdown menu with a light gray background and a thin border. The selected item, "1. C Program Structure", is highlighted in orange. Below it, a list of seven options is shown in black text: "1. C Program Structure", "2. C Data Types", "3. Variable Declaration", "4. C Functions", "5. C Pointers", "6. C Memory Management", and "7. Hash Tables".

Figure 5.5: *Chapter Selection*

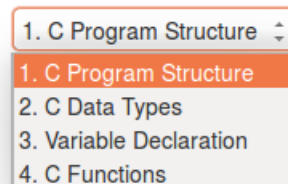
Please select a chapter to see the files:



A form with a light gray background and a thin border. At the top, a dropdown menu shows "2. C Data Types" selected. Below it is a "Submit Query" button. Underneath are four radio buttons, each followed by a filename: "Cap2-Array.c", "Cap2-BadArray.c", "Cap2-lfs.c", and "Cap2-lfs2.c". At the bottom is another "Submit Query" button.

Figure 5.6: *File Selection*

Please select a chapter to see the files:



A dropdown menu with a light gray background and a thin border. The selected item, "1. C Program Structure", is highlighted in orange. Below it, a list of four options is shown in black text: "1. C Program Structure", "2. C Data Types", "3. Variable Declaration", and "4. C Functions".

Figure 5.7: *Chapters for Students*

5.2.3 Test 3

In Figure 5.8 the simulation tool is shown. The RAM is displayed along with the C code and the output window. Moreover, the line that is executed is highlighted in red [Requirements: 3a, 3b, 3c, 3d, 3e and 3f].

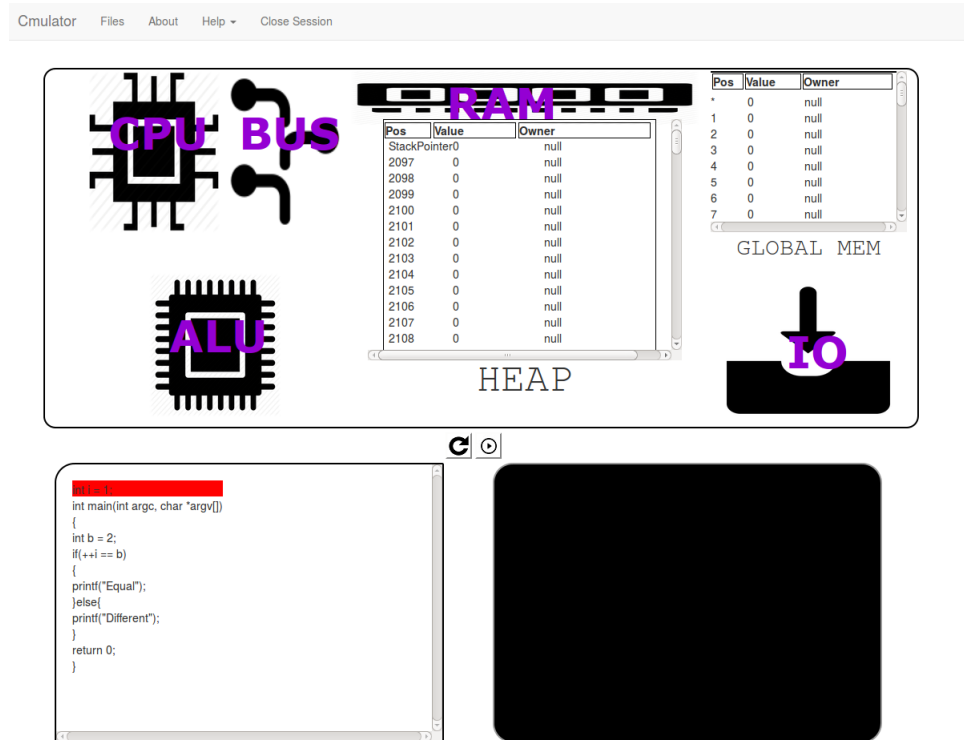


Figure 5.8: *Simulation Page*

5.2.4 Test 4

In the About page the contact information of the professor and the administrator is displayed as can be seen in Figure 5.9 [Requirement: 4a and 4b].

Lucia Uguina (4.1.C03): 100305498@alumnos.uc3m.es

Iria Estevez (4.1.A06): ayres@it.uc3m.es

Figure 5.9: *Contact Information*

5.2.5 Test 5

In order to check the administrative functionalities some files are going to be analysed.

When the user logs in the PHP code shown in Table 5.1 is executed.

```
<?php
[...]
if(!is_dir("/users/" . $ccid)){
    mkdir("/users/" . $ccid, 0700);
}
$string = date("d-m-Y_H:i:s");
$string = "{$string}_User:{$ccid}_logged.";
file_put_contents("/users/{$ccid}/log",$string . PHP_EOL, FILE_APPEND | LOCK_EX);
[...]
```

Table 5.1: Generating Log File

And this is done every time a user logs in, selects a file or logs out [Requirement: 1a].

