UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN, MENCIÓN EN TELEMÁTICA

Diseño e implementación de una herramienta de gestión de los Trabajos Fin de Grado de la E.T.S.I. de Telecomunicación

Autor:
D. Gonzalo Lara Camarón

Tutores:
D. Carlos Alonso Gómez
Dña. María Jesús Verdú Pérez

Valladolid, 4 de Septiembre de 2018

TÍTULO: **Diseño e implementación de una herramienta de gestión de los Trabajos Fin de Grado de la E.T.S.I. de Telecomunicación**

AUTOR: **D. Gonzalo Lara Camarón**

TUTORES: **D. Carlos Alonso Gómez**
Dña. María Jesús Verdú Pérez

DEPARTAMENTO: **Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE: **D. Juan Pablo de Castro Fernández**
VOCAL: **Dña. María Jesús Verdú Pérez**
SECRETARIO: **Dña. Luisa María Regueras Santos**
SUPLENTE: **D. Francisco Merino Caminero**
SUPLENTE: **D. Jaime Gómez Gil**

FECHA: cinco de septiembre de 2018
CALIFICACIÓN:

Resumen

El presente trabajo de fin de grado analiza el actual sistema de gestión de trabajos de fin de grado que se realiza en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Valladolid y plantea algunos problemas que se derivan del mismo, principalmente la alta carga de trabajo humano que supone. Para dar solución a este problema se plantea el desarrollo de una aplicación web distribuida, que asista y simplifique dicho trabajo de gestión. A parte de esto se proponen otros objetivos deseables como la posibilidad de realizar algunos trámites relacionados con los trabajos de fin de grado de forma telemática a través de dicha aplicación.

Para crear esta aplicación, primero se ha estudiado revisado el estado del arte y las diferentes tecnologías web basadas en Java que podían ser útiles en el desarrollo de la misma. Pasada esta fase, se han analizado los requisitos funcionales y no funcionales del sistema de información que se iba a implementar. Posteriormente se ha procedido al diseño de un software específico que cumpliese con dichos requisitos. Por último, se ha implementado este software, y queda pendiente su entrada en el entorno de producción.

Palabras clave

Trabajo de Fin de Grado, Aplicación Web, Gestión integrada, Aplicación distribuida, Spring, Framework.

Abstract

This work analyzes the current management system of final degree projects that takes place in the Higher Technical School of Telecommunications Engineering of the University of Valladolid and raises some problems deriving from it, mainly the high workload that it entails. To solve this problem, the development of a distributed web application that assists and simplifies the management work is proposed. Apart from this, other desirable objectives are the possibility of carrying out some paperwork related to the final degree projects in a telematic way.

To create this application, the first step was to review the state of the art and the different Java-based web technologies that could be useful in its development. After this phase, the functional and not functional requirements were analyzed. Then, we proceeded to design a specific software that met these requirements. Finally, this software has been implemented and the deployment to the production environment is pending.

Keywords

Final degree project, Web application, Integrated management, Distributed application, Spring, Framework.

ÍNDICE

Índice de Figuras

Índice de diagramas

1. Introducción y objetivos

1.1 Planteamiento del problema

En la Escuela Técnica Superior de Ingenieros de Telecomunicación de Valladolid se imparten dos titulaciones de grado. En estas titulaciones es necesario cursar el Trabajo de Fin de Grado (TFG) para finalizar el plan de estudios y obtener el título académico.

El TFG consiste en una propuesta de realización por parte del alumno de un trabajo original e inédito bajo la supervisión de un tutor académico. Es un trabajo de integración con cuya elaboración el alumno debe demostrar que ha adquirido el conjunto de competencias de grado. El trabajo de fin de Grado deberá permitir evaluar los conocimientos y capacidades adquiridos por el alumno teniendo en cuenta su carácter de prueba global [1].

A nivel de plan de estudios, el TFG es considerado como una asignatura más; sin embargo, este tiene un carácter personalizado para cada estudiante, mientras que el resto de asignaturas tienen un carácter general e igualitario para todos los alumnos que la cursan. Esto se traduce en que cada alumno tendrá que desarrollar un TFG individual y propio que, en general, no va a guardar relación con los TFG que cursen otros alumnos.

Esta circunstancia obliga a la ETSIT a llevar a cabo un sistema de gestión de trabajos de fin de grado. Si describimos de forma muy superficial el sistema, podemos decir que hay diferentes actores implicados: tutores, alumnos, el Comité de Titulaciones de Grado (CTG), secretarías... y cada uno de ellos

con diferentes funciones y trámites asignados: los tutores proponen trabajos, los alumnos los solicitan, los defienden o renuncian a ellos, el CTG ejerce como una especie de supervisor de todo lo relacionado con la gestión de los mismos, las secretarías tienen que tramitar diferentes escritos como por ejemplo la solicitud de defensa de los trabajos, etc...

En resumidas cuentas, hay una serie de trámites y procedimientos implicados, que afectan a diferentes actores en el ciclo de vida de un TFG. Si se tiene en cuenta que hay un número elevado de alumnos realizando simultáneamente su TFG durante un curso académico (al menos, tantos como alumnos que obtienen la titulación) y que es un proceso que se repite todos los años, se empieza a vislumbrar la cantidad de trabajo y de recursos que supone toda la gestión humana que se viene realizando hasta ahora.

A parte de esto, con el sistema actual los trámites y solicitudes relacionadas con los trabajos de fin de grado se realizan principalmente de forma presencial y mediante documentos y formularios impresos en papel. En este aspecto resulta muy interesante modernizar la metodología de trabajo e implementar algún tipo de sistema digital que permita a los usuarios realizar estos trámites de mantera telemática, pudiendo efectuarlos desde cualquier ubicación.

Por estos dos motivos principalmente, es muy conveniente implementar algún tipo de sistema informático para estos propósitos, que simplifique y facilite en la medida de lo posible todos los trámites relacionados con la administración y gestión de los trabajos de fin de grado.

1.2 Objetivos

La aplicación telemática que diseñe e implemente como fruto del presente trabajo de fin de grado deberá servir para alcanzar los siguientes objetivos:

1. **Realización de trámites vía telemática**: Con la puesta en marcha de esta aplicación, se pretende facilitar a los tutores y alumnos del centro la solicitud y realización de los diferentes trámites. Estos podrán efectuarse de forma telemática desde cualquier PC o dispositivo móvil con conexión a Internet en vez de la forma presencial mediante la cual se venían efectuando hasta ahora. Esto va a suponer eliminación del papel, un ahorro de tiempo, y en ocasiones también un ahorro de desplazamientos a la ETSIT cuando estos tengan como único motivo la realización de alguno de estos trámites. En última instancia se conseguirá una mejora en la productividad de dichos actores.

2. Plataforma de gestión centralizada: La aplicación deberá operar como una plataforma de gestión centralizada que aglutine toda la información relacionada con los TFG y la lógica de negocio. El objetivo es crear una lógica digital que emule el sistema de gestión humano que se ha venido utilizando hasta ahora. Esta lógica será la encargada de controlar los diferentes estados en los que se encuentran los trabajos, y en función de eso, los trámites que pueden o no realizarse sobre ellos, así como la actualización del estado de dichos TFG como resultado de dichos trámites.

3. Centro de información: Los miembros del Comité de Titulaciones de Grado tienen un rol de supervisión y coordinación de todo lo relacionado con la administración de los TFG. La aplicación brindará a estos usuarios un panel de control centralizado con toda la información del sistema. Podrán monitorizar todos los trabajos de fin de grado: consultar sus datos, consultar el estado en el que se encuentran y consultar si hay alguna solicitud, trámite o evento pendiente relacionado con ellos, etc. Este panel de control será una herramienta potente y eficaz que facilitará el trabajo de gestión a estos usuarios.

4. Automatización de tareas repetitivas: La aplicación también deberá encargarse de automatizar ciertas tareas con un alto grado de repetición que hasta ahora venían ejecutándose de forma humana. El principal ejemplo de este objetivo es la generación de listados en PDF con los diferentes trabajos almacenados en el sistema, en función de ciertos parámetros y filtros de entrada.

5. Integración con los servicios web de la ETSIT y su base de datos: La aplicación deberá estar integrada con la base de datos de usuarios de la ETSIT. De esta forma los usuarios que necesiten acceder a ella podrán hacerlo mediante sus credenciales UVA sin necesidad de ningún tipo de registro o creación de una nueva cuenta. Además, la información de la base de datos permitirá a la aplicación saber qué tipo de usuario es el que está accediendo y qué rol tiene asignado dentro de la misma (Alumno, Profesor, CTG, Secretarías...). Además de esto, la aplicación también deberá comunicarse con la página web de la ETSIT para publicar en el tablón de anuncios diferentes notificaciones y ficheros PDF relacionados con los TFG de la escuela.

1.3 Estructura de la memoria

La presente memoria comienza planteando el problema de la gestión de los TFG en la ETSIT y de cómo podría mejorarse esta mediante el desarrollo de una aplicación telemática, para lo cual se plantean una serie de objetivos deseables a conseguir mediante la implementación de la misma.

En el capítulo 2 se analizan diferentes tecnologías para el desarrollo de aplicaciones telemáticas con el fin de averiguar cuál o cuáles son las más apropiadas para desarrollar el software. Se explican someramente algunas ideas sobre arquitecturas software como el concepto de aplicación web, que es el paradigma de diseño que mejor se adapta a la problemática y el que va a permitir la consecución de los objetivos antes mencionados con una mayor facilidad. Después se explican algunas ideas sobre el patrón de desarrollo de software Modelo-Vista-Controlador (MVC) y por qué es muy recomendable que el código desarrollado como fruto del presente trabajo cumpla con dicho patrón.

Llegados a este punto se justifica la necesidad de escoger tecnologías de desarrollo que trabajen sobre la plataforma J2EE empleando esta como soporte o marco de trabajo. A mayores se analizan diferentes frameworks y

tecnologías específicas basadas en dicha plataforma para ver cuál de ellas resulta más recomendable. Se analizan el uso de la API Servlets, los framework Spring y Struts, orientados al desarrollo de aplicaciones basadas en el patrón de diseño MVC, y por último las herramientas BPM para la gestión de procesos de negocio. Después de explicar someramente dichas tecnologías, se hace un análisis comparativo de todas ellas para elegir cuál es la más adecuada para la implementación de la aplicación.

En el capítulo 3 se analizan los requerimientos funcionales de la aplicación a desarrollar. Se explica en profundidad qué es lo que tiene que hacer la aplicación mediante el desarrollo de unos casos de uso detallados. A parte de esto también se especifican otros requisitos no funcionales que debe cumplir la aplicación.

Por último, el capítulo 4 aborda todo el proceso de diseño de la aplicación como tal. Se comienza explicando brevemente la metodología empleada. Esta ha consistido básicamente en dividir la aplicación en diferentes funcionalidades individuales que se han ido implementado de forma independiente e iterativa.

A continuación, se detalla el entorno de desarrollo que se ha utilizado para trabajar sobre la aplicación. El entorno de desarrollo consta de un servidor web Tomcat, un servidor de base de datos postgreSQL y el IDE NetBeans.

Por último, se aborda el diseño de la aplicación propiamente dicho, dividiendo este en bloques o partes bien diferenciadas, siguiendo el patrón de diseño MVC.

Se comienza detallando el diseño del controlador para después pasar al diseño del modelo de la aplicación. A partir de ahí se explica el diseño de la base de datos del sistema y por último el diseño referente a las vistas y la interfaz gráfica.

Finalmente, la memoria consta de un capítulo reservado a las conclusiones y a las líneas futuras del proyecto. Este último aspecto es muy importante ya que el trabajo fruto de la presente memoria no es algo cerrado o acabado. Existen numerosas posibilidades de mejora, ya sea modificando las funcionalidades existentes como añadiendo otras nuevas que sean interesantes para los usuarios de la aplicación.

2. Estado del arte y elección de tecnologías subyacentes:

Las tecnologías de la información y la comunicación (TIC) han experimentado un avance vertiginoso e imparable en las últimas décadas. La mejora constante del hardware con nuevos y cada vez más potentes dispositivos como los smartphones, portátiles, tabletas, etc.… unido al avance generalizado de las redes de comunicación, han conseguido que hoy

en día las TIC sean una parte fundamental de nuestra vida y estén presentes en muchos aspectos de la misma.

Hay infinidad de servicios para infinidad de problemas que se le pueden presentar a los ciudadanos. Y esto a su vez hace que haya un amplio abanico de tecnologías, arquitecturas y posibilidades para desarrollar nuevas aplicaciones y servicios basados en las TIC.

Para desarrollar una herramienta que facilite la gestión de los TFG de la ETSIT es preciso elegir previamente una arquitectura, patrón de desarrollo y tecnologías que permitan lograr este objetivo.

2.1 Aplicaciones web

De todas las posibilidades que ofrece el diseño de software hoy en día, la arquitectura de diseño basada en aplicaciones web es la que mejor se adapta al problema descrito en el capítulo anterior y la que más fácilmente y con menor esfuerzo permite conseguir los objetivos propuestos en dicho capítulo.

Una aplicación web es una particularización del modelo de aplicación cliente-servidor. Como puede apreciarse en la Figura 1, en el modelo cliente-servidor hay una máquina llamada servidor que gestiona recursos, mientras que otras máquinas llamadas clientes pueden acceder a esos recursos a través del servidor. [2]

Las aplicaciones web toman como base la arquitectura cliente servidor, pero empleando una tecnología muy asentada, versátil y que está presente en una gran variedad de dispositivos electrónicos de consumo como es la web.

Una aplicación web es una página web dinámica (o una serie de páginas) a la que el usuario accede mediante un navegador web y que utiliza para realizar una tarea específica [3]. Esta es la principal diferencia respecto a una página web convencional, que se suele emplear para consumir contenidos por parte del usuario.

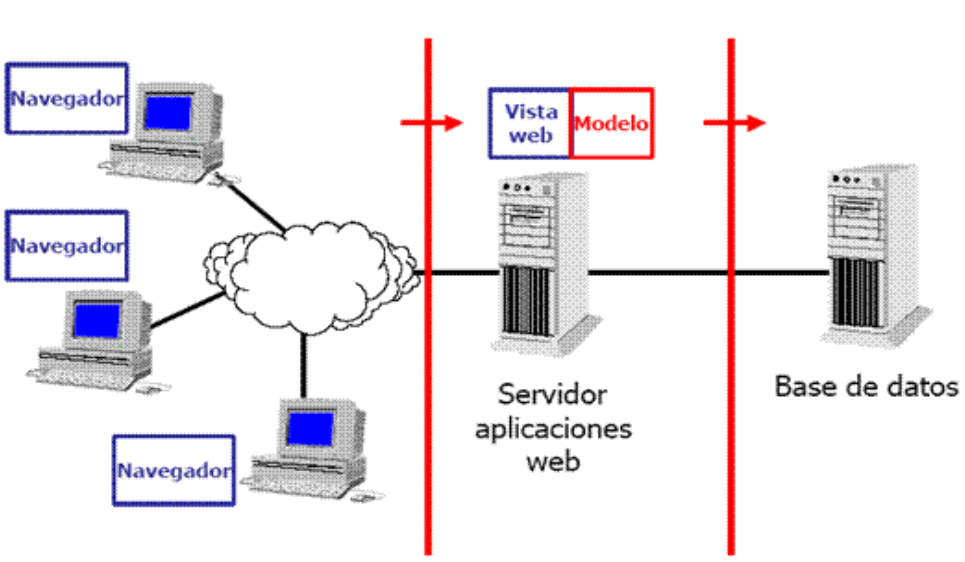

Figura 1 – Modelo Cliente-Servidor

El hecho de elegir la tecnología web supone un buen abanico de ventajas respecto a otras arquitecturas y tecnologías menos extendidas. A continuación, se describen someramente algunas de ellas:

Multiplataforma: La aplicación cliente mediante la cual los usuarios acceden al sistema es un sencillo navegador web. El navegador web se encarga de adaptarse a las particularidades de cada dispositivo (PC, Tableta, teléfono inteligente...) y de cada sistema operativo (Windows, Linux, MAC OS, Android, IOS...) de tal forma que la aplicación desarrollada se podrá ejecutar en todo tipo de máquinas independientemente de su tecnología subyacente con el único requisito de que tengan un navegador web instalado y conexión a Internet disponible.

Actualización: Las aplicaciones web están siempre actualizadas al último lanzamiento. Si se realiza algún cambio en el software, este se actualiza en los ficheros almacenados en servidor de la aplicación. A su vez, el servidor web enviará al cliente la última versión de la aplicación para que esta se ejecute en su navegador.

Inmediatez de acceso: las aplicaciones basadas en web no necesitan ser descargadas e instaladas localmente en las máquinas de los clientes. Casi todos los dispositivos electrónicos ya tienen un navegador web instalado, y cuando un usuario quiera ejecutar la aplicación simplemente abrirá el navegador y accederá a ella a través de su URL.

Menos Bugs: Este tipo de aplicaciones son menos propensas a detenerse inesperadamente. También generan menos problemas técnicos debidos al software o a conflictos ya sean debidos al hardware o a otras aplicaciones.

Acceso remoto: Los usuarios pueden acceder a la aplicación desde cualquier parte del mundo y desde cualquier tipo de máquina con el único requisito de que esta tenga un navegador web y conexión a Internet.

2.2 Patrón de diseño Modelo-Vista-Controlador

El uso de patrones de diseño es una práctica muy recomendable en ingeniería de software. Según [4], los patrones describen un problema que ocurre una y otra vez en un entorno, y determinan la solución a ese problema, de tal modo que pueda utilizarse esta solución un millón de veces sin siquiera hacerlo de la misma manera dos veces.

Utilizar patrones de software es beneficioso porque se emplean fórmulas, prácticas y soluciones que se sabe de antemano que han funcionado en otros escenarios similares. De esta forma es más fácil lograr un diseño coherente para la aplicación, que cumpla con los requisitos funcionales y que tenga un mantenimiento sencillo de cara al futuro.

Para el diseño de la aplicación, se ha hecho uso del patrón de diseño Modelo-Vista-Controlador. El patrón MVC no se desarrolló pensando en los problemas de diseño relacionados con aplicaciones web. Este patrón se describió por primera vez en [5], en el año 1979, para Smalltalk, mucho

antes de que existiese la web tal y como la conocemos hoy en día; como respuesta a la necesidad de construir sistemas software más robustos y con un mayor ciclo de vida, cuando estos requerían el empleo de interfaces de usuario. El patrón MVC está ampliamente extendido en multitud de proyectos y, además, pese a no haberse desarrollado específicamente para ello, se adapta muy bien a los requisitos y funcionalidades que demandan las aplicaciones web.

