UNIVERSIDAD
REY JUAN CARLOS

INGENIERíA EN SISTEMAS DE TELECOMUNICACIÓN

Curso Acadï¿½mico 2017/2018

Trabajo Fin de Grado

PLANGO

Autor : Sara López Zambrano Tutor : Pedro de las Heras

Trabajo Fin de Grado

planGO

Autor : Sara López Zambrano

Tutor : Dr. Pedro de las Heras

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2017, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2017

Dedicado a

mi familia / mi abuelo / mi abuela

II

Agradecimientos

Aquï¿½ vienen los agradecimientos... Aunque estï¿½ bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutarï¿½ tanto de tus logros como tï¿½... Ademï¿½s, la pareja quizï¿½s no sea para siempre, pero tu madre sï¿½.

Resumen

Aquï¿½ viene un resumen del proyecto. Ha de constar de tres o cuatro pï¿½rrafos, donde se presente de manera clara y concisa de quï¿½ va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ï¿½De quï¿½ va este proyecto? ï¿½Cuï¿½l es su objetivo principal?
- ï¿½Cï¿½mo se ha realizado? ï¿½Quï¿½ tecnologï¿½as estï¿½n involucradas?
- ï¿½En quï¿½ contexto se ha realizado el proyecto? ï¿½Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Índice general

Índice de figuras

Capítulo 1

Introducción

En poco tiempo se ha producido un gran avance en el mundo de las tecnologías y en concreto en los dispositivos móviles, tanto ha sido así que estamos al alcance de la mayoría de las cosas solo disponiendo de un dispositivo móvil con conexión a Internet. Si navegamos en el tiempo, desde dispositivos que solo nos ofrecían la capacidad de mantener una comunicación instantánea entre dos personas, hemos llegado a incluso poder realizar pagos. Pero es tan importante la evolución a nivel de dispositivo como a nivel de aplicaciones. Se han desarrollado aplicaciones muy potentes que nos facilitan la vida. Un claro ejemplo es el poder gestionar tus cuentas del banco e incluso realizar operaciones desde cualquier lugar. Existen aplicaciones para todos los gustos, necesidades e intereses.

1.1. Historia de la web

La historia de la web abarca ya más de 25 años, en los que se han alternado períodos de intenso desarrollo con otros de estancamiento. El primer servidor de páginas web de la historia se puso en marcha en diciembre de 1990. El inventor de la web, Tim Berners-Lee, pretendía crear un sistema que permitiera a los investigadores del CERN compartir fácilmente la información. La primera versión del lenguaje de marcas inventado por Berners-Lee nunca fue publicado como documento oficial, pero si lo hubiera sido se hubiera llamado HTML 1.0.

Los investigadores del CERN, diseminaron en sus universidades de origen el sistema creado por Berners-Lee, puesto que se trataba de un sistema abierto y libre. En aquella época ya existía Internet, pero su acceso estaba limitado principalmente a Universidades y centros de

investigación.

En noviembre de 1993 se publicó la versión 1.0 de Mosaic, un navegador creado en la Universidad de Illinois por Marc Andreessen y que superaba a todos al permitir, por ejemplo, incluir imágenes en las páginas web.

En 1994 se permitió el acceso de particulares y empresas a Internet. La web se convirtió enseguida en el servicio más empleado para ofrecer información. La web empezó a verse como una gigantesca oportunidad de negocio y Marc Andreessen dejó la universidad para fundar Netscape, que publicaría la versión 1.0 de su navegador en diciembre de 1994.

En octubre de 1994 Berners-Lee fundó el World Wide Web Consortium (W3C), el W3C está organizado en grupos de trabajo. Los primeros grupos de trabajo que se crearon se dedicaron al HTML y a las CSS.

En 1995 Microsoft incluyó en Windows un navegador, Internet Explorer, que poco a poco comenzó a crecer en el mercado en detrimento de Netscape. Entre 1995 y 2000 Microsoft y Netscape publicaron nuevas versiones cada año. Para diferenciar sus productos, cada navegador fue incorporando nuevas etiquetas, lo que supuso un riesgo de fragmentación de la web.

En esos años, el W3C también publicó recomendaciones a ritmo frenético. Por un lado, para consensuar un HTML común para todos los navegadores. Pero por otro lado, proponiendo innovaciones muy importantes, como la separación entre contenido y presentación mediante hojas de estilo (CSS).

