UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Sistema de Gestión para Trabajos Fin de Máster y Prácticas en Empresas en el MII

Miguel González Cano

Septiembre, 2016

Sistema de Gestión para Trabajos Fin de Máster y Prácticas en Empresas en el MII

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

TECNOLOGÍA ESPECÍFICA DE
TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

Sistema de Gestión para Trabajos Fin de Máster y Prácticas en Empresas en el MII

Autor: Miguel González Cano

Director: David Vallejo Fernández

Septiembre, 2016

Miguel González Cano

Ciudad Real – Spain

E-mail: Miguel.Gonzalez3@alu.uclm.es
Teléfono: 660172363

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

Resumen

En la ingeniería del software se denomina aplicación web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador. En otras palabras se podría decir que es un aplicación software que se codifica e implementa en lenguajes soportados por los navegadores web en la que se confía la ejecución a los navegadores. Éstas son muy populares debido a lo pŕactico de navegador web como cliente de la aplicación, a independencia de sistemas operativos y de instalaciones nativas. Existen aplicaciones como los webmails, wikis, weblogs, tiendas en línea y la propia Wikipedia que son ejemplos bastante conocidos de aplicaciones web.

En la actualidad la Universidad de Castilla-La Mancha ofrece una titulación oficial de Máster en Ingeniería Informática. Las instituciones educativas disponen de plataformas virtuales de gestión de contenidos de aprendizaje, lo que aporta una plataforma que permita llevar la gestión de los cursos y titulaciones, así como todos los contenidos que se discuten. Es por ello que existen medios para poder gestionar todos los procesos educativos que se refieren a estos asuntos y así realizar su labor didáctica de una manera más adecuada. Una vez se han conseguido aprobar todas las asignaturas, es necesario presentar un Trabajo Fín de Máster que consiste en un proyecto integral de Ingeniería en Informática de naturaleza profesional que aplica todos los conocimientos adquiridos en el máster.

Lo que propone con este Trabajo Fin de Grado es la realización de una aplicación web que permita gestionar dichas propuestas por parte del coordinador, ya que el proceso involucra no solo a los tutores y Coordinador del Máster, sino también a una Comisión Académica del Máster que verificará y validará las propuesta. Por lo tanto, al involucrar a ciertos individuos con varias funciones de gestión y la posibilidad de simplificar el tedioso proceso gracias a las plataformas de desarrollo de aplicaciones web, se pone de manifiesto la idea de este proyecto, el cual se relatará en los capítulos venideros. A lo largo de este documento se describe como se inició y desarrolló dicho proyecto: cómo se planteó, qué problemas surgieron, cómo se solucionaron, el resultado final y las conclusiones.

Abstract

In software engineering it is called web applications to that aplications which users can use a web server to access via Internet or an intranet using a web browser. In other words you could say it is a software application that is coded and implemented in several languages supported by web browsers in which execution is entrusted to browsers. These are very popular because of his practicality as a client application, to independence from operating systemas and native setups. There are applications such as webmails, wikis, weblogs, online stores and Wikipedia itself that are well know examples of web applications.

Today, the University of Castilla-La Mancha offers an official Master's degree in Computer Engineering. Educational Institutions have virtual platforms for learning content management, which provides a platform to bring the management of courses and degrees, as well as all content discussed. That's why there are means to manage all educational processes that relate to these matters and so do their teaching work in a more appropiate manner. Once you have managed to pass all subjects, you must show a Master's Thesis consisting of a comprehensive project in Computer Engineering professional nature that applies all the knowledge acquired in the master.

What proposes this Final Project is the implementation of a web application that can manage these proposals by the coordinator, as the process involves not only the tutor and Coordinator, but also an Academic Committee of the Master to verify and validate the proposal. Therefore, by involving certain individuals with various management functions and the ability to simplify the tedious process through development platforms web applications it highlights the idea of this project, wich will be told in the coming chapters.

Throughout this document is described as initiated and developed the project: how it was raised, what problems arose, how were settled, the final results and conclusions.

Agradecimientos

A toda mi familia que siempre ha estado ahí apoyándome y no dejándome caer, a todos los compañeros de los cuales me llevo gratos recuerdos, a todos los profesores que me han formado no sólo como estudiante, sino también como persona. Al mejor apoyo de la carrera, a Fede, Antonio, Carlos, Fernando, Rodrigo, Santiago etc. por compartir con vosotros grandes años de mi vida. A mi tutor y mentor en todo esto, David Vallejo, el tutor que siempre quise y siempre estuvo ahí para guiarme. Y sobretodo a mi media parte, la parte que me complementa y me hace seguir hacia delante, la parte que me equilibra y hace mejor persona, esa gran parte de mi en la que se ha convertido, sin ella nada de esto habría ocurrido. ¡Mil Gracias Marta!

Migue[1]

[1] Miguel González Cano

A todos aquellos que siempre creyeron en mí....

Índice general

Índice de cuadros

Índice de figuras

Índice de listados

Listado de acrónimos

TFM	Trabajo Fin de Master
PE	Prácticas en Empresas
HTML	HyperText Markup Language
ESI	Escuela Superior de Informática
CSS	Cascading Style Sheets
RAE	Real Academia Española de la Lengua
IBM	International Bussines Machines Corp.
WIMP	Windows, Icons, Menus and Pointers
MS-DOS	MicroSoft Disk Operating System
W3C	World Wide Web Consortium
MVC	Modelo Vista Controlador
DOM	Document Object Model
AJAX	Asynchronous JavaScript And XML
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
ZF	Zend Framework
ASP	Active Server Pages
JSP	Java Server Pages
UCLM	Universidad de Castilla-La Mancha
TFG	Trabajo Fin de Grado
CAM	Comisión Académica del Máster
ESI	Escuela Superior de Informática
MII	Máster Universitario en Ingeniería Informática
DDP	Distributed Data Protocol

Capítulo 1

Introducción

En esta sección se introducirá el proyecto poniendo en contexto al lector de la problemática, de la situación actual y haciendo uso de la propuesta de trabajo, la motivación que ha llevado a realizarlo. Se completa el capítulo presentando un esquema del documento con las secciones que podrá encontrar.

1.1 Contexto

En un contexto de alto crecimiento de la información y de la conectividad de empresas, administraciones públicas y hogares de todo el mundo, la formación de profesionales en el ámbito de las tecnologías de la información y la comunicación es un factor decisivo para el progreso tecnológico y económico y la cohesión de nuestra sociedad.

En relación con esto y tras muchos años de esfuerzo, en la actualidad la Universidad de Castilla-La Mancha (UCLM) ofrece el **Máster Universitario en Ingeniería Informática**[1] **(MII)**, una Titulación Oficial que tiene como objetivo desarrollar el dominio de competencias y destrezas de dirección a distintos niveles (proyecto, equipo de trabajo, departamental y corporativo) y potenciar el conocimiento de tecnologías punteras. Este Máster está destinado a aquellos ingenieros que desean adquirir las competencias necesarias que se predisponen para optar a puestos de mayor responsabilidad en el ámbito laboral. Al tratarse de un Máster Oficial, éste tiene reconocimiento en todo el Espacio Europeo de Educación Superior. Además, este Máster permite explorar las tendencias y tecnologías informáticas más actuales en el campo profesional, como, por ejemplo, el análisis de grandes volúmenes de información, la computación distribuida o el desarrollo de el internet de las cosas.

Para completar la formación del estudiante de cara al mercado laboral se deberá realizar una de las actividades más importantes del programa, el **Trabajo Fin de Máster (TFM en adelante)**. El TFM consiste en la planificación, realización, presentación y defensa, una vez obtenidos todos los créditos del plan de estudios, de un ejercicio original realizado individualmente ante un tribunal universitario, consistente en un proyecto integral de Ingeniería en Informática de naturaleza profesional en el que se sinteticen las competencias adquiridas en las enseñanzas. Su finalidad es propiciar la aplicación de las habilidades y los conoci-

[1] http://webpub.esi.uclm.es/spa/paginas/formacion-master-ing-informatica-1

mientos adquiridos en el resto de las materias del Máster, así como facilitar el desarrollo de competencias relevantes. La importancia de un trabajo de estas características radica en que fomenta en el alumno habilidades tan relevantes como ser capaz de seleccionar un tema; planificar un proceso de análisis y estudio del tema seleccionado, establecer unos objetivos a alcanzar en el mismo; y ofrecer y defender una respuesta lógica y justificada al problema planteado de manera formal.

Por otro lado, las **Prácticas en Empresas (PE en adelante)** suelen representar la primera puerta de entrada al mundo laboral de futuros egresados, y también suele ser la primera oportunidad para aquellas personas que quieren reorientar su carrera profesional y han adquirido una nueva formación. Tiene como principal objetivo permitir a los/las estudiantes aplicar y complementar los conocimientos adquiridos en su formación académica favoreciendo, al mismo tiempo, la adquisición de competencias que les preparen para el ejercicio de actividades profesionales, faciliten su empleabilidad y fomenten su capacidad de emprendimiento. Al hacer las prácticas no sólo se descubre si las expectativas (muy elevadas por lo general) son ciertas en relación a lo que es un puesto de trabajo acorde a lo que se ha estudiado, sino que además sirve como toma de contacto con el mundo de la empresa, ese gran desconocido para muchos/as de los/as futuros titulados/as.

1.2 Motivación

En la actualidad, las instituciones educativas y en este caso la UCLM disponen de plataformas virtuales de gestión de contenidos de aprendizaje también conocidas por sus siglas en inglés *LCMS (Learning Content Management System)* [FA10]. La más utilizada es *Moodle*, también conocida como *Campus Virtual*, un sistema de gestión de cursos, de distribución libre, que ayuda a los educadores y profesores a crear comunidades de aprendizaje en línea.

A pesar de las continuas actualizaciones de la plataforma, aún no posee un sistema que pueda dar el potencial que necesitan los tutores y directores de proyectos a la hora de automatizar la gestión de los TFM con todo lo que ello conlleva, sin hablar de las particularidades y necesidades que tiene al final cada titulación. Actualmente y debido a la ausencia de un sistema que lo gestione, el Coordinador del Máster es el encargado de crear formularios web que sirven para recoger la información necesaria de las propuestas, y de gestionar el envío de propuestas por medio de e-mails a la Comisión Académical del Máster (CAM) que las valora. En esencia, este proceso aún no está bien automatizado ni integrado en ningún otro sistema de gestión por lo que las partes implicadas, y principalmente, el Coordinador del Máster, se ven obligados a emplear otras herramientas externas. Se adjunta un diagrama de flujo con la secuencia de trabajo que se detalla en las siguientes líneas.

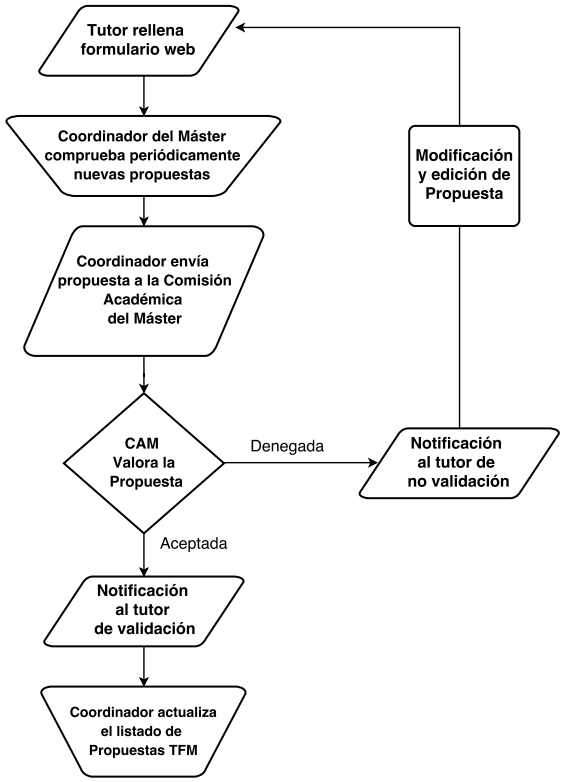

Figura 1.1: Diagrama de Flujo de la inserción de Trabajos Fin de Máster en la actualidad.

A continuación se detalla brevemente el flujo de información (ver figura 1.1) se sigue para este cometido en la actualidad:

1. El tutor del TFM rellena un formulario web con la información del TFM. La información de la propuesta serán los datos del tutor, del TFM y los datos del potencial estudiante al que se asigna.
2. El Coordinador del Máster recibe notificaciones o bien comprueba periódicamente si hay nuevas propuestas de TFMs a través del formulario.
3. Por cada nueva propuesta de TFM, el Coordinador envía la información de la misma a los miembros de la Comisión Académica del Máster (CAM).
4. El Coordinador del Máster notifica al tutor del TFM mediante e-mail si la propuesta ha sido aceptada o no. Si no lo ha sido se vuelve al punto 1 para editarla y se repite el flujo. Si ha sido asignada, entonces el Coordinador actualiza el listado de TFMs propuestos o asignados.
5. Si la propuesta ya estaba preasignada a un estudiante, el tutor le notifica a éste que la propuesta ha sido validada por la CAM.

Como puede observarse el procedimiento actual es tedioso y no tiene un enfoque práctico para el Coordinador. Además es totalmente opaco al alumno, ya que es el tutor el que

realiza las propuestas y el que recibe las notificaciones. El Coordinador hace uso de herramientas externas para poder recibir las peticiones y éste debe comunicarse vía e-mail con los miembros de la CAM, de nuevo haciendo uso de servicios externos, cuando lo ideal sería tener todas estas características incluidas en un sistema o plataforma como Campus Virtual, facilitando las tareas a los miembros implicados en el proceso de gestión e incentivando la interacción y la participación por medio de notificaciones o alertas vinculadas a un sistema que las automatice.

La figura 1.2 ilustra el flujo de trabajo que se sigue en la actualidad para este cometido, el cual se describirá en las siguientes líneas.

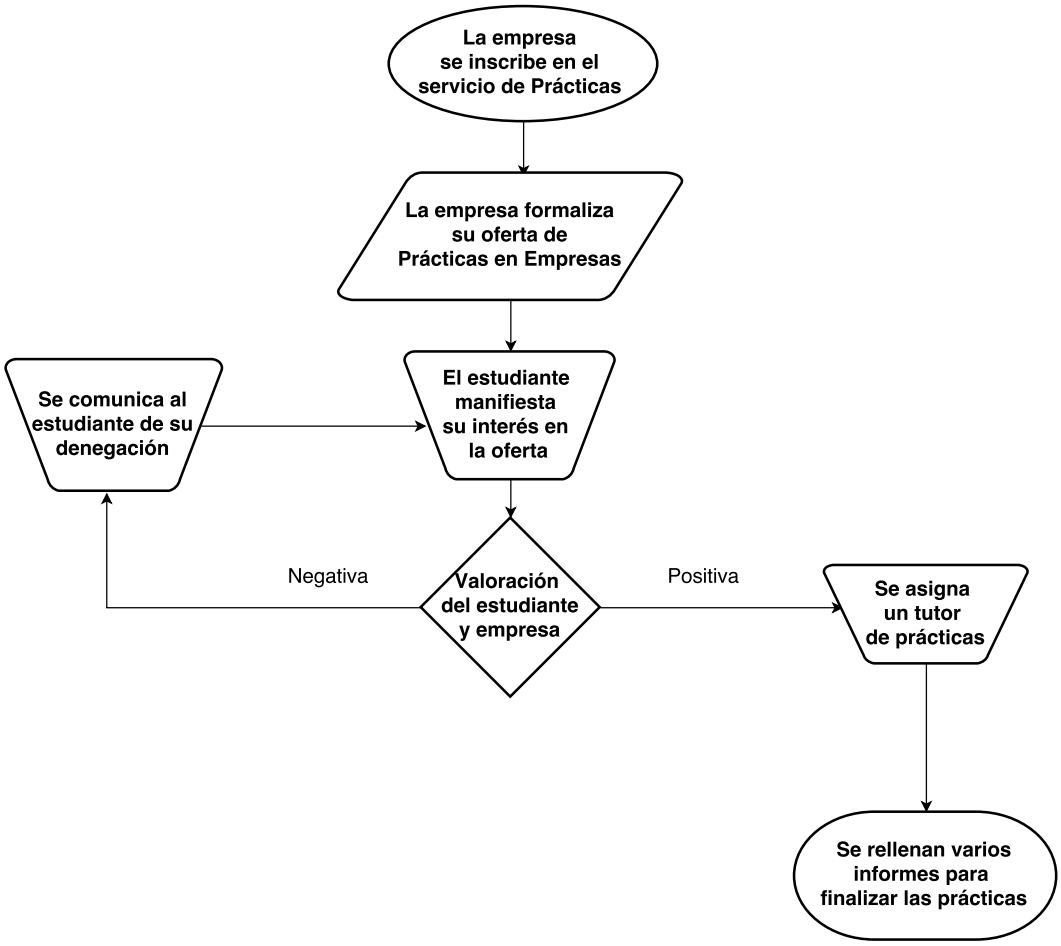

Figura 1.2: Diagrama de Flujo de la gestión de PE en la actualidad.

La parte referente a las PE actualmente son gestionadas a nivel institucional mediante laUCLM, siguiendo el siguiente flujo de trabajo:

1. La empresa se da de alta en el servicio de Practicas en Empresas de laUCLM.
2. La empresa realiza su oferta de PE.
3. El estudiante manifiesta su interés en la oferta de PE.

4. En coordinación con el Coordinador del Máster o del responsable de PE se nombra un tutor (profesor del Máster) que supervise el proceso de PE.
5. Se rellenan varios informes para finalizar las PE.

Debido a que el proceso esta gestionado por la UCLM, se podría plantear la posibilidad de reflejar el interés de los estudiantes en diversos temas para que las empresas pudieran identificar rápidamente si alguno los estudiantes tiene un perfil que encaje con sus líneas de trabajo. Las ofertas de PE se podrían formalizar y de esta forma poder ser notificadas a los alumnos inscritos en el sistema, dando la posibilidad de recibir la información en sus dispositivos. En definitiva, incluir todo esto en un sistema y centralizar todos los procesos en una herramienta o aplicación confiere una serie de ventajas y facilidades muy extensa.

En este contexto, el objetivo principal de este TFG consiste en diseñar, implementar y desplegar un sistema web que automatice y facilite la gestión de TFM y PE, y que además permita y fomente la interacción entre estudiante y profesor. Con la implantación de este TFG se pretende dar un enfoque más práctico y accesible a la hora de realizar una propuesta de TFM, tanto por la parte de la coordinación y tutores, como por la parte del alumnado, además de poder estudiar las ultimas novedades en plataformas de desarrollo web, con la finalidad de adaptarlo a las tecnologías móviles y haciéndolo accesible y transparente a todas las partes implicadas. Por la parte de las PE permitirá asociar una serie de líneas de trabajo, identificadas por palabras clave, al perfil del estudiante, además de poder añadir empresas del sector en las que estén interesados en realizar las prácticas. De este modo se pueden identificar rápidamente los intereses del estudiante, a nivel individual, y del grupo, a nivel colectivo.

1.3 Propuesta

El uso de tecnologías, plataformas y sistemas web cada vez está más extendido gracias a los teléfonos inteligentes. Esto nos da la posibilidad de poder ingresar y gestionar cualquier aplicación web en cualquier lugar. Aprovechando todas las opciones que nos brindan hoy en día estas plataformas, se plantea la posibilidad de desarrollar un sistema web, que a su vez se pueda acceder en forma de aplicación móvil, para la automatización y gestión de TFMs y PE, además de facilitar e incentivar la interacción entre el estudiante y profesor.

La aplicación o sistema web a desarrollar en este TFG tendría como principales funciones la gestión de TFM y PE por parte del Coordinador, la posibilidad de formalizar las propuestas de los TFM y PE por parte del tutor, y la posibilidad de visualizar las propuestas por parte del alumno interesado.

Cada usuario al ingresar en la aplicación tendrá unas funciones diferentes en función al rol seleccionado y, posteriormente, validado por el administrador. Esto permite enfocar el sistema directamente al usuario y permitir el acceso a las funciones que le competen:

- **El Coordinador del Máster** hará las funciones de administrador del sistema, lo que se traduce en la validación de usuarios y la total gestión de TFM y PE. Será el encargado de verificar, enviar y gestionar las propuestas de TFM y su posterior veredicto como representante de la CAM. Respecto a la parte de PE, surgirá el rol de Coordinador de PE y podrá visualizar el estado de cada estudiante, para identificar si esta cursando o no las PE y poder visualizar las lineas de trabajo asociadas a PE más demandadas por los alumnos. Es el rol más importante de esta parte del sistema y se encargará de notificar y validar a la CAM en el sistema.
- **Un miembro de la Comisión Académica del Máster** por su parte podrá verificar las propuestas de TFM que le han sido notificadas y validarlas mucho más fácilmente que con el método que actualmente se lleva a cabo.
- **El tutor** será el encargado de inscribir las propuestas de TFM en nombre del alumno, dándole la posibilidad de realizarlas de una manera sistemática, simple y funcional. Una vez aprobadas le será enviada una notificación con la resolución de la propuesta.
- **El alumno** tendrá la posibilidad de ver las propuestas tanto de TFM como de PE y realizar un seguimiento de su resolución, además de facilitar la interacción entre el tutor y el alumno. En el caso de las PE el alumno podrá editar su perfil asociando una serie de lineas de trabajo, en forma de palabras clave, además de una serie de empresas en las que estaría interesado en realizar las prácticas. De este modo se pueden identificar rápidamente los intereses del alumno y poderle ofrecer unas ofertas más personalizadas. Además de estar informado en todo momento de las vacantes o plazas libre que se oferten por medio de notificaciones personalizadas.

Con este sistema se propone facilitarle a todas las partes implicadas la automatización del proceso de gestión de inscripciones y propuestas de TFM y PE, promoviendo y fomentando la participación y el seguimiento por parte del alumno y tutor, y ayudando en la tarea de gestión a la parte encargada de la coordinación del Máster.

1.4 Estructura del documento

Capítulo 1 - Introducción

Capítulo actual, donde se presenta e introduce el proyecto y la motivación que ha llevado a realizarlo.

Capítulo 2 - Objetivos

Se presentan los objetivos que pretenden cumplirse en este proyecto,

Capítulo 3 - Estado del arte

Se comenta el estado actual y la evolución de las interfaces de usuario, así como en el diseño y desarrollo de la web. Se habla de las plataformas de computación en la nube y se introduce a la plataforma utilizada, Meteor.

Capítulo 4 - Metodología de trabajo

Se describe la metodología de trabajo seguida en el proyecto, así como las herramientas utilizadas durante su desarrollo.

Capítulo 5 - Arquitectura

Se presenta y desarrolla la arquitectura de la aplicación web, indicando todos los aspectos relevantes y detallando todos los módulos y funcionalidades que contiene el sistema.

Capítulo 6 - Evolución y resultados

Se discute la evolución que ha seguido el proyecto a través del tiempo entre iteraciones, desde el inicio hasta las pruebas finales. Además se reflejará el resultado de la aplicación web, así como un caso de estudio sobre el despliegue y los primeros pasos a realizar.

Capítulo 7 - Conclusiones y trabajo futuro

Se dará la conclusión del proyecto y se hablará del trabajo futuro que queda para la mejora y adaptación de la aplicación.

Anexo A - Manual de usuario de la aplicación web

Se comentará a modo de manual de usuario, cómo realizar el despliegue de la aplicación, el acceso a la misma y unos detalles sobre el uso básico de las funciones según el tipo de usuario.

Capítulo 2

Objetivos

En este capítulo se planteará el objetivo general del proyecto junto a una descripción más detallada de los distintos objetivos específicos, más funcionales o didácticos.

2.1 Objetivo general

El objetivo principal de este proyecto es el diseño, implementación y despliegue de una aplicación web que que automatice y facilite la gestión de Trabajo Fin de Master (TFM) y Prácticas en Empresas (PE) y que además permita y fomente la interacción entre estudiante y profesor. Esto viene motivado por los siguiente aspectos:

- Necesidad de **implementación una aplicación que sustituya al actual método** de propuestas de Trabajos Fin de Máster.
- **Aprendizaje de un *framework***[1] que posibilite el despliegue de una aplicación en la nube y que permita el diseño de una aplicación web que permita lainteracción de los usuarios y la gestión de la plataforma.
- **La integración de una plataforma de desarrollo novedosa** como es *Meteor* en un proyecto para un fin educativo y que tiene como objetivo la propia integración en la gestión administrativa.
- **Aprender el enfoque didáctico** orientado, no solo a realizar un proyecto real, sino también al estudio de problemas que surgen en un proyecto de esta magnitud.
- **La puesta en práctica de todo lo aprendido en la intensificación** de Tecnologías de la información y todo lo relacionado con las tecnologías web.

Conla implantación de este TFM se pretende dar un enfoque más práctico y accesible a la hora de realizar una propuesta de TFM, tanto por la parte de la coordinación y tutores, como por la parte del alumnado, adaptándolo a las tecnologías móviles y haciéndolo accesible y transparente a todas las partes implicadas.

[1] Framework es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software

2.2 Objetivos específicos

Los objetivos específicos están divididos en funcionales (funcionalidades que se esperan del proyecto) y didácticos (conocimientos que se esperan adquirir con el proyecto).

2.2.1 Objetivos Funcionales

- **Diseño e implementación de un sistema o aplicación web** que de soporte a la gestión y automatización de propuestas de TFM y PE para el Máster Universitario en Ingeniería Informática de la ESI.
- **Diseño de una arquitectura** para pa adecuación de plataformas de desarrollo innovadoras que fomentan la integración y el uso centrado al usuario.
- **Proporcionar un sistema fiable** basado en autenticación y validación de usuarios.
- **Creación de un sistema reutilizable** a pesar de posibles cambios en la administración y coordinación del máster.
- **Diseño de un sistema web escalable y fácilmente accesible** desde todo tipo de dispositivos conectados a internet, principalmente orientado al usuario y al acceso a la información.
- **Fomentar el uso de estándares, tecnologías y plataformas libres** y novedosas en el desarrollo de aplicaciones y sistemas web, dando lugar a aplicaciones versátiles, intuitivas y escalables.

2.2.2 Objetivos Didácticos

- **Estudio y aprendizaje de HTML[2], JavaScript[3] y CSS[4]**: Se utilizarán estos lenguajes como lenguajes para el proyecto. HTML para la parte exclusiva del cliente. Javascript para dotar de funcionalidad tanto al cliente como al servidor. CSS para dar estilo y maquetar la aplicación. Esto permitirá el estudio en profundidad de tecnologías web y el desarrollo de aplicaciones web.
- **Estudio del Framework[5] de diseño de aplicaciones Web:** se estudiará y utilizará el Framework de aplicaciones web Meteor que será el encargado de motorizar nuestra aplicación.
- **Aplicación de la intensificación: Tecnologías y Sistemas Web:** La aplicación web que se desarrolle con este *framework* será, a efectos prácticos, un ejemplo de sistema

[2]Siglas en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto)

[3]JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

[4]Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

[5]Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

o aplicación web. Por tanto, se estudiarán conceptos relacionados con las tecnologías y sistemas web.

- **Utilización de paquetes externos:** El *framework* permite la posibilidad de agregar o instalar paquetes al proyecto. Esto es similar a la utilización de librerías externas que permiten la utilización de funciones y variables ya definidas para poder utilizarlas y fomentar la reutilización de código.
- **Aprendizaje entre distintos lenguajes interoperables** este proyecto utilizará varios lenguajes de desarrollo y cada uno para un fin específico, por lo que será de gran interés para la formación del alumno.
- **Realización de proyecto complejo:** Al tratarse de un proyecto complejo, prolongado en el tiempo y con múltiples requisitos, se pretende diseñar un organigrama de trabajo para la organización del proyecto.
- **Organización del trabajo:** deberá encontrarse una metodología de trabajo que permita terminar el proyecto en el mínimo tiempo posible.

Capítulo 3

Estado del Arte

En este capítulo de realizará un estudio acerca del estado del arte en las áreas de trabajo ligadas a ese Trabajo Fin de Grado. En concreto, se estudiarán las interfaces de usuario y todo lo relacionado, el diseño y el desarrollo en el ámbito de la web incluyendo las herramientas y lenguajes más utilizados y un repaso sobre la computación en la nube y como se desarrolla ese modelo. Por último se darán unas pinceladas al *framework* que se ha utilizado en este proyecto, llamado *Meteor*.

3.1 Diseño de interfaces de usuario

3.1.1 Introducción y evolución

La Interfaz de Usuario, también llamada Interfaz Gráfica de Usuario, en inglés *GUI*[1] , de un programa, aplicación o web, es un conjunto de elementos principalmente software que representan información al usuario y las acciones disponibles que le permiten interactuar con dicha información y con el dispositivo. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo, aplicación o web, por lo tanto, es la pieza fundamental de un entorno gráfico.

La etimología de la palabra **interfaz** viene del inglés interface, formado del prefijo *inter* (entre) y *face* (cara), es decir *entre-caras*, o como la RAE lo recoge en su diccionario, *"superficie de contacto"*[2].

En el contexto de la interacción persona-ordenador, hablamos de interfaz gráfica de usuario, para referirnos a la relación general que existe entre un sujeto y un ordenador, sistema interactivo o dispositivo electrónico. La interfaz de usuario, es esa *ventana* de un sistema informático, que ofrece la posibilidad a una persona interactuar con él y *manejarlo*. El concepto de interfaz gráfica de usuario tiene una localización determinada y definida: Si la interfaz etimológicamente supone la cara o la superficie de contacto, la interfaz gráfica de usuario, supone un tipo específico de interfaz que usa metáforas de tipo visual y signos gráficos como paradigma interactivo entre la persona y el dispositivo.

[1] Graphical User Interface, Interfaz gráfica de usuario
[2] http://dle.rae.es/?w=interfaz&origen=REDLE

La interfaz gráfica de usuario como tal, exige por parte del usuario una serie de condicionantes y necesita del uso de herramientas que permitan el contacto con el dispositivo; estas herramientas son llamadas dispositivos de entrada, como el ratón o el teclado, que permiten interactuar con la interfaz y ser parte fundamental de la misma. Por lo tanto, la interfaz cobraría mucho más sentido cuando el usuario es capaz de comprender el significado y el proceso de interacción y es capaz de interpretar adecuadamente los elementos que componen la interfaz por medio de las herramientas [SPC+16].

Elementos de la Interfaz de Usuario

Los elementos de la interfaz suponen metáforas o elementos simbólicos que tienen su propia gramática de representación e interacción, suponiendo que este modelo es comprendido y entendible para todos los usuarios que estén dispuestos a interactuar con el sistema o dispositivo.

Las **barras de navegación, ventanas, iconos, menús y dispositivos externos** son los primeros elementos interactivos que surgen para relacionarse con los ordenadores o sistemas. Para ayudar en el proceso entre usuarios y computadores, se crean metáforas como la del *escritorio*, que consiste en representar un espacio de trabajo con los elementos y acciones del sistema a través de algo que conocemos [LA09]. Aquí surgen los elementos de carpeta, documento, herramienta, pincel, puntero, etc.

Historia de la Interfaz de Usuario

La historia de la interfaz gráfica ha estado marcada por dos factores importantes: el negocio y la investigación. El verdadero origen de la interfaz de usuario estaba motivado en la búsqueda de un modelo de interacción sencillo y amigable con los ordenadores que superase a la línea de comandos, las primeras herramientas que a modo de interfaz proveían una interacción con la máquina. La *guerra de las interfaces* ha estado ligada a la guerra de los sistemas operativos que gobiernan los dispositivos desde hace ya bastantes años. Poseer una interfaz propia, es de algún modo, poseer una herramienta poderosa de control sobre los usuarios que la utilizan. Esta es la razón por la que las principales empresas del sector han mantenido verdaderas luchas de poder por dominar el territorio de la interfaz demandándose unas a otras, de ahí la *guerra*.

La interfaz gráfica de usuario podemos dividirla en 3 periodos, los cualidades han venido marcados por su investigación y madurez en la sociedad. Estos 3 periodos vienen motivados por la investigación y aproximación semiótica y cognitiva de la interfaz gráfica de usuario, trabajo en el cual se ha influenciado para relatar la historia. [ME06].

Primer Período. Nacimiento de la Interfaz Gráfica (1970-1981)

El primer periodo de la historia de la interfaz gráfica viene motivado por la investigación y la búsqueda de un *paradigma*[3] de interacción definitivo y óptimo, que sustituye-

[3]Ejemplo o modelo de algo

se la práctica, pero a su vez compleja, interfaz de línea de comandos. En estos primeros años cabe destacar la labor de los pioneros en la investigación, los cuales trataron de dar forma a la interfaz gráfica. Uno de estos primeros protagonistas fue el centro de investigación *PARC* [4], una división que la empresa *Xerox Corporation* construyó en el 1970. En este centro se desarrollaron las investigaciones más importantes de este periodo de computación.

En el año 1973, nace el primer ordenador que incluiría la primera interfaz gráfica de la historia, el **Xerox Alto.** La primera intención por su parte fue la de desarrollar un sistema informático pequeño y transportable con capacidad para ser instalado en una oficina. Debería de tener capacidad para poder manejar un sistema operativo con interfaz gráfica y compartir información de manera sencilla. Esto había puesto en alza la primera aproximación al concepto de ordenador moderno.