El patrón Modelo-Vista-Controlador consiste en separar la aplicación en tres bloques funcionales con papeles bien diferenciados y con interfaces de conexión entre estas partes claras y definidas. De esta manera se consigue un código más modular y sencillo de mantener ya que se pueden realizar cambios o actualizar alguno de los bloques dejando inalterado el funcionamiento de los otros, siempre que se respeten las interfaces. La relación entre los tres bloques funcionales se puede ver en la Figura 2.

Modelo: El modelo está formado por un conjunto de clases que representan la información del mundo real que se quiere emular en el sistema. El modelo contiene e implementa las reglas de negocio de la aplicación. Este componente también se encarga de mantener la persistencia de los datos, guardando o recuperando la información independiente del medio que se utilice para almacenarla (ficheros XML, bases de datos relacionales, etc.…). A nivel teórico, un modelo bien diseñado no debería tener conocimiento acerca de la existencia de las vistas y del controlador.

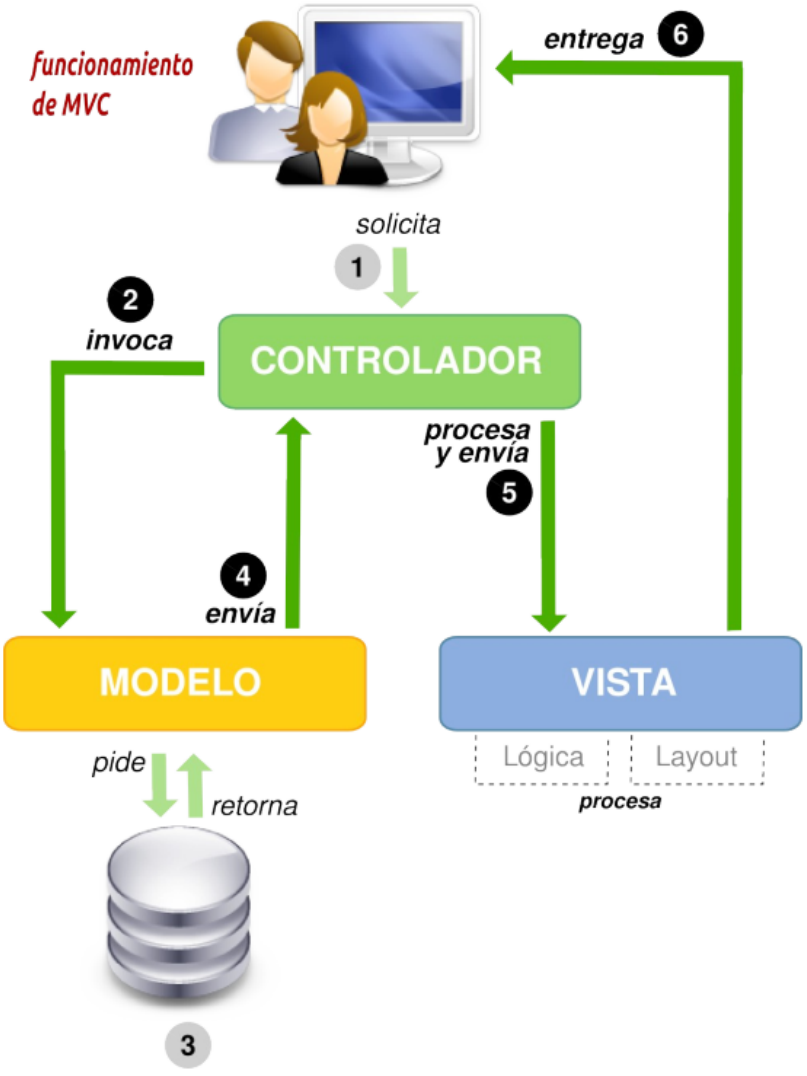

FIGURA 2 – ESQUEMA SOBRE LAS RELACIONES EN EL PATRÓN MODELO VISTA CONTROLADOR

Vista: La vista es la parte que se encarga de representar los datos del modelo al usuario, dándoles el aspecto y formato necesarios para su correcta interpretación. La vista y el modelo tienen una relación de varias a uno. Solo hay un modelo que contiene todos los datos del sistema, pero estos datos se pueden representar mediante muchas vistas diferentes. En el paradigma de diseño basado en aplicaciones web, las vistas contienen el código de presentación, que se va a enviar al navegador para que se represente.

Controlador: El bloque del controlador contiene los componentes necesarios para que la aplicación interaccione con el usuario. El controlador actúa de intermediario entre el usuario, el modelo y la vista. El controlador recoge las peticiones del usuario, interacciona con el modelo trasladándole dichas peticiones y finalmente selecciona qué vista es la adecuada para mostrar los datos fruto de la operación solicitada. Puede haber varios controladores independientes interactuando con el modelo.

La clave del éxito de este patrón es la implementación modular. Cualquier modificación que afecte al dominio, a cómo se hacen las cosas en la organización o en la empresa (modificar datos o aumentar métodos), supondrá cambios en el modelo, y en las interfaces de este con la vista, pero no requerirá cambios en el controlador. Por otra parte, las modificaciones en la parte de la vista no afectan a la implementación del modelo. Se modifica la forma de presentar la información, pero no el tratamiento de la misma.

Para desarrollar una aplicación bajo el patrón de diseño MVC es necesario una mayor dedicación en los tiempos iniciales del desarrollo. Normalmente el patrón exige al programador desarrollar un mayor número de clases que, en otros entornos de desarrollo, no son necesarias. Sin embargo, esta desventaja es muy relativa ya que posteriormente, en la etapa de mantenimiento de la aplicación, una aplicación MVC es mucho más sencilla de mantener, también es más fácil de extender y modificar que una aplicación que no lo implementa.

2.3 Análisis de frameworks Java

Como se explicará más adelante en el apartado 3.2, la aplicación desarrollada fruto del presente trabajo ha de estar basada en Java puesto que el servidor web de producción donde deberá correr la misma ejecuta un contenedor de Servlets.

Según [6], un servlet es una clase java que admite peticiones a través del protocolo HTTP. Los servlets reciben peticiones desde un navegador web, las procesan y devuelven una respuesta al navegador, normalmente en HTML. Para realizar estas tareas podrán utilizar otras clases en lenguaje Java. Estos programas son los intermediarios entre el cliente (casi siempre navegador web) y los datos (bases de datos).

La tecnología Servlets tiene un buen número de ventajas. La primera de ellas es que permite crear aplicaciones web dinámicas basadas en componentes y que son independientes de la plataforma o máquina en la que se ejecuten [7]. Los servlets se ejecutan en un contenedor de aplicaciones con una máquina virtual de java. Eso supone libertad para elegir el sistema operativo de la máquina y el contenedor de aplicaciones que se prefiera.

Por otra parte, trabajar con servlets implica compatibilidad total con la API de Java que es muy extensa y rica en posibilidades, lo cual es de gran ayuda a la hora de diseñar aplicaciones.

Dentro de la especificación Servlets de Java, existen diferentes alternativas para desarrollar aplicaciones web. Sin ánimo de analizar todas ellas, el presente trabajo se ha centrado en estudiar y comparar cuatro posibles opciones bastante representativas y finalmente elegir una de ellas para implementar la aplicación.

2.3.1 Java Servlets

La primera aproximación para desarrollar aplicaciones web basadas en la tecnología de Servlets es trabajar directamente con dicha API de Java, sin framework, capa intermedia o ayuda de ningún tipo. La principal ventaja de esta posibilidad es que se trabaja a bajo nivel, directamente con lenguaje java empleando clases específicas de la plataforma Java EE para programación de aplicaciones web, principalmente la clase HttpServlet. De esta manera no es necesario aprender ni familiarizarse con ningún framework específico, ni con su arquitectura, ni con sus bibliotecas particulares. Obviamente hay que tener un cierto dominio de la propia API Servlets de Java, de sus principales clases y métodos. La curva de aprendizaje de esta alternativa es la más asequible de todas cuantas se contemplan en esta comparativa.

Esta forma de hacer las cosas puede resultar interesante e incluso práctica y recomendable cuando se trata de crear aplicaciones o sistemas sencillos ya que permite una implementación rápida y directa para resolver el problema. El gran inconveniente de este paradigma es que resulta muy ineficiente en cuanto las aplicaciones empiezan a ser grandes y a incorporar bastantes funcionalidades. En estos casos la complejidad en el diseño empieza a no ser manejable.

El principal problema de esta aproximación es que en una misma clase java se va a aglutinar código con diferentes funcionalidades. Código para el acceso a la base de datos, código para generar el diseño de la vista y código para el control de flujo de la aplicación. Es decir, no se estaría aplicando el patrón de diseño Modelo Vista Controlador que se ha mencionado anteriormente.

No obstante, se podría construir la aplicación con una serie de clases estructurales diseñadas e implementadas por el propio desarrollador, que ayudasen a separar las funcionalidades de la aplicación y conseguir un código más claro, separado, específico y modular. Esta posibilidad, a fin de cuentas, sería algo parecido a diseñar e implementar el código de un framework personal para desarrollar aplicaciones web MVC basadas en java.

Esta posible solución sería algo así como programar algo que ya está programado (el framework). Resulta mucho más interesante valerse del trabajo que ya han realizado otras personas, seguramente con más conocimientos y experiencia. Por estos motivos, trabajar directamente con la API servlets de java no es una buena alternativa en este caso y resulta muy recomendable decantarse por algún framework disponible (libre o propietario) que facilite el desarrollo, y el posterior mantenimiento de la aplicación.

2.3.2 Framework Struts

Según la página del proyecto [8], Struts es un framework de código abierto cuyo propósito es facilitar la creación de aplicaciones web basadas en Java. Mediante Struts se pueden crear aplicaciones web que puedan interactuar

con bases de datos o con lógica de negocio que esté implementada en Java. Esta suite se desarrolla como parte del proyecto Jakarta de la *Apache Software Foundation*. Es por tanto un software "open-source" creado por un grupo de voluntarios que comparten valores comunes con respecto al desarrollo colaborativo de código abierto basado en la comunidad.

Struts vio la luz en el año 2000 y desde entonces ha ido ganando popularidad y aceptación hasta convertirse en una de las principales soluciones elegida por los desarrolladores para construir aplicaciones Web REST basadas en Java siguiendo el patrón Modelo Vista Controlador.

Una de las principales ventajas de Struts es que, al ser una tecnología muy extendida y utilizada, tiene una gran comunidad detrás de ella. Esto implica que cuenta con un buen soporte, lo cual puede facilitar bastante la tarea de aprendizaje del framework. Con Struts es más fácil implementar y posteriormente mantener la aplicación desarrollada. Estamos ante una tecnología muy probada, de la que existe gran cantidad de documentación, código y ejemplos.

Struts también ofrece mucha flexibilidad a la hora de diseñar una aplicación ya que es sencillo de integrar con otras tecnologías existentes. Para la capa de persistencia de datos ofrece compatibilidad con diferentes ORM (Object Relation Manager) como por ejemplo Hibernate, Apache Cayenne, o sencillamente empleando JDBC (Java Data Base Conectivity) a bajo nivel.

También ofrece variedad de posibilidades al elegir la tecnología que se encargue del front-end de la aplicación. Lo tradicional hasta hace no mucho tiempo era emplear este framework junto con JSP (JavaServer Pages) para levantar todo el apartado de las vistas, pero recientemente han salido al mercado nuevas opciones como Velociti [9], Freemarker o incluso iText para la generación dinámica de vistas en PDF. De hecho, también puede trabajar con HTML estático y Angular JS corriendo en el lado del cliente.

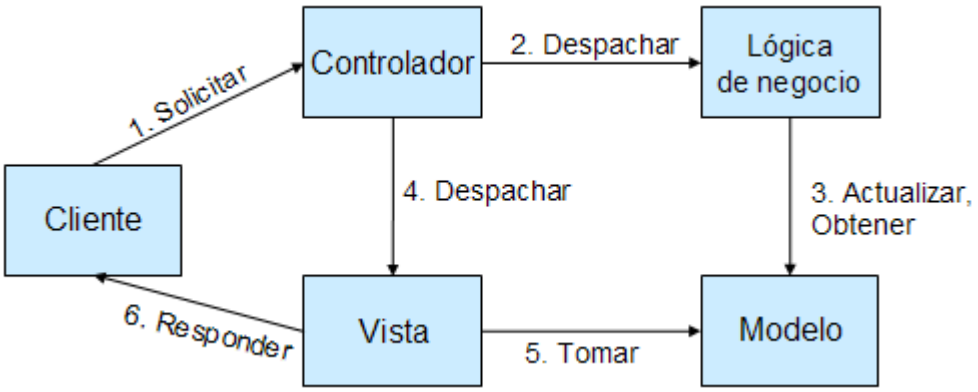

FIGURA 3 - CICLO DE TRABAJO DE STRUTS ANTE UNA PETICIÓN

En la Figura 3 se puede apreciar el flujo de trabajo de una aplicación basada en Struts que implementa el patrón MVC. A continuación, se explican someramente las diferentes fases del mismo:

1. El cliente realiza una petición HTTP al contenedor de aplicaciones, solicitando una URL. Esta URL no hace referencia a un objeto alojado en el servidor como un documento HTTP u otro tipo de recurso, tampoco a un servlet en concreto que atienda esa petición. Todas las URL relacionadas con la aplicación se mapean a un único Servlet interno de Struts que hace el papel de controlador. Para la implementación de este servlet se suele hacer uso de la clase org.apache.struts.action.ActionServlet.

2. El servlet controlador identifica la petición que ha llegado, en función de la URL que se solicita y llama a la acción correspondiente asociada a esa URL.

3. Las acciones son clases Java que se llaman desde el controlador y que implementan la lógica de negocio. Tienen un método llamado “execute”, que es el que se lanza como consecuencia de una petición por parte del cliente. Son las que actualizan el modelo modificando sus datos o recuperándolos para atender la petición. Desde el método “execute” de las clases “action” se tiene acceso a la petición y respuesta HTTP, es decir, el método tiene acceso a diferentes parámetros como por ejemplo los datos contenidos en un formulario o en una URL (métodos POST o GET).

4. El método “execute” de la acción devuelve un objeto “ActionFordward” que especifica la vista concreta que se tiene que utilizar para devolver la petición al cliente, y también los datos del modelo necesarios para rellenar o completar esa vista.

5. Por último, la vista elegida para representar los datos, completa la respuesta a la petición mediante los datos del modelo y se envía al cliente.

Para concluir, Struts es un framework basado en Java pensado para levantar aplicaciones web basadas en el patrón MVC. Tiene una serie de clases internas y de mecanismos de control que facilitan el manejo de eventos de la aplicación, además de otras herramientas que pueden simplificar notablemente el desarrollo de la misma.

2.3.3 Framework Spring Web MVC

Según la página del proyecto [10], Spring Web MVC es un framework basado en la API Servlets de Java. Esta herramienta es uno de los numerosos módulos de los que se compone el framework Spring, que es un paquete muy potente y extenso, que aborda gran diversidad de soluciones para desarrollar software. Spring es una herramienta de código libre y está desarrollada por Pivotal.

Spring Web MVC surgió después de Struts y es por eso que sus desarrolladores se fijaron en todo lo bueno que había funcionado en este framework para crear su propio producto, pero sobre todo mejorando aquello que no funcionaba del todo bien en el software de Apache.

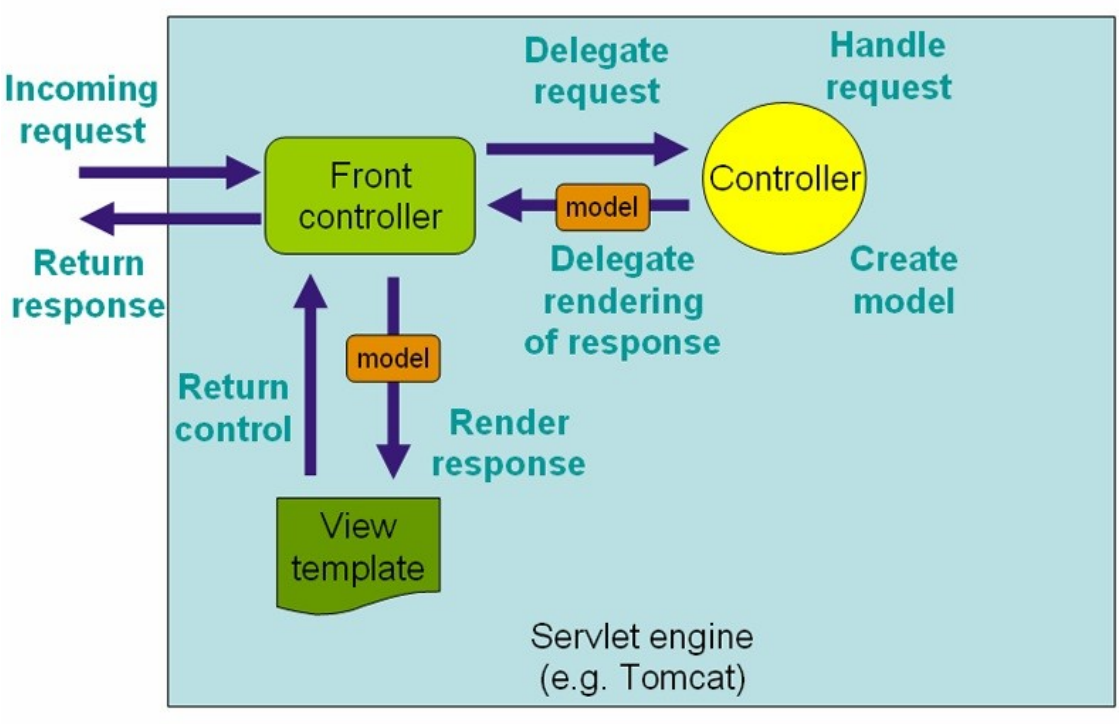

FIGURA 4 - EL FLUJO DE TRABAJO DE PROCESAMIENTO DE SOLICITUDES EN SPRING WEB MVC

Al igual que Struts, Spring Web está pensado para desarrollar aplicaciones web basadas en la plataforma servlets, por lo que es compatible con la amplia API de Java, que cuenta con módulos y bibliotecas de lo más potentes y variadas. Además, también está orientado al desarrollo de aplicaciones siguiendo el patrón de diseño Modelo Vista Controlador del que ya se han comentado sus ventajas en apartados anteriores.