En 1995 Brian Eitch creó para Netscape 2.0 el lenguaje de programación Javascript, cuyos programas se podían incluir directamente en las páginas web para ser ejecutados por el navegador. Microsoft creó su propia variante parcialmente incompatible. La normalización de Javascript la llevó a cabo la organización ECMA, que en 1997 empezó a publicar normas para unificar y desarrollar el lenguaje.

Ante la necesidad por incluir nuevos campos el W3C creó el XML. El problema era que el HTML no cumplía las nuevas reglas del XML y el W3C planteó reformular el HTML de acuerdo con ellas (ese nuevo lenguaje se llamaría XHTML).

En 1998 Netscape creó la organización Mozilla.

En el año 2000 la guerra de navegadores culminó con la victoria de Internet Explorer y la desaparición de Netscape. Microsoft decidió no seguir innovando y no habría nuevas versiones después de Internet Explorer 6.

Durante estos años, la organización Mozilla desarrolló un nuevo navegador, Mozilla, de uso muy minoritario pero que respetaba las recomendaciones del W3C, apareciendo como una alternativa a Internet Explorer.

Debido a la competencia, Microsoft retomó el desarrollo de Internet Explorer. En 2004 se creó también el WHATWG (grupo formado por Mozilla, Apple y Opera), para retomar el desarrollo del HTML que el W3C había abandonado en favor del XHTML, bajo el nombre de HTML 5.

En 2007 el W3C formó un grupo de trabajo sobre HTML, que trabajaría conjuntamente con el WHATWG para publicar la recomendación HTML 5.

En 2009 Google publicó su propio navegador: Google Chrome. En 2011 el W3C abandonó el desarrollo del XHTML y se concentró en el HTML 5.

En 2011 el WHATWG abandonó por su parte el nombre de HTML 5 y pasó a denominarlo simplemente HTML, abandonando la idea de versiones en favor de una norma ”líquida”, continuamente modificada y mejorada.

En 2013 Microsoft consiguió con Internet Explorer 11 cumplir de forma correcta las antiguas recomendaciones HTML 4 y CSS 2 y admitir lenguajes XML como SVG. Pero para sacar todo el partido a HTML 5, Microsoft creó un nuevo navegador (Edge), que ya no estará ligado a las nuevas versiones de Windows.

En estos años Google Chrome ha desbancado a Internet Explorer, entre otros motivos debido al gran uso de los teléfonos móviles y al hecho de que los usuarios de Windows 7 no pueden usar Edge.

1.2. Trabajo relacionado

1.3. Tecnoloías

1.3.1. HTML5

HTML, que significa Lenguaje de Marcado para Hipertextos, es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad. Hiper Texto se refiere a en-

laces que conectan una páginaWeb con otra, ya sea dentro de una página web o entre diferentes sitios web. Un lenguaje de marcado hace referencia a aquellos lenguajes que emplean etiquetas. Estas etiquetas ya están predefinidas dentro del lenguaje respectivo y contienen la información que ayudan a leer el texto. Su principal diferencia con los lenguajes de programación es que éstos últimos poseen funciones aritméticas o variables, mientras que los lenguajes de marcado no.

HTML5 es la quinta revisión del estándar que fue creado en 1990 y su versión definitiva se publicó en octubre de 2014. Con HTML5, los navegadores como Firefox, Chrome, Explorer, Safari y más pueden saber cómo mostrar una determinada página web, saber dónde están los elementos, dónde poner las imágenes, dónde ubicar el texto.

En HTML5, se han tomado en cuenta mejoras en la creación de la estructura del código web y en el manejo óptimo de las etiquetas web. De esta manera, se convierte en un estándar mucho más versátil, que permitirá realizar una interacción mucho más poderosa y simple, mejorando la experiencia de uso por parte del usuario y facilitando la depuración del código web.

