ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Herramienta de Gestión de Trabajos de Fin de Grado

(Final Year Projects Management Tool)

Para acceder al Título de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autor: Jorge Orallo Rodríguez

Junio - 2018

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Jorge Orallo Rodríguez
Director del TFG: José Luis Crespo Fidalgo
Título: "Herramienta de Gestión de Trabajos de Fin de Grado"
Title: "Final Year Projects Management Tool"
Palabras clave: Aplicaciones Web, Aplicaciones Full-Stack, Desarrollo de software, TFG, Proyectos

Presentado a examen el día: 20 de Junio de 2018

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre):
Secretario (Apellidos, Nombre):
Vocal (Apellidos, Nombre):

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente Fdo.: El Secretario

Fdo.: El Vocal Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Índice

1. Introducción

En los actuales estudios de grado, el paso final y previo a la obtención del título consiste en la elaboración de un **Trabajo de Fin de Grado (TFG)**, que se lleva a cabo bajo la supervisión de un profesor de la titulación denominado tutor o director del proyecto. Un TFG debe estar relacionado con las materias impartidas en la titulación, y el director deberá ser un profesor que imparta docencia en esa misma titulación.

Cuando el alumno tiene claro el proyecto sobre el que va a realizar su TFG, deberá presentar una solicitud de Inicio del Trabajo de Fin de Grado. Una vez comenzado el desarrollo del TFG, éste se va resolviendo de forma personalizada entre el tutor y el alumno, mediante tutorías. Es personalizado porque no existe una metodología a seguir que se aplique en todos los casos, sino que depende de las características y circunstancias del proyecto, del alumno y del tutor.

Cuando el TFG se acerca a su finalización, se deben realizar los trámites para la designación de los miembros del tribunal encargado de evaluarlo, así como la fecha, hora y aula elegidos. Al concluir el desarrollo del TFG, el alumno deberá defenderlo ante un tribunal de evaluación compuesto por docentes de su titulación y cuya estructura puede variar de un plan de estudios a otro.

Previo a la defensa del TFG, el alumno deberá hacer llegar a cada miembro del tribunal una copia electrónica de la memoria del TFG, con una semana de antelación como máximo.

En el marco de la asignatura de *Ingeniería Web* surgió la idea de diseñar y desarrollar una aplicación de tipo web que permita gestionar todo este volumen de información de una forma sencilla y accesible a todo el personal de la universidad. Se trata de cubrir todo el ciclo de vida de un TFG, desde que el tutor decide publicarlo hasta que es presentado ante un tribunal para su evaluación.

La idea no solo cuenta con el apoyo del profesorado de la asignatura de Ingeniería Web, sino también del departamento de Transportes y Tecnología de Proyectos y Procesos de la ETSICCP.

Los beneficiados por esta aplicación serían todos aquellos que realicen, oferten o gestionen los trabajos de fin de grado. Los alumnos, por la agilidad que se ganaría en la búsqueda de un TFG con el que terminar el grado; los profesores, por la difusión que podrían tener sus proyectos y la mejora en la comunicación con el alumno; los responsables de una titulación, por la simplificación de los procesos y gestiones necesarias.

Punto de partida

En la ETSIIT de la Universidad de Cantabria, cada departamento tiene un sitio web donde guardar su documentación y ofertar sus Trabajos de Fin de Grado. Al no existir un repositorio común, cuando un alumno se acerca al final de sus estudios, comienza la búsqueda y selección de un proyecto sobre el que realizar su TFG. Esta búsqueda a menudo implica visitar los distintos espacios web de varios departamentos, descartar información no actualizada, y desplazarse “puerta por puerta” preguntando a los profesores.

A partir de ahí, el ciclo de vida de un TFG tiene numerosos pasos y documentación asociados. La gestión se realiza de forma manual y analógica y, con el paso del tiempo, puede acumularse y/o perderse. Surge la necesidad de adoptar un sistema que permita digitalizar este proceso, haciéndolo más accesible y, sobre todo, permitiendo registrar todos los pasos en un soporte informático para su posterior consulta o estudio.

En muchos casos, la documentación guardada no refleja la realidad en cuanto a las fechas de presentación de cada documento. En ocasiones, los formularios de inicio y final de TFG se presenten en un intervalo breve de tiempo, y cercano a la conclusión. Esto no representa de forma correcta la situación, puesto que, al no disponer de las fechas de inicio reales, no se pueden obtener métricas fiables.

Nuestro punto de partida es, por tanto, un sistema manual, descentralizado, que no registra eventos más allá de la presentación física de los papeles pertinentes en secretaría.

2. Objetivos

El objetivo que se persigue es agilizar todos los procesos relativos a la oferta y búsqueda de trabajos de fin de grado, así como proporcionar una vía de comunicación y seguimiento durante el tiempo de vida del mismo. También se pretende desarrollar un medio digitalizado para la gestión de los tribunales de evaluación, y ofrecer una nueva interfaz para la exploración del histórico de trabajos que permita obtener información y estadísticas sobre todos los Trabajos de Fin de Grado.

Las funcionalidades propuestas se pueden dividir en tres grupos según el tipo de usuario que se vería beneficiado por ellas:

2.1 Desde el punto de vista de un alumno:

- La posibilidad de buscar todos los proyectos disponibles basándose en su titulación o plan de estudios.
- La comunicación con el director de cada proyecto, así como la posibilidad de destacar los proyectos que le interesan y, llegado el caso, solicitar al director que le asigne dicho proyecto.
- Una vez que el alumno esté trabajando en un proyecto, proporcionar una vía de comunicación con el director y de control que permita llevar un registro de los objetivos o hitos alcanzados que el alumno considere importante destacar.
- Cuando el alumno lo considere necesario, podrá cargar los archivos relativos al proyecto (memoria, planos, manuales, etc.) en la aplicación, y enviarlo todo al director para que lo apruebe.

2.2 Desde el punto de vista de un profesor:

- Le permite crear un proyecto para una de las titulaciones o planes de estudio en los que imparta docencia. La información del proyecto incluye el título, descripción, área y palabras clave. Al ser el responsable de ese proyecto también tiene poder para editar cualquiera de los campos anteriores o incluso eliminar el proyecto.
- Como director del proyecto podrá ver qué alumnos han indicado su interés en el proyecto, y si han solicitado su asignación, y de ser así, el director tiene la opción de asignárselo a un alumno y dar por iniciado el proyecto.
- Acceso a la interfaz de seguimiento de sus proyectos activos. Esta interfaz, similar a la que ve el alumno, permite comunicarse con el alumno y ver los hitos que éste ha guardado.
- Confirmar que el proyecto realizado por el alumno está listo para presentarse, y dar el paso a la designación del tribunal.
- En caso de que el profesor haya sido designado miembro de algún tribunal, le permite ver el proyecto que se va a presentar y la información sobre el tribunal (fecha, hora, lugar), y posteriormente, calificar el proyecto.

2.3 Desde el punto de vista de un jefe de estudios o responsable de titulación:

- Si el usuario con el rol de jefe de estudios o responsable también imparte docencia en la titulación, tendrá acceso a todas las funciones que corresponden al rol de profesor.
- Función de designación de tribunales, esto es, podrá formar el tribunal para un proyecto cumpliendo con los parámetros establecidos para la titulación.
- Vista general de todos los proyectos, que mostrará información filtrada de todos los proyectos pertenecientes a la titulación de la cual es responsable, independientemente del estado (en curso o finalizado) en el que se encuentren.

En definitiva, la implantación de este nuevo sistema digital mejoraría de forma significativa los trámites necesarios en el ciclo de vida de un trabajo de fin de grado y puede ofrecer estadísticas muy útiles para la dirección de cada plan de estudios.

3. Arquitectura del sistema

3.1 Formato de Aplicación Web

Una aplicación se define como un programa informático diseñado como herramienta para permitir a un usuario realizar diversas tareas. Por lo general una aplicación requiere la instalación o presencia de software específico en el equipo, y esto resulta en una peor experiencia para el usuario, así como mayor dificultad de uso y la posible incompatibilidad del software.

Una aplicación Web es aquella a la que se accede conectándose a un servidor a través de un navegador web. De esta forma, no requieren de ningún software específico más que un navegador web, que está presente en cualquier tipo de plataforma (PC con indiferencia del SO, Mac, dispositivos móviles, etc.). Por este motivo la popularidad de las aplicaciones web ha crecido en los últimos años, utilizándose en multitud de situaciones y entornos profesionales.

Se ha decidido usar el formato de aplicación web para la realización de este Trabajo de Fin de Grado, por su capacidad para utilizarse desde cualquier dispositivo, la creciente importancia que está ganando este formato, la facilidad de integración con el resto de aplicaciones de la universidad (Moodle, BUC, etc.) y también por las posibilidades de ampliación y mejora que ofrece de cara al futuro.

3.1.1 Estructura de una aplicación web

Una aplicación web se compone principalmente de dos partes: El *Front-end* y el *Back-end.*

Fig1. Front End y Back End

3.1.2 Front-End

Es la cara visible de la aplicación web, la parte del software que interactúa con el usuario. Se ejecuta en el lado del cliente (*client side*) y *se* compone de distintas tecnologías, siendo Html,

Css y Javascript las más representativas. La función de esta capa es la de presentar los datos de forma correcta, legible y útil, y recoger las acciones del usuario para enviarlas al *back-end.*

Fig 2. Tecnologías de Front End

3.1.3 Back-End

Comprende todo el funcionamiento interno de la aplicación y se ejecuta en el servidor (*server side*). En este ámbito existen muchas tecnologías aplicables, como por ejemplo Java, PHP, o Ruby, y SQL como motor de base de datos. La función de esta capa es la de obtener los datos solicitados por el front-end o realizar las acciones necesarias sobre la base de datos o el servidor.

Fig 3. Servidor donde se ejecuta el Back End

3.2 Modelo – Vista – Controlador (MVC)

Durante el desarrollo del proyecto se ha aplicado el paradigma de Modelo – Vista – Controlador (**MVC**). MVC es un patrón de diseño que se fundamenta en la división del código en capas con diferente funcionalidad. La idea es que cada capa se encargue de una función en particular, de forma que el código esté mejor estructurado, sea más legible, y facilite la depuración de errores y los desarrollos futuros.

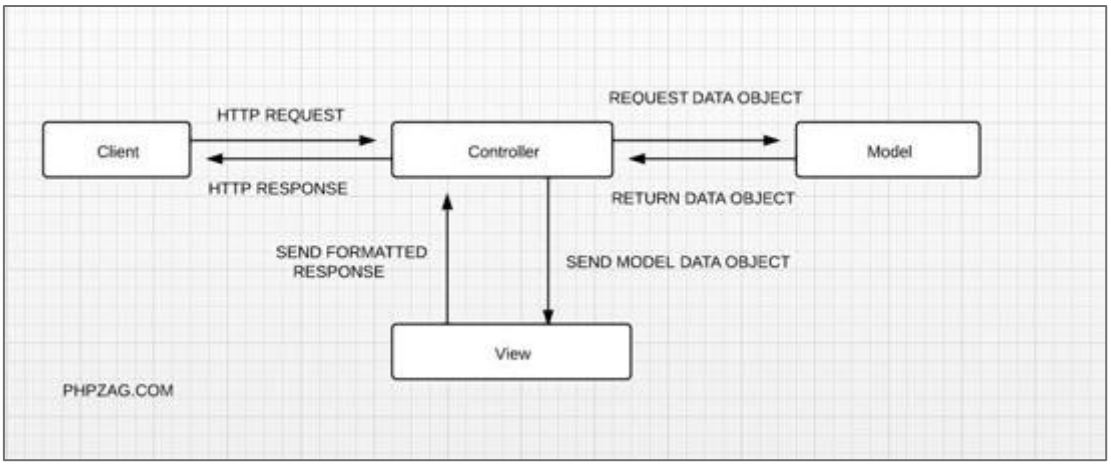

Fig 4. Relaciones entre los componentes de un patrón MVC

El patrón establece una división en tres capas:

- Modelo: Corresponde a la lógica del negocio, la entrada-salida de datos. Es la capa que se comunica con la base de datos y realiza las operaciones de búsqueda, inserción, modificación o borrado de datos.
- Vista: Se encarga de interactuar con el usuario. Muestra la información que devuelve la base de datos en un formato legible, y recoge las acciones del usuario para hacer nuevas peticiones.
- Controlador: Es el nexo entre las dos capas anteriores. Responde a las acciones de la vista y hace llamadas al modelo para ejecutar las peticiones y recopilar los datos solicitados. Posteriormente devuelve a la vista los resultados que le proporciona el modelo.