Este framework implementa la idea del patrón Front-Controller, que es una particularización del paradigma Modelo Vista Controlador. Este patrón es ampliamente utilizado en el ámbito web y consiste en emplear un controlador como el punto de entrada de todas las peticiones al sistema y en función del tipo de petición que recibe, esta se delega a unos u otros componentes de la aplicación.

En la Figura 4 se puede observar el flujo de procesamiento de las peticiones HTTP en Spring, bastante similar al de Struts. Todas las peticiones HTTP relacionadas con la aplicación se dirigen al Front-Controller, que no es más que un servlet Java cuya implementación viene dada por el framework. En el caso de Spring, se emplea la clase DispacherServlet.

El Front-Controller, mediante la URL que solicita el cliente, llama al controlador propio de la aplicación web. Esto se hace mediante el

HandlerMaping, que asocia las URL entrantes con la clase controladora que se va a encargar de atenderlas.

El controlador de la aplicación (circulo amarillo del esquema) es una clase java que implementa varios métodos, cada uno de los cuales se encarga de atender una URL diferente. Por lo que en función de la URL que solicita el cliente, se llama a uno de ellos o a otro.

El método que atiende una URL concreta se encarga de realizar las operaciones necesarias sobre el modelo, ya sea modificando sus datos o solicitándolos. Una vez realizado esto, el método devuelve un objeto de tipo Model al Front-Controller. En este objeto se incluyen tanto el nombre de la vista que hay que emplear para responder a la petición HTTP como los datos y variables necesarios para confeccionar y completar esa vista.

Llegados a este punto, el Front-Controller hace uso de un ViewResolver que se encarga de averiguar qué tipo de vista es la que hay que emplear, en función del nombre lógico de la vista que devuelve el objeto Model. Normalmente esta vista será un fichero JSP, aunque Spring es muy flexible en este aspecto y también puede ser una clase Java que confeccione un PDF u otro tipo de documento dinámico.

Por último, el Front-Controller responde a la petición del cliente enviándole la vista apropiada, que se ha confeccionado mediante los datos que devolvió el modelo.

2.3.4 Herramientas de gestión de procesos de negocio (BPM)

La gestión de procesos de negocio o BPM por sus siglas en inglés (Bussines Process Management) es una nueva metodología que está cobrando bastante relevancia en los últimos tiempos. No es un nuevo patrón de diseño de software, arquitectura o innovadora forma de organizar el proceso de desarrollo de un proyecto. BPM es una filosofía de negocio. Es una nueva forma de hacer las cosas en la empresa.

Según [11], BPM hace un enfoque orientado a los procesos productivos de una empresa a diferencia de la clásica organización jerárquica basada en departamentos. Se pasa de una estructura vertical a algo transversal y continuo donde lo importante es cómo son los procesos que generan valor en la empresa y cómo pueden optimizarse estos, ahorrando tiempo o recursos para aumentar la competitividad.

Para conseguir esto, con BPM se modelan los diferentes procesos de negocio, que a su vez se subdividen en diferentes tareas que incluyen personas, aplicaciones, eventos y actividades. Los procesos son estudiados y modelados, pero además se monitorizan constantemente con diferentes métricas y parámetros para comprobar que todo funciona adecuadamente.

Para que todo esto sea viable, rentable y eficaz hacen falta una serie de herramientas software que den soporte a BPM. Es lo que se conoce como

BPMS o *Bussines Process Management* Software. Hay diferentes suites BPMS en el mercado. Algunas son libres y gratuitas como Camunda y otras son de pago, aunque el objetivo de todas ellas es similar. Estas suites BPMS cuentan con diferentes herramientas gráficas, como un gestor de procesos, diferentes asistentes para el modelado de procesos y otro tipo de software auxiliar.

Por lo tanto, BPM es una forma de organización productiva para las empresas que incorpora las TIC como parte fundamental que vertebra ese nuevo paradigma. Hay un software BPM orientado a modelar y a gestionar esta nueva forma de organización productiva, y por último está el software y las aplicaciones específicas para el desarrollo de las tareas propias de la empresa.

Llegados a este punto queda claro que una solución basada en BPM no es adecuada para el problema que se está tratando de resolver con el presente trabajo. BPM está orientado al ámbito empresarial y a cómo organizar tareas, personas, aplicaciones y recursos para ser lo más competitivos posibles. El objetivo de este trabajo es desarrollar una plataforma para la gestión integrada de los TFG de la ETSIT. En ese contexto no hay procesos productivos, no hay mano de obra implicada y no hay una cadena de valor.

No merece la pena utilizar una herramienta BPM o paradigmas basados en gestión de procesos para lograr los objetivos requeridos. La complejidad de este tipo de tecnología es bastante elevada ya que ofrece una solución integral a los problemas organizativos de una empresa. Es un sistema complicado compuesto de numerosos módulos. Otro problema derivado de este tipo de herramientas es que no están muy extendidas en el mercado ni hay una gran comunidad detrás de ellas creando valor añadido como pueden ser guías, ejemplos, documentación, ayuda en foros etc. Empezar a desarrollar aplicaciones con esta tecnología implica superar una barrera de entrada demasiado elevada.

2.5 Análisis comparativo

Una vez analizadas estas cuatro tecnologías, se va a elegir cuál de ellas es la más conveniente para el diseño y desarrollo de la aplicación.

Emplear la API Servlets de Java a bajo nivel no es recomendable puesto que la complejidad fruto del desarrollo sería demasiado elevada sin ningún tipo de framework intermedio que facilite el trabajo. Además, la carga de trabajo derivada del mantenimiento futuro de la aplicación también se incrementaría notablemente.

Por otra parte, también se ha estudiado someramente el paradigma de diseño basado en gestión de procesos o BPM. Esta tecnología no está pensada para el cometido que aquí se está buscando. BPM hace un enfoque integral de los procesos productivos de una compañía o empresa para mejorar estos. El problema de la gestión de los TFG de la ETSIT es un caso diferente, es un escenario diferente. El problema que se pretende resolver

con el presente trabajo es un problema puntual que afecta a un ámbito muy concreto dentro de la escuela. No tiene sentido aplicar herramientas globales que tengan en cuenta una visión de conjunto de la organización.

Las dos alternativas restantes son Struts y Spring Web MVC. A favor de Struts se puede decir que es la tecnología con la que está implementada la página corporativa de la ETSIT. Esto es interesante de cara a simplificar el trabajo de mantenimiento y actualización de ambas plataformas, ya que la persona que se encargue de ello solo tendría que familiarizarse con el funcionamiento de un sólo framework en vez de dos.

Según [12] ambos frameworks tienen objetos responsables del manejo de eventos y peticiones de la aplicación y de la llamada a la lógica contenida en el modelo que proceda. En Struts se conocen como Action, mientras que en Spring Web MVC se llama Controller. La diferencia radica en que en Struts las acciones se instancian cada vez que se recibe una petición, mientras que en Spring MVC el controlador se instancia una sola vez al desplegarse la aplicación en el servidor y se almacena en memoria. Esto implica que Spring Web MVC framework es bastante más eficiente que Struts a la hora de manejar las peticiones HTTP.

Algunas ventajas de Struts son:

- Es más sencillo
- Es fácil de integrar con otros plug-ins
- Mejor soporte a características basadas en etiquetas
- Soporta diferentes tecnologías para generar vistas

Por el contrario, Struts peca de una documentación menos completa y extensa que su competidor y, además, en ciertos casos presenta problemas de compatibilidad.

Spring MVC por otra parte ofrece:

- Una separación más clara entre el controlador, los Java Beans que componen el modelo y las vistas
- Es más flexible puesto que forma parte de la plataforma Spring que está compuesta por diferentes módulos que ofrecen un amplio abanico de soluciones
- El código es más fácil de testear, en parte gracias a la inversión de control (IoC), también conocida como inyección de dependencias
- No proporciona un framework para el desarrollo del modelo. Esto facilita a los desarrolladores la implementación de la parte del controlador y las vistas.

La parte más negativa de Spring es que se trata de un framework que abarca ámbitos de desarrollo muy variados. Precisamente en esa

riqueza de soluciones radica su mayor debilidad: es demasiado amplio y complejo.

En conclusión, tanto Spring Web MVC como Struts son dos herramientas que se adaptan perfectamente a los requerimientos del presente proyecto. Ambas son similares en su objetivo: El desarrollo de aplicaciones web modulares basadas en el patrón MVC. Llegados a este punto no hay una decisión mala, sino más bien un conjunto de gustos o preferencias. En este caso se va a optar por desarrollar la aplicación empleando Spring Web MVC principalmente porque tiene una documentación más completa, más cantidad de ejemplos y una mayor comunidad detrás para resolver dudas relacionadas con la implementación.

3. Análisis de requisitos

3.1 Descripción del sistema a implementar

En este apartado se describe minuciosamente cómo debe de ser el comportamiento del sistema de información que se va a implementar como resultado del presente trabajo de fin de grado.

La plataforma va a tener una arquitectura de aplicación web. Los diferentes usuarios que vayan a interaccionar con la misma deberán acceder a ella a través de un navegador web. Para la autenticación de los usuarios se hará uso del servidor LDAP de la UVA en el que tanto alumnos como profesores, como personal administrativo, tienen ya una cuenta creada, evitando así la molestia a los usuarios de tener que crear diferentes cuentas para los diferentes servicios relacionados con la universidad.

A continuación se explican detalladamente los diferentes atributos que conforman un TFG. Estos campos son necesarios para tener toda la información referente a los trabajos perfectamente clasificada dentro de la aplicación. Además, de algunos de estos atributos, como el estado, va a depender el que se puedan o no realizar la mayoría de trámites a los que da soporte la plataforma.

- ID: identificador unívoco del TFG. No puede haber dos trabajos con el mismo.
- Estado: Durante su ciclo de vida, un TFG atraviesa diferentes estados, y en cada uno de ellos tiene una serie de trámites posibles asociados y otros que no están disponibles.
- Fecha de creación: La fecha en la que se crea la propuesta de TFG.
- Título
- Tutor/es: Por defecto es el profesor que crea la propuesta de TFG. Puede modificarse de tal forma que un profesor cree una

propuesta que no le tenga a él como tutor sino a otra persona que también imparta docencia en la ETSIT.

- Segundo tutor: Un TFG puede tener dos tutores. El segundo de ellos no tiene por qué impartir docencia en la ETSIT. En caso de que el segundo tutor no pertenezca a la UVA, se almacena su nombre y apellidos, así como la empresa o corporación a la que pertenece.
- Departamento. Uno de los departamentos que imparten docencia en la ETSIT.
- Resumen oferta: Se explica brevemente en qué consiste la propuesta de trabajo
- Titulación-mención para la que se oferta. Puede ser una o varias de las siguientes:
  - Grado en Ingeniería Tecnologías Específicas de Telecomunicación - Mención en Sistemas de Telecomunicación
  - Grado en Ingeniería Tecnologías Específicas de Telecomunicación - Mención en Sistemas Electrónicos
  - Grado en Ingeniería Tecnologías Específicas de Telecomunicación - Mención en Telemática
  - Grado en Ingeniería en Tecnologías de Telecomunicación
  - Asociado a prácticas en empresa: Si/No/Las dos posibilidades
- Comisión Evaluadora: Está compuesta por cinco profesores con docencia en la ETSIT con los siguientes roles:
  - Presidente
  - Vocal
  - Secretario
  - Suplente1
  - Suplente2
- Pre-Asignado: Puede estar acordado que el TFG se asigne a un alumno en concreto antes del momento de su creación. Dicho estudiante tendrá preferencia al optar por ella.

El ciclo de vida de los TFG puede apreciarse en la Figura 5.

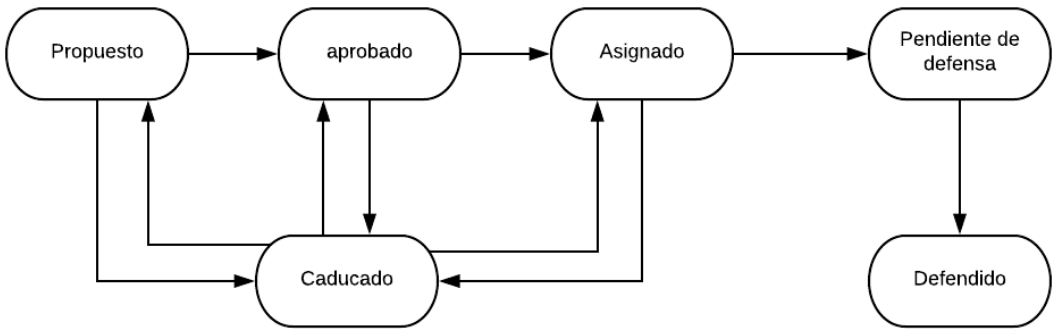

FIGURA 5 - ESQUEMA DE LOS DIFERENTES ESTADOS QUE PUEDE ATRAVESAR UN TFG

Un tutor crea un TFG nuevo. Su estado de origen es **propuesto**

Un TFG **propuesto** puede:

- Pasar a estado **aprobado** si el CTG decide que la propuesta es apta para que un alumno realice su trabajo sobre ella.
- Pasar a estado **caducado** si la fecha actual es superior a su fecha de validez.

Un TFG **aprobado** puede:

- Pasar a estado **asignado** si un alumno lo solicita y el CTG decide asignárselo.
- Pasar a estado **caducado** si la fecha actual es superior a su fecha de validez.

Un TFG **asignado** puede:

- Pasar a estado **pendiente de defensa** si el alumno solicita defenderlo.
- Pasar a estado **caducado** si la fecha actual es superior a su fecha de validez.
- Pasar a estado **caducado** si se aprueba la renuncia de un alumno a un TFG que tiene asignado.

Un TFG **caducado** puede:

- Pasar a estado **propuesto** si el tutor lo renueva.
- Pasar a estado **asignado** si el CTG renueva su asignación a un alumno.

Por último, Un TFG **pendiente de defensa** puede pasar a estado **defendido** si la fecha actual es superior a la fecha de defensa. El estado defendido es el estado final en el que se termina el ciclo de vida del mismo.

Para su normal funcionamiento la aplicación se rige por una serie de plazos y fechas estipuladas que condicionan algunos de los trámites o acciones. Dichas fechas podrán ser modificadas por el CTG o el administrador de la plataforma.

Los periodos son los siguientes:

- **Envío de propuestas de TFG:** Es el periodo durante el cual los profesores de la ETSIT deberán enviar sus propuestas de TFG.
- **Plazo oficial de solicitud de TFG**: Es el periodo durante el cual los alumnos pueden solicitar un TFG de la oferta publicitada oficialmente

Además de estos periodos, la plataforma también tiene que registrar las siguientes fechas:

- Curso académico actual.
- Curso académico siguiente.
- Fecha finalización curso académico actual.
- Fecha finalización curso académico siguiente.

Las fechas de finalización de los cursos académicos son importantes puesto que van a ser los parámetros de caducidad por los que se rige el estado de los TFG.

El resto de trámites relacionados con la plataforma no están regidos por plazos o periodos, puesto que la ETSIT ofrece flexibilidad para la tramitación de los TFG.

A continuación, se recogen los casos de uso de todos los procesos que se podrán llevar a cabo en la plataforma. Se describe de forma exhaustiva cual es la interacción deseada entre un usuario y la aplicación en cada uno de esos casos y cuáles son los resultados fruto de dicha interacción.

Título:	Crear propuesta de TFG (tutor)	
Precondición:	El usuario tutor se ha identificado previamente como personal docente de la ETSIT mediante sus credenciales UVA.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario tutor solicite al mismo la creación de una nueva propuesta de TFG.	
Secuencia normal	**Paso**	**Acción**
	1	El tutor solicita al sistema la creación de un nuevo TFG.
	2	El sistema solicita que el tutor introduzca los siguientes datos en el formulario: título del TFG, tutor, Departamento, si el TFG está asociado a prácticas en empresa, resumen, palabras clave, titulaciones para las que se oferta, si está pre-asignado a un estudiante y a qué estudiante si fuera el caso y por último los miembros de la comisión evaluadora.
	3	El tutor introduce los datos solicitados.
	4	El tutor envía la solicitud.
	5	El sistema almacena la propuesta de TFG en la base de datos e indica al tutor que se ha creado con éxito.
Post-condición	Si la propuesta de TFG se crea fuera del periodo de envío de propuestas establecido en el sistema, el sistema notifica al CTG	

	la creación de dicha propuesta mediante un correo electrónico.		
Excepciones	**Paso**	**Acción**	
	4	Si el tutor ha dejado sin completar algún campo obligatorio del formulario o los ha completado incorrectamente. Si se indica un alumno preasignado al TFG y dicho alumno ya tiene asignado o preasignado un TFG diferente.	
		E.1	El sistema notificará al tutor el error cometido en la introducción de los datos.
		E.2	El tutor comprueba el formulario y corrige los errores.
		E.3	El tutor envía el formulario nuevamente.
		E.4	El sistema comprueba la solicitud. Si los datos no son correctos se vuelve al paso E.1. Si son correctos, el sistema almacena la propuesta de TFG en la base de datos e indica al tutor que se ha creado con éxito.

Título:	Modificar TFG (tutor)	
Precondición:	El usuario tutor se ha identificado previamente como personal docente de la ETSIT mediante sus credenciales UVA. El usuario tiene algún TFG asociado a él, susceptible de ser modificado. No se puede modificar TFG con estado pendiente de defensa o defendido.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario tutor solicite al sistema la modificación de un TFG.	
Secuencia normal	**Paso**	**Acción**
	1	El tutor selecciona en el sistema el TFG que desea modificar entre una lista de los diferentes TFG que ha creado o que están asociados a su usuario.
	2	El sistema muestra un formulario cargado con los datos del TFG a modificar.
	3	El tutor modifica alguno de los campos del TFG que estén permitidos.
	4	El tutor envía el formulario con los datos modificados.

	5	El sistema almacena las modificaciones en la base de datos e indica al usuario que el TFG se ha modificado con éxito.	
Excepciones	**Paso**	**Acción**	
	4	Si el tutor ha dejado sin completar algún campo obligatorio del formulario o los ha completado incorrectamente.	
		E.1	El sistema notificará al tutor el error cometido en la introducción de los datos
		E.2	El tutor comprueba el formulario y corrige los errores.
		E.3	El tutor envía el formulario nuevamente
		E.4	El sistema comprueba la solicitud. Si los datos no son correctos se vuelve al paso E.1. Si son correctos, se almacenan los cambios en la base de datos y se informa al tutor que la operación se ha realizado con éxito.
	5	Si el TFG está aprobado por el CTG o asignado a algún alumno.	
		E.1	Se notifica al tutor que las modificaciones solicitadas se enviarán al CTG para que las estudie. El sistema envía un correo electrónico al CTG informando de la solicitud de modificación del TFG. El sistema almacena en la base de datos los cambios propuestos por el tutor.

