Universidade de Aveiro
2016

Departamento de Eletrónica,
Telecomunicações e Informática

Fábio André Ferreira Borges Lourenço

Ambiente Integrado para Aprendizagem e Desenvolvimento de Aplicações Móveis

Integrated Learning and Development Environment for Mobile Applications

Universidade de Aveiro
2016

Departamento de Eletrónica,
Telecomunicações e Informática

Fábio André Ferreira Borges Lourenço

Ambiente Integrado para Aprendizagem e Desenvolvimento de Aplicações Móveis

Integrated Learning and Development Environment for Mobile Applications

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Eletrónica e Telecomunicações, realizada sob orientação científica do Professor Doutor Aníbal Manuel de Oliveira Duarte, Professor Catedrático da Universidade de Aveiro e do Doutor Rui Pedro Figueiredo Marques, Equiparado a Assistente 1º Triénio da Universidade de Aveiro.

"Just play. Have fun. Enjoy the game."
— Michael Jordan

o júri / the jury

presidente / president

Professor Doutor José Carlos da Silva Neves
Professor Catedrático da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Aníbal Manuel de Oliveira Duarte
Professor Catedrático da Universidade de Aveiro (orientador)

Professor Doutor Luís Manuel de Jesus Sousa Correia
Professor Associado Com Agregação, Departamento de Engenharia Eletrotécnica e de Computadores - Ist (arguente principal)

agradecimentos / acknowledgements

Para começar, um obrigado a esta academia por disponibilizar um curso de excelência nesta área de topo.

Um obrigado ao meu orientador, o Professor Doutor Manuel de Oliveira Duarte, pela oportunidade de desenvolver um projeto do meu interesse.

Agradeço ao meu coorientador, o Doutor Rui Marques, pela colaboração e conselhos essenciais para a realização deste projeto.

Ao colaborador que me acompanhou e ajudou nesta etapa final da minha formação, o Mestre António Alves, um obrigado.

Aos meus amigos, aqueles que posso chamar de "minha malta", que levo para a vida e a quem devo grandes momentos, um obrigado por toda ajuda académica e não académica.

Agradeço aos meus avós e aos meus tios por todo o apoio e força que me deram e por sempre acreditarem e confiarem em mim.

Um grande obrigado à minha namorada, Ana Mano, por todo o apoio nas alturas complicadas, pela motivação e preocupação.

Por fim, mas sem dúvida de maior importância, obrigado aos meus pais e irmã, por serem a minha família, por me aturarem e ajudarem, por todas as oportunidades que me dão e por nunca me terem falhado. Obrigado pela educação e princípios que me incutiram, sem eles nunca chegaria onde cheguei.

palavras-chave Sistemas de Informação Web; Educação na Engenharia; Auto didata

resumo

Esta dissertação contribuiu para o desenvolvimento de uma ferramenta didática estruturada que integrasse informação de qualidade já existente sobre Sistemas de Informação Web, proveniente de fontes devidamente credenciadas, disponibilizando-a ao público-alvo (alunos de engenharia fora da área das tecnologias da informação e comunicação), indicando e ajudando a encontrar o melhor caminho a seguir nessa aprendizagem.

A ferramenta didática encontra-se atualmente em três formatos diferentes – Website que utiliza um software de Sistema de Gestão de Conteúdos, aplicação móvel Android e documento de texto – tentando assim abranger o maior número de indivíduos dentro do público-alvo.

A ferramenta didática encontra-se orientada para a autoaprendizagem devidamente estruturada. Aborda os fundamentos teóricos principais sobre os Sistemas de Informação Web: conceitos, definições, arquitetura e as suas camadas, componentes de construção (bases de dados, Web *Services*, interface com o utilizador nos formatos de aplicação móvel Android e Website). Disponibiliza receitas para casos de uso prático que são expansíveis a diversas áreas, como, por exemplo, o desenvolvimento de um sistema que faz uso de um servidor LAMP, no qual existem serviços RESTful que fornecem a informação presente numa base de dados para ser acedida numa aplicação móvel Android.

A ferramenta não se encontra ainda testada e validada na comunidade estudantil, mas as primeiras impressões são muito promissoras.

keywords Web Information Systems; Engineering Education; Self-tutoring

abstract

This dissertation has contributed to develop a structured didactic tool that integrates existing quality information on Web Information Systems, from properly accredited sources, making it available to the target audience (engineering students outside the area of information and communication technologies), indicating and helping them to find the best course of action in this learning process.

The didactic tool can be currently found in three different formats – Website that uses a Content Management System software, Android mobile application and text document – thus attempting to reach the largest number of individuals within the target audience.

The didactic tool is oriented to properly structured self-learning. It discusses the main theoretical foundations on Web Information Systems: concepts, definitions, architecture and its layers, building components (databases, Web Services, user interface in Android mobile application and Website formats). Provides recipes for cases of practical use that are expandable to different areas, for example, the development of a system that makes use of a LAMP server, in which there are RESTful services that provide the information stored in a database to be accessed in an Android mobile application.

The tool has not been yet tested and validated in the student community, but first impressions are very promising.

Contents

List of Abbreviations

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BEEP	Blocks Extensible Exchange Protocol
CDA	Content Delivery Application
CMA	Content Management Application
CSS	Cascading Style Sheets
DBMS	Database Management System
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
DNS	Domain Name System
DOM	Document Object Model
DQL	Data Query Language
EJB	Enterprise JavaBeans
FK	Foreign Key
FTP	File Transfer Protocol
GPLv2	General Public License version 2
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IP	Internet Protocol
IS	Information Systems
JS	JavaScript
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LAMP	Linux, Apache, MySQL and PHP
MVC	Model-View-Controller
NTP	Network Time Protocol
OO	Object-Oriented
OOP	Object-Oriented Programming
OSI	Open Systems Interconnection

PF	Primary Foreign
PK	Primary Key
REST	REpresentational State Transfer
RPC	Remote Procedure Call
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Access Object Protocol
SQL	Structured Query Language
SSH	Secure Shell
TCP	Transport Control Protocol
UBR	UDDI Business Registry
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
WIS	Web Information System
WSDL	Web Service Description Language
WYSIWYG	What You See Is What You Get
XML	eXtensible Markup Language

List of Figures

List of Tables

1. Introduction

1.1 Background and Motivation

Nowadays, Web Information Systems are present in almost all activity sectors, including health, education, entertainment, commerce, manufacturing, banking, insurance, tourism, transportation, and many others. Their impact is twofold:

- Used as a means of reinventing and making more effective the conventional activities and procedures of many sectors;
- Make possible to deal with new complexities, constant transformations, rapidly evolving business models and ultra-large-scale operations – which, themselves, are, to a high extent, consequences of the increasing usage of information and communication technologies.

The acute need for professionals skilled on web based information technologies that, some years ago, happened mostly inside leading digital industries, has spread to many other activity sectors. The capacity of taking advantage of such resources is now a strong competitive factor for individuals and enterprises.

However, many engineering *curricula* outside the areas of information and communications technologies include little or none disciplines addressing the basic knowledge required for the incorporation of Web Information Systems in their business domains. As a consequence, many young graduates loose opportunities of more effective professional integration and success For these reasons, it is common to find students that, realizing the importance of the subject, decide to learn about Web based Information Systems on their own. Usually this is done resorting to materials available in the internet space, for example, in YouTube. Most of these materials have no pedagogic or scientific quality control, resulting frequently in poor learning outcomes and even misconceptions.

Thus, revealing the need for the existence of simple to use but quality guaranteed materials adjusted for the introduction of the essential steps and resources involved in the creation and exploitation of Web Information Systems.

1.2 Objectives

With the motivation in mind, the objective of this project is to contribute towards the development of a structured didactic tool that addresses Web Information Systems overall concepts and aggregates them in a simple manner using quality proved references. The tool is envisaged to self-paced learning and targets engineering students outside the area of information and communication technologies.

1.3 Approach

In setting up the development of a tool envisaged to self-paced learning of Web Information Systems, the idea was not that of replacing the many high quality texts and online courses existent on the subject.

The objective was to provide students with a roadmap able to provide support in the different steps involved with the development of Web Information Systems. This roadmap, itself should be supported as much as possible by already existent educational materials and only when absolutely necessary new courseware should be developed.

In order to achieve the wanted goal, it is necessary to elaborate a portfolio of theoretical concepts and practical implementations that constitute a Web Information System, which will then

be structured so that it can be made available in different formats, in order to reach as many users as possible.

To boost student confidence three usage scenarios were considered:

- Usage scenario 1 – The student has not yet started the development of a Web based Information System and needs full guidance about concepts, development methodology, building components, procedures and other resources;
- Usage scenario 2 – The student has already started the development and is faced with difficulties which is not able to overcome;
- Usage scenario 3 – The student already knows about some of the building components and wants to learn the remaining ones, in order to be able to develop a full stack Web Information System.

In the three cases, the tool provides support and guidance helping the user to understand the fundamental concepts and then gaining access to a collection of building components, procedures and other resources.

Besides the text document format, it is intended that the tool is available on Website and mobile application formats.

The structured tool has essentially the following components:

- Fundamental WIS theoretical concepts;
- Building components, implementation procedures and resources;
- A presentation of possible building components and basic procedures necessary for developing WIS;
- An application case study illustrating how the previous materials can be used in a practical context.

Several case studies are also provided where the student can find exemplificative situations that can be adapted to a large variety of use cases. In the main case study, a system three-tiers architecture is adopted, composed by the following building blocks:

- The usage of a LAMP (Linux, Apache, MySQL and PHP) environment regarding the server;
- The usage of REST (Representational State Transfer) architecture as an approach to Web Services;
- The usage of Android devices as User Interfaces.

1.4 Dissertation Structure

This dissertation is structured initially introducing the Web Information Systems and theirs structure, following by referencing its building components, and finally elucidating the development of the didactic structured tool as practical use case.

This document is organized as the following:

- Chapter 2 – Information Systems and Web Information Systems are defined and its elements are introduced;
- Chapter 2.3 – the Web Information Systems three-tiers architecture is approached and its tiers are illustrated, starting in the User Interface tiers, moving to the Data Transfer tier, and finishing with the Data Processing and Storage tier;
- Chapter 2.4 – here TCP/IP and OSI models are approached as Data Flow and System Interconnection Representation, comparing the models and explaining their layers;
- Chapter 3 – in this chapter Web Information Systems components are introduced, which correspond to the building blocks of this type of system. This chapter is divided in:

- Databases and Database Management Systems, which are introduced as information storing and processing elements;
- Web Services, who are approached as the element that processes data exchange between architectural tiers;
- Android applications and Websites, that are referenced as User Interface elements.
- Chapter 4 – Content Management Systems are introduced as a complete solution that represents a WIS. Different CMS's open source software is compared;
- Chapter 5 – Use case implementation and development is described, where the three formats of the didactic tool and their functionalities are illustrated;
- Chapter 6 – this chapter includes conclusions, future work and expected outcome.

2. Web Information Systems Definition and Architecture

2.1 Information Systems Definition

An Information System is a set of interrelated elements or components that collect, process, store and disseminate data and information. [1]

An Information System (IS) is composed the following elements:

- Input – gathers and captures raw data;
- Processing – converts and transforms data into useful outputs;
- Output – production of useful information, normally in the form of documents or reports;
- Feedback – used to make changes to Input or Processing activities;
- Storage – stores all the information in the system.

2.2 Web Information Systems

A Web Information System refers to an Information System that is based on the Internet. It consists of five basic resources: [2]

- Personnel – software Engineer, database administrator, end-users, among others;
- Hardware – server, terminals, computer peripherals, among others;
- Software – operative system, applications, utilities, DBMS, API's, GUI's, and so on;
- Networks – communication media, network support, and others;
- Data – information, knowledge, personal data, databases, and many other.

"*In the 21st century, the idea of an information system that does not rely on sophisticated information technologies is almost unimaginable*." [2]

Throughout the years, with the arrival of smartphones and the change in User Interfaces software development, the Web Information Systems have experienced some changes in a way that its top-most level can be divided into two main components: [3]

- Website – set of inter-linked objects (text, image, video, and others) associated to a domain, organized in pages that are displayed in a Web browser software;
- Web application – application that uses data provided by the Web to display content in many devices, like computer or mobile applications; the information is obtained by the use of Web Services which are mechanisms designed to implement machine-to-machine interaction over the Web, in a simple manner.
-

2.3 Web Information System Architecture

It is necessary to define an architecture to plan the development of any system. In Web Information System it is required an architecture that allows both the bulk storage of information as providing all the contents necessary for the user to interact with the system and its applications. Thus, there is also the need to have information interpreters and translators that enable the display of information to the user in an intelligible form.

A three-tier architecture has characteristics that provide the development of Web Information System in a structured and organized way, without giving up flexibility and scalability. This architecture is formed by three tiers, which are below introduced and are represented in Figure 1. [4]

- User Interface tier – User Interface; accounts for singular characteristics of various devices; mobile applications (Android, iOS, Windows, others); Internet of things; Web browser; responsible for experience delivery;
- Data Transfer tier – describes services (API's); determines the optimal way to deliver content; integrates services and transforms data bi-directionally; handles the information requested and delivered by the client; applies logical rules that combine databases data to user applications;
- Data Processing and Storage tier – is the bridge between internally and externally supplied data and functionality; database data access; visualization and creation of data; proprietary and third-party services (API's).

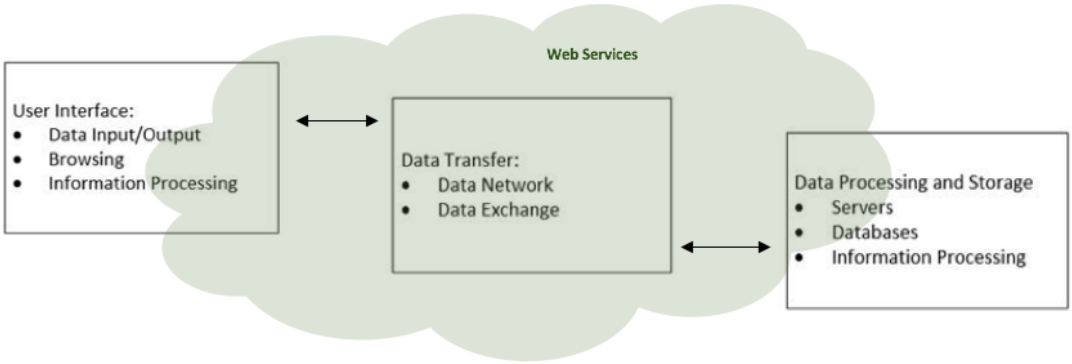


Figure 1 – Three-tier architecture of a Web Information System

The three-tier architecture approached, which divides the Web Information Systems into three major development groups, is responsible for every action in these systems: [4]

- The User Interface tier interacts with user through Web browsers, computer and mobile applications and others;
- The Data Transfer tier controls and performs detailed processing of requests, information and services;
- The Data Processing and Storage tier accesses and processes data stored in databases, and allocates services.

For simplicity, all concepts will be explained towards the development of these three groups, avoiding the abstraction of a four-tier architecture that is recently coming to life.

The architecture components are represented as an example in Figure 2. The components are usually independent from each other.

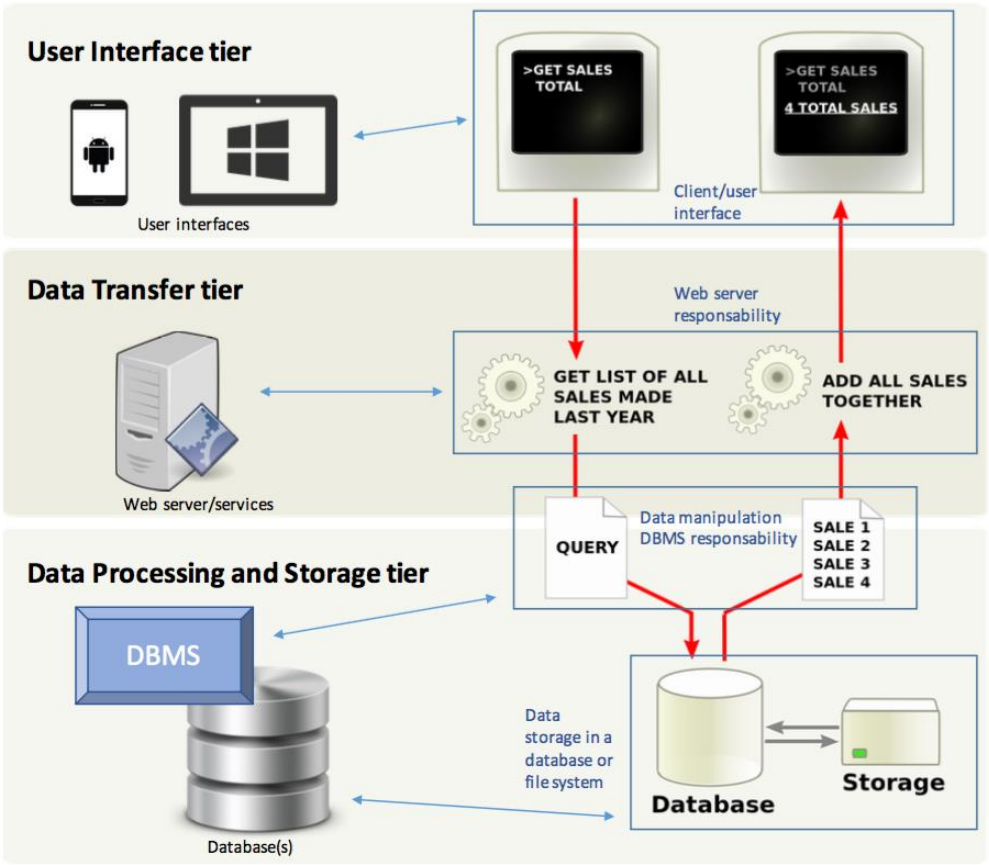


Figure 2 – The WIS architecture components of its main group [5]

2.3.1 User Interface Tier

The User Interface tier, as its name indicates, is the tier that interacts with the user, namely the User Interface. Interaction with the user can be done in many ways, such as a Web browser application that is no more than a Website, a computer application or program, a mobile application like and Android or iOS app, internet of things, among others.

In this tier information as to be displayed so that the user perceives what is being shown to him. It also collects information that needs to be processed by the lower tiers. [4]

Due to its big popularity and usability, Android applications will be approached in this tool as main the User Interface tier solution, although Websites it will also be mentioned.

2.3.2 Data Transfer Tier

The Data Transfer tier is a thin layer that can have great impact on the performance of the whole. Though it does not store neither displays information, it is responsible for all the logical work in the system, being its coordinator. This tier processes commands, makes logical evaluations and decisions, performs calculations, moves and processes data between the surrounding tiers, and so on.

In example, when using an Android application in different size and resolution devices, one of the Data Transfer tier jobs is to properly display the information in each screen type, applying screen size calculation methods and deciding the best definitions for the components to show (like the font size).

Another example can be the user filling a form in a Website, where the Data Transfer tier must be able to gather the input information and call the right API's, filling their fields correctly, to provide the information to the database. [4]

2.3.3 Data Processing and Storage Tier

The Data Processing and Storage tier is the lowest level tier that will be approached in this guide.

Its main functionalities are hosting a Web server and provide database access to store and retrieve information. Also it may include third-party services.

Behind this tier there is one or several databases or file systems, and this tier provides access to them all without the next tier knowing which database or file system it is accessing. [4]

Although there are several ways and a billion combinations that resume in a Data Processing and Storage tier, in this document it will be approached a LAMP server solution. In this document's case, LAMP stands for Linux, Apache, MySQL and PHP. This allows building a simple Data Processing and Storage tier by creating a MySQL database to store and retrieve information using an Apache server and PHP scripts to provide access to the database for Data Transfer tier. The access to the database is granted by providing an API using Web Services.

2.3.4 Netflix's Architectural Design as Example

One of the most notorious companies that uses four-tier architectural design (an abstraction of the three-tier model) is Netflix. Supporting more than 1000 devices types, handling more than 50000 requests per second during peak hours, while deploying new features almost every day, Netflix defines the need for performance, flexibility, scalability and agility. In addition, Netflix's platform is largely built on open source software. [4] Figure 3 represents its architecture.

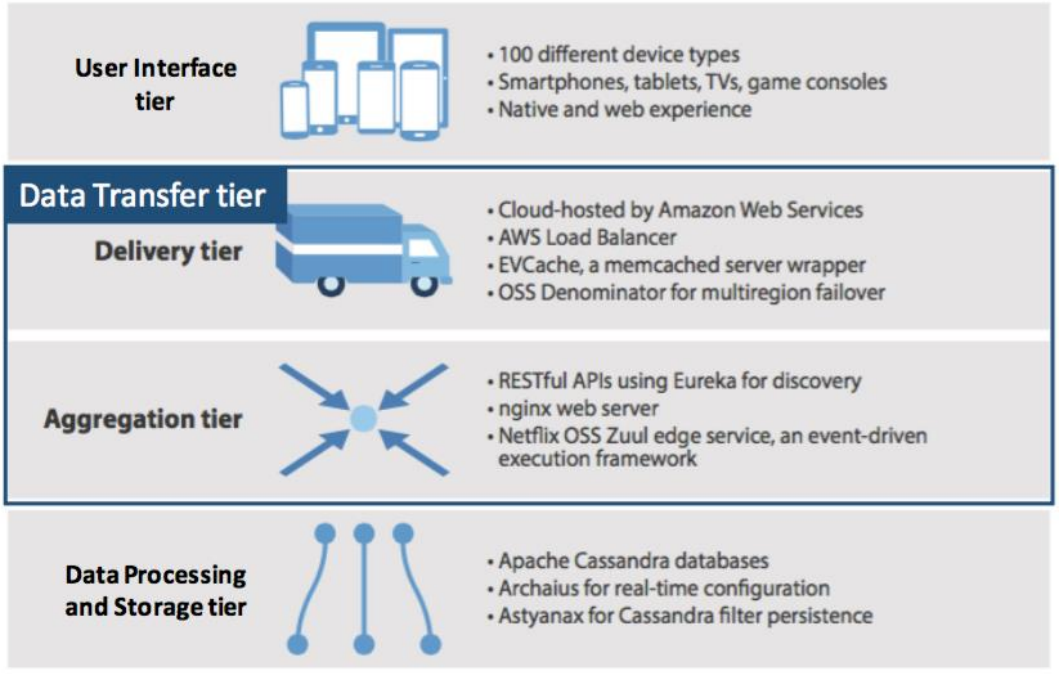


Figure 3 – Netflix architecture tiers [4]

2.4 Data Flow and System Interconnection Representation

When information is accessed through a network connection, it generates a process by which data is packaged and transmitted from a sending application through the physical wires to the receiving application. This process is represented in the following models: [6]

- OSI model – the Open Systems Interconnection reference model describes how the different software and hardware components involved in a network connection should divide labour and interact together. Defining a seven-layer set of elements, starting in the Physical layer through the Application layer. It is considered an effort to standardize networking;
- TCP/IP model – Transport Control Protocol and Internet Protocol are a part of the network standards that define Internet. TCP/IP is based on a loose approach to layering, resulting in a four-layer set of components. It aims to ensure that communications can survive any conditions and that data integrity is not compromised under malicious attacks. IP defines how devices can interact with each other over a routed and interconnected set of networks, while TCP defines how an application can create reliable channels of communication across a network.

The Figure 4 compares the layers of the two models, relating each layer with each other. As it can be seen bellow, the Network Access layer is equivalent to the combination of the Physical and Data Link layers, the Internet layer is associated to the Network layer while the Transport layer remains the same in both models. The Application layer in the TCP/IP model is equivalent to the Session, Presentation and Application layers of the OSI model all together.

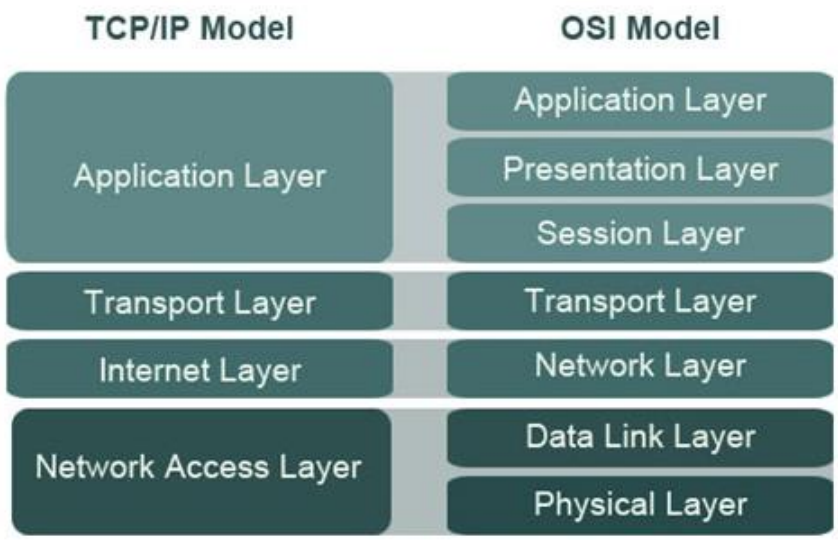


Figure 4 – TCP/IP vs OSI layers comparison [7]

Figure 5 compares both models in a most complete, oriented to protocol geography way, where it is compared both transport protocols (TCP and UDP) and several application protocols (HTTP, SSH, DNS, NTP, and others).

TCP/IP DoD Model			OSI Model
Application Layer (Services Layers 5,6,7) **PDU: Data**	HTTP: port **80** HTTPS/TLS/SSL: port **443** NNTP: port **119** FTP: port **21, 20** Telnet: port **23** SSH: port **22** POP3: port **110** IMAP4: port **143** SMTP: port **25**	DNS: port **53** TFTP: port **69** DHCP/BootP: port **67,68** SNMP: port **162, 161** NTP: port **123** Syslog: port **514**	Application Layer (7) **Scribe.** APIs, network services Serves the King/User Presentation Layer (6) **Translator.** Reformats, encrypts/de-crypts, compress/de-compress Session Layer (5) **Negotiator.** Establishes, manages and ends sessions.
Transport Layer (Host to Host Layer 4) **PDU: Segments**	TCP: protocol 6	UDP: protocol 17	Transport Layer (4) **Middle Manager.** Segment ID/Assembly
Internet Layer (Network Layer 3) **PDU: Packets**	IP	IP	Network Layer (3) **Mail Room Guy.** IP Addressing/Routing
Network Acces Layer 1 & 2 **PDU: Frame**	Ethernet, PPP Frame Relay MAC addresses, ARP	Ethernet, PPP Frame Relay MAC addresses, ARP	Data-Link Layer (2) **Envelope Stuffer.** Organizes bits into frames
Network Access Layer 1 & 2 **PDU: Bits or Data Stream**	Electrons, RF or Light	Electrons, RF or Light	Physical Layer (1) **The Truck.** Movement of bits.

Figure 5 – TCP/IP vs OSI protocol geography comparison [8]

It is also possible to locate in the OSI model the network in practice. Figure 6 defines which devices and applications (as something applied in practice) belong to each layer. As an example, the Presentation layer in the OSI model is applied in practice to the Syntax layer, which encrypts and decrypts, encodes and decodes, translates information and it uses protocols like JPEG, ASCII, TIFF, and many others.

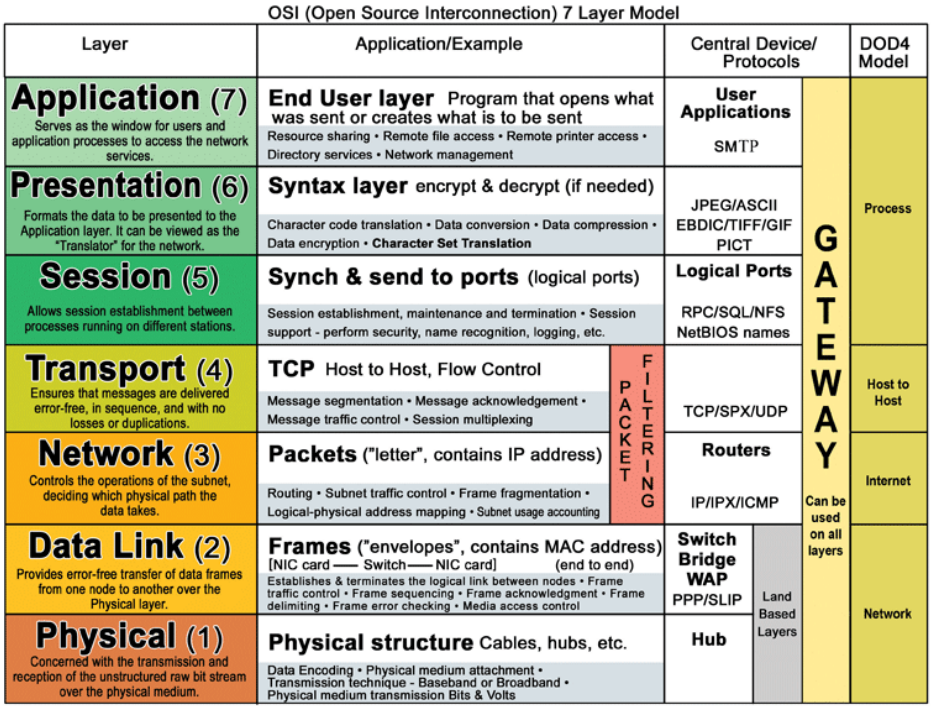


Figure 6 – OSI model applicability example [9]

2.4.1 Data Encapsulation in OSI model

The OSI model also takes part in defining the data encapsulation in a network communication. While on the Application layer it has the data that the user is interested in acquire, on the Physical layer the information is interpreted as just bits.

In order to have a performant data delivery system the data to transmit must be processed with all the information required to be delivered to the right place and then converted to bit code.

This means that the intelligence displayed at the application layer as only data will contain a piece of information regarding each layer. It is represented on Figure 7.

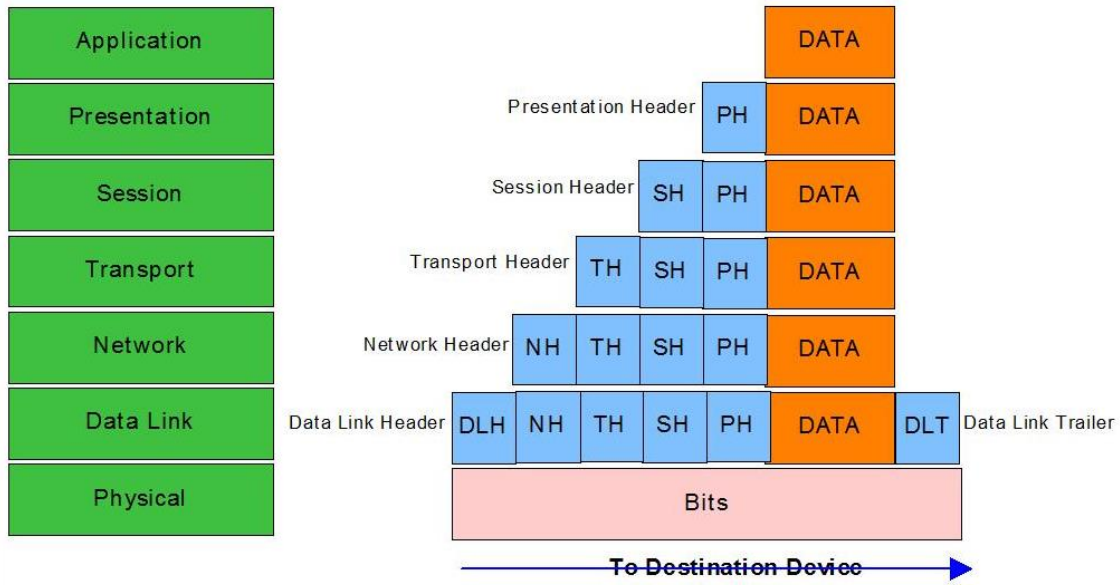


Figure 7 – OSI model data encapsulation [10]

3. Web Information Systems Components

In this document, it will be introduced and explained some building components of a Web Information System. The components that will be addressed are:

- Databases and DBMS, Database Management Systems, introducing a SQL like database;
- Web Services using the SOAP protocol and REST architecture;
- As front-end Android mobile applications and Websites.

3.1 Databases

A Database can be considered an organised collection of data of some aspect of an organisation reality. Whatever form of a database must be designed, which is the process of representing entities, attributes and relationships in a database. [11]

There are two parts that compose a database: [11]

- Intensional part – set of definitions which describe the structure of a given database;
- Extensional part – total set of data in the database.

Integrity is the problem of ensuring that the database remains an accurate representation of its organisation reality and it is ensured with the application of two integrity constraints: [11]

- Static constraint – a restriction defined on a database state;
- Transition constraint – a rule that relates different given states of a database.

3.1.1 Entities

In the database world, an entity is a uniquely identifiable element about which data is stored in a database [12]. Entities exist in four kinds:

- People;
- Things;
- Events;
- Locations.

A good example is the creation of a website for a shop, where it is necessary thinking about the type of information that one will have to handle - a seller sells products to a customer. From this situation, one can state that:

- “Shop” is a location;
- “Sales” are events;
- “Products” are things;
- “Sellers” are people;
- “Costumers” are people also. [13]

All these entities need to be included in the website database. Figure 8 represents the required entities for the example above.

Figure 8 – Entities in the database of the shop website example [14]

Each record in a table is an instance of an entity. For example, there is an entity which type is Person. Angelia Jolie is an instance of that entity.

3.1.2 Attributes

Attributes are properties of the entities and are allocated taking into account the type of properties provided by the database and the type of property that the entity needs [12]. Figure 9 represents the attributes of each entity on the database of the above example.

Figure 9 – Attributes of the database entities [15]

3.1.3 Relationships

Relationships, or Entity-Relationships, is a conceptual database feature that describes associations and interdependencies between entities.

It can be described as what do two entities have to do with each other, how they can relate [16]. Returning to the above example, simple relationships are:

- Costumers buy products;
- Likewise, products are sold to costumers;
- A sale comprises products;
- A sale takes place in a shop.

With these relationships, it is possible to achieve relationships that are more complex:

- How many costumers belong to one sale?
- How many sales belong to one costumer?
- How many sales take place in one shop?

By relating entities in simple processes like the ones referenced above it is possible to obtain critical relationships between entities that help one manipulating and maintaining a database. Relationships that relate one entity to another like: [13]

- Costumers -> Sales – one costumer can buy a product several times;
- Sales -> Costumers – one sale is always made by a costumer at the time;
- Sales -> Products – one sale can consist in a multiple products sale.

Cardinality

Cardinality identifies the maximum number of instances in a relationship that an entity participates. There are four types of cardinality: [16]

- 1:1 – one to one;
- 1:M – one to many;
- M:1 – many to one;
- M:N – many to many.

The above Costumers -> Sales example (one costumer can buy a product several times) is an example where cardinality is 1:M, while the Sales -> Costumers example (one sale is always made by a costumer at the time) is an example where cardinality is 1:1.

A relationship between Products and Shops can have cardinality being M:N, for example: Products -> Shops – many products can be sold in many shops. Figure 10 represents several relationships between entities.

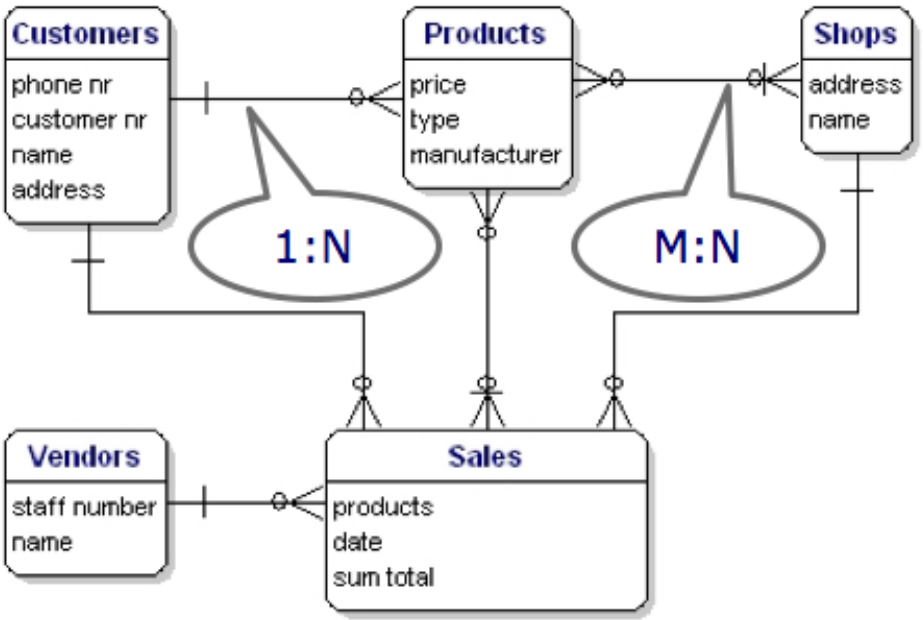


Figure 10 – Relationships between database entities with redundancy [17]

In the above Figure 10, it is possible to make the relationship Costumers -> Products through the Sales entity, as shown in Figure 11. This is called a Redundant relationship [13], and Figure 16 alterations to relationships solves the redundancy.

