Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería de Software

Gestión de TFGs en la Escuela de Ingeniería Informática de Valladolid

---

Alumno: Borja Arnáez Pérez

Tutor: Yania Crespo González-Carvajal

Agradecimientos

Con este trabajo cierro una etapa que con sus luces y sombras ha sido maravillosa. Y si ha sido así es por las personas que me habéis acompañado hasta aquí.

A mis padres, por hacer de mi quien soy hoy y en día y estar siempre al pie del cañón, por vuestro trabajo diario y dedicación a mi y mis hermanas.

A mis hermanas, por estar conmigo cada día y porque no ha habido nada más divertido que haber crecido junto a vosotras.

A mi pareja, por haberme echado alguna que otra mano en este trabajo, por ayudarme a mejorar a diario, por todos los momentos juntos y enseñarme a amar.

A mis amigos, por las risas, charlas y buenos momentos.

A todos, gracias, os quiero.

Resumen

Este Trabajo de Fin de Grado es un proyecto que busca la renovación de algunos de los sistemas con los cuáles la Escuela de Ingeniería Informática gestiona los proyectos comprendidos en la asignatura Trabajo Fin de Grado (TFG). Este proyecto implementa una aplicación web dedicada a la gestión de TFGs, junto a una API como intermediario con la base de datos.

La aplicación facilita el flujo de trabajo relacionado con la adjudicación de TFGs, creación de tribunales y asignación de tribunal para la defensa de TFGs. Los principales usuarios son los profesores del centro, que proponen la adjudicación de TFGs bajo su tutorización, y el coordinador del comité de título que, una vez aprobado por el comité, aprueba la adjudicación, crea tribunales, y prepara las defensas asignando tribunal a los TFGs que se entregan. La aplicación permite al coordinador publicar las asignaciones, los tribunales y las defensas, que una vez públicas pueden ser consultadas por cualquier usuario sin necesidad de identificarse. El sistema desarrollado se comunica con el SSO 5.4.1 de la UVa para el inicio de los usuarios, siendo autorizados a iniciar sesión solamente los profesores de la Escuela.

En el desarrollo de este trabajo se han utilizado los *framework* React con Typescript y Springboot con Java, así como HTML5, CSS y bajo el marco de trabajo Scrum.

Abstract

This Final Degree Project (FDP) aims to update some of the systems that are used by our *Escuela de Ingeniería Informática* manages the projects that are developed during the *Trabajo Fin de Grado* subject. This project implements a web-app for the management of the FDPs along with an API in between this app and the database.

The application facilitates the management flow related with FDPs adjudication, courts creation and court assignment for the FDP defense. The main users of this app are the teaching staff, those who propose new adjudications of FDPs under their tutelage, and the title committee coodinator who acepts the adjudication, creates the courts and prepares the defenses once it has been revised by the committee. The app allows the coordinator to publish the adjudications, courts and defenses to be read by any user without the need for authenticate. The system makes use of the SSO of the University for the sign-in of the teachers of the School.

For the development of this project we have made use of the frameworks React with Typescript and Springboot with Java, together with HTML5 and CSS. The project has been organised under the SCRUM framework.

Índice general

Agradecimientos **III**

Resumen **V**

Abstract **VII**

Lista de figuras **XV**

Lista de tablas **XVII**

1. Introducción **1**

1.1. Contexto 1

1.2. Motivación 2

1.3. Estructura de la memoria 2

2. Requisitos y Planificación **5**

2.1. Proceso de desarrollo basado en Scrum 5

2.1.1. Roles 6

2.1.2. Sprints 6

2.1.3. Metodología de estimación 7

2.1.4. Eventos Scrum 8

2.2. Stakeholders 8

2.3. Análisis de riesgos 9
2.4. Planificación y calendarización 11
2.4.1. Épicas 12
2.4.2. Backlog inicial 12
2.4.3. Backlog final 12
2.4.4. Calendarización 13
2.5. Presupuesto 13

3. Análisis **17**
3.1. Modelo de proceso 17
3.2. Modelado de dominio 21
3.3. Casos de uso 21

4. Tecnologías utilizadas **33**
4.1. Tecnologías de desarrollo 33
4.1.1. React 33
4.1.2. Vite 34
4.1.3. Yarn 34
4.1.4. TypeScript 34
4.1.5. Java 34
4.1.6. Springboot 35
4.1.7. Postman 35
4.1.8. Git y GitLab 35
4.1.9. Jetbrains 35
4.1.10. Docker 35
4.1.11. NGINX 35
4.2. Tecnologías de comunicación 36

4.2.1. Teams 36
4.3. Tecnologías de gestión 36
4.3.1. GitLab Issue Tracker 36
4.4. Tecnologías de documentación 36
4.4.1. LaTeX y Overleaf 36
4.4.2. Visual Paradigm 36

5. Diseño **37**

5.1. Arquitectura Frontend 37
5.1.1. MVVM 38
5.1.2. Diferencias de MVVM con MVC 38
5.1.3. Estructura del proyecto 39
5.1.4. Bibliotecas 43
5.2. Arquitectura Backend 43
5.2.1. API REST 43
5.2.2. Arquitectura por capas 44
5.3. Arquitectura del sistema 45
5.4. Identificación y autenticación 45
5.4.1. Servicio CAS 45
5.4.2. Token JWT 48
5.4.3. Diseño del flujo de autenticación en la aplicación 48
5.5. HTTPS 49

6. Implementación y pruebas **51**

6.1. Implementación 51
6.1.1. Base de datos y consultas 51
6.1.2. Implementación de una pantalla en la aplicación web 53

6.1.3. Casos de uso en el servicio backend 53
6.2. Pruebas 54
6.2.1. Pruebas de Casos de uso 55
6.2.2. Pruebas de navegación de la aplicación web 55
6.2.3. Pruebas con usuarios finales 59

7. Seguimiento del proyecto **61**
7.1. Fase inicial 61
7.2. Sprint 1 61
7.3. Sprint 2 63
7.4. Sprint 3 63
7.5. Sprint 4 64
7.6. Sprint 5 64
7.7. Sprint 6 65
7.8. Sprint 7 66
7.9. Fase final 67
7.10. Resumen de la ejecución del proyecto 67
7.10.1. Calendarización 67
7.10.2. Gestión de Riesgos 68
7.10.3. Coste 69

8. Conclusiones **71**
8.1. Líneas de trabajo futuras 71

Bibliografía **73**

A. Manuales **75**
A.1. Manual de despliegue e instalación 75
A.2. Manual de mantenimiento 76

A.2.1. Certificado SSL 76
A.2.2. Instalación de entorno de desarrollo 76
A.3. Manual de usuario 77
A.3.1. Menú principal 77
A.3.2. Propuestas 77
A.3.3. Detalles de propuesta 78
A.3.4. Adjudicaciones 78
A.3.5. Tribunales 78
A.3.6. Defensas 81

B. Resumen de enlaces adicionales **85**

Lista de Figuras

3.1. Fragmento del modelo de proceso centrado en la creación de propuesta, asignación y adjudicación 18

3.2. Fragmento del modelo de proceso centrado en la adjudicación de una asignación 19

3.3. Fragmento del modelo de proceso centrado en la creación de tribunales y defensas 20

3.4. Modelo del dominio 22

3.5. Diagrama de Casos de Uso 23

3.6. Diagrama de estados de una propuesta o adjudicación 23

5.1. MVC vs MVVM 40

5.2. Árbol de directorios del frontend 41

5.3. Diagrama de Arquitectura del frontend 42

5.4. Arquitectura de capas de referencia 44

5.5. Arquitectura del backend. 46

5.6. Subdiagrama de controlador. 46

5.7. Subdiagrama de servicio. 47

5.8. Diagrama de despliegue 47

5.9. Flujo de identificación 49

6.1. Diagrama de Diseño de la Base de Datos 52

6.2. Detalle de la Arquitectura de un Caso de Uso en el backend 54

A.1. Menú principal 77

A.2. Listado de propuestas 78

A.3. Creación de propuesta 79

A.4. Detalles de propuestas 79

A.5. Listado de adjudicaciones 80

A.6. Detalles de adjudicación 80

A.7. Listado de tribunales 81

A.8. Creación tribunales 82

A.9. Listado de defensas 82

A.10.Creación de defensa 83

Lista de Tablas

2.1. Comparación de escalas para la estimación de historias de usuario 7
2.2. Resumen metodología Scrum definida 9
2.3. Descripción de los riesgos 10
2.4. Exposición de los riesgos 10
2.5. Medidas de mitigación y contingencia 11
2.6. Épicas del proyectos 12
2.7. Backlog inicial del proyecto 12
2.8. Backlog final del proyecto 13
2.9. Calendarización del proyecto 14
2.10. Presupuesto simulado 15

3.1. CU Identificación 24
3.2. CU Solicitud de adjudicación TFG 25
3.3. CU Adjudicación de TFG 26
3.4. CU Publicación de adjudicaciones 26
3.5. CU Modificación de adjudicación publicada 27
3.6. CU Renovación de adjudicación 28
3.7. CU Creación de defensa 29
3.8. CU Publicación de defensa 30
3.9. CU Creación de tribunal 30

3.10. CU Publicación de tribunales 31
3.11. CU Modificación de tribunal 31
3.12. CU Modificación de defensa 32

6.1. CP08. Creación de tribunales sin publicar 55
6.2. CP02. Creación de propuesta 56
6.3. CP03. Asignación de propuesta 56
6.4. CP04. Adjudicación de propuesta 56
6.5. CP05. Adjudicación de propuesta 56
6.6. CP06. Edición de adjudicación 57
6.7. CP07. Publicar una adjudicación 57
6.8. CP08. Creación de tribunales sin publicar 57
6.9. CP09. Creación de tribunales con publicación 57
6.10. CP10. Edición de los detalles de un tribunal 58
6.11. CP11. Borrado de un tribunal durante la creación 58
6.12. Pruebas de navegación 58

7.1. Sprint 0 62
7.2. Sprint 1 62
7.3. Sprint 2 62
7.4. Sprint 3 63
7.5. Sprint 4 64
7.6. Sprint 5 65
7.7. Sprint 6 65
7.8. Sprint 7 66
7.9. Fase final 66
7.10. Calendarización final del proyecto 68
7.11. Presupuesto real 69

Capítulo 1

Introducción

1.1. Contexto

El Trabajo de Fin de Grado (TFG) es una asignatura de docencia obligatoria en el marco del grado de Ingeniería Informática, perteneciente al segundo cuatrimestre del cuarto año de dicho grado y siendo el cierre al paso del estudiante por la universidad en el contexto de esta titulación.

Es por esto que todos los alumnos deberán pasar por los trámites de esta asignatura al menos una vez durante su etapa universitaria, si esta es finalizada. También todos los profesores tutores, con una frecuencia mucho mayor, realizan diferentes trámites en relación con esta asignatura tales como: proponer trabajos de fin grado, proponer al Comité de Título la aprobación de TFG y su adjudicación. El Comité de Título, representado fundamentalmente por el Coordinador de la titulación, realiza con mayor frecuencia gestiones como aprobacíón de TFGs y adjudicaciones, creación de tribunales y defensas, asignando un tribunal para la presentación de un TFG. De esta forma, una actualización en los procesos de gestión de esta asignatura es de gran valor para la comunidad de nuestra Escuela.

Para el desarrollo de este trabajo se ha dispuesto del Reglamento sobre la elaboración y evaluación del Trabajo Fin de Grado de la Universidad de Valladolid[19] y de la Normativa de Evaluación del Trabajo Fin de Grado de E.T.S de Ingeniería Informática[18]. El segundo extiende al primero y lo especializa a las características de nuestra Escuela.

También se ha trabajado con el Servicio de Tecnologías de la Información y las Comunicaciones de la Universidad (S.T.I.C., o STIC para simplificar), con los Técnicos de la Escuela y con el actuar Coordinador de la Titulación.

1.2. Motivación

El proyecto producto de este Trabajo Fin de Grado es fruto de la necesidad de unificar y refinar los procesos por los cuales se realiza la gestión de esta asignatura.

Actualmente estos procesos son realizados de forma manual por el Coordinador de Grado, requiriendo estos de una gran inversión de tiempo en tareas repetitivas. Por otra parte, los procesos informatizados actualmente se encuentran en plataformas antiguas que necesitan actualizarse. Estos son los motivos por los cuáles se oferta este trabajo.

Por parte del estudiante, la principal motivación para llevar a cabo este proyecto es buscar un final para los estudios universitarios de grado que suponga un reto, que refuerce todo lo aprendido en esta etapa y que permita demostrarme que puedo tomar cualquier desafío en el futuro.

1.3. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: Este capítulo sirve como entrada al documento, describe las motivaciones de este trabajo, los objetivos y las soluciones actuales.

Capítulo 2 Requisitos y planificación: Este capítulo explica la planificación de trabajo bajo el marco de trabajo Scrum, así como las modificaciones de este para adaptarlo a un equipo de trabajo de un solo estudiante. También se repasa la calendarización y planificación, el presupuesto y el análisis de riesgos del proyecto.

Capítulo 3 Análisis: Este capítulo recoge toda la información recopilada necesaria para llevar a cabo el proyecto, y que comprende el modelado de proceso y dominio, e identificación de casos de uso.

Capítulo 4 Tecnologías utilizadas: En este capítulo se repasan todas las herramientas utilizadas durante el desarrollo del proyecto y que comprenden a aquellas usadas para el desarrollo, la comunicación, la gestión y la documentación.

Capítulo 5 Diseño: Este capítulo describe todas las decisiones de diseño tomadas a lo largo del proyecto. Estas comprenden la arquitectura de la aplicación desarrolladas y los sistemas que permiten su comunicación, acompañados de los diagramas correspondientes.

Capítulo 6 Implementación y pruebas: Este capítulo recoge y detalla las implementaciones de los casos de uso en las aplicaciones, las diferencias entre el esquema de la base de datos con respecto a su diseño y las pruebas realizadas a la aplicación.

Capítulo 7 Seguimiento del proyecto: En este capítulo se explican las tareas desarrolladas a lo largo de cada sprint, los problemas encontrados en estos y el trabajo dedicado al proyecto. También contiene un resumen de la ejecución final del proyecto y sus diferencias frente a lo establecido en la planificación.

Capítulo 8 Conclusiones: Este capítulo contiene las conclusiones extraídas tras la finalización del trabajo de desarrollo y una descripción de trabajos futuros necesarios para la aplicación desarrollad y la integración de nuevas funcionalidades.

Anexo A Manuales: En este anexo se detallan los manuales necesarios para la instalación, despliegue y mantenimiento de la aplicación, y el manual de usuario.

Anexo B Resumen de enlaces adicionales: Este último anexo se indican los enlaces de repositorios que contienen el código desarrollado y el enlace a la aplicación desplegada para su uso.

Capítulo 2

Requisitos y Planificación

En esta sección se estudiarán los riesgos que pueden manifestarse a lo largo del proyecto, así como la creación de un plan para el desarrollo de este que tenga cuenta la realidad del estudiante y estos riesgos.

2.1. Proceso de desarrollo basado en Scrum

En el desarrollo del presente proyecto vamos a hacer uso de la combinación de dos modelos de proceso de desarrollo como son el proceso en cascada y el proceso ágil basado en el marco de trabajo Scrum [14].

El proceso en cascada se utilizará durante las primeras fases del proyecto que incluyen la elicitación de requisitos, el análisis del problema y el diseño de arquitecturas y aplicaciones. Esta metodología se caracteriza por tener un acercamiento progresivo a cada fase de desarrollo, es decir, una fase sucede a otra cuando la anterior termina hasta que el proyecto es finalizado. En nuestro caso será hasta que el diseño quede finalizado.

Por otro lado, una vez finalizado el diseño, se realizará un proceso de desarrollo ágil basado en Scrum. Con esto se llevará a cabo la fase de implementación y desarrollo de las funcionalidades que componen las aplicaciones del proyecto. Esto nos permitirá entregar valor en cada sprint lo que no sería posible utilizando un proceso en cascada durante todo el proyecto. De cara a poder aplicar Scrum a un equipo compuesto únicamente por una persona y su tutora, se deben hacer una serie de concesiones[21] y adaptaciones al modelo general propuesto por este marco de trabajo. Para ello necesitaremos redefinir la repartición de roles, sprints y eventos propuestos por Scrum.

2.1.1. Roles

Empezaremos repasando los roles que conforma un equipo que trabaja bajo Scrum y la repartición de estas responsabilidades entre los implicados en el proyecto. Queda anotar que idealmente un equipo Scrum debe estar compuesto por al menos 4 personas, un Product Owner, un Scrum Master y dos desarrolladores, roles que se repartirán entre alumno y tutora.

- Product Owner (PO): Hace las funciones de representante de los intereses de los stakeholders y entiende las necesidades de estos. Es responsabilidad de este rol asegurarse de que se está entregando el mayor valor posible en cada sprint. Será la tutora la que tomará este rol en el desarrollo de este proyecto.
- Equipo de Desarrollo (DT): Comprende a todas las personas que trabajan en los incrementos de valor propuesto en cada sprint ya sean ingenieros, diseñadores, escritores, investigadores etc. En el contexto del proyecto será el alumno el único componente de este equipo y desarrollará todo el trabajo relacionado con la compleción de las tareas.
- Scrum Master (SM): Actúa como rol intermediador entre el PO y el DT, y participa en tareas con ambos roles. Por un lado trabaja con el PO en la definición del backlog, en el particionado del trabajo en partes alcanzables y en la comunicación del valor del trabajo realizado por el equipo; por otro lado su responsabilidad con el DT se centra en la organización, gestión de bloqueos y dirección de los esfuerzos en tareas que entreguen valor.
  Dadas estas responsabilidades, es necesario que sea el alumno quien tome también este rol pues no cagar este rol en la tutora supondría una delegación de responsabilidades por parte del alumno que no contemplamos.

Como se ha visto en la asignación de roles, el alumno toma el papel de roles que no deberían estar concentrados en un mismo individuo en un proceso basado en Scrum. De esta forma, y con el fin de establecer una vía para que estos roles se separen lo máximo posible, el alumno se compromete a no desarrollar en el mismo espacio de tiempo tareas pertenecientes a estos dos roles. También se deberá trabajar y prestar atención a centrarse en los intereses de cada uno de estos roles a fin de extraer el mayor beneficio de ambos roles a pesar de tratarse de una única persona.

2.1.2. Sprints

De cara a la definición de un sprint que nos permita entregar valor sin extenderse demasiado en el tiempo, perdiendo de esa forma las ventajas de metodologías ágiles, tendremos en cuenta las horas de dedicación a este proyecto por parte del Equipo de Desarrollo y el tiempo necesario para entregar valor al fin de cada sprint.

Comenzaremos con el tiempo de desarrollo que se dedicará a este proyecto semanalmente. Cabe destacar que el alumno compatibilizará el desarrollo de este proyecto con trabajo externo en jornada de mañana de 6 horas. Por este motivo, el tiempo de de dedicación de

Lineal	Fibonacci
Mayor facilidad de uso	Complejo hasta adquirir experiencia
Uso más extendido	Menor popularidad y uso
Mayor volatilidad	Mayor precisión en la estimación
Esconde las tareas complejas	Mejor indicativo de tareas mal definidas

Tabla 2.1: Comparación de escalas para la estimación de historias de usuario

Lunes a Viernes se pretende que sea de una media de 2 horas de trabajo diario. Los sábados se marca el objetivo de invertir de 6 a 8 horas, dejando los domingos como días de descanso. De esta forma tenemos un tiempo de trabajo semanal de 17 horas de media.

Debemos, también, definir un tiempo de sprints suficientemente largo como para ser capaces de entregar valor en cada uno de estos sin alargarlo demasiado en el tiempo, idealmente 2 o 3 semanas. También se ha acordado con la tutora basándose en su experiencia que un sprint en un Trabajo de Fin de Grado debería contener entre 30 y 40 horas. Si consideramos que semanalmente se dedicarán 17 horas, resulta ideal considerar como duración de sprint las 2 semanas.

2.1.3. Metodología de estimación

Podemos encontrar diferentes escalas de estimación del esfuerzo que lleva desarrollar las Historias de Usuario como pueden ser la estimación lineal, de Fibonacci o en potencias de 2, entre otras. De cara a simplificar esta decisión vamos a considerar las dos más extendidas por su uso que son la lineal y la de Fibonacci. A continuación se repasan las ventajas y debilidades de cada una de estas alternativas, para escoger aquella que se alinee mejor con nuestra visión y proyecto.

- Lineal: Las estimaciones realizadas en esta escala se define en incrementos lineales unitarios. Su principal ventaja es la facilidad de uso que plantea y la popularidad de esta. Por contrapartida, y basándonos en la experiencia recogida por herramientas como PivotalTracker[20], esta escala presenta una mayor volatilidad por punto de historia y que nos indica una peor estimación por parte de los equipos que usan esta escala.
- Fibonacci: Esta escala establece incrementos según la serie de Fibonacci. Su uso está soportado principalmente en la idea de que el aumento de la complejidad no es lineal y la realización de, por ejemplo, 3 tareas de 1 punto toman conjuntamente menos tiempo que 1 sola tarea de 3 puntos. Por otro lado, esta escala ofrece un indicativo más obvio de aquellas tareas que deberían ser partidas en otras más simples e impone la necesidad de una evaluación más crítica de las tareas de cara a su valoración en puntos. Como principal desventaja esta escala presenta una mayor complejidad de uso.

En la Tabla 2.1 se muesra un resumen de la comparativa entre la escala lineal y la de Fibonacci.

Presentadas las fortalezas y debilidades de ambas escalas, hemos decidido implementar la escala Fibonacci principalmente por coincidir en que la complejidad de las tareas no es lineal y una menor volatilidad en sus estimaciones.

