UNIVERSIDAD
REY JUAN CARLOS

Grado en ingeniería en tecnologías de telecomunicación

Curso Académico 2015/2016

Trabajo Fin de Grado

Panel 3D de seguimiento de desarrollo de software

Autor : Adrián Alonso Barriuso

Tutor : Dr. Jesús M. González Barahona

Proyecto Fin de Carrera

FIXME: Título

Autor : Adrián Alonso Barriuso **Tutor :** Dr. Jesús M. González Barahona

La defensa del presente Proyecto Fin de Grado se realizó el día de de 2016, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 20XX

Dedicado a

mi familia / mi abuelo / mi abuela

II

Agradecimientos

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

El presente proyecto tiene como objetivo facilitar la visualización de datos y estadísticas de desarrollo de software y hacerlo en 3D dentro de cualquier navegador web sin la necesidad de instalar ningún controlador, plugin o software adicional.

Para realizarlo se ha utilizado principalmente el lenguaje de programación JavaScript,y HTML5 como base para poder servirse de webGL, una librería que nos permite renderizar gráficos en 3D con aceleración gráfica de forma nativa en casi cualquier navegador. No obstante, dado que webGL es de relativo bajo nivel y sería muy complicado realizar aplicaciones directamente sobre su API, se ha utilizado una librería que proporciona un mayor nivel de abstracción que webGL, esta librería es Three.js.

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

This project aims to facilitate the visualization of data and statistics from software development in 3D, inside any web browser without the need to install any drivers , plugin or additional software.

To do this has been mainly used the JavaScript programming language , and HTML5 as the basis to use WebGL, a library that allow us to render 3D graphics with hardware acceleration natively in almost any browser. However , since webGL is relatively low level and would be very difficult to make applications directly on its API , it has been used a library that provides a higher level of abstraction than WebGL, this library is Three.js.

Índice general

Índice de figuras

Capítulo 1

Introduction

En este capítulo se introduce el proyeto. Debería tener información general sobre el mismo, dando la información sobre el contexto en el que se ha desarrollado.

1.1. Objectives

1.1.1. Problem description

1.1.2. Main Objective

1.1.3. Requeriments

1.2. Demos

Sobre el uso de las comas[1]

```
From gaurav at gold-solutions.co.uk  Fri Jan 14 14:51:11 2005
From: gaurav at gold-solutions.co.uk (gaurav_gold)
Date: Fri Jan 14 19:25:51 2005
Subject: [Mailman-Users] mailman issues
Message-ID: <003c01c4fa40$1d99b4c0$94592252@gaurav7klgnyif>

Dear Sir/Madam,
```

[1] http://narrativabreve.com/2015/02/opiniones-de-un-corrector-de-estilo-11-recetas-par
html