Título:	Eliminar TFG (tutor)	
Precondición:	El usuario tutor se ha identificado previamente como personal docente de la ETSIT mediante sus credenciales UVA. El tutor tiene algún TFG asociado a él, susceptible de ser eliminado. Son susceptibles de ser eliminados los TFG que tengan estados: propuesto, aprobado, caducado.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario tutor solicite al mismo la eliminación de un TFG	
Secuencia normal	**Paso**	**Acción**
	1	El tutor selecciona en el sistema el TFG que desea eliminar entre una lista de los diferentes TFG que estén asociados a él.
	2	El sistema elimina el TFG de la base de datos y notifica al tutor que la operación se ha realizado correctamente.

Excepciones	**Paso**	**Acción**	
	2	Si el TFG tiene estado APROBADO.	
		E.1	El sistema notifica al tutor que la solicitud de eliminación se enviará al CTG para que la apruebe.
		E.2	El sistema envía un correo electrónico al CTG indicando que un tutor ha solicitado la eliminación de un TFG con estado APROBADO. En el correo se incluyen los datos tanto del tutor como del TFG que se pretende eliminar.

Título:	Nueva propuesta de TFG (Estudiante)	
Precondición:	El usuario se ha identificado previamente como alumno de la ETSIT mediante sus credenciales UVA. El alumno no tiene asignado o preasignado ningún TFG.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando un alumno solicite crear una propuesta de tema para un TFG.	
Secuencia normal	**Paso**	**Acción**
	1	El alumno solicita al sistema proponer un nuevo tema para un TFG.
	2	El sistema solicita que el alumno introduzca los siguientes datos en el formulario: título del TFG, tutor, resumen, titulación y mención (si procede) del alumno y si el TFG va a estar asociado a prácticas en empresa.
	3	El alumno introduce los datos solicitados.
	4	El alumno envía la solicitud.
	5	El sistema almacena la propuesta en la base de datos. El sistema indica al alumno que la propuesta de TFG se ha creado con éxito.
Postcondición	El sistema envía un correo electrónico al CTG indicando la creación de una nueva propuesta de TFG por parte de un alumno.	
Excepciones	**Paso**	**Acción**

	4	Si el alumno ha dejado sin completar algún campo obligatorio del formulario o los ha completado incorrectamente.	
		E.1	El sistema notificará al alumno el error cometido en la introducción de los datos
		E.2	El alumno comprueba el formulario y corrige los errores.
		E.3	El alumno envía el formulario nuevamente
		E.4	El sistema comprueba la solicitud. Si los datos no son correctos se vuelve al paso E.1. Si son correctos, se comunica al alumno que la propuesta de TFG se ha creado con éxito. Se almacena la propuesta en la base de datos

Título:	Listar propuestas de TFG (CTG)		
Precondición:	El usuario se ha identificado previamente como miembro del CTG mediante un login específico para la cuenta de CTG.		
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario CTG solicite listar todos los TFG contenidos en la base de datos del sistema que tengan estado PROPUESTO.		
Secuencia normal	**Paso**	**Acción**	
	1	El usuario solicita al sistema el listado en PDF de los TFG con estado PROPUESTO.	
	2	El sistema devuelve un documento en PDF con la lista de los TFG propuestos y los siguientes datos de cada TFG: ID, título, tutor, Departamento, Titulación y mención (si procede) para la que se oferta, si está asociado a prácticas y si está preasignado a un alumno.	
Excepciones	**Paso**	**Acción**	
	2	Si no hay ningún TFG con estado propuesto en el sistema	
		E.1	El sistema notificará al usuario que no hay ningún TFG propuesto en el sistema.

Título:	Aprobar TFG (CTG)	
Precondición:	El usuario se ha identificado previamente como miembro del CTG mediante un login específico para la cuenta de CTG.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario CTG solicite la aprobación de un TFG de la plataforma que tuviese un estado PROPUESTO.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema que liste todos los TFG con estado PROPUESTO.
	2	El usuario selecciona de la lista el TFG o lista de TFGs que desea aprobar.
	3	El sistema modifica el estado del TFG en la base de datos y notifica al usuario que la solicitud se ha realizado con éxito.

Título:	Modificar TFG (CTG)	
Precondición:	El usuario se ha identificado previamente como miembro del CTG mediante un login específico para la cuenta de CTG. No pueden modificarse los TFG con estado PENDIENTE DE DEFENSA o DEFENDIDO.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario CTG solicite la modificación de un TFG de la plataforma.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema que se listen todos los TFG con un determinado estado (propuesto, aprobado, asignado o caducado).
	2	El sistema devuelve al usuario una lista con todos los TFGs que tienen el estado seleccionado en el paso anterior y a mayores muestras algunos atributos significativos de los mismos.
	3	El usuario selecciona el TFG de la lista que desea modificar.
	4	El sistema devuelve un formulario completado con los datos del TFG que se desea modificar.

	5	El usuario modifica los campos del formulario que precise.	
	6	El usuario envía el formulario.	
	7	El sistema almacena los cambios en la base de datos y notifica al usuario que su solicitud se ha completado correctamente.	
Excepciones	**Paso**	**Acción**	
	4	Si no se han introducido datos en los campos obligatorios del formulario o hay algún campo completado erróneamente.	
		E.1	El sistema notificará al usuario los errores presentes en el formulario.
		E.2	El usuario modifica los datos introducidos y corrige los errores existentes.
		E.3	El usuario envía nuevamente el formulario
		E.4	Si los datos introducidos no son correctos se vuelve al paso E.1. Si todo está bien el sistema guarda las modificaciones en la base de datos y notifica al usuario que los cambios se han realizado satisfactoriamente.

Título:	Crear solicitud de TFG (Alumno)	
Precondición:	El usuario se ha identificado previamente como alumno de la ETSIT mediante sus credenciales UVA. El alumno no tiene asignado ningún TFG.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando un alumno desee crear una solicitud de TFG.	
Secuencia normal	**Paso**	**Acción**
	1	El alumno solicita al sistema la creación de una solicitud de TFG.

	2	El sistema devuelve un listado con todos los TFG aprobados susceptibles de ser solicitados. En dicho listado se muestra la siguiente información de los TFG: título, tutor, departamento, mención para la que se ofrece, resumen, si está asociado a prácticas en empresa y si está pre asignado ya a un estudiante.	
	3	El alumno solicita los TFG de la lista que le puedan interesar (hasta un máximo de 10), pudiendo ordenar estos en función de su preferencia.	
	4	Cuando el alumno ha terminado de confeccionar su lista de trabajos solicitados y los ha ordenado según sus prioridades, envía la lista.	
	5	El sistema muestra una pantalla nueva en la que aparece un formulario con datos que hay que cumplimentar para realizar la solicitud de TFG: Titulación que está cursando el alumno, dirección postal, correo electrónico y número de teléfono. El DNI del alumno se muestra en dicho formulario, pero no es modificable.	
	6	El alumno completa los datos del formulario y lo envía.	
	7	El sistema muestra una pantalla nueva en la que solicita al alumno que suba un fichero (JPG, PDF etc..) que justifique que está matriculado en el TFG y que por tanto se le puede asignar un trabajo.	
	8	El alumno selecciona un fichero de su sistema (PC, Smartphone, etc.) con dicho resguardo y lo sube al servidor.	
	9	El sistema notifica al alumno que su solicitud se ha creado correctamente. A mayores envía un correo a la dirección del CTG indicando que un alumno ha creado una solicitud de TFG nueva.	
Postcondición	Si se está fuera del plazo de envío de solicitudes de TFG, el sistema envía un correo electrónico al CTG indicando que un alumno ha realizado una solicitud fuera de plazo.		
Excepciones	**Paso**	**Acción**	
	3	Si el alumno ya ha solicitado diez TFG y quiere solicitar más.	
		E.1	El sistema notificará al alumno que ha llegado al máximo de TFG que puede incluir en su solicitud y por tanto para añadir otro TFG a la misma es preciso que antes elimine alguno de los que ya ha elegido.

		E.2	El alumno puede eliminar algún TFG de los que ya ha seleccionado e incluir uno diferente o por el contrario dejar los inicialmente solicitados. Se continua en el paso 4.
	3	Si el alumno no ha seleccionado que desea realizar ningún TFG.	
		E.1	El sistema notifica al alumno que tiene que elegir al menos un TFG para poder crear su solicitud.
		E.2	El alumno selecciona uno o más TFG de la lista de trabajos disponibles. Continua en el paso 4.
	6	Si el alumno no ha completado correctamente los datos que se le solicitan en el formulario, o ha dejado algún campo vacío.	
		E.1	El sistema notifica al alumno el error en los datos de la solicitud.
		E.2	El alumno completa adecuadamente el formulario y lo envía de nuevo.
		E.3	El sistema comprueba que el formulario se ha completado correctamente. Si no es así se vuelve al paso E.1. Si por el contrario el formulario se ha completado correctamente, se vuelve al paso 7.
	8	Si el alumno no ha subido ningún fichero al servidor.	
		E.1	El sistema notifica al alumno que tiene que adjuntar el justificante de matrícula.
		E.2	El alumno selecciona un fichero válido de su máquina y lo sube al servidor.
		E.3	El sistema almacena la solicitud de TFG en la base de datos e informa al alumno que su petición se ha realizado con éxito.

Título:	Modificar solicitud de TFG (Alumno)
Precondición:	El usuario se ha identificado previamente como alumno de la ETSIT mediante sus credenciales UVA. El alumno tiene que haber creado una solicitud de TFG previamente Este caso de uso sólo puede realizarse dentro del plazo oficial de

	solicitud de TFG		
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando un alumno desee modificar su solicitud de TFG.		
Secuencia normal	**Paso**	**Acción**	
	1	El usuario solicita al sistema la modificación de su solicitud de TFG	
	2	El sistema devuelve un formulario cargado con los datos de su solicitud y el listado de TFG elegido. En dicho listado se muestra la siguiente información de los TFG: título, tutor, departamento, mención para la que se ofrece, resumen, si está asociado a prácticas y si está pre asignado ya a un estudiante.	
	3	El usuario modifica los datos que desee de su solicitud, también puede modificar los TFG de la lista de trabajos solicitados, así como su orden de preferencia.	
	4	El usuario confirma la solicitud	
	5	El sistema notifica al usuario que su solicitud se ha modificado correctamente	
Excepciones	**Paso**	**Acción**	
	3	Si el usuario ya ha solicitado 10 TFG y quiere añadir más	
		E.1	El sistema notificará al usuario que ha llegado al máximo de TFG que puede incluir en su solicitud y por tanto para añadir otro TFG a la misma, será preciso que antes elimine alguno de los que ya ha elegido
		E.2	El usuario puede eliminar algún TFG de los que ya ha seleccionado e incluir uno diferente o por el contrario dejar los inicialmente solicitados. Se continua en el paso 4.
	5	Si el alumno no ha completado correctamente los datos que se le solicitan, o ha dejado algún campo vacío o su lista de TFG solicitados está vacía	
		E.1	El sistema notifica al usuario el error en los datos de la solicitud
		E.2	El usuario corrige el error
		E.3	El sistema comprueba que el error se ha subsanado y notifica al usuario que su solicitud se ha creado correctamente.

Título:	Eliminar solicitud de TFG (Alumno)		
Precondición:	El usuario se ha identificado previamente como alumno de la ETSIT mediante sus credenciales UVA. El alumno ha creado previamente una solicitud de TFG. Este caso de uso sólo puede realizarse dentro del plazo oficial de solicitud de TFG.		
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando un alumno desee eliminar su solicitud de TFG.		
Secuencia normal	**Paso**	**Acción**	
	1	El alumno solicita al sistema eliminar su solicitud de TFG.	
	2	El sistema notifica al usuario que su solicitud se ha eliminado correctamente.	
Excepciones	**Paso**	**Acción**	
	3	Si el usuario no ha creado ninguna solicitud o si esta ya ha sido tramitada por el TFG.	
		E.1	El sistema notificará al usuario que no tiene ninguna solicitud de TFG y que por tanto no se puede borrar.

Título:	Asignar TFG (CTG)	
Precondición:	El usuario se ha identificado previamente como miembro del CTG mediante un login específico para la cuenta de CTG.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario CTG desee asignar un TFG a un alumno.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la acción de asignar TFG
	2	El sistema muestra un listado con todas las solicitudes de TFG que han creado los alumnos. En ese listado se muestra el DNI del alumno que realiza la solicitud, la fecha de creación, la titulación del alumno y la lista de TFG que solicita el alumno en orden de preferencia.
	3	El usuario elige un TFG de la lista de trabajos solicitados por el alumno para asignárselo. También puede modificar la fecha de asignación que por defecto será la fecha actual.

	4	El sistema notifica al usuario que la asignación se ha realizado correctamente

Título:	Listar TFG asignados (CTG)	
Precondición:	El usuario se ha identificado previamente como miembro del CTG mediante un login específico para la cuenta del comité.	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el usuario CTG desee generar un listado en PDF con todos los TFG del sistema que estén asignados a un alumno.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la acción de listar los TFG asignados.
	2	El sistema ofrece la posibilidad de listar los TFG por fecha de asignación, pudiendo filtrar los resultados especificando una fecha inicial y/o otra fecha final. Los TFG que no estén asignados entre esas fechas no se incluirían en el listado. Por defecto, si no se indican fechas, se incluyen todos los TFG asignados.
	3	El usuario selecciona en un formulario las fechas de comienzo y/o fin del listado si lo requiere. También puede dejar alguna de ellas o las dos en blanco. Solicita generar el listado.
	4	El sistema genera un PDF con el listado solicitado por el usuario. En ese listado se incluye el ID del TFG, su título, su tutor y su departamento, el nombre de alumno que realiza el TFG, su titulación y el número de ECTS del TFG.

Título:	Renuncia a un TFG asignado (Alumno)	
Precondición:	El usuario se ha identificado previamente como alumno de la ETSIT mediante sus credenciales UVA. El alumno tiene un TFG ya asignado	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando un alumno solicite la renuncia a su TFG.	
	Paso	**Acción**

Secuencia normal	1	El usuario solicita al sistema que desea renunciar al TFG que tiene asignado.
	2	El sistema muestra un formulario indicando al usuario que explique el motivo de su renuncia
	3	El usuario introduce sus motivos en el formulario y lo envía
	4	El sistema envía un correo electrónico a la dirección del CTG y del tutor del TFG al que desea renunciar el alumno, avisándoles de la renuncia
	5	El sistema comunica al usuario que su solicitud de renuncia se ha guardado correctamente y que el CTG estudiará la misma.

Título:	Consultar renuncias pendientes (Tutor)	
Precondición:	El usuario se ha identificado previamente como profesor de la ETSIT mediante sus credenciales UVA.	
Descripción	El sistema deberá comportarse como se describe a continuación cuando un tutor accede al mismo y se le notifica que hay solicitudes de renuncia pendientes	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede a la aplicación mediante sus credenciales UVA
	2	El sistema notifica al usuario que uno de sus tutorandos ha solicitado renunciar a su TFG
	3	El usuario selecciona la función “consultar renuncias” en el sistema
	4	El sistema muestra una tabla con todas las renuncias pendientes que hayan realizado los tutorandos del usuario
	5	El usuario selecciona una de las renuncias mostradas
	6	El sistema muestra un formulario en el que se solicita al usuario que escriba algunas observaciones sobre la renuncia del alumno y si está conforme con dicha renuncia o no.
	7	El usuario completa dichos campos
	8	El sistema almacena la información emitida por el tutor en la base de datos y envía un correo a la dirección de email del CTG para notificar que el tutor ya ha revisado la renuncia

Título:	Tramitar renuncia de TFG (CTG)	
Precondición:	El usuario se ha identificado previamente como miembro del CTG mediante un login específico para la cuenta del comité.	
Descripción	El sistema deberá comportarse como se describe a continuación cuando el CTG solicite tramitar las renuncias presentadas por los alumnos.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita mostrar todas las solicitudes de renuncia pendientes.
	2	El sistema muestra una tabla que contiene todas las renuncias a trabajos que estén pendientes de tramitar en el sistema. Esta tabla muestra el nombre del alumno, el nombre del tutor, el id del TFG, el motivo por el cual el alumno solicita la renuncia, las observaciones del tutor y si el tutor está conforme con la renuncia o no.
	3	El usuario elige la renuncia que desea tramitar y aprieta el botón de “aprobar renuncia”
	4	El sistema informa de que la renuncia se ha tramitado satisfactoriamente. Además, el sistema actualiza el alumno en la base de datos para eliminar su relación con el TFG. Además, el sistema actualiza el estado del TFG a caducado.

Título:	Solicitar defensa de TFG (Alumno)	
Precondición:	El usuario se ha identificado previamente como alumno de la ETSIT mediante sus credenciales UVA. El alumno tiene un TFG ya asignado	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el un alumno solicite la defensa de su TFG.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema que desea defender su TFG
	2	El sistema devuelve un texto con información sobre la defensa y un formulario con los siguientes campos: Palabras clave, resumen, fecha de la defensa, lugar de defensa
	3	El usuario introduce los datos en el formulario y lo envía
	4	El sistema genera un documento PDF con el impreso de solicitud de defensa del TFG.

	5	El usuario imprime dicho documento PDF, completa los datos del mismo y lo firma.
	6	El usuario entrega el documento en la Secretaría Administrativa del centro como tarde cinco días antes de la fecha elegida para la defensa. Además, también adjuntará una copia de la memoria del trabajo en formato PDF incluida en algún soporte físico.

Título:	Tramitar defensa de TFG (Secretaría ETSIT)	
Precondición:	El usuario se ha identificado previamente con un login especial para la Secretaría de la ETSIT.	
Descripción	Cuando un alumno presenta la documentación en la Secretaría de la ETSIT solicitando la defensa de su TFG, el sistema deberá comportarse como se describe en el siguiente caso de uso.	
Secuencia normal	**Paso**	**Acción**
	1	La Secretaría de la ETSIT accede al sistema y elige la opción tramitar defensa.
	2	El sistema muestra una pantalla solicitando al usuario que introduzca el DNI del alumno que ha solicitado defender su TFG
	3	El usuario introduce el DNI del alumno.
	4	El sistema muestra una pantalla con datos del alumno, del TFG que desea defender, así como información sobre la defensa
	5	El usuario comprueba que los datos son correctos y confirma la tramitación.
	6	El sistema actualiza el estado del TFG que pasa a Pendiente de Defensa.