The administrator is also able to log in the MySQL server and manage files and the chapters that each students' group can see [Requirements: 2d and 2f].

5.2.6 Test Comparison

In Table 5.2 the different tests and requirements are associated.

Requirement	Test 1	Test 2	Test 3	Test 4	Test 5
<i>1a</i>					✓
<i>1b</i>	✓				
<i>1c</i>	✓				
<i>1d</i>	✓				
<i>1e</i>	✓				
<i>2a</i>		✓			
<i>2b</i>	✓				
<i>2c</i>		✓			
<i>2d</i>					✓
<i>2e</i>		✓			
<i>2f</i>					✓
<i>3a</i>			✓		
<i>3b</i>			✓		
<i>3c</i>			✓		
<i>3d</i>			✓		
<i>3e</i>			✓		
<i>3f</i>			✓		
<i>3g</i>	✓				
<i>4a</i>				✓	
<i>4b</i>				✓	

Table 5.2: Tests Comparison

5.3 Conclusion

The application meets all the requirements specified in chapter 3.

It is also remarkable that this application has been used in class as a concept test and a reduced group of volunteer students has been able to test the *C-mulator* tool. This has been done in order to use, in a near future, *C-mulator* as an useful educational tool that provides a better understanding of concepts related to Systems Architecture subject.

Figures 5.10, and 5.11 show how the tool was used to explain concepts during several lessons.

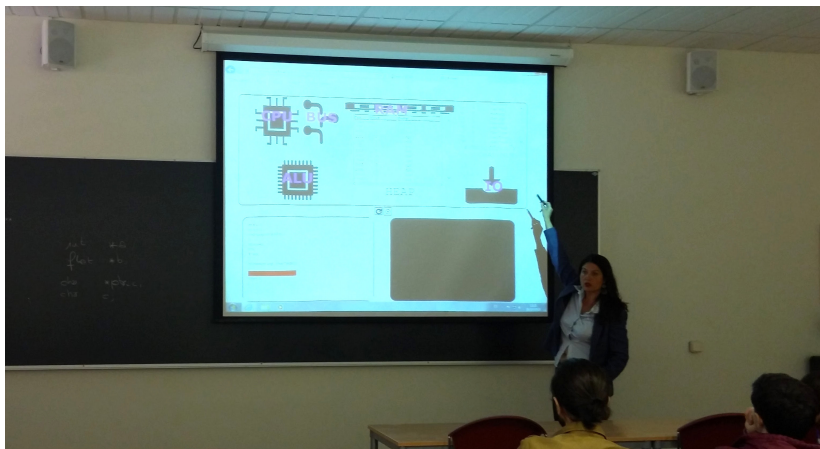


Figure 5.10: C-mulator *in class*

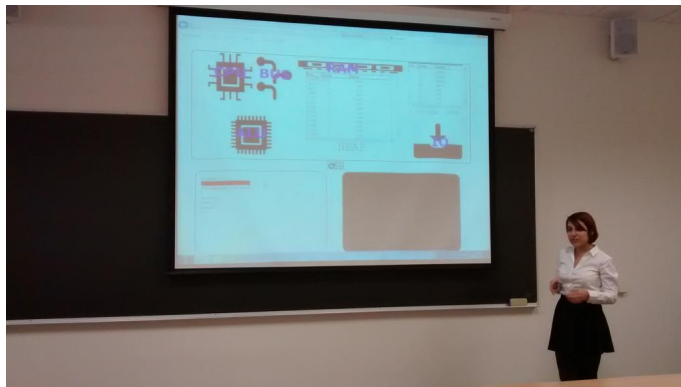


Figure 5.11: C-mulator *in class*

Chapter 6

Conclusions and Future Work

6.1 Introduction

Once the objectives and requirements of the project have been fulfilled and several tests have been done in order to prove that the application works as expected, the project has been brought to an end.

Even though the project has concluded with a satisfactory ending, some improvements that can be included in the application are proposed in this chapter.

6.2 Conclusions

This project has shown the development of a web application based in a simulation tool for C language. The application was a Client-Server web based in a three tiers architecture.

This tool has been designed from bottom to top in order to simplify the use of the simulation device and embed it into the web page. This web application has been used during the first four-month period of the university year in Systems Architecture subject.

This whole project has been useful for internalising and learning how a good project is done. That means, it has been useful in order to have a better and clearer idea of what will be the functions developed by a graduate engineer in a real work environment.

Finally, the development of the web application was good option in order to simplify the use of the simulation program. This also goes along with the fact that the public of this project is a group of students, then the simpler it is to run the

simulator, the better they will accept it as an useful tool.