Las ventajas principales de esta versión son:

- Nueva estructura de etiquetas: permite separar el encabezado, la barra de navegacion, secciones de la página, textos, diálogos y el pié de página.
- Introduccion de etiquetas video y audio: por medio de las etiquetas ¡video¿y ¡audio¿de HTML5, ahora puedes añadir videos o audio sin necesidad de usar Adobe Flash o cualquier otro plugin de tercero. Toda la acción sucede desde el propio navegador, lo que puede ayudar a disminuir al tamaño del archivo final de tu página.
- Geolocalización: permite al sitio detectar la ubicación de cada usuario que ingresa al sitioweb.
- Aplicaciones web: desarrollar aplicaciones HTML5 tiene la ventaja de que el resultado final es completamente accesible, es decir, se puede acceder a esta aplicación desde un ordenador, tablet o móvil.
- Capacidad de realizar ejecucciones offline: esto permite realizar aplicaciones de escritorio.
- Canvas: nueva etiqueta de dibujo sobre la página web.

1.3.2. JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Surgió por la necesidad de ampliar las posibilidades del HTML. En efecto, al poco tiempo de que las páginas web apareciesen, se hizo patente que se necesitaba algo más que las limitadas prestaciones del lenguaje básico, ya que el HTML solamente provee de elementos que actuan exclusivamente sobre el texto y su estilo, pero no permite, como ejemplo sencillo, ni siquiera abrir una nueva ventana o emitir un mensaje de aviso. La temprana aparición de este lenguaje, es posiblemente la causa de que se haya convertido en un estándar soportado por todos los navegadores actuales.

Los documentos HTML permiten incrustar fragmentos de código JavaScript, bien dentro del propio archivo HTML o bien realizando una carga de ese código indicando el archivo donde se encuentra el código JavaScript. Dentro de un documento HTML puede haber ninguno, uno o varios scripts de JavaScript.

Además, tambien es utilizado del lado del servidor, ya que tiene la ventaja de poseer un excelente modelo de eventos, ideal para la programación asíncrona.

1.3.3. TypeScript

TypeScript es un lenguaje de programación de alto nivel que implementa muchos de los mecanismos más habituales de la programación orientada a objetos, pudiendo extraer grandes beneficios que serán especialmente deseables en aplicaciones grandes, capaces de escalar correctamente durante todo su tiempo de mantenimiento. Puede ser usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor (Node.js).

TypeScript convierte su código en Javascript común. Es llamado también Superset de Javascript, lo que significa que si el navegador está basado en Javascript, este nunca llegará a saber que el código original fue realizado con TypeScript y ejecutará el Javascript como lenguaje original.

1.3.4. CSS3

Es el lenguaje utilizado para describir la presentación de documentos HTML o XML. Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página y una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento.

El objetivo inicial de CSS, separar el contenido de la forma, se cumplió ya con las primeras especificaciones del lenguaje. Sin embargo, el objetivo de ofrecer un control total a los diseñadores sobre los elementos de la página ha sido más difícil de cubrir. Las especificaciones anteriores del lenguaje tenían muchas utilidades para aplicar estilos a las webs, pero los desarrolladores aun continúan usando trucos diversos para conseguir efectos tan comunes o tan deseados como los bordes redondeados o el sombreado de elementos en la página.

1.3.5. Sass

Sass es un metalenguaje de Hojas de Estilo en Cascada (CSS). Es un lenguaje de script que es traducido a CSS.

Extiende CSS proveyendo de varios mecanismos que están presentes en los lenguajes de programación tradicionales, particularmente lenguajes orientados a objetos, pero éste no está disponible para CSS3 como tal. Cuando SassScript se interpreta, éste crea bloques de reglas CSS para varios selectores que están definidos en el fichero SASS. El intérprete de SASS traduce SassScript en CSS.

Sass permite la definición de variables de tipo number, string, colores y booleanos. Otra ventaja es que soporta mixins Un mixin es una sección de código que contiene código Sass. Cada vez que se llama un mixin en el proceso de conversión el contenido del mismo es insertado en el lugar de la llamada. Los mixin permiten una solución limpia a las repeticiones de código, así como una forma fácil de alterar el mismo.

1.3.6. Node.js

Node.js es un entorno de ejecución multiplataforma de código abierto para desarrollar aplicaciones web. Esta librería se ejecuta sobre JavaScript y está basado en el motor V8 de Javascript de Google. Este motor está diseñado para correr en un navegador y ejecutar el código de

Javascript de una forma extremadamente rápida.