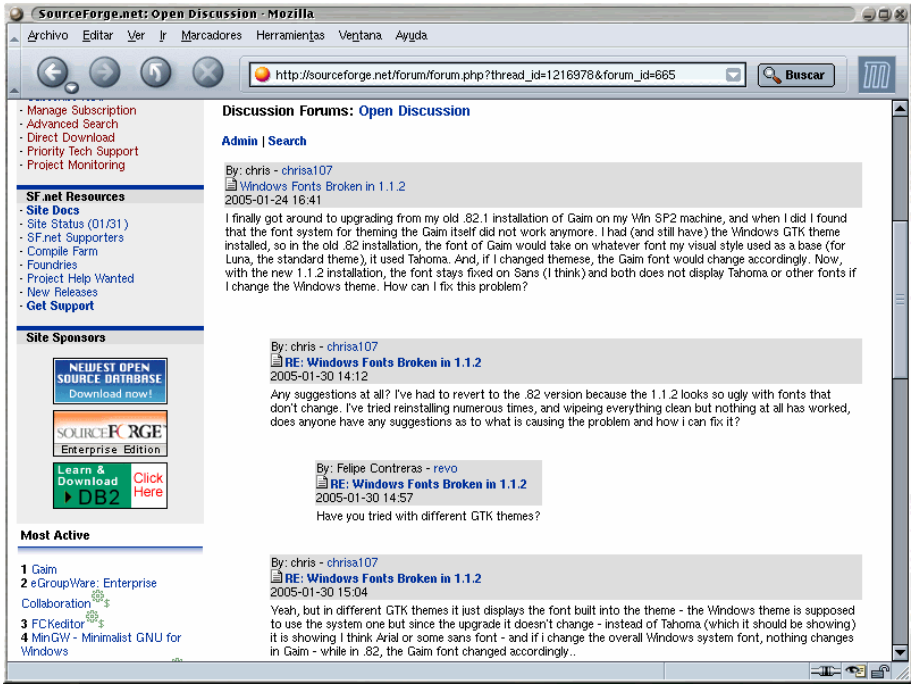

Figura 1.1: Página con enlaces a hilos

```
How can people reply to the mailing list?  How do i turn off
this feature? How can i also enable a feature where if someone
replies the newsletter the email gets deleted?
Thanks

From msapiro at value.net  Fri Jan 14 19:48:51 2005
From: msapiro at value.net (Mark Sapiro)
Date: Fri Jan 14 19:49:04 2005
Subject: [Mailman-Users] mailman issues
In-Reply-To: <003c01c4fa40$1d99b4c0$94592252@gaurav7klgnyif>
Message-ID: <PC17302005011410485105780lb04d55@msapiro>

gaurav_gold wrote:
>How can people reply to the mailing list?  How do i turn off
this feature? How can i also enable a feature where if someone
replies the newsletter the email gets deleted?

See the FAQ
>Mailman FAQ: http://www.python.org/cgi-bin/faqw-mm.py
article 3.11
```

Capítulo 2

Proyect context

2.1. Free software

2.2. Objetivos específicos

2.3. Planificación temporal

Capítulo 3

Objetivos

3.1. Objetivo general

labelsec:objetivo-general

3.2. Objetivos específicos

labelsec:objetivos-especificos

3.3. Planificación temporal

labelsec:planificacion-temporal

Capítulo 4

Used technologies

4.1. HTML5

4.1.1. General description

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It was finalized, and published, on 28 October 2014 by the World Wide Web Consortium (W3C) This is the fifth revision of the HTML standard since the inception of the World Wide Web. The previous version, HTML 4, was standardized in 1997.

Its core aims are to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices (web browsers, parsers, etc.). HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

In particular, HTML5 adds many new syntactic features. These include the new video, audio and canvas elements, as well as the integration of scalable vector graphics (SVG) content (replacing generic object tags) and MathML for mathematical formulas. These features are designed to make it easy to include and handle multimedia and graphical content on the web without having to resort to proprietary plugins and APIs. Other new page structure elements, such as main, section, article, header, footer, aside, nav and figure, are designed to enrich the semantic content of documents. New attributes have been introduced, some elements and attributes have been removed and some elements, such as a, cite and menu¿have been changed, redefined or standardized. The APIs and Document Object Model (DOM) are no longer afterthoughts, but

are fundamental parts of the HTML5 specification.HTML5 also defines in some detail the required processing for invalid documents so that syntax errors will be treated uniformly by all conforming browsers and other user agents.

4.1.2. In this project

As we said in the previous subsection, HTML5 adds the new syntactic feature known as Canvas, allowing us to render dynamic graphics and animations on our web pages. Almost all web browsers supports Canvas today. To render 3D charts, we must to tell Three.js the canvas element that will contain our beautiful crafted charts:

```
// attach div element to variable to contain the renderer
container = document.getElementById( 'ThreeJS' );
// attach renderer to the container div
canvas=renderer.domElement
container.appendChild( canvas );
```

Where the container is the div with ThreeJS tag(for example).

4.2. Javascript

JavaScript is a high-level, dynamic, untyped, and interpreted programming language.It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, it is one of the three essential technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern Web browsers without plug-ins.JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented,imperative, and functional programming styles.It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Despite some naming, syntactic, and standard library similarities, JavaScript and Java are otherwise unrelated and have very different semantics. The syntax of JavaScript is actually derived from C, while the semantics and design are influenced by the Self and Scheme programming languages.

