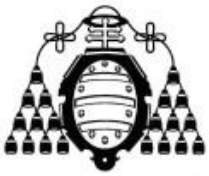
UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

"Plataforma Web para la coordinación de trabajos académicos"

DIRECTOR: Vicente García Díaz

AUTOR: Javier Urones Monteserín

Vº Bº del Director del Proyecto

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Agradecimientos

Esta sección no es en absoluto obligatoria, pero es el lugar correcto para dedicar el proyecto a las personas/instituciones/empresas/… que se desee. Si no se va a realizar, eliminarla.

Resumen

Un texto breve (una cara aproximadamente) que describa qué se ha hecho en el proyecto, sus principales objetivos, la utilidad que se le quiere dar, si está destinado a algún cliente real, aspectos sobre la tecnología usada y cosas similares que permitan hacerse una idea rápida del trabajo realizado.

Se trata de describir brevemente todos los aspectos más importantes del proyecto destacando en lo posible sus puntos fuertes para permitir comprenderlo fácilmente en una lectura rápida sin tener más referencias del mismo. Por tanto, no debe ser un texto demasiado largo ni complejo.

Palabras Clave

Palabra1, Palabra2, Palabra3,…

De 5 a 7 palabras[1] clave que mencionen conceptos de capital importancia en el proyecto: Cosas que el proyecto manipula (Ej.: "Máquinas Expendedoras", "Automóviles"), tecnologías usadas (Ej.: J2EE, RMI), utilidad del proyecto (Ej.: Gestión de Existencias), temática (Ej.: Historia Medieval, Aeronáutica) y cosas similares.

Si finalmente saliesen demasiados términos, conviene hacer una selección de los más relevantes para quedarse con el número indicado.

[1] No necesariamente debe ser una única palabra, pueden ser varias. Por ejemplo, si el proyecto tratase sobre la gestión de calificaciones de Alumnos, una palabra clave válida podría ser "Expediente Académico".

Abstract

Traducción al inglés del resumen anterior. Conviene hacerlo una vez se tenga la versión definitiva de dicho resumen. Se recomienda consultar al director del proyecto acerca de si considera adecuado que aparezca esta sección.

Keywords

Ídem a la sección anterior. Se recomienda consultar al director del proyecto acerca de si considera adecuado que aparezca esta sección.

Índice General

NOTA: No debemos olvidarnos de generarlo cuando se cierre la documentación, puesto que en caso contrario pueden quedar referencias mal actualizadas o texto erróneo (tanto éste como el de figuras). **BORRAR ESTA NOTA EN LA DOCUMENTACIÓN FINAL**.

Índice de Figuras

Capítulo 1. Memoria del Proyecto

El concepto de memoria de un proyecto es, en esencia, un resumen del proyecto para personas que desconozcan o no posean conocimientos avanzados de la naturaleza del proyecto y/o sus tecnologías, o incluso no posean conocimientos específicos de informática. Por tanto, debemos orientarla de manera que cualquier persona pueda entender que se ha hecho durante todo el proyecto.

Los puntos obligatorios de la memoria varían mucho de unos proyectos a otros, pero en este documento se proponen unos mínimos. En muchos casos, la memoria tiene un apartado por cada parte importante del proyecto, por ejemplo (Introducción, Requerimientos, Análisis y Diseño, Presupuesto, etc.) y en cada apartado se resume (para el perfil de lector mencionado anteriormente) el contenido del apartado técnico correspondiente. En cualquier caso, podemos orientar la memoria de la siguiente forma:

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

1.2 Resumen de Todos los Aspectos

Resumen de todos los aspectos del proyecto comentados en las secciones posteriores, tal y como se dijo anteriormente.

1.3 Otros Apartados

Otros apartados que el alumno o el director del proyecto consideren relevantes.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

No es una novedad que la realización de trabajos pertenecientes al ámbito educativo cuya extensión e importancia sean significativas se llevan ya tiempo coordinando de manera telemática. El uso de la informática simplifica y optimiza en gran medida la realización de estas tareas de coordinación ya que pone a disposición muchas funciones tecnológicas ventajosas: almacenamiento de archivos compartidos, edición de archivos compartidos de manera paralela, comunicación de manera rápida, seguimiento de tareas realizadas y una larga lista de funcionalidades más. Para suplir estas necesidades tecnológicas existen infinidad de herramientas y aplicaciones sin embargo, en ocasiones, la utilización de tantas herramientas distintas para gestionar un proyecto académico genera problemas a la hora de llevar un seguimiento de manera óptima.

Este proyecto nace con el objetivo de unificar en una misma plataforma las necesidades de coordinación de proyectos de carácter académico donde se necesite una comunicación y un feedback entre profesor y alumno. Dicha plataforma debe satisfacer las necesidades de comunicación, seguimiento y gestión documental que un proyecto de carácter académico pueda necesitar.

~~2.1~~ Esta plataforma debe proveer de todos y cada uno de los servicios que las aplicaciones de comunicación, alojamiento de archivos, seguimiento de tareas y cualquier otro tipo de aplicación que sea utilizada para coordinar un proyecto académico. Todos estos servicios deben de ser ofrecidos de manera conjunta y con independencia de otras tecnologías tanto de comunicación, mensajería, alojamiento en la nube etc..

Con formato: Normal

En esta sección se debe describir el motivo por el que se desarrolla el proyecto. Se trataría de describir brevemente, en lenguaje no técnico, de que va a consistir el proyecto y el motivo de su desarrollo, incluyendo la descripción de las necesidades que va a cubrir. Se necesita hacer entonces un resumen que describa las aplicaciones del proyecto y que justifique su creación, procurando compararlo favorablemente con otras posibles soluciones.

Figura 1.1. Ejemplo

La imagen de más arriba muestra un ejemplo de cómo se pueden colocar las imágenes en todo el proyecto, incluyendo el estilo de su descripción (que está vinculado con la lista de figuras de más arriba) y la forma de numerar las imágenes propuesta (nº de capítulo – nº de imagen dentro del capítulo).

2.2 Objetivos del Proyecto

Dentro de esta sección deben describirse (por ejemplo en forma de lista numerada) todos los objetivos que se quieren alcanzar dentro del proyecto (lo que se quiere lograr). No se debe entrar en excesivos detalles técnicos, sólo describir claramente todo aquello qué se quiere lograr con el proyecto, describiendo cada uno de estos objetivos de forma lo más completa posible para que queden lo suficientemente claros.

2.3 Estudio de la Situación Actual

Hoy en día es una realidad que existen multitud de opciones tecnológicas distintas para desarrollar y gestionar proyectos de todo tipo de manera online. Normalmente las tecnologías empleadas vienen dadas por las necesidades del proyecto, la experiencia y la afinidad del usuario/equipo con una tecnología. Los motivos de la utilización de una tecnología u otra pueden variar dependiendo del ámbito en el que se desarrolle el proyecto, ya sea ámbito profesional, educativo o personal. En esta sección nos centraremos exclusivamente en describir algunas de las tecnologías empleadas en proyectos de ámbito educativo y más concretamente en las utilizadas para la coordinación de proyectos relacionados con trabajos fin de grado, trabajos fin de máster y tesis doctorales.

La realidad es que actualmente existen multitud de herramientas que pueden servir para lo que englobaría coordinar el tipo de trabajos educativos que se han comentado anteriormente. Sin embargo, no existen muchas herramientas que nos permitan hacer todo lo que engloba la coordinación de estos trabajos académicos sin tener que recurrir a otras complementarias. Es decir, por lo general durante la realización de un proyecto académico se recurren a varias herramientas distintas, por ejemplo, se utiliza un servicio de almacenamiento en la nube para compartir documentos y recursos, se utiliza un servicio de correo electrónico para comunicarse entre los distintos miembros del proyecto y los tutores y además se suele utilizar una herramienta de comunicación que permita realizar reuniones online periódicamente. Además de estos tres ejemplos de herramientas, dependiendo de las características del proyecto podrían incluso ser más.

En los siguientes apartados procederá a realizar una breve descripción de alguna herramienta similar a la plataforma que se va a crear con el objetivo de agrupar, descubrir y clasificar el conjunto de funcionalidades a implementar.

2.3.1 Proyectos estudiados

En los siguientes subapartados se va a realizar una revisión de algunas plataformas cuyas funcionalidades serán inspiradoras para el proyecto que se va a llevar a cabo, con el objetivo de identificar posibles funcionalidades interesantes y descartar otras que lo sean menos.

Con formato: Normal

Con formato: Fuente: +Cuerpo (Calibri)

2.3.1.1 Google Wave

Esta herramienta estaba diseñada con el objetivo de congregar varias de las funcionalidades de las descritas en apartados anteriores. Permitía en una misma plataforma el empleo de correo electrónico, mensajería instantánea, almacenamiento de archivos, edición en tiempo real de documentos, el uso de hilos de conversación y la posibilidad de tener una wiki, entre algunas de sus funciones.

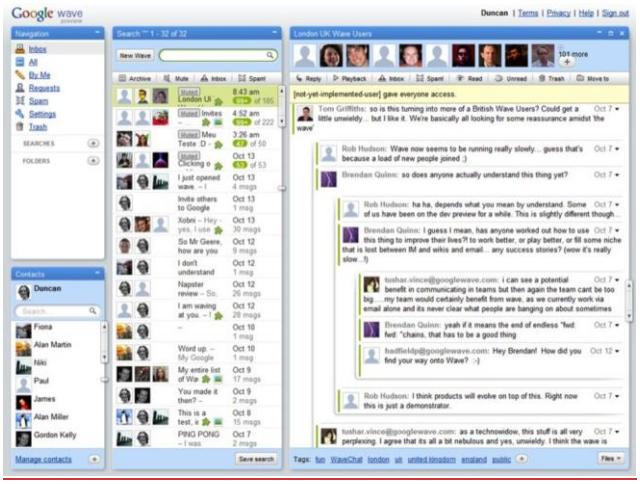


Figura 2.1. Google Wave

Con formato: Descripción

Realmente esta herramienta no encajaría en un estudio de la situación actual ya que en el año 2010 dejó de trabajar en su desarrollo y mantenimiento y dos años después fue completamente eliminada. La razón por la que se incluye en esta memoria es porque realmente desde entonces no existe una aplicación similar en el mercado y en el pasado era utilizada para la coordinación de proyectos académicos. También debe aparecer aquí porque a diferencia de las siguientes alternativas estudiadas, Google Wave debido a la época en la que fue desarrollada y utilizada, no pertenecía a un gran ecosistema de aplicaciones cuyas funcionalidades estaban sincronizadas entre sí, es decir, todas sus funcionalidades eran de manera nativa y funcionaban sobre la propia herramienta que es lo que se busca en este proyecto, es decir, una herramienta sencilla que permita tener todo lo referente a un proyecto académico reunido en un mismo lugar y que pueda ser fácilmente accesible tanto para el alumno como para el profesor.

2.3.1.2 Microsoft Teams

Es una plataforma de colaboración y comunicación que permite como función principal la realización de reuniones, de llamadas y la gestión y almacenamiento de archivos. En este apartado se hace referencia exclusivamente a Microsoft Teams porque es lo más conocido por los usuarios, sin embargo, lo más interesante de esta herramienta es que permite crear un espacio de trabajo compartido y acceder a varios servicios de la plataforma Microsoft 365.

Esta opción es muy interesante, como ya se ha comentado, debido a la integración y sincronización con otros servicios de la plataforma de Microsoft. Permite entre muchas cosas sincronizar los archivos con el sistema de almacenamiento en la nube (OneDrive),

sincronización con el servicio de correo electrónico (Outlook) y así con muchas de las herramientas de Microsoft (OneNote, Forms, PowerBI…) que van sumando cada una de sus funcionalidades a las posibilidades de Teams. Esta herramienta está dentro de un gran ecosistema y de ahí que sea tan utilizada para coordinar grupos de trabajo y proyectos de todo tipo tanto en el ámbito empresarial como educativo.

Por las limitaciones del alcance de este proyecto no se puede crear un ecosistema de aplicaciones que funcionen coordinadas y sincronizadas entre sí a esta escala. Sin embargo se intentará que la plataforma creada si incluya algunas de las distintas funcionalidades que ofrecen por separado las herramientas de Microsoft 365 en una sola herramienta.

2.3.1.3 Google Workspace

Esta plataforma es similar a Microsoft 365 pero ofrecida por Google, permite trabajar de manera colaborativa empleando las aplicaciones web de la plataforma Google (Gmail, Drive, Docs, Sheets…) que incluyen tanto aplicaciones de ofimática como de comunicación y gestión de tareas. Además también permite la instalación de nuevas aplicaciones tanto gratuitas como de pago para cubrir las necesidades de la organización o equipo de trabajo.

Al igual que Microsoft 365 está basado en la nube y proporciona a los usuarios las ventajas que esto implica como el acceso remoto desde cualquier lugar y dispositivo y las facilidades para el trabajo colaborativo y la comunicación entre otras.

Al igual que en el caso anterior, no podemos pretender en este proyecto incluir todas y cada una de las funcionalidades que Google Workspace ofrece en una sola aplicación, si no que se buscará crear una plataforma que permita realizar algunas de las funciones que ofrecen las distintas herramientas de Google Workspace en una sola.

Con formato: Normal

~~2.2.1~~

Con formato: Fuente: +Cuerpo (Calibri)

En esta sección deben identificarse y describirse sistemas similares al que se va a desarrollar, estableciendo una comparación entre lo que ofrecen estos sistemas y lo que pretendemos lograr con el proyecto, para de esta forma diferenciar nuestro desarrollo de lo ya existente.

No tienen por qué ser sistemas que hagan lo mismo que el nuestro, sino que pueden ser sistemas que contengan funcionalidad en común con una parte significativa o bien que estén orientados a un conjunto de potenciales usuarios similar.

En esta sección también es adecuado evaluar las posibles herramientas o lenguajes de programación utilizables para el proyecto y determinar cuál (o cuales) se adaptan mejor a nuestras necesidades concretas (de forma justificada).

Conviene en general destacar los puntos en común y las principales discrepancias entre estos sistemas y el nuestro, con la idea de ver en qué sentido nuestro desarrollo supone una ganancia o mejora sobre ellos (también puede orientarse a resolver ciertos defectos de los mismos, mejorar algunas funciones para hacerla más completa, rápida o fácil de usar, etc.). Si

los sistemas carecen de alguna funcionalidad que el nuestro va a incorporar, conviene también destacarlo (precisamente esto puede ser una de las principales aportaciones del mismo).

En general, conviene usar esta sección como un primer paso para "promocionar" las "bondades" de nuestro proyecto.

2.3.2 Evaluación de Alternativas

~~2.2.2~~

Con formato: Normal

En esta sección se describirán, una por una, todas las alternativas estudiadas. Conviene estudiar 3 o 4 alternativas importantes, salvo que por algún motivo justificado se deba incluir un número menor o mayor de las mismas. En todo caso, siempre es conveniente cuidar de que en esta sección haya un conjunto de sistemas significativo. En función de lo dicho anteriormente, cada sistema podrá dividirse en tres secciones: "Descripción", "Ventajas" e "Inconvenientes", aunque es posible cualquier otra división que contenga los aspectos descritos, dependiendo de qué tipo de sistemas se estudien.

La naturaleza de este proyecto hace que no exista ninguna limitación a la hora de desarrollarlo, la implementación del mismo puede llevarse a cabo en multitud de lenguajes, tecnologías y frameworks distintos, por lo que a continuación se evaluarán algunas alternativas que se han estudiado teniendo en cuenta conocimientos y experiencia previa del desarrollador.

2.3.2.1 ~~Sistema 1~~FrontEnd

Existen multitud de frameworks distintos para desarrollar aplicaciones web. En lo que respecta al lado del cliente hoy en día los siguientes son los más utilizados, todos utilizan JavaScript como lenguaje de programación o algún derivado de este como TypeScript.

Con formato: Normal

2.3.2.1.1 Angular

Es una framework desarrollado y mantenido por Google y permite crear SPAs (Simple Page Applications). Utiliza como lenguaje de programación TypeScript que es un lenguaje derivado de JavaScript.

2.3.2.1.1.1 Ventajas

Es un framework mantenido por una empresa altamente reconocida por ello existe mucha documentación y también muchos proyectos realizados en él. También hay multitud de tutoriales y referencias que pueden ayudar a la hora de desarrollar.

En términos más técnicos, utiliza componentes que son reutilizables lo que permite mantener de manera sencilla la aplicación y reducir el código escrito. Por otro lado, el renderizado es muy rápido ya que permite que el usuario solo cargue el código que necesita en ese momento para cargar la vista. Como última ventaja "reseñable" estaría la utilización de la herramienta de línea de comandos "Angular CLI" que permite generar estructuras de código de manera automática.

Con formato: Normal

Por último y más importante es un framework con el que he trabajado en el pasado en proyectos reales y tengo experiencia desarrollando con él por lo que la curva de aprendizaje sería mínima.

2.3.2.1.1.2 Inconvenientes

Los proyectos desarrollados en Angular suelen tener gran tamaño debido a los paquetes de JavaScript del CLI de Angular. Es un poco más difícil de aprender que otros frameworks, pero en este caso no lo es debido a que poseo experiencia con él. Por último, existe la posibilidad de confundir Angular con AngularJS, que aunque este último también era un framework para desarrollar aplicaciones web, su estructura no se parece. Esto puede generar confusión a la hora de buscar documentación o referencias.

2.3.2.1.2 React

Es una librería de JavaScript desarrollada y mantenida por Facebook y al igual que Angular permite desarrollar SPAs.

2.3.2.1.2.1 Ventajas

Es minimalista y no tiene funciones ni plantillas complicadas, la curva de aprendizaje es baja debido a que si se conoce JavaScript es fácil de aprender. La documentación y recursos para aprender a manejar React son muy amplios debido a que tiene una de las comunidades de desarrolladores más grandes de todo el mundo. En términos de rendimiento es superior a Angular debido a que los árboles DOM que genera son más ligeros y reduce la carga del navegador.

2.3.2.1.2.2 Inconvenientes

Utiliza la librería Redux que es fundamental aprender a utilizarla para poder desarrollar en React y esto implica tiempo de aprendizaje. También hay un conjunto de prácticas deseables que son necesario aprender para escribir código de manera limpia. Finalmente, es complicado aprender a configurar proyectos debido a que no existe un proyecto predefinido.

2.3.2.1.3 Vue.js

Es un framework de JavaScript que, al igual que los dos anteriores, también permite desarrollar SPAs. Fue desarrollado y es mantenido por Evan You y por desarrolladores de las empresas Netlify y Netguru.

2.3.2.1.3.1 Ventajas

Es muy ligero ya que pesa únicamente 18 KB. Es fácil de aprender y utilizar si ya se tienen conocimientos previos de JavaScript y además permite dividir la aplicación en componentes lo que implica poder probarlos de manera aislada.

~~*2.2.2.1*~~*2.3.2.1.3.2 Inconvenientes*

Existe documentación pero no tanta como para los otros dos frameworks anteriores, además mucha de esta documentación no está en inglés ni español por lo que es menos accesible. Al no estar mantenido por una gran empresa no hay tantos proyectos importantes y además es un framework relativamente nuevo.

Con formato: Normal

Con formato: Normal

Con formato: Normal

2.3.2.2 ~~Sistema 2~~BackEnd

2.3.2.2.1 .NET Core

Este framework es una evolución del pasado .Net Framework y además es una de las apuestas más recientes de Microsoft en lo que respecta a software libre. Es un ecosistema de desarrollo que permite el desarrollo de aplicaciones de consola, librerías, aplicaciones web, aplicaciones de escritorio y otras muchas más

2.3.2.2.1.1 Ventajas

Con formato: Título 6

Este framework ofrece por una parte la ventaja del lenguaje de programación C#, uno de los lenguajes más utilizados del mundo y con unas características que lo hacen muy interesante. Por otro lado, este framework es multiplataforma y puede funcionar tanto sobre Windows como Linux, macOS o dispositivos móviles. Otra de las ventajas es la gran cantidad de librerías y paquetes que existen dentro de este framework que resuelven multitud de problemas.

2.3.2.2.1.2 Desventajas

La principal desventaja de este framework es el tiempo de aprendizaje de las tecnologías que son englobadas por la plataforma. Esto hace referencia, por ejemplo a Entity Framework que es la tecnología por excelencia dentro de .NET que se encarga del mapeo objeto-relacional o a Microsoft SQL Server/SQL Server Express como sistemas de gestión de bases de datos. Estas tecnologías son muy potentes pero a la vez tienen muchas características que harían que la curva de aprendizaje se disparase.

Con formato: Normal, Sangría: Izquierda: 0 cm, Primera línea: 0 cm

2.3.2.2.2 Node.JS/Express

Node.js es un entorno de ejecución de Javascript que está orientado a eventos asíncronos y orientado a la capa de servidor. Por otro lado, Express.js es un framework para Node.Js que ofrece la posibilidad de utilizar javascript como lenguaje para desarrollar en el servidor y está orientado al desarrollo de aplicaciones web y APIs.

2.3.2.2.2.1 Ventajas

La primera y más importante de las ventajas es que la curva de aprendizaje sería mucho más reducida en comparación con otras alternativas ya que tengo experiencia en el desarrollo de aplicaciones con esta tecnología. Por otro lado, a nivel más teórico esta tecnología permitiría desarrollar en el mismo lenguaje tanto el frontend como el backend (javascript). Por otro lado, el sistema de paquete de Node (npm) permite la inclusión de módulos que facilitan muchas tareas. Como última ventaja esta tecnología permite desarrollar de manera sencilla y rápida ya que en muy pocas líneas de código te permite tener una API funcional.

2.3.2.2.2.2 Desventajas

El principal inconveniente de esta tecnología es el manejo asíncrono de los eventos, ya que a veces puede hacer que cueste encontrar dónde está el error a la hora de desarrollar. Tampoco es adecuado para aplicaciones muy grandes y no está optimizado para la programación multihilo.

Con formato: Normal

~~2.2.2.2~~2.3.2.2.3 PHP

PHP es uno de los lenguajes más famosos para desarrollar en el servidor. Este lenguaje es de código abierto y está orientado al desarrollo web. Además existen frameworks para desarrollar con este lenguaje orientados al desarrollo de aplicaciones web MVC como lo son Symfony y Laravel.

2.3.2.2.3.1 Ventajas

Es un lenguaje ampliamente utilizado con lo que ello significa: multitud de documentación, problemas comunes resueltos, cursos, tutoriales etc. Esto implicaría facilidad a la hora de desarrollar funciones básicas con este lenguaje. Las ventajas de este lenguaje a nivel más técnico podrían ser su eficiencia, su alta compatibilidad con sistemas de gestión de bases de datos y la rapidez a la hora de desarrollar.

2.3.2.2.3.2 Desventajas

La principal desventaja es la curva de aprendizaje y el tiempo invertido para aprender las suficientes características del lenguaje como para llevar a cabo un proyecto. Si a esto sumamos el aprendizaje de uno de sus frameworks para desarrollo de aplicaciones web, el tiempo empleado se dispara.

2.3.2.3 Bases de datos

Para estudiar el sistema de gestión de base de datos a emplear se va a considerar la división de las mismas entre relacionales y no relacionales.

Con formato: Normal

2.3.2.3.1 Relacionales (SQL)

Es el modelo de base de datos clásico y tradicional, se basa en tablas (que contienen columnas y registros) y relaciones entre estas utilizando claves primarias y claves foráneas. Para hacer consultas a las tablas se emplea el lenguaje SQL (Structured Query Language). Algunos de los sistemas de gestión de bases de datos relacionales más famosos son: MySql, MariaDB, SQLite, Microsoft SQL Server, Oracle y PostgreSQL.

2.3.2.3.1.1 Ventajas

Este tipo de bases de datos llevan muchos años existiendo y durante mucho tiempo fueron la única alternativa, por lo que existe mucha información, documentación (que varía dependiendo del SGBD utilizado) y proyectos que utilizan bases de datos SQL. La curva de aprendizaje sería baja ya que se ha trabajado previamente con este lenguaje, realizando consultas y procedimientos almacenados. Son fiables y mantienen la integridad referencial evitando registros repetidos.

2.3.2.3.1.2 Desventajas

Su rendimiento no es tan alto como el que se puede obtener con las bases de datos NoSql ya que precisan de más recursos para realizar la consulta de sus datos y para poder mantenerse. La complejidad del modelo de datos es mayor e implica más tiempo empleado en el diseño del mismo (relaciones, tablas, etc.).

Con formato: Normal

2.3.2.3.2 No Relacionales (NoSQL)

Las bases de datos no relacionales o NoSQL son la alternativa a las anteriormente estudiadas, como principal característica en contraposición destaca la no utilización de tablas. Como sistemas de gestión de bases de datos destaca Neo4J (orientada a grafos) y MongoDB (orientada a documentos).

2.3.2.3.2.1 Ventajas

A nivel de rendimiento es superior que las bases de datos relacionales ya que precisan de menos recursos para funcionar. Por otro lado, la escalabilidad es sencilla y muy barata de llevar a cabo y además permite trabajar con datos no necesariamente estructurados lo que facilita mucho el desarrollo al no tener un modelo de datos fijo con tablas y estructuras relacionadas. También en lo que respecta al desarrollo no se necesita utilizar un mapeador objeto-relacional que convierta en objetos los registros de las tablas por lo que se disminuye el tiempo de desarrollo.

2.3.2.3.2.2 Desventajas

No son óptimas para trabajar con cantidades de datos muy grandes. No llevan tanto tiempo existiendo por lo que no hay una comunidad tan grande detrás en comparación a las bases de datos relacionales. Pueden no ser tan sencillas de comprender en comparación con los bases de datos SQL cuya estructura está bien definida siempre.

Con formato: Título 6

Con formato: Justificado

Con formato: Fuente: 11 pto, Color de fuente: Automático

~~Capítulo 3. Aspectos Teóricos~~

Esta sección está destinada a describir brevemente aquellos conceptos, herramientas y tecnologías existentes que vamos a usar en nuestro proyecto. Conviene limitarse a un máximo de 2 hojas por cada uno de ellos aproximadamente (es una descripción, no un tutorial sobre ello), describiendo por qué y para qué usamos esto en nuestro proyecto pero sin hacer descripciones muy grandes de características y similares (para ello podemos referenciar enlaces Web, libros, etc. que las describan más detalladamente, que posteriormente podemos incluir en la bibliografía):

- En el caso de conceptos sobre los que trata el proyecto, debemos decir su origen, aplicación e importancia, para tener una idea de lo que se va a tratar. A modo de ejemplo, si usamos bases de datos orientadas a objetos debemos decir que son, que importancia tienen y describir brevemente su forma de trabajo. Si nuestro proyecto es sobre emisión de video en directo, debemos describir la técnica usada para ello y como funciona, etc.
- En el caso de tecnologías debemos decir su creador u origen, para lo que sirve y que aplicaciones ha tenido, para qué la vamos a usar, la versión que hemos empleado, etc.
- En el caso de herramientas debemos decir su fabricante y la utilidad que le vamos a dar dentro de nuestro proyecto.
- Dentro de este apartado también podemos incluir las notaciones empleadas para representar los diagramas del proyecto (como UML) o la metodología usada para el desarrollo de la documentación (como Métrica 3). Este documento está desarrollado usando la metodología Métrica 3, pero de forma adaptada, contemplando solo aquellos apartados que se han considerado más adecuados para un proyecto de fin de carrera. No obstante, conviene hacer una reseña a esta metodología si se usa esta plantilla.

Por último, debemos recordar describir de esta forma todo aquello que usemos en el proyecto y dar especial importancia a todo aquello que sea “novedoso”.

WEB-RTC, BBDD Orientadas a objetos, Angular, .Net Core, MySql, API REST

3.1 Concepto 1

Definición

3.2 Concepto 2

Definición

Capítulo 4. Planificación del Proyecto y Presupuesto Iniciales

~~**NOTA: Consultar al director del proyecto acerca de si considera conveniente la inclusión de una sección como esta en la documentación del PFC o que partes de la misma se deben incluir.**~~

4.1 Planificación Inicial

Una vez se ha definido de manera general cual va a ser el objetivo principal del proyecto y lo que se quiere conseguir, se ha realizado una planificación inicial la cual contiene estimaciones de las distintas tareas en las que se va a dividir este proyecto y su duración. Las tareas han sido agrupadas por hitos los cuales distinguen bien las distintas etapas del proyecto.