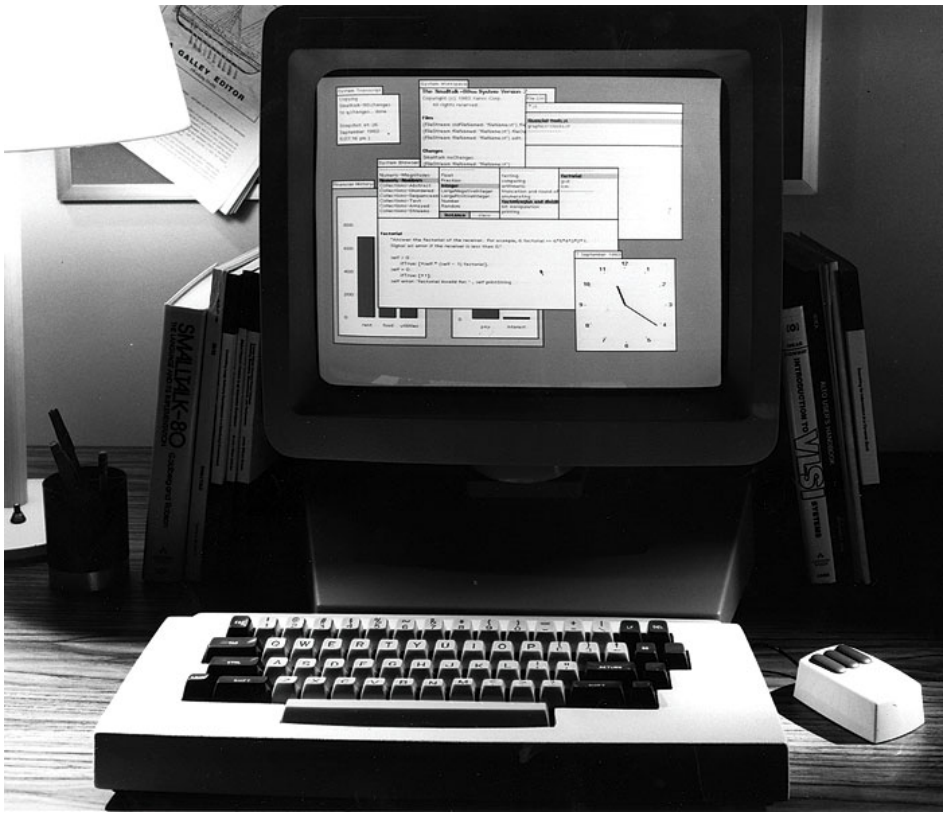

Figura 3.1: Imagen del Xerox Alto. [Im116]

El *Xerox Alto* (Imagen 3.1) poseía una interfaz gráfica muy básica en blanco y negro, con la que se podía interactuar por medio de un ratón. No fue implantado un sistema de ventanas en primera instancia y fue la primera aproximación realizada al paradigma de interacción WIMP[5], sin llegar a la metáfora de escritorio.

[4]Palo Alto Research Center
[5]Windows,Icons,Menus and Point Devices, es español, Ventanas,Iconos,Menús e Interfaces Humanos

En el Año 1981 fue concebido el sucesor del Xerox Alto, el **Xerox Star 8010**. El ordenador fue etiquetado como *oficina del futuro* y fue el primer ordenador que introducía una interfaz de usuario incluyendo y aplicando la metáfora de escritorio. Con ello se desarrolla e implementa el concepto de *WYSIWYG*[6] en la interfaz, lo que ves, es lo que puedes tener. El sistema Xerox Star disponía de los dispositivos de entrada ratón y teclado, que ya habían sido incluidos en el Xerox Alto. También estandarizó una serie de funciones generales para facilitar la interacción con todo tipo de archivos y de este modo mejorar la usabilidad del sistema. Estos comandos básicos serían los de copiar, abrir, mover, borrar, mostrar y borrar propiedades [Ray05].

Segundo Periodo. Desarrollo de las interfaces de usuario (1981-1995)

El segundo periodo de la evolución histórica de las interfaces gráficas está íntimamente ligada a la revolución de los ordenadores personales surgida en 1981. Para la interfaz gráfica, este periodo significa su implantación definitiva en los hogares y oficinas de trabajo. A partir de este año se produce un enorme despliegue industrial en la venta de ordenadores personales y su implantación en la sociedad, se introduce el concepto de ordenador personal (PC)[7], y este éxito vendrá condicionado en gran parte por la capacidad interactiva de la interfaz de usuario. Durante este periodo es cuando definitivamente los modelos vigentes que definían la interacción persona-ordenador son definidos y ampliados. Por un lado *Apple* acabaría definiendo el modelo incluido en su MAC OS[8], y por otro lado el modelo de Windows quedaría desarrollado por *Microsoft* al final de este periodo, ambos inspirados y herederos del modelo desarrollado en Xerox Parc.

Los principales protagonistas de este periodo son IBM, *Microsoft*, *Apple* y el proyecto de software libre *GNU/Linux* [9]. Cada una de estas empresas ha tenido un papel importante en la evolución de la interfaz y hará sus aportaciones definiendo patrones de interacción que hoy en día siguen utilizándose.

Comenzando en 1976 con el ordenador *Apple I* y tras haber invertido todos sus beneficios en el diseño y producción del sucesor *Apple II en 1977*, La empresa *Apple Computers*, formada por dos aficionados a la electrónica llamados Steve Jobs y Steve Wozniak, junto a antiguos miembros de Xerox Parc, lanzarían al mercado el **Apple Lisa** en 1983. Este supondría el primer ordenador *Apple* con interfaz gráfica de usuario. La novedad de esto está en el hecho de realizar el primer prototipo de ordenador personal con interfaz gráfica listo para ser introducido en el mercado y poder ser vendido de forma masiva. El *Apple* Lisa se diseñó en primer lugar orientado al mercado de oficina y grandes empresas, y tras sus fracaso, apostaría por el modelo orientado al

[6] What You See Is What You Get, lo que ves es lo que tienes
[7] Personal Computer
[8] Abreviatura de Macintosh Operating Sistem
[9] https://www.gnu.org/gnu/linux-and-gnu.es.html

mercado de la informática personal, con el **Apple Macintosh.**

Un aspecto importante fue la consistencia de la interfaz. Este concepto es integrado a la hora de organizar los menús donde se mantiene un orden que es aplicado de forma consistente, y esto ayudaba a recordar donde estaban las funciones, favoreciendo y reduciendo la curva de aprendizaje. Otro aspecto importante era la simplicidad, eliminando información redundante de la pantalla.

Ese mismo año, se hizo posible la colaboración entre IBM y *Microsoft*, la cual adaptaría su sistema operativo MS-DOS a una interfaz gráfica de usuario para operar en ordenadores IBM en el año 1985. Este sistema operativo seria **Windows 1.0.**

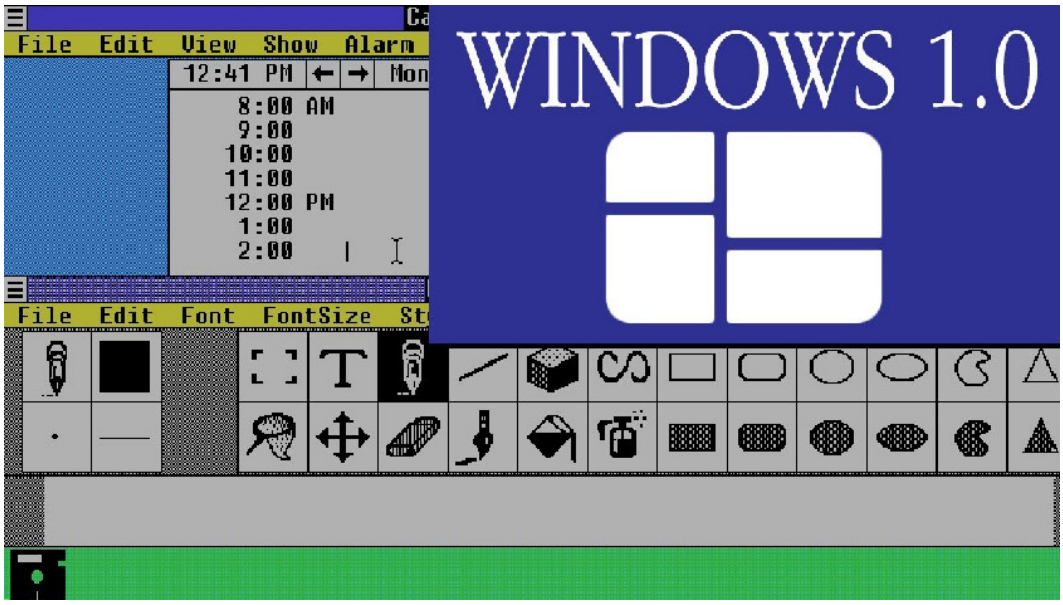

Figura 3.2: Interfaz gráfica Windows 1.0 [Im316]

La interfaz gráfica de *Windows 1.0* (Imagen 3.2) poseía un administrador de archivos, una calculadora, un calendario, un reloj, un blog de notas y un terminal, antigua interfaz de línea de comandos. La interfaz aún presentaba un aspecto rudimentario, muy alejado de la actualizad y como peculiaridad, su sistema de ventanas que establecía por defecto la ventana maximizada, al contrario de lo que hacían otras empresas solapando ventanas, pero incluyendo iconos de redimensión y la posibilidad de acceso a menús contextuales. Una de las novedades incluidas era una serie de iconos en la parte inferior que representaban las aplicaciones activas en el sistema, lo que daría lugar al origen de la futura barra de tareas que tanto éxito tuvo en la versión 95.

Tercer Período. Maduración y actulidad de la Interfaz Gráfica(1995-Actualidad)

Este período en el que ya se han desarrollado los conceptos básicos y se ha llegado a un modelo óptimo de interacción, la interfaz gráfica entra en un proceso de desarrollo centrado en la estética, provocadas por las necesidades mercantiles del producto frente a la competencia.

En octubre de 1995 *Microsoft* lanzó **Windows 95** (Imagen 3.3. Este sistema operativo significa el inicio del imperio *Microsoft* en el mercado de software informático. Uno de los cambios importantes que introduce este sistema operativo es convertir la interfaz inicial orientada a aplicaciones, en una orientada a objetos. Esto tiene implicaciones sobre la ampliación del paradigma WIMP, provocando cambios importantes. Ahora el escritorio, por ejemplo, no sólo servirá para representar aplicaciones minimizadas o activas, sino además iconos que pueden ser manipulados, agrupados y activados desde allí. Otra de las novedades que incluye Windows 95 es el menú de inicio, al que se asocian las aplicaciones,archivos y funciones del sistema. El botón de inicio será uno de los grandes hallazgos que aún se mantiene en todas las interfaces actuales.

Los proyectos de Software libre **KDE (1998)**[10] **y GNOME (1999)**[11] tenían como objetivo desarrollar un interfaz gráfico que opere sobre sistemas operativos Unix[12], especialmente GNU/LINUX y que sea similar a lo que ofrecian Windows o Mac. Estos incorporaban ya elementos conocidos como las barras de tareas o lanzador de aplicaciones, incluso lo menús. Todo diseñado para ser personalizable y de uso libre.

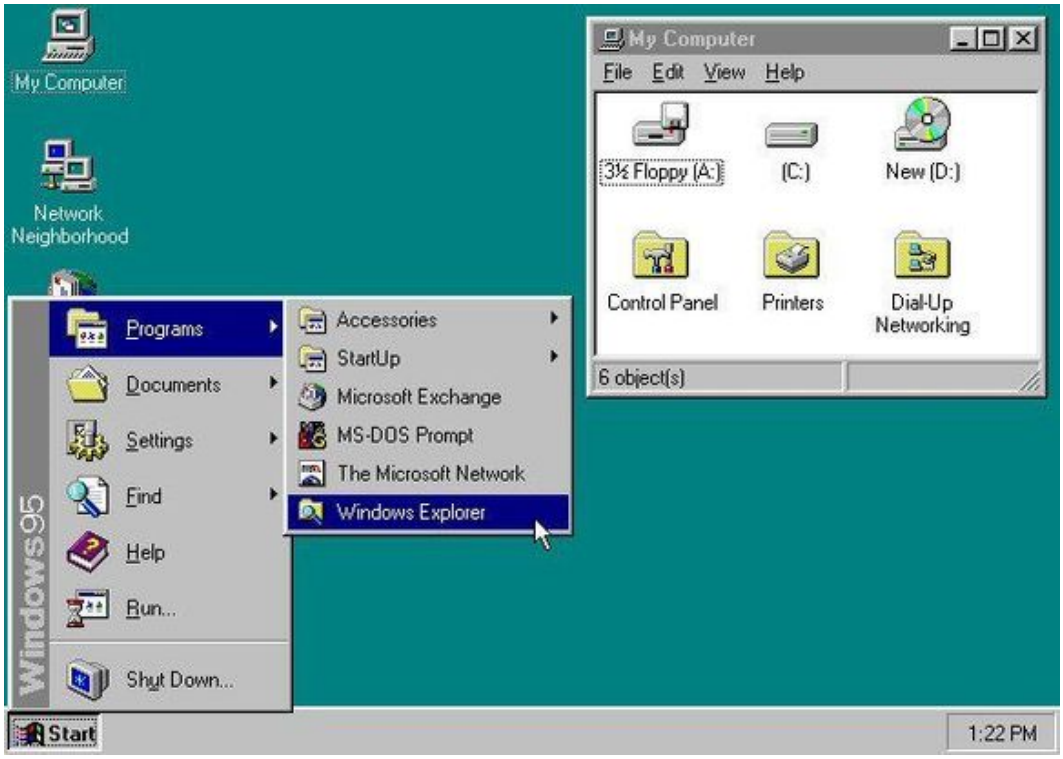

Figura 3.3: Interfaz gráfica Windows 95 [Im416]

En el año 2001 *Apple* y *Microsoft* lanzaron sendas versiones de sus sistemas operativos. *Apple* lanzaba **MAC OS X** y *Microsoft*, su **Windows XP.**

[10] https://www.kde-espana.org/
[11] https://www.gnome.org/
[12] http://www.opengroup.org/unix

MAC OS X estaba basado en tecnología *Unix*, al contrario de sus versiones anteriores y no solo cambiaba su arquitectura interna, sino que además renunciaba a toda una iconografía desarrollada hasta el momento para introducir su nuevo entorno gráfico *Aqua*. El primer elemento diferenciador que introduce la interfaz de Macintosh en el llamado *Dock*[13], ya introducido por *NEXTSTEP*[14], que hacía sus labores de barra de tareas. También incluía controles situados en la parte superior de cada ventana, posicionados en el lazo izquierdo y que representaban las funciones de cerrar, minimizar y maximizar ventanas. En relación a la personalización de la interfaz, *MAC OS X* ofrece varios tipos de visualización (forma de árbol, modo multicolumna, forma de iconos.. etc) y tenía la posibilidad de cambiar de apariencia.

Por su lado Windows, con su Windows XP se preocupaba por adaptar la interfaz al perfil del usuario. A través de diversos personajes, a modo de asistentes, añadía automatización inteligente a la interfaz y fortalecía la metáfora del asistente, dando consejos al usuario a la hora de realizar tareas sobre alguna acción concreta en el sistema.

Como hemos podido comprobar, la interfaz como artefacto está escrita en los ciclos de vida de cualquier producto industrial informático. Todo elemento fundamental tiene su historia que contar y corresponde a muchas investigaciones y pruebas de diseño. Las interfaces gráficas han pasado de convertirse, de un artefacto tecnológico que posibilitaba la interacción con el ordenador, a constituirse como artefacto inteligente capaz de orientar al usuario y provocar cambios sobre sí misma.

Las interfaces gráficas están actualmente abiertas a procesos de personalización, permitiendo que el usuario modifique todo lo relacionado con el aspecto visual para adaptarlo a sus gustos.

3.1.2 Principios Básicos

A la hora de construir interfaces de usuario debemos tener presente una serie de principios básicos para intentar alcanzar los niveles de interacción exigibles para las aplicaciones actuales. Estos principios básicos guiarán posteriormente el proceso de especificación de los requisitos de interfaz de usuario e inspiran buenas soluciones en la fase de construcción de interfaces de usuario.

Estos principios fueron estandarizados por IBM en 2001 para crear guías de estilo robustas y de calidad [Sim86].

1. **Consistencia.**

   La consistencia implica un conjunto de estándares que han de ser seguidos dentro

[13]Dock o Muelle es un elemento de interfaz gráfica de usuario que permite a los usuarios iniciar, cambiar y monitorear aplicaciones

[14]NeXTSTEP es el sistema operativo orientado a objetos, multitarea que NeXT Computer, Inc. diseñó para ser ejecutados en los computadores NeXT.

de la aplicación y entre aplicaciones. Los objetos que poseen una apariencia o comportamiento similar son interpretados de manera consistente. Los beneficios pueden resumirse en los siguientes: ayudar a los usuarios a aprender y reconocer fácilmente el lenguaje gráfico de esa interfaz, sistemas más predecibles, mejora de la curva de aprendizaje, uso de distintas aplicaciones sin instrucciones adicionales.

2. **Familiaridad.**
   Las metáforas del mundo real, el lenguaje orientado al usuario, las metáforas visuales (iconos), etc. aportan familiaridad al usuario. El principio de la familiaridad puede ser vista como la consistencia con el mundo real. Este concepto ayuda a los usuarios a usar su conocimiento del mundo real para interactuar con el sistema, simplificando el proceso de aprendizaje ya que se apoya en conceptos del mundo real que el usuario ya ha experimentado antes.

3. **Simplicidad.**
   Cuanto más simple mejor. El usuario no debe sentirse desbordado en ningún momento por los detalles. Cuanto más simple es la interfaz, más fácil es que el modelo mental del usuario coincida con el de la aplicación. La simplicidad facilita mucho el aprendizaje del sistema. Sin embargo, la simplicidad puede ser un lastre para los usuarios experimentados que dificulta otro principio, a veces contrapuesto como puede ser el principio de la eficiencia. En estos casos, hay que alcanzar un equilibrio entre ambos principios.

4. **Flexibilidad.**
   Las interfaces deben adaptarse a las necesidades del usuario. La personalización del sistema permite ajustar el sistema para un usuario especifico con unas determinadas necesidades. La mayoría de guías de estilo sugieren que debería proporcionarse más de una manera de llevar a cabo una misma tarea.

5. **Retroalimentación.**
   Todo usuario debe saber en todo momento qué está sucediendo en el sistema, por consiguiente, todas las acciones del usuario han de tener como respuesta una rápida retroalimentación que sea consistente y entendible para el usuario. De este modo, los usuarios permanecen centrados en su tarea.

6. **Aproximación gradual.**
   Toda interfaz de usuario, debe organizar y presentar la información al usuario de manera gradual. Siguiendo la regla de 7 ± 2 elementos, debemos cuidar no abrumar al usuario cuando vaya a tomar una decisión con un número elevado de opciones disponibles ya que el número de elementos que los humanos podemos recordar en la memoria inmediata está en torno a nueve elementos. La agrupación o clasificación en grupos y subgrupos se convierte en necesidad cuando el número de opciones crece. Los menús de aplicaciones desplegables son un buen ejemplo de ello.

7. **Diseño visual.**
   El diseño visual engloba a la posición y la apariencia de los objetos. Las guías de estilo o *guidelines* favorecen la simplicidad y la claridad de diseños recargados con efectos visuales. De este modo, la atención recae sobre la tarea que el usuario realiza evitando despistes innecesarios sobre los detalles y adornos de la interfaz.

8. **Robustez.**
   Los usuarios son propensos a cometer errores, unas veces por la propia naturaleza humana y otras por que el usuario prefiere aprender mediante el ensayo-error antes que recurrir a leer la documentación. En el caso de que se produzcan acciones no deseadas, la posibilidad de deshacer las últimas acciones llevadas a cabo (deshacer) puede enmendar gran parte de los errores cometidos. Los usuarios valoran mucho las prestaciones de tipo deshacer puesto que les proporciona sensación de seguridad el hecho de saber que si comenten un error pueden enmendar lo con facilidad [MM03].

3.1.3 Aplicaciones de Gestión

Entenderemos como aplicación de gestión aquella que se diseña para sustituir uno o varios procedimientos, tanto comerciales como administrativos, que habitualmente realiza una persona en una empresa o institución de forma presencial, por una serie de pantallas en un ordenador u otro dispositivo, que permitan realizar al cliente los mismos procedimientos de forma no presencial. Esta definición también sería válida para la *oficina virtual*, vocablo de nuevo cuño muy en boca últimamente.

Luego podemos dar una definición de Aplicaciones de Gestión como la siguiente :

Son todos aquellos programas utilizados a nivel empresarial o particular, que por su definición genera acción de emprender algo y por su aplicación persigue fines lucrativos y no lucrativos. También es un software que permite gestionar todos los procesos de un negocio o de una empresa en forma integrada. Por lo general está compuesto por módulos cruzados de los procesos del negocio [Ano16].

Algunas características del Aplicaciones de Gestión en relación a una empresa son las siguientes [Web16]:

- Debe estar enfocado en una empresa u organización, donde debe existir coordinación y debe estar sustentada por la administración.
- Todas las áreas de la empresa integran el proceso de gestión de administración, que transforma los objetivos en resultados.
- Permiten construir la contabilidad y los movimientos financieros acuerdo a las normas establecidas en la empresa.
- Obtener beneficios que ayudaran a controlar y administrar los bienes de la Empresa.

- Permiten dar respuesta inmediata y oportuna a las necesidades de registro presupuestario, patrimonial, contable y administrativo en las organizaciones.
- Permite mantener el control de cada departamento, unidad o sección de la empresa.

3.2 Diseño y desarrollo web

Confundir el diseño y desarrollo web suele ser algo habitual, nombrando dichos conceptos indiferentemente, y refiriéndose, en algunos casos, a ambos a la vez. Los dos ocupan y tienen lugar en el mismo proceso, el de la elaboración de una aplicacion web, pero cada uno ocupa una parte bien diferenciada del proyecto por sí solo. Esto no quiere decir que estén separados por completo, de hecho el uno no existe sin el otro. [Luj16].

El encargado de la parte visual solía ser el mismo programador o un incluso un diseñador gráfico, habituado a trabajar con formatos de impresión. Por ello, las páginas carecían de movimiento e interacción. Era importante que el que fuera a encargarse de la parte visual comprendiera también las capacidades y limitaciones de la web. Fue cuestión de tiempo que los diseñadores se adaptaran a esta materia y se hablara de la figura del diseñador web. Además de la parte gráfica, el diseño web se ocupa de la experiencia de usuario. Determina los objetivos del proyecto y las necesidades de los usuarios. Define la arquitectura web, el número de páginas y bloques de contenido de los que dispondrá el sitio. Estudia la interacción de usuario, las funciones y la navegación que se realizará en la web y estructura el contenido en las diferentes páginas del sitio, mediante composiciones o plantillas, con el objetivo de elaborar prototipos o wireframes.

En definitiva, el **diseño web** cubre las fases del proyecto que se ocupan de la navegación, la usabilidad, la interacción, la arquitectura de la información y de la parte gráfica de la web.

Por otra parte, el **desarrollo web** es la programación necesaria para la construcción del sitio web. Se divide en dos partes que pueden estar o no conectadas, la parte del cliente y la parte del servidor. En la parte del cliente estaríamos hablando de HTML y CSS, código básico para creación de páginas web, *JavaScript*, para la interacción con el usuario. En la parte del servidor se trabaja con código más complejo, como es PHP, ASP.NET, JSP, etc. Con este código se construye el *back-end*, la parte de la web que el usuario no ve. Su objetivo es el diseño de bases de datos y asegurar la seguridad de la web. Cuando ambas partes se comunican, se habla de programación cliente-servidor. Esta comunicación permite la interacción del usuario con los contenidos alojados en bases de datos, el registro de nuevo contenido y de cuentas de usuario.

"La distinción entre "diseño" y "programación" o incluso la terminología aún más perturbadora de "técnico" y "creativo", es artificial. Están tan entrelazadas como el arte y la ciencia del propio diseño web [VM01]."

En resumen, es necesaria la existencia de estos dos perfiles en un equipo web y una buena comunicación entre ambos. *Esta comunicación, y finalmente la conversión entre disciplinas, es de máxima importancia*.Pensar como un equipo de desarrollo, incluso cuando el equipo está integrado por una sola persona, es un buen punto de partida [Cha06].

3.2.1 Evolucion de la Web

El diseño ha sido considerado, durante mucho tiempo, secundario en la creación de páginas y aplicaciones web. El principal motivo de esto fueron las limitaciones de la época a la hora de dar estilo a las paginas web. Las primeras etiquetas HTML permitían definir unos atributos básicos para introducir textos e imágenes, dejando al navegador la interpretación del estilo. Para dar presencia a la web se utilizaban gráficos que ocupaban el lugar de textos. Con la aparición del CSS[15] los diseñadores comenzaron a tener mayor control. La unión de HTML y CSS permitía separar el contenido de su presentación.

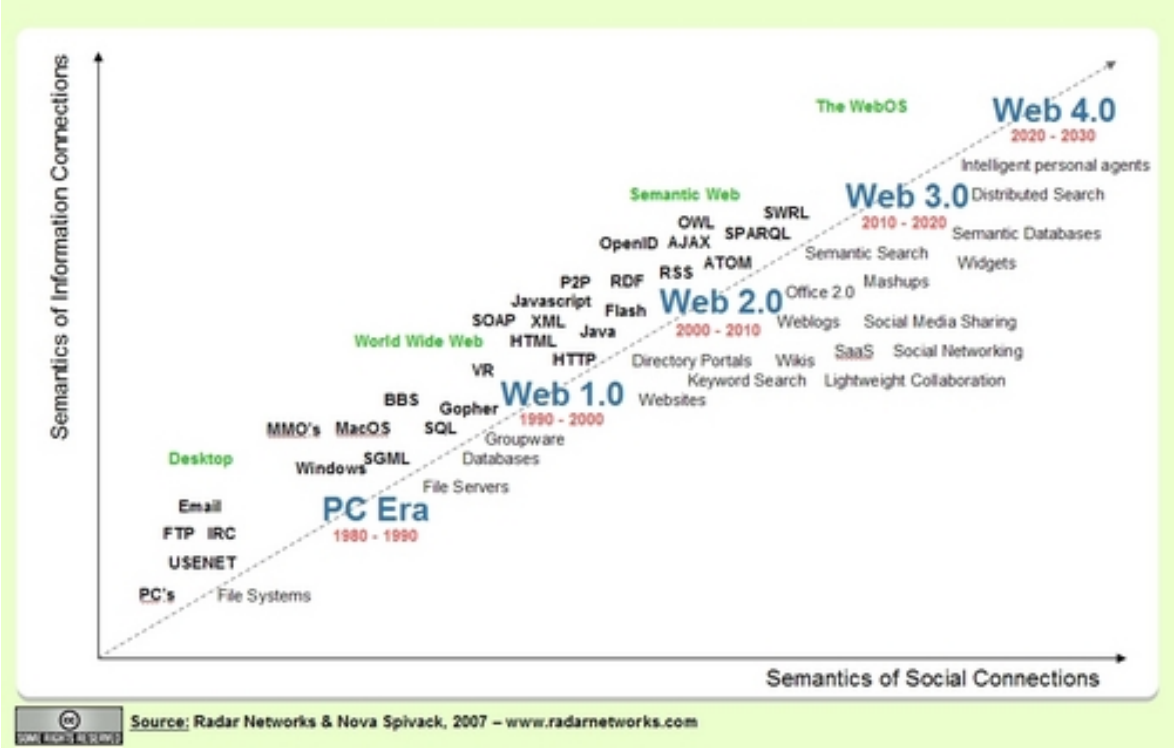

Figura 3.4: Evolución de la Web. [Im216]

Sin embargo, la web es algo más que palabras e imágenes, debe tener interacción. Aparecieron nuevas etiquetas, <*embed*>, <*object*>y <*applet*>, las cuales permitían incluir elementos multimedia. Los navegadores reproducían un número limitado de formatos de archivo, lo que dificultaba la visualización óptima de algunos contenidos de la web. La alternativa fue el uso de flash, un formato que se ganó su popularidad rápidamente y que permitía mayor interacción con usuario. La realidad de los sitios desarrollados con esta tecnología era que

[15]http://www.w3schools.com/css/default.asp

sacrificaban la accesibilidad del contenido, dificultando el posicionamiento en los buscadores.

La resolución de pantalla fue aumentando con el tiempo. Las páginas se desarrollaban en un tamaño fijo, habitualmente 800x600 píxeles de resolución. El objetivo era que la página pudiera verse correctamente en todas las resoluciones sin activar el scroll en el eje x, es decir, sin exceder el ancho de las pantallas. Entre otras cosas, esto llevó al uso de medidas relativas que permitían que los elementos de la web se adaptaran, en cierta medida, al tamaño de la ventana. El formato flash calló en desuso, pues carecía de tamaño relativo y los elementos se adaptaban todos a la vez, deformando imágenes y reduciendo el texto al límite de lo ilegible.

Los desarrolladores web buscaban siempre alternativas para la simplificación del código. Aparecieron librerías JavaScript y frameworks que facilitaron el desarrollo de interacciones avanzadas. La librería más popular es JQuery y su filosofía es "escribe menos, haz más". Se produjo una gran cantidad de material programado que representaba elementos básicos de las páginas web, con un estilo fácil de editar para su uso en distintos proyectos. Todo esto llevó a un nivel mayor de complejidad los diseños y la interacción de usuario.

Una sola web

Con la aparición de dispositivos móviles con acceso a internet, se produjo un conflicto con el tamaño de las páginas web. Ya no sobraba con trabajar en medidas relativas. Los dispositivos móviles requerían un diseño diferente, una tipografía ajustada al tamaño de la pantalla, botones más grandes, etc. Como respuesta, se realizaron webs con versiones para escritorio y móvil.

Los desarrolladores se percataron del trabajo que suponía desarrollar varias webs para un mismo proyecto y buscaron alternativas. El consorcio W3C[16] introdujo ya la idea de "una web" en su recomendación del 28 de julio de 2008 bajo el subtítulo *One Web* [W3C16]:

"The recommendations in this document are intended to improve the experience of the Web on mobile devices. While the recommendations are not specifically addressed at the desktop browsing experience, it must be understood that they are made in the context of wishing to work towards "One Web". [...], One Web means making, as far as is reasonable, the same information and services available to users irrespective of the device they are using."[17]

[16] World Wide Web Consortium

[17] "Las recomendaciones de este documento están destinadas a mejorar la experiencia de la web en los dispositivos móviles. Aunque las recomendaciones no están dirigidas específicamente a la experiencia de navegación de escritorio, debe entenderse que se hacen en el contexto de querer trabajar por *Una Web.* [...], Una web significa hacer accesible, en la medida de lo posible, la misma información y los servicios disponibles para los usuarios, independientemente del dispositivo que utilicen."

3.2.2 Mercado actual y tendencias

En el diseño web, como en toda disciplina del diseño, existen estilos y tendencias influenciadas por una serie de factores sociales y tecnológicos. El factor tecnológico ha condicionado el diseño web desde el inicio de la *World Wide Web*, no solo en aspectos técnicos, también en su concepción, como es el caso del diseño orientado a dispositivos de menor tamaño, que se ha extendido a lo largo de la Web, aplicándose sus normas en todos los contextos.

La web se presenta con mayor claridad y simplicidad, dando prioridad al contenido y jugando con el espacio en blanco. El texto quiere llamar la atención del usuario, adquiere mayor tamaño y densidad con el uso habitual de la negrita. Se mantienen las tipografías sin serifa, aunque aumenta el uso de tipografías más elaboradas. Las imágenes y los vídeos pasan a ser de gran tamaño, ocupando el ancho de la pantalla y funcionando en ocasiones como background para el mensaje principal. La interfaz de usuario tiende a botones de gran tamaño, como mejora destinada a la experiencia de usuario en las pantallas de tamaño reducido. El menú principal se mantiene fijo, se queda visible o escondido en la parte superior o lateral de la pantalla para facilitar el acceso a las opciones de navegación.

El ***Flat Design* o diseño plano** se convierte en tendencia en la web, tanto para el diseño de contenidos como para la interfaz y sus elementos. Estos diseños se componen de colores planos y formas simplificadas, sin relieves ni sombreados difuminados. La influencia de la ilustración es evidente en esta tendencia. Se extiende el uso del *SVG*, un formato vectorial escrito en código que permite que las ilustraciones se adapten a grandes tamaños sin aumentar el peso del documento. Se utilizan paletas de colores habituales en la ilustración vectorial, con colores frescos y tonos pastel, con cierta tendencia a la calidez. Las ilustraciones animadas se vuelven cada vez más comunes.

Cada vez más diseñadores optan por una propuesta ***responsive*** ante una versión móvil y otra de escritorio. Las webs que continúan teniendo una versión móvil suelen desarrollar aplicaciones para estos dispositivos, ofreciendo su descarga en el sitio web. Los diseño flexibles han dado lugar a estructuras de una sola página, construyendo páginas web verticales, con efectos impresionantes y webs de galerías y contenidos con scroll infinito. Otras propuestas carecen de menú principal, construyendo un recorrido a través de la web o apareciendo las opciones de navegación según va navegando el usuario, animando a explorar los elementos de la web.

La experiencia de usuario se ha convertido en un campo de investigación, donde los diseñadores dan rienda suelta a su creatividad, experimentando con la interacción y buscando nuevas vías de comunicación [BS14].

3.2.3 Frameworks, Herramientas y Lenguajes de desarrollo

Es habitual en los últimos años encontrarse con el concepto de Framework[18] (cuya traducción aproximada sería "marco de trabajo"), cuyo concepto no es sencillo de definir, a pesar de que un programador con experiencia captará su sentido de manera casi intuitiva, y es muy posible que esté utilizando su propio framework (aunque no lo llame así).