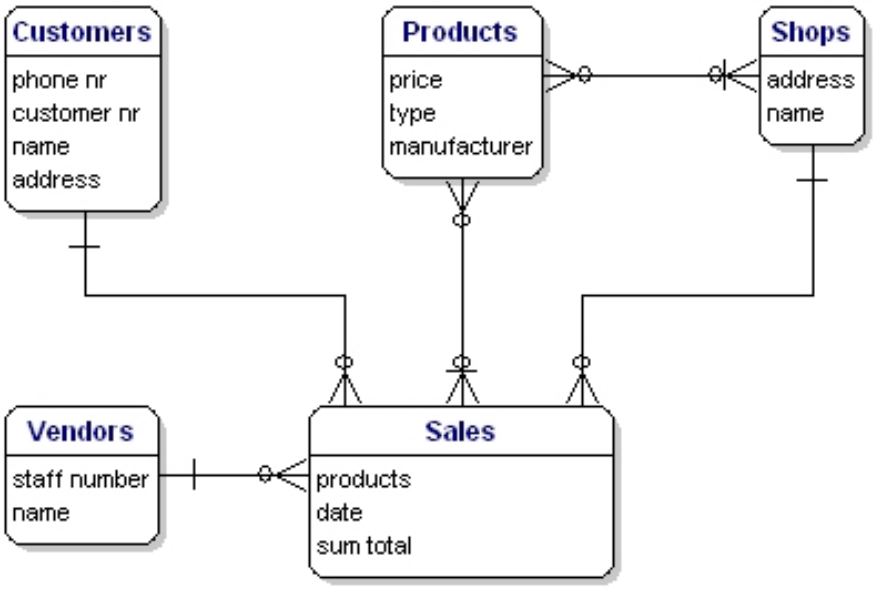


Figure 11 – Relationships between database entities without redundancy [18]

Mandatory Relationships

A Mandatory relationship is a relationship with such a dependency that for every occurrence of entity A there must exist an occurrence of entity B. For example, the relationship between the entity ID and the entity Costumers is a 1:1 Mandatory relationship, because for every ID there must be a Costumer and every Costumer must have an ID. [16]

Optional Relationships

An Optional relationship between two entities indicates that it is not necessary for every entity occurrence to participate in the relationship. In other words, an entity can exist without the one that it is related to.

For example, in the relationship Man is married to Woman, both entities may be represented in an Entity-Relationship Model because they are of interest to the organization. However, not every man, or woman, is necessarily married. [16]

Contingent Relationships

A Contingent relationship represents an association that is mandatory for one of the involved entities, but optional for the other. In other words, an instance of entity A can exist without existing an instance of the other entity involved in the relationship (entity B), but for an instance of entity B to exist there must exist an instance of entity A.

As example, the relationship Man fathers Child, where not all instances of the entity Man will result in an entity Child. However, if an instance of Child exists, it must be related to a Man instance. [16]

3.1.4 Keys

Each attribute in an entity has its own key. Attributes in an entity can be linked to attributes in other entities by adding a column for the key of the linked attribute. In a relational database, there are mainly used two types of keys: Primary keys and Foreign keys.

Primary keys

A Primary key is one or more attributes that identify uniquely an entity. A Primary key attribute should be identifiable by a 'PK' before its name.

When a Primary key consists in two or more attributes it is called a primary Composite key. [13]

Foreign keys

Foreign keys are the Primary Keys of an entity that are used as arguments in another entity in order to create a relationship between the entities. It is a good practise to identify the Foreign Key by a 'FK' before its attribute name.

The Foreign Key of an entity can also be part its Primary Key, which becomes a Primary Foreign key and it is a good practise to identify it with a 'PF' before its attribute name.

When a Foreign Key consists in two or more attributes it is called a foreign Composite key. [13]

There are seven usual situations in which to use a Foreign Key to relate entities. These situations are described in the following Table 1 [19]. (The • symbol represents the need for a mandatory relationship to one of the entities, or both).

Case	Nr. of Tables	Description
•1:1•	1	One table – it is chosen as Primary Key (PK) of the table, the PK of one of the entities
•1:1 1:1•	2	Two tables – one table for each entity, with the respective PKs. The PK of the entity without mandatory participation is added as attribute (FK) to the entity with mandatory participation
1:1	3	Three tables – one table for each entity, with the respective PKs, and one for the relationship that will have, among its attributes, the two PKs of the entities as FKs. One of the attributes is used as PK for the relationship table
1:n• •1:n• •n:1• •n:1	2	Two tables – one table for each entity, with the respective PKs. The PK of the grade 1 entity is added as attribute (FK) to the grade n entity.
1:n •1:n n:1• n:1	3	Three tables – one table for each entity, with the respective PKs, and one for the relationship that will have, among its attributes, the two PKs of the entities as FKs. The FK corresponding to the PK of the grade n entity is defined as primary Foreign Key (PF) of the relationship.
n:n •n:n n:n• •n:n•	3	Three tables – one table for each entity, with the respective PKs, and one for the relationship that will have, among its attributes, the two PKs of the entities as FKs. Both FKs corresponding to the PKs of the grade n entities are defined as primary Foreign Keys (PFs) of the relationship.

Case	Nr. of Tables	Description
Ternary	4	Four tables – one table for each entity, with the respective PKs, and one for the relationship that will have, among its attributes, the three PKs of the entities as FKs. All FKs corresponding to the PKs of the entities are defined as primary Foreign Keys (PFs) of the relationship.

Table 1 – Usual situations for relationships between entities

Assigning keys example

In Figure 12 there are assigned the correct keys to each entity of the database in the example above.

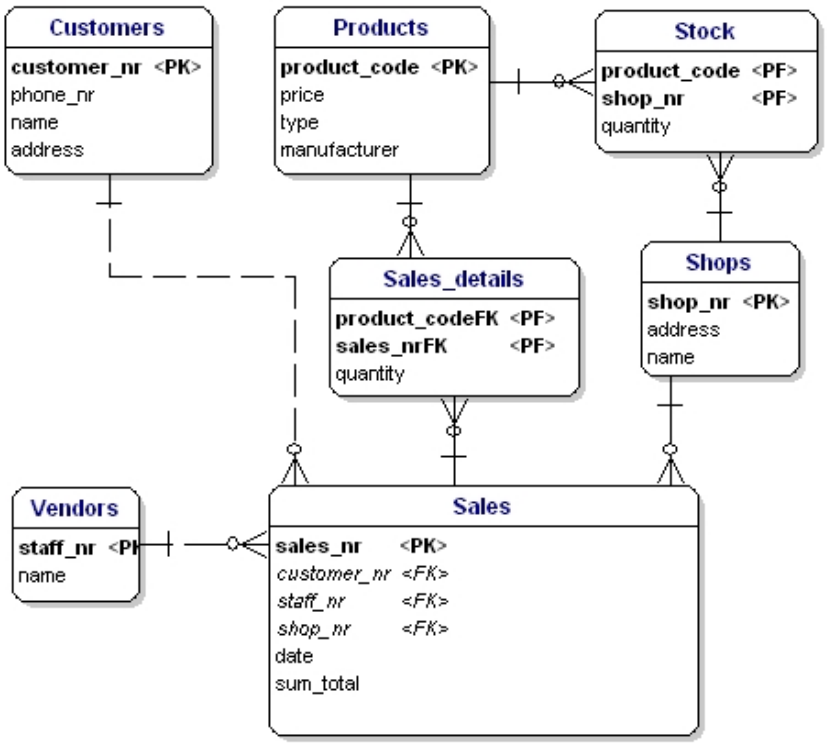


Figure 12 – Keys assigned to each entity of the example [20]

3.1.5 Assigning types example

In Figure 13 – Attribute types assigned to each entity of the example there are assigned the correct attribute types to each entity of the database in the example above.

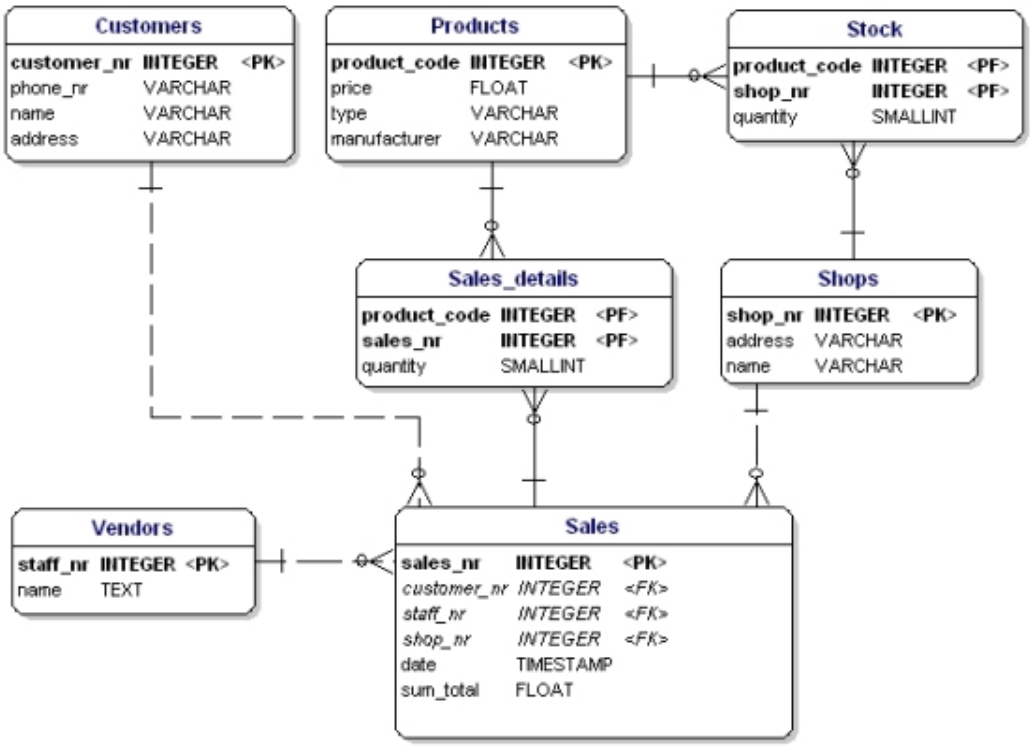


Figure 13 – Attribute types assigned to each entity of the example [21]

3.1.6 Normalization

Normalization is the process of organizing and refining relationships between entities in a relational database. There are three forms of normalization. [12]

1st Form of Normalization

The first form of normalization states that there must not be repeating groups of columns in an entity. Below in the example in Figure 14, in the left image there are three “product” attributes, which limit the number of sold products in a sale to three. In the right image there was created an additional entity with the relationship 1:M (Sales -> Sales_details) which results in unlimited products sold in a sale. [13]

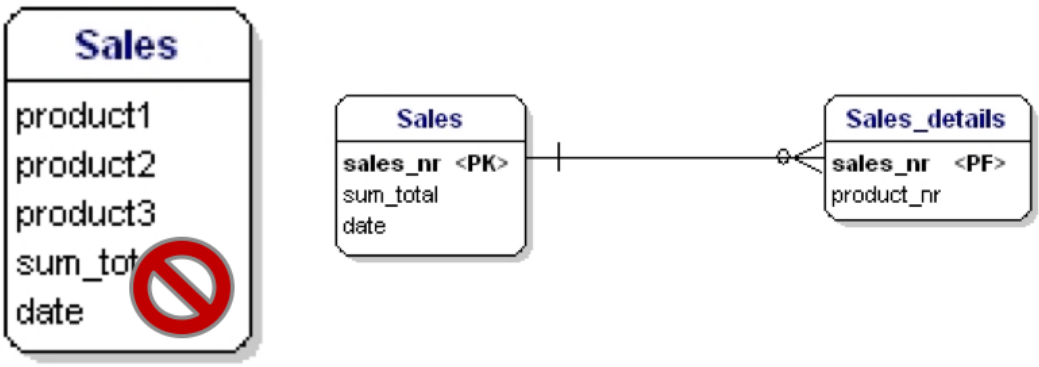


Figure 14 – Application of normalization 1st form [22] [23]

2nd Form of Normalization

The second form of normalization states that all attributes of an entity should be fully dependent on the whole Primary Key, meaning that each attribute of an entity can only be identified through the whole Primary Key. [13]

In the example of Figure 15 the change is justified because looking for the date of a sale should not require knowing the product number attribute.

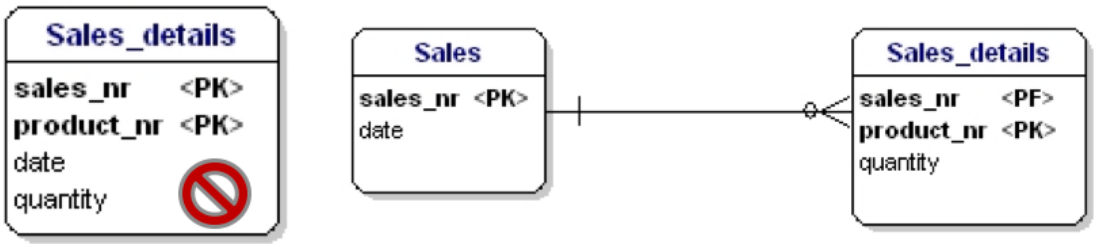


Figure 15 – Application of normalization 2nd form [24] [25]

3rd Form of Normalization

The third form of normalization states that all attributes need to be directly dependent on the Primary Key, and not on other attributes. [13]

Figure 16 reflects the application of the third normalization form.

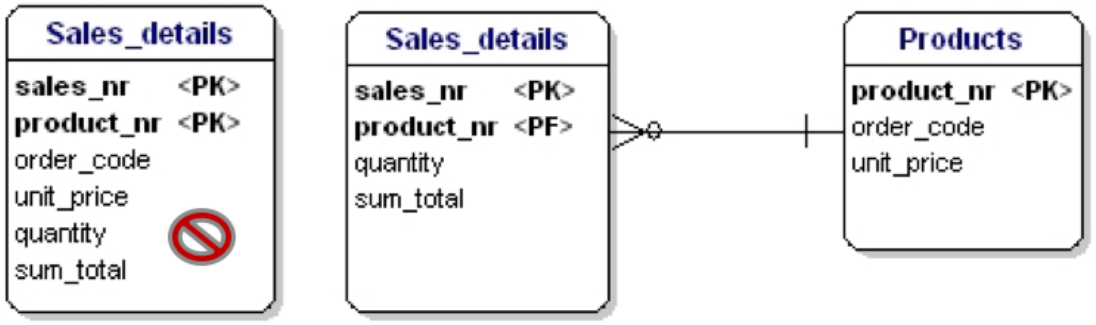


Figure 16 – Application of normalization 3rd form [26] [27]

3.1.7 Data Integrity

In a database that has integrity, there is a close correspondence between the facts stored in the database and the real world that it models. Integrity is guaranteed by applying integrity rules and constrains. In the relational model, there are two rules that are inherent to every relational database and that should be demonstrated by them. These are Referential Integrity and Entity Integrity. [11]

Entity Integrity

Entity Integrity is related with Primary Keys, and is an integrity rule that states that every table must have a Primary Key and that the column or columns chosen to be the Primary Key should be unique and not null.

A direct consequence of the entity integrity rule is that duplicate rows cannot exist in a table. [11]

Referential Integrity

Referential Integrity is related with Foreign Keys. This integrity rule states that any Foreign Key value can only be in one of two states: the usual state is that the Foreign Key value refers to a Primary Key value of some table in the database; the occasional state is that a Foreign Key value can be null (in this case, either there is no relationship between the objects represented in the database or that this relationship is unknown). [11]

3.1.8 Structure example

After applying all forms of normalization and correcting all relationships, it is possible to visualize the example database final structure in Figure 17.

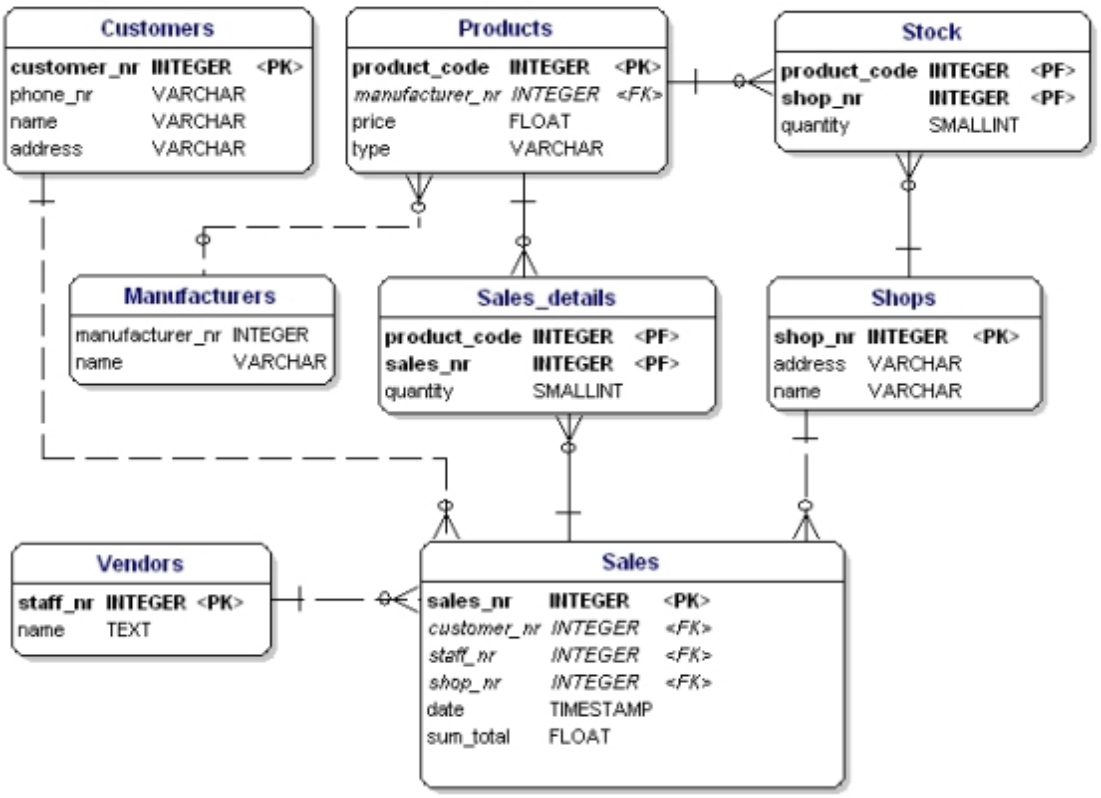


Figure 17 – Final structure of the example database [28]

3.1.9 Database Management Systems

Database Management Systems (DBMS) are software for creating and manipulating database. These systems are designed to: [29]

- Manage the access to the database;
- Define the contents of the database's data through Data Definition Language (DDL);
- Manipulate database's data through Data Manipulation Language (DML).

3.1.10 Data Definition, Manipulation, Control and Query Languages

DDL – Data Definition Language

Data Definition Language is a computer language that is used to create and modify database structure objects', including tables, schemas, among others. [30] Table 2 displays the DDL commands and their descriptions.

Command	Description
create	Creates a new table, a view of a table, or other object in the database
alter	Modifies an existing database object, such as a table
drop	Deletes an entire table, a view of a table or other object in the database

Table 2 – Data Definition Language commands and their description [31]

DDL syntax example [32]

Creation of an example database called testDB:

```
SQL> CREATE DATABASE testDB;
```

Delete of the example database testDB:

```
SQL> DROP DATABASE testDB;
```

Select the created database to be able to edit it:

```
SQL> USE testDB;
```

Create the costumers example table in the testDB database:

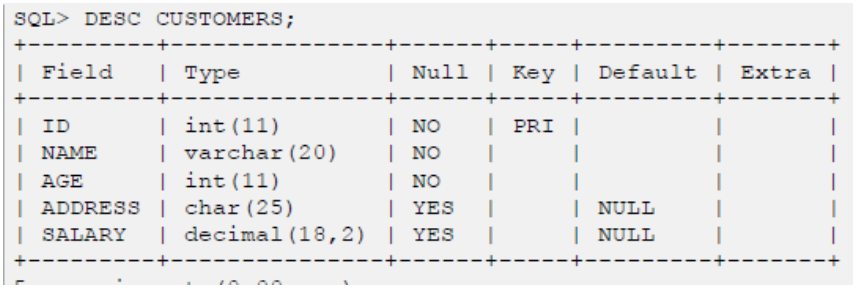
```
SQL> CREATE TABLE costumers( id INT NOT NULL, name VARCHAR (20)
NOT NULL, age INT NOT NULL, address CHAR (25) , salary DECIMAL
(18, 2), PRIMARY KEY (id) );
```

Delete the created table:

```
SQL> DROP TABLE costumers;
```

Describing a created table (Figure 18):

```
SQL> DESC costumers;
```



```
SQL> DESC CUSTOMERS;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| ID      | int(11)       | NO   | PRI |         |       |
| NAME    | varchar(20)   | NO   |     |         |       |
| AGE     | int(11)       | NO   |     |         |       |
| ADDRESS | char(25)      | YES  |     | NULL    |       |
| SALARY  | decimal(18,2) | YES  |     | NULL    |       |
+---------+---------------+------+-----+---------+-------+
```

Figure 18 – Costumers table example description [33]

DML – Data Manipulation Language

Data Manipulation Language (DML) is a computer language used to insert, delete and modify information in the database, in other words allowing users to manipulate data in it. [34] Table 3 shows DML commands and their description.

Command	Description
insert	Creates a record in a table of the database
update	Modifies a record in a table of the database
delete	Deletes a record in a table of the database

Table 3 – Data Manipulation Language commands and their description [31]

DQL – Data Query Language

Data Query Language (DQL) is a computer language used to retrieve information stored in the database. [35] Table 4 shows the DQL command a its description.

Command	Description
select	Retrieves certain records from one or more tables

Table 4 – Data Query Language commands and their description [31]

DML and DQL syntax example [32]

Insert (DML) entry into costumers table, first way:

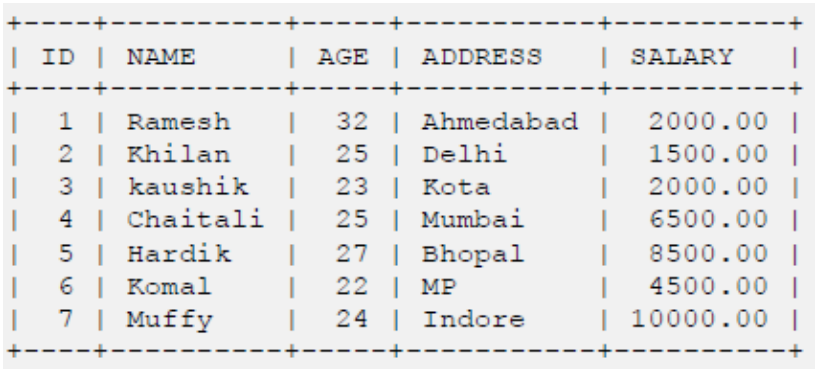
```
SQL> INSERT INTO costumers (id,name,age,address,salary) VALUES (1,
'Ramesh', 32, 'Ahmedabad', 2000.00 );
```

Insert (DML) entry into costumers table, second way:

```
SQL> INSERT INTO costumers VALUES (7, 'Muffy', 24, 'Indore',
10000.00 );
```

View (DQL) all items and fields from costumers table (Figure 19):

```
SQL> SELECT * FROM costumers;
```

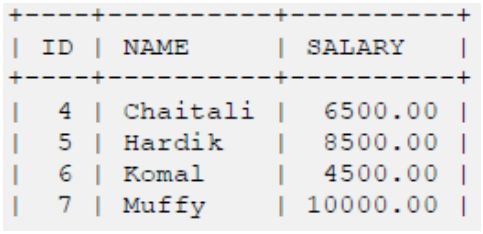


ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Figure 19 – Selection of all items and fields from costumers table [33]

View (DQL) items where salary is more than 2000 and show only id, name and salary fields from costumers table (Figure 20):

```
SQL> SELECT id, name, salary FROM costumers WHERE salary > 2000;
```

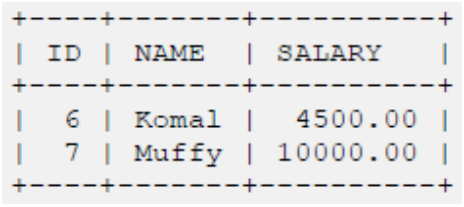


ID	NAME	SALARY
4	Chaitali	6500.00
5	Hardik	8500.00
6	Komal	4500.00
7	Muffy	10000.00

Figure 20 – Selection of items where salary is over 2000 from costumers table [33]

View (DQL) items where salary is more than 2000, age less than 25 and show only id, name and salary fields from costumers table (Figure 21):

```
SQL> SELECT id, name, salary FROM costumers WHERE salary > 2000
AND age < 25;
```



ID	NAME	SALARY
6	Komal	4500.00
7	Muffy	10000.00

Figure 21 – Selection of items where salary is over 2000 and age is less than 25 from costumers table [33]

Delete (DML) item from costumers table where id is equal to 6 (Figure 22):

```
SQL> DELETE FROM costumers WHERE id = 6;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
7	Muffy	24	Indore	10000.00

Figure 22 – Costumers table without item with id equal to 6 [33]

DCL – Data Control Language

Data Control Language (DCL) is a computer language that allows database administrators to configure security access to databases, by granting, revoking and denying privileges to database users. [36] Table 5 displays DCL commands and their description.

Command	Description
grant	Gives a privilege to user
revoke	Takes back privileges from user
deny	Prevent a user from receiving a particular privilege

Table 5 – Data Control Language commands and their description [31]

Data Dictionary

A Data Dictionary contains a list of all files in a database, the names and types of each field, and the number of records in each file. Normally it is hidden from users to prevent data damage. Although the data dictionary is used to manage the data in a database, it does not contain any actual data. [37]

3.1.11 SQL Database Language

SQL is a database computer language designed for the retrieval and management of data in relational database. It is an example of application of the above languages.

The SQL database language allows users to: [31]

- Access data in in relational database management systems;
- Describe the data in the database;
- Define and manipulate the data in the database;
- Create and drop (equivalent to delete) databases and its tables;
- Create views, procedures and functions in the database;
- Set permissions on views, procedures and functions of the database.

The SQL language is possible to embed within other languages using SQL modules, libraries and pre-compilers.

Figure 23 represents the working modules in a SQL database.

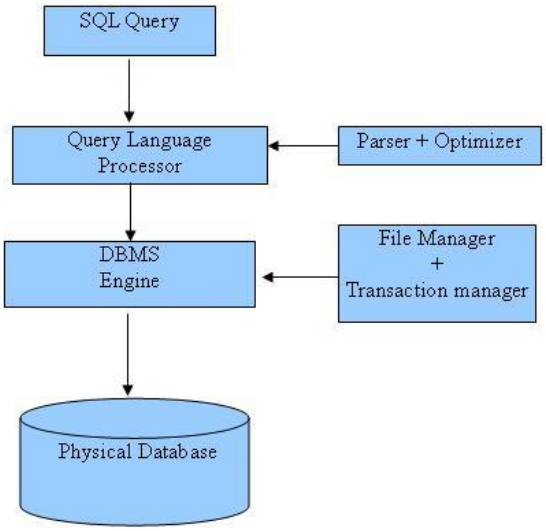


Figure 23 – Modules in a SQL database management system [38]

3.2 Web Services

Web Services are normally defined as a software system designed to support interoperable machine-to-machine interaction using a network (usually the Web) as communication medium. Web Services are located in a Web server. [39]

In other words, Web Services, consist of a group of standards intended to make it possible for diverse systems to communicate, without requiring a particular type of middleware, programming language or even operating system. It can be seen as a machine using resources and methods that reside in other machine, which requires a service request from a service consumer (client), following a service response from the service provider (server), as represented in Figure 24. [40]

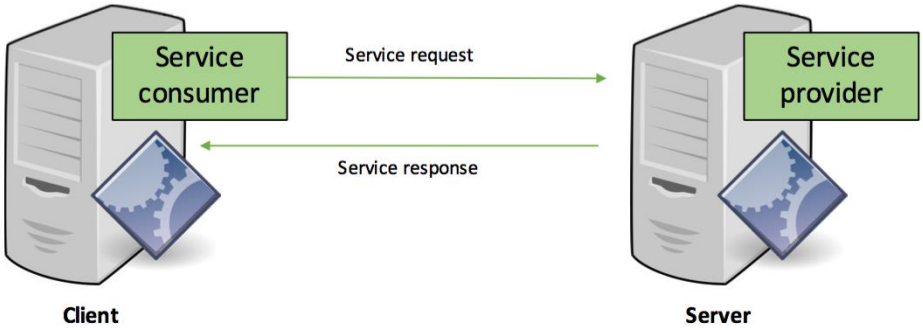


Figure 24 - Web Services machine-to-machine interaction

In Figure 25 it is represented an example of Web Services usage. Machine 2 running a .NET software holds a method that given an integer argument it returns the squared value of said argument. Also machine 2 makes this method available through the Web. Machine 1, that runs a JAVA software, wants to use the method "`squared`" that resides on machine 2. With that in mind, machine 1 sends a service request to machine 2, with the value "`4`" as argument. Machine 2 responds with the value "`16`" (the squared value of "`4`"). The client, machine 1, accessed the server, machine 2, through the Web to use its method.

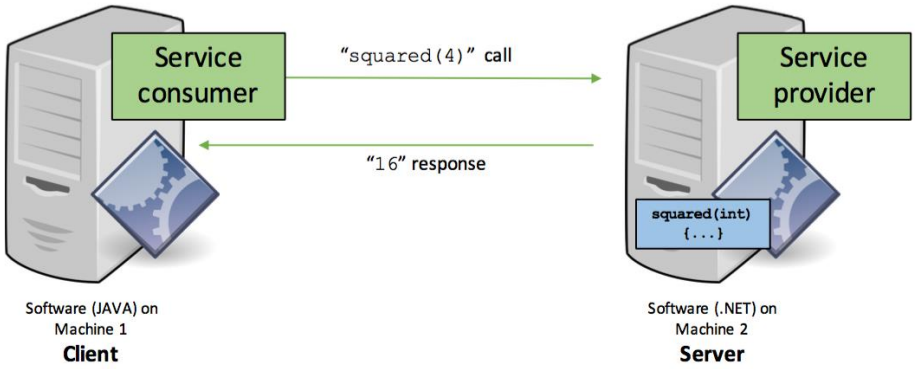


Figure 25 - Web Services machine-to-machine interaction example

In the Web Services architecture, there are three major roles that are introduced below and represent in Figure 26.

- Service provider – the **provider** of the Web Service, implements the service and makes it available in the network:
- Service requestor – any **consumer** of the Web Service that uses it by opening a network connection with the provider and sending a request;

- Service registry – a logically centralized **directory** of services, which provides a central known location where developers can publish new services and find existing ones. [41]

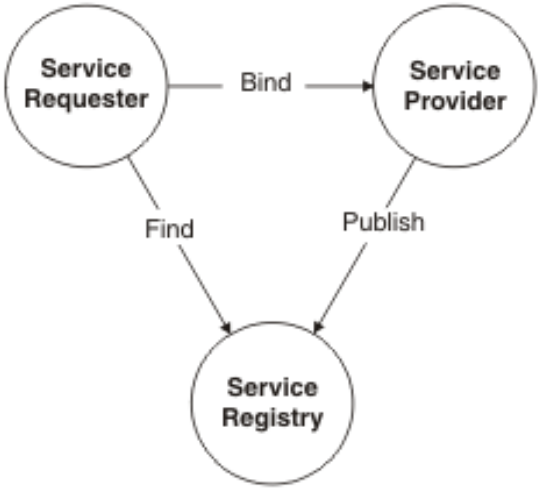


Figure 26 – Web Services architecture [42]

Figure 27 represents the main function of Web Services in a Web Information System, where it is seen several different devices accessing several different database management systems.

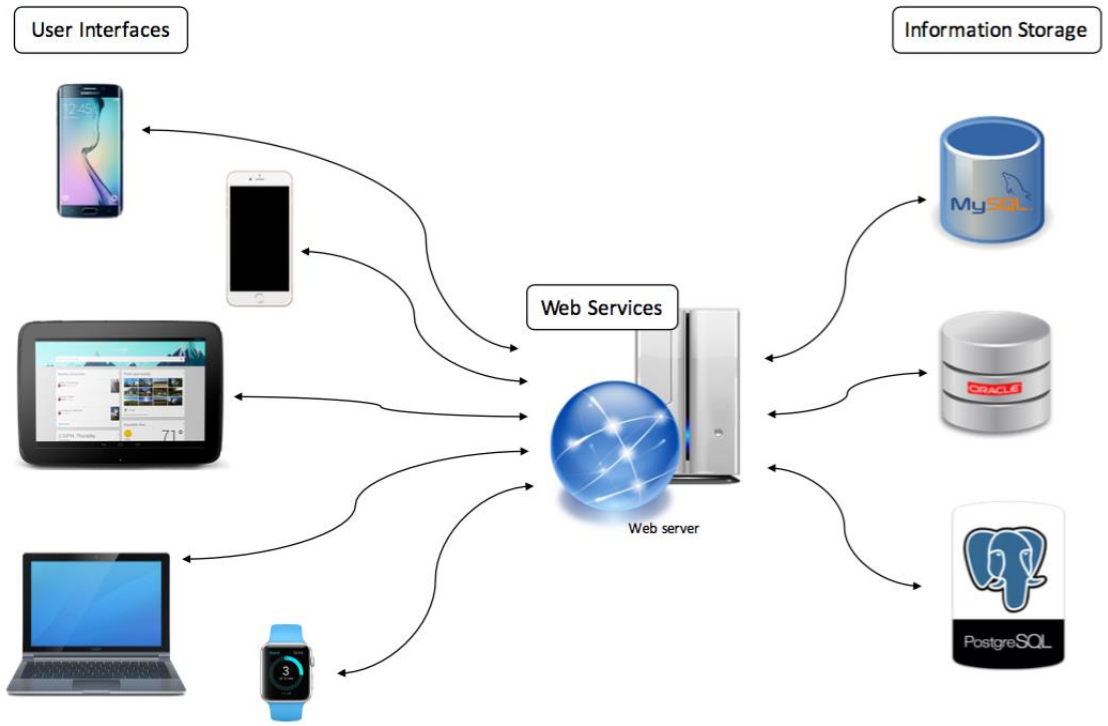


Figure 27 – Web Services representation in a Web Information System

The main characteristics of Web Services are: [43]

- Available over the Internet or private networks (Intranet);

- Use standardized messaging systems (XML, JSON);
- Programming language and operating system independent;
- Self-describing over, for example, a common XML grammar;
- Discoverable via a simple API find mechanism;
- Identified through a URI (Uniform Resource Identifier).

Also, Web Services: [44]

- Are loosely-coupled – the Web Service interface can change over time without compromising its functionality for the user (without the user knowing it);
- Are coarse-grained – several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service;
- Have the ability to be synchronous or asynchronous – synchronous clients receive their result when the service has completed, while asynchronous clients retrieve their result at a later point in time. Asynchronous capability is a key factor in enabling loosely-coupled systems;
- Support Remote Procedure Calls (RPC) – Web Services can provide RPC services of their own, or can translate an incoming invocation into an invocation of an EJB or .NET component.

Tightly-coupled: when a group of services are highly dependent on each other. [45]

Loosely-coupled: achieved by promoting single-responsibility and separation of concerns. [45]

Fine-grained: smaller components of which the larger ones are composed, lower level services. [46]

Coarse-grained: larger components than fine-grained, large sub-components. Simply wraps one or more fine-grained services together into a more discrete operation. [46]

3.2.1 XML-based SOAP Web Services

XML-based Web Services are highly interoperable applications at core level. One of the advantages of using XML is its generic way of representing not only data but also complex documents, supporting document exchange. [41]

3.2.1.1 Web Services Protocol Stack

Service transport
• This layer is responsible for **transporting** messages between applications. Currently, this layer includes hypertext transfer protocol (***HTTP***), Simple Mail Transfer Protocol (SMTP), file transfer protocol (FTP), and newer protocols, such as Blocks Extensible Exchange Protocol (***BEEP***).
XML messaging
• This layer is responsible for **encoding** messages in a common XML format so that messages can be understood at either end. Currently, this layer includes ***XML-RPC*** and ***SOAP*** (document's target).
Service description
• This layer is responsible for **describing** the public interface to a specific web service. Currently, service description is handled via the Web Service Description Language (***WSDL***).
Service discovery
• This layer is responsible for centralizing services into a common registry, and providing easy **publish/find functionality**. Currently, service discovery is handled via Universal Description, Discovery, and Integration (***UDDI***).

Table 6 – Web Services Protocol Stack

The above Table 6 introduces the four layers of the XML-based SOAP Web Services Protocol Stack. The four layers are: Service transport, XML messaging; Service Description; Service discovery. [41]

3.2.1.2 Service Transport

Web Services that use SOAP protocol are not tied to any specific transport protocol, though the most used is HTTP. In fact, SOAP can be used over FTP, SMTP, among others. One promising transport protocol is BEEP. [41]

HTTP and BEEP are introduced in Table 7.

Hyper Text Transfer Protocol (HTTP)
• Currently, HTTP is the most popular option for service transport. HTTP is simple, stable, and widely deployed. Most firewalls allow HTTP traffic. This allows XML-RPC or SOAP messages to masked as HTTP messages, enabling easily integration with remote applications, though raising a number of security concerns.
Blocks Extensible Exchange Protocol (BEEP)
•One promising alternative to HTTP is BEEP. BEEP is a recent IETF framework of best practices for building new protocols. BEEP is layered directly on TCP and includes a number of built-in features, including an initial handshake protocol, authentication, security, and error handling. Using BEEP, one can create new protocols for a variety of applications, including instant messaging, file transfer, content syndication, and network management

Table 7 – Service Transport

3.2.1.3 XML Messaging

In this document there will be approached two XML messaging protocols, XML-RPC and SOAP, introduced in the Table 8 below. [47]

XML Remote Procedure Call (XML-RPC)
• Simplest XML-based protocol to exchange messages between devices, consists in using XML messages to perform remote procedure calls; • Requests are encoded in XML and sent via a HTTP POST method; • XML responses are embebed in the HTTP response's body; • XML-RPC is platform independent and allows diverse applications to communicate, for example, a Java client can communicate with a Perl server; • The applications can be as simple as a single function call or a complex API.

Simple Object Access Protocol (SOAP)
•Simple and open standard XML-based protocol to exchange messages between devices; •SOAP is a communication protocol from communication between applications; •SOAP is a format to send XMP messages, designed to communicate via network; •SOAP is platform and language independent; •Allows to get around firewalls;

Table 8 – XML Messaging

XML-RPC Components

The XML-RPC protocol consists in three components: [47]

- XML-RPC data model – set of types to use while passing parameters, return values and faults (error messages);
- XML-RPC request structure – a HTTP POST request containing method and parameters information;
- XML-RPC – a HTTP response that contains return values or faults information;

Figure 28 illustrates the basic data types in the XML-RPC data model containing int, or i4, and double types for numbers, Boolean type for logic, string type for characters, dateTime in ISO8601 format for time and date and base64 for binary information.

Basic data types in XML-RPC		
Type	**Value**	**Examples**
int or i4	32-bit integers between -2,147,483,648 and 2,147,483,647.	<int>27<int> <i4>27<i4>
double	64-bit floating-point numbers	<double>27.31415</double> <double>-1.1465</double>
Boolean	true (1) or false (0)	<boolean>1</boolean> <boolean>0</boolean>
string	ASCII text, though many implementations support Unicode	<string>Hello</string> <string>bonkers! @</string>
dateTime.iso8601	Dates in ISO8601 format: CCYYMMDDTHH:MM:SS	<dateTime.iso8601> 20021125T02:20:04 </dateTime.iso8601> <dateTime.iso8601> 20020104T17:27:30 </dateTime.iso8601>
base64	Binary information encoded as Base 64, as defined in RFC 2045	<base64> SGVsbG8sIFdvcmxkIQ== </base64>

Figure 28 – Basic data types in XML-RPC data model

The following XML message example is used to make a request to a method named circleArea, which uses as input parameter a double value, the circle radius, and returns also a double value, the circle area. It is possible to observe that there is a field (which contains the input parameters) named methodCall.