JavaScript is also used in environments that are not Web-based, such as PDF documents, site-specific browsers, and desktop widgets. Newer and faster JavaScript virtual machines (VMs) and platforms built upon them have also increased the popularity of JavaScript for server-side Web applications. On the client side, JavaScript has been traditionally implemented as an interpreted language, but more recent browsers perform just-in-time compilation. It is also used in game development, the creation of desktop and mobile applications, and server-side network programming with runtime environments such as Node.js.

4.2.1. Main features

- Imperative and structured: JavaScript supports much of the structured programming syntax from C (e.g., if statements, while loops, switch statements, do while loops, etc.). One partial exception is scoping: JavaScript originally had only function scoping with var. ECMAScript 2015 adds a let keyword for block scoping, meaning JavaScript now has both function and block scoping. Like C, JavaScript makes a distinction between expressions and statements. One syntactic difference from C is automatic semicolon insertion, which allows the semicolons that would normally terminate statements to be omitted.

- Dynamic: As with most scripting languages, JavaScript is dynamically typed; a type is associated with each value, rather than just with each expression. For example, a variable that is at one time bound to a number may later be re-bound to a string. JavaScript supports various ways to test the type of an object, including duck typing. JavaScript includes an eval function that can execute statements provided as strings at run-time.

- Prototype-based (Object-oriented): JavaScript is almost entirely object-based. In JavaScript, an object is an associative array, augmented with a prototype (see below); each string key provides the name for an object property, and there are two syntactical ways to specify such a name: dot notation (obj.x = 10) and bracket notation (obj['x'] = 10). A property may be added, rebound, or deleted at run-time. Most properties of an object (and any property that belongs to an object's prototype inheritance chain) can be enumerated using a for...in loop.

  JavaScript has a small number of built-in objects, including Function and Date.

- Functional: A function is first-class; a function is considered to be an object. As such, a function may have properties and methods, such as .call() and .bind().A nested function is a function defined within another function. It is created each time the outer function is invoked. In addition, each nested function forms a lexical closure: The lexical scope of the outer function (including any constant, local variable, or argument value) becomes part of the internal state of each inner function object, even after execution of the outer function concludes.JavaScript also supports anonymous functions.

Syntax examples:

```
var x; // defines the variable x, the special value 'undefined' (not to be
    confused with an undefined value) is assigned to it by default
var y = 2; // defines the variable y and assigns the value of 2 to it

//A simple recursive function:
function factorial(n) {
    if (n == 0) {
        return 1;
    }
    return n*factorial(n - 1);
}

//Anonymous function (or lambda) syntax and closure example:
var displayClosure = function() {
    var count = 0;
    return function () {
        return ++count;
    };
}
var inc = displayClosure();
inc(); // returns 1
inc(); // returns 2
inc(); // returns 3
```

4.2.2. In this project

Is have been used JavaScript for all the scripts in the project, and also, all the libraries we have included are written in JavaScript, these libraries are described in detail in the following sections.

4.3. JQuery

PREGUNTAR SI LO INCLUYO, YA QUE SOLO SE USA EVENTUALENTE PARA RECOGER LOS DATOS Y NO ES PARTE DE LA LIBRERIA

4.4. WebGL

WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 3D computer graphics and 2D graphics within any compatible web browser without the use of plug-ins. WebGL is integrated completely into all the web standards of the browser allowing GPU accelerated usage of physics and image processing and effects as part of the web page canvas. WebGL elements can be mixed with other HTML elements and composited with other parts of the page or page background.WebGL programs consist of control code written in JavaScript and shader code that is executed on a computer's Graphics Processing Unit (GPU). WebGL is designed and maintained by the non-profit Khronos Group.

WebGL 1.0 is based on OpenGL ES 2.0 and provides an API for 3D graphics.It uses the HTML5 canvas element and is accessed using Document Object Model interfaces. Automatic memory management is provided as part of the JavaScript language.

