Grado en Ingeniería Informática

Trabajo Final de Grado

Extensión de funcionalidades del software de gestión de órganos colegiados, GOC

Autor:
Sergio Jimenez Chovares

Supervisor:
Nicolás Manero Carbó
Tutor académico:
Pedro Latorre Carmona

Fecha de lectura: 28 de junio de 2019
Curso académico 2018/2019

Resumen

Este documento recoge la implementación de una serie de nuevas funcionalidades a un software *open-source* por parte de Sergio Jimenez, alumno de cuarto año de ingeniería informática de la Universitat Jaume I de Castellón.

Durante el documento se describe el proyecto desde diferentes puntos, desde la perspectiva de usuario de la aplicación hasta la descripción de detalles técnicos. El proyecto retratado en el documento es GOC, Gestión de Órganos Colegiados, perteneciente a un consorcio de universidades. La aplicación se encuentra actualmente en funcionamiento y su objetivo es la gestión, seguimiento y convocatoria de reuniones entre los diferentes órganos colegiados que componen estas instituciones.

El documento también incluye la descripción del proceso llevado a cabo para la implementación de estas nuevas funcionalidades.

Las funcionalidades a implementar serán siete en total, pasarán a dividirse en tres *sprints*, que serán detallados para explicar tanto el objetivo de las diferentes historias de usuario contenidas en cada *sprint*, como el resultado final obtenido tras la finalización de cada uno de los *sprints*.

Palabras clave

GOC, órgano colegiado , reunión.

Keywords

GOC, collegiate body, meeting.

Índice general

1. Introducción **5**

1.1. Contexto y motivación del proyecto 5

1.2. Objetivos del proyecto 6

2. Descripción del proyecto **9**

2.1. Descripción de conceptos 9

2.2. Descripción funcional de la aplicación 10

2.3. Alcance previo 12

2.4. Alcance esperado 12

2.5. Alcance logrado 13

3. Planificación del proyecto **15**

3.1. Metodología 15

3.2. Planificación 15

3.3. Planificación real 17

3.4. Estimación de recursos y costes del proyecto 17

4. Análisis y diseño del sistema **21**

4.1. Modelo de datos del sistema 21

4.2. Arquitectura del sistema 21

4.3. Diseño de la interfaz 25

5. Implementación **27**

5.1. Detalles de implementación 27

5.2. Control de versiones 28

6. Desarrollo de la implementación **31**

6.1. *Backlog* 31

6.2. Sprint 1 32

6.2.1. Planificación del *sprint* 32

6.2.2. GOC4TIC-123 32

6.2.3. GOC4TIC-141 33

6.2.4. Comentarios 36

6.3. Sprint 2 38

6.3.1. Planificación del *sprint* 38

6.3.2. GOC4TIC-545 38

6.3.3. GOC4TIC-381 41

6.4. Sprint 3 42

6.4.1. Planificación del *Sprint* 42

6.4.2. GOC4TIC-163 42

6.4.3. GOC4TIC-143 45

7. Conclusiones **49**

7.1. Futuras ampliaciones de funcionalidad 50

8. Bibliografía **51**

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

El proyecto será llevado a cabo dentro de la empresa 4TIC Castellón 2009 S.L. cuyas instalaciones se encuentran dentro de la universidad Jaume I de Castellón en el edificio Espaitec. 4TIC es un empresa especializada en ofrecer diferentes soluciones para proyectos de las tecnologías de la información y la comunicación (TIC), especializada en el desarrollo de proyectos para la administración pública. Esta empresa desarrolla, mantiene, y mejora 3 herramientas a destacar:

- **Alejandría:** Software de gestión documental y de archivo junto con la gestión de depósitos físicos de estos, cumpliendo con el Esquema Nacional de Interoperabilidad (ENI).
- **Escena Online:** Software para la gestión de entradas para eventos, tanto para la venta de entradas a nivel *online* como en taquilla.
- **Gestión de Órganos Colegiados:** Software para la gestión de reuniones y órganos colegiados de las universidades, el cual mantiene en la nube toda la información siempre disponible y accesible.

El proyecto, que será incrementado mediante la implementación de nuevas funcionalidades y descrito en la siguiente memoria técnica, será GOC, Gestión de Órganos Colegiados.

Algunos ejemplos de usuarios de esta aplicación son los órganos de la universidad, el Claustro, Consejo de Gobierno, Consejo de Dirección, Consejo Social, Consejo de Estudiantes, o Juntas de Centro, entre otros. Esta herramienta cubre todo el proceso de organización de una reunión, desde su convocatoria hasta el cierre del acta. De esta forma, la aplicación facilita los trámites y posibilita, desde una misma plataforma, convocar una reunión, añadir puntos del orden del día, adjuntar documentación y enviar la convocatoria a los miembros de un órgano colegiado para que reciban la información de la reunión y confirmen su asistencia. Adicionalmente, una vez realizada la reunión, permite incluir el resultado de los acuerdos y deliberaciones y, seguidamente, cerrar el acta, validarla, firmarla digitalmente y archivarla para que quede guardada en un histórico.

GOC no pertenece a la empresa 4TIC, sino que es una herramienta *open-source*, gestionada, utilizada y desarrollada por un conjunto de universidades. 4TIC ha sido contratada para el mantenimiento y mejora de funcionalidades de forma externa por el conjunto de universidades. Entre las universidades participantes se encuentran Universitat Jaume I de Castellón (UJI), Universitat de Barcelona (UB), Universitat Oberta de Catalunya (UOC), Universidad Politécnica de Madrid (UPM) o la Universitat Politécnica de Cartagena (UPCT).

Su naturaleza *open-source* hace que el código fuente sea libre y que cualquier agente externo pueda contribuir desarrollando e implementando funcionalidades. Las nuevas funcionalidades o aportaciones provenientes de cualquier contribución externa siempre serán analizadas, revisadas y aceptadas o rechazadas por los responsables designados por las diferentes universidades que comparten la gobernanza de la herramienta.

Los agentes externos no solo pueden contribuir generando código, implementando nuevas funcionalidades o corrigiendo posibles anomalías del software, sino que también pueden sugerir la implementación de nuevas mejoras de funcionalidad o la corrección de errores. Si la propuesta del agente externo es de nueva funcionalidad, tras ser analizada y aceptada será dividida en el conjunto total de historias de usuario [9] necesarias para su implementación.

La creación, mantenimiento y evolución de estas historias de usuario se realiza en el*backlog* y pizarra Kanban [5] que las universidades miembro comparten en JIRA [4] y el JIRA propio de 4TIC.

Si el agente externo aporta el código de la nueva funcionalidad o de la corrección de la anomalía, este pasará a ser revisado y por una fase de testeo funcional antes de ser aprobado y unido al proyecto de forma oficial. Esto significa que el código necesario para que estas historias de usuario puedan ser implementadas deberá ser proporcionado por el agente externo, para su revisión.

La implementación de la nueva funcionalidad debe realizarse por la persona o empresa que contribuye y el código deberá ser subido a un repositorio para ser validado, por los diferentes responsables al cargo de las universidades.

El mecanismo de entrega del código aportado será el de *pull-request* [19] al repositorio que aloja el proyecto. Siendo este el mecanismo habitual en los proyectos abiertos.

1.2. Objetivos del proyecto

Para la satisfactoria consecución de este proyecto se pretende implementar una serie de mejoras en la herramienta GOC que resultan en un incremento considerable de las funcionalidades de la herramienta. Dichas mejoras fueron extraídas del *backlog* que las universidades propietarias del producto controlan en la plataforma web JIRA y del JIRA propio de la empresa 4TIC. En dicha plataforma tanto las universidades como el propietario del producto [18] por parte de la empresa, añaden a modo de historias en una pizarra *Kanban* las diferentes mejoras a implementar.

La empresa por su parte también tenía prevista una serie de mejoras a implementar, siendo también alguna, petición expresa por parte de alguna entidad concreta.

La implementación de estas nuevas funcionalidades permitirá a los usuarios de la aplicación realizar diligencias en las actas ya cerradas de las reuniones. También se permitirá asignar firmantes en las reuniones de forma dinámica, mejorando la flexibilidad en el cierre de actas y en la realización de reuniones. Será añadida la capacidad de duplicar reuniones para que el flujo de trabajo de los administradores de la aplicación y entes convocantes de las reuniones sea más fluido y sencillo, también se incluirá una mejora en la forma de tratar con los adjuntos de los puntos de orden del día.

- **GOC4TIC-123:** Como administrador quiero poder editar los firmantes de la reunión de manera dinámica.
- **GOC4TIC-141:** Como administrador quiero poder asignar un sustituto de un firmante si este no asiste a la reunión.
- **GOC4TIC-163:** Como administrador quiero poder gestionar las personas que tienen acceso a la aplicación desde la parte privada de la aplicación.
- **GOC-381:** Como administrador quiero poder realizar diligencias sobre las actas ya cerradas y firmadas.
- **GOC-593:** Como administrador quiero poder duplicar reuniones para poder volver a convocarlas.
- **GOC-545:** Como administrador quiero poder gestionar desde un solo punto los adjuntos de un punto de orden del día.
- **GOC4TIC-143:** Como usuario quiero poder acceder a la documentación de las reuniones sin ser miembro.

El proyecto es una aplicación web compuesta por dos partes: el *frontend* creado con el *framework* EXTJS [21] y JavaScript [24], el *backend* formado básicamente por tecnologías Java, incluyendo también SQL [23] y bases de datos. Para la realización del proyecto el alumno trabajará en las dos partes, modificando la base de datos, capa inferior, perteneciente a la parte de servidor y modificando también la parte del cliente.

A nivel formativo el trabajo tiene otros objetivos enfocados a adquirir distintas competencias y habilidades. Algunas de ellas serían:

- Aprender la metodología de trabajo de la empresa
- Ampliar los conocimientos de desarrollo con lenguajes de orientación a objetos
- Mejorar conocimientos de desarrollo de aplicaciones con servicios REST
- Mejorar los conocimientos de lenguajes no tipados como JavaScript
- Incrementar los conocimientos de *testing* añadiendo nuevas tecnologías como Cypress.

- Afianzar el paradigma de programación en los conocimientos del alumno.
- Conocer la estructura de la aplicación y diferentes buenas prácticas de programación.

Capítulo 2

Descripción del proyecto

La empresa 4TIC ha sido contratada por diferentes universidades clientes de la aplicación, con la finalidad de implementar un conjunto de nuevas funcionalidades para la misma. Gracias a la implementación de estas funcionalidades las universidades conseguirán entre otras mejoras, aumentar la flexibilidad de la plataforma, permitiendo dar de alta nuevos usuarios/as así como la administración de forma dinámica de los firmantes de cada reunión y órgano.

2.1. Descripción de conceptos

Dentro del proyecto y en el documento se trabajará con terminología de la aplicación, en esta sección se pretende esclarecer de forma concreta el significado de algunas de las palabras recurrentes en el desarrollo del proyecto y el documento, dando la correcta acepción y significado dentro del aplicativo.

- **Reunión:** Es el componente central de la aplicación, se trata del nexo del resto de componentes, y objetivo final de la plataforma. Ésta contiene los diferentes puntos de orden del día, acuerdos, temporalidad de la misma etc.
- **Miembro:** Componentes mínimos de un órgano que asisten a las reuniones, con características, como autorizado, asistencia, capacidad de firma, etc.
- **Órgano:** Conjunto de miembros, unidad mínima que puede ser convocada a una reunión. En una reunión pueden asistir uno o múltiples órganos.
- **Puntos de orden del día:** Temas a tratar en una reunión. Dentro de estos se encuentran los diferentes acuerdos y deliberaciones a los que se llega tras la realización de la reunión.
- **Acta:** Refleja en forma de pdf los acuerdos y deliberaciones de la reunión así como los puntos del orden del día. Deberá ser firmada para certificar la veracidad de los acuerdos y deliberaciones a los que se ha llegado durante la reunión.
- **Persona:** Se trata de una entidad no existente en el estado actual de la plataforma, que será implementada durante el desarrollo del proyecto. Ésta diferencia a los miembros de

los órganos de los usuarios, en este caso, personas, con acceso a la aplicación, es decir, el cómputo total de usuarios de la aplicación que actualmente esta externalizado.

2.2. Descripción funcional de la aplicación

La aplicación está formada por dos secciones desde el punto de vista del usuario. Una parte privada y una pública.

Desde la sección privada, los usuarios con los permisos adecuados pueden crear órganos, reuniones, añadir puntos de orden del día, miembros, enviar convocatorias, etc.

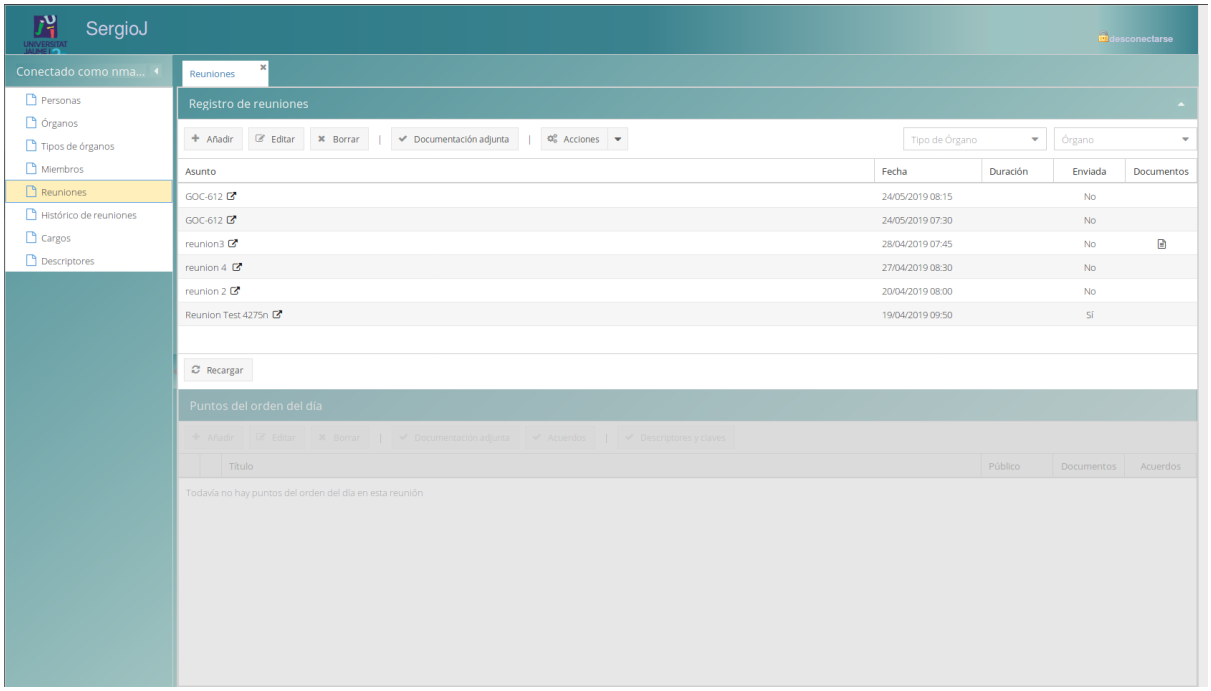

Figura 2.1: Parte privada de la aplicación.