6.3 Future Work

Once the requirements are fulfilled and the application is done, the next step is improving the functionalities covered by this tool.

Some of the possible future work lines are:

- Improving the response time of the server.
- Developing an scalable web application.
- Having a larger collection of files.
- Creating a service of files upload.
- Improving the code of the simulation part in order to be able to upload any kind of C files and simulate them.
- Enhancing the web simulation experience by creating more effects and showing the operations that are taking place not only inside the RAM but in any other component.
- Adding a GUI for the teaching staff in order to manage files and permissions without contacting the administrator.
- Completely separating the simulation tool module from the web application one.
- Implementing Java Compiler-Compiler (JavaCC) in order to be able to simulate C projects with more than one C file.
- Testing if there exists Learning Gain by using *C-mulator*.
- A paper about this project is being writing in order to present it in a conference.

All of these lines of future work are intended for creating a web service that could be use as a competitive tool for educational purposes. This tool would be able to simulate any kind of C program and tell the user what is happening at every moment in his code. This is only an enhancement of what this project is doing in the restricted environment of the Systems Architecture subject.

Chapter 7

Planning and Budget

7.1 Planning

In this section the duration of each project phase is going to be shown by itemising the tasks in how many working days each task lasted.

Firstly, in Table 7.1 the duration of the first project phase: the previous study, is shown.

Task Name	Duration	Start Date	End Date
<i>Previous Study Phase Block</i>	10d	Thu 02/19/2015	Wed 03/04/2015
<i>Possible Project Study</i>	5d	Thu 02/19/2015	Wed 02/25/2015
<i>Viability Study</i>	5d	Thu 02/26/2015	Wed 03/04/2015

Table 7.1: Previous Study Duration

Secondly, the next block of the application was the simulation tool. In Table 7.2 this part planning is shown.

Thirdly, the client implementation and design was done. This phase block is shown in Table 7.3

After that, the server and the database along with the communication between server and client and server and database was done. This duration is shown in Table 7.4.

The time dedicated to the tests phase is shown in Table 7.5.

Finally, this report was written time after the project was finished because the

Task Name	Duration	Start Date	End Date
<i>Simulation Tool Phase Block</i>	50d	Thu 03/05/2015	Wed 05/13/2015
<i>Simulation Design</i>	5d	Thu 03/05/2015	Wed 03/11/2015
<i>Simulation Implementation</i>	45d	Thu 03/12/2015	Wed 04/29/2015
<i>Simulation Tests</i>	10d	Thu 04/30/2015	Wed 05/13/2015

Table 7.2: Simulation Block Duration

Task Name	Duration	Start Date	End Date
<i>Client Phase Block</i>	25d	Thu 05/14/2015	Wed 06/17/2015
<i>Client Design</i>	5d	Thu 05/14/2015	Wed 05/20/2015
<i>Client Implementation</i>	15d	Thu 05/21/2015	Wed 06/10/2015
<i>Client Tests</i>	10d	Thu 06/11/2015	Wed 06/17/2015

Table 7.3: Client Block Duration

Task Name	Duration	Start Date	End Date
<i>Server and Database Phase Block</i>	51d	Thu 06/18/2015	Thu 08/27/2015
<i>Database Design</i>	3d	Thu 06/18/2015	Mon 06/22/2015
<i>Server Design</i>	3d	Tue 06/23/2015	Thu 06/25/2015
<i>Database Implementation</i>	5d	Fri 06/26/2015	Thu 07/02/2015
<i>Server Implementation</i>	25d	Fri 07/03/2015	Thu 08/06/2015
<i>Client-Server Communication</i>	10d	Fri 08/07/2015	Thu 08/20/2015
<i>Client-Server Communication</i>	5d	Fri 08/21/2015	Thu 08/27/2015

Table 7.4: Database and Server Block Duration

Task Name	Duration	Start Date	End Date
<i>Test Phase Block</i>	27d	Fri 08/28/2015	Mon 10/05/2015
<i>Application Tests</i>	3d	Fri 08/28/2015	Tue 09/01/2015
<i>Error Correction</i>	10d	Wed 09/02/2015	Tue 09/15/2015
<i>Improvements</i>	10d	Wed 09/16/2015	Tue 09/29/2015
<i>Final Tests</i>	4d	Wed 09/30/2015	Mon 10/05/2015

Table 7.5: Test Phase Block Duration

Systems Architecture subject began and the students were using the tool during the first four-month period of the university year. The duration of the report writing is shown in Table 7.6.

Task Name	Duration	Start Date	End Date
<i>Report Phase Block</i>	30d	Mon 01/11/2016	Fri 02/19/2016
<i>Report Writing</i>	30d	Mon 01/11/2016	Fri 02/19/2016

Table 7.6: Report Duration

To conclude, if all of these phases are summed up, skipping the report part as it was done afterwards, the result is Table 7.7.