Like OpenGL ES 2.0, WebGL does not have the fixed-function APIs introduced in OpenGL 1.0 and deprecated in OpenGL 3.0. This functionality can instead be provided by the user in the JavaScript code space. Shaders in WebGL are expressed directly in GLSL(OpenGL Shading Language,a high-level shading language based on the syntax of the C programming language).

4.4.1. Support

WebGL is widely supported in modern browsers. However its availability is dependent on other factors like the GPU supporting it. The official WebGL website offers a simple test page[1].

More detailed information (like what renderer the browser uses, and what extensions are available) is provided at third-party websites.

Desktop Browsers:

- Google Chrome: WebGL has been enabled on all platforms that have a capable graphics card with updated drivers since version 9, released in February 2011.By default on Windows, Chrome uses the ANGLE (Almost Native Graphics Layer Engine) renderer to translate OpenGL ES to Direct X 9.0c or 11.0, which have better driver support.On Linux and Mac OS X the default renderer is OpenGL however.It is also possible to force OpenGL as the renderer on Windows.Since September 2013, Chrome also has a newer Direct3D 11 renderer, which however requires a newer graphics card.

- Mozilla Firefox: WebGL has been enabled on all platforms that have a capable graphics card with updated drivers since version 4.0.Since 2013 Firefox also uses DirectX on the Windows platform via ANGLE.

- Safari: Safari 6.0 and newer versions installed on OS X Mountain Lion, Mac OS X Lion and Safari 5.1 on Mac OS X Snow Leopard implemented support for WebGL, which was disabled by default before Safari 8.0.

- Opera: WebGL has been implemented in Opera 11 and 12, although was disabled by default in 2014.

- Internet Explorer:WebGL is partially supported in Internet Explorer 11.It initially failed the majority of official WebGL conformance tests, but Microsoft later released several updates. The latest 0.94 WebGL engine currently passes 97 % of Khronos tests. WebGL support can also be manually added to earlier versions of Internet Explorer using third-party plugins such as IEWebGL.

- Microsoft Edge: The initial stable release supports WebGL version 0.95.

[1] http://get.webgl.org/

Mobile Browsers:

- BlackBerry 10: WebGL is available for BlackBerry devices since OS version 10.00
- BlackBerry 10: WebGL is available via WebWorks and browser in PlayBook OS 2.00
- Android Browser: Basically unsupported, but the Sony Ericsson Xperia range of Android smartphones have had WebGL capabilities following a firmware upgrade.Samsung smartphones also have WebGL enabled (verified on Galaxy SII (4.1.2) and Galaxy Note 8.0 (4.2)). Supported in Google Chrome that replaced Android browser in many phones (but is not a new standard Android Browser).
- Internet Explorer:WebGL is available on Windows Phone 8.1.
- Firefox for mobile WebGL is available for Android and MeeGo devices since Firefox 4.
- Firefox OS
- Google Chrome: WebGL is available for Android devices since Google Chrome 25 and enabled by default since version 30.
- Maemo: In Nokia N900, WebGL is available in the stock microB browser from the PR1.2 firmware update onwards.
- MeeGo:WebGL is unsupported in the stock browser ”Web.”However, it is available through Firefox.
- Opera Mobile: Opera Mobile 12 supports WebGL (on Android only).
- Sailfish OS: WebGL is supported in the default Sailfish browser.
- Tizen.
- Ubuntu Touch.
- WebOS.
- iOS:WebGL is available for mobile Safari, in iOS 8.

4.4.2. In this project

The WebGL API may be too tedious to use directly without some utility libraries due to its low abstraction level. Loading scene graphs and 3D objects in the popular industry formats is also not directly provided for. JavaScript libraries have been built (or sometimes ported to WebGL) to provide the additional functionality. A non-exhaustive list of libraries that provide many high-level features includes BabylonJS, three.js, O3D, OSG.JS, CopperLicht and GLGE. There also has been a rapid emergence of game engines for WebGL,including Unreal Engine 4 and Unity 5.The Stage3D/Flash-based Away3D high-level library also has a port to WebGL via TypeScript.A more light-weight utility library that provides just the vector and matrix math utilities for shaders is sylvester.js.It is sometimes used in conjunction with a WebGL specific extension called glUtils.js.