Esta planificación inicial será modificada de manera continuada durante la elaboración del proyecto, ya que únicamente se trata de una estimación previa que pueda ir variando a lo largo del discurso de las distintas etapas de trabajo. La planificación inicial que se va a mostrar ha sido creada utilizando un diagrama de Gantt, que es básicamente un cronograma que representa las tareas a realizar frente al tiempo.

Id	Nombre de tarea	Comienzo	Fin	Duración
1	**Plataforma de Gestión de Trabajos Académicos**	**mar 15/02/22**	**lun 23/05/22**	**138,75 días?**
2	**Fase previa**	**mar 15/02/22**	**mar 22/02/22**	**10,75 días**
3	Reunión de inicio del proyecto	mar 15/02/22	mar 15/02/22	0,5 días
4	Planteamiento inicial del proyecto	mar 15/02/22	mar 15/02/22	0,5 días
5	Estudio de proyectos similares	mar 15/02/22	mié 16/02/22	1 día
6	Estudio de tecnologías a emplear	mié 16/02/22	jue 17/02/22	1,25 días
7	Presupuesto inicial	jue 17/02/22	vie 18/02/22	1,5 días
8	Planificación inicial	vie 18/02/22	mar 22/02/22	2 días
9	**Análisis del sistema**	**mar 22/02/22**	**jue 03/03/22**	**14 días**
10	Definición del alcance del sistema	mar 22/02/22	mar 22/02/22	0,75 días
11	Identificación de casos de uso	mié 23/02/22	vie 25/02/22	3 días
12	Identificación de requisitos	vie 25/02/22	lun 28/02/22	1,5 días
13	Análisis de interfaces de	mar	mié 02/03/22	1,75 días

	usuario	01/03/22		
14	Especificación del plan de pruebas	mié 02/03/22	jue 03/03/22	1,5 días
15	**Diseño**	**jue 03/03/22**	**mar 22/03/22**	**26,25 días**
16	Arquitectura del proyecto	jue 03/03/22	lun 07/03/22	2 días
17	Interfaces de usuario	lun 07/03/22	jue 10/03/22	3,75 días
18	Base de datos	jue 10/03/22	vie 11/03/22	2 días
19	Diseños de módulos	lun 14/03/22	vie 18/03/22	6 días
20	Especificación técnica del plan de pruebas	lun 21/03/22	mar 22/03/22	2 días
21	**Desarrollo**	**mar 22/03/22**	**mar 03/05/22**	**59,25 días?**
22	**Puesta en marcha del proyecto y esqueleto de la aplicación**	**mar 22/03/22**	**jue 24/03/22**	**3,5 días**
23	Preparación del entorno	mar 22/03/22	mié 23/03/22	1 día
24	Instalación de tecnologías necesarias	mié 23/03/22	jue 24/03/22	1 día
25	**Módulo de autenticación, registro y sesión**	**jue 24/03/22**	**mar 29/03/22**	**6,25 días?**
26	Desarrollo	jue 24/03/22	lun 28/03/22	3 días?
27	Pruebas unitarias	mar 29/03/22	mar 29/03/22	0,75 días?
28	**Módulo de gestión de proyectos académicos**	**mar 29/03/22**	**lun 04/04/22**	**7,75 días?**
29	Desarrollo	mar 29/03/22	vie 01/04/22	3,75 días?
30	Pruebas unitarias	vie 01/04/22	lun 04/04/22	1 día?
31	**Módulo de almacenamiento de archivos**	**lun 04/04/22**	**vie 08/04/22**	**8 días?**
32	Desarrollo	lun 04/04/22	jue 07/04/22	3,75 días?
33	Pruebas unitarias	jue 07/04/22	vie 08/04/22	1,25 días?
34	**Módulo de foro**	**vie 08/04/22**	**jue 14/04/22**	**8 días?**
35	Desarrollo	vie 08/04/22	mié 13/04/22	3,75 días?
36	Pruebas unitarias	mié 13/04/22	jue 14/04/22	0,75 días?
37	**Módulo de gestión de tareas**	**jue 14/04/22**	**mar 19/04/22**	**6 días?**
38	Desarrollo	jue 14/04/22	lun 18/04/22	3 días?
39	Pruebas unitarias	mar 19/04/22	mar 19/04/22	0,5 días?
40	**Módulo de calendario**	**mar 19/04/22**	**jue 21/04/22**	**4 días?**
41	Desarrollo	mar 19/04/22	mié 20/04/22	2 días?
42	Pruebas unitarias	jue 21/04/22	jue 21/04/22	0,5 días?
43	**Módulo de videoconferencia**	**jue 21/04/22**	**jue 28/04/22**	**10 días?**

44	Desarrollo	jue 21/04/22	mar 26/04/22	4,5 días?
45	Pruebas unitarias	mié 27/04/22	jue 28/04/22	1,5 días?
46	**Módulo de notificaciones**	**jue 28/04/22**	**mar 03/05/22**	**5,75 días?**
47	Desarrollo	jue 28/04/22	lun 02/05/22	2,5 días?
48	Pruebas unitarias	lun 02/05/22	mar 03/05/22	0,75 días?
49	**Pruebas**	**mar 03/05/22**	**jue 05/05/22**	**4,5 días**
50	Pruebas de accesibilidad	mar 03/05/22	mar 03/05/22	0,75 días
51	Pruebas de usabilidad	mar 03/05/22	mié 04/05/22	0,75 días
52	Pruebas de integración	mié 04/05/22	mié 04/05/22	0,75 días
53	Pruebas de rendimiento	jue 05/05/22	jue 05/05/22	0,75 días
54	Despliegue	jue 05/05/22	lun 09/05/22	2,5 días
55	**Fase final**	**lun 09/05/22**	**lun 23/05/22**	**20 días**
56	Desarrollo de manuales del sistema	lun 09/05/22	mar 17/05/22	3 días
57	Reunión final del proyecto	mar 17/05/22	mié 18/05/22	0,5 días
58	Entrega de la documentación del proyecto	mié 18/05/22	jue 19/05/22	0,5 días
59	Preparación de la defensa	jue 19/05/22	lun 23/05/22	2,5 días
60	Cierre del proyecto	lun 23/05/22	lun 23/05/22	0,25 días

El proyecto consta de 60 tareas en total. Algunas de estas tareas están marcadas como hitos y constituyen las distintas fases del proyecto. La primera fase, denominada "Fase previa" contiene aquellas tareas básicas y principales necesarias para empezar el proyecto. En la reunión de inicio del proyecto se discute de los distintos objetivos y el planteamiento inicial de este. Además, en dicha reunión, se comenta de las distintas maneras de abordar el proyecto: tecnologías, necesidades, características… Y a partir de ahí se plantea el inicio de la documentación con las tareas de planteamiento inicial del proyecto, estudio de otros proyectos y estudio de tecnologías a emplear. Una vez se tiene el "stack" tecnológico decidido se plantea la realización del presupuesto y la planificación iniciales los cuales deben contener el punto de partida del proyecto.

Una vez se ha completado la fase inicial con la finalización del presupuesto y la planificación inicial se pasa a la fase de análisis del sistema. Esta fase contiene las tareas de identificación de alcance del sistema y de identificación de requisitos, de análisis de interfaces y de especificación del plan de pruebas. Este conjunto de tareas está principalmente relacionado con saber lo que tiene que hacer el sistema y hasta donde debe llegar, dejando claros así los requisitos y las funcionalidades que debe satisfacer el producto final, independientemente de las tecnologías a utilizar.

Tras la fase de análisis del sistema viene la fase de diseño, esta fase debe consistir en establecer la arquitectura del proyecto y diagramar todas las características del sistema que se

va a crear. Durante esta etapa se plantea el diseño de las interfaces, donde se deberán realizar prototipos de pantalla, el diseño de la base de datos, dónde debe aparecer reflejado todo el esquema de datos de la aplicación, el diseño de los módulos que incluirá los distintos diagramas de clases, paquetes, etc. Y finalmente, la especificación del plan de pruebas.

Una vez se ha obtenido todo el diseño de la aplicación comienza la fase más larga del proyecto, el desarrollo. Básicamente esta fase se ha dividido entre los distintos módulos que se pretende que la aplicación tenga, y además, cada uno de estos módulos está partido casi de forma similar en las mismas tareas: una de desarrollo y otra de pruebas. En principio la idea del proyecto es llevar a cabo un desarrollo orientado a pruebas (TDD), es decir, a la vez que se trabaja en el desarrollo como tal, ir implementando las pruebas especificadas para cada módulo en la fase de diseño. El inicio de esta fase viene marcado por una tarea de puesta en marcha, la cual consistirá en implementar la base de la aplicación, que consistirá en la parte de sesión y en el esqueleto y de la instalación de las tecnologías necesarias para empezar a funcionar. Cabe destacar que en la planificación no se ha separado la implementación del backend y del frontend ya que la idea es trabajar en paralelo con ambos, es decir, desarrollar primero la parte del backend de una característica/caso de uso del módulo en específico y posteriormente, una vez se tiene el servicio concreto funcionando desarrollar la parte del frontend que le realice las consultas, una vez se termine la característica o caso de uso implementado, deberá pasar las pruebas unitarias para poder proseguir con el siguiente.

Cuando se termina la fase de desarrollo, la idea principal es utilizar un periodo de tiempo específico únicamente para pruebas (de carácter no unitarias), es decir, pruebas de accesibilidad, integridad, usabilidad y rendimiento, que permitan probar distintas características de la aplicación. En caso de que esta fase implique un gran cambio en el código, la idea es ir especificándolo y replanificándolo en la planificación final que se publicará en los apartados finales de este documento.

La siguiente fase del proyecto sería la relacionada con el despliegue, donde la aplicación se deberá poner en producción y desplegarla.

La última fase del proyecto sería la denominada fase final, que incluye varias tareas que hay que realizar antes de dar por terminado el proyecto como pueden ser: desarrollo de manuales, reuniones finales, preparación de la defensa etc.

4.1.1 Calendario de trabajo

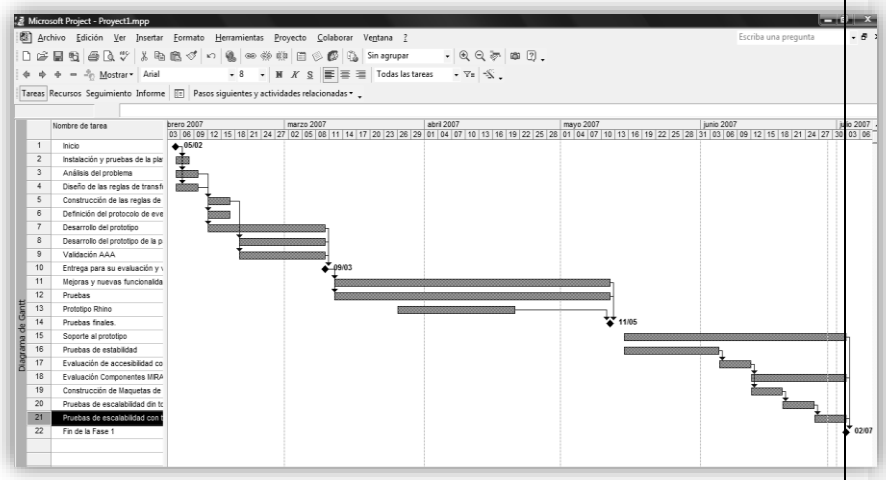
Con formato: Título 3

La planificación del proyecto da un total de 312 horas de trabajo, para llevar estas horas a cabo se plantea un horario de lunes a viernes en el cual se trabajará los lunes, miércoles y viernes cuatro al día horas y los martes y jueves tres horas al día. Hay que tener en cuenta que este horario está planteado para ser compaginado con mis horas de trabajo por lo que puede ser que en la práctica haya que modificarlo por posibles imprevistos.

En lo que respecta a la planificación, solo se ha definido un recurso de trabajo ya que seré el encargado de realizar las distintas tareas de tester, programador, analista etc. Sin embargo, en el presupuesto no se reflejará como tal, por ejemplo, las horas en las que haga trabajo de arquitecto de software no costarán lo mismo que las horas de programador o tester.~~En esta sección, se debe describir el plazo de ejecución del proyecto de forma que puedan fijarse las expectativas de quienes van a recibir el producto resultado del mismo.~~ ~~**Debe realizarse antes de comenzar el proyecto intentando estimar la duración de los módulos en base a los conocimientos.**~~

~~Debe contener un cronograma explicitando las entregas parciales, reuniones, hitos intermedios y duración del proyecto a partir de la fecha de iniciación del mismo. Esto debe realizarse mediante alguna metodología de gestión de proyectos, que sirva de guía sobre la descomposición en tareas, la asignación de recursos, etc.~~

~~Es buena idea incluir diagramas que describan la planificación del avance del proyecto, por ejemplo es posible utilizar un diagrama Gantt o Pert:~~



~~Figura 4.1. Ejemplo de diagrama de Gantt~~

Con formato: Normal

~~Figura 4.2. Ejemplo de diagrama de Pert~~

Con formato: Normal

~~En función del tipo de proyecto, también es posible utilizar otros tipos de cronogramas utilizando otras herramientas como podría ser Excel o tablas:~~

Con formato: Justificado

~~Figura 4.3. Ejemplo de cronograma~~

Con formato: Normal

~~Una vez creada, **se recomienda hacer una copia** sobre la que ir añadiendo los cambios que vayan ocurriendo a lo largo del desarrollo del proyecto. De esta manera, se puede utilizar esta copia como planificación actualizada e ir incorporándola al proyecto en sus respectivas secciones.~~

~~Hay que tener en cuenta que, si se utiliza SCRUM u otra metodología ágil, lo conveniente sería reflejar una pila de tareas en vez del uso de un diagrama de GANT, pues este no es usable en una metodología ágil por cómo funcionan estas. Por ello, se puede realizar una tabla con las diferentes tareas.~~

Con formato: Fuente: 11 pto, Sin Negrita, Color de fuente: Automático

4.2 Presupuesto Inicial

En este apartado se recoge la presupuestación completa del proyecto, para ello se ha dividido en dos subapartados. En el primer subapartado se presentará el presupuesto detallado de costes del proyecto en el cual se analizarán los costes directos e indirectos y se llegará a un presupuesto detallado. En el segundo subapartado se definirá el presupuesto del cliente, es decir, un resumen del presupuesto de costes donde se recojan los conceptos principales y se establezca el precio final que deberá abonar el cliente.~~Desarrollo de Presupuesto Detallado (Empresa)~~

~~4.1.1~~

Con formato: Normal

4.2.1 Desarrollo del Presupuesto Detallado (Empresa)

Para la llevar a cabo la realización del presupuesto inicial del proyecto se han tenido en cuenta los diversos costes generados por el mismo. Estos costes son agrupables entre costes directos, generados principalmente por las actividades llevadas a cabo durante el proyecto y costes indirectos, que son costes que influyen en el proyecto y vienen implícitamente con el proceso de desarrollo de este.

4.2.1.1 Costes indirectos

Con formato: Título 4

Para empezar a presupuestar este proyecto se comenzará explicando los costes indirectos del proyecto. Estos costes indirectos se han clasificado en tres principales grupos: costes de los medios de producción, costes de las licencias y costes de la instalación.

4.2.1.1.1 Costes de las licencias

Este tipo de costes vienen relacionados con el coste monetario que tienen los distintos programas y aplicaciones utilizados para desarrollar el proyecto.

Concepto	Unidades	Coste/mes	Uso (meses)	Coste	Amortización (%)	Coste total	Tipo
Licencia Windows 10 Home	1	-	-	145,00 €	1,18%	1,65 €	Amortización
Licencia Microsoft Office 365 Personal	1	5,75 €	5	28,75 €	-	28,75 €	Alquiler
Licencia MS Project	1	8,40 €	5	42,00 €	-	42,00 €	Alquiler
Licencia MS Visio	1	4,20 €	5	21,00 €	-	21,00 €	Alquiler
MongoDB Dedicated	1	57,00 €	5	285,00 €	-	285,00 €	Alquiler
Docker (version Pro)	1	5,00 €	5	25,00 €	-	25,00 €	Alquiler
Visual Studio Code	1	0,00 €	5	0,00 €	-	0,00 €	Alquiler
Angular 13	1	0,00 €	5	0,00 €	-	0,00 €	Alquiler
NodeJS 14.17.0	1	0,00 €	5	0,00 €	-	0,00 €	Alquiler
					TOTAL	**403,40 €**	

Dentro del coste de las licencias se puede distinguir entre licencias de tipo "alquiler" y licencias de tipo "amortización". La diferencia principal entre ambas es que las licencias que se han considerado de tipo alquiler son las que se paga de manera mensual, y por ello, solo se ha considerado el precio para los cinco meses que dura el desarrollo del proyecto.

Las licencias de tipo amortización están clasificadas así porque son licencias que ya se tenían previamente y solo se ha considerado el precio de lo que costarían en el periodo de cinco meses durante los cuales transcurre el desarrollo del proyecto. Para calcular este porcentaje se ha considerado que la vida total de la licencia son 3 años, que equivalen a 26280 horas, luego se han dividido las 312 horas que dura el proyecto entre este número total de horas y el resultado es el porcentaje que ocupa el tiempo de duración del proyecto respecto a la vida total de la licencia.

4.2.1.1.2 Costes de la instalación

Este tipo de costes vienen asociados al uso de las instalaciones donde se va a trabajar durante las horas que dure el proyecto, para ello se han tenido en cuenta factores como el coste del agua, de la luz, material de oficina, limpieza…

Concepto	Coste/mes	Nº meses	Porcentaje influyente	Total
Coste luz	50,00 €	5	20,00%	50,00 €

Coste agua	30,00 €	5	10,00%	15,00 €
Coste calefacción	30,00 €	5	20,00%	30,00 €
Material de oficina	20,00 €	5	70,00%	70,00 €
Alquiler del local	400,00 €	5	100,00%	2.000,00 €
Limpieza	200,00 €	5	100,00%	1.000,00 €
			TOTAL	**3.165,00 €**

Se ha tenido en cuenta un porcentaje de influencia del proyecto en el total de cada concepto. Esto es así ya que durante el tiempo que dura el proyecto dichos conceptos no sólo son generados única y exclusivamente por la actividad provocada por el desarrollo de este proyecto.

4.2.1.1.3 Costes de los medios de producción

Con formato: Título 5

Este tipo de costes vienen asociados con los costes generados por las tecnologías y medios que se utilizan en el proyecto.

Concepto	Unidades	Coste	Vida útil (meses)	Uso (meses)	Amortización (%)	Total coste	Tipo
Equipo de desarrollo	1	1.500,00 €	36	5	1,18%	210,79 €	Amortización
Conexión a internet	1	29,99 €	-	5	-	149,95 €	Alquiler
					TOTAL	**360,74 €**	

Con formato: Fuente: 9 pto
Tabla con formato
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Tabla con formato

De la misma manera que en el apartado 4.2.1.1.1 se han tenido en cuenta dos tipos de costes, los que son amortizaciones y los que son alquileres. En el caso de la conexión a internet se ha tenido en cuenta el coste generado por los cinco meses que dura el proyecto, y en el caso del equipo de desarrollo se ha tenido en cuenta el porcentaje de amortización calculado anteriormente.

4.2.1.2 Costes directos

Con formato: Título 4

Los costes directos son los que vienen relacionados directamente con el coste de las actividades llevadas a cabo para que el proyecto pueda finalizarse. Para realizar la presupuestación de este apartado se han tenido en cuenta las tareas definidas en el apartado 4.1 junto a las horas que estas implican.

Para poder asignar un precio a cada hora de trabajo del proyecto, se ha tenido en cuenta la siguiente tabla donde a cada rol dentro del proyecto se ha asignado un precio por hora. Para asignar estos precios se han investigado los salarios que cobran distintos perfiles dentro del mundo de TI.

Puesto	Precio/hora
Programador	30,00 €
Analista	35,00 €
Arquitecto de Software	40,00 €
Tester	35,00 €
DevOps	45,00 €

Una vez se tienen los precios por hora y las horas totales de cada tarea solo queda establecer para cada tarea su precio total, y el precio total del proyecto.

Item (I)	Item (II)	Item (III)	Descripción	Horas	Puesto	Subtotal (III)	Subtotal (II)	Coste Total
001			**Fase previa**	**27**				**945,00 €**
	001		Reunión de inicio del proyecto	2	Analista		70,00 €	
	002		Planteamiento inicial del proyecto	2	Analista		70,00 €	
	003		Estudio de proyectos similares	4	Analista		140,00 €	
	004		Estudio de tecnologías a emplear	5	Analista		175,00 €	
	005		Presupuesto inicial	6	Analista		210,00 €	
	006		Planificación inicial	8	Analista		280,00 €	
002			**Análisis del sistema**	**34**				**1.190,00 €**
	001		Definición del alcance del sistema	3	Analista		105,00 €	
	002		Identificación de casos de uso	12	Analista		420,00 €	
	003		Identificación de requisitos	6	Analista		210,00 €	
	004		Análisis de interfaces de usuario	7	Analista		245,00 €	
	005		Especificación del plan de pruebas	6	Analista		210,00 €	
003			**Diseño**	**63**				**2.520,00 €**
	001		Arquitectura del proyecto	8	Arquitecto de Software		320,00 €	
	002		Interfaces de usuario	15	Arquitecto de Software		600,00 €	
	003		Base de datos	8	Arquitecto de Software		320,00 €	
	004		Diseños de módulos	24	Arquitecto de Software		960,00 €	

Con formato: Fuente: 9 pto
Tabla con formato
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto

	005		Especificación técnica del plan de pruebas	8	Arquitecto de Software		320,00 €	
004			**Desarrollo**	**141**				**4.360,00 €**
	001		Puesta en marcha del proyecto y esqueleto de la aplicación	8	Programador		240,00 €	
	002		Módulo de autenticación, registro y sesión	15			465,00 €	
		001	Desarrollo	12	Programador	360,00 €		
		002	Pruebas unitarias	3	Tester	105,00 €		
	003		Módulo de gestión de proyectos académicos	19			590,00 €	
		001	Desarrollo	15	Programador	450,00 €		
		002	Pruebas unitarias	4	Tester	140,00 €		
	004		Módulo de almacenamiento de archivos	20			625,00 €	
		001	Desarrollo	15	Programador	450,00 €		
		002	Pruebas unitarias	5	Tester	175,00 €		
	005		Módulo de foro	18			555,00 €	
		001	Desarrollo	15	Programador	450,00 €		
		002	Pruebas unitarias	3	Tester	105,00 €		
	006		Módulo de gestión de tareas	14			430,00 €	
		001	Desarrollo	12	Programador	360,00 €		
		002	Pruebas unitarias	2	Tester	70,00 €		
	007		Módulo de calendario	10			300,00 €	
		001	Desarrollo	8	Programador	240,00 €		
		002	Pruebas unitarias	2	Tester	60,00 €		
	008		Módulo de videoconferencia	24			750,00 €	
		001	Desarrollo	18	Programador	540,00 €		
		002	Pruebas unitarias	6	Tester	210,00 €		
	009		Módulo de notificaciones	13			405,00 €	
		001	Desarrollo	10	Programador	300,00 €		
		002	Pruebas unitarias	3	Tester	105,00 €		

Con formato: Fuente: 9 pto

005			**Pruebas**	**12**	Tester			**420,00 €**
006			**Despliegue**	**8**	DevOps			**360,00 €**
007			**Fase final**					**885,00 €**
	001		Desarrollo de manuales del sistema	12	Programador		360,00 €	
	002		Reunión final del proyecto	2	Analista		70,00 €	
	003		Entrega de la documentación del proyecto	2	Analista		70,00 €	
	004		Preparación de la defensa	10	Analista		350,00 €	
	005		Cierre del proyecto	1	Analista		35,00 €	
							TOTAL	**10.680,00 €**

Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto
Con formato: Fuente: 9 pto

4.2.1.3 Presupuesto de costes

Una vez se han calculado todos los costes indirectos y directos del proyecto, se puede detallar el presupuesto interno de costes del proyecto. Para calcular este presupuesto se ha tenido en cuenta que el beneficio que se espera obtener es del 10%.

Item(I)	**Item (II)**	**Concepto**	**Total**
001		**Costes Directos**	**10.680,00 €**
	001	Fase Previa	945,00 €
	002	Análisis del sistema	1.190,00 €
	003	Diseño	2.520,00 €
	004	Desarrollo	4.360,00 €
	005	Pruebas	420,00 €
	006	Despliegue	360,00 €
	008	Fase final	885,00 €
002		**Costes Indirectos**	**3.929,20 €**
	001	Costes de licencias	403,46 €
	002	Costes de los medios de producción	360,74 €
	003	Costes de la instalación	3.165,00 €
003		**TOTAL (Costes Directos + Costes Indirectos)**	**14.609,20 €**
004		**Beneficio (10%)**	**1.460,92 €**
		TOTAL	**16.070,12 €**

~~4.2~~

Con formato: Normal

Explicitar el presupuesto de cliente resumido que se ha previsto para la ejecución, de forma que pueda tomarse la decisión de proseguir o no con la ejecución de lo valorado. **Esta sección puede o no tener sentido en función del tipo de proyecto desarrollado. Por ello, se aconseja consultar al director del proyecto acerca de este tema antes de hacer nada relativo a la misma.**

- **Debe realizarse antes de comenzar el proyecto intentando estimar el software y hardware que se necesitará y hacer un presupuesto estimado en base a lo que puede costar el proyecto.**
- Debe incluir el coste total de la ejecución para la organización que ha de hacerse cargo de este proyecto.
- En este apartado debe tenerse especial cuidado en presentar las cifras de manera no ambigua, completa, sin costes ocultos y dando un total general desglosado por partidas.
- Habitualmente sólo se ponen los ítems, eliminando del presupuesto los sub-ítems.
- Si aun así resultase excesivamente complejo, se puede resumir en grandes partidas de trabajo que tengan sentido práctico y económico para el cliente.
- Notas:
    - Hay que incluir todos los elementos requeridos en el proyecto, incluyendo licencias de programas (Microsft Windows, Microsoft Office, Microsoft Projects,…). Podéis utilizar Internet para ver el coste de estas licencias. Si es software gratuito, añadirlo con un coste de 0€.
    - Hay que añadir el coste de cada módulo.
    - Hay que añadir el diferente Hardware que se utiliza: portátil, servidor, cuenta en Amazon Web Services o Microsoft Azure, Raspberry Pi, Arduino,…
    - Acordarse del material de oficina. Este es todo lo relacionado a aquel material que se gasta de continuo y que es necesario para que la empresa funcione: Internet, Electricidad, papel y lápices, tóner, impresora,…
    - Hay que amortizar los elementos utilizados. Por ejemplo, un software te cuesta 1.100€ y prevés amortizarla en un año. Vas amortizarlo sobre 1200 por si hay subidas/reparaciones. Si el proyecto te lleva 6 meses, le tienes que añadir 600€ a ese proyecto para esa licencia/hardware. Esta parte se ha de explicar de qué manera y en qué tiempo se amortiza cada elemento.
    - Hay que añadir el beneficio de la empresa en el caso de que no se añada en cada módulo por separado.
    - Hay que acordarse de añadir el IVA actual y no uno de épocas pasadas.
    - Se pueden crear roles para los diferentes trabajadores (analista, jefe de proyecto, programador senior, programador junior,…) o bien utilizar uno solo como es el caso. Esto depende hasta qué punto se quiera simular el proyecto. En cualquier caso, hay que especificar lo que cobra cada rol.
- Si se usase cualquier otro criterio para hacer este resumen del presupuesto, deberá justificarse adecuadamente. La siguiente tabla muestra un ejemplo de cómo se puede realizar el mismo.

Ítem	Concepto	Cantidad	Amortización	Precio	Total (€)

				Unitario (€)	
1	*Investigación Inicial*				
1.1	Posibles herramientas	-	100%	3776	3776
2	*Recursos Software*				
2.1	Microsoft Windows 8	2	5%	149	7.45
2.2	Microsoft Office Proffesional 2013	1	7%	539	37.73
3	*Recursos Hardware*				
3.1	Servidor Microsoft Azure	1	100%	350	350
3.2	Portátil	2	10%	900	180
4	*Módulos del proyecto*				
4.1	Módulo 1	1	100%	2689	2689
...					
Subtotal					...
Beneficio (7%)					...
IVA (21%)					...
TOTAL					...