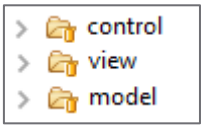

Fig 5. Estructura de archivos en esquema MVC

3.2.1 Comunicación entre la vista y el controlador

La comunicación entre la vista y el controlador se puede llevar a cabo de diversas formas. En esta aplicación, todo el flujo de información entre vistas y controladores se realiza mediante llamadas *AJAX*. Una llamada *AJAX, Asynchronous JavaScript And XML,* es una función de javascript y soportada por todos los navegadores web, que realiza una petición vía HTTP a una ruta o dirección determinadas. Permite enviar datos al servidor, recibir una respuesta, y modificar la información de la vista, todo de forma asíncrona. Esta capacidad de funcionar de forma asíncrona, es decir, que el flujo de ejecución continúe mientras se espera la respuesta del servidor, consigue que las actualizaciones sean totalmente fluidas y transparentes al usuario.

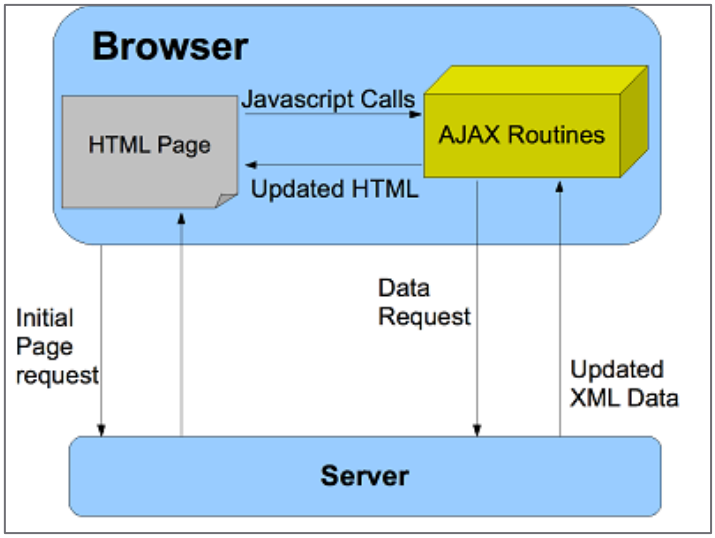

Fig 6. Esquema de una llamada AJAX

3.3 Tecnologías elegidas

Para realizar este proyecto se han elegido las tecnologías frecuentes de *front-ent*, HTML5, CSS3, JavaScript y su framework jQuery. Para la parte *back-end*, se ha optado por utilizar PHP por la rapidez y estabilidad que proporciona, así como la compatibilidad con cualquier servidor, y la facilidad de instalación inicial y de despliegues con actualizaciones de software. Para almacenar todos los datos se ha utilizado una base de datos MySQL. Sencilla y potente, es una de las mejores opciones para este caso.

3.4 Herramientas usadas para el desarrollo

Para el desarrollo del proyecto se han usado diversas herramientas o entornos de desarrollo integrado (IDE) muy presentes en el ámbito profesional. Un IDE se define como una aplicación informática que proporciona servicios integrales para facilitar al programador el desarrollo de software. En este caso, el IDE empleado ha sido Eclipse en sus versiones Mars y Neon. Eclipse es uno de los IDE más utilizados en el mundo y cuenta con multitud de plugins que permiten añadirle nuevas funcionalidades o mejorar y personalizar las que ya tiene.

Una de las herramientas más importantes en el desarrollo de software es el conocido como *debugger* o depurador, que permite analizar el flujo de ejecución de un código, así como visualizar el valor de sus variables internas mientras se ejecuta línea a línea. A día de hoy se hace imposible desarrollar cualquier aplicación mínimamente compleja sin el uso de un debugger. Se ha empleado uno de los principales debuggers para PHP, conocido como *XDebug*. Consiste en una extensión que se instala junto con el servidor PHP y un plugin para eclipse.

Para la gestión y configuración de la base de datos MySql he utilizado la herramienta MySql Workbench 6.3 en su versión *Community* bajo licencia *open source*.

Una aplicación web de este tipo requiere tanto de un servidor HTTP + PHP como de un servidor SQL debidamente instalados para funcionar. Por motivos de rapidez de configuración y comodidad, se ha usado un entorno de desarrollo conocido como XAMPP que proporciona un servidor web local de Apache con PHP y un servidor local de bases de datos MySQL. Es la alternativa rápida a instalar todos los componentes por separado y su uso es habitual en entornos de desarrollo profesionales.

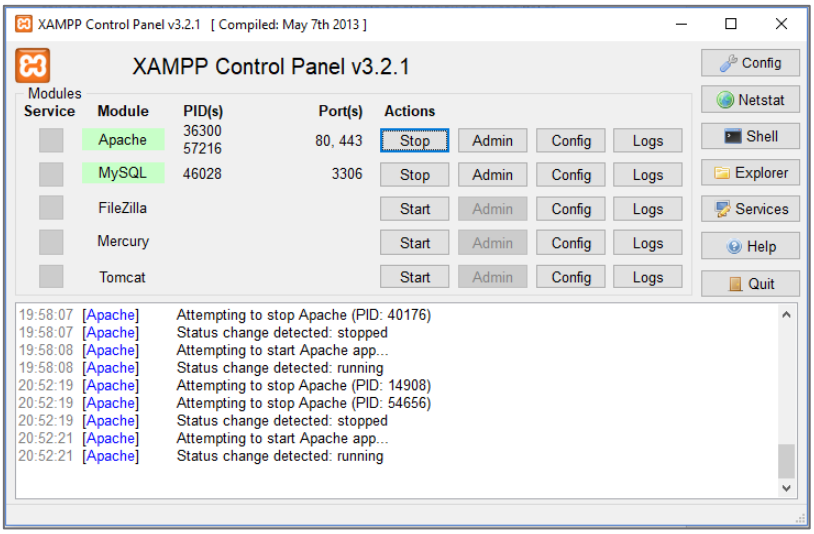

Fig 7. Panel de control de XAMPP

La versión de Apache utilizada es la 2.4, la de MySQL es 5.6.26 y la de PHP es 5.6.12.

3.5 Sistema de control de versiones

El proyecto se ha llevado a cabo a lo largo de un periodo de tiempo muy extenso y usando más de un equipo, y esto ha creado la necesidad de emplear un **sistema de control de versiones**.

Un **Sistema de control de versiones** es un software utilizado para gestionar los cambios que se realizan sobre los distintos elementos que componen un proyecto. Es algo imprescindible en cualquier equipo de trabajo compuesto por varias personas, ya que permite el desarrollo paralelo y la posterior fusión o *merge* de los avances de cada miembro. Aunque el equipo conste de una sola persona, sigue siendo una herramienta muy útil dado que sirve como *backup* o copia de seguridad de nuestro software y además permite llevar un control de los cambios siempre que se realicen subidas diarias o periódicas del progreso. De esta forma se puede volver (*rollback*) a cualquier estado anterior en el caso de que esto fuera necesario, o simplemente saber cuándo se hizo un cambio, qué se cambió exactamente, y quién lo hizo.

Para el desarrollo del proyecto se ha optado por usar un repositorio privado de **Github**, uno de los sistemas más extendidos en el entorno profesional y que además está muy extendido en el desarrollo de software de código abierto. La ruta al repositorio es https://github.com/oralloj/tfg, aunque no se puede acceder sin invitación al ser privado.

El entorno de desarrollo Eclipse incluye por defecto una extensión para el uso de repositorios Git que, además de facilitarnos las últimas versiones del software, permite la sincronización de

archivos que han sido modificados por diferentes personas, de modo que se puede trabajar simultáneamente sobre el mismo fichero sin miedo a perder los avances.

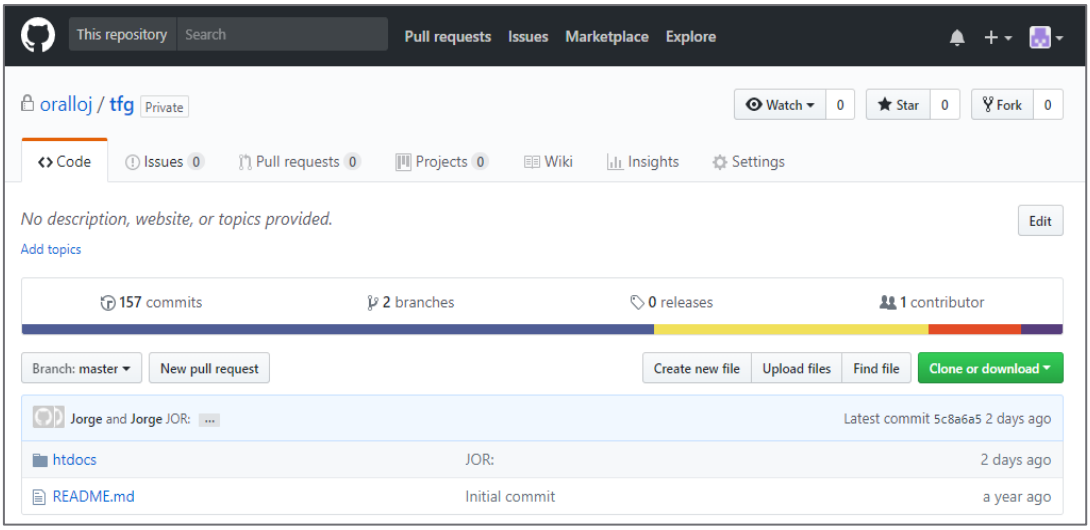

Fig 8. Vista del repositorio Git

4. Modelo de datos

4.1 Modelo relacional

En la definición del modelo de datos de la aplicación podemos encontrar la estructura en la que se organiza la base de datos, los tipos de datos que guarda y las relaciones que se han establecido entre ellos. A continuación, se explicará la función de cada entidad y su estructura.

Glosario de términos SQL que se usarán más adelante:

- **Primary Key (PK):** Clave primaria de una entidad. No puede haber dos iguales.
- **Foreign Key (FK):** Clave foránea de una entidad. Hace referencia a otro campo de otra entidad en el que debe existir el mismo dato.
- **String:** Tipo de dato asociado a una cadena de caracteres.
- **Int:** Tipo de dato asociado a un número entero.
- **Float:** Tipo de dato asociado a un número decimal.
- **Insert:** Función para insertar datos en una entidad.
- **Update:** Función para modificar datos en una entidad.
- **Delete:** Función para borrar datos de una entidad.

Relación de entidades:

Users

Contiene la información básica de todos los usuarios de la aplicación.

Nombre	Tipo	Referencia	Descripción
Userid	String		PK – Identificador único de usuario
Groupid	Int		Identificador de grupo de usuarios
Nombre	String		Nombre del usuario
Apellidos	String		Apellidos del usuario
Correo	String		Mail del usuario
Idi_codigo	String		Idioma de la aplicación

Archivos_proyecto

Registra los archivos relativos a los TFGs que se suben al servidor.

Nombre	Tipo	Referencia	Descripción
Id_archivo	Int		PK – Identificador único de archivo
Id_proyecto	Int	Proyectos_global	Identificador del proyecto
Nombre_archivo	String		Nombre del archivo
Tamano_archivo	Int		Tamaño del archivo en Bytes
Fecha_subida	Date		Fecha de subida
Ruta	String		Ruta al archivo

Áreas

Contiene la relación entre las áreas disponibles y cada plan de estudios.

Nombre	Tipo	Referencia	Descripción
Id_plan	Int	Planes_estudio	Identificador de la titulación
Nombre_area	String		Nombre del área

Calificaciones

Registra las calificaciones otorgadas por cada tribunal.

Nombre	Tipo	Referencia	Descripción
Id_proyecto	Int	Proyectos_global	PK - Identificador del proyecto
Calificacion	Float		Calificación del proyecto
Comentario_tribunal	String		Anotaciones o comentarios
Matricula_honor	String		Indica si se otorga matrícula

Estados_proyecto

Relación de los distintos estados en los que se puede encontrar un proyecto.

Nombre	Tipo	Referencia	Descripción
Id_estado	Int		PK – Identificador único de estado
Idi_codigo	String		Idioma de la descripción
Desc_estado	String		Descripción del estado
Str_estado	String		Código de estado en string

Hitos

Contiene información de los hitos registrados por los alumnos.

Nombre	Tipo	Referencia	Descripción
Id_hito	Int		PK – Identificador único de hito
Id_proyecto	Int	Proyectos_global	Identificador del proyecto
Fecha_hito	Date		Fecha de registro
Desc_hito	String		Descripción del hito

Idiomas

Idiomas existentes en la aplicación.

Nombre	Tipo	Referencia	Descripción
Idi_codigo	String		PK – Identificador único de idioma
Idi_abreviatura	String		Iniciales del idioma
Idi_nombre	String		Nombre del idioma

Login

Nombre	Tipo	Referencia	Descripción

Userid	String	Users	PK – Identificador único de usuario
Password	String		Contraseña
Groupid	Int		Identificador de grupo de usuarios

Entidad auxiliar para el inicio de sesión.

Orden_seleccion_miembros

Entidad auxiliar para la selección de los miembros de un tribunal por criterio de ordenación.