Framework es por tanto, un esquema, un esqueleto, un patrón...etc, para el desarrollo y la implementación de una aplicación de diversa índole. Su utilización fomenta la reutilización de código, promueve buenas prácticas de desarrollo y proporciona una reducción de tiempo en los procesos de desarrollo.

Debido a la cantidad de frameworks que existen hoy en día, esta sección se centrará en los que rige la arquitectura MVC o Modelo Vista Controlador.

Figura 3.5: Frameworks Web

Un *Framework* MVC involucra la filosofía de trabajo de la separación de las funciones principales dentro de una estructura configurable, obteniendo los beneficios que un patrón de estas características otorga. Las ventajas que ofrece el diseñar aplicaciones web utilizando un framework MVC consiste principalmente en obtener un desarrollo estructurado con código reutilizable, aportando un desarrollo rápido debido a que se disminuye el esfuerzo de trabajo al aprovechar funcionalidades ya implementadas. En este aspecto, JavaScript, se considera una solución interesante para la creación de *frameworks* ya que permite incorporar a ellos un sin fin de funciones diferentes, entre las cuales predomina el ofrecer una interfaz única para todos los navegadores, corrigiendo fallos e incompatibilidad entre ellos, añadiendo funcionalidades de alto nivel especialmente en el DOM[19].

[18] define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. WIKIPEDIA

[19] es esencialmente una interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML,1 un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. WIKIPEDIA

Aunque PHP[20] también es hoy en día muy utilizado y se nombrarán varios frameworks, en general los frameworks JavaScript están más a la orden del día.

- **[FRAMEWORKS MVC]**

Angular.js[21]

Es un framework de código libre desarrollado por Google para crear aplicaciones web en lenguaje para el cliente con JavaScript, extendiendo el código HTML con etiquetas propias, tales como *ng-app, ng-controller, ng-model, ng-view*

Angular es reconocido como un framework sólido, que implementa el patrón MVC, separando en su totalidad el modelo, la vista y el controlador, con inyección de dependencia que nos permite crear pequeños módulos, está orientado a código testable, e iguala el desarrollo front-end con el back-end.[22]

Sus principales características son [HA15] :

- Permite extender HTML con etiquetas personalizadas, definir y vincular variables vista/controlador, consultas AJAX[23] con peticiones HTTP[24], manipulación de datos en JSON[25], formularios de validación, desacoplamiento del DOM de JavaScript, filtros, sistemas de pruebas etc.
- Es compatible con los navegadores de última generación (Chrome, Firefox, Safari, Opera, Webkits, IE 9+) y se puede hacer compatible para Internet Explorer 8 o anterior mediante varias modificaciones.
- Flexibilidad en conexiones REST[26].
- Disminución de código.
- Soporte de animaciones a través de los eventos *hide, show, enter, leave y move.*

Sin lugar a duda, Angular.js marca la transición entre páginas webs y aplicaciones web sin recarga de página, extendiendo las limitaciones de HTML. Por supuesto, existen otros frameworks basados en JavaScript como podrían ser Backbone.js y Ember.js que comparten la misma filosofía que Angular.js.

Meteor, *framework* utilizado en este proyecto, está basado en *angular.js*.

[20] PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico

[21] https://angularjs.org/

[22] Front-end y back-end son términos que se refieren a la separación de intereses entre una capa de presentación y una capa de acceso a datos, respectivamente. Pueden traducirse al español el primero como interfaz, final frontal o frontal y el segundo como motor, dorsal, etc aunque es común dejar estos términos en inglés. WIKIPEDIA

[23] Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas

[24] Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

[25] JSON, acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos.

[26] La Transferencia de Estado Representacional (Representational State Transfer) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

Spine.js[27]

Spine.js es considerado un *framework* ligero para la creación de aplicaciones web en un entorno real. Aunque está dentro de la clasificación de un *framework* MVC, no cumple con esta concepción en su totalidad, ya que al principio estructura la aplicación bajo el enfoque de este patrón de diseño pero luego se aparta un poco de ello. Está escrito en *CoffeeScript* pero también se puede utilizar JavaScript; por lo que existen varias formas de trabajar con Spine.js para la creación de aplicaciones web, estas son:

- Utilizando JavaScript: descargar los archivos JavaScript e incluirlos en el código HTML.
- Usando Node.js8[28], CoffeeScript[29] y Hem[30]: mientras que Spine.app genera la estructura de directorios para la aplicación "Hem" será utilizado para gestionar dichos directorios de forma dinámica, y mostrarlos al usuario.
- A través de la integración con Rails9[31]: mientras que Rails hace una gran tarea para el almacenamiento de datos y autenticación, Spine proporciona a la aplicación con una experiencia de usuario front-end.

Las principales características son:

- Posee una ligera implementación del controlador (basado en la API[32] de Backbone) [Spi16].
- La capa del modelo es lo bastante completa, además incluye ORM[33].
- Ejerce una comunicación asíncrona hacia el servidor.
- Incorpora adaptadores de almacenamiento local de AJAX y HTML5.
- Incorpora una extensión móvil, que le permite crear aplicaciones móviles fácilmente.
- Incorpora un gestor de dependencia y servidor de desarrollo denominado *Hem*.
- Funcionamiento en varios navegadores (Internet Explorer, Safari, Chrome, Firefox, otros).
- Posee una biblioteca que contienen las clases principales como modelo y controlador.

[27] http://spinejs.com/
[28] https://nodejs.org/en/
[29] http://coffeescript.org/
[30] https://github.com/spine/hem
[31] http://rubyonrails.org/
[32] La interfaz de programación de aplicaciones, abreviada como API (del inglés: Application Programming Interface), es el conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
[33] El mapeo objeto-relacional, ORM en inglés, es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

Zend Framework[34]

Zend Framework es un *framework* para desarrollo de aplicaciones web y servicios web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source[35] y trabaja con PHP 5.Está implementado usando código 100 % orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes.

ZF implementa el modelo MVC, es orientado a objetos, y tiene un estándar de codificación que seguir en el proyecto. Tiene un gran soporte para las aplicaciones con multi-idioma, convertir formatos de fechas, monedas, etc.

Algo muy importante es que es adaptable para distintos tipos de base de datos y además posee componentes para la autenticación y autorización de usuarios ya testeados, por lo que ahorra una gran parte de trabajo. Dicho framework posee gran documentación y manuales en la web, pero el idioma en el que se encuentra la mayoría es inglés [Bal16].

Symphony2[36]

Symfony2 es un framework de PHP que utiliza la arquitectura MVC. Ha sido construido con componentes independientes que han sido creados por el proyecto Symfony. Se usa bajo la licencia MIT[37], es decir es un software libre con el que se puede crear aplicaciones web de una forma rápida y profesional. Existe una gran comunidad de programadores en todo el mundo que participan activamente en el desarrollo de este framework, esto garantiza la continuidad de los proyectos [Egu15].

Grandes proyectos como Drupal[38] han sido construidos usando componentes de Symfony. Pero lo más importante de este framework es la documentación existente en la web en español, lo que ayuda al rápido aprendizaje y a resolver los problemas encontrados de una forma sencilla e eficaz.

- **[LENGUAJES Y HERRAMIENTAS]** [GM15]

HTML5[39]

Desde el inicio de internet se han publicado sitios web gracias al lenguaje HTML. Es un lenguaje estático para el desarrollo de sitios web (acrónimo en inglés de HyperText Markup Language, en español Lenguaje de Marcas Hipertextuales). Desarrollado por el *World Wide Web Consortium (*W3C*)*. Los archivos pueden tener las extensiones (htm, html).

Características

[34] https://framework.zend.com/
[35] Código abierto, de licencia libre
[36] http://symfony.es/
[37] Massachusetts Institute of Technology
[38] https://www.drupal.org/
[39] https://www.w3.org/TR/html5/

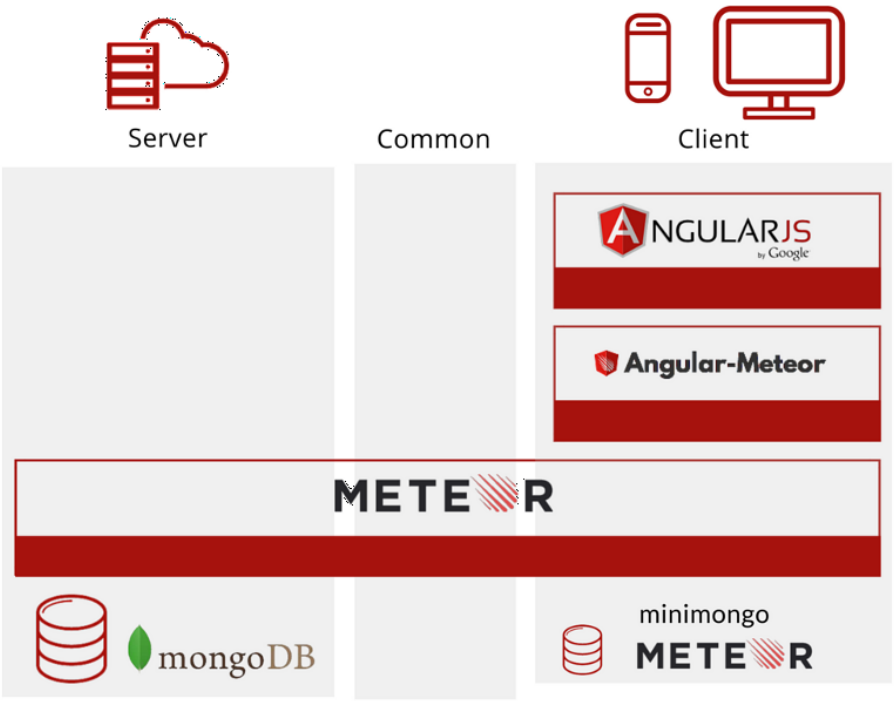

Figura 3.6: Comparación de Meteor con otros Frameworks mencionados [Im516]

- Texto presentado de forma estructurada y agradable.
- No necesita de grandes conocimientos cuando se cuenta con un editor de páginas web o WYSIWYG.
- Archivos pequeños.
- Despliegue rápido.
- Lenguaje de fácil aprendizaje.
- Lo admiten todos los exploradores.

JavaScript[40]

JavaScript es un lenguaje interpretado, es decir, no requiere compilación. Fue creado por Brendan Eich en la empresa *Netscape Communications* y es utilizado principalmente en páginas web. Es similar a Java[41], aunque no es un lenguaje orientado a objetos, ya que no dispone de herencias. La mayoría de los navegadores en sus últimas versiones interpretan código JavaScript.

El código JavaScript puede ser integrado dentro de nuestras páginas web. Para evitar incompatibilidades el *World Wide Web Consortium (*W3C*)* diseño un estándar denominado DOM (en inglés Document Object Model, en su traducción al español Modelo de Objetos del Documento).

[40] https://www.javascript.com/

[41] Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos https://www.java.com/es/

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

Tradicionalmente se venía utilizando en páginas web *HTML* para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Características:

- Lenguaje de *scripting* seguro y fiable. Los *scripts* son porciones de código ejecutable en el cliente.
- Los scripts tienen capacidades limitadas, por razones de seguridad.
- El código Javascript puede ejecutarse en el servidor o en el cliente de forma indistinta.

PHP[42]

Es un lenguaje de programación utilizado para la creación de sitio web. PHP es un acrónimo recursivo que significa *PHP Hypertext Pre-processor*, (inicialmente se llamó Personal Home Page). Surgió en 1995, desarrollado por PHP Group.

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas *HTML* y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión (php).

Características:

- Muy fácil de aprender.
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta en cierta medida la orientación a objeto. Clases y herencia.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- Capacidad de conexión con la mayoría de los manejadores de base de datos.
- Capacidad de expandir su potencial utilizando módulos.

[42] http://php.net/

- Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

ASP.NET[43]

Este es un lenguaje comercializado por *Microsoft*, y usado por programadores para desarrollar entre otras funciones, sitios web. ASP.NET es el sucesor de la tecnología ASP, fue lanzada al mercado mediante una estrategia de mercado denominada .NET.

El ASP.NET fue desarrollado para resolver las limitaciones que brindaba tu antecesor ASP. Creado para desarrollar web sencillas o grandes aplicaciones. Para el desarrollo de ASP.NET se puede utilizar C#[44], VB.NET[45] o J#[46]. Los archivos cuentan con la extensión (aspx). Para su funcionamiento de las páginas se necesita tener instalado IIS[47] con el Framework .Net. Microsft Windows 2003 incluye este framework, solo se necesitará instalarlo en versiones anteriores.

Características:

- Completamente orientado a objetos.
- Controles de usuario y personalizados.
- División entre la capa de aplicación o diseño y el código.
- Facilita el mantenimiento de grandes aplicaciones.
- Se puede utilizar C#, VB.NET o J# como lenguajes.
- Comunicación óptima con SQL Server.
- Soporta el lenguaje JScript (Javascript de *Microsoft*).

JSP[48]

Es un lenguaje para la creación de sitios web dinámicos, acrónimo de Java Server Pages. Está orientado a desarrollar páginas web en Java. JSP es un lenguaje multiplataforma. Creado para ejecutarse del lado del servidor.

[43] http://www.asp.net/

[44] C# (pronunciado si sharp en inglés) es un lenguaje de programación orientado a objetos desarrollado y estandarizado por *Microsoft* como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA

[45] Visual Basic .NET (VB.NET) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET.

[46] El lenguaje de programación J# (o J-sharp) es un lenguaje transicional para programadores del lenguaje de programación Java y del lenguaje J++ de *Microsoft*, creado con la intención de que ambos puedan usar sus conocimientos actuales para crear aplicaciones en la plataforma .NET de *Microsoft*.

[47] Internet Information Services o IIS1 es un servidor web y un conjunto de servicios para el sistema operativo *Microsoft* Windows.

[48] http://www.jsp.es/index.php/es/

JSP fue desarrollado por Sun Microsystems. Comparte ventajas similares a las de ASP.NET, desarrollado para la creación de aplicaciones web potentes. Posee un motor de páginas basado en los servlets de Java. Para su funcionamiento se necesita tener instalado un servidor Tomcat[49].

Características:

- Código separado de la lógica del programa.
- Las páginas son compiladas en la primera petición.
- Permite separar la parte dinámica de la estática en las páginas web.
- Los archivos se encuentran con la extensión (jsp).
- El código JSP puede ser incrustado en código HTML.

Ruby[50]

Es un lenguaje interpretado de muy alto nivel y orientado a objetos. Desarrollado en el 1993 por el programador japonés Yukihiro "Matz" Matsumoto. Su sintaxis está inspirada en Phyton, Perl. Es distribuido bajo licencia de software libre (Opensource).

Ruby es un lenguaje dinámico para una programación orientada a objetos rápida y sencilla. Para los que deseen iniciarse en este lenguaje pueden encontrar un tutorial interactivo de ruby. Se encuentra también a disposición de estos usuarios un sitio con informaciones y cursos en español.

Características:

- Existe diferencia entre mayúsculas y minúsculas.
- Múltiples expresiones por líneas, separadas por punto y coma ";".
- Dispone de manejo de excepciones.
- Ruby puede cargar librerías de extensiones dinámicamente si el (Sistema Operativo) lo permite.
- Portáil.

3.3 Cloud Computing Stack

El *Cloud Computing Stack* o Modelo de Computación en la Nube, es un modelo que habilita el acceso a un conjunto de recursos o servicios computacionales configurables bajo demanda y con ubicuidad. Este conjunto de recursos puede ser rápidamente suministrado y lanzado con un esfuerzo de mantenimiento mínimo o interacción por parte del proveedor.

[49] Apache Tomcat (también llamado Jakarta Tomcat o simplemente Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation

[50] https://www.ruby-lang.org/es/

Las principales características de la computación en la nube son:

- **Servicio propio bajo demanda:** Un consumidor puede contratar los servicios de manera unilateral y los recursos computacionales que requiera su sistema, como tiempo de servidor o capacidad almacenamiento, según sea necesario de forma automática sin requerir interacción humana con el proveedor del servicio.
- **Acceso desde la red:** Las capacidades del sistema están disponibles a través de la red o internet, accediendo mediante unas plataformas que pueden ser usadas por distintos tipos de dispositivos (smartphones, tablets, ordenadores,...)
- **Puesta en conjunto de recursos:** Los recursos informáticos suministran servicios a múltiples consumidores usando un modelo multi-usuario, con un servidor físico o virtual que aloja unaaplicación que está diseñada para permitir el uso de múltiples usuarios diferentes. Cada usuario se siente como si tuviera un uso exclusivo de la aplicación. Los recursos son dinámicamente asignados y reasignados de acuerdo con la demanda del consumidor
- **Flexibilidad:** Gracias a la computación en la nube tenemos un nuevo abanico de servicios a la vez que se ofrece la posibilidad de convertir gastos fijos en variables, lo que permite conocer mejor los costes reales y minimizar los riesgos.
- **Escalabilidad:** Sin la nube las empresas que querían aumentar su oferta tenían que invertir primero una gran cantidad de capital en infraestructura y otros recursos informáticos. Por ejemplo, una empresa que tenga picos en sus actividades debe de invertir mucho capital en recursos que mantendrá inactivos durante la mayor parte del tiempo o reducir su actividad en determinadas épocas. Gracias a la computación en la nube, la empresa podrá adaptar los servicios y recursos contratados a sus necesidades.
- **Monitorización:** Los sistemas de computación en la nube controlan y optimizan sus recursos de manera automática gracias a los sistemas de medida, que chequean el rendimiento del sistema en el nivel de abstracción apropiado para un determinado tipo de servicio (almacenamiento, procesamiento de datos, ancho de banda, usuarios activos,...). Los recursos usados pueden ser monitorizados y controlados, además permiten enviar información propia. Todo esto proporciona información de gran utilidad tanto para el proveedor como para el consumidor de esos servicios.

El servicio ofrecido en un entorno de computación en la nube puede categorizarse en función del nivel de control ofrecido sobre la infraestructura subyacente. Así, se dispone de tres modelos de servicio fundamentales: IaaS, PaaS, SaaS [GC15].

3.3.1 Software as a Service *(SaaS)*

El modelo de servicio más completo es aquel que ofrece el software y el hardware como un servicio conjunto, es decir, *SaaS* provee la infraestructura, el software, la solución y toda la

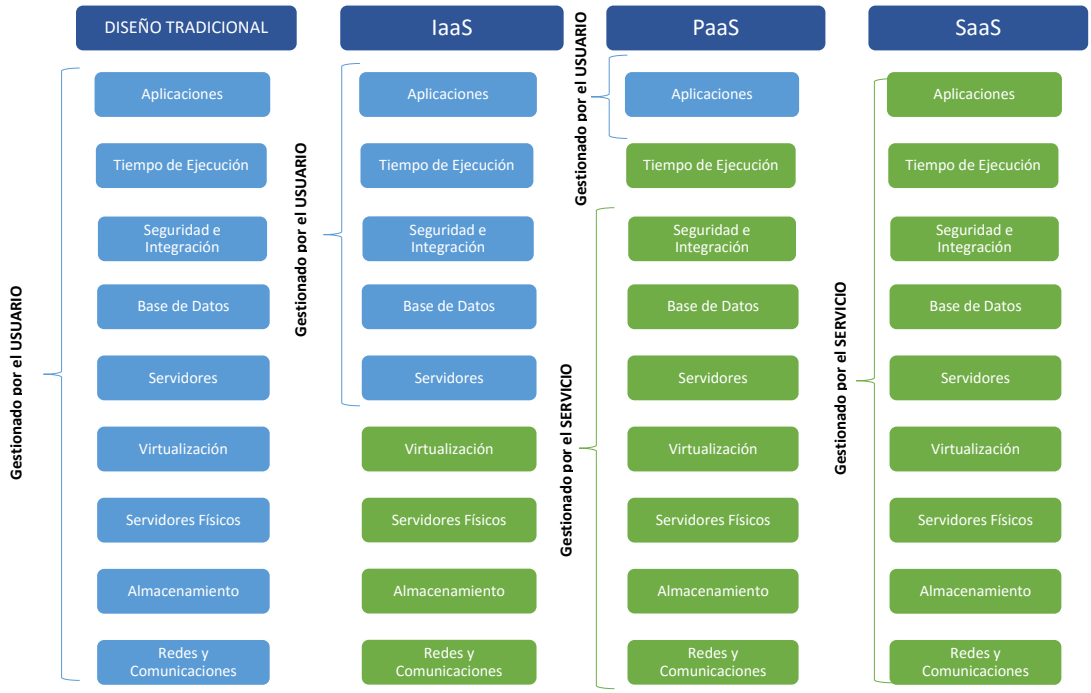

Figura 3.7: Comparativa de los modelos de computación en la nube

pila de aprovisionamiento como un servicio global. Como puede observarse en la figura 3.7 es el modelo opuesto al diseño tradicional, puesto que ofrece todo el servicio al completo.

Software as a Service (SaaS) se puede describir como software que está desplegado en un servicio de hosting y puede ser accedido globalmente a través de Internet mediante navegador, móvil, tablet, etc. Y donde todos los aspectos que no sean la propia interacción con la aplicación son transparentes al usuario. En el modelo SaaS, los usuarios pagan por el uso del servicio mediante cuotas de suscripción, válidas por un determinado período de tiempo, como en el caso de un alquiler. Las características fundamentales de este modelo se pueden resumir en:

- El software está disponible globalmente y en todo momento a través de Internet y bajo demanda.
- El modelo de subscripción suele ser mediante licencias o basado en uso y es facturado por mensualidades de forma periódica.
- Todo lo relativo a operaciones es responsabilidad del proveedor.
- Las actualizaciones, mejoras, evoluciones o parches en el aplicativo, debe ser siempre transparente al usuario y por supuesto no debe hacer ningún tipo de configuración.
- SaaS soporta múltiples usuarios generalmente con un modelo multi-usuario.

Este modelo de servicios normalmente pretende llegar a pequeñas y medianas empresas (PYMES) y a veces a usuarios individuales. El acceso suele ser a través de un portal o de un CMS web, que puede ser en abierto o bien está sujeto a una subscripción previa de otro servicio en la compañía que ofrece SaaS (ADSL, línea móvil, .etc.). Dicho portal puede ser muy variado, pero generalmente contiene los servicios y aplicaciones adquiridos previamente con un acceso mediante 'mashups'. Y por otro lado un catálogo de aplicaciones que se ofrecen. De ahí en adelante el portal puede ser tan avanzado como se quiera pero siempre será el punto de acceso y uso a los servicios.

3.3.2 Platform as a Service *(PaaS)*

Este modelo de servicio se sitúa por encima de IaaS y por debajo del SaaS en cuanto a nivel de abstracción de los recursos de tecnología de la información. Este modelo propone un entorno software en el cuál un desarrollador puede crear y modificar soluciones dentro de un contexto de herramientas de desarrollo que la plataforma proporciona.

En *Platform as a Service (PaaS)* los clientes pueden interactuar con el software para introducir o recuperar datos, realizar acciones, etc.; pero no tienen responsabilidad de mantener el hardware, software o el desarrollo de las aplicaciones, sólo se tiene responsabilidad de la interacción con la plataforma, es decir, el proveedor es el responsable de todos los aspectos operacionales. Además, la plataforma ofrece herramientas de desarrollo y despliegue de aplicaciones.

Las plataformas como servicio vienen a suponer que el desarrollador de aplicaciones web se olvida de almacenaje de ficheros, de gestión de la base de datos, de balanceo entre máquinas, de ancho de banda, de escalabilidad, de picos de demanda, de estabilidad, de configurar una máquina servidor...; en definitiva, únicamente se presta atención al desarrollo de la aplicación web, olvidándose de la infraestructura.

Poder abstraerse del entorno supone dos ventajas importantes. La primera sería el ahorro de costes y la segunda la posibilidad de centrar toda la atención en la aplicación. El ahorro de coste no proviene únicamente de la posibilidad de contratar un servicio de almacenamiento más barato, sino porque el conocimiento necesario para crear arquitecturas escalables es muy costoso. Aunque también cuenta con algunos inconvenientes: depender de un único proveedor y sufrir sus caídas.

3.3.3 Infraestructure as a Service *(IaaS)*

Podría definirse como un modelo de servicios de computación. Estos servicios podrían utilizarse para resolver necesidades computacionales sin problemas de escalabilidad. El presente trabajo centra su atención en este modelo de servicio.

El modelo *IaaS* hace referencia al hecho de ofrecer servicios de computación y almacenamiento, de tal manera que sea posible disponer de recursos como ciclos de CPU, memoria,

disco o equipamientos de red. El consumidor alquila los recursos de hardware en vez de comprarlos e instalarlos en su propio centro de procesamiento de datos, lo que le permite ir variando el consumo de los recursos en funciones de sus necesidades, esto se conoce como elasticidad de la infraestructura.

Este modelo está siendo adoptado por una multitud de *startups*[51] que han comenzado a emprender en tiempos de crisis y que no se pueden permitir tener su propio centro de datos o una infraestructura propia. En este modelo los desarrolladores encuentran una forma dinámica y flexible de trabajar, ya que se puede interactuar con la IaaS mediante servidores virtuales, almacenamiento virtual.

Normalmente se generan instancias de estas máquinas virtuales desde un portal web. En ese momento los desarrolladores tienen vía libre para personalizar y encontrar una solución apropiada. El acceso y la interacción de la aplicación con la IaaS suelerealizarse a través de SOA (Service-Oriented Architecture).

IaaS está enfocado a cualquier empresa que desea delegar la implantación de sus sistemas software y aplicaciones en la infraestructura hardware de un proveedor externo (hosting) o que requiera de servicios de almacenamiento externo, copias de seguridad de sus datos, cálculos complejos que requieran software de elevadas prestaciones, etc., el proveedor les permitirá gestionar dichos sistemas en un entorno virtualizado.

Los proveedores de servicios son los propietarios de las máquinas físicas, y las ofrecerán como un servicio a los usuarios, además de proporcionarles un entorno que les permita gestionarlas (sitio web) [Sos10].

3.4 Meteor

Meteor o *MeteorJS* es un framework para aplicaciones web con JavaScript libre y de código abierto y basado en *Node.js*. *Meteor* facilita la creación rápida de prototipos y produce código multiplataforma (web, Android, iOS). Se integra con MongoDB[52] como base de datos y usa Distributed Data Protocol como protocolo de comunicación, y un patrón publicación-subscripción para propagar automáticamente al cliente cambios en los datos sin requerir que el desarrollador escriba algún código de sincronización. En el cliente, Meteor depende de jQuery y puede ser usado con cualquier librería de Interfaz de usuario para JavaScript. Meteor es desarrollado por el Meteor Development Group [Ber16].

Es importante recalcar que *Meteor* es un proyecto open source o de código abierto. Es posible participar en su desarrollo y resulta muy interesante ver su código para entender ciertas partes. Una de las diferencias claves a la hora de seleccionar *Meteor* como la plataforma en la que construir una aplicación es su arquitectura cliente-servidor.

[51] Una compañía de arranque, compañía incipiente o, simplemente, compañía emergente

[52] es un sistema de base de datos *NoSQL* orientado a documentos, desarrollado bajo el concepto de código abierto. https://www.mongodb.com/es

Cliente-Servidor

Esta arquitectura lleva usándose para el desarrollo de aplicaciones web desde el principio y sigue siendo muy utilizada hoy en día. En partes completamente diferenciadas de la aplicación se tiene el código que se ejecuta en el cliente y en el servidor. Para garantizar la seguridad de la aplicación solo el código en el servidor tiene acceso a la base de datos. Por último, en este tipo de arquitectura de una aplicación web el renderizado suele hacerse en el servidor, aunque las aplicaciones web modernas han llevado la parte de renderizado al cliente. También se ha ido progresando hacía aplicaciones RIA (Rich Internet Application) o SPA (Single Page Application) manteniendo siempre la separación entre cliente y servidor.

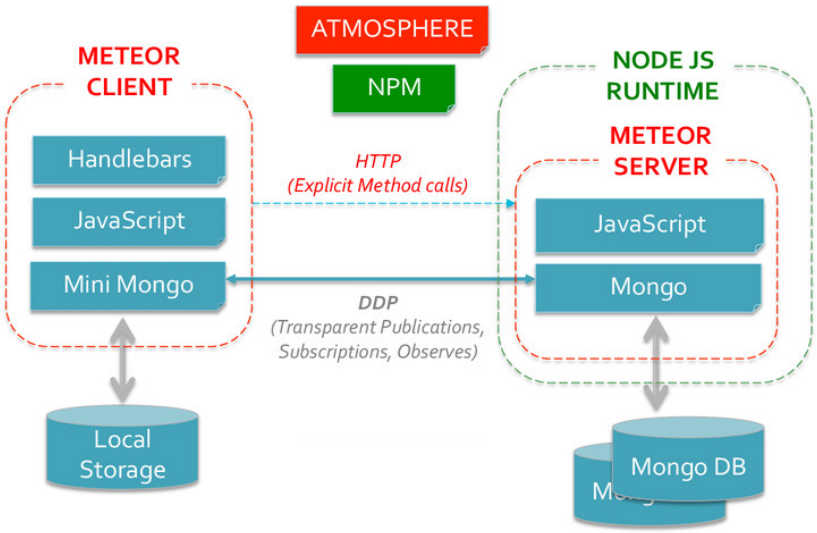

Figura 3.8: Arquitectura del Framework Meteor [Wel16]

Cliente-Servidor con Node.js

Creado en 2009 *Node.js* ha supuesto muchos cambios en el desarrollo web. Grandes empresas han movido sus servicios a Node.js y pese a no ser la mejor opción para todos los casos, aporta suficientes ventajas como para ser tenido en cuenta en muchos proyectos actualmente. Desarrollar una aplicación cliente/servidor con Node.js supone que ahora la parte del servidor también va a estar programada en javascript. Es la gran diferencia con respecto a la arquitectura anterior, en la que se utilizaban lenguajes distintos. Esta oportunidad de utilizar el mismo lenguaje permite reducir la barrera entre ambas partes.

Llegados a este punto ya se han visto algunos de los problemas que suponen las arquitecturas clásicas para desarrollo web: varios frameworks, grandes diferencias entre cliente y servidor, resulta imposible compartir código entre ambas partes... Debido a todos estos problemas Meteor se planteó desde el inicio como una plataforma isomórfica, es decir, que debía funcionar en cliente y servidor. Esta idea más tarde se amplió para incluir dispositivos móviles. Una vez tenemos una API compartida entre cliente y servidor, todo bajo un mismo

lenguaje, hay algo que falta, una parte que sea única del servidor: el acceso a la base de datos.

En una aplicación web moderna no desarrollada bajo Meteor lo normal es que la aplicación que se ejecuta en el servidor exponga al cliente una API a la que "atacar" para obtener información. De esta forma el cliente "pide" al servidor el listado de usuarios y es el servidor el que conecta a la base de datos, extrae los mismos y los envía codificados normalmente como json al cliente. Este sistema tiene un gran problema: el tiempo de carga. Es imposible evitar que la información consuma tiempo en viajar del cliente al servidor y realice el recorrido inverso, lo que provoca que la experiencia del usuario sea inferior.

Por esta razón *Meteor* lleva la base de datos también al cliente y la mantiene sincronizada con el servidor, todo utilizando un protocolo extremadamente sencillo llamado DDP. Este protocolo funciona sobre *websockets*[53]. La base de datos se deja solo al alcance del servidor por una gran razón: la seguridad.

Las principales características de Meteor son [Str12]:

Solo transporta datos

Las aplicaciones desarrolladas con *Meteor* llevan el *renderizado* al cliente. El protocolo que une la capa cliente con la capa servidor únicamente transporta datos. Nunca transporta html, css o assets de otro tipo.

JavaScript como lenguaje único

Como Meteor usa Node.js, se utiliza JavaScript en el cliente y en el servidor. Y más aún, Meteor es capaz de compartir código entre ambos entornos. Utilizando la técnica de *long pooling* cada vez que un cliente contacta a un servidor Meteor, deja abierta una petición extra con el cliente reteniendo indefinidamente la respuesta hasta tener algo que enviarle. De esta forma mantiene la reactividad del sistema.

Base de Datos en todas partes

Meteor brinda dos bases de datos: una base de datos del lado del cliente (cache) y una base de datos *MongoDB* en el servidor. Cuando un usuario modifica algún dato —haciendo clic en Save, por ejemplo,— el código JavaScript que se ejecuta en el navegador actualiza la base de datos en *MongoDB* local y luego realiza una solicitud de DDP al servidor. El código actúa de inmediato como si la operación hubiera sido exitosa ya que no necesita esperar la respuesta por el servidor. Mientras tanto, el servidor se actualiza en segundo plano. Si se produce un error en la operación del servidor o si devuelve un resultado inesperado, el código JavaScript del lado del cliente se reajusta inmediatamente según los datos devueltos recientemente del servidor. Este ajuste se

[53]WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor.

denomina *compensación de latencia* y genera una cantidad de velocidad adicional que el usuario sí percibe.

Este proceso de actualización y comunicación con la base de datos, produce un salto en el rendimiento hasta ahora conocido por los usuarios web. Dicho de otro modo, cuando el usuario actualiza un dato, lo está actualizando localmente y en segundo plano con el servidor, hecho que con tecnologías convecionales ocurre de forma secuencial.