2.1.4. Eventos Scrum

El marco de trabajo Scrum define eventos [15] como son la planificación de sprint, *dailys* (reuniones diarias), revisiones de sprint y retrospectiva del sprint. La organización de estos eventos está definida para proyectos de tamaño mayor por lo que nosotros las adaptaremos según lo siguiente:

- Planificación de sprint: Estas reuniones toman lugar en la fecha de inicio del sprint. En estas se definen las historias de usuario del backlog que se van a llevar a cabo en el siguiente sprint. La duración de estas en un equipo tradicional es de 1 hora. En nuestro caso la duración de esta será 45 minutos y estaremos presentes tanto alumno como tutor.
- *Daily*: Las reuniones diarias que plantea Scrum se realizarían en forma de stand-up en la que cada miembro del equipo actualiza al resto sobre el estado de sus tareas y estado general del sprint, y tienen comúnmente una duración de 15 minutos.
  En el caso de nuestro proyecto estas reuniones se suprimen dado que solo asistiría a esta el alumno y en su lugar se establece un tiempo antes de iniciar con el trabajo diario en el que repasar el estado de las tareas, fijar los objetivos del día e identificar problemas o bloqueos para poder comunicárselo a la tutora lo antes posible.
- Revisión de sprint y retrospectiva de sprint: Estas dos reuniones se celebran al finalizar el sprint y se centrar, por una parte, en revisar las tareas para valorar si se pueden considerar hechas y, por otra parte, en identificar y analizar problemas que se hayan manifestado en el anterior sprint. Estas dos reuniones se eliminan como tal en nuestro proceso pero los temas que se tratarían en estas reuniones se desplazan al inicio de las reuniones de planificación de sprint.

Las reuniones definidas anteriormente se realizarán por videollamada, o en persona si la disponibilidad de ambos componentes lo permite.
De esta forma, hemos hecho uso de los principios que plantea el marco de trabajo ágil Scrum para definir las bases de un proceso que se adapta a las necesidades especiales de nuestro proyecto. En la Tabla 2.2 se resumen las principales características de esta adaptación.

2.2. Stakeholders

Los stakeholders de un proyecto son aquellas personas, organizaciones o empresas que están interesadas y/o son afectadas por el proyecto en cuestión. En el presente proyecto tendremos en cuenta a los siguientes:

Roles	
Rol	Personal
Producto Owner	Tutora
Scrum Master	Alumno
Equipo de Desarrollo	Alumno
Sprint	
Concepto	Detalles
Duración de Sprint	2 semanas
Trabajo semanal	17 horas
Estimación	Escala Fibonacci
Eventos	
Evento	Detalles
Daily	Sustituida por reflexión del alumno
Planificación de Sprint	Inicio de cada sprint. Alumno y Tutora.
Revisión y Retrospectiva de Sprint	Incluidas en Planificación de Sprint

Tabla 2.2: Resumen metodología Scrum definida

1. **Alumnos**: Comprende a todos los alumnos, matriculados y no matriculados, que tienen intención de comenzar su proyecto TFG. Estos alumnos tienen interés en poder encontrar con mayor facilidad toda la información relativa a la asignatura TFG así conocer todos los temas disponibles presentados por los profesores de una forma más ágil a la actual.
2. **Tutores**: En este grupo se encuentran aquellos profesores que se ofrecen y tutelan proyectos a los alumno en el contexto del TFG. Estos tienen interés en una plataforma moderna que les permite publicar sus ofertas de TFG y la presentación al Comité de Título de la propuesta de TFG y la propuesta de adjudicación.
3. **Coordinador de título**: Este rol es el encargado de la gestión referente a los TFG y que actualmente se realiza de forma manual, por lo que una automatización de los diferentes procesos es su principal interés.
4. *Técnicos de la Escuela*: Serán ellos los encargados del mantenimiento de la aplicación una vez finalizado el TFG.

2.3. Análisis de riesgos

En la Tabla 2.3 se describen los riesgos identificados para el desarrollo del proyecto. También, con el objetivo de evaluar la necesidad de tomar medidas de prevención y contención utilizaremos la tabla mostrada en la Tabla 2.4 a fin de identificar aquellos que suponen un mayor peligro. Se prestará especial atención a aquellos que supongan una exposición media o alta.

A partir de esta información, en la Tabla 2.5 se describen los planes de mitigación y contingencia tomados para los riesgos identificados. Entendemos por plan de mitigación al

Riesgos		
ID	**Nombre**	**Descripción**
R01	Enfermedad	Durante los meses de trabajo en el proyecto puedo caer enfermo y que esto afecte a mi productividad y disponibilidad durante ese tiempo
R02	Comunicación con tutora	A lo largo del proyecto podrían darse momentos en los que la comunicación con la tutora no fuese posible provocando bloqueos en el desarrollo del proyecto
R03	Trabajo externo	A la vez que el desarrollo de este proyecto me encontraré trabajando, lo que podría limitar mi tiempo disponible y la productividad.
R04	Estimación de tareas	Al principio del proyecto es natural no tener afinado el tiempo estimado que puede requerir cada tarea, además es común que estos errores sean subestimando el trabajo que es requerido para cada tarea.
R05	Problemas con el entorno	Un cambio de versión en alguna de las herramientas usadas durante el proyecto puede ocasionar problemas de compatibilidad ralentizando el plan de desarrollo del proyecto.
R06	Cambios en los requisitos	Tras, o durante, la realización de una tarea pueden surgir cambios en los requisitos de esa parte que requiera de una modificación total o parcial del trabajo realizado.

Tabla 2.3: Descripción de los riesgos

ID	Probabilidad	Impacto	Exposición
R01	Alta	Medio	Alta
R02	Baja	Alto	Media
R03	Alta	Bajo	Media
R04	Alta	Medio	Alta
R05	Medio	Medio	Media
R06	Baja	Medio	Baja

Tabla 2.4: Exposición de los riesgos

ID	Plan de contingencia	Plan de mitigación
R01	Riesgo ajenos a nuestro control	Recuperar las horas perdidas al recuperarse. Replanificar y retrasar la fecha de entrega del proyecto
R02	Establecer reuniones periódicas con la tutora. Utilizar medios de comunicación ágiles	Migrar el trabajo a tareas que no requieran de la atención inmediata de la tutora. Replanificar y retrasar la fecha de fin del proyecto
R03	Establecer un horario de trabajo que me permita compatibilizar trabajo y proyecto.	Negociar vacaciones con la empresa si se requiere en alguna semana. Replanificar y retrasar la fecha de finalización.
R04	Programar los primeros sprints de forma generosa con el fin de obtener un conocimiento asentado del coste real de las historias.	Utilizar el colchón de horas de sprints anteriores para suplir el mal cálculo en sprints siguientes. Mover tareas a los sprints siguientes.
R05	Trabajar sobre versiones estables. Respaldar el proyecto en diferentes medios de almacenamiento.	Utilizar el colchón de horas para arreglo del entorno. Replanificar y retrasar la finalización del proyecto.
R06	Establecer medios para identificar cambios en los requisitos antes de trabajar en su implementación. Mantener un canal de comunicación fluido con las diferentes partes interesadas en el proyecto.	Utilizar el colchón de horas para modificar módulos afectados. Replanificar y retrasar la fecha de finalización del proyecto.

Tabla 2.5: Medidas de mitigación y contingencia

conjunto de acciones que tienen como objetivo reducir la probabilidad de que el riesgo se manifieste, y como plan de contingencia aquellas que se llevarán a cabo si el riesgo se produce.

2.4. Planificación y calendarización

A continuación se tratará la planificación y la calendarización. En esta sección se repasará el *backlog* y épicas iniciales que se obtiene a partir de lo que se desarrollará en la Sección 3. Únicamente, y para situar el origen y motivo de estas tareas, comentaremos que esto se han obtenido a partir de la documentación comentada en el Capítulo 1 y de entrevistas con el actual Coordinador de Grado, el profesor José Manuel Marqués Corral.

ID	Épica
EP01	Identificación
EP02	Gestión de propuestas de TFGs
EP03	Gestión de adjudicaciones de TFGs
EP04	Gestión de defensas de TFGs
EP05	Gestión de tribunales de TFGs

Tabla 2.6: Épicas del proyectos

ID	Épica	Nombre
B01	EP01	Identificación con el sistema CAS de la UVa
B02	EP01	Autenticación *backend-frontend*
B03	EP02	Creación de propuesta
B05	EP02	Publicación de propuesta
B06	EP02	Creación de propuesta
B07	EP03	Creación de adjudicación
B08	EP03	Publicación de adjudicación
B09	EP03	Modificación de adjudicación
B10	EP04	Creación de defensa
B11	EP04	Publicación de defensa
B12	EP05	Creación de tribunales de
B13	EP05	Publicación de tribunales
B14	EP05	Modificación de tribunal

Tabla 2.7: Backlog inicial del proyecto

2.4.1. Épicas

Al inicio del proyecto establecemos las épicas como se definen en la Tabla 2.6 partiendo de lo que se tratará en la Sección 3.

2.4.2. Backlog inicial

A partir de las épicas establecidas se han obtenido, mediante la identificación de subtareas en cada épica, las tareas mostradas en la Tabla 2.7.

2.4.3. Backlog final

A lo largo del proyecto se añadió una tarea al backlog, el estado del backlog al final del proyecto se resume en la Tabla 2.8. En concreto la tarea en dicha tabla identificada como BF16 se crea a partir de un requerimiento técnico encontrado al desarrollar la autenticación con el servicio SSO de la UVa, basado en CAS.

ID	Épica	Deriva de	Nombre
B01	EP01		Identificación con el sistema CAS de la UVa
B02	EP01		Autenticación *backend-frontend* con JWT
B16	EP01	B01	Despliegue de aplicaciones en HTTPS
B03	EP02		Creación de propuesta
B05	EP02		Publicación de propuesta
B06	EP02		Creación de propuesta
B07	EP03		Creación de adjudicación
B08	EP03		Publicación de adjudicación
B09	EP03		Modificación de adjudicación
B10	EP04		Creación de defensa
B11	EP04		Publicación de defensa
B12	EP05		Creación de tribunales de
B13	EP05		Publicación de tribunales
B14	EP05		Modificación de tribunal

Tabla 2.8: Backlog final del proyecto

2.4.4. Calendarización

Al inicio del proyecto se establece una calendarización inicial para el proyecto en base a toda la información que se desarrollará y se ha desarrollado en esta sección. La calendarización para el trabajo se definió con inicio el 14 de septiembre de 2023 y con fecha de finalización el 12 de marzo de 2024 tal y como se recoge en la Tabla 2.9.

2.5. Presupuesto

Para calcular el presupuesto simulado real de este proyecto debemos tener en cuenta los siguientes factores a analizar como son, las licencias de las aplicaciones o herramientas utilizadas durante el desarrollo y despliegue.
Con respecto a las licencias de todas las herramientas, estas son gratuitas para su uso académico. Por otro lado, para el cálculo del coste humano tomaremos como referencia el salario medio en España de un desarrollador Junior y que se sitúa en los 1800 €/mes [6] en jornada completa, en nuestro caso y como se indicó en la Sección 2.1.2 estaremos a 42.5 % (17 horas semanales) correspondiente a una jornada completa. A este coste se le añadirá un 30 % de Seguridad Social. Considerando que la duración del proyecto es de 300h utilizamos la siguiente fórmula para estimar el coste de personal (CP).

$$CP = \frac{1800\frac{€}{mes} * 0{,}425 * 300h * 1{,}3}{68\frac{h}{mes}} = 3375€ \tag{2.1}$$

Por último tendremos en cuenta el costo de materiales (CH) para lo cuál solo se tiene en cuenta el equipo sobre el que se ha desarrollado el proyecto y que es un portátil Acer Swift 13

Hito	Inicio	Fin	Comentarios
Fase inicial	14/09/2023	07/11/2023	
Reunión Semanal	Se realizan cada miércoles.		Se realizaron reuniones cada semana en las que se planificaba el inicio del proyecto, recogían requisitos y alguna reunión con el Coordinador de Grado. La forma de trabajo durante esta fase fue en cascada, desde la elicitación y análisis de requisitos hasta el diseño de la arquitectura
Sprint 1	08/11/2023	21/11/2023	
Reunión Semanal	15/11/2023		
Sprint Review, Retrospectiva y Planificación	21/11/2023		
Sprint 2	22/11/2023	05/12/2023	
Reunión Semanal	29/11/2023		
Sprint Review, Retrospectiva y Planificación	05/12/2023		
Sprint 3	06/12/2023	19/12/2023	
Reunión Semanal	13/12/2023		
Sprint Review, Retrospectiva y Planificación	19/12/2023		
Sprint 4	20/12/2023	02/01/2024	
Reunión Semanal	27/12/2023		
Sprint Review, Retrospectiva y Planificación	02/01/2024		
Sprint 5	03/01/2024	16/01/2024	
Reunión Semanal	10/01/2024		
Sprint Review, Retrospectiva y Planificación	16/01/2024		
Sprint 6	17/01/2024	30/01/2024	
Reunión Semanal	24/02/2024		
Sprint Review, Retrospectiva y Planificación	30/01/2024		
Sprint 7	31/01/2024	13/02/2024	
Reunión Semanal	07/02/2024		
Sprint Review, Retrospectiva y Planificación	13/02/2024		
Sprint 8	15/02/2024	27/02/2024	
Reunión Semanal	21/02/2024		
Sprint Review, Retrospectiva y Planificación	27/02/2024		
Cierre de proyecto	28/02/2024	12/03/2024	
Reunión Semanal	06/03/2024		
Sprint Review, Retrospectiva y Planificación	12/03/2024		Este último período se dedicará a corregir la memoria y finalizar la documentación que quede pendiente.

Tabla 2.9: Calendarización del proyecto

equipado con una CPU AMD Ryzen 4700U 16GB de memoria física, 1TB de almacenamiento corriendo ArchLinux@6.x-arch1-x adquirido en 2020 por 800 €. A razón de calcular el coste real de uso de este dispositivo actualmente durante el tiempo del desarrollo nos remitiremos a las tablas de amortización simplificada de la Agencia Tributaria [1] y que establecen un coefiente de amortización lineal máximo para bienes de este tipo del 26 %, con un periodo de amortización máximo de 10 años. Para el cálculo de esto tomamos el coeficiente de amortización ya que nos permitiría amortizar el bien en menor tiempo, de esta forma lo cálculos quedan como sigue:

$$CH = \frac{800 € * 0{,}26 \frac{\text{amort}}{\text{año}} * 4{,}41\text{meses}}{12 \frac{\text{mes}}{\text{año}}} = 76,44 € \tag{2.2}$$

Con lo repasado anteriormente podemos desglosar el presupuesto en la siguiente tabla y que supondría un coste total simulado del proyecto de 3451,44 €.

Concepto	Importe (€)
Personal	4 387,5
Software	0
Hardware	76,44
Total	4 463,94

Tabla 2.10: Presupuesto simulado

Al final del Capítulo 7 se muestra un resumen de la ejecución del proyecto y un análisis de los costes frente a lo presupuestado.

Capítulo 3

Análisis

De cara a iniciar el desarrollo de este proyecto se requiere antes de una fase de análisis del problema con el objetivo de identificar el dominio de este, y de las necesidades que se pretenden resolver. Como primera etapa en esta fase de análisis se han llevado a cabo de forma paralela el modelado de dominio y el modelado de procesos cuyo resultado se presenta a continuación.

3.1. Modelo de proceso

Los procesos a automatizar o simplificar comprenden la creación y renovación de asignaciones y adjudicaciones de TFG, creación de tribunales y creación de sesiones de defensa con la consiguiente asignación de tribunal. Estos procesos, realizados actualmente por medio de correos o medios analógicos, se describen en los modelos de proceso representados mediante diagramas de actividades UML. La definición y validación de estos se ha realizado a través de comunicaciones y reuniones con el Coordinador de Título. En las Figuras 3.1, 3.2 y 3.3 se muestran los diagramas de proceso frutos de estas entrevistas.

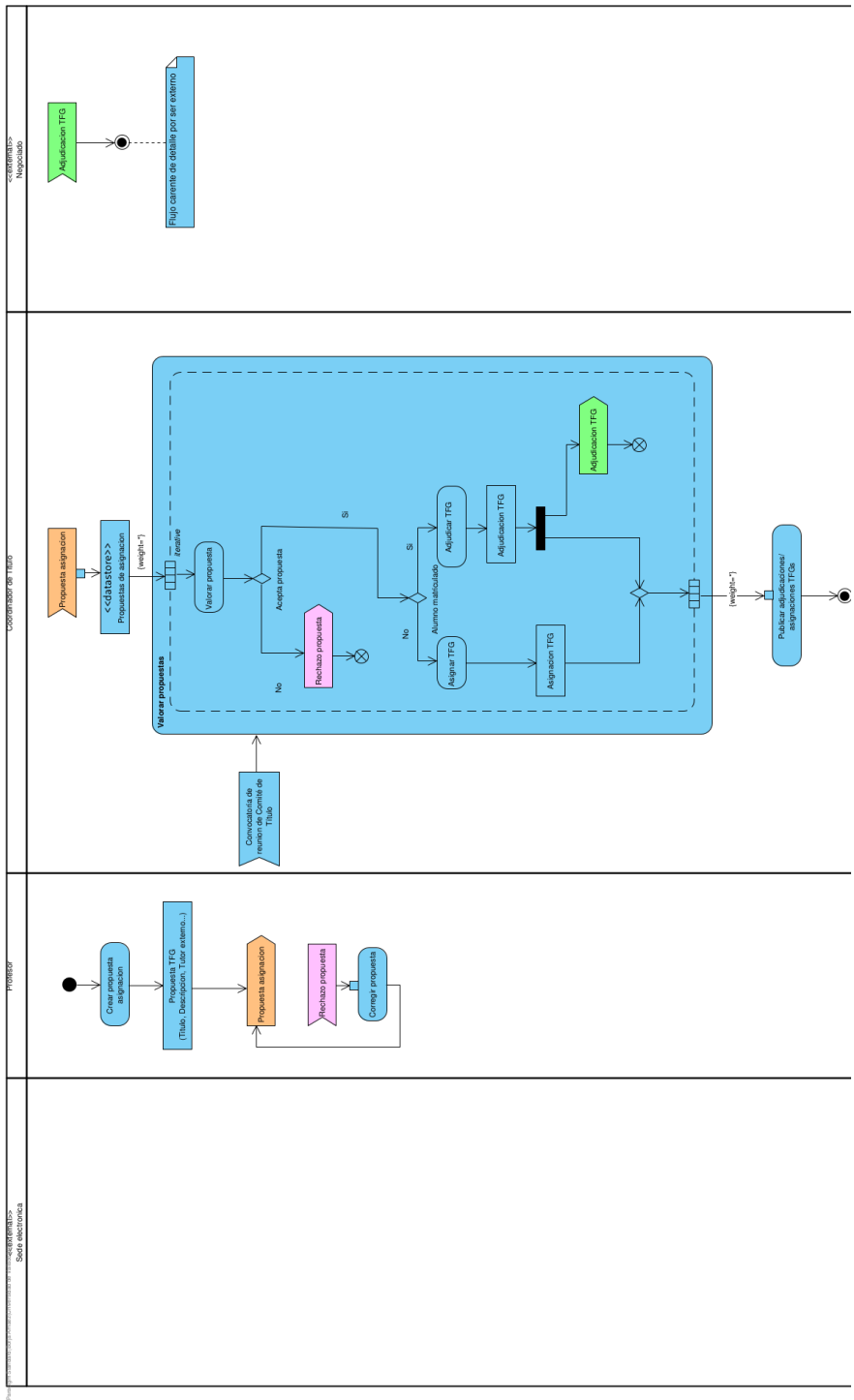

Figura 3.1: Fragmento del modelo de proceso centrado en la creación de propuesta, asignación y adjudicación

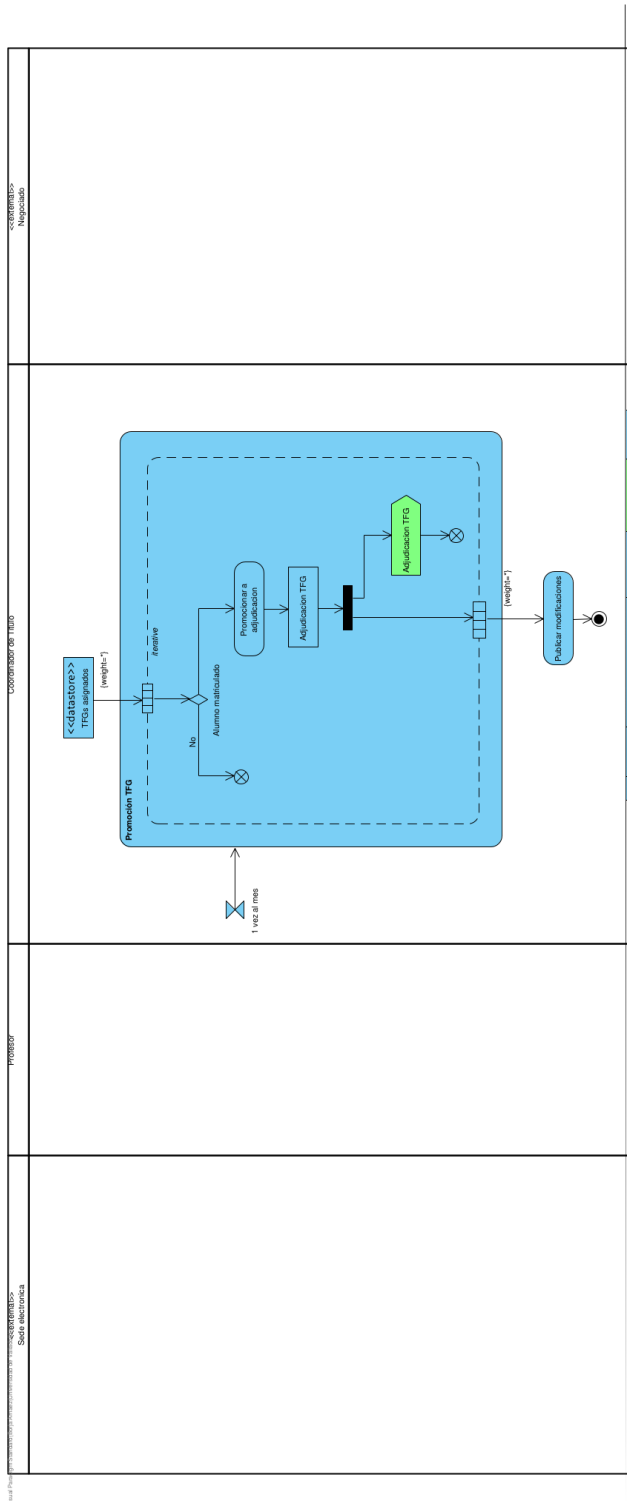

Figura 3.2: Fragmento del modelo de proceso centrado en la adjudicación de una asignación