Desde la sección pública (figura 2.2), los usuarios pueden confirmar asistencia, asignarse un suplente que asista en su lugar, acceder a la documentación adjunta de la reunión, revisar información sobre los diferentes puntos de orden del día a tratar, ver el nombre del convocante, consultar la temporalidad de la reunión, fecha o duración de la reunión, etc. Todo esto desde la URL recibida en la convocatoria, que les es remitida por correo electrónico.

Así pues, el flujo de trabajo de la aplicación es, desde de la sección privada por parte de un administrador o autorizado, primero la creación de los diferentes órganos, en los que se añadirán los miembros formantes, seguidamente la creación de una reunión, la adición de órganos a la reunión y los diferentes puntos de orden del día, posteriormente la configuración de la temporalidad y otras opciones, y finalmente el envío de la convocatoria. Tras esto, cada uno de los miembros de órganos invitados recibirá un correo con un enlace con la información sobre la reunión, que les permitirá también gestionar la suplencia o la asistencia. Una vez realizada la reunión, el autorizado rellenará los acuerdos y deliberaciones de la misma en cada punto del

orden del día y estos serán añadidos al acta.

Existen dos tipos de usuarios. Usuarios con acceso a la sección privada y usuarios con acceso a la sección pública, aunque dentro de cada una de estas categorías existen varios tipos de roles con diferentes niveles de acceso y privilegios.

Los usuarios con acceso a la sección privada (figura 2.1 en la página 10) tienen acceso a un menú que contiene las siguientes diferentes opciones:

- **Reuniones**
- **Histórico Reuniones**
- **Órganos**
- **Miembros**
- **Puntos de orden del día**
- **Descriptores**

Los usuarios de la parte pública (figuras 2.2 y 2.3) solo tienen acceso a la información de la reunión a la que han sido invitados y en ningún caso capacidad de modificación de la misma.

Figura 2.2: Parte pública de la aplicación, vista "reunión".

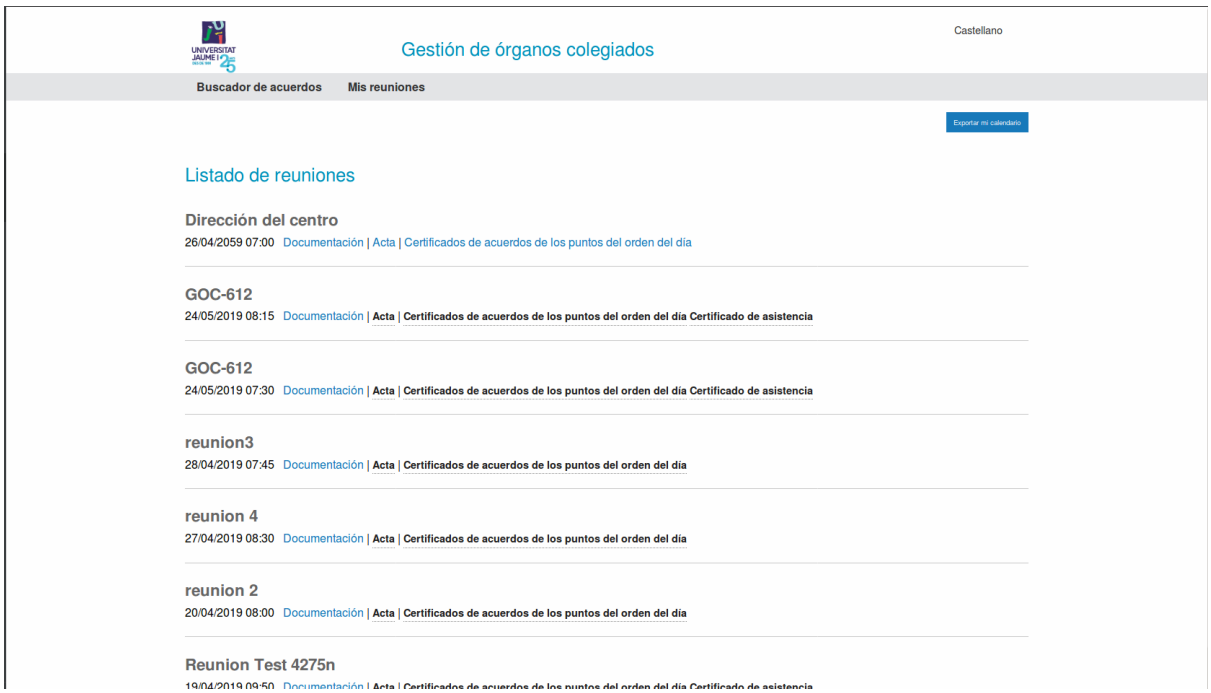

Figura 2.3: Parte pública de la aplicación, vista "mis reuniones".

2.3. Alcance previo

El alcance de la aplicación incluye todas las funcionalidades descritas anteriormente. En la actualidad no se permite la adición de usuarios a la aplicación, si no es a través de un sistema externo que envíe la información requerida por el aplicativo para la adición de una persona. Por tanto para que una persona pueda ser usuaria de la aplicación en la actualidad, ésta debe de forma obligatoria, estar dada de alta en un sistema externo, distinto en cada uno de los diferentes clientes.

Al realizar las actas, éstas son firmadas al pie. Actualmente solo pueden ser firmadas por el presidente y el secretario del órgano, sin ningún tipo de identificación individualizada. Se trata de textos fijos en el pie del acta.

El aplicativo permite la adición de acuerdos y documentos a través de diferentes modales. Al cerrar una reunión y adjuntar las actas y deliberaciones, las reuniones quedan cerradas pasando a ser reuniones históricas. Las reuniones históricas en el alcance actual no admiten ningún tipo de modificación. Solo los miembros de un órgano tienen acceso a la reunión.

2.4. Alcance esperado

Al finalizar el proyecto, deberían haberse alcanzado los siguientes objetivos:

- La capacidad de añadir de forma local (dentro de la propia aplicación) nuevos usuarios,

pudiendo gestionar también sus datos incluso e modificarlos, dejando de depender de un sistema externo para esto.

- Podrán ser seleccionados tantos firmantes como se deseen en cada órgano de forma que estos serán los que aparecerán en el pie de la documentación como firmantes.
- Los firmantes podrán ser sustituidos por un suplente designado.
- Una nueva ventana de documentación adjunta en los puntos de orden del día, permitirá gestionar todos los ficheros y acuerdos que se adjunten al punto del orden del día.
- Se permitirá a los usuarios con privilegios de convocatoria de reuniones, duplicar reuniones, para evitar así tener que hacer desde el principio reuniones que se realizan de forma periódica o recurrente.
- Se permitirá, en caso de ser necesario, reabrir una reunión ya cerrada, permitiendo la edición de sus acuerdos o deliberaciones y sus puntos de orden del día, marcando qué puntos han sido modificados o quién ha reabierto la reunión.

2.5. Alcance logrado

Al finalizar la implementación del proyecto se han logrado implementar con éxito todos los puntos de alcance preestablecidos a excepción de:

- Se permitirá a los usuarios convocantes de reuniones, desde la sección privada, duplicar reuniones para evitar así tener que hacer desde el principio reuniones que se realizan de forma periódica o recurrente.

Este punto de alcance se planteó inicialmente como tarea perteneciente al primer *sprint* (GOC-593). Finalmente, no pudo ser incluida en el *sprint* debido a un consumo de tiempo excesivo por parte de las otras historias. Este hecho será explicado en mayor profundidad en el apartado implementación del *sprint* 1.

Capítulo 3

Planificación del proyecto

3.1. Metodología

Se trata de un proyecto maduro cuya implementación se realiza utilizando una metodología de desarrollo ágil basada en *scrum* [5] con ligeras modificaciones. La empresa recibe peticiones específicas por parte de las distintas universidades por diferentes medios, actuando las universidades como propietarias del producto. Un ejemplo de cómo las universidades hacen llegar las historias a la empresa podría ser la Universidad Jaume I de Castellón, utilizando una pizarra Kanban, mientras que otro ejemplo sería la Universidad de Cartagena que lo hace a través de reuniones similares a las de planificación del *sprint* en las que describe cuáles son los incrementos esperados.

El equipo en conjunto de la empresa se encuentra en una misma oficina, lo que facilita la comunicación y hace la existencia del *daily scrum* innecesaria, pues la comunicación es directa y constante.

Dentro de la empresa, el desarrollo y mantenimiento del proyecto es realizado en su totalidad por el autor de este documento, Sergio Jimenez, mientras que el gerente y propietario de la empresa, Nicolás Manero, actúa directamente como propietario del producto para esclarecer de forma continua y directa todos los requerimientos y detalles que las nuevas funcionalidades precisen. Al tratarse de una metodología basada en la metodología ágil las diferentes funcionalidades son divididas en historias de usuario. Por su parte la empresa tiene un *backlog* propio donde se introducen todas las historias requeridas por las universidades y una pizarra *Kanban* gestionada en un *board* de JIRA.

3.2. Planificación

La duración estimada para la implementación de estas historias de usuario es de 300 horas. Para valorar el tiempo que una historia de usuario puede necesitar para su consecución, se utilizará la unidad de punto de historia de usuario. Cada punto de historia de usuario equivaldrá

a una jornada de trabajo, las jornadas de trabajo del alumno son de 8 horas, por lo que la previsión es que las historias de usuario sean completadas en unas 8 semanas de trabajo.

Se necesitarán dos semanas para el aprendizaje de las nuevas tecnologías utilizadas dentro del proyecto, para familiarizarse con todo el código existente y su funcionamiento, así como con el funcionamiento de la empresa.

Se realizarán 3 *sprints* de dos semanas cada uno. Cabe recalcar que este documento ha sido realizado con la totalidad de historias de usuario implementadas, pues el alumno se encontraba en situación de exención de practicas, desempeñando funciones a nivel profesional para la empresa, por lo que se han seleccionado los primeros *sprints* realizados por el autor del documento en 4TIC 2009 S.L.(véase tabla 3.1).

	Fecha inicio	Fecha fin	Historias de usuario	Coste estimado (PH)
Aprendizaje	3/12/18	14/12/18	Toma de contacto con el proyecto, metodología y nuevas tecnologías.	10
Sprint 1	17/12/18	4/1/19	GOC4TIC-123 Poder seleccionar los firmantes del acta de forma dinámica. GOC-593 Como administrador quiero poder duplicar reuniones para reconvocarlas. GOC4TIC-141 Seleccionar el sustituto de un firmante.	GOC4TIC-123 4 GOC4TIC-593 4 GOC4TIC-141 2
Sprint 2	14/1/19	25/1/19	GOC-545 Poder gestionar desde un solo punto los adjuntos de un punto de orden del día. GOC-381 Poder realizar diligencias sobre reuniones ya cerradas	GOC4TIC-141 5 GOC-381 5
Sprint3	18/2/19	8/2/2019	GOC4TIC-163 Gestionar las personas que tienen acceso a la aplicación desde la parte privada de la aplicación GOC4TIC-143 Como usuario quiero poder acceder a la documentación de las reuniones sin ser miembro.	GOC4TIC-163 6 GOC4TIC-143 4

Cuadro 3.1: Sprints previsión

Durante el primer *sprint* se implementarán dos historias de usuario que permitirán la adición

de firmantes de forma dinámica por cada órgano y en caso de que éste no pueda asistir y le sea asignado un suplente, el suplente pasará a ser el firmante del acta, apareciendo en cualquier caso su nombre en las plantillas pdf de las actas.

El segundo *sprint* está enfocado a dos historias completamente diferenciadas con dos objetivos diferentes y claros. Estos son añadir un punto de entrada a la documentación adjunta, para los puntos de orden del día, tanto documentos como acuerdos y poder realizar diligencias sobre las reuniones.

El último *sprint* está compuesto por dos tareas que contienen una pequeña dependencia entre ellas. Una permitirá la inserción y gestión de las personas que pueden acceder a la aplicación desde la parte privada, mientras que la otra aprovechará esta nueva funcionalidad para permitir que usuarios no miembros de órganos en la plataforma, puedan acceder a reuniones invitados como personas externas.

3.3. Planificación real

Durante el desarrollo del proyecto se modificó la planificación temporal, eliminando una de las historias de usuario (en rojo en la tabla 3.2), ya que la realización de una de las tareas supuso un esfuerzo mayor al esperado (en verde en la tabla 3.2), la principal causa de esta modificación fue el hecho de tratarse de la primera tarea realizada por el autor del documento en el proyecto GOC y por tanto tratarse de una primera toma de contacto con el software.