7.1.1 Gantt Chart

In Figure 7.1 the Gantt Chart of the project is shown. In this graphic the itemisation described before is displayed.

7.2 Budget

In this section the project budget is detailed, in this budget the staff cost, machine cost and so on, are included. This cost computation is based in the Universidad Carlos III de Madrid budget manual [46].

Task Name	Duration	Start Date	End Date
End of Degree Project	163d	Fri 02/19/2015	Mon 10/05/2015
<i>Previous Study Phase Block</i>	10d	Thu 02/19/2015	Wed 03/04/2015
<i>Simulation Tool Phase Block</i>	50d	Thu 03/05/2015	Wed 05/13/2015
<i>Client Phase Block</i>	25d	Thu 05/14/2015	Wed 06/17/2015
<i>Server and Database Phase Block</i>	51d	Thu 06/18/2015	Thu 08/27/2015
<i>Test Phase Block</i>	27d	Fri 08/28/2015	Mon 10/05/2015

Table 7.7: Project Duration

It is established that a man can dedicate a maximum of 131.25 hours per month, as it is said in [46].

If 3 hours per day are dedicated by average. As not all of the days have been working days and not all of the days have the same dedicated time. The average dedication per month follows the formula:

$$Dedication(month) = \frac{Hours/day * Totaldays * NumberofMen}{MaximumHourspermonth}$$

Then the dedication of the different professionals that have being part of this project is:

- Senior Engineer

$$Dedication(month) = \frac{3*(10(previousstudy)+20(projectadvances)+30(reportwriting))*1}{131.25} = 1.37months$$

- Junior Engineer

$$Dedication(month) = \frac{3*(193)*1}{131.25} = 4.41months$$

- Test Verificator

$$Dedication(month) = \frac{3*(3+4+10+10)*3}{131.25} = 1.85months$$

Full Name	Category	Dedication	Cost/month	Cost
<i>Iria Manuela Estévez Ayres</i>	Senior Engineer	1.37	4289.54€	5876.67€
<i>Lucía Uguina Gadella</i>	Junior Engineer	4.41	2694.39€	11882.26€
<i>Not Especificed</i>	Test Verifier [47]	1.85	2086.41€	3859.86€
	Total	7.63	9070.34€	21618.79€

Table 7.8: Staff Cost

Then the staff cost is computed and it is shown in Table 7.8.

In order to compute the cost of personnel hiring the following formula has been used:

$$Cost = Dedication(months) * Salary/month$$

The gross cost of the machines used in this project is shown in Table 7.9.

Object	Cost	Taxes	Total Cost
<i>Computer</i>	1298.76€	21%	1571.50€
<i>Laptop</i>	495.79€	21%	599.90€
<i>Raspberry Pi 2</i>	77.56€	21%	93.85€
Total	1872.11€	21%	2265.25€

Table 7.9: Equipment Cost

The amortization is shown in Table 7.10.

Object	Cost	Dedicated Use	Dedication (months)	Devaluation period	Chargeable Cost
<i>Computer</i>	1298.76€	100%	9	60	194.81€
<i>Laptop</i>	495.79€	100%	9	60	74.37€
<i>Raspberry Pi 2</i>	77.56€	100%	3	60	3.88€
Total	1872.11€				273.06€

Table 7.10: Amortization

The chargeable cost is computed in the following way:

$$ChargeableCost = \frac{A*B}{C*D}$$

Being:

- **A** : the equipment cost without taxes.
- **B** : the dedication in months.
- **C** : the dedicated use.
- **D** : the devaluation period in months.

Once all of the costs are detailed, they have to be summed up with indirect costs of 20%. This is shown in Table 7.11.

Description	Total Cost
<i>Staff</i>	21618.79€
<i>Amortization</i>	273.06€
<i>Indirect Costs(20%)</i>	4378.33€
Total	26269.98€

Table 7.11: Total cost

Finally, the total budget with taxes is shown in Table 7.12.

Description	Total Cost
<i>Staff</i>	26269.98€
<i>Taxes</i>	21%
Total	31768.68€

Table 7.12: Total cost with taxes

Appendix A

Raspberry Configuration

In this appendix the configuration made in the Raspberry Pi 2 Model B will be explained. This Raspberry was configured to be used as an Apache-Tomcat server with PHP usability and a database using MySQL.

1 Installing Raspbian

First of all, the Raspbian image is downloaded from [48]. In this case, the Raspbian Jessie version was downloaded. Then, the Secure Digital (SD) card needs to be prepared for installing the images. For Windows systems, the Win32DiskImager utility has to be downloaded from [49]. Once it is downloaded, it has to be run as administrator.