Título:	Consultar TFG adjudicados, tribunales y fechas de defensa	
Descripción	El sistema deberá comportarse como se describe a continuación cuando un usuario solicite el listado de TFG adjudicados, tribunales y fechas de defensa.	
	Paso	**Acción**

Secuencia normal	1	El usuario accede a la plataforma y solicita consultar el listado de TFG adjudicados, tribunales y fechas de defensa.
	2	El sistema genera un documento PDF formado por una tabla que contendrá todos los TFG con estado asignado o pendiente de defensa. En dicha tabla se mostrarán los siguientes campos: número de TFG, Nombre del alumno al que se ha adjudicado, título, tutor/es, Departamento, Miembros del tribunal, fecha de defensa y lugar de la defensa. .

Título:	Consultar TFG ofertados	
Descripción	El sistema deberá comportarse como se describe a continuación cuando un usuario solicite el listado de TFG ofertados por la escuela.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede a la plataforma y solicita consultar el listado de TFG ofertados.
	2	El sistema genera un documento PDF formado por una tabla que contendrá todos los TFG con estado aprobado. En dicha tabla se mostrarán los siguientes campos: número de TFG, título, tutor/es, departamento, titulaciones para las que se ofrece y un breve resumen

A continuación, se adjuntan unos diagramas de los principales procesos y trámites que implementa la aplicación telemática.

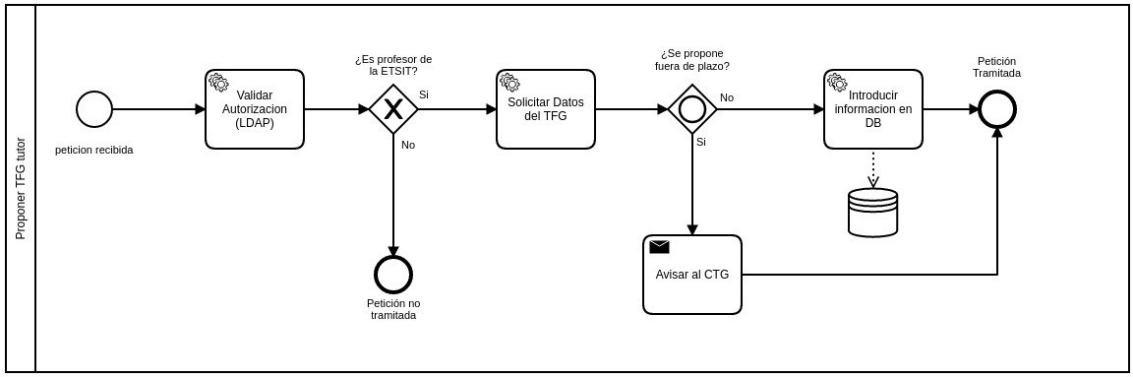

DIAGRAMA 1 – PROPONER TFG (TUTOR)

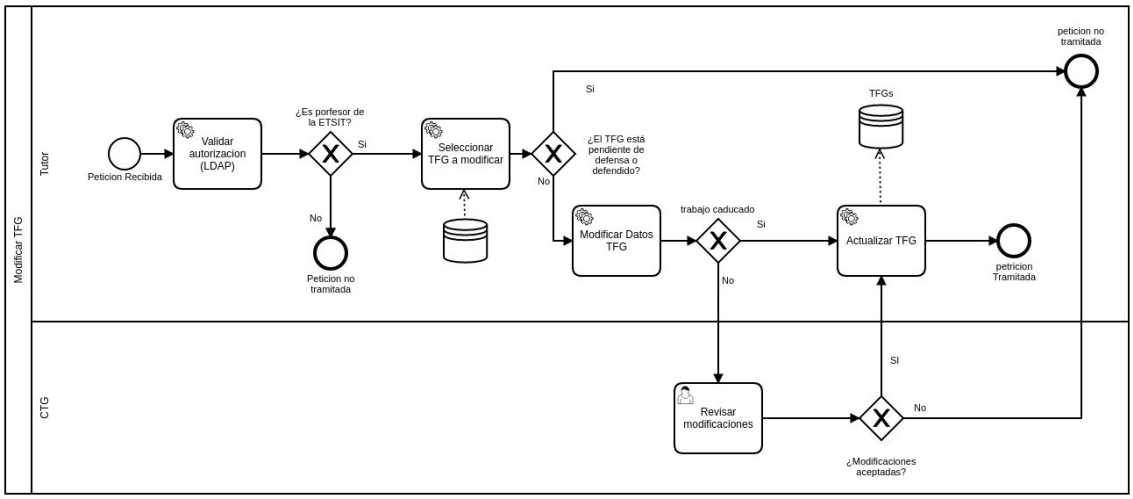

DIAGRAMA 2 – MODIFICAR TFG

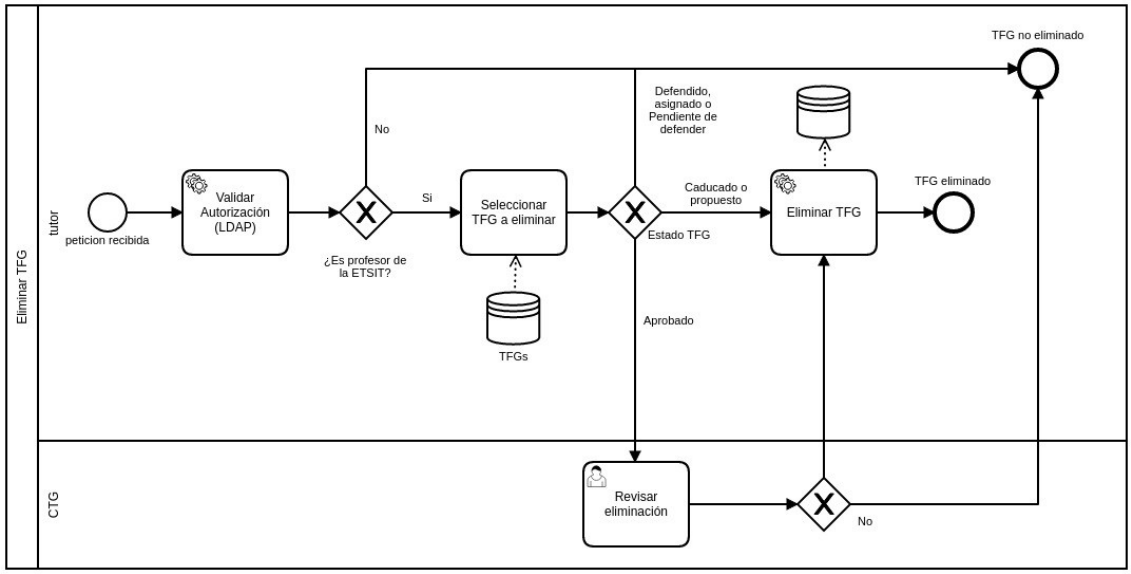

DIAGRAMA 3 – ELIMINAR TFG

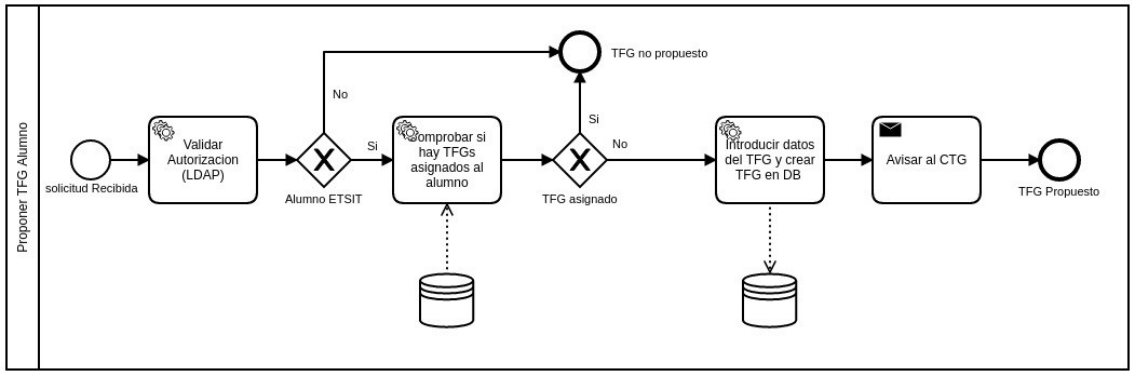

DIAGRAMA 4 – PROPONER TFG (ALUMNO)