3.4. Estimación de recursos y costes del proyecto

La estimación de recursos y costes, los dividiremos en 2 categorías:

- **Costes humanos**
- **Costes materiales**

Los costes humanos (figura 3.3) asociados al proyecto son los equivalentes al trabajo de una persona durante el lapso de 300 horas. El proyecto utiliza una serie de tecnologías, con coste asociado, como ExtJs con un coste posible de 8171,75 € [16], aunque éste pertenece a las universidades y no genera coste alguno para la empresa. Esta clase de productos están siendo utilizados en más proyectos por las universidades, por lo que se desconoce el coste prorrateado qué correspondería a este producto. Como los gastos son transparentes para la empresa, el coste de las tareas descritas en este documento, desciende y se simplifica al mero coste humano y de material.

Se sumarán los gastos de equipos utilizados en el entorno de desarrollo, valorados en 700 € y el pago de la licencia de Intellij Idea como IDE (en este caso la versión *ultimate*), con un coste asociado de 49.90 € al mes (3.4) [14].

	Fecha inicio	Fecha fin	Historias de usuario	Coste estimado (PH)
Aprendizaje	3/12/18	14/12/18	Toma de contacto con el proyecto, metodología y nuevas tecnologías.	∼
Sprint 1	17/12/18	4/1/19	GOC4TIC-123 Poder seleccionar los firmantes del acta de forma dinámica. GOC-593 Como administrador quiero poder duplicar reuniones para reconvocarlas. GOC4TIC-141 Seleccionar el sustituto de un firmante.	GOC4TIC-123 8 GOC4TIC-593 4 GOC4TIC-141 2
Sprint 2	14/1/19	25/1/19	GOC-545 Poder gestionar desde un solo punto los adjuntos de un punto de orden del día. GOC-381 Poder realizar diligencias sobre reuniones ya cerradas	GOC4TIC-141 5 GOC-381 5
Sprint3	18/2/19	8/2/2019	GOC4TIC-163 Gestionar las personas que tienen acceso a la aplicación desde la parte privada de la aplicación GOC4TIC-143 Como usuario quiero poder acceder a la documentación de las reuniones sin ser miembro.	GOC4TIC-163 6 GOC4TIC-143 4

Cuadro 3.2: Sprints real

Horas	Coste €/hora	Total
300	15	4500 €

Cuadro 3.3: Tabla costes humanos

Material	Coste
Ordenador	700 €
Alquileres, consumo de luz y derivados	100 €

Cuadro 3.4: Tabla costes materiales

La suma de los costes calculados nos llevaría a un total aproximado de 5.400 €, se añade la totalidad del equipo de desarrollo (ordenador) de 700 € puesto que el proyecto sigue y el desarrollo de la plataforma es su principal funcionalidad.

Capítulo 4

Análisis y diseño del sistema

4.1. Modelo de datos del sistema

Las figuras 4.1 y 4.2 muestran todos los modelos de datos de la aplicación. Al tratarse de una aplicación de tamaño considerable, el modelo de datos ha tenido que ser dividido en dos figuras. Para facilitar la lectura se ha creado un enlace que permite descargarla.

(https://drive.google.com/open?id=1zfFLJa0ff65zV9HTJUjkNDgYzlLWDjog)

4.2. Arquitectura del sistema

La arquitectura de la aplicación está dividida en dos módulos distintos que operan a diferentes niveles: goc-base y goc-core (figura 4.3). Ambos módulos forman parte de un sistema cliente servidor.

En este sistema existe:

- Cliente web: Encargado de mostrar los datos oportunos, ordenarlos y dar la capacidad de ejecutar acciones como modificaciones sobre estos datos.
- Servidor: Sistema que captura las peticiones creadas por el sistema cliente para responder acorde aplicando la lógica de negocio necesaria, realizando almacenamientos, modificaciones, y borrados sobre los diferentes datos de la plataforma.

Dentro de cada una de las partes existe una arquitectura diferente. En la parte de cliente existe una arquitectura MVC (*model*, *view*, *controller*). En la arquitectura MVC, la mayoría de las clases son Modelos, Vistas o Controladores. El usuario interactúa con las vistas que son las que muestran los datos que definen los modelos. Estas interacciones son monitorizadas por los controladores, quienes responden a las interacciones del usuario actualizando la vista y el modelo.

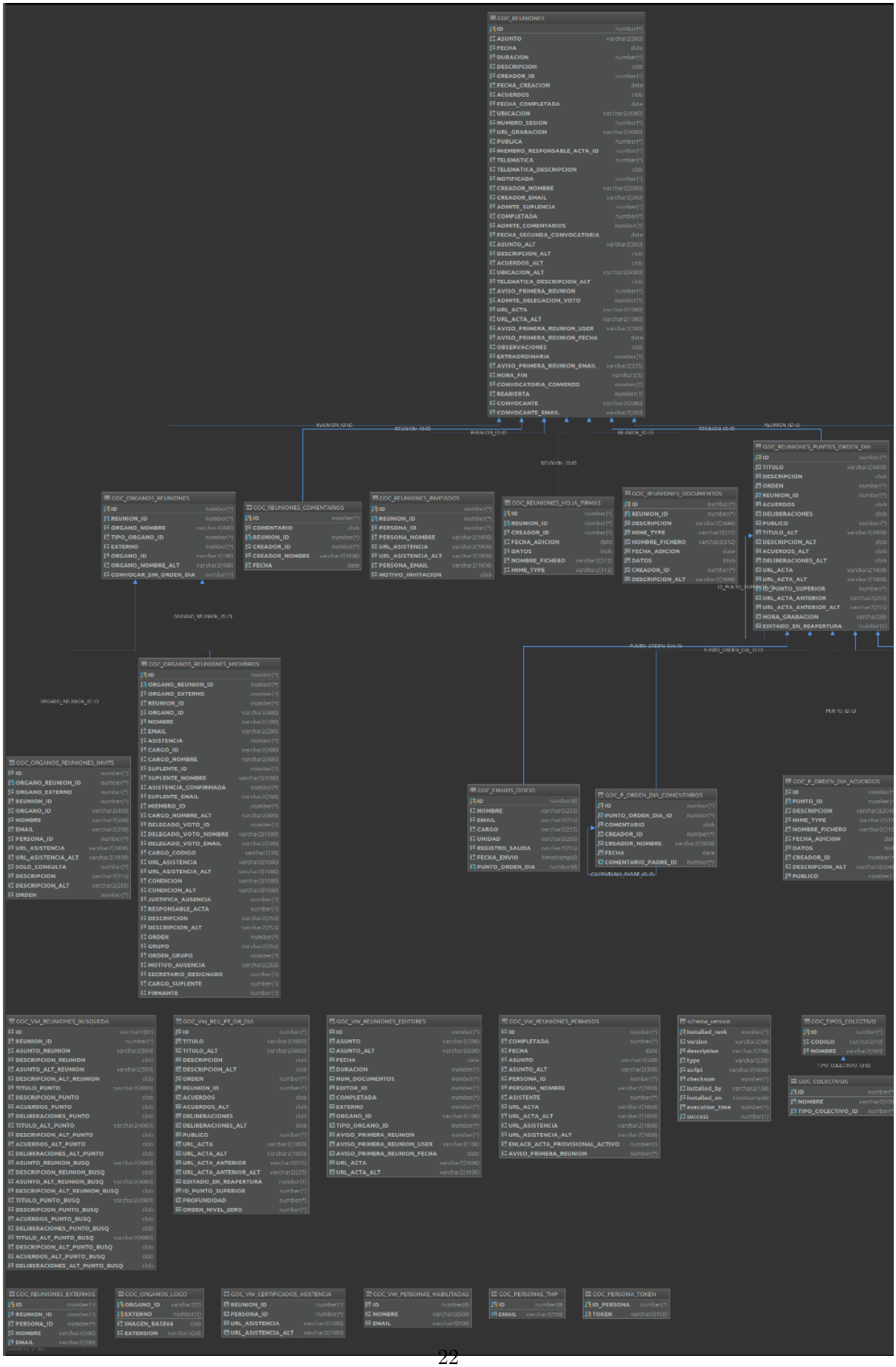

Figura 4.1: Base de datos y clases, parte 1.

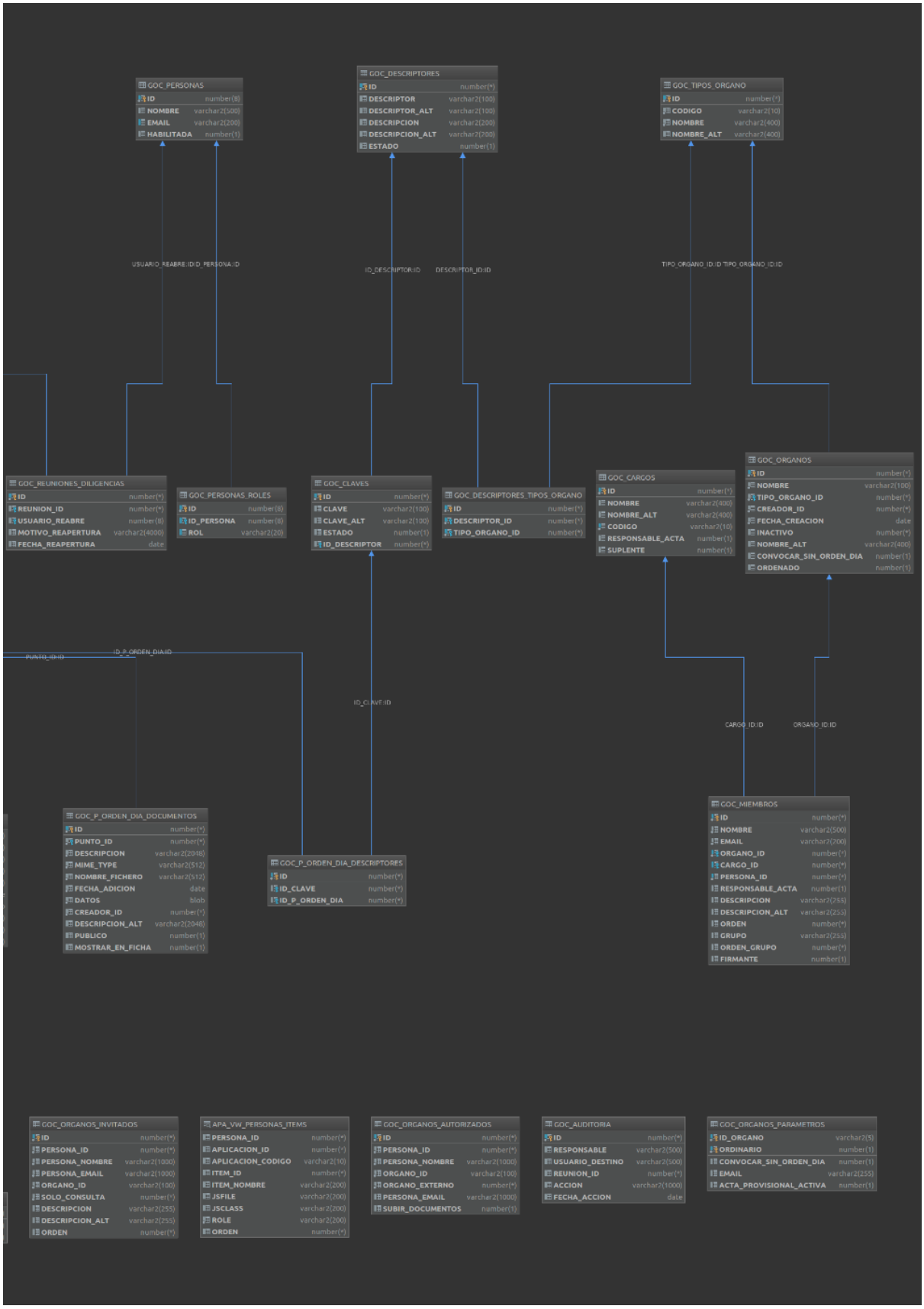

Figura 4.2: Base de datos y clases, parte 2.

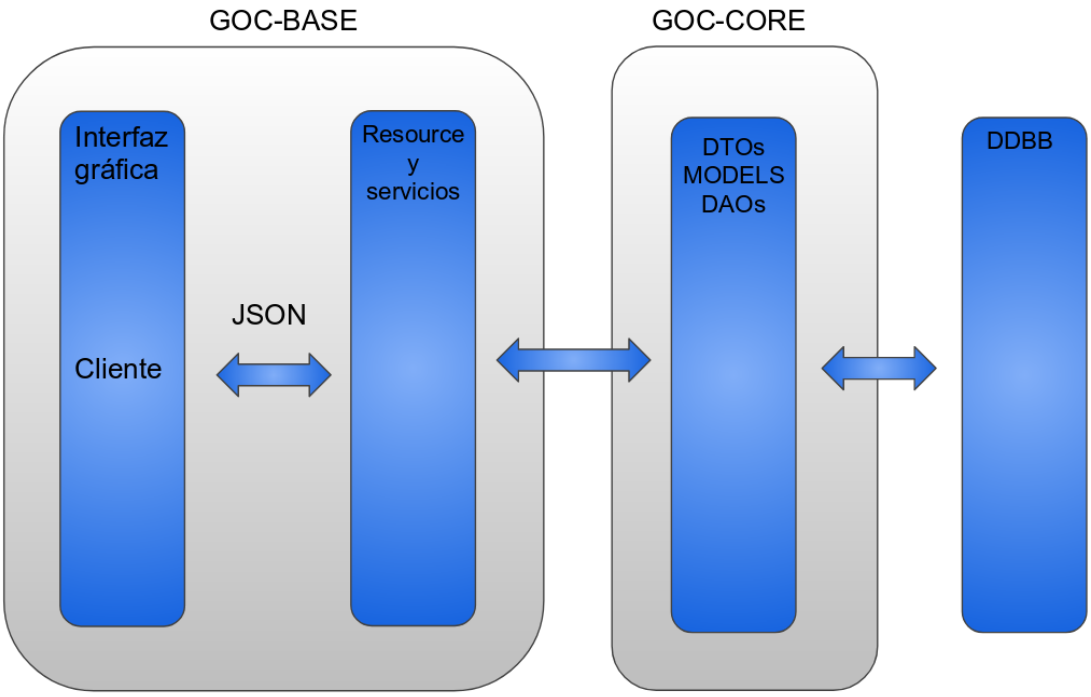

Figura 4.3: Arquitectura de sistema

La vista y el modelo desconocen la existencia el uno del otro ya que el controlador es el que tiene la responsabilidad de dirigir las actualizaciones. Los controladores son las clases que contienen la mayor parte de la lógica de negocio en una arquitectura MVC.