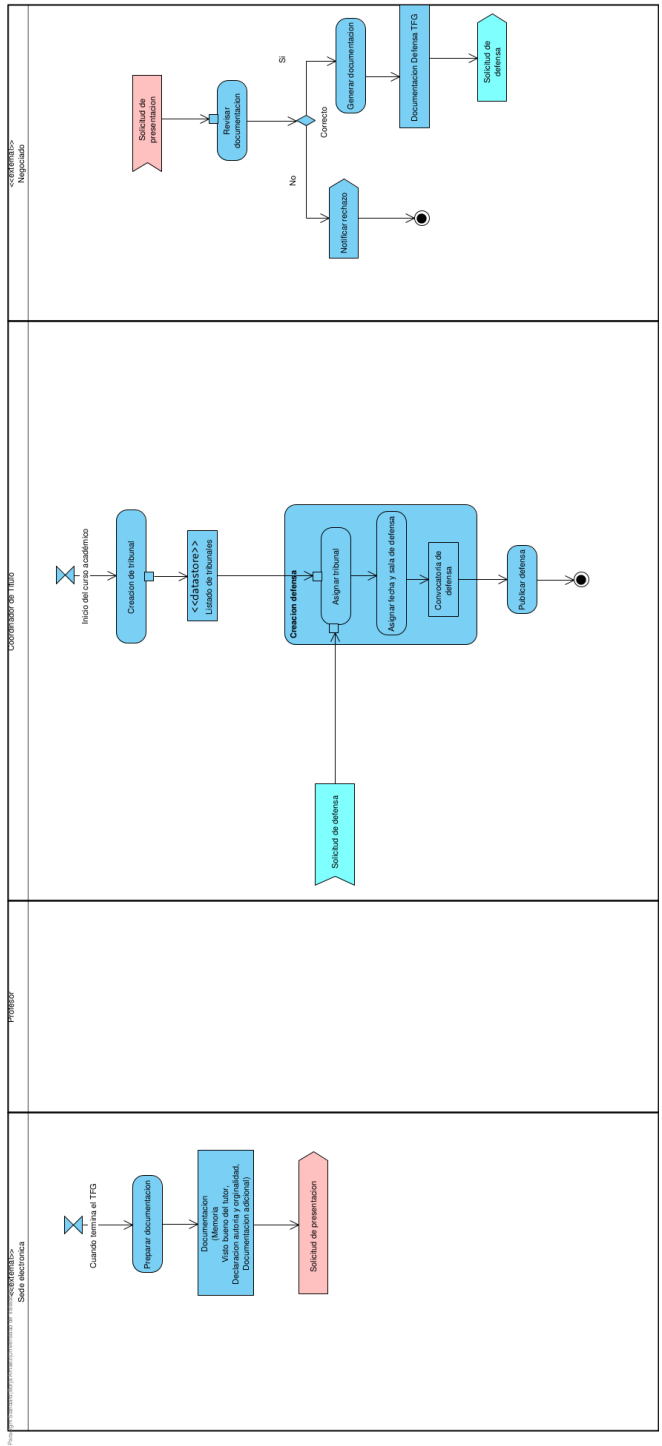

Figura 3.3: Fragmento del modelo de proceso centrado en la creación de tribunales y defensas

3.2. Modelado de dominio

Para la modelización del dominio se ha obtenido información de las fuentes mencionadas en la Sección 1 como son [19] y [18]. En la Figura 3.4 se muestra el modelo del dominio analizado representado mediante un diagrama de clases UML.

En relación con la adjudicación de un TFG se distinguen los términos asignación vs adjudicación. Para poder realizar una adjudicación el estudiante debe estar matriculado en la asignatura Trabajo de Fin de Grado. Se produce una adjudicación formal que se envía a Secretaría Administrativa (Negociado) para que quede reflejada en el expediente del estudiante. Cuando el estudiante no está matriculado, si el Comité de Título recibe la propuesta de un profesor de adjudicar un TFG a un estudiante, se anota como ASIGNADO. Es una asignación interna pero aún no puede reflejarse en el expediente.

3.3. Casos de uso

En la Figura 3.5 se presentan, en un diagrama UML de casos de uso, todos los casos de uso identificados durante el análisis del proyecto. Posteriormente, se especifica textualmente la interacción que ocurre en cada caso de uso en las Tablas de la 3.1 a la 3.12.

En algunas de las descripciones de los siguientes casos de uso se hablará de transiciones de estados permitidos para lo cual se dispone del diagrama de estado que se muestra en la Figura 3.6. En dicho diagrama se marcan las transiciones con eventos que se nombran con el identificador del caso de uso que pueda provocar el cambio de estado.

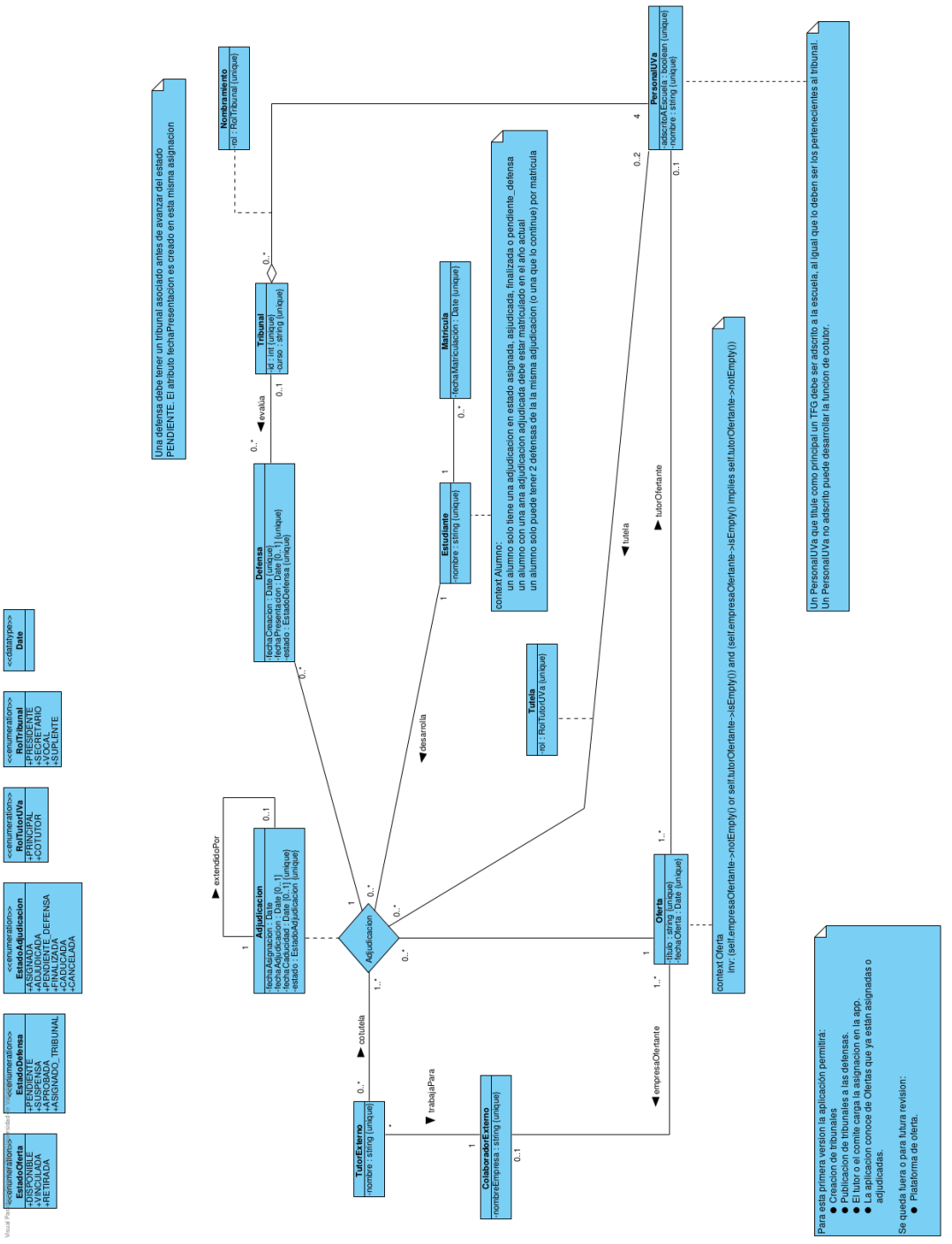

Figura 3.4: Modelo del dominio

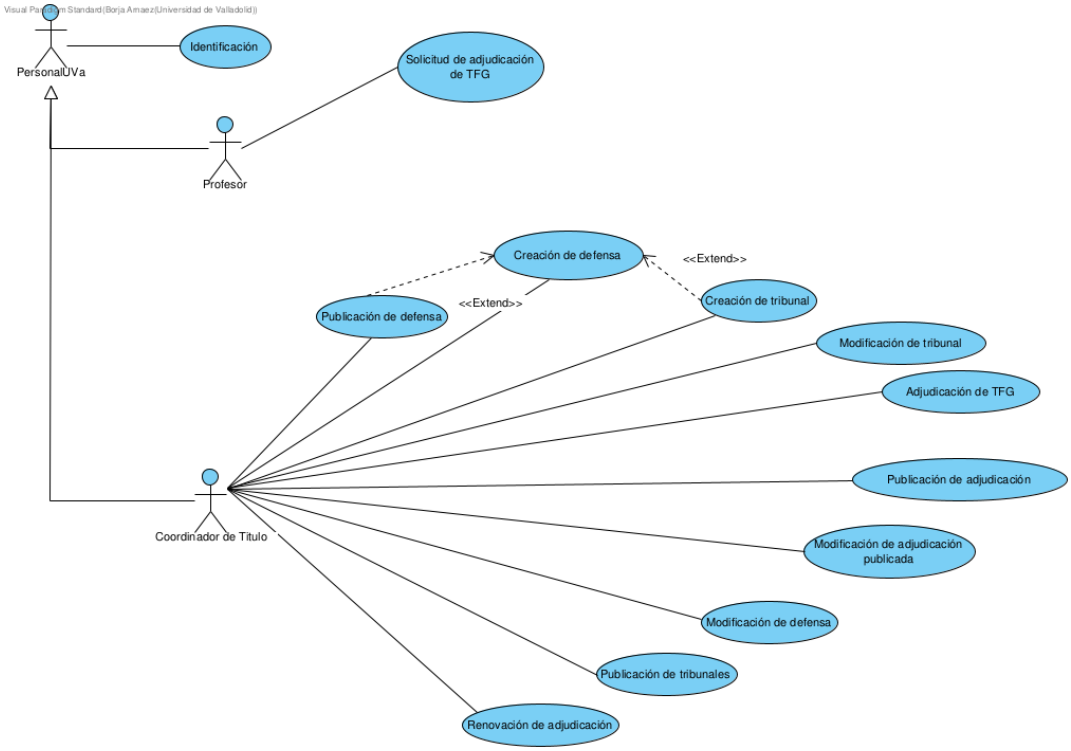

Figura 3.5: Diagrama de Casos de Uso

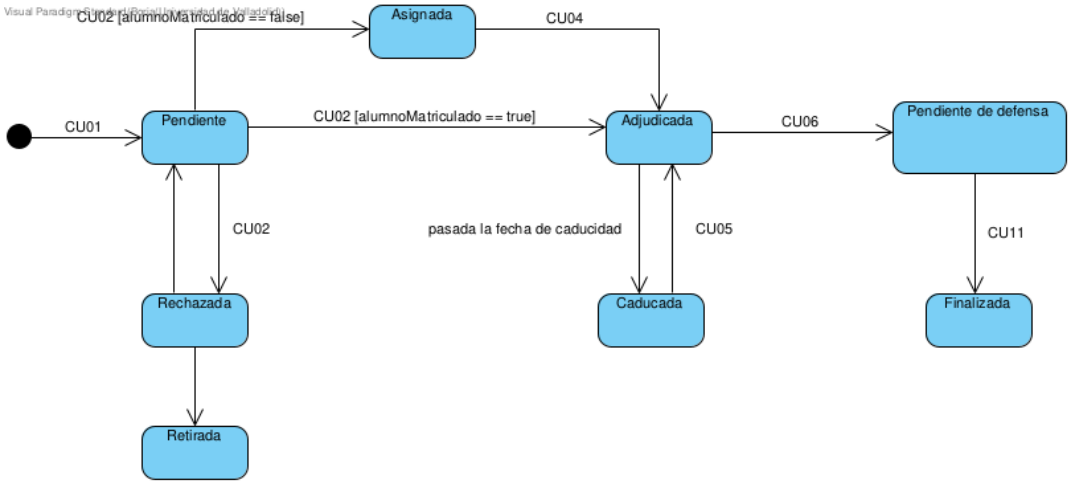

Figura 3.6: Diagrama de estados de una propuesta o adjudicación

CU00	Identificación
Actor	PersonalUVa
Descripción	El actor se identifica en el sistema con sus credenciales UVa.
Precondiciones	
1	El actor no está identificado en el sistema.
Flujo principal	
1	Actor solicita identificarse en el sistema.
2	SISTEMA solicita el usuario y contraseña.
3	Actor introduce usuario y contraseña.
4	SISTEMA comprueba la validez de los datos introducidos e identifica al usuario en el sistema.
El caso de uso termina.	
Flujos alternativos	
4.1	Si los datos no son correctos, SISTEMA informa al usuario y se vuelve al paso 2.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.1: CU Identificación

CU01	Solicitud de adjudicación de TFG
Actor	Profesor
Descripción	El actor crea una nueva propuesta de tutoría de TFG.
Precondiciones	
1	El actor está identificado en el sistema.
2	El actor posee los permisos para realizar la solicitud.
Flujo principal	
1	El actor solicita la creación de una nueva adjudicación de TFG.
2	SISTEMA solicita al usuario el alumno que va a realizar el TFG.
3	El actor introduce el alumno.
4	SISTEMA comprueba que el alumno no posea otros TFG en los estados asignado, adjudicado, pendiente de defensa o finalizado.
5	SISTEMA solicita al usuario el título y descripción del TFG.
6	El actor introduce el título y descripción del TFG.
7	SISTEMA muestra el alumno, título y descripción aportados y solicita confirmación al usuario para enviar la solicitud.
8	El actor confirma el envío de la solicitud
9	SISTEMA almacena la solicitud
El caso de uso termina.	
Flujos alternativos	
4.1	Si el alumno no es elegible para crear la solicitud se informa al usuario y se vuelve al paso 2.
8.1	Si el actor no confirma el envío se vuelve al paso 2.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.2: CU Solicitud de adjudicación TFG

CU02	**Adjudicación de TFG**
Actor	Coordinador de Título
Descripción	El actor revisa y aprueba o rechaza una propuesta de TFG pendiente.
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.
	Flujo principal
1	El actor solicita las propuestas pendientes de revisión.
2	SISTEMA muestra las propuestas de TFG pendientes de revisión.
3	El actor selecciona una propuesta.
4	SISTEMA muestra los detalles de la propuesta.
5	El actor revisa la propuesta y la acepta.
6	SISTEMA solicita que se indique si se hará una asignación o adjudicación.
7	El actor indica el tipo de la adjudicación.
8	SISTEMA almacena la nueva adjudicación y modifica el estado de la propuesta.
El caso de uso termina.	
	Flujos alternativos
8.1	Si el tipo es adjudicación, el SISTEMA enviará un correo al negociado con los detalles de la adjudicación.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.3: CU Adjudicación de TFG

CU03	**Publicación de adjudicaciones**
Actor	Coordinador de Título
Descripción	El actor publica en la plataforma las adjudicaciones de TFG aprobadas.
	Precondiciones
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.
	Flujo principal
1	El actor inicia el caso de uso.
2	SISTEMA muestra las adjudicaciones pendientes de publicación y solicita que se acepte la publicación de estas.
3	El actor revisa las adjudicaciones y acepta su publicación.
4	SISTEMA actualiza los estados de las propuestas según la elección en el paso anterior.
El caso de uso termina.	
	Flujos alternativos
3.1	Si el actor cancela la publicación, el caso de uso se cancela y queda sin efecto.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.4: CU Publicación de adjudicaciones

CU04	**Modificación de adjudicación publicada**
Actor	Coordinador de Título
Descripción	El actor modifica una adjudicación de TFG ya publicada.
Precondiciones	
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.
Flujo principal	
1	El actor selecciona la adjudicación que desea modificar.
2	SISTEMA muestra los detalles de la adjudicación y permite la modificación del estado, título y descripción de esta.
3	El actor modifica la adjudicación.
4	SISTEMA comprueba que el cambio de estado está permitido y actualiza la adjudicación.
5	SISTEMA permite la modificación del tutor de la adjudicación.
6	El actor introduce el nuevo tutor.
7	SISTEMA crea una nueva adjudicación con los mismos detalles a la anterior con el nuevo tutor.
El caso de uso termina.	
Flujos alternativos	
3.1	Si no realiza modificación se salta al paso 5.
4.1	Si el cambio de estado es de ASIGNADO a ADJUDICADO, se enviará un correo al negociado con los detalles del TFG.
6.1	Si no realiza cambio de tutor el caso de uso finaliza.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.5: CU Modificación de adjudicación publicada

CU05	**Renovación de adjudicación**
Actor	Coordinador de Título
Descripción	El actor renueva una adjudicación caducada o próxima a caducar.
Precondiciones	
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.
Flujo principal	
1	El actor selecciona la adjudicación que desea renovar.
2	SISTEMA muestra los detalles de la adjudicación y solicita de la confirmación de renovación.
3	El actor autoriza la renovación.
4	SISTEMA comprueba que el cambio de estado está permitido.
5	SISTEMA crea una nueva adjudicación sucesora con los mismos detalles a la anterior y actualiza el estado de la anterior.
El caso de uso termina.	
Flujos alternativos	
3.1	Si no se autoriza la renovación el caso de uso se cancela y queda sin efecto.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.6: CU Renovación de adjudicación

CU06	**Creación de defensa**
Actor	Coordinador de Título
Descripción	El actor crea una nueva defensa para una adjudicación de TFG.

Precondiciones	
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.

Flujo principal	
1	El actor selecciona la adjudicación que para la que se desea crear defensa.
2	SISTEMA comprueba que el estado de esta sea ADJUDICADA o PENDIENTE_DE_DEFENSA.
3	SISTEMA muestra las información de la asignación.
4	SISTEMA solicita el tribunal de la defensa.
5	El actor selecciona el tribunal.
6	SISTEMA comprueba que el tribunal seleccionado es apto para el TFG.
5	SISTEMA solicita que se indique la fecha y aula de defensa.
6	El actor introduce la fecha y aula.
7	SISTEMA registra la defensa y actualiza el estado de la adjudicación.
8	SISTEMA ofrece al usuario publicar la defensa.
9	El actor acepta publicar la defensa.
10	SISTEMA lanza el caso de uso *07 Publicación de defensa*.
El caso de uso termina.	

Flujos alternativos	
2.1	Si el estado de la adjudicación no es ADJUDICADA, se informa al usuario y el caso de uso se cancela y queda sin efecto.
5.1	El actor puede solicitar crear un tribunal en vez de elegir de los disponibles, se lanzará el caso de uso *08 Creación de tribunal*. Al finalizar este, el caso de uso continúa en el paso 6.
9.1	Si no acepta publicar la defensa el caso de uso finaliza.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.7: CU Creación de defensa

CU07	**Publicación de defensa**
Actor	Coordinador de Título
Descripción	El actor publica en la plataforma una defensa de una adjudicación de TFG.
	Precondiciones
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.
	Flujo principal
1	El actor inicia el caso de uso.
2	SISTEMA muestra los detalles de la defensa y solicita autorización para su publicación.
3	El actor acepta publicar la defensa.
4	SISTEMA actualiza la defensa.
El caso de uso termina.	
	Flujos alternativos
3.1	Si El actor rechaza publicar la defensa el caso de uso se cancela y queda sin efecto.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.8: CU Publicación de defensa

CU08	**Creación de tribunal**
Actor	Coordinador de Título
Descripción	El actor crea un tribunal para las defensas de TFGs.
	Precondiciones
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.
	Flujo principal
1	El actor inicia el caso de uso.
2	SISTEMA solicita que se introduzcan los 4 profesores que compondrán el tribunal.
3	El actor introduce los profesores con sus respectivos roles.
4	SISTEMA comprueba la validez de las relaciones profesor-rol.
5	SISTEMA crea el tribunal.
El caso de uso termina.	
	Flujos alternativos
4.1	Si las relaciones no son válidas, se informa al usuario y el caso de uso vuelve al paso 2.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.9: CU Creación de tribunal

CU09	**Publicación de tribunales**
Actor	Coordinador de Título
Descripción	El actor publica en la plataforma los tribunales ya creados.
Precondiciones	
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.
Flujo principal	
1	El actor inicia el caso de uso.
2	SISTEMA muestra los tribunales pendientes de publicación con sus detalles y solicita la confirmación de publicación.
3	El actor confirma publicar los tribunales.
4	SISTEMA actualiza los tribunales.
El caso de uso termina.	
Flujos alternativos	
3.1	Si El actor rechaza publicar los tribunales el caso de uso se cancela y queda sin efecto.
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.10: CU Publicación de tribunales

CU10	**Modificación de tribunal**
Actor	Coordinador de Título
Descripción	El actor modifica los componentes de un tribunal o despublicarlo.
Precondiciones	
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.
Flujo principal	
1	El actor selecciona el tribunal a modificar.
2	SISTEMA muestra los detalles del tribunal y permite la modificación de sus integrantes y roles y eliminar su publicación.
3	El actor realiza las modificaciones.
4	SISTEMA actualiza el tribunal.
El caso de uso termina.	
Flujos alternativos	
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.11: CU Modificación de tribunal

CU11	**Modificación de defensa**
Actor	Coordinador de Título
Descripción	El actor modifica las características de una defensa.
Precondiciones	
1	El actor está identificado en el sistema.
2	El actor posee el rol de Coordinador de Título.
Flujo principal	
1	El actor selecciona la defensa que desea editar.
2	SISTEMA muestra los detalles de esta y permite la modificación de la fecha, aula y estado.
3	Coordinador de Título realiza las modificaciones.
4	SISTEMA actualiza la defensa.
El caso de uso termina.	
Flujos alternativos	
*	El actor puede cancelar el caso de uso en cualquiera de sus pasos dejándolo sin efecto.