To avoid this low abstraction level , we use Three.js in our case, but here[2] we can find other useful webGL frameworks. To see Three.js in detail, we will go to the next section.

4.5. Three

Three.js is a cross-browser JavaScript library/API used to create and display animated 3D computer graphics in a web browser. Three.js uses WebGL. The source code is hosted in a repository on GitHub[3].

Three.js allows the creation of GPU-accelerated 3D animations using the JavaScript language as part of a website without relying on proprietary browser plugins.This is possible thanks to the advent of WebGL.

High-level libraries such as Three.js or GLGE, SceneJS, PhiloGL or a number of other libraries make it possible to author complex 3D computer animations that display in the browser without the effort required for a traditional standalone application or a plugin.

Main features:

1. Effects: Anaglyph, cross-eyed and parallax barrier.

2. Scenes: add and remove objects at run-time; fog.

[2] https://en.wikipedia.org/wiki/List_of_WebGL_frameworks/
[3] https://github.com/mrdoob/three.js/

3. Cameras: perspective and orthographic; controllers: trackball, FPS, path and more.
4. Animation: armatures, forward kinematics, inverse kinematics, morph and keyframe.
5. Lights: ambient, direction, point and spot lights; shadows: cast and receive.
6. Materials: Lambert, Phong, smooth shading, textures and more.
7. Shaders: access to full OpenGL Shading Language (GLSL) capabilities: lens flare, depth pass and extensive post-processing library.
8. Objects: meshes, particles, sprites, lines, ribbons, bones and more - all with Level of detail.
9. Geometry: plane, cube, sphere, torus, 3D text and more; modifiers: lathe, extrude and tube.
10. Data loaders: binary, image, JSON and scene.
11. Utilities: full set of time and 3D math functions including frustum, matrix, quaternion, UVs and more.
12. Export and import: utilities to create Three.js-compatible JSON files from within: Blender, openCTM, FBX, Max, and OBJ.
13. Support: API documentation is under construction, public forum and wiki in full operation.
14. Examples: Over 150 files of coding examples plus fonts, models, textures, sounds and other support files
15. Debugging: Stats.js, WebGL Inspector, Three.js Inspector.

An example of how complex would be our rendered graphics with Three.js: 4.1.

4.5.1. Usage example

The Three.js library is a single JavaScript file. It can be included within a web page by linking to a local or remote copy:

Figura 4.1: Car rendered with Three.js

```
<script src="js/three.js"></script>
```

The following code creates a scene, adds a camera and a cube to the scene, creates a WebGL renderer and adds its viewport in the document.body element. Once loaded, the cube rotates about its X- and Y-axis.

```
var camera, scene, renderer;
var geometry, material, mesh;

init();
animate();

function init() {

    camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.
        innerHeight, 1, 10000);
    camera.position.z = 1000;

    scene = new THREE.Scene();

    var light = new THREE.PointLight(0xffffff,0.8);
    light.position.set(0,0,200);
    scene.add(light);
```

```
    geometry = new THREE.BoxGeometry(200, 200, 200);
    material = new THREE.MeshLambertMaterial({
        color: 0xff0000,
        wireframe: false
    });

    mesh = new THREE.Mesh(geometry, material);
    scene.add(mesh);

    renderer = new THREE.WebGLRenderer();
    renderer.setSize(window.innerWidth, window.innerHeight);
    renderer.setClearColor( 0xd8d8d8 );


    document.body.appendChild(renderer.domElement);

}

function animate() {

    requestAnimationFrame(animate);

    mesh.rotation.x += 0.01;
    mesh.rotation.y += 0.02;

    renderer.render(scene, camera);

}
```

If all went well, we will see something like this: figure 4.2.