Nombre	Tipo	Referencia	Descripción
Index	Int		PK – Índice
Id_plan	Int		PK – Identifica al plan de estudios
Userid	String		Identifica al docente

Parámetros_tribunales

Parametriza la estructura de los tribunales según el plan de estudios.

Nombre	Tipo	Referencia	Descripción
Id_plan	Int	Planes_estudio	PK – Identificador único de titulación
Num_vocales	Int		Número de vocales
Presidente	String		Modo de selección de presidente
Secretario	String		Modo de selección de secretario
Vocal	String		Modo de selección de vocales

Permisos

Relación de permisos para acceder a cada controlador según grupo de usuario.

Nombre	Tipo	Referencia	Descripción
Controlador	String		PK – Controlador (MVC)
Groupid	Int		PK – Grupo de usuarios

Planes_alumnos

Contiene la información relativa al plan de estudios asignado a cada alumno.

Nombre	Tipo	Referencia	Descripción
Userid	String		PK – Identificador único de alumno
Id_plan	Int	Planes_estudio	Identificador de la titulación

Planes_estudio

Registro de los planes de estudio o titulaciones presentes en la aplicación.

Nombre	Tipo	Referencia	Descripción
Id_plan	Int		PK – Identificador único de titulación
Cod_plan	Strin		Código de la titulación
Nombre_plan	String		Nombre de la titulación

Planes_impartidos

Relación de los planes de estudio o titulaciones en los que participa un docente.

Nombre	Tipo	Referencia	Descripción
Userid	String	Users	PK – Identificador único de profesor
Id_plan	Int	Planes_estudio	Identificador de la titulación

Planes_resp

Relación de los planes de estudio o titulaciones de los que es responsable el jefe de estudios.

Nombre	Tipo	Referencia	Descripción
Userid	String	Users	PK – Identificador único de usuario
Id_plan	Int	Planes_estudio	Identificador de la titulación

Proyectos_confirmar

Contiene los proyectos que han sido propuestos por el alumno para que el director de el aprobado final antes de pasar a convocar el tribunal.

Nombre	Tipo	Referencia	Descripción
Id_proyecto	Int	Proyectos_global	PK – Identificador único de proyecto
Comentario_alumno	String		Comentario o anotación del alumno

Proyectos_favoritos

Relación de alumnos que han marcado un proyecto como favorito.

Nombre	Tipo	Referencia	Descripción
Id_proyecto	Int	Proyectos_global	PK – Identificador único de proyecto
Userid	String	Users	Identificador del alumno

Proyectos_global

Es la entidad que contiene la información general de todos los proyectos de la aplicación.

Nombre	Tipo	Referencia	Descripción
Id_proyecto	Int		PK – Identificador único de proyecto
Id_tutor	String	Users	Identificador del director
Id_alumno	String		Identificador del alumno

Titulo_proyecto	String		Título del proyecto
Desc_proyecto	String		Descripción del proyecto
Id_plan	Int	Planes_estudio	Identificador de la titulación
Keywords_proyecto	String		Palabras clave del proyecto
Area_proyecto	String		Identificador del área
Fecha_oferta_proyecto	Date		Fecha de oferta
Fecha_inicio_proyecto	Date		Fecha de inicio
Fecha_fin_proyecto	Date		Fecha de fin
Estado_proyecto	Int	Estados_proyecto	Estado del proyecto
Proyecto_confirmado	String		Indica si ha sido confirmado por el tutor
Tribunal_asignado	String		Indica si se ha asignado un tribunal al proyecto

Proyectos_solicitados

Relación de proyectos que han sido solicitados por un alumno para iniciarlos.

Nombre	Tipo	Referencia	Descripción
Id_proyecto	Int	Proyectos_global	PK – Identificador único de proyecto
Userid	String	Users	Identificador del alumno

Tribunales

Contiene la información de los tribunales de cada proyecto.

Nombre	Tipo	Referencia	Descripción
Id_tribunal	Int		PK – Identificador único de tribunal
Id_proyecto	Int	Proyectos_global	Identificador del proyecto
Fecha_tribunal	Date		Fecha de presentación
Hora_tribunal	String		Hora de presentación
Id_presidente	String		Identificador del presidente
Id_secretario	String		Identificador del secretario
Aula	String		Lugar de la presentación

Vocales

Relación de vocales asignados a cada tribunal.

Nombre	Tipo	Referencia	Descripción
Id_tribunal	Int	Proyectos_global	PK – Identificador único de proyecto
Userid	String	Users	PK - Identificador del vocal

Diagrama relacional

Podemos ver en la siguiente imagen un diagrama que muestra todas las entidades y las relaciones de claves foráneas (FK) entre ellas. La generación automática de diagramas es una de las funcionalidades que incluye el gestor de bases de datos MySQL Workbench.

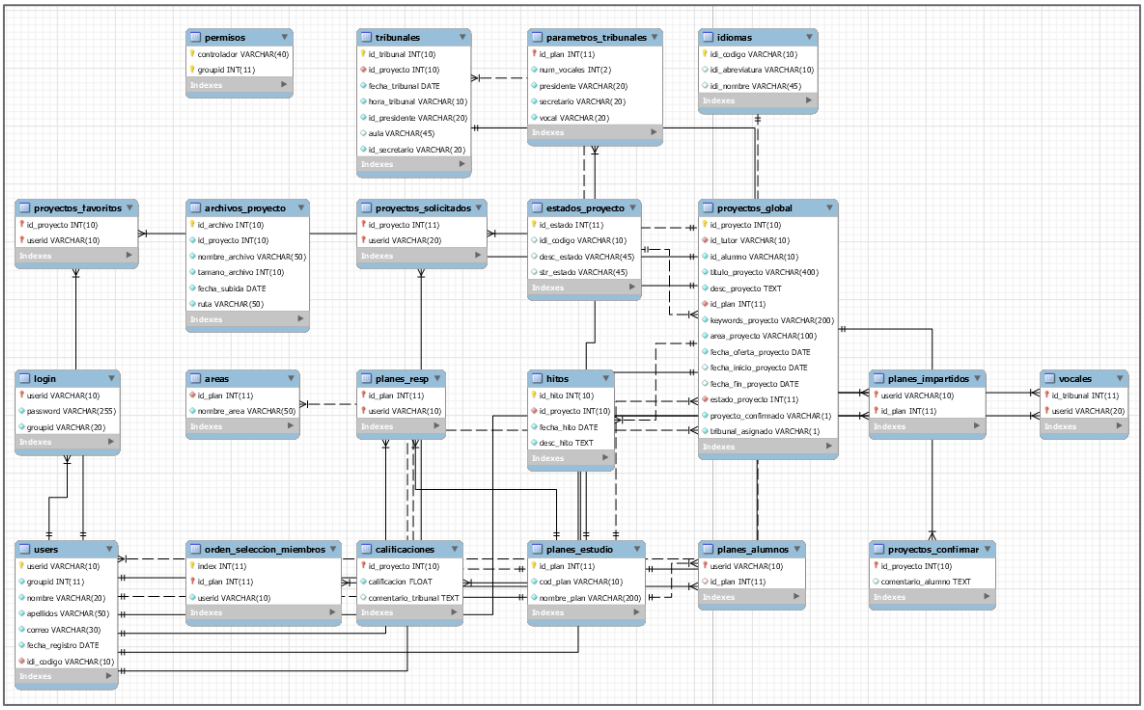

Fig 9. Diagrama relacional BBDD

5. Componentes desarrollados

A continuación, se explicarán los distintos componentes que se han desarrollado.

5.1 Enrutamiento dinámico

El enrutamiento es una parte fundamental en un diseño **MVC**. Consiste en dirigir las **URLs** que introduce el usuario a través de un enrutador (a partir de ahora *router*) para enviar las peticiones hacia el controlador adecuado. Se implanta modificando la configuración del servidor para permitir la modificación de rutas dinámicamente y creando un script que hace de router.

Un **ejemplo** de en qué consiste un router y qué aporta:

Para acceder al archivo *view/ver_proyecto.php* y cargar los datos del proyecto con id 1, la URL tendría la forma *www.dominio.com/view/ver_proyecto.php?id=1*. Utilizando redireccionamiento dinámico y un router, la ruta pasa a ser de la forma *www.dominio.com/verproyecto/id/1*. El router se encargará de interpretar esta ruta y dirigir la llamada hacia el controlador y método adecuados, mientras que la ruta real permanece oculta.

Implantación.

La configuración del servidor se basa en el uso del módulo **mod_rewrite** de Apache. Este módulo permite modificar de forma dinámica las URLs a las que acceden los usuarios. Se configura desde el fichero *.htaccess* que se encuentra en el directorio raíz de la aplicación.

```
1 RewriteEngine On
2 RewriteCond %{REQUEST_FILENAME} !-d
3 RewriteCond %{REQUEST_FILENAME} !-f
4 RewriteRule ^(.+)$ index.php?a=$1 [QSA,L]
```

Fig 10. Mod Rewrite

En nuestro caso redirige el tráfico hacia el archivo *index.php*, y pasando la URL como parámetro a la variable *a*. El fichero *index.php* es el encargado de obtener la URL solicitada, interpretarla y enviarla al *router*. Realiza un tratamiento sobre la URL introducida para separarla en tres partes que se corresponden con un controlador, una acción o método, y un parámetro. Cuando estas tres partes están identificadas, se invoca al método *call* del *router*, que recibe los parámetros de la petición.

```
// Le pasamos el control al router
require_once('config/router.php');
$router = new Router();
$router->call($controller, $action, $params);
```

Fig 11. Llamada al router

En el método *call* se comprueba que la llamada es correcta, es decir, que tanto el controlador como la acción o método requeridos son válidos. Si no existiera alguno, se utilizarían los predeterminados según el tipo de usuario.

```php
// Comprobamos que controller y action son correctos
require_once('config/routes.php');
if(array_key_exists($controller, $controllerList)){
    if(in_array($action,$controllerList[$controller])){
```

Fig 12. Se comprueba la ruta

```php
$this->route($controller, $action, $params);
```

Fig 13. Llamada al método route

El método route cumple con la función final de cargar un controlador e invocar un método. Primero, comprueba que el usuario tenga privilegios de acceso a ese controlador basándose en el grupo de usuario, y realiza la llamada.

```php
// Comprobar permisos del usuario antes de cargar el controller
require_once('model/permisos.php');
$permisos = new Permisos();
$check = $permisos->checkPermisosUser($controller);

if($check == true){
```

Fig 14. El router comprueba los permisos del usuario

Se llama al controlador y método de la siguiente forma:

```php
require_once('control/'.$controller.'Controller.php');
$_SESSION['curView'] = $controller;
$controllerName = $controller.'Controller';
$inst = new $controllerName();
$inst->{$action}($params);
```

Fig 15. El router llama al método necesario dentro del controlador

En el caso de no tener los suficientes permisos para acceder al controlador se mostraría la vista de acceso no autorizado:

```php
else {
  $vista = 'error_403';
  require_once('view/layout.php');
```

Fig 16. Se carga la vista de acceso no autorizado

Todas las rutas admitidas por la aplicación están especificadas en el fichero *config/routes.php*. La finalidad es establecer todas las posibles llamadas que puede realizar el usuario y evitar posibles problemas de seguridad. La estructura del fichero es

{Controlador} : [{Método_1}, {Método_2}, {Método_3}, …]

```
$controllerList = array(
        'index' => ['index'],
        'login' => ['login', 'authenticate', 'logout', 'login_ok', 'login_ko'],
        'header' => ['datosuser'],
        'alumnos' => ['alumnos', 'load', 'buscar', 'proyecto', 'loadAreas'],
        'profesores' => ['profesores', 'load', 'addProyecto', 'loadPlanes', 'loadAreas'],
        'verproyecto' => ['verproyecto', 'load', 'id', 'addFavorito', 'delFavorito', 'solicitar', 'cancelarSolicitud'],
        'miproyecto' => ['miproyecto', 'load', 'addhito', 'edithito', 'delhito'],
        'calificar' => ['calificar', 'load', 'addCalificacion'],
        'terminacion' => ['terminacion', 'load', 'submit'],
        'vistageneral' => ['vistageneral', 'load', 'load_planes', 'load_areas', 'buscar'],
        'director' => ['director', 'id', 'load', 'asignar', 'editproyecto', 'delproyecto', 'loadAreas', 'loadPlanes'],
        'designartribunal' => ['designartribunal', 'buscar', 'loadPlanes', 'verTribunal', 'guardarTribunal', 'asignarFecha', 'buscarTrabajos'
        'seguimiento' => ['id', 'load'],
        'tribunales' => ['tribunales', 'getTribunales'],
        'confirmacion' => ['confirmacion', 'loadProyectosConfirmar', 'confirmar'],
        'files' => ['upload', 'deleteFile', 'download']
);
```

Fig 17. Lista de rutas admitidas

5.2 Comunicación mediante AJAX.

En el apartado 3.2 se habló de Ajax como forma de comunicación entre las vistas y los controladores. Una llamada Ajax tiene la siguiente forma:

```
$.ajax({
    url: "login/authenticate",
    type: "POST",
    data: login,
    dataType: "json",
    success: function(data){
        if(data.code == 200){
            window.location = data.datos;
        } else if(data.code == 401){
            // Login incorrecto
            $('#alertLoginKo').show();
            $('#alertLoginKo').fadeOut(5000);
        }
    },
    error: function(xhr, status){
        mostrarModalError(window.diccionario.msg_error_1, true, false);
    }
});
```

Fig 18. Estructura de una llamada Ajax

Se pueden apreciar las distintas partes que forman la estructura Ajax:

- **URL**: La ruta a la que llamamos.
- **Type**: El tipo de petición. Puede ser *GET, POST, PUT, DELETE.*
- **Data**: Los datos que enviamos.
- **DataType**: El formato de datos de respuesta que se espera recibir.
- **Success**: Código a ejecutar si la respuesta es correcta.
- **Error**: Código a ejecutar si la respuesta es incorrecta.