```
POST /xmlrpc HTTP 1.0
User-Agent: myXMLRPCClient/1.0
Host: 192.168.124.2
Content-Type: text/xml
Content-Length: 169
<?xml version="1.0"?>
<methodCall>
      <methodName>circleArea</methodName>
      <params>
            <param>
                  <value><double>2.41</double></value>
            </param>
      </params>
</methodCall>
```

A XML message corresponding to the desired HTTP response of the above call is represent below. Now the methodCall field was replaced with a methodResponse field, which contains the desired output parameter.

```
HTTP/1.1 200 OK
Date: Sat, 06 Oct 2001 23:20:04 GMT
Server: Apache.1.3.12 (Unix)
Connection: close
Content-Type: text/xml
Content-Length: 124
<?xml version="1.0"?>
<methodResponse>
      <params>
            <param>
                  <value><double>18.24668429131</double></value>
            </param>
      </params>
</methodResponse>
```

It is also possible to obtain a fault message, in case of anything went wrong. The following XML message represents a fault message stating that the circleArea method does not exist. The params field present in the above sample was replaced with the fault field.

```
<?xml version="1.0"?>
<methodResponse>
      <fault>
```

```
                <value><string>No such method!</string></value>
            </fault>
        </methodResponse>
```

SOAP Elements

A SOAP message is a normal XML document containing four elements, displayed above. [47]

- Envelope (mandatory) – defines the start and end of the message;
- Header (optional) – contains any optional message attributes to be used when processing the message, either at an intermediary point or ate the end point;
- Body (mandatory) – contains the XML message data which is being sent;
- Fault (optional) – provides information regarding the occurrence of errors while processing the message.

Figure 29 represents a BabelFish method request, which translates the input into French, where the input parameter is the string “Hello, world!”. It is possible to observe that both Envelope and Body elements are present.

```
POST /perl/soaplite.cgi HTTP/1.0
Host: services.xmethods.com
Content-Type: text/xml; charset=utf-8
Content-Length: 538
SOAPAction: "urn:xmethodsBabelFish#BabelFish"

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:BabelFish
xmlns:ns1="urn:xmethodsBabelFish"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<translationmode xsi:type="xsd:string">en fr</translationmode>
<sourcedata xsi:type="xsd:string">Hello, world!</sourcedata>
</ns1:BabelFish>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 29 – SOAP message to send a request to BabelFish method

The SOAP response to the above request is displayed in Figure 30. The method request was successful and the method responded with the string “Bonjour, monde!” as output. One has to be aware that, once again, both Envelope and Body elements are present.

```
HTTP/1.1 200 OK
Date: Sat, 09 Jun 2001 15:01:55 GMT
Server: Apache/1.3.14 (Unix) tomcat/1.0 PHP/4.0.1pl2
SOAPServer: SOAP::Lite/Perl/0.50
Cache-Control: s-maxage=60, proxy-revalidate
Content-Length: 539
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<namesp1:BabelFishResponse xmlns:namesp1="urn:xmethodsBabelFish">
<return xsi:type="xsd:string">Bonjour, monde!</return>
</namesp1:BabelFishResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 30 – SOAP message with BabelFish method response

A SOAP Web Service request and response is represented in Figure 31 below.

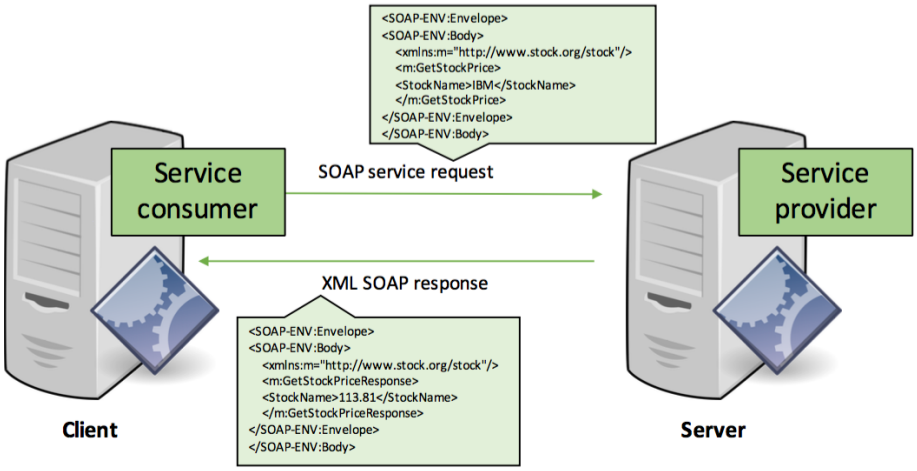


Figure 31 - SOAP Web Services machine-to-machine interaction

3.2.1.4 Service Description

Web Services Description Language (WSDL)
• WSDL is the standard format for describing a Web Service in XML format; • XML-based language for describing Web Services - how to access them and what actions it will perform; • Protocol for information exchange in decentralized and distributed environments; • WSDL is an integral part of UDDI, an XML-based worldwide business registry; • Developed jointly by Microsoft and IBM; • WSDL is pronounced as "wiz-dull"; • WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special data types used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

Table 9 – Service Description

The Table 9 above describes what is WSDL.

3.2.1.5 Service Discovery

Universal Description, Discovery and Integration (UDDI)

• UDDI is an XML-based standard for describing, publishing, and finding Web Services;
• UDDI is a specification for a distributed registry of Web Services;
• Protocol for information exchange in decentralized and distributed environments;
• UDDI is platform independent, open framework;
• Can communicate via SOAP, CORBA, Java RMI Protocol;
• Uses WSDL to describe interfaces to Web Services;
• UDDI is seen with SOAP and WSDL as one of the three foundation standards of Web Services;
• UDDI is an open industry initiative enabling businesses to discover each other and define how they interact over the Internet;
• UDDI has two parts:
  • A registry of all a web service's metadata including a pointer to the WSDL description of a service;
  • A set of WSDL port type definitions for manipulating and searching that registry. [41]

Table 10 – Service Discovery - UDDI

There are three types of pages in the UDDI standard: White pages, Yellow pages and Green pages. The above Table 10 describes the UDDI standard and the below Table 11 describes the UDDI types of pages.

White pages
• Basic information about companies and their business; • Basic contact information including business name, address, phone number, among others; • A unique identifier for company tax ID, allowing others to discover company's Web Service based upon its business identification.
Yellow pages
• More detailed company information. Includes descriptions of the kind of electronic capabilities the company can offer to interested parties; • It uses commonly accepted industrial categorization schemes, industry codes, product codes, business identification codes, making it easier for companies to search through the listings and find exactly what they want.
Green pages

• Contains technical information about a Web Service. This is what allows someone to bind to a Web Service after it has been found;
• Includes:
  • Web Services various interfaces;
  • URL locations;
  • Discovery information and similar data required to find and run the Web Service.

Table 11 – UDDI types of pages

UDDI is composed by three elements, which are: [47]

- UDDI data model – a XML Schema with the purpose of finding businesses and Web Services;
- UDDI API specification – specification of the API to search and publish UDDI data;
- UDDI cloud services – provide implementations of the UDDI specification and synchronize all data on a scheduled basis.

Figure 32 represents the UDDI technical architecture. The UDDI Business Registry (UBR), also known as the Public Cloud, is a conceptually single system built from multiple nodes that has their data synchronized through replication.

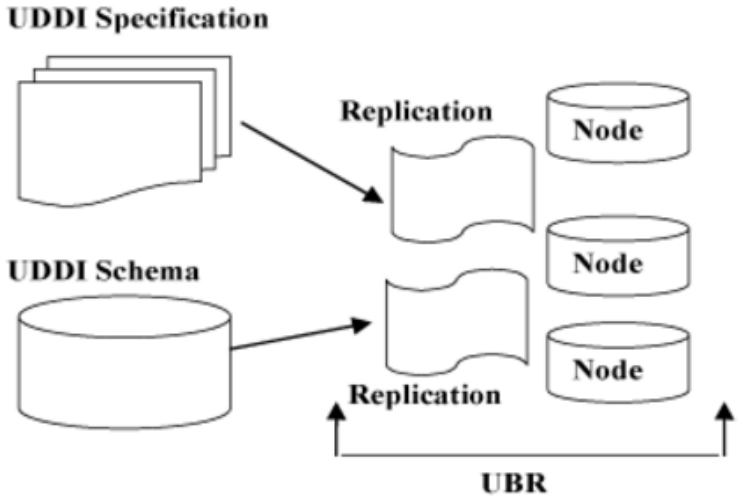


Figure 32 – UDDI technical architecture [48]

3.2.1.6 Web Services Components Example

Figure 33 represents the components of a possible Web Services implementation. It uses SOAP protocol, WSDL language and UDDI standard.

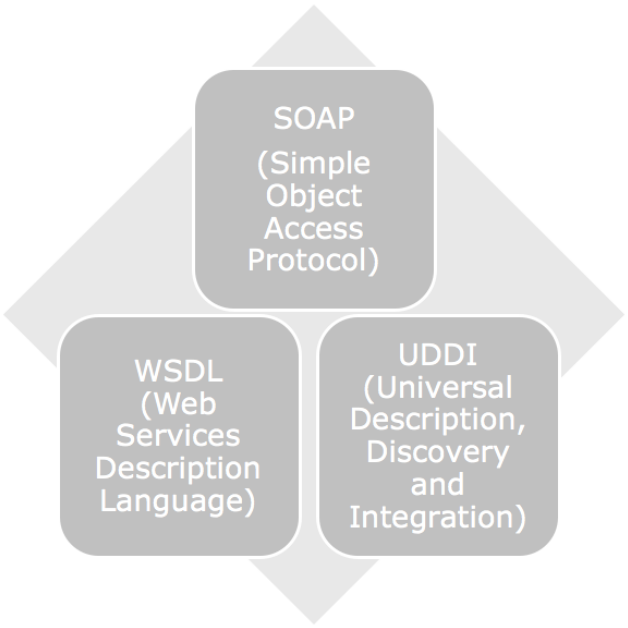


Figure 33 – Web Services implementation example components

3.2.2 RESTful Web Services

Web Services based on a REST architecture have been gaining more and more acceptance in the Web development community across recent years rather than the Web Services that use SOAP and WSDL. A proof to that crescent acceptance is the change in interface design form several mainstream service providers, such like Google and Facebook. This change happens mainly because RESTful Web Services are easier to use and resource oriented.

Web Services based on a REST, or RESTful Web Services, use HTTP methods to implement the concept of REST architecture. These Web Services normally define: [49]

- A URI (Uniform Resource Identifier), like a path to the service;
- A service;
- A resource representation (using JSON or XML languages);
- A set of HTTP methods.

3.2.2.1 REST Definition

REST stands for Representational State Transfer and is an architecture based on Web standards that uses HTTP protocol for data communication. Its principle resides in every component being a resource, which is accessed by a common HTTP method and described by an URI. REST can use various representations to represent resources, such as text, XML or JSON. [50]

In this architecture, the REST server simply provides access to resources, while the REST client accesses and presents them. [50]

3.2.2.2 HTTP Methods

The usual HTTP methods used in RESTful Web Services are: [49]

- GET – provides a read only access to a resource;
- PUT – used to create a new resource;
- POST – used to create a new resource or to update and existing resource, or both;
- DELETE – used to remove a resource.

A RESTful Web Service request and response is represented in Figure 34 below.

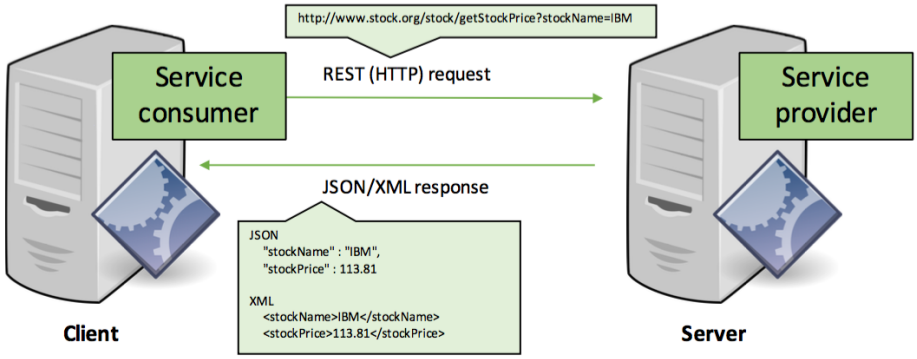


Figure 34 - RESTful Web Services machine-to-machine interaction

3.3 Android applications

Android applications are a possible solution to implement a User Interface. The Android applications development is accessible to any interested party without much of an invest and providing a great developers community with many open source libraries and free components.

The Android system, developed by the Open Handset Alliance, which belongs to Google, is built above a Linux kernel. It includes a set of programming languages libraries, the Android Runtime tool, the Application Framework and a set of inherent applications [51]. The Android architecture is represented in Figure 35.

Android is built above an optimize Java Virtual Machine, supports several communication technologies like Bluetooth, Wi-Fi, 2G, 3G, 4G, NFC, among others, supports interaction with several components and sensors like GPS, camera, proximity, accelerometer, and so on. [51]

Android devices represent an average of 80% of the total number of active smart mobile devices (smartphone and tablets mostly) in the world in 2015.

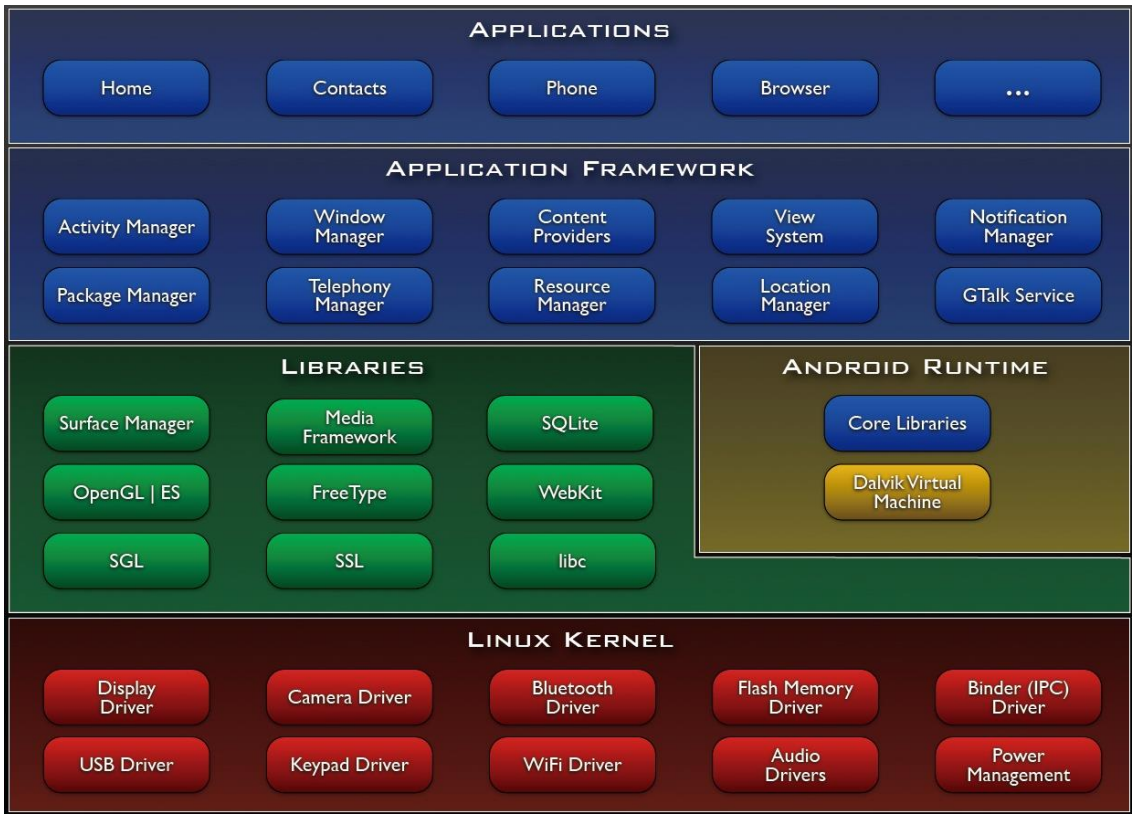


Figure 35 – Android system architecture [52]

Applications

They can be defined as software that allow the user to perform a specific task, corresponding to the mobile device actual applicability. Applications can be tools, games, maps, among others.

An Android application is developed using Java programming language and Google provides its on development open source tool, the Android Studio. Android Studio is the official Android Integrated Development Environment (IDE). It consists in a software program that provides comprehensive facilities to develop Android applications, including a code editor, an automatic compiler and a debugger, among other utilities. [51]

Application Framework

Consists in the application structure, to which developers have access, and its designed in a way to simplify and boost the utilities/components reutilization, as it allows applications to use the capabilities of other applications (though limited to restrictions imposed by its own structure). In Figure 33, under the Application Framework layer, there are represented several system services to which developers can access and use in their applications. [51]

Android Runtime

Basically, Android Runtime translates Java code to an encoding in bytes, byte-code, so the high-level programming is feasible at low-level. It consists in a set of core libraries and a virtual machine. Core libraries provide most of the functionality available in the core libraries of the Java programming language.

In conventional computing devices, run software directly on the kernel of the operating system, but an Android application runs in its own process, in its instance of the Dalvik Virtual Machine. This virtual machine is designed so that devices are able to run several instances of it in an efficient manner. Although their very similar, it is not the typical Java Virtual Machine (JVM), and it runs files on its own format, Dalvik Executable (.dex), which is optimized for the lowest possible impression in system memory. This virtual machine is resource-based and runs classes compiled by the Java compiler that have been transformed in. dex file with an included tool.

The Dalvik VM relies on the Linux kernel for the underlying functionality such as threading and low-level memory management. [51]

Linux kernel

In the Android environment base there is a simple Linux kernel, or at least simpler than for other operating systems based on this kernel. Fundamentally, this makes the device hardware connection to its software, being responsible for lower-level task, which means that Android work itself is mostly done in the layers that are above the kernel. [53]

3.3.1 Connection to the Database

Being the User Interface, most Android applications need to access the database to obtain a several number of data and information that are required for its proper performance. The access to the server database via an Android application is not made directly because of security, flexibility and transparency reasons. It is needed a middleware solution, like a Web Service (approached in the previous section of this document). For example, the Web Service can reside in an Apache server and use PHP as a programming language.

In a normal usage case, when an application needs to access the database, it is made a request (for example a SQL Query) through the Web Service to the database, and a response is obtained (which can be a JSON file containing the wanted information). Figure 36 represents a simple request and response handled by the middleware solution, granting the access from the mobile device to the server's database.

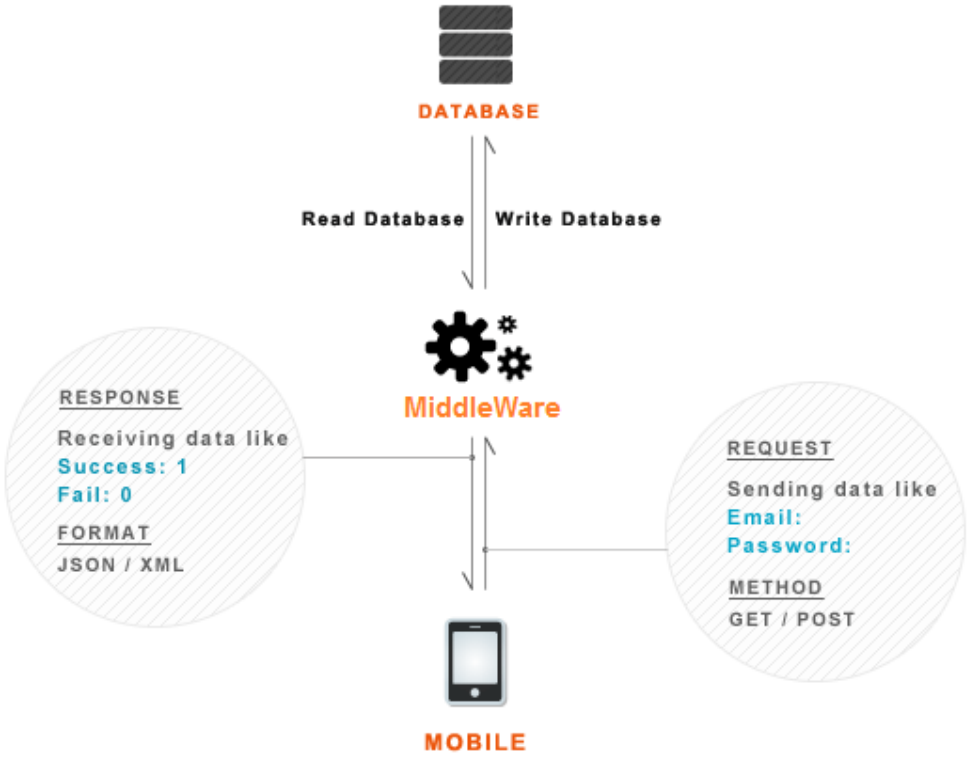


Figure 36 – Database access through middleware solution

3.3.2 Application Components

Android application components constitutes the essential part of the application itself, being its basis. These components, and how they interact, are described in a file called AndroidManifest.xml, that exists in every application.

The following Table 12 introduces the most frequently used components.

Component	Description
Activity	An Activity describes and defines the User Interface (UI) and handles the user interaction with the device.
Fragment	A Fragment represents a portion of the UI inside an activity.
View	A View corresponds to every UI element that is drawn in the device screen.

Component	Description
Layout	A Layout, or several layouts, correspond to the hierarchies that encapsulate and control the appearance and format of every view.
Intent	An Intent is simply a message that connects different components.
Resource	A Resource is an external element such as colours, dimensions, strings or figures.
Manifest	The Manifest is the configuration file for the application
Service	A Service manages background processing related with the application.
Broadcast Receiver	A Broadcast Receiver manages communication between the application and the Android operative system.
Content Provider	A Content Provider handles received data and manages it.

Table 12 – Application Components

The principal component of concern of an Android developer is the Activity as it mainly consists in the usage of the application. An application is composed from several activities, usually one activity for one “screen”, and each of this activity is responsible to perform and handle actions. In the manifest file it is necessary to specify which activity is launched when the application is started. [54]

Table 13 describes the moment when the methods are called.

Method	Description
onCreate()	The method onCreate() is the first to be called when the activity is summed. Results in the creation of the activity.
onStart()	The method onStart() is called when the activity becomes visible in the device screen.
onResume()	The method onResume() is called just before the user starts interacting with the activity.
onPause()	The method onPause() is called when the system is about to start resuming another activity.
onStop()	The method onStop() is called when the activity becomes invisible in the device screen.
onDestroy()	The method onDestroy() is called to finish the activity, being necessary to re-create the activity when it opens.
onRestart()	The method onRestart() is called when the intention is to make visible an activity in stopped state.

Table 13 – Activity Methods

When an activity is summed there are several methods that are called in a pre-determined order. The same happens when an activity is closed. This sequence of methods calling is name Activity Lifecycle and it is represented in Figure 37. These methods are described in the table below. [55] [56]

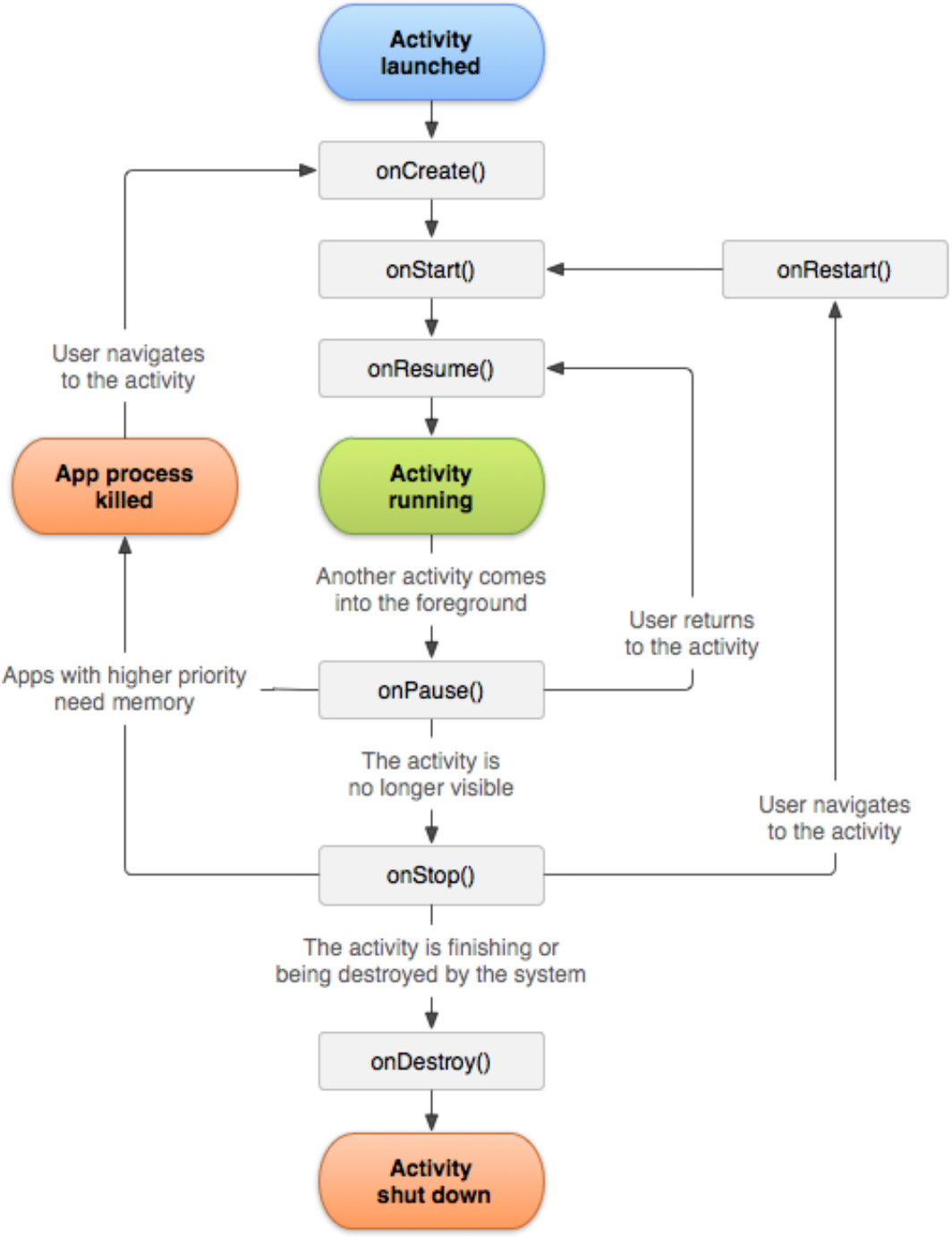


Figure 37 – Android activity lifecycle [57]

3.3.3 User Interface Element Types

3.3.3.1 UI Layout Types

A Layout element is the element in which every other element of the User Interface is drawn into, consisting in a view group (because it has several views inside). There are different types of layout elements with different drawing restrictions and specific for certain designs. [58]

A layout file defines the visual structure for a User Interface, for example the UI for an activity. [59]

Below there is Table 14 with the description of the mostly used layout element types.

Layout type	Description
LinearLayout	A LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally, where it is possible to specify the layout direction. [59]
RelativeLayout	A RelativeLayout is a view group that displays child views in relative positions, where the position of each view can be specified as relative to sibling elements or in positions relative to the parent view. [60]
ListView	A ListView is a view group that displays a list of scrollable items, that are automatically inserted to the list using an Adapter that pulls content from a source and converts each item result into a view that is placed into the list. [61]
GridView	A GridView is a view group that displays items in a two-dimensional, scrollable grid, where the grid items are automatically inserted to the layout using an Adapter, like the one in the ListView. [62]

Table 14 – Main UI Layout types

3.3.3.2 UI Control Types

The UI Controls are the interactive elements in the User Interface of the application, i.e. user makes use of them to perform actions when using the application.

To add a new control to the application UI one can just add it in the layout file.

Some of the most used UI Control types are described in the Table 15 below, and are displayed in Figure 38. [63] [64]

Control type	Description
TextView	A TextView is a control type that displays text to the user.
EditText	A EditText (.B) is a sub-class of TextView that allows the user to input text.
Button	A Button (.A) is no more than a push-button that can be pressed, or clicked, by the user to perform an action. It can also be used an ImageButton.
CheckBox	A CheckBox (.C) is an on/off switch that can be toggled by the user, and it should be used when presenting users with a group of selectable options that are not mutually exclusive. Has two states: checked or unchecked.
RadioButton	A RadioButton (.D) is similar to the CheckBox control type, but only one option of the group can be selected. Also has two states: checked or unchecked.
RadioGroup	A RadioGroup (.D) is used to group together one or more RadioButton controls.
ToggleButton	A ToggleButton (.E) is an on/off button with a light indicator.
Spinner	A Spinner (.F) is a drop-down list that allows the user to select one of its items.
Picker	A Picker (.G) is a dialog for users to select a single value for a set by using up/down buttons or via a swipe gesture, normally associated to Date or Time.

Table 15 – Principal UI Control types

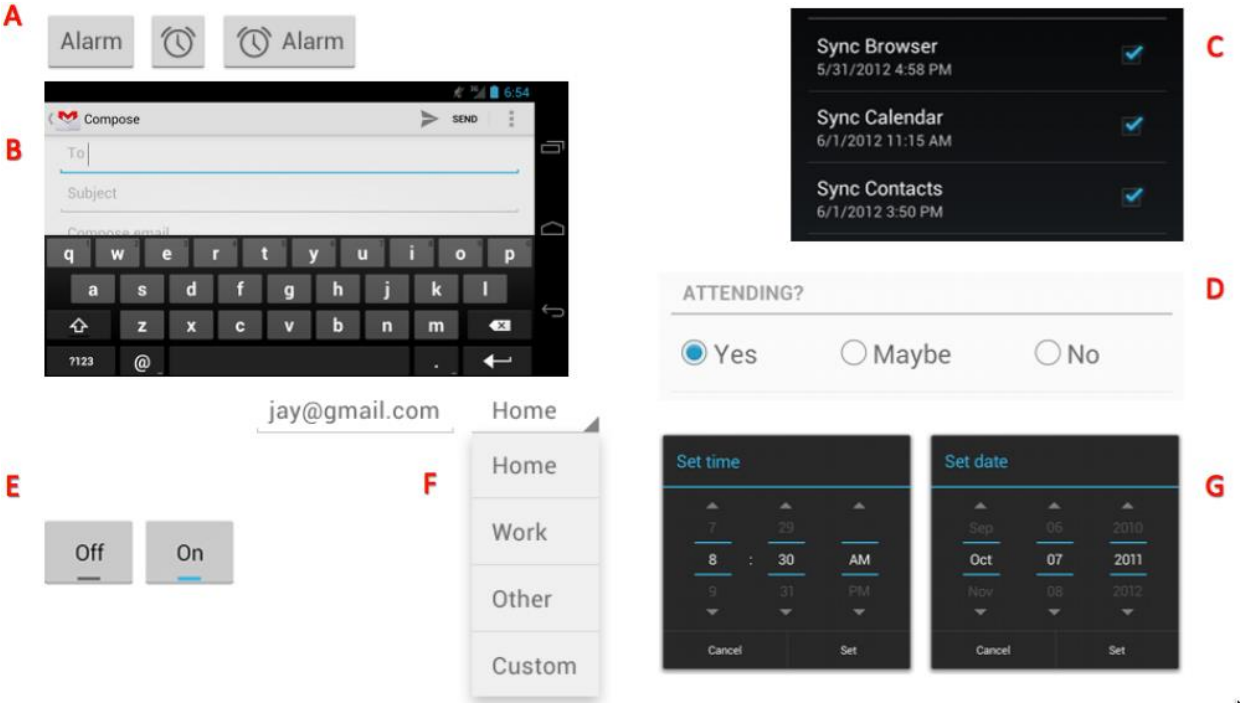


Figure 38 – Android UI Control types [63]

3.4 Websites and Web Applications Example

Websites and Web applications are a much common User Interface, that normally are accessed via a Web browser, which allows the user to access it in almost every existing device that has a browser client.

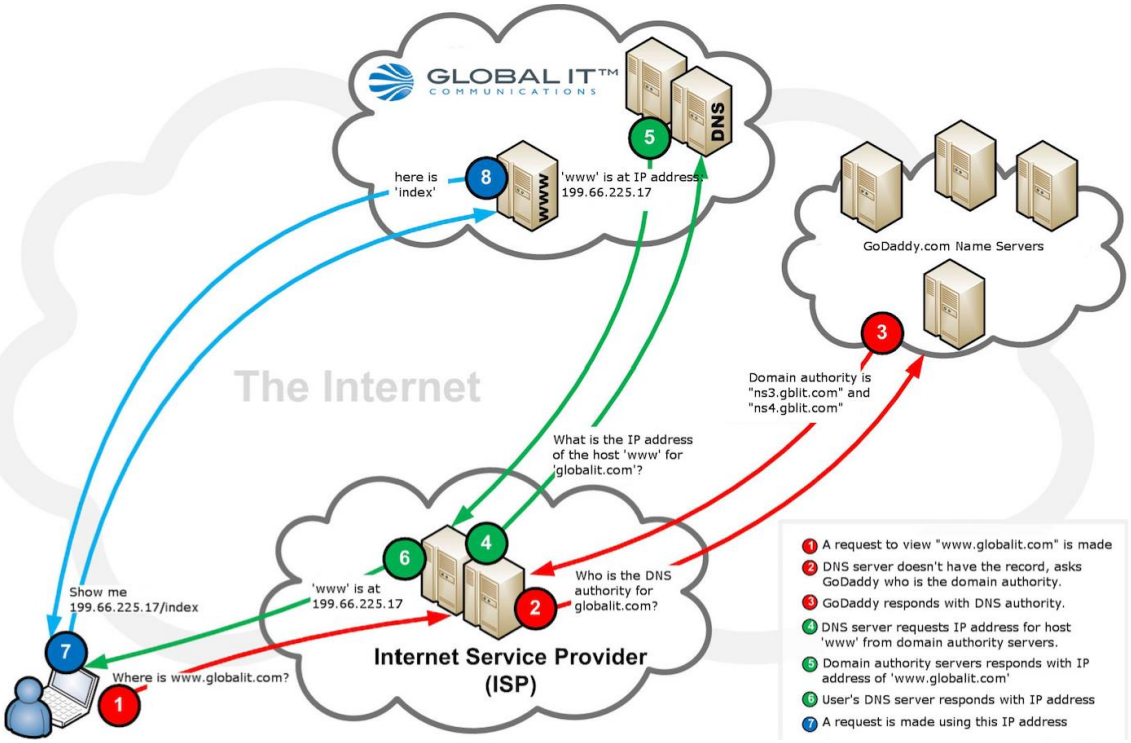


Figure 39 – DNS query diagram [65]

Figure 39 represents a DNS query diagram, which corresponds to the process between client and server when the client intends to access a website in a browser. The numbers in the figure represent actions and are described below:

1. A request to view www.globalit.com is made;
2. DNS server doesn't have the record, asks GoDaddy who is the domain authority;
3. GoDaddy responds with DNS authority;
4. DNS server requests IP address for host 'www' from domain authority servers;
5. Domain authority servers responds with IP address of www.globalit.com;
6. User's DNS server responds with IP address;
7. A request is made using this IP address;
8. The server at 199.66.225.17 responds to the request.

In Figure 40 there is a page running a script of PHP embedded in HTML code that displays a form in where two numbers are inserted in order to be summed, and the sum result displayed after submitting. The two numbers are processed by a script that runs in the server.

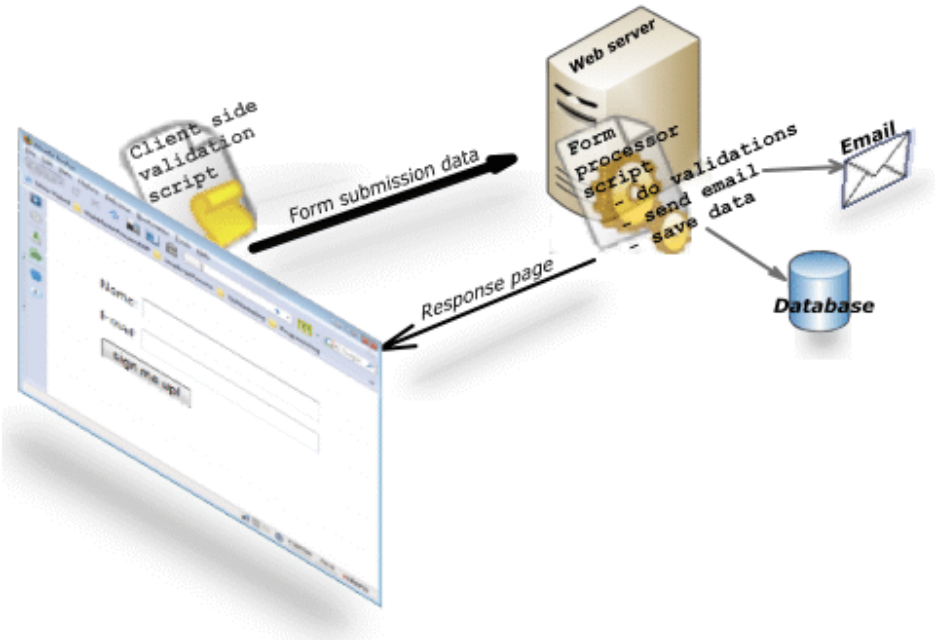


Figure 40 – PHP calculation example [66]

An “index.php” script will be run when the submit button is clicked and also be the return page.

The browser uses HTTP to request the page, by default, “index.php” from the Web server. Server runs the PHP script on the page and sends the final result with the HTML.

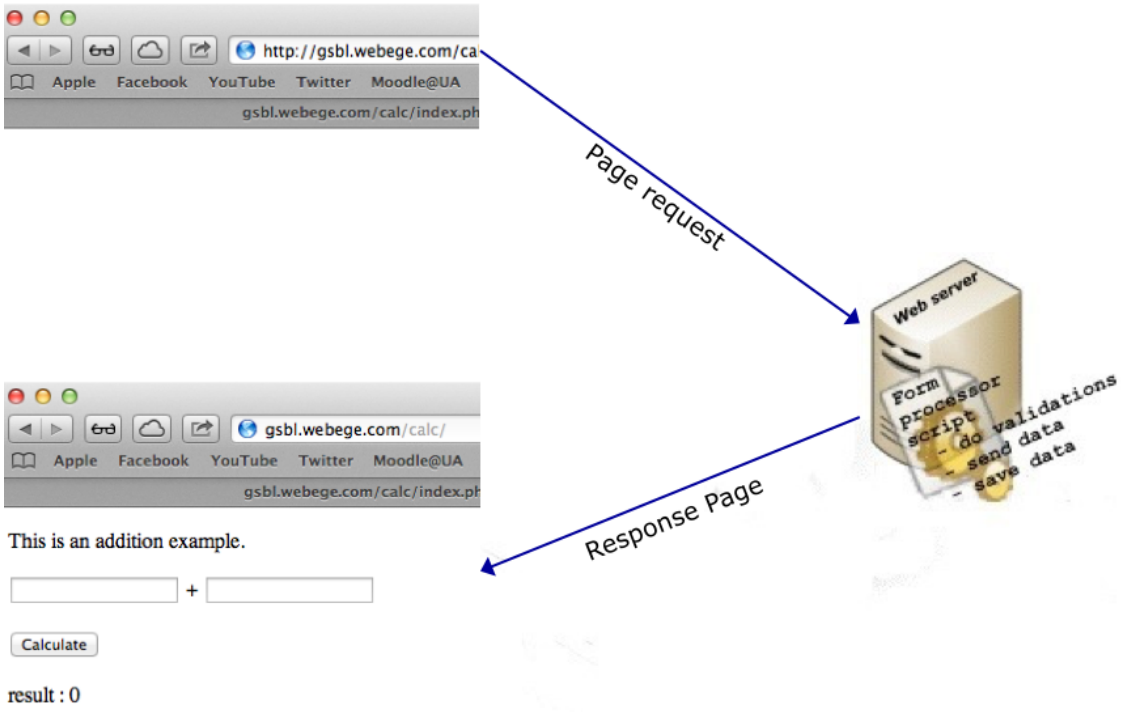


Figure 41 – PHP calculation example page access

The process described above is represented in Figure 41 where no user data was inputted at time the PHP script was run, therefore the result at the output was zero.

Figure 42 shows that after the user inputs data in the form and the data is sent to the server, the server returns a page as HTML tag, which contains the result and also the HTML code for the browser to draw the whole page again.

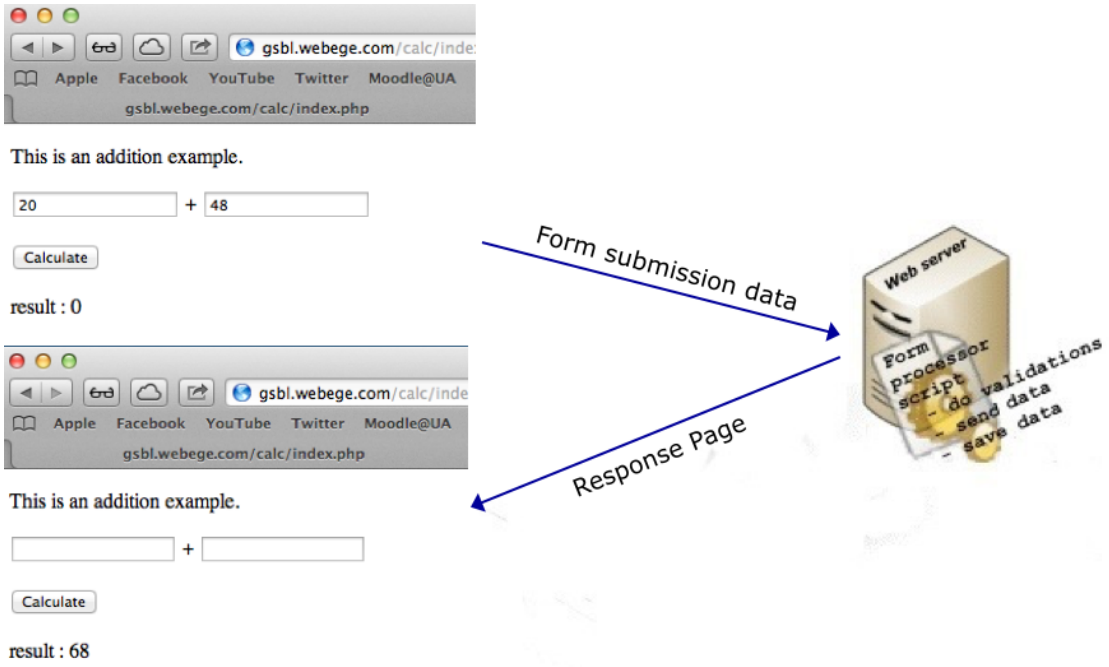


Figure 42 – PHP calculation example result returned

3.4.1 HTML, CSS and JavaScript

In order to properly develop a website, a Web developer should be comfortable using the following three languages:

- HTML to build the content structure of the website pages;
- CSS to style and specify the layout of the website;
- JavaScript to give dynamic properties to the website.

These languages allow the developer to build the static structure of the website, style it and give it dynamic capabilities. [67]

Hyper Text Markup Language

HTML stands for Hyper Text Markup Language and is a markup language for describing Web pages. A HTML file that describes a Web page is called a document.

HTML is defined as a markup language because HTML documents are described by HTML tags, and a markup language is a set of markup tags. Markups mark a certain type of text. Each HTML tag defines different page content.

HTML code is built in a document, will the page that it creates is viewed in a Web browser. The Web browser translates the text in the HTML document into a visible form. [68]

HTML tags are keywords surrounded by angle brackets that contain different Web page content. Normally HTML tags come in pairs, where the first tag of the pair is the start tag and the second tag is the end tag. HTML tags are used like in the example below:

```
<tagname>This is an HTML Tag!</tagname>
```

The end tag is similar to the start tag but with a slash after the first angle bracket. HTML tags are what separate normal text from HTML code. Different tags will perform different actions. While not appearing themselves in the page visualization using a browser, the effects of the tags are visible. The following example uses a tag that displays text in bold.

```
<b>This text will be bold.</b>This text will be normal.
```

HTML is a powerful and easy to learn language, but it can be powered by other languages. Cascading Style Sheets are used to control the website presentation, while JavaScript is used to provided effects and interactions, thus adding lot of power to HTML.

Next it is given and example of a HTML document.

```
<!DOCTYPE html>
<html>
	<head>
		<title>This is the page title!</title>
	</head>
	<body>
		<h1>This is heading type 1!</h1>
		<p>This is a paragraph!</p>
	</body>
</html>
```

- `DOCTYPE` defines the document type to be HTML. This declaration helps the browser to display the page correctly in the browser. Each version of HTML has a different declaration that represents both is type and version;
- The code between <html> and </html> describes the HTML document;
- The code between <head> and </head> gives information about the document;
- The code between <title> and </title> provides a title to the document;
- Between <body> and </body> one can find the description of the visible content in the page;
- The code between <h1> and </h1> describes a heading type 1;
- The code between and represents a simple paragraph.

By using a browser to view the document described above, one can visualize a heading and a paragraph.

The current version of HTML is HTML5 and it was released in 2014. [69]

Cascading Style Sheets

CSS stands for Cascading Style Sheets and it is used to define styles for the websites, including the design, layout and variations in displaying the pages in different devices and screen sizes. Thus it describes how HTML elements are to be displayed in the screen. One .css file can control multiple Web pages at the same time.

CSS was created to simplify the development of large websites, with different fonts, colours, shapes, among others, for every single page, reducing once a long and expensive process. [70]

CSS rules state that there must be a selector and at least one declaration, where the selector indicates to which HTML element (or an element's attribute) is the style being applied and the declaration indicates which property is being styled and its value. Below it is exemplified the selection of a heading type 1, declaring that its text colour should be red.

```
h1 {color: red;}
```

The selector is h1, while, in the declaration, the property is color and the value is red.

In CSS there can be applied six different style sheets. They are: [71]

- Browser style – the default style sheet in a browser. It is applied when no style rules are specified;
- User style – allows the user to write a style sheet that overrides any styles created by the developer. It is applied by changing a setting in the browser. Not very usual, but can be very helpful for users with special needs;
- Inline style – used in the individual element, and applied by the style attribute. Useful for one-time element styling;
- Embedded style – controls just one Web page as it is placed inside the style element in the HTML document;
- Linked style – this style is applied by linking the HTML document to a style sheet .css file. The linkage is made in the head of the HTML document. The style sheet is applied to any document that links to the .css file;
- Imported style – similar to the above style, but it also allows to import styles directly into the HTML document.

Cascading comes from the term Cascade and is a style application hierarchy, that works a system to determine how rules must be applied. For example, if there more than one type of style sheets applied to the same element in a Web page, it is necessary to decide which style sheet to apply. Cascade is applied by the following hierarchy:

- User style overrides all other styles;
- Inline style has priority over embedded, linked and imported styles;
- Embedded style is prioritized above linked and imported styles;
- Linked and imported styles are treated equally;
- Browser style only takes place if none of the others is applied.

Another cascade responsibility is to decide how multiple style sheets of the same type are applied. If there are more than a style sheet of the same type, and there are any conflict between then, the one applied is the one at the bottom.

Inheritance is a property that means that styling is inherited from parent elements to child elements. For example, if there are two elements inside a body tag, like a h1 tag and a p tag, if there are any styles applied to the body element it will be reflected in the h1 and p elements.