Se trata de un intérprete Javascript del lado del servidor, lo que permite utilizar el mismo lenguaje de programación tanto para cliente como para servidor. Node sirve para facilitar la creación de aplicaciones web escalables de manera sencilla y con gran estabilidad, pudiendo ser utilizado para desarrollar cualquier tipo de aplicación. Además, es importante volver a destacar su altísima velocidad y su flexibilidad, dos de sus cualidades más importantes.

Trabaja con un único hilo de ejecución que es el encargado de organizar todo el flujo de trabajo que se deba realizar. Gestiona sus tareas de manera asíncrona y para trabajar de manera óptima delega todo el trabajo en un poll de threads. La libreria que construye esto es Libuv, una vez que el trabajo ha sido completado emite un evento recibido por Node.js.

1.3.7. Ionic 2

Se trata de un framework destinado al desarrollo de aplicaciones híbridas, aunque también puede ser utilizado para para implementar aplicaciones web. Una aplicación híbrida es aquella desarrollada por las tecnologías web: HTML, CSS Y JavaScript. Este tipo de aplicaciones tienen una serie de ventajas como ser compatibles para una gran cantidad de sistemas operativos con un tiempo de desarrollo menor, pero a cambio de esta gran ventaja el rendimiento es menor que en una aplicación nativa.

Su caracteríítica fundamental es que usa por debajo Angular, esto le da ventajas como tener una buena estructura de proyecto y contar con una buena gama de componentes y directivas.

Componentes

Los componentes se utilizan unos a otros para la obtención de objetivos globales de la aplicaci ón. Están pensados para, de manera modular y encapsulada, resolver pequeños problemas. Ionic ofrece componentes fáciles de utilizar, pero para comportamientos más especííicos de nuestro modelo de negocio, será necesario crear nuestros propios componentes.

Los componentes de Ionic 2 se adaptan al dispositivo estéticamente. Manteniendo el mismo código, en un dispositivo iOS tiene diferente vista que en un dispositivo Android, ya que se visión de nativa y además da al usuario una experiencia cercana a la que está acostumbrada en su teléfono. Sin embargo, es decisión del desarrollador mantener esta visión en su aplicación o

personalizar la estética a su gusto. adapta al sistema operativo en el que se compila. Esto permite que una aplicación híbrida de

Apache Cordova

Para el acceso a componentes nativos desde la aplicación de Ionic, como la cámara, aceler ómetro, teclado, usa plugins que proporciona Apache Cordova. También nos permite compilar el desarrollo realizado con Ionic con tecnologías web en aplicaciones para móviles instalables mediante tiendas de aplicaciones.

Estructura

Un proyecto con Ionic contiene una lista de capetas y archivos. Cada parte tiene su función:

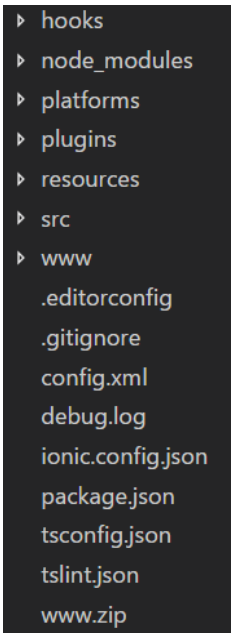

Figura 1.1: Estructura ionic

- SRC: carpeta que contiene los archivos fuente con el código desarrollado de la aplicación.
- WWW: contiene los archivos que se producen al realizar la transpilación del TypeScript y compilado de los archivos Sass, es decir, la trasformación de todos los archivos de 'src', de tal manera que el navegador sea capaz de entender.
- PLUGINS: contiene todos los plugins nativos que se utilizan para la aplicación.

- PLATFORM: archivos de cada plataforma a la que da soporte la aplicación. Suele ser Android e iOS.

- RESOURCES: contiene los iconos de la aplicación y el splash screen.

- HOOKS: scripts que se crean para ser ejecutados automáticamente después de algo específico.

- NODE MODULES: dependencias de npm que vienen definidas en el package.json e instaladas en local dentro de tu proyecto.

1.3.8. Express

Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Contiene muchos métodos de programa de utilidad HTTP y middleware.

Node.js es una plataforma construida sobre el motor de JavaScript de Google Chrome (V8) que permite la ejecución de JavaScript en el lado del servidor. Permite montar un servidor HTTP utilizando el modulo http que viene incluido en el core de Node.