Los formatos de la tabla anterior no son obligatorios, deben adaptarse a cada proyecto.

4.2.11.1.1 Desarrollo de Presupuesto Detallado (Empresa)

A la hora de elaborar un presupuesto hay que tener en cuenta dos aspectos importantes:

- ¿Cuánto cuesta desarrollar el proyecto? **(Presupuesto de Costes)**
- ¿Cuánto voy a cobrar por los trabajos? **(Presupuesto del Cliente)**

El alumno debe realizar ambos. Por lo que respecta al presupuesto del cliente, se hará constar en esta sección y deberá estar basado en el presupuesto de costes realizado. El presupuesto de costes puede realizarse en esta sección o figurar en un anexo específico, a criterio del director del proyecto.

Dicho presupuesto debe ser claro y exhaustivo, adelantándose a las dudas del cliente y presentando todos los elementos, de manera que no exista ningún "punto oscuro". Para ello deben constar únicamente ítems que entienda el cliente. Como ejemplo, el coste de amortización de los equipos de desarrollo no es un dato que deba constar en este presupuesto, sino en el de costes. Un curso de formación, el análisis del sistema, la instalación de un SGBD, etc. si son elementos que entregamos al cliente, sí deben constar en este apartado.

En general el presupuesto del cliente debe cubrir el presupuesto de costes de desarrollo para que el proyecto resulte rentable y debe tener un margen de beneficio con respecto a aquel.

El presupuesto del cliente no debe ser una tabla de valores monetarios vacía, sino que debe ser acompañado por una explicación de estos valores.

Las características fundamentales que debe cumplir son:

- **Completo:** Desde la presentación de nuestra empresa, hasta la solución que pensamos darle al proyecto del cliente, ejemplos de proyectos ya realizados, presupuesto cerrado indicando que incluye y tan importante como esto, qué no incluye.
- **Claro:** Siempre hay que pensar que el cliente no sabe. Si sabe puede pensar que somos muy básicos, pero si no sabe y le pasamos un presupuesto excesivamente técnico, no le podremos hacer llegar nuestra idea de manera que él pueda visualizar el resultado.
- **Conciso:** No hay que andarse por las ramas. A nosotros nos interesa poder transmitir la mayor cantidad de ideas en el menor espacio posible. Para ello, es muy recomendable utilizar tablas y/o puntos.

A la hora de desarrollar el presupuesto, debemos pensar en hacer referencias a:

- **Metodología de trabajo:** Explicar al cliente como vamos a hacer las cosas, comentar por ejemplo que primero se va a diseñar la web o la base de datos, que hasta que no se apruebe (firmado), no se va a pasar a la siguiente fase.
- **Tiempos de ejecución:** Hay que cumplirlos. También hay que ajustar mucho e indicar cuáles son nuestras responsabilidades y cuáles son las suyas en cuanto a entregas de material, validaciones etc., y la influencia que tendrá en los tiempos, el retraso en las entregas y las aprobaciones.
- **Presupuesto detallado propiamente dicho:** Es muy útil tabular cada una de las secciones que van a componer el proyecto y desglosarlo por su coste.
- **Formas de pago (Si procede):** Indicando también los plazos que habrá que cumplir.

Todos estos apartados, pueden no llevar este orden, incluso estar mezclados, pero es importante tener transmitir todo esto claramente al cliente. Un ejemplo de un desglose de un presupuesto en una tabla se muestra a continuación:

Item	SubItem	Concepto	Cantidad	Precio Unitario	TOTAL
01		***Desarrollo de la Aplicación***			**7.400,00 €**
	001	*Analisis*	20,00	60,00 €	1.200,00 €
	002	*Diseño*	25,00	60,00 €	1.500,00 €
	003	*Implementación*	100,00	37,00 €	3.700,00 €
	004	*Pruebas*	25,00	40,00 €	1.000,00 €
02		***Formación***			**8.000,00 €**
	001	*Directivos*	20,00	50,00 €	1.000,00 €
	002	*Usuarios*	40,00	50,00 €	2.000,00 €
	003	*Mantenimiento*	100,00	50,00 €	5.000,00 €
				Subtotal	30.800,00 €
				IVA (16%)	4.928,00 €
				TOTAL	**35.728,00 €**

~~El presupuesto debe contener:~~

- ~~Cuando proceda, un cuadro de precios de las unidades de medida correspondientes: componentes de hardware, elementos de software, horas persona de diferentes categorías, elementos auxiliares y otros.~~
- ~~Cuando proceda, costes de unidades lógicas con entidad propia dentro del proyecto, con la descomposición correspondiente de componentes de hardware, elementos de software, horas persona, elementos auxiliares y otros.~~
- ~~El presupuesto propiamente dicho debe contener la valoración económica global, descompuesta siguiendo la estructura de desglose de los elementos utilizada en la planificación y ejecución del proyecto.~~
- ~~El presupuesto debe especificar claramente las bases con las que se confecciona el mismo.~~

4.2.2 Desarrollo de Presupuesto Simplificado (Cliente)

El presupuesto que se envía al cliente debe mostrar de manera sencilla y adecuada la presupuestación general del proyecto, excluyendo determinadas partidas del presupuesto de costes que no deben ser enviadas, así como los beneficios que se desea obtener. Además, este presupuesto debe ser fácilmente comprensible para cualquier persona que no posea conocimientos especialmente técnicos de la materia y le permita ver de una manera sencilla el coste total de los servicios que va a contratar.

En el caso de este proyecto lo primero que hay que determinar son las partidas que no son enviables al cliente, para ello se deben de calcular los costes generados por las partidas que no pueden ser enviables al cliente. Se han seleccionado principalmente la partida inicial, donde se realiza un trabajo previo de documentación e iniciación y partida final que va relacionada con las tareas de cierre del proyecto, la generación de manuales etc. Además de esto hay que tener en cuenta que el beneficio tampoco puede ser enviado al cliente.

Una vez se ha calculado este coste total no enviable lo que se debe hacer es restar de los costes directos las partidas no enviables y a partir de ese valor calcular el porcentaje que se ha de incrementar cada partida enviable al cliente. Una vez se ha calculado dicho porcentaje, se aplica a cada uno de los conceptos del presupuesto, se suman y se aplica el IVA final.

Item (I)	**Concepto**	**Total**
001	Análisis del sistema	2.160,84 €
002	Diseño	4.575,90 €
003	Desarrollo	7.917,03 €
004	Pruebas	762,65 €
005	Despliegue	653,70 €
	Subtotal	16.070,12 €
	IVA (21%)	3.374,73 €
	TOTAL	**19.444,85 €**

Tabla con formato

Con formato: Fuente: Negrita

4.2.2

Con formato: Normal

~~En esta sección debemos rellenar esta tabla Excel para calcular el coste económico de desarrollar el sistema planteado en la documentación. Debemos considerar cosas como las horas de análisis, diseño y programación, el personal necesario, materiales, programas, equipos, etc.~~

Concepto	Cantidad	Precio Unitario	Coste Total Concepto
Horas de Programador			0,00 €
Horas de Análisis y Diseño			0,00 €
…			0,00 €
			0,00 €
			0,00 €
		Subtotal	0,00 €
		IVA (16%)	0,00 €
		TOTAL	**0,00 €**

~~Si fuese necesario, podemos incluir en esta sección una planificación temporal de las distintas etapas del proyecto (análisis, diseño, etc.).~~

Capítulo 5. Análisis

Este apartado contendrá toda la especificación de requisitos y toda la documentación del análisis del sistema que se pretende crear ~~la aplicación,~~ a partir de la cual se elaborará posteriormente el diseño.

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

El sistema completo a crear permitirá a los estudiantes dados de alta en la aplicación gestionar sus trabajos académicos de manera coordinada con su tutor. Un estudiante dentro de la aplicación podrá visualizar la información sobre los trabajos académicos a los cuales está asignado, acceder a un apartado independiente para cada trabajo que actuará como foro para poder comunicarse con su tutor, gestionar su propio apartado (también independiente para cada trabajo) en el cual podrá subir archivos, crear directorios y gestionarlos, con el objetivo de tener un lugar de almacenamiento de los archivos relacionados con el trabajo. También podrá visualizar un calendario en el cual se podrán crear y marcar eventos, y se deberá tener acceso a un apartado de tareas que deberá presentarse como un tablero Kanban en el cual puedan personalizarse las columnas y desde el cual llevar un control del estado de las tareas a realizar durante el trabajo. Los profesores tendrán la posibilidad de acceder a toda la información de los trabajos académicos creados por ellos mismos, así como a los que han sido asignados. Además, serán encargados de la creación de los trabajos y de invitar a los alumnos a los mismos (que deberán confirmar la invitación para poder acceder). Se plantea también un sistema de notificaciones que deberá informar a las personas implicadas de cualquier cambio, actualización o novedad en los proyectos a los cuales está asignada. Finalmente el sistema deberá también ofrecer la posibilidad de establecer comunicación por llamada entre el alumno y el tutor con opción tanto de audio como de video.

Por otro lado, la aplicación estará internacionalizada en dos idiomas y el diseño de esta deberá de ser adaptable a distintos dispositivos y pantallas, es decir, “responsive”.

~~5.1.1~~ El proyecto consistirá en la creación de una aplicación web que utilizará el framework Angular en el cliente y se comunicará mediante peticiones HTTP con el servidor. Para realizar está comunicación, el servidor expondrá una API utilizando NodeJS con el framework Express.js que además será el encargado de hacer la comunicación con la base de datos MongoDB. Finalmente, el despliegue consistirá en crear un contenedor Docker que correrá una imagen de la aplicación.

~~Se trata de describir de nuevo el sistema, pero en lugar de repetir lo que ya hemos dicho de él, tenemos que constatar en este apartado hasta donde vamos a llegar en su construcción, es~~

Con formato: Normal

5.4.15.2.1 Obtención de los Requisitos del Sistema

El producto de esta sección se crea para su aprobación formal, es decir, que los potenciales clientes deben ver a partir de él las especificaciones completas del sistema. Además, esta sección construirá una base para solicitar cambios en los requisitos antes de avanzar más en la construcción del sistema.

5.2.1.1 Requisitos funcionales

A continuación van a exponerse los distintos requisitos funcionales que la aplicación debe satisfacer. Para ello se dividirá de manera que queden comprendidos cada uno de los módulos en apartados distintos.

5.2.1.1.1 Usuario no autenticado

- RF.1. El sistema permitirá la creación de cuentas de usuario mediante un formulario a los usuarios no autenticados:
  - RF.1.1. El usuario deberá especificar obligatoriamente:
    - RF.1.1.1. Nombre.
    - RF.1.1.2. Apellidos.
    - RF.1.1.3. Dirección de correo electrónico.
    - RF.1.1.4. Contraseña.
      - RF.1.1.4.1. Deberá tener una longitud de al menos 8 caracteres.
      - RF.1.1.4.2. El usuario deberá introducirla en un campo "Repite contraseña" que deberá coincidir con la contraseña especificada en el requisito RF.1.1.4.
    - RF.1.1.5. Un rol de usuario que será uno de los siguientes:
      - RF.1.1.5.1 Estudiante
      - RF.1.1.5.2. Profesor
  - RF.1.2. El sistema deberá almacenar los datos:
    - RF.1.2.1. Los datos especificados por el usuario del requisito RF.1.1.1 al RF.1.1.5.
    - RF.1.2.2. El dato proporcionado en el requisito RF.1.1.4. deberá almacenarse de manera encriptada.
    - RF.1.2.2. Fecha de registro.
- RF.2. El sistema permitirá a un usuario no autenticado mediante un formulario iniciar sesión:
  - RF.2.1. El usuario deberá especificar obligatoriamente:

Con formato

- RF.2.1.1 Correo electrónico.
- RF.2.1.2. Contraseña

- RF.2.2 El sistema deberá de comprobar los datos introducidos por el usuario en los RF.2.1.1 y RF.2.1.2 y comprobar que coinciden con los almacenados en el sistema.
  - RF.2.2.1. En caso de que no coincidan deberá denegarse la autenticación.
  - RF.2.2.2. En caso de que coincidan el usuario pasará a estar autenticado.

- RF.3. Un usuario no autenticado podrá también cambiar el idioma en el que se muestra la aplicación.

5.2.1.1.2 Usuario autenticado

Todos los requisitos expuestos en este apartado hacen referencia a las funcionalidades las cuales podrá realizar un usuario previamente autenticado.

5.2.1.1.2.1 Sesión

- RF.4. Un usuario autenticado podrá desconectar su sesión pasando de nuevo a estar como no autenticado.

5.2.1.1.2.2 Gestión del perfil

- RF.5. El sistema debe permitir que el usuario realice las siguientes acciones respectivas a su perfil:
  - RF.5.1. Consultar sus datos de perfil.
    - RF.5.1.1. Nombre
    - RF.5.1.2. Apellidos
    - RF.5.1.3. Dirección de correo electrónico.
    - RF.5.1.4. Rol.
    - RF.5.1.5. Imagen del usuario.
  - RF.5.2. Modificar sus datos de perfil.
    - RF.5.2.1. El usuario podrá modificar los siguientes datos que le pertenecen:
      - RF.5.2.1.1. Nombre
        - RF.5.2.1.1.1. Es obligatorio.
      - RF.5.2.1.2. Apellidos
        - RF.5.2.1.2.1. Es obligatorio.
      - RF.5.2.1.3. Imagen del usuario.
        - RF.5.2.1.3.1. Es opcional.
    - RF.5.2.2. El sistema deberá actualizar los datos almacenados del usuario con las actualizaciones.
  - RF.5.3. Eliminar su perfil.
    - RF.5.3.1. El sistema debe eliminar al usuario almacenado.

Con formato
Con formato
Con formato
Con formato: Normal
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato

5.2.1.1.2.3 Gestión de trabajos académicos.

- RF.6. Un usuario con rol profesor podrá crear nuevos trabajos académicos.
  - RF.6.1. Deberá especificar los datos del proyecto:
    - RF.6.1.1 Título
      - RF.6.1.1.1. Es obligatorio
    - RF.6.1.2. Descripción
      - RF.6.1.2.1. Es obligatoria.
    - RF.6.1.3. Curso.
      - RF.6.1.3.1. El usuario deberá seleccionar un año.
      - RF.6.1.3.2. Es opcional.
    - RF.6.1.4. Alumno/s implicados en el proyecto.
      - RF.6.1.4.1. Es opcional
      - RF.6.1.4.2. El usuario tendrá acceso a una lista con todos los alumnos dados de alta en el sistema.
      - RF.6.1.4.3. El usuario podrá seleccionar uno o varios alumnos.
    - RF.6.1.5. Profesor/es implicados en el proyecto.
      - RF.6.1.5.1. Es opcional.
      - RF.6.1.5.2. El usuario tendrá acceso a una lista con todos los profesores dados de alta en el sistema.
      - RF.6.1.5.3. El usuario podrá seleccionar uno o varios profesores.
      - RF.6.1.5.4. El sistema deberá incluir automáticamente al profesor que crea el trabajo académico dentro de esta lista.
    - RF.6.1.6. Categoría del trabajo.
      - RF.6.1.6.1. Es obligatorio.
      - RF.6.1.6.2. Deberá ser de uno de los siguientes tipos:
        - RF.6.1.6.2.1. Trabajo Fin de Grado.
        - RF.6.1.6.2.2. Trabajo Fin de Máster.
        - RF.6.1.6.2.3. Tesis Doctoral.
    - RF.6.1.7. El sistema deberá introducir automáticamente al usuario que crea el proyecto académico como propietario del trabajo.
  - RF.6.2. Cuando se crea un nuevo trabajo académico, el sistema debe crear una solicitud de incorporación tanto para los profesores especificados en RF.6.1.5, como a los alumnos especificados en RF.6.1.4.
  - RF.6.3. El usuario debe poder consultar la lista de solicitudes pendientes.
  - RF.6.4. Los usuarios que reciben una solicitud de incorporación podrán:
    - RF.6.4.1. Aceptar la solicitud
      - RF.6.4.1.1. El usuario pasará a estar implicado en el trabajo académico.
    - RF.6.4.2. Denegar la solicitud
      - RF.6.4.2.1. El usuario no pasará a estar implicado en el trabajo académico.
- RF.7. El usuario debe poder visualizar los trabajos académicos en los que está implicado.

Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato
Con formato

- RF.7.1. Si el rol es estudiante visualizará únicamente los trabajos en los que está asignado.
- RF.7.2. Si el rol es profesor visualizará los trabajos que ha creado y los trabajos en los que está asignado.
- RF.7.3. EL usuario podrá filtrar por categoría de trabajo:
  - RF.7.3.1. Trabajo Fin de Grado.
  - RF.7.3.2. Trabajo Fin de Máster.
  - RF.7.3.3. Tesis doctoral.

Con formato

- RF.8. El usuario con rol profesor que sea propietario de un trabajo académico debe poder modificar sus datos.
  - RF.8.1. Título
  - RF.8.2. Descripción
  - RF.8.3. Categoría.
    - RF.8.3.1. Deberá de ser una de las categorías especificadas entre RF.6.1.6.2.1 y RF.6.1.6.2.3.

Con formato

  - RF.8.4. Alumnos implicados.
    - RF.8.4.1. El usuario debe poder añadir o eliminar alumnos del proyecto.

Con formato

  - RF.8.5. Profesores implicados.
    - RF.8.4.2. El usuario debe poder añadir o eliminar profesores del proyecto.

Con formato

- RF.9. El usuario con rol profesor que sea propietario de un trabajo académico debe poder eliminarlo.
  - RF.9.1. El sistema deberá eliminar los datos relacionados con el trabajo.
- RF.10. El usuario debe poder acceder a todas las funcionalidades de un trabajo académico en el cual está implicado:
  - RF.10.1. Foro

Con formato: Izquierda

  - RF.10.2. Gestión de archivos.
  - RF.10.3. Tareas.
  - RF.10.4. Calendario.

Con formato: Párrafo de lista, Con viñetas + Nivel: 2 + Alineación: 1,9 cm + Sangría: 2,54 cm

5.2.1.1.2.3.1 Foro

- RF.11. El usuario implicado en el proyecto podrá crear un nuevo tema en el foro especificando:
  - RF.11.1. Título descriptivo del tema.

Con formato

    - RF.11.1.1. Es obligatorio.

Con formato

  - RF.11.2. Contenido del tema.
  - RF.11.3. El sistema almacenará la fecha y la hora de creación del tema.
- RF.12. El usuario podrá marcar un tema como favorito.
- RF.13. El usuario implicado en el proyecto podrá consultar la lista de temas existentes en el proyecto académico.
  - RF.13.1. Deberán mostrarse las propiedades especificadas en RF.11.1, RF.11.2 y RF.13.3 y el número de respuestas para cada tema.
  - RF.13.2. La lista de temas mostrada debe poder ordenarse según:

        - RF.13.2.1. Más reciente
        - RF.13.2.2. Menos reciente.
        - RF.13.2.3. Temas favoritos.
- RF.14. El usuario debe poder eliminar un tema.
    - RF.14.1. Si el usuario es un estudiante solo podrá eliminar los temas que él ha creado.
    - RF.14.2. Si el usuario es un profesor podrá eliminar cualquier tema del foro.
- RF.15. El usuario debe poder responder a un tema.
    - RF.15.1. Cada respuesta debe tener un contenido obligatorio.
    - RF.15.2. El sistema debe incluir automáticamente la hora y el día a cada respuesta.
- RF.16. El usuario debe poder consultar las respuestas de un tema.
    - RF.16.1. Las respuestas deben mostrarse por orden cronológico de acuerdo con la fecha y la hora en la que fueron publicadas.
- RF.17. El usuario debe poder eliminar una respuesta de un tema.
    - RF.17.1. Si el usuario es un estudiante solo podrá eliminar las respuestas que el haya publicado.
    - RF.17.2. Si el usuario es un profesor podrá eliminar cualquier respuesta.

Con formato

Con formato: Normal

5.2.1.1.2.3.2 Gestión de archivos

- RF.18. El sistema debe crear un directorio raíz automáticamente para gestionar los archivos de un trabajo académico cuando este se crea.
- RF.19. El usuario debe poder crear directorios en su apartado de archivos dentro del directorio raíz.
    - RF. 19.1. El sistema debe comprobar que el directorio creado no exista.
        - RF.19.1.1. En caso de que exista no debe permitir la creación del directorio.
        - RF.19.1.2. En caso de que no exista debe permitir la creación del directorio.
    - RF.19.2. El sistema debe comprobar que el directorio a crear no tenga más de cinco niveles de profundidad en el árbol de directorios.
        - RF.19.2.1 En caso de que la profundidad sea mayor a cinco, el directorio no debe crearse.
        - RF.19.2.2. En caso de que la profundidad sea mayor de cinco, el directorio debe crearse.
- RF.20. El usuario debe poder subir archivos a sus directorios.
    - RF.20.1. En caso de que el archivo ya exista en ese directorio, debe sobrescribirse.
    - RF.20.2. En caso de que el archivo no exista en ese directorio, debe subirse a dicho directorio.
- RF.21. El usuario debe poder descargar los archivos subidos a sus directorios.
    - RF.21.1. El sistema debe proveer la descarga del archivo manteniendo el estado original en el que fue subido.

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

- RF.22. El usuario podrá eliminar los archivos subidos a sus directorios.
- RF.23. El usuario podrá eliminar sus directorios.
    - RF.23.1. En caso de que el directorio esté vacío, es decir, no contenga ningún archivo subido, debe poder eliminarse.
    - RF.23.2. En caso de que el directorio no esté vacío no se debe permitir su eliminación.
    - RF.23.3. En caso de que el directorio a borrar sea el directorio raíz del proyecto no debe permitirse la eliminación.

Con formato: Párrafo de lista, Con viñetas + Nivel: 2 + Alineación: 1,9 cm + Sangría: 2,54 cm

5.2.1.1.2.3.3 Calendario

- RCalendario.1. El usuario tendrá acceso a un calendario independiente para cada trabajo académico.
- RCalendario.2. El usuario podrá crear eventos en los distintos días del calendario.
    - RCalendario2.1. Cada evento constará de los siguientes atributos:
        - RCalendario.2.1.1. Título del evento.
            - RCalendario.2.1.1.1. Es obligatorio
        - RCalendario.2.1.2. Descripción del evento.
        - RCalendario.2.1.3. Etiquetas.
            - RCalendario.2.1.3.1. El usuario podrá elegir entre crear nuevas etiquetas o asignar etiquetas existentes.
            - RCalendario.2.1.3.2. El usuario podrá eliminar etiquetas.
                - RCalendario.2.1.3.2.1. Si el usuario elimina una etiqueta, todos los eventos con esa etiqueta deben eliminarla.
            - RCalendario.2.1.3.3. Por defecto el sistema incluirá las etiquetas “Reunión”, “Tarea” y “Entrega”.
- RCalendario.3. El usuario podrá eliminar cualquier evento previamente creado.
- RCalendario.4. El calendario deberá mostrar automáticamente un evento para la fecha aproximada de inicio y fin de las tareas del apartado de Tareas del trabajo académico.
- RCalendario.5. El usuario podrá filtrar los elementos a mostrar en el calendario:
    - RCalendario.5.1. Mostrar únicamente sus eventos.
    - RCalendario.5.2. Mostrar los eventos relacionados con una o varias etiquetas.
    - RCalendario.5.3. Mostrar únicamente las fechas de inicio y fin de las tareas.
    - RCalendario.5.4. No mostrar nada.

Con formato

Con formato: Párrafo de lista, Con viñetas + Nivel: 2 + Alineación: 1,9 cm + Sangría: 2,54 cm

5.2.1.1.2.3.4 Tareas

- RTareas.1. El usuario tendrá acceso a un apartado para gestionar sus tareas.
- RTareas.2. El sistema ofrecerá la posibilidad de crear nuevas tareas.
    - RTareas.2.1. El usuario tendrá que especificar los siguientes atributos para cada tarea a crear:
        - RTareas.2.1.1. Título.
            - RTareas.2.1.1. Obligatorio.
        - RTareas.2.1.2. Descripción.
            - RTareas.2.2.1. Opcional

Con formato

- RTareas.2.1.3. Responsable/s.
    - RTareas.2.1.3.1. Opcional.
    - RTareas.2.1.3.2. Será una lista de los usuarios implicados en la realización de la tarea.
- RTareas.2.1.4. Fecha estimada de inicio.
    - RTareas.2.1.4.1. Es opcional
- RTareas.2.1.5. Fecha estimada de finalización.
    - RTareas.2.1.5.1. Es opcional
- RTareas.2.2. La tarea almacenará también automáticamente su fecha de creación.

Con formato

Con formato

Con formato

- RTareas.3. El usuario podrá crear apartados para clasificar sus tareas.
    - RTareas.3.1. Por defecto cuando se crea un trabajo académico el sistema establecerá cuatro apartados:
        - RTareas.3.1.1. Tareas nuevas.
        - RTareas.3.1.2. Tareas en progreso.
        - RTareas.3.1.3. Tareas bloqueadas.
        - RTareas.3.1.4. Tareas terminadas.
    - RTareas.3.2. Cada apartado constará de un título obligatorio.
- RTareas.4. El usuario podrá modificar los títulos de los apartados en cualquier momento.
- RTareas.5. El usuario podrá eliminar los apartados creados.
    - RTareas.5.1. Si un apartado tiene tareas asociadas, estas se eliminarán también.
- RTareas.6. El usuario podrá eliminar tareas creadas previamente.
- RTareas.7. El usuario podrá modificar todos los atributos especificados desde RTareas.2.1 hasta RTareas.2.3 de las tareas creadas previamente.
- RTareas.8. El usuario podrá cambiar el apartado en el que las tareas se encuentran.
-

Con formato: Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

RNotificaciones.1. El usuario recibirá notificaciones cuando se produzcan cambios en cualquier trabajo académico en el que se encuentre asociado.

- RNotificaciones.2. Los eventos que producirán una notificación serán los siguientes:
    - RNotificaciones.2.1. Foro.
        - RNotificaciones.2.1.1. Nuevo tema creado.
        - RNotificaciones.2.1.2. Nueva respuesta a un tema.
    - RNotificaciones.2.2. Archivos.
        - RNotificaciones.2.2.1. Crear nuevo directorio.
        - RNotificaciones.2.2.2. Crear nuevo archivo.
        - RNotificaciones.2.2.3. Eliminar archivo.
        - RNotificaciones.2.2.4. Eliminar directorio.
    - RNotificaciones.2.3. Calendario.
        - RNotificaciones.2.3.1. Nuevo evento creado.
        - RNotificaciones.2.3.2. Evento eliminado.
        - RNotificaciones.2.3.3. Eventos que suceden hoy.

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 2 + Alineación: 1,9 cm + Sangría: 2,54 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 3 + Alineación: 3,17 cm + Sangría: 3,81 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 2 + Alineación: 1,9 cm + Sangría: 2,54 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 3 + Alineación: 3,17 cm + Sangría: 3,81 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 2 + Alineación: 1,9 cm + Sangría: 2,54 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 3 + Alineación: 3,17 cm + Sangría: 3,81 cm

- RNotificaciones.2.3.4. Eventos que van a suceder mañana.
- RNotificaciones.2.4. Tareas.
  - RNotificaciones.2.4.1. Nueva tarea creada.
  - RNotificaciones.2.4.2. Nuevo apartado de clasificación de tareas creado.
  - RNotificaciones.2.4.3. Tarea eliminada.
  - RNotificaciones.2.4.4. Apartado de clasificación de tareas eliminado.
- RNotificaciones.2.5. Videollamadas.
  - RNotificaciones.2.5.1. Videollamada perdida.

5.2.1.1.4 Videollamadas

- RVideo.1. El usuario tendrá la opción de realizar una videollamada con cualquier otro usuario participante en un trabajo académico que se encuentre conectado.
- RVideo.2. El usuario tendrá la opción de compartir recursos audio, video o ambas.
  - RVideo.2.1. El usuario podrá dejar de compartir los recursos de audio, video o ambos en cualquier momento.
- RVideo.3. El usuario tendrá la opción de terminar una videollamada en cualquier momento.