Tabla 3.12: CU Modificación de defensa

Capítulo 4

Tecnologías utilizadas

En este Capítulo se presenta un resumen de las tecnologías utilizadas tanto para la gestión del proyecto, como para el desarrollo.

4.1. Tecnologías de desarrollo

4.1.1. React

React [22] es una librería para el desarrollo de interfaces desarrollada por Meta y la comunidad, con un enfoque en la componentización de elementos de interfaz de usuario para lo cuál utiliza JSX y CSS. JSX [10] es una extensión de sintaxis para JavaScript o TypeScript que permite escribir marcado similar a HTML. Entre las principales ventajas de esta herramienta destaca la sencillez en su aprendizaje.

Como puntos débiles encontramos la falta de algunas funcionalidades de forma nativa, que sí están presentes en frameworks como Angular [2], como puede ser el enrutado de pantallas por dirección o validación de formularios, para lo cuál existen librerías externas que lo implementan.

Antes de continuar con el resto de tecnologías vamos a explicar en que consiste un componente de *React*, que se diferencia en gran medida de uno de *Angular*, con el fin de despejar dudas y a ayudar a entender futuras referencias a estos en texto y figuras.

Componentes React

Empezando por la definición de un componente, teniendo en cuenta de que se trata en todo de caso de lo que denominamos "componentes en el *front*". Un componente es simplemente

una pieza de código reusable y completamente funcional por sí misma, sin significar esto que no dependan de ficheros externos como, por ejemplo, hojas de estilo CSS. Se puede entender como una función de *Javascript* con la peculiaridad de que un componente React debe retornar HTML. Originalmente existían los *Class Components*, actualmente deprecados, que aprovechaban la herencia de objetos de *Javascript*, y que han sido sustituidos por los *Function Components* que, como su nombre indica y de aquí la analogía de las funciones anteriores, son funciones.

Estas funciones, si bien pueden ser definidas en cualquier lugar del código y utilizadas en cualquier otro, hemos decidido definirlas en ficheros independientes, con el fin de tener una estructura del código fácilmente entendible. Estos fichero no contienen tampoco las hojas de estilo de los componentes, las cuáles sen encuentran en ficheros *.css* junto a su componente o componentes en el árbol de directorios. Esta estructura se detalla en las secciones 5.1.3 y 6.1.2.

4.1.2. Vite

Vite [13] es una herramienta de desarrollo cuyo objetivo es facilitar la experiencia del trabajo creando un servidor de desarrollo que permite, entre otras funcionalidades, el reemplazo de módulos de forma rápida ante cambios en el código.

4.1.3. Yarn

Yarn [7] es un gestor de paquetes para Javascript alternativo a NPM para la instalación, actualización, configuración y borrado de paquetes en el proyecto. Se diferencia de NPM en ser más rápido y seguro.

4.1.4. TypeScript

Typescript es un lenguaje de programación desarrollado por Microsoft que ofrece todas las funcionalidades de Javascript con algunos añadidos extra como el tipado, tipos, interfaces y genericidad. Actualmente, en proyectos complejos está sustituyendo a Javascript por la seguridad que ofrece sobre este.

4.1.5. Java

Java es un lenguaje de programación desarrollado por Oracle de orientación a objetos consolidado y altamente utilizado para el desarrollo de aplicaciones backend, móviles y de escritorio entre otras. En el desarrollo se ha decidido contar con la versión 21 LTS del lenguaje.

4.1.6. Springboot

Springboot [12] es un framework de Java enfocado al backend que facilita el desarrollo de aplicaciones y microservicios. Cuenta con extensiones que permiten gestionar la conexión con diferentes bases de datos, implementación de un servidor REST y control de la seguridad.

4.1.7. Postman

Postman es una herramienta para la construcción y uso de APIs que ofrece al usuario de gran variedad de soluciones de cara a acelerar el desarrollo y las pruebas de estas APIs. En nuestro caso se ha utilizado para realizar pruebas sobre el servicio backend.

4.1.8. Git y GitLab

Git es un software de control de versiones para el código. Este lo complementaremos con GitLab como gestor de repositorios y soluciones DevOps.

4.1.9. Jetbrains

Para el desarrollo se usará Webstorm, para Typescript con React, e IntelliJ IDEA, para Springboot, respectivamente. Se ha decidido usar estas herramientas de Jetbrains, frente a otras como Visual Studio Code, principalmente por las funcionalidades integradas de forma nativa en estos IDE. La versión de este software es la Ultimate, las cuáles Jetbrains ofrece gratuitamente a estudiantes.

4.1.10. Docker

Docker es una herramienta que permite el empaquetado de software en lo que este llama contenedores que incluyen todo lo necesario para que este software se ejecute. De esta forma se crea una capa de abstracción entre software y hardware, asegurando el funcionamiento correcto independientemente del entorno [16].

4.1.11. NGINX

NGINX es un servidor web de código abierto que además ofrece soluciones como proxy inverso, caché HTTP o balanceador de carga. En nuestro proyecto lo utilizaremos únicamente como servidor web para el despliegue de la aplicación web.

4.2. Tecnologías de comunicación

4.2.1. Teams

Teams es la plataforma oficial definida por la Universidad de Valladolid para la comunicación y será utilizada para tales fines entre alumno y tutora.

4.3. Tecnologías de gestión

4.3.1. GitLab Issue Tracker

Herramienta integrada en el entorno de GitLab y que permite la creación de sprints, historias de usuario y *logging* de tiempo. También permite la gestión ágil mediante tableros Kanban.

4.4. Tecnologías de documentación

4.4.1. LaTeX y Overleaf

LaTeX es una herramienta para la creación de documentos cuyo uso está ampliamente extendido en los campos de las ciencias y la tecnología. Como editor de esta herramienta utilizaremos Overleaf una aplicación web que consta de un editor y entorno para LaTeX, además de ofrecer herramientas para la publicación y edición colaborativa de documentos.

4.4.2. Visual Paradigm

Visual Paradigm es una herramienta para la creación de diagramas UML en la ingeniería del software.

Capítulo 5

Diseño

En este capítulo se documentan las decisiones de diseño tomadas para llevar a cabo ambas aplicaciones de nuestro trabajo, la aplicación web y el servidor *backend*. Por ser estas dos aplicaciones independientes dividiremos esta sección en sus dos respectivas partes dando inicio con la aplicación web o *frontend*.

5.1. Arquitectura Frontend

En esta sección repasaremos las decisiones de diseño relativas a la aplicación web que se han tomado durante el desarrollo del proyecto.

En primer lugar, React no establece ninguna estructura para el proyecto. Su enfoque está principalmente en la reutilización de componentes y funcionalidades a largo de la aplicación. Es por esto que la estructura del proyecto debe permitir una organización coherente y natural para los elementos de la aplicación. Comúnmente se recomienda organizar clasificando por tipos de elementos de React o por casos de uso, nosotros nos quedaremos con la primera por ser esta más flexible a la hora de reutilizar elementos de funcionalidad en diferentes casos de uso. Antes de definir la estructura de la aplicación es necesario definir algunos elementos que componen a la biblioteca de React.

1. Componentes: Los componentes son piezas de interfaz, bien puedes ser tan simples como un botón o complejos como un formulario completo o página web. El objetivo de estos es renderizar elementos en la pantalla y manejar lógica necesaria para ese renderizado. De esta forma estaría dentro de su cometido, por ejemplo, el consumo de un *array* para su visualización en pantalla pero no lo estaría su obtención.
2. Hooks: Los hooks son encapsulados que dotan de funcionalidad a los componentes y que permite que esta sea reutilizada. Esta lógica puede comprender la obtención de datos de servicios externos o el procesado de estos.

3. Contextos: Los contextos son la forma que ofrece React de almacenar y exponer estados globales a la aplicación. Uno de los usos más comunes y evidentes de estos son los estilos de la aplicación, usuario identificado o preferencias de este.

A partir de los elementos definidos podemos establecer una relación entre la arquitectura diseñada para nuestra aplicación y el patrón arquitectónico MVVM.

5.1.1. MVVM

El patrón Model-View-View Model, conocido por sus siglas MVVM, es un patrón de arquitectura de software creado con el objetivo de separar la interfaz de la aplicación (UI) de la lógica de aplicación y de la lógica de negocio [3]. Este patrón consta de tres elementos principales que se describen a continuación:

- View (Vista): La vista es la pieza responsable de mostrar al usuario la información, así como la estructura y el diseño de esta en la pantalla. Este elemento también puede contener lógica, pero solo debe ser lógica relacionada con estados de la vista. Es decir, indicar tiempos de carga, la no disponibilidad de una opción o mensajes mediante el cambio de los estados de los elementos que componen la vista.

  En nuestra arquitectura de React, son los componentes los que realizan esta función.

- Model (Modelo). El modelo representa el modelo de dominio de la aplicación que comprende la lógica de negocio y la capa de datos. En ningún caso este debe depender de ninguna forma con la vista.

  En nuestra aplicación React son los contextos los que contienen los datos de la aplicación.

- View Model (Modelo de vista). Actúa como una abstracción entre la vista y el modelo. Ofrece los datos que se muestran a la vista y nutre de propiedades y comandos a esta. También notificará a la vista de los cambios de estado en el modelo para que sea la vista quién decida cómo responder a dichos cambios.

  Con respecto a nuestra arquitectura React, son los hooks los que actúan como modelos de vista, suministrando a los componentes de los datos para su representación y de funciones o comandos. Si bien estos hooks también contienen lógica para la obtención de datos desde el servidor, la cuál en algunas propuestas de MVVM deberían encontrarse en el modelo, son lo más cercano al Modelo de la vista que tenemos.

5.1.2. Diferencias de MVVM con MVC

Resulta interesante detenernos para desarrollar en mayor profundidad la motivación detrás de la elección de MVVM siendo MVC (Modelo-Vista-Controlador) un patrón bien conocido y aplicado en algunas asignaturas de este grado. En nuestra aplicación podríamos haber

utilizado cualquiera de estas arquitecturas dado que, y como se han comentado ya, *React* no es un framework y por lo tanto es agnóstico a la estructura de tu proyecto.

Las diferencias entre estas dos arquitecturas son sutiles pero cambian drásticamente la forma en que la aplicación funciona. Para empezar en ambas tenemos tres elementos y que fácilmente por su nombre podemos relacionar como son el Modelo, Vista y Controlador; y Modelo, Vista y Vista-Modelo. En ambas podemos entender el Modelo y la Vista como conceptualmente el mismo elemento con las mismas responsabilidades que se han detallado en la anterior sección.

Apoyándonos en la Figura 5.1 podemos ver como en el MVVM la vista no conoce de la existencia del modelo, dependencia que sí puede existir en MVC, y además como en MVVM la dirección de dependencia es Vista, Vista-Modelo, Modelo permitiendo mayor independencia. Por otro lado, la forma en la que se actualizan los datos tiene lugar desde el Modelo hacia la Vista mediante estrategias como *data-binding*, la cual es la que utilizaremos en nuestra aplicación mediantes los estados de *React*, o eventos. Esto en MVC se lleva a cabo mediante dependencias directas por las cuales el Controlador informa a la Vista de cambios y esta los lee desde el Modelo.

Una arquitectura MVVM nos ofrece de esta forma una mayor independencia entre las partes de la arquitectura haciendo más fácil probar los diferentes componentes en aislamiento.

5.1.3. Estructura del proyecto

A continuación vamos a repasar los directorios que componen la aplicación. Y en las Figuras 5.2 y 5.3 se muestras el árbol de directorios de la aplicación y el diagrama de arquitectura de la aplicación respectivamente.

- assets: recursos usados en la aplicación como pueden ser imágenes, fuentes, iconos etc.
- components: componentes de React complejos de la interfaz. Por esto nos referimos a elementos como tablas, tarjetas o encabezados entre otros.
- contexts: los contextos definidos anteriormente estarán bajo este directorio.
- hooks: aquí se encontrarán los hooks.
- services: en este directorio se encontrará la lógica necesaria para comunicarse con la API externa.
- screens: en screens se encontrarán las páginas web, que también son, a su vez, componentes de React.

En el Capítulo 6, concretamente en el apartado 6.1.2, se explica un poco más en detalle cómo con la arquitectura definida se desarrolla cada caso de uso.

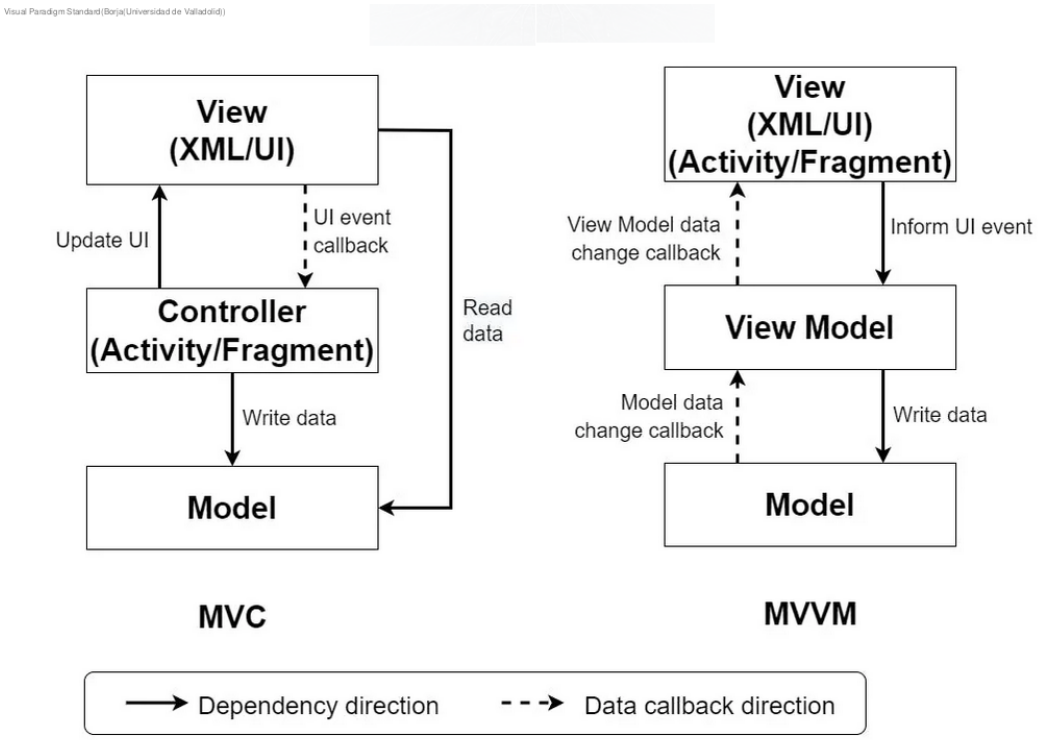

Figura 5.1: MVC vs MVVM

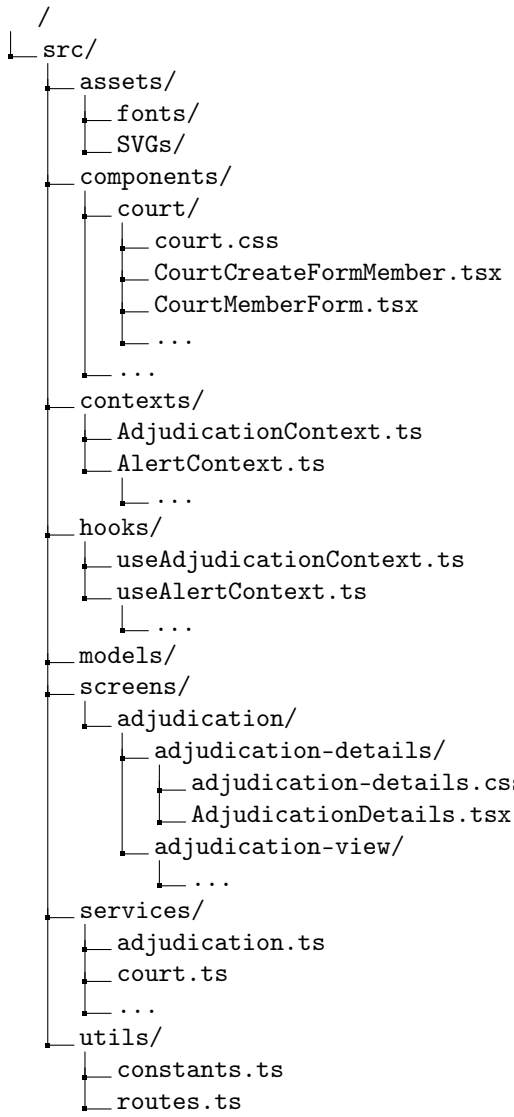

Figura 5.2: Árbol de directorios del frontend

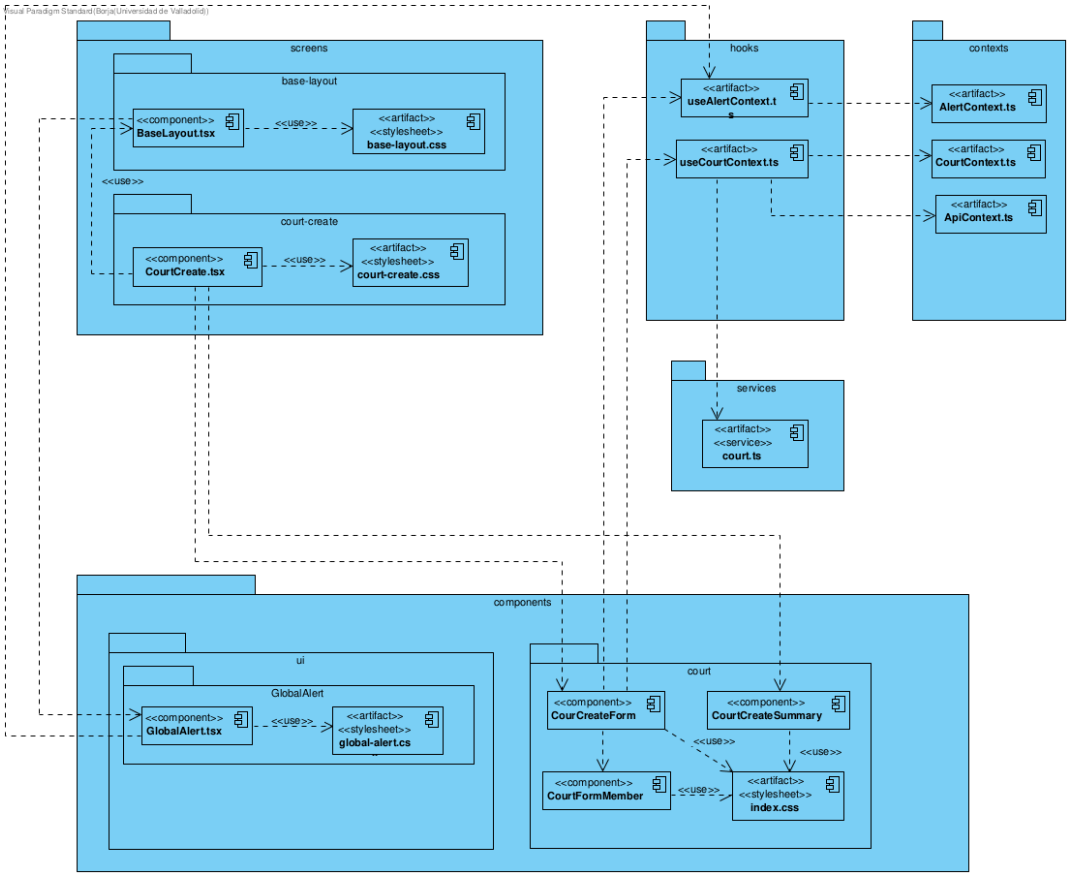

Figura 5.3: Diagrama de Arquitectura del frontend

5.1.4. Bibliotecas

Con el fin de paliar algunas de las carencias que presenta React, se hará uso de las siguientes bibliotecas externas:

- Axios: Axios es un cliente HTTP que nos permitirá la creación de *interceptors* para establecer headers necesarios a todas las peticiones a la API, así como cancelación de peticiones o serialización automática de objetos.
- React-Router: Una de las mayores carencias de React es el no disponer de un enrutador para navegar entre páginas de la aplicación a través de la dirección de ruta del navegador. Esta herramienta se encarga de ofrecer una solución a este problema.
- React-Query: *React Query* es un gestor asíncrono de estados. Esto nos permite un control más sencillo de los estados de tareas asíncronas como son las peticiones a servicios externos, su aplicación más general. También ofrece soluciones a problemas como el disparo simultáneo de diferentes peticiones idénticas y gestión de cachés de estas peticiones a nivel global de aplicación. Puede ser una alternativa a los Contextos cuando este estado depende de servicios externos o debe de ser actualizado frecuentemente.
- MUI: Si bien esta biblioteca no suple ninguna carencia de *React*, sí que permite el desarrollo más veloz ofreciendo componentes de interfaz que implementan el patrón de diseño de interfaz de usuario *Material Design* de Google.
- useForm: *useForm* es un *hook* que ofrece una forma sencilla, extensible y eficiente de validar formularios. Con esta herramienta podemos reducir la cantidad de código repetitivo y poco reusable propio de la validación de formularios.