Reactividad

Hoy en día la reactividad es algo imprescindible en muchos aspectos del desarrollo. No es un concepto propio de un lenguaje, pero *Meteor* lo lleva implícito. Una función *reactiva* es una función que vuelve a calcular su resultado cuando una de sus fuentes cambia. Muchas partes de Meteor utilizan reactividad de forma transparente y también es posible utilizarla bajo el control del programador.

Adopción del Ecosistema

El ecosistema web es enorme y se hace imprescindible aprovecharlo. Por ello es posible utilizar *Angular, React, GraphQL, Blaze, paquetes npm, etc.* Cada día Meteor es más flexible y sus partes más independientes.

Simplicidad = Productividad

La *API* y todo el ecosistema que rodea a Meteor intentan ser lo más sencillos posibles. Por ello Meteor integra la sincronización en tiempo real de base y sin ninguna complicación, por ello crear una aplicación de *Meteor* es tan sencillo como ejecutar un comando en el terminal...

Capítulo 4

Metodología de Trabajo

En este capítulo se determinará la metodología de trabajo que se ha seguido para el desarrollo del presente Trabajo Fin de Grado, así como la descripción del ciclo de vida y fases que se han empleado. Además se especifican las tecnologías, frameworks, herramientas y lenguajes que se han utilizado en la implementación de la aplicación web.

4.1 Metodología de Trabajo

Debido a que el proyecto es de desarrollo individual, la metodología no tiene un interés tan abordante a nivel organizativo como podría tenerlo con un grupo de desarrolladores. Por este motivo se detallará la forma de trabajo que se ha seguido y el estándar más adecuado al que se ha adaptado este proyecto.

4.1.1 Metodología de Desarrollo

La metodología de trabajo que ha sido utilizada en este proyecto ha sido la del *desarrollo iterativo e incremental*. Se ha elegido dicha metodología ya que, el proyecto está planificado en varios bloques temporales que definiremos iteraciones. En cada una de estas iteraciones se repite un proceso de trabajo que proporciona un resultado completo y concreto al final. De este modo cada bloque se debe completar en una sola iteración completando todas las tareas que competen a dicho módulo o bloque. Después de cada iteración se obtiene un módulo que va evolucionando en el tiempo y que será necesario interconectar a otros módulos, añadiendo nuevas funcionalidades y mejorando los completados anteriormente. Por tanto la priorización de requisitos se establece según la cantidad de valor que añadan al producto final [Her14].

En principio se identifican unos requisitos fundamentales y una serie de módulos básicos, que tras cada una de las iteraciones que se establecen fueron dando lugar a los módulos de gestión que debería de tener la aplicación, aportando funcionalidad y comunicación entre ellos. Despues de cada una de las iteraciones se obtenía un prototipo de la aplicación. Definimos prototipo como *el desarrollo de un sistema con la funcionalidad necesaria para abordar diferentes requisitos*. Dicho prototipo iba creciendo con el paso de las iteraciones dando lugar a un prototipo completo de la aplicación, el cual se valorará como un todo. En

cada una de las iteraciones se debe seguir el mismo ciclo de desarrollo, el cual tiene las siguientes fases [FdC02] :

1. **Análisis.** En esta fase, el desarrollador y el cliente realizan una puesta en común de los requisitos que el módulo a tratar debe recoger. Además, se valora el módulo junto a otros módulos para que su integración sea la adecuada. También se realizan, si es necesario, modificaciones en prototipos o módulos anteriormente implementados para mejorar su funcionamiento o diseño.
2. **Diseño.** En función de los requisitos definidos en la etapa de análisis se procede al diseño de bocetos y esquemas que describan la estructura del módulo.
3. **Implementación.** El módulo diseñado en la anterior etapa se implementa en código y se da forma al diseño del mismo, generando un primer prototipo del módulo.
4. **Pruebas de Integración.** Una vez completada la implementación del módulo, se procede a realizar todo tipo de pruebas de verificación e integración con el sistema hasta ahora implementado. Una vez las pruebas han sido realizadas de manera satisfactoria se genera un prototipo de sistema, el cual será el que se muestra al cliente.
5. **Valoración del Prototipo.** El prototipo creado en las etapas anteriores es mostrado al cliente, en nuestro caso, el director del proyecto. El cliente puede decidir que nuevas funcionalidades desea incluir en el prototipo o modificar el comportamiento de los módulos que hasta el momento se han añadido, así como el diseño del mismo. Tanto con el visto bueno del prototipado como con el rechazo del mismo, se debería volver al principio de este ciclo, ya sea con la iteración del mismo módulo a raíz de su modificación, o con un nuevo módulo.

Los principales motivos por los que se ha optado por esta metodología de trabajo son:

- **Planificación** el cliente (o su representante) escribirá sus necesidades para definir concretamente las actividades que el sistema debe realizar. En esta fase se creará un documento que contendrá historias de usuario que forman el plan de liberación, el cual define los tiempos de entrega de la aplicación para poder recibir *feedback*[1] por parte del cliente.
- **El cliente deberá formar parte del equipo de desarrollo.** Se le dará poder para determinar los requisitos de la aplicación, definir la funcionalidad y dar prioridad a determinadas cosas. Gracias a esto, habrá una fuerte interacción con el programador, disminuyendo así el tiempo de comunicación y la cantidad de documentación a redactar.
- **Integración continua:** consiste en implementar progresivamente las nuevas características del software. En lugar de crear versiones estables en función de una planifi-

[1] Retroalimentación

cación previamente realizada, el programador diseñará los módulos de manera incremental e integrándolos con los módulos existentes.

- **Desarrollo incremental:** Al tratarse de un desarrollo incremental permite que el desarrollador adquiera experiencia de iteraciones y modulos anteriores.
- **Prototipado:** Mediante la generación de prototipos de manera regular se produce una sensación de avance la cual no se da en otras metodologías de trabajo. Con esto el cliente puede obetner una visión de los resultados desde las primeras iteraciones.

En resumen, el desarrollo iterativo e incremental es una de las mejores metodologías a seguir para el estudio y desarrollo de una aplicación web, debido a la fácil adaptación de los nuevos requisitos o funcionalidades y al hecho de crear un prototipo de web que va creciendo según se crean los módulos que definirán su funcionalidad.

4.1.2 Aplicación de la Metodología

La metodología de desarrollo iterativo e incremental está creada para cualquier entorno y equipo de desarrollo, indiferentemente del número de personas que formen ese equipo. En el caso particular de este Trabajo Fin de Grado, al tratarse de un único desarrollador no existen restricciones a la hora de llevar a cabo el proceso de desarrollo haciendo uso de esta metodología.

El desarrollador de la aplicación es Miguel González Cano. David Vallejo Fernández es el encargado de dirigir el proyecto y el usuario final de la aplicación web, por tanto también hace el rol de cliente. Los prototipos obtenidos han sido creados de manera ágil, ya que, durante el proceso de desarrollo del proyecto se mantenían reuniones periódicas cada 1 ó 2 semanas, las cuales ayudaban a decidir que nuevos requisitos incluir, modificar, o cambiar, valorando el prototipo y siguiendo el modelo iterativo de fases que se ha comentado anteriormente.

4.2 Herramientas

4.2.1 Hardware

El proyecto a realizar no tiene requisitos hardware específicos más allá de los que puede tener cualquier ordenador o dispositivo moderno con conexión a internet. El único requisito para poder acceder a la aplicación web, es obviamente, poseer una conexión a internet. Durante el desarrollo se han utilizado 2 equipos, un ordenador portátil y uno de sobremesa con las siguientes características:

Equipo portátil:

- Procesador Intel Core 2 Duo P8400, 2,26 Ghz
- 4 GiB de memoria RAM
- Disco duro Toshiba 320 GiB.

- Tarjeta Grafica Radeon 256Mb

Equipo sobremesa:

- Procesador AMD FX8350 4.2 Ghz - 8 núcleos
- 8 GiB de memoria RAM
- Disco duro WD 1024 GiB
- Tarjeta Grafica AMD Radeon 7950 3GiB

4.2.2 Software

Sistema Operativo

El sistema operativo en el que se ha desarrollado la mayor parte del proyecto ha sido Debian GNU/Linux. Aunque el proyecto fue iniciado en un sistema operativo Windows 7, más tarde se migró todo el proyecto a una distribución Linux ya que ofrecía mayor simplicidad a la hora del desarrollo y una mayor rapidez a la hora de cargar el framework para programar.

Se ha utilizado una versión estable y final de Debian 7 Wheezy.[2]

Documentación

Tanto para los manuales como para la escritura del presente documento se ha utilizado LaTeX[3]. LaTeXes un lenguaje de marcas y sistema de preparación de documentos, que permite expresar su contenido y forma por medio de texto plano siguiendo una sintaxis que, posteriormente, se compila para generar el documento en un determinado formato. En este caso el formato de salida será PDF.

Gráficos

Para diseñar los gráficos y figuras del proyecto se han utilizado varios programas de diseño de figuras como son *Gimp*[4] y *Microsoft Word*. Para los diagramas se ha utilizado la herramienta online de diseño *draw.io*[5].

Editores de Texto

Se han utilizado 2 editores de texto a la hora de la escritura de código tanto en lenguajes HTML,Javascript y MongoDB, de los cuales hablaremos posteriormente.

Atom[6]

Atom, es el fruto de 6 años de trabajo por parte de los desarrolladores de Github; un editor de textos orientado al desarrollo de código y que, como no podía ser de otra forma, se ha desarrollado como una herramienta en software libre. Atom se ha construido

[2]https://www.debian.org/releases/wheezy/
[3]https://www.latex-project.org/
[4]http://www.gimp.org.es/
[5]https://www.draw.io/
[6]https://atom.io/

usando los mismos componentes que se usarían para desarrollar una web: HTML y CSS. Además, como buena herramienta de trabajo, Atom se ha concebido como un editor flexible al que se le pueden añadir nuevas funcionalidades y complementos, además de temas, que nos permitan configurar un espacio de trabajo adaptado a nuestras necesidades. La mayor parte del código HTML y Javascript se ha realizado con esta herramienta.

Sublime Text 3[7]

Sublime Text es un editor de texto y editor de código fuente está escrito en C++ y Python para los plugins. Desarrollado originalmente como una extensión de Vim, con el tiempo fue creando una identidad propia, por esto aún conserva un modo de edición tipo vi llamado Vintage mode.

Se puede descargar y evaluar de forma gratuita. Sin embargo no es software libre o de código abierto y se debe obtener una licencia para su uso continuado, aunque la versión de evaluación es plenamente funcional y no tiene fecha de caducidad. Toda la documentación se ha realizado con la ayuda de esta herramienta.

Frameworks

Meteor[8]

Como se ha comentado anteriormente en su correspondiente sección (véase Apartado 4.4 Meteor), se ha utilizado este framework para la realización de la aplicación web. El framework incluye el motor y todo lo necesario para compilar y ejecutar el código en un cliente web. El framework incluye la creación de proyectos de prueba y esquemas de código para facilitar la iniciación del proyecto. En la sección de Arquitectura se pasará a detallar con mayor grado el funcionamiento del framework, así como su metodología de desarrollo [DeB16].

Bootstrap[9]

Bootstrap es un framework CSS desarrollado inicialmente (en el año 2011) por Twitter que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web. Aunque el desarrollo del framework Bootstrap fue iniciado por Twitter, fue liberado bajo licencia MIT en el año 2011 y su desarrollo continua en un repositorio de GitHub.

Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño.

[7] https://www.sublimetext.com/3
[8] https://www.meteor.com/
[9] http://getbootstrap.com/

Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías.

La mayor parte del código del cliente se ha desarrollado bajo bootstrap, lo que facilita mucho trabajo de desarrollo en CSS.

MongoDB[10]

MongoDB (que proviene de «humongous») es la base de datos NoSQL líder orientada a documentos y desarrollado bajo el concepto de código abierto. MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En lugar de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Es una base de datos ágil que permite a los esquemas cambiar rápidamente cuando las aplicaciones evolucionan, proporcionando siempre la funcionalidad que los desarrolladores esperan de las bases de datos tradicionales, tales como índices secundarios, un lenguaje completo de búsquedas y consistencia estricta. MongoDB ha sido creado para brindar escalabilidad, rendimiento y gran disponibilidad, escalando de una implantación de servidor único a grandes arquitecturas complejas de centros multidatos.

MongoDB es la base de datos que integra Meteor, por lo que al venir ya integrada no es necesario desplegarla por separado, siempre está enlazada a nuestro proyecto desde su creación. La versión utilizada es la más reciente hasta la fecha, la versión MongoDB 3.2.8.

Navegadores Web

Para la verificación y pruebas del entorno, y obviamente, para la ejecución del código en el cliente, son necesarios los navegadores web. El proyecto se ha probado en los siguientes navegadores web:

- Microsoft Edge (Windows 10)
- Google Chrome v52
- Mozilla Firefox v48

Gestor de Tareas

Para gestionar las tareas dentro del proyecto se ha hecho uso de **Trello**[11].

Trello es un gestor de tareas que permite el trabajo de forma colaborativa mediante tableros (board) compuestos de columnas (llamadas listas) que representan distintos estados. Se basa en el método Kanban para gestión de proyectos, con tarjetas que viajan por diferentes listas en función de su estado.

[10] https://www.mongodb.com/
[11] https://trello.com/

Control de versiones

El control de versiones consiste en utilizar herramientas o aplicaciones software que faciliten la gestión de los cambios en el código durante el desarrollo de proyectos software, aunque también sirve para gestionar la documentación. Esto se realiza mediante el uso de un repositorio que almacena los cambios realizados y permite descargarlos y clonarlos, de forma que distintos desarrolladores puedan trabajar en el proyecto desde distintos equipos y mantener sincronizado el código, además de añadir seguridad en todos los sentidos.

Para el control de versiones se ha utilizado Git, a través del servicio de alojamiento en web de **BitBucket**[12], escrito en Python y que ofrece planes gratuitos y comerciales.

Despliegue Web

Para el despliegue de la aplicación en internet y que esté disponible en un servidor se ha hecho uso de una plataforma que admite el motor Meteor, llamada **Modulus**.[13]

Modulus es un servicio web que sirve como contenedor de aplicaciones, y que permite desplegar y aportar escalabilidad a las aplicaciones web. Característicamente forma parte de las Plataformas como Servicio (PaaS).

4.2.3 Lenguajes

HTML5

HTML es el lenguaje que se emplea para el desarrollo de páginas de internet, más concretamente en el cliente. Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. Es el lenguaje principal que se ha utilizado en este proyecto para el cliente, o más comúnmente dicho, para la parte que el usuario ve. Más concretamente se ha hecho uso de la versión más reciente de este lenguaje, la versión 5 o HTML5. Ésta se trata de la última versión adaptada a los nuevos navegadores y que posee mayor variedad de etiquetas y una gran cantidad de funcionalidades añadidas. Algunos de los cambios respecto a acsHTML son:

- Semántica: Permite describir con mayor precisión cual es su contenido.
- Conectividad: Permite comunicarse con el servidor de formas nuevas e innovadoras.
- Sin conexión y almacenamiento: Permite a las páginas web almacenar datos localmente en el lado del cliente y operar sin conexión de manera más eficiente.
- Multimedia: Nos otorga un excelente soporte para utilizar contenido multimedia como lo son audio y vídeo nativamente.
- Gráficos y efectos 2D/3D: Proporciona una amplia gama de nuevas características que se ocupan de los gráficos en la web como lo son canvas 2D, WebGL,

[12] https://bitbucket.org
[13] https://modulus.io/

SVG, etc. Rendimiento e Integración: Proporciona una mayor optimización de la velocidad y un mejor uso del hardware.

- Acceso al dispositivo: Proporciona APIs para el uso de varios componentes internos de entrada y salida de nuestro dispositivo.
- CSS3: Nos ofrece una nueva gran variedad de opciones para hacer diseños más sofisticados.

Para potenciar el código HTML también se ha utilizado el sistema de renderizado de frontend llamado Blaze. Este paquete o plugin nos ofrece la posibilidad de utilizar etiquetas entre corchetes para la inserción de código y aportar dinamismo al codigo HTML. Esto nos aporta definir variables y funciones que se ejecutan en el cliente por medio de JavaScript y se muestran gracias al código HTML.

JavaScript

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

El lenguaje JavaScript ha sido utilizado en este proyecto tanto en el lado del cliente, como en el lado del servidor, como anteriormente se ha comentado en la sección Meteor. Por lo que es el principal lenguaje de desarrollo y el encargado de brindar toda la funcionalidad a la aplicación web.

CSS3[14]

Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML2. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser definida en un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales con el elemento «style» o en cada etiqueta particular mediante el atributo «style».

CSS se ha creado en varios niveles, cada nivel de CSS se construye sobre el anterior, generalmente añadiendo funciones al previo. En este proyecto se utilizará CSS3, siendo la más reciente de las versiones y con la mayor funcionalidad de todas.

CSS3 se ha utilizado junto a *Bootstrap* para dar estilo y maquetación a toda la aplicación.

[14]http://www.css3.info/

Capítulo 5

Arquitectura

En este capítulo se describe detalladamente la arquitectura en la que está basado el presente Trabajo Fin de Grado. Se trata de una arquitectura modular y adaptativa, esto es, dividida en módulos interconectados entre sí que dan lugar a una aplicación web, sobre los cuales se pueden realizar cambios e incorporar nuevas funcionalidades de una forma sencilla.

Para la descripción y diseño de la arquitectura del Sistema de Gestión de Trabajos Fin de Máster y Prácticas en Empresas se ha utilizado un *diseño descendente o top-down.* Es decir, una vez que se ha descrito y especificado la funcionalidad general de la aplicación, se discuten los módulos o componentes que la forman, de manera que el lector pueda tener una visión general de la aplicación y sus funcionalidades.

5.1 Descripción General

La Arquitectura del sistema (ver Figura 5.1) compuesta por una serie de módulos que se comunican con el servidor y éste a su vez con la base de datos para dar funcionalidad a nuestra aplicación web. Como ya se ha explicado en ocasiones anteriores, toda aplicación web dispone de una parte de cliente y una parte de servidor. Todo módulo del sistema necesita hacer peticiones al servidor a través del cliente, por lo que la interconexión y la necesidad de intercambio de información viene intrínseca en las propiedades del modelo y su funcionalidad.

Es oportuno destacar que cada módulo posee una funcionalidad y un fin específico que a su vez otros módulos utilizarán, por ejemplo, el módulo de autenticación al encargarse de la autenticación del usuario en el sistema será el que dote de información a los demás módulos a la hora de recoger información del usuario, o simplemente para el acceso a los diferentes módulos.

Una breve descripción de la funcionalidad de cada módulo es la siguiente:

- ***Módulo de Autenticación.*** La función principal de este módulo es la de la autenticación y registro de usuarios en el sistema. Será el encargado de recoger los datos de registro y procesarlos para registrarlos en la base de datos como nuevo usuario. Si el

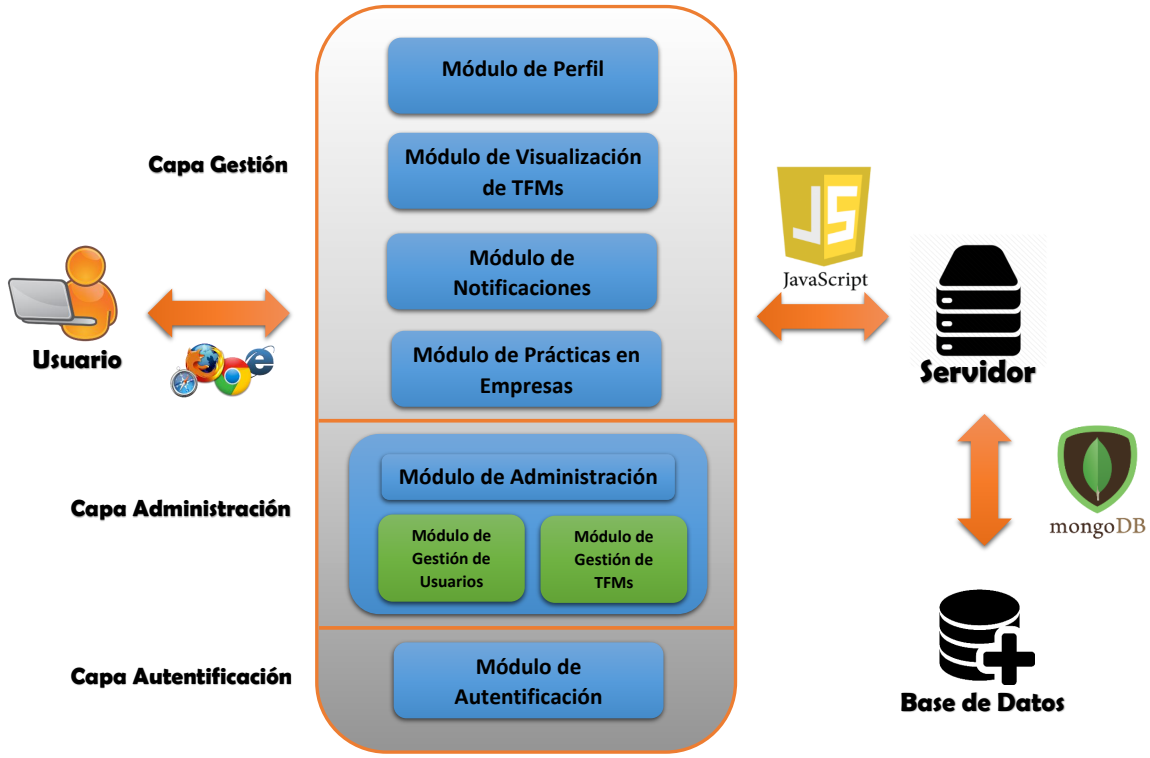

Figura 5.1: Arquitectura del Sistema

usuario ya está registrado en el sistema simplemente hará las funciones de *login* del sistema. Cabe destacar que existirán varios roles que identifique el estatus o rol del usuario en el sistema, el rol de alumno, el de tutor, el de miembro de la comisión del Máster, y el Coordinador del , o también llamado administrador del sistema.

- ***Módulo de Visualización de* TFMs.** Este módulo es el encargado de mostrar las diferentes propuestas de TFM al cliente. Según el rol que posea el usuario la información será de una u otra índole, aportando un contenido dinámico particular. Las principales funcionalidades de este módulo están ligadas a los tutores y miembros de la comisión, a los cuales les permitirá llevar un seguimiento de las propuestas que inscriba o le sean asignadas.

  Para poder visualizar las propuestas, primero el sistema deberá de aportar un método para poder inscribirlas. En este módulo se presenta un botón que redirige al usuario al submódulo de inscripción de TFM. En esta vista se recogerá a modo de formulario detallado los datos de la propuesta que el tutor vaya a cumplimentar, de este modo se guardarán en la base de datos las propuestas y todo su contenido detallado para su posterior validación. Una vez insertada, la visualización de TFM cobrará sentido, por lo que aparecerá dicha propuesta con etiquetas mostrando su estado de validación y dando la posibilidad de realizar el seguimiento anteriormente citado.

- ***Módulo de Administración.*** El módulo de Administración es probablemente el más importante del sistema ya que contiene todas las funciones para que la aplicación cobre sentido. El Coordinador será el encargado de hacer las funciones de administrador del sistema. Estas funciones tienen que ver con la gestión de usuarios y roles, y con la gestión de TFM las cuales se detallaran a continuación:
  - ***Módulo de Gestión de Usuarios y Roles.*** Este módulo es encargado de verificar el ingreso y registro de usuarios en el sistema. Para que el usuario pueda acceder al grueso de la aplicación deberá ser validado en el sistema, acción que deberá de permitir el Coordinador. Además podrá editar información de cualquier usuario y añadir o eliminar los roles que tenga asignados.
  - ***Módulo de Gestión de* TFM*s.*** El módulo más importante del sistema, encargado de recoger todas las propuestas de TFMy gestionarlas durante el proceso de validación por parte de los miembros de la comisión. Las funciones de este módulo son las de enviar la propuesta a la comisión para su validación, realizar el seguimiento de dichas valoraciones, editar cualquier propuesta de TFMy la posibilidad de validar la propuesta y aceptarla como TFM.
- ***Módulo de Notificaciones.*** Este módulo es el encargado de notificar al usuario en todo momento sobre los cambios que surjan en el sistema. Según el rol de usuario, estas notificaciones serán de un modo u otro. Permitirá avisar al usuario de cambios en su TFM, o al propio administrador de que un nuevo usuario se ha inscrito en el sistema y requiere su validación. Estas notificaciones nos redirigirán al módulo que contenga la funcionalidad que se requiera.
- ***Módulo de Prácticas en Empresas*.** El módulo de Prácticas en Empresas provee de una retroalimentación al Coordinador del Máster sobre los intereses del usuario respecto a las líneas de trabajo que más se adecúen a las pretensiones de su futuro empresarial. De este modo podrá estar al tanto de qué es lo más solicitado y poder enviar ofertas de prácticas mucho más personalizadas.
- **Módulo de Gestión de Perfil de Usuario.** Este módulo permitirá gestionar el perfil del usuario y poder dotar al sistema de información congruente con el individuo.

5.2 Modelo de Base de Datos

Para comenzar la explicación del modelado de la arquitectura de la aplicación, primero se deberá de definir que base de datos va a tener dicha aplicación y como está definida. La base de datos en una aplicación en uno de los componentes principales y de los que se debe partir para poder diseñar un sistema bien estructurado, seguro y efectivo. En el caso de *Meteor*, el framework que usaremos en este Trabajo Fin de Grado, como base de datos utiliza *MongoDB*. *MongoDB* es una base de datos NoSQL orientada a colecciones de documen-

tos. Como sistema de bases de datos NoSQL, éstas se caracterizan por diferir del modelo clásico del sistema de gestión de bases de datos relacionales. Por lo tanto no requieren de estructuras fijas o tablas, en su lugar utilizan estructuras de datos en documentos similares a JSON, llamado BSON[1], haciendo que la integración de datos en ciertas aplicaciones sea más fácil y rápida. Al no poseer relaciones intrínsecas al lenguaje, será labor del programador el encargado de relacionar los documentos y sus atributos como dese y realizar las consultas apropiadamente para conseguir la información que se requiera. Uno de las principales ventajas de *MongoDB* es la flexibilidad de sus colecciones de documentos, los cuales pueden o no tener los mismos campos o atributos, algo impensable en una base de datos SQL [AA15].

El modelo de base de datos utilizado en la aplicación está representado por el siguiente diagrama de entidad-relación:

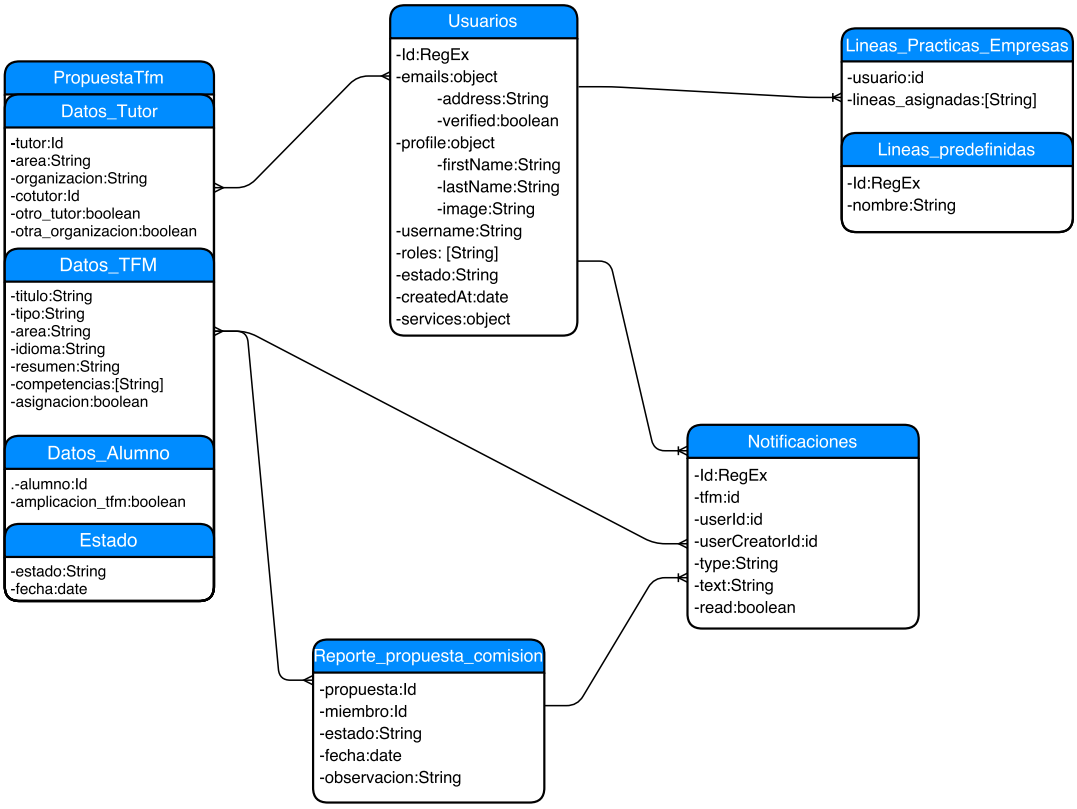

Figura 5.2: Diagrama Base de Datos

En este diagrama podemos apreciar las colecciones de documentos principales y como están relacionalmente enlazadas. Las colecciones importantes son la de *PropuestaTfm*, la colección de usuarios del sistema (*Usuarios*), la de *Notificaciones*, las relacionadas con las líneas de trabajo de prácticas en empresas y los reportes de las propuestas a valorar.

[1]es un formato de intercambio de datos usado principalmente para su almacenamiento y transferencia en la base de datos *MongoDB*. Es una representación binaria de estructuras de datos y mapas

Como puede observarse existen atributos dentro de algunas colecciones que vienen predefinidos en inglés, como por ejemplo en la colección usuarios que existen atributos como *profile* o *services*, ésto se debe a que se utiliza directamente los atributos de las clases proporcionadas por los paquetes o *plugins* que se utilizan en el proyecto para algunas funcionalidades.

- **La colección *PropuestaTfm*** tiene como propósito almacenar todos los datos que una propuesta de TFM necesita para ser guardada en el sistema. Estos datos tienen que ver con los datos del propio TFM, los datos del tutor que lo solicita y los datos del alumno si el TFM va a ser asignado. Además tendrá un campo de estado que identificará el estado de la solicitud o propuesta. La manera de representación tiene que ver con su funcionalidad dentro de la base datos, de esta forma cada uno de las subclases que posee la colección de *PropuestaTfm* es un objeto accesible a través de JavaScript, lo que facilita en gran medida su acceso y modificación.

```
Schema.PropuestaTfm = new SimpleSchema({
  datostutor: {
    type: Schema.DatosTutor,
    label: "datos del tutor"
  },

  datostfm: {
    type: Schema.DatosTfm,
    label: "datos del tfm"
  },

  datosalumno: {
    type: Schema.DatosAlumno,
    optional: true
  },
  estado: {
    type: Schema.EstadoTfm,
    optional: true{}
  }
});
}
```

Listado 5.1: Colección *PropuestaTfm*

Es posible acceder a los Datos del Tutor a traves de dicho objeto dentro de la colección *PropuestaTfm*, entendiendo por objeto a una subpropiedad de cada uno de los documentos dentro de la colección. Por lo tanto, como se puede apreciar en el código 5.1 la variable *Schema* que contiene la colección *PropuestaTfm* está compuesta por varios *subesquemas* que se tratan como objetos en el código y que poseen los atributos que definen a las colecciones. Estas *subcolecciones* son *DatosTutor*, *DatosTfm*, *DatosAlumno* y *EstadoTfm*. Cada una de ellas recogerá información por separado y modulando dicha información como si de un objeto se tratase. Por ejemplo, una de las

subcolecciones es la del código 5.2

```
Schema.DatosTutor = new SimpleSchema({
  tutor: {
    type: String,
    label: "Tutor del TFM",
  },
  area: {{}
    type: String,
    label: "Area",
    allowedValues: ['LSI','ATC','OE','MA']
  },
  organizacion: {
    type: String,
    label: "Organizacion",
    defaultValue: "UCLM"

  },
  cotutor:{
    type: String,
    label: "Co-Tutor del TFM",
    defaultValue: "",
    optional: true
  },
  otrotutor:{
    type: Boolean,
    optional: true,
    defaultValue: false
  },
  otraorganizacion:{
    type: Boolean,
    label: "Otro Tutor no inscrito en el Sistema",
    optional: true,
    defaultValue: false
  }
});
```

Listado 5.2: Colección *PropuestaTfm* - Datos Tutor

Tras apreciar los listados de código se puede entender cómo están definidas las colecciones y subcolecciones de las que aquellas se complementan, además de la descripción de cada uno de los atributos, con el tipo de datos que guarda y sus características. El listado 5.2 es un ejemplo de las cuatro *subcolecciones* que posee la colección *PropuestaTfm* y las cuales la definen. del lenguaje JavaScript se utilizan y referencian a través de objetos, por ello su similitud en el código y cómo éste se define.

- **La colección *Usuarios*** es la encargada de guardar todas las instancias de los usuarios que se registren en el sistema. Dicha colección es una colección que viene predefinida en la mayoría del código gracias a los paquetes externos de los que el framework y su maravillosa comunidad brinda a los desarrolladores. El propio framework ya incluye varios paquetes que gestionan a los usuarios y en este proyecto se utilizan algunos.