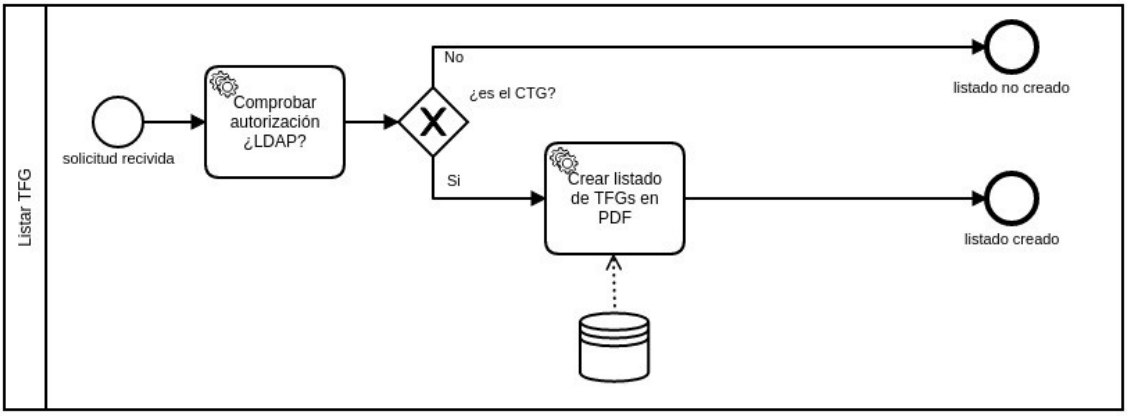

DIAGRAMA 5 – LISTAR TFG PROPUESTOS

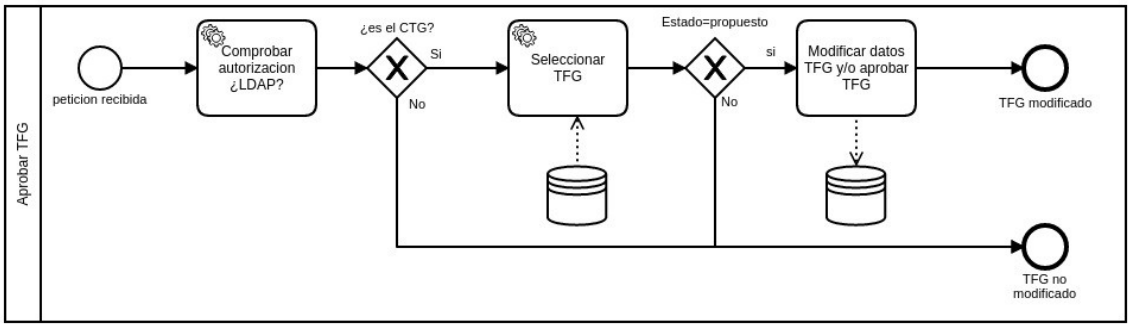

DIAGRAMA 6 – APROBAR TFG

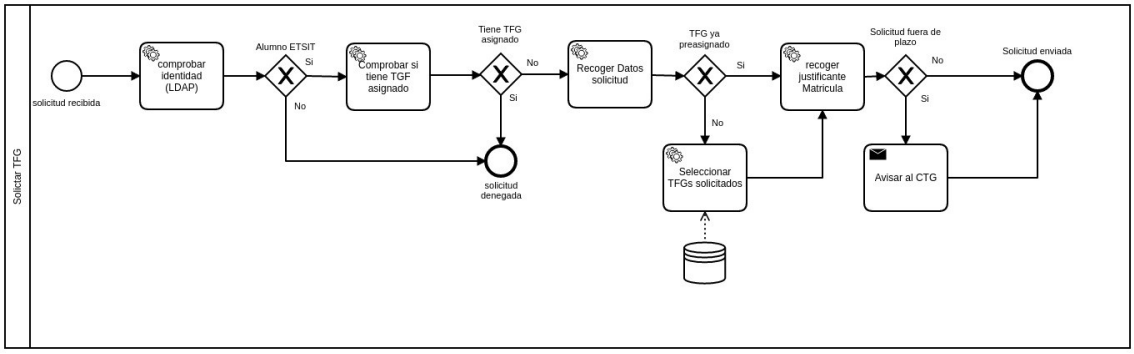

DIAGRAMA 7 – SOLICITAR TFG

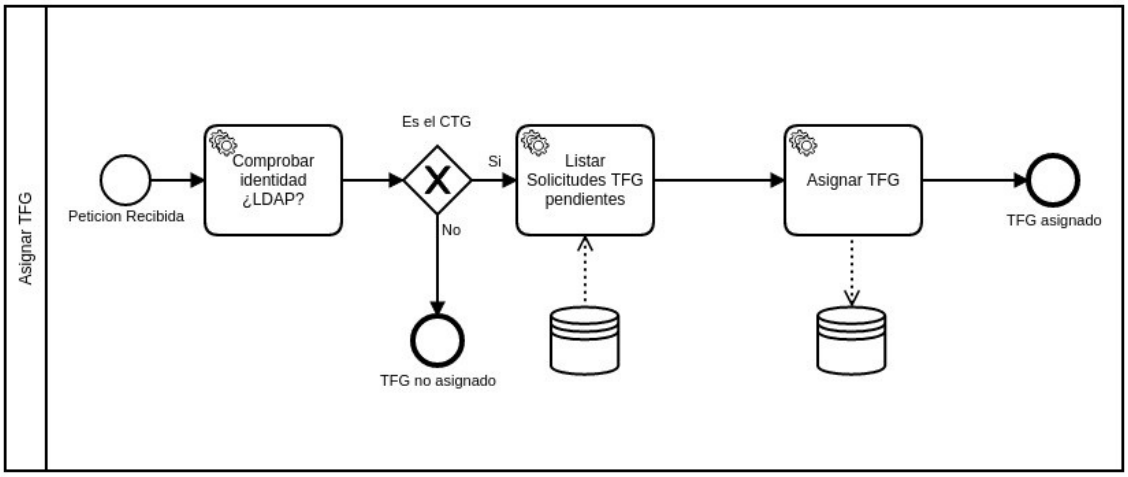

DIAGRAMA 8 – ASIGNAR TFG

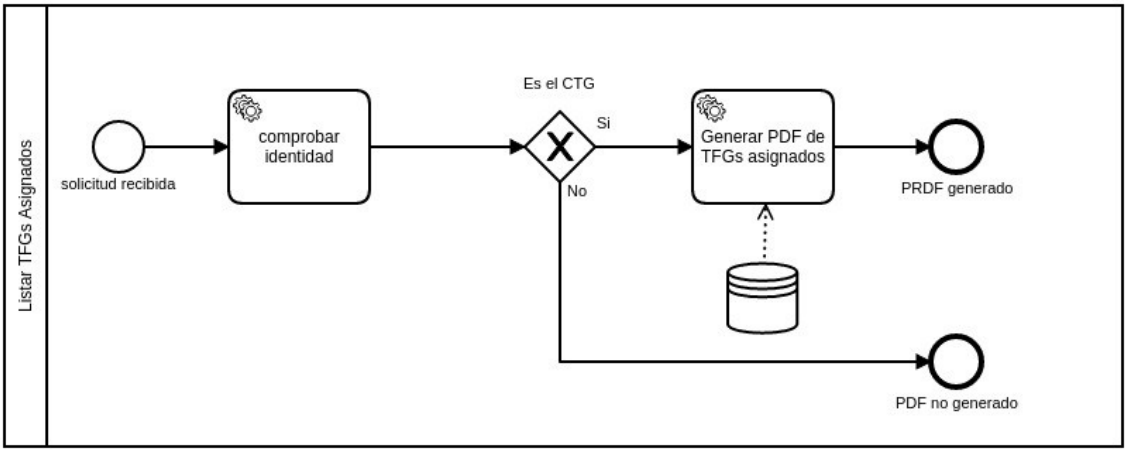

DIAGRAMA 9 – LISTAR TFG ASIGNADOS

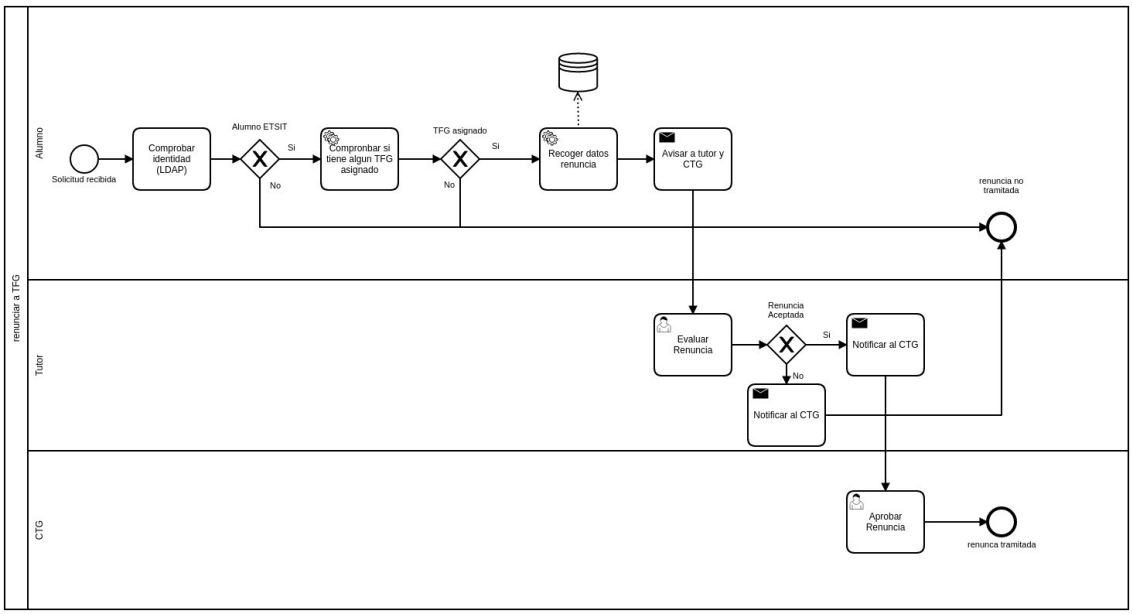

DIAGRAMA 10 – RENUNCIAR A TFG

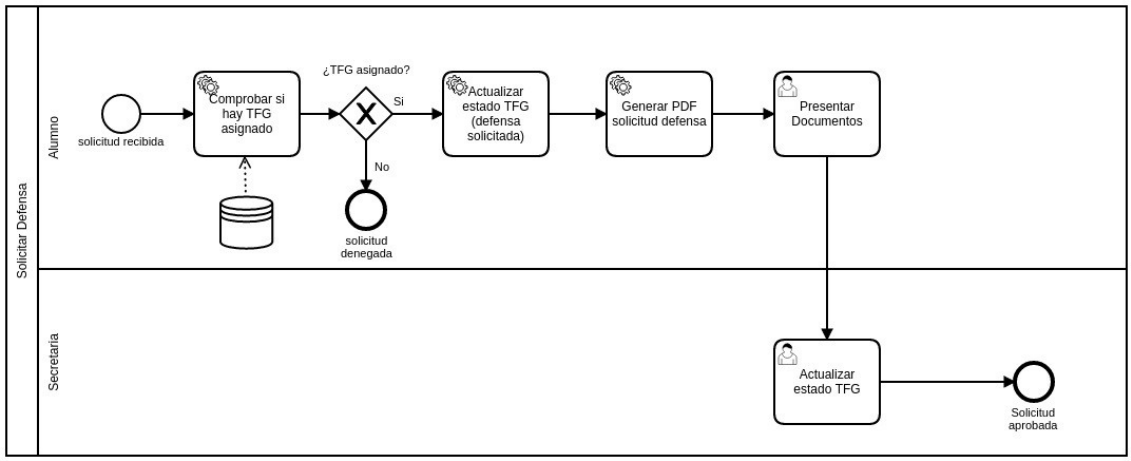

DIAGRAMA 11 – SOLICITAR DEFENSA DE TFG

3.2 Los servicios WEB de la ETSIT

Para el desarrollo de la aplicación, se van a reaprovechar ciertos servidores, servicios e infraestructura de red que la ETSIT ya utiliza para otros proyectos, de tal forma que no sea necesario comprar nuevas máquinas que supondrían un coste añadido tanto en adquisición como en el mantenimiento de las mismas. El aprovechamiento de estos componentes en ocasiones va a suponer una restricción o ligadura con determinada tecnología. A continuación, se resumen los diferentes servicios que ya están operativos y que se pretenden emplear para el funcionamiento de la aplicación de gestión de los TFG.

Servidor Web: La aplicación web tiene que alojarse en un servidor. La ETSIT cuenta con un servidor web de producción operativo, que entre otros cometidos tiene el de alojar la página web corporativa de la escuela (www.tel.uva.es). Es importante destacar que el programa que se ejecuta en esta máquina para prestar el servicio web es Apache Tomcat. Este software es un contenedor web de páginas dinámicas que soporta Servlets y JSP. Esto ha condicionado el empleo del lenguaje de programación Java en el lado del servidor para el desarrollo de la aplicación y la utilización de la tecnología de servlets.

Base de datos: Para la persistencia de los datos relacionados con la aplicación, también se ha optado por reaprovechar el servidor de bases de datos que ya tiene la escuela en funcionamiento y que da soporte a diferentes proyectos. En concreto se trata de un sistema de base de datos relacional PostgreSQL.

LDAP: También se va a hacer uso del servidor de directorio LDAP de la ETSIT, que en última instancia depende del servidor LDAP de la UVA. El empleo del servicio de directorios será necesario para las tareas de autenticación de los usuarios y para la gestión de roles y permisos. De esta manera el acceso a la aplicación por parte de los usuarios será mucho más sencillo y cómodo ya que se realizará mediante las credenciales de sus cuentas UVA.

4. Desarrollo e implementación del sistema

4.1 Metodología empleada

El trabajo realizado ha partido de un borrador con las especificaciones funcionales que debía cumplir la aplicación a desarrollar. Este borrador ha sido redactado por miembros del Comité de Titulaciones de Grado y, por tanto, las personas al cargo de toda la gestión de los TFG en la ETSIT.

Este borrador agrupa las diferentes funcionalidades de la aplicación en lo que se podría llamar procedimientos o casos de uso. Por caso de uso se entiende un determinado trámite o acción con significado completo que

solicita un usuario sobre el sistema. Así se puede tener un caso de uso para proponer un TFG que será realizado por un tutor, o el caso de uso de solicitar la renuncia de un TFG, que será realizado por un alumno. Se pueden consultar todos los casos de uso que componen la aplicación en el capítulo 3.1. Teniendo esta agrupación de funcionalidades en mente, la aplicación inicial, tal y como se ha planteado está compuesta por 12 grandes casos de uso.

La metodología empleada ha consistido en una primera fase de estudio de las especificaciones funcionales y análisis de las mismas, donde el borrador anteriormente mencionado ha jugado un papel fundamental. Fruto de este estudio se crea un primer diseño de la base de datos y de la estructura de clases de la aplicación.

Sobre este primer diseño se comienzan a implementar los casos de uso. El desarrollo de cada uno de estos procedimientos se ha separado en iteraciones independientes. En cada una de estas iteraciones se han estudiado los requisitos que tenía que cumplir el procedimiento, se ha diseñado una solución para satisfacer dichos requisitos, se ha implementado el código y por último se han probado las funcionalidades desarrolladas para confirmar que tenían el comportamiento adecuado.

Cada uno de estos procedimientos se ha desarrollado de forma independiente y secuencial, de uno en uno. No obstante, a medida que se avanzaba en la implementación de los mismos, ha habido que acometer cambios, normalmente en el modelo de representación de los datos, así como en el diseño de la base de datos para poder dar cabida a las nuevas funcionalidades que se iban desarrollando. Por el contrario, la arquitectura general de la aplicación, así como su estructura de clases, ha permanecido mayoritariamente inalterada desde las primeras iteraciones.

4.2 Configuración del entorno de desarrollo

Para el desarrollo e implementación de la aplicación se han utilizado cuatro herramientas principales. Por un lado, NetBeans en su versión 7.4, el contenedor de aplicaciones web Apache Tomcat 8 con soporte para tecnologías Servlet y JSP; un servidor de base de datos PostgreSQL, en su versión 9.3 y por último la herramienta PhpPgAdmin para la administración de la base de datos de una forma gráfica y sencilla. Todas ellas corriendo sobre un sistema operativo Ubuntu, basado en GNU/Linux. La elección de la tecnología del servidor web y del servidor de base de datos, así como sus respectivas versiones, ha sido tomada pensando en tener un entorno de pruebas lo más similar que fuera posible al entorno de producción donde va a tener que ejecutarse finalmente la aplicación.

NetBeans es un entorno de desarrollo integrado (IDE) enfocado principalmente en la gestación de aplicaciones basadas en la plataforma Java; ya sean aplicaciones de escritorio, móviles o basadas en web, puesto que tiene soporte para tecnologías HTML, CSS y JS. También se puede ampliar su funcionalidad añadiéndole módulos con los que se da soporte a

otros lenguajes y tecnologías como C/C++ o PHP. Es software libre, gratuito y de código abierto por lo que no hay que pagar licencias para utilizarlo.

Para instalar Netbeans sólo hay que acceder a su portal web (https://netbeans.org/downloads/index.html) y descargar el archivo ejecutable de la versión que más convenga (ver Figura 6). En este caso se ha optado por la versión Java EE, por tener soporte para algunas tecnologías necesarias para el desarrollo de la aplicación como HTML5 y JS; además de incluir soporte para contenedores de aplicaciones web basados en Servlets como Tomcat o GlashFish Server.

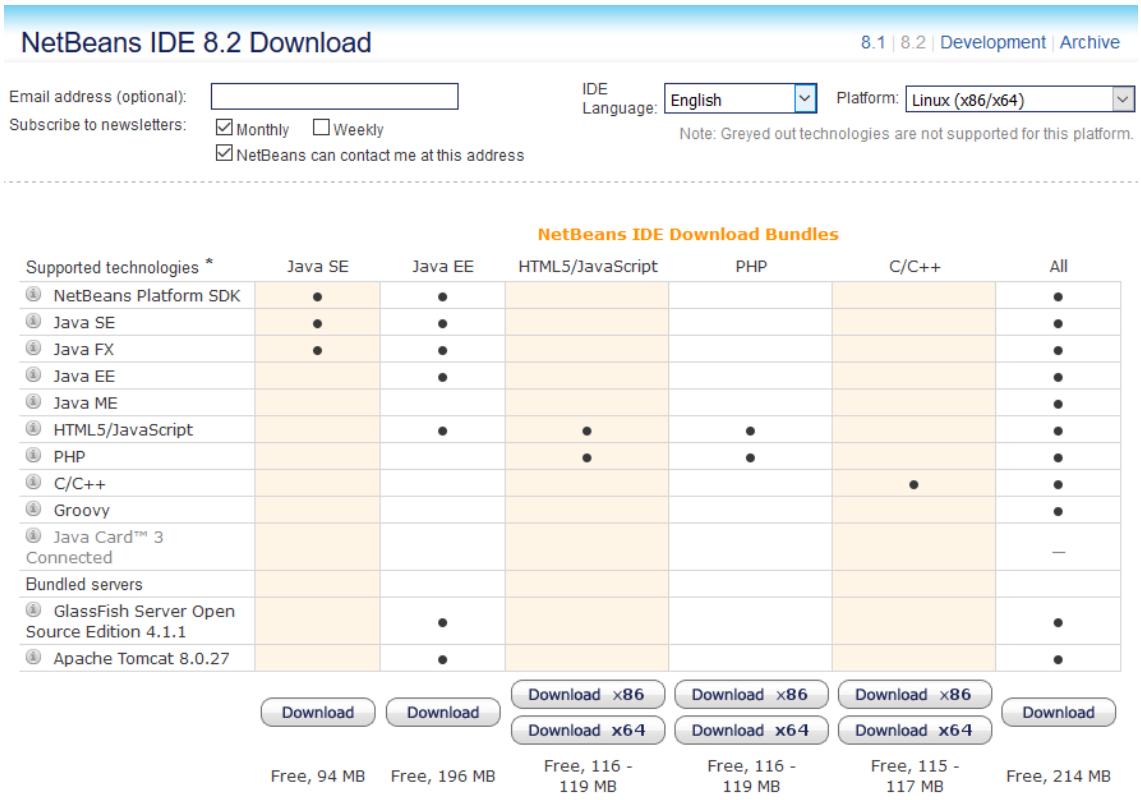

Figura 6 – Diferentes versiones de NetBeans y las tecnologías que soportan

Un aspecto importante de Netbeans es que se puede integrar con Maven. Según la página oficial del proyecto [13], Maven es una herramienta pensada para simplificar el proceso de compilación y generación de ficheros ejecutables a partir del código fuente de cualquier proyecto basado en java. Maven establece un proceso estándar para compilar los proyectos y define en qué consisten los mismos, de una forma clara y específica.

En la práctica Maven funciona mediante un fichero XML en la raíz del proyecto llamado POM (Project Object Model). En ese fichero se describen diversos aspectos importantes relacionados con el proyecto como el nombre del mismo, su versión, información referente a su compilación y algo muy importante a la hora de desarrollar el propio proyecto: las dependencias del

mismo. Mediante el fichero POM se pueden incluir diferentes librerías de software que se han empleado para el desarrollo del proyecto. En este fichero es donde se indican todas las dependencias que necesita la aplicación para poder ser compilada, empezando por el propio framework Spring. El IDE se encarga de descargar todos los ficheros y librerías descritas en el POM para tenerlas disponibles de cara a la compilación de la aplicación en un fichero WAR.

A continuación, se describen brevemente todas las dependencias empleadas y para qué se han empleado

- **Dependencias del framework Spring**: las dependencias spring-context, spring-webmvc y spring-context-support son propias del framework Spring y se necesitan para poder desarrollar nuestra aplicación bajo esta plataforma. Estas dependencias incorporan todas las clases núcleo que requiere el framework para funcionar.

- **Dependencias de seguridad**: Spring-security-web, spring-security-ldap y spring-security-config son dependencias más específicas del framework Spring que se necesitan para proveer de seguridad a la aplicación. Desde la autenticación mediante el servidor LDAP hasta la gestión de roles de acceso y permisos (qué usuarios pueden acceder a qué paginas).

- **Dependencias para el acceso a base de datos:** La librería JDBC (Java Data Base Conectivity) nos provee de una serie de clases necesarias para poder trabajar con consultas y peticiones a la base de datos mientras que org.postgresql es el driver propio de la implementación concreta de la base de datos que estamos empleando, y lo que nos permite realizar conexiones a dicha base de datos.

- **Dependencias Java estándar**: javaee-web-api es un api genérico para el desarrollo de aplicaciones WEB incluida en la plataforma J2EE.

- **Dependencias para la subida de archivos al servidor**: commons-fileupload y commons-io son dependencias de Apache que se emplean para posibilitar la subida de archivos al servidor Tomcat.

- **Dependencias para la generación de vistas en PDF:** Mediante la librería itextpdf se pueden crear vistas de la aplicación basadas en documentos PDF generados dinámicamente.

- **JSP Standar Tag Library**: Esta dependencia permite mucha flexibilidad a la hora de generar vistas dinámicas basadas en JSP. Mediante el empleo de ciertas etiquetas resulta mucho más sencillo el manejo de colecciones y la definición de formularios en los ficheros JSP.

- **Lombok**: Esta librería sirve para simplificar la creación de las clases Java pertenecientes al modelo. Mediante una sencilla anotación al comienzo de la clase esta herramienta se encarga de generar automáti-

camente todos los métodos get y set de la misma, ahorrándonos dicha tarea.

- **Dependencias para el servicio de mensajería:** javax.mail y javax.activation son dos librerías necesarias para que la aplicación pueda conectarse a un servidor de correo electrónico y mandar diferentes correos electrónicos de aviso a algunos usuarios de la misma.

La instalación del servidor de bases de datos PostgreSQL, así como un cliente por línea de comandos para acceder a la misma es muy sencilla y se puede realizar fácilmente mediante la terminal del sistema escribiendo:

$ sudo apt-get install postgresql postgresql-contrib

Una vez realizada la instalación es interesante administrar la base de datos creando un usuario específico o rol, como se denomina en PostgreSQL, para facilitar la conexión de la aplicación a la misma. Para hacer esto hay numerosos manuales en Internet que explican detalladamente este procedimiento.

Para la instalación del servidor web Apache Tomcat, lo primero es descargar la aplicación de la página oficial del proyecto (https://tomcat.apache.org/) como un fichero comprimido. Después procedemos a descomprimir ese fichero en el directorio donde más convenga. El servidor web Tomcat es una aplicación escrita en Java por lo que no requiere una instalación en el sistema operativo como tal. Una vez que la descomprimimos, se generan una serie de scripts .sh para Linux o .bat para Windows, que se emplean para arrancar la aplicación, detenerla o realizar configuraciones de algún tipo.

En este caso no es necesario manipular ni lanzar ninguno de esos scripts. Tampoco va a ser necesario colocar los ficheros compilados del proyecto en el directorio raíz del servidor para su ejecución y prueba. Todas esas tareas van a ser realizadas automáticamente por NetBeans, que se encargará de toda la gestión de Tomcat. Para conseguir esto tan solo hay que ir al apartado de servidores dentro de NetBeans y seleccionar la opción de añadir un nuevo servidor Tomcat, especificando dónde se encuentra la carpeta que contiene los ficheros y scripts del servidor, dentro del árbol de directorios del sistema operativo.

PhpPgAdmin es una aplicación web cuyo objetivo es la administración y monitorización de bases de datos basadas en PostgreSQL. Es una herramienta muy similar al conocido PhpMyAdmin con la salvedad de que este último está orientado a la administración de bases de datos mySQL. PhpPgAdmin es una aplicación escrita en PHP y se accede a ella a través de un navegador web. Por lo tanto, para instalarla, basta con descargar la aplicación de la página oficial de la herramienta (http://phppgadmin.sourceforge.net/) como un fichero ZIP y descomprimirlo en el directorio raíz de un servidor web que soporte PHP. En este caso se ha optado por el conocido servidor Apache2 para este cometido. Una vez realizados estos pasos, se podrá acceder a la aplicación insertando su URL

en un navegador web. A continuación, tan solo hay que introducir el usuario y clave que se tenga configurado en la base de datos para que PhpPgAdmin pueda conectarse a la misma y mostrar gráficamente su contenido.

4.3 Diseño de la aplicación

Como se ha mencionado en capítulos previos, la aplicación desarrollada, que hemos denominado AGT (Adjudicación y Gestión de Trabajos de fin de grado), sigue un patrón de diseño Modelo Vista Controlador. La aplicación se compone de 3 bloques principales con funciones bien diferenciadas. En este capítulo se va a describir el diseño que ha seguido cada uno de esos bloques.

4.4.1 Módulo del controlador

El controlador que va a emplear la aplicación para su funcionamiento ya viene parcialmente implementado en el propio framework Spring mediante la clase DispacherServlet. Según [13] el DispacherServlet se define como:

A central dispatcher for HTTP request handlers/controllers, e.g. for web UI controllers or HTTP-based remote service exporters. Dispatches to registered handlers for processing a web request, providing convenient mapping and exception handling facilities.

Esta clase implementa el servlet que se va a ejecutar en el contenedor de aplicaciones Java (Apache Tomcat en este caso) y que va a recibir en primera instancia todas las peticiones HTTP cuya URL pertenezca a la aplicación AGT. En función de la URL que se reciba, se direcciona la petición a un determinado controlador de la aplicación u otro. En este caso, la aplicación consta de un único controlador principal, por lo que todas las peticiones se mapearán a este.

No hay que implementar la clase DispacherServlet puesto que ya viene creada, pero sí hay que configurarla correctamente para que funcione acorde a las necesidades de la aplicación. En este aspecto Spring ofrece flexibilidad a la hora de la configuración [14] ya que se puede realizar mediante ficheros XML (método tradicional) o programáticamente mediante código Java utilizando una serie de clases determinadas (Soportado desde la versión 3.1 de Spring).