Para comenzar un proyecto en Express es necesario configurar en que puerto e IP va a estar escuchando el servidor para atender a las peticiones. Además, es necesario añadir las urls a las que tiene que atender y que método realizar para cada caso.

Estructura

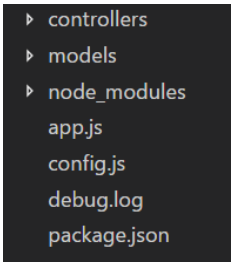

Figura 1.2: Estructura express

1.3.9. MongoDB

Se trata de una base de datos no relacional, es decir, NoSQL. Esto significa que los datos no son almacenados en tablas y no garantiza consistencia. Este tipo de bases de datos tienen ciertas desventajas, como: poca eficiencia en aplicaciones que necesiten usar los datos intensivamente, o si contiene gran número de indexaciones. A cambio de esta desventaja tiene la capacidad de manejar mucha cantidad de datos, no generan cuellos de botella y se ejecutan en clusters de máquinas baratas.

Este tipo de bases de datos surgieron por el Big Data, donde la información que se genera es muy grande, de manera rápida y constante, y que, además, en ocasionas la forma es no estructurada y cambiante. Las bases de datos relacionales teníín carencias para afrontar esto.

Colecciones

MongoDB es una base de datos orientada a documentos, es decir, los documentos son almacenados en BSON, que es una representación binaria de JSON. No siguen un esquema dijo, los documentos de una misma colección pueden tener esquemas diferentes.

En MongoDB los documentos se agrupan en colecciones. Aunque lo normal es que los documentos de una colección compartan estructura, puede ser flexible porque la estructura no se impone a ninguÂ´n documento y dinmÂ´ ica porque la estructura puede cambiar. En el caso de que se decida variar la estructura de datos, no sería necesarios crear ni modificar las colecciones, bastaría con almacenar los nuevos documentos, con una estructura distinta, en la misma colección o en otra.

No proporciona integridad referencial. Esto significa que, si en una colección hacemos referencia a un documento de otra colección, la base de datos no tiene la responsabilidad de comprobar que el documento referenciado existe.

Velocidad

MongoDB tiene una baja velocidad en generar o modificar información. Esto provoca que el acceso a la base de datos este más tiempo bloqueado. Una operación de escritura bloquea el acceso a toda la base de datos en la que se efectúa la operación. Para solucionar esto hay que evitar actualizaciones que provoquen movimientos, es decir, que el crecimiento de un documen-

to sea tan grande que suponga un cambio de cajón. Cada documento se encuentra almacenado en un cajón el cual cuenta con un espacio extra para posibles crecimientos futuros.

A cambio de esta baja velocidad de escritura, proporciona una gran rapidez de lectura.

1.4. Estructura de la memoria

En esta secciï¿½n se deberï¿½a introducir la esctura de la memoria. Asï¿½:

Capítulo 2

Objetivos

2.1. Objetivo general

Aquï¿½ vendrï¿½a el objetivo general en una frase: Mi trabajo fin de grado consiste en crear de una herramienta de anï¿½lisis de los comentarios jocosos en repositorios de software libre alojados en la plataforma GitHub.

Recuerda que los objetivos siempre vienen en infinitivo.

2.2. Objetivos especï¿½ficos

Los objetivos especï¿½ficos se pueden entender como las tareas en las que se ha desglosado el objetivo general. Y, sï¿½, tambiï¿½n vienen en infinitivo.

2.3. Planificaciï¿½n temporal

A mï¿½ me gusta que aquï¿½ pongï¿½is una descripciï¿½n de lo que os ha llevado realizar el trabajo. Hay gente que aï¿½ade un diagrama de GANTT. Lo importante es que quede claro cuï¿½nto tiempo llevas (tiempo natural, p.ej., 6 meses) y a quï¿½ nivel de esfuerzo (p.ej., principalmente los fines de semana).

Capítulo 3

Estado del arte

Descripciï¿½n de las tecnologï¿½as que utilizas en tu trabajo. Con dos o tres pï¿½rrafos por cada tecnologï¿½a, vale. Se supone que aquï¿½ viene todo lo que no has hecho tï¿½.

Puedes citar libros, como el de Bonabeau et al. sobre procesos estigmï¿½rgicos [1].