El paquete *accounts-password*[2] proporciona la colección *users* la cual se define y se añaden los atributos pertinentes para el proyecto.

```
Schema.User = new SimpleSchema({
    username: {
        type: String,
        regEx: /^[a-z0-9A-Z_]{3,15}$/,
        optional: true
    },
    emails: {
        type: [Object]
    },
    "emails.$.address": {
        type: String,
        regEx: SimpleSchema.RegEx.Email
    },
    "emails.$.verified": {
        type: Boolean
    },
    createdAt: {
        type: Date
    },
    profile: {
        type: Schema.UserProfile,
        optional: true
    },
    services: {
        type: Object,
        optional: true,
        blackbox: true
    },
    roles: {
        type: [String],
        optional: true,
        allowedValues : ['admin','tutor','comision','alumno','coordinador']
    },
    estado: {
      type: String,
      allowedValues: ['Validado','Pendiente',],
      autoValue: function () {
        if(this.isInsert){
          return "Pendiente";
        }
      },
      optional: true
    },
    area: {
      type: String,
      allowedValues: ['LSI','ATC','OE','MA'],
      optional: true
    }
});
```

Listado 5.3: Colección *Users*

[2]https://atmospherejs.com/meteor/accounts-password

Como se puede apreciar en la estructura del código, algunos atributos vendrán definidos en inglés para identificar a los atributos que trae el paquete, y algunos son diseñados e incluídos en el código para propósito del sistema.

Cada una de las instancias de usuario se trata como un documento. Cada documento será un usuario único en el sistema, y tendrá su información así como sus roles que lo definan. Los *roles* admitidos en el sistema son los de *admin* (Coordinador del Máster), *comisión* (CAM), *tutor* y *alumno*. Un usuario podrá poseer varios roles, pero no ninguno, por lo que hasta que no se le asigne uno y sea validado en el sistema por el Coordinador no podrá acceder a las funcionalidades de la aplicación. Una característica importante de esta colección es que no posee a simple vista un campo con la contraseña, atributo que lo gestionará automáticamente la base de datos por medio de un objeto llamado *services* (atributo con *blackbox*) el cual codifica la contraseña y provee de seguridad al sistema, a parte de otras muchas funciones ocultas.

- **La colección *Notificaciones*** tiene como objetivo almacenar las notificaciones que son creadas automáticamente por el sistema, de este modo se crearán y destruirán las notificaciones que sean pertinentes. Las notificaciones tienen varios tipos debido a que todas poseen la misma estructura y son identificadas según el usuario que las crea, el usuario destinatario y la finalidad de la misma. La colección viene predefinida en inglés y a pesar de la inclusión de otros atributos sin predefinir, éstos están plasmados en inglés para su cohesión con otros paquetes y funciones. La colección *Notificaciones* viene predefinida por los atributos *id* que identifica a la notificación dentro del sistema, *tfm* identificador del TFM dentro del sistema, *userId* identificador del usuario destinatario, *userCreatorId* creador de la notificación, *type* que define el tipo de la notificación, *text* que podrá incluir un texto identificativo y *read* que cambiará el estado de no leído a leído mediante un valor booleano.
- **La colección *Lineas_Prácticas_Empresas*** posee la información sobre las líneas de trabajo elegidas por cada usuario, a modo de intereses personales, para poder recibir ofertas de prácticas en empresas. Éstas líneas de trabajo son creadas, predefinidas y gestionadas por el administrador o Coordinador del sistema, el cual le sustraerá de información para su posterior envío de ofertas de prácticas. La colección guarda el usuario y las líneas de trabajo que ha elegido para poder acceder a ellas y obtener un resumen de los perfiles de los usuarios y las líneas más solicitadas. La colección de *Lineas_predefinidas* sólo contendrá su identificador (*Id*) y un *nombre* que la defina. Sin embargo la colección *Lineas_Prácticas_Empresas* contiene el *usuario* que crea la instancia y un *array* de *Lineas_predefinidas*. De esta forma se diseña qué líneas ha seleccionado cada usuario.
- **La colección *Reporte_Propuesta_Comision*** se crea con el fin de almacenar los informes de cada una de las valoraciones personales de todos los miembros de la CAM.

Cada vez que una propuesta de TFM es creada, el administrador recibe una notificación y envía la propuesta para su valoración a los miembros de la comisión. Dichos miembros deben de realizar su valoración y fallar en su veredicto. Cuando la decisión es tomada por dichos miembros, se crean unos reportes de informes en forma de documentos los cuales recogen esta información. Estos documentos son los que almacena esta colección y a los cuales solo el administrador o Coordinador tiene acceso.

Esta colección viene definida por los atributos *propuesta*, que será el identificador de la propuesta en el sistema, *miembro* que define el identificador del miembro de la comisión que debe de hacer la valoración, el *estado* que tendrá la propuesta dentro del sistema, la *fecha* de modificación de los estados asociados y una posible *observacion* a modo de texto explicativo para el Coordinador. Por lo tanto por cada propuesta, se crearán tantos reportes como miembros de la CAM formen dicho comisionado.

5.3 Estructura de la Aplicación

Esta sección pretende dar a conocer la estructura interna de la aplicación, a través de la definición y explicación generalizada de como está interiormente diseñado el sistema. En primer lugar comentar que la estructura de directorios dentro de un proyecto *Meteor* es de suma importancia para la comprensión e implementación de las funcionalidades de toda aplicación bajo este marco de trabajo. Para crear un proyecto *Meteor* desde cero, basta con escribir un comando en una consola o terminal de comandos:

```
meteor create <nombre del proyecto>
```

Una vez hecho esto, *Meteor* nos creará una estructura de proyecto en el directorio por defecto seleccionado, la cual está indicando al desarrollador la estructura que debe seguir el diseño y modulación de la aplicación o sistema web. La estructura de directorios que va a aparecer es similar a la de la siguiente imagen:

Como se puede apreciar en la imagen, se ha creado una serie de carpetas o directorios con unos archivos de ejemplo para ayudar a comprender esta estructura. Cada una de las carpetas deberá contener los archivos que definan y diseñen la aplicación con unas reglas básicas:

- **Directorio *client*.** Este directorio deberá contener todos los archivos referentes al cliente de la aplicación, es decir, todos las funcionalidades y cualidades que el cliente, por defecto un navegador, debe poseer para su buen funcionamiento, y que por lo tanto será información accesible a través del cliente. Cabe mencionar que cada archivo *html* deberá tener un archivo *js* que lo complemente en funcionalidad, otorgando dinamismo al código estático y funcionalidad al cliente.

  Por ejemplo, cualquier visualización del cliente de una página, por ejemplo la página de inicio, vendrá definida por su codigo *html* y su código *js*. Dentro de este directorio se encuentra todo el código que define la interfaz de la aplicación, así como sus fun-

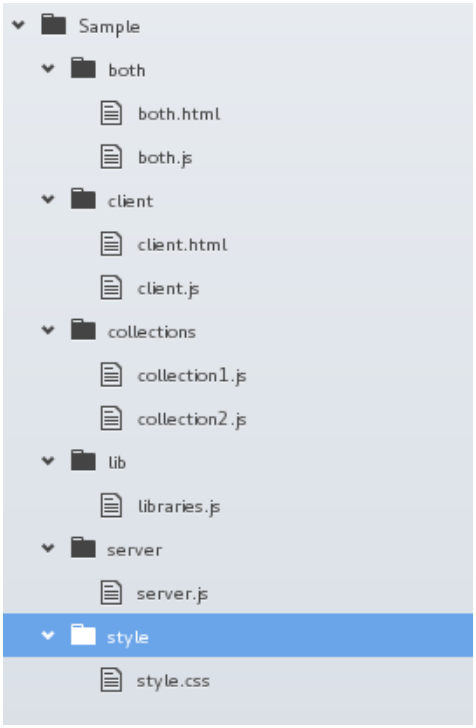

Figura 5.3: Estructura de directorio de un proyecto *Meteor*

cionalidades básicas que no requieren de una acción por parte del servidor, por lo tanto una gran parte del código total de la aplicación se encuentra bajo este directorio.

- **Directorio *both*.** Este directorio contendrá todos aquellos aspectos de la funcionalidad que son visibles y accesibles tanto al servidor, como al cliente. Por lo tanto serán objetos compartidos por ambas partes de la aplicación. Bajo este aspecto hay que tener cuidado de no introducir funcionalidades sensibles de información, y de no publicar datos con un carácter privado.

  Un ejemplo de funcionalidades que se ubican bajo este directorio es la de los controladores y enrutado de páginas. Esta funcionalidad debe ser visible tanto para el cliente como para el servidor, los cuales necesitan de esta información y funcionalidad para poder enviar información de una página a otra de la aplicación.

- **Directorio *server*.** El directorio server es quizá el directorio más interno y con mayor carácter secreto y privativo. Bajo este directorio, se deben encontrar todas aquellas funcionalidades, módulos o datos que el cliente requiera de una acción del servidor, por ejemplo, la devolución de algunos datos informativos de la base de datos, o la publicación de cierta información privada.

  Además contiene todos los mecanismos de creación de cuentas y de envío de información por parte del sistema a través de unos entes llamados *methods*. Estos *methods* o métodos en castellano, permiten al sistema aplicar el modelo vista controlador. De este modo se definen en el servidor una serie de métodos que solo son accesibles desde el

cliente por medio de una llamada a función. Como cualquier llamada a procedimiento, es una comunicación cliente-servidor que se hace efectiva y que sólo envía datos si el cliente se los pide bajo una llamada, si esto no sucede, el servidor no devolverá información alguna.

Una funcionalidad añadida de este modelo, es el de las publicaciones. El modelo publicador-suscriptor viene definido intrínsecamente en el modelo de base de datos. De esta forma se deberán de definir unas funciones que permitan la publicación de datos de la propia base de datos, pero de una manera segura, a través de objetos con la información que el programador desee, pudiendo elegir los campos y la información que se publica en el cliente. Además en el cliente se deberá suscribir a dicha publicación para que esos datos sean accesibles. De este modo se provee al sistema de una barrera segura entre el cliente y el servidor.

- **Directorio *collections*.** Este directorio, como su propio nombre indica contendrá toda la información referente a las colecciones de la base de datos. Se definirán dentro de esta carpeta y se definirá su estructura para su posterior utilización. Cada una de las colecciones se podrá definir en un archivo diferente para poder acceder y distinguir cada una de las colecciones que forman la base de datos de la aplicación. También se podrán definir las reglas de acceso, modificación e inserción además de varias características relacionadas con el modelo de base de datos utilizado.
- **Directorio *lib*.** El directorio lib contendrá todas las librerías o archivos de código de terceros que contengan servicios o funcionalidades externas a la aplicación.
- **Directorio *style*.** El directorio style existe para insertar todos los archivos de estilo o *css* qué darán forma a la aplicación, teniendo la posibilidad de tener toda la maquetación de la aplicación dentro de un mismo directorio. Otra opción es la de crear archivos *css* individuales y adjuntarlos en la carpeta client junto a su archivo *html*

5.4 Barra de Navegación

La barra de navegación no está incluida como un módulo dentro de la arquitectura o el diseño, pero sí cabe mencionarla y definirla para su posterior comprensión. Toda aplicación posee de un menú de navegación que provee a la propia aplicación de los métodos de navegación y posicionamiento dentro del sistema,de este modo se podrá navegar y acceder a los distintos módulos funcionales a través de una barra intuitiva y de fácil manejo. Cabe mencionar que dicha barra de navegación posee la cualidad de ser adaptativa al rol que el usuario porte en el sistema, de este modo, si el usuario es un alumno, las opciones de la barra de navegación serán mucho menores que si el usuario es el administrador, ya que deberá poseer, por ejemplo, la opción de navegar al modulo administrativo, obviamente oculto para el resto de usuarios.

Para visualizarlo mejor, se adjuntan 2 imágenes con la barra de navegación que le aparecerá al administrador (ver Figura 5.4) y la que le aparecerá aun alumno o tutor (ver Figura 5.5)

Figura 5.4: Barra navegación administrador

Figura 5.5: Barra navegación alumno

Como se puede apreciar, las barras de navegación son dinámicas y se adaptan a la funcionalidad que vaya a desempeñar el usuario, proporcionando al sistema la privacidad y métodos de seguridad que le permiten ser una aplicación robusta y confiable. Cabe destacar que un usuario sin validación, no podrá acceder a ninguno de los módulos o funcionalidades de la aplicación, mostrando una barra de navegación vacía con botones de inicio y perfil.

Para entender mejor su dinamismo en el código se adjunta una porción de dicho código en lenguaje HTML.

```
<div class="collapse navbar-collapse" id="main-nav">
        {{#if currentUser}}
          <ul class="nav navbar-nav navbar-right">
            <li class="{{ isActiveRoute regex='profile|mitutor'}} dropdown">
              <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                {{#with images}}
                <img id="avatar" src="{{this.url}}" width="25" height="25"><img/>
                {{/with}} <strong>{{ currentUser.emails.[0].address }}</strong> <i class="fa fa-
                    caret-down"></i>
              </a>
              <ul class="dropdown-menu" style="min-width: 167px">
                <li class="{{ isActiveRoute regex='profile'}}">
                <a href="{{pathFor 'profile'}}">
                    Perfil de Usuario </a>
                </li>
                {{#if isInRole "alumno"}}
                <li role="separator" class="divider"></li>
                <li class="{{ isActiveRoute regex='mitutor' }}">
                <a href="{{pathFor 'mitutor'}}">
                    Mi Tutor </a>
                </li>
                {{/if}}
                {{#if isInRole "tutor"}}

                {{/if}}
                ...
```

Listado 5.4: Código HTML Barra de Navegación

Como se puede apreciar en el código, el dinamismo se lo proporciona la capacidad del lenguaje JavaScript en el cliente para visualizar o no el botón correspondiente, de este modo, si no existe un usuario registrado en el sistema el código *{{#if currentUser}}* se ejecutaría como falso y no se mostraría nada de lo dedinido a continuación. Del mismo modo se otorga el dinamismo según el rol ocupado por el usuario utilizando las funciones *{{#if isInRole <rol>}}*. Este código se ejecuta en el cliente y muestra las opciones según se cumplan dichas condiciones o no.

5.5 Módulo de Autenticación

El módulo de autenticación es el principal módulo de acceso a la aplicación web. Este módulo es transversal a toda la aplicación y por ello se sitúa en el diagrama en la parte inferior del modelo de capas, en la capa de autenticación.

El módulo de autenticación tiene como principales funciones la del registro y creación de cuentas de usuario y la del acceso a la aplicación si el usuario ya posee cuenta registrada y validada en el sistema. La primera vez que el usuario acceda a la aplicación web, en la parte superior de la barra de navegación se podrán encontrar 2 botones que contendrán las funciones que anteriormente han sido nombradas, un botón dará la posibilidad de registrarse en el sistema y otro el de acceder al sistema.

El botón de registro en el sistema de un nuevo usuario nos llevará a un formulario de registro el cuál se deberá de rellenar con los datos del usuario para registrarse como nuevo usuario. El formulario a rellenar contendrá los campos necesarios para identificar al usuario y recoger los datos que se necesita de él, así como su email de contacto, su nombre y apellidos, una contraseña y la posibilidad de seleccionar el rol que poseemos dentro de la Universidad, esto es, el de alumno o el de tutor del Máster. Una vez rellenado el formulario y enviado a la aplicación, se redirigirá a la página de inicio de la aplicación, la cual aparecerá prácticamente vacía. El motivo es que cada usuario registrado en el sistema requiere de una validación en el mismo por parte del administrador o Coordinador del sistema. Una vez el administrador haya validado al usuario en la aplicación según los criterios establecidos, dicho usuario podrá acceder a todas las funcionalidades dispuestas en la aplicación.

El formulario de registro posee las características necesarias para detectar que el email introducido tiene un formato correcto, así como la repetición de la contraseña y la necesidad de que sea de un mínimo de longitud y la inclusión de algún dígito.

Por otra parte, el botón de acceso al sistema, mostrará el típico formulario de acceso a toda aplicación o sistema web, un campo para email registrado y otro para la contraseña que se facilitó al completar el registro en el sistema. Si los datos introducidos son correctos, la aplicación redirigirá al usuario a la página principal de la misma.

Una vez dentro de la aplicación, también se tendrá la posibilidad de salir de dicha aplica-

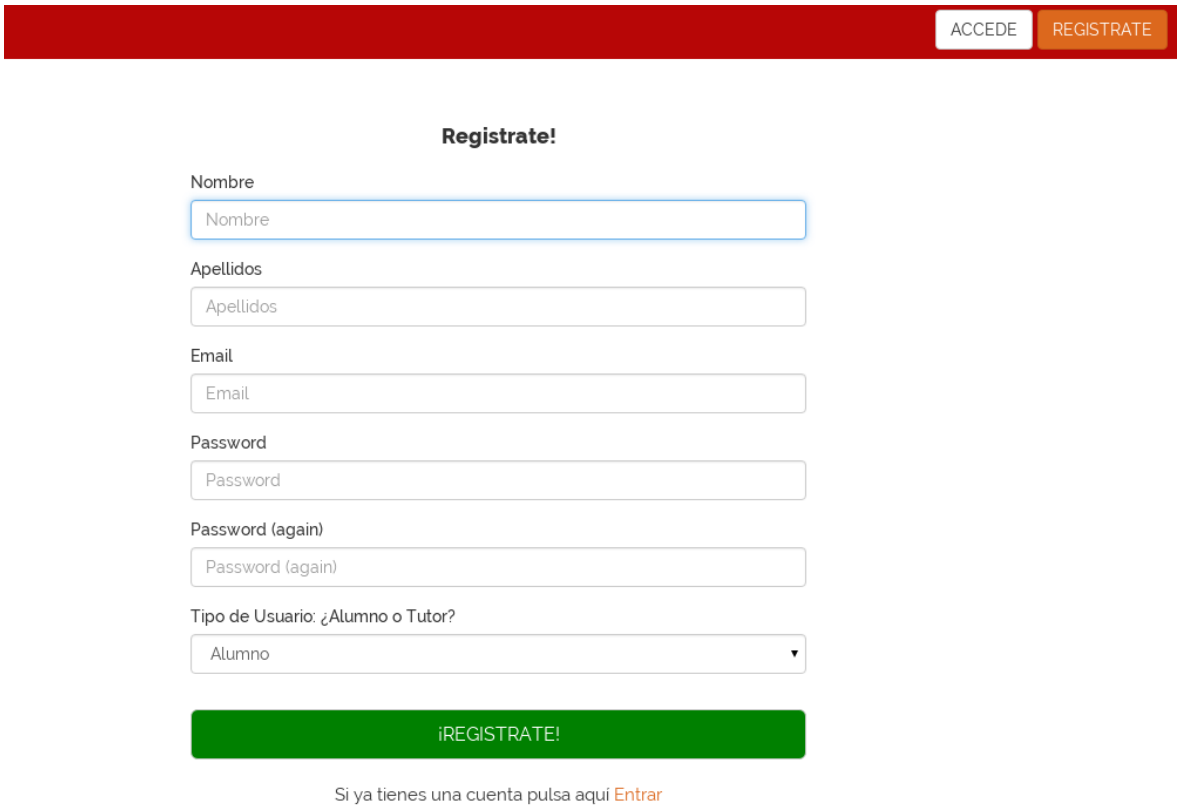

Figura 5.6: Registro en la aplicación web

ción y realizar el *logout*, para ello bastará con pulsar sobre el boton de la barra de navegación destinado al perfil de usuario y en el desplegable hacer *click* sobre el botón de Salir, lo que hará que se cierre la sesión de trabajo dentro de la aplicación.

El módulo de autenticación también posee algunas funcionalidades ocultas que son automáticas y transparentes a todo usuario. Una de ellas es por ejemplo el envío automático de correo electrónico al administrador para avisar de que un nuevo usuario se ha registrado en el sistema y necesita de verificación, de esta forma el administrador está informado en todo momento sobre algunas de las acciones que se llevan acabo en el sistema y que requieren de su atención.

El registro y autenticación en el sistema, así como la introducción de datos sensibles al usuario y de carácter personal, es siempre una tarea laboriosa de modelar. En diversas reuniones de diseño se han barajado y probado varias opciones de diseño que pasaban por utilizar cuentas de organizaciones externas como la UCLM hasta funciones de registro con cuentas de redes sociales actuales. Como opción final y dado el ámbito de la aplicación web, se optó por no usar ninguna autenticación externa y crear la propia del sistema, utilizando algunos paquetes relacionados con la creación de cuentas de usuario. Los usuarios del sistema son únicos y la información es la dada en el momento por el usuario, esta información solo estará visible para el administrador del sistema, siendo el Coordinador del Máster el único manipulador de datos.

5.6 Módulo de Administración

El Módulo de Administración es el módulo inmediatamente superior en el modelo de capas que se ha propuesto. Este módulo es práctica y funcionalmente el principal módulo del sistema y el que posee las funcionalidades primordiales que dan sentido a toda la aplicación web. A este módulo solo le está permitido acceder al Coordinador del Máster que hace las funciones de administrador del sistema, y es el principal gestor de las funcionalidades permitidas.

Figura 5.7: Estructura de directorio de un proyecto *Meteor*

Como principales funcionalidades del módulo de administración están las de gestionar las propuestas de TFM y las de gestionar a los usuarios del sistema y sus roles. Estas funcionalidades estarán detalladamente explicadas en las siguientes secciones.

5.6.1 Módulo de Gestión de TFM

Figura 5.8: Lista de propuestas TFM

El módulo de gestión de TFM es el principal submódulo del módulo de administración. Este submódulo contrendrá todas las funcionalidades y operaciones que un Coordinador del Máster necesita para gestionar cualquier propuesta de TFM y su validación en el sistema.

Como objeto principal, este panel alberga una lista de todos los TFM que se han inscrito en el sistema, con toda la información referente a la propia inscripción, tales como el titulo del TFM, el area al que pertenece, el idioma elegido para su realización, los datos del tutor que lo inscribe y si va a asignarse a algún alumno o no. Todos estos datos son de carácter obligatorio para que el Coordinador y los miembros de la CAM puedan valorar la propuesta.

Para facilitar la tarea de búsqueda se provee a la lista de un filtrado por estado el cual ayudará al gestor en la búsqueda y clasificación de las propuestas según su estado. De esta forma podrá filtrar para que sólo aparezcan propuestas por validar, o que ya estén validadas.

Figura 5.9: Envío de propuestas a la comisión

Seleccionando cada una de las propuestas que aparecerán en la lista, el Coordinador posee la funcionalidad de enviar dicha propuesta a la CAM para que realice su labor de validación, a través de un botón.

El Coordinador del Máster, al también formar parte de la CAM será el que de su último voto, al validar o no definitivamente la propuesta según el fallo de los miembros de la comisión. Para llevar a cabo este seguimiento se define el siguiente sistema:

Una vez se haya enviado la propuesta a la comisión, aparecerán una serie de elementos que identifican a través de botones a los miembros de la comisión y su veredicto. Con color azul el sistema muestra que el miembro de la comisión aún no ha valorado la propuesta, con color verde el miembro la ha validado positivamente y por el contrario si aparece en rojo, esto significará que ha fallado negativamente a la propuesta, añadiendo el por qué de la denegación a modo de observación. Para facilitar la labor de entendimiento y comprensión se dota a la vista de una leyenda que indicará lo anteriormente explicado.

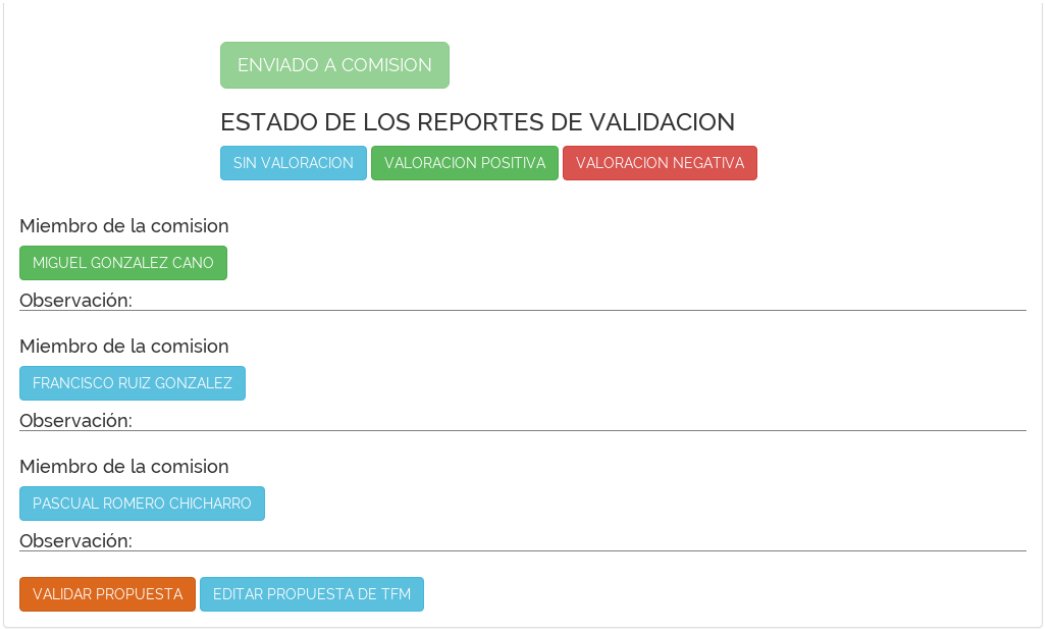

Figura 5.10: Seguimiento de los reportes de los miembros de la comisión.

El Coordinador podrá realizar el seguimiento en tiempo real y deberá realizar la labor de mediador pudiendo editar la propuesta y enviarle dicha información al tutor. Una vez crea conveniente el Coordinador, podrá validar finalmente la propuesta, enviándole una notificación automática al tutor con el objeto de avisarle de dicha modificación.

Por último se tendrá la posibilidad de acceder fácilmente a la otra parte de gestión, la de usuarios y roles, por medio de un simple botón, el cual redirigirá automáticamente.

5.6.2 Módulo de Gestión de Usuarios y Roles

El módulo de gestión de usuarios y roles es un módulo obligatorio y recurrente a la hora de realizar cualquier aplicación que albergue usuarios de distinta índole y con un fin diferenciado. El administrador debe de tener la posibilidad de gestionar a los usuarios que se inscriben para poder tener el sistema congruente y seguro.

Una vez se accede al panel de gestión de usuarios y roles, aparecerá una lista mostrando cada uno de los usuarios que han pedido ingreso o forman parte de la aplicación 5.11 Los campos que contendrá cada instancia de usuario serán los del nombre o email en su defecto, el rol que han solicitado o poseen en el sistema, y su estado de validación dentro de la propia aplicación. Además de las funcionalidades básicas de gestión.

La principal función del administrador dentro de este módulo es sin duda la de validación de usuarios en el sistema. Esta función es de crucial importancia debido a que sin su aprobación, el usuario no podrá acceder a las funciones y al contenido de la aplicación. El administrador es el único que posee las cualidades y características para validar al usuario en el sistema. Una vez validado, el usuario recibirá una notificación de que ha sido aceptado

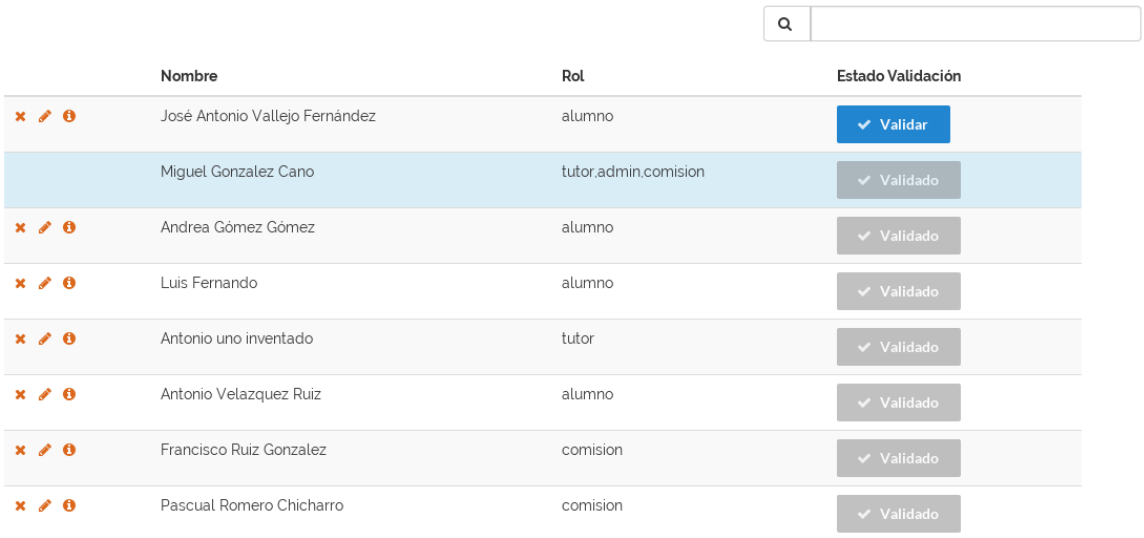

Figura 5.11: Imágen del módulo de gestión de usuarios

y puede acceder a las funcionalidades pertinentes. El hecho de incluir una función de validación en el sistema viene dado por la posibilidad que tiene un usuario cualquiera el hecho de poder registrarse en el sistema como tutor, seleccionando dicha opción en el registro. De este modo el administrador recibe la petición de entrada en el sistema y valora si es en realidad un tutor del Máster o simplemente es un impostor. De esta forma sólo habrá tutores verificados previamente por el administrador en el sistema.

Otra de las funciones principales es la de asignación o eliminación de roles de usuario, así como la posibilidad de poder crear unos nuevos. Los roles que se han propuesto en el diseño son los de alumno,tutor, miembro de la comisión y administrador del sistema. Dichos roles solo podrán ser asignados o cambiados por el administrador a través del panel correspondiente incluido en este módulo.

Como todo panel de gestión, dicha lista tendrá funciones de edición y vista de la información del usuario, así como la posibilidad de eliminar al usuario del sistema. También posee un cuadro de búsqueda para poder identificar con mayor facilidad a algún usuario en especial.

Para el diseño de la vista de gestión de usuarios, se ha hecho uso de las clases que *bootstrap* proporciona para tal desempeño, y en concreto se ha utilizado el siguiente fragmento de código para diseñar la tabla de usuarios:

```
<!-- Cabezera -->
<div class="ui four column internally celled center aligned pageable grid container" style="
    background-color: #F0F0F0;" id="grid">
  <div class="row">
      <div class="column">
        <h2 class="ui center aligned block black header">Email</h2>
      </div>
      <div class="column">
          <h2 class="ui center aligned block black header">Roles</h2>
      </div>
      <div class="column">
          <h2 class="ui center aligned block black header">Estado de Validacion</h2>
      </div>
      <div class="column">
    </div>
  </div>
  <!--Por cada resultado -->
  {{#each searchResults}}
  <div class="row"> <!-- FILA -->
    <div class="column">
      {{emails.[0].address}}
    </div>
    <div class="column">
        <input type="checkbox" class="userRolesInput" name="tutor_{{_id}}" checked={{isChecked
            this 'tutor'}}>
        <label>Tutor</label>
        <br>
        <input type="checkbox" class="userRolesInput" name="alumno_{{_id}}" checked={{isChecked
            this 'alumno'}}>
        <label>Alumno</label>
        <br>
        <input type="checkbox" class="userRolesInput" name="comision_{{_id}}" checked={{isChecked
            this 'comision'}}>
        <label>Comision</label>
    </div>
    <div class="column"> <!-- COLUMNA -->
      {{estado}}
      <br>
      <br>
      {{#unless validado estado}}
        <button id="boton-valida" user={{_id}} class="huge blue ui button"><i class="checkmark
            icon"></i>Validar</button>
      {{/unless}}
    </div>
    <div class="column">  <!-- COLUMNA -->
      <button id="actualizar" user={{_id}} data-dismiss="alert" class="huge blue ui button"> <i
          class="refresh icon"></i>Actualizar Usuario</button>
    </div>
  </div>
  {{/each}}
</div>
```

Listado 5.5: Código HTML Módulo Gestión de Usuarios

Como se puede apreciar en el código, la cabezera es estática, y siempre se muestra lo mismo. Sin embargo la lista de usuarios debe de ser dinámica y se utiliza el objeto *searchResults*, que se modifica según se haga uso de la búsqueda o no, para dotar de dinamismo al código y mostrar los usuarios de la base de datos. Cada una de las variables o identificadores que aparecen entre corchetes vienen definidas en el código javascript como funciones o variables que recogen la funcionalidad del código javascript que complementa al código HTML estático y le otorga variación ante eventos.

5.7 Módulo de Visualización e Inserción de TFMs

Desde el punto de vista del alumno o tutor, este módulo es el que cobra la mayor importancia dentro de la funcionalidad del sistema. Según sea uno u otro el rol del usuario, el módulo y su visualización será de una forma u otra.

Figura 5.12: Visualización de propuestas TFM

Desde el punto de vista de un tutor del Máster, este módulo posee todas las funcionalidades que necesita para presentar y obtener la validación de una propuesta de TFM. El tutor es el encargado de presentar las propuestas ya que el alumno no posee las cualidades ni características para su realización, y deberá de hacerlo bajo alguna tutorización. Para presentar la propuesta, el módulo está dotado de un botón que redirige al usuario al panel de inscripción de propuestas TFM. Este panel o submódulo presentará al usuario un formulario prácticamente idéntico e influenciado al que actualmente está creado bajo tecnología de terceros, y que en la actualidad es el método que debe seguir el tutor para inscribir la petición y el administrador para valorarla, principal motivo de la creación de este sistema. El flujo de trabajo que se sigue para que un tutor inscriba una propuesta de TFM es el indicado en la figura 5.13.