En este caso se ha optado por la configuración programática. Usar ficheros XML tiene la ventaja de que se separa claramente la parte de configuración de la aplicación de lo que es el propio código de la misma. Por el contrario, la configuración programática suele ser más breve y concisa.

```
public class MyWebApplicationInitializer implements
WebApplicationInitializer {

    @Override
    public void onStartup(ServletContext servletContext) {
```

```
        AnnotationConfigWebApplicationContext appContext = new
AnnotationConfigWebApplicationContext();
        appContext.setConfigLocation("config.*");

        servletContext.addListener(new
ContextLoaderListener(appContext));

        ServletRegistration.Dynamic registration
                = servletContext.addServlet("springmvc", new
DispatcherServlet(appContext));
        registration.setLoadOnStartup(1);
        registration.addMapping("/");

        }
}
```

La configuración se ha de realizar creando una clase que implementa la interfaz WebApplicationInitializer. Dentro de esta clase hay que definir un método llamado onStartup con la anotación @Override. Se crea un contexto basado en anotaciones y después se le indica cuál es la clase o el paquete completo donde están las clases de configuración de la aplicación. En la aplicación desarrollada todas las clases de configuración están contenidas dentro del paquete config.

A continuación se crea el Dispacherservlet con el contexto de la aplicación que hemos creado anteriormente. Y después se registra en el contenedor de aplicaciones.

SetLoadOnStartup sirve para especificar la prioridad con la que se van a inicializar los servlets en el caso de que la aplicación desarrollada tenga más de uno, teniendo más prioridad los que tengan el número entero más bajo. Como la aplicación sólo cuenta con un DispacherServlet no es importante el parámetro que se ponga.

Por último, addMapping se utiliza para especificar las URL que van a ser procesadas por el servlet. En este caso, como se ha mencionado antes, la aplicación se compone de un único servlet por lo que este tendrá que atender todas las peticiones que lleguen (“/”).

El DispacherServlet redirige todas las peticiones HTTP que se reciban a la clase Controlador de la aplicación. Esta clase es el centro neurálgico de la app. Cuenta con una serie de métodos anotados con la etiqueta @RequestMapping. Con esta anotación se indica la URL a la que va a responder cada método. De esta forma, el DispacherServlet llama a unos u otros métodos de la clase Controlador en función de la URL que se solicite por el cliente.

Cada uno de los métodos de la clase controlador va a atender a una URL concreta de la aplicación. Los métodos a veces reciben variables de entrada que, entre otras cosas, pueden encapsular datos que el usuario envíe mediante formularios o información sobre variables de sesión de dicho usuario. Dentro de cada método se encuentra la funcionalidad que tiene que

ejecutarse para dar respuesta a la petición del usuario. Normalmente esto se consigue mediante llamadas a la lógica de negocio de la aplicación (implementada en las clases estáticas Gestores). Estas clases devuelven, si procede, información sobre el modelo que haya de representarse a través de las vistas al usuario. Por último, Los métodos devuelven un objeto ModelAndView que contiene el nombre de la vista que se va a emplear para responder a la petición del usuario y diferentes datos y variables del modelo que pudieran hacer falta para completar dicha vista.

La clase controlador contiene un método por cada una de las diferentes “pantallas” que componen la aplicación. La navegación a través de estas pantallas se realiza cuando el usuario clica sobre diferentes botones o interactúa con algún formulario. Estos recursos redirigen el navegador a una nueva URL de la aplicación y esta carga la nueva pantalla.

4.4.2 Diseño del modelo del sistema

El bloque del modelo del sistema se puede separar en dos componentes. Por un lado, está el modelo de objetos o modelo de dominio. Este modelo se compone de beans java que representan los diferentes tipos de datos que maneja la aplicación para su funcionamiento. La aplicación AGT se compone de los siguientes Beans: Alumno, Renuncia, SolicitudDefensa, SolicitudTfg, Tfg, Tutor y Alumno. Cada uno de estos beans, sus atributos y métodos, pueden verse en la Figura 7. Los diferentes métodos SET y GET para interactuar con dichos atributos se han eliminado del esquema para mayor claridad del mismo.

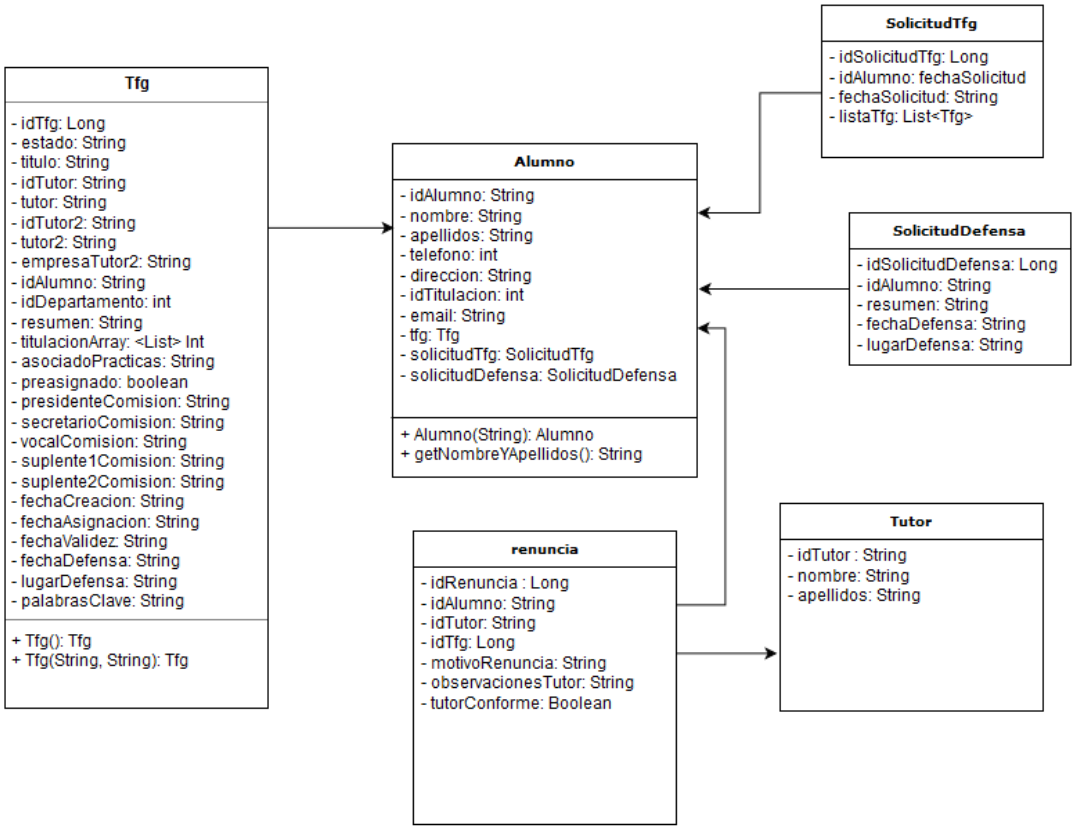

Figura 7 – Esquema de las clases que componen el modelo de dominio de la aplicación

Sobre estos beans y sus atributos cabe hacer algunas aclaraciones. El parámetro ID que identifica a los objetos que se instancien de estas clases, es un identificador serial auto incrementado, creado por la base de datos donde se almacenan dichos objetos. Excepcionalmente, en las clases Tutor y Alumno, se emplea el uid asociado con el sistema LDAP de la UVA compuesto por el DNI del usuario precedido de la letra e.

Por otra parte, está el modelo lógico de la aplicación que contiene todas las reglas de negocio del sistema, es decir, modela las reglas del mundo real por las que se rige el sistema de gestión de los TFG en la escuela. Este componente se encarga de la manipulación de los datos del modelo de dominio, creando nuevos objetos y modificando o eliminando los ya existentes, como respuesta a las peticiones del controlador. El modelo lógico también se encarga de interactuar con la base de datos del sistema ya sea para almacenar nueva información, como para extraer los datos guardados en ella y devolvérselos al controlador. El modelo lógico está compuesto por clases estáticas no instanciables que contienen los diferentes métodos que implementan la lógica de la aplicación. Estas clases pertenecen al paquete Gestores y para una mayor facilidad en la lectura y mantenimiento del código se han separado en función del tipo de objetos que manipulan.

De esta forma hay un GestorTfg que se encarga de manejar las peticiones relacionadas con los TFG; un GestorRenuncias que se encarga de manipular las solicitudes de renuncia de los alumnos y todo lo relacionado con ellas; un GestorSolicitudesTfg que se encarga de cursar las solicitudes de TFG que hacen los alumnos, un GestorSolicitudesDefensa que se encarga de las peticiones de defensa que solicitan los alumnos, y por último un GestorUsuarios que se encarga de recuperar el contexto de los usuarios cuando estos acceden a la aplicación.

4.4.3 Diseño de la base de datos

La aplicación desarrollada implementa una serie de clases que componen el modelo de dominio y que pueden verse en la Figura 7. Estas clases modelan y representan diferentes entidades del mundo real involucradas en la gestión de los trabajos de fin de grado.

Estas clases se instancian formando colecciones de objetos. De esta forma se tiene una colección de todos los TFG que se han creado en la plataforma, una colección de renuncias de TFG pendientes de tramitar, una colección de solicitudes de defensa que han cursado los alumnos, etc. Estas colecciones de objetos han de almacenarse de forma persistente en una base de datos que garantice la integridad de la información a largo plazo.

Para este cometido se ha recurrido a una base de datos relacional basada en PostgreSQL por ser el gestor de bases de datos relacionales que ya está implementado y funcionando en la red de la ETSIT. A la hora de diseñar la base de datos se ha pretendido ordenar la información de la manera más eficaz posible, evitando la redundancia de la misma, protegiendo su integridad y evitando problemas en la actualización de dicha información en las diferentes tablas. Para cumplir con estos objetivos se ha diseñado una base de datos normalizada en tercera forma, siguiendo las pautas habituales del diseño de base de datos.

Los objetos que han de almacenarse están compuestos tanto por atributos que son de tipo primitivo (int, string, float, long, boolean…) y que están soportados por el gestor de base de datos; como por atributos que son de tipo objeto, los que a su vez tendrán otros atributos encapsulados dentro de ellos.

Como se puede ver en la Figura 8, por cada clase que compone el modelo de dominio hay una tabla o relación en la base de datos que la representa, y a su vez, hay otras tablas a mayores. Las relaciones entre clases pueden ser de 1:1. Por ejemplo: un alumno tiene un solo TFG asignado o una única solicitud de defensa cursada asociada a él. En estos casos estas relaciones se representan en la base de datos mediante la inclusión de una clave foránea en una de las tablas, que es a su vez una clave primaria en la tabla con la que se quiere reaccionar. La tabla de alumnos tiene un id_alumno, que se almacena como clave foránea en la tabla tfgs.

Cuando las relaciones son de varios a uno o incluso de varios a varios, entonces se hace uso de tablas de unión. Estas tablas contienen

únicamente dos columnas: las claves foráneas de las dos relaciones que se pretenden unir. Esta técnica puede apreciarse en la relación tfgs_solicitudes y tfg_titulaciones. Una solicitud de TFG puede tener varios trabajos asociados a ella, y un TFG puede estar relacionado con varias solicitudes (cuando dos alumnos diferentes solicitan el mismo trabajo). Algo similar ocurre en la tabla tfg_titulaciones donde un trabajo puede ofertarse para diferentes titulaciones.

De esta forma se ha conseguido un diseño robusto y fácilmente escalable en caso de que las necesidades de la aplicación cambien en el futuro y sea necesario modificar esta base de datos.

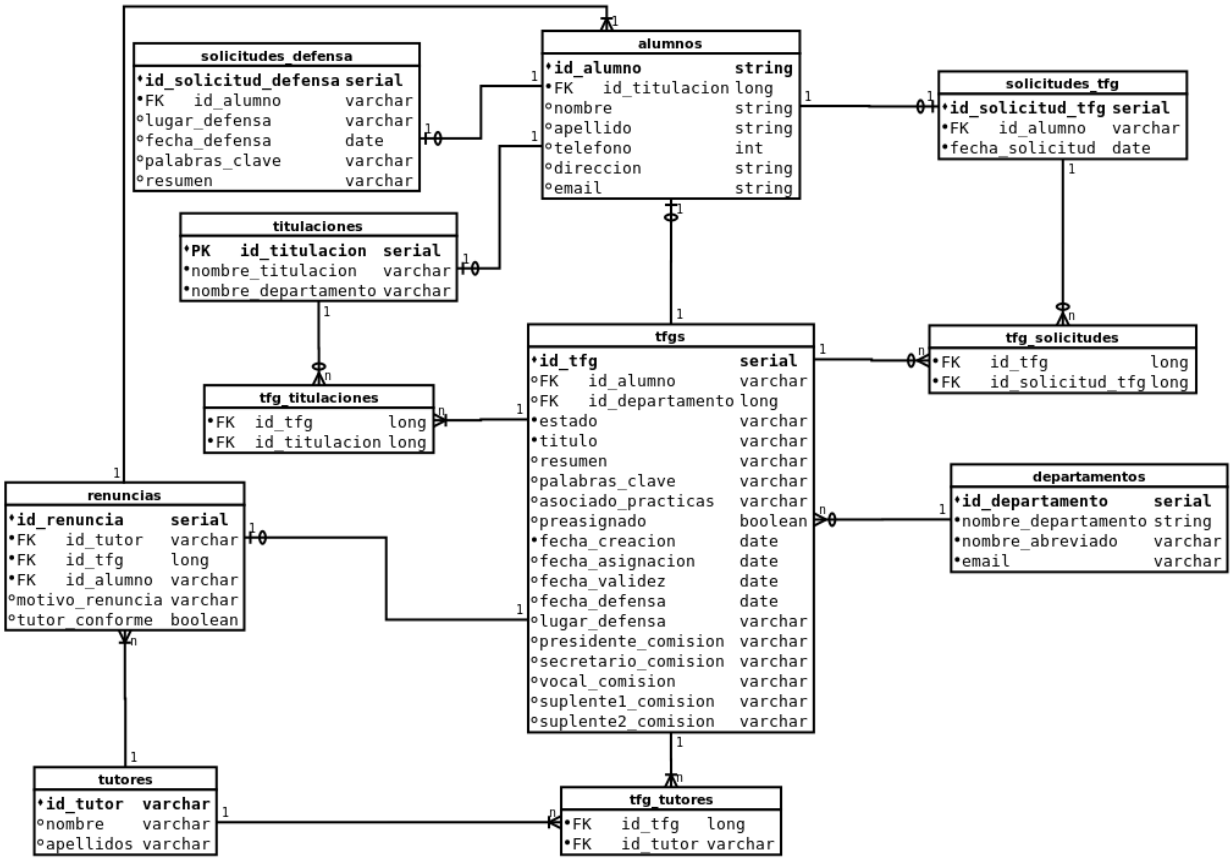

Figura 8 - Esquema entidad-Relación de la base de datos del sistema

4.4.4 Diseño de las vistas

En cuanto al diseño de las vistas de la aplicación, se ha recurrido a diferentes tecnologías con el objetivo de proporcionar una interfaz gráfica clara e intuitiva. El propósito ha sido conseguir una experiencia de uso lo más sencilla y agradable posible a los usuarios de la plataforma, de tal forma que sean capaces de interactuar con la misma sin necesidad de ningún tipo de guía, instrucciones o conocimiento previo.

La principal tecnología utilizada para desarrollar las vistas ha sido JSP. Según [15] Java Server Pages es una especificación desarrollada por Sun Microsystems, que forma parte de la plataforma J2EE y permite generar páginas web dinámicas en el lado del servidor. En la práctica es una tecnología similar a PHP. Los documentos JSP son páginas web con contenido HTML estático a las que se les puede incorporar pequeños bloques de código Java, que serán procesados por el servidor para generar el contenido dinámico antes de enviar el documento final al cliente.

El software desarrollado se compone de diferentes ficheros JSP que contienen todo el código relacionado con la interfaz gráfica. Hay un fichero JSP por cada una de las diferentes pantallas que conforman la aplicación.

Figura 9 – La información se presenta al usuario mediante el empleo de tablas

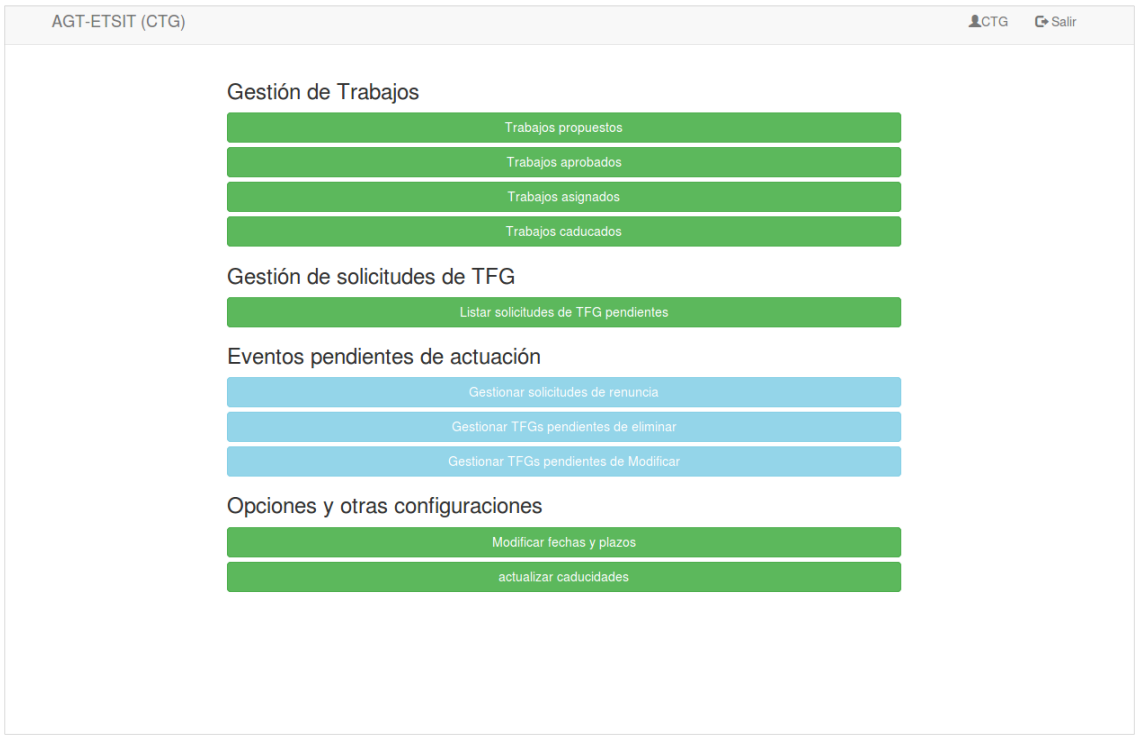

Figura 10 – Diferentes botones para la navegación a través de los distintos trámites que ofrece la aplicación

Los elementos que componen la interfaz gráfica de usuario son los propios de la tecnología web. Se ha recurrido al uso de tablas (ver Figura 9) para mostrar colecciones de elementos de una forma clara y ordenada. Para la navegación por los diferentes menús de la aplicación se ha hecho uso de botoneras de gran tamaño, en las cuales se ha empleado un código de colores intuitivo (Figura 10) con el fin de remarcar el riesgo de ciertas acciones (color rojo), así como para conducir a los usuarios por el itinerario habitual de los diferentes trámites dentro de la misma (colores verdes). También se ha recurrido al uso de formularios HTML para el envío de datos por parte de los usuarios.