5.2.1.2 Requisitos no funcionales

- RNF.1. El navegador debe soportar la tecnología WebRTC.
- RNF.2. El navegador debe soportar HTML5 y Javascript.
- RNF.3. Las contraseñas deben almacenarse de manera encriptada.
- RNF.4. El sistema será desarrollado utilizando la versión 14.17.0 de NodeJs y la versión 6.4.13 de npm.
- RNF.5. El sistema será desarrollado utilizando la versión 13.1.0 de angular/core.
- RNF.6. El sistema será desarrollado utilizando la versión 5.0.6 de MongoDB Community en local.
-

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 2 + Alineación: 1,9 cm + Sangría: 2,54 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Título 4

Con formato: Normal, Sin viñetas ni numeración

~~5.4.2~~5.2.2 Identificación de Actores del Sistema

~~Un actor es algo o alguien que reside fuera del sistema y que interactúa con el mismo (actor primario) o bien es algo o alguien sobre el que el sistema actúa (actor secundario). Un actor puede ser una persona, un dispositivo, otro sistema o un subsistema.~~

La utilización de la aplicación por parte de los usuarios no registrados estará muy limitada. Este tipo de usuarios son los que acceden por primera vez a la aplicación y todavía no han tenido la oportunidad de completar el proceso de registro. Los usuarios no registrados únicamente podrán acceder a las funciones de inicio de sesión y también podrán cambiar de idioma la aplicación mediante las opciones de internacionalización.

5.2.2.2 Usuario registrado

Los usuarios registrados son los usuarios que ya han completado el proceso de registro, y una vez lo han completado también han iniciado sesión en la aplicación. Las funciones de estos usuarios variarán en función de su rol, el cual será estudiante o profesor (que se seleccionará mediante el proceso de registro en la aplicación).

5.2.2.2.1 Estudiante

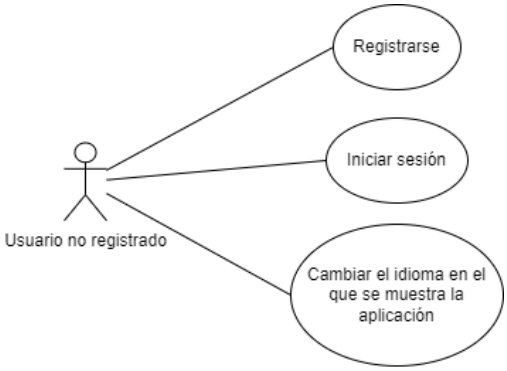
Si el usuario registrado tiene rol de estudiante tendrá acceso a diversas funciones: podrá acceder a visualizar la información de los trabajos académicos a los que ha sido asignado, podrá acceder a la información de su perfil, editarlo o eliminarlo, podrá acceder a la ayuda del sistema y al apartado de aceptar solicitudes de incorporación a trabajos académicos. Luego para cada trabajo académico podrá realizar las siguientes funciones: gestionar sus archivos (creación, eliminación y modificación), gestionar su apartado de tareas (creación, eliminación y modificación de tareas y creación, eliminación y modificación de apartados de clasificación de tareas), gestionar su calendario (creación, modificación y eliminación de sus eventos) y utilización del foro (creación, modificación y eliminación de sus temas, creación y eliminación de sus respuestas), y también podrá acceder a iniciar una videollamada con otro usuario (ya sea estudiante o profesor) asignado al trabajo académico.

5.2.2.2.2 Profesor

El rol de profesor podrá realizar las mismas funciones que un estudiante y además tendrá privilegios extra. Este rol será el encargado de crear nuevos trabajos académicos desde los cuales se invitará a otros alumnos y profesores a pertenecer. Además, para cada uno de los trabajos académicos que el profesor haya creado tendrá las opciones de modificar la información y de eliminarlo. Dentro de cada trabajo académico también tendrá privilegios elevados, en el apartado del foro podrá eliminar respuestas y temas de cualquier otro profesor y estudiante. En el apartado de calendario podrá también eliminar y modificar cualquier evento creado por un estudiante o un profesor.

5.2.3 Especificación de Casos de Uso

5.2.3.1 Usuario no registrado



Nombre del Caso de Uso
Registrarse
Descripción
El usuario que no esté registrado en la aplicación podrá crear su cuenta en el sistema. Deberá completar la información: • Nombre • Apellidos • Correo electrónico • Contraseña • Repetir su contraseña • Rol (estudiante o profesor). Todos los datos requeridos serán obligatorios.

Nombre del Caso de Uso
Iniciar sesión
Descripción
Un usuario que esté registrado en el sistema podrá iniciar sesión en la aplicación rellenando un formulario compuesto de: • Correo electrónico • Contraseña

Nombre del Caso de Uso
Cambiar el idioma en el que se muestra la aplicación
Descripción
Un usuario no registrado tendrá la posibilidad de cambiar el idioma en el que la aplicación se

Tabla con formato

Con formato: Normal

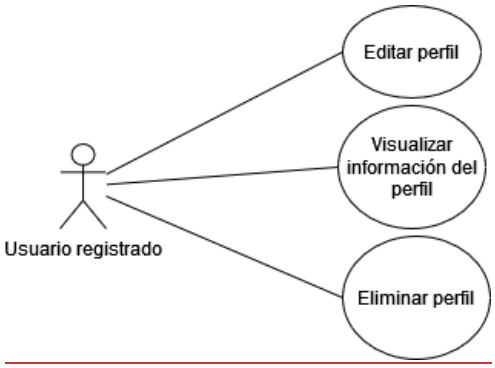
Tabla con formato

Con formato: Párrafo de lista, Sangría: Izquierda: 1,27 cm

muestra. Podrá elegir entre español o inglés.

Con formato: Normal, Sangría: Izquierda: 0 cm

5.2.3.2 Gestionar perfil

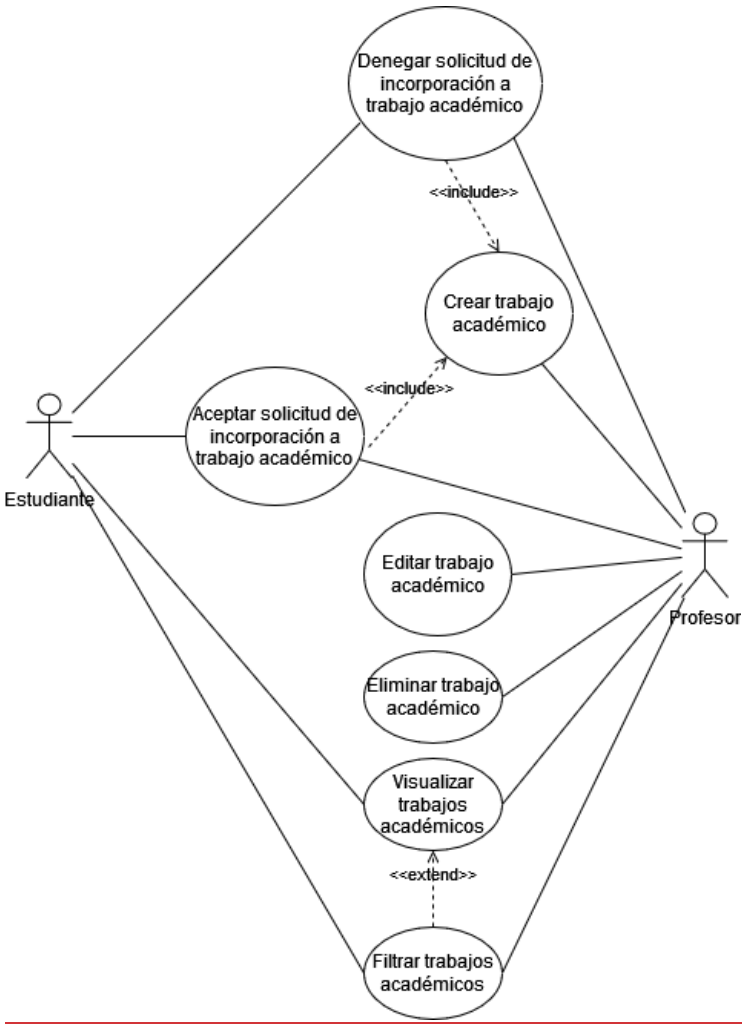


Nombre del Caso de Uso
Editar perfil
Descripción
Un usuario registrado tendrá la posibilidad de editar su perfil modificando los datos introducidos durante el registro. También podrá incluir una foto de perfil de manera opcional.

Nombre del Caso de Uso
Visualizar información del perfil
Descripción
Un usuario registrado tendrá la posibilidad de ver la información de su perfil.

Nombre del Caso de Uso
Eliminar perfil
Descripción
Un usuario registrado tendrá la opción de eliminar su perfil. El sistema debe hacer que su cuenta sea eliminada impidiendo así que pueda volver a iniciar sesión.

5.2.3.3 Gestionar trabajos académicos



Nombre del Caso de Uso
Crear trabajo académico
Descripción
El profesor será el encargado de crear los trabajos académicos que requiera. Para ello deberá completar los siguientes datos. • Título • Descripción • Categoría (TFG, TFM, Tesis doctoral). • Lista de estudiantes asociados • Lista de profesores asociados

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

- Curso

Los datos de título, descripción, categoría y curso son obligatorios. La lista de profesores y alumnos asociados es opcional, se podrá editar.
Crear un trabajo académico generará solicitudes para incorporarse a los usuarios pertenecientes a la lista de estudiantes asociados y a la lista de profesores asociados.

Nombre del Caso de Uso
Editar trabajo académico
Descripción
El profesor que ha creado el trabajo académico podrá editarlo en cualquier momento, pudiendo así modificar cualquiera de los datos introducidos en la creación del trabajo.

Nombre del Caso de Uso
Visualizar trabajos académicos
Descripción
Los estudiantes y profesores asignados a trabajos académicos deberán poder visualizar estos junto a su información general.

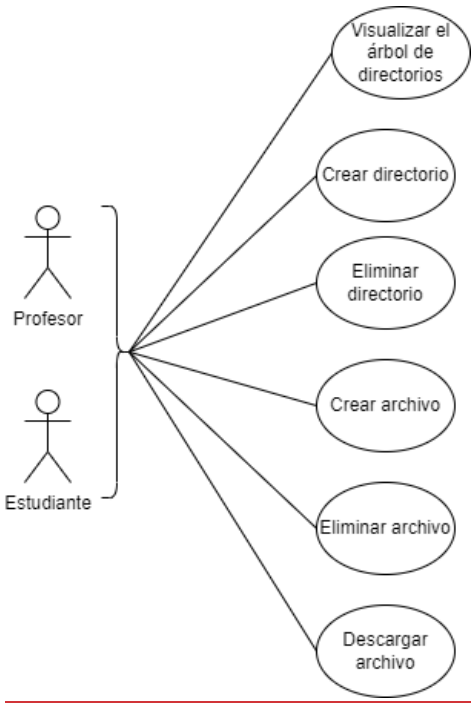
Nombre del Caso de Uso
Filtrar trabajos académicos
Descripción
Los estudiantes y profesores podrán filtrar la lista de trabajos académicos a visualizar, para ello podrán filtrar por categoría de trabajo (TFM, TFG, Tesis doctoral) y por curso.

Nombre del Caso de Uso
Eliminar trabajo académico
Descripción
El profesor que ha creado un trabajo académico podrá eliminarlo cuando requiera.

Nombre del Caso de Uso
Aceptar solicitud de incorporación a trabajo académico
Descripción
Los estudiantes y profesores invitados a participar en un trabajo académico podrán aceptar la solicitud de incorporación. Acto seguido pasarán a pertenecer al trabajo académico.

Nombre del Caso de Uso
Denegar solicitud de incorporación a trabajo académico
Descripción
Los estudiantes y profesores invitados a participar en un trabajo académico podrán denegar la solicitud recibida para incorporarse, no pasarán a formar parte del trabajo académico.

5.2.3.4 Gestionar archivos



Nombre del Caso de Uso
Visualizar árbol de directorios
Descripción
Los estudiantes y los profesores asignados a un trabajo académico podrán visualizar el árbol de directorios que contiene los ficheros subidos y relacionados con dicho trabajo académico. Podrán visualizar el nombre, tamaño y fecha de modificación de cualquier directorio y archivo.

Nombre del Caso de Uso
Crear directorio
Descripción
Los estudiantes y profesores asignados a un trabajo académico podrán crear directorios y subdirectorios para almacenar archivos. El límite de subdirectorios dentro del árbol de jerarquía será de cinco. No se permitirá crear dos directorios con el mismo nombre en el mismo nivel del árbol de directorios.

Tabla con formato

Nombre del Caso de Uso
Eliminar directorio
Descripción

Los estudiantes y los profesores asignados a un trabajo académico podrán eliminar directorios. Se comprobará antes de eliminar que el directorio esté vacío y no contenga nada en su interior. En caso de que el directorio tenga algún otro subdirectorio o archivo incluido dentro de él no se permitirá la eliminación.

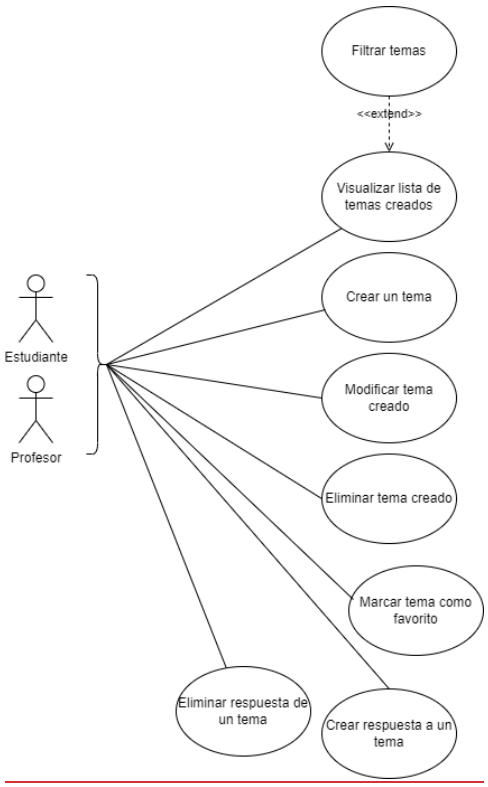
Nombre del Caso de Uso
Subir archivo
Descripción
Los estudiantes y los profesores asignados a un trabajo académico podrán subir archivos desde su ordenador a la aplicación. No se permitirá subir dos archivos con un mismo nombre a un mismo directorio.

Nombre del Caso de Uso
Eliminar archivo
Descripción
Los estudiantes y los profesores asignados a un trabajo académico podrán eliminar cualquier archivo contenido dentro de los directorios de un trabajo académico.

Nombre del Caso de Uso
Descargar archivo
Descripción
Los estudiantes y los profesores asignados a un trabajo académico podrán descargar en su ordenador cualquiera de los archivos subidos a un trabajo académico.

Con formato: Normal

5.2.3.5 Gestionar foro



Nombre del Caso de Uso
Visualizar lista de temas creados
Descripción
El usuario podrá visualizar la lista de temas del foro específico de cada trabajo académico. Dicha lista debe mostrar la información del tema, el número de respuestas y tener la opción de ser marcado como favorito.

Nombre del Caso de Uso
Filtrar temas
Descripción
El usuario podrá ordenar la lista de temas a visualizar por: • Más recientes: el foro debe mostrar primero los temas cuya fecha de creación es más nueva. • Menos recientes: el foro debe mostrar primero los temas cuya fecha de creación es más antigua. • Favoritos: el foro debe mostrar primero los temas que han sido marcados como favoritos.

Con formato: Izquierda, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Nombre del Caso de Uso
Crear tema
Descripción
El usuario podrá crear un nuevo tema, para ello deberá introducir un título obligatorio y un contenido del tema, que deberá ser un mensaje con el contenido a elegir por el usuario.

Nombre del Caso de Uso
Modificar tema
Descripción
El usuario tendrá la posibilidad de modificar un tema creado previamente, pudiendo así cambiar el contenido de este, el título no será modificable. Solo podrá modificar temas creados por él mismo.

Nombre del Caso de Uso
Eliminar tema creado
Descripción
El usuario podrá eliminar cualquiera de los temas que ha creado. En caso de que el usuario sea un estudiante, solo podrá eliminar los temas que le pertenezcan. En caso de que el usuario sea un profesor, podrá eliminar cualquier tema del foro.

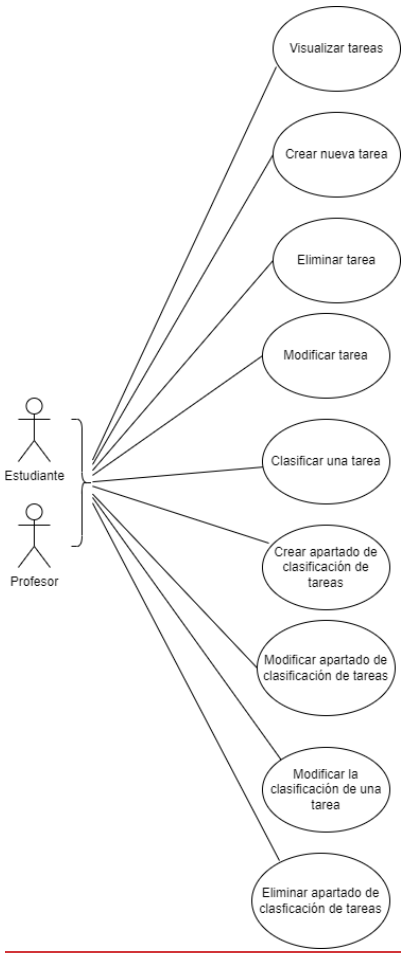
Nombre del Caso de Uso
Marcar tema como favorito
Descripción
El usuario podrá especificar que cualquiera de los temas del foro sea favorito. El tema deberá reflejar esta característica que servirá para el filtrado.

Nombre del Caso de Uso
Crear respuesta a un tema
Descripción
El usuario podrá enviar respuestas dentro de los temas que se encuentren creados en el foro. Dicha respuesta tendrá un contenido textual.

Nombre del Caso de Uso
Eliminar respuesta de un tema
Descripción
El usuario podrá eliminar cualquiera de las respuestas que haya publicado. En caso de que el usuario sea un profesor, podrá eliminar tanto sus respuestas como las de otros estudiantes y profesores. En caso del estudiante solo podrá eliminar las respuestas que haya creado él con anterioridad.

Con formato: Normal

5.2.3.6 Gestionar tareas



Nombre del Caso de Uso
Visualizar tareas
Descripción
El estudiante y el profesor deben poder visualizar la lista de las tareas creadas para un trabajado académico. Deben poder visualizarse las tareas clasificadas y las no clasificadas.

Nombre del Caso de Uso
Crear nueva tarea
Descripción
El profesor y el estudiante deben poder crear nuevas tareas, especificando: • Título • Descripción

- Responsables (opcional)
- Fecha de inicio (opcional)
- Fecha de fin (opcional)

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Nombre del Caso de Uso
Eliminar tarea
Descripción
El estudiante y el profesor podrán eliminar tareas creadas con anterioridad. En este caso tanto el alumno como el profesor podrán eliminar cualquiera de las tareas existentes para el proyecto.

Nombre del Caso de Uso
Modificar tarea
Descripción
El estudiante y el profesor deben poder editar y modificar los atributos especificados durante la creación de las tareas.

Nombre del Caso de Uso
Clasificar una tarea
Descripción
El estudiante y el profesor podrán clasificar las tareas dentro de los distintos apartados de clasificación de tareas.

Nombre del Caso de Uso
Crear apartado de clasificación de tareas
Descripción
El estudiante y el profesor podrán crear nuevos apartados para clasificar tareas. Para crear el apartado solo deberán especificar de manera obligatoria un título. Por defecto al crear un nuevo trabajo académico se crearán automáticamente los siguientes apartados: • Tareas nuevas • Tareas en progreso • Tareas bloqueadas • Tareas terminadas

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Nombre del Caso de Uso
Modificar apartado de clasificación de tareas
Descripción
El estudiante y el profesor deben poder modificar el título de cualquier apartado de clasificación de tareas.

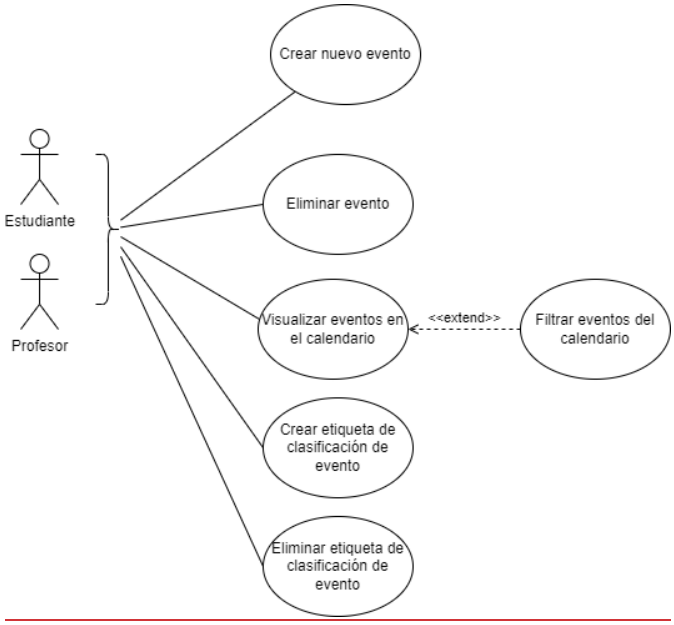
Nombre del Caso de Uso
Modificar la clasificación de una tarea
Descripción

La clasificación de las tareas debe poder ser cambiada tanto por el alumno como el profesor entre los distintos apartados de clasificación de tareas.

Nombre del Caso de Uso
Eliminar apartado de clasificación de tareas
Descripción
El alumno y el profesor deben poder eliminar cualquiera de los apartados de clasificación de tareas. No se podrá eliminar un apartado de clasificación de tareas que contenga tareas clasificadas en dicho apartado.

Con formato: Normal

5.2.3.7 Gestionar calendario



Nombre del Caso de Uso
Crear nuevo evento
Descripción
El alumno y el profesor deben poder crear nuevos eventos en el calendario, para ello deben indicar: título del evento (obligatorio), descripción y el grupo de etiquetas (previamente creadas) para clasificar el evento, estas últimas son opcionales. También deberán indicar la fecha y la hora del evento.

Nombre del Caso de Uso
Eliminar evento
Descripción

El alumno y el profesor deben poder eliminar eventos que han creado previamente. En el caso del alumno, podrá eliminar únicamente los eventos que él haya creado. En el caso del profesor, podrá eliminar cualquier evento del calendario del trabajo.

Nombre del Caso de Uso
Visualizar eventos en el calendario
Descripción
El alumno y el profesor deben poder visualizar los eventos asociados con un trabajo académico que hayan sido creados previamente. Por defecto, se mostrará también en el calendario las tareas en las fechas de inicio y fin.

Nombre del Caso de Uso
Filtrar eventos del calendario
Descripción
El alumno y el profesor deben poder filtrar la lista de eventos a visualizar de acuerdo con las siguientes condiciones: • Mostrar únicamente los eventos creados por el usuario. • Mostrar los eventos relacionados con una o varias etiquetas. • Mostrar únicamente las fechas de inicio y fin de las tareas. • No mostrar nada.

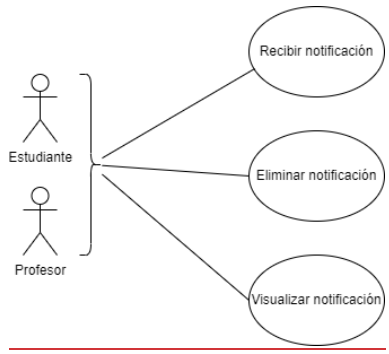
Nombre del Caso de Uso
Crear etiqueta de clasificación de evento
Descripción
El alumno y el profesor deben poder crear etiquetas para clasificar eventos, por defecto el sistema debe incluir las etiquetas: “Reunión”, “Tarea” y “Entrega”.

Nombre del Caso de Uso
Eliminar etiqueta de clasificación de evento.
Descripción
El alumno y el profesor podrán eliminar etiquetas de clasificación de eventos. Si una etiqueta de clasificación se elimina, todos los eventos con dicha etiqueta deben eliminar la etiqueta de su contenido.

Con formato

Con formato: Normal, Sin viñetas ni numeración

5.2.3.8 Notificaciones

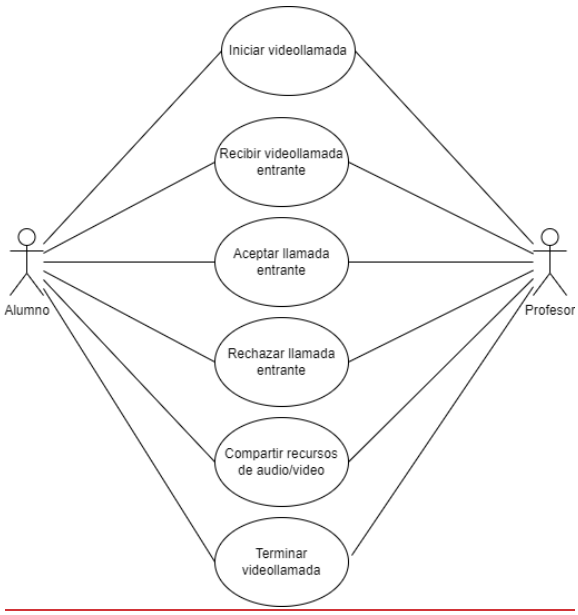


Nombre del Caso de Uso
Recibir notificación.
Descripción
Tanto el alumno como el profesor deben recibir notificaciones cuando se producen alguno de los eventos especificados en los requisitos de notificaciones 5.2.1.1.3.

Nombre del Caso de Uso
Eliminar notificación.
Descripción
Tanto el profesor como el estudiante deben poder eliminar notificaciones recibidas.

Nombre del Caso de Uso
Visualizar notificación.
Descripción
Tanto el profesor como el alumno deben poder visualizar las alertas de notificaciones.

5.2.3.9 Videollamadas



Nombre del Caso de Uso
Iniciar videollamada.
Descripción
Tanto el alumno como el profesor podrán iniciar una videollamada. El otro usuario debe estar conectado en ese momento en la aplicación para que esta pueda llevarse a cabo.

Nombre del Caso de Uso
Recibir videollamada entrante.
Descripción
Tanto el alumno como el profesor cuando otro usuario inicie una videollamada con ellos, deberán recibir la información de que están siendo llamados por otro usuario.

Nombre del Caso de Uso
Compartir recursos de audio/video.
Descripción
Tanto el alumno como el profesor en cualquier momento de la llamada podrán dejar de compartir o empezar a compartir recursos de audio y video de su dispositivo.

Nombre del Caso de Uso
Terminar videollamada.
Descripción

Tanto el alumno como el profesor en cualquier momento de la llamada podrán colgar la llamada, poniendo fin a esta.

Con formato: Normal

5.4.3

Un caso de uso es una descripción del comportamiento de un sistema cuando responde a una petición que se origina fuera del mismo (por parte de los actores del sistema). Un caso de uso describe "quién" puede hacer "qué" con el sistema. La creación de casos de uso se utiliza para capturar los requisitos funcionales del sistema y su comportamiento, haciéndose esto último a través de los escenarios que forman parte del mismo.

Un caso de uso se representa como una secuencia de pasos simples iniciadas por un actor de los identificados en la sección anterior, el cual interactúa con el sistema para llevar a cabo algún objetivo específico. A modo de ejemplo se pueden citar casos de uso como "*Hacer una búsqueda*", "*Enviar datos de Compras*", "*Abandonar Operación en curso*", etc. Podemos encontrar ejemplos de caso de uso adicionales y de los diagramas *UML* que representan gráficamente los casos de uso aquí:

- http://www.agilemodeling.com/artifacts/useCaseDiagram.htm
- http://www.visualcase.com/kbase/use_case_sample.htm

Cada caso de uso puede generar uno o varios escenarios, que describen detalladamente cada posible vía para alcanzar el objetivo del caso de uso y que serán descritos en una sección posterior. A la hora de describir casos de uso no es necesario recurrir a terminología técnica, sino que es posible usar lenguaje cercano al usuario final. Además, un caso de uso puede derivar en más subcasos de uso si es necesario describir el comportamiento del sistema con un detalle mayor.