Como se puede apreciar en el diagrama cuando el tutor rellena el formulario de inserción de TFM el Coordinador recibe una notificación informándole de que dicha propuesta se ha registrado en el sistema, esto hará que el propio Coordinador envíe la propuesta a la CAM para que proceda a su labor de validación. Según los reportes ocasionados por los informes de los miembro de la CAM el Coordinador tomará la decisión de validar o denegar, lo que hará que se mande una notificación al tutor informándole sobre el estado de su proposición.

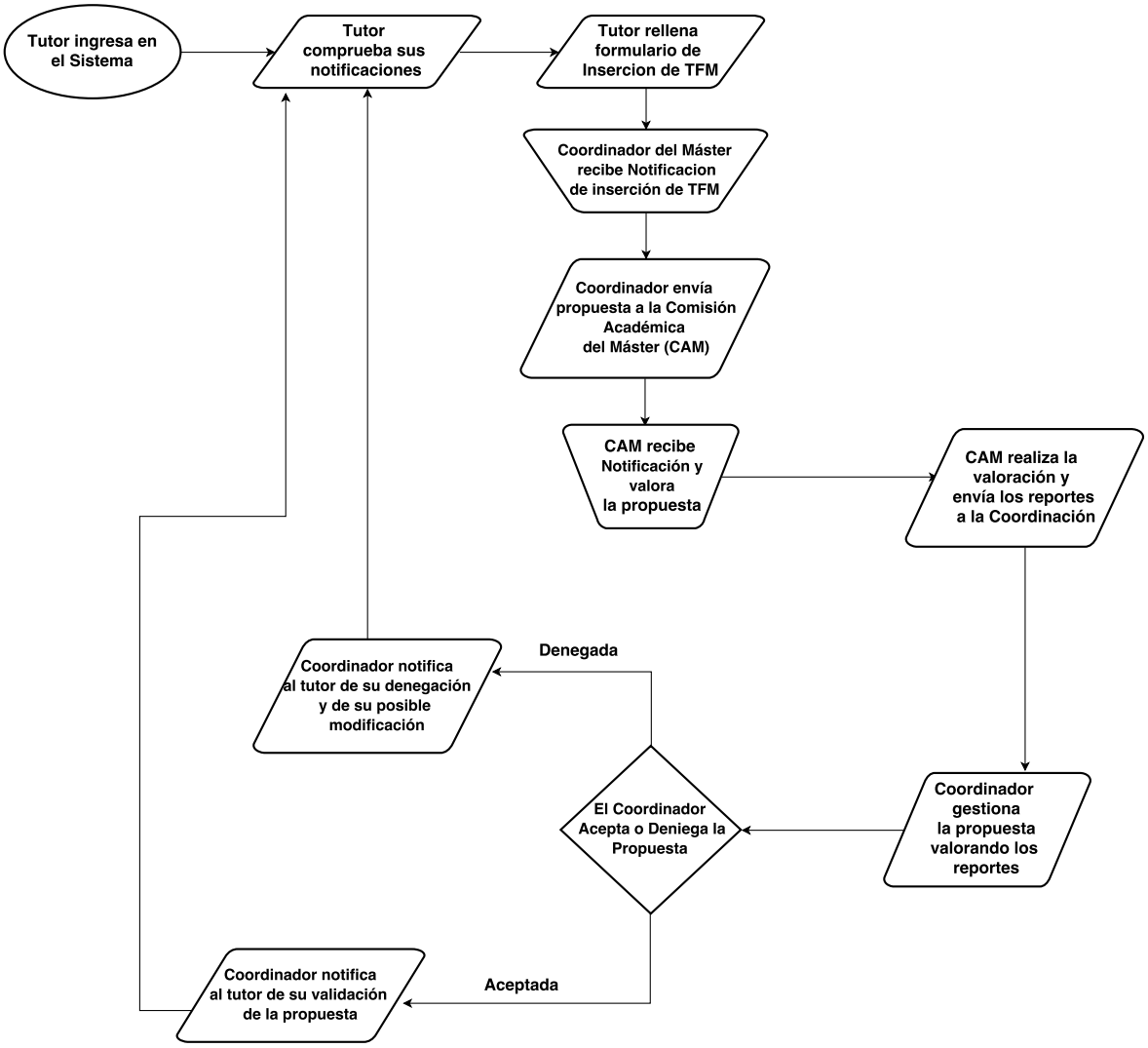

Figura 5.13: Diagrama de flujo de trabajo que sigue la inserción de TFM

El formulario propuesto recogerá toda la información necesaria para identificar la propuesta en el sistema y el tutor deberá cumplimentarla adecuadamente con todos los datos que se piden en este formulario. Dicho formulario puede apreciarse en la imagen 5.14

El formulario incluye todo tipo de sugerencias y está diseñado para que la tarea sea lo más sencilla e intuitiva. En la sección de Datos Tutor, el tutor deberá de introducir sus datos y algunos vendrán predefinidos o con opción única, debido a que recoge los datos del perfil del tutor para facilitar la funcionalidad del formulario. Además podrá elegir un co-tutor del Máster si éste lo necesita. En la sección de Datos TFMse recogerán los datos de propuesta,

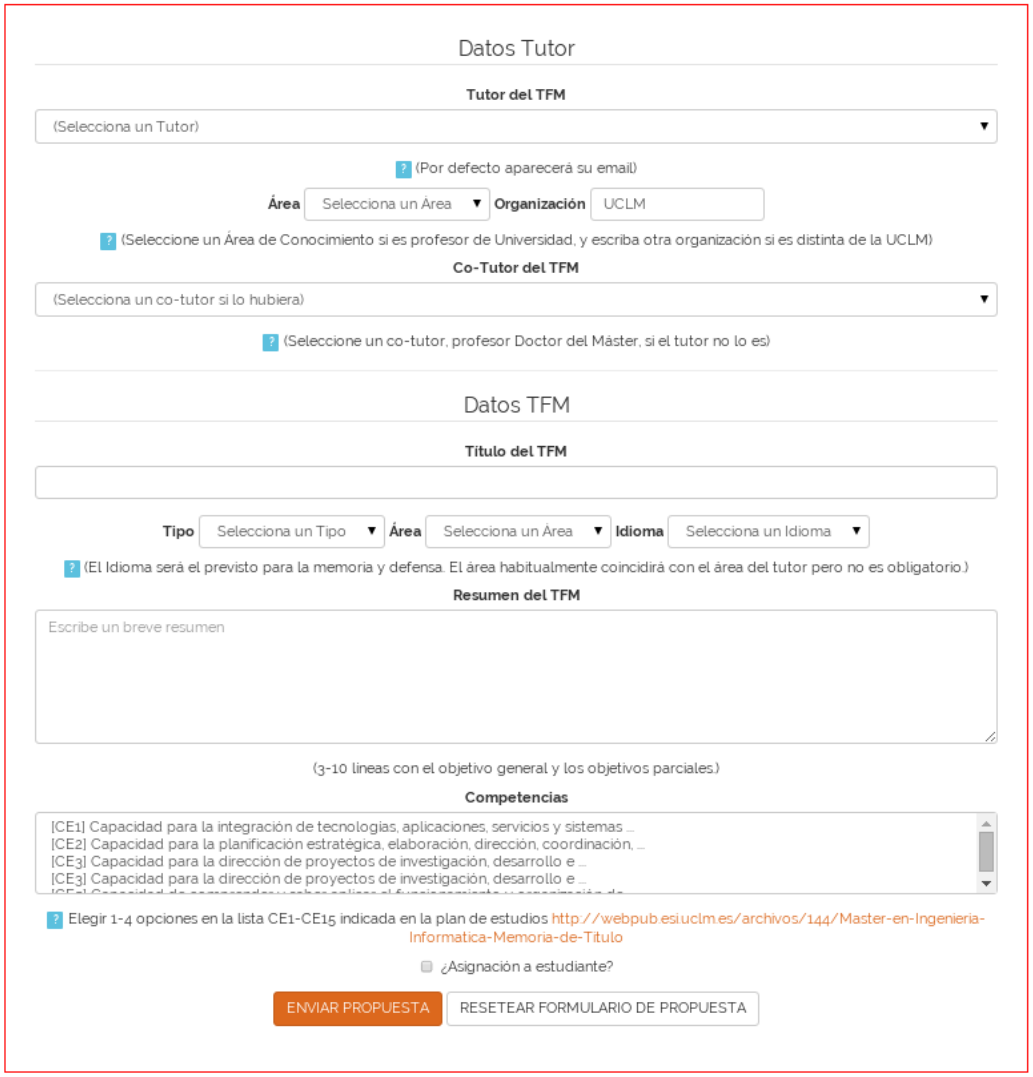

Figura 5.14: Formulario de inserción de propuesta TFM

los cuales pasan por el título del TFM, el tipo, el area y el idioma, así como un resumen de la propia propuesta. Además se deberán añadir las competencias que se recogen dentro del TFM. Por último se dará la posibilidad de asignación directa a un alumno y la indicación de si es o no ampliación de TFG. Una vez cumplimentado todo lo anterior y pulsando sobre el botón correspondiente a enviar la propuesta, dicha propuesta será enviada y almacenada en la base de datos, enviando las notificaciones automáticas pertinentes al Coordinador para su posterior evaluación, avisando al usuario de que su petición se ha completado de manera válida, o por el contrario, la inscripción no está completamente aceptada.

Para el diseño del formulario de inserción de TFM se ha hecho uso del paquete *meteor-autoform*[3]. Este paquete proporciona una herramienta de generación de formularios muy extensa, que no es necesario definir todo el código HTML para su diseño, sino que utiliza un lenguaje propio de definición de formularios que hacen uso de las colecciones para su inser-

[3]https://github.com/aldeed/meteor-autoform

ción y almacenamiento de la información. Para visualizarlo mejor, se adjunta una porción del código utilizado para este propósito.

```
{#autoForm collection="PropuestaTfm" type="insert" id="insertTFM"}}

<div id="formulario">
  <legend>Datos Tutor</legend>
  <div class="form-group">
    <label class="control-label">{{afFieldLabelText name='datostutor.tutor'}}</label>

    {{> afFieldInput name="datostutor.tutor" options=tutores_validados firstOption="(Selecciona un
        Tutor)" }}

    ....
    ....
```

Listado 5.6: Código HTML inserción de TFM

Como se puede apreciar, se define el formulario *autoForm* con la colección que manejará dicho formulario y acto seguido se definen cada uno de los campos del formulario. Por ejemplo, el primer campo del formulario es una selección del tutor de la propuesta, el cual viene definido por la linea de texto *{{ > afFieldInput ... }}*. Esta línea de código lo que realiza es la llamada a la creación de un campo de entrada el cual vendrá definido por el atributo *datostutor.tutor* de la colección *PropuestaTfm*. Las opciones vendrán definidas por una función y se asignará la primera opción a un texto específico. De esta forma se pueden definir cada uno de los campos del formulario sea cual sea su naturaleza y abstrayendo al programador de aspectos de diseño HTML que son tediosos de diseñar.

La visualización de TFMs y el submódulo que lo compete es el encargado de mostrar y permitir el seguimiento de todas las propuestas que se han presentado al sistema. De este modo aparecerá una lista con cada una de las propuestas (ver Figura 5.12) una serie de etiquetas que harán de labor informativa según el estado de dicha propuesta. El seguimiento se podrá llevar de una manera sencilla a través de notificaciones y retroalimentación por parte de la lista de propuestas. ajo el punto de vista de el alumno se podrá seguir la propuesta que se le haya sido asignada, de este modo podrá seguir simultáneamente la resolución de la propuesta junto a su tutor, permitiendo la comunicación a modo de mensajes internos si esto fuese necesario.

En esta vista se puede apreciar cada una de las propuestas que tiene tutorizadas y la información asociada, así como el estado y la fecha de su última modificación. El tutor tiene la posibilidad de cambiar la asignación del alumno o asignarlo por primera vez desde aquí. Un apunte importantes es el hecho de que un Coordinador o miembro de la comisión también puede actuar con su rol de tutor y proponer TFMs al sistema, por lo que en la vista de TFMle aparecerán las propuestas que tiene tutorizadas bajo el rol de tutor, y la lista total de

propuestas bajo el rol de administrador.

Figura 5.15: Vista de Valoración de Propuestas TFM

Por ultimo, los miembros de la CAM tendrán su sección de TFMs para valorar (ver Figura 5.15) , los cuales aparecerán bajo una lista titulada con todos los datos del TFM para su posterior validación. Una vez verificada la propuesta, fallarán en su resolución gracias a unos botones que validarán o no dicha propuesta, dando la posibilidad de incluir una observación del motivo de su denegación si esto fuese necesario. Estos reportes llegarán y se mostrarán en el panel de administración del Coordinador, el cual hará su trabajo pertinente explicado anteriormente en el módulo correspondiente.

5.8 Módulo de Notificaciones

El módulo de notificaciones es el encargado de automatizar el proceso de envío de advertencias o resoluciones de interés para el usuario de tal forma que siempre esté informado en todo momento de los cambios que produce el sistema. Las notificaciones pueden ser de muchos tipos y de carácter variado, de tal modo que según el rol y las funciones que lleve a cabo el usuario se podrán obtener y manejar unas notificaciones u otras. El hecho de crear un módulo exclusivo para este fin y presentarlo en el modelo de capas es que tiene una vital importancia a la hora de tener informado en todo momento y tener la sensación de que estás realizando el seguimiento en tiempo real.

Notificaciones hay de diversos tipos y destinadas a diferentes roles en el sistema, éstas se explican a continuación:

- **Notificaciones dirigidas al Coordinador.** Este tipo de notificación va destinada única y exclusivamente al administrador del sistema y pueden ser de varios tipos. Un ejemplo

de ellos serán las notificaciones que comuniquen un nuevo registro de usuario o una nueva inserción de propuesta TFM. También habrá del tipo: Ün miembro de la CAM ha validado un TFM". Como puede observarse, se intenta dotar al administrador de un medio que mantenga informado en tiempo real y le presente, cada vez que entra al sistema, los movimientos y nuevas actividades que se han resuelto mientras no estaba conectado al sistema.

- **Notificaciones dirigidas a los miembros de la CAM.** Los miembros de la CAM necesitan estar informados de las nuevas propuestas que les han sido enviadas para su labor de validación o denegación de las mismas. Por lo tanto, cada vez que el administrador les envíe una nueva propuesta para evaluar, recibirán una notificación que les informe de tal asunto, redirigiéndoles a la vista de verificación de propuestas.
- **Notificaciones dirigidas a los tutores.** Los usuarios que adquieran el rol de tutor del Máster dentro del sistema recibirán en todo momento notificaciones que les informarán sobre el estado de su propuesta de TFM. De este modo sabrán si ha sido validada por los miembros de la comisión o si su propuesta está denegada a falta de alguna modificación o simplemente que la propuesta no cumple con los requisitos establecidos. Si la propuesta es aceptada recibirán una notificación comunicándoles que ya puede ser asignada a un alumno o definitivamente puesta en marcha. Otro tipo de notificaciones que podrá recibir son los de aviso de mensaje interno. El sistema provee al usuario de un sistema de mensajería básico para que pueda comunicarse con los alumnos que tiene tutorizados, de manera que puedan enviarse mensajes y avisos de parte del alumno.
- **Notificaciones dirigidas al alumno.** Los alumnos recibirán dos tipos de notificaciones que están ligadas al TFMque tiene asignado y está en proceso de validación. El alumno en cuestión recibirá notificaciones por parte del sistema avisándole de que la propuesta que tiene asignada va siguiendo el proceso de validación. También recibirá alertas sobre nuevos mensajes internos por parte del tutor que le ha sido asignado.

5.9 Módulo de Prácticas en Empresas

El módulo de prácticas en empresas está funcionalmente diseñado y creado para los alumnos que decidan ingresar en la aplicación web. El principal motivo de la creación de este módulo es la falta de conocimiento por parte de la Coordinación del Máster y el personal administrativo de los intereses formativos y de carácter empresarial que tiene el alumnado. Actualmente existen medios de carácter organizativo dentro de la UCLM que permite al alumnado el recibir ofertas de prácticas que son ofertadas por las propias empresas a través de la UCLM utilizando la web oficial de la universidad. Estas ofertas llegan al alumno por medio de correo electrónico y necesitan de algunas características previas para poder acceder a ellas, por ejemplo la de haber cumplimentado el currículum vitae a modo de borrador.

El flujo de trabajo que existe actualmente se puede apreciar en la figura 1.2

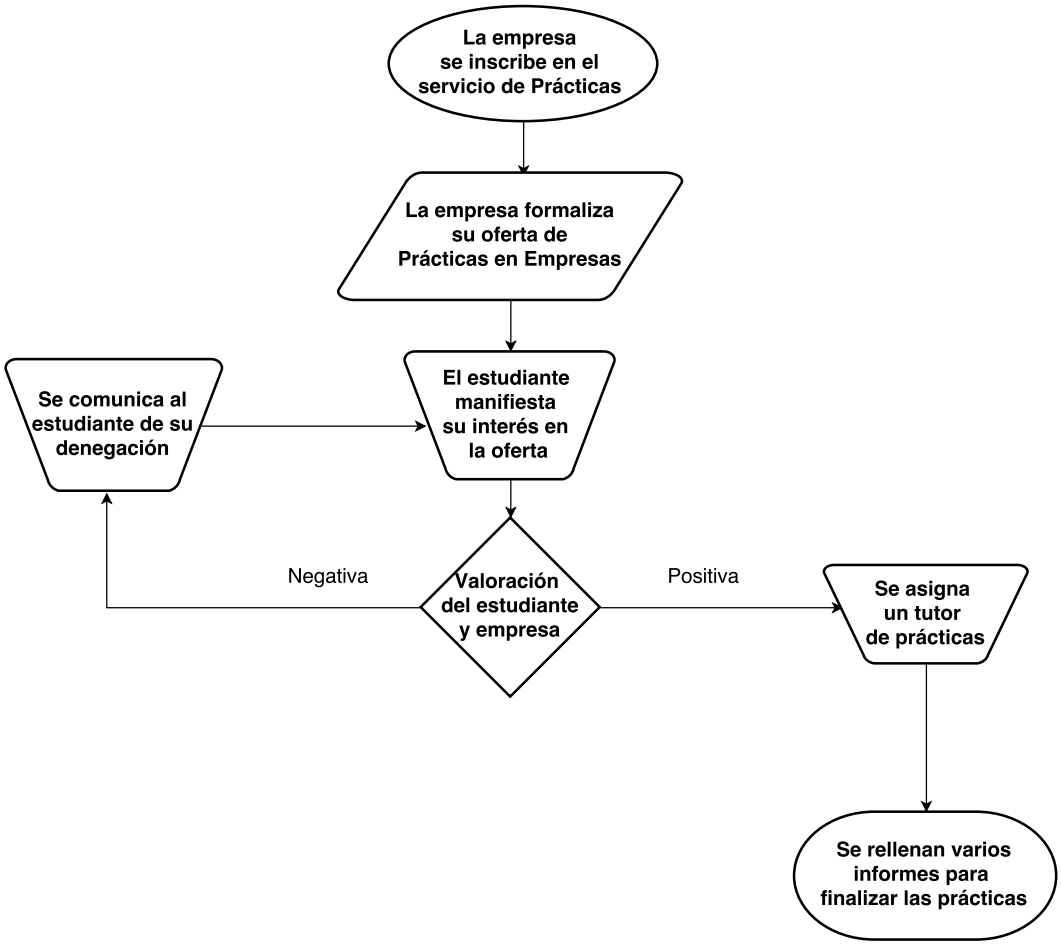

Figura 5.16: Diagrama de Flujo de la gestión de PE en la actualidad.

Todos estos procedimientos se hacen de manera automática y el alumno recibe notificaciones en forma de correo electrónico de ofertas que pueden o no interesarles directamente. De este modo, con un método que proporcione el conocimiento necesario sobre los intereses personales de cada alumno las ofertas que se presenten pueden ir mejor destinadas y pueden ser de mayor importancia para los individuos que están directamente interesados. Por todo lo ello y para facilitar un método interno a la aplicación y que sirva de ayuda a la Coordinación del Máster, se propone la inclusión de este módulo de prácticas en empresas.

Este módulo consiste en la definición y propuesta de una serie de conceptos que a modo selección de opciones se presenten al usuario y pueda elegir que líneas de trabajo son las más adecuadas a sus intereses profesionales de cara a su futura entrada al mercado laboral. De este modo el Coordinador expone las líneas de trabajo que tiene disponibles y puede tener un total conocimiento sobre la cantidad de usuarios que elige cada una de las líneas de trabajo.

Desde el punto de vista del alumnado, en este módulo se presentarán a modo de *checkboxes* las distintas opciones de líneas de trabajo que se han incluido en el sistema. El alumno

deberá elegir en las que está interesado y que tienen un carácter motivador en el terreno laboral. Una vez pulse el botón de guardar líneas de trabajo, éstas se guardaran en su perfil y esta información estará a disposición del Coordinador del Máster. La figura 5.17 muestra una vista del administrador del sistema.

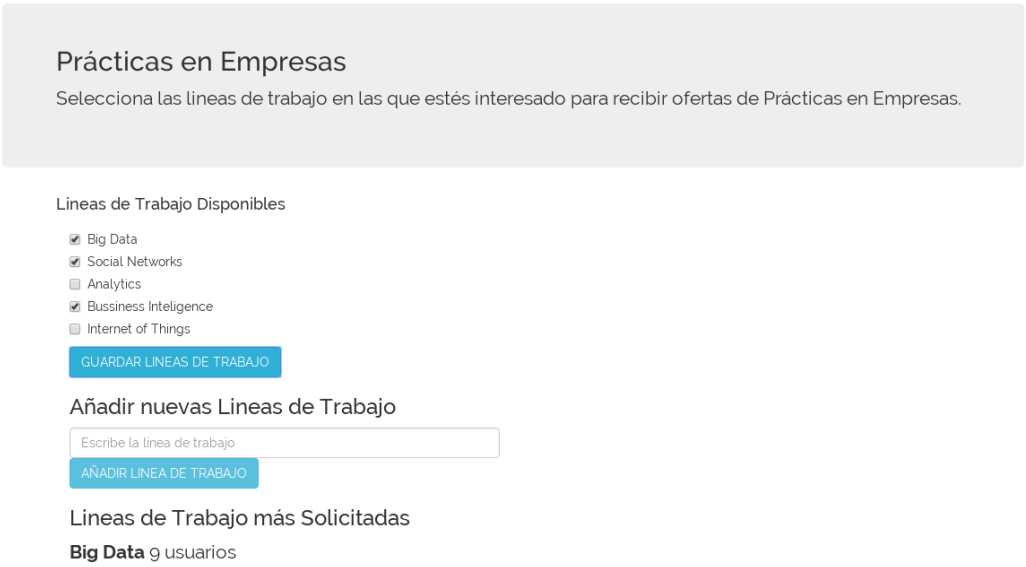

Figura 5.17: Vista del módulo Prácticas en Empresas según el Coordinador

Desde el punto de vista del Coordinador del Máster, este módulo tiene como funcionalidad principal la labor informativa que se ha comentado anteriormente, para el posterior envío de ofertas mucho más personales de prácticas en empresas. El Coordinador del Máster presenta una serie de conceptos a modo de líneas de trabajo, los cuales podrá crear y eliminar en todo momento, que escenifican y definen unas capacidades y aptitudes diferenciadas que segmentan las cualidades que el individuo prefiere o le interesan para su futuro laboral. De este modo, una vez el alumnado vaya ingresando sus preferencias, el Coordinador irá recibiendo información segmentada y cuantificada de que líneas son más solicitadas y cuales no son muy interesantes o elegidas. A modo de estudio estadístico se integrará una serie de gráficos que a simple vista le facilite esta labor al Coordinador.

Gracias a esto, la administración poseerá un método que le sustraiga de información, la cual podrá utilizar para seguir utilizando el mismo método de envío de prácticas, pero esta vez, podrá hacerlo mucho más selectivo y personal, convirtiendo así las ofertas en algo más valorable y directo.

5.10 Módulo de Gestión de Perfil

El módulo de gestión de perfil es indispensable y obligatorio de diseñar en toda aplicación web que permita el registro de usuarios. De este modo el usuario puede cumplimentar información referente a su perfil y rol en el sistema y proveer la retroalimentación de información

para los diferentes fines que la aplicación vaya a tener para su uso. La información del perfil de usuario le permite al administrador el poseer los datos necesarios para poder formalizar cualquiera de las instancias que se detallan en los Trabajos Fin de Máster y que datarán de carácter verídico a los individuos que la solicitan.

Figura 5.18: Gestión de perfil de usuario

Los datos de carácter meramente informativo pueden ser el nombre y apellidos, la fecha de nacimiento, el área del departamento al que pertenece si éste procediera, el género y la posibilidad de subir una foto o imagen a modo de avatar identificativo del usuario.

Además de la opción de cumplimentar y modificar los datos del perfil de usuario, también se dotara a éste módulo de una función de mensajería. A través del menú desplegable que el usuario encontrará en la sección del perfil de usuario, encontrará también una opción que le lleve a la mensajería interna. Como ya se ha explicado en secciones anteriores, los alumnos y tutores poseen un método de mensajería básica e interna que les provee de comunicación para asuntos referentes a la tutorización de los TFM.

Si el usuario es un tutor, éste podrá tener varios TFM bajo su tutorización y por lo tanto a varios alumnos asignados, por lo tanto, le podrán aparecer tantas instancias de alumnos como tenga. De este modo por ejemplo si sólo tuviera a uno, le aparecería la siguiente pantalla:

Como se puede apreciar, el sistema de mensajería es básico pero efectivo, y sirve para su propósito. Si el usuario es en cambio un alumno, la vista sería igual pero solo aparecería su

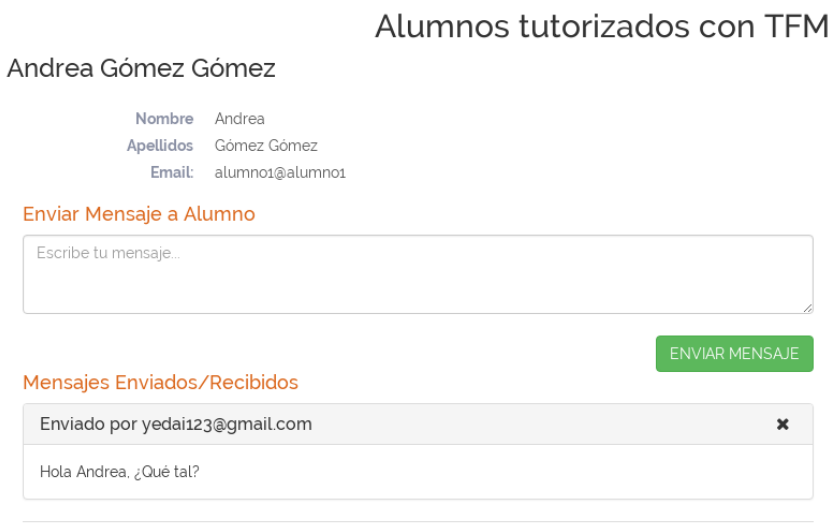

Figura 5.19: Imagen de la vista de mensajería

tutor asignado, y en su defecto si lo hubiera, su cotutor. De este modo se puede mantener una conversación interna sin necesidad de utilizar medios externos a esta aplicación.

5.11 Patrones de Diseño

Los patrones de diseño son una solución simple y recurrente a la hora de resolver problemas comunes del diseño orientado a objetos. Así pues no es necesario volver a determinar una solución diferente frente a problemas similares. Hay patrones de diversos tipos y para diversidad de problemas, además unos representan estructura y otros comportamiento. Según se diseñe una solución u otra se debe adoptar un patrón u otro. En este caso, se utilizarán algunos explicados a continuación [GHJV94]:

5.11.1 Patrón Modelo Vista Controlador (MVC)

El patrón MVC es un patrón de arquitectura de las aplicaciones software y que puede ser adoptado por todo tipo de aplicaciones que se rijan por un comportamiento específico a la hora de mostrar y manipular la información. Este patrón separa la lógica de negocio de la interfaz de usuario, facilitando la evolución por separado de sendos aspectos lo que supone un incremento en la reutilización y flexibilidad. El patrón se compone de varios módulos que se pueden apreciar en la siguiente figura:

La explicación a cada uno de los módulos es la siguiente:

- ***Modelo.*** El modelo contiene el núcleo funcional de la aplicación. Proporciona funciones para acceder a sus datos que son utilizadas por la vista para adquirir dicha información y mostrarlos al usuario
- ***Vista.*** La vista presentará la información al usuario de forma legible y comprensible,

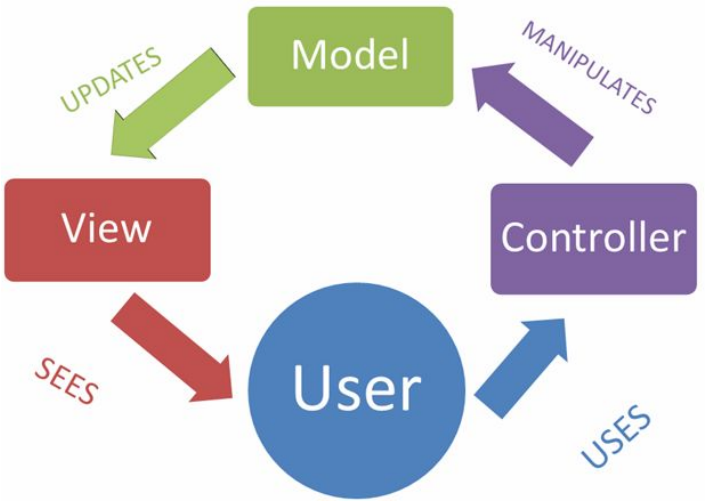

Figura 5.20: Modelo Vista Controlador (MVC)

normalmente mediante la interfaz de usuario, por lo que requiere de la información que el Modelo le proporciona.

- ***Controlador.*** El controlador define las entradas como eventos. Un controlador implementa procedimientos de atención a eventos y llamada a procedimientos que serán invocados cuando se produzca la interacción con el Modelo y se basan en representar y llevar a cabo todas esas peticiones.

Este patrón define la arquitectura y el diseño de la aplicación, por lo que es el patrón más importante de los utilizados.

5.11.2 Patrón Publicador-Subscriptor

El patrón publicador-subscriptor yace en la capacidad de que un subscriptor conozca el estado de un ente registrándose para ello o subscribiéndose a un publicador para conocer o recibir notificaciones cuando algún evento suceda en el publicador. Por lo tanto una aplicación que adquiera este patrón es aquella que posea un servidor que proporciona una serie de servicios a los usuarios para gestionar, almacenar y presentar todo tipo de información a los que otros usuarios se subscriben para ser avisados de nuevas actualizaciones de la información.

En esta aplicación, el patrón de publicador-subscriptor está aplicado en la metodología de la base de datos, de esta forma y como se explicó en su correspondiente sección, el servidor a modo de publicador, solo publica información que éste cree relevante, proporcionando a los subscriptores que se hayan suscrito a la publicación de la información que el servidor tiene para ellos. Por lo tanto proporciona los métodos para publicar solo la información que el desarrollador quiera, y permitir de métodos de seguridad a la base de datos de la aplicación.

Capítulo 6

Evolución y Resultados

En este capítulo se describen las fases e iteraciones de la evolución de la aplicación web diseñada, se discutirán y expondrán los diferentes resultados que se han obtenido después del desarrollo de este proyecto, además de una consecución de los distintos tipos de objetivos a lo largo del trabajo. También se incluye una sección con los resultados obtenidos de la aplicación web y una simulación de su uso.

6.1 Evolución de la aplicación web

En esta sección se detallan las diferentes iteraciones llevadas a cabo para el desarrollo de la aplicación web *Sistema de Gestión de Trabajos Fin de Máster y Prácticas en Empresas en el MII*. Se describe como información relevante los hitos alcanzados en cada iteración, la complejidad o el esfuerzo. La motivación principal de este Trabajo Fin de Grado es la **inexistencia de un método interno o aplicación más allá del uso de terceros para la gestión de propuestas de Trabajo Fin de Master (TFM) y su posterior veredicto de la CAM.** Tras un primer contacto por correo, se comentó la idea de querer realizar una aplicación web que utilizara una tecnología novedosa en el desarrollo de aplicaciones y sistemas web. El tutor de este TFG, David Vallejo Fernández, ese mismo año ostentaba el cargo de Coordinador del Máster Universitario en Ingeniería Informática y propuso la idea de un realizar una aplicación web que automatizara la gestión de trabajos ya que la metodología que se usaba para tal asunto era la del uso del correo electrónico y la aplicación de formularios de terceros. Acto seguido la idea se puso de manifiesto realizando una reunión de diseño del plan de trabajo y la captación de los requisitos de la plataforma. La puesta de manifiesto y comienzo del trabajo tuvo lugar en *Septiembre del 2015*.

6.1.1 Iteraciones e Hitos

El desarrollo del presente proyecto ha seguido la metodología *iterativa e incremental*, por lo que se presenta el seguimiento y evolución del trabajo a modo de iteraciones que iban formando poco a poco la aplicación web final. Al estar segmentado en módulos, varias iteraciones tendrán lugar a hitos que compondrán la implementación y puesta en marcha del módulo en cuestión.