Las rutas que se usan en el campo URL deben ser las mismas rutas que hemos permitido en el fichero *config/routes.php*. Si la llamada se ejecuta correctamente, la función *success* tiene un parámetro *data* devuelto por el servidor que contiene toda la información que hayamos solicitado.

Se ha implementado un método para formatear esta respuesta de forma que siga el mismo patrón en toda la aplicación. La estructura usada consta de tres campos: Un campo código que indica el resultado basándose en los códigos del protocolo HTTP, un mensaje que dará información sobre el error que se ha producido, si se ha producido, y el campo donde viajarán los datos, si los hubiera.

Si se ha ejecutado la petición según lo esperado, se devuelve un código 200, un mensaje de error vacío y los datos que haya devuelto el modelo. Si la petición solicitada no se ha podido ejecutar, o ha dado algún error en tiempo de ejecución, se enviará un código que dependerá del error si este ha sido controlado, un mensaje descriptivo del error y el apartado de datos irá vacío.

```php
/* Devolver estructura de respuesta OK */
public function respuestaOk($datos){
    $respuesta = array(
            'code' => '200',
            'message' => '',
            'datos' => $datos
    );
    return $respuesta;
}

/* Devolver estructura de respuesta KO */
public function respuestaKo($code, $msg){
    $respuesta = array(
            'code' => $code,
            'message' => $msg,
            'datos' => ''
    );
    return $respuesta;
}
```

Fig 19. Generación de estructura de respuestas Ok y Ko

Ambas funciones *respuestaOk* y *respuestaKo* se encuentran en la clase *Modelo* que se incluye por defecto en todos los modelos, por lo que están disponibles en toda la aplicación.

Las peticiones de tipo *GET* se utilizan para recopilar datos del servidor, como por ejemplo las numerosas búsquedas que se repiten en la aplicación. Las de tipo *POST* y *PUT* se utilizan para hacer modificaciones en la base de datos, y el tipo *DEL* se utiliza para borrar registros, aunque también reciben algún tipo de respuesta a modo de confirmación.

5.3 Control de permisos y accesos

La aplicación se enfoca a tres tipos distintos de usuarios (alumnos, profesores y jefes de estudios) y, dentro de un tipo, cada usuario tiene sus propios datos que le pertenecen únicamente a él mismo. Por esto es necesario contar con los medios necesarios para diferenciar a cada usuario y establecer una separación entre los distintos grupos.

Se ha implementado un sistema de control de acceso basado en usuarios, grupos y objetos. Los objetos son las partes o recursos de la aplicación a las que se demanda acceso, como controladores o vistas. Cada usuario pertenece a un grupo y hereda los permisos de ese grupo. Además de estos permisos de grupo, cada usuario tiene sus propios permisos para los objetos que le pertenecen solo a él.

En la entidad *login* de la base de datos tenemos una relación entre el identificador del usuario, el grupo al que pertenece y una contraseña que se encuentra encriptada mediante el método de php *password_hash*. Al introducir la contraseña en el login, se comprueba contra el hash almacenado en base de datos mediante el método de php *password_verify*. Estos métodos de encriptación hacen uso del algoritmo *Bcrypt*.

```
$sql = sprintf('
        SELECT
            password
        FROM
            login
        WHERE
            userid = \'%s\'
        ', $username);
$encryptedPass = $this->query($sql);
if( sizeof ($encryptedPass ) == 1 ){
    // Comprobamos que los hashes coinciden
    $check = password_verify($password, $encryptedPass[0][0]);
```

Fig 20. Se comprueba la contraseña introducida contra la guardada en base de datos

Si las contraseñas coinciden, se considera que el login ha sido correcto y se crea una **sesión** de PHP con los datos del usuario, que estará disponible desde toda la aplicación y a lo largo del tiempo.

```
$oUser = array(
        'userid' => $oDatosUser['userid'],
        'groupid' => $oDatosUser['groupid'],
        'usernombre' => $oDatosUser['username'],
        'language' => $oDatosUser['language']
);
$_SESSION['user'] = $oUser;
```

Fig 21. Creación de la sesión

5.4 Sesiones de PHP

Una sesión, *$_session*, de PHP se trata de una forma de almacenar información del usuario de forma que sea accesible desde cualquier parte de la aplicación, sin recurrir a almacenarla en base de datos.

La funcionalidad es similar a la de las famosas *cookies* de internet, pero en la práctica su funcionamiento varía totalmente. Las *cookies* guardan información del usuario en el equipo del usuario, con lo cual son fácilmente accesibles y por tanto modificables. Esto puede dar lugar a serios problemas de seguridad al no poder confiar en ningún dato que provenga de la *cookie* del usuario.

Las sesiones de PHP, por el contrario, almacenan la información en el servidor. Para reconocer a qué cliente pertenece determinada información, se le asigna un *token* que se guardará localmente en el equipo del usuario y servirá para identificarle. De esta forma no es posible robar o modificar los datos al no estar guardados en el equipo del usuario.

Con *cookies* se puede alcanzar el mismo nivel de seguridad si la *cookie* se utiliza solamente para guardar el *token* que servirá para identificar al usuario, y la información sensible se almacena en el servidor, imitando el comportamiento nativo de las sesiones de PHP.

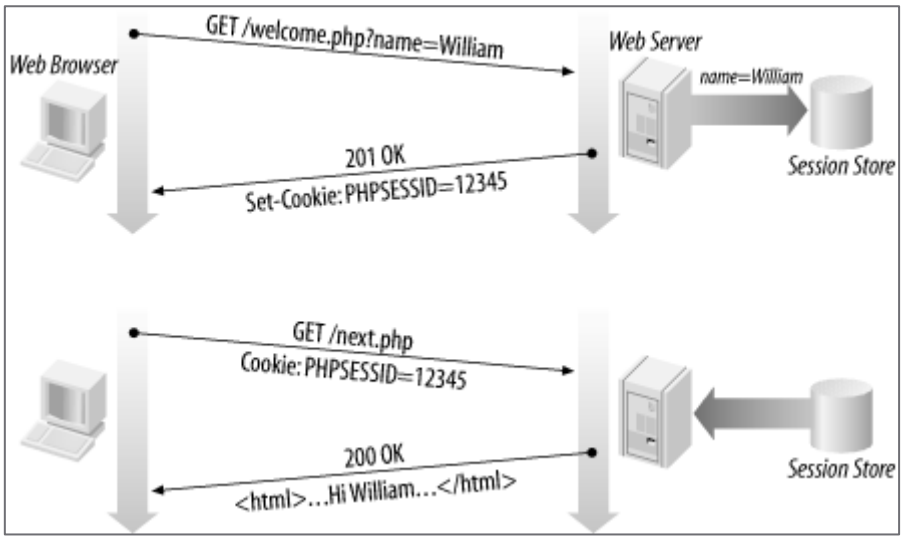

Fig 22. Funcionamiento de una sesión de PHP

Cuando un usuario inicia sesión se guarda su identificador, nombre, grupo al que pertenece e idioma en la sesión. Cada vez que intente acceder a una ruta dentro de la aplicación, se comprueba que el grupo al que pertenece tenga permisos para acceder a la ruta. Para llevar a cabo esta comprobación, se introduce el control de permisos en el *router*, y se utiliza como parámetro el controlador al que el usuario intenta acceder.

```
require_once('model/permisos.php');
$permisos = new Permisos();
$check = $permisos->checkPermisosUser($controller);
```

Fig 23. El router comprueba los permisos del usuario antes de servir cada llamada

El método que comprueba si el usuario tiene permisos, obtiene los datos del usuario de la variable de sesión y los comprueba contra la entidad *permisos*. Recordamos su estructura:

- **Controlador**: De tipo *String*, hace referencia al nombre de un controlador.
- **Groupid**: De tipo *Int*, hace referencia al grupo de usuarios que tiene acceso.

```
$sql = sprintf('
        SELECT 1
        FROM permisos
        WHERE
            controlador = \'%s\' AND
            groupid = %d
    ', $controlador, $oUser['groupid']);

$res = $this->query($sql);
```

Fig 24. Validación de permisos de acceso a un controlador

Dependiendo del resultado de la validación se dirigirá al usuario al controlador correspondiente o por el contrario se mostrará una pantalla de error.

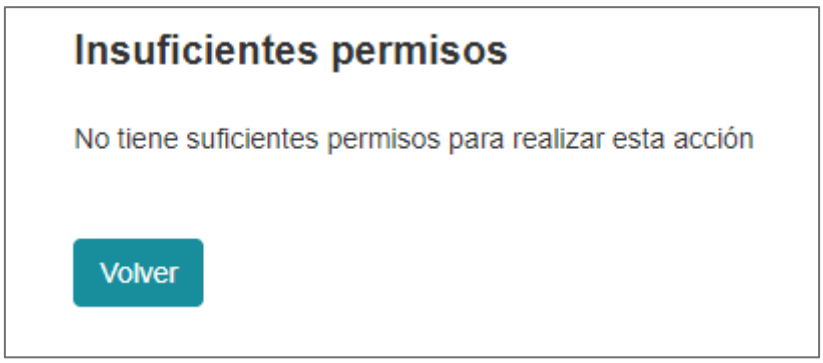

Fig 25. Pantalla que informa de que el usuario no tiene permisos para realizar una acción

5.5 Seguridad de la aplicación

La seguridad informática es un tema que está a la orden del día desde hace muchos años. Cada año surgen nuevas amenazas que afectan a millones de equipos, desde usuarios particulares hasta grandes empresas. Los ataques pueden ser de muchos tipos distintos y afectar a distintas partes del software que existe en el equipo, pero vamos a centrarnos solamente en los principales ataques que afectan a aplicaciones web.

Según la fundación *OWASP (Open Web Application Security Project)*, referente mundial en materia de seguridad informática, las vulnerabilidades más comunes en aplicaciones web son las inyecciones *SQL* y el *XSS (Cross Site Scripting).* A continuación, se explican los dos conceptos y por qué se considera que la aplicación es relativamente segura.

La aplicación tiene usuarios de distintos perfiles (alumnos, profesores, jefes de estudios) y a priori algunos parecen más fiables y menos propensos a excederse más allá de los límites. Sin embargo, la experiencia nos dicta que al desarrollar software nunca se puede confiar en los

usuarios. Durante el desarrollo se ha seguido la premisa de que **todos los usuarios son maliciosos** y tienen malas intenciones, sin importar su perfil.

5.5.1 Cross Site Scripting

El *Cross Site Scripting* ***'XSS'*** consiste en ejecutar un *script*, un trozo de código *JS* o *HTML*, en el equipo de la víctima. Es un ataque que tiene lugar en el navegador que ejecuta el código. Para ser capaz de ejecutar dicho código, el atacante tiene que introducirlo de forma permanente en el servidor, o conseguir cargarlo de forma temporal directamente en el equipo de la víctima (por ejemplo, mediante una URL elaborada). Dado que el código malicioso proviene del servidor, el navegador del cliente lo ejecutará al considerarlo fiable.

Para evitar la manipulación de URLs, se ha evitado el envío de parámetros mediante el método *GET* de *HTTP*. En los casos en los que ha sido necesario usarlo, los parámetros enviados se tratan directamente en el controlador y nunca se representan en la vista.

Para evitar la inserción permanente se ha optado por la opción más eficaz, filtrar toda entrada de datos del usuario a la aplicación, lo cual es adecuado en términos generales y no solamente para este caso. Se utiliza la función de filtrado nativa de PHP *filter_var* pasándole como parámetros la cadena de caracteres que queremos limpiar y como tipo de filtro *FILTER_SANITIZE_STRING*. Este filtrado eliminará cualquier *tag* o etiqueta de HTML que da pie a explotar una vulnerabilidad.

5.5.2 Inyección SQL

Este es uno de los más comunes y peligrosos ataques que se pueden producir en una aplicación web. Consiste en que un atacante consigue "romper" una consulta SQL para introducir a continuación otra consulta preparada por él, que se ejecutará junto a la primera consulta. Para romper dicha consulta es necesario introducir caracteres como las comillas simples (') o dobles (") y terminarla con un punto y coma (;) para poder ejecutar una consulta maliciosa a continuación.