El objetivo de la arquitectura MVC es definir claramente las responsabilidades de cada clase en la aplicación. Debido a que cada clase tiene claramente definido su objetivo, implícitamente se desacoplan entre ellas. De esta forma conseguimos código más limpio, legible, fácil de mantener y más reusable [10].

Las vistas, los controladores y modelos se definen con el lenguaje Javascript, mediante el framework ExtJs de Sencha, realizan peticiones asíncronas al servidor mediante Ajax.

Tanto el envío como la recepción de la información entre la parte de servidor y la parte de cliente, se realiza mediante la serialización de objetos en ficheros JSON utilizando para ello un procesador de JSON de alto rendimiento llamado Jackson.

Extjs contiene unos elementos llamados *stores* que son capaces de almacenar datos en la parte de cliente, otorgando flexibilidad y rapidez en las acciones realizadas en las vistas, ya que realiza modificaciones de datos en su *store* y los muestra sin que las modificaciones hayan tenido que esperar una respuesta del servidor.

En el servidor existe una arquitectura por capas, donde encontramos diferentes tipos de componentes.

- **DTO (Data Transfer Object):** Es un objeto para la transferencia de datos entre proce-

sos. Cuando trabajamos con interfaces remotas, cada llamada entre interfaces implica un elevado consumo de recursos, por lo que puede resultar conveniente la reducción de estas llamadas. Una solución es crear un objeto de transferencia de datos, que puede contener todos los datos necesarios en la transacción entre las interfaces en una sola llamada. Para lograr la transferencia de estos objetos se requiere serializar el objeto para que pueda ser enviado a través de la conexión. En la aplicación los DTOs son los objetos que la tecnología Hibernate transfiere entre las entidades de base de datos y el servidor mediante anotaciones.

- **DAO (Data Acces Object):** Abstrae y encapsula todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos, almacena y extrae datos.
- **Service:** Contiene la lógica de negocio, realiza cálculos a partir de unos datos de entrada y datos almacenados, validación de datos y preparación de los datos que deben ser presentados. La capa de servicios, contiene todas las posibles operaciones que la aplicación puede realizar al interactuar con el cliente [20].

En la aplicación estos y otros componentes se separan en dos módulos.

- **Goc-base:** Módulo superior, formado por los *resources* y servicios. En los *resources* se encuentran los métodos que capturan las peticiones de la parte del cliente. Los datos son cedidos a la capa de servicio ya deserializados para que se realicen las correspondientes transformaciones en los modelos si es requerido y aplique la lógica de negocio.
- **Goc-core:** Segundo módulo o módulo inferior, realiza el paso de los modelos a DTOs sirviéndolos a los DAOs para que se encarguen en esta capa final de realizar la persistencia de datos y el acceso a ellos. Para esta última capa se utilizan diferentes tecnologías, destacando el uso de Hibernate y Querydsl.

4.3. Diseño de la interfaz

En este apartado se muestra una captura de pantalla del diseño de la aplicación a nivel genérico (figura 4.4). En posteriores secciones se podrán observar los diseños de las vistas creadas para las distintas tareas que se han implementado y se describen en este documento.

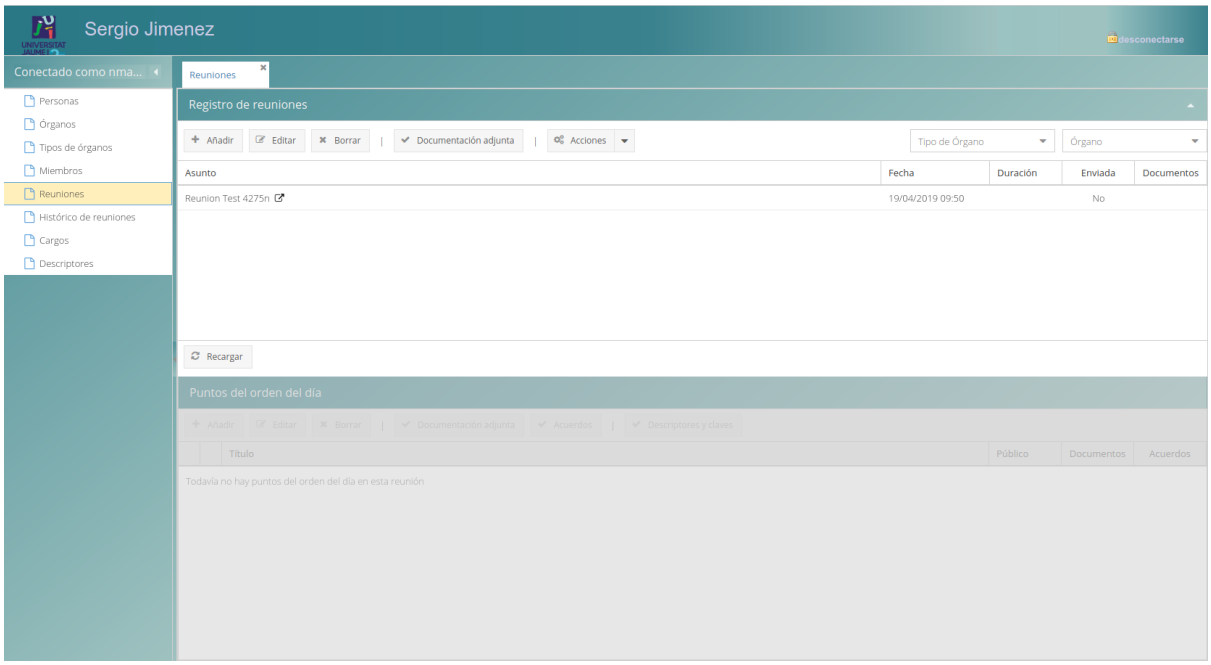

Figura 4.4: Diseño genérico de la aplicación

Capítulo 5

Implementación

5.1. Detalles de implementación

Durante la implementación de las tareas se ha utilizado una serie de librerías de terceros:

- **Junit:** Librería para la implementación de pruebas unitarias que permite el testeo funcional de partes de código, así como la creación de agrupaciones llamadas baterías [12].
- **Cypress:** Librería que permite el testeo integral de la aplicación, *end-to-end*, simulando el uso de un usuario. Esta librería utiliza Javascript [25].
- **Jersey:** Permite el desarrollo de servicios rest añadiendo soporte a JAX-RS [11] en el proyecto [17].
- **Hibernate:** Añade al proyecto la capacidad de realizar el mapeo objeto-relacional, permitiendo llevar las clases de Java que el proyecto incluye a base de datos, en el caso del proyecto mediante el uso de las anotaciones [8].
- **Hibernate JPA:** JPA permite usar Hibernate mediante las anotaciones en los proyectos Java. Esto permite el mapeo de los objetos en base de datos sin necesidad de definir nada más [1].
- **Log4j:** Permite mejorar el *log* del proyecto para mostrar más información [13].
- **ExtJs:** Permite la creación de aplicaciones multiplataforma [10].
- **Querydsl:** Permite a los usuarios de esta tecnología escribir *queries* SQL sin usar cadenas de *strings*, mejorando la escritura y lectura de los queries y añadiendo un capa de tipo de datos [6].
- **Jackson:** Permite la serialización y deserialización de objetos en ficheros JSON mediante anotaciones Java [15].
- **Spring:** Framework con múltiples funcionalidades. En el caso particular de este proyecto, se aprovecha la capacidad de insertar dependencias entre clases mediante anotaciones Java de una forma sencilla y cómoda [22].

5.2. Control de versiones

Dado que el proyecto aquí documentado consiste en incrementar un producto existente ya alojado en repositorios, fue una parte importante del aprendizaje y la adaptación al proyecto, conocer las plataformas que lo alojan y las tecnologías que lo implementan. El proyecto se encuentra alojado en la plataforma perteneciente a la compañía Atlassian, BitBucket [3] (figura ejemplo de la aplicación BitBucket 5.1). Además, para el manejo de versiones se utiliza Git [7].

Las funcionalidades nuevas siempre se realizan en nuevas ramas Git. Cada vez que una nueva historia de usuario comienza, ésta pasa a ser una nueva rama, que después (tras una validación) será unida al proyecto.

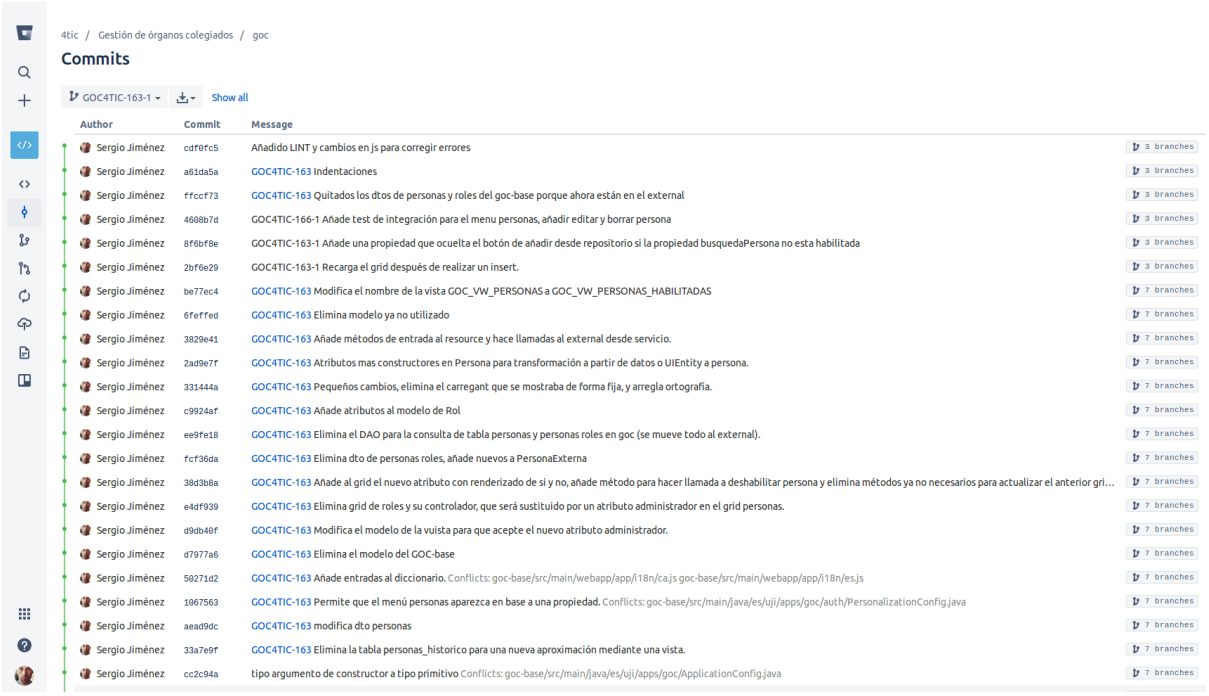

Figura 5.1: Muestra de la aplicación Bitbucket

El control de versiones sigue una metodología de trabajo llamada *Git flow* [2] donde el flujo de trabajo está basado en dos ramas principales, la rama *master* donde se encuentran las versiones desplegadas y la rama *develop*. La rama *develop* es la rama desde la que parten las nuevas *features* o funcionalidades. Esto implica la creación de una nueva rama *feature* para cada una de estas nuevas funcionalidades. Posteriormente las ramas *feature* son incorporadas a *develop*, para finalmente *develop* ser unida a *master*. En caso de encontrarse un *bug* en *master* se generaría una nueva rama *hot-fix* que partiría de la rama *master*, para ser directamente incorporada a la siguiente versión *release* (véase figura 5.2).

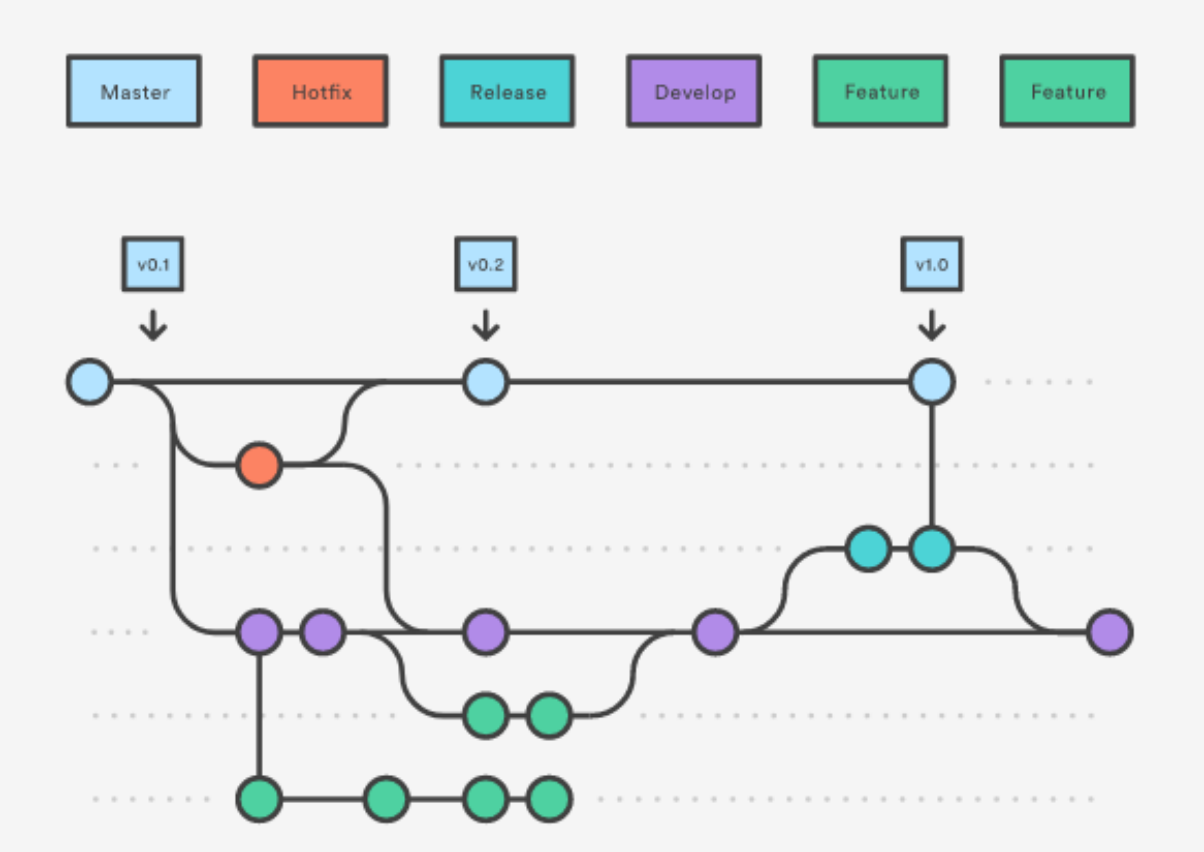

Figura 5.2: esquema git flow

Capítulo 6

Desarrollo de la implementación

Este capítulo describirá el desarrollo de los *sprints* y la implementación de las nuevas funcionalidades descritas con anterioridad en las diferentes secciones, tales como objetivos o planificación temporal. Partiremos de un *backlog* con las historias de usuario. Algunas de estas historias contienen una serie de subtareas para asegurar que se cumple la lógica de negocio, y los resultados obtenidos a lo largo del *sprint* son los esperados.