```
<body>
       <h1>First element!</h1>
       <p>Second element!</p>
</body>
```

If there are any styles applied to the children elements, they will override the ones applied to the body element. However, not all properties are inherited. Properties such margin and padding, and others, are not inherited from children to parents.

Specificity is the term to describe the conflict solving algorithm that CSS has to resolve conflicts that the cascade can not resolve. Basically, the algorithm is based on how specific a rule is. [71]

JavaScript

JavaScript is a programming language designed to perform dynamic tasks. It can be placed inside a HTML document, inside a script tag. JavaScript is a cross-platform, object oriented scripting language, that, inside a host environment, like a Web browser, can be connected to objects in the environment and provide programmatic control over them. [72]

JavaScript is the only scripting language that supports client-side scripting that is supported by almost all Web browsers. JavaScript contains a standard library of objects (array, date, math, among others) and a core set of language elements like operators, structures and statements, but it can also can be extended by supplementing it with additional objects. So it is extended in ways such as: [73]

- Client-side JavaScript core language is extended by supplying objects to the Web browser and its Document Object Model (DOM);
- Server-side core language is extended by supplying objects relevant to running JavaScript on a server.

4. Content Management Systems

The concept of Content Management System (CMS) fits on the architecture of a Web Information System, making use of its components to stand as backbones in several nowadays professional and business Websites.

Content Management Systems are computer applications that allow publishing, editing, organizing and deleting content as well as managing a Website-based User Interface from an administrative interface. CMS's are often used to run Websites containing blogs, news, shopping, and learning platforms. [81]

At the highest level, content management is the process behind matching what does an organization have to what a set of definable audiences want. An organization has information and functionality of value, and there is a specific set of users who want that value. This definition and the processes behind it work as well in other outlets as on the Web, for example in a library which offer of value are books and whose crowd are readers. [82]

It can be said that CMS is similar to a framework of a pre-structured Website with basic resources readily available. Resources like usability, visualization, and administration.

A CMS allows a company to have complete autonomy over content and evolution of its presence in the Web, dismissing third-party assistance companies for routine maintenance, thus saving money. [83]

Considering the basic concept of user and content, CMS's have two main components - the Content Management Application (CMA) and the Content Delivery Application (CDA). The CMA is a Graphical User Interface (GUI) that allows the user, who may not necessarily have technical knowledge, to control the creation, modification and removal of content from a Website without the assistance of a webmaster. The CDA component provides the back-end services that support management and delivery of the content once it has been created in the CMA. [84]

4.1 Characteristics of CMS's

Features can vary amongst the various CMS offerings, but the core functions are often considered to be indexing, search and retrieval, format management, revision control and publishing.

- Intelligent indexing, search and retrieval features index all content for easy access through search functionality, allowing Website users to search by attributes such as keywords, category or author;
- Revision features allow content to be updated and edited after it was first published, while tracking any changes made to files by individuals.
- Publishing functionality allows individuals to use a set of templates approved by the organization, as well as wizards and other tools to create, modify and display content.

CMS may also provide marketing tools, like one-to-one marketing, using the ability of a Website to tailor its content and advertising to a Website user's specific interests. This function uses information gathered by the CMS or provided by the Website user. For instance, if the Website user visited a search engine and searched for digital camera, the advertising banners would display businesses that sell digital cameras and related products, instead of businesses that sell kitchen products, or other unrelated articles. [84]

4.2 Typical CMS Architecture

At first blush, content management may seem a way to create large Websites, but on closer examination, it's in fact an overall process for collecting, managing, and publishing content to any outlet, as the following list describes: [82]

- Collection – one can either create or acquire information from an external source. Depending on the source, it may be necessary to convert the information to a generic format, such as XML. Then the information is aggregated in the system by segmenting it into chunks, and adding appropriate metadata (which is defined as data about data [85] to each chunk;
- Management – a repository that consists in database records and/or files containing content chunks and administrative data is created;
- Publishing – content is made available by extracting chunks out of the repository and constructing targeted publications, which normally are Websites, but can also be printable document, and e-mail newsletters. The publications consist of appropriately arranged components, functionality, standard surrounding information, and navigation.

Taking this information into account, it can be seen in Figure 43 the typical architecture of Web CMS's, where content is provided by the Contributors and is stored in the CMS Database. Then HTML files are generated to display the content in a user friendly way in a Web browser.

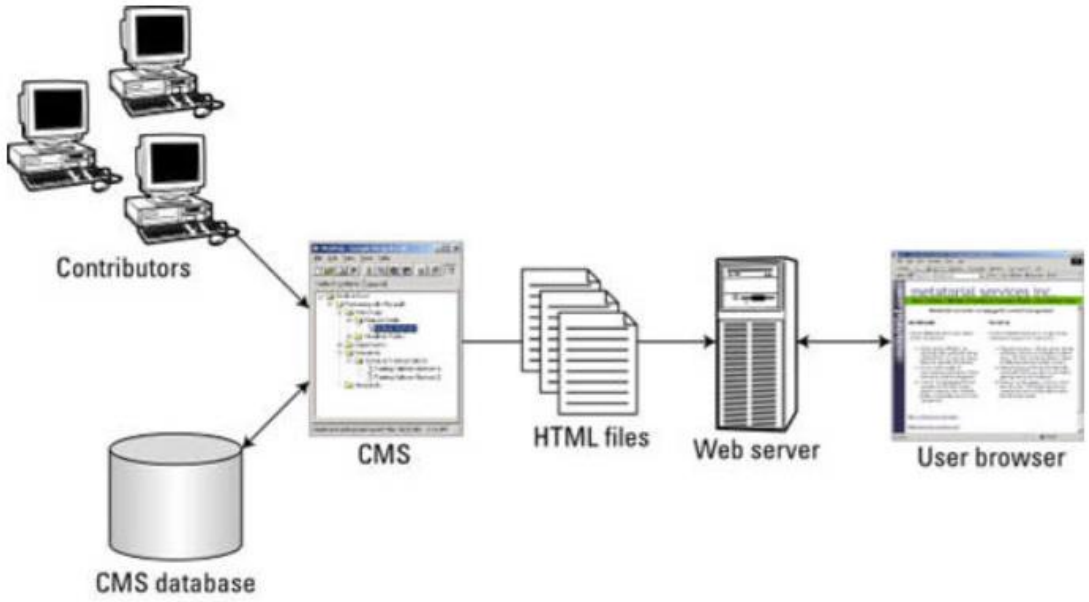


Figure 43 – CMS Typical Architecture [82]

4.3 Choosing a CMS

In order to choose a CMS software that is as close as possible to the ideal three possible options have been considered. All these are open source software, or free for use with access to source code (which also allows editing), and have a large community that gives them supports. The three software analysed were the following:

- WordPress;
- Joomla;
- Drupal.

It should be noted that although the software to implement the CMS is free of cost, it is necessary to make it available online, or in other words, install and configure the software on a Web server. This means that, if one does not wish to purchase a server for own use, it is required to hire services from third-party Web servers, commonly known as Web hosting services, which entails costs, often cyclic (monthly or annual). As it will be used a private server for self-hosting the CMS, it will only be contemplated the process of manual installation of each software.

4.3.1 WordPress

WordPress is an open source blogging tool that is based on *PHP* and *MySQL* technologies and licensed under the GPLv2 or later versions. It was designed how of the desire to create an elegant and structured personal publishing system. Being an open source project with many users there are hundreds of people around the world working on improving it. Figure 44 displays the WordPress characteristics.

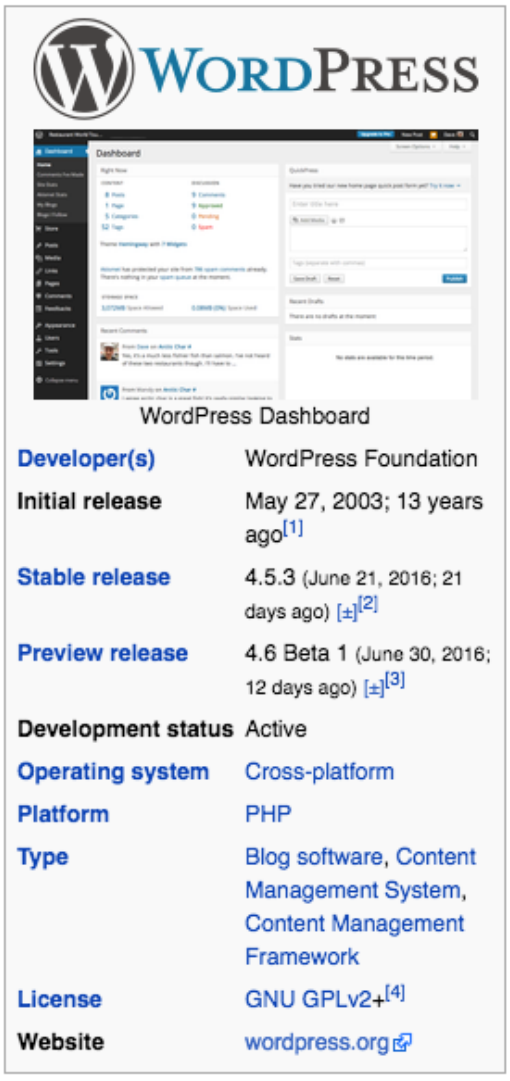


Figure 44 – WordPress characteristics published on Wikipedia [86]

The requirements for installing WordPress are displayed in Table 16. The recommended Web servers are the ones in the table, but any server that supports *PHP* and *MySQL* will also work. [87]

Software	Version
PHP	5.6 or greater
Supported Databases	
MySQL	5.6 or greater
MariaDB	10.0 or greater
Recommended Web Servers	
Apache	
Nginx	

Table 16 – WordPress minimum requirements

The manual installation of the software involves downloading its latest released and placing it in the server directory reserved for Website hosting.

After the WordPress software is installed, the administrator can access to the WordPress administration dashboard, where it is possible to customise the Website, publish new content, among others. Figure 45 represents a sample of the administration dashboard. [84]

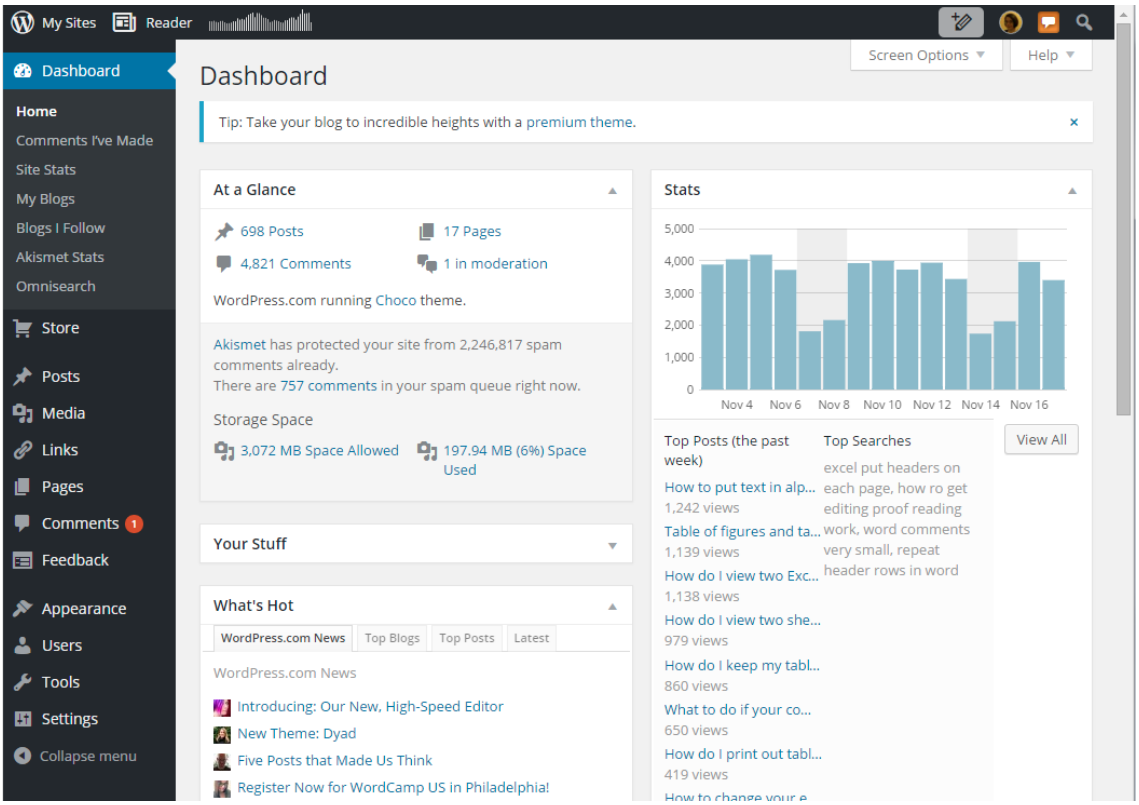


Figure 45 – WordPress administration dashboard [88]

Themes

WordPress users can install new themes in the CMS, or switch between already owned ones, which will reflect in changing Website look without changing its structure and content.

Themes can be found in two versions: free or paid. While free themes do not mean costs to the WordPress user, paid themes are often more flexible and complex at the same time, and seek to satisfy user needs, offering professional technical support. [84]

Plugins

While themes regulate the WordPress based Websites appearance, plugins extend functionalities and features the Websites. Each plugin comes with its own functionalities and features which allows WordPress users to carve their Websites as required.

These functionalities go from client portals, used to display account information to logged in users, to content displaying elements such as the navigation bars and other widgets. [84]

4.3.2 Joomla

Joomla is an open source Content Management System for Web content publishing, built on a model-view-controller (MVC) Web application framework, using object-oriented programming (OOP) techniques. The idea of developing Joomla was to provide a flexible platform for publishing and collaboration among its users.

Joomla is found being used in corporate Websites, online magazines, newspapers and publications, online commerce Websites (e-commerce), governmental applications, community portals, personal platforms, among others, making a Website as extensible as the imagination and capability of its creator. [84]

This CMS software combines content and modules using templates to produce Webpages dynamically, thus removing the need to store, update, link together and upload to the server HTML pages. Figure 46 illustrates Joomla characteristics.

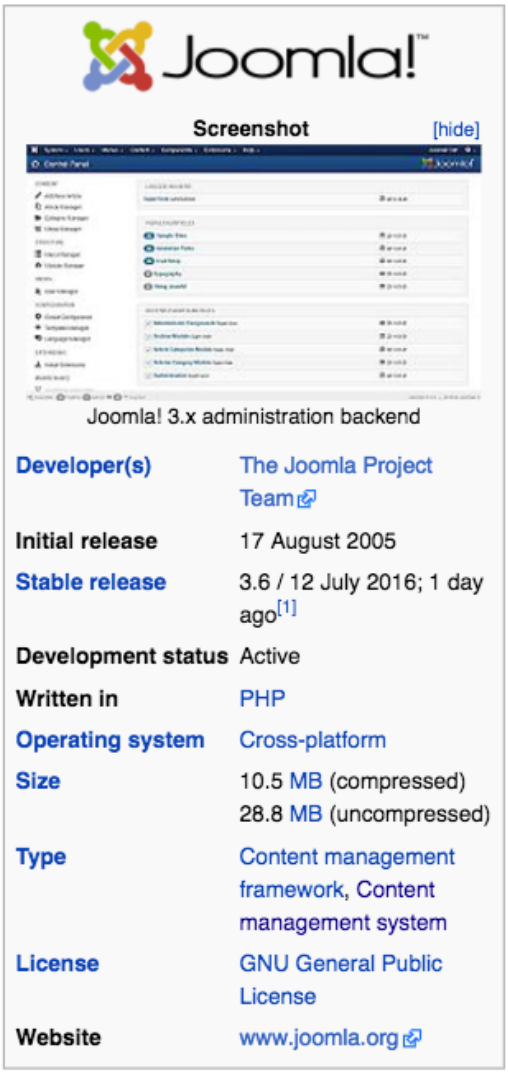


Figure 46 – Joomla characteristics published on Wikipedia [89]

Table 17 displays the minimum requirements for installing Joomla in a Web server.

Software	Version
PHP	5.3.10

Supported Databases	
MySQL	5.1
SQL Server	10.50.1600.1
PostgreSQL	8.3.18
Supported Web Servers	
Apache	2.0
Nginx	1.0
Microsoft IIS	7

Table 17 – Joomla minimum requirements

In order to install Joomla in a self-hosted Web server it is necessary to download the Joomla software and then relocate it to the host specific directory. Afterwards, it is necessary to create a database that will be assign to the CMS. After the software is installed it is possible to access the administrations dashboard, as shown in Figure 47. [84]

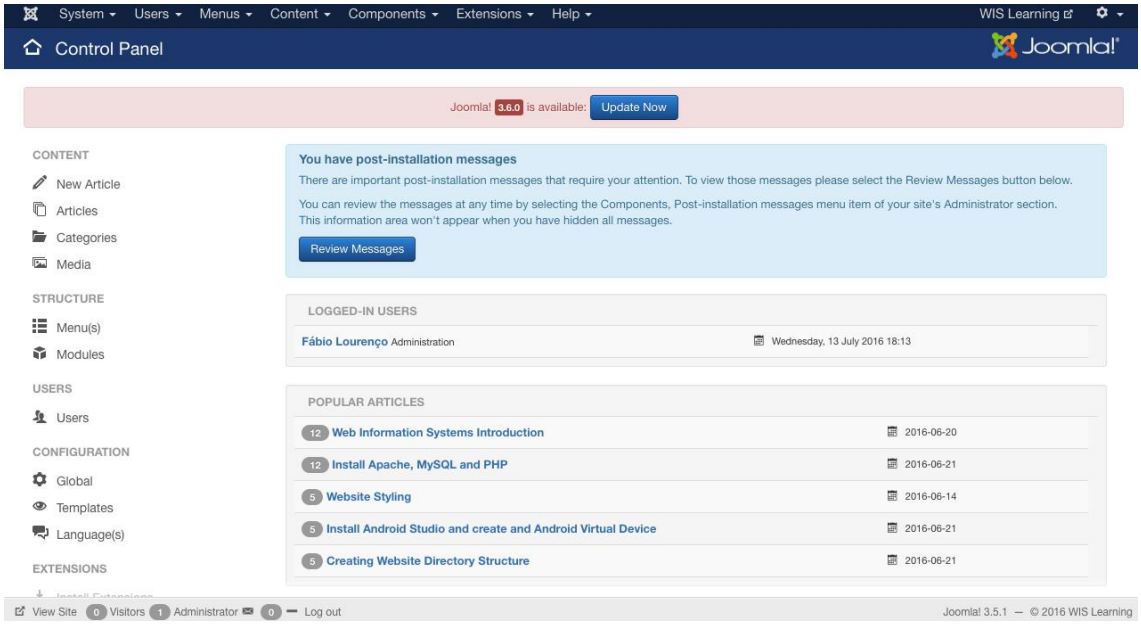


Figure 47 – Joomla administration dashboard

4.3.2.1 Joomla Extensions

Extensions are software packages that add functionalities to Joomla Websites, by providing custom features that can integrate the Website, be used by other extensions, display varied information in a side box, modify content in articles, define the platform layout, among others. [84]

Components

Components are the most complex extensions, being called to render the page body each time a Joomla webpage loads. Components are an essential portion of a Joomla based CMS because every menu item runs a component, and Joomla Websites are driven by menu items.

Usually components have two parts: a frontend part, which appears on the Website and with which Website users interact; a backend part, which is used by the CMS administrator for content and Website editing. [84]

Plugins

Plugins are advanced extensions and event. In the execution of any part of Joomla, an event can be triggered and, whenever that happens, plugins that are assigned to that function handle the event. In example, if an article is submitted by a Website, a plugin can be used to intercept that event and check the article for bad words.

When a particular type of event occurs, all plugin functions related to that type are sequentially executed. [84]

Templates

Templates are responsible for the presentation of Joomla based Websites. Creating or editing a template is simple because templates are flexible when it comes to styling the Website, allowing changes from colours, font types, among others.

Templates exist in two types: frontend templates and backend templates. Frontend templates are responsible for defining how the Website is presented to the user, styling its contents. Backend templates are where the administrator controls how administrative tasks are displayed to control everything related to the Joomla powered CMS. [84]

Modules

Modules are lightweight extensions used for rendering a Joomla page. Modules are usually associated to boxes around a component, and are assigned to menu item which enables it to show or hide depending on the menu item that the Website user is viewing.

One common example of a module is the login menu, which is allocated to a placeholder. A placeholder identifies one or several positions within the template, and are the reference for the Joomla application to assign a module to a particular position. [84]

Languages

Language extensions are the most basic type of extensions. The language packs are divided into two types: one regarding the language of the Website and the regarding the language of the administration dashboard. [84]

4.3.3 Drupal

Drupal is an open source software for creating a CMS that can be used to organize, manage and publish content in the format of a Website. It is more oriented to developers with some level of knowledge in programming areas as Drupal CMS software only provides the raw materials for developing a Drupal Website. Figure 48 displays the characteristics of Drupal software.

Figure 48 – Drupal characteristics published on Wikipedia [90]

The minimum requirements that the server as to fulfil in order to run Drupal are represented in Table 18. [91]

Software	**Version**
PHP	5.5.9
Supported Databases	
MySQL	5.5.3

MariaDB	5.5.20
PostgreSQL	9.1.2
SQLite	3.7.11
Supported Web Servers	
Apache	2.0
Nginx	0.7
Microsoft IIS	5

Table 18 – Drupal minimum requirements

The installation process is similar to the other two CMS software's. After the process is completed the Drupal user can access the administration dashboard, represented in Figure 49.

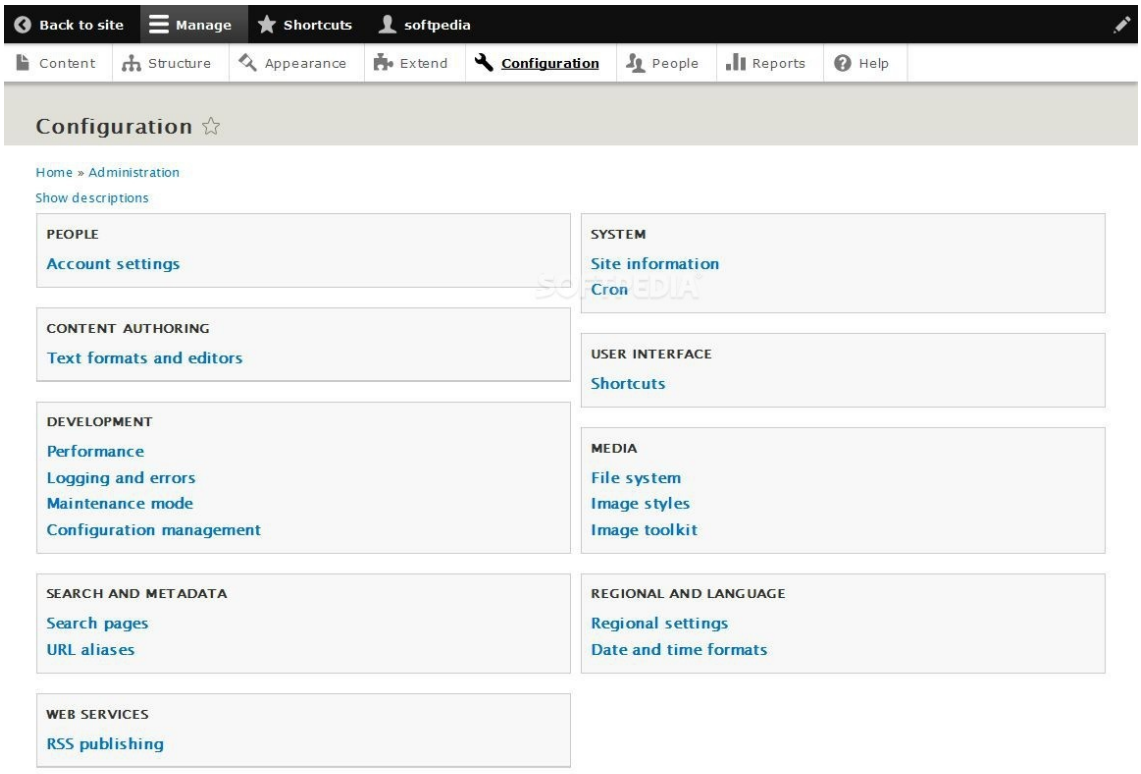


Figure 49 – Drupal administration dashboard [92]

4.3.3.1 Drupal Extensions

Basic Drupal functionalities include community features like forums, blogging, contact forms, and can easily be extended by modules and themes. [84]

Modules

Modules extend Drupal platforms functionalities and add features such as images galleries, custom content types and content listings, WYSIWYG editors, private messaging systems, third-party integration tools, among others. Some of the most commonly used modules are:

- Content Construction kit – enables the CMS administrator to dynamically create content types by extending the database schema;
- Panels – enables the possibility for the administrator to visually design the platform layout by using a drag and drop layout manager;
- Views – facilitates the presentation and retrieval of contents to the Website users. [84]

Themes

Themes are extensions used to forge the looks of a Drupal Website. The Drupal platform administrator can use themes developed by the Drupal user's community or create new ones. Community themes may not be optimized to meet all the layout objectives of a Drupal Website, so there is also the possibility to create a sub-theme which inherits the resources of an existing theme.

Instead of being used only to change the appearance of an entire Website, Drupal themes can change the appearance of only certain sections, selected types of content, or even individual pages. [84]

4.3.4 CMS Comparison

In order to properly choose a CMS software for the use case implementation it was necessary to compare the three options. Table 19 displays the main features of each CMS software, bringing out the key insights of each one, making it easier to the future decision of which software to use.

	WordPress	Joomla	Drupal
Ease of use	Requires low technical expertise	More complex than WordPress but less complex than Drupal	Requires medium/high technical knowledge
Manual installation	5 minutes	10 minutes	10 minutes
Best use cases	Simple Websites, such as everyday blogging, news, and others	Websites with more complex content and structure but still with fairly easy, intuitive usage. Used for e-commerce, e-learning, among others	Complex, advanced and versatile Websites, that require complex data organization. Community platform Websites with multiple users
Minimum requirements	PHP 5.6; MySQL 5.6 or MariaDB 10.0; Apache or Nginx	PHP 5.3.10; MySQL 5.1 or SQL Server 10.50.1600.1 or PostgreSQL 8.3.18; Apache 2.0 or Nginx 1.0 or Microsoft IIS 7	PHP 5.5.9; MySQL 5.5.3 or MariaDB 5.5.20 or PostgreSQL 9.1.2 or SQLite 3.7.11; Apache 2.0 or Nginx 0.7 or Microsoft IIS 5

Table 19 – CMS software's comparison

Following the CMS software's comparison, it was chosen Joomla CMS because it appeared to have the required features necessary for the use case implementation, without requiring a too complex development.

5. Implementation of a Use Case for Engineering Education

As main objective of this dissertation, it was developed a use case aiming to complement the formation and knowledge of engineering students in relation to the subject hereby addressed, Web Information Systems.

This use case is composed by three different elements, in order to reach as many people as possible. It is composed by:

- A Content Management System using the Joomla software;
- A mobile application for Android devices;
- A properly structured and sectioned document.

It should be noted the highly educational nature of the use case, with which it was intended to allocate more and better weapons for newly graduates to approach the employment market.

5.1 Content Management System using Joomla

As referenced in the previous chapter, Joomla CMS is an open source software that allows the developer to build the wanted platform without needing to be expert Web developer.

5.1.1 Requirements, Installation and Configuration

Requirements

As described in Section 4.3.2, it is necessary to fulfil some requirements in order to successfully install Joomla CMS.

- PHP language version 5.3.10;
- MySQL 5.1;
- Apache 2.0.

Installed Software

In order to install the Joomla CMS, it was necessary to configure the server with the following server software:

- Ubuntu Mate (lightweight Linux distribution) as operative system;
- Apache as server software;
- PHP installed;
- MySQL as database management system installed.

CMS Software Installation

At first, it was necessary to install the basis for the CMS. It occurred in the following order:

- Download of the Joomla CMS software repository from the Joomla site and relocation to the *localhost* repository;
- Creation of a database to be used by the CMS and association to its *mysql* user;
- Assignment of permissions to CMS *mysql* user.

Configuration

When the CMS is first accessed it is displayed the Setup page where some features are configured:

- Database assignment and *mysql* user credentials indication;
- Creation of the CMS administrator user account;
- CMS naming (**WIS Learning**) and other configurations.

5.1.2 Content

The CMS has given rise to an educational Website, whose content is oriented to learning concepts about Web Information Systems for interested engineering students.

Its content is organized in the following categories:

- **WIS Concepts** – introductory category, where it is explained what are WIS, where it is referred its typical architecture and the tiers that constitute it are explained. Also it is mentioned basic data flow and system interconnection representation;
- **WIS Components** – category where some possible elements (with possible it is meant that there are more elements, but are not addressed) that composed each of the WIS tiers are approached and thoroughly explained. This is a theoretical approach and concepts of Databases, Database Management Systems, Web Services (SOAP and RESTful), Android applications, Websites and UML.;
- **WIS Tutorials** – category which illustrates a practical case that aims to interrelate the components of each tier of WIS. So, it is available a tutorial that explains how to build a MySQL database, provide its data using RESTful Web services and develop an Android application that can display the data from the database in a User Interface. It also demonstrates the creation of a simple Website using HTML, CSS and JavaScript;
- **Development Environment** – category that explains how to install the development environment necessary for implementing the examples present in the tutorials. It also contains elements for testing.

Each category is divided in sections that contain articles about each and every subject here addressed. All categories are accessible in the main menu of the Website, as shown in Figure 50. By putting the mouse over each category, it shows a hyperlink to its sections.

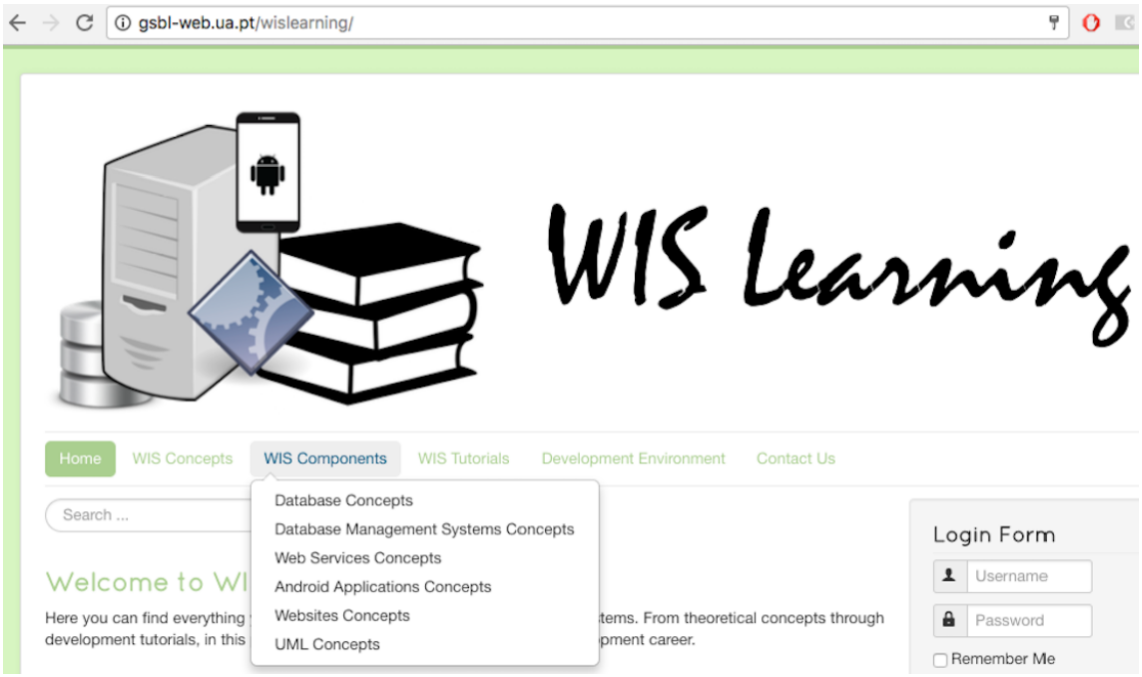


Figure 50 – WIS Learning main menu and its sections

5.1.3 Login

With the Login feature (Figure 51) it is intended to distinguish the users, thereby allowing, for example, to restrict the access to some articles to only one type of users, to give ability to publish and edit to some users, and to allocate moderating capabilities to specific users.

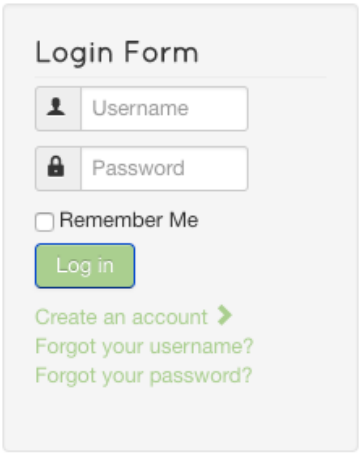


Figure 51 – Login form of the WIS Learning Website

It also enables tracking user activity, whose information can be used to store user preferences, allow the user to save their favourite items, show dynamic advertisement related to the interests of each different user, among others.

5.1.4 Roadmap

In order to implement a "learning pathway", an interactive table was developed (using HTML and JavaScript), located in the Website Homepage, which recommends a path that a user can follow in the learning of WIS. The roadmap cells can be opened, which will show steps to Theoretical and Practical learning about Web Information Systems. Figure 52 illustrates the roadmap with both categories in hidden state.

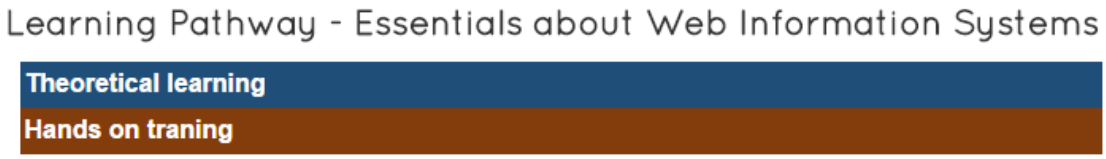


Figure 52 – Roadmap to learning essentials about WIS in the Website (in hidden state)

The theoretical learning section of the roadmap is divided in two steps, which guide the user through learning WIS concepts and its building blocks, as displayed in Figure 53.

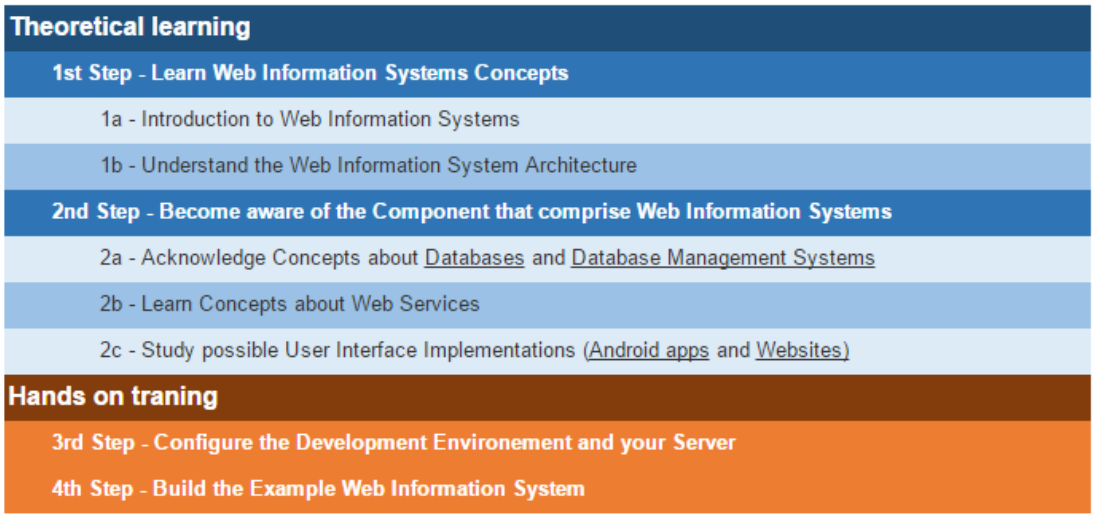


Figure 53 – Theoretical steps for WIS learning present in the Roadmap

Hands on training category is also divided in two steps, conducting the user first through development environment configuration and then through the implementation of a practical use case that consists in the construction of a complete Web Information System, that can be customised to any project type wanted. Figure 54 represents the Hands on training section of the roadmap.

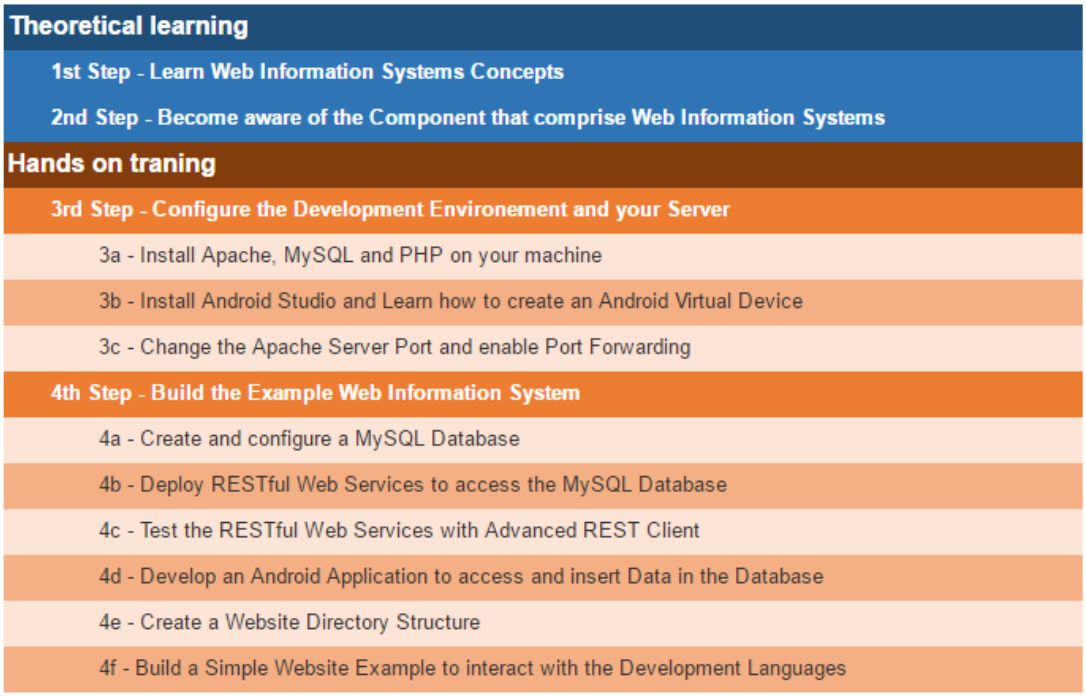


Figure 54 – Practical steps for WIS learning present in the Roadmap

With the roadmap to WIS leaning it is intended that the user evaluates its current capacity and chooses the adequate step to start into.

In the future, the aim is to develop an interactive Roadmap using figures, that will be located at the side position of the Website, displaying information accordingly to the category and section where the user is. It is also intended to enable the markup of check marks (automatic or not) for each article of the sections that the user completes. The Roadmap intended is something inspired in Figure 55.

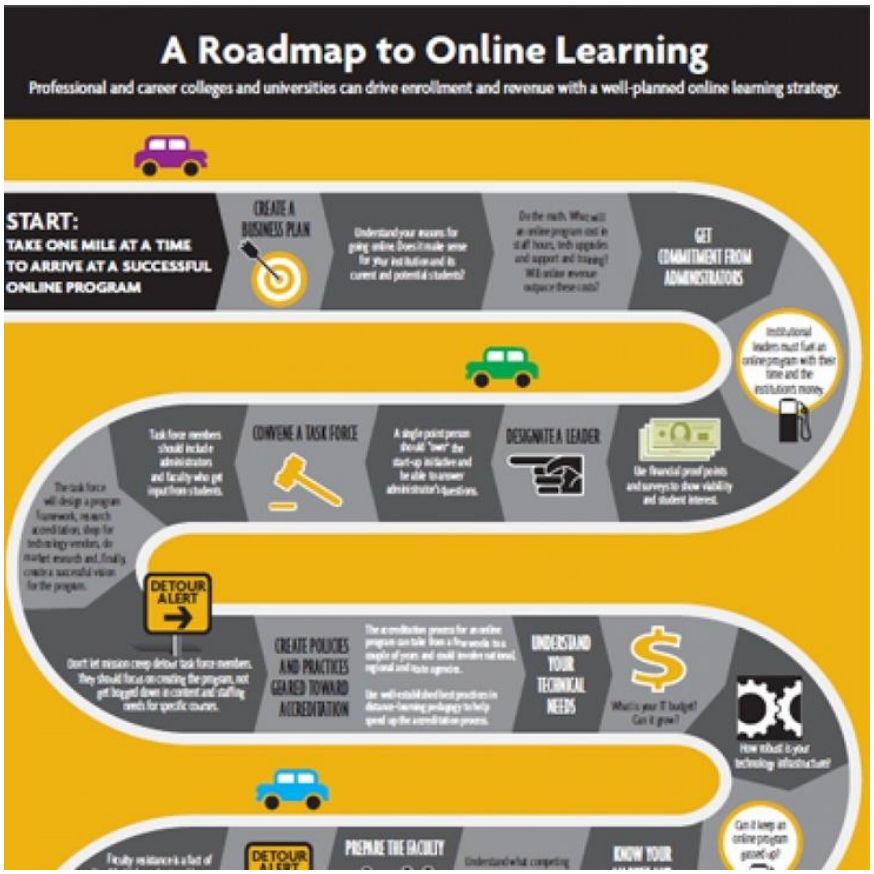


Figure 55 – Inspirational Roadmap for future implementation [93]

5.1.5 Search Module

It was chosen to include a search module on the Website, which allows users to quickly search for published articles related to certain topics and keywords.

Figure 56 – Search module located under the Main Menu

The module figures below the Main Menu in order to be easily detected (Figure 56) and generates search results in articles, categories, sub-categories, among others, that can be applied filters (Figure 57).

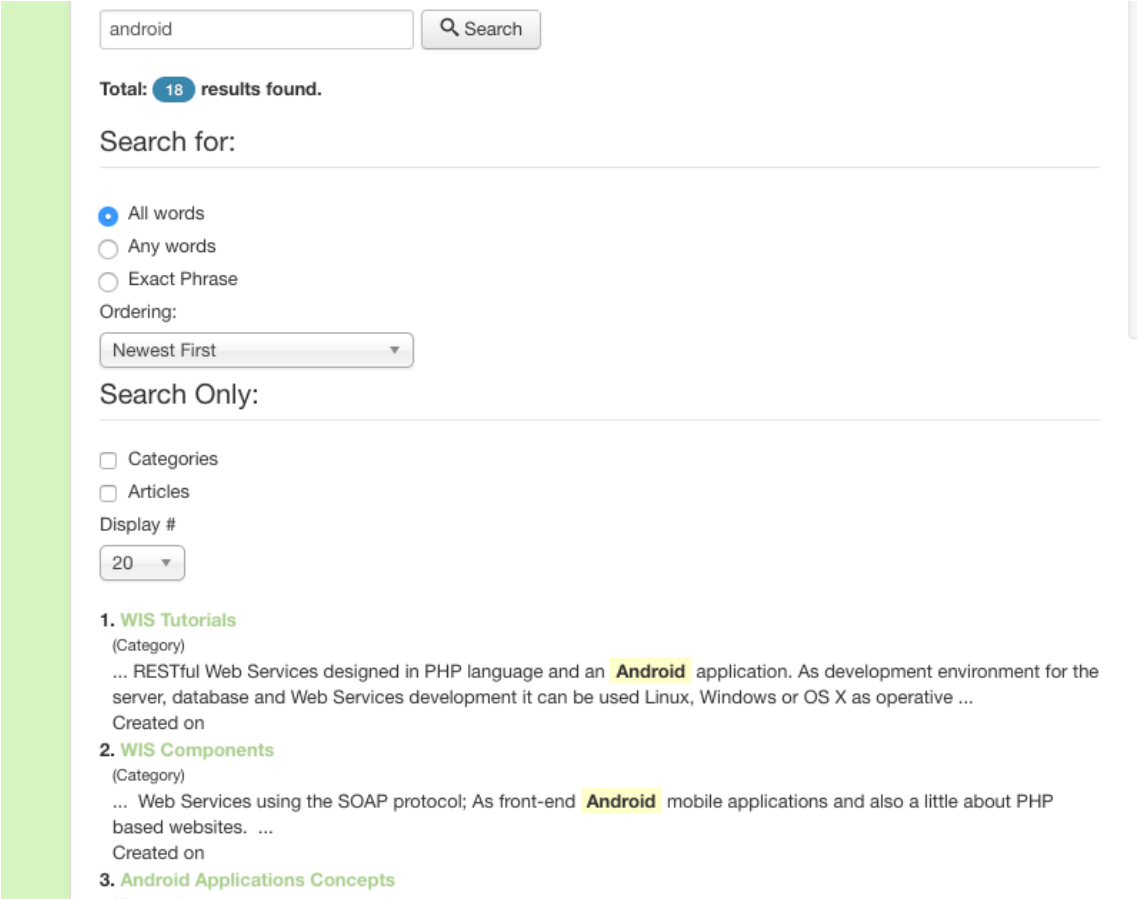


Figure 57 – Search results for "android" keyword

5.1.6 Future Features

In order to give continuity and improve the entire platform, it is intended to implement some features that, these days, are found present in many other Websites of this and other genres.

To start it is intended to take advantage of the existence of user accounts to limit the content available to different groups of users. It is also intended to allow users to create and edit articles, thereby facilitating the achievement of new content, the expansion of existing information on the platform to new areas and to correct any typos and errors that may exist in the articles already published. After submitting the article, whether created articles or edited articles, it should wait on hold until approved by a user with privileges of moderation, namely a moderator. After being approved the article will be published on the Website and in addition to being displayed in the respective area of the category to which it belongs, it will also appear in a section of new articles.

It is planned to integrate two extensions on the content manager that enhance the quality and quantity of content that may appear on the Website. These extensions have as functionalities the capacity of file uploading to the content manager (Simple File Uploader) and displaying documents with extension .docx, .pdf, .xlsx, among others (Ari docs). Combining these features it enables the inclusion in the created articles, either by users or by administrators, documents corresponding to scientific articles, publications, books, and many more that are related to the information in the article. It also allows to make use of existing quality resources, duly crediting their authors, rather than creating new resources that mean more work and where there is a risk to obtain something with less quality (and perhaps less credibility).

In the present times the social networks occupy an increasingly larger space in the daily lives of every citizen, which makes it almost mandatory that they are part of any Website and online platform. That said, the intention is to integrate at least one social network on the platform using for this purpose an extension that provides such functionality. Including a social network will benefit when creating new user accounts and logging in at the page, since the user, whether creating a new account or logging in an existing account, will just have to input his social network credentials to be authenticated, thereby facilitating these processes. It will also facilitate the dissemination of the Website itself, as well as allow the sharing of articles that are already published.

In the future it may be considered the possibility of making the existing content at the platform marketable. Combining the fact that there are user accounts, which keep the record of each user, with a potential extension (VirtueMart) that allows placing articles that the user wishes to acquire in a shopping cart and enables forwarding to payment pages (PayPal for example) and invoice generation, it is possible to reach said purpose. This feature brings business value to the platform, which content quality level may benefit with this value, as potential users may be interested in publishing their articles on the platform because they can get paid for it. So, this feature can be translated into an enrichment both at commercially and intellectual level for the project.

Access to the WIS Learning

The WIS Learning Website can be accessed by anyone that is connected to the *eduroam* network or to its VPN, by following the URL below:

http://gsbl-Web.ua.pt/wislearning2/

5.2 Android Mobile Application

Taking advantage of the fact that information about WIS is already distributed in a database, the one database used by the CMS Joomla, and with the aim of getting information to as many people it was decided to continue the development by creating a mobile application for Android systems that can display the content present on the Website and its database.

The content is found in the form of articles, divided into categories and sub-categories.

Requisites

In order to be able to provide the contents of the database to the mobile application it was necessary to implement services that allow access to information. It was chosen to use RESTful Web Services because of its simplicity and the fact that they use HTTP methods. As data format JSON was chosen.

5.2.1 Access to Content

RESTful Web services were created using the PHP programming language, making use of *mysqli* library to perform the connection to the MySQL database and make queries to it, thus obtaining the required information.

The connection is made including the *connection.php* file in each of the services to be developed. This file calls a method that makes the connection to the database, to which are sent as arguments the location of the database, on this case *localhost*, the name of the database to be accessed, and the access credentials to it. Therefore, the method called is the following:

```
mysqli("localhost", "user", "password", "wislearning_db")
```

Were created three main services to access the data in the database:

- A service that selects all categories in the database, excluding some whose information does not matter for the application;
- A service that selects the sub-categories that belong to a particular category;
- A service that selects the articles belonging to a particular category or sub-category.

Select Categories

The service that selects the categories in the database has as address `http://gsbl-web.ua.pt/wislearning2/mobile/select_categories.php` and calls the following SQL command (using the *query()* method):

```
SELECT id, title, description FROM categories WHERE level = 1 AND
NOT title = 'Uncategorised' AND NOT title = 'About';
```

Thus it obtains a set of information that contains the id, name and description (description is in HTML format) of all categories except categories About and Uncategorised. This information is transformed into an associative array by *mysqli_fetch_assoc()* method. Below one can see an excerpt of the information obtained, in JSON format, when invoking the service.

```
[
  {
    "id": "8",
    "title": "WIS Concepts",
    "description": ""
  },
```

```
  {
    "id": "9",
    "title": "WIS Tutorials",
    "description": "<div class=\"page\" title=\"Page
60\">\r\n<div class=\"layoutArea\">\r\n<div
class=\"column\">\r\n<p style=\";\..."
  }
]
```


Select Sub-Categories

The service that selects the sub-categories that belong to a determined category in the database has as address `http://gsbl-web.ua.pt/wislearning2/mobile/select_subcategories_by_parent_id.php`, to which it is added a parameter, since the request is *HTTP GET*. The submitted parameter is *parent_id*, which is the id of the category.

The SQL command executed, by calling the *query()* method, is:

```
SELECT id, title, description FROM categories WHERE parent_id =
'$parent_id';
```

Once again it obtains a set of information that contains the id, name and description (description is in HTML format) of the sub-categories that belong to the searched category. This information is transformed into an associative array by *mysqli_fetch_assoc()* method. Below one can see an excerpt of the information obtained, in JSON format, when invoking the service.


```
[
  {
    "id": "20",
    "title": "MySQL Database",
    "description": ""
  },
  {
    "id": "21",
    "title": "RESTful Web Services",
    "description": ""
  },
  {
    "id": "22",
    "title": "Android Application",
    "description": "<p style=\"text-align: justify;\"><span
style=\"font-size: 10pt; font-family: Arial;\">In this part of
the example it is assumed that the Android Studio IDE is already
installed and that the user either has an Android smartphone or
that an AVD (Android Virtual Device) is configured, as described
in annex.</span></p>..."
  }
]
```


Select Articles

The service that selects the articles that belong to a determined category or sub-category in the database has as address `http://gsbl-web.ua.pt/wislearning2/mobile/`

`select_content_by_catid.php`, to which it is added a parameter, since the request is *HTTP GET*. The submitted parameter is *catid*, which is the id of the category or sub-category.

The SQL command executed, by calling the *query()* method, is:

```
SELECT id, title, introtext FROM content WHERE catid = '$catid';
```

Once again it obtains a set of information that contains the id, title and HTML content of the article that belongs to the category or sub-category. This information is transformed into an associative array by *mysqli_fetch_assoc()* method. Below one can see an excerpt of the information obtained, in JSON format, when invoking the service.

```
[
  {
    "id": "8",
    "title": "Creating an Android Project",
    "introtext": "<p style=\"text-align: justify;\"><span
style=\"font-size: 10pt;\">Let&rsquo;s start by opening Android
Studio and creating a new Android Studio project. The application
will be called RestaurantDB and the company domain
praticalexample.com, according to the database. A configu new
package </p>..."
  },
  {
    "id": "9",
    "title": "Menu Activity",
    "introtext": "<p style=\"text-align: justify;\"><span
style=\"font-size: 10pt;\">Now it&rsquo;s time to design the user
interface for the Menu activity. So open the
startActivity(queryClientsIntent);</span><br /><span
style=\"font-family: 'courier new'...
  }
]
```

5.2.2 Android Application

After making available the contents of the CMS database, it was departed to the development of the Android mobile application.

As target devices, it was chosen smartphones and tablets that run an Android version equal or above Android 4.1, Jelly Bean, as these constitute more than 90% [94] of the Android world.

In AndroidManifest.xml it was necessary to include permission for Internet access in order to be able to communicate with the server. In the project it was necessary to add two libraries: Volley, to consume the services previously created, and Picasso, to treat and display images in a simple way.

5.2.2.1 Classes

In order to be possible to process the data received in the application, it was necessary to create two classes: Category, to handle the received Information about categories and sub-categories, and Article, to handle the received information about articles.

Below it can be seen the structure of each class where one can see the constructor and methods to return each of its attributes.

```
public class Category {
    private int mId;
    private String mTitle,
mDescription;

    public Category (int id, String
title, String description) {
        mId = id;
        mTitle = title;
        mDescription = description;
    }

    public int getId () {
        return mId;
    }

    public String getTitle () {
        return mTitle;
    }

    public String getDescription () {
        return mDescription;
    }
}
```

```
public class Article {

    private int mId;
    private String mTitle, mHtml;

    public Article (int id, String
title, String html) {
        mId = id;
        mTitle = title;
        mHtml = html;
    }

    public int getId () {
        return mId;
    }

    public String getTitle () {
        return mTitle;
    }

    public String getHtml () {
        return mHtml;
    }
}
```

5.2.2.2 Activities

Four activities were created in order to be possible to display all the information. Each activity is hierarchically invoked as the user progresses in the application. It is associated with each activity their respective layout, which defines and puts all its elements in the respective position. Invocation of activities from other activity is performed through *Intents* and the exchange of information between them through the *Bundle Extra*.

Categories Menu Activity

The activity CategoriesMenuActivity is composed by two elements: an *ImageView* and a *ListView*.

The *ImageView* element displays the WIS Learning logo and the *ListView* element shows a list of available categories. The *ImageView* element is populated using methods from the *Picasso* library. The categories are obtained in JSON format through *select_categories.php* service and then transformed into a *List* element of the *Category* class, executing the *getCategoriesList()* method (which uses methods that belong to the *Volley* library).

When the user presses one of the categories on the list it is triggered an event that calls SubCategoriesMenuActivity activity if the category contains sub-categories, or ArticlesMenuActivity activity if the category does not contain sub-categories. The id, title and description fields are sent for the following activity.

The Figure 58 shows the activity in operation.

Figure 58 – WIS Learning application main activity, CategoriesMenuActivity, populated with categories

Sub-Categories Menu Activity

The activity SubCategoriesMenuActivity is composed by two elements: a *WebView* and a *ListView*.

The *WebView* element displays the parent category description and the *ListView* element shows a list of available sub-categories that belong to the parent category. The sub-categories are obtained in JSON format through *select_subcategories_by_parent_id.php* service, sending the parent category id (receive in the *Bundle Extra*) as *GET* parameter, and then transformed into a *List* element of the *Category* class, executing the *getSubCategoriesList()* method (which uses methods that belong to the *Volley* library). The text in the *ActionBar* is set to the title of the parent category. If the parent category has a description it is used to populate the *WebView* element as it is formatted in HTML, if it has not the element will be hidden.

When the user presses one of the sub-categories on the list it is triggered an event that calls ArticlesMenuActivity activity and sends it the id, title and description fields.

The Figure 59.A shows the activity in operation when the parent category has a description and Figure 59.B when it has not.

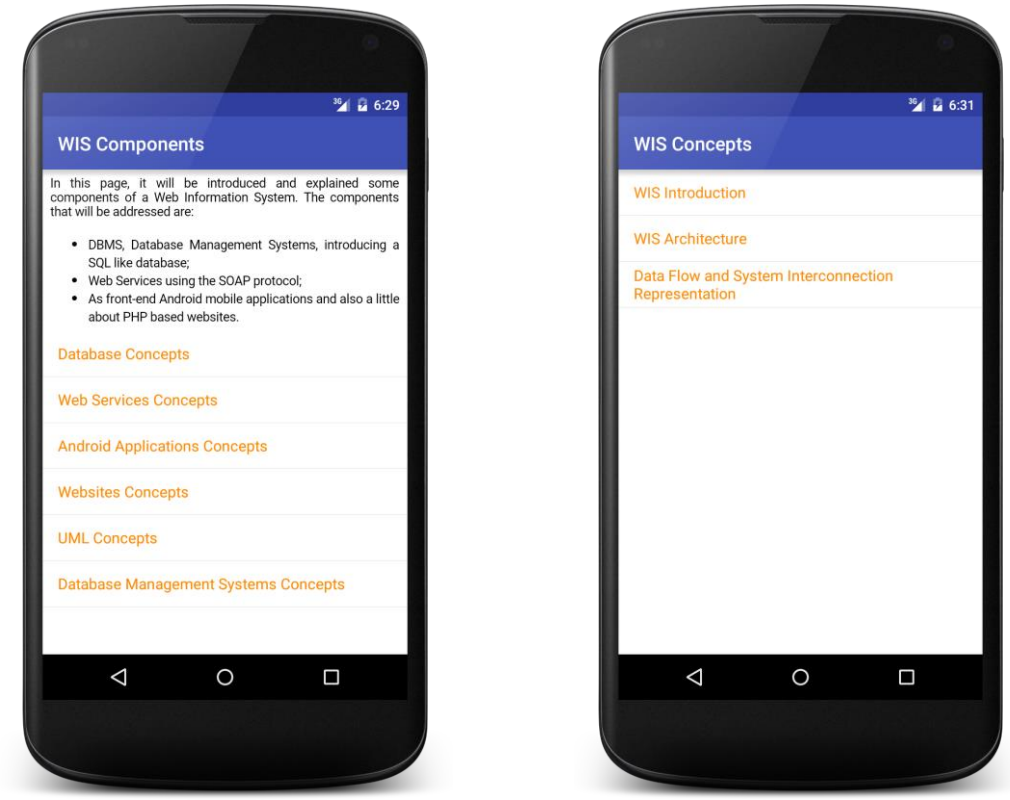


Figure 59 – SubCategoriesMenuActivity, populated with sub-categories. A (left) with description, B (right) without description

Articles Menu Activity

The activity ArticlesMenuActivity is composed by two elements: a *WebView* and a *ListView*.

The *WebView* element displays the parent category or sub-categoty description and the *ListView* element shows a list of available articles that belong to the parent category or sub-category. The a are obtained in JSON format through *select_content_by_catid.php* service, sending the parent category or sub-category id (receive in the *Bundle Extra*) as *GET* parameter, and then transformed into a *List* element of the *Article* class, executing the *getArticlesList ()* method (which uses methods that belong to the *Volley* library). The text in the *ActionBar* is set to the title of the parent category or sub-category. If the parent category or sub-category has a description it is used to populate the *WebView* element as it is formatted in HTML, if it has not the element will be hidden.

When the user presses one of the articles on the list it is triggered an event that calls ArticleActivity activity and sends it the title and HTML content fields.

The Figure 60.A shows the activity in operation when the parent category or sub-category has a description and Figure 60.B when it has not.

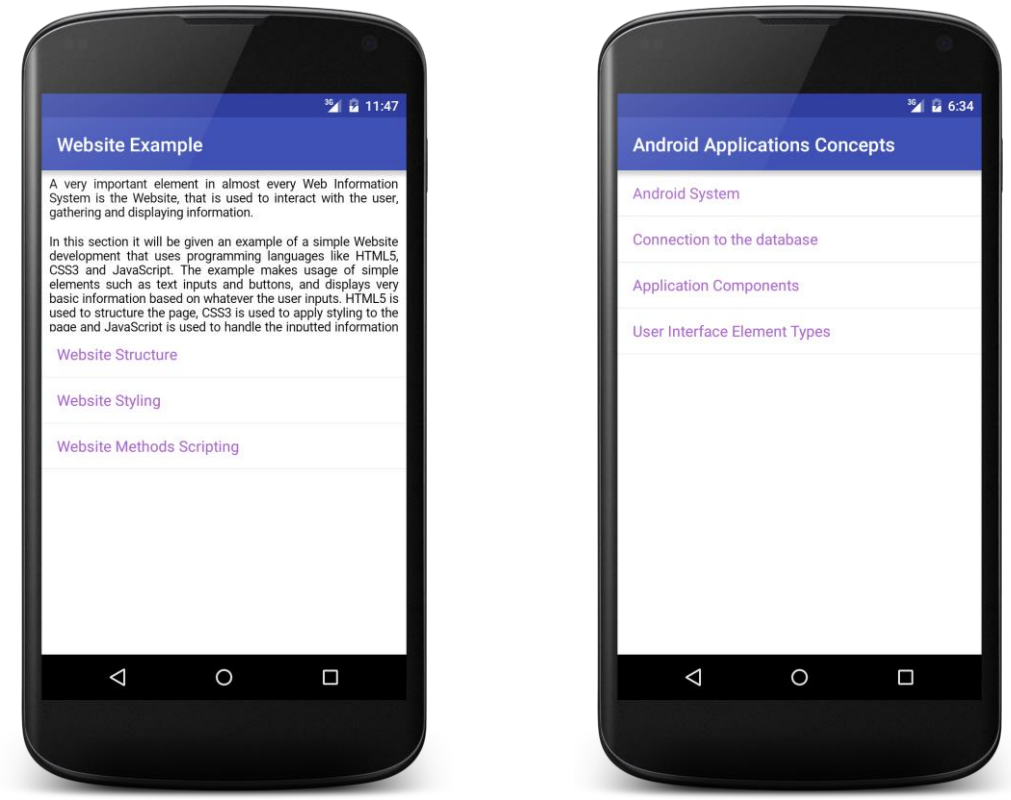


Figure 60 – ArticlesMenuActivity, populated with articles. A (left) with description, B (right) without description

Article Activity

The activity ArticleActivity is the top activity of the application and it is composed by a *WebView* element.

The *WebView* element displays the associated article content, as it is formatted in HTML, which is obtained in the *Bundle Extra*. In order to properly show images in the *WebView* it is necessary to edit the HTML content *String*, completing the images URL's and adding CSS styling in order for the images to fit the device screen. The text in the *ActionBar* is set to the title of the article.

The Figure 61.A shows the activity in operation displaying a simple article, Figure 61.B displays an article that has a table properly scaled and Figure 61.C shows an article that has a image properly scaled to the device screen.

Figure 61 – ArticleActivity displaying article. A (left) simple article, B (right) article with table, C (centre) article with image

5.2.2.3 Login Feature

In order to be able to build the login feature in the mobile application it was necessary to develop components not only at the level of the Android application, but also in terms of the content manager (Joomla CMS).

From the CMS side, whereas for accessing the informative content it was enough to create the services that access the database and select categories, sub-categories and articles, the authentication process is more complex. Since the passwords of users are complexly encrypted in the database table that contains information about users, you can simply compare not the credentials submitted by the user in the mobile application with the credentials that are found in the database. In the situation below is possible to see the contents of the database for the user “test_user”.