After configuring Win32DiskImager by selecting the Raspbian Jessie image file and the appropriate drive letter, the one that corresponds to the SD. The SD will be prepared.

Now, the SD card has to be inserted inside the Raspberry Pi. Once the keyboard, mouse and the High-Definition Multimedia Interface (HDMI) are plugged in, the Raspberry is switched on.

After booting it will show the Raspbian symbol and will configure the system.

2 Installing and Configuring Apache-Tomcat

Firstly, in order to install Tomcat the commands shown in Table A.1 need to be run in the Raspberry Pi.

Secondly, for running Java applications in Tomcat the default JDK is needed. In

```
sudo apt-get update
sudo apt-get install tomcat7
```

Table A.1: Tomcat Installation

order to install it, the command shown in Table A.2 is executed. Then, Tomcat has to be restarted, this command is also shown in Table A.2.

```
sudo apt-get install default-jdk
sudo /etc/init.d/tomcat7 restart
```

Table A.2: JDK Installation

Once Tomcat is successfully installed, if the url `http://localhost:8080` is visited the page shown in Figure A.1 will be displayed.

It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at:
`/var/lib/tomcat7/webapps/ROOT/index.html`

Tomcat7 veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat7` and `CATALINA_BASE` in `/var/lib/tomcat7`, following the rules from `/usr/share/doc/tomcat7-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

tomcat7-docs: This package installs a web application that allows to browse the Tomcat 7 documentation locally. Once installed, you can access it by clicking [here](#).

tomcat7-examples: This package installs a web application that allows to access the Tomcat 7 Servlet and JSP examples. Once installed, you can access it by clicking [here](#).

tomcat7-admin: This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the [manager webapp](#) and the [host-manager webapp](#).

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager-gui". The host-manager webapp is restricted to users with role "admin-gui". Users are defined in `/etc/tomcat7/tomcat-users.xml`.

Figure A.1: *Tomcat Welcome Page*

3 Installing and Configuring MySQL and phpMyAdmin

The next step of the server configuration is installing the database management system and a GUI that will simplify the database management.

In order to do so, MySQL needs to be installed. First of all, all the packages management tools need to be up-to-date and the latest software available has to be installed. For this, the commands shown in Table A.3 need to be executed.

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Table A.3: Update System

Later on, when the system has installed all the necessary packages, the MySQL ones are going to be installed. In Table A.4 the command needed is shown.

```
sudo apt-get install mysql-server mysql-client
```

Table A.4: MySQL Installation

While MySQL is being configured, the system will ask the user to set a 'root' password. A window similar to the shown in Figure A.2 will be displayed.

Now that MySQL has been successfully installed and configured. The server also needs the phpMyAdmin tool and PHP. In order to get them, the commands shown in A.5 have to be executed.

```
sudo apt-get install php5-mysql
sudo apt-get install phpmyadmin
```

Table A.5: PHP and phpMyAdmin Installation

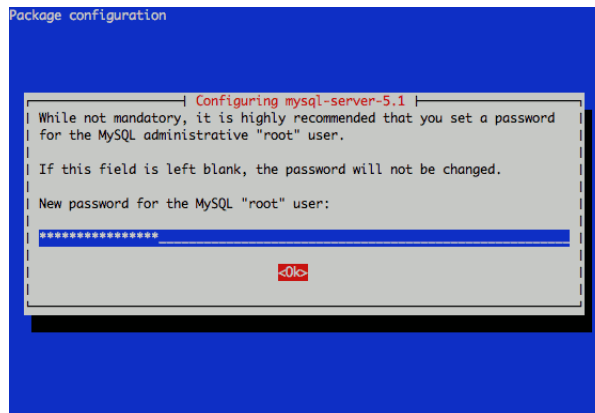


Figure A.2: *MySQL Configuration*

For the phpMyAdmin configuration, the system will ask for some information. First of all, phpMyAdmin will demand the default web server as shown in Figure A.3. Apache 2 has to be selected.