5.2. Arquitectura Backend

A continuación, se describen las decisiones de diseño tomadas con respecto al servidor (*backend*).

5.2.1. API REST

El backend consistirá en una API RESTful. Una API REST consiste en un servicio que busca desacoplar el cliente-servidor, lo único que el cliente debe conocer acerca de la API es el punto de acceso al recurso. Otra característica vital es la carencia de estado o sesión, es decir, cada petición a una API REST debe contener toda la información necesaria para la resolución de dicha petición y no almacenar información entre una petición y otra. Es por otra parte recomendable, el uso de caches por parte del servidor o cliente y contar con una arquitectura por capas en el servidor, lo cuál definiremos a continuación.

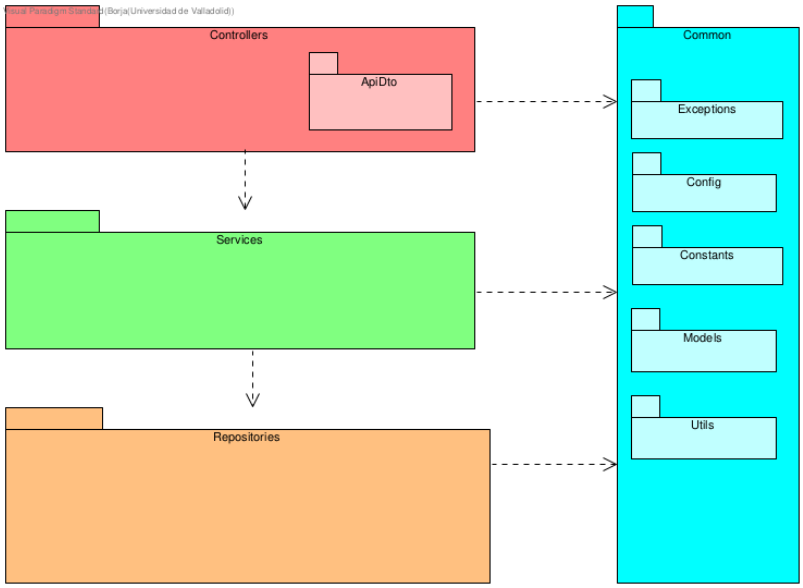

Figura 5.4: Arquitectura de capas de referencia

5.2.2. Arquitectura por capas

La arquitectura por capas [5] es un modelo arquitectónico genérico que permite adaptarse a las características del software en el que se aplica. En nuestro proyecto utilizaremos la siguiente arquitectura de tres capas, que se muestra en la Figura 5.4:

- Capa API: Consta de los controladores. Es la fachada o punto de entrada de la API para todo servicio, aplicación o humano que lo utilice. Se comunican con los servicios para dotarse de funcionalidad.
- Capa de servicios: Responsable de la lógica y, también, controladores de los casos de uso. Son invocados por los controladores u otros servicios que requieran de ellos. Cuentan con los repositorios para la obtención de datos externos a su lógica.
- Capa de repositorios: Los repositorios son los responsables de la comunicación con servicios externos bien sean estos la BD, API web externas o FTP, por nombrar algunos.
- Capa común: En esta capa se encuentran utilidades que pueden ser requeridas por cualquier otra capa. En ella nos encontramos principalmente las excepciones.

Una arquitectura por capas como la anterior nos proporciona las siguientes ventajas.

- Separación de responsabilidades. Cada capa tiene una función bien definida que proporciona una mayor modularidad de la aplicación, así como permitir la reusabilidad de código reduciendo la duplicidad y el tiempo de desarrollo de funcionalidades.

- Testabilidad. Cada capa puede ser probada en aislamiento mediante tests unitarios con *mocks* de otros módulos que sean necesarios para el test.
- Mantenimiento. La modularidad entre capas permite la sustitución, mejora o extensión de alguno de los módulos sin requerir de cambios al resto de componentes. Un ejemplo muy frecuente de esto se da al migrar de BD, y que solo requeriría de la sustitución de la capa de repositorios.

Si bien los beneficios superan en gran medida a los perjuicios de hacer uso de una arquitectura como esta, es justo comentar algunos de los principales problemas que pueden surgir de la implantación de esta arquitectura y que son: impactos al rendimiento, una mayor complejidad de la aplicación y de mayor tiempo de aprendizaje respecto a arquitecturas más simples.

Tomando el diagrama en la Figura 5.4 como base de nuestra arquitectura, la aplicación se ha estructurado tal y como se muestra en la Figura 5.5. Para facilitar la compresión de este diagrama, los paquetes pertenecientes al paquete *controller* se muestran vacíos, ya que todos estos comparten internamente la misma estructura, y un ejemplo de como estos se estructuran se muestra en la Figura 5.6 con el paquete *adjudication*. De forma análoga, la estructura de los paquetes *service* se presentan en la Figura 5.7.

5.3. Arquitectura del sistema

Para el despliegue y puesta en funcionamiento de los diferentes módulos de la aplicación se ha desarrollado el siguiente diagrama de despliegue. Todos los servidores necesarios para dicho despliegue son ofrecidos por la Escuela y el STIC.

5.4. Identificación y autenticación

En la identificación de la aplicación se utilizará el Inicio de Sesión Único (SSO, por sus siglás en inglés), basado en Servicio de Identificación Central (CAS, por sus siglas en inglés), de la Universidad de Valladolid. En conjunción con este se requiere de una forma de autenticación y autorización en el consumo de la API del servicio backend.

5.4.1. Servicio CAS

CAS [4] es un protocolo de autenticación de inicio de sesión único. Esto significa que el usuario es capaz de identificarse mediante un único ID a múltiples servicios independientes pero relacionados, por ejemplo porque estos pertenezcan a la misma organización

Este protocolo requiere de tres sistemas, un navegador web, la aplicación que realiza la petición de identificación y el servicio CAS. El funcionamiento consiste en que al acceder a

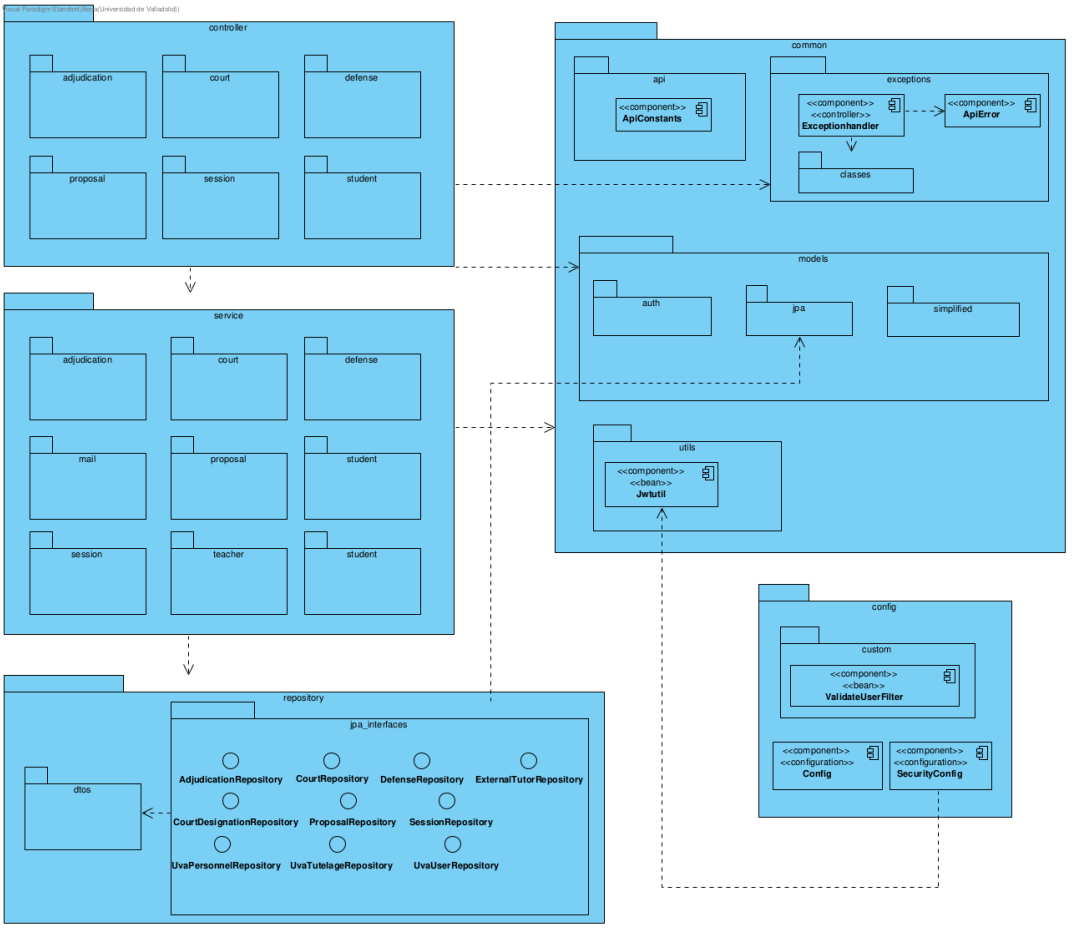

Figura 5.5: Arquitectura del backend.

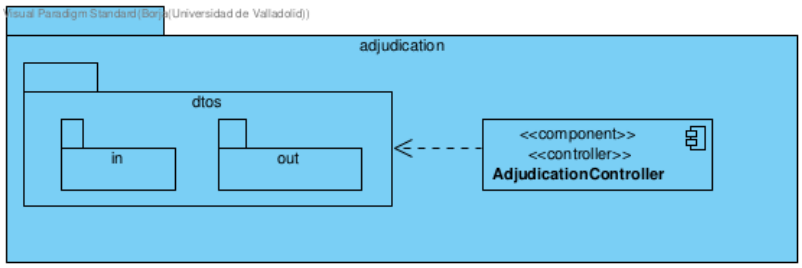

Figura 5.6: Subdiagrama de controlador.

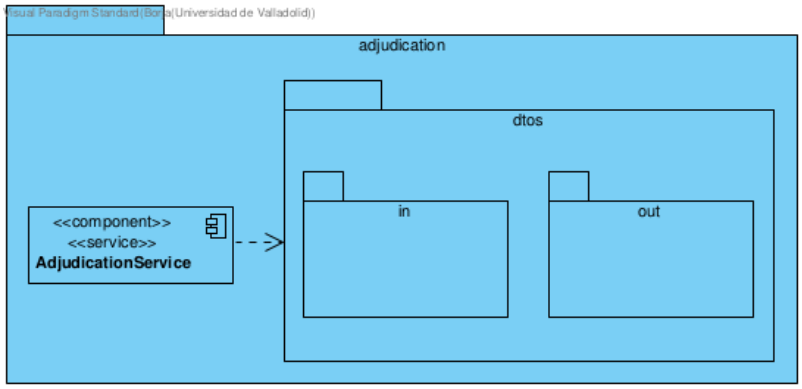

Figura 5.7: Subdiagrama de servicio.

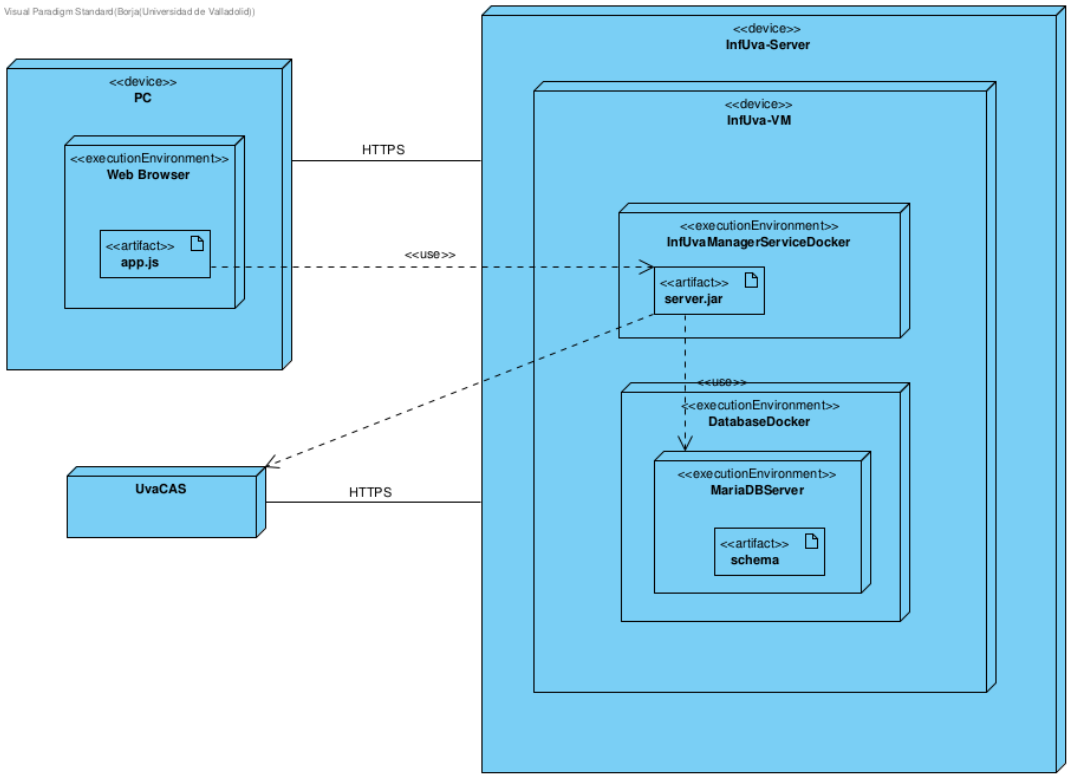

Figura 5.8: Diagrama de despliegue

una aplicación que requiera de autenticación, esta redirigirá a la página de identificación del CAS, que es quien valida los credenciales suministrados, en nuestro caso un usuario y una contraseña. En caso de una verificación exitosa el servidor CAS devuelve un ticket de servicio, el cual posteriormente la aplicación verificará con el servidor CAS y que este responderá con la información confidencial.

5.4.2. Token JWT

JSON Web Token (JWT) [8] es un estándar que define una forma compacta y autocontenida para la transmisión segura de información entre dos partes en forma de objeto JSON.

Existen dos formas de uso de estos tokens, tokens firmados y tokens encriptados. Estos primeros, los que vamos a usar, no esconden la información que contienen a terceros pero aseguran su integridad mediante firmas digitales generadas con diferentes algoritmos criptográficos. Los token encriptados por otro lado también protegen el contenido a agentes externos.

Estos token consisten en las siguientes partes:

- Header. La cabecera contiene generalmente dos campos, *typ* que indica el tipo del token, en nuestro caso JWT; y *alg*, el algoritmo utilizado para la firma. Nosotros usaremos HMAC SHA256, si bien también se pueden utilizar algoritmos más seguros de clave privada y pública como RSA.
- Payload. Este campo contiene la información que se transmite. El contenido de este puede ser diferente en base a la implementación, por un lado están las *claims* públicas y que son establecidas en RFC 7519[11]

  Además de estas se consideran claims privadas que contienen la información que se desea compartir.
- Signature. Es la parte del token que se encripta. Para ello se codifican a base 64 las dos partes anteriores del token, se juntan separándolas por un punto y se coloca detrás la clave de encriptación usada. Finalmente, se encripta todo con el algoritmo seleccionado.

En nuestras aplicaciones, el token JWT será generado en el *backend* tras la identificación del SSO y contendrá los datos del usuario identificado entre los que se comprenden su rol de acceso, su nombre y su dirección de correo electrónico. Este será enviado por el *frontend* en la cabecera HTTP *Authentication* para identificar al usuario que está realizando la petición.

5.4.3. Diseño del flujo de autenticación en la aplicación

Como se ha mencionado, el sistema hará uso del servicio de identificación de la Universidad de Valladolid. Esta identificación tendrá lugar en la aplicación web por lo que se necesita de

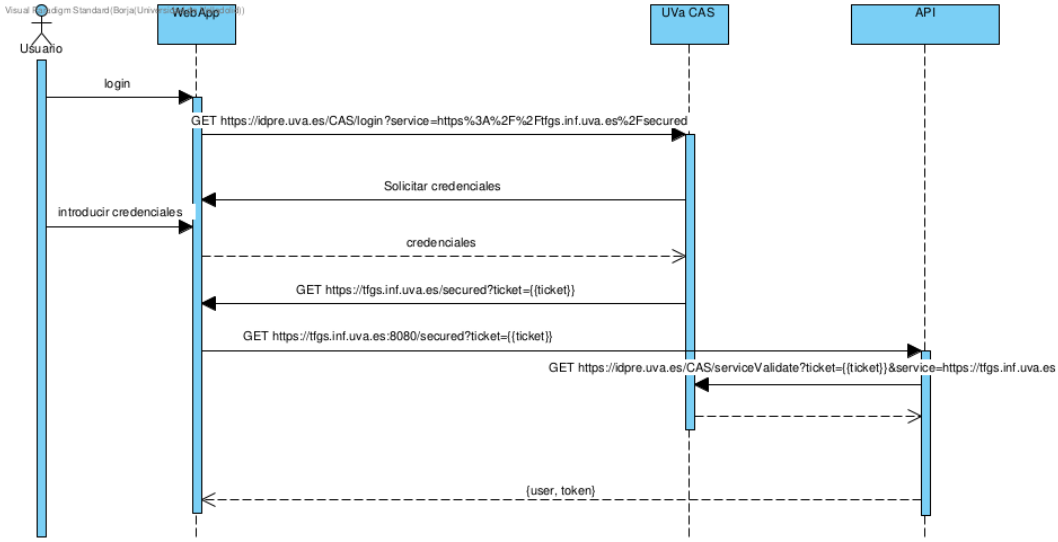

Figura 5.9: Flujo de identificación

una arquitectura que permita la identificación del usuario en el servicio backend. Para ello se ha desarrollado el siguiente flujo para la identificación, y que se detalla en la Figura 5.9.

1. El usuario inicia el proceso de identificación y se le redirige a la página de inicio de sesión de la Universidad de Valladolid. Esta devolverá en caso de éxito el ticket de servicio.

2. La aplicación web enviará este ticket al servicio backend que será el encargado de verificar la validez de este ticket con el servicio CAS y que devolverá la información correspondiente al usuario identificado. El servicio backend entonces, creará un token JWT con esta información del usuario y que será la respuesta a la aplicación web.

3. Una vez la aplicación web cuenta con el token este será guardado en el almacenamiento local del navegador y será insertado en la cabecera de todas las peticiones al servicio backend.

4. Una vez el token haya caducado o este sea borrado, el proceso de identificación será realizado desde el primer paso.

5.5. HTTPS

A raíz de la comunicación con CAS de la Universidad de Valladolid, tanto la aplicación web como el servidor deben ser desplegados bajo HTTPS.

HTTPS[17] es el protocolo de transferencia de hipertexto seguro, que es a su vez la versión segura de HTTP, que es el principal protocolo utilizado para enviar datos entre un navegador web y un sitio web. HTTPS es el mismo protocolo que HTTP utilizando el protocolo Transport Layer Security (TLS), o su predecesor Secure Sockets Layer (SSL). TLS permite que las conexiones realizadas por HTTPS estén cifradas, íntegras y la autenticación de extremos. De esta forma se protege la conexión de ataques frecuentes en la red como son ataque de intermediario y el *eavesdropping*.

Para la implementación de HTTPS se ha utilizado la autoridad de certificación (CA) *Let's Encrypt*[9] que expide certificados X.509 gratuitos para el cifrado TLS/SSL.

Capítulo 6

Implementación y pruebas

6.1. Implementación

En esta sección se detallarán algunas de las implementaciones de las decisiones de diseño establecidas en su correspondiente apartado.

6.1.1. Base de datos y consultas

Para la base de datos se ha hecho uso de Hibernate para la creación de la BD y mapeo de las entidades a sus correspondientes tablas. Esto nos permite acelerar el desarrollo de software al eliminar la necesidad de implementar mappers manualmente para la persistencia de objetos.

En la Figura 6.1 tenemos el diagrama de la base de datos final. Tras haber realizado varias iteraciones sobre las que se encontraron dificultades para la implementación de ciertos casos de uso así como para mantener la integridad de los datos hemos obtenido este diagrama. Las principales diferencias de este respecto del definido en 3.4 son la fusión de las entidades Propuesta, llamada *proposal* en este diagrama, con Adjudicación. Estas dos entidades son almacenadas en la misma tabla discriminando una de la otra por el campo *type*. Análogamente para la fusión de las entidades PersonalUVa y Estudiante, ahora representadas en la tabla *uva_ user*. Por último, en el diseño final de la BD no se ha tenido en cuenta a los colaboradores externos por ser estos únicamente necesarios en los TFGs ofertados por empresas en convenio con la universidad, funcionalidad que no se ha realizado en este trabajo.