```
+-----+-----------+-----------+----------------+--------------------------------------+
| id  | name      | username  | email          | password                             |
+-----+-----------+-----------+----------------+--------------------------------------+
| 734 | Test User | test_user | user@gmail.com | $2y$10$zziYJPUTO1LyZlwvCE5FWvTBdOSUGq |
+-----+-----------+-----------+----------------+--------------------------------------+
```

The password that was entered when creating this user account was "pass", although one can verify something different in the *password* field in the excerpt above.

To create the component, and since it must respect a directory structure which subsequently is compressed in a *.zip* file, it was used the Component Creator [95] tool (accessible via Web browser) that after entering all essential information for the component it generates the basic structure needed to create it (which is then downloaded).

The component was named *com_mobilelogin*. After downloaded its base structure it was edited the files needed to create the desired functionality. As main editing it was altered the file *Controller.php* from *site* directory. This file creates the MobileloginController class that extends the Joomla class *JControllerLegacy* and invokes methods from *JFactory* class.

```
class MobileloginController extends JControllerLegacy{
function login() {
       app = getApplication();

       credentials = (
                    username => app ->get('username'),
                    password => app-> get('pass')
             );

             if (app.login(credentials))
                    // Success
                    user = JFactory::getUser();
                    data = array(
                           'message' = 'success',
                           'id' = user.id,
                           'username' = user.username,
                           'name' = user.name,
                           'email' = user.email
                    );
             else
                    // login failed
                    data = array(
```

```
                                    'message' = 'login failed'
                                );

                            header('Content-Type: application/json');
                            send(data);
                        }
                }
```

After editing the files, the component structure was compressed, creating the *.zip* file to be installed on the CMS. Then the CMS administration dashboard was accessed, where the created component was installed by using its *.zip* file, as in the Figure 62.

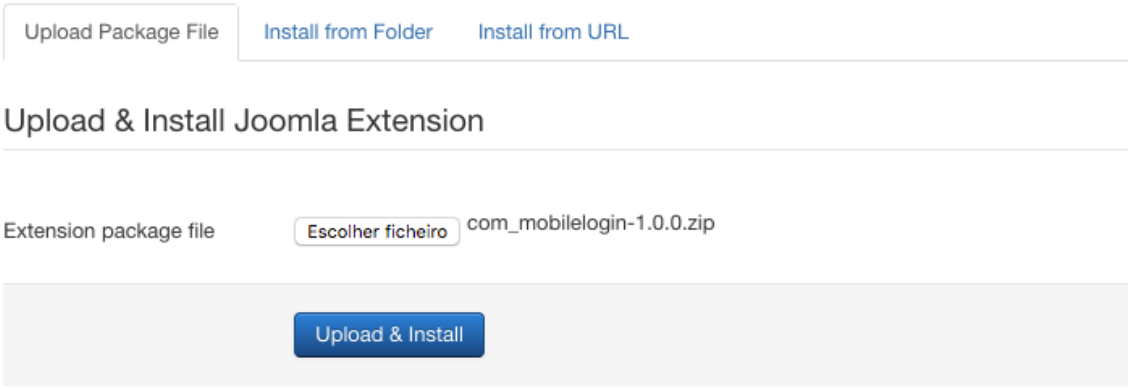


Figure 62 – Mobile login component installation on Joomla CMS

The authentication service is called using the address `http://gsbl-web.ua.pt/wislearning2/index.php?option=com_mobilelogin&task=login&username=username&pass=password` including the parameters *username* and *pass*. In the case of a failed authentication the service responds, in JSON format, as follows:

```
{"message":"login failed"}
```

If the authentication is performed correctly the service responds with information about the user, in JSON format.

```
{
  "message": "success",
  "id": "734",
  "username": "test_user",
  "name": "Test User",
  "email": "user@gmail.com"
}
```

Login Activity

In the the mobile application it was created a new activity, *LoginActivity*, which makes use of *Volley* library to consume the authentication service previously released.

The activity consists of two *EditText* elements that allow the user to enter its credentials, *username* and *password*, and a *Button* element to submit the credentials. The activity is called when there is that there is no user logged in.

When the user enters its credentials and submits them to login, the service is called using the HTTP GET method and receives a JSONObject, from which is withdrawn the *message* field to check whether the authentication was successful or not.

If the user does not fill in the fields *username* or *password*, or enters a *password* with fewer than four characters, errors are displayed.

Figure 63 displays LoginActivity running.

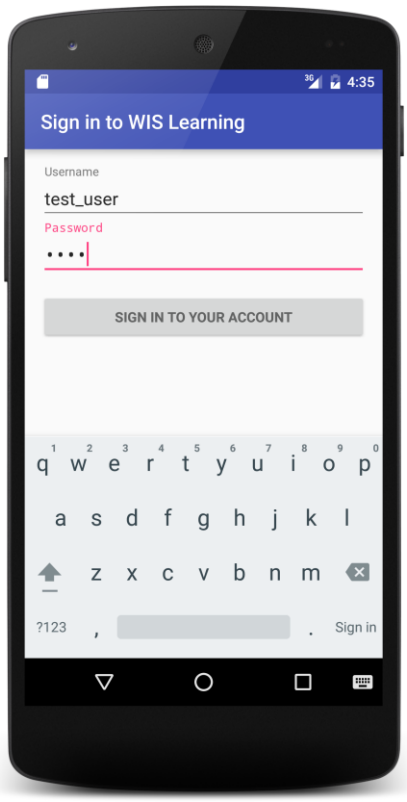


Figure 63 – Login Activity from WIS Learning application

5.2.3 Future Functionalities

As future functionality, it is intended to include a Registration activity, in order to allow new users to access the informative contents that reside on the CMS database. Firstly, it would be necessary to expand the *com_mobilelogin* by adding a registration method. Then, it would be necessary to give users the possibility to register in the mobile application, by proving elements for them insert their data and by calling the registration service.

It is also intended to include social networks in the application that would facilitate the content sharing by the users with whom they want. This functionality can also be used to create new user accounts and authenticate users logging in easily.

The Android mobile application, **WIS Learning**, will work on every Android smartphone and tablet that runs at least Android 4.1, Jelly Bean. It is required to be connected to a *eduroam* network or to its *VPN*.

In order to install the application it is required to download the *.apk* file to the device and to run it. The *.apk* file is available for download in the following link:

https://db.tt/z8x47vrU

5.3 Didactic Document

It was considered the creation of the didactic tool in document format to concatenate all of the content covered in this project, from the theoretical formation to the practical instruction. That said, the document allows the user to have a tool available with all the information about WIS that can be stored in a directory (offline), online shared and even provided in the printed version (version which neither the CMS or the mobile application provide).

5.3.1 Structure

The didactic document is organized along similar lines as the Website, first introducing the theoretical concepts and then exemplifying their application in practice. At the end of the document are sections that explain how to install and configure the tools and work environment. The didactic document therefore has the following structure:

3. Information Systems
    - 3.1 Definition
    - 3.2 Web Information Systems

4. Web Information System Architecture
    - 4.1 User Interface tier
    - 4.2 Data Transfer tier
    - 4.3 Data Processing and Storage tier
    - 4.4 Netflix's Architectural Design as example

5. Data flow and system interconnection representation
    - 5.1 Data encapsulation in OSI model
    - 5.2 Example – Accessing ua.pt

6. Web Information Systems components
    - 6.1 Databases
        - 6.1.1 Entities
        - 6.1.2 Attributes
        - 6.1.3 Relationships
        - 6.1.4 Keys
        - 6.1.5 Assigning types example
        - 6.1.6 Normalization
        - 6.1.7 Data Integrity
        - 6.1.8 Structure example
        - 6.1.9 Database Management Systems
        - 6.1.10 Data Definition, Manipulation, Control and Query Languages
        - 6.1.11 SQL Database Language
    - 6.2 Web Services
        - 6.2.1 XML-based SOAP Web Services
            - 6.2.1.1 Web Services Protocol Stack
            - 6.2.1.2 Service Transport
            - 6.2.1.3 XML Messaging
            - 6.2.1.4 Service Description
            - 6.2.1.5 Service Discovery

The chapters 7. Practical Example and Annex can be found on the Appendix of this dissertation.

The didactic document is available for download in the following link:
https://db.tt/LDrPER7M

6. Conclusions

In the process of setting up the structured didactic tool it was necessary to devote plenty of time to obtain verified quality content about Web Information Systems, both by being quite a few topics to be covered, as sometimes by being difficult to find sufficient specific and coherent concepts definitions.

Nevertheless, it was possible to construct three different formats of the indicated tool, allowing to cover different types of individuals within the target audience, the students of engineering courses outside the area of information and communication technologies.

With the didactic tool, it is intended that the interested individual will be able to:

- Understand databases and DBMS, learning to design them according to the architecture of the system in which they operate. Translate this knowledge to practice by implementing a MySQL database;
- Perceive the machine-to-machine process provided by Web Services. Apply this knowledge by developing RESTful Web Services to make available online the data stored in the MySQL database;
- Assimilate concepts of Android mobile applications and Websites, carrying this information to develop a User Interface that consumes the RESTful Web Services to display in a user-friendly manner the data stored in the MySQL database.

Therefore, it was offered an integrated and structured mechanism for the target audience to learn how to develop a system that fits in the architecture of a WIS.

6.1 Results

In the format of Website, that makes use of a Content Management System based on the open source Joomla software, besides allowing the management of information that is intended to provide to users, also allows displaying the information in a highly organized and customizable manner. All the information present on the Website, about categories, content, users, among others, is stored in a MySQL database.

The latter situation facilitated information providing and handling to the Android mobile application format. With the creation of RESTful Web Services, the information has become available in the Android application, thus providing the tool in a format that allows tablets and smartphones users to learn the concepts here approached in a native way for their devices. The CMS along with a created extension also permitted user authentication in the Android application.

The tool in document format provides the possibility to maintain a formal record of its addressed contents. In addition, it provides an alternative format to the type of user that prefers more traditional formats of learning, since the document can be stored in a device or printed.

The Website provides an interactive roadmap that is intended to guide the user in what should be the path to follow in learning WIS. Furthermore, at the level of operation it displays the contents as articles that are divided into categories and sub-categories, allows the creation of user accounts and login to them, and enables content search using keywords with the possibility of filtering the search.

In the mobile application login to user account is required, which unlocks the informative contents. Initially, it displays a list of categories that contain information about WIS and, upon selection of a category by the user, evolves into a list of sub-categories, a list of articles and, finally, displays the information about the selected article.

Preliminary trials indicate that involved engineering students are reacting very positively to the utilization of the tool:

- The time from familiarization with the basic WIS concepts to the start of simple developments is in the order of a few weeks, with approximately 4 hours of work per week.
- The self-confidence of non-information and communications technologies students in dealing with the incorporation of Web Information Systems in their projects grows very rapidly, therefore contributing to the creation of a new form of transversal technical literacy.

It is also worth mentioning that the work developed gave rise to a scientific paper to be presented at the International Conference on Interactive Collaborative Learning, held in September in Belfast, United Kingdom.

6.2 Future Work

In the Website and Android application formats, it is intended to take advantage of the existence of user accounts to limit the content available to different groups of users, to allow users to create and edit articles following by a moderator revision.

It is planned to integrate two extensions on the content manager that enhance the quality and quantity of content that may appear in the Website. Simple File Uploader gives the capacity of file uploading to the content manager. Ari docs displays documents with extension .docx, .pdf, .xlsx, among others. These extensions together enable the inclusion of documents corresponding to scientific papers, publications, books, and others, in the informative contents

As social networks occupy an increasingly larger space in the daily lives of every citizen, it is intended to integrate at least one social network on the platform, that would facilitate the content sharing by the users with whom they want. This functionality can also be used to create new user accounts and authenticate users logging in easily.

It may be considered the possibility of making the existing content in the platform marketable and possibly expand it to other areas.

Besides the features that are intended to develop in the future, there is a need to validate the applicability of the tool among the student community.

Bibliography

[1] R. Stair and G. Reynolds, An Introduction to Information Systems, 10 ed., New York: Cengage Learning, 2013.

[2] R. Vidgen, D. Avison, B. Wood and T. Wood-Harper, Information System Development, Butterworth-Heinemann, 2002.

[3] C. Borodescu , “Web Sites vs. Web Apps: What the experts think,” Vision Mobile, 29 07 2013. [Online]. Available: http://www.visionmobile.com/blog/2013/07/web-sites-vs-web-apps-what-the-experts-think/.

[4] M. Facemire, T. Schadler and J. C. McCarthy, “Mobile Needs A Four-Tier Engagement Platform,” *Forrester Research,* October 2013.

[5] “The WIS architecture components of its main group,” [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/5/51/Overview_of_a_three-tier_application_vectorVersion.svg/2000px-Overview_of_a_three-tier_application_vectorVersion.svg.png. [Accessed 29 03 2016].

[6] J. Burke, “What is the difference between OSI model and TCP/IP other than the number of layers?,” [Online]. Available: http://searchnetworking.techtarget.com/answer/What-is-the-difference-between-OSI-model-and-TCP-IP-other-than-the-number-of-layers.

[7] “TCP/IP vs OSI layers comparison,” [Online]. Available: http://i.stack.imgur.com/CG9lg.jpg.

[8] “TCP/IP vs OSI protocol geography comparison,” [Online]. Available: http://blog.buildingautomationmonthly.com/wp-content/uploads/2013/05/TCP-Protocols.png.

[9] “OSI model applicability example,” [Online]. Available: http://www.escotal.com/Images/Network%20parts/osi.gif.

[10] “OSI model data encapsulation,” [Online]. Available: http://cdn.routemybrain.com/wp-content/uploads/2010/04/encapsulation.jpg.

[11] P. Beynon-Davies, Database systems, Macmillan, 2000.

[12] “Relational Database Fundamentals,” [Online]. Available: https://quizlet.com/64682173/cjv1-lesson-02-relational-database-fundamentals-flash-cards/.

[13] “Introduction to database design,” [Online]. Available: http://www.datanamic.com/support/lt-dez005-introduction-db-modeling.html.

[14] “Entities: types of information,” [Online]. Available: http://www.datanamic.com/support/lt-dez005-introduction-db-modeling.html.

[15] “Entities with attributes,” [Online]. Available: http://www.datanamic.com/supimg/dez005-attributes.jpg.

[16] C. Borysowich, “Understanding Relationships in E-R Diagrams,” 6 February 2007. [Online]. Available: http://it.toolbox.com/blogs/enterprise-solutions/understanding-relationships-in-er-diagrams-14310.

[17] “Relationships between the entities,” [Online]. Available: http://www.datanamic.com/supimg/dez005-examplefirstrelationships.jpg.

[18] “Relationships between the entities with redundancy,” [Online]. Available: http://www.datanamic.com/supimg/dez005-redundantrelationships.jpg.

[19] D. Simões, Método Entidade-Relacionamento, Universidade de Aveiro, 2012.

[20] “Primary keys and foreign keys,” [Online]. Available: http://www.datanamic.com/supimg/dez005-modelwithpksandfks.jpg.

[21] “Data model displaying data types,” [Online]. Available: http://www.datanamic.com/supimg/dez005-modelwithdatatypes.jpg.

[22] “Not in 1st normal form,” [Online]. Available: http://www.datanamic.com/supimg/dez005-normalization1nf_1.jpg.

[23] “In accordance with 1st normal form,” [Online]. Available: http://www.datanamic.com/supimg/dez005-normalization1nf_2.jpg.

[24] “Not in 2nd normal form,” [Online]. Available: http://www.datanamic.com/supimg/dez005-normalization2nf_1.jpg.

[25] “In accordance with 2nd normal form,” [Online]. Available: http://www.datanamic.com/supimg/dez005-normalization2nf_2.jpg.

[26] “Not in 3rd normal form,” [Online]. Available: http://www.datanamic.com/supimg/dez005-normalization3nf_1.jpg.

[27] “In accordance with 3rd normal form,” [Online]. Available: http://www.datanamic.com/supimg/dez005-normalization3nf_2.jpg.

[28] “Data model in accordance with 1st, 2nd and 3d normal form,” [Online]. Available: http://www.datanamic.com/supimg/dez005-modelwithmanufacturers.jpg.

[29] M. Rouse, “Definition - Database Management System,” [Online]. Available: http://searchsqlserver.techtarget.com/definition/database-management-system.

[30] “Data Definition Language (DDL),” [Online]. Available: https://www.techopedia.com/definition/1175/data-definition-language-ddl.

[31] “SQL - Overview,” Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/sql/sql-overview.htm.

[32] “SQL - Syntax,” Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/sql/sql-syntax.htm.

[33] “SQL - RDBMS Concepts,” Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm.

[34] M. Chapple, “Data Manipulation Language,” [Online]. Available: http://databases.about.com/od/sql/a/sqlfundamentals_3.htm.

[35] D. M. L. a. D. Q. Language. [Online]. Available: http://www.cs.toronto.edu/~nn/csc309-20085/guide/pointbase/docs/html/htmlfiles/dmlDql.html.

[36] M. Chapple, “Data Control Language (DCL),” [Online]. Available: http://databases.about.com/od/sql/a/What-Is-Sql.htm.

[37] V. Beal, “Data dictonary,” [Online]. Available: http://www.webopedia.com/TERM/D/data_dictionary.html.

[38] “Modules in a SQL database management system,” [Online]. Available: http://www.tutorialspoint.com/sql/images/sql-architecture.jpg.

[39] “Web Services Glossary,” W3C, 11 February 2004. [Online]. Available: https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice.

[40] N. Chase, “Understanding web services specifications, Part 1: SOAP,” vol. 12, p. May, 2006.

[41] “Web Services - Architecture,” Tutorials Point, [Online]. Available: Web Services - Architecture.

[42] “Web Services architecture model,” [Online]. Available: https://www.ibm.com/support/knowledgecenter/api/content/nl/pt-pt/SSGMCP_5.2.0/com.ibm.cics.ts.webservices.doc/graphics/model.gif.

[43] “What are Web Services?,” Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/webservices/what_are_web_services.htm.

[44] “Web Services - Characteristics,” Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/webservices/web_services_characteristics.htm.

[45] “What is the difference between loose coupling and tight coupling,” [Online]. Available: http://stackoverflow.com/questions/2832017/what-is-the-difference-between-loose-coupling-and-tight-coupling-in-object-orien.

[46] “Coarse-grained vs fine-grained,” [Online]. Available: http://stackoverflow.com/questions/3766845/coarse-grained-vs-fine-grained.

[47] “Web Services - Components,” Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/webservices/web_services_components.htm.

[48] “UDDI technical architecture,” [Online]. Available: http://www.tutorialspoint.com/uddi/images/uddi_arch.gif.

[49] “RESTful Web Services - Introduction,” Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/restful/restful_introduction.htm.

[50] “RESTful Web services: The basics,” IBM developerWorks, [Online]. Available: http://www.ibm.com/developerworks/library/ws-restful/.

[51] Overview Developers, “A Developer's First Look At Android,” *Linux For You,* pp. 48-50, January 2008.

[52] “Android system architecture,” [Online]. Available: http://elinux.org/images/c/c2/Android-system-architecture.jpg.

[53] H. T. AlRaye, “Studying main differences between android & studying main differences between android & linux operating systems,” *International Journal of Electrical & Computer Sciences,* p. 12, 2012.

[54] Tutorials Point, “Android Application Components,” [Online]. Available: http://www.tutorialspoint.com/android/android_application_components.htm.

[55] Tutorials Point, “Android Activities,” [Online]. Available: http://www.tutorialspoint.com/android/android_acitivities.htm.

[56] Developer Android, “Activity Lifecycle,” [Online]. Available: http://developer.android.com/guide/components/activities.html#Lifecycle.

[57] Developer Android, “Activity Lyfecycle model,” [Online]. Available: http://developer.android.com/images/activity_lifecycle.png.

[58] Tutorials Point, “Android User Interface Layout,” [Online]. Available: http://www.tutorialspoint.com/android/android_user_interface_layouts.htm.

[59] Developer Android, “Linear Layout,” [Online]. Available: http://developer.android.com/guide/topics/ui/layout/linear.html.

[60] Developer Android, “Relative Layout,” [Online]. Available: http://developer.android.com/guide/topics/ui/layout/relative.html.

[61] Developer Android, “ListView,” [Online]. Available: http://developer.android.com/guide/topics/ui/layout/listview.html.

[62] Developer Android, “GridView,” [Online]. Available: http://developer.android.com/guide/topics/ui/layout/gridview.html.

[63] Developer Android, “UI controls,” [Online]. Available: http://developer.android.com/guide/topics/ui/controls.html.

[64] Tutorials Point, “Android User Interface Controls,” [Online]. Available: http://www.tutorialspoint.com/android/android_user_interface_controls.htm.

[65] “DNS query,” [Online]. Available: http://info.globalit.com/wp-content/uploads/2013/05/DNS-Query-Diagram-HQ.jpg.

[66] “PHP calculation,” [Online]. Available: http://www.javascript-coder.com/wp-content/uploads/2010/07/web-form-working.png.

[67] W3Schools, [Online]. Available: http://www.w3schools.com/js/default.asp.

[68] “What is HTML?,” [Online]. Available: http://www.yourhtmlsource.com/starthere/whatishtml.html.

[69] W3Schools, “HTML Introduction,” [Online]. Available: http://www.w3schools.com/html/html_intro.asp.

[70] W3Schools, “CSS Introduction,” [Online]. Available: http://www.w3schools.com/css/css_intro.asp.

[71] Brainbell, “CSS Theory Simplified,” [Online]. Available: http://www.brainbell.com/tutorials/HTML_and_CSS/CSS_Theory_Simplified.htm.

[72] “JavaScricpt Introduction,” [Online]. Available: https://developer.mozilla.org/pt-PT/docs/Web/JavaScript/Guide/Introduction.

[73] S. Chapman, “What is JavaScript?,” [Online]. Available: http://javascript.about.com/od/reference/p/javascript.htm.

[74] M. Fowler, UML Distilled, Third Edition, Addison-Wesley, 2004.

[75] Smartdraw, “UML Diagram,” [Online]. Available: https://wcs.smartdraw.com/uml-diagram/.

[76] “Package diagram,” [Online]. Available: https://wcs.smartdraw.com/uml-diagram/img/package-diagram.jpg.

[77] R. Miller, “Practical UML™: A Hands-On Introduction for Developers,” [Online]. Available: http://edn.embarcadero.com/article/31863.

[78] “Sequence diagram,” [Online]. Available: https://wcs.smartdraw.com/uml-diagram/img/sequence-diagram.jpg.

[79] “Use case diagram,” [Online]. Available: http://edn.embarcadero.com/article/images/31863/actorsmultipleno3d.gif.

[80] “State diagram,” [Online]. Available: http://edn.embarcadero.com/article/images/31863/statediagno3d.gif.

[81] H. Oriahi, “Content Management Sustems (CMS),” Oulu, 2014.

[82] B. Boiko, “Understanding Content Management,” Metatorial Services Inc. & HungryMinds Inc., 2002.

[83] [Online]. Available: https://pt.wikipedia.org/wiki/Sistema_de_gerenciamento_de_conte%C3%BAdo.

[84] “Content Management System,” 06 2016. [Online]. Available: http://searchsoa.techtarget.com/definition/content-management-system.

[85] D. Michelinakis, “Open Source Content Management Systems:,” 2004.

[86] [Online]. Available: https://en.wikipedia.org/wiki/WordPress.

[87] WordPress, [Online]. Available: https://wordpress.org/about/requirements/.

[88] [Online]. Available: https://libroediting.files.wordpress.com/2015/11/2-old-admin-page.png.

[89] [Online]. Available: https://en.wikipedia.org/wiki/Joomla.

[90] [Online]. Available: https://en.wikipedia.org/wiki/Drupal.

[91] [Online]. Available: https://www.drupal.org/requirements.

[92] [Online]. Available: http://i1-news.softpedia-static.com/images/news2/drupal-8-released-here-s-a-list-of-all-the-new-features-496447-4.jpg.

[93] [Online]. Available: https://s-media-cache-ak0.pinimg.com/736x/bd/6a/5b/bd6a5b68af0804d4873cd1f8d3300847.jpg.

[94] “Share of Android platforms on mobile devices with Android OS,” 2016. [Online]. Available: http://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/.

[95] “Component Creator,” [Online]. Available: https://www.component-creator.com.

[96] “WSDL - Example,” Tutorials Point, [Online]. Available: http://www.tutorialspoint.com/wsdl/wsdl_example.htm.

Appendix

Practical Example

In this chapter it is given a practical example of implementation of a Web Information System.

The example is composed by an Apache server, a database that uses MySQL as the management system, RESTful Web Services designed in PHP language and an Android application.

As development environment for the server, database and Web Services development it can be used Linux, Windows or OS X as operative system. Although a LAMP server approach is recommended, a WAMP (Windows, Apache, MySQL and PHP) or MAMP (Macintosh, Apache, MySQL and PHP) can also be done and is an equivalent approach.

Android Studio is the recommended IDE (Integrated Development Environment) for developing Android applications and it can be installed in the major operative systems.

Install guides for the tools and environments are available as annex.

The objective of this example is to create a database for a restaurant. The restaurant database will have a table in which it is kept the information of its clients. The information in the table is then handled in an Android application, where it is possible to add new clients (insert), visualize the clients in the table (select), edit client information (update) and delete a client (delete).

MySQL Database

After installing MySQL it's time to access it in order to create a database with tables for the example. Open the command line in your development environment and type the following (Important: the command lines in the guide are oriented for an UNIX environment and there might be some differences comparing to Windows environments):

```
$ mysql –u root –p
Enter password:
```

Insert MySQL root password when prompted. After that it's time to create the database for the restaurant. The following first command creates the database while the second shows all databases in the system.

```
mysql> CREATE DATABASE restaurant;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| restaurant         |
| events             |
| mysql              |
| performance_schema |
+--------------------+
6 rows in set (0.00 sec)
```

Now it's time to create the table in the database which will be manipulated, via Web Services, in the Android application. It is a table to keep track of the restaurant clients so let's call clients.

The attribute id will be the Primary Key for each instance of the entity client, and it will be an auto-incremental variable, which means that there is no need to define it in each creation of an instance. The attribute invoice is a character that will keep track if the client wants an invoice or not, and it is supposed to have value ‘y’ or ‘n’, as in “yes” or “no”. The remaining attributes are all VARCHAR type which correspond to the usual String type.

The following instructions select the restaurant database to use, creates the clients table with its attributes, show the tables in the restaurant database and describe the clients table.

```
mysql> USE restaurant;
Database changed

mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql> CREATE TABLE clients (id INT NOT NULL PRIMARY KEY
       AUTO_INCREMENT, name VARCHAR(30), type VARCHAR(10), invoice
       CHAR(1), tax_nr VARCHAR(15), address VARCHAR(50), contact
       VARCHAR(15));
Query OK, 0 rows affected (0.07 sec)

mysql> SHOW TABLES;
 +--------------------+
| Tables_in_clients |
+--------------------+
| restaurant         |
+--------------------+
1 row in set (0.00 sec)

mysql> DESCRIBE restaurant;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int(11)     | NO   | PRI | NULL    | auto_increment |
| name    | varchar(30) | YES  |     | NULL    |                |
| type    | varchar(10) | YES  |     | NULL    |                |
| invoice | char(1)     | YES  |     | NULL    |                |
| tax_nr  | varchar(15) | YES  |     | NULL    |                |
| address | varchar(50) | YES  |     | NULL    |                |
| contact | varchar(15) | YES  |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

RESTful Web Services

Now that the database is created it is time to access create the means to access and manipulate it from a remote location. It is possible to enable the remote manipulation by generating simple PHP scripts that use RESTful Web Services to provide basic CRUD (Create, Read, Update and Delete) operations to the database.

The following instructions are specified to the Ubuntu 15.04 environment, but can easily be adapted for other environments. Also, it is expected at this stage to have every required package installed by following the guidelines in annex.

Let's start by modifying the permissions on the default Web pages' directory. This will allow access from an external location. After that let's verify the permissions in the www folder. The following instructions are made in superuser mode, so insert the password when prompted.

```
$ sudo chmod -R 755 /var/www
$ sudo ls -l /var/
total 48
drwxr-xr-x  2 root root     4096 Fev  4 07:57 backups
drwxr-xr-x 19 root root     4096 Fev  3 11:30 cache
drwxr-xr-x 73 root root     4096 Fev  3 22:22 lib
drwxrwsr-x  2 root staff    4096 Abr 17  2015 local
drwxrwxr-x 15 root syslog   4096 Fev  8 07:48 log
drwxrwsr-x  2 root mail     4096 Abr 22  2015 mail
drwxr-xr-x  2 root root     4096 Abr 22  2015 opt
drwxr-xr-x  8 root root     4096 Abr 22  2015 spool
drwxrwxrwt  3 root root     4096 Fev  8 17:50 tmp
drwxr-xr-x  3 root root     4096 Fev  3 11:30 www
```

In order to keep the everything organized, it will be created a directory in the www folder for the project that it's been created. This directory will be called restaurantDB and will held every PHP scripts that enable access and manipulation to the restaurant database.

```
$ sudo mkdir /var/www/html/restaurantDB
```

The first method that is necessary is the method that connects to the database. In this method it is needed to input the database name, the user name and its password. The method will be named connection.php, and the following instruction creates and opens it:

```
$ sudo nano /var/www/html/restaurantDB/connection.php
```

The connection.php file must be filled with the following PHP code lines, where it is necessary to replace the "mysqlrootpassword" field. This script uses an improved MySQL extension for PHP, which is mysqli.

```
<?php
$connection = new mysqli("localhost", "root", "mysqlrootpassword",
"restaurant") or die(mysqli_error());
?>
```

Next it will be built a method for making inserts in the clients table from the restaurant database. At first create a file named insert.php and open it. Then use the code lines above to edit the file.

In the code lines it can be seen several arguments getting their value from the body of a POST request, but first it is necessary to include the connection.php script in order to connect to the database. Next a MySQL instruction is created using the arguments received in the POST request and then submitted to the database.

```
$ sudo nano /var/www/html/restaurantDB/insert.php

<?php
include 'connection.php';
$name = $_POST['name'];
$type = $_POST['type'];
$invoice = $_POST['invoice'];
$tax_nr = $_POST['tax_nr'];
$address = $_POST['address'];
$contact = $_POST['contact'];

$sql = "INSERT INTO `restaurant`.`clients` (`id`, `name`, `type`,
`invoice`, `tax_nr`, `address`, `contact`) VALUES (NULL, '$name',
'$type', '$invoice', '$tax_nr', '$address', '$contact');";

if ($connection->query($sql)) {
$msg = array("status" =>1 , "msg" => "Your record inserted
successfully");
} else {
echo "Error: " . $name . "<br>" . mysqli_error($connection);
}

$json = $msg;

header('content-type: application/json');
echo json_encode($json);

@mysqli_close($connection);

?>
```

The following step is to build a script that allows to visualize the inputs that were made in the clients table. It is necessary to create and then edit a new file called select.php, where it will be made an instruction to send to the MySQL database. This instruction selects every input in the clients table.

```
$ sudo nano /var/www/html/restaurantDB/select.php

<?php
include 'connection.php';

$getData = "select * from clients";
$qur = $connection->query($getData);

while($r = mysqli_fetch_assoc($qur)){

$msg[] = array("id" => $r['id'], "name" => $r['name'], "type" =>
$r['type'], "invoice" => $r['invoice'], "tax_nr" => $r['tax_nr'],
"address" => $r['address'], "contact" => $r['contact']);
}
$json = $msg;
header('content-type: application/json');

echo json_encode($json);

@mysqli_close($connection);
?>
```

Then it is also required an update.php script that enables updating the information of an input in the clients table. This script works similarly to the insert.php script, except that the user needs to know the id of the input that he wants to update and send it as a parameter in the POST request.

Let’s start by creating the update.php file and then open it. Then edit the file with the code lines below.

```
$ sudo nano /var/www/html/restaurantDB/update.php

<?php
include 'connection.php';

$name = $_POST['name'];
$type = $_POST['type'];
$invoice = $_POST['invoice'];
$tax_nr = $_POST['tax_nr'];
$address = $_POST['address'];
$contact = $_POST['contact'];
$id = $_POST['id'];

$query = "UPDATE `clients` SET `name`='$name', `address`='$address',
`type`='$type', `invoice`='$invoice', `tax_nr`='$tax_nr',
`contact`='$contact'  WHERE  `id`='$id'";
if ($connection->query($query)) {
       $msg = array("status" => 1 , "msg" => "Record Updated
successfully");
}else {
    echo "Error: " . $query . "<br>" . mysqli_error($connection);
}

$json = $msg;

header('content-type: application/json');
echo json_encode($json);

@mysqli_close($connection);
?>
```

The last method necessary to create is the delete method. Deleting will be made by sending as parameter the id of the input in the clients table that is required to delete, through a POST request.