Esta sección debe incluir el clásico diagrama de casos de uso de la aplicación. Si el número de casos de uso fuese muy elevado, se pueden crear múltiples diagramas para que quede todo de forma más clara. A modo de ejemplo, se presentan los siguientes diagramas de un sistema para la creación y corrección de exámenes de tipo test:

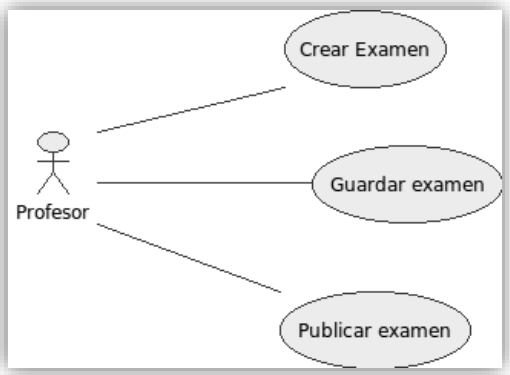


Figura 5.1. Ejemplo de caso de uso 1

~~5.5~~5.3 Identificación de los Subsistemas en la Fase de Análisis

~~El objetivo de esta sección es analizar el sistema para poder descomponerlo en sistemas más pequeños (subsistemas) que faciliten su posterior análisis.~~En este apartado se realizará una descripción de los distintos subsistemas que van a formar el sistema final y de la comunicación que se va a establecer entre ellos.

5.3.1 Descripción de los Subsistemas

El sistema estará fundamentalmente compuesto por tres subsistemas principales: el cliente de Angular, el servidor NodeJS y el módulo de base de datos encargado de mantener la persistencia de los datos del sistema.

- **Cliente Angular:** es el subsistema que actuará como cliente de la aplicación. Será el encargado de exponer la vista de la aplicación al cliente, por lo que servirá al cliente del código HTML, CSS y Javascript necesario, en términos generales expondrá la interfaz gráfica de usuario. Internamente se compondrá de los distintos elementos que contiene una aplicación Angular: componentes, servicios, módulos, directivas etc.
- **Servidor NodeJS:** será el encargado de exponer la API REST con la que el cliente se comunicará mediante peticiones a las rutas de dicha API. Contendrá la lógica necesaria para poder interactuar con la aplicación y servirá al subsistema de cliente de todos los recursos que necesite.

~~5.5.1~~• **Persistencia:** será el subsistema encargado de conectarse con la base de datos MongoDB para poder servir al servidor de todo aquello que requiera interactuar con esta (altas, bajas, modificaciones etc.). Estará formado por los modelos y el código Javascript necesario para poder llevar a cabo todas estas acciones.

~~En esta sección debemos enumerar todos los subsistemas que identifiquemos inicialmente en la aplicación. Los subsistemas son agrupaciones de paquetes y clases que tienen un objetivo propósito común. Ejemplos de subsistemas pueden ser todas las clases que manejen la base de datos (subsistema "base de datos"), clases que agrupen un conjunto de servicios relacionados, clases del cliente de esos servicios, etc.~~

La comunicación entre el cliente Angular y el servidor NodeJS se hará mediante peticiones a La comunicación entre el cliente Angular y el servidor NodeJS se hará mediante peticiones a La comunicación entre el cliente Angular y el servidor NodeJS se hará mediante peticiones a La comunicación entre el cliente Angular y el servidor NodeJS se hará mediante peticiones a una API REST. El servidor expondrá una API que tendrá distintas rutas o endpoints, el cliente se encargará de tener servicios que realicen peticiones asíncronas a dichas rutas de la API. Las peticiones realizadas por el cliente serán de

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Fuente: Negrita

tipo HTTP: POST, GET, PUT… El servidor enviará las respuestas a dichas peticiones utilizando códigos de respuesta: 200 OK, 404 Not found, 403 Forbidden…

La aplicación deberá estar preparada para soportar que el cliente y el servidor estén desplegados en la misma máquina o que el cliente y el servidor estén en máquinas distintas, por lo que la configuración de la API REST deberá adaptarse a esto.

La comunicación entre servidor y el subsistema de persistencia se realizará mediante comunicación interna. De la misma manera, el subsistema de persistencia deberá conectarse con la base de datos MongoDB, por ello deberá soportar que la base de datos esté desplegada en la misma máquina o en otra máquina de otra red. Para poder conectarse desde este subsistema a la base de datos se utilizará la biblioteca Mongoose.

~~5.6~~5.4 Diagrama de Clases Preliminar del Análisis

En la fase de análisis podemos identificar ya posibles clases del sistema, a partir de los casos de uso y subsistemas ya vistos. Estas clases no tienen por qué ser definitivas ni contener todas las operaciones y atributos que finalmente tendrán (sólo los que sean obvios según los requisitos y casos de uso), pero sirven como punto de partida para el esquema completo que se desarrollará en la fase de diseño. Por ello, estas clases no deben tener nombres de operaciones con sus parámetros exactos, sino más bien una indicación de lo que deben hacer esas operaciones. Por ejemplo, en lugar de "`insertarUsuario(Usuario:u)`" es mejor poner simplemente "`insertar usuario`". La descripción de las clases se hará agrupándolas por el subsistema al que pertenecen.

~~5.6.1~~5.4.1 Diagrama de Clases

Previamente a describir las clases una por una, debemos incluir **un diagrama de clases global que muestre la relación entre todas ellas**. Se recuerda que esta es sólo la fase de análisis y no es necesario lograr un nivel de profundidad de detalle muy elevado, sino que lo que se busca es una idea aproximada (pero precisa) de cómo va a ser el sistema a construir. En este diagrama también pueden aparecer los subsistemas identificados anteriormente. A continuación se muestra un ejemplo:

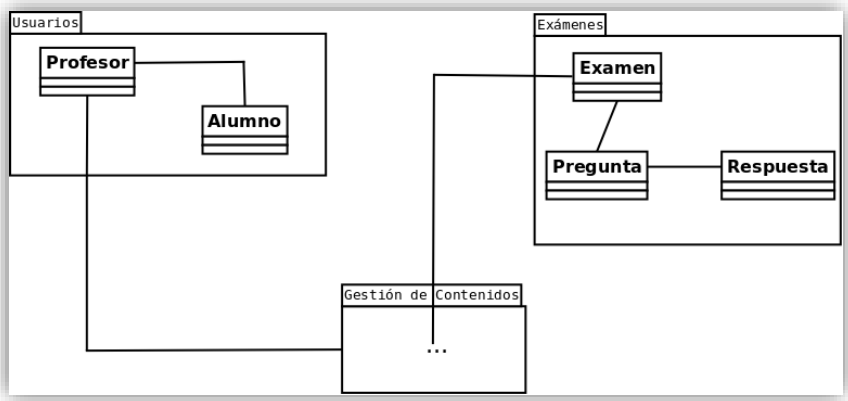


Figura 5.3. Diagrama de clases de ejemplo

~~5.6.2~~5.4.2 Descripción de las Clases

Las clases deberían estar organizadas por los subsistemas identificados en anteriormente, rellenando una tabla como la siguiente por cada clase del mismo:

~~5.6.2.1~~5.4.2.1 Subsistema 1

Nombre de la Clase
Descripción
Responsabilidades
Atributos Propuestos
NombreAtributo: Descripción de su propósito **NombreAtributo2**: Descripción de su propósito
Métodos Propuestos
NombreMétodo: Descripción de lo que hace **NombreMetodo2**: Descripción de lo que hace

~~5.6.2.2~~5.4.2.2 Subsistema 2

Nombre de la Clase
Descripción
Responsabilidades
Atributos Propuestos
NombreAtributo: Descripción de su propósito **NombreAtributo2**: Descripción de su propósito
Métodos Propuestos
NombreMétodo: Descripción de lo que hace **NombreMetodo2**: Descripción de lo que hace

~~5.7~~5.5 Análisis de Casos de Uso y Escenarios

En esta sección se describirán los casos de uso identificados anteriormente de forma detallada, a través de sus escenarios. Los escenarios describen las interacciones entre los usuarios y el sistema e incluyen información acerca de los objetivos, expectativas, motivaciones, acciones y reacciones que se llevan a cabo. Los escenarios deben reflejar la forma en la que un sistema se comporta y se suelen describir en lenguaje coloquial, intentando no recurrir excesivamente a tecnicismos para poder ser entendidos por el usuario final. La intención de los mismos es definir el comportamiento deseado del *software* de manera que complementen a los requisitos funcionales antes descritos. Si se está usando una metodología de desarrollo ágil, los escenarios se detallan como acciones breves del usuario.

Es también frecuente incluir escenarios que describan acciones erróneas o equivocadas que el programa debe tener en cuenta, para asegurar un comportamiento adecuado y seguro en todos los posibles casos. A la hora de incluir estos escenarios en la descripción de un sistema se deben tener en cuenta aquellos casos en los que el sistema pueda tener un posible problema de seguridad o de fiabilidad. Por ejemplo, casos en los que el usuario introduzca información errónea, se produzca algún error al hacer algún cálculo o bien temas relacionados con algunos requisitos no funcionales.

Como ya se ha dicho, los escenarios se generan a partir de los casos de uso identificados. Como mínimo debe existir un escenario primario o principal que describa el flujo normal de los eventos que transcurran en el caso de uso, es decir, lo que debe ocurrir normalmente cuando este se ejecuta. Por ejemplo, para un caso de uso "*Registro en el sistema*", la secuencia de pasos asociada al escenario principal del mismo "*El usuario se da de alta correctamente*" sería:

1. El sistema muestra la pantalla de logon.
2. El usuario introduce su nombre y clave de usuario
3. El sistema valida la información introducida
4. El usuario entra correctamente en el sistema.

Además, también existirán escenarios que describan caminos secundarios o alternativos que son variantes del principal mostrado anteriormente. Por ejemplo, para el caso de uso anterior unos posibles escenarios secundarios serían:

- **Escenario Alternativo 1**: Alta errónea porque el usuario ya existe
- **Escenario Alternativo 2**: Alta errónea porque la contraseña no cumple las especificaciones requeridas.
- **Escenario Alternativo 3**: Alta errónea porque faltan campos obligatorios en el formulario.
- etc.

Los casos de uso es frecuente que se vuelvan complejos y generen un gran nº de escenarios secundarios (pueden tener un gran nº de pasos y cada uno de ellos genera varios escenarios secundarios). Por ello, se usan diagramas de actividad o robustez para poder representar mejor esa complejidad.

Por otro lado, los escenarios alternativos no deben contemplar los errores a nivel de sistema (fichero no encontrado, error de conexión), sino que suelen incluirse en su propia sección de "Excepciones" para tratar de no repetir la misma información de errores entre escenarios. Los escenarios alternativos son más secuencias de pasos alternativas a la "normal" o "principal", pero que luego pueden estar relacionados con la misma. Por ejemplo, el escenario alternativo anterior "*Alta errónea porque faltan campos obligatorios en el formulario*" finalizará en "*Ir al paso 1 del escenario principal*" (volver a pedir los datos de login).

Bajo esta perspectiva, el número de escenarios que caben dentro de una aplicación medianamente compleja puede ser muy elevado. Entonces, ¿dónde poner el límite? ¿Cuáles deben ser representados? El analista debe ser capaz de seleccionar aquellos escenarios que aporten información útil al diseño. Por ejemplo, en toda aplicación web cabe considerar el escenario de "fallo de conexión". Sin embargo, si no se requiere un tratamiento específico para esta circunstancia, esa información es redundante dado que más que informar, despista al diseñador. Sin embargo, si queremos que, por ejemplo, para dar de alta un cargo en una actividad de la empresa ésta deba estar abierta, y en consecuencia que esto sea controlado por el sistema en desarrollo, sí deberemos considerar el escenario "Intento de asignación de cargo a actividad cerrada" dentro del caso de uso "Asignar cargo a actividad".

Una vez vista una pequeña descripción de los conceptos involucrados en esta sección, pasaremos a describir qué elementos deberían aparecer en la documentación final para contemplar todo lo dicho. Para ello se debe describir la secuencia de pasos de la que constan los escenarios y sus alternativas, clasificándolos según el caso de uso al que correspondan de una forma similar a la que se muestra en el ejemplo siguiente. Debe además tenerse en cuenta que no hay una plantilla estándar para describir los casos de uso y sus escenarios en una documentación, sino que es frecuente adaptar su descripción al proyecto que se está describiendo. A continuación se da un ejemplo de tabla para la descripción de los mismos, que puede adaptarse reduciéndose o ampliándose en función de las necesidades:

Nombre del caso de uso	
Precondiciones	Descripción de todas las condiciones que deben cumplirse para iniciar el caso de uso. Esto quiere decir que si el sistema no está en estado descrito por sus precondiciones, el comportamiento del caso de uso no está determinado.
Poscondiciones	Describe que cambios en el estado del software se producirán tras completar el caso de uso
Actores	Qué actores están involucrados en el caso de uso (quién lo inicia, quién lo termina)
Descripción	Se usará para capturar la esencia del caso de uso (su escenario principal), describiendo el contenido del mismo y sus operaciones
Variaciones (escenarios secundarios)	Aquí deben describirse todas las posibles variaciones contempladas sobre el escenario principal, es decir, la descripción de todos los escenarios secundarios identificados
Excepciones	Condiciones excepcionales o errores que puedan ocurrir en el escenario principal y/o los secundarios descritos antes
Notas	Cualquier aclaración necesaria que no se haya contemplado en los

	puntos anteriores

Para la documentación, los casos de uso complejos o de importancia elevada es mejor documentarlos con un diagrama de actividad o robustez además del texto que hagamos siguiendo la tabla anterior. De esta forma, se debería dividir la sección por cada caso de uso, y dentro de cada una de **ellas incluir primero un diagrama con la secuencia de pasos que contempla el mismo seguido de tablas como la mostrada antes, una por cada caso de uso**. A continuación se muestran unos ejemplos de esto para aclarar el contenido de esta sección.

5.5.1 Caso de Uso 1: Registrarse

Registrarse	
Precondiciones	
Poscondiciones	Debe crearse un nuevo usuario con id único en el sistema.
Actores	Lo inicia usuario no registrado y lo termina usuario registrado.
Descripción	El usuario no registrado: 1- Accederá a la pantalla de registro de la aplicación. 2- Rellenará la información necesaria del registro. 3- El sistema validará en el cliente y en el servidor los datos introducidos. 4- Accederá a la pantalla principal de la aplicación.
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el correo introducido por el usuario ya existe en la aplicación. Volver al paso 2 del escenario principal. • **Escenario alternativo 2:** alguno de los datos introducidos es erróneo. Volver al paso 2 del escenario principal.
Excepciones	• **Problema de conexión con el servidor:** indicar al usuario que ha ocurrido un error de conexión.

Con formato: Párrafo de lista, Numerado + Nivel: 1 + Estilo de numeración: 1, 2, 3, … + Iniciar en: 1 + Alineación: Izquierda + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Fuente: Negrita

Con formato: Fuente: Negrita

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Fuente: Negrita

Con formato: Párrafo de lista, Con viñetas + Nivel: 1 + Alineación: 0,63 cm + Sangría: 1,27 cm

Con formato: Fuente: Negrita

Con formato: Normal

Con formato: Normal

~~5.7.1~~5.5.33 Caso de Uso 1

NOTA: No debemos olvidarnos de incluir una explicación del diagrama con aquello que pudiese no quedar del todo claro en el mismo.

(El siguiente ejemplo de diagrama de robustez corresponde al escenario principal del caso de uso "Crear examen" dado en un ejemplo anterior)

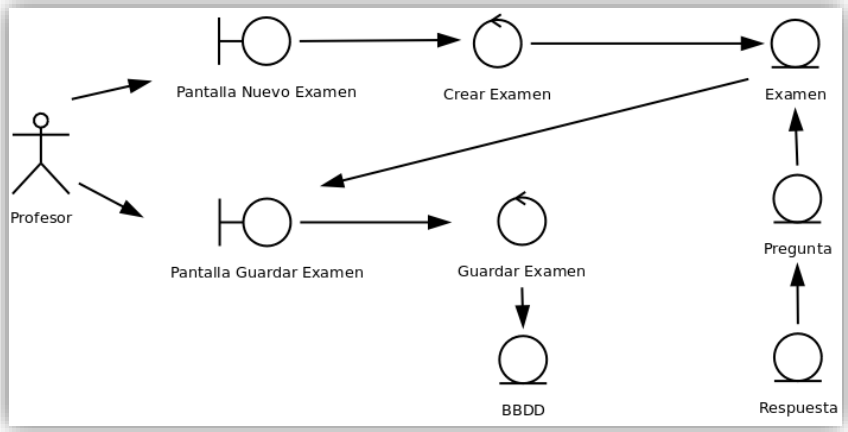


Figura 5.4. Descripción de las actividades de un escenario con un diagrama de robustez (I)

Crear Examen	
Precondiciones	El usuario debe estar validado con rol de profesor
Poscondiciones	Debe existir un nuevo examen con identificador único en el sistema
Actores	Iniciado y terminado por el profesor
Descripción	El profesor: 1. Accederá a la pantalla de nuevo examen 2. Rellenará la información necesaria para confeccionar el examen 3. Asignará las preguntas al examen escogiendo entre las existentes 4. Guardará el examen 5. Recibirá una notificación del éxito en la operación

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El identificador del examen indicado ya existe en el sistema
	  ○ Volver al paso 2 del escenario principal, manteniendo el resto de información en la pantalla
	• **Escenario Alternativo 2**: El usuario no encuentra la pregunta que busca entre las existentes
	  ○ Dar la posibilidad al usuario de editar nuevas preguntas, conectando con el escenario principal del caso de uso "Editar Nuevas preguntas"

	○ Actualizar la lista de preguntas disponible ○ Volver al paso 3 del escenario principal
Excepciones	• **La base de datos no está disponible**: No se pueden obtener preguntas ni guardar exámenes ○ Notificar un error asociado al problema encontrado
Notas	-

~~5.7.2~~5.5.34 Caso de Uso 2

(El siguiente ejemplo de diagrama de robustez corresponde al escenario principal del caso de uso "Registro en el sistema" mencionado en un ejemplo anterior)

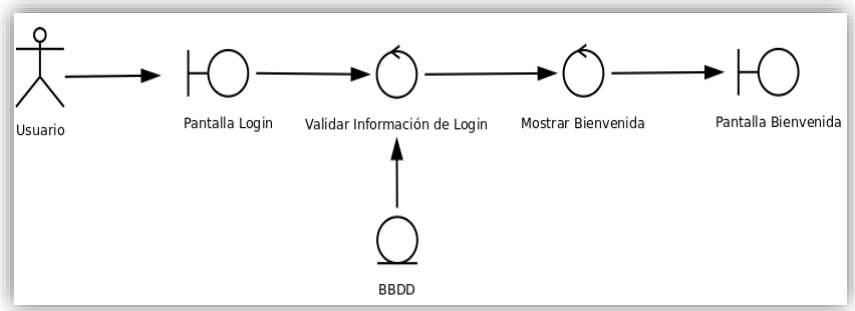


Figura 5.5. Descripción de las actividades de un escenario con un diagrama de robustez (II)

Registro en el Sistema	
Precondiciones	No
Poscondiciones	El usuario debe estar validado y con un rol asignado
Actores	Iniciado por un usuario de cualquier tipo de la aplicación finalizado por un usuario con el rol asociado a la información de logon introducida
Descripción	1. El sistema muestra la pantalla de logon. 2. El usuario introduce su nombre y clave de usuario 3. El sistema valida la información introducida 4. El usuario entra correctamente en el sistema
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Alta errónea porque faltan campos obligatorios en el formulario ○ Notificar el hecho al usuario ○ Volver al paso 1 del escenario principal • **Escenario Alternativo 2**: Usuario y/o contraseña inválidos ○ Notificar el hecho al usuario, sin dar detalles de lo que falta por seguridad

	○ Volver al paso 1 del escenario principal • **Escenario Alternativo 3**: Usuario y/o contraseña inválidos introducidos más de 5 veces seguidas ○ Notificar el hecho al usuario, sin dar detalles de lo que falta por seguridad ○ Volver al paso 1 del escenario principal, pero cambiando el contenido de la pantalla de login por un mensaje de forma que no se pueda volver a introducir información de login ○ Mantener esa invalidación para la IP del usuario que se conecta durante 30 minutos
Excepciones	• **La base de datos no está disponible**: No se pueden obtener nombres ni contraseñas de usuario ○ Notificar un error asociado al problema encontrado
Notas	En caso de que la BBDD no esté disponible, se debe mostrar un error pidiéndolo al usuario que lo intente en unos minutos en la pantalla de login

Podemos encontrar más información en: http://en.wikipedia.org/wiki/Scenario_(computing).

Por último, dado que los diagramas de robustez que se pueden usar para la descripción de la secuencia de pasos a dar en un escenario son un concepto relativamente nuevo y poco conocido, se ha incluido el siguiente anexo para su mejor comprensión.

Anexo: Diagramas de Robustez

El propósito principal de estos diagramas es poder analizar los pasos de cada caso de uso para validar que la lógica de negocio que modela es correcta, y que la terminología usada es consistente con la de otros casos de uso que se han analizado previamente. Por tanto, pueden usarse para comprobar si nuestros casos de uso son lo suficientemente robustos para representar a los requisitos del sistema construido. Otra ventaja de hacerlos es que permite identificar objetos o responsabilidades de objetos que son necesarias para soportar la lógica modelada por cada caso de uso, pero que se llaman fuera del mismo, sirviendo como puente hacia otros diagramas como diagramas de secuencia o diagramas de clase.

En un diagrama de robustez aparecen los siguientes conceptos:

- **Actores**: Los identificados en la sesión correspondiente.
- **Elementos limítrofes o *"Boundary Elements"*:** Representan elementos software como pantallas, informes, páginas HTML o interfaces del sistema con los que cada actor interactúa. También se les denomina "Elementos de Interface"

- **Elementos de Control o *"Control Elements"*:** Sirven como unión entre elementos *"Boundary"* y las entidades que veremos a continuación, implementando la lógica necesaria para gestionar los elementos y sus interacciones. También se les suele denominar elementos de proceso o controladores. Es importante entender que es posible implementar controladores dentro del sistema con elementos que no sean objetos (si son muy simples, pueden representarse con métodos de una entidad o clase *"Boundary"* simplemente).
- **Entidades o *"Entity Elements"*:** Estos son los tipos de entidad que se encuentran en el modelo conceptual ("Estudiante", "Aula", etc.).
- **Otros casos de uso (opcional):** Dado que en muchas ocasiones unos casos de uso invocan a otros, puede ser necesario representar esto en los diagramas de robustez.

El uso de estos diagramas será el correcto si vemos que nos reportan las siguientes ventajas:

- **Control de corrección:** Ayudan a estar seguro de que la descripción de cada caso de uso y sus escenarios es correcta y que no describe comportamientos no razonables o imposibles. En ocasiones puede ser necesario cambiar los nombres usados en la descripción del caso de uso con los nombres que aparecen en los diagramas de robustez.
- **Control de completitud:** Ayuda a asegurarnos que los casos de uso encajan con las operaciones que pretendemos hacer.
- **Identificación de objetos:** Es posible que nos hayamos olvidado de identificar algunos objetos en las partes del análisis hechas anteriormente y estos diagramas nos ayudarán a saberlo. También pueden ayudarnos a identificar discrepancias y conflictos entre nombres que hayamos asignado a las entidades anteriormente. En caso de encontrar estos fallos, debemos modificar los diagramas de manera acorde.
- **Ayuda a una fase preliminar del diseño:** Estos diagramas suelen ser más sencillos y fáciles de leer que los de secuencia.

Para una realización correcta de estos diagramas debemos tener en cuenta:

- Que las entidades que representemos en este diagrama DEBEN aparecer en el diagrama de clases de entidades del análisis hecho anteriormente.
- Que los diagramas tienen que describir los procesos de los casos de uso/escenarios que hayamos identificado. En caso de encontrar discrepancias, se debe identificar cual es el error y arreglarlo en aquel diagrama que corresponda.

Podemos encontrar más información en:

- http://www.agilemodeling.com/artifacts/robustnessDiagram.htm
- http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/SoftwareEngineering/UCDOM_robustness.html (Especialmente recomendado si este tipo de diagramas no se ha visto anteriormente, detallando además posibles errores a la hora de construir estos diagramas)

Por último, debemos recordar que si los escenarios representan operaciones muy sencillas o triviales no es necesario hacer un diagrama para los mismos. Tampoco tiene mucho sentido

desarrollar muchos diagramas casi iguales; si varias operaciones funcionan prácticamente de la misma forma, bastaría con indicar que el diagrama correspondiente a la operación X es muy similar al mostrado para la operación Y, indicando en texto las diferencias. Es también importante tener herramientas que nos permitan realizar fácilmente estos diagramas. Actualmente herramientas como *DIA* (instalable mediante el "centro de *software"* de versiones recientes de Ubuntu) o *Enterprise Architect* son ejemplos de herramientas que lo permiten.

~~5.8~~5.6 Relación Escenarios – Casos de Uso – Requisitos

Esta sección trata de mostrar de una manera resumida y visual, a través del uso de una tabla, la relación entre los Escenarios y los Casos de Usos explicados anteriormente.

Casos de Uso:	1.1	1.2	1.3	1.4	…
Escenario 1.1	X				
Escenario 1.2		X			
Escenario 2.1		X			
Escenario 3.1			X		
Escenario 3.2				X	
…					…

~~5.9~~5.7 Análisis de Interfaces de Usuario

A la hora de diseñar un interfaz de usuario, debemos cumplir con las normas de comunicación persona-máquina existentes, procurando que el interfaz sea usable, permita manejar el programa de manera eficiente y que no sea propenso a provocar errores en los usuarios. Esto debe hacerse así porque los diseños obedecen al resultado de hacer un diseñado centrado en el usuario, que simplemente nos lleva a simular en la pantalla el trabajo que realiza sobre una mesa o, en general, su entorno de trabajo existente hasta el momento. Un enlace que puede ser de ayuda a la hora de tomar determinadas decisiones a la hora de construir el interface de la aplicación es el siguiente: http://www.ambysoft.com/essays/userInterfaceDesign.html.

~~5.9.1~~5.7.1 Descripción de la Interfaz

En esta sección debemos crear la especificación de las interfaces entre el usuario y el sistema a construir, incluyendo todos los diferentes tipos de pantallas que van a existir, los cuadros de diálogo o los informes que le proporcionarán al usuario.

En este apartado también es importante identificar posibles grupos de usuarios para así aplicar las pantallas a dichos grupos, así como detallar otros aspectos, como lo que vamos a incluir en las pantallas para cumplir con normas de accesibilidad y usabilidad.

Para los distintos tipos de pantallas, se debe hacer un esquema que muestre la disposición de las mismas, que permita identificar donde irá cada elemento y las diferentes zonas de trabajo. Se muestra un ejemplo con este dibujo:

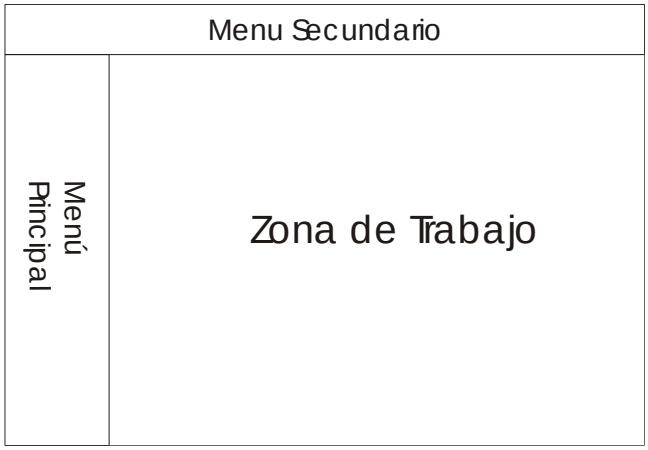
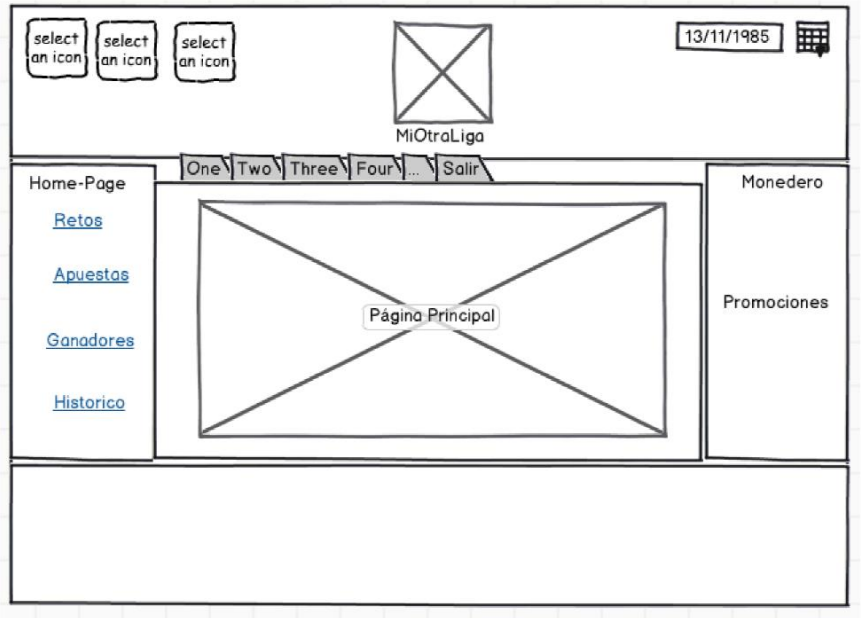


Figura 5.6. Boceto de una interfaz

Para hacer esto, se pueden utilizar programas que nos permitan hacer Mockups. Algunos ejemplos de estos programas son *Balsamiq Mockups* (https://balsamiq.com/products/mockups), *Draw.io* y *www.fluidui.com* (Este permite mockups interactivos).