Respecto a las consultas se ha utilizado un acercamiento híbrido entre SQL nativo y los métodos ofrecidos por JPA en su *JpaRepository*, con el objetivo de reducir la necesidad de realizar filtrados y ordenaciones en el código Java. Esto aprovecha mejor el ancho de banda de la BD y no recuperar entidades que serían de otra forma desestimadas en el código.

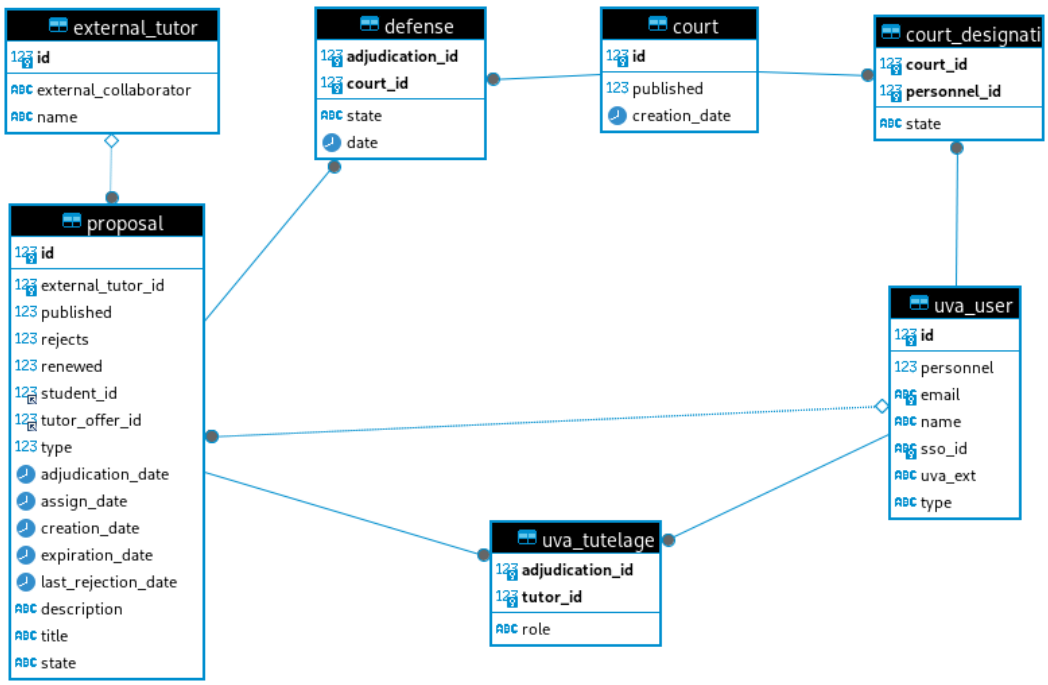

Figura 6.1: Diagrama de Diseño de la Base de Datos

6.1.2. Implementación de una pantalla en la aplicación web

A continuación nos retrotraemos a lo descrito en la Sección 5.1.3 para ampliar como esa arquitectura definida da forma a una pantalla y caso de uso en la aplicación para lo cuál nos apoyaremos en la Figura 5.3. Con razón de reducir el número de diagramas tomaremos como ejemplo la pantalla de creación de tribunales de la aplicación para este ejemplo. La estructura se traduce al resto de pantallas sin ningún cambio que impida entenderlo una vez explicado este.

En primer lugar nos situaremos en el componente central de la pantalla, *CourtCreate.tsx*, el cuál se monta sobre el componente *BaseLayout.tsx* que no es más que una plantilla que proporciona la cabecera y pie de la web y utilidades como el sistema de alertas. Volviendo a *CourtCreate.tsx*, este se compone de los componentes *CourtCreateForm.tsx* y *CourtCreateSummary.tsx* que a su vez cuentan con su propias hoja de estilo *.css*. Hasta ahora hemos repasado como una pantalla, que es un componente, se forma a partir de otros componentes.

Para que estos componentes cuenten con funcionalidad utilizamos los *hooks*. En nuestro ejemplo contamos con dos *hooks*. Por un lado tenemos *useCourtContext.ts* que contiene la funcionalidad relacionada con los tribunales como la creación, borrado u obtención de datos del servicio backend. Este *hook* cuenta con su contexto pareja *CourtContext.ts* que almacena y provee el estado a la aplicación. Como se puede intuir, existe una pareja *hook*-contexto análoga a esta para las diferentes entidades del proyecto como son defensa, propuesta y adjudicación. Por otro lado, también contamos con *useAlertContext.ts* que permite al componente *CourtCreateForm.tsx* generar una alerta para notificar al usuario de una acción y que será consumida en otro punto de la aplicación por el componente *GlobalAlert.tsx* que es el encargado de su renderizado. Por último, contamos con el contexto *ApiContext.tsx* que contiene la información para realizar la conexión con el backend. Esta información es requerida por los *hooks* al utilizar los servicios, como *court.ts*, para comunicarse con el servicio.

6.1.3. Casos de uso en el servicio backend

Tal y como se ha descrito en la Sección 5.2.2 Diseño, se ha seguido una arquitectura por capas en el servicio backend. En esta sección ejemplificaremos esta arquitectura con el caso de uso de Creación de una adjudicación.

En este caso de uso debemos realizar, a alto nivel, las tareas de obtener la propuesta la cuál se va a adjudicar a un alumno y recuperar tanto el tutor como el alumno. Crear la nueva entidad, relacionarla a estas y persistir los cambios, además se deberá notificar a la Secretaría del centro de esta operación.

Para ello, y como se muestra en la Figura 6.2, el Controlador (*AdjudicationController*), la frontera de la API, recibe procesa la petición y la empaqueta en el DTO *Request* (*CreateAdjudicationRequest*) correspondiente al caso de uso. Este DTO es enviado al Servicio (*AdjudicationService*) que contiene la lógica de negocio para este caso de uso.Para completar la operación este cuenta con otros servicios y el Repositorio correspondiente (*AdjudicationRepository*). Cabe destacar que ningún servicio se comunica directamente con repositorios

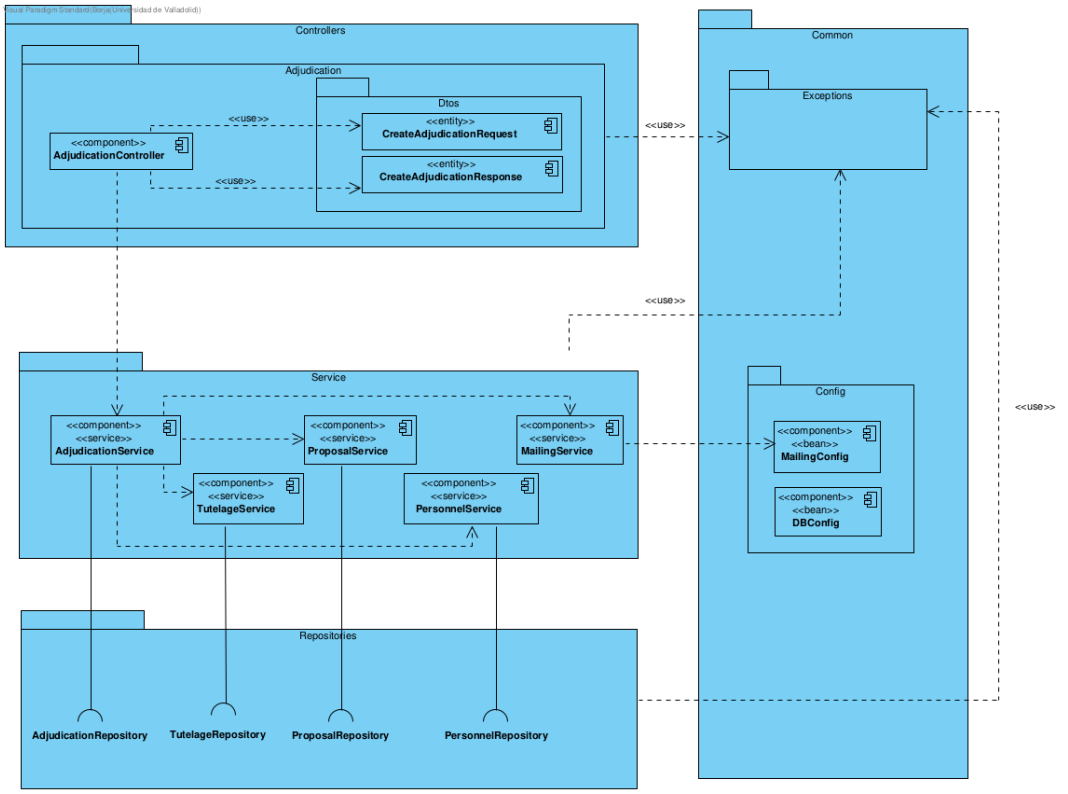

Figura 6.2: Detalle de la Arquitectura de un Caso de Uso en el backend

ajenos a él o, dicho de otra manera, el servicio de adjudicaciones sólo se comunica con el repositorio de adjudicaciones. A lo largo de este flujo, cualquier componente puede hacer uso de componentes presentes en la capa de comunes como son las excepciones o configuraciones de la aplicación según las requieran. Finalmente, es el Controlador de nuevo que con lo recibido desde el servicio genera un nuevo DTO *Response* (*CreateAdjudicationResponse*) y que es la respuesta de la API.

6.2. Pruebas

En el desarrollo de software existen diferentes métodos para probar la calidad y funcionamiento del sistema o de sus módulos. Estos se pueden dividir en dos grandes grupos que son pruebas de caja negra y de caja blanca. Las pruebas de este primer grupo ignoran los detalles internos del sistema y únicamente se centran en el correcto funcionamiento del sistema con respecto a los requisitos definidos y lo tratan como lo haría un usuario final. En el segundo grupo de pruebas tenemos aquellas que tienen en cuenta la implementación del sistema y de sus módulos, y se centran en encontrar el correcto funcionamiento del código bajo diferentes entradas de datos.

CP01	Identificación.
Descripción	Identificación del usuario mediante el CAS de la UVa.
Entrada	El usuario inicia la identificación, se le redirigirá a la pantalla de inicio de sesión del CAS y se recuperará su rol y detalles de usuario.
Resultado esperado	El usuario está identificado en el sistema.
Resultado obtenido	Error.
Corrección	Se arregló la forma en que se recuperaban los datos del usuario obtenidos por el CAS en la solicitud de verificación.

Tabla 6.1: CP08. Creación de tribunales sin publicar

En nuestro proyecto y debido a la naturaleza de este y de no disponer de tiempo para realizar un trabajo de pruebas completo, nos hemos centrado en la realización de pruebas de sistema, que son pruebas de caja negra. Las pruebas de sistema tratan el sistema en conjunto, desde un punto de vista de usuario validando si el resultado obtenido del sistema cumple con los requisitos definidos.

Estas pruebas se han realizado tanto para el servicio backend como para la aplicación web siguiendo el flujo de pruebas definido a continuación:

- Tras la finalización de un caso de uso se han realizado las pruebas correspondientes a este sobre el servicio backend con la herramienta Postman y comparando la respuesta obtenida con lo que se esperaba en cada caso.
- Posteriormente, se realizaban las pruebas correspondientes a la aplicación web y que comprendían a la navegación hasta el nuevo caso de uso, y la correcta visualización de la información con datos mockeados.
- Una vez probado ambos sistemas por separado, se han probado los casos de uso desde la aplicación web conectada con el servicio backend.

Siguiendo estos pasos hemos realizado pruebas de sistema a ambos sub-sistemas (aplicación web y servicio) y al sistema completo comprendido por ambos. A continuación se detallan las diferentes pruebas llevadas a cabo. Para las pruebas se inicializa la BD con el script que se proporciona en [TODO].

6.2.1. Pruebas de Casos de uso

6.2.2. Pruebas de navegación de la aplicación web

También se han realizado pruebas a la navegación de la aplicación, según roles de acceso a esta. Las pruebas se realizan accediendo a la aplicación y comprobando que según el rol del usuario ingresado pueda acceder a las pantallas permitidas según se muestra en la Figura 6.12

CP02	Creación de una propuesta
Descripción	Creación de una nueva propuesta de TFG.
Entrada	Siendo un usuario con rol de profesor acceder al listado de propuestas, acceder a la pantalla de creación y cumplimentar el formulario
Resultado esperado	Se muestra un mensaje de éxito y la nueva propuesta aparece en el listado de propuestas
Resultado obtenido	Correcto

Tabla 6.2: CP02. Creación de propuesta

CP03	Asignación de una propuesta
Descripción	Asignación de una propuesta previamente creada.
Entrada	Siendo un usuario con rol de coordinador acceder a los detalles de una propuesta pendiente, seleccionar asignar.
Resultado esperado	Se muestra un mensaje de éxito y la nueva propuesta aparece en el listado de adjudicaciones.
Resultado obtenido	Correcto

Tabla 6.3: CP03. Asignación de propuesta

CP04	Adjudicación de una propuesta
Descripción	Asignación con publicación de una propuesta previamente creada.
Entrada	Siendo un usuario con rol de coordinador acceder a los detalles de una propuesta pendiente, seleccionar adjudicar con publicación.
Resultado esperado	Se muestra un mensaje de éxito y la nueva adjudicación aparece en el listado de adjudicaciones publicada y con la fecha correcta de expiración.
Resultado obtenido	Correcto

Tabla 6.4: CP04. Adjudicación de propuesta

CP05	Adjudicación de una propuesta
Descripción	Asignación sin publicación de una propuesta previamente creada.
Entrada	Siendo un usuario con rol de coordinador acceder a los detalles de una propuesta pendiente, seleccionar adjudicar sin publicación.
Resultado esperado	Se muestra un mensaje de éxito y la nueva adjudicación aparece en el listado de adjudicaciones publicada y con la fecha correcta de expiración.
Resultado obtenido	Correcto

Tabla 6.5: CP05. Adjudicación de propuesta

CP06	Edición de adjudicación
Descripción	Edición del título, descripción, estado, tutores de una adjudicación.
Entrada	Siendo un usuario con rol de coordinador acceder a los detalles de una adjudicación, editar los campos permitidos y aplicar cambios.
Resultado esperado	Se muestra un mensaje de éxito y al refrescar la web los datos han persistido.
Resultado obtenido	Correcto

Tabla 6.6: CP06. Edición de adjudicación

CP07	Publicar una adjudicación
Descripción	Publicación de una adjudicación ya creada.
Entrada	Siendo un usuario con rol de coordinador acceder a los detalles de una adjudicación, cambiar el estado del deslizable de publicación y aplicar cambios.
Resultado esperado	Se muestra un mensaje de éxito y al refrescar la web los datos han persistido.
Resultado obtenido	Correcto.

Tabla 6.7: CP07. Publicar una adjudicación

CP08	Creación de tribunales sin publicar.
Descripción	Creación de múltiples tribunales sin publicarlos en momento de la creación.
Entrada	Siendo un usuario con rol de coordinador acceder a la pantalla de tribunales y acceder a la pantalla de creación. En esta pantalla cumplimentar el formulario y hacer clic en siguiente, una vez se ha creado todos los tribunales hacer clic en terminar
Resultado esperado	Se muestra un mensaje de éxito y los tribunales creados aparecen en el listado de tribunales con estado no publicado.
Resultado obtenido	Correcto.

Tabla 6.8: CP08. Creación de tribunales sin publicar

CP09	Creación de tribunales con publicación.
Descripción	Creación de múltiples tribunales publicándolos en el momento de la creación.
Entrada	Siendo un usuario con rol de coordinador acceder a la pantalla de tribunales y acceder a la pantalla de creación. En esta pantalla cumplimentar el formulario y hacer clic en siguiente, una vez se ha creado todos los tribunales seleccionar publicar y hacer clic en terminar.
Resultado esperado	Se muestra un mensaje de éxito y los tribunales creados aparecen en el listado de tribunales con estado publicados.
Resultado obtenido	Correcto.

Tabla 6.9: CP09. Creación de tribunales con publicación

CP10	Edición de los detalles de un tribunal.
Descripción	Edición de los detalles de un tribunal en el momento de la creación.
Entrada	Siendo un usuario con rol de coordinador acceder a la pantalla de tribunales y acceder a la pantalla de creación. En esta pantalla y tras haber creado un tribunal hacer clic en el botón de editar y modificar los detalles del tribunal.
Resultado esperado	El tribunal vuelve a aparecer en el listado de tribunales pendientes de crear con los nuevos datos.
Resultado obtenido	Correcto.

Tabla 6.10: CP10. Edición de los detalles de un tribunal

CP11	Borrado de un tribunal durante la creación.
Descripción	Borrado de un tribunal pendiente de envío durante la creación de múltiples tribunales.
Entrada	Siendo un usuario con rol de coordinador acceder a la pantalla de tribunales y acceder a la pantalla de creación. En esta pantalla y tras haber creado un tribunal hacer clic en el botón de eliminar un tribunal.
Resultado esperado	El tribunal desaparece del listado de tribunales pendientes de crear.
Resultado obtenido	Correcto.

Tabla 6.11: CP11. Borrado de un tribunal durante la creación

Role	Accesos	Resultado obtenido
Coordinador	Detalles de adjudicación	Correcto
	Detalles de tribunal	Correcto
	Todos los permisos del rol Profesor	Correcto
Profesor	Creación de propuesta	Correcto
	Todos los permisos del rol Anónimo	Correcto
Anónimo	Todos los listados	Correcto

Tabla 6.12: Pruebas de navegación

6.2.3. Pruebas con usuarios finales

Las últimas pruebas realizadas sobre las aplicaciones se realizaron con la colaboración de usuario finales como son el Coordinador de Comité de Título y la tutora de este TFG, por ser estos capaces de iniciar sesión a través del SSO de la universidad.

Esto causó dificultades a la hora de identificar problemas debido a que el alumno no contaba con un entorno en las mismas condiciones que con el que los usuarios interactuaban. También, la necesidad de desplegar las aplicaciones tras cada cambio en la máquina autorizada por el SSO ralentizaba el proceso de pruebas además de ocasionar errores en estos despliegues debido a la necesidad de adaptar los cambios a un entorno al que el alumno no tenía acceso.

Capítulo 7

Seguimiento del proyecto

Como se ha definido en su sección 2.4.4, el proyecto se desarrolla con una primera fase a la que nos referiremos como Sprint 0 y que comprende la elicitación de requisito, análisis y modelado del problema.

7.1. Fase inicial

Esta fase está marcado por un modelo de proceso en cascada. Durante este sprint se han tenido reuniones, por parte tanto del estudiante como de la tutora, con el Coordinador de Título. La larga duración de este está marcado por las múltiples revisiones de los modelos y dificultades para establecer las reuniones mencionadas.

7.2. Sprint 1

El sprint en el que da el inicio al desarrollo e implementación. Durante este sprint se ha creado la BD. También se han completado las implementaciones de ambos casos de uso en el servicio backend y se ha revisado la arquitectura de la BD, con algunos problemas de diseño identificados al momento de implementar los casos de uso.
Se ha completado el flujo de éxito del caso de uso (01) de la aplicación web. Debido a los problemas identificados al no contar con acceso al SSO de la UVa hasta la mitad de este sprint se ha desestimado completar la implementación de los casos de uso en la aplicación web y se ha dedicado tiempo al estudio y experimentación del SSO de pruebas provisto para este proyecto.

Sprint 0	
Fecha de inicio	14 septiembre 2023
Fecha de fin	7 noviembre 2023
Metas	- 01: Elicitación y análisis de requisitos. - 02: Planificación del proyecto. - 03: Modelado de procesos: diagrama de los tres procesos definidos. - 04: Modelado de dominio. - 05: Casos de uso: definición y descripción. - 06: Arquitecturas de aplicaciones - 07: Diseño de la BD
Metas alcanzadas	Todas
Tiempo invertido	50 horas

Tabla 7.1: Sprint 0

Sprint 1	
Fecha de inicio	8 noviembre 2023
Fecha de fin	21 noviembre 2023
Metas	- 01: CU01: Solicitud de adjudicación TFG. 3 Puntos Historia - 02: CU02: Adjudicación TFG. 3 Puntos Historia - 03: Creación de BD - 04: Documentación acerca del SSO UVa
Metas alcanzadas	03
Metas parciales	01, 02, 04
Tiempo invertido	25 horas

Tabla 7.2: Sprint 1

Sprint 2	
Fecha de inicio	22 noviembre 2023
Fecha de fin	05 diciembre 2023
Metas	- 01: Diseño e implementación de interfaz de la aplicación web. 5 PH - 02: Documentación del sprint. 1 PH
Metas alcanzadas	Todas
Tiempo invertido	20 horas

Tabla 7.3: Sprint 2

Sprint 3	
Fecha de inicio	06 diciembre 2023
Fecha de fin	19 diciembre 2023
Metas iniciales	- 01: Implementación de los casos de uso desarrollados en la nueva interfaz. 3 PH - 02: Diseño de autenticación de la aplicación web y servidor backend. 2 PH - 03: Documentación del sprint. 1 PH
Metas alcanzadas	02 con extras, 03
Metas parciales	01
Tiempo invertido	20 horas

Tabla 7.4: Sprint 3

7.3. Sprint 2

En este sprint se decidió dejar paradas las HU no finalizadas completamente en el anterior al encontrarnos en ese con dificultades asociadas a la falta de un diseño de la interfaz previo y falta de documentación acerca del servicio de identificación de la UVa. Respecto a estos problemas para obtener acceso al servicio CAS de la universidad debemos solicitar la autorización para obtener el acceso a este servicio, así como de disponer de un servidor con un puerto de acceso abierto para la comunicación del servicio con nuestra aplicación.