El riesgo que entraña es que un atacante podría tener acceso total a la base de datos de la aplicación. Ejecutando múltiples consultas llegaría a conocer la estructura de la base de datos y podría ver, modificar o borrar cualquier registro.

Para evitar este tipo de ataques se lleva a cabo un filtrado de la totalidad de datos que provienen del usuario, sea del tipo que sea. Igual que en el caso del *XSS*, recurrimos a la función nativa *filter_var* con el filtro FILTER_SANITIZE_STRING que se encarga de eliminar todos los caracteres susceptibles de romper nuestras consultas SQL.

5.5.3 Control de permisos de usuario

Otro problema común en las aplicaciones web es la falta de control sobre las acciones que realiza un usuario. A veces la aplicación proporciona demasiada información al usuario, aunque esta no se vea a simple vista. Esa información puede ser aprovechada por un usuario malicioso para acceder a determinados recursos que no deberían estar a su alcance.

Un ejemplo aplicado a esta aplicación sería el borrado de archivos. Un alumno puede borrar los archivos asociados a su TFG que haya cargado en la aplicación. Si un usuario malicioso interceptase la llamada legítima que indica que hay que borrar el archivo con una id X, y modificara esa id, podría borrar cualquier archivo en el servidor.

El enfoque que se ha utilizado es validar todas las acciones que realice el usuario sobre cualquier objeto o recurso, y comprobar siempre que el usuario que realiza la acción está autorizado para ello.

5.5.4 Ocultación de rutas

Cuanta menos información tenga un posible usuario malicioso sobre nuestra aplicación, más segura será. En el caso de las rutas de acceso a los distintos ficheros '*.php'* que componen la aplicación, las redirecciones se gestionan internamente a través del router por lo que el usuario no conoce la ruta exacta a ninguno de ellos.

El caso de las descargas de archivos puede ser conflictivo, dado que la forma habitual en la que funcionan las descargas vía *HTTP* es con el navegador accediendo directamente a la ruta del archivo y solicitando dicho archivo al servidor en la petición. Un usuario con dudosas intenciones podría modificar esta ruta y apuntar hacia otros archivos o carpetas. Por ejemplo:

El alumno con identificador *alum1* tiene asignada una carpeta para sus ficheros de la forma *files/alum1/ejemplo.docx*. Viendo esta ruta, uno podría tratar de acceder a una carpeta vecina, como *files/alum2/*. Estaría accediendo a recursos que no deberían estar disponibles para él.

Para evitarlo, se ha creado un gestor de descargas que funciona pasándole como parámetro el identificador del archivo que se quiere descargar. El gestor comprueba si el usuario tiene permisos para ver ese archivo, y de ser así, obtiene la ruta desde la entidad *archivos_proyecto*, lee el archivo y lo sirve al navegador como descarga. La diferencia radica en que el usuario no accede directamente a la ruta del archivo, sino a una dirección parametrizada gestionada por un controlador.

A modo de sumario, en la aplicación se controlan todas las entradas de datos por parte del usuario, todas las acciones que puedan requerir permisos, y se considera al usuario un atacante en potencia. Esto no convierte la aplicación en segura puesto que no existe ningún sistema seguro al 100%, pero se puede considerar que está protegida de los principales y más frecuentes ataques.

5.6 Gestión de archivos

La gestión de archivos es necesaria en tanto que la aplicación ofrece la funcionalidad de cargar al servidor los archivos relativos al proyecto, y es necesario llevar un registro de quién sube qué archivo, y dónde se encuentra dicho archivo. También es necesario controlar quién y cómo lo descarga.

La subida de archivos se realiza desde la interfaz de terminación del TFG. La descarga puede realizarse desde la misma interfaz, la interfaz de confirmación de TFG del tutor, la interfaz de

tribunales en los que se participa y la interfaz de vista general. El controlador encargado de gestionar subidas, descargas y borrados es *FilesController*. Consta de los siguientes métodos:

- **Upload**: Realiza la subida del archivo al directorio especificado y comprueba que no haya errores.
- **LogUploadedFile**: Lleva a cabo el guardado lógico del archivo en la base de datos.
- **DeleteFile**: Realiza el borrado de un archivo.
- **Download**: Se encarga de servir la descarga de los archivos.

A continuación, se explica brevemente cada función.

5.6.1 Subida de ficheros

Se realiza a través de un formulario de Html con un objeto *input file*. Este componente de Html es el que nos permite seleccionar un archivo de nuestro sistema y posteriormente enviarlo al servidor, donde se guarda en una ruta temporal. En el servidor, la llamada es recibida por el controlador *FilesController*, que mueve el archivo de la ubicación temporal en la que se carga a su destino final y lo renombra añadiendo un *timestamp* por motivos de control y para evitar duplicados. Para cada usuario que realice subidas, se creará una carpeta nueva con su identificador de usuario.

5.6.2 Registro de los ficheros subidos

Una vez que el fichero ha sido guardado en la carpeta correspondiente con su nuevo nombre, el controlador llama a la función que se encarga de registrar la subida en la base de datos en la entidad *archivos_proyecto*. Recordamos su estructura:

- **Id_proyecto:** Identifica al proyecto.
- **Nombre_archivo:** Es el nombre con el que se sube el archivo, pero no con el que se guarda. Este nombre es el que se mostrará en la aplicación.
- **Tamano_archivo:** Controla el tamaño en kilobytes del archivo.
- **Fecha_subida:** Fecha en la que se cargó el archivo.
- **Ruta:** Ruta exacta al archivo. Es fundamental para poder descargarlo o eliminarlo.

5.6.3 Borrado de ficheros

Solo el dueño de un archivo puede realizar el borrado del mismo, es decir, el alumno que lo cargó al sistema. El controlador comprueba que el usuario que hace la llamada de borrado sea el mismo que subió inicialmente el archivo. De ser correcto, el modelo se encarga de obtener la ruta del archivo para borrarlo físicamente. Después, se realiza el borrado lógico en la entidad *archivos_proyecto*. En el modelo se incluye una doble validación de los permisos del usuario al hacer un cruce en las dos consultas SQL entre el campo a*rchivos_proyecto.id_alumno* y el id del usuario que hace la llamada.

5.6.4 Descarga de ficheros

En este apartado entra en juego uno de los puntos que hemos destacado en el apartado de seguridad, y es la ocultación de rutas internas de la aplicación. Generalmente, una descarga a través de HTTP consiste en que el navegador accede a la ruta de un fichero y el servidor se lo sirve. En el desarrollo de la aplicación se ha tratado de evitar dar información sobre la

estructura interna de carpetas y ficheros con el fin de dificultar cualquier intento de apropiación indebida o manipulación de ficheros.

Las descargas funcionan de la siguiente forma. Cuando se muestra un enlace, en vez de contener la ruta directa al archivo, contiene una ruta genérica de la forma *files/download/{id}*, donde {id} es el identificador del archivo. Al seguir ese enlace, lo que sucede es que se envía el valor de *{id}* como parámetro al controlador *filesDownloader* y al método *download*.

En el controlador, se filtra el parámetro *id* con el filtro *FILTER_VALIDATE_INT* y posteriormente se comprueba que el usuario tenga permisos sobre dicho archivo, teniendo en cuenta el grupo al que pertenece el usuario.

```
switch($oUser['groupid']){
    case 10:
        $sql = sprintf('
                SELECT 1
                FROM archivos_proyecto AS f
                INNER JOIN proyectos_global AS g ON g.id_proyecto = f.id_proyecto
                WHERE
                    f.id_archivo = %d AND
                    g.id_alumno = \'%s\'
        ', $id, $oUser['userid']);
        break;
    case 20:
    case 30:
        $sql = sprintf('
                SELECT 1
                FROM archivos_proyecto AS f
                INNER JOIN proyectos_global AS g ON g.id_proyecto = f.id_proyecto
                INNER JOIN tribunales AS t ON t.id_proyecto = f.id_proyecto
                INNER JOIN vocales AS v ON v.id_tribunal = t.id_tribunal
                WHERE
                    f.id_archivo = %d AND
                    (
                        g.id_tutor = \'%s\' OR
                        t.id_presidente = \'%s\' OR
                        t.id_secretario = \'%s\' OR
                        v.userid = \'%s\'
                    )
        ', $id, $oUser['userid'], $oUser['userid'], $oUser['userid'], $oUser['userid']);
        break;
}
$p = $this->query($sql);
```

Fig 26. Validación de permisos para la descarga de un fichero

Si el usuario pertenece al grupo 10 (alumnos), se comprueba que sea el alumno que realiza el proyecto. Si pertenece a los grupos 20 o 30 (personal docente y jefes de estudio), se comprueba que el usuario sea tutor del proyecto o miembro del tribunal. Si la validación de permisos es correcta, se genera un *header* que indica al navegador que debe descargar un archivo y se sirve el contenido.

```
$permisos = $this->files->checkPermisos($oUser, $id);
if($permisos == 'ok'){
    $r = $this->files->getRuta($id);
    $ruta = $r['ruta'];
    $nombre = $r['nombre'];

    header('Content-Disposition: attachment; filename="'.$nombre.'"');
    readfile($ruta);
```

Fig 27. Llectura y descarga del archivo

5.7 Envío de correos

La funcionalidad de envío de correos electrónicos permite a la aplicación notificar un evento a los usuarios. Estos eventos corresponden a cambios en el estado de un TFG, tales como la asignación a un alumno, el envío por parte del alumno, la designación del tribunal o el fin del TFG y su calificación.

El envío de correos se hace mediante el método *sendMail* dentro de la clase *modelo*.

```
public function sendMail($to, $title, $msg){
    try{
        // Obtener dirección del usuario al que se envía el mail
        $addr = $this->getAddrByUser($to);
        mail($addr, $title, $msg);
        return true;
```

Fig 28. Método que envía los correos electrónicos

Este método tiene como parámetros de entrada el id del destinatario, el título y el cuerpo del mensaje. A partir del id del destinatario se obtiene su dirección de correo, que se pasa a la función nativa de PHP *mail*. Para que la función *mail* funcione correctamente, es necesario configurar los datos del servidor *SMTP (Simple Mail Transfer Protocol)* que se encargará de enviar los correos. A modo de ejemplo y por la falta de un servidor propio de pruebas, se ha configurado el servidor de Gmail.

```
smtp_server=smtp.gmail.com
smtp_port=587
```

Fig 29. Configuración del servidor SMTP

```
auth_username=xxxxx@gmail.com
auth_password=0123456789
```

Fig 30. Configuración de la cuenta de usuario del servidor SMTP

Actualmente, dado que la aplicación se encuentra todavía en desarrollo, el método *sendMail* está desactivado por la falta de un servidor de correo dedicado y cuentas de correo reales a las que enviar los correos sin incurrir en el spam a direcciones ficticias.

5.8 Multi idioma

Para mejorar la usabilidad de la aplicación y hacerla más atractiva para estudiantes acogidos a los programas internacionales como *Erasmus*, se ha decidido añadir un soporte para permitir que la aplicación se muestre en varios idiomas.

La implantación se realiza parametrizando todos los textos que aparecen en la aplicación, de forma que se rendericen mediante un método que obtenga un texto determinado de una fuente diferente en función del idioma seleccionado. Estas fuentes serán archivos codificados que contendrán todas las traducciones para cada idioma que se desee implantar.

Para obtener las traducciones se utiliza un método nativo de PHP llamado *getText*, mediante su abreviatura "*_()*". Necesita dos parámetros, un dominio en el que buscará las traducciones, y el identificador del texto a traducir. La función del dominio es indicar en qué fichero se deben buscar las traducciones y permite organizar los diccionarios, separándolos según vista, funcionalidad u otro criterio deseado. En este caso y dado que la aplicación no es tan grande, se ha usado un dominio llamado *messages* para casi toda la aplicación y uno llamado *mail* donde irían las traducciones de los correos que envía la aplicación. La gestión del idioma y el dominio según se puede ver en la figura 31.

```
function __($domain, $name){
    if(isset($_SESSION['language'])){
        $language = $_SESSION['language'];
    } else{
        $language = 'es_ES'; //default
    }
    putenv("LANG=$language");
    setlocale(LC_ALL, $language);
    $realpath = $_SERVER['DOCUMENT_ROOT'] . '/locale';
    bindtextdomain($domain, $realpath);
    bind_textdomain_codeset($domain,'UTF-8');
    textdomain($domain);
    $traduc = _($name);
    return $traduc;
}
```

Fig 31. Método para traducir a distintos idiomas

El idioma para la traducción se obtiene de la base de datos. En la entidad *users*, el campo *idi_codigo* identifica el idioma en el que el usuario va a ver la aplicación. Este dato se guarda en la variable de sesión que se crea al iniciar sesión, de forma que está presente y accesible en toda la aplicación. Por defecto este idioma será español, hasta que se incluyan traducciones a más idiomas.

```
$oUser = array(
        'userid' => $oDatosUser['userid'],
        'groupid' => $oDatosUser['groupid'],
        'usernombre' => $oDatosUser['username'],
        'language' => $oDatosUser['language']
);
$_SESSION['user'] = $oUser;
```

Fig 32. Estructura de la variable de sesión del usuario

6. Interfaces desarrolladas

La aplicación se compone de diversas interfaces que permiten llevar a cabo todas las acciones que se describen en los objetivos.