Otra posible opción para este apartado es diseñar ya las pantallas definitivas sin funcionalidad, solo para ver cómo quedarán en el producto final (es decir, crear un **prototipo**), lo que tiene la ventaja de poder enseñarle al cliente el aspecto de la aplicación desde un primer momento. En

este caso, habría que incluir estos prototipos en el Archivo adjunto e indicar en esta sección la ruta de los prototipos, junto con la explicación correspondiente para cada caso.

~~5.9.2~~5.7.2 Descripción del Comportamiento de la Interfaz

En este apartado debemos especificar cosas como los convenios que vamos a crear para validar la entrada de datos de la aplicación, los mensajes de error que mostraremos y el tipo de ayuda que vamos a proporcionar al usuario.

~~5.9.3~~5.7.3 Diagrama de Navegabilidad

En esta sección incluiremos un diagrama que muestre la navegación que habrá entre las pantallas del programa y su relación con las computaciones que tienen lugar en las mismas. Debemos mostrar solo las transiciones entre pantallas y no el contenido de cada pantalla en sí, ya que este diagrama tiene simbología especial para todos sus elementos. Podemos encontrar más información en:

- www.agilemodeling.com/artifacts/uiFlowDiagram.htm.

~~5.9.4~~5.7.4

~~5.10~~5.8 Especificación del Plan de Pruebas

En esta sección crearemos y diseñaremos el plan de pruebas de la aplicación y sus funciones, así como todos los mecanismos que utilizaremos para detectar errores y corregirlos ya en la fase de implementación.

Las pruebas contemplarán aspectos tanto de funcionalidad de la aplicación como de aspectos de los usuarios o clientes de la misma.

Se contemplarán hasta cinco tipos de pruebas:

- **Pruebas Unitarias**: Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código, o en este caso una clase individual que cumple con una función concreta. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. A partir de los casos de uso, los escenarios y clases vistos anteriormente, debemos desarrollar pruebas unitarias que consideremos necesarias y especificar los resultados que se espera encontrar una vez ejecutada la operación sobre cada una de ellas. Es conveniente tabular estas pruebas para su aplicación posterior.
- **Pruebas de Integración**: Las pruebas de integración comprenden verificaciones asociadas a grupos de componentes, verificando que éstos funcionan correctamente cuando estos son ensamblados para cumplir con una función concreta. Para ello, cada escenario debe probarse con el mayor número de entradas posibles (y relevantes) que sea posible, incluyéndose entradas con datos correctos y con datos incorrectos para probar que el sistema reacciona correctamente ante errores de los usuarios. Para elaborar estas pruebas debemos tener en cuenta las características de la aplicación.
- **Pruebas del sistema**: Las pruebas del sistema son pruebas de integración del sistema construido completo, que permiten probar el conjunto de todo el sistema y que sus relaciones con otros sistemas que necesite son correctas, verificando así que todas sus especificaciones funcionales y técnicas se cumplen.
- **Pruebas de Usabilidad**: Este tipo de pruebas determinan la satisfacción del cliente con el producto final. Podemos especificar aquí cuales de estos aspectos son los más importantes en la aplicación a crear y establecer unas pautas generales por las cuales queremos medir en qué medida hemos conseguido estos aspectos. No se trata de diseñar ya los mecanismos de prueba de estos aspectos, ya que eso se hará posteriormente.
- **Pruebas de Código**: Para determinar la existencia de código muerto. Se recomienda hablar si incluir este tipo de pruebas o no con el director del proyecto.

Para elaborar las pruebas debemos desarrollarlas a partir de los casos de uso y escenarios antes descritos, empleando tablas como las que se muestran a continuación. Para elaborarlas debemos tener en cuenta el escenario principal del caso de uso y sus posibles alternativas y excepciones. Estas tablas sirven como ejemplo para pruebas de integración o del sistema. Si resultase más sencillo, puede hacerse con pequeñas tablas independientes para cada caso. A continuación se muestra un pequeño ejemplo para un caso de uso:

Caso de Uso 1: Añadir Usuario	
Prueba	**Resultado Esperado**

Añadir un usuario no existente	El sistema posee un usuario más
Prueba	**Resultado Esperado**
Añadir un usuario que ya existe	El sistema no posee un usuario más y se muestra un dialogo notificándolo
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

6.1.1 Diagramas de Paquetes

Aunque el concepto de paquete se puede aplicar a varios elementos del modelo, lo más típico en un PFC es agrupar clases con ellos. Los paquetes agruparán clases que estén relacionadas con una determinada funcionalidad y este diagrama debe mostrar también las relaciones existentes entre dichos paquetes (por ejemplo, dos paquetes estarán relacionados si algunas de sus clases se usan entre ellas o se envían mensajes). En esencia se trata de mostrar la organización lógica de la aplicación (en contraposición con la organización física, que aparecerá en los diagramas posteriores), presentando como se agrupan las clases de dicha aplicación en paquetes y la relación existente entre los mismos. Más información sobre estos diagramas se puede encontrar en:

http://www.agilemodeling.com/artifacts/packageDiagram.htm

http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_packagediagram.html

Los diagramas de paquetes son especialmente útiles cuando el diagrama de clases de todo el sistema es demasiado grande para visualizarse correctamente. Podemos entonces hacer un diagrama de paquetes (cada paquete contendrá N clases del sistema) y luego detallar individualmente cada uno de ellos donde corresponda.

6.1.1.1 Paquete 1

Descripción de los paquetes del diagrama (que tipo de elementos contiene, para qué sirven los mismos,…).

6.1.1.2 Paquete 2

6.1.2 Diagramas de Componentes

Los diagramas de componentes muestran los diferentes componentes de un sistema y sus dependencias. Un componente representa un módulo de código físico. Muchas veces se suele identificar un componente con un paquete, pero no siempre es así ya que los componentes representan el empaquetado físico del código. Por tanto, una misma clase puede estar presente en varios componentes, pero definida en un único paquete.

Es también una práctica común el incluir estos diagramas dentro de los de despliegue (que veremos a continuación) para que la distribución física de las distintas partes de la aplicación (y los componentes que las forman) en las distintas máquinas disponibles quede más clara.

Para tener más información sobre este tema, se pueden consultar los siguientes enlaces:

http://www.agilemodeling.com/artifacts/componentDiagram.htm

http://www.visual-paradigm.com/VPGallery/diagrams/Component.html

http://www.ibm.com/developerworks/rational/library/dec04/bell/

Si se integra con el diagrama siguiente (el de despliegue), en las descripciones que hagamos de los elementos debemos describir también los componentes. En caso contrario, describiremos los componentes en esta sección.

6.1.3 Diagramas de Despliegue

El sistema creado puede estar compuesto por varios procesos *software* y máquinas que colaboran para llevar a cabo la tarea encomendada, y esta sección es la indicada para representar estos procesos y máquinas así como la relación existente entre ellos. Debemos ofrecer un diagrama (un ejemplo extremadamente simple de cómo podemos hacerlo se muestra a continuación) y posteriormente una descripción de que es cada parte de la aplicación y su función (relacionándolo con las clases y paquetes vistos en el análisis).

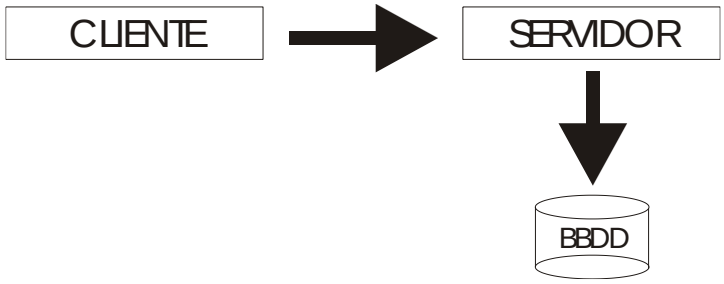


Figura 6.1. Ejemplo simple de arquitectura del sistema

Aunque el nivel de detalle puede variar en función del sistema diseñado, los elementos mostrados deben dejar clara la distribución de la aplicación. Podemos encontrar más información al respecto en los siguientes enlaces:

http://www.agilemodeling.com/artifacts/deploymentDiagram.htm

http://www.visual-paradigm.com/VPGallery/diagrams/Deployment.html

6.1.3.1 Elemento 1

Descripción de los elementos del diagrama anterior.

6.1.4

6.2 Diseño de Clases

En esta sección representaremos diagramas que muestren los paquetes (y la relación existente entre ellos) y las clases que formarán parte de la implementación final del sistema (incluyendo también las relaciones existentes entre ellas), explicando todo aquel dato acerca de la utilidad de los mismos y justificando todo aquello que consideremos necesario.

- Debemos extraer la información de métodos y atributos a incluir en cada clase (así como las relaciones existentes entre ellas) de los diagramas que hemos desarrollado en el punto anterior.
- También debemos mostrar las clases adicionales que pueden ser necesarias en función de lo que hemos desarrollado anteriormente.
- Debemos mostrar todas las asociaciones y agregaciones (en general, relaciones) que necesitemos, en función de dichos diagramas.
- El conjunto de atributos de las clases debemos crearlo según los métodos, relaciones y en general de las necesidades de cada clase del sistema.
- La jerarquía de clases debemos pensarla de acuerdo a las necesidades de cada subsistema y aplicación.
- Debemos aclarar todos aquellos aspectos que no queden claros en el diagrama, usando notas.
- En general, debemos tener en cuenta que el diseño es una evolución del análisis, por lo que las clases que incluyamos en el diseño deben corresponderse razonablemente con todo el trabajo hecho en esa parte del documento.

6.2.1 Diagrama de Clases

Es necesario mostrar por lo menos un diagrama global de clases. Si el diagrama de clases global fuese muy grande, podemos dividir el diagrama en varios diagramas más pequeños según la estructura de la aplicación, mostrando el principal abreviado tal y como se explico anteriormente en los diagramas de paquetes. Otra opción es hacer que la hoja que tenga el diagrama se imprima en otro formato que nos proporcione más espacio (en horizontal, A3,…).

6.3 Diagramas de Interacción y Estados

Esta sección se usará, entre otras cosas, para evolucionar y detallar los diagramas de robustez que hemos desarrollado en el análisis usando diagramas de interacción y de estados. La estructura a seguir es la de incluir el diagrama en sí (dibujo) y luego hacer una lista explicando cada uno de los pasos existentes en dicho diagrama que lo requieran. Los diagramas deben incluir nombres de clases, métodos y parámetros "reales", con la intención de que puedan trasladarse directamente a la implementación del sistema (el objetivo de un buen diseño). Por este motivo no debemos escatimar en detalles a la hora de desarrollarlos, ya que se supone que de estos diagramas podremos extraer directamente la implementación de la aplicación.

Cuando se menciona que los diagramas del diseño son una evolución de los del análisis significa que podemos aprovechar el trabajo ya realizado para su creación, adaptando la información en los diagramas correspondientes del análisis para desarrollar los nuevos diagramas (se debe lograr coherencia entre las entidades desarrolladas en ambas fases).

En cuanto al número de diagramas a contemplar, al igual que se dijo anteriormente no es necesario incluir en el diseño todos y cada uno de los posibles diagramas, sino que podremos ahorrar extender aquellos en los que:

- Las operaciones sean muy simples (en cuyo caso se puede optar por no decir nada al respecto o colocar simplemente un texto explicativo).
- La operación sea muy similar o idéntica a una que ya está desarrollada en esta sección. En este caso lo mejor es poner en el diagrama original a que otros casos se aplica o representa y no repetirlo.

6.3.1 Caso de Uso 1.1

6.3.1.1 Diagramas de Interacción (Comunicación y Secuencia)

Diagramas de secuencia de diseño correspondiente al escenario principal del caso de uso y a aquellos alternativos que merezcan ser desarrollados (estableciendo un paralelismo con la nomenclatura usada en el análisis). Podemos encontrar más información en:

http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/SoftwareEngineering/UCDOM_interaction.html

En el caso de que creamos necesario mostrar el comportamiento de varios objetos que colaboran para la consecución de un determinado fin común, debemos crear un diagrama de comunicación de objetos que represente este comportamiento. Podemos encontrar más información acerca de los mismos en esta dirección:

http://www.agilemodeling.com/artifacts/communicationDiagram.htm

6.3.1.2 Diagramas de Estados de las Clases

En caso necesario, si es conveniente dar más detalles de cómo se comporta un determinado escenario, podemos incluir diagramas de estados de las clases involucradas en el mismo para indicar las distintas fases por las que una clase pasará durante el mismo. Podemos encontrar más información sobre estos diagramas en:

http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm

6.3.2 Caso de Uso 1.2

6.4 Diagramas de Actividades

Si hay alguna operación o funcionalidad dentro del sistema que merezca la pena destacar se documentará mediante un diagrama de actividades. Puede haber tantos como consideréis necesarios. Más información acerca de estos diagramas se puede encontrar en estos enlaces:

http://dn.codegear.com/article/31863#activity-diagrams

http://www.agilemodeling.com/artifacts/activityDiagram.htm

6.5 Diseño de la Base de Datos

Esta sección es la indicada para describir todo lo relativo al sistema de gestión de bases de datos que vamos a usar en nuestra aplicación (si usamos alguno). Esto incluye hablar del sistema en sí, la versión utilizada, las clases empleadas al usarlo, como se integra en el sistema y el diagrama E-R de las tablas y relaciones creadas, todo ello en los apartados que se muestran a continuación.

6.5.1 Descripción del SGBD Usado

6.5.2 Integración del SGBD en Nuestro Sistema

6.5.3 Diagrama E-R

6.6 Diseño de la Interfaz

Esta sección debe mostrar ya la interfaz definitiva de la aplicación (que evidentemente deberá ser una evolución del diseño mostrado en el análisis) y las diferentes partes de las que consta. Como es más que probable que todas las pantallas tengan elementos de interfaz comunes, esta es la sección donde se va a hablar de cada uno de esos elementos, su propósito y su función (barra de menús, barras de estado, etc.).

En caso de no haber hecho un prototipo realista en la fase de análisis (es decir, mostrado el aspecto real del programa), debemos hacerlo aquí. En todo caso, cualquier cosa que quedase pendiente en la fase de análisis acerca de la descripción del interfaz debe concretarse aquí.

6.7 Especificación Técnica del Plan de Pruebas

El proceso de pruebas se extiende durante todo el proceso de construcción del software y por tanto en esta sección debemos describir cómo vamos a aplicar las pruebas diseñadas y especificar más otro tipo de pruebas que vamos a realizar al software.

No debemos olvidarnos mencionar entonces bajo que máquina (incluyendo sistema operativo y entorno de desarrollo) se realizan las pruebas, y en qué orden se realizan las diferentes pruebas contempladas, entre otros aspectos.

NOTA: Es perfectamente posible que, aunque un sistema conste de múltiples procesos independientes que colaboran entre sí (por ejemplo un cliente y un servidor), las pruebas se hagan en sólo una máquina (Ej.: cliente y servidor que se conectan a *localhost*).

6.7.1 Pruebas Unitarias

Esta sección contendrá la descripción de la aplicación de las pruebas unitarias que hemos descrito anteriormente (qué datos se van a introducir finalmente sobre qué clases, ahora que conocemos finalmente todas las clases que se van a implementar gracias al diseño), indicando cuando las vamos a realizar y que medidas tomaremos en caso de encontrar fallos. También debemos hablar de si hemos dividido las pruebas entre los diferentes subsistemas de alguna forma.

En este apartado debemos también describir las posibles **pruebas de regresión** que queramos implementar. Estas pruebas de regresión se deben automatizar para poder pasarlas periódicamente cada vez que hagamos cambios importantes en el sistema.

Se recomienda encarecidamente el empleo de una herramienta de automatización de estas pruebas unitarias para facilitar mucho esta labor. Posibles herramientas son *JUnit* (Java) o *NUnit*. Realmente esta sería la forma más adecuada de afrontar esta tarea.

6.7.2 Pruebas de Integración y del Sistema

Las pruebas funcionales y del sistema que hemos especificado en el análisis deben aplicarse para garantizar que el sistema funciona correctamente. Debemos describir así cómo y cuándo vamos a aplicar este tipo de casos de prueba dentro del sistema, así como las entradas y salidas de estas pruebas. No debemos olvidar que la aplicación de las pruebas y qué ocurre cuando se pasan irá en una sección posterior.

Debemos recordar pues que, tanto en esta sección como en la anterior, ya tenemos las pruebas diseñadas en la fase de análisis, y que ahora se trataría de describir cómo y cuando las vamos a aplicar, qué datos vamos a introducir en cada caso y qué vamos a hacer en caso de acierto o fallo de las pruebas. No se trata de repetir lo mismo de nuevo.

A modo de guía, debemos pensar que lo que diseñemos aquí debe ser directamente aplicable sobre el código de la aplicación.

6.7.3 Pruebas de Usabilidad y Accesibilidad

Dado que es posible que el lector de este documento no esté familiarizado con este tipo de pruebas, haremos una descripción un poco más en detalle de en qué consisten. Este tipo de pruebas determinan la satisfacción del cliente con el producto final y por tanto debemos darle mucha importancia a las mismas, no dejándolas de lado en ningún caso.

Las pruebas de usabilidad tratan de evaluar la aplicación en su funcionamiento real. Debemos realizar un conjunto de pruebas para verificar que las distintas partes del programa se pueden usar adecuadamente. Los objetivos son:

- Mejorar la calidad de la interacción de los usuarios con nuestra aplicación.
- Reducir el tiempo necesario para hacer las distintas tareas en la aplicación y de esta forma aumentar la productividad.

Debemos tener los siguientes elementos en cuenta, para describirlos completamente en esta sección antes de hacer las pruebas en sí:

- **Usuarios**: Distintos grupos de usuarios de la aplicación sobre los que vamos a realizar las pruebas. Cada usuario tendrá un perfil de uso distinto y tenemos que tenerlo en cuenta a la hora de hacer las pruebas. Por lo general es suficiente con efectuar las pruebas con un número reducido de usuarios (de 3 a 5). También debemos dejar bien claro, en caso de que los usuarios lo deseen, la profesión y la edad de los mismos.
- **Lugar de realización**: En un laboratorio o bien en la propia casa u oficina del cliente.
- **Metodología**: Describir que vamos a hacer en estas pruebas (el conjunto de pasos a seguir en las mismas).

Aunque en la siguiente sección se dan detalles acerca de cómo hacer cuestionarios para efectuar estas pruebas, estos no son la única herramienta para ello. Existen otros tipos de herramientas como pruebas basadas en las opiniones dadas de forma oral por los usuarios, heurísticas, etc. Dependiendo fundamentalmente del producto y para quien este destinado. No obstante, los cuestionarios porque son una herramienta adecuada para gran parte de los casos y no exigen muchos recursos. Se recomienda consultar al director de proyecto acerca de la mejor forma de hacer este tipo de pruebas en el proyecto concreto a desarrollar.

6.7.3.1 Diseño de Cuestionarios

Detallamos a continuación algunas pautas a seguir en el diseño de los cuestionarios que podamos necesitar, si se estima oportuno el uso de esta herramienta. Una norma muy importante a cumplir es que él tiempo necesario para rellenar los cuestionarios no deben superar los 15 minutos.

6.7.3.1.1 Cuestionario de Evaluación

Debemos elaborar un cuestionario para que los usuarios lo hagan y determinar así distintos aspectos del mismo y de su interacción con la aplicación. Los puntos a tocar son (esto es un esquema que puede ampliarse si se desea para adaptarlo a la aplicación en sí):

- **1º**: **Preguntas de carácter general** a través de las cuales se determine el tipo de usuario y su nivel de conocimiento informático.
- **2º**: **Actividades guiadas** para hacer con nuestra aplicación.
- **3º**: **Batería de preguntas cortas** con los distintos aspectos de la aplicación que se pretendan evaluar.
- **4º**: **Observaciones**, para que el usuario aporte todo lo que considere oportuno de nuestra aplicación.

6.7.3.1.2 Cuestionario para el Responsable de las Pruebas

Además, debemos desarrollar un cuestionario para que el responsable de las pruebas pueda anotar distintos aspectos que observe durante la realización de las pruebas. Un posible desarrollo de todos estos aspectos se describe a continuación.

6.7.3.2 Actividades de las Pruebas de Usabilidad

6.7.3.2.1 Preguntas de carácter general

Se muestra un esbozo de un posible cuestionario, que debemos desarrollar y adaptar a nuestras necesidades:

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar

2. Que sea intuitivo
3. Que sea rápido
4. Que tenga todas las funciones necesarias

6.7.3.2.2 Actividades guiadas

Hacer un compendio de actividades que se puedan hacer con la aplicación para que nuestros usuarios las hagan y comenten los problemas y dificultades que según su opinión presenta la aplicación (si existiesen). Debemos recoger estas opiniones en el documento. Posibles actividades a probar son:

- Añadir un usuario a la aplicación
- Eliminar un artículo de la aplicación
- …

6.7.3.2.3 Preguntas Cortas sobre la Aplicación y Observaciones

Un posible cuestionario de preguntas cortas (a desarrollar más en cada proyecto) es el siguiente:

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe donde está dentro de la aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?				
¿El tiempo de respuesta de la aplicación es muy grande?				
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				
Observaciones				
Cualquier comentario del usuario				

Evidentemente debemos extender este cuestionario de ejemplo para adaptarlo a nuestras necesidades.

6.7.3.2.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	
Tiempo en realizar cada tarea	
Errores leves cometidos	
Errores graves cometidos	
<Incluir otras cosas>	
…	
…	

Los resultados de las pruebas se deben analizar conjuntamente para así establecer una conclusión respecto a cuatro aspectos de usabilidad:

1. **Facilidad de aprendizaje**: Capacidad para aprender la funcionalidad de la aplicación desarrollada y desarrollar las tareas de manera adecuada, midiendo cuanto se tarda en hacer las distintas tareas.
2. **Eficiencia**: Cuanto mejora la labor de los usuarios por usar la aplicación respecto a lo que se hacía anteriormente.
3. **Errores**: Cuantos errores cometen los usuarios en las distintas tareas, lo que decrementa la usabilidad del mismo.
4. **Satisfacción del usuario**: Impresión general de los usuarios al usar la aplicación.

6.7.3.3 Pruebas de Accesibilidad

El grueso de las **pruebas de accesibilidad** será descrito en la parte de desarrollo de las pruebas. A nivel de diseño simplemente puede indicarse qué tipo de pruebas van a realizarse para el programa concreto, que normas *WCAG* van a seguirse u otras consideraciones generales acerca de las mismas, dentro de las posibles vías de actuación. *WAI* propone básicamente tres tipos de revisiones:

- **Revisión preliminar**: Identifica rápidamente problemas de accesibilidad de un sitio *Web*. Todos los procesos de esta revisión también tienen lugar en la siguiente revisión.
- **Evaluación de conformidad**: Permite indicar si un sitio web cumple con los estándares de accesibilidad (Ej.: *WCAG*). Puede encontrarse más información en: www.w3.org/WAI/eval, aunque en esta plantilla se intentarán dar pautas para hacer una evaluación adecuada.
- **Otras evaluaciones**: *WAI* también proporciona otra serie de documentos para evaluar aplicaciones en contextos específicos y para implicar a usuarios discapacitados en la evaluación. En el primer caso se tratan de documentos complementarios a los anteriores que describen consideraciones para la evaluación de sitios grandes y complejos durante el proceso de desarrollo y mantenimiento, o bien para sitios existentes cuyas páginas son generadas dinámicamente.

6.7.4 Pruebas de Rendimiento

El sistema desarrollado consumirá una determinada cantidad de recursos (memoria y tiempo de proceso) que debemos procurar que sean los menores posible. Por ello, la aplicación debe al menos medirse para ver cuántos recursos consume y se debe intentar eliminar posibles cuellos de botella en el rendimiento que se puedan presentar en uno o varios subsistemas de la misma. Debemos diseñar medios para hacer estas mediciones en nuestra aplicación.

Para esta tarea debemos pues medir el tiempo que tardan las operaciones más importantes que haga el sistema y comprobar si es posible mejorarlas de algún modo una vez que el sistema esté en funcionamiento. En este apartado debemos diseñar que actuaciones vamos a llevar a cabo para hacer estas pruebas y las actuaciones que vamos a llevar a cabo para mejorar el rendimiento en aquellos puntos en los que encontremos problemas. Una herramienta que podemos aplicar para este fin es la opción "*Tools – View Speed Report*" de las "*Web Developer Extensions*" de *Firefox*.

Una consideración a tener en cuenta es determinar qué actividades serán las más frecuentes y si su rendimiento es adecuado o no, ya que por esta vía conseguiremos una mejor optimización de la aplicación.

En el caso de que el programa deba tener algún requisito respecto al tiempo que debe tardar o a la memoria ocupada, debe tenerse especial cuidado en este aspecto.

Por último, también debemos diseñar pruebas de carga para determinar cómo responde el sistema con un alto número de usuarios o un gran volumen de datos. Para esta tarea puede ayudar una aplicación como *JMeter* (http://jakarta.apache.org/jmeter/).

Capítulo 7. Implementación del Sistema

7.1 Estándares y Normas Seguidos

Descripción breve de los estándares y normas que hayamos usado en nuestra aplicación a la hora de desarrollar su código y si nos hemos ocupado de validar que esos estándares se cumplan efectivamente.

7.2 Lenguajes de Programación

Descripción de los lenguajes de programación usados, su versión y distribución, los módulos o complementos de los mismos empleados, etc.

7.3 Herramientas y Programas Usados para el Desarrollo

Descripción de todas las herramientas de desarrollo (Eclipse, Visual Studio, etc.), sistemas adicionales existentes, complementos y otros productos software que necesitemos para la implementación de nuestro sistema. Debemos dejar claro que versión usamos, para qué y cómo interactuará con nuestro sistema.

7.3.1 Programa 1

7.3.2 Programa 2

7.4 Creación del Sistema

Todos los aspectos con los que nos hemos encontrado durante la implementación debemos describirlos aquí.

7.4.1 Problemas Encontrados

Enumeramos los problemas encontrados en el desarrollo y la solución que le hemos dado, si hubo alguno que merezca la pena destacar. **Se recomienda ir documentándolos cuando se encuentran para así conseguir una buena documentación de ellos y no olvidarnos de ninguno**.

7.4.1.1 Problema 1

7.4.1.2 Problema 2

7.4.2 Descripción Detallada de las Clases

En esta sección debemos decir la ruta o rutas (en caso de ser varios proyectos) donde se encuentra la/s documentación/es del código de nuestra aplicación dentro del Archivo adjunto. Esta deberá tener detalladamente todos sus atributos y métodos así como su descripción y función exacta, tal y como aparecen en el código de la aplicación, y debe incluir el autor de cada clase/fichero/módulo. Para hacerlo lo mejor posible, **debemos elaborar esta descripción a la vez que se programa** o como muy tarde, en la finalización de cada módulo, para así seguir la metodología de la forma adecuada y sernos de ayuda en el caso de que debamos tocar una clase o módulo que hace tiempo no tocamos y poder usar nuestra propia documentación para recordarnos que hacía y por qué. Una vez cerrado el desarrollo de la aplicación, se deberá de generar la documentación de la forma en que se crea más conveniente (HTML, Word, ODF,…).