Si bien no era una meta, durante este sprint se han seguido estudiando las diferentes formas de implementación de la autenticación. En un principio, y siguiendo con las instrucciones aportadas por el Servicio de Tecnologías de la Información y las Comunicaciones (STIC) de la UVa se planteó implementar esta autenticación en el servidor backend pero tras analizarlo y tener en cuenta que en nuestro backend implementa una API RESTful esto no es posible sin comprometer dicha arquitectura debido a que, como se explica en 5, estas APIs no almacenan estados. De este modo el flujo de autenticación y almacenamiento de la sesión deberá llevarse a cabo en la aplicación web.

Para finalizar, concluimos con que en este sprint se han llevado a cabo con éxito todas las metas planeadas.

7.4. Sprint 3

En este sprint se dedicaron menos horas de las que se planearon inicialmente, influenciadas por factores externos al alcance de este trabajo. A pesar de ello se han alcanzado los objetivos en su mayoría quedando como pendiente la mejora de la visualización de un caso de uso en la aplicación web.

En la planificación del sprint se decidió dar comienzo con el diseño de la autenticación entre el front y el back, surgido tras la decisión de mover la identificación con el servicio

Sprint 4	
Fecha de inicio	20 diciembre 2023
Fecha de fin	2 enero 2024
Metas iniciales	- 01: CU04: Modificación de adjudicación. 3 PH - 02: CU05: Renovación de adjudicación. Solo backend. 2 PH - 03: Documentación del sprint. 2 PH
Metas alcanzadas	Todas
Tiempo invertido	20 horas

Tabla 7.5: Sprint 4

de la Universidad de Valladolid a la aplicación web. Este objetivo se vió ampliado durante el desarrollo de este sprint, al decidir incluir no solo el diseño de este si no también la implementación en ambas aplicaciones de cara a evitar re-trabajos en futuros sprints.

De esta forma y a pesar de haberle dedicado menos trabajo del planeado inicial, se ha trabajado rápido y concluyendo con la mayoría de las metas iniciales, metas añadidas a mitad de sprint y sin problemas en el desarrollo.

7.5. Sprint 4

En este sprint se finalizó la tarea que quedó pendiente de cerrar en el anterior sprint y se completaron todas las tareas planeadas para este.

Durante el desarrollo de este sprint se realizaron modificaciones profundas al diseño de la BD, fusionando en una sola tabla Propuesta y Adjudicación con la motivación de reducir el número de accesos a la BD y facilitar la gestión de la integridad de los datos entre ambas entidades. En este momento, se deciden que ambas se almacenaran en la misma tabla diferenciándolas por el estado en el que esta se encuentra. De forma colateral, también se ha modificado la relación entre una Adjudicación y su sucesora cuando esta se renovara, para lo cual se ha sustituido la relación por una nueva columna en la tabla.

En el lado de la aplicación web no ha habido cambios significativos y se ha incluido el nuevo caso de uso sin contratiempos.

Para concluir, cerramos el sprint satisfactoriamente tras haber cerrado todas las tareas planificadas y pendientes, y habiendo mantenido el ritmo de trabajo del anterior sprint.

7.6. Sprint 5

En este sprint se ha dedicado tiempo a la definición y realización de pruebas más completas a los casos de uso realizados hasta el momento. También y aunque no estaba planificado al

Sprint 5	
Fecha de inicio	3 enero 2024
Fecha de fin	16 enero 2024
Metas iniciales	- 01: Pruebas. 3 PH - 02: Documentación del sprint. 2 PH
Metas alcanzadas	Todas
Tiempo invertido	20 horas

Tabla 7.6: Sprint 5

Sprint 6	
Fecha de inicio	17 enero 2024
Fecha de fin	30 enero 2024
Metas iniciales	- 01: Creación de tribunal. 3 PH - 02: Creación de defensa 3 PH - 03: Documentación del sprint. 2 PH
Metas alcanzadas	01
Metas parciales	02 y 03
Tiempo invertido	15 horas

Tabla 7.7: Sprint 6

inicio de este, se han reescrito, parcial o totalmente, algunos de estos casos de uso tras encontrarnos con errores que no se habían detectado anteriormente.

Se trata por lo tanto un sprint de afianzar lo que hay construido de cara a próximos sprints que se centrarán en abordar los casos de uso que restan del proyecto.

7.7. Sprint 6

En este Sprint se atacaron dos nuevos casos de uso. La creación de tribunales y la extensión de este caso de uso sobre el caso de uso de la creación de defensa. El desarrollo de estos se desarrolló sin contratiempos. Durante este Sprint se introdujo una nueva forma de manejar formularios (ver apartado 5.1.4) y que se aplicó para formularios creados en anteriores (creación de propuesta y modificación de adjudicación).

Por falta de tiempo en este sprint se han quedado pendientes algunas tareas de documentación y que se abordarán en el siguiente sprint.

Sprint 7	
Fecha de inicio	31 enero 2024
Fecha de fin	13 febrero 2024
Metas iniciales	- 01: Modificación defensa. 3 PH - 02: Modificación tribunal. 3 PH - 03: Publicación defensa. 2 PH - 04: Publicación de tribunal. 2 PH
Metas alcanzadas	Todas
Tiempo invertido	30 horas

Tabla 7.8: Sprint 7

Fase final	
Fecha de inicio	14 febrero 2024
Fecha de fin	13 mayo 2024
Tareas	**Tiempo invertido**
Despliegue *frontend*	13 horas
Despliegue *backend*	12 horas
Configuración HTTPS	17 horas
Documentación	35 horas
Pruebas	47 horas
Configuración CAS	25 horas
Presentación y defensa	10 horas
Tiempo total	159 horas

Tabla 7.9: Fase final

7.8. Sprint 7

Para este sprint se realizaron el restante de casos de uso que quedaban por finalizar, solo en su parte de desarrollo y las pruebas y documentación relacionadas a este se planificarían al siguiente sprint.

También destacar que se planificaron un mayor número de puntos de historia si los comparamos al resto de sprints. Esta decisión viene influenciada por la experiencia obtenida durante el desarrollo del proyecto, aumentando la *velocity*, y de disponer de más tiempo para el TFG en estas semanas.

Todas las tareas planificadas para este sprint se completaron en su totalidad.

7.9. Fase final

A partir de este momento en el desarrollo se desestimó el uso de sprints debido a bloqueos que impedían una entrega constante de valor en el tiempo establecido y razonable de un sprint.

Estos bloqueos fueron la solicitud e implementación del servicio CAS de la Universidad de Valladolid y el despliegue bajo HTTPS de ambas aplicaciones, así como las pruebas correspondientes a estas tareas.

Respecto al primero de estos, se intentó evitar este bloqueo mediante la solicitud prematura del servicio. Aún así y debido a la falta de comunicación por parte del STIC con nosotros este proceso se alargó semanas hasta que puedo ser resuelto con la colaboración del Director de esta Escuela. Tras, finalmente, obtener acceso al servicio CAS se detectó un problema en la configuración de red de la máquina virtual ofrecida por los Técnicos de la Escuela y que tomó un par de días para su resolución. Posteriormente nos encontramos con que ambos servicios (aplicación web y servidor *backend*) debían ser desplegados bajo HTTPS y que llevó otra semana en resolverse debido a la poca experiencia con la configuración de despliegues de aplicaciones, este proceso se detalla en su sección correspondiente 5.5.

Como trabajo paralelo a estos desarrollos, se llevaron a cabo el grueso de las pruebas de las aplicaciones, así como la documentación del proyecto relativa a estas tareas, finalización de la memoria y preparación de la presentación. Todas estas tareas y las horas dedicadas a cada una de ellas se detalla en la Tabla 7.9.

7.10. Resumen de la ejecución del proyecto

Para finalizar vamos a realizar una comparación entre lo analizado en la Sección 2.5 y la realidad tras la ejecución del proyecto. Analizaremos la calendarización, la revisión del presupuesto y los riegos manifestados en el proyecto.

7.10.1. Calendarización

Tal y como se mostraba en la Tabla 2.9, el proyecto consistiría de ocho sprints con uno extra dedicado al cierre del proyecto.

Un resumen de la calendarización final del proyecto se muestra en la Figura 7.10. El plan se llevó a cabo como planeado hasta el sprint 7, momento en el que se decidió prescindir del marco de trabajo en sprints ya que el valor generado en la duración de un sprint no era suficiente. Esto fue causado por bloqueos que se tratarán a continuación en la Sección 7.10.2.

La duración del proyecto, por lo tanto, se alargó en 3 meses con respecto a la duración planeada y con ello el número de horas dedicadas al proyecto que ascendieron a 358 horas con respecto a las 300 horas que ocupa esta asignatura.

	Inicio	Fin	Comentarios	Horas
Fase inicial	14/09/2023	07/11/2023		50
Reunión Semanal	Se realizan cada miércoles.		Se realizaron reuniones cada semana en las que se planificaba el inicio del proyecto, recogían requisitos y alguna reunión con el Coordinador de Grado, siguiendo la forma de trabajo en cascada.	
Sprint 1	08/11/2023	21/11/2023		25
Sprint 2	22/11/2023	05/12/2023		20
Sprint 3	06/12/2023	19/12/2023		20
Sprint 4	20/12/2023	02/01/2024		20
Sprint 5	03/01/2024	16/01/2024		20
Sprint 6	17/01/2024	30/01/2024		15
Sprint 7	31/01/2024	13/02/2024		30
Fase final	14/02/2024	13/05/2024	Durante estos meses de trabajo se realizaron reuniones con una frecuencia irregular debido a los bloqueos encontrados	159
			Horas totales	359

Tabla 7.10: Calendarización final del proyecto

7.10.2. Gestión de Riesgos

Para esta sección recordamos el contenido de la Figura 2.3 que definía los riesgos identificados para el desarrollo de este proyecto.

Con respecto a los riesgos descritos en la planificación no tuvieron lugar *Comunicación con la tutora*, *Problemas con entorno* y *Cambios en los requisitos*. El primero de estos se evitó planeando con la tutora las reuniones y previendo aquellas semanas o días en los que no se fuese posible reunirse, adaptando la comunicación entre ambos en consecuencia. Con respecto al segundo, se estableció una arquitectura y selección de tecnología robusta que no fue necesario modificar en el desarrollo del proyecto. Y para el último, se obtuvieron los requisitos con claridad al inicio del proyecto lo que permitió un desarrollo sin bloqueos ni retrabajos.

No todo fue acorde al plan y como ejemplos contamos con los riesgos de *Estimación de tareas* y *Trabajo externo* que se tuvieron en cuenta y finalmente se manifestaron durante el proyecto. La manifestación del primero se dio en los primeros sprints, como podía preverse, debido a la inexperiencia principalmente. Acerca del segundo, este supuso una bajada del rendimiento durante algunas semanas del proyecto en las que la carga en el trabajo externo eran superiores, retroalimentando al riesgo anterior aumentando la probabilidad de su ocurrencia.

Fuera de los riesgos identificados en la tabla anteriormente referenciada, se han manifestado los siguientes problemas:

- Adaptaciones en las aplicaciones: Se refiere a la necesidad de adaptar las aplicaciones para funcionar con el CAS de la Universidad de la Valladolid bajo el protocolo HTTPS.

Esto en un primer momento no estaba contemplado y tuvo lugar durante el desarrollo de la identificación de la aplicación y contactar con el STIC, que nos informó de este requerimiento debido a la solicitud de los Técnicos de la Escuela.

- Comunicación con terceros: En este vamos a agrupar los bloqueos que tuvieron lugar debido a la comunicación y a la espera de soluciones a problemas con personas u organismos externos a la tutora o al estudiante como fueron, principalmente, los Técnicos y el STIC. Con los primeros se tuvieron que gestionar problemas con la máquina virtual, los entornos de despliegue, accesos a estos que ocasionaron bloqueos hasta solucionar todo lo necesario; y con los segundos, se dieron problemas de comunicación para la solución de dificultades con el SSO de la UVa, basado CAS.

7.10.3. Coste

A partir del presupuesto estimado en la Sección 2.5 vamos a analizar a continuación la variación con los costes finales que habría supuesto este proyecto de tratarse de un proyecto real. Para ello utilizaremos las mismas fórmulas usadas en dicha sección corrigiendo los valores que hayan cambiado.

Los cambios a realizar son los de aquellos valores que dependan del tiempo de desarrollo como son las horas de trabajo humano y el tiempo utilizado en la amortización de las herramientas hardware utilizadas en el proyecto. Las fórmulas con lo nuevos datos y resultados se encuentran en las Ecuaciones 7.1 y 7.2.

$$CP = \frac{1800\frac{€}{mes} * 0{,}425 * 359 * 1{,}3}{68\frac{h}{mes}} = 5250{,}38\,€ \tag{7.1}$$

$$CH = \frac{800\,€ * 0{,}26\frac{\text{amort}}{\text{año}} * 7\text{meses}}{12\frac{\text{meses}}{\text{año}}} = 121{,}33\,€ \tag{7.2}$$

Concepto	Importe (€)
Personal	5 250,38
Software	0
Hardware	121,33
Total	5 371,71

Tabla 7.11: Presupuesto real

Comparando la Tabla 2.10 y la Tabla 7.11 nos encontramos con un incremento de los costes con respecto a lo presupuestado de 4463.94 €, o lo que supone un 120.33 % del presupuesto original.

Con fin aclarativo, durante el desarrollo de este proyecto no ha coste de personal alguno y los cálculos realizados anteriormente son con fin de simulación.

Capítulo 8

Conclusiones

A lo largo de este documento se han comentado las motivaciones, necesidades, las ideas de diseño y proceso de desarrollo, así como los problemas encontrados a lo largo de este. Para finalizar, en este capítulo se repasará brevemente el resultado final de este trabajo.

Con respecto a las motivaciones que llevaron al planteamiento de este trabajo, se ha obtenido un resultado que sirve como buen punto a partir del cuál seguir trabajando para conseguir la plataforma de gestión de TFGs de la Escuela. El plan inicial se ha visto a lo largo del desarrollo que superaba en envergadura a lo que un único TFG podría abarcar y aún así se han conseguido muchos éxitos.

Al inicio del trabajo se enumeraban las motivaciones que me llevaban a elegir un tema como este para realizar mi TFG. El resultado es satisfactorio, me he encontrado con problemas a lo largo del desarrollo y diseño que he resuelto con habilidad y decisión. También ha sido un trabajo que se ha alargado en el tiempo más de lo que se planeó en un primer momento y lo he notado en mi motivación a lo largo de este, llegando a una fase final del proyecto agotado. No sirva esto para minusvalorar el trabajo realizado pues realmente creo que el resultado es satisfactorio.

8.1. Líneas de trabajo futuras

Dado que este trabajo se tiene planeado implementar para la gestión de TFGs de Escuela requerirá de un trabajado de mantenimiento extendido en el tiempo como es natural en todo sistema informático. Además de esto, este puede ampliarse en varios sentidos y los cuáles pueden dar oportunidad nuevos TFGs que construyan a partir de la base de este trabajo.

- Optimización: Sería interesante la implementación de mejoras en las peticiones al servicio backend, como la paginación y filtrado en las peticiones de listados, que supondrían una gran mejora con volúmenes de datos grandes.

- Usabilidad: Dado el tiempo que abarca un TFG no se ha podido tener en cuenta todas las recomendaciones de usabilidad. Un estudio de esta y sus mejoras podría abrir nuevas oportunidades de trabajo.
- Dispositivos portátiles: Si bien esta aplicación se ha conceptualizado como una aplicación web usada principalmente en escritorio podría resultar interesante, sobre todo para los alumnos, que la app se comportara mejor en este tipo de dispositivos. Esto puede abrir la posibilidad de portar la app a SSR (*Server-side Rendering*) y adaptaciones de la app a pantallas pequeñas y verticales.
- DevOps: Implementar un ciclo de vida DevOps aumentaría en gran medida la velocidad de testeo y automatizaría el despliegue de las aplicaciones.
- Mejora del SSO: Algunas mejoras relacionadas con el SSO incluyen facilitar las pruebas por medio de posibilitar añadir a usuarios de pruebas individualmente y de esta forma agilizar el proceso de pruebas al no requerir de los usuarios finales para llevarlas a cabo.

Bibliografía

[1] Agencia Tributaria. tabla de amortización simplificada. https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/folleto-actividades-economicas/3-impuesto-sobre-renta-personas-fisicas/3_5-estimacion-directa-simplificada/3_5_4-tabla-amortizacion-simplificada.html. Accessed: 2023-8-16.

[2] Angular. https://angular.io/docs. Accessed: 2023-8-15.

[3] Arquitectura MVVM. https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm. Accessed: 2023-9-1.

[4] CAS - home. https://apereo.github.io/cas/7.0.x/index.html. Accessed: 2024-3-24.

[5] From Request to Database: Understanding the Three-Layer Architecture in API Development. https://konstantinmb.medium.com/from-request-to-database-understanding-the-three-layer-architecture-in-api-development-1c44c973c7af. Accessed: 2024-03-21.

[6] Glassdoor. sueldo programador junior en españa. https://www.glassdoor.es/Sueldos/programador-junior-sueldo-SRCH_KO0,18.htm. Accessed: 2023-9-18.

[7] Introduction to Yarn. https://yarnpkg.com/getting-started. Accessed: 2023-8-29.

[8] JTW. https://jwt.io/introduction. Accessed: 2024-3-4.

[9] Let's Encrypt. https://letsencrypt.org/es/. Accessed: 2024-3-19.

[10] React. marcado JSX. https://es.react.dev/learn/writing-markup-with-jsx. Accessed: 2023-8-29.

[11] RFC 7519. https://datatracker.ietf.org/doc/html/rfc7519. Accessed: 2024-3-4.

[12] Spring Boot Guide. https://spring.io/projects/spring-boot. Accessed: 2023-8-15.

[13] Vite. https://vitejs.dev/. Accessed: 2023-8-15.

[14] What is scrum? https://www.scrum.org/resources/what-scrum-module. Accessed: 2023-9-9.

[15] What is Scrum and how to get started. `https://www.atlassian.com/agile/scrum`. Accessed: 2023.

[16] ¿Qué es Docker? `https://aws.amazon.com/es/docker/`. Accessed: 2023-9-1.

[17] ¿Qué es HTTPS? `https://www.cloudflare.com/es-es/learning/ssl/what-is-https/`. Accessed: 2024-3-19.

[18] Universidad de Valladolid. Normativa de evaluación del Trabajo Fin de Grado de E.T.S. de Ingeniería Informática. Fecha de la última actualización del documento antes de consultarlo: 27/04/2023. Accedido: 10/11/2023.

[19] Universidad de Valladolid. Reglamento sobre la elaboración y evaluación del Trabajo Fin de Grado. Fecha de la última actualización del documento antes de consultarlo: 27/04/2023. Accedido: 10/11/2023.

[20] Laura Fletcher. Pivoltaltracker. pointing Scales: Linear or Fibonacci? `https://www.pivotaltracker.com/blog/2020-01-14-pointing-scales-linear-fibonacci`. Accessed: 2023-9-9.

[21] Michael Gant. Medium. Scrum and the Solo Dev. `https://medium.com/@jmgant.clearyeconsulting/scrum-and-the-solo-dev-fb8e810ed42b`. Accessed: 2023-9-9.

[22] David Herbert. What is React.js? `https://blog.hubspot.com/website/react-js`. Accessed: 2023-8-15.

Apéndice A

Manuales

A.1. Manual de despliegue e instalación

En esta sección se explica la forma de proceder para poder desplegar y modificar las aplicaciones desarrolladas en este proyecto.

- Entorno de red: La máquina debe disponer de los puertos 8080 y 443 abiertos. La máquina también debe estar configurada con el dominio *tfgs.inf.uva.es*.
- NGINX: Es necesario contar con este servidor instalado. La configuración de este se provee en la máquina suministrada para el desarrollo de este proyecto en la ruta */etc/nginx/sites-available/tfgs.inf.uva.es*
- Docker: Necesario para el despliegue del servidor *backend*. En el momento de realización de este documento se utiliza la versión 24.0.5 .

La compilación del servicio *backend* se realiza mediante el Dockerfile en el repositorio del proyecto por lo que no son necesarias dependencias para su compilación.

Para la compilación de la aplicación web se necesitan, como mínimo, Node 18.18 y npm 10.5.0. El proceso de compilación se detalla a continuación.

1. Ejecutar el comando a continuación que generará un directorio con el nombre de dist/

   ```
   $ vite build
   ```

2. Copiar el contenido de este directorio a la ruta */var/www/tfgs.inf.uva.es/html/* de la máquina de despliegue. Si el directorio *dist/* se comprime a fichero *tar* y se mueve a */home/usuario*, puede utilizarse el alias *deploy_webapp* para el copiado automático a la ruta indicada antes.

Para el despliegue del *backend* se puede utilizar el alias *pull_ deploy_ service* siendo necesario tener el repositorio del servicio en la ruta */home/usuario/git/*.

A.2. Manual de mantenimiento

A.2.1. Certificado SSL

La renovación del certificado es necesario realizarla cada 90 días. Esto se puede realizar con el siguiente comando A.2.1 en la máquina que aloja la aplicación web. Esta acción puede configurarse en un cron para que se realice automáticamente.

```
# cerbot renew
```

La actualización del certificado en el *backend* requiere de la ejecución de los comandos a continuación tras haber realizado el proceso anterior.

```
# openssl pkcs12 -export
    -in /etc/letsencrypt/live/tfgs.inf.uva.es/fullchain.pem
    -inkey /etc/letsencrypt/live/tfgs.inf.uva.es/privkey.pem
    -out domain.p12
    -name myalias
# keytool -importkeystore -srckeystore domain.p12
    -srcstoretype PKCS12 -destkeystore domain.jks
    -deststoretype JKS -alias myalias
# cp domain.jks ~/git/infuva_tfg_manager_service/src/main/resources/
# docker restart api_service
```

A.2.2. Instalación de entorno de desarrollo

Para el mantenimiento de ambas aplicaciones se necesita de un entorno de desarrollo que lo permita. A continuación se listarán los *software* recomendados para llevar a cabo estas tareas.