Tambiï¿½n existe la posibilidad de poner notas al pie de pï¿½gina, por ejemplo, una para indicarte que visite la pï¿½gina de LibreSoft[1].

3.1. Seccï¿½n 1

Hemos hablado de cï¿½mo incluir figuras. Pero no hemos dicho nada de tablas. A mï¿½ me gustan las tablas. Mucho. Aquï¿½ un ejemplo de tabla, la Tabla 3.1.

[1] `http://www.libresoft.es`

1	2	3
4	5	6
7	8	9

Cuadro 3.1: Ejemplo de tabla

Capítulo 4

Diseï¿½o e implementaciï¿½n

Aquï¿½ viene todo lo que has hecho tï¿½ (tecnolï¿½gicamente). Puedes entrar hasta el detalle. Es la parte mï¿½s importante de la memoria, porque describe lo que has hecho tï¿½. Eso sï¿½, normalmente aconsejo no poner cï¿½digo, sino diagramas.

4.1. Arquitectura general

Si tu proyecto es un software, siempre es bueno poner la arquitectura (que es cï¿½mo se estructura tu programa a "vista de pï¿½jaro").

Por ejemplo, puedes verlo en la figura 4.1.

Si utilizas una base de datos, no te olvides de incluir tambiï¿½n un diagrama de entidad-relaciï¿½n.

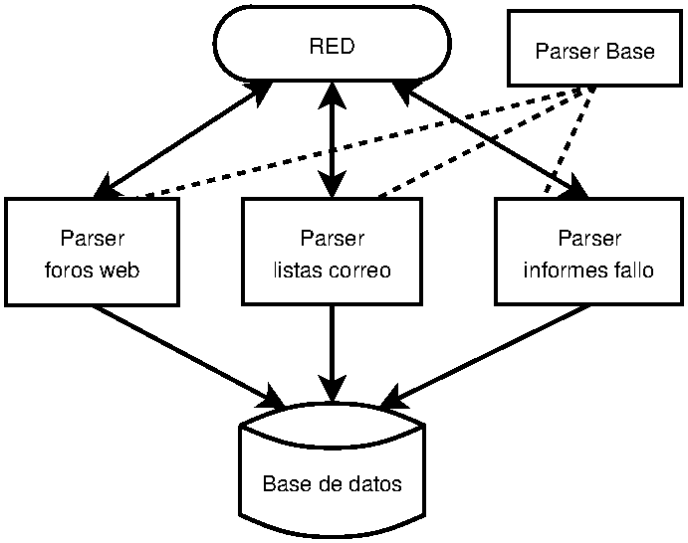

Figura 4.1: Estructura del parser bï¿½sico

Capítulo 5

Resultados

En este capï¿½tulo se incluyen los resultados de tu trabajo fin de grado.

Si es una herramienta de anï¿½lisis lo que has realizado, aquï¿½ puedes poner ejemplos de haberla utilizado para que se vea su utilidad.

Capítulo 6

Conclusiones

6.1. Consecucii¿½n de objetivos

Esta secciï¿½n es la secciï¿½n espejo de las dos primeras del capï¿½tulo de objetivos, donde se planteaba el objetivo general y se elaboraban los especï¿½ficos.

Es aquï¿½ donde hay que debatir quï¿½ se ha conseguido y quï¿½ no. Cuando algo no se ha conseguido, se ha de justificar, en tï¿½rminos de quï¿½ problemas se han encontrado y quï¿½ medidas se han tomado para mitigar esos problemas.

6.2. Aplicacii¿½n de lo aprendido

Aquï¿½ viene lo que has aprendido durante el Grado/Mï¿½ster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas mï¿½s relacionadas y comentar en un pï¿½rrafo los conocimientos y habilidades puestos en prï¿½ctica.

1. a

2. b

6.3. Lecciones aprendidas

Aquï¿½ viene lo que has aprendido en el Trabajo Fin de Grado/Mï¿½ster.

1. a

2. b

6.4. Trabajos futuros

Ningï¿½n software se termina, asï¿½ que aquï¿½ vienen ideas y funcionalidades que estarï¿½a bien tener implementadas en el futuro.

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

Apéndice A

Manual de usuario

Bibliografía

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Articial Systems*. Oxford University Press, Inc., 1999.