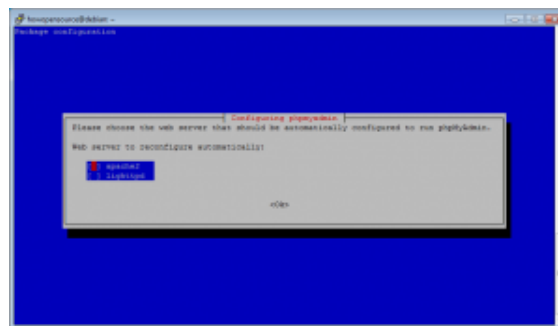
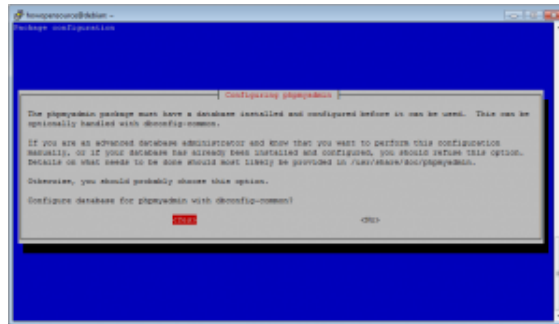
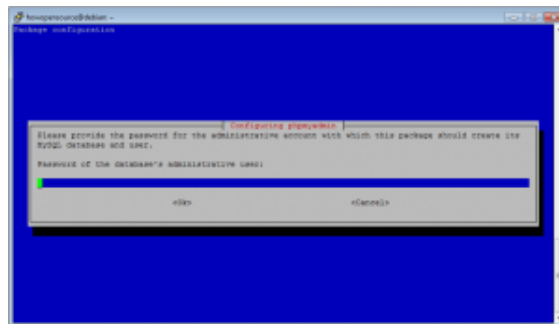


Figure A.3: *phpMyAdmin Configuration*

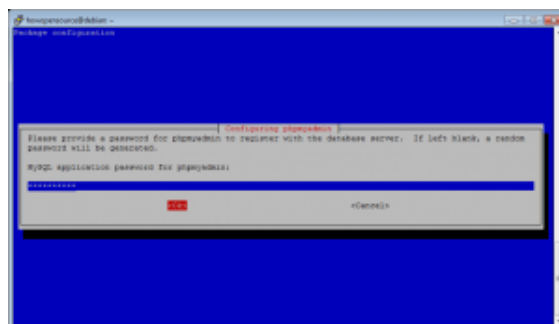
The following question will ask for creating a database needed for phpMyAdmin. This can be seen in Figure A.4.

Figure A.4: *phpMyAdmin Configuration*

The next step is to introduce the MySQL root password. This is shown in Figure A.5.

Figure A.5: *phpMyAdmin Configuration*

The final step is to configure the root password for phpMyAdmin as shown in Figures A.6 and A.7.

Figure A.6: *phpMyAdmin Configuration*

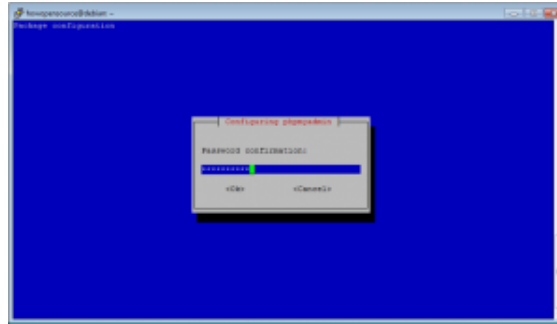


Figure A.7: *phpMyAdmin Configuration*

Now that everything is configured, MySQL server can be accessed through command or by going to <http://localhost/phpmyadmin>.

4 Configuring PHP in Tomcat

One of the main issues of this configuration is that Tomcat do not allow a native PHP configuration. Thus, JavaBridge [50] is going to be installed.

JavaBridge provides a PHP environment for Tomcat. Firstly, the tool has to be downloaded from <http://php-java-bridge.sourceforge.net/doc/download.php>, there are several ways to download it, the most complete one is the Binary option.

Once it is downloaded, Tomcat has to be stopped. Then the files `JavaBridge.jar`, `php-servlet.jar` and `php-script.jar` has to be moved to the tomcat library directory, in this case this directory is `/usr/share/tomcat7/lib`.

The next step is to edit the file `/var/lib/tomcat7/conf/web.xml` the lines marked with a '+' sign in Table A.6 need to be added.

```

<web-app xmlns =... >

+ <listener><listener-class>php.java.servlet.ContextLoaderListener
  </listener-class></listener>
+ <servlet><servlet-name>PhpJavaServlet</servlet-name><servlet-
  class>php.java.servlet.PhpJavaServlet</servlet-class>
+ </servlet>
+ <servlet><servlet-name>PhpCGIServlet</servlet-name><servlet-
  class>php.java.servlet.fastcgi.FastCGIServlet</servlet-class>
+ <init-param><param-name>prefer_system_php_exec</param-name><
  param-value>On</param-value></init-param>
+ <init-param><param-name>php_include_java</param-name><param-
  value>Off</param-value></init-param>
+ </servlet>
+ <servlet-mapping><servlet-name>PhpJavaServlet</servlet-name><url
  -pattern>*.phpjavabridge</url-pattern></servlet-mapping>
+ <servlet-mapping><servlet-name>PhpCGIServlet</servlet-name><url-
  pattern>*.php</url-pattern></servlet-mapping>
...

</web-app>

```

Table A.6: JavaBridge Configuration

Finally, the last step is to start Tomcat again.