6.1. *Backlog*

Las tareas contenidas en la pila del producto fueron:

- **GOC4TIC-123:** Como administrador quiero poder editar los firmantes de la reunión de manera dinámica.
- **GOC4TIC-141:** Como administrador quiero poder asignar un sustituto de un firmante, si no asiste a la reunión.
- **GOC4TIC-163:** Como administrador quiero poder gestionar las personas que tienen acceso a la aplicación desde la parte privada de la aplicación.
- **GOC-381:** Como administrador quiero poder realizar diligencias sobre las actas ya cerradas y firmadas.
- **GOC-593:** Como administrador quiero poder duplicar reuniones para poder volver a convocarlas.
- **GOC-545:** Como administrador quiero poder gestionar desde un solo punto los adjuntos de un punto de orden del día.
- **GOC4TIC-143:** Como usuario quiero poder acceder a la documentación de las reuniones sin ser miembro.

6.2. Sprint 1

6.2.1. Planificación del *sprint*

Esta sección muestra las historias de usuario elegidas del *backlog* para realizar el primer *sprint*. Dentro de cada una de estas historias encontramos subtareas cuyo objetivo es esclarecer el comportamiento de la lógica de negocio de la aplicación bajo diferentes situaciones que podrían acontecer, explicando cómo actuaría la nueva funcionalidad, dada cierta casuística.

6.2.2. GOC4TIC-123

Como administrador quiero poder editar los firmantes de la reunión de manera dinámica (figura 6.1).

- Al cerrar una reunión, es obligatorio que como mínimo haya un firmante de todos los órganos.

- Los firmantes de la reunión han de aparecer en la plantilla del acta.

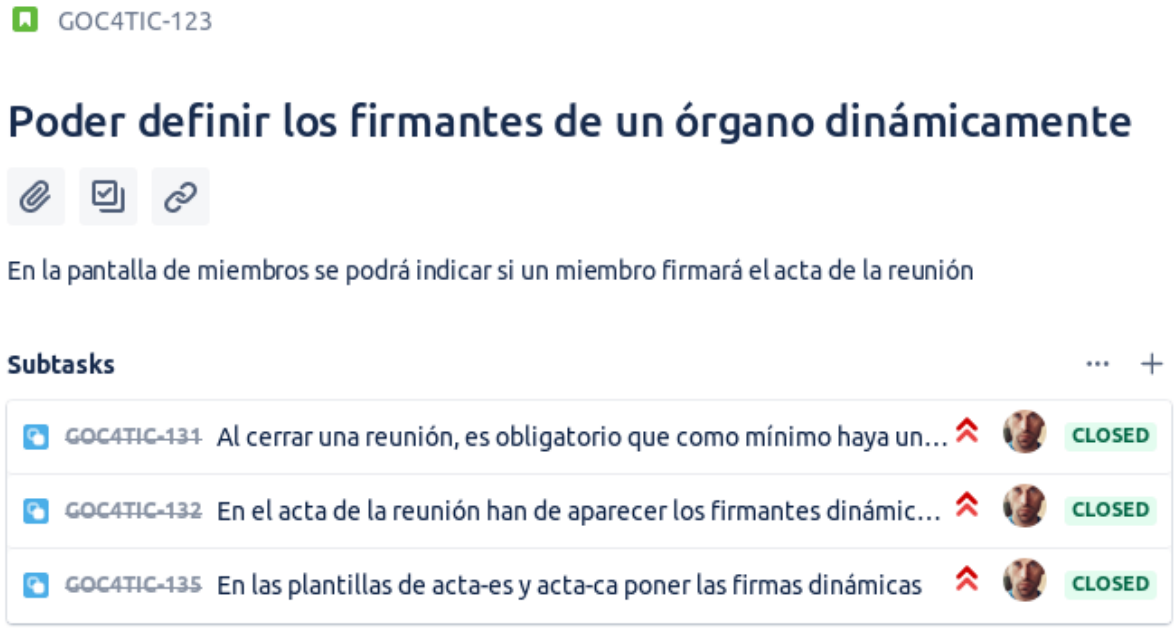

Figura 6.1: GOC4TIC-123

6.2.2.1. Implementación

Modelo de datos: La realización de esta tarea supuso pequeños cambios en base de datos. Concretamente, se añadió un atributo a la entidad Miembros de la reunión y uno al de miembros del órgano. Este cambio supuso la modificación de los DTOs y modelos en consecuencia.

Modificaciones en la vista: En la vista, supuso cambios en el modelo de datos que el *grid* (tipo de vista en forma de tabla de la librería Extjs) de miembros recibía para aceptar la nueva propiedad añadida en el DTO y que debía ser mostrada al usuario (figura 6.2). Supuso un cambio en el *grid* perteneciente al menú miembros y otro cambio en el *grid* que muestra los miembros asistentes a una reunión en un modal perteneciente al formulario de la reunión (6.3).

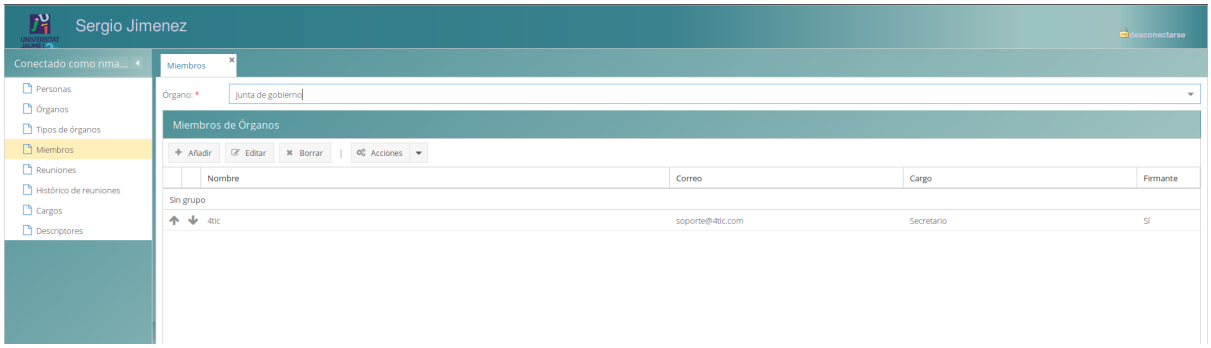

Figura 6.2: *grid* miembros principal

6.2.2.2. Resultado

Al finalizar la tarea, el resultado es el que se aprecia en la figura 6.4. La figura muestra un acta con un firmante a modo de ejemplo. La lógica de negocio realiza una serie de comprobaciones y contiene excepciones personalizadas. Una comprobación a destacar sería el hecho de que debe existir al menos un firmante para poder realizar el cierre del acta. Esta tarea incrementa el producto inicial, permitiendo la adición de tantos firmantes como considere o necesite el editor de la reunión, generando de forma dinámica dos columnas en las que se insertan los firmantes junto con sus cargos en los pdf's.

6.2.3. GOC4TIC-141

Como administrador quiero poder seleccionar un sustituto de un firmante si este no puede asistir a la reunión (figura 6.5).

- Devolver el firmante (persona y cargo) a las plantillas. Si el firmante no asiste, hay que enviar al suplente, si está asignado.

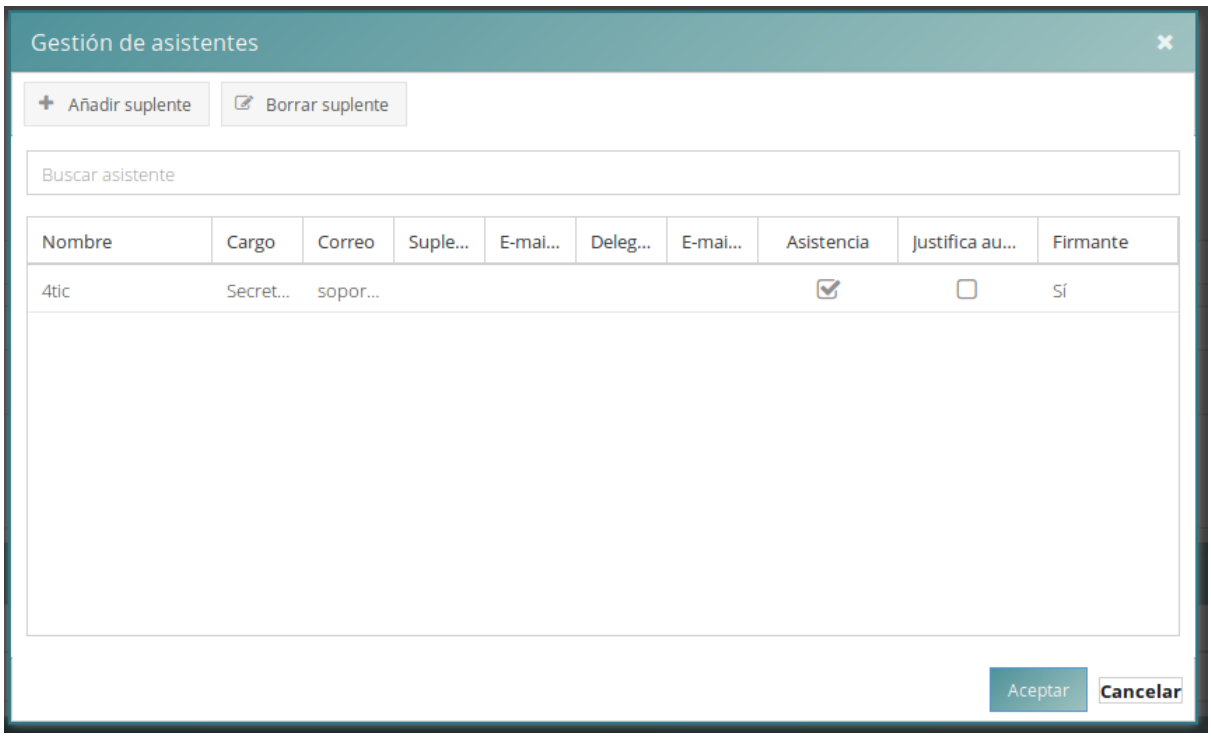

Figura 6.3: *grid* miembros en modal

- Si el firmante no asiste y el suplente no está asignado, devolver error.
- Si el suplente es un miembro del órgano de la reunión, recuperar su cargo para mostrarlo en las plantillas.
- Si el suplente no es un miembro del órgano del miembro que suple, no mostrar su cargo.

6.2.3.1. Implementación

Modelo de datos: En la tabla miembros que muestra la figura 6.6, Miembros de órganos, aparecen las modificaciones que experimentó el modelo de datos durante la implementación de esta tarea. Se añadieron dos atributos a esta entidad, el nombre del suplente y el cargo de este.

Modificaciones en la vista: La tarea no supuso cambios en la vista de la aplicación, solo supuso cambios visibles en las plantillas que la aplicación posee.

6.2.3.2. Resultado

El resultado de esta segunda tarea del *sprint* es una extensión de la tarea inmediatamente anterior (GOC4TIC-123) lo que supuso una modificación de la lógica anteriormente implementada para permitir la asignación de un suplente a un firmante.