- Docker: Para el desarrollo del backend este se despliega por defecto en un contenedor Docker. Si bien no es obligatorio, no disponer de este requeriría de configurar la BD y despliegue en desarrollo de diferente forma a como se entrega en el proyecto.
- Editores de texto o IDEs: Durante el desarrollo, y como se indica en este documento, se utilizan los producto de JetBrains, IntelliJ y Webstorm. El desarrollador puede optar por otros de su gusto si así lo decide.
- Node 18.18: Necesario para correr la aplicación web.
- JDK 21 LTS.

Figura A.1: Menú principal

A.3. Manual de usuario

En esta sección se repasarán las diferentes pantallas que comprenden la aplicación y una explicación de uso de cada una de estas.

A.3.1. Menú principal

Esta es la pantalla inicial en la aplicación y que se muestra en Figura A.1. Desde esta se puede navegar a las diferentes secciones que contiene la aplicación y realizar el inicio de sesión en el botón de arriba a la derecha, el cuál redirigirá a la pantalla de inicio de sesión de la Universidad de Valladolid.

A.3.2. Propuestas

Desde esta pantallaA.2 se podrán consultar las propuestas de TFG pendientes de revisión que tiene el Coordinador de Grado. Si se posee el rol de profesor también se podrá acceder desde el botón de *Crear* a la creación de una nueva propuesta. Por otro lado al desplegar una propuesta y siendo Coordinador se puede acceder a la pantalla de detalles de propuesta. Ambas se explican a continuación.

La pantalla de creación de propuestas A.1 es accesible para cualquier usuario profesor.

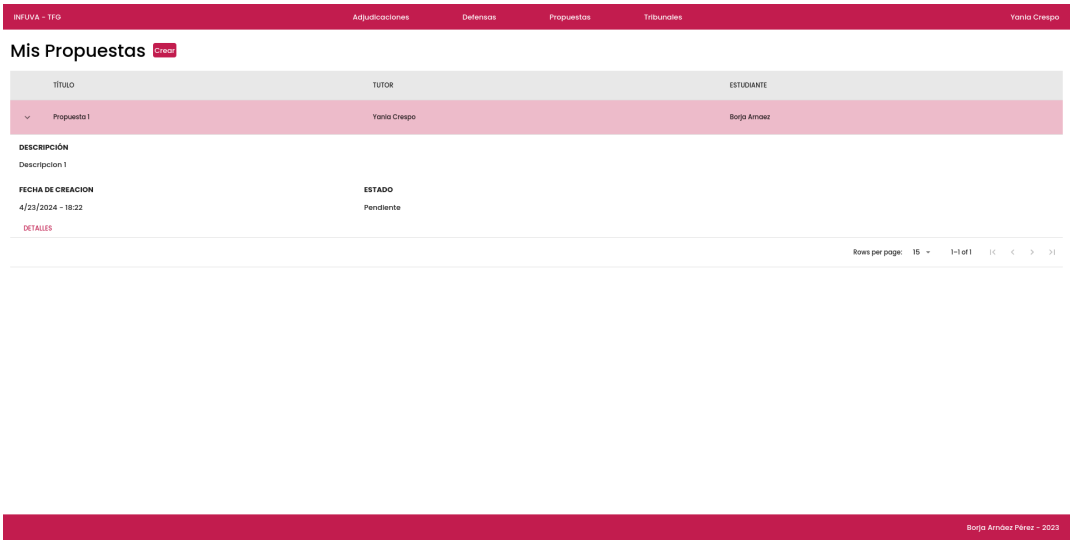

Figura A.2: Listado de propuestas

Desde esta se puede crear una nueva propuesta de TFG para enviarla para revisión al Coordinador de Grado, para ello se debe rellenar y enviar el formulario.

En la pantalla de detalles de la propuesta A.4 el Coordinador puede revisar y modificar el estado de esta.

A.3.3. Detalles de propuesta

A.3.4. Adjudicaciones

Accediendo desde el menú principal a la sección de *Adjudicaciones* se llega a esta pantalla A.5, desde la cual se pueden consultar las adjudicaciones y acceder, si se es Coordinador, a la pantalla de detalles de una adjudicación.

En esta pantalla de detalles A.6, el Coordinador puede modificar algunos detalles de la adjudicación como son el título, descripción, tutores o el estado de esta, así como gestionar la renovación si esta ha caducado. También, si el estado de la adjudicación lo permite, se puede crear una defensa de la adjudicación en un botón con el texto *Crear defensa* que se encuentra junto al resto de botones del formulario.

A.3.5. Tribunales

Análogo a las secciones anteriores podemos acceder a esta. En esta pantalla A.7 nos encontramos con un listado de los tribunales creados, si no somos el Coordinador solo se en-

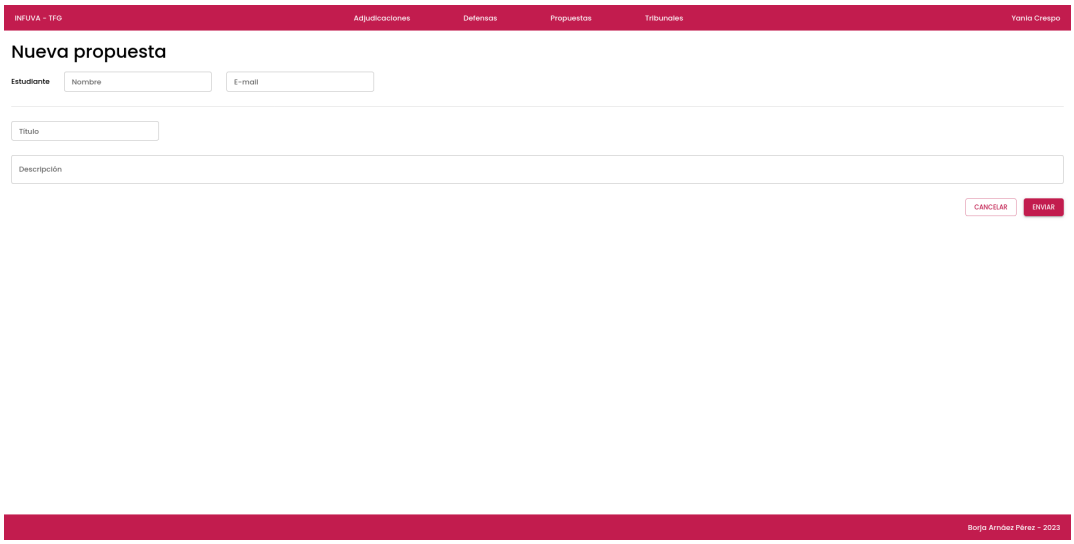

Figura A.3: Creación de propuesta

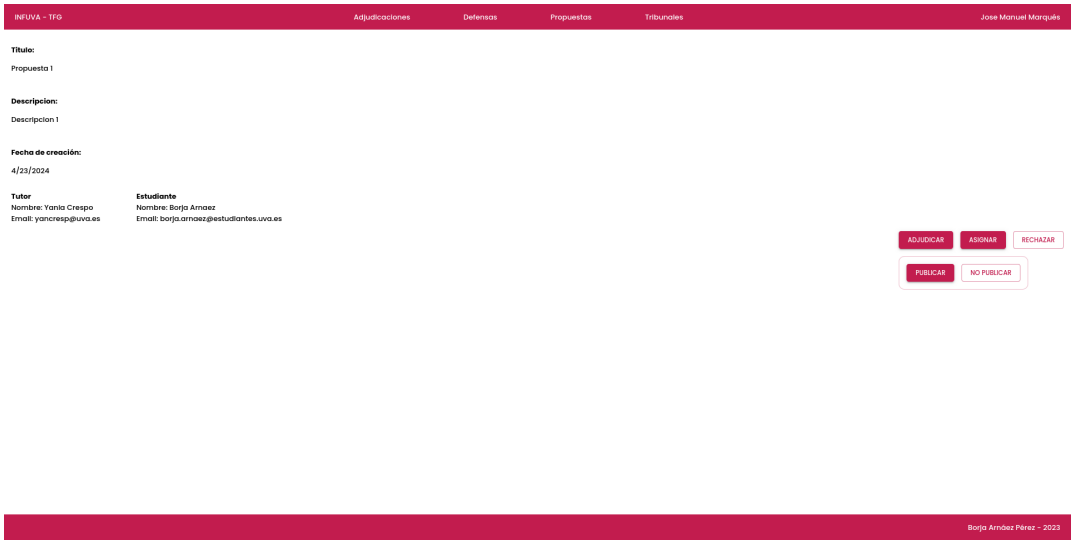

Figura A.4: Detalles de propuestas

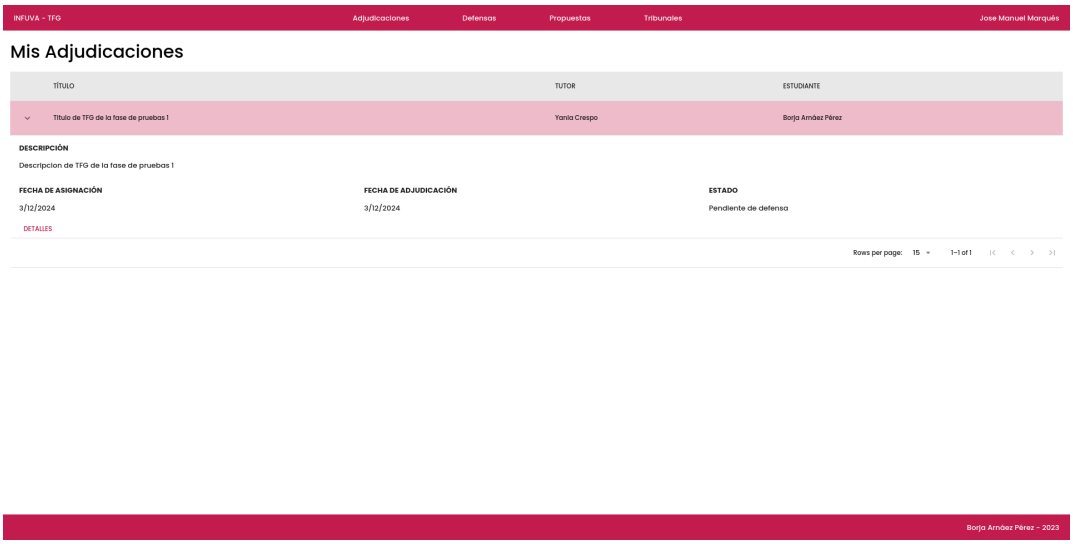

Figura A.5: Listado de adjudicaciones

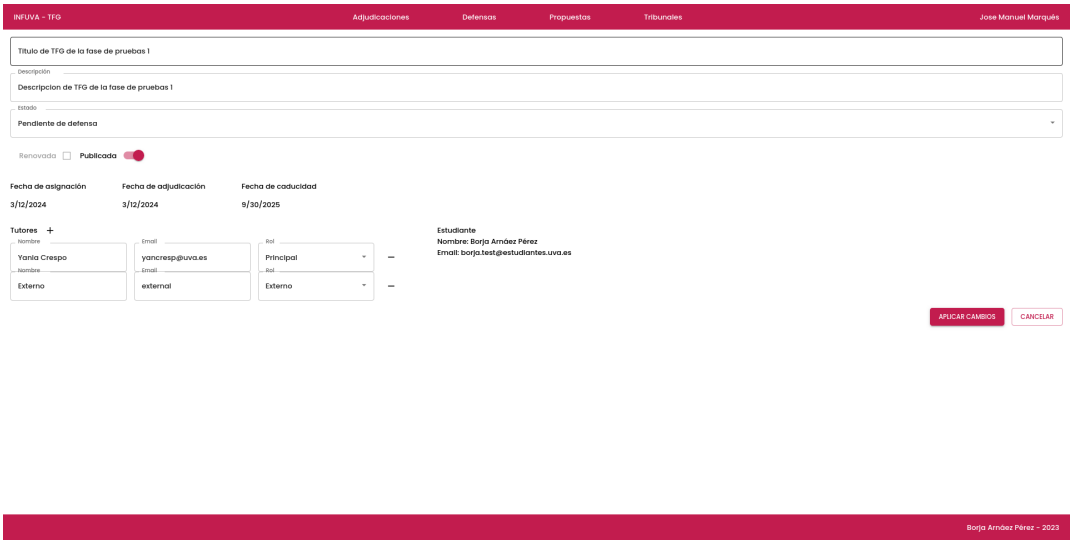

Figura A.6: Detalles de adjudicación

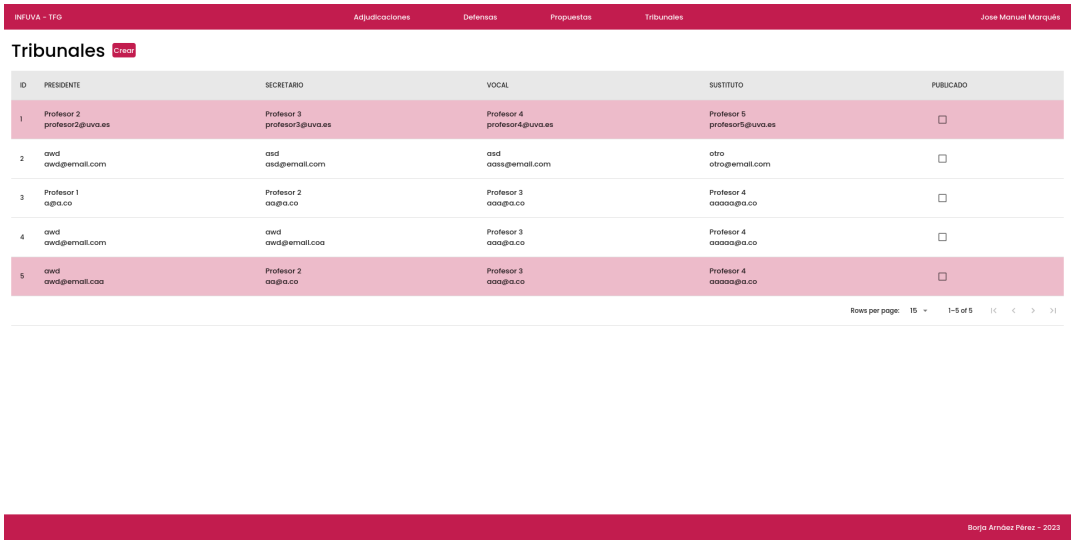

Figura A.7: Listado de tribunales

contrarán aquellos publicados. Desde esta, y de nuevo siendo Coordinador podremos acceder a la pantalla de creación de tribunales desde el botón *Crear*.

Una vez en esta pantalla se pueden crear uno o más tribunales. Los tribunales creados quedan a la espera de ser enviados en la columna de la derecha desde la cuál haciendo clic en el icono del lápiz y de la papelera, se puede editar o eliminar respectivamente el tribunal seleccionado.

A.3.6. Defensas

En la pantalla de defensas A.9 se muestran las defensas. El Coordinador puede navegar a los detalles de una defensa en concreto desde el icono de lápiz en la primera columna del listado.

Retomamos lo comentado en la pantalla de detalles de adjudicación y la forma de crear defensas. En la pantalla A.10 se muestran algunos detalles de la adjudicación para la cuál vamos a crear la defensa y se nos permite seleccionar de los tribunales ya existentes o crear uno *ad hoc* desde el par de botones bajo los detalles. La fecha y aula de la defensa se establecen desde la pantalla de detalles de la defensa mediante un formulario.

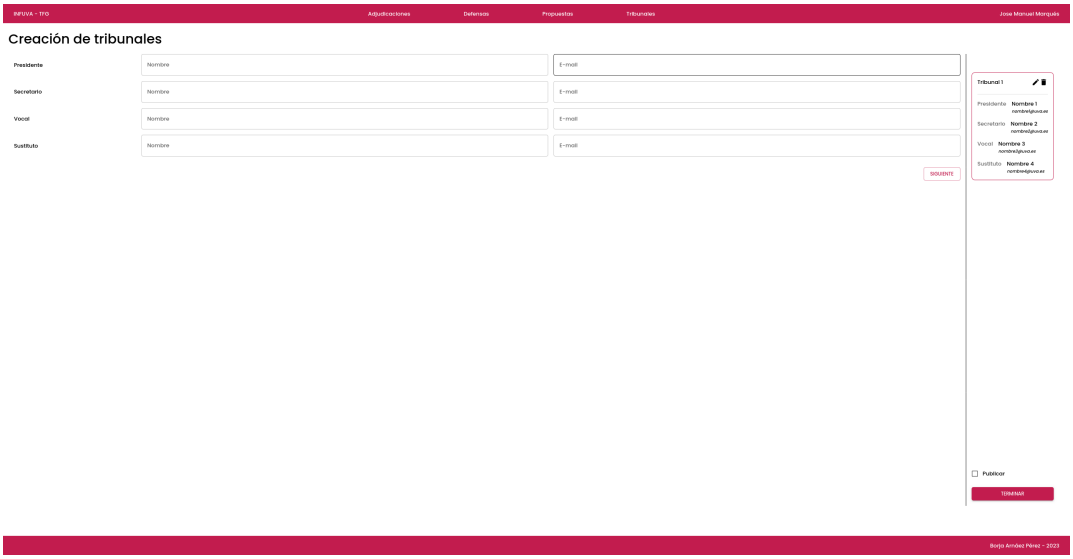

Figura A.8: Creación tribunales

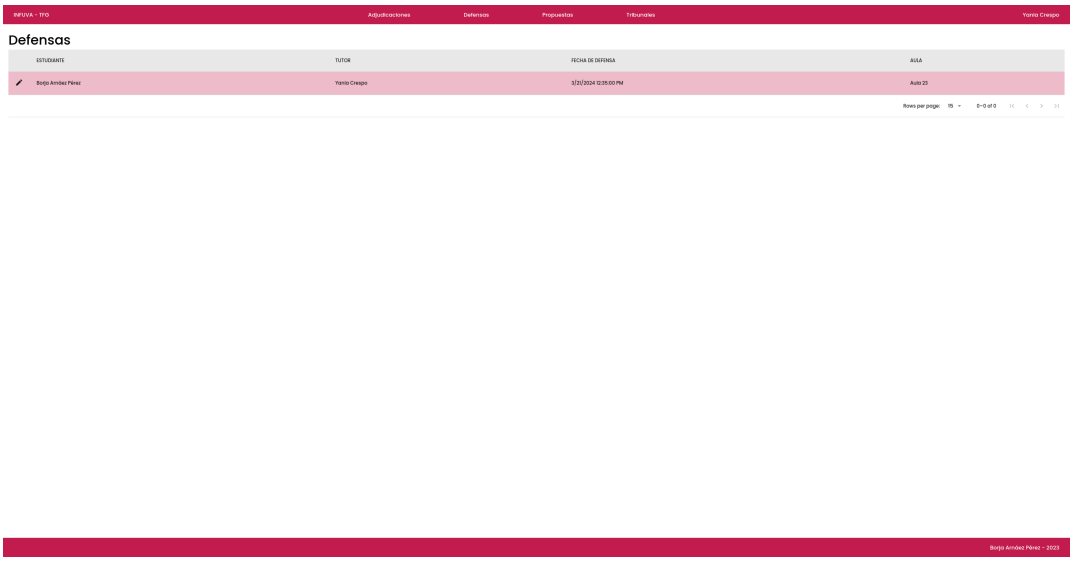

Figura A.9: Listado de defensas

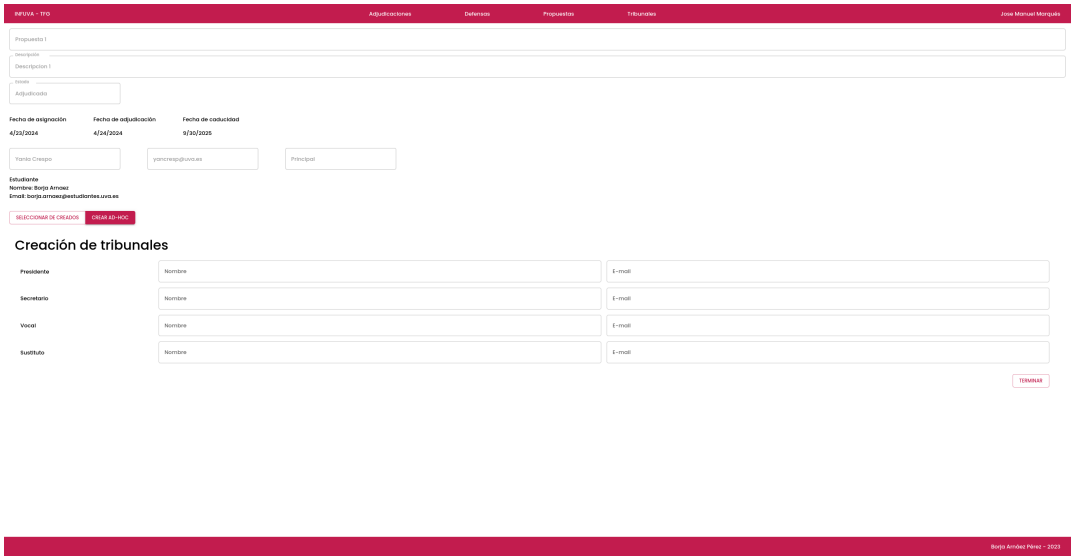

Figura A.10: Creación de defensa

Apéndice B

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- Repositorios.
  - Frontend: `https://gitlab.inf.uva.es/borja_arnaez_tfg/infuva_tfg_manager_webapp`
  - Backend: `https://gitlab.inf.uva.es/borja_arnaez_tfg/infuva_tfg_manager_service`
  - Documentación: `https://gitlab.inf.uva.es/borja_arnaez_tfg/tfg_documentacion`
- URL de despliegue: `https://tfgs.inf.uva.es/`