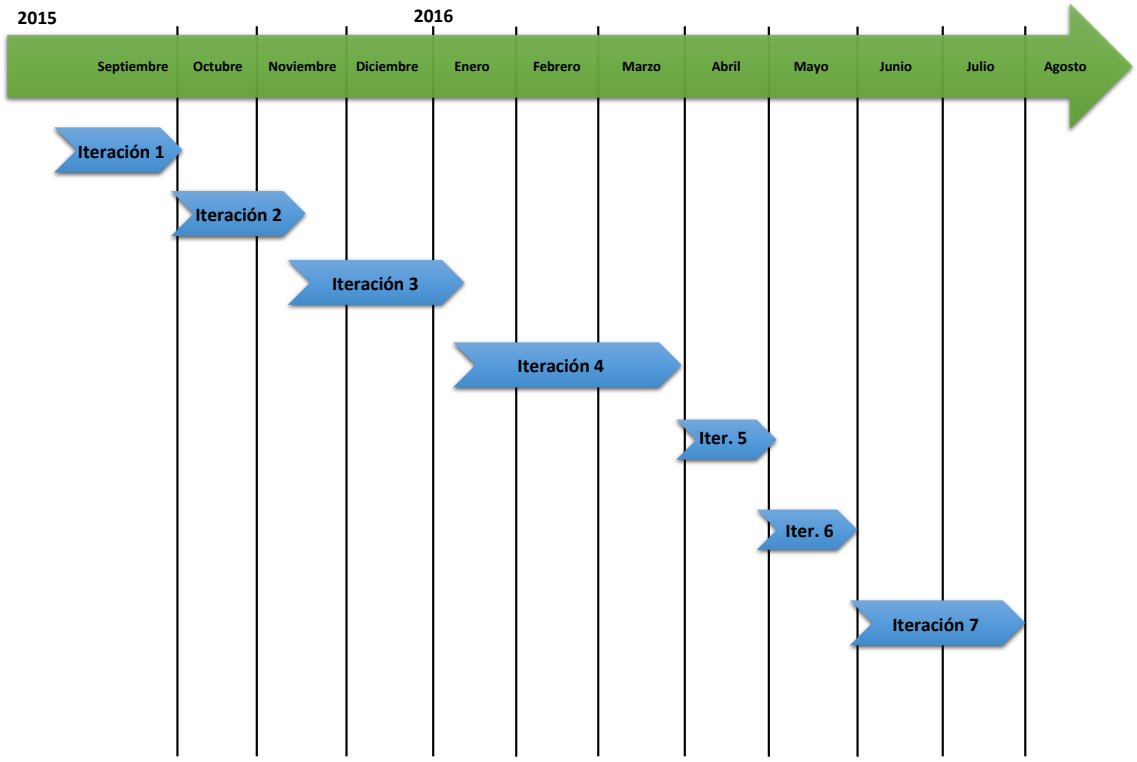

Figura 6.1: Evolución del TFG con respecto al tiempo

Iteración 1 - Introducción a Meteor y realización de aplicaciones de prueba

Durante esta primera iteración se propuso la lectura de manuales de documentación y tutoriales de introducción e iniciación a la creación de aplicaciones bajo esta tecnología. Se creó un portal básico de pruebas que sirvió de propuesta para una primera toma de contacto entre lo que se quería (reunión de requisitos) y lo que la plataforma ofrecía. Tras algunas semanas de pruebas, iniciación y comprensión de la plataforma, además como la adecuación a los lenguajes de programación se propuso la **reunión de diseño del esquema de trabajo y definición de la arquitectura de la aplicación**.

El *19 de Septiembre de 2015* se realiza dicha reunión entre alumno y tutor en la que ponen de manifiesto el esquema de desarrollo iterativo que se seguirá y que se expone a continuación en las diversas iteraciones.

Iteración 2 - Diseño y Definición de la Base de Datos

En esta primera iteración de desarrollo, se pone de manifiesto la creación de la base de datos que la aplicación web tendrá en un futuro. El *framework* utiliza un tipo de base de datos específico de tipo documental y no relacional, llamada MongoDB y comentada en secciones anteriores. *MongoDB* utiliza una definición de bases de datos basada en colecciones (ver Figura 5.3). Estas colecciones deben estar definidas de alguna forma siguiendo la estructura de definición apropiada.

Existen diversas formas de estructurar esta información y la que propone seguir en este trabajo es la denominada definición esquemática utilizando el paquete *Schemas*[1]. En estos esquemas se definen las colecciones a utilizar y qué campos pueden o no tener dichos documentos dentro de cada colección, es decir, se define la estructura de la base de datos por completo.

En primer lugar, se **definen las colecciones** *Usuarios* (ver listado 5.3), *PropuestasTfm*, que son las principales colecciones de documentos que se utilizan en primera instancia. En diversas iteraciones futuras se vuelve a este punto para diseñar las colecciones que son creadas a posteriori, venidas de la agregación de funcionalidad o módulos.

Una vez definidas las colecciones, se proveen de información y se **prueba el acceso y recuperación de datos del servidor**, actuando como un sistema de pruebas que permite la verificación del diseño y su implementación. Por último se instala una extensión para navegador la cual proporciona a modo de gestor una visualización de la base de datos y los documentos que aloja, de este modo se proporciona al diseñador de una herramienta de gestión básica de base de datos Mongo. Esta herramienta es *Mongol*[2].

El objetivo de esta iteración y por tanto el hito a comentar es el hecho de tener una aplicación web con una base de datos conectada y funcional, de modo que se pueda acceder, modificar y eliminar los datos mediante las funciones y búsquedas que proporciona la tecnología usada.

El resultado de esta iteración no supone ninguna aplicación funcional, sin embargo, se construye un esqueleto o esquema de base de datos que es accesible a través de funciones escritas en JavaScript siguiendo el esquema del modelo MVC.

Iteración 3 - Diseño e Implementación del Módulo de Gestión de Usuarios

En esta iteración se desarrolla uno de los módulos específicos de la aplicación. Este módulo en principio se diseño para que fuese constituido por varios submódulos funcionales, los cuales estarán aplicados en diversas funciones dentro de la aplicación. Para entender la funcionalidad de este módulo y concluir con la realización de un hito específico, se realizan sub-iteraciones o iteraciones más cortas para la implementación de los submódulos que componen al módulo.

- **Iteración 3.1 - Submódulo de autenticación.**

  En esta fase del desarrollo, se pone de manifiesto el uso y posible rediseño de la colección de usuarios. Para la realización del módulo de autenticación se adapta la colección creada con los paquetes que incluye el framework en relación con la gestión de usuarios. En concreto se usa el paquete *accounts-password*[3]. Este

[1] https://github.com/aldeed/meteor-simple-schema
[2] https://github.com/msavin/Mongol
[3] https://atmospherejs.com/meteor/accounts-password

paquete provee una estructura específica, así como el uso de diferentes funciones para la gestión de los usuarios.

En este periodo se tiene una reunión de diseño la cual se discute sobre la apariencia de la aplicación a la hora del **registro y el ingreso a la página de los usuarios**, así como las diversas tecnologías que se pueden utilizar para tal asunto. En primer lugar se dió la posibilidad de usar un sistema de autenticación existente como es el sistema de servidor *LDAP*[4] de la UCLM. Este sistema permitiría acceder a la aplicación web con las credenciales que poseen los alumnos y personal de la UCLM para acceder a todos los servicios que ésta organización brinda. Pero surgía el problema de la seguridad. Al tratarse de una validación externa, se debía extraer dicha información relacionada con el usuario de la información almacenada en el servidor *LDAP* para ser guardada en la aplicación web, y esto trajo varios problemas colaterales, los cuales surgían de la problemática del acceso consecutivo a un servidor oficial y en continuo uso para otros fines. El hecho también de la inseguridad del usuario al insertar credenciales de acceso oficial fue el detonante para dejar de investigar esta vía e intentar seguir otra.

Tras algunas semanas de pruebas se decide por crear un **método propio de registro y autenticación**, que no haga uso de aplicaciones de terceros ni de credenciales ajenas a la aplicación. El objetivo de implementación de este submódulo compete a la creación un método de acceso, el cual muestra botones de acceso y registro, además de los formularios pertinentes para tales asuntos. Para tal diseño se crea un diagrama de secuencia el cual representa las distintas partes involucradas en el registro de un usuario (ver imagen 6.2).

Como puede observarse en el diagrama se da pié a una gestión de usuarios por parte del Coordinador o administrador del sistema, funcionalidad de la cual se tratará en el módulo de gestión de usuarios y que se diseñará una interación más tarde.

El resultado de esta iteración, supone la creación de un prototipo que proponga la creación de cuentas de usuario, el ingreso del usuario si ya posee cuenta creada y la distinción entre la información que se muestra a un usuario que no se ha registrado con un usuario registrado en el sistema. A lo largo de esta iteración recurrente se ha desarrollado la funcionalidad del módulo de autenticación (ver sección 5.5).

- **Iteración 3.2 - Sumbódulo de Gestión de Usuarios y Roles.**

  En esta fase del desarrollo, se propone la realización de un submódulo que contenga la funcionalidad de gestión de los usuarios, y la creación de un sistema de

[4]Es el método de autenticación más adecuado en entornos de red medios y grandes. Los usuarios ingresan sus credenciales cuando accedan a sitios web. Las credenciales son verificadas contra un servidor externo usando el Protocolo Ligero de Acceso a Directorios (LDAP)

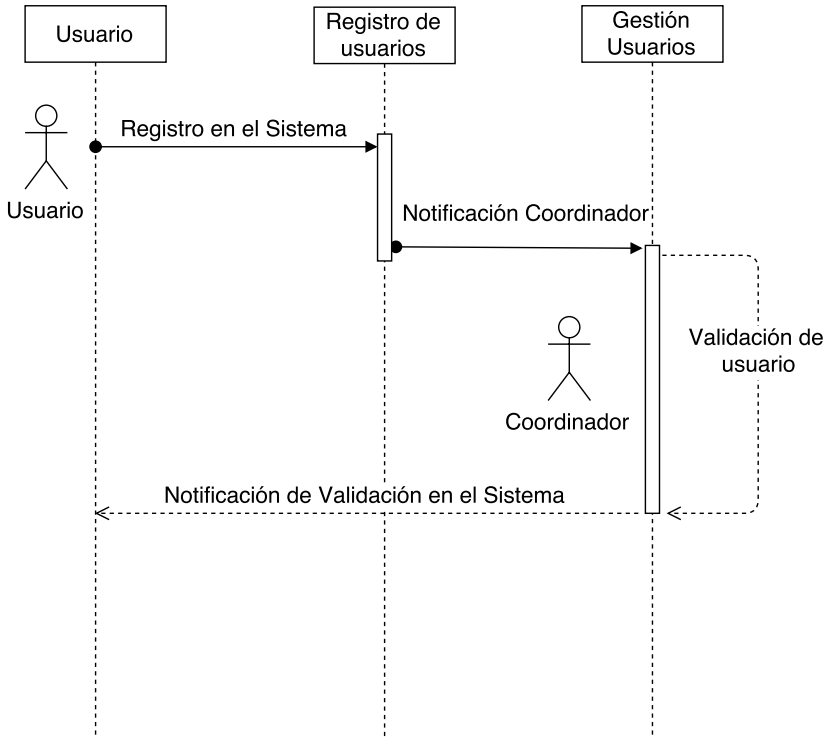

Figura 6.2: Diagrama de Secuencia del *registro de usuarios*

roles asociado a los usuarios. Para el diseño de esta vista se propone la utilización de un sistema de lista de usuarios que contengan diferentes datos identificativos y que sea posible editar dicha información, de modo que el administrador pueda modificar datos sensibles o incorrectos. Además se propone la inserción de un campo a la base de datos que contenga un estado de validación. De modo que el coordinador pueda validar al usuario en el sistema (ver imagen 6.2). Como segundo punto a desarrollar, se comenta la utilización de un paquete meteor que **añada la funcionalidad de los roles** a la colección usuario, además de proporcionar diversos métodos y funciones que permiten su gestión de una forma más eficiente. Este paquete es *meteor-roles*[5]. Una vez añadido el paquete al proyecto, éste incluye a la colección de usuarios un campo nuevo llamado roles, que identifica qué roles posee el usuario y la totalidad de los roles que tendrá o usará la aplicación. Gracias a la inclusión de este campo y los diversos métodos o funciones que contiene el paquete, se puede incluir en la implementación diversas vistas que cambien o sean diferenciadas según el rol que posea el usuario. Toda esta funcionalidad está diseñada y enfocada a la gestión por parte del administrador del sistema.

Como resultado de esta iteración se obtiene, por una parte, de una vista del submódulo de gestión de usuarios, que contendrá la lista con todos los usuarios del

[5]https://github.com/alanning/meteor-roles

sistema, junto a un botón para validar al usuario, y por otra parte, la posibilidad añadir, eliminar o modificar los roles establecidos a cada usuario, así como los totales que se han creado para la aplicación. Los roles que se proponen son los de tutor y alumno básicos en el diseño, el rol de administrador del sistema, y el rol de miembro de la Comisión Académica del Máster.

- **Iteración 3.3 - Submódulo de Gestión de Perfil.**

  Una vez se ha dotado al sistema de un método para la creación de cuentas de usuario, así como la propia gestión por parte de un administrador, toca la definición de un módulo que permita al usuario de gestionar su perfil. El diseño principal de esta vista viene influenciada por la mayoría de sitios web que proporcionan un método para la **gestión del perfil de usuario**, de modo que pueda introducir su información personal y dotar al sistema y más en concreto a la colección de usuarios de datos congruentes a su persona. De esta forma el sistema se retroalimentará de información que posteriormente utilizará para diversas funcionalidades como por ejemplo la inserción de TFM. El módulo de gestión de perfil ha variado poco en diseño debido a que la idea se tuvo en un principio fue diseñada para un uso y esa gestión no ha necesitado de apenas modificaciones o inserción de otras funcionalidades. Se propuso una vista básica de formulario, con la inserción de datos tales como el nombre de pila, los apellidos, la fecha de nacimiento, el género del usuario y algunos datos de carácter académico como puede ser el área u organización al que pertenece. Se añadió la posibilidad de subir e incluir una imagen de perfil para dar la intención de subir fotos u otras imágenes a modo de avatar de usuario, facilitando la verificación de los usuarios por parte de la administración. Se debe de incluir en esta iteración el diseño e implementación de una barra de navegación a modo de menú de aplicación. De esta forma se podrá acceder a través de dicha barra de navegación a las distintas vistas o módulos que hasta ahora componía la aplicación.

  Como resultado de esta iteración se obtiene una vista o página accesible mediante la barra de navegación y que bajo el nombre del usuario como botón de la misma se podría acceder al perfil de usuario. Bajo esta vista se puede gestionar toda la información y campos de la base de datos que sean de carácter informativo.

Como resultado de la tercera Iteración se obtiene un módulo que está formado por varios submódulos que se complementan, que manipulan y gestionan información referente a lo usuarios por parte de la administración y de los usuarios a modo personal. Estos submódulos estará localizados en diversas partes de la aplicación, pero se diseñan de manera conjunta y se realizan bajo una misma iteración general.

Iteración 4 - Diseño e Implementación del Módulo de Gestión de TFM

En esta fase se tiene implementado un prototipo de aplicación el cual posee una fun-

cionalidad básica de registro y autenticación de usuarios y algunas funcionalidades referentes a la gestión de dichos usuarios. A continuación se diseña un módulo que se complemente con varios submódulos para crear la funcionalidad principal de la aplicación, la inserción y gestión de propuestas de Trabajo Fin de Máster. De manera similar a la iteración anterior donde se diseña e implementa la gestión de los usuarios y en la que se opta por una subdivisión de módulos más básicos que complementen el módulo al completo, se pretende seguir con este mismo método para llevar a cabo esta tarea. Por lo tanto este módulo se subdivide en otros tres, los cuales se explicarán en las iteraciones siguientes:

- **Iteración 4.1 - Submódulo de inserción de propuestas TFM.**

  En esta fase de desarrollo de la aplicación, se pretende diseñar uno de los submódulos más importantes de la aplicación, el módulo de inserción de las propuestas TFM. En primer lugar se crea a modo de diagrama secuencial, tal y como se hizo con la gestión de usuarios, un diagrama que escenifique el proceso que debe seguir tanto los actores involucrados en el proceso como los objetos o módulos que forman parte de la gestión. Dicho diagrama es el de la figura 6.3.

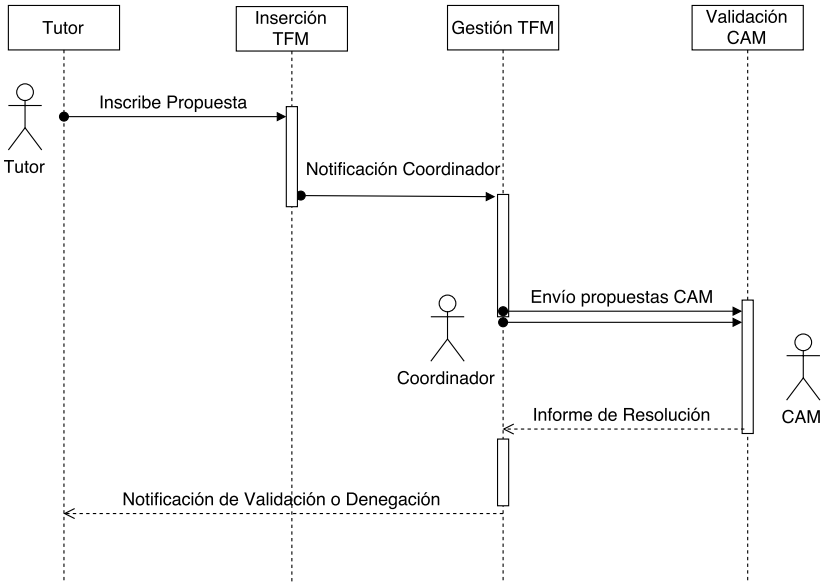

Figura 6.3: Diagrama de Secuencia de *Inserción de Propuestas TFM*

Como puede observarse en el diagrama, los actores involucrados son varios, tales como el *tutor*, el *Coordinador* del sistema y los *Miembros de la* CAM. Dicho diagrama establece la cantidad de módulos que deben crearse en el sistema, módulos que se crearán en sucesivas iteraciones y que darán funcionalidades varias al sistema.

Para diseñar el módulo que compete a esta iteración se pretende crear una vista que a modo de **formulario** permita la inserción y estandarización de las propues-

tas en el sistema. Partiendo de la base de la existencia de un *formulario Google* que se utiliza para formalizar dichas propuestas y que el coordinador pueda gestionarlas, se crea dicho formulario utilizando los elementos que nos proporciona las tecnologías usadas en el proyecto. De este modo y tras varias pruebas se diseña un formulario que a modo de inserción en diversos campos de información referente a la propuesta, se introduzca dicho documento en la base de datos y se identifique de modo que pueda ser gestionado desde la aplicación. Por lo tanto se rediseñan los esquemas de la base de datos para adecuarlo a las características y requisitos que se piden, obteniendo una página que incluye un formulario para enviar propuestas y almacenarlas de una manera legible tanto para la base de datos como para el desarrollador a la hora de su implementación. Para la realización del formulario se utiliza el paquete *meteor-autoform*[6] el cual proporciona a la plataforma de unos formularios más fácilmente accesibles y manipulables por parte del desarrollador, además de la facilitación del uso de código tanto HTML como Javascript para tal fin. Como resultado se obtiene una vista que permita a modo de formulario la inserción y almacenamiento de propuestasTFM en una base de datos, para posteriormente poder gestionar y utilizar dichos documentos en el proceso de validación.

- **Iteración 4.2 - Submódulo de visualización de propuestas TFM**.

  En esta iteración, se tiene como objetivo la obtención de una vista o página que a modo de módulo de la aplicación permita la visualización legible por parte de todo tipo de usuarios de las propuestasTFM. Como primera aproximación se diseña una **lista que muestre las propuestas** que están almacenadas en el sistema, mostrando los datos de dicha propuestas de una manera legible y estructurada. Una vez se obtiene dicha vista, se pasa a la elaboración de un método que permita dinamismo a esta vista, de modo que según el usuario que se encuentre esta vista le proporcionará una serie de funcionalidades y de información diferente a otra. Este dinamismo viene aportado por el paquete de roles mencionado anteriormente. De este modo, se puede implementar diferentes vistas y páginas, así como elementos visuales diferentes, según el rol del usuario que accede. Si el usuario es un alumno, la información mostrada debe de ser diferente a la información que se muestra a un tutor, o a un miembro de la CAM:

  - Si el usuario es un **alumno**, este solo podrá ver la propuesta que tiene o tendrá asignada, además de un posible listado de propuestas que han sido propuestas pero no asignadas así a ningún alumno.
  - Si el usuario es un **tutor**, éste sólo podrá acceder a las propuestas que como tutor ha inscrito anteriormente en el sistema.

[6] https://github.com/aldeed/meteor-autoform

- Si el usuario es un **miembro de la CAM**, éste tendrá funciones de validación de propuestas.(Iteración 4.3) Otro objetivo por tanto de este submódulo es la de dotar a esta vista de una funcionalidad que permita recibir y validar las propuestas que el coordinador le mande, de este modo se complementa al sistema de una validación interna, sin utilización de medios externos o de terceros.
- Si el usuario es el **administrador del sistema**, coordinador del máster por consiguiente y que tiene la potestad de enviar las propuestas a la CAM, realizando las labores de intermediario y además teniendo la última palabra como miembro de la CAM del máster, validando finalmente o no cada una de las propuestas que le sean recibidas. (Iteración 4.3)

Como resultado de esta iteración, se va obteniendo una vista con funcionalidades variopintas y que realizan tareas de mucha importancia dentro de la visualización deTFM. Por lo que no solo se visualizarán las propuestas, sino que se tendrá a mano herramientas y funcionalidades que están relacionadas con la gestión de Trabajos Fin de Máster.

- **Iteración 4.3 - Submódulo de validación de propuestas TFM**.

  Como se ha comentado en la Iteración anterior, este submódulo podemos incluirlo dentro de otro submódulo, pero se ha preferido tratarlo como único para su diseño, implementación y documentación adecuada. La funcionalidad que compete a este submódulo es la del **proceso de validación o denegación de la propuesta de TFM**. Una vez se ha formalizado y almacenado la propuesta que alguno de los tutores inscritos en el sistema ha creado, el administrador del sistema deberá de evaluar dicha propuesta y enviarla a la CAM. Para este propósito se diseñó e implementó la colección de *Reportes_propuesta_comision.* Esta colección automáticamente crearía instancias de reportes en forma de documentos por cada uno de los miembros de la CAM. Éstos contendrían la valoración positiva o negativa y una posible observación de los miembros de la CAM. El administrador podría **seguir el proceso de la valoración, viendo la actividad relacionada** y teniendo la última palabra a la hora de la verificación final, una vez tenga en cuenta las valoraciones. Para diseñar esto, se dotó al módulo de visualización de TFM de la posibilidad de si el usuario es miembro de la CAM, éste tuviera funcionalidades añadidas y creadas para dotar al módulo de la propia gestión adecuada. Este módulo dio lugar a que la funcionadidad que poseía el administrador para enviar propuestas y validarlas, estuviera estructurado bajo un **panel de administración**, junto a la gestión de usuarios.Por ello, al final se traslada dicha funcionalidad y se crea el módulo de gestión de propuestas TFM por parte del coordinador o administrador. Las funcionalidades en cuanto al diseño son las mismas.

El objetivo y por tanto el resultado de esta iteración es la adición de contenido y funcionalidades a las clases que implementan la vista de TFM y dotar al sistema de distintas acciones de gestión según el usuario que acceda al sistema.

Como resultado de la cuarta iteración, se obtiene el módulo de visualización deTFMs y de validación de propuestas por parte de la CAM. Esta visión da lugar a la creación de otro submódulo que forma parte del módulo de administración, ya que el administrador posee funcionalidades añadidas. Por lo que esta iteración ha sido recurrente e implementada a la vez que algunas de las iteraciones siguientes.

Iteración 5 - Diseño e Implementación del Módulo de Notificaciones

En esta iteración se adapta el sistema a la posibilidad de **envío y recepción de notificaciones de usuario**. En una de las reuniones que se llevaban a cabo, se tomó la decisión de que bajo ciertas acciones o eventos se enviara una notificación visual que sirviera al sistema de un gestor de eventos o acciones. De este modo un usuario sabría si ha recibido algún mensaje durante su ausencia, o si se le requiere para que lleve a cabo una acción específica en el sistema. Para diseñar tal asunto, se creó la colección *"Notificaciones"*. Para enviar y recibir notificaciones hacía falta que éstas se guardaran y almacenaran en alguna colección para posteriormente poder acceder a ellas, leerlas y mostrarlas. Una vez vistas, deberían de eliminarse automáticamente. Una vez la mayoría de las acciones ya habían sido implementadas, se adaptó este módulo de notificaciones a las cuales simplemente se le añadía la funcionalidad de disparar este evento. Esta tarea se ha ido implementando también en las sucesivas iteraciones dando lugar al módulo completo.

Como resultado de esta iteración se tiene la implementación de un sistema básico de notificaciones entre usuarios de todo tipo y la comunicación de eventos, funcionalidad que ha ido modificando y añadiendo aspectos de diseño en iteraciones siguientes.

Iteración 6 - Diseño e Implementación del Módulo de Gestión de PE

En esta etapa la visión general de la propuesta se había concluido y quedaban por añadir algunas funcionalidades que diseñaran el aspecto de las prácticas en empresas. Partiendo de la existencia de un mecanismo que bajo la web de la UCLM es gestionada y que envía correos con solicitudes de prácticas en empresas, se quería añadir funcionalidad y valor a estas proposiciones, de modo que los alumnos que se registraran en la aplicación tuvieran la posibilidad de proveer información acerca de sus intereses, y que se aplicara dicha información a la hora de los envíos de ofertas de prácticas en empresas. De este modo surgió la posibilidad de añadir una serie de conceptos relacionados con las tecnologías a modo de líneas de trabajo que especificaran en qué áreas se sentían más agusto trabajando o simplemente que gustos o prioridades tenía por unas líneas u otras. Para diseñar e implementar esta parte, se crearon dos colecciones de documentos los cuales guardarían información referente a estos aspectos. Una colección

sería predefinida con algunos ejemplos de **líneas de trabajo**, dando la posibilidad de añadir nuevos elementos, y otra colección que guardara las **líneas elegidas por cada uno de los usuarios** que las almacenara. De este modo quedaría registrado en el sistema esta información y se podría presentar a modo de gráficos o de informe de estudio de la situación. Para implementar dicha sección se emplearon varios elementos que darían forma a una selección de elementos por parte del alumno, y la posibilidad de eliminar y añadir nuevas líneas de trabajo por parte del administrador.

Como resultado de esta iteración se da lugar a la pestaña Prácticas en Empresas que tiene una funcionalidad informadora por parte del alumno, y una cualidad estadística y administrativa por parte del coordinador o administrador del sistema.

Iteración 7 - Pruebas finales, integración y despliegue de la aplicación.

Una vez se tuvo un prototipo final de la aplicación, se pasó a dar estilo y a maquetar el **diseño visual final** de la aplicación. Este aspecto cabe destacarlo porque es un aspecto importante del diseño de la interfaz de usuario. Al mismo tiempo que se iban puliendo los aspectos de diseño visual, se iban realizando varias **pruebas de integración** en varios sistemas y navegadores, como dispositivos móviles, tablets y diversos sistemas operativos. Se realizan pruebas de verificación y validación de usuarios y TFM.

Para probar el **despliegue de la aplicación en remoto** Meteor proveía, en los tiempos de inicio de este proyecto, un método automático que desplegaba cualquier aplicación creada por un desarrollador en remoto, en sus servidores particulares. Con un sencillo comando la aplicación se desplegaba y śe obtenía una url en la cual la aplicación estaba alojada. En Abril de 2016 la compañía dejó de prestar el servicio y se pasó al modelo de pago, el cual se debía pagar por tiempo alojado. Con el paso del tiempo se fue investigando hasta que después de varias pruebas se optó por usar un modelo de cobro por tiempo, al principio subvencionado por periodo de prueba. Este modelo es el que está usado en la actualidad para desplegar la aplicación propuesta. La plataforma es *Modulus* y acepta multitud de plataformas entre ellas *Meteor*.

6.1.2 Evolución Temporal

La evolución del presente proyecto con respecto al tiempo puede observarse en la siguiente figura 6.1.

El tiempo estimado en la elaboración de la presente aplicación ha sido de 50 semanas aproximadamente, las cuales han estado divididdas según indica la siguiente tabla 6.1.

6.1.3 Recursos y Costes

En este apartado se recogeran unas estadisticas asociadas a los recursos y costes del proyecto. En primer lugar se ofrece de manera estadística una tabla con la representación del número de archivos de los cuales depende este sistema y que estarán recogidos en su corres-

Numero de Iteración	Fecha Inicio	Fecha Fin
Iteración 1	1 Septiembre	5 Octubre
Iteración 2	5 Octubre	15 Noviembre
Iteracion 3	15 Noviembre	10 Enero
Iteracion 4	10 Enero	20 Marzo
Iteracion 5	28 Marzo	125 Abril
Iteracion 6	25 Abril	23 Mayo
Iteracion 7	23 Mayo	31 Julio

Cuadro 6.1: Tabla de duracion de iteraciones

Tipo de Archivo	Número de Archivos	Líneas en Blanco	Líneas de Comentario	Líneas de Código	Total Líneas
JavaScript	59	1299	819	4969	7087
HTML	30	364	138	1368	1870
CSS	10	19	0	195	214
XML	1	0	0	23	23
TOTAL	100	1676	2633	6555	**9194**

Cuadro 6.2: Representación del número de archivos y líneas de código totales de las que se compone el proyecto

pondiente repositorio para su verificación. Dicha tabla (ver tabla 6.2) indica la cantidad de archivos y líneas de código que presenta la aplicación web. Como puede observarse, un proyecto de esta magnitud y de esta naturaleza posee una gran cantidad de archivos de diverso tipo y muchas líneas de código.

Los costes asociados al proyecto han sido los recogidos en la siguiente tabla 6.3:

El *hosting* asociado al proyecto ha tenido que reflejarse debido a que el sistema gratuito de despliegue de aplicaciones *Meteor* dió de baja el servicio. En su lugar se contrató uno de prueba. La estimación del *programador* incluye las labores de diseño y maquetación, trabajo que de cualquier otro modo debería realizarlo otro individuo, con los costes que esto supondría.

Recursos	Cantidad	Coste (€/unidad)	TOTAL
Programador	1200 horas	20	24000 €
Ordenador Portátil	1	750	750 €
Hosting	100 horas	1.5	150 €
TOTAL			24900 €

Cuadro 6.3: Estimación del coste total del proyecto

6.2 Resultados

La aplicación obtenida con la realización del presente proyecto permite la posibilidad de gestión de unas de las facetas más importantes del máster en ingeniería informática como es la gestión de los trabajos finales, con todo lo que ello conlleva. Gracias a la aplicación de este sistema se da la posibilidad de utilizar una tecnología web y en definitiva una aplicación para la gestión de los TFM. Además se provee a la aplicación de unas funcionalidades que sirven como beneficio y enriquecimiento de la comunicación y del desarrollo de ciertos aspectos como pueden ser las prácticas en empresas. La facilidad de uso y de acceso por medio de diversos dispositivos o navegadores provee al sistema la entrada a la nueva generación de aplicaciones de tipo web o nativas, siendo ésta modificable con unos cambios en el código.

6.2.1 Caso de Estudio - Despliegue y puesta en marcha

En esta sección se relatarán los pasos que debe seguir un Coordinador del Máster para desplegar la aplicación web y su puesta en marcha.

1. En primer lugar el Coordinador en sus labores como administrador del sistema, deberá de desplegar la aplicación web en remoto, lo que podrá utilizar cualquier servicio que se preste y acepte las características de *Meteor*. En este proyecto se pondrá el ejemplo de *Modulus* y su servicio de Hosting.
2. Para desplegar la aplicación en remoto deberán seguirse los pasos descritos en el Anexo A.1
3. El Coordinador desplegará la aplicación por medio de la interfaz web de *Modulus* o por medio de comandos. Dicho comando será:

```
Modulus start
```

4. Una vez desplegada la aplicación en remoto y obtenido la dirección web de dicha aplicación, la introduciremos en el navegador y aparecerá la pantalla de inicio o de bienvenida (Ver Anexo A.2).
5. Una vez en la pantalla de bienvenida se podrá apreciar que la base de datos está totalmente vacía, por lo tanto no posee ningún usuario registrado en la base de datos. Para esta cuestión se provee a la aplicación de una ruta segura a modo de enlace que sólo conocerá el Coordinador y que le permitirá acceder a una vista protegida con un formulario para ingresar una contraseña.
6. Esta contraseña será la que proteja a esta sección de un ataque contra la integridad, por eso será introducida sólo una vez y pedida cada vez que entre a este módulo.
7. Una vez introducida dicha contraseña, podrá establecer e ingresar a los primeros usuarios del sistema a modo de texto plano, el cual se recogerá dentro de un cuadro de

texto. Para ello se acompaña de unas indicaciones para la inserción de usuarios y sus datos.

8. De esta forma lo primero que deberá hacer es crear su usuario con los permisos de Administrador, indicando en el campo *rol* el rol de *admin*. Por ejemplo:

   Miguel.Gonzalez3@alu.uclm.es,admin,LSI,Miguel,González Cano

   Donde LSI será el área del departamento al que el usuario pertenezca.

9. Una vez realizado y enviado dicha información al servidor, se enviará un correo con una dirección que deberá seguir para introducir su contraseña de usuario.
10. Si se desea también se pueden dar de alta a otros usuarios como los miembros de la CAM.
11. Hecho esto el usuario quedará validado en el sistema y podrá realizar las diversas funciones.