LOCAL | Actas de reuniones

ACTA DE REUNIÓN

Reunion Test 4275n

Fecha: 19/04/2019 09:50

Asistentes

4tic - Secretario
Sergio Jimenez - Rector
Externo - Vocal

Invitados

Sergio Jimenez

Orden del día

1. Reunion Test 4275n

Desarrollo de la sesión

1. Reunion Test 4275n

Firmantes

4tic
Secretario

Figura 6.4: Resultado de la tarea GOC4TIC-123

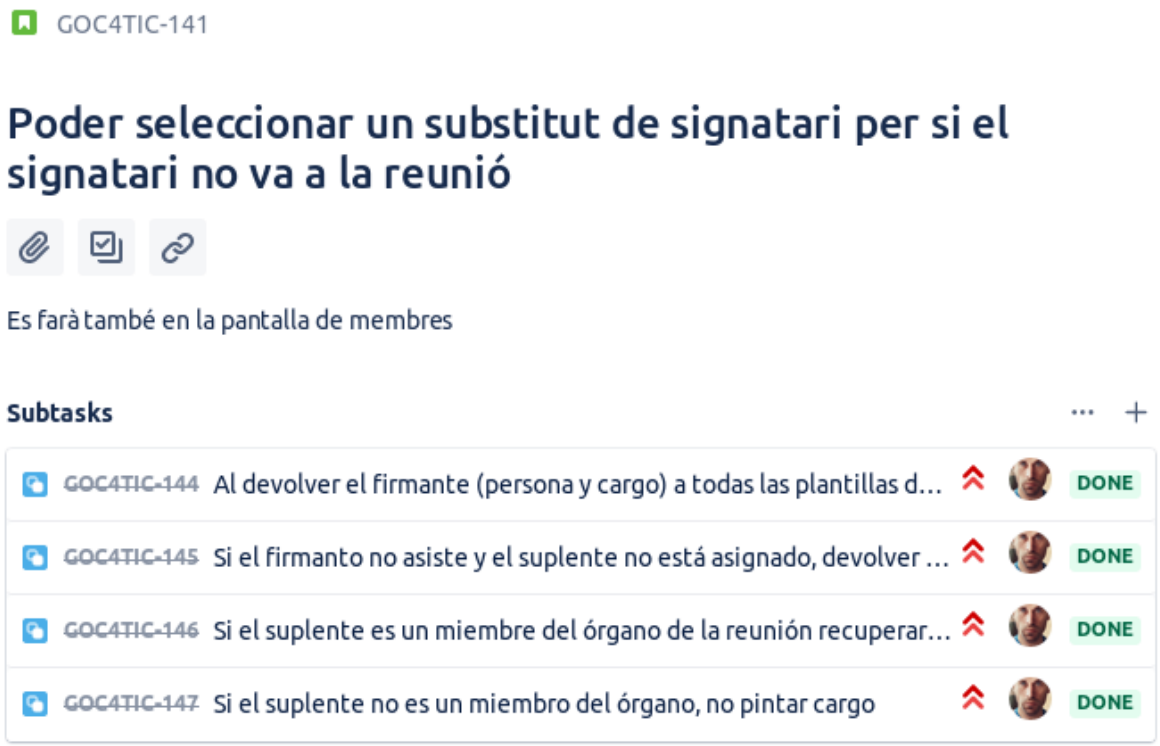

Figura 6.5: GOC4TIC-141

En la figura 6.7(la cual muestra los miembros asistentes a la reunión) observamos cómo un miembro no firmante (Externo) ha sido asignado como suplente de un firmante (4tic). Al ser dos miembros pertenecientes al mismo órgano, el resultado será la aparición de los firmantes Externo (sustituto de 4tic) y Sergio Jimenez, ambos con la aparición del cargo, ya que externo pertenece al mismo órgano de la persona a la que sustituye (figura 6.8).

6.2.4. Comentarios

La previsión final de este *sprint* pasaba por la realización de tres tareas pero la implementación de la tarea GOC4TIC-123 supuso un esfuerzo mayor del esperado, consumiendo mucho más tiempo del previsto debido a diferentes factores. Como principal factor podría destacarse el hecho de ser una primera toma de contacto con el proyecto, un proyecto con miles de lineas de código ya escritas, lo que supuso una demora, pues había que adaptar toda la funcionalidad nueva al código, aprovechar código existente y evitar que el mismo dejara de comportarse de manera correcta. Como segundo factor, la aproximación a nuevas tecnologías utilizadas en el proyecto, como por ejemplo la modificación de la plantilla pdf que utiliza la tecnología Thymeleaf. El desconocimiento de esta tecnología supuso una demora importante, así como el desconocimiento de la librería principal utilizada en la parte de cliente ExtJs. El retraso generado supuso la eliminación de forma provisional de una de las tareas previamente planteadas, GOC-593.

Figura 6.6: GOC4TIC-123

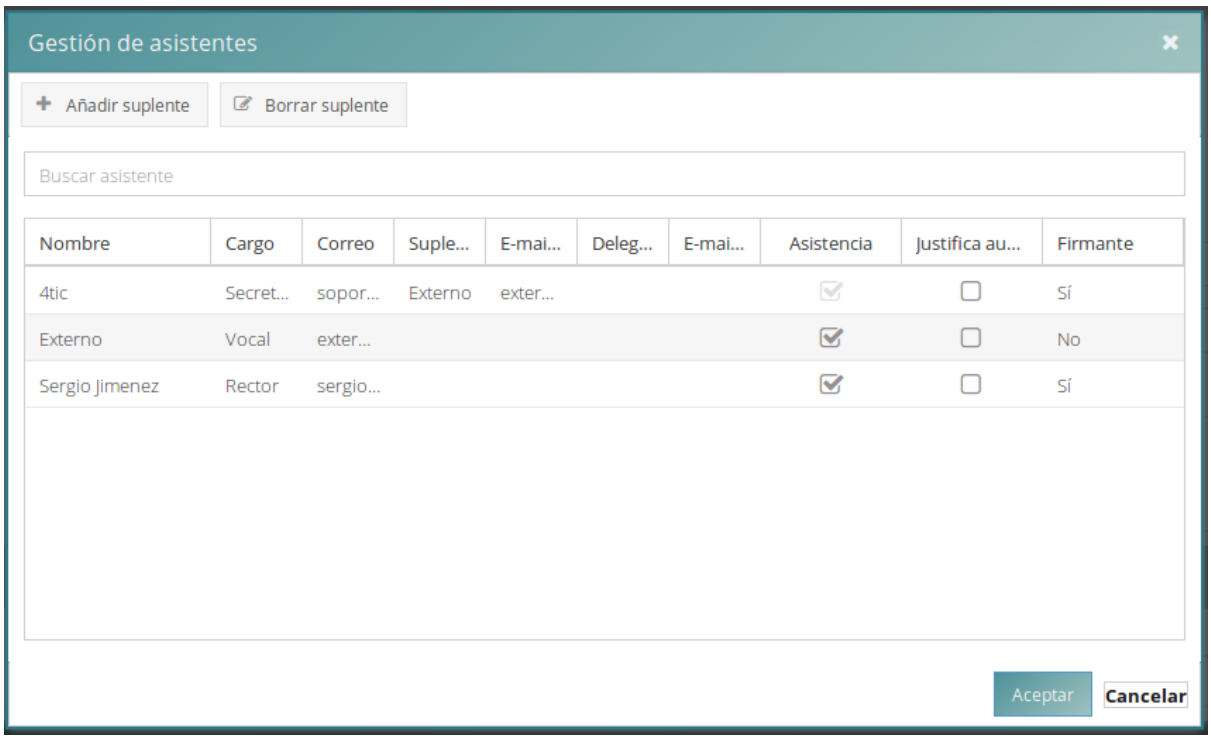

Figura 6.7: Miembros asistentes a la reunión

6.3. Sprint 2

6.3.1. Planificación del *sprint*

Esta sección muestra las historias de usuario elegidas del *backlog* para realizar el segundo *sprint*. Dentro de cada una de estas historias encontramos subtareas cuyo objetivo es esclarecer el comportamiento de la lógica de negocio de la aplicación bajo diferentes situaciones que podrían acontecer, explicando cómo actuaría esta nueva funcionalidad, dada cierta casuística.

6.3.2. GOC4TIC-545

Como administrador quiero poder gestionar desde un solo punto los adjuntos de un punto de orden del día (figura 6.9).

- Si indicamos que el adjunto es un acuerdo, no hará falta especificar nada más. Al ser un acuerdo, deberá enlazarse en el acta y si en el punto del orden del día hemos marcado el check “Publicar acuerdos”, entonces además aparecerá en el buscador público de acuerdos.
- Si indicamos que es un adjunto, entonces podremos habilitar dos opciones (checkbox):
  - Mostrar en ficha: El documento se mostrará en la ficha de la reunión. Así, el convocante puede ocultar/mostrar el documento en el momento en que decida.

LOCAL | Actas de reuniones

ACTA DE REUNIÓN

Reunion Test 4275n

Fecha: 19/04/2019 09:50

Asistentes

4tic - Secretario
Asiste como suplente Externo
Sergio Jimenez - Rector
Externo - Vocal

Orden del día

1. Reunion Test 4275n

Desarrollo de la sesión

1. Reunion Test 4275n

Firmantes

Externo Vocal	Sergio Jimenez Rector

Figura 6.8: Acta resultado de la tarea GOC4TIC-141

- Publicar en el buscador de acuerdos: Se mostrará un enlace al documento en el buscador de acuerdos.

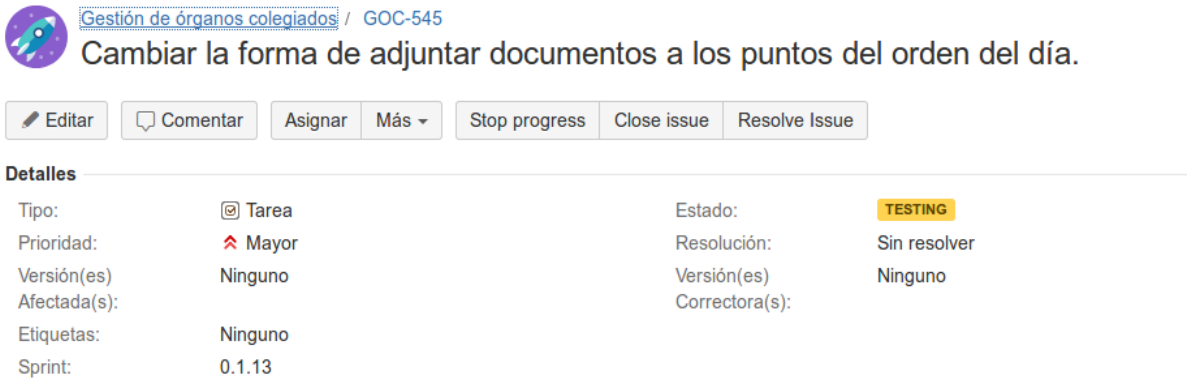

Figura 6.9: GOC4TIC-545

6.3.2.1. Implementación

Modelo de datos: La tarea aprovecha DTOs existentes en la aplicación, moviendo las modificaciones a la capa de servicio donde se generan modelos combinados del tipo documentación, así como nuevos modelos para nuevos elementos de la vista, en la parte del servidor y también nuevos modelos en la parte del cliente en consonancia con los del servidor.

Modificaciones en la vista: Se creó toda una vista nueva, un modal (figura 6.10), con diferentes elementos, un *grid* y un formulario, desde la parte del *grid*, se posibilita la descarga y el borrado de ficheros y se muestra información relevante del fichero. La adición de elementos al *grid* se realiza desde el formulario situado en la parte inferior que permite añadir también una descripción al fichero. Además una vez subidos estos ficheros admiten modificaciones en su estado tales como, edición de su descripción o la modificación de su nivel de visibilidad a nivel del buscador de acuerdo o ficha.

6.3.2.2. Resultado

Tal y como se aprecia en la figura 6.10 (Vista modal principal de la tarea GOC-545 en la página 41), el resultado de la tarea es un incremento que permite la adición de acuerdos y documentación a un punto de orden del día desde una ventana modal con múltiples opciones, que van desde la descarga, el borrado o la modificación a la adición.

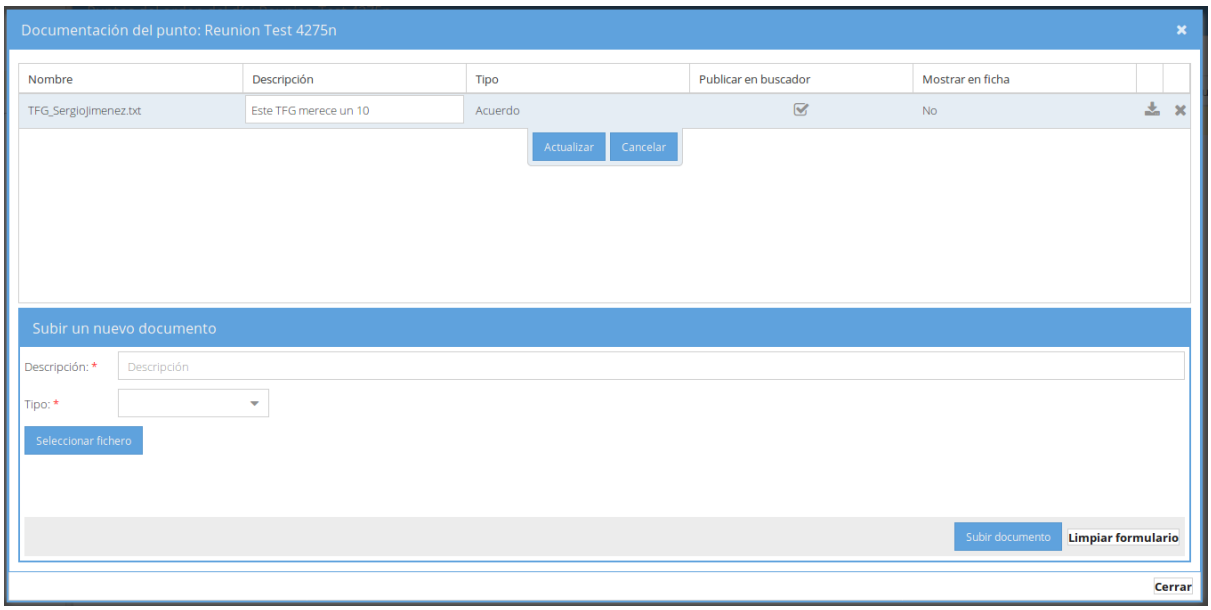

Figura 6.10: Vista modal principal de la tarea GOC-545

6.3.3. GOC4TIC-381

Como administrador quiero poder realizar diligencias sobre las actas ya cerradas y firmadas (figura 6.11).

- Al reabrir una reunión, ésta se puede editar de forma normal
- Al abrir una reunión, se debe almacenar una auditoría
- Al editar un punto de orden del día de una reunión reabierta, se debe añadir un flag para saber que ha sido editado
- Al reabrir, todos los *flags* de la reunión se inicializan a estado de no modificación.
- Al añadir un punto de orden del día nuevo, se añade un *flag* que lo marca como nuevo.
- Si se editan o añaden puntos a una reunión no reabierta, el atributo añadido para indicar modificaciones, no sufre modificación alguna.