Also, you can play with the example by following this JSfiddle link [4]

[4] https://jsfiddle.net/no0aeknq/

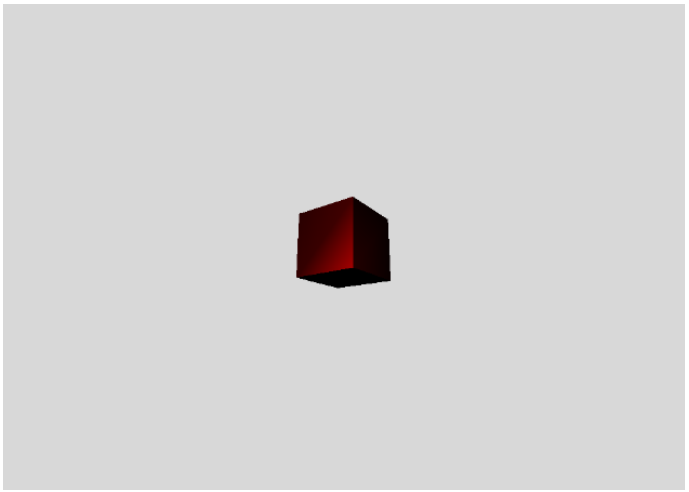

Figura 4.2: Basic cube made with Three.js

4.5.2. In this project

We choose Three.js to write our library because is one the most extended webGL graphics library written on JavaScript and it has a growing community of programmers. However, not all good news, due to it is a relatively new technology, its documentation is a little incomplete and outdated. Therefore, the learning mostly consists of reading code from other developers.

We can find some highly commented use examples in several sites , but they are too old in many cases, using old versions of Three.js that require some changes. One of the used tutorials is a good example of this and we can find it on GitHub [5]. It has a large number of examples of increasing complexity, but we will need to make some changes on its basis if we use the latest version of Three.js.

Also, we have a very complete Udacity course where we can learn the foundations of interactive 3D graphics using Three.js. It contains videos, examples, exercises and even exams. You can register here [6].

[5] http://stemkoski.github.io/Three.js

[6] https://www.udacity.com/course/interactive-3d-graphics--cs291

4.6. Crossfilter

4.7. Domevents

4.8. Orbitcontrols

Puedes citar libros, como el de Bonabeau et al. sobre procesos estigmérgicos [**?**].

También existe la posibilidad de poner notas al pie de página, por ejemplo, una para indicarte que visite la página de LibreSoft[7].

4.9. Sección 1

[7] http://www.libresoft.es

Capítulo 5

Diseño e implementación

5.1. Arquitectura general

figura 5.1.

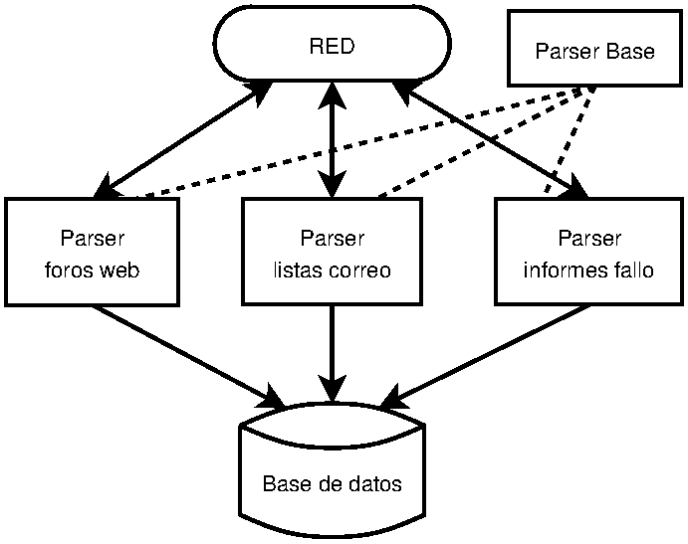

Figura 5.1: Estructura del parser básico

Capítulo 6

Resultados

Capítulo 7

Conclusiones

7.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

7.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a
2. b

7.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. a

2. b

7.4. Trabajos futuros

Ningún software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

7.5. Valoración personal

Finalmente (y de manera opcional), hay gente que se anima a dar su punto de vista sobre el proyecto, lo que ha aprendido, lo que le gustaría haber aprendido, las tecnologías utilizadas y demás.

Apéndice A

Manual de usuario