Aunque JSP es la tecnología que vertebra la interfaz gráfica de la aplicación, sobre ella se ha hecho uso de otras herramientas con el fin de facilitar el desarrollo de la misma y también para conseguir una mejor experiencia de uso.

Sobre JSP se ha empleado Java Server Pages Standard Tag Library [16]. JSTL encapsula, mediante simples etiquetas, diversas funcionalidades comunes a muchas aplicaciones web. Esta tecnología se ha empleado principalmente a la hora de trabajar con colecciones de elementos en las vistas de la aplicación ya que tiene funcionalidades propias para el manejo de listas y para realizar iteraciones en las mismas. También permite trabajar con sentencias condicionales, lo cual ha servido para mostrar, o no, diversas partes del contenido de la interfaz gráfica en función de algún parámetro o condición previa.

A mayores de JSTL también se ha empleado Spring JSP Tag Library [17]. Esta librería de Tags amplía la funcionalidad de JSTL y añade funcionalidades muy interesantes como el manejo de una forma sencilla de los formularios en la aplicación, ya que trata los diferentes campos del mismo como atributos de un objeto java que se envía al navegador del cliente para que los rellene y devuelva dicho objeto cumplimentado. También permite procesar diferentes mensajes de error cuando el usuario completa incorrectamente los datos del formulario.

Los documentos JSP que forman la aplicación contienen código HTML ya sea generado estática o dinámicamente. Para maquetar los diferentes elementos que componen estos documentos, para dotarles de color, de estilo y de estructura, se ha empleado el framework Bootstrap que consigue estos objetivos mediante el empleo de hojas de estilo y java script.

Al utilizar Bootstrap se ha conseguido implementar un diseño visual sencillo, limpio e intuitivo en la plataforma. Otra ventaja de Bootstrap es que este framework ya se utiliza en un gran número de webs, por lo que su empleo implica que los usuarios ya suelen estar familiarizados con la apariencia de los diferentes elementos, por haber entrado en contacto con ellos previamente en otras webs.

En ciertos procedimientos o trámites, el objetivo de la aplicación es generan diferentes documentos en PDF para su visualización, impresión o descarga por parte del usuario. Esta funcionalidad también forma parte del bloque de las vistas, solo que se apoya en una tecnología diferente llamada iText. Esta

herramienta de código libre permite crear y manipular de forma dinámica ficheros PDF en Java. Para el diseño de esta funcionalidad se han tomado como referencia los documentos PDF que se emplean en el actual sistema de gestión y se ha intentado replicar exactamente su diseño, formas y contenido.

4.5 Estructura de directorios del proyecto software

A la hora de desarrollar el proyecto de software se ha seguido el patrón de convención sobre configuración (CoC). Esto implica que los diferentes ficheros que componen el proyecto, están organizados de una forma estandarizada atendiendo a su tipo y función, dentro del árbol de directorios del mismo. En la Figura 11 se puede apreciar dicho árbol de directorios, así como las principales carpetas que lo componen. A continuación, se describe someramente esta organización:

Figura 11 – árbol de directorios del proyecto software

En el directorio raíz del se encuentra el fichero pom.xml que es utilizado por Maven para la gestión de dependencias del proyecto, así como para establecer los parámetros de compilación y empaquetamiento.

El directorio /src/main/java contiene todos los ficheros de código java de la aplicación. Estos ficheros se organizan en diferentes paquetes java que se describen a continuación.

- **Config:** contiene clases relacionadas con la configuración inicial de la aplicación. Configuración sobre el despliegue del servlet en el contenedor web, así como diferentes configuraciones sobre seguridad.
- **Controladores:** Este paquete contiene una única clase que implementa la funcionalidad completa del módulo controlador de la aplicación.
- **Gestores:** Contiene clases estáticas que implementan la lógica de negocio de la aplicación. Es decir, aglutina el modelo de negocio del sistema
- **Modelo:** Este paquete contiene las diferentes clases que componen el modelo de dominio de la aplicación. Es decir, las clases que modelan las entidades del mundo real que debe emular el sistema: alumno, tfg, SolicitudTfg, Renuncia…
- **Views:** Contiene clases que se encargan de crear las vistas de la aplicación que consisten en la generación dinámica de ficheros en PDF. Para el resto de vistas del sistema se usan ficheros JSP y no clases Java.
- **Auxiliares:** Este paquete contiene diferentes clases que se encargan de tareas concretas como las conexiones a la base de datos o las conexiones a un servidor de correo electrónico para la generación y envió de email a diferentes usuarios del sistema.

El directorio /src/main/webapp/WEB-INF contiene ficheros principalmente relacionados con las vistas de la aplicación. Dentro del mismo se encuentran las siguientes sub-carpetas:

- **css:** Contiene el fichero de hoja de estilos de Bootstrap.
- **views:** contiene los ficheros JSP que componen las diferentes páginas de la aplicación
- **js:** Contiene ficheros de código Java Script, tanto los asociados al framework Bootstrap como otros ficheros de código desarrollados en el presente trabajo para dotar a la aplicación de algunas funcionalidades.
- **Classes:** contiene un fichero de propiedades que redirige los nombres lógicos de ciertas vistas hacia clases java, para que sean

estas las que se encarguen de confeccionar dichas vistas. Esto ocurre con la generación dinámica de vistas PDF, que no utiliza ficheros JSP para su construcción.

El directorio target almacena diferentes ficheros que se generan automáticamente como resultado de la compilación del proyecto. Entre otros ficheros, ahí se almacena el .WAR que empaqueta toda la aplicación y que ha de desplegarse en el servidor web o contenedor de servlets para ejecutarla.

5. Conclusiones y líneas futuras

5.1 Conclusiones

Como resultado del presente trabajo, se ha analizado el problema existente en la Escuela Técnica Superior de Ingenieros de Telecomunicación de Valladolid en relación con la gestión y administración de los trabajos de fin de grado de los alumnos. Dicho problema consiste en el importante esfuerzo humano que supone manejar de forma manual todo ese volumen de información ya que es una tarea costosa y que se repite todos los años.

Se ha realizado una propuesta de mejora de este sistema mediante el diseño y posterior creación de una aplicación telemática que automatice y facilite, en la medida de lo posible, la realización de los trámites relacionados con los TFG a los diferentes colectivos implicados. También se ha conseguido implementar una plataforma digital centralizada que facilite la tarea de administración y gestión de dichos trabajos por parte del Comité de Titulaciones de Grado.

5.2 Líneas futuras

El trabajo recogido en este TFG deja las puertas abiertas a futuras mejoras en lo ya realizado, así como nuevas líneas de desarrollo. A continuación, se mencionan algunas de las más interesantes:

Soporte a trabajos de fin de Master:

El presente trabajo recoge el problema de la gestión de los TFG en la ETSIT y cómo se ha mejorado esta mediante el desarrollo de una aplicación telemática para facilitar y automatizar la mayoría de trámites. Casi todos Los problemas derivados de la gestión de los TFG, así como las soluciones que se han desarrollado para dichos problemas, son extrapolables a la gestión de los TFM que tienen que realizar los alumnos que cursan algún máster. En todo caso, hay menos alumnos cursando másteres en la ETSIT que grados, por lo que todo el problema de la gestión de los TFM es de menor magnitud. Llegados a este punto se proponen dos aproximaciones posibles para resolver el problema.

Por un lado, se puede implementar una aplicación web nueva y diferenciada de AGT-ETSIT que en esencia tendrá unas características y funcionalidades muy similares a la original, por lo que el grado de reutilización del código de AGT sería muy elevado. Simplemente habría que realizar modificaciones en el modelo y en las vistas, para adaptarlos a las particularidades de los TFM. La segunda aproximación consiste en integrar toda la gestión de los TFM dentro de la aplicación AGT ya existente, añadiendo a esta las nuevas funcionalidades que sean necesarias.

Ambas posibilidades tienen ventajas e inconvenientes. Centralizar TFG y TFM en una misma aplicación hace que sólo haya que mantener un producto, lo cual evitará tener que realizar por duplicado futuros cambios y actualizaciones. Por otra parte, implementar dos aplicaciones independientes nos proporcionaría una mayor flexibilidad al poder desarrollar un código más individualizado para cada uno de los dos sistemas. Además, cada una de las aplicaciones por separado sería más sencilla de mantener al incluir menos lógica y funcionalidades. Por último, al tratarse de dos sistemas independientes sería una solución con mayor fiabilidad ya que, aunque uno de ellos dejara de funcionar por algún motivo, el otro podría continuar prestando su servicio.

Mejoras en la implementación del código:

A la hora de enfrentarme al diseño de la aplicación, y su posterior implementación, he partido de los conocimientos que he adquirido a lo largo del grado que he cursado en la ETSIT, pero el desarrollo del presente trabajo ha sido un aprendizaje constante y muy intenso en sí mimo. Con esto se quiere decir que el código desarrollado, aunque cumple con las funcionalidades requeridas no es perfecto, ya que está escrito por un principiante en estas lides. Si la plataforma AGT cumple con las expectativas y resulta útil y funcional para la comunidad ETSIT, sería recomendable hacer una revisión de este código y modificarlo implementando buenas prácticas de programación que yo pudiese desconocer en el momento del desarrollo, de cara a conseguir un producto más robusto y con una mayor sencillez en el mantenimiento.

Mejoras en la eficiencia computacional:

Relacionado con el punto anterior también se podría mejorar la eficiencia computacional de la aplicación. Es decir, optimizar el código para que consuma menos recursos, para que haga lo mismo, pero de forma más eficiente. Con esto se ahorra carga de trabajo en las máquinas donde se vaya a ejecutar la aplicación, lo cual implica a su vez un menor consumo energético.

Implementación de la funcionalidad auto-completar en los formularios de la aplicación:

En los formularios de la aplicación, algunos campos a veces hacen referencia a ciertos usuarios del propio sistema. Por ejemplo, a la hora de pre-asignar un TFG a un estudiante, el usuario tutor de la aplicación ha de indicar quién es el estudiante al que se pre-asigna dicho TFG. También

ocurre algo similar cuando un tutor crea una propuesta de TFG y quiere asociarla con un segundo tutor UVA aparte de consigo mismo.

Actualmente la aplicación está implementada para que se tenga que introducir el uid UVA de dichas personas (eXXXXXXXXX) y así identificarlas de forma unívoca en el sistema. El problema de esta solución es que los usuarios de la aplicación que tengan que introducir este tipo de campos necesitan saber el dni de la persona en cuestión que quieran escoger.

Para simplificar este proceso, la solución más recomendable es que se añada una funcionalidad auto-completar a los campos de los formularios que sirvan para identificar personas dentro del sistema. De esta forma, el usuario de la aplicación que quiera escoger a una persona para alguno de estos campos, no necesitará conocer el dni de dicha persona, sino que comenzará a introducir el nombre de esta en el campo y la aplicación cotejará en tiempo real la cadena de caracteres que vaya escribiendo el usuario, con la base de datos de LDAP para devolver las personas cuyo nombre y apellidos coincidan con la cadena introducida.

Implementación de alguna herramienta para visualizar y gestionar los justificantes de matrícula desde la propia plataforma:

Cuando los alumnos crean una solicitud de TFG, uno de los requisitos que se les pide es que adjunten un fichero (PDF, JPEG, PNG…) en el que se demuestre que se han matriculado de la asignatura Trabajo de Fin de Grado. Actualmente la aplicación está diseñada para que el servidor almacene localmente dichos ficheros, renombrado cada uno de ellos con el uid del alumno (eXXXXXXXXX) que lo subió a la plataforma.

Si se quiere acceder a dichos ficheros es necesario realizar una conexión FTP al sistema de ficheros del servidor web, para poder descargarlos en otra máquina y proceder a su comprobación. Una propuesta muy interesante es que dentro de la aplicación se desarrolle algún tipo de funcionalidad para poder gestionar dichos ficheros a través de la propia aplicación web. De esta forma el usuario CTG podría comprobar el contenido de dichos justificantes y administrarlos convenientemente dentro del contexto de la propia aplicación. Esto supondría una mayor facilidad en la gestión y un aumento en la productividad del CTG.

Añadir nuevas funcionalidades deseables:

La aplicación desarrollada como fruto del presente TFG no es un producto cerrado ni acabado completamente. Se han implementado todas las funcionalidades imprescindibles que se recogían en el borrador de especificaciones, pero seguramente puedan incluirse de cara al futuro funcionalidades añadidas que ayuden aún más a los usuarios del sistema.

Mejora de la interfaz gráfica:

Uno de los puntos en los que más margen de mejora puede haber es en la parte de las vistas. Este apartado es quizá en el que menos tiempo y trabajo he invertido. A la hora de diseñarlo busqué sencillez y claridad en la presentación de los contenidos. Pero una vez que los usuarios reales

interactúen con el sistema, seguramente tengan sugerencias que aportar sobre la forma que tienen de interactuar con el mismo y cómo esta podría ser mejor y más eficaz, presentando la información de una forma más clara, y que todos los contenidos puedan visualizarse de forma intuitiva sin tener que realizar apenas esfuerzo.

Mejoras en el código de acceso a base de datos:

A la hora de implementar la aplicación se ha desarrollado una API para la comunicación de esta con la base de datos. Este código trabaja a bajo nivel, adaptando la información de las clases java del modelo de la aplicación a consultas SQL que pueda interpretar el driver JDBC, que en última instancia es el que realiza la conexión a la base de datos. Esta forma de hacer las cosas a bajo nivel no es la más eficaz de cara al mantenimiento y a futuros cambios en la aplicación. Si se realiza algún cambio en las clases de la aplicación, seguramente haya que modificar muchas líneas del código de comunicación con la base de datos.

La solución que se propone es que la sustitución de esta parte de la aplicación en futuras versiones, por alguna API ORM (Object Relation Manager) como Hibernate, por citar uno. La ventaja de utilizar este tipo de herramientas es que son transparentes para el programador. Por un lado, se entrega la colección de objetos que se desea almacenar en la base de datos relacional, y la API trabaja como una caja negra almacenando dicha información en la base de datos sin que el programador tenga conocimiento de lo que está pasando a bajo nivel. De esta forma si cambia la forma o el tipo de los objetos, la API actúa en consecuencia cambiando la forma de traducir estos al modelo relacional sin que haya que intervenir para nada en el proceso.

6. Referencias

[1] ETSIT Valladolid, *Guía docente del trabajo de fin de grado* [online], 2017. Disponible en https://www.tel.uva.es/descargar.htm?id=24251 [Consulta: 23 julio 2018]

[2] M. L. Liu, *Computación distribuida. Fundamentos y aplicaciones*. San Luís: Addison-Wesley, 2004.

[3] Estrategia digital, *Diferencias entre una aplicación web y un sitio web* [online]. 2016. Disponible en https://blog.ida.cl/estrategia-digital/diferencias-aplicacion-web-sitio-web/ [Consulta: 23 julio 2018]

[4] E. Gamma, R. Helm, J. Vlissides, R. Johnson, *Design Patterns: Elements of Reusable Object-Oriented Software.* Indianapolis: Addison-Wesley, 1995.

[5] T. Reenskaug, *THING-MODEL-VIEW-EDITOR, an Example from a planningsystem* [online]. 1979. Disponible en:

http://heim.ifi.uio.no/~trygver/1979/mvc-1/1979-05-MVC.pdf [Consulta: 23 julio 2018]

[6] M. Hall, *Core Servlets and JavaServer Pages (JSP)*. 2000. Prentice Hall.

[7] Oracle. Inc, *Java Servlet Technology Overview* [online]. Disponible en http://www.oracle.com/technetwork/java/javaee/servlet/index.html [Consulta: 23 julio 2018]

[8] The apache software foundation, *The Apache Struts web framework* [online]. Disponible en https://struts.apache.org/ [Consulta: 23 julio 2018]

[9] The apache velocity Project, *What is velocity* [online]. Disponible en http://velocity.apache.org/ [Consulta: 23 julio 2018]

[10] Spring doc, *Web on Servlet Stack* [online]. Disponible en https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html [Consulta: 23 julio 2018]

[11] P. Fingar, C. Robles, G. Bouchon, M. Mora, R. Campos, R. de Laurentiis, *El libro del BPM y la trasformación digital*. Madrid: Club-BPM, 2017.

[12] M. Mehta, *STRUTS 2 vs SPRINGMVC: Know the Difference & Choose the Best One Based On Your Requirements* [online]. *2015.* Disponible en https://dzone.com/articles/struts-2-vs-springmvc-know-the-difference-choose-t [Consulta: 23 julio 2018]

[13] Apache Maven Project, *What is Maven* [online]. Disponible en https://maven.apache.org/what-is-maven.html [Consulta: 23 julio 2018]

[14] Baeldung, *Web.xml vs Initializer whit Spring* [online]. 2018. Disponible en http://www.baeldung.com/spring-xml-vs-java-config [Consulta: 23 julio 2018]

[15] Oracle, *Overview of Java Server Pages (JSP)* [online]. Disponible en https://docs.oracle.com/cd/E13222_01/wls/docs81/jsp/intro.html [Consulta: 23 julio 2018]

[16] Oracle, *Java Server Pages Standard Tag Library* [online]. Disponible en http://www.oracle.com/technetwork/java/index-jsp-135995.html [Consulta: 23 julio 2018]

[17] Spring doc, *Spring JSP Tag library* [online]. Disponible en https://docs.spring.io/spring/docs/4.2.x/spring-framework-reference/html/spring-tld.html [Consulta: 23 julio 2018]