A continuación, se explicarán las interfaces que se han desarrollado.

6.1 Interfaces comunes

Son las partes comunes a todos los tipos de usuario.

6.1.1 Interfaz de login

Como la aplicación está destinada a tres grupos distintos de usuarios (alumnos, profesores, jefes de estudios), y ofrece una funcionalidad distinta a cada uno, es necesario identificar qué tipo de usuario está accediendo para mostrarle el contenido adecuado en cada caso. Además, cada usuario tiene sus propios datos que le corresponden únicamente a él, tales como proyectos ofertados, proyecto en curso, planes de estudios, etc.

Por este motivo ha sido necesario crear un login para poder diferenciar a cada usuario. La interfaz de login consta de una vista sencilla con los campos habituales en un login, esto es, un nombre de usuario y una contraseña. Al introducir la contraseña en el login, se comprueba contra la almacenada en la base de datos.

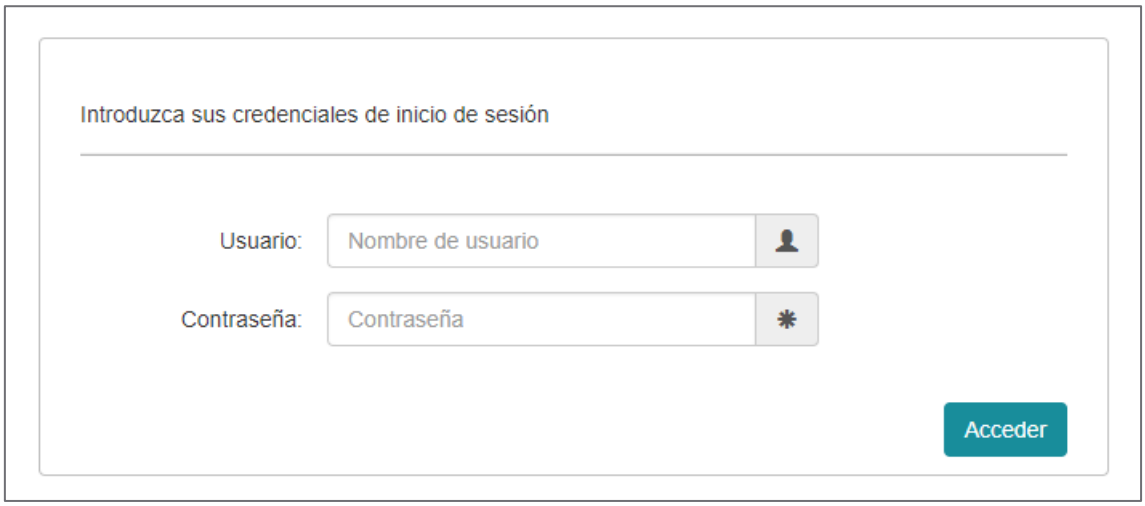

Fig 33. Vista de inicio de sesión

Si la verificación es correcta, entraremos en la aplicación y seremos redirigidos a un portal distinto según el tipo de usuario. Podremos volver a la pantalla de login siempre que usemos el botón de cerrar sesión o caduque nuestra sesión. La parte técnica del login está explicada en el apartado 5.3.

Fig 34. Botón de cierre de sesión

Si la verificación es incorrecta, se mostrará un mensaje de error que se desvanecerá pasados unos segundos. No se realiza ninguna métrica de intentos de inicio de sesión fallidos, ni hay un control que evite una gran cantidad de inicios de sesión en un tiempo reducido, como sucede, por ejemplo, en un intento de login por fuerza bruta. Se plantean como posibles mejores en desarrollos futuros en el apartado 7.2.

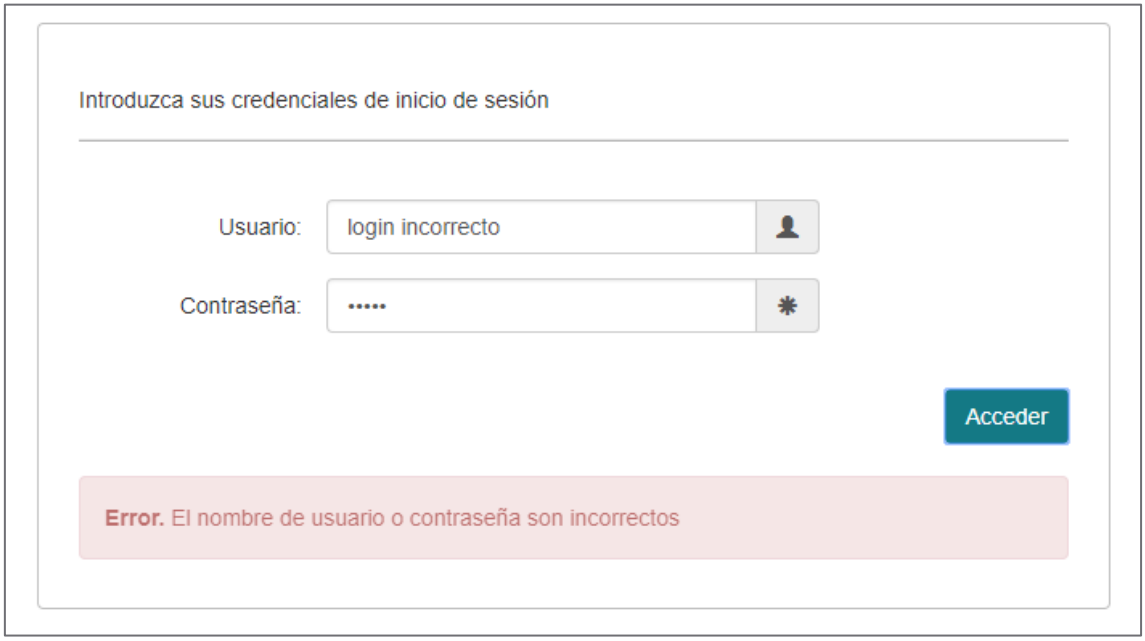

Fig 35. Inicio de sesión incorrecto

6.1.1 Cabecera o header

La aplicación consta de una cabecera que está presente en todas las vistas menos en la de login. Se muestra en un marco superior que contiene un logo, el título de la aplicación, el nombre del usuario, la selección de idioma, el botón de inicio y el de cerrar sesión.

Fig 36. Cabecera de la aplicación

La forma de integrar la cabecera en todas las vistas es incluyéndola en la vista que hace de plantilla, llamada *layout.php,* a partir de la cual se cargan el resto de vistas.

Para recuperar el nombre del usuario se obtiene el *id* de la variable de sesión y continuación se buscan sus datos en la entidad *users*. El botón de inicio realiza una redirección con javascript hacia la página inicial de la aplicación. El de cierre de sesión o *logout*, destruye la variable de sesión de PHP y redirige al login.

Los botones de selección de idioma permitirían alternar entre castellano, inglés y cualquier otro idioma que pudiera introducirse, aunque las traducciones no se han incluido actualmente.

6.1.3 Barra de navegación o navbar

Por debajo de la cabecera o header se sitúa una barra de navegación o navbar que sirve para tener a mano algunos de los enlaces de la aplicación. La barra carga desde la misma fuente para cada tipo de usuario, por lo que se realiza una comprobación dinámica para determinar qué enlaces se deben mostrar y cuáles no en función del tipo de usuario que acceda.

La correspondiente a alumnos (grupo 10) es la más sencilla:

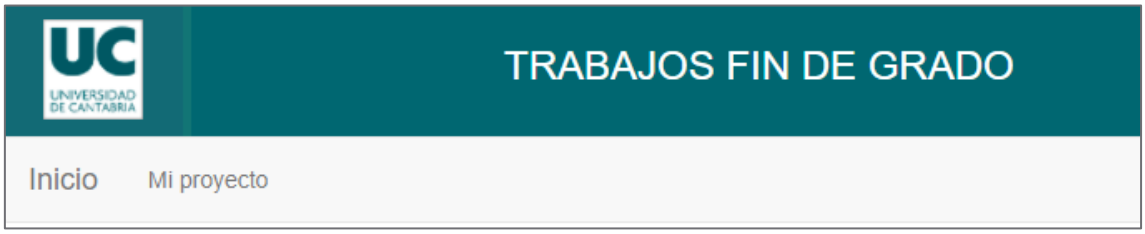

Fig 37. Barra de navegación para alumnos

En el caso de profesores (grupo 20) incluye más opciones:

Fig 38. Barra de navegación para docentes

Por último, para responsables de un plan de estudios (grupo 30) incluye también los enlaces a aquellas funciones que están reservadas para ellos:

Fig 39. Barra de navegación para responsables

Los enlaces de la barra de navegación se colorearán al estar activo para una mejor experiencia de usuario.

Fig 40. Barra de navegación con enlaces resaltados

6.2 Interfaces desde el punto de vista de un alumno

A continuación, las interfaces a las que tienen acceso los usuarios de tipo 10.

6.2.1 Interfaz de inicio para alumnos

Es la interfaz a la que se redirige a los alumnos al iniciar sesión y consta de 3 secciones.

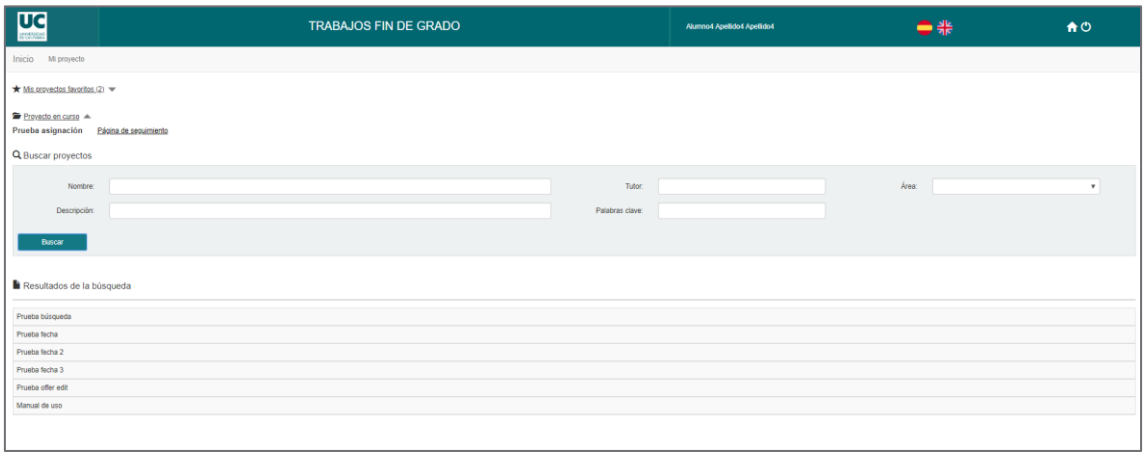

Fig 41. Vista de la interfaz de inicio para alumnos

La primera es una lista breve de los proyectos que el alumno ha marcado como favoritos, indicando su interés en ellos. Cada proyecto en esa lista se representa mediante su título, pero al pinchar sobre él, se abre un desplegable en el que podremos ver una descripción breve y un enlace a la página del proyecto.

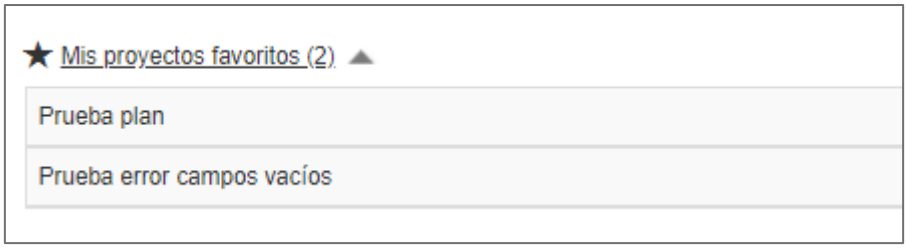

Fig 42. Listado de proyectos favoritos

La segunda permite ver el proyecto que el alumno está actualmente realizando, en el caso de que lo hubiera. La presentación es similar a la lista de favoritos, pero en este caso el enlace lleva a la interfaz de seguimiento de un proyecto en curso.