Para esto se debe emplear un generador de documentación. Esto son herramientas que vuelcan las clases y sus comentarios a un formato que podamos usar y que contemple todos los aspectos vistos antes (es decir, que el resultado sea similar a la tabla mostrada más abajo). Ejemplos de estas herramientas son *Javadoc y Doxygen, entre otros (https://en.wikipedia.org/wiki/Comparison_of_documentation_generators)*. Hay IDEs y lenguajes de programación que ya cuentan con una propia o librerías para este fin.

En caso de ser un proyecto hecho en parejas, conviene crear una tabla con las clases, para así indicar quien ha sido el autor de cada clase. No obstante, la documentación del código debería de llevar el autor en ella.

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

Describir aquí el resultado de las pruebas unitarias que ya hemos diseñado y descrito anteriormente, así como los resultados obtenidos y las actuaciones que hemos llevado a cabo ante los errores y problemas encontrados.

8.2 Pruebas de Integración y del Sistema

Ejecutamos las pruebas funcionales ya diseñadas anteriormente y anotamos el resultado que obtenemos, comparándolo con el que especificamos anteriormente. Podemos hacerlo a partir de una tabla modificada de la anterior como ésta (si es más sencillo, puede hacerse con pequeñas tablas independientes para cada caso):

Caso de Uso 1.1: Añadir Usuario	
Prueba	**Resultado Esperado**
Añadir un usuario no existente	El sistema posee un usuario más
	Resultado Obtenido
	El sistema efectivamente posee un usuario más
Prueba	**Resultado Esperado**
Añadir un usuario que ya existe	El sistema no posee un usuario más y se muestra un dialogo notificándolo
	Resultado Obtenido
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

8.3 Pruebas de Usabilidad y Accesibilidad

8.3.1 Pruebas de Usabilidad

A partir de los cuestionarios que se diseñaron anteriormente y de los procedimientos explicados, mostramos aquí el resultado de aplicarlos sobre todos los usuarios que hayamos empleado para estas pruebas.

Es muy importante a su vez indicar los cambios producidos en la interfaz a partir de las sugerencias recogidas por los usuarios. Una forma adecuada de representar esto sería poner la pantalla inicial y poner la pantalla después de los cambios efectuados.

Además de los cuestionarios diseñados anteriormente, podemos pasar esta guía de usabilidad desarrollada por Yusef Hassan Montero [Hassan08] (http://www.nosolousabilidad.com/articulos/heuristica.htm):

Criterios	¿Cumplido?
Generales	
¿Cuáles son los objetivos del sitio web? ¿Son concretos y bien definidos? ¿Los contenidos y servicios que ofrece se corresponden con esos objetivos?	**SI**
¿Tiene una URL correcta, clara y fácil de recordar? ¿Y las URL de sus páginas internas? ¿Son claras y permanentes?	
¿Muestra de forma precisa y completa qué contenidos o servicios ofrece realmente el sitio web? El diseño de la página de inicio debe ser diferente al resto de páginas y cumplir la función de 'escaparate' del sitio.	
¿La estructura general del sitio web está orientada al usuario? Los sitios web deben estructurarse pensando en el usuario, sus objetivos y necesidades. La estructura interna de la empresa u organización, cómo funciona o se organiza no interesan al usuario.	
¿El *look & feel* general se corresponde con los objetivos, características, contenidos y servicios del sitio web? Ciertas combinaciones de colores ofrecen imágenes más o menos formales, serias o profesionales.	
¿Es coherente el diseño general del sitio web? Se debe mantener una coherencia y uniformidad en las estructuras y colores de todas las páginas. Esto sirve para que el usuario no se desoriente en su navegación.	
¿Es reconocible el diseño general del sitio web? Cuánto más se parezca el sitio web al resto de sitios web, más fácil será de usar.	
¿El sitio web se actualiza periódicamente? ¿Indica cuándo se actualiza? Las fechas que se muestren en la página deben corresponderse con actualizaciones, noticias, eventos...no con la fecha del sistema del usuario.	
Identidad e Información	
¿Se muestra claramente la identidad de la empresa-sitio a través de todas las páginas?	

El Logotipo, **¿es significativo, identificable y suficientemente visible?**	
El eslogan o *tagline*, **¿expresa realmente qué es la empresa y qué servicios ofrece?**	
¿Se ofrece algún enlace con información sobre la empresa, sitio web, 'webmaster',...?	
¿Se proporciona mecanismos para ponerse en contacto con la empresa? (email, teléfono, dirección postal, fax...)	
¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?	
En artículos, noticias, informes... **¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?**	
Lenguaje y Redacción	
¿El sitio web habla el mismo lenguaje que sus usuarios? Se debe evitar usar un lenguaje corporativista. Así mismo, hay que prestarle especial atención al idioma, y ofrecer versiones del sitio en diferentes idiomas cuando sea necesario.	
¿Emplea un lenguaje claro y conciso?	
¿Es amigable, familiar y cercano? Es decir, lo contrario a utilizar un lenguaje constantemente imperativo, mensajes crípticos, o tratar con "desprecio" al usuario.	
¿1 párrafo = 1 idea? Cada párrafo es un objeto informativo. Trasmita ideas, mensajes...Se deben evitar párrafos vacíos o varios mensajes en un mismo párrafo.	
Rotulado	
Los rótulos, **¿son significativos?** Ejemplo: evitar rótulos del tipo "haga clic aquí".	
¿Usa rótulos estándar? Siempre que exista un "estándar" comúnmente aceptado para el caso concreto, como "Mapa del Sitio" o "Acerca de...".	
¿Usa un único sistema de organización, bien definido y claro? No se deben mezclar diferentes. Los sistemas de organización son: alfabético, geográfico, cronológico, temático, orientado a tareas, orientado al público y orientado a metáforas.	
¿Utiliza un sistema de rotulado controlado y preciso? Por ejemplo, si un enlace tiene el rótulo "Quiénes somos", no puede dirigir a una página cuyo encabezamiento sea "Acerca de"	
El título de las páginas, **¿Es correcto? ¿Ha sido planificado?** Relacionado con la capacidad para poder buscar y encontrar el sitio *web*.	
Estructura y Navegación	
La estructura de organización y navegación, **¿Es la más adecuada?** Hay varios tipos de estructuras: jerárquicas, hipertextual, facetada,...	
En el caso de estructura jerárquica, **¿Mantiene un equilibrio entre Profundidad y Anchura?**	
En el caso de ser puramente hipertextual, **¿Están todos los clúster de nodos comunicados?** Aquí se mide la distancia entre nodos.	

¿Los enlaces son fácilmente reconocibles como tales? ¿Su caracterización indica su estado (visitados, activos,...)? Los enlaces no sólo deben reconocerse como tales, sino que su caracterización debe indicar su estado, y ser reconocidos como una unidad	
En menús de navegación, **¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística?** No se deben superar los 7±2 elementos, ni los 2 o, como mucho, 3 términos por elemento.	
¿Es predecible la respuesta del sistema antes de hacer clic sobre el enlace? Relacionado con el nivel de significación del rótulo del enlace, aunque también con: el uso de globos de texto, información contextual, la barra de estado del navegador,...	
¿Se ha controlado que no haya enlaces que no llevan a ningún sitio? Enlaces que no llevan a ningún sitio: Los enlaces rotos, y los que enlazan con la misma página que se está visualizando (por ejemplo enlaces a la "home" desde la misma página de inicio)	
¿Existen elementos de navegación que orienten al usuario acerca de dónde está y cómo deshacer su navegación? ...como *breadcrumbs*, enlaces a la página de inicio,...recuerde que el logo también es recomendable que enlace con la página de inicio.	
Las imágenes enlace, **¿se reconocen como clicables? ¿Incluyen un atributo 'title' describiendo la página de destino?** En este sentido, también hay que cuidar que no haya imágenes que parezcan enlaces y en realidad no lo sean.	
¿Se ha evitado la redundancia de enlaces?	
¿Se ha controlado que no haya páginas "huérfanas"? Páginas huérfanas: que aún siendo enlazadas desde otras páginas, éstas no enlacen con ninguna.	
Layout de la Página	
¿Se aprovechan las zonas de alta jerarquía informativa de la página para contenidos de mayor relevancia? (como por ejemplo la zona central)	
¿Se ha evitado la sobrecarga informativa? Esto se consigue haciendo un uso correcto de colores, efectos tipográficos y agrupaciones para discriminar información. Los grupos diferentes de objetos informativos de una página deben ser 7±2.	
¿Es una interfaz limpia, sin ruido visual?	
¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?	
¿Se hace un uso correcto del espacio visual de la página? Es decir, que no se desaproveche demasiado espacio con elementos de decoración, o grandes zonas en "blanco", y que no se adjudique demasiado espacio a elementos de menor importancia.	
¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página? (La jerarquía visual se utiliza para orientar al usuario)	
¿Se ha controlado la longitud de página? Se debe evitar en la medida de lo posible el *scrolling*. Si la página es muy extensa, se debe fraccionar.	
Búsqueda (si es necesario, por la extensión del sitio, incorporar un buscador interno)	

¿Se encuentra fácilmente accesible? Es decir: directamente desde la home, y a ser posible desde todas las páginas del sitio, y colocado en la zona superior de la página.	
¿Es fácilmente reconocible como tal?	
¿Permite la búsqueda avanzada? (siempre y cuando, por las características del sitio web, fuera de utilidad que la ofreciera)	
¿Muestra los resultados de la búsqueda de forma comprensible para el usuario?	
¿La caja de texto es lo suficientemente ancha?	
¿Asiste al usuario en caso de no poder ofrecer resultados para una consultada dada?	
Elementos Multimedia	
¿Las fotografías están bien recortadas? ¿Son comprensibles? ¿Se ha cuidado su resolución?	
¿Las metáforas visuales son reconocibles y comprensibles por cualquier usuario? (prestar especial atención a usuarios de otros países y culturas)	
¿El uso de imágenes o animaciones proporciona algún tipo de valor añadido?	
¿Se ha evitado el uso de animaciones cíclicas?	
Ayuda	
Si posee una sección de Ayuda, **¿Es verdaderamente necesaria?** Siempre que se pueda prescindir de ella simplificando los elementos de navegación e interacción, debe omitirse esta sección.	
En enlace a la sección de Ayuda, **¿Está colocado en una zona visible y "estándar"?** La zona de la página más normal para incluir el enlace a la sección de Ayuda, es la superior derecha.	
¿Se ofrece ayuda contextual en tareas complejas? (transferencias bancarias, formularios de registro...)	
Si posee *FAQs*, **¿Es correcta tanto la elección como la redacción de las preguntas? ¿Y las respuestas?**	
Accesibilidad (debería cubrirse con los test de Accesibilidad posteriores)	
¿El tamaño de fuente se ha definido de forma relativa, o por lo menos, la fuente es lo suficientemente grande como para no dificultar la legibilidad del texto?	
¿El tipo de fuente, efectos tipográficos, ancho de línea y alineación empleadas facilitan la lectura?	
¿Existe un alto contraste entre el color de fuente y el fondo?	
¿Incluyen las imágenes atributos 'alt' que describan su contenido?	
¿Es compatible el sitio web con los diferentes navegadores? ¿Se visualiza correctamente con diferentes resoluciones de pantalla? Se debe prestar atención a: *JScript*, *CSS*, tablas, fuentes...	
¿Puede el usuario disfrutar de todos los contenidos del sitio web sin necesidad de tener que descargar e instalar *plugins* adicionales?	
¿Se ha controlado el peso de la página? Se deben optimizar las imágenes, controlar el tamaño del código *JScript*...	

¿Se puede imprimir la página sin problemas? Leer en pantalla es molesto, por lo que muchos usuarios preferirán imprimir las páginas para leerlas. Se debe asegurar que se puede imprimir la página (no salen partes cortadas), y que el resultado es legible.	
Control y Retroalimentación	
¿Tiene el usuario todo el control sobre el interfaz? Se debe evitar el uso de ventanas pop-up, ventanas que se abren a pantalla completa, banners intrusivos...	
¿Se informa constantemente al usuario acerca de lo que está pasando? Si el usuario tiene que esperar hasta que se termine una operación, se debe mostrar un mensaje indicándoselo y que debe esperar, con el tiempo de espera estimado o una barra de progreso.	
¿Se informa al usuario de lo que ha pasado? Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente.	
Cuando se produce un error, **¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema?** Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error.	
¿Posee el usuario libertad para actuar? NO restringir la libertad del usuario: Uso de animaciones que no pueden ser "saltadas", páginas en las que desaparecen los botones de navegación, no impida al usuario poder usar el botón derecho de su ratón...	
¿Se ha controlado el tiempo de respuesta? Esto tiene que ver con el peso de cada página (accesibilidad) y tiene relación con el tiempo que tarda el servidor en finalizar una tarea y responder. El tiempo máximo que esperará un usuario son 10 segundos	
Aclaraciones	
¿Se ha evaluado adecuadamente la orientación del usuario? (Donde estoy, como volver, que he visitado, que va a pasar)[2]	
¿Se ha usado correctamente la publicidad?[3]	

Tras rellenar esta tabla se deben poner a continuación una enumeración de todas aquellas aclaraciones y/o observaciones que queramos hacer acerca de la misma.

[2] La orientación del usuario se puede evaluar a través de varios criterios: elementos de navegación orientativos, caracterización de los enlaces e información contextual en elementos de interacción (estructura y navegación); distribución visual de la página (*layout*); coherencia del diseño (generales); nivel de significación de los rótulos (rotulado) y retroalimentación del usuarios (control y retroalimentación)

[3] Respecto a la publicidad (normalmente en forma de banners), se puede evaluar desde varios criterios: lenguaje (lenguaje y redacción), nivel de significación de los rótulos (rotulado), jerarquía informativa y ruido visual (*layout* de la página), pop-ups y banners intrusivos (control y retroalimentación)...

8.3.2 Pruebas de Accesibilidad

A continuación se detallan las tareas que se recomiendan hacer para asegurarnos de que el programa creado cumple con los estándares de accesibilidad. Esta sección está muy enfocada a proyectos web, pero algunas ideas pueden extrapolarse a programas de escritorio en caso necesario. Un proyecto debería hacer una evaluación de conformidad completa del mismo, pero se ha incluido un procedimiento más abreviado (revisión preliminar) para aquellos casos que por alguna razón se necesite pasar un procedimiento más rápido o menos exhaustivo. Una forma de proceder es hacer la revisión preliminar y, una vez superada arreglando todos los defectos encontrados, intentar ir a por la evaluación de conformidad completa, que permitirá reutilizar todo el estudio hecho en la revisión preliminar para hacerla.

8.3.2.1 Revisión Preliminar

Como paso previo a hacer los pasos descritos, es necesario enumerar el material utilizado para ello, como navegadores (*IE*, *Firefox*,…), lectores de pantalla o cualquier otra herramienta usada para hacer las pruebas.

Paso 1. Selección de un grupo de páginas representativo de la aplicación

Para pasar las pruebas de accesibilidad es necesario escoger una muestra de páginas de la aplicación representativa para así no tener que someter a toda la aplicación al proceso de revisión. Una muestra representativa está formada por:

- La página de entrada al sitio / principal / home
- Páginas con distinta organización y funcionalidad, como por ejemplo:
    - Páginas con tablas
    - Páginas con formularios
    - Páginas con diagramas o gráficos
    - Páginas con scripts o aplicaciones
    - Páginas con resultados generados dinámicamente (por ejemplo, con un gestor de contenidos)

Paso 2. Examinar las páginas usando un navegador gráfico

Esta actividad comprende las siguientes operaciones:

- Desactivar las imágenes y probar como queda el aspecto de la aplicación. Se puede hacer fácilmente mediante el menú "*Images*" de la "*Web Developer Extension*" de *Firefox* de la que se habló anteriormente.
- Apagar los altavoces (si la página tuviese alguna clase de narración oral o sonido)
- Cambiar el tamaño del texto y comprobar que sigue siendo usable (muchos navegadores permiten hacer esto fácilmente con *CTRL + Rueda del ratón*).

- Cambiar solamente el tamaño de letra de la página para ver cómo se comporta. En *Firefox* podemos ir a "Herramientas – Opciones – Contenido", aunque también podemos hacerlo cambiando la *CSS* de la propia página si el navegador no soporta esta opción. Si un tamaño de letra estándar es 16, se puede probar con un tamaño mínimo (9) y un máximo (72) para ver qué ocurre. Posteriormente podemos hacer pruebas con dos tamaños intermedios (32 y 48), para ver qué ocurriría con nuestra página si los usuarios necesitan un tamaño de letra muy superior al estándar.
- Cambiar la resolución y el tamaño de la ventana (verificando que no es necesario el *scroll* horizontal). Esto puede hacerse fácilmente con el *plugin* "*Web Developer Extension*" de *Firefox*. Este *plugin*, una vez instalado tiene una opción "*Resize*". Mediante su sub-opción "*Edit Resize Dimensions*" podemos introducir nuevas resoluciones que podemos usar para comprobar cómo se comporta nuestra aplicación ante ellas. Resoluciones interesantes a probar son: 800x600, 1024x768, 1152x864, 1280x720, 1280x768, 1280x960, 1280x1024 y 1600x1200. Las resoluciones a probar están condicionadas por la máxima resolución de nuestro monitor, por lo que se recomienda poner el mismo a la máxima resolución posible antes de hacer estas pruebas.
- Relacionado con lo anterior, también se recomienda, si es posible, comprobar cómo se ve nuestra aplicación en múltiples tamaños de pantalla (15 pulgadas, 17 pulgadas,…). Otra posible prueba es redimensionar la página múltiples veces (probando a hacerla cada vez más grande y cada vez más pequeña) para ver cómo se comporta su contenido ante esta situación. Si el aspecto de la página se estropea por ello, entonces habremos encontrado un problema.
- Probar a visualizar la página sin sus hojas de estilo *CSS* para asegurarse de que aún es legible y usable. Se puede hacer fácilmente mediante el menú "*CSS*" de la "*Web Developer Extension*" de *Firefox* de la que se habló anteriormente.
- Probar a visualizar la página usando una escala de grises. Para esto podemos usar la herramienta web *GrayBit* (http://graybit.com/main.php).
- Usar el teclado para navegar a través de los enlaces y controles de formularios, usando *Tab* para desplazarse.

Herramientas útiles para comprobar estos aspectos pueden ser:

- *AIS Toolbar* para *Internet Explorer* y *Opera*
- *WAVE Toolbar* para *Firefox*, *Internet Explorer* y *Netscape*
- *Web Developer Extension* para *Firefox (*recomendada*)*

Paso 3. Examinar las páginas usando uno o varios navegadores especializados

Es muy necesario comprobar cómo nuestras páginas se comportan ante diferentes clases de navegadores, como por ejemplo:

- Navegadores por voz como *Firevox* (http://firevox.clcworld.net/) (gratuito) o *JAWS* (http://www.freedomscientific.com/jaws-hq.asp). También puede encontrarse una lista de herramientas similares aquí: http://en.wikipedia.org/wiki/Comparison_of_screen_readers

- Navegadores de texto como *Lynx*. Si se trabaja en *Windows* es posible ejecutarlo usando *Cygwin*.

En estos navegadores es necesario verificar que la información transmitida por ambos tipos es similar a la mostrada en el navegador gráfico y asegurarse de que el orden en el que se transmite dicha información es coherente.

Por otro lado, también es conveniente probar la página con diferentes navegadores (*IE*, *Firefox*, *Opera*, *Chrome*,…) y con distintas versiones de cada navegador. Aunque podría hacerse mediante *VMware*, instalar un gran nº de navegadores y versiones de los mismos es muy costoso. No obstante, una herramienta que nos proporcionará fácilmente acceso a un elevadísimo número de navegadores distintos, con múltiples versiones de cada uno y con distintos sistemas operativos, es *BrowserShots* (http://browsershots.org/). En esta página *Web* podremos introducir la *URL* de nuestro sitio (en caso de que no la tengamos disponible se puede probar con la *IP* de la máquina de desarrollo, asegurándose de que es accesible). Esto invocará a un servicio *web* que distribuirá la petición a múltiples máquinas con distintos navegadores instalados, devolviéndonos las capturas de pantalla en todos y cada uno de esos navegadores de nuestro sitio web, transcurrido un tiempo de proceso de la petición (en función de la hora del día y de la carga de trabajo del servicio puede tardar bastante). Si no cerramos la página o cancelamos la petición y tenemos paciencia, una vez terminado el proceso tendremos imágenes de nuestra web en multitud de navegadores y SO diferentes fácilmente, algo que de otra manera nos supondría un coste muy elevado. Debido al coste en tiempo de esta herramienta, se recomienda hacerlo con la versión final del interfaz de la aplicación.

Paso 4. Utilizar herramientas automáticas de evaluación de accesibilidad

Este último paso consiste en pasar a la muestra de páginas seleccionadas herramientas de evaluación automática de la accesibilidad como:

- *TAW* (http://www.tawdis.net/taw3/cms/es)
- *HERA* (http://www.sidar.org/hera/index.php.en)
- *EvalAccess* (http://sipt07.si.ehu.es/evalaccess2/index.html)
- *WAVE* (http://wave.webaim.org/)

Código de campo cambiado

Código de campo cambiado

Debemos tener en cuenta:

- Que hay que pasar un mínimo de dos herramientas de esta clase.
- Que sólo pueden probar algunos aspectos de la accesibilidad. Una vez eliminados todos los errores de comprobación automática de la página, debemos hacer todo lo posible por eliminar todos los de comprobación manual (de todas las herramientas pasadas).

Paso 5. Resumen de resultados

Finalmente, se incluirá un pequeño informe final con los siguientes aspectos:

- Tipos de problemas encontrados, como se han resuelto y aspectos positivos de la página.
- Indicar como se detectó cada problema.

8.3.2.2 Evaluación de Conformidad

La evaluación de conformidad combina la evaluación manual de la página con la evaluación semiautomática. Se suele emplear cuando se desarrolla un sitio nuevo o bien cuando se evalúa un sitio existente. En el caso de un PFC, debemos intentar pasar una evaluación de conformidad completa al sitio para considerar que realmente ha hecho un esfuerzo adecuado por lograr un nivel de accesibilidad óptimo.

Paso 1. Determinar el alcance de la evaluación

Este paso contempla:

- Definir y divulgar el nivel deseado de conformidad *WCAG 1.0* (*A*, *AA*, *AAA*). Se recuerda que las web destinadas a la administración pública deben tener al menos un nivel AA, siendo este un nivel mínimo exigible en un PFC. También pueden emplearse las normas *WCAG 2.0*. El siguiente enlace proporciona información de cómo pasar de la versión 1.0 a la versión 2.0 de forma resumida: http://www.w3.org/WAI/WCAG20/from10/comparison/
- Seleccionar un conjunto representativo de páginas para la evaluación manual, siguiendo el criterio expuesto en la revisión preliminar.

Paso 2. Utilizar herramientas de evaluación automática de accesibilidad

Este paso contempla:

- Validar los lenguajes utilizados (*HTML*, *CSS*,…) con las herramientas adecuadas que existan para ello. La "*Web Developer Extension*" de *Firefox* posee opciones para ello.
- Usar al menos dos herramientas de evaluación automática en la muestra de páginas seleccionada (ver la revisión preliminar). También se debe usar al menos 1 herramienta en la totalidad del sitio.
- Anotar y resolver los problemas encontrados.

Paso 3. Evaluar manualmente la muestra de páginas

Para ello debemos:

- Utilizar el *checklist* de puntos de control del *WCAG* que aparecerá tras esta explicación, según versión. Como es mucho más probable que en los PFC se use la versión 1.0 de las normas, al final de esta sección se ha incluido este *checklist* adaptado a *Word* y listo para rellenarse:
  - *WCAG 1.0*: http://www.w3.org/TR/WCAG10/full-checklist.html

Código de campo cambiado

- *WCAG 2.0*: http://www.w3.org/TR/2006/WD-WCAG20-20060427/appendixB.html

Código de campo cambiado

- Examinar las páginas con al menos 3 navegadores gráficos en diferentes versiones y en diferentes plataformas (para lo cual puede usarse el servicio *web Browsershots* ya visto). Tener en cuenta especialmente estos puntos, que son una lista extendida de los que se enunciaron en la revisión preliminar. Para su comprobación podemos usar las mismas herramientas y procedimientos recomendados en dicha sección:
  - Desactivar las imágenes y probar como queda el aspecto de la aplicación. Se puede hacer fácilmente mediante el menú "*Images*" de la "*Web Developer Extension*" de *Firefox* de la que se habló anteriormente.
  - Apagar los altavoces (si la página tuviese alguna clase de narración oral o sonido)
  - Cambiar el tamaño del texto y comprobar que sigue siendo usable (muchos navegadores permiten hacer esto fácilmente con *CTRL + Rueda del ratón*).
  - Cambiar solamente el tamaño de letra de la página para ver cómo se comporta. En *Firefox* podemos ir a "Herramientas – Opciones – Contenido", aunque también podemos hacerlo cambiando la *CSS* de la propia página si el navegador no soporta esta opción. Si un tamaño de letra estándar es 16, se puede probar con un tamaño mínimo (9) y un máximo (72) para ver qué ocurre. Posteriormente podemos hacer pruebas con dos tamaños intermedios (32 y 48), para ver qué ocurriría con nuestra página si los usuarios necesitan un tamaño de letra muy superior al estándar.
  - Cambiar la resolución y el tamaño de la ventana (verificando que no es necesario el *scroll* horizontal). Esto puede hacerse fácilmente con el *plugin* "*Web Developer Extension*" de *Firefox*. Este *plugin*, una vez instalado tiene una opción "*Resize*". Mediante su sub-opción "*Edit Resize Dimensions*" podemos introducir nuevas resoluciones que podemos usar para comprobar cómo se comporta nuestra aplicación ante ellas. Resoluciones interesantes a probar son: 800x600, 1024x768, 1152x864, 1280x720, 1280x768, 1280x960, 1280x1024 y 1600x1200. Las resoluciones a probar están condicionadas por la máxima resolución de nuestro monitor, por lo que se recomienda poner el mismo a la máxima resolución posible antes de hacer estas pruebas.
  - Relacionado con lo anterior, también se recomienda, si es posible, comprobar cómo se ve nuestra aplicación en múltiples tamaños de pantalla (15 pulgadas, 17 pulgadas,…). Otra posible prueba es redimensionar la página múltiples veces (probando a hacerla cada vez más grande y cada vez más pequeña) para ver cómo se comporta su contenido ante esta situación. Si el aspecto de la página se estropea por ello, entonces habremos encontrado un problema.
  - Probar a visualizar la página sin sus hojas de estilo *CSS* para asegurarse de que aún es legible y usable. Se puede hacer fácilmente mediante el menú "*CSS*" de la "*Web Developer Extension*" de *Firefox* de la que se habló anteriormente.
  - Probar a visualizar la página usando una escala de grises. Para esto podemos usar la herramienta web *GrayBit* (http://graybit.com/main.php). En lo referente a colores, también debemos usar herramientas de verificación de color:
    - ***Color Contrast Analyzer*** (http://www.visionaustralia.org.au/info.aspx?page=628): permite verificar que los colores de fondo y texto de una imagen o página web contrastan lo suficiente para ser distinguibles por cualquier persona, según los algoritmos desarrollados por el W3C.

        - ***Vischeck*** (http://www.vischeck.com/): muestra como ven una página web personas con distintos tipos de daltonismo: deuteranopia, protanopia y tritanopia.
    - Usar el teclado para navegar a través de los enlaces y controles de formularios, usando *Tab* para desplazarse.
    - Desactivar *scripts*, *applets*, *Flash*, etc. y comprobar que la página sigue siendo navegable. Para esto es especialmente útil el *plugin NoScript* de *Firefox*
- Examinar las páginas usando uno o varios navegadores especializados: al menos uno de texto y uno de voz (ver revisión preliminar).
- Leer y evaluar el contenido de las páginas, para buscar texto no claro o incongruente. Sobre todo debe primar que el texto de las páginas sea lo más claro posible.