Cuando el usuario se encuentre validado en el sistema, tendrá acceso al panel de administración el cual le permitirá realizar la gestión de las peticiones que se vayan enviando.

Capítulo 7

Conclusiones y Trabajo futuro

En este capítulo se se realiza una valoración sobre los objetivos alcanzados en la realización del presente proyecto. La aplicación ha cumplido con los objetivos que se detallan a continuación. Además se realiza unas propuesta de trabajo futuro para la mejora y adecuación de la aplicación web.

7.1 Conclusiones

Para comentar las conclusiones, éstas se detallarán junto a los objetivos comentados en el capítulo 2, que a su vez están divididos en objetivos generales y específicos.

7.1.1 Objetivo general cumplido

Al final de todo el proyecto se ha diseñado, implementado y desplegado una aplicación web que permite la automatización y gestión de los proyectos finales de máster, acción que se llevaba a cabo utilizando programas y servicios de terceros. En concreto se han respondido a las siguientes cuestiones:

- El framework permite desarrollar y desplegar aplicaciones web en servidores remotos.
- Se ha creado una aplicación que sustituye al actual método de propuestas TFM.
- Se ha implementado una herramienta novedosa gracias a la ayuda de una nueva tecnología que posibilita la creación de contenido y sistemas web de fácil uso y manejo.
- Se ha utilizado con éxito el framework *Meteor*.
- Se ha puesto en práctica todo lo aprendido en la intensificación de Tecnologías de la Información.

Es importante recalcar la complejidad que ha alcanzado el proyecto de aplicación, ya que cuanto más crecía y se aprendía más acerca de este framework, más ideas se intentaban llevar a cabo y surgía problematicas de tipo funcional, ya sea en el código como en la intercomunicación de los diferentes paquetes y tecnologías usadas. El haber podido enfrentarse a esta complejidad y el haber acabado el proyecto con todo ello también es considerado por el autor como un importante logro cumplido.

7.1.2 Objtivos específicos cumplidos

Los objetivos específicos están divididos en funcionales (funcionalidades que se esperan obtener del proyecto) y didácticos (conocimientos que se esperan adquirir con el desarrollo del proyecto).

Objetivos funcionales cumplidos

- Se ha diseñado una arquitectura basada en iteraciones para el desarrollo de la aplicación.
- Se ha proporcionado un sistema fiable basado en autentificación y validación de usuarios.
- Se ha reutilizado código y paquetes externos para la creación de un sistema que proporcione las funcionalidades que se desean.
- Se ha creado un sistema web que es fácilmente escalable y modulable, que permite la adición de nuevos módulos, así como la fácil comprensión de los actuales, dando la facilidad de implementación por parte de terceros.
- Se crea un sistema de simple uso y fácilmente accesible desde todo tipo de dispositivos conectados a la red.
- Se han utilizado estándares, tecnologías y plataformas libres novedosas en el desarrollo de aplicaciones y sistemas web, dando lugar a una aplicación versátil y muy escalable.

Objetivos didácticos cumplidos

- Se ha aprendido durante el proyecto los lenguajes principales en tecnologías web, como son el lenguaje HTML, JavaScript y CSS.
- El autor ha aprendido a realizar aplicaciones y sistemas bajo el framework Meteor.
- El autor ha obtenido conocimientos de las tecnologías y sistemas web.
- Se han utilizado y probado multitud de paquetes externos proporcionados por los usuarios, los cuales están desarrollados en el mismo lenguaje que se ha utilizado en el proyecto, por lo que refuerza el primer punto.
- Se ha diseñado un sistema interoperable por varios lenguajes, los cuales afectan a distintos aspectos de la arquitectura. De este modo se ha aprendido a utilizar y manejar los lenguajes para su interoperabilidad.
- Se ha trabajado con éxito en un proyecto complejo y prolongado en el tiempo
- Se ha cumplido parcialmente la organización del trabajo inicial.

7.2 Trabajo Futuro

Para continuar con el desarrollo del proyecto a partir de lo presentado, se sugieren las siguientes líneas de trabajo:

- **Puesta en explotación y refinamiento del prototipo.** Debido a la naturaleza del proyecto y a la realización de diversos prototipos, la llegada al final del proyecto supone una aplicación en estado de pruebas o estado *beta*. De este modo una de las líneas a trabajar sería el refinamiento del prototipo en aspectos de diseño y funcionalidad, de modo que se aporten funciones de refuerzo y seguridad frente a errores no deseados. Otro línea importante en la que trabajar sería la explotación y puesta en marcha del sistema, lo que haría necesitaría de un despliegue en un servidor de uso propio o específico y la realización de pruebas de carga.
- **Integración de un Módulo de notificaciones *push* para móviles**. Dado que la aplicación web está diseñada para dispositivos móviles y la plataforma provee un sistema de generación de archivos *.apk* e *.ipa*, los cuales son compatibles con los sistemas operativos *Android* e *iOs* respectivame, se da la posibilidad de la futura creación de un módulo que gerere notificaciones para móviles como las de otras aplicaciones móviles.
- **Expansión del Módulo de PE**. Dado que ya existe en la actualidad un servicio que provee la funcionalidad de gestión de prácticas en empresas, este proyecto ha intentado completar o complementar dicho servicio con uno propio que provea información acerca de este asunto. Dicho módulo tiene mucho potencial de desarrollo futuro, y dado que no se ha comentado dentro de este proyecto, se puede complementar con nuevas funcionalidades como por ejemplo la inserción del rol de *tutor de prácticas en empresas*. Otro actor que formará parte del sistema y que servirá de enlace con la empresa para facilitar la gestión y elaboración de dichas prácticas durante su proceso. De este modo se proveerá a la aplicación de otro gestor al que delegar funcionalidades básicas pero necesarias.

ANEXOS

Anexo A

Manual de usuario de la aplicación web

A.1 Despliegue de la aplicación web

Como se ha comentado en el capitulo de Resultados (Capítulo 6.2.1), para realizar el despliegue de la aplicación debemos contar con un servidor que sea compatible con *Meteor* y por supuesto, tener instalado el framework en dicho servidor. Para esta sección y este propósito se utilizará un servicio externo que proporciona alojamiento de aplicaciones web en remoto. Dicho servicio es proporcionado por *Modulus.io* y es un servicio de pago, el cual proporciona una cuenta de prueba para desarrolladores que quieran probar su aplicación.

Por lo tanto, para desplegar la aplicación se necesitará una cuenta de usuario de *Modulus*. Una vez registrados en la web, aparecerá un panel de administración de aplicaciones el cual estará vacio. Los pasos a seguir son los siguientes:

1. En primer lugar se necesitará crear un proyecto nuevo. Para ello pulsamos sobre el botón *CREATE NEW PROYECT*.
2. Una vez en la pantalla de creación de nuevos proyectos, se seleccionará el entorno de desarrollo, en este caso *Meteor*, se indicará la cantidad de memoria que se desea por sirviente, que en aplicaciones de este tipo con 192Mb debería bastar, y el proveedor y la region de la base de datos, en este caso el más recomendable es *joyent - eu-ams-1*. En la imagen A.1 puede verse un ejemplo de los parámetros a introducir.
3. Creado el proyecto, el siguiente paso es crear la base de datos y enlazarla a dicho proyecto. Para ello, en la sección *DATABASES*, se pulsará sobre el botón *CREATE DATABASE*. Una vez hecho esto aparecerá un formulario para dar nombre a la base de datos, elegir el proveedor y la región (los mismos que en el punto anterior) y por último el usuario por defecto de la base de datos.
4. Una vez creada la base de datos, se pulsará sobre ella para acceder a su información. En el menú de *ADMINISTRATION* aparecerán los *Connection Strings*, que son cadenas de texto que ayudarán a enlazar al proyecto con la base de datos. Dichas cadenas se copiarán y guardarán aparte para utilizarlas en el punto siguiente.
5. Una vez anotadas las *Connection Strings*, habrá que dirigirs al proyecto que se ha creado en el punto 2 y dirigirse a la pestaña *ADMNISTRATION*. En dicho panel aparecerá

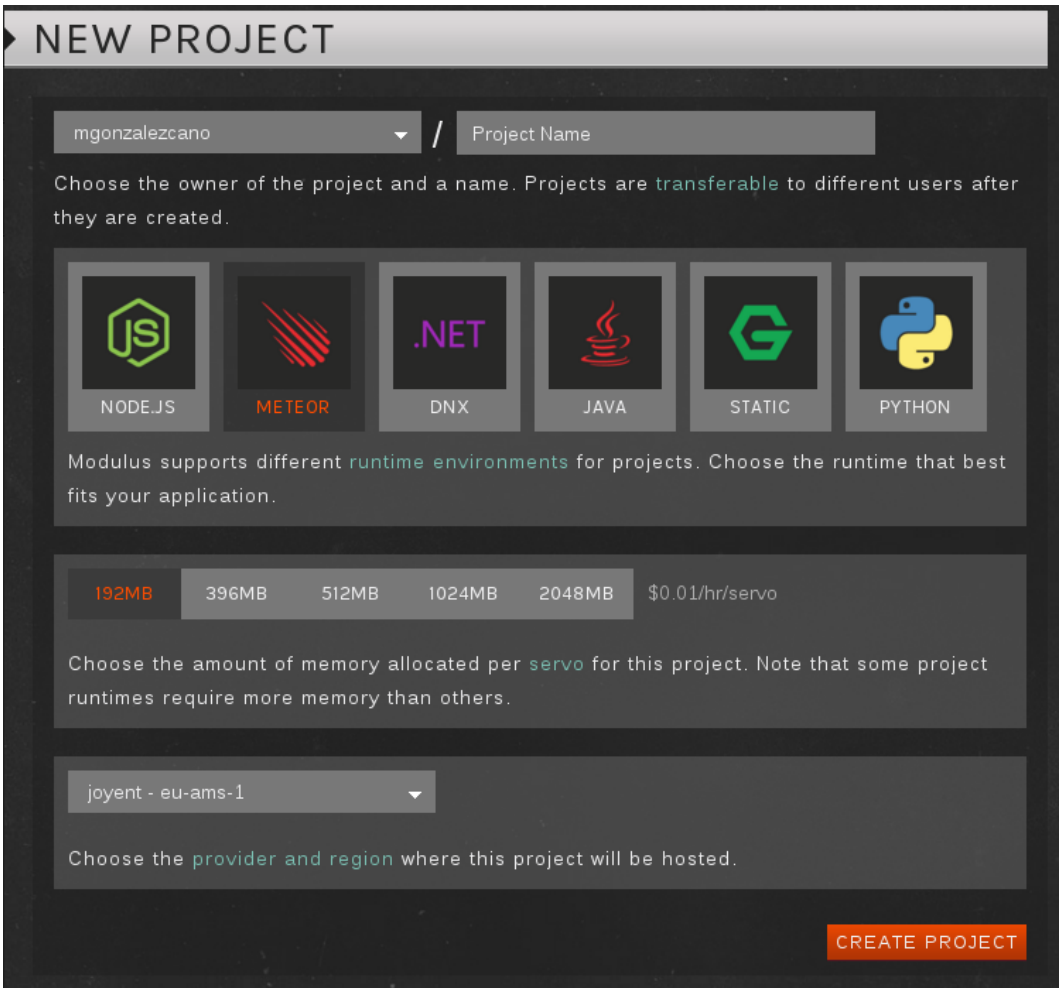

Figura A.1: Modulus.io - Creación de proyectos

la sección de *ENVIROMENT VARIABLES*, las cuales recoge las variales de entorno, importantes para enlazar todo el proyecto. Dichas variables son 2:

- *MONGO_URL*. Dicha url aparecerá en el panel de administración de la base de datos creada en el punto anterior.
- *ROOT_URL*. Esta url se proporciona una vez creado el proyecto. Es la url que la aplicación tendrá en remoto.

Un ejemplo de dichas variales puede verse en la imagen A.2. Acto seguido se pulsará sobre el botón *SAVE* para guardar los datos introducidos.

6. Una vez llegado a este paso, el último requisito es subir la información del proyecto al servidor, para ello habrá que dirigirse al nuevo proyecto creado y en la sección *DEPLOY FILES* que aparece en la parte superior derecha, pulsar sobre el botón *BROWSE*. Esto llevará a elegir la ruta del proyecto, el cual se subirá al servidor pulsando sobre el botón *UPLOAD*.
7. Por defecto el servidor arrancará solo y si no aparecen mensajes de error de ningún tipo y se generará una url con la ruta del proyecto a la cual acceder.

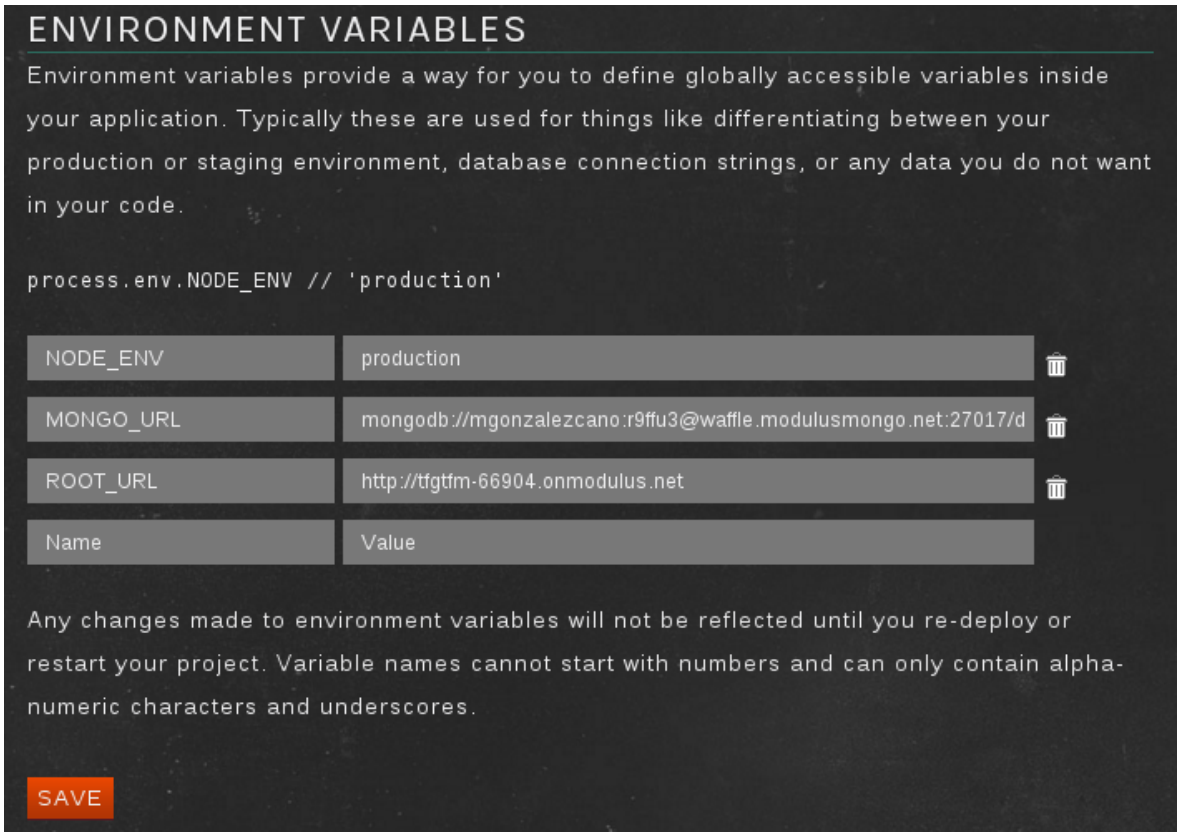

Figura A.2: Modulus.io - Variables de Entorno

Dicho proceso también se podrá realizar por medio de un terminal, pero la versión más intuitiva se encuentra en el uso de la herramienta web, como se ha comentado en los pasos anteriores.

A.2 Acceso a la aplicación

Para realizar el acceso a la aplicación web, se deberá acceder por medio de una *url* en la que esté alojado el sistema y activo. Para ello se deberá seguir el apartado anterior, el cual se generará una *url* válida a la que acceder, o entrar en la aplicación de prueba desplegada por el autor del proyecto cuya *url* es la siguiente:

```
tfgtfm-66904.onmodulus.net
```

Una vez accedido al sitio web, aparecerá la aplicación web cargada en el cliente lista para su uso.

A.3 Uso de la aplicación web

La aplicación web tiene diversos usos y sobretodo está dirigida a varios tipos de usuarios, es decir, que según sea el rol del usuario dentro del sistema (rol que debe corresponderse con el que ostenta en la universidad) tendrá unas funciones predefinidas u otras. Por lo tanto la

mejor forma de explicar el uso de la aplicación web es segmentarlo o estructurarlo según el punto de vista de cada tipo de usuario.

- **Uso de la aplicación para el Coordinador del Máster.** El Coordinador del Máster es quizá la figura principal e imprescindible del sistema, ya que todan las funciones giran en torno a este rol. El Coordinador hace las funciones de administrador del sistema, por lo que posee las funciones principales de gestión dentro de la aplicación web, además de tener la potestad de dirigir el sistema y validar todas las acciones que se lleven a cabo dentro de él.

  - **Validar a los nuevos usuarios registrados en el sistema.** Para ello deberá dirigirse a la barra de navegación y pulsar sobre el panel de administración. Una vez en el panel de administración, aparecerá el menú de gestión el cual deberá seleccionar la gestión de usuarios. Dentro del panel de gestión de usuarios, podrá validar a los usuarios que él crea conveniente pulsando sobre el voton *Validar* (imagen A.3).

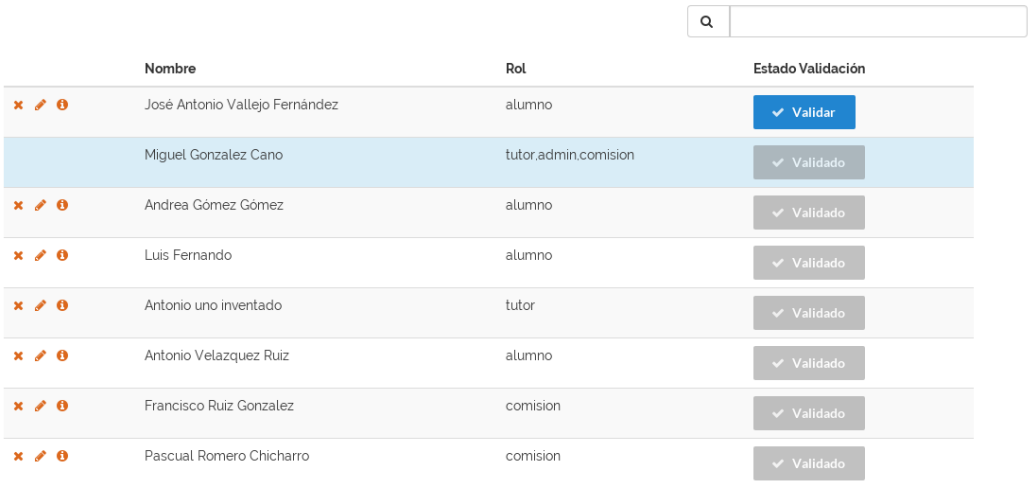

Figura A.3: Gestión de Usuarios

  - **Modificar información del usuario o eliminarlo del sistema.** Dentro del menú de gestión de usuarios aparecerán varias opciones a la izquierda de cada fila de la lista. Estas opciones darán la posibilidad de eliminar la usuario del sistema, modificar sus datos o roles y ver la información detallada.
  - **Verificar las nuevas propuesta de TFM y enviarlas a la comisión.** Una de las tareas importantes de la aplicación es la de verificar la nueva inserción de propuestas por parte de los tutores. Cuando se realize la inserción de un nuevo TFM en el sistema, el Coordinador recibirá una notificación avisándole de ello. El Coordinador deberá dirigirse al panel de administración, y pulsar sobre la gestión de TFM. Una vez en el panel de gestión de TFM aparecerá una lista de las

propuestas junto a una etiqueta que indicará su estado. El estado *propuesto* indica que se acaba de formalizar la inscripción y necesita de su acción para enviarlo a la comisión para su evaluación. Para ello deberá de pulsar sobre dicha propuesta de la lista y aparecerán las opciones de *Editar la Propuesta* o *Enviar Propuesta a la Cómisión*. La propuesta quedará en estado de *Revisión* una vez se haga efectivo el envío de dicha propuesta a los miembros de la CAM.

- **Validar Propuestas TFM.** Cuando la propuesta se ha enviado a la CAM, queda por esperar a la resolución de los miembros de la CAM, por lo que mientras tanto, se podrá hacer un seguimiento de la propuesta observando la actividad y el fallo de los miembros de la CAM. El Coordinador, al ser miembro también de la CAM tiene su voto, el cual se hará efectivo una vez decida si validar o no la propuesta, habiendo hecho uso del seguimiento y los reportes de validación de los Miembros de la CAM. Para validar la propuesta tan sólo habrá que pulsar sobre el botón que está marcado como *Validar Propuesta*.

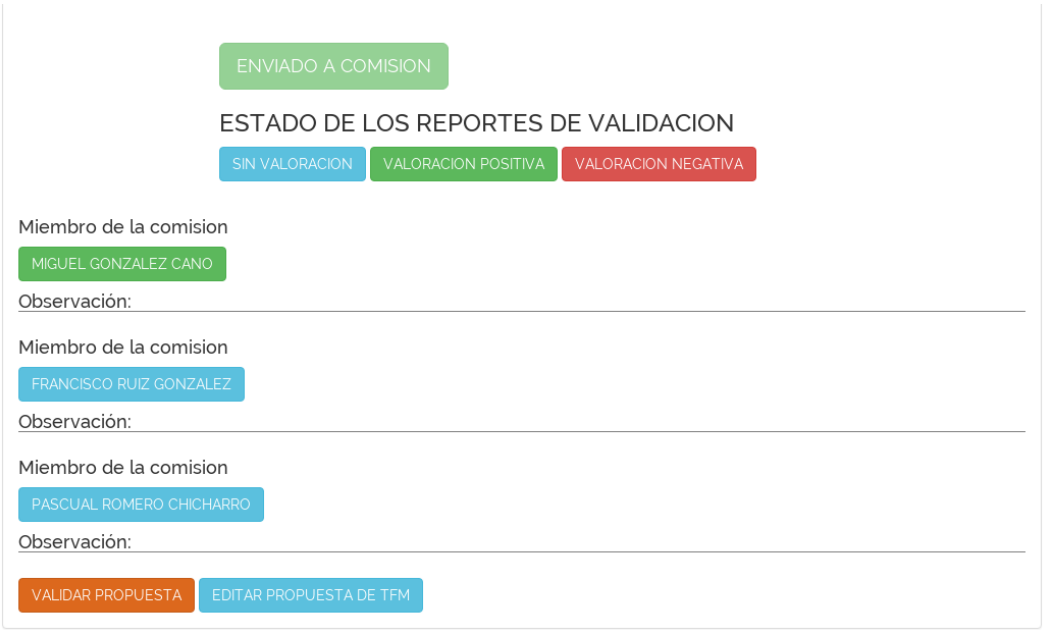

Figura A.4: Seguimiento de Propuestas TFM

- **Seguimiento de las Líneas de Trabajo de PE.** Si el Coordinador se dirige a la sección de Prácticas en Empresas, aparecerá una vista en la cual se pueden observar la cantidad de usuarios por línea de trabajo, es decir, a modo estadístico, qué lineas han elegido más los usuarios.
- **Modificación de las Líneas de Trabajo de PE.** El Coordinador podrá eliminar o añadir líneas de trabajo nuevas a la colección de modo que siempre estén actualizadas.

- **Uso de la aplicación para un Miembro de la CAM.** Todo Miembro de la Comisión Académica del Máster deberá estar registrado en el sistema para poder llevar a cabo su función. Dicha función es la de validar o denegar, desde su objetividad, las propuestas de TFM que lleguen al sistema. El Coordinador les irá enviando periódicamente propuesta que deberán validar. Dicha información le vendrá dada por medio de notificaciones que le avisarán de que hay propuestas que validar. Para realizar dicha tarea tan solo deberá de dirigirse al panel de *Vista de Propuestas* TFM (ver imagen A.5), allí aparecerán las propuestas marcadas por el estado y la fecha. Si el usuario despliega dicha propuesta en la lista podrá ver toda la información y realizará su tarea por medio de los botones de *Validar* o *Denegar* la propuesta. Dicha información se verá reflejada en la gestión de TFM del panel de administración del Coordinador.

Figura A.5: Validación de TFM por parte de la CAM

- **Uso de la aplicación para un Tutor del Máster.** Un tutor del máster tendrá como principal propósito el de inscribir propuestas de TFM en el sistema. Dichas propuestas podrán venir o no acompañadas de una asignación a un alumno, lo que no es requisito indispensable en un principio. Una vez inscritas, podrá seguir su resolución desde la *Vista de Trabajos Fin de Máster* y realizar las pertinentes acciones de asignación.
  - **Insertar Propuestas de TFM.** Para insertar nuevas propuestas, deberá dirigirse a la *Vista de Propuestas* TFM y pulsar sobre el botón *Inscribir nueva propuesta.* Una vez allí aparecerá un formulario que deberá de rellenar para formalizar la propuesta. Una vez rellenada la propuesta, deberá pulsar en *Enviar Propuesta* para realizar su envío a la base de datos y que el Coordinador pueda gestionarla.
  - **Realizar el seguimiento de la propuesta TFM.** Una vez haya formalizado al-

guna propuesta y ésta haya sido enviada al Coordinador, podrá realizar el seguimiento de dicha propuesta desde la *Vista de Propuestas* TFM. Desde este panel podrá asignar las propuestas a algún alumno que se encuentre registrado en el sistema, para ello bastará con pulsar en la propuesta, y acto seguido en el botón de *Asignar Propuesta a Alumno*. Aparecerá un desplegable con los alumnos que estén validados en el sistema.

- **Mantener Comunicación con el Alumno del TFM.** Si se ha realizado la asignación de la propuesta a algún alumno del sistema, se podrá realizar una comunicación a modo de mensajes cortos dentro de la aplicación. Para ello deberá dirigirse al menú superior, en la pestaña de su perfil, y pulsar sobre la opción *Mis Alumnos Tutorizados*. En este panel se registrarán todos los alumnos que el tutor tenga asignados a las propuestas de TFM. De este modo podrá comunicarse con todos ellos e informar a los alumnos sobre la propuesta.

- **Uso de la aplicación para un Alumno del Máster.** Todo alumno del Máster que vaya a realizar el TFM deberá registrarse en el sistema. Para ello una vez rellenado el formulario de registro, deberá esperar a que sea validado por el administrador. Un aspecto importante es el de tener actualizado el perfil de usuario, con la información requerida válida para poder ser aceptado en el sistema por el Coordinador. Una vez validado podrá ser elegido para la asignación a un TFM. Dicha asignación vendrá formalizada por su tutor del TFM. Formalizada la asignación del alumno, podrá comunicarse con el tutor mediante mensajería interna. Para ello deberá dirigirse al menú de *perfil de usuario* y pulsar sobre *Mi Tutor del TFM*. Desde este panel podrá enviar y leer los mensajes que se vayan enviando al tutor. Otra funcionalidad importante es la de la asignación de líneas de trabajo en las TFM. Para registrar las líneas en las que el alumno está interesado deberá dirigirse al panel de *Prácticas en Empresas* y seleccionar a modo de *checkboxes* las líneas de trabajo interesadas en el envío de ofertas de PE y tenerlas actualizadas según sus intereses.

Referencias

[AA15] G.G. Avilés y I.T.C. Academy. *Seguridad en Bases de Datos y Aplicaciones Web:*. CreateSpace Independent Publishing Platform, 2015. ISBN: 9781511544474.

[Ano16] Anonimo. Software para la gestión administrativa, 2013 (último acceso Agosto 2016). url: `https://sofwareparalagestionadministrativa.wikispaces.com/DEFINICION+DEL+SOFTWARE+PARA+LA+GESTION+ADMINISTRATIVA`.

[Bal16] Balu Baluart.net. Zend Framework desde cero, 2015 (último acceso Agosto 2016). url: `http://www.baluart.net/articulo/zend-framework-desde-cero`.

[Ber16] David Berube. ¿Qué es Meteor?, 2015 (último acceso Agosto 2016). url: `https://www.ibm.com/developerworks/ssa/library/wa-meteor/`.

[BS14] Juan Pedro Barba Soler. Diseño y Desarrollo Web. Análisis de Casos. 1, 2013-2014.

[Cha06] J. Chapman. *Web Design: A Complete Introduction*. John Wiley & Sons Australia, Limited, 2006. ISBN: 9780470012260.

[DeB16] Matt DeBergalis. Discover Meteor 1.0 Edition. Technical report, 2016.

[Egu15] Javier Eguiluz. *Desarrollo Web Ágil con Symfony2*. 2015.

[FA10] N.F. Ferrer y J.M. Alfonso. *Content Management for E-Learning*. Springer New York, 2010. ISBN: 9781441969590.

[FdC02] B.C. Falgueras y Universitat Oberta de Catalunya. *Ingeniería del software*, capítulo 1.2 Ciclo de Vida del Software. Biblioteca multimedia: Informática. Universitat Oberta de Catalunya, 2002. ISBN: 9788484297932.

[GC15] Mario González Cifuentes. Despliegue de una infraestructura Cloud basada en XenServer para Bilib. 1, 2015.

[GHJV94] E. Gamma, R. Helm, R. Johnson, y J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994. ISBN: 9780321700698.

[GM15] Marjorie Aydeé Ganchala Mora. Desarrollo de una aplicación Web empresarial de ayuda a la producción de campañas publicitarias. 1, 2015.

[HA15] M.T. Hernández y I.T.C. Academy. *Symfony Framework. Desarrollo Rápido de Aplicaciones Web:*. CreateSpace Independent Publishing Platform, 2015. ISBN: 9781511755078.

[Her14] J. Hernández. *Análisis y Desarrollo Web:*, capítulo 1.1.2 Tipos de Metodologías - Metodologías Ágiles. Jesús Hernández, 2014.

[Im116] Imagen del Xerox Alto, 2010 (último acceso Agosto 2016). url: `http://toastytech.com/guis/alto3.html`.

[Im216] Imagen de la Evolución de la Web, 2007 (último acceso Agosto 2016). url: `www.radarnetworks.com`.

[Im316] Imagen de Windows 1.0, 2015 (último acceso Agosto 2016). url: `http://www.muycomputer.com/2015/11/20/windows-1-0-cumple-hoy-30-anos`.

[Im416] Imagen de Windows 95, 2015 (último acceso Agosto 2016). url: `http://www.pcworld.com/article/2975025/windows/20-years-later-windows-10-follows-in-windows-95-s-footsteps.html`.

[Im516] Imagen de comparativa de Meteor con otros frameworks, 2015 (último acceso Agosto 2016). url: `https://medium.com/all-about-meteorjs/angular-meteor-architecture-explained-782f5bbeb0ff`.

[LA09] Ana Milagro Luzardo Alliey. *Diseño de la interfaz gráfica web en función de los dispositivos móviles*, volume 1, capítulo Elementos de las Interfaces Gráficas de Usuario. 2009.

[Luj16] S. Luján. ¿Qué es el desarrollo web?, 2014 (último acceso Agosto 2016). url: `http://idesweb.es/temario/que-es-el-desarrollo-web`.

[ME06] Carlos Marrero Expósito. *Interfaz Gráfica de Usuario. Aproximación semiótica y cognitiva*, volume 1, capítulo Interfaz gráfica de usuario: definición y problemáticas. 2006.

[MM03] Pedro Juan Molina Moreno. Especificación de Interfaz de Usuario. De los requisitos a la generación automática. 1, 2003.

[Ray05] Eric S. Raymond. *The first GUIs*, volume 1, capítulo 2. History: A Brief History of User Interfaces. 2005.

[Sim86] H. Simpson. *Programming the IBM PC User Interface*. Byte books. McGraw-Hill, 1986.

[Sos10] B. Sosinsky. *Cloud Computing Bible*. Bible. Wiley, 2010. ISBN: 9781118023990.

[SPC+16] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, y N. Diakopoulos. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Pearson Education, 2016. ISBN: 9780134380728.

[Spi16] Spine. Spine Web Documentation, 2015 (último acceso Agosto 2016). url: `http://spinejs.com/docs/`.

[Str12] I. Strack. *Getting Started with Meteor JavaScript Framework*. Community experience distilled. Packt Publishing, Limited, 2012. ISBN: 9781782160830.

[VM01] J. Veen y M.M. Moyano. *Arte y ciencia del diseño Web*. Pearson educación. Pearson Educación, 2001. ISBN: 9788420531571.

[W3C16] W3C. Mobile Web Best Practices, 2008 (último acceso Agosto 2016). url: `http://www.w3.org/TR/mobile-bp/#OneWeb`.

[Web16] Software para la gestion administrativa, 2014 (último acceso Agosto 2016). url: `https://sofwareparalagestionadministrativa.wikispaces.com`.

[Wel16] Gravity Well. Imagen de Arquitectura Meteor, 2015 (último acceso Agosto 2016). url: `https://www.gravitywell.co.uk/latest/how-to/posts/meteor-bring-the-awesome-part-0/`.

Este documento fue editado y tipografiado con LaTeX empleando
la clase **esi-tfg** (versión 0.20160828) que se puede encontrar en:
`https://bitbucket.org/arco_group/esi-tfg`