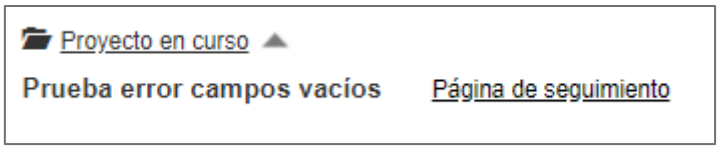

Fig 43. Enlace a la página de seguimiento del proyecto en curso

La tercera se trata de una interfaz de búsqueda de proyectos. Ofrece una serie de filtros para acotar las búsquedas, tales como, título del proyecto, profesor que lo oferta, descripción, palabras clave y área. Si no se utiliza ningún filtro, mostraría todos los datos disponibles. En cualquier caso, la búsqueda está limitada al plan de estudios o titulación del alumno, y a proyectos que no se encuentren en curso, es decir, cuyo estado en la aplicación sea de 1 (ofertado).

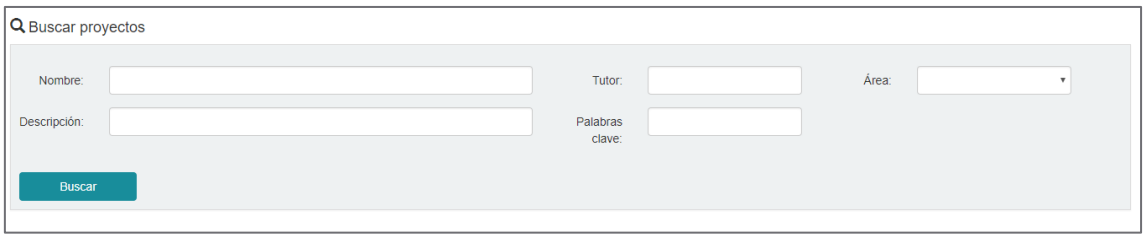

Fig 44. Filtros disponibles del buscador de proyectos

```php
public function buscarProyectos($oUser, $titulo, $desc, $keywords, $area, $tutor){

    // Crear la query según los datos introducidos
    $sql= " SELECT *
            FROM proyectos_global AS proyectos
            INNER JOIN users AS tutor ON proyectos.id_tutor = tutor.userid
            INNER JOIN planes_alumnos AS planes ON planes.id_plan = proyectos.id_plan
            INNER JOIN planes_estudio ON planes_estudio.id_plan = planes.id_plan
            ";
```

Fig 45. Query SQL del buscador

Los resultados son presentados en una tabla utilizando la misma estética vista anteriormente. Al desplegar una de las filas obtendremos una información más amplia, con la descripción completa y palabras clave. Incluye un enlace a la página de la oferta del proyecto en la que se puede ver la ficha completa, y las acciones disponibles.

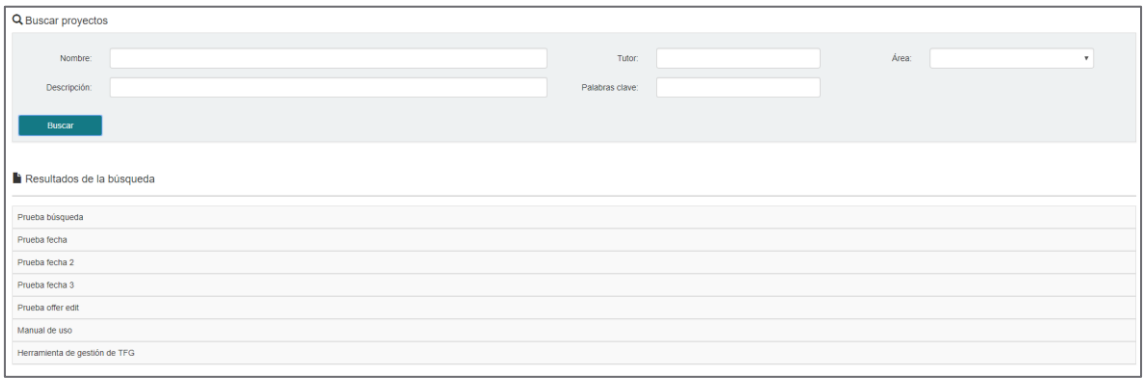

Fig 46. Resultados de la búsqueda

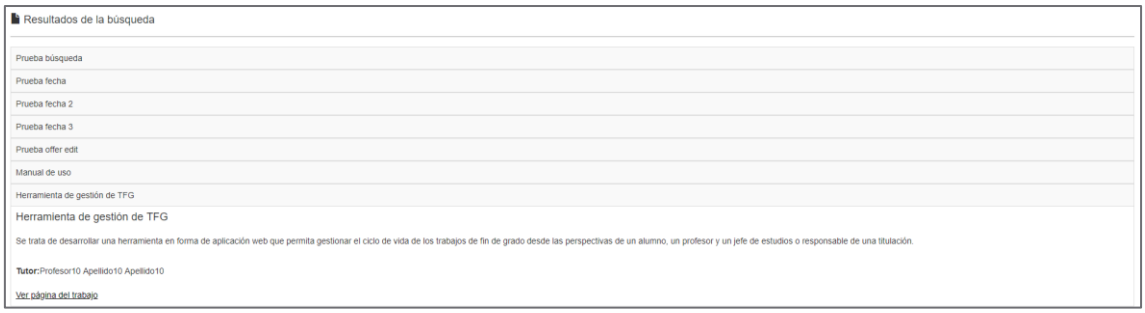

Fig 47. Desplegable con información más detallada

6.2.2 Interfaz de proyecto ofertado

Esta interfaz ofrece la ficha completa de un proyecto. Tiene tres funcionalidades: Mostrar toda la información relativa y que no cabía por motivos de diseño en los resultados del buscador, marcar el proyecto como favorito, y solicitar que el profesor te asigne el proyecto.

Para la carga inicial de la vista es necesario conocer el identificador del proyecto que queremos visualizar. Este parámetro, de tipo entero, se envía utilizando el método GET del protocolo HTTP. De esta forma es posible obtener una URL personalizada para cada proyecto, que puede ser apuntada en marcadores, enviada por correo, o simplemente recordada en el historial del dispositivo, opuestamente a lo que sucedería si el parámetro se envía por POST.

Para obtener el Id del proyecto hay que extraerlo a partir de la llamada HTTP. En este tipo de llamadas GET, el parámetro se envía en la propia URL de la forma *dominio.com/verproyecto/id/7*. El valor del parámetro *id* es 7.

La información del proyecto se obtiene de la entidad *proyectos_global* realizando una búsqueda a partir del identificador de proyecto que se recoge desde la URL.

Fig 48. Información ampliada del proyecto y botonera

Desde la botonera están disponibles las opciones de guardar el proyecto como favorito o solicitarlo.

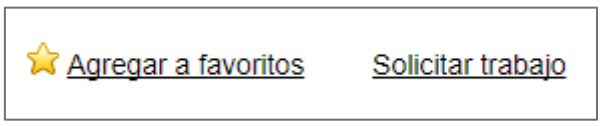

Fig 49. Botonera para agregar un proyecto a favoritos o solicitarlo

Al marcar el proyecto como favorito se realiza una inserción sobre la entidad *proyectos_favoritos*. En el caso de solicitarlo, la inserción se hace sobre la entidad

proyectos_solicitados. En ambos casos se trata de una relación directa entre un proyecto y un alumno.

```
$sql = sprintf("
        INSERT IGNORE INTO proyectos_solicitados (id_proyecto, id_alumno)
        VALUES (%d, '%s')
        ", $idProyecto, $oUser['userid']);

$this->query($sql);
```

Fig 50. Query de solicitar proyecto

```
// Query
$sql = sprintf("
        INSERT IGNORE INTO proyectos_favoritos (id_proyecto, id_alumno)
        VALUES (%d, '%s')
        ", $idProyecto, $oUser['userid']);
try
{
    $this->query($sql);
```

Fig 51. Query de agregar un proyecto a favoritos

Las acciones sobre el proyecto fuerzan una recarga dinámica de los enlaces para alterar el texto mostrado según el estado actual.

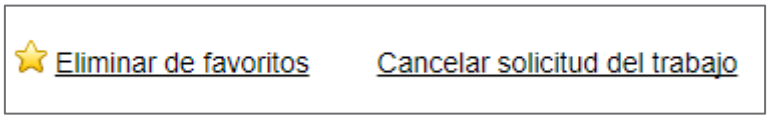

Fig 52. Botonera de acciones cambia de texto según el estado

6.2.3 Interfaz de seguimiento de proyecto (alumno)

Esta interfaz se usa para que el alumno mantenga una línea de comunicación abierta con el tutor y pueda registrar los avances de su proyecto. Consta de una sección con la información del proyecto que se puede ocultar mediante un desplegable, la información de contacto del tutor, y una sección en la que se podrán crear, editar y borrar entradas que representen avances, logros o hitos en el desarrollo del proyecto.

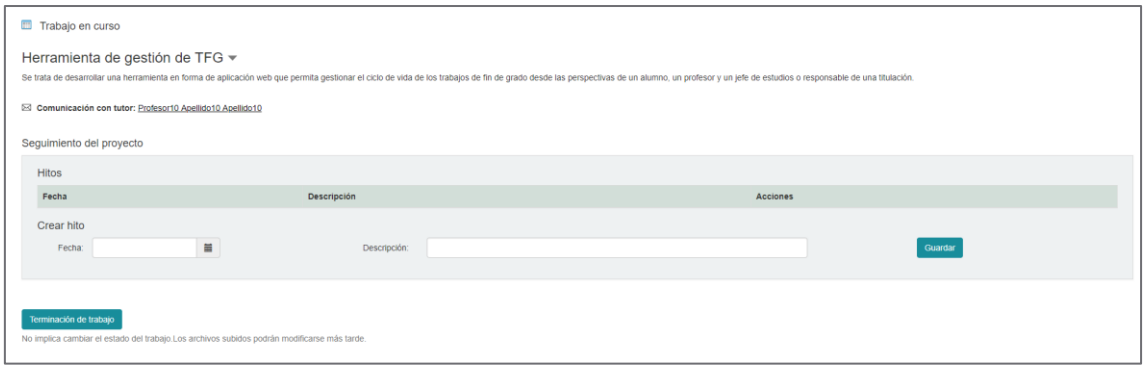

Fig 53. Vista de la interfaz de seguimiento

En esta sección se puede encontrar un *CRUD* (Create Read Update Delete) para gestionar los hitos.

Creación

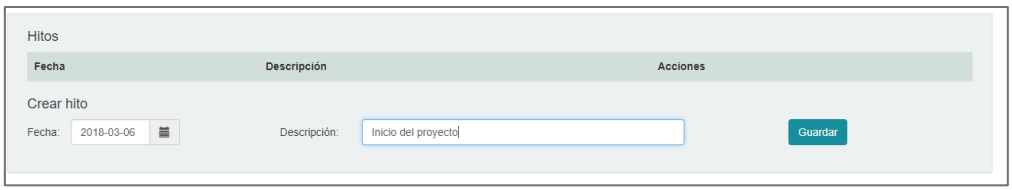

Fig 54. Formulario de creación de hitos

```php
public function addHito($oUser, $idProyecto, $desc_hito, $fecha_hito){
    try
    {
        $sqlPermisos = sprintf('
                SELECT 1
                FROM proyectos_global
                WHERE
                    estado_proyecto = 2 AND
                    id_alumno = \'%s\' AND
                    id_proyecto = %d
            ', $oUser['userid'], $idProyecto);
        $res = $this->query($sqlPermisos);
        if($res[0][1] == 1){
            $sqlquery = sprintf("
                    INSERT INTO hitos (id_proyecto, fecha_hito, desc_hito)
                    VALUES ('%s', '%s', '%s')
                ", $idProyecto, $fecha_hito, $desc_hito);

            $this->query($sqlquery);
```

Fig 55. Query de creación de un hito

Edición

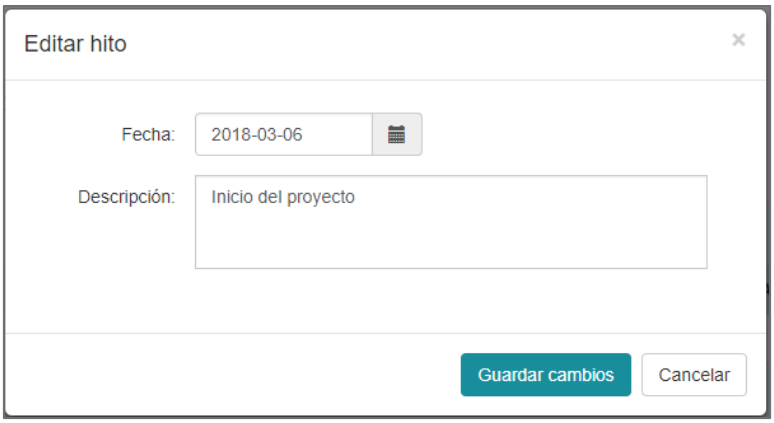

Fig 56. Formulario de edición de hitos.

```
public function editHito($oUser, $id_hito, $idProyecto, $desc_hito, $fecha_hito){
    try
    {
        $sqlPermisos = sprintf('
                SELECT 1
                FROM proyectos_global
                WHERE
                    estado_proyecto = 2 AND
                    id_alumno = \'%s\' AND
                    id_proyecto = %d
            ', $oUser['userid'], $idProyecto);

        $res = $this->query($sqlPermisos);

        if($res[0][1] == 1){
            $username = $oUser['userid'];
            $desc_hito = preg_replace( "/\r|\n/", ". ", $desc_hito);

            $sqlquery = sprintf("
                UPDATE hitos h
                JOIN proyectos_global p ON h.id_proyecto = p.id_proyecto
                SET h.fecha_hito = '%s', h.desc_hito = '%s'
                WHERE
                    h.id_hito = '%s' AND
                    h.id_proyecto = '%s' AND
                    p.id_alumno = '%s'
            ", $fecha_hito, $desc_hito, $id_hito, $idProyecto, $username);
            $this->query($sqlquery);
```