6.3.3.1. Implementación

Modelo de datos: La implementación de esta segunda tarea del *sprint* supuso una modificación de varias entidades de la aplicación, tanto a nivel de reunión, donde se crea un atributo para indicar si las reuniones han sido reabiertas, como a nivel de puntos de orden del día, donde se añade también un atributo para indicar si han sido o no editados.

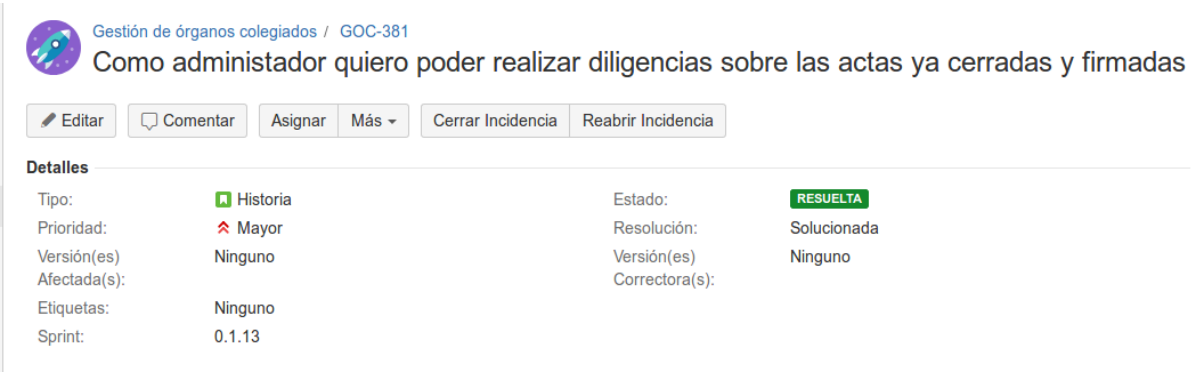

Figura 6.11: Descripción tarea GOC-381

Modificaciones en la vista: La vista no sufrió en este caso grandes modificaciones ya que la mayor parte de la tarea transcurre a nivel de servidor, donde se modifican modelos y consultas para conseguir el objetivo. A nivel de vista se añade un botón, que permite que una reunión pase del estado completada, al estado reabierta. Esto consigue que la reunión se traslade al *grid* de reuniones desde el de histórico de reuniones, donde se encuentran el resto de reuniones y se pueda editar de forma normal.

6.4. Sprint 3

6.4.1. Planificación del *Sprint*

Esta sección muestra las historias de usuario elegidas del *backlog* para realizar el último *sprint* previsto para este documento. Dentro de cada una de estas historias encontramos subtareas cuyo objetivo es esclarecer el comportamiento de la lógica de negocio de la aplicación bajo diferentes situaciones que podrían acontecer. Es decir como actuaría esta nueva funcionalidad dada cierta casuística.

6.4.2. GOC4TIC-163

Como administrador, quiero poder gestionar las personas que tienen acceso a la aplicación desde su parte privada (figura 6.12).

- Se debe añadir un nuevo ítem, personas, en el menú principal en que se permita la gestión de la entidad personas. El *grid* debe contener dos vistas, una vista general, con dos *grids* uno para mostrar las personas y otro para mostrar el detalle de los roles de la persona seleccionada. Al modificarse cualquier dato de una persona, el cambio deberá ser replicado en base de datos y donde sea necesario (ejemplo modificar el nombre o correo electrónico en miembros si es requerido)

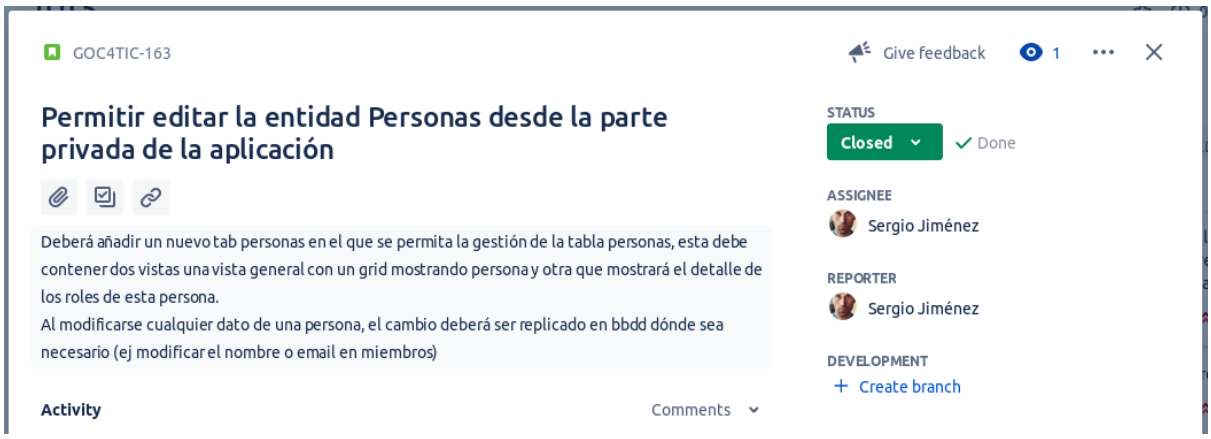

Figura 6.12: GOC4TIC-163

6.4.2.1. Implementación

Modelo de datos: Esta tarea supuso una modificación de una de las principales entidades de la aplicación, en la que se comprueba qué usuarios pueden o no acceder a la aplicación. Supuso también la modificación de muchas de las consultas a la base de datos, que utilizaban esta entidad, se creó para su consulta, una nueva vista para ocultar algunos de los atributos y dar una capa de seguridad mayor, evitando así posibles indebidas inserciones en la entidad o el acceso a datos privados. Toda la lógica de esta tarea se trasladó a un módulo externo, quedando los módulos principales goc-base y goc-core, solo para la realización de peticiones a este modulo externo llamado goc-external, donde otro resource recibe las peticiones y tras realizar la lógica adecuada, devuelve una respuesta.

Modificaciones en la vista: Se creó toda una vista nueva (figura 6.13), desde el menú principal, un *grid* que muestra la información de la nueva vista y permite la adición directamente desde un botón en el *grid* sin tener que utilizar un repositorio externo y un botón para la adición desde repositorio externo (activo solo si la universidad tiene repositorio externo configurado) que abre un nuevo modal para la búsqueda de personas en el repositorio externo a través de peticiones a dicho servicio (figura 6.14). También se permite el borrado, que implicaría la denegación de acceso a la plataforma y la edición del nombre de usuario. Se muestra también una columna en la que aparece si el usuario posee el rol de administrador, siendo éste no modificable.

6.4.2.2. Resultado

El resultado de la tarea difirió del previsto pues la descripción dada fue evolucionando tras diversas aclaraciones con el propietario del producto, en este caso representado por el gerente de la empresa 4 TIC. Se partió desde el planteamiento de un doble *grid* en una misma ventana para mostrar toda la información de roles y la capacidad de editar estos. No obstante por motivos de seguridad y visibilidad de la información, se acabó modificando la idea principal hacia la finalmente implementada, con un solo *grid* que contiene en una de sus columnas, si la persona posee el rol administrador o no. También se implementó, por sugerencia propia, el botón de

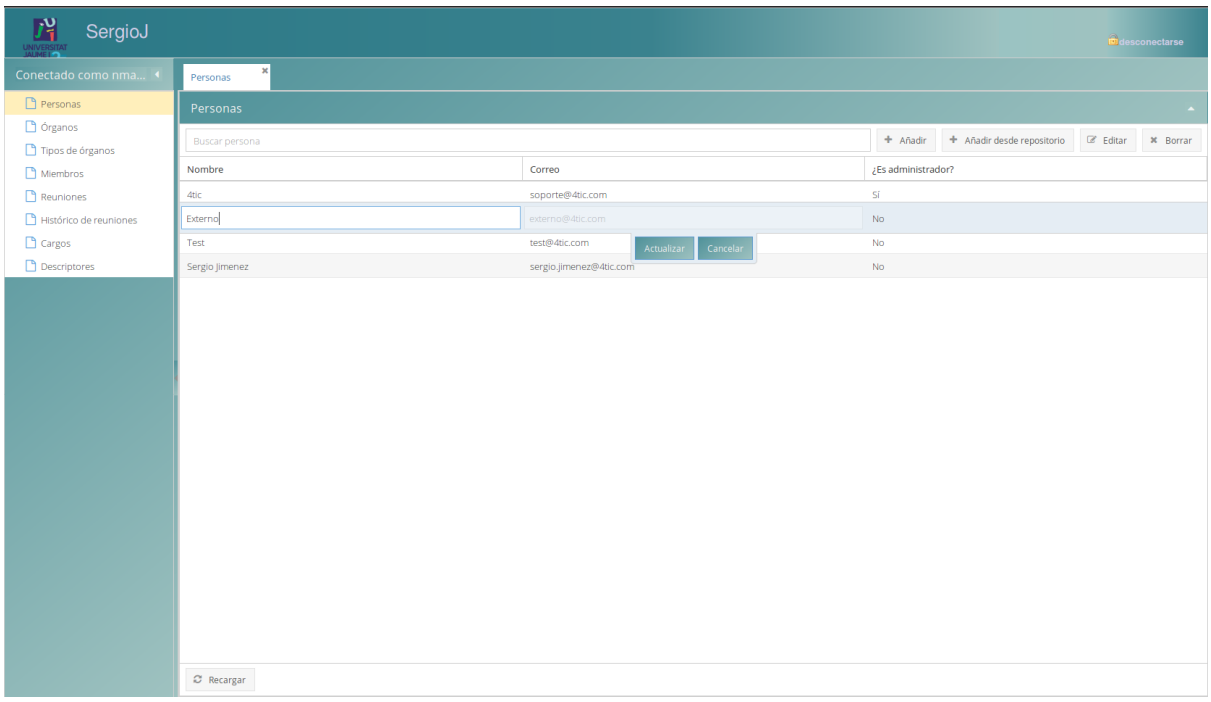

Figura 6.13: Vista principal de la tarea GOC4TIC-163

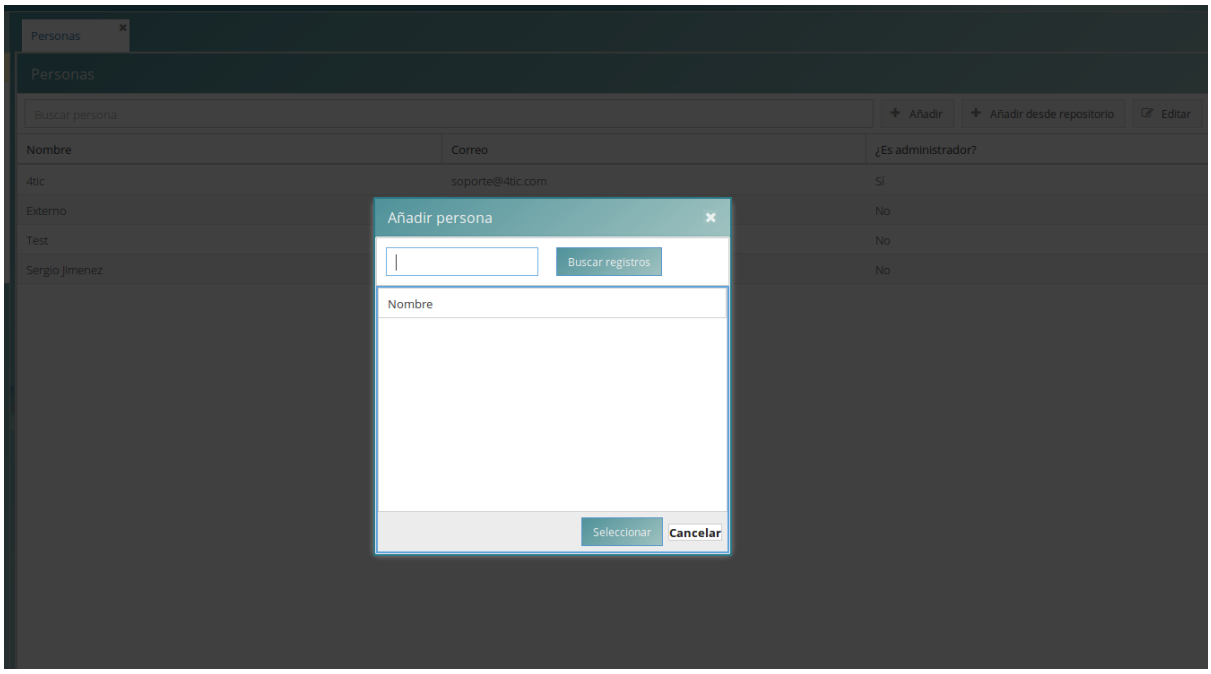

Figura 6.14: Vista modal principal de la tarea GOC-163

añadir desde el repositorio, que se muestra en base a la propiedad global de la aplicación activa, si la aplicación tiene configurado un repositorio externo.

6.4.3. GOC4TIC-143

Como usuario quiero poder acceder a la documentación de las reuniones sin ser miembro (figura 6.15).