5 TLS Configuration

In order to add the cryptography protocol Transport Layer Security (TLS) to the Apache-Tomcat server the following configuration is necessary.

A new keystore has to be created. The commands shown in Table A.7 create the keystore. This command will ask for a password, the default one for Apache-Tomcat is "changeit".

```
/usr/bin/java/bin/keytool -genkey -alias tomcat -keyalg RSA
-keystore /
```

Table A.7: Keystore Creation

Then, the lines in Table A.8 are added to the configuration file `/var/lib/tomcat7/conf/server.xml`.

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    port="8443" maxThreads="200"
    scheme="https" secure="true" SSLEnabled="true"
    keystoreFile="/.keystore" keystorePass="changeit"
    clientAuth="false" sslProtocol="TLS"/>
```

Table A.8: Keystore Creation

Finally, Tomcat has to be restarted.

6 Application Deployment

Once the server is fully configured. The *war* files can be moved to `/var/lib/tomcat7/webapps`. In this case the *war* is named as 'JavaServlet'. So, in order to access the website the URL that has to be typed is `https://SERVER_IP:8443/JavaServlet`.

Bibliography

- [1] David R Krathwohl. “A revision of Bloom’s taxonomy: An overview”. In: *Theory into practice* 41.4 (2002), pp. 212–218.
- [2] Richard M. Felder and Linda K. Silverman. “Learning and Teaching Styles”. In: *Engineering Education* 78.7 (1988), pp. 674–681.
- [3] John L. Dobson. “A comparison between learning style preferences and sex, status, and course performance”. In: *Advances in Physiology Education* 34.4 (Dec. 2010), pp. 197–204.
- [4] Abbas Pourhosein Gilakjani Fouzieh Sabzian and Sedigheh Sodouri. “Use of Technology in Classroom for Professional Development”. In: *Journal of Language Teaching and Research* 4.4 (July 2013), pp. 684–692.
- [5] Kirsti Ala-Mutka Essi Lahtinen and Hannu-Matti Järvinen. “A study of the difficulties of novice programmers”. In: *ACM SIGCSE Bulletin* 37.3 (Sept. 2005), pp. 14–18.
- [6] Athanassios Jimoyiannis and Vassilis Komis. “Computer simulations in physics teaching and learning: a case study on students’ understanding of trajectory motion”. In: *Computers & Education* 36.2 (Feb. 2001), pp. 183–204.
- [7] Jefatura del Estado. “Ley Orgánica de Protección de Datos de Carácter Personal”. In: *BOE* (Dec. 1999).
- [8] Agencia Española de Protección de Datos. “Guía de Seguridad de Datos”. In: *NILO Industria Gráfica* (2010). URL: https://www.agpd.es/portalwebAGPD/canaldocumentacion/publicaciones/common/Guias/GUIA_SEGURIDAD_2010.pdf.
- [9] Universidad Carlos III de Madrid. *Datos Personales: utilización y derechos*. URL: http://portal.uc3m.es/portal/page/portal/varios_cg/dat_pers_uti_der (visited on 01/31/2016).

- [10] John M. Gallaugh and Suresh C. Ramanathan. “Choosing a client/server architecture”. In: *Information Systems Management* 13.2 (Jan. 1996), pp. 7–13.
- [11] Adam Freeman and Steven Sanderson. *Pro ASP.NET MVC 3 Framework*. Apress, 2011.
- [12] Rick Miller and Raffi Kasparian. *Java for artists : the art, philosophy, and science of object-oriented programming*. Falls Church, 2006.
- [13] Douglas E. Comer and David L. Stevens. *Internetworking with TCP/IP*. vol. III: client-server programming and applications. Prentice-Hall, 1993.
- [14] ITL Education Solutions Limited. *Express Learning: Database Management Systems*. Pearson India, Jan. 2012.
- [15] Youssef Bassil. “A Comparative Study on the Performance of the Top DBMS Systems”. In: *Journal of Computer Science & Research (JCSCR)* 1.1 (Feb. 2012), pp. 20–31.
- [16] Robert Stackowiak Rick Greenwald and Jonathan Stem. *Oracle Essentials: Oracle Database 12c*. O’Reilly, 2013.
- [17] Paul Nielsen with Mike White and Uttam Parui. *Microsoft SQL Server 2008 Bible*. Wiley, 2009.
- [18] Jason Buffington. *Data Protection for Virtual Data Centers*. Sybex, July 2010. Chap. Microsoft SQL Server.