Create a file named delete.php and editi it with the following code lines.

```
$ sudo nano /var/www/html/restaurantDB/delete.php

<?php
include 'connection.php';

$id = $_POST['id'];

$query = "DELETE FROM `clients` WHERE   `id`='$id'";
if ($connection->query($query)) {
    $msg = array("status" =>1 , "msg" => "Record Deleted successfully");
} else {
    echo "Error: " . $query . "<br>" . mysqli_error($connection);
}

$json = $msg;

header('content-type: application/json');
echo json_encode($json);

@mysqli_close($connection);
?>
```

After every file is created and configured it is necessary to configure the listening ports in the Apache server, forward the ports in the network router to allow access from a remote location

and testing the Web Services using a REST client. These can all be done by following the guidelines available in annex.

Android application

In this part of the example it is assumed that the Android Studio IDE is already installed and that the user either has an Android smartphone or that an AVD (Android Virtual Device) is configured, as described in annex.

Let's start by opening Android Studio and creating a new Android Studio project. The application will be called RestaurantDB and the company domain praticalexample.com, according to the database. A configuration example is show in Figure 64.

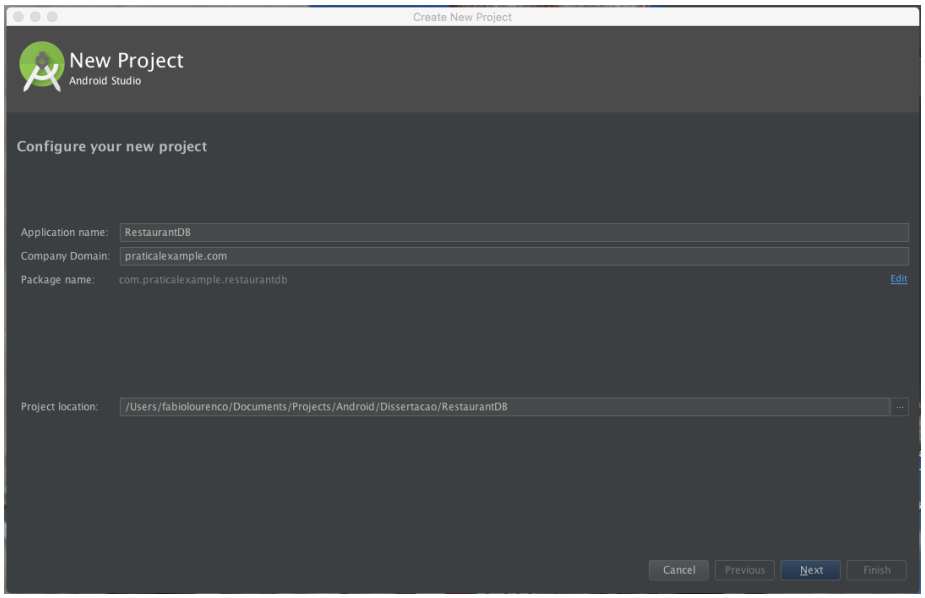


Figure 64 – New project on Android Studio

In the next window it is necessary to choose the type of device and the minimum Android SDK version in which the application is supposed to work properly. For now, let's choose Phone and Tablet, using the API 19 that corresponds to Android 4.4 Kitkat version (Figure 65).

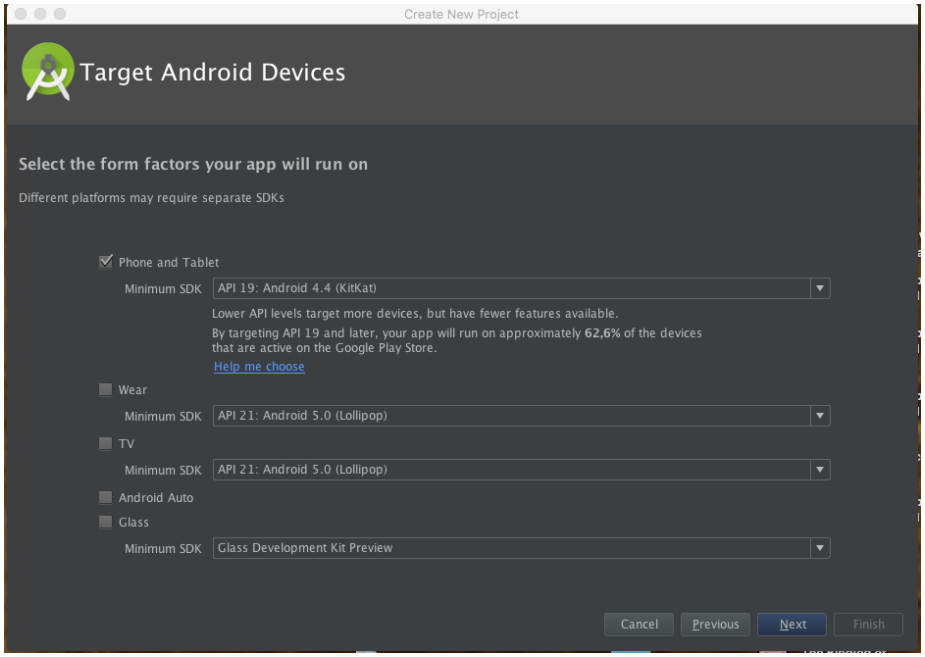


Figure 65 – Definitions of new project in Android Studio

Then it is time to choose the main activity of the application. Let's choose an Empty Activity and hit Next. Then edit the name of the activity to MenuActivity and the name of the layout to activity_menu_layout, as in Figure 66.

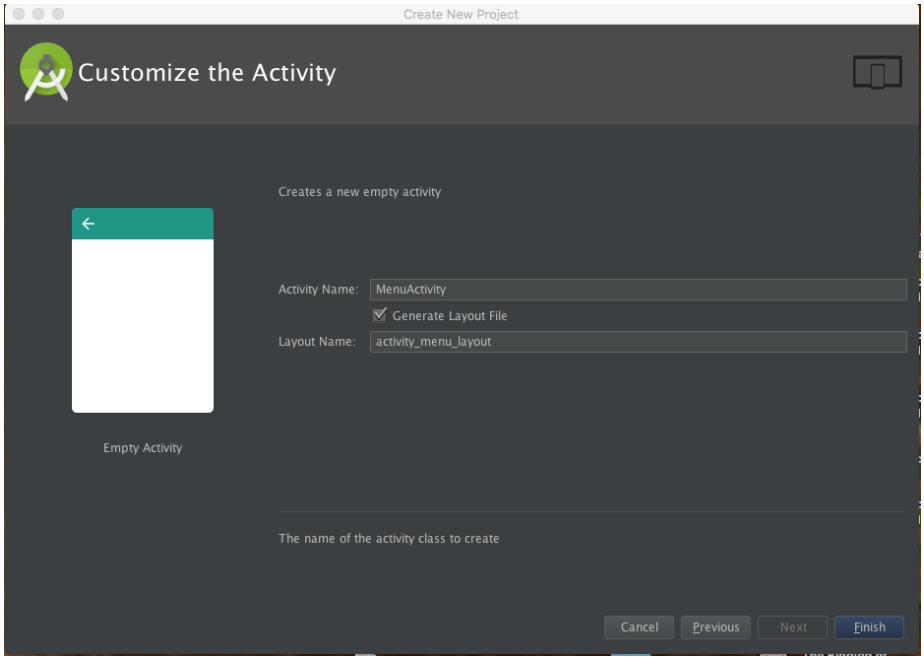


Figure 66 – Creating the Menu activity for the application

After the project is generated the next step is to create a package (similar to a directory) that will held the Activity files that are related to the User Interface. Let's make a right click in the com.praticalexample.restaurantDB package and create a new package named UI, as in Figure 67. Then let's drag the MenuActivity file to the UI package.

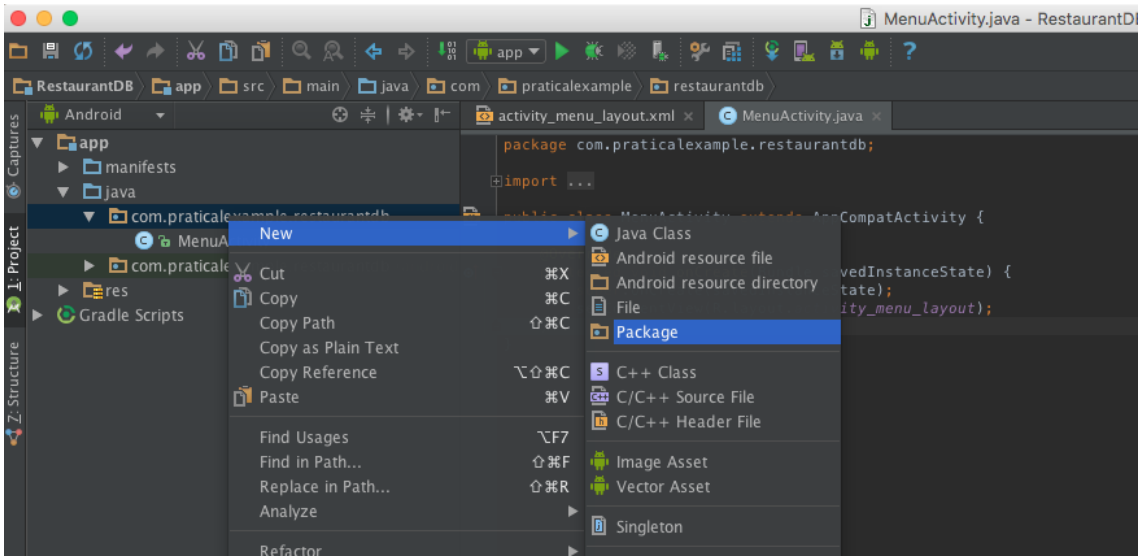


Figure 67 – Creating a new package

Now it's time to design the User Interface for the Menu activity. So open the activity_menu_layout.xml file that resides in the res/layout directory. Delete its content and replace it with the code lines displayed below.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UI.MenuActivity"
    android:orientation="vertical"
    android:weightSum="2"
    android:id="@+id/menuLinearLayout">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:paddingBottom="10dp"
        android:id="@+id/menuNewClientRelLayout">

        <Button
            android:layout_width="200dp"
            android:layout_height="70dp"
            android:id="@+id/menuNewClientButton"
            android:layout_centerHorizontal="true"
            android:layout_alignParentBottom="true"
            android:text="New Client"
            android:textSize="20sp" />
    </RelativeLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:paddingTop="10dp"
        android:id="@+id/menuQueryClientsRelLayout">

        <Button
            android:layout_width="200dp"
            android:layout_height="70dp"
            android:id="@+id/menuQueryClientsButton"
            android:layout_centerHorizontal="true"
            android:layout_alignParentTop="true"
            android:text="Query Clients"
            android:textSize="20sp" />
    </RelativeLayout>
</LinearLayout>
```

These code lines originate a layout with two buttons, centered in the screen, one to insert a new client in the database table and other to query the inputs in the clients table. Figure 68 displays the layout created above.

The LinearLayout is the main element for this activity graphical design and it has two child elements, two RelativeLayouts (each having a Button element as child element). Instead of assigning fixed values to the height of each child RelativeLayout, it was assigned a weight of 1 to each of then and a weight sum of 2 to the parent LinearLayout. This means that each RelativeLayout has the same weight, which means that they’re equally divided in the partent layout.

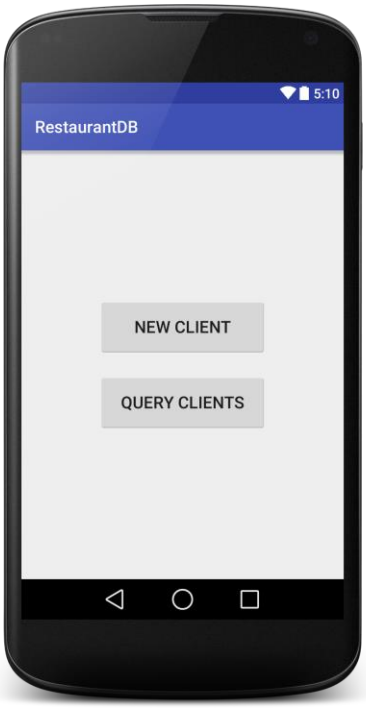


Figure 68 – Application menu activity layout

The next step is to assign the layout to the MenuActivity, referencing its file and adding its elements and their behaviour when an action is triggered (for example, what happens when the New Client button is pressed.

Open the MenuActivity file and replace its content with the content above exposed. When the New Client button is clicked a new intent is created with the objective to open a new activity (to be built ahead). The same happens with the Query Clients button.

```
package com.praticalexample.restaurantdb.UI;

import android.content.Context;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import com.praticalexample.restaurantdb.R;

public class MenuActivity extends AppCompatActivity {

    private Button newClientButton, queryClientsButton;
    private Context mContext;
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu_layout);

        mContext = getApplicationContext();

        newClientButton = (Button)
findViewById(R.id.menuNewClientButton);
        queryClientsButton = (Button)
findViewById(R.id.menuQueryClientsButton);

        newClientButton.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                Intent newClientIntent = new Intent(mContext,
NewClientActivity.class);
                startActivity(newClientIntent);
            }
        });

        queryClientsButton.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                Intent queryClientsIntent = new Intent(mContext,
QueryClientsActivity.class);
                startActivity(queryClientsIntent);
            }
        });
    }
}
```

Now add a new activity inside de UI package, clicking with the right mouse button in the package given by com.praticalexample.restaurantdb.UI then clicking New, Activity and finally choosing Empty Activity. Name the activity NewClientActivity and its layout activity_new_client_ layout.

Open the activity_new_client_ layout.xml located in the res/layout directory and replace its content by the XML code lines below.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".UI.NewClientActivity"
    android:weightSum="8"
    android:id="@+id/newClientLinearLayout"
    android:padding="5dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:id="@+id/newClientNameLinearLayout"
        android:orientation="horizontal"
        android:weightSum="4">

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Name:"
            android:id="@+id/newClientNameTextView"
            android:layout_weight="1"
            android:textAlignment="center"
            android:layout_gravity="center"
            android:textStyle="bold" />

        <EditText
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:inputType="textPersonName"
            android:ems="10"
            android:id="@+id/newClientNameEditText"
            android:layout_weight="3"
            android:layout_gravity="center_vertical" />
```

```
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:id="@+id/newClientTypeLinearLayout"
            android:orientation="horizontal"
            android:weightSum="4">

            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Type:"
                android:id="@+id/newClientTypeTextView"
                android:layout_weight="1"
                android:layout_gravity="center"
                android:textStyle="bold"
                android:textAlignment="center" />

            <Spinner
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:id="@+id/newClientTypeSpinner"
                android:layout_weight="3"
                android:layout_gravity="center_vertical"
                android:spinnerMode="dropdown" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:id="@+id/newClientInvoiceLinearLayout"
            android:orientation="horizontal"
            android:weightSum="4">

            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Invoice?"
                android:id="@+id/newClientInvoiceTextView"
                android:layout_gravity="center_vertical"
                android:layout_weight="1"
                android:textStyle="bold"
                android:textAlignment="center" />

            <RadioGroup
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="3"
                android:id="@+id/newClientInvoiceRadioGroup"
                android:orientation="horizontal"
                android:layout_gravity="center_vertical"
                android:weightSum="2">

                <RadioButton
    android:id="@+id/newClientInvoiceYesRadioButton"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:text="Yes"
                    android:onClick="onRadioButtonClicked"
                    android:layout_weight="1"
                    android:checked="true" />
```

```
            <RadioButton android:id="@+id/newClientInvoiceNoRadioButton"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:text="No"
                android:onClick="onRadioButtonClicked"
                android:layout_weight="1" />

        </RadioGroup>

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:id="@+id/newClientTaxNRLinearLayout"
        android:orientation="horizontal"
        android:weightSum="4">

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Tax nr:"
            android:id="@+id/newClientTaxNRTextView"
            android:layout_weight="1"
            android:textStyle="bold"
            android:textAlignment="center"
            android:layout_gravity="center" />

        <EditText
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:inputType="number"
            android:ems="10"
            android:id="@+id/newClientTaxNREditText"
            android:layout_weight="3"
            android:layout_gravity="center_vertical" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp" android:layout_weight="1"
        android:id="@+id/newClientAddressLinearLayout"
        android:orientation="horizontal"
        android:weightSum="4">

        <TextView
            android:layout_width="0dp"
android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Address:"
            android:id="@+id/newClientAddressTextView"
            android:layout_weight="1"
            android:textStyle="bold"
            android:textAlignment="center"
            android:layout_gravity="center" />

        <EditText
            android:layout_width="0dp"
android:layout_height="wrap_content"
            android:inputType="textPostalAddress"
            android:ems="10"
            android:id="@+id/newClientAddressEditText"
            android:layout_weight="3"
            android:layout_gravity="center_vertical" />
    </LinearLayout>
```

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp" android:layout_weight="1"
        android:id="@+id/newClientContactLinearLayout"
        android:orientation="horizontal"
        android:weightSum="4">

        <TextView
            android:layout_width="0dp"
android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Contact:"
            android:id="@+id/newClientContactTextView"
            android:layout_weight="1"
            android:textStyle="bold"
            android:textAlignment="center"
            android:layout_gravity="center" />

        <EditText
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:inputType="phone"
            android:ems="10"
            android:id="@+id/newClientContactEditText"
            android:layout_weight="3"
            android:layout_gravity="center_vertical" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp" android:layout_weight="2"
        android:id="@+id/newClientButtonsLinearLayout"
        android:orientation="horizontal"
        android:weightSum="2">

        <RelativeLayout
            android:layout_width="0dp"
android:layout_height="match_parent"
            android:layout_weight="1"
            android:id="@+id/newClientSubmitRelLayout">

            <Button
                style="?android:attr/buttonStyleSmall"
                android:layout_width="100dp"
android:layout_height="wrap_content"
                android:text="Submit"
                android:id="@+id/newClientSubmitButton"
                android:layout_centerInParent="true" />
        </RelativeLayout>

        <RelativeLayout
            android:layout_width="0dp"
android:layout_height="match_parent"
            android:layout_weight="1"
            android:id="@+id/newClientCancelRelLayout">

            <Button
                style="?android:attr/buttonStyleSmall"
                android:layout_width="100dp"
android:layout_height="wrap_content"
                android:text="Cancel"
                android:id="@+id/newClientCancelButton"
                android:layout_centerInParent="true" />
        </RelativeLayout>
    </LinearLayout>

</LinearLayout>
```

In the above code lines, it is described the layout for the NewClient avtivity. It contains several elements of the Android interface that can be interesting to learn how to work with them. Elements like Submit and Cancel buttons to submit a new client to the database, or cancel it, a Spinner to select the type of client from two given types, radio group compose by two Radio buttons to select whether the client wants invoice or not, and Edit text elements to input information from using the keyboard. Figure 69 represents the layout for the NewClient activity.

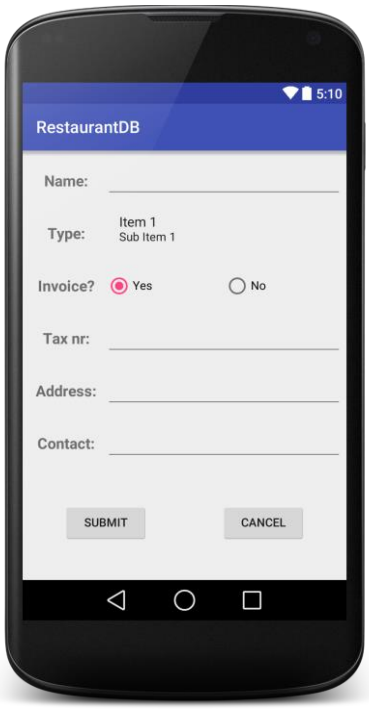


Figure 69 – Application New Client layout

Now open the file NewClientActivity.java and paste de code lines below. This activity is responsible for assigning the activity_new_client_layout.xml to itself, assign the Spinner element information (so that the user can choose one of its options), process the data inputted by the user in the Edit Text elements and the options chosen in the Radio Group and the Spinner elements and send it to the server using the previously built API's.

To send the information to the server using the RESTful API's it is necessary to use an Android library called Volley. This library will be included on a forward part.

```
package com.praticalexample.restaurantdb.UI;

import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Spinner;
import android.widget.Toast;

import com.android.volley.AuthFailureError;
import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
```

```
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import com.praticalexample.restaurantdb.R;

import org.json.JSONObject;

import java.lang.reflect.Array;
import java.util.HashMap;
import java.util.Map;

public class NewClientActivity extends AppCompatActivity {

    private EditText nameEditText, taxNREditText, addressEditText,
contactEditText;
    private Spinner typeSpinner;
    private Button submitButton, cancelButton;
    private ArrayAdapter<String> typeSpinnerAdapter;

    private boolean invoice = true, typeIsPrivate = true;
    private String name, taxNR, address, contact;

    private String[] strings = {
            "Private",
            "Company"
    };

    private RequestQueue queue;
    private Context mContext;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_client_layout);
        mContext = getApplicationContext();
        queue = Volley.newRequestQueue(this);

        nameEditText = (EditText)
findViewById(R.id.newClientNameEditText);
        taxNREditText = (EditText)
findViewById(R.id.newClientTaxNREditText);
        addressEditText = (EditText)
findViewById(R.id.newClientAddressEditText);
        contactEditText = (EditText)
findViewById(R.id.newClientContactEditText);
        typeSpinner = (Spinner) findViewById(R.id.newClientTypeSpinner);
        submitButton = (Button)
findViewById(R.id.newClientSubmitButton);
        cancelButton = (Button)
findViewById(R.id.newClientCancelButton);

        // Create an ArrayAdapter using the string array and a default
spinner layout
        typeSpinnerAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, strings);
        // Specify the layout to use when the list of choices appears

typeSpinnerAdapter.setDropDownViewResource(android.R.layout.simple_spinn
er_dropdown_item);
        // Apply the adapter to the spinner
        typeSpinner.setAdapter(typeSpinnerAdapter);

        typeSpinner.setOnItemSelectedListener(new
```

```
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                typeIsPrivate =
parent.getItemAtPosition(position).toString().equals("Private");
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {}
        });

        submitButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                name = nameEditText.getText().toString();
                taxNR = taxNREditText.getText().toString();
                address = addressEditText.getText().toString();
                contact = contactEditText.getText().toString();

                addNewClient();
                finish();
            }
        });

        cancelButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }

    public void onRadioButtonClicked(View view) {
        // Is the button now checked?
        boolean checked = ((RadioButton) view).isChecked();

        // Check which radio button was clicked
        switch(view.getId()) {
            case R.id.newClientInvoiceYesRadioButton:
                if (checked)
                    invoice = true;
                break;
            case R.id.newClientInvoiceNoRadioButton:
                if (checked)
                    invoice = false;
                break;
        }
    }

    private void addNewClient() {
        // Prepare the request
        String requestURL =
"http://94.60.154.138:1200/restaurantDB/insert.php";
        final Map<String, String> bodyParams = new HashMap<String,
String>();

        // Fill body parameters
        bodyParams.put("name", name);
        bodyParams.put("type", typeIsPrivate ? "Private" : "Company");
        bodyParams.put("invoice", invoice ? "y" : "n");
        bodyParams.put("tax_nr", taxNR);
        bodyParams.put("address", address);
        bodyParams.put("contact", contact);

        // Request a string response from the provided URL.
        StringRequest stringRequest = new
```

```
StringRequest(Request.Method.POST, requestURL, new
Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                VolleyLog.d("Response: %s", response.toString());
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                VolleyLog.e("Error: %s",
error.getClass().getSimpleName());
            }
        }){
            @Override
            protected Map<String, String> getParams() throws
AuthFailureError {
                return bodyParams;
            }
        };

        // establish a sufficient timeout value
        stringRequest.setRetryPolicy(new DefaultRetryPolicy(10000,
                DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
                DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));

        // Add the request to the RequestQueue.
        queue.add(stringRequest);
    }
}
```

In order to add the Volley library to Android Studio it is necessary to make some actions. First open the Terminal console in the Android Studio IDE and paste the following instruction:

```
git clone https://android.googlesource.com/platform/frameworks/volley
```

Then select File -> New -> Import Module... and select the volley directory in the root directory of the project. After adding the module open the build.gradle(Module: app) file located in the Gradle Scripts section of the project. Add the following code line under dependencies (as in Figure 70): `compile project (':volley')`. Make sure that in the settings.gradle file the is the volley input like the following: `include ':app', ':volley'`.

After editing the code line click on Sync Now.

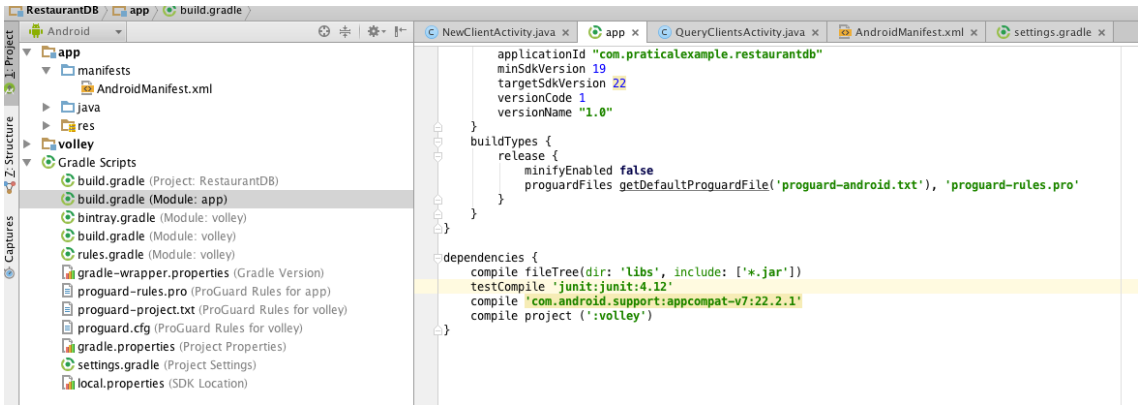


Figure 70 – Adding the Volley library

It is necessary to add permissions to the application manifest in order to allow the application to access the Internet. So open the AndroidManifest.xml file located at the manifests folder and add the following line below the package input.

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

Let's add a new activity inside de UI package, clicking with the right mouse button in the package given by com.praticalexample.restaurantdb.UI then clicking New, Activity and finally choosing Empty Activity. Name the activity QueryClientsActivity and its layout activity_query_clients_ layout.

The layout that this activity will have is displayed at Figure 71.

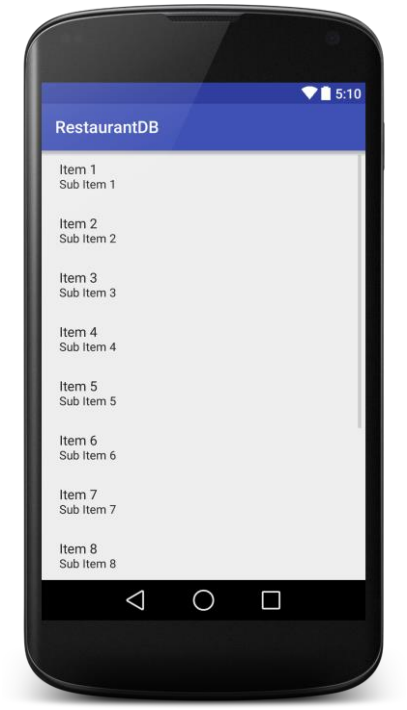


Figure 71 – Application Query Clients layout

Open the activity_query_clients_ layout.xml located in the res/layout directory and replace its content by the XML code lines below. This layout file only creates a ListView, which will be filled with items described in another layout file.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="5dp"
    tools:context=".UI.QueryClientsActivity">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/queryClientsListView" />
</RelativeLayout>
```

In order to fill the ListView it is necessary to create an Adapter that will take data as input and transform it into an item of the ListView. The items of the list need to have their own layout, that will be associated to the Adapter, and then to the data.

Let's start by creating a Client class to facilitate data exchange between elements. First create a new package called Classes under the com.praticalexample.restaurantdb directory (switch the view, on the top left corner under RestaurantDB, from Android to Project, and in the end return to Android view). Create a new Java Class under the Classes package and name it Client. Replace its code line for the ones below.

```
package com.praticalexample.restaurantdb.Classes;

public class Client {

    private int mId;
    private String mName, mType, mTaxNR, mAddress, mContact;
    private boolean mInvoice;

    public Client(int id, String name, String type, boolean invoice,
String taxNR, String address,
                  String contact) {
        mId = id;
        mName = name;
        mType = type;
        mInvoice = invoice;
        mTaxNR = taxNR;
        mAddress = address;
        mContact = contact;
    }

    public int getId() {
        return mId;
    }

    public String getName() {
        return mName;
    }

    public String getType() {
        return mType;
    }

    public boolean getInvoice() {
        return mInvoice;
    }

    public String getTaxNR() {
        return mTaxNR;
    }

    public String getAddress() {
        return mAddress;
    }

    public String getContact() {
        return mContact;
    }
}
```

By analysing this class, it is possible to see that this class contains the attributes of a client as defined in the database and contains methods to access them when an instance of the class is created.

It's now time to create a new layout for the ListView items. Add a new Layout resource file under the res/layout directory and name it query_clients_list_view_item_layout. Open the file

and replace its XML lines with the ones below. These code lines create a layout (represented in Figure 72) with two visible elementes, two TexView fields, one bigger to display the client name and one smaller to display wether the client is private or company.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="match_parent"
    android:layout_height="50dp"
    android:weightSum="10"
    android:id="@+id/queryClientsItemLinearLayout">

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Medium Text"
        android:layout_weight="9"
        android:layout_gravity="center_vertical"
        android:id="@+id/queryClientsItemNameTextView"
        android:textColor="@android:color/black" />

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Medium Text"
        android:layout_weight="1"
        android:layout_gravity="center_vertical"
        android:id="@+id/queryClientsItemTypeTextView" />
</LinearLayout>
```

Figure 72 – Application ListView item layout

Afterwards it is time to make the adapter to associate data to the ListView, which will be assigned to the layout above created. Start by creating a new package named Adapters under the com.praticalexample.restaurantdb directory. After that create a new Java Class named

QueryClientsListAdapter inside that package. Open the file and replace its content with the Java lines below.

```
package com.praticalexample.restaurantdb.Adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.praticalexample.restaurantdb.Classes.Client;
import com.praticalexample.restaurantdb.R;

import java.util.List;

public class QueryClientsListAdapter extends ArrayAdapter<String> {

    private Context mContext;
    private List<Client> mClients;

    public QueryClientsListAdapter(Context context, List<Client>
clients) {
        super(context, R.layout.query_clients_list_view_item_layout);

        mContext = context;
        mClients = clients;
    }

    @Override
    public int getCount() {
        return mClients.size();
    }

    @Override
    public View getView(int position, View view, ViewGroup parent) {

        final ViewHandler handler;

        if (view == null) {
            LayoutInflater inflater = (LayoutInflater)

mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

            view =
inflater.inflate(R.layout.query_clients_list_view_item_layout, parent,
false);

            handler = new ViewHandler();

            handler.nameTextView = (TextView)
view.findViewById(R.id.queryClientsItemNameTextView);
            handler.typeTextView = (TextView)
view.findViewById(R.id.queryClientsItemTypeTextView);

            view.setTag(handler);
        } else {
            handler = (ViewHandler) view.getTag();
        }

        Client client = mClients.get(position);
        handler.nameTextView.setText(client.getName());

        if (client.getType().equalsIgnoreCase("Company")) {
            handler.typeTextView.setText("C");
```

```
handler.typeTextView.setTextColor(mContext.getResources().getColor(andro
id.R.color.holo_blue_dark));
        } else {
            handler.typeTextView.setText("P");

handler.typeTextView.setTextColor(mContext.getResources().getColor(andro
id.R.color.holo_red_dark));
        }

        return view;
    }

    private static class ViewHandler {
        private TextView nameTextView, typeTextView;
    }
}
```

This adapter class constructor receives as one of the parameters the list of clients (represented by a list of Client class elements) that will be saved in a local variable. Then when the getView method gets called, the adapter accesses the saved data and gets a client by position, adds its information to the item TextView elements and returns the view.

Finally, all that remains is replacing the code lines of the activity QueryClients by the code lines below. This activity is assigned to the respective layout, the adapter is to the ListView element, being (the adapter) initialized with an empty list. Afterwards customers' data is obtained by using the API with that purpose, using the Volley library, and updating the adapter information, that places all clients' information arranged in the ListView. Each customer ID is saved for possible future use.

```
package com.praticalexample.restaurantdb.UI;

import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ListView;

import com.android.volley.DefaultRetryPolicy;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.Volley;

import
com.praticalexample.restaurantdb.Adapters.QueryClientsListAdapter;
import com.praticalexample.restaurantdb.Classes.Client;
import com.praticalexample.restaurantdb.R;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class QueryClientsActivity extends AppCompatActivity {
    private ListView queryClientsListView;
    private QueryClientsListAdapter queryClientsListAdapter;

    public List<Client> clientList;
    private RequestQueue queue;
    private Context mContext;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_query_clients_layout);
        mContext = getApplicationContext();
        queue = Volley.newRequestQueue(this);

        queryClientsListView = (ListView)
findViewById(R.id.queryClientsListView);
        clientList = new ArrayList<>();
        queryClientsListAdapter = new QueryClientsListAdapter(mContext,
clientList);
        queryClientsListView.setAdapter(queryClientsListAdapter);

        getClientList();
    }

    private void getClientList() {
        // Prepare the request
        String requestURL =
"http://94.60.154.138:1200/restaurantDB/select.php";

        JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, requestURL,
                null, new Response.Listener<JSONArray>() {
            @Override
            public void onResponse(JSONArray response) {
                VolleyLog.d("Response: %s", response.toString());
                for (int i = 0; i < response.length(); i++) {
                    try {
                        JSONObject jsonObject =
response.getJSONObject(i);
                        int id = jsonObject.getInt("id");
                        String name = jsonObject.getString("name");
                        String type = jsonObject.getString("type");
                        boolean invoice =
jsonObject.getString("invoice").equalsIgnoreCase("y");
                        String taxNR = jsonObject.getString("tax_nr");
                        String address =
jsonObject.getString("address");
                        String contact =
jsonObject.getString("contact");

                        Client client = new Client(id, name, type,
invoice, taxNR, address, contact);
                        clientList.add(client);

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }

                queryClientsListAdapter.notifyDataSetChanged();
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                VolleyLog.e("Error: %s",
error.getClass().getSimpleName());
            }
        });

        // establish a sufficient timeout value
        jsonArrayRequest.setRetryPolicy(new DefaultRetryPolicy(10000,
                DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
                DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
```

```
            // add request to the queue
            queue.add(jsonArrayRequest);
        }
    }
```

Now just click on the green play button to compile and run the Android application. Make sure to select the Launch emulator option, as in Figure 73, to run the application on the previously create AVD, wait for the emulator to initialize and the application to launch.

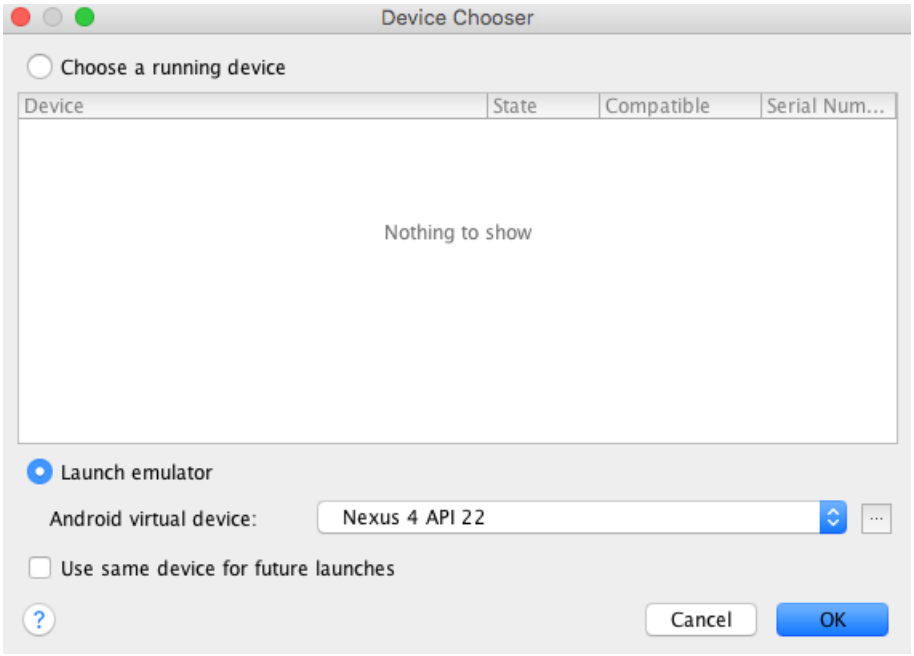


Figure 73 – Choose emulator to run and test the application

If everything went well you should have the Android application running in the virtual device as in Figure 74.

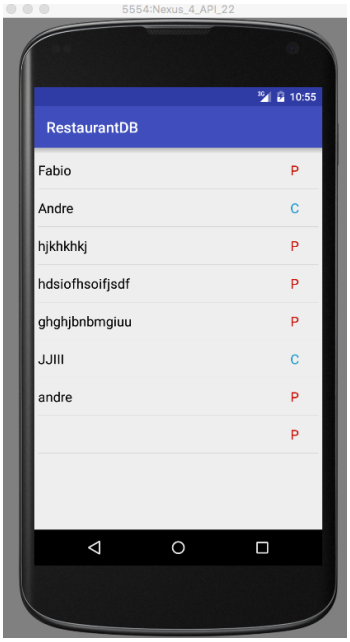


Figure 74 – Android application working on virtual device

Website Example

A very important element in almost every Web Information System is the Website, that is used to interact with the user, gathering and displaying information.

In this section it will be given an example of a simple Website development that uses programming languages like HTML5, CSS3 and JavaScript. The example makes usage of simple elements such as text inputs and buttons, and displays very basic information based on whatever the user inputs. HTML5 is used to structure the page, CSS3 is used to apply styling to the page and JavaScript is used to handle the inputted information and decide what will be displayed.

There will be two sections on the Website – one for the basic usage, know as the Home page, and the other to information related with the developer, known as About page.

Website Structure

The first step is to create the file structure as described in Annex E and rename the source directory to MoodTeller. Then open the file called index.html in a text or code editor to start editing the Website structure (if you open the file using a browser it will preview the Website).

The editing target will be the <body> part of the file, so don't bother looking elsewhere in this file. Start by deleting the paragraph (<p>) that has the Hello World! string, and by completing the instruction below with an "http:" input. Then add a with the ID "container" that will contain the displayable part of the Website. The body at this point should look like:

```
<body>
	<div id="container">
	</div>

	<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"><
/script>
	<script>window.jQuery || document.write('<script
src="js/vendor/jquery-1.11.2.min.js"><\/script>')</script>
	<script src="js/main.js"></script>
</body>
```

Now it is time to add the title for our Website. In order to do that lets add a <header> inside our container (and identify it as mainHeader), and inside the header lets add <h1> element (heading type 1) with the string "Mood Teller Example". Our container should now look like this:

```
<div id="container">
	<header id="mainHeader">
		<h1>Mood Teller Example</h1>
	</header>
</div>
```

To create the option to navigate among different pages or places of the Website, lets add a <nav> element with an <ul> element (unordered list) inside. This list will have two elements, two <li> elements (list item), one to jump to the Home page and the other to jump to the About page. The item reference (href) for the Home page will be "index.html", and for the About page the item reference will be "about.html" (that will be created later). Do this by adding the code lines below to your file.