Paso 4. Elaborar el informe de conclusiones

Finalmente se incluirá un informe de conclusiones que contemplará los siguientes aspectos:

- Resumen de los principales problemas encontrados y la solución dada a los mismos
- Resumen de los aspectos positivos que potencian la accesibilidad de la página
- Recomendación de fututas actividades para mejorar la accesibilidad de la página:
    - Reparación de barreras de accesibilidad encontradas y que no hayan podido ser solucionadas del todo o adecuadamente.
    - Ampliación de aspectos positivos de accesibilidad
    - Monitorización del sitio.

Por último, a modo de resumen se destacan aquí algunos aspectos muy importantes que no deben dejarse nunca de lado cuando se desarrolla una página, para minimizar los problemas cuando se haga una revisión de cualquier clase sobre la misma. No obstante, todo esto se contempla en el *checklist* del *WCAG*:

- Poner un texto alternativo para las imágenes
- Poner texto alternativo en los hipervínculos
- Revisar los formularios una vez creados (navegación, claridad)
- Marcos: títulos en los marcos y existencia de *tags <noframes>*
- Desactivar *scripts* y *Flash* y comprobar que la página no pierde funcionalidades
- Navegar sólo con el teclado y comprobar que se puede acceder a todo el sitio.
- Hacer especial hincapié en la revisión de la página de inicio (más importante).

8.3.2.3 Checklist del WCAG 1.0

La siguiente tabla es el *checklist* que el *WCAG* proporciona para verificar las pautas de accesibilidad de la aplicación *web*, pero adaptado a *Word* para poder ser editado fácilmente. Cada punto tiene antes un hiperenlace que va directamente a la web del *WCAG* para

proporcionar explicaciones adicionales sobre el mismo (que pueden ir bien para arreglar los problemas detectados).

Puntos de verificación Prioridad 1:

En general (Prioridad 1)	Sí	No	N/A
1.1 Proporcione un texto equivalente para todo elemento no textual (Por ejemplo, a través de "alt", "longdesc" o en el contenido del elemento). *Esto incluye:* imágenes, representaciones gráficas del texto, mapas de imagen, animaciones (Por ejemplo, *GIFs* animados), "applets" y objetos programados, "ascii art", marcos, scripts, imágenes usadas como viñetas en las listas, espaciadores, botones gráficos, sonidos (ejecutados con o sin interacción del usuario), archivos exclusivamente auditivos, banda sonora del vídeo y vídeos.		**X**	
2.1 Asegúrese de que toda la información transmitida a través de los colores también esté disponible sin color, por ejemplo mediante el contexto o por marcadores.	**X**		
4.1 Identifique claramente los cambios en el idioma del texto del documento y en cualquier texto equivalente (por ejemplo, leyendas).			**X**
6.1 Organice el documento de forma que pueda ser leído sin hoja de estilo. Por ejemplo, cuando un documento HTML es interpretado sin asociarlo a una hoja de estilo, tiene que ser posible leerlo.			
6.2 Asegúrese de que los equivalentes de un contenido dinámico son actualizados cuando cambia el contenido dinámico.			
7.1 Hasta que las aplicaciones de usuario permitan controlarlo, evite provocar destellos en la pantalla.			
14.1 Utilice el lenguaje apropiado más claro y simple para el contenido de un sitio.			
Y si utiliza imágenes y mapas de imagen (Prioridad 1)	**Sí**	**No**	**N/A**
1.2 Proporcione vínculos redundantes en formato texto para cada zona activa de un mapa de imagen del servidor.			
9.1 Proporcione mapas de imagen controlados por el cliente en lugar de por el servidor, excepto donde las zonas sensibles no puedan ser definidas con una forma geométrica.			
Y si utiliza tablas (Prioridad 1)	**Sí**	**No**	**N/A**
5.1 En las tablas de datos, identifique los encabezamientos de fila y columna.			
5.2 Para las tablas de datos que tienen dos o más niveles lógicos de encabezamientos de fila o columna, utilice marcadores para asociar las celdas de encabezamiento y las celdas de datos.			
Y si utiliza marcos ("frames") (Prioridad 1)	**Sí**	**No**	**N/A**
12.1 Titule cada marco para facilitar su identificación y navegación.			
Y si utiliza "applets" y "scripts" (Prioridad 1)	**Sí**	**No**	**N/A**
6.3 Asegure que las páginas sigan siendo utilizables cuando se desconecten o no se soporten los scripts, *applets* u otros objetos programados. Si esto no es posible, proporcione información equivalente en una página alternativa accesible.			
Y si utiliza multimedia (Prioridad 1)	**Sí**	**No**	**N/A**
1.3 Hasta que las aplicaciones de usuario puedan leer en voz alta automáticamente el texto equivalente de la banda visual, proporcione una descripción auditiva de la información importante de la banda			

visual de una presentación multimedia.			
1.4 Para toda presentación multimedia dependiente del tiempo (por ejemplo, una película o animación) sincronice alternativas equivalentes (por ejemplo, subtítulos o descripciones de la banda visual) con la presentación.			
Y si todo lo demás falla (Prioridad 1)	**Sí**	**No**	**N/A**
11.4 Si, después de los mayores esfuerzos, no puede crear una página accesible, proporcione un vínculo a una página alternativa que use tecnologías *W3C*, sea accesible, tenga información (o funcionalidad) equivalente y sea actualizada tan a menudo como la página (original) inaccesible.			

Puntos de verificación Prioridad 2:

En general (Prioridad 2)	**Sí**	**No**	**N/A**
2.2 Asegúrese de que las combinaciones de los colores de fondo y primer plano tengan el suficiente contraste para que sean percibidas por personas con deficiencias de percepción de color o en pantallas en blanco y negro [Prioridad 2 para las imágenes. Prioridad 3 para los textos].			
3.1 Cuando exista un marcador apropiado, use marcadores en vez de imágenes para transmitir la información.			
3.2 Cree documentos que estén validados por las gramáticas formales publicadas.			
3.3 Utilice hojas de estilo para controlar la maquetación y la presentación.			
3.4 Utilice unidades relativas en lugar de absolutas al especificar los valores en los atributos de los marcadores de lenguaje y en los valores de las propiedades de las hojas de estilo.			
3.5 Utilice elementos de encabezado para transmitir la estructura lógica y utilícelos de acuerdo con la especificación.			
3.6 Marque correctamente las listas y los ítems de las listas.			
3.7 Marque las citas. No utilice el marcador de citas para efectos de formato tales como sangrías.			
6.5 Asegúrese de que los contenidos dinámicos son accesibles o proporcione una página o presentación alternativa.			
7.2 Hasta que las aplicaciones de usuario permitan controlarlo, evite el parpadeo del contenido (por ejemplo, cambio de presentación en periodos regulares, así como el encendido y apagado).			
7.4 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener las actualizaciones, no cree páginas que se actualicen automáticamente de forma periódica.			
7.5 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener el redireccionamiento automático, no utilice marcadores para redirigir las páginas automáticamente. En su lugar, configure el servidor para que ejecute esta posibilidad.			
10.1 Hasta que las aplicaciones de usuario permitan desconectar la apertura de nuevas ventanas, no provoque apariciones repentinas de nuevas ventanas y no cambie la ventana actual sin informar al usuario.			
11.1 Utilice tecnologías *W3C* cuando estén disponibles y sean			

apropiadas para la tarea y use las últimas versiones que sean soportadas.			
11.2 Evite características desaconsejadas por las tecnologías *W3C*.			
12.3 Divida los bloques largos de información en grupos más manejables cuando sea natural y apropiado.			
13.1 Identifique claramente el objetivo de cada vínculo.			
13.2 Proporcione metadatos para añadir información semántica a las páginas y sitios.			
13.3 Proporcione información sobre la maquetación general de un sitio (por ejemplo, mapa del sitio o tabla de contenidos).			
13.4 Utilice los mecanismos de navegación de forma coherente.			
Y si utiliza tablas (Prioridad 2)	**Sí**	**No**	**N/A**
5.3 No utilice tablas para maquetar, a menos que la tabla tenga sentido cuando se alinee. Por otro lado, si la tabla no tiene sentido, proporcione una alternativa equivalente (la cual debe ser una versión alineada).			
5.4 Si se utiliza una tabla para maquetar, no utilice marcadores estructurales para realizar un efecto visual de formato.			
Y si utiliza marcos ("frames") (Prioridad 2)	**Sí**	**No**	**N/A**
12.2 Describa el propósito de los marcos y cómo éstos se relacionan entre sí, si no resulta obvio solamente con el título del marco.			
Y si utiliza formularios (Prioridad 2)	**Sí**	**No**	**N/A**
10.2 Hasta que las aplicaciones de usuario soporten explícitamente la asociación entre control de formulario y etiqueta, para todos los controles de formularios con etiquetas asociadas implícitamente, asegúrese de que la etiqueta está colocada adecuadamente.			
12.4 Asocie explícitamente las etiquetas con sus controles.			
Y si utiliza "applets" y "scripts" (Prioridad 2)	**Sí**	**No**	**N/A**
6.4 Para los *scripts* y *applets*, asegúrese de que los manejadores de eventos sean independientes del dispositivo de entrada.			
7.3 Hasta que las aplicaciones de usuario permitan congelar el movimiento de los contenidos, evite los movimientos en las páginas.			
8.1 Haga los elementos de programación, tales como *scripts* y *applets*, directamente accesibles o compatibles con las ayudas técnicas [Prioridad 1 si la funcionalidad es importante y no se presenta en otro lugar; de otra manera, Prioridad 2].			
9.2 Asegúrese de que cualquier elemento que tiene su propia interfaz pueda manejarse de forma independiente del dispositivo.			
9.3 Para los "scripts", especifique manejadores de evento lógicos mejor que manejadores de eventos dependientes de dispositivos.			

Puntos de verificación Prioridad 3:

En general (Prioridad 3)	**Sí**	**No**	**N/A**
4.2 Especifique la expansión de cada abreviatura o acrónimo cuando aparezcan por primera vez en el documento.			
4.3 Identifique el idioma principal de un documento.			
9.4 Cree un orden lógico para navegar con el tabulador a través de vínculos, controles de formulario y objetos.			
9.5 Proporcione atajos de teclado para los vínculos más importantes			

(incluidos los de los mapas de imagen de cliente), los controles de formulario y los grupos de controles de formulario.			
10.5 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten claramente los vínculos contiguos, incluya caracteres imprimibles (rodeados de espacios), que no sirvan como vínculo, entre los vínculos contiguos.			
11.3 Proporcione la información de modo que los usuarios puedan recibir los documentos según sus preferencias (por ejemplo, idioma, tipo de contenido, etc.).			
13.5 Proporcione barras de navegación para destacar y dar acceso al mecanismo de navegación.			
13.6 Agrupe los vínculos relacionados, identifique el grupo (para las aplicaciones de usuario) y, hasta que las aplicaciones de usuario lo hagan, proporcione una manera de evitar el grupo.			
13.7 Si proporciona funciones de búsqueda, permita diferentes tipos de búsquedas para diversos niveles de habilidad y preferencias.			
13.8 Localice la información destacada al principio de los encabezamientos, párrafos, listas, etc.			
13.9 Proporcione información sobre las colecciones de documentos (por ejemplo, los documentos que comprendan múltiples páginas).			
13.10 Proporcione un medio para saltar sobre un *ASCII art* de varias líneas.			
14.2 Complemente el texto con presentaciones gráficas o auditivas cuando ello facilite la comprensión de la página.			
14.3 Cree un estilo de presentación que sea coherente para todas las páginas.			
Y si utiliza imágenes o mapas de imagen (Prioridad 3)	**Sí**	**No**	**N/A**
1.5 Hasta que las aplicaciones de usuario interpreten el texto equivalente para los vínculos de los mapas de imagen de cliente, proporcione vínculos de texto redundantes para cada zona activa del mapa de imagen de cliente.			
Y si utiliza tablas (Prioridad 3)	**Sí**	**No**	**N/A**
5.5 Proporcione resúmenes de las tablas.			
5.6 Proporcione abreviaturas para las etiquetas de encabezamiento.			
10.3 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten correctamente los textos contiguos, proporcione un texto lineal alternativo (en la página actual o en alguna otra) para *todas* las tablas que maquetan texto en paralelo, en columnas de palabras.			
Y si utiliza formularios (Prioridad 3)	**Sí**	**No**	**N/A**
10.4 Hasta que las aplicaciones de usuario manejen correctamente los controles vacíos, incluya caracteres por defecto en los cuadros de edición y áreas de texto.			

8.3.2.4 Accesibilidad con Dispositivos Móviles

En caso de que la página a analizar esté destinada a un dispositivo móvil (o tenga una parte o una versión destinada para los mismos), se proporcionan aquí enlaces a herramientas e información útil en estos casos:

- ***W3C mobileOK Checker***: http://validator.w3.org/mobile/
- ***TAW Ok Basic***: http://validadores.tawdis.net/mobileok/es
- ***Ready.mobi***: http://ready.mobi/launch.jsp?locale=en_EN

Código de campo cambiado
Código de campo cambiado
Código de campo cambiado

Otro aspecto a tener en cuenta con estos dispositivos es que si tenemos que evaluar la página en varios de ellos puede ser muy difícil conseguir el hardware necesario. No obstante, para esto nos pueden servir emuladores de los mismos, que permitan comprobar sobre un mismo PC y de forma rápida y sencilla como se ve nuestro sitio en distintos dispositivos móviles con gran fiabilidad. Podemos pues usarlos para enriquecer nuestras pruebas de usabilidad, mostrando cómo se ve nuestra aplicación en un dispositivo móvil. Ejemplos son:

- **.mobi**: http://mtld.mobi/emulator.php
- ***The Openwave Phone Simulator***: http://developer.openwave.com/dvl/tools_and_sdk/phone_simulator/

Código de campo cambiado
Código de campo cambiado

Una vez pasadas las pruebas indicadas en estos enlaces (para las que también pueden tomarse ideas de las pruebas de accesibilidad anteriores), se debe hacer un informe similar a los de la sección anterior indicando:

- Tipos de problemas encontrados, como se han resuelto y aspectos positivos de la página.
- Indicar como se detectó cada problema.

8.4 Pruebas de Rendimiento

Explicamos el desarrollo, resultados y cambios derivados de la ejecución de todas las pruebas de rendimiento que hayamos especificado para el sistema, según lo que hemos diseñado en el anteriormente.

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

Elaborar un manual que contemple todos los pasos necesarios para instalar nuestro sistema, incluyendo la instalación de otras herramientas o software cualquiera (sea o no comercial) necesario para que funcione. Debemos explicarlo todo paso a paso de forma clara y acompañarlo por capturas de pantalla adecuadas.

9.2 Manual de Ejecución

Este manual contemplará todos los pasos necesarios para el arranque de nuestro sistema, lo que es especialmente importante en caso de sistemas con clientes y servidores o distintos procesos que deban arrancarse independientemente.

Por otra parte, también debemos incluir procedimientos para parar adecuadamente la aplicación.

9.3 Manual de Usuario

El manual de usuario es algo muy importante debido a que es el documento que servirá a los usuarios de nuestro sistema para saber cómo funciona cada una de las partes de nuestra aplicación. Debemos pues describir cómo funcionan todas las opciones de las misma, que parámetros tiene, que cosas debemos hacer para que todas las operaciones funcionen correctamente y cualquier otro aspecto que consideremos oportuno para explicar el funcionamiento del sistema.

No debemos escatimar detalles en este manual ya que es la herramienta para que los usuarios comprendan nuestro sistema. También debemos hacer el mayor uso posible de capturas de pantalla para mejorar nuestras explicaciones.

9.4 Manual del Programador

En este manual debemos describir cualquier aspecto que pueda ayudar a otros programadores a ampliar, modificar o entender aspectos de la construcción de nuestra aplicación. Debemos por tanto hacer una descripción general de los distintos aspectos involucrados en la construcción del sistema que puedan ser más difíciles de entender y también describir los procedimientos necesarios para hacer ciertas ampliaciones que hayamos contemplado en el diseño del sistema (añadir nuevas entidades, nuevos atributos a entidades existentes, nuevos servicios que usen a los ya desarrollados, modificaciones en la interfaz, etc.).

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Conclusiones del sistema: Qué hemos elaborado, si los resultados están dentro de lo esperado, si hemos cumplido las expectativas, justificación de haber escogido las mejores opciones para cada uno de los aspectos del sistema, etc.

10.2 Ampliaciones

Cualquier labor de ampliación que tengamos contemplada en el sistema debe ser descrita aquí, mencionando en qué consiste, cómo ampliará el sistema, qué ventajas nos aporta y porqué no se ha incluido en el sistema diseñado, entre otros aspectos.

Capítulo 11. Planificación del Proyecto y Presupuesto finales

A continuación se presentan dos posibles alternativas para el desarrollo del presupuesto del proyecto. La primera de ellas (recomendada) contempla muchos más aspectos y es más completa en líneas generales que la segunda. Seleccionar una u otra depende del criterio del director y del tipo de proyecto. Se presentan ambas para que se seleccione la que se considere más adecuada en cada caso. Otra posible opción es tomar elementos de ambas opciones a conveniencia. **En cualquier caso, consultar al director de proyecto acerca de cuál de las dos es más adecuada para el proyecto desarrollado.**

11.1 Planificación Final

En esta sección se añadirá la planificación final y se explicará las diferencias respecto a planificación inicial y el porqué de estas. Se recuerda, que esta debería de corresponderse con todas las iteraciones, pues debería de ser la copia que se fue actualizando continuamente durante el proyecto.

11.2 Presupuesto Final

En esta sección se realizará una reevaluación del presupuesto añadiendo o quitando, si se da e laso, de aquellos materiales, tanto hardware como software o módulos o extensiones que no tuvimos en cuenta en el presupuesto y planificación inicial. Después, compararemos estos presupuestos con los iniciales y explicaremos a que son debidas sus variaciones.

11.2.1 Desarrollo de Presupuesto Detallado (Empresa)

Item	SubItem	Concepto	Cantidad	Precio Unitario	TOTAL
01		***Desarrollo de la Aplicación***			**7.400,00 €**
	001	*Analisis*	20,00	60,00 €	1.200,00 €
	002	*Diseño*	25,00	60,00 €	1.500,00 €
	003	*Implementación*	100,00	37,00 €	3.700,00 €
	004	*Pruebas*	25,00	40,00 €	1.000,00 €
02		***Formación***			**8.000,00 €**
	001	*Directivos*	20,00	50,00 €	1.000,00 €
	002	*Usuarios*	40,00	50,00 €	2.000,00 €
	003	*Mantenimiento*	100,00	50,00 €	5.000,00 €
				Subtotal	30.800,00 €
				IVA (16%)	4.928,00 €
				TOTAL	**35.728,00 €**

11.2.2 Desarrollo de Presupuesto Simplificado (Cliente)

Concepto	Cantidad	Precio Unitario	Coste Total Concepto
Horas de Programador			0,00 €
Horas de Análisis y Diseño			0,00 €
…			0,00 €
			0,00 €
			0,00 €
		Subtotal	0,00 €
		IVA (16%)	0,00 €
		TOTAL	**0,00 €**

Capítulo 12. Referencias Bibliográficas

12.1 Libros y Artículos

Libros y artículos usados de alguna forma durante el desarrollo del proyecto o su documentación. **Para que esta tarea sea fácil de hacer se recomienda utilizar un gestor de referencias como puede ser Mendeley**. Los gestores de referencias permiten introducir libros, artículos, webs y otros documentos similares para así ser citados en un texto y se encargan ellos de numerar, generar la bibliografía y cambiar el estilo de una manera fácil y rápida.

Formato sugerido:

[<PrimerApellidoAutor><DosUltimosDigitosDelAño>] <Apellidos1, Nombre1; Apellidos2, Nombre2;…>. ”<Título del libro o Articulo>”. <Editorial o lugar de publicación>. <Año (4 cifras)>.

Ejemplo:

[Redondo07] Redondo L., J. Manuel; De Tal y Cual, Menganito. ”Ejemplo para la plantilla de PFC”. Universidad de Oviedo. 2007.

Si tenemos el ISBN, debemos también ponerlo al final.

12.2 Referencias en Internet

Páginas Web consultadas para cualquier aspecto relacionado con el desarrollo del sistema o su documentación.

Formato sugerido:

[<PrimerApellidoAutor><DosUltimosDigitosDelAño>] <Apellidos1, Nombre1; Apellidos2, Nombre2;…>. "<Título de la página Web>". <URL>. <Año en el que se consultó (4 cifras)>.

Ejemplo:

[Redondo07] Redondo L., J. Manuel; De Tal y Cual, Menganito. "Título de la página Web de ejemplo". www.unaurlcualquiera.com. 2007.

Si tenemos más datos que permitan localizar la información dentro de la página, podemos ponerla donde consideremos oportuno.

Esta referencia es real (se usa dentro del documento) y debe dejarse aquí siempre que usemos el cuestionario que la menciona en la sección de usabilidad.

[Hassan08] Hassan Montero, Y. "Guía de Evaluación Heurística de Sitios Web". http://www.nosolousabilidad.com/articulos/heuristica.htm

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

Por orden alfabético, todos los términos que se consideren importantes en la aplicación con una descripción breve de su significado dentro de la aplicación.

- **Término1**: Descripción del significado.
- **Término2**: Descripción del significado.

13.2 Contenido Entregado en el Archivo adjunto

13.2.1 Contenidos

Descripción del contenido del archivo adjunto (directorios y para qué sirve cada cosa), descripción de esta documentación y de cualquier material que adicionalmente se entregue en la presentación. En esta sección se presenta una estructura de directorios de ejemplo para el archivo adjunto que se puede seguir para distribuir todos sus contenidos en el mismo. Conviene por tanto tenerla en cuenta desde el principio de la implementación.

13.2.1.1.1 Introducción

La estructura de directorios del proyecto debe poder recoger todos los ficheros relacionados con el proyecto, clasificándolos por su propósito dentro del mismo. Los tipos más frecuentes son: ficheros fuente, ficheros de configuración, ficheros de documentación…

Se deben crear directorios para contener cada uno de los tipos de ficheros. Tener una estructura estandarizada de los directorios del proyecto es importante por varias razones:

- Ayuda a localizar la información del proyecto. Por ejemplo, los ficheros fuente siempre deben estar en la carpeta *src*.
- Ayuda a los desarrolladores a determinar dónde debe ir cada fichero.
- Permite crear scripts de construcción estandarizados.

13.2.1.1.2 Recomendación estructura general directorios del Archivo adjunto

NOTA: El nombre del Archivo adjunto debe corresponder con el nombre del proyecto.

Directorio	Contenido
./ Directorio raíz del Archivo adjunto	Contiene un fichero leeme.txt explicando toda esta estructura. Se puede incluir *autorun* e icono del proyecto si existe.
./<nombre_proyecto>	Contiene toda la estructura de directorios del proyecto para desarrollo. **Ver la tabla Recomendación de estructura de directorios de desarrollo**. <nombre_proyecto> debe sustituirse por el nombre corto del proyecto.
./instalacion	Ficheros utilizados para la instalación del proyecto.
./documentacion	Contiene toda la documentación asociada al proyecto. Es necesario incluir un fichero con el documento final del proyecto (en formato *.doc* o *.docx* de *Word* o bien *.sxw* de *Open Office*, por ejemplo) además de un fichero *.PDF*
./documentacion/img	Directorio que contiene las imágenes

	utilizadas en la documentación. Estas imágenes tendrán formato *.png* si son capturas de pantalla, *.wmf* si son diagramas o esquemas y *.jpg* sólo si son fotografías.
./documentacion/uml	Ficheros que genera la herramienta (Rose, *ArgoUML*, etc.) con la que se han generado los diagramas UML y de entidad relación.
./presentacion	Directorio que contiene la presentación en *Powerpoint* o equivalente utilizada el día de la defensa del proyecto, si está disponible en el momento de realizar el Archivo adjunto.
./herramientas	Contiene los ficheros de instalación de las herramientas utilizadas para el desarrollo o puesta en marcha del proyecto (lógicamente sólo las que sean distribuibles).
./herramientas/desarrollo	Ficheros de instalación de las herramientas utilizadas en el desarrollo
./herramientas/explotacion	BD, servidor Web y herramientas en general.

13.2.1.1.3 Recomendación de la Estructura de Directorios de "desarrollo"

Se muestra aquí el contenido del directorio de desarrollo de la tabla anterior, incluyendo todos los directorios que deben depender del mismo. Algunos de los elementos sea han incluido suponiendo que se están usando ciertas tecnologías Java. En caso de no usarlas, buscaremos un equivalente existente (si lo hay) en la que estemos usando nosotros.

Directorio	Contenido
./ Directorio raíz de "desarrollo"	Contiene los ficheros de proyecto del IDE utilizado.
./build	Contiene el build.xml de *ant* (si lo usamos). Debemos situarnos dentro para poder invocarlo.
./conf	Contiene los diferentes ficheros de configuración del proyecto. Podría contener distintos subdirectorios, en función de la tecnología usada. En este ejemplo se muestra un ejemplo de un proyecto *Web* hecho con tecnologías *Java*: • **web**: contiene los ficheros de configuración de la aplicación Web (por ejemplo: web.xml). • **ear**: contiene los ficheros de configuración de una aplicación empresarial (por ejemplo: application.xml). • **ejb**: contiene los descriptores de despliegue de los EJB.
./dist	Directorio donde se sitúan los ficheros para la distribución del proyecto. Por ejemplo: los ficheros *.war* o *.ear.*

./doc	Contiene toda la documentación relativa al proyecto, incluyendo los ficheros generados por herramientas de generación de documentación automática como *Javadoc* o similar.
./lib	Bibliotecas externas (*.jar*, *.dll*,…) necesarias para compilar y distribuir, de las que depende este proyecto.
./compile-lib	Bibliotecas externas (*.jar*, *.dll*,…) necesarias para compilar pero que no deseamos distribuir.
./src	Ficheros fuente
./src/java	Todos los ficheros *Java*, lógicamente agrupados en los paquetes correspondientes.
./src/sql	Este directorio contiene los scripts de *SQL* que permiten construir y meter los datos iniciales en la base de datos del proyecto (si existe).
./web	Este directorio contiene los ficheros (*.JSP*, *.ASPX*, *.HTML*,…) de la *Web* (si el proyecto incluyese una).
./web/images	Contiene las imágenes utilizadas por los ficheros de la web del proyecto.
./classes	Directorio donde se guardan los ficheros compilados (como por ejemplo los *.class*)
./test	Directorio base para todos los ficheros utilizados en la automatización del proceso de prueba.
./test/java	Contiene todas las pruebas unitarias utilizadas en el proceso de prueba automatizado.
./test/sql	Scripts *SQL* utilizados en la carga de datos de prueba
./bak	Directorio donde se pueden guardar versiones antiguas de los ficheros fuente del proyecto.

13.2.2 Código Ejecutable e Instalación

Descripción de los contenidos del código ejecutable y de la instalación de la aplicación en un ordenador. Breve manual de instalación y puesta en marcha de la aplicación (solamente unos pasos sencillos que faciliten este proceso, sin explicar nada (para eso está el manual de instalación propiamente dicho).

13.2.3 Ficheros de Configuración

Descripción de todos los ficheros necesarios para poder hacer funcionar la aplicación (ficheros de configuración, ficheros de datos, etc.).

13.3 Índice Alfabético

Este índice alfabético esta generado automáticamente por Word e incluirá todos los términos que nosotros marquemos adecuadamente con la herramienta que *Word* posee para ello (en *Word 2007*, seleccionamos la palabra o frase a indexar y vamos luego a *Referencias* - *Marcar Entrada*. En el cuadro que sale seleccionados luego "*Marcar*", o "*Marcar todas*" para que se busquen todas las apariciones de dicha palabra en el documento). No debemos pues incluir palabras "a mano" en él. Este índice tiene una serie de ejemplos para ilustrar como funciona. No debemos olvidarnos de usar la opción "Actualizar campos" al finalizar la documentación.

NOTA: Quitar esta explicación en la documentación final.

13.4 Código Fuente

El código fuente se incluirá en el archivo adjunto que se ha de subir. En esta sección se indicará la ruta donde se puede encontrar el código fuente y se describirá brevemente su estructura por carpetas y que contiene cada una.