- [19] Michael Kofler. *The Definitive Guide to MySQL 5*. Vol. 3rd Edition. Apress, 2005.
- [20] Adrian Neagu and Robert Pelletier. *IBM DB2 9.7 Advanced Administration Cookbook*. Packt Publishing Ltd., Mar. 2012.
- [21] Joyce Cox and Joan Lamber. *Microsoft Access 2013 Step by Step*. Microsoft Press, 2013.
- [22] Jess Thompson. “Avoiding a middleware muddle”. In: *IEEE Software* 14.6 (Dec. 1997), pp. 92–95.
- [23] Margaret Ticknor & Alan Corcoran & Balazs Csepregi-Horvath & Addison Goering & José Pablo Hernandez & Julien Limodin & Sergio Straessli Pinto. *IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide*. IBM Redbooks, Aug. 2011.
- [24] Aleksa Vuktic and James Goodwill. *Apache Tomcat 7*. Apress, Sept. 2011.

- [25] Ben Laurie and Peter Laurie. *Apache: The Definitive Guide*. Vol. Third Edition. O'Reilly, 2003.
- [26] Netcraft. *November 2015 Web Server Survey*. Nov. 2015. URL: <http://news.netcraft.com/archives/2015/11/16/november-2015-web-server-survey.html> (visited on 01/10/2016).
- [27] Dipankar Sarkar. *Nginx 1 Web Server Implementation Cookbook*. Packt Publishing Ltd., May 2011.
- [28] Clément Nedelcu. *Nginx HTTP Server Second Edition*. Packt Publishing Ltd., July 2013.
- [29] Jason Brittain with Ian F. Darwin. *Tomcat: The Definitive Guide, Second Edition*. O'Reilly, Oct. 2007.
- [30] Joe Kraynak. *The Complete Idiot's Guide® To HTML5 and CSS3*. Alpha Books, June 2011.
- [31] Jon Duckett. *Beginning HTML, XHTML, CSS, and JavaScript®*. Wrox, Dec. 2009.
- [32] Paul Wilton and Jeremy McPeak. *Beginning JavaScript®, Fourth Edition*. Wrox, Oct. 2009.
- [33] Patrick Carey. *New Perspectives on Creating Web Pages with HTML, XHTML, and XML*. Course Technology, May 2009.
- [34] Jim Boulton. *100 Ideas that Changed the Web*. Laurence King, Aug. 2014.
- [35] Bogdan Brinzarea Audra Hendrix and Cristian Darie. *AJAX and PHP*. Packt Publishing, Dec. 2009.
- [36] Tim Bray et al. *Extensible Markup Language (XML)*. W3C, Aug. 2006.
- [37] Clark Evans Oren Ben-Kiki and Ingy döt Net. *YAML Ain't Markup Language (YAML™) Version 1.2*. Patched, Oct. 2009.
- [38] Ben Smith. *Beginning JSON*. Apress, Feb. 2015.
- [39] William Yurcik and Hugh Osborne. "A crowd of little man computers: visual computer simulator teaching tools". In: *Simulation Conference, 2001. Proceedings of the Winter*. Vol. 2. IEEE. 2001, pp. 1632–1639.
- [40] Eduardo S Cordeiro et al. "DCMSim: Didactic cache memory simulator". In: *Frontiers in Education, 2003. FIE 2003 33rd Annual 2 (2003)*, F1C–14.
- [41] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. "DRAMSim2: A cycle accurate memory system simulator". In: *Computer Architecture Letters* 10.1 (2011), pp. 16–19.

- [42] Herbert Grünbacher. “Teaching computer architecture/organisation using simulators”. In: *Frontiers in Education Conference, 1998. FIE’98. 28th Annual*. Vol. 3. IEEE. 1998, pp. 1107–1112.
- [43] Maria Grigoriadou, Evangelos Kanidis, and Agoritsa Gogoulou. “A Web-based educational environment for teaching the computer cache memory”. In: *Education, IEEE Transactions on* 49.1 (2006), pp. 147–156.
- [44] *GNU C reference manual*. URL: <http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html> (visited on 02/05/2016).
- [45] Sue Jenkins. *Web Design All-in-One For Dummies*. Wiley Publishing, 2009.
- [46] Universidad Carlos III de Madrid. *Guía Presupuestaria para TFG*. URL: https://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG2028329_1.xlsx (visited on 01/30/2016).
- [47] *Test Verificator Salary*. URL: <http://www.expoqa.com/pdf/expoqa12/SalariosTestersEspaña-ES-V03.pdf> (visited on 01/30/2016).
- [48] *Raspbian Download*. URL: <https://www.raspberrypi.org/downloads/raspbian/> (visited on 05/15/2015).
- [49] *Win32DiskImager Utility*. URL: <http://sourceforge.net/projects/win32diskimager/> (visited on 05/15/2015).
- [50] *JavaBridge*. URL: <http://php-java-bridge.sourceforge.net/pjb/> (visited on 05/17/2015).