- En la pantalla de reuniones, añadir una acción extra para poder añadir personas externas, con el menú "Dar acceso a personas externas a la reunión"
- Guardar auditoría de quién proporciona y retira acceso a estas personas
- Las personas externas han de poder acceder a la ficha de la reunión concreta a la que tengan acceso
- Guardar auditoría cuando una persona acceda a la ficha de la documentación
- Guardar auditoría cuando una persona descarga documentación de la ficha

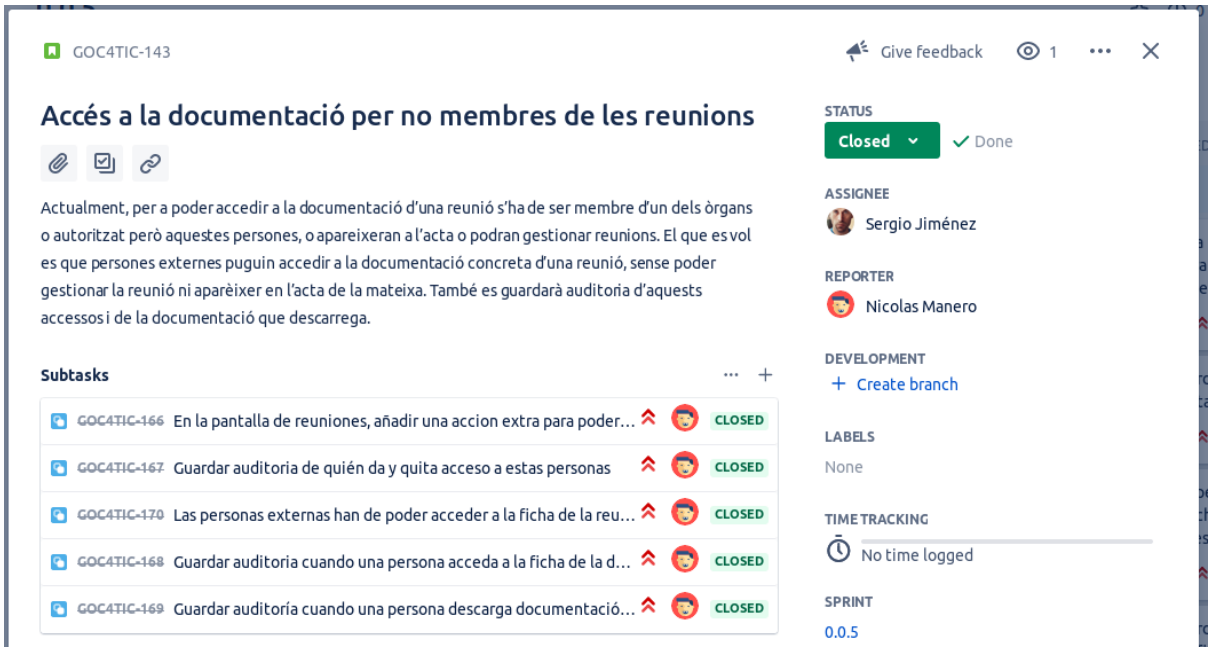

Figura 6.15: Descripción tarea GOC-143

6.4.3.1. Implementación

Modelo de datos: La implementación de esta segunda tarea del *sprint* supuso la creación de una nueva entidad llamada externos, que contenía los datos necesarios para el externo, el id de la reunión a la que se le otorga acceso, su nombre y dirección de correo electrónico. También se

añade una nueva enumeración que describe un tipo por cada acción que puede ser almacenada en la nueva entidad diligencias. Así como la adición de los diferentes modelos y *stores* en la vista para el manejo de los nuevos modelos de datos en la parte del cliente.

Modificaciones en la vista: Se creó un botón en el *grid* de las reuniones que daba acceso a este nuevo modal de gestión de acceso a personas externas a una reunión (figura 6.16). El modal contiene un *grid* con la información de las persona que tienen acceso a la reunión sin ser miembros, los botones de adición y borrado de personas en el *grid*. Para la adición se creo un modal que realiza una búsqueda en la entidad personas, implementada en la tarea anterior (figura 6.17).

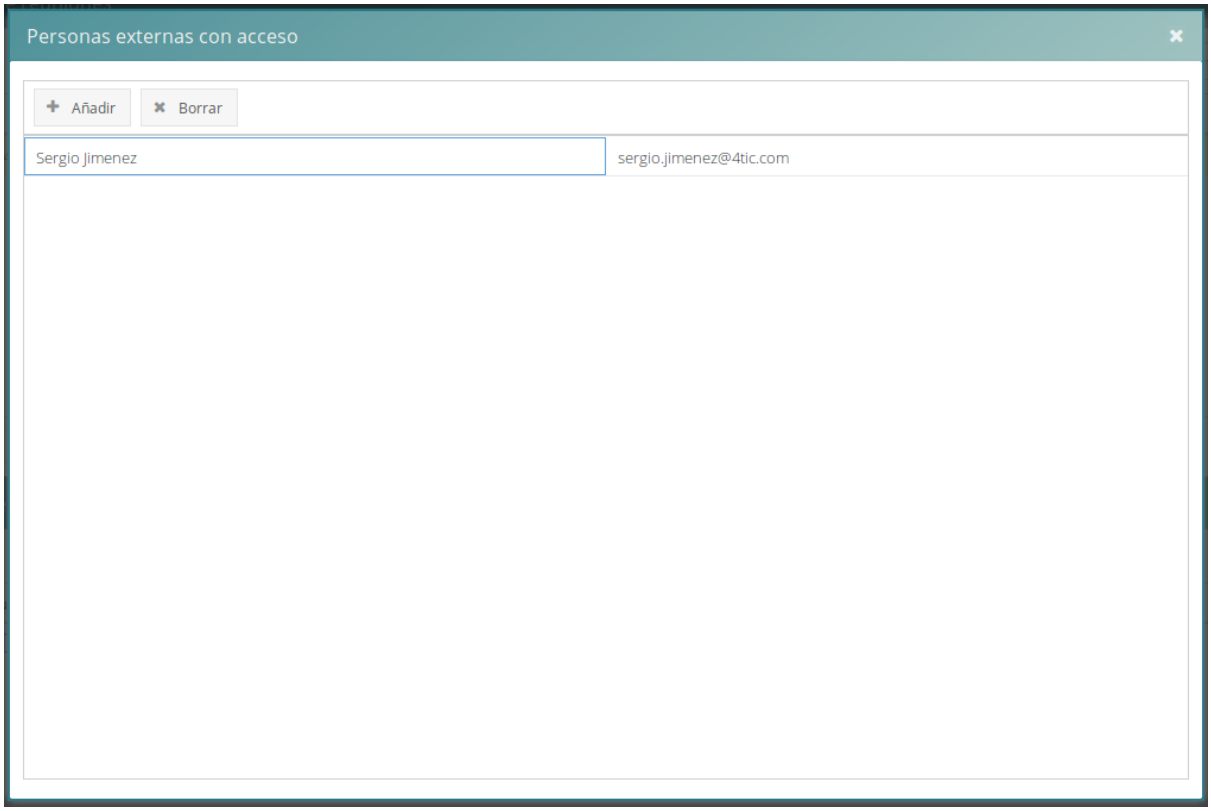

Figura 6.16: Modal de gestión de personas externas con acceso a la reunión GOC-143

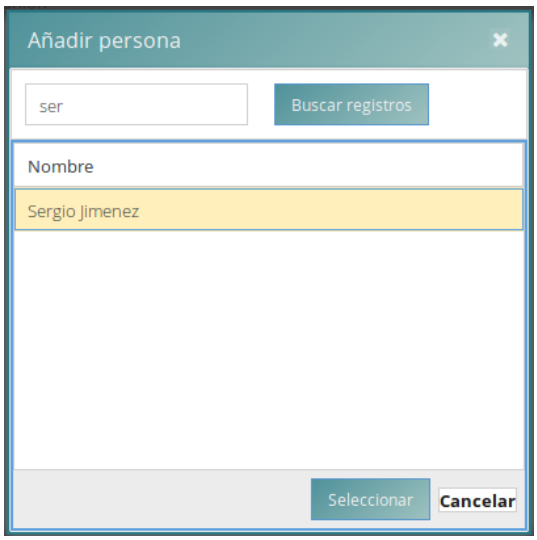

Figura 6.17: Modal de adición personas externas GOC-143

Capítulo 7

Conclusiones

Este documento relata una aproximación al ámbito profesional por parte de un alumno de cuarto curso de ingeniería informática, Sergio Jimenez. Se trata de un TFG que describe cómo se implementa una serie de nuevas funcionalidades a un proyecto *open-source*. Estas funcionalidades se integraron en el producto final quedando disponibles y en funcionamiento en la plataforma real.

A nivel personal, esto supone una gran primera experiencia profesional, una aproximación a la profesión de ingeniero informático desde la posición de un desarrollador *full-stack*. Supone un gran crecimiento personal y un sentimiento de realización al ofrecer la posibilidad de ver el trabajo añadido a un producto real, siendo este trabajo constantemente juzgado y analizado por otros profesionales con una dilatada experiencia en el sector.

En cuanto a objetivos de aprendizaje, las previsiones eran buenas, pero superaron todas las expectativas. No solo se aprendieron nuevas tecnologías y metodologías de trabajo, sino que también se mejoraron y comprendieron muchos de los conocimientos que se adquirieron durante los cuatro años que el grado supone.

De forma particular y a destacar, se han mejorado conocimientos de múltiples tecnologías, como en el uso de los lenguajes de programción Javascript y Java, el uso de librerías como ExtJs, Cypress, JPA, Spring, Jersey etc.

La creación de este documento supuso la aplicación combinada de lo que se aprendió en múltiples materias del grado, por lo que la redacción y la preparación fueron una actividad enriquecedora, desde el principio.

El transcurso de los *sprints* fue un éxito y supuso la integración de interesantes funcionalidades para el software. En estos momentos algunas de estas funcionalidades ya están siendo usadas a diario por parte de los usuarios de la aplicación.

7.1. Futuras ampliaciones de funcionalidad

La aplicación sigue evolucionando. Muchas de las futuras ampliaciones ya se encuentran en el *backlog* de la aplicación. Añadir búsquedas en el histórico de reuniones, filtrar resultados por órgano y tipo de órgano en reuniones, seleccionar el convocante de la reunión de forma dinámica, o permitir que una reunión sea convocada por un órgano, son algunas de estas ampliaciones. Existe también la tarea que quedó desplazada en la implementación del *sprint* 1, y que permitirá duplicar reuniones desde la parte privada de la aplicación.

Sin duda la aplicación aún espera una notable evolución, en la que sigo trabajando.

Capítulo 8

Bibliografía

Bibliografía

[1] Cecilio Álvarez. *JPA vs Hibernate*. es. Jul. de 2014. URL: https://www.genbeta.com/desarrollo/jpa-vs-hibernate (visitado 19-04-2019).

[2] Atlassian. *Gitflow Workflow — Atlassian Git Tutorial*. en. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow (visitado 20-04-2019).

[3] Atlassian. *Herramienta de gestión de código Git de Atlassian Bitbucket para equipos*. es. URL: https://es-4f4071804890f12e0.getsmartling.com (visitado 20-04-2019).

[4] Atlassian. *Jira — Software de seguimiento de proyectos e incidencias*. es. URL: https://es.atlassian.com/software/jira (visitado 13-04-2019).

[5] Atlassian. *Kanban - A brief introduction*. es. URL: https://es.atlassian.com/agile/kanban (visitado 13-04-2019).

[6] baeldung. *Intro to Querydsl*. en-US. Jul. de 2016. URL: https://www.baeldung.com/intro-to-querydsl (visitado 20-04-2019).

[7] *Git*. URL: https://git-scm.com/ (visitado 20-04-2019).

[8] *Hibernate. Everything data. - Hibernate*. URL: https://hibernate.org/ (visitado 19-04-2019).

[9] *Historias de usuario en Agile Scrum*. es. Mayo de 2016. URL: https://managementplaza.es/blog/historias-de-usuario-en-scrum-agile/ (visitado 29-06-2019).

[10] *Intro to App Architecture — Ext JS 6.0.2*. URL: https://docs.sencha.com/extjs/6.0.2/guides/application_architecture/application_architecture.html (visitado 27-04-2019).

[11] *JAX-RS Servicios RESTFull en Java*. es. Jun. de 2013. URL: https://www.arquitecturajava.com/jax-rs-servicios-restfull-en-java/ (visitado 19-04-2019).

[12] *JUnit 5 User Guide*. URL: https://junit.org/junit5/docs/current/user-guide/ (visitado 19-04-2019).

[13] *Log4j – Apache Log4j 2 - Apache Log4j 2*. URL: https://logging.apache.org/log4j/2.x/ (visitado 05-05-2019).

[14] *Monthly and yearly plans with JetBrains Toolbox*. URL: https://www.jetbrains.com/store/ (visitado 19-04-2019).

[15] Eugen Paraschiv. *Jackson Annotation Examples*. en-US. Jun. de 2015. URL: https://www.baeldung.com/jackson-annotations (visitado 20-04-2019).

[16] *Product Packages and Pricing*. en-US. Feb. de 2017. URL: https://www.sencha.com/pricing/ (visitado 19-04-2019).

[17] *Project Jersey*. en. Page Version ID: 892187079. Abr. de 2019. URL: https://en.wikipedia.org/w/index.php?title=Project_Jersey&oldid=892187079 (visitado 19-04-2019).

[18] *Propietario del producto - Scrum Manager BoK*. URL: https://www.scrummanager.net/bok/index.php?title=Propietario_del_producto (visitado 17-04-2019).

[19] *Pull Request Tutorial by yangsu*. URL: https://yangsu.github.io/pull-request-tutorial/ (visitado 29-06-2019).

[20] *rest - What are the DAO, DTO and Service layers in Spring Framework?* URL: https://stackoverflow.com/questions/35078383/what-are-the-dao-dto-and-service-layers-in-spring-framework (visitado 01-05-2019).

[21] *Sencha Ext JS - Comprehensive JavaScript Framework and UI Components*. en-US. Abr. de 2015. URL: https://www.sencha.com/products/extjs/ (visitado 20-04-2019).

[22] *Spring Projects*. URL: https://spring.io/projects (visitado 20-04-2019).

[23] *SQL Tutorial*. URL: https://www.w3schools.com/sql/ (visitado 29-06-2019).

[24] *What is JavaScript?* es. URL: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript (visitado 29-06-2019).

[25] *Why Cypress?* en. URL: https://docs.cypress.io/guides/overview/why-cypress.html (visitado 19-04-2019).