Fig 57. Query de edición de hitos

Borrado

Fig 58. Confirmación de borrado de un hito

```php
public function delHito($oUser, $id_hito, $idProyecto){
    try
    {
        $sqlPermisos = sprintf('
                SELECT 1
                FROM proyectos_global
                WHERE
                    estado_proyecto = 2 AND
                    id_alumno = \'%s\' AND
                    id_proyecto = %d
            ', $oUser['userid'], $idProyecto);

        $res = $this->query($sqlPermisos);

        if($res[0][1] == 1){
            $username = $oUser['userid'];

            $sqlquery = sprintf("
                DELETE hitos
                FROM hitos
                JOIN proyectos_global ON hitos.id_proyecto = proyectos_global.id_proyecto
                WHERE
                    hitos.id_hito = '%s' AND
                    hitos.id_proyecto = '%s' AND
                    proyectos_global.id_alumno = '%s'
                ", $id_hito, $idProyecto, $username);

            $this->query($sqlquery);
```

Fig 59. Query de borrado de hitos

Las funciones de editar y borrar están disponibles mediante los botones de la última columna de la tabla.

Fig 60. Botonera de la tabla de hitos con las acciones de editar y borrar

En la parte inferior de la interfaz se encuentra el botón que da acceso a la interfaz de terminación del proyecto. No implica un cambio en el estado del proyecto.

6.2.4 Interfaz de terminación de trabajo y subida de archivos

La funcionalidad de esta interfaz es la de subir al servidor los distintos archivos que componen el proyecto e indicar al tutor que el proyecto está listo para ser revisado y propuesto para presentar.

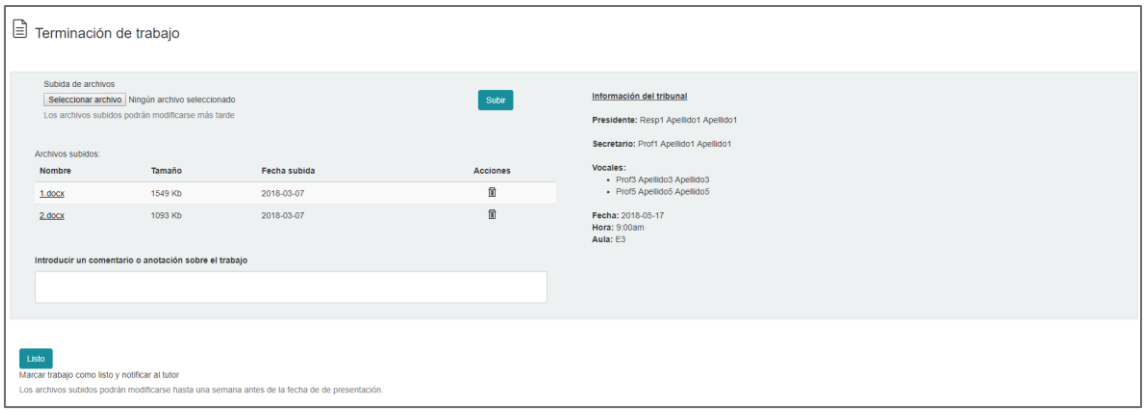

Fig 61. Interfaz de terminación de trabajo

El **módulo de gestión de archivos** permite la subida, descarga y borrado de archivos en el servidor.

Fig 62. Gestión de archivos del proyecto

Gestión de archivos:

Los archivos subidos por un usuario se ubican en una carpeta creada en la estructura de archivos del servidor con el identificador del usuario como nombre. Si esta carpeta no existe, es creada al momento. Al cargar el archivo, se guarda en una carpeta temporal del sistema y posteriormente se realizan una serie de comprobaciones.

Se comprueba si ha habido errores en la subida, si el archivo tiene un nombre nulo y si tiene un tamaño nulo. Si pasa estas validaciones se procede a crear la carpeta del usuario, en caso de que no exista ya, y se transfiere el archivo temporal a la nueva ruta. El archivo se renombra, añadiendo delante un *timestamp*. La ruta establecida para las descargas es */utils/uploads/{userid}/*.

```php
if (isset($_POST['submit'])) {

    if (empty($errors)) {
        // Comprobamos si hay archivo o está vacío
        if(!($fileName == '' || $fileSize == 0)){
            // Comprobamos si ya existe el archivo
            if(!file_exists($uploadPath)){
                // Compruebo si existe la ruta del usuario, y la creo
                if (!file_exists($folderPath)) {
                    mkdir($folderPath, 0777, true);
                }
                $didUpload = move_uploaded_file($fileTmpName, $uploadPath);
```

Fig 63. Subida de archivos

Una vez que el fichero ha sido correctamente movido a la carpeta de destino, se realiza un registro del archivo en la base de datos dentro de la entidad *archivos_proyecto*.

```php
$sql = sprintf('
        INSERT INTO archivos_proyecto
            (id_proyecto, nombre_archivo, tamano_archivo, fecha_subida, ruta)
        VALUES
            (%d, \'%s\', %d, \'%s\', \'%s\')
        ', $idProyecto, $fileName, $fileSize, $date, $filePath);
$this->query($sql);
```

Fig 64. Registro en base de datos del fichero cargado

De esta forma tenemos constancia de que el archivo ha sido subido al servidor y podremos listarlo cada vez que el usuario adecuado desee verlo.

Si por algún motivo se desea borrar el archivo, existe una función para ello que borrará el archivo tanto físicamente como lógicamente (en la base de datos). El proceso a seguir por el método de borrado es comprobar los permisos del usuario sobre el archivo en concreto, y sólo si se trata del mismo usuario que subió el archivo, proceder al borrado.

El borrado se realiza en tres pasos. Primero se obtiene de la base de datos la ruta al archivo. Después se comprueba que la ruta es correcta y, finalmente, se elimina el archivo (borrado físico). Por último, se elimina también de la base de datos (borrado lógico).

Paso 1:

```
// Obtenemos la ruta para borrar el archivo físico
$sqlPath = sprintf('
        SELECT
            ruta,
            nombre_archivo
        FROM
            archivos_proyecto AS f
        INNER JOIN proyectos_global AS g
            ON g.id_proyecto = f.id_proyecto
        WHERE
            g.id_alumno = \'%s\' AND
            id_archivo = %d
        ', $oUser['userid'], $fileId);
$r = $this->query($sqlPath);
$path = $r[0]['ruta'];
$name = $r[0]['nombre_archivo'];
```

Fig 65. Obtención de la ruta del fichero

Paso 2:

```
if(file_exists($pathToFile)){
    unlink($pathToFile);
}
```

Fig 66. Borrado físico del archivo

Paso 3:

```
// Después lo borramos de la tabla
$sql = sprintf('
    DELETE FROM archivos_proyecto
    WHERE
        id_archivo = %d AND
        id_proyecto = (
            SELECT
                id_proyecto
            FROM
                proyectos_global
            WHERE
                id_alumno = \'%s\'
        )
        ', $fileId, $oUser['userid']);
$this->query($sql);
```

Fig 67. Borrado lógico del archivo

Terminación de trabajo

La segunda funcionalidad de esta interfaz es la de marcar el trabajo como listo, de forma que será propuesto al tutor para que lo revise y apruebe, y pase a estar propuesto para la designación de un tribunal. Esto incluye también un área de texto donde el alumno puede incluir un comentario para el tutor.

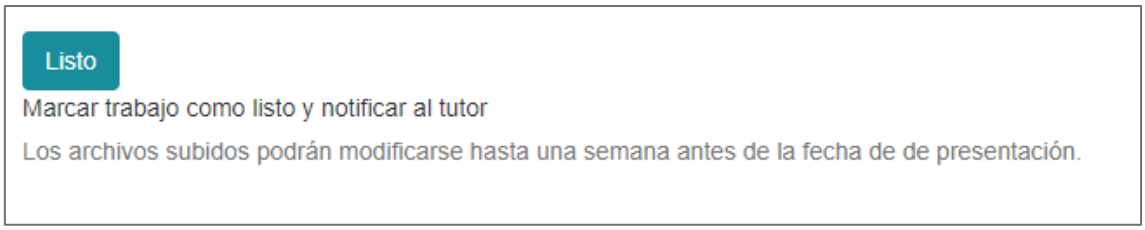

Fig 68. Funcionalidad de enviar el proyecto al tutor para su corrección

Internamente, se inserta una fila de datos en la entidad *proyectos_confirmar* de la base de datos, que incluye el identificador del proyecto y el comentario del alumno, y se notifica al tutor mediante un email.

Información del tribunal

Una vez que el tutor haya dado el visto bueno y confirmado el proyecto, este habrá pasado a estar propuesto para designar un tribunal. Cuando el jefe de estudios formalice dicho tribunal, el alumno podrá ver la información desde esta misma vista.

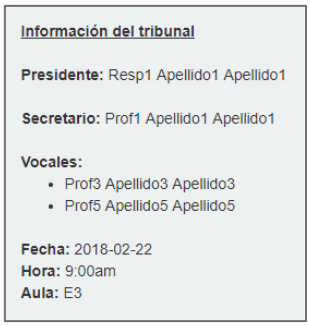

Fig 69. Información del tribunal

6.3 Interfaces desde el punto de vista de un profesor

6.3.1 Interfaz de inicio para profesores

Es la interfaz a la que se redirige a un profesor cuando inicia sesión o pulsa en el botón de Inicio. Consta de 3 partes: Interfaz de oferta de nuevo proyecto, lista de proyectos actualmente ofertados y lista de proyectos en curso.

Creación de nuevo proyecto

Se trata de un formulario que permite agregar un nuevo proyecto a la herramienta. Contiene una serie de campos que forman la información básica del proyecto:

- **Título:** Campo de texto libre, corresponde al título del TFG y tiene un máximo de 400 caracteres.
- **Descripción:** Campo de texto libre, corresponde a la descripción del TFG. Sin límite de caracteres.

- **Plan de estudios:** Campo de selección desplegable, se carga con la información de la tabla *planes_impartidos* y muestra los planes en los que imparte docencia el usuario.
- **Área:** Campo de selección desplegable, depende de la selección de Plan de estudios y carga desde la entidad *areas*.
- **Palabras clave:** Campo de texto libre, corresponde a las palabras clave del TFG y tiene un límite de 200 caracteres.

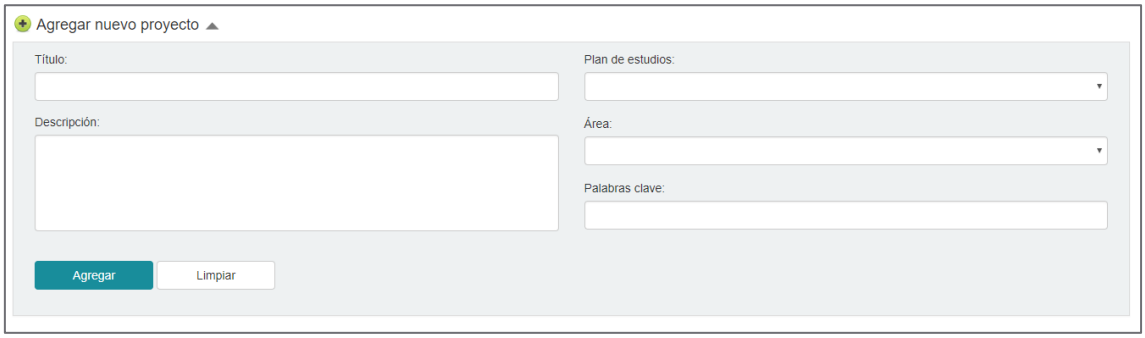

Fig 70. Formulario para la creación de un nuevo proyecto

El formulario tiene dos botones, Limpiar y Agregar. La función del botón de **Limpiar** es vaciar el texto de todos los campos y dejarlos en blanco.

El botón de **Agregar** lleva a cabo la función de dar de alta un proyecto en el sistema. Al pulsarlo se realiza una validación inicial en el lado del cliente para comprobar que se han rellenado todos los campos. Si esto es correcto, se envían los datos al servidor. En caso contrario se muestra un mensaje de error.