```
	...
	</header>
	<nav>
		<ul>
			<li><a href="index.html">Home</a></li>
			<li><a href="about.html">About</a></li>
		</ul>
	</nav>
```

```
</div>
```

It is also important to add a <footer> element to the Website with a <p> element (paragraph) with information such as “Copyright © 2016, Mood Teller Example Website”. Add the footer below the navigation part but leave space in the middle to add the main content of the page. Also, before the footer, lets include a with class “clearfix” just to make sure that the <footer> element is not affected with properties from other elements.

```
        ...
        </nav>
        ...

        <div class="clearfix"></div>
        <footer>
              <p>Copyright &copy; 2016, Mood Teller Example Website</p>
        </footer>
</div>
```

In order to properly organized the main space of the page, create a <main> tag that will hold all the elements to display in the page, other than de header, navigation bar and the footer. It should look like:

```
...
</nav>
<main>

</main>
<div class="clearfix"></div>
<footer>
...
```

Inside the <main> tag lets add a <section> element with the ID of “mainSection” to display the principal information in the page, and an <aside> element to display secondary information (typically used as side bar).

```
<main>
        <section id="mainSection">

        </section>
        <aside>

        </aside>
</main>
```

Now lets add a <h2> element (heading 2) inside the mainSection element, that will display as title for this section the string “Mood Teller”, and a <p> element that will introduce a <form>, with ID “moodForm”, and that will not trigger any action itself. Also inside the mainSection, and below the <form> element, add a element, with ID “holder”, to display the information that results from the user’s input.

```
<section id="mainSection">
        <h2>Mood Teller</h2>
        <p>Please type in your name and your current mood</p>
        <form id="moodForm" action="#">

        </form>
        <div id="holder"></div>
</section>
```

Inside the <form> element lets add one <input> element preceded by a <label> element. The label will display the string “Name: “, which introduces an input, with ID “name” and type text, that will be used for the user to input his name. The input element has a placeholder field that suggests to the user what to input.

The moodForm element will not trigger any action itself because on its inside it will have two <button> elements that will trigger the required actions. Both these buttons call methods that are described in main JavaScript file of the project. One of the buttons displays a “Tell Me” string and calls a showMood() method that concatenates the name of the user with a random mood string (obtained by a simple RESTful service) and displays it in the holder element. The other button displays a “Clear” string and calls a clearForm() method, that wipes information from the input and holder elements.

The form should now look like the following:

```
<form id="moodForm" action="#">
	<label>Name: </label>
	<input type="text" id="name" placeholder="Enter your name...">
	<br><br>
	<button onclick="showMood()">Tell me</button>
	<button onclick="clearForm()">Clear</button>
</form>
```

The <br> element is used to add line breaks to the page.

Now let’s add illustrative content to the <aside> element. First we’ll add a <h3> element (heading type 3) displaying “Secondary Bar”, because, although it is normally used as side bar, this element can appear anywhere. Below the heading, let’s add an example paragraph.

```
<aside>
	<h3>Secondary Bar</h3>
	<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
	Aenean quis ipsum venenatis, varius lectus non, sodales tortor.
	Suspendisse posuere quam vitae lorem dictum placerat ac sit amet
	quam. Vestibulum rutrum metus ac finibus volutpat.</p>
</aside>
```

At this point, a preview of the Website should look like the one in Figure 75.

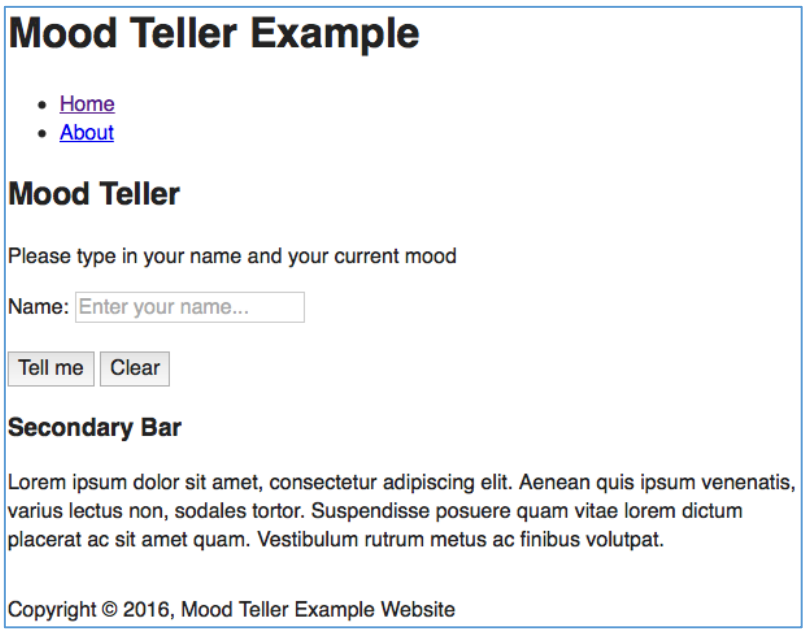


Figure 75 – Mood Teller Example Website structure

In order to allow jumps from the Home page to the About page, it is necessary to create a new about.html file, in the root directory of the project. Copy the content from index.html to the new file, but delete the code lines inside the mainSection. Now just fill the mainSection with your about information, like in the code snippet below:

```
<section id="mainSection">
       Developed by Fábio Lourenço
       <br>
       Eng. Electrónica e Telecomunicações
       <br>
       Universidade de Aveiro
       <br> <br>
       E-mail: fabiolourenco23@hotmail.com
</section>
```

Website Styling

To customize the Website, it is necessary do use CSS3 programming language, in order to apply styling to the wanted elements.

Open the main.css file, inside the css directory. Move to “Author’s custom styles” zone, in order to start customizing.

The CSS3 code snippet below applies a background colour to the body of the page and defines its font family and size.

```
body {
       font-family: helvetica;
       font-size: 13px;
       background: #d07e64;
}
```

Now apply styling to the container element that limits its width to 85% of the full page width, defines a different background colour, applies a margin of 10 pixels to top and bottom and automatic to the sides, configure the overflow as automatic and create rounded corners with 10 pixels of radius.

```
#container {
       width: 85%;
       background: #f4be84;
       margin: 10px auto;
       overflow: auto;
       border-radius: 10px;
}
```

Choose a different colour to the header background, switch the text colour and apply 10 pixels of padding all around the header.

```
header {
       background: #f79362;
       color: #3c393c
       padding: 10px;
}
```

Now define the height of the navigation bar to 40 pixels, switch the colour of the nav bar background and set the overflow to hidden mode.

Set the style of the unordered list of the navigation bar to none, and leave it with no padding or margin (padding will be applied to the list items).

Order the list items to horizontal and to float to left and add 10 pixels of padding to them, except to the bottom.

In the <a> elements of the list, set the text colour to white, without text decoration (so the bullets disappear) and style the font to bold. Also set the text colour to black when the mouse is over each item.

```
nav {
	height: 40px;
	background: #e83232;
	overflow: hidden;
}

nav ul {
	list-style-type: none;
	padding: 0;
	margin: 0;
}

nav li {
	float: left;
	padding: 10px 10px 0 10px;
}

nav a {
	color: white;
	text-decoration: none;
	font-weight: bold;
}

nav a:hover {
	color: black;
}
```

Apply 20 pixels of padding to the elements inside the <main> tag.

```
main {
	padding: 20px;
}
```

To the main section let’s set the position to relative, a width of 70% and for it to float to the left side.

To the side bar let’s define its width to be 20% and its floating to the right.

```
section#mainSection {
	position: relative;
	width: 70%;
	float: left;
}

aside {
	width: 20%;
	float: right;
}
```

To the holder part, where the result will be displayed, let’s set its position to absolute, indicating its place by assigning a top and a left value. Also set the size of the font to be big and visible, and set a fixed width to the holder.

```
#holder {
	position: absolute;
```

```
        top: 10px;
        left: 340px;
        font-size: 35px;
        width: 260px;
}
```

Set the footer to have a 10 pixels padding all around and to be aligned to the centre of the page.

```
footer {
        padding: 10px;
        text-align: center;
}
```

Now move to the “Media Queries” zone and delete completely the first (which is empty). Replace it with code snippet below. This snippet gives the page some flexibility by rearranging the elements position for smaller screen sizes (like smartphones). For every device with a screen smaller than 600 pixels, the main section will be placed n the centre of the page instead of the left side, and the side bar will also be placed in the centre of the page, below the main section, instead of being on the right side.

```
@media only screen and (max-width: 600px) {
        section#mainSection {
                width: 100%;
                float: none;
                text-align: center;
        }

        aside {
                width: 100%;
                float: none;
                text-align: center;
        }
}
```

If everything went well, the Website should now look like the one on the Figure 76.

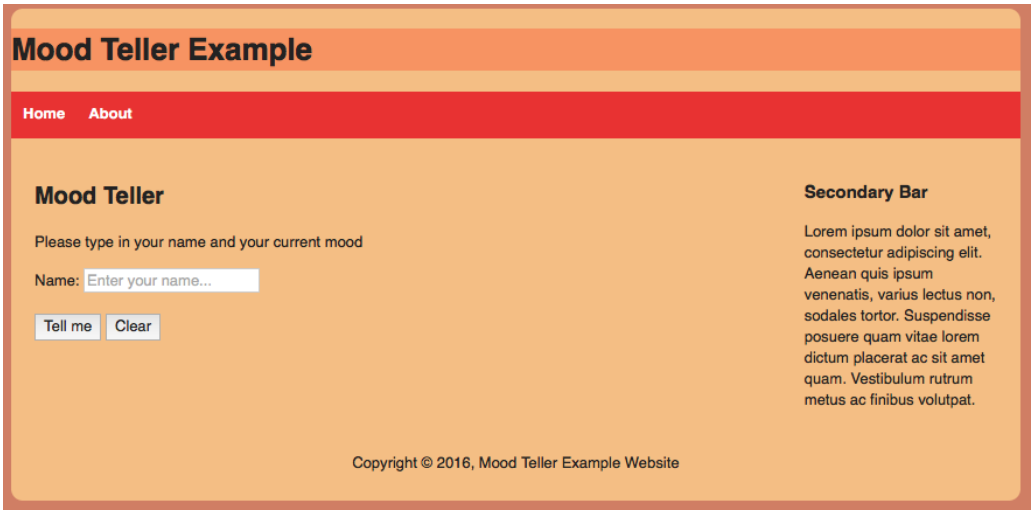


Figure 76 – Mood Teller Example Website after style customization

Website Methods Scripting

Methods can be created by using JavaScript programming language. These methods can be called from the HTML files and perform several actions. For example, when the Tell Me button is pressed the method showMood() is called, which reads the input in the form, consumes the RESTful mood sevice, and displays a simple text.

Open the main.js file, located at the js directory. For the showMood() method copy the code snippet below. It starts by getting the value in the name input at the form. Afterwards it uses the ajax function (from jQuery library) to get the random mood from the RESTful service (this Webservice returns a random mood string formatted in JSON). Then it verifies if the input and the mood the string where both filled and, if not, it prints an alert message and exits from the function. Then it applies some decision to what the output will be, by comparing the mood string received. After that the complete output string is created. Finally, the holder element is loaded and its equalled to the created output string.

```
function showMood() {
	name = document.getElementById("name").value;
	$.ajax({
		type: "GET",
		url: "http://94.60.154.138:1200/mood.php",
		dataType: "json",
		success: function(mood){
			if(!name || !mood) {
				alert("Please fill in all field!");
				return false;
			}

			if(mood == "happy") {
				face = ":)";
			} else if(mood == "sad") {
				face = ":(";
			} else {
				face = ":|";
			}

			tellMeString = name + " is " + mood + " today " +
face;

			holder = document.getElementById("holder");
			holder.innerHTML = tellMeString;

	},
	error: function(e){
	  alert('Error: ' + e);
	}
     });
}
```

The clearForm() method, which intends to clear the inputs and outputs in the page, just needs to load the moodForm element and reset it, and to load the holder element and equal it to an empty string. Put the method below the showMood().

```
function clearForm() {
	document.getElementById("moodForm").reset();

	holder = document.getElementById("holder");
	holder.innerHTML = "";
}
```

If everything went well there should be action on the user presses one of the buttons (Figure 77).

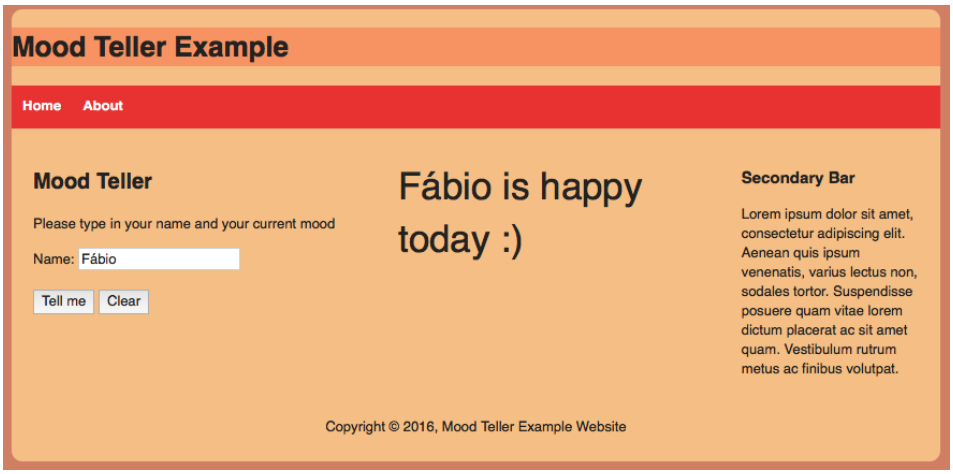


Figure 77 – Mood Teller Example Website input response

Annex

A. Install Apache, MySQL and PHP

These steps will guide you through the installation of Apache, MySQL and PHP packages in a Linux environment. It is recommended to follow these steps using an Ubuntu distribution.

All packages will be installed using the command line. The first step is to install Apache and it is achieved by following the instructions below.

```
$ sudo apt-get update
$ sudo apt-get install apache2
```

After running this instructions Apache will be installed. To check if it is correctly working just open a browser and write `http://localhost/` and it should a similar figure to Figure 78. `localhost` can be replaced by the user local IP or external IP if port forwarding is enabled (section C).

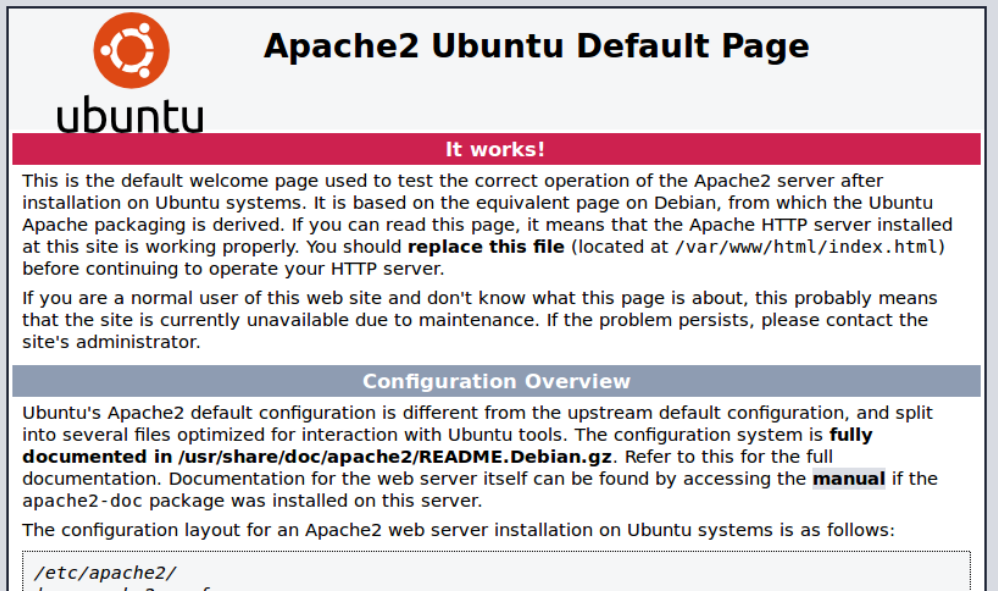


Figure 78 – Apache working on Ubuntu

The second step is to install MySQL database management system. The instructions below install MySQL packages and some secondary packages that are useful for MySQL. During the installation it will be asked to input a password for the MySQL root account, which is the administrator account for MySQL and has extended privileges. Be careful to pick a password that will not be forgotten.

```
$ sudo apt-get install mysql-server php5-mysql
```

Next it is required to build the MySQL database directory. That can be achieved by the following instruction:

```
$ sudo mysql_install_db
```

The only action missing to conclude MySQL installation is to run some security measures. The instruction bellow will prompt some questions that you should answer. The first one is up to you to decide what to answer (you probably want to answer with and ‘n’, since you have chosen the password a moment ago), but the other questions should be answer only by hitting the ENTER key.

```
$ sudo mysql_secure_installation
```

The third and last step of this section is to install PHP. Starting by running an instruction to install PHP packages and then by editing the file in which the Web server looks into when a user requests a directory. The file will be edited using the implicit editor of the command line. This editor is called nano and a file can be open by calling an instruction. To close and save the file it is required to press CTRL+X keys, type Y or N (according if the intention is to save or lose changes made in the file) and then ENTER.

The instruction below the installs PHP packages.

```
$ sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

After PHP is installed, open the dir.conf file and switch the position of the index.php input from the first code snippet to the position in the second snippet. The instruction bellow opens the file in the nano editor. Then take note of the position of the index.php input in the code snippet.

```
$ sudo nano /etc/apache2/mods-enabled/dir.conf

<IfModule mod_dir.c>
	DirectoryIndex index.html index.cgi index.pl index.php
index.xhtml index.htm
</IfModule>
```

Make sure that the index.php input is in the position below.

```
<IfModule mod_dir.c>
	DirectoryIndex index.php index.html index.cgi index.pl
index.xhtml index.htm
</IfModule>
```

Restart the Apache server for the changes to make effect.

```
sudo service apache2 restart
```

To test if PHP packages were correctly installed it is necessary to run a simple script that displays the PHP information of the system. Start by creating a file called info.php that will reside in the Apache scripts directory (which in this development environment is /var/www/html) and edit it with the code lines below. Use the following instruction to create and open the file, edit it and then save it.

```
$ sudo nano /var/www/html/info.php

<?php
phpinfo();
?>
```

By accessing the file through a Web browser (http://localhost/info.php) it is possible to visualize the information displayed in Figure 79.

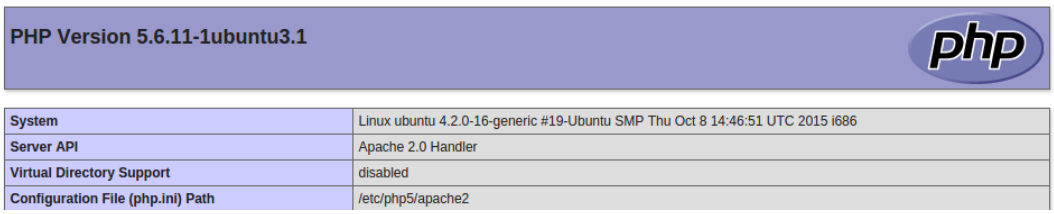


Figure 79 – PHP info script working on Ubuntu server

B. Install Android Studio and create and Android Virtual Device

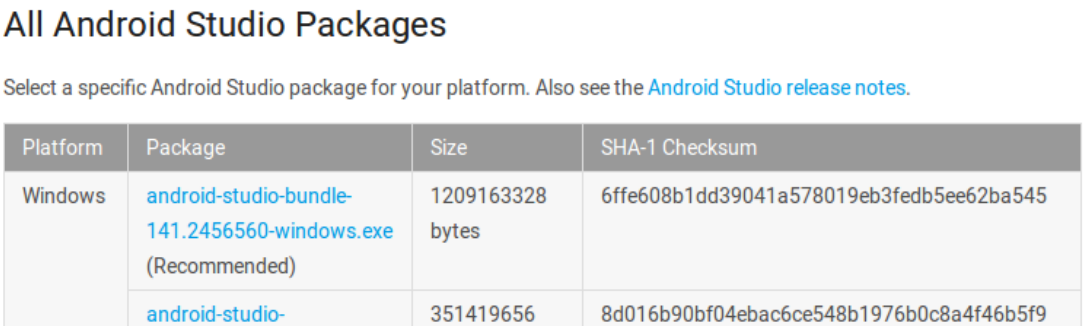
Android Studio is a recommend Integrated Development Environment for Android applications development. It is compatible with all major operative systems but the only one that will be approached in here is Windows.

Before installing Android Studio, it is necessary to install an updated version of Java Development Kit (JDK). Just follow the link below, accept the license to download the package and install JDK.

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Next download and install Android Studio. Be sure to select the recommended package in the All Android Studio Packages section, as in Figure 80. Follow the following link:

http://developer.android.com/intl/pt-br/sdk/index.html



All Android Studio Packages

Select a specific Android Studio package for your platform. Also see the Android Studio release notes.

Platform	Package	Size	SHA-1 Checksum
Windows	android-studio-bundle-141.2456560-windows.exe (Recommended)	1209163328 bytes	6ffe608b1dd39041a578019eb3fedb5ee62ba545
	android-studio-	351419656	8d016b90bf04ebac6ce548b1976b0c8a4f46b5f9

Figure 80 – Android Studio recommended package

After Android Studio is installed, open it and let it install all recommended SDK packages. Then create a new Android Virtual Device (AVD) so that you can test your applications without the need for a physical device.

Click on the AVD Manager and on Create Virtual Device… to start the configuration of a new AVD. Select the Phone category and, for example, hardware like Nexus 4. Select an Android version equal to 4.4 Kitkat or superior. Input the parameters similar to Figure 81 and it Finish.

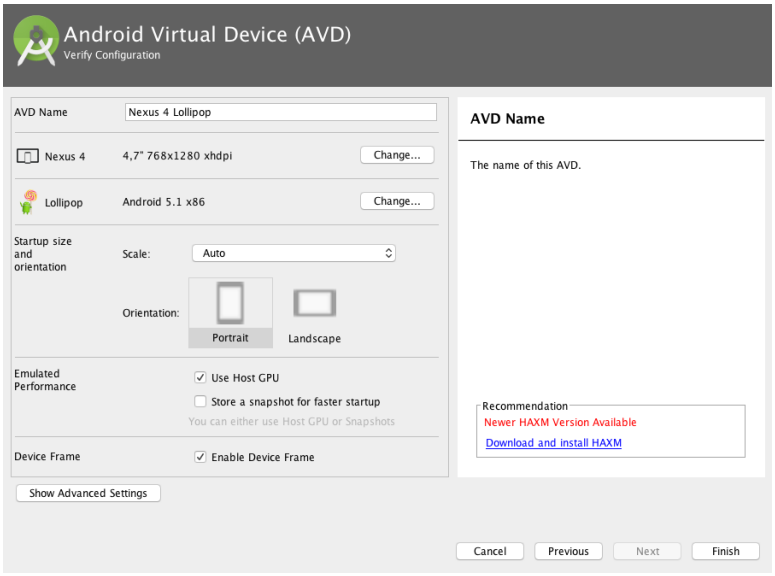


Figure 81 – Android Virtual Device creation definitions

C. Change Apache Server Port and enable Port Forwarding

Choosing the right listening port for the Apache server and enabling port forwarding on your router are important steps to provide access to the server content from a remote external location.

First let's choose the listening port or ports on the Apache server. The following instruction opens the ports.conf file in the nano editor. Replace the listening port from 80 to 1200 for example as in the code snippet below, and the save and close the file.

```
$ sudo nano /etc/apache2/ports.conf

...
# /etc/apache2/sites-enabled/000-default.conf

Listen 1200

<IfModule ssl_module>
       Listen 443
</IfModule>
...
```

Restart the Apache server for the changes to make effect. Be aware that the access to your Apache server must be made detailing which port to access, for example http://localhost:1200/.

```
sudo service apache2 restart
```

Now it is necessary for you to find your local IP address. Run `ifconfig` in the command line and search for a used connection type. It will have your machine's local IP, which should be something like `192.168.X.X`. You can replace `localhost` by this IP address.

```
$ ifconfig

eno16777736 Link encap:Ethernet  HWaddr 00:0c:29:10:5f:20
          inet addr:192.168.10.128  Bcast:192.168.10.255
Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe10:5f20/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4379 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2822 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4608667 (4.6 MB)  TX bytes:313107 (313.1 KB)
          Interrupt:19 Base address:0x2000
```

It is also required for you to find your external IP address, your IP address in the Web. Open a browser and access an Internet speed tester. Run a test and it should display your external IP address.

The next step is to enable port forwarding in your home router. Start by accessing your router interface. This is usually made by input a local IP like `192.168.X.1` or `192.168.X.254` in a Web browser. Search for the port forwarding part and enable forwarding from your local IP port 1200 to your external IP port 1200. It should be something like Figure 82.

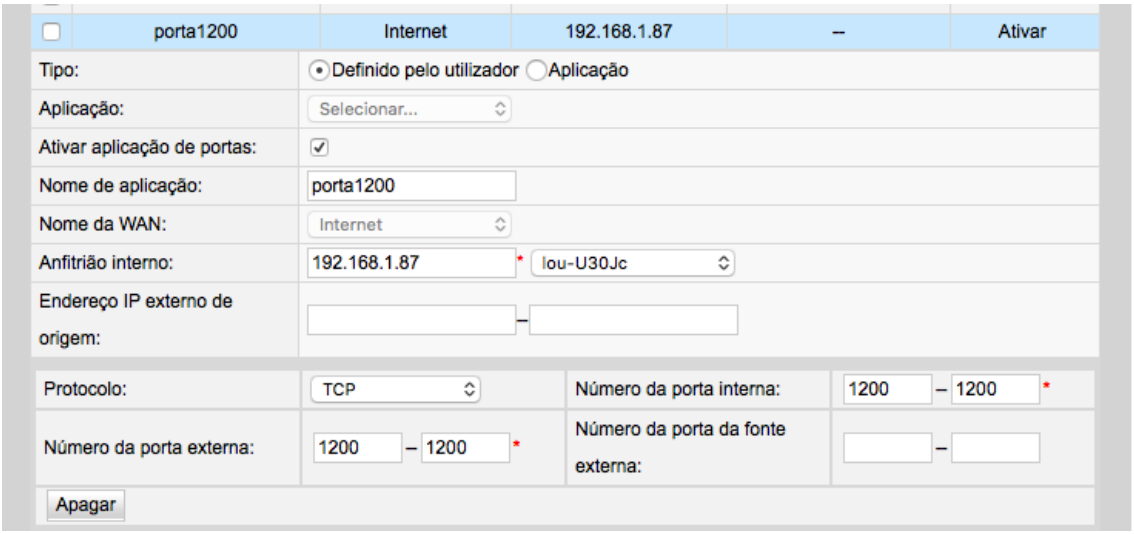


Figure 82 – Opening ports on a Huawei Vodafone router

If it was done correctly you can now access your Web content using your external IP address instead of `localhost`, thereby access it from a place external to your home network.

D. Testing Web Services – Advanced REST Client

In order to test RESTful Web Services while developing them it is necessary to install a simple tool. This is tool is Advanced REST Client and it is a plugin for Chrome Web browser.

This section will help you install and use the tool to test your created API's.

Start by installing Chrome Web browser. Then follow the link below and install the plugin.

https://chrome.google.com/webstore/detail/advanced-rest-client/

After it is installed, open the tool. Select the Request option in the left side of the screen. Input the URI for the Web Service, or API, and input the content type in the Headers section in Raw format as in Figure 83.

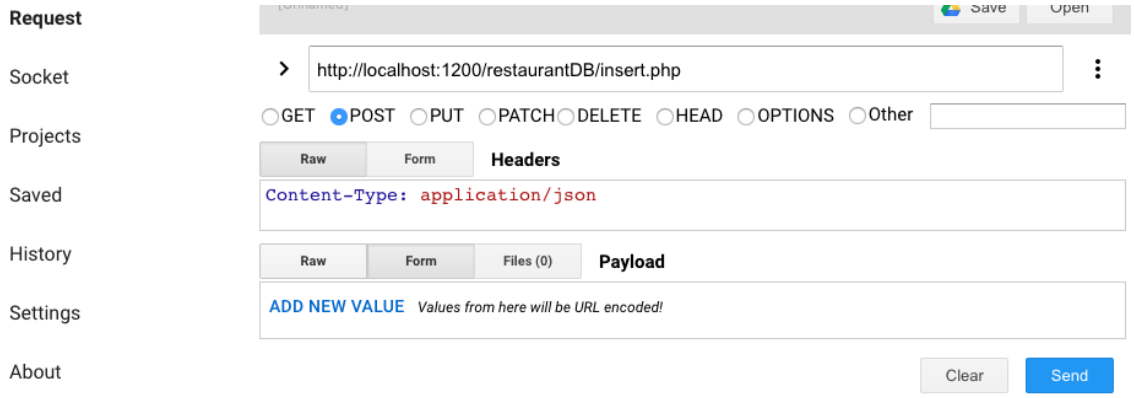


Figure 83 – Calling insert API on Advanced REST Client

In the Payload section select the Form format and add the API parameters by clicking in the ADD NEW VALUE button. The parameter name is placed in the left side of the form and the parameter content in the right side, as in Figure 84.

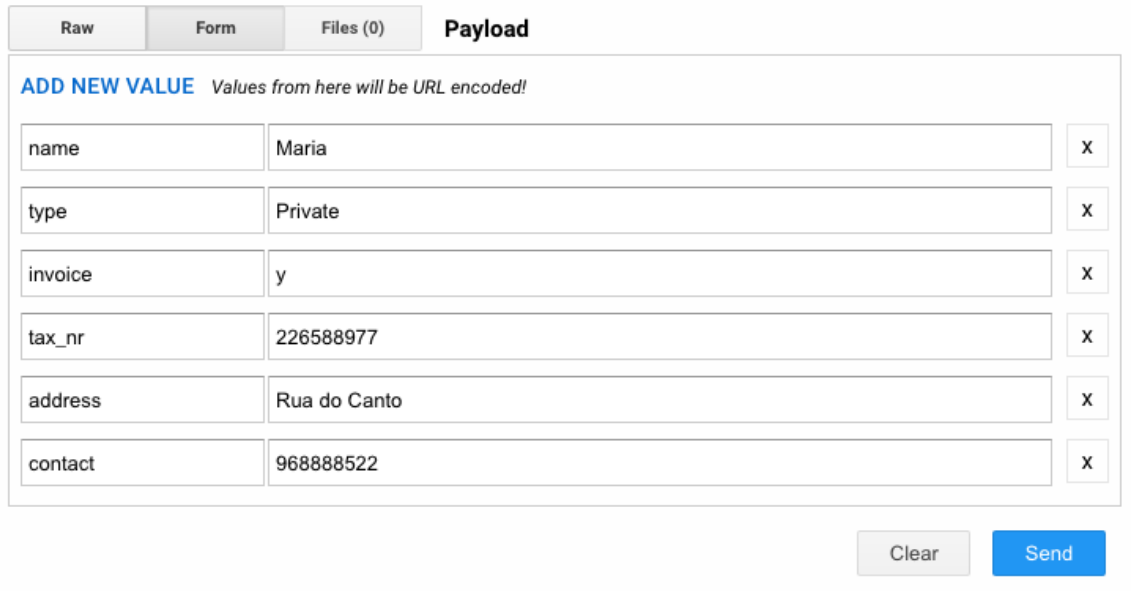


Figure 84 – Inserting body parameters on Advanced REST Client

Now hit send. If everything went well the server should have responded with a 200 code.

E. Creating Website Directory Structure

Access the http://www.initializr.com/ webpage to download a pre-defined Website source. This will serve as basis to the development of a Website.

In 1 – Pre-configuration select Classic H5BP (Figure 85).

Figure 85 – Initializr step 1 – Pre-configuration

In 2 – Fine tuning select the options in Figure 86 and unselect all others.

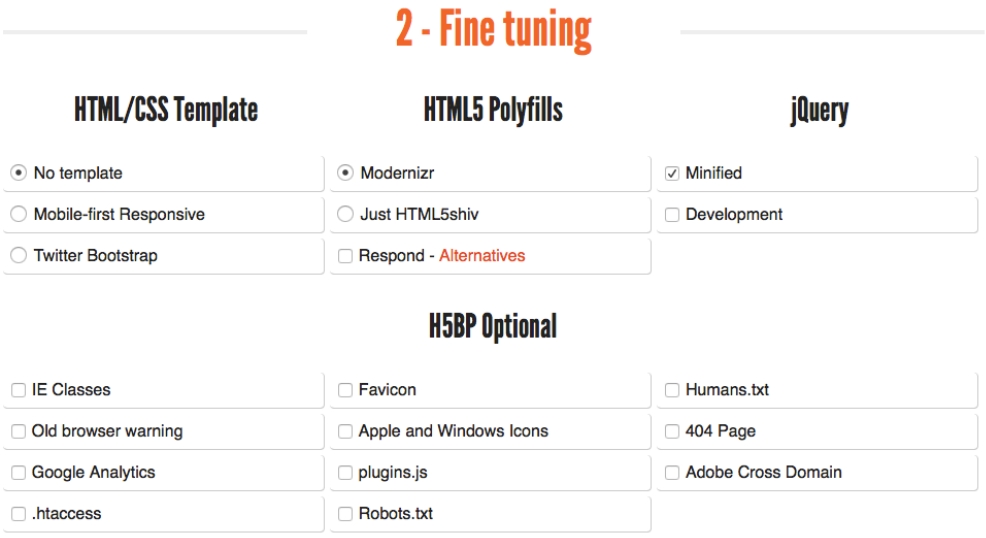


Figure 86 – Initializr step 2 – Fine-tuning

Now click on the download button. It should download a .zip file. Extract it and you will have your project structure (Figure 87) ready to start developing.

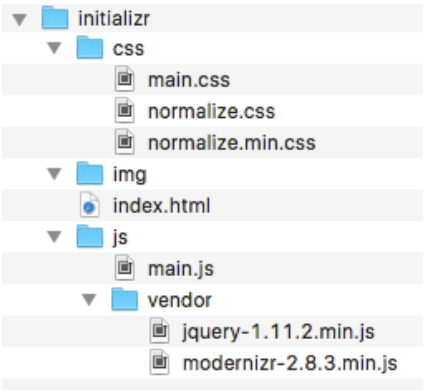


Figure 87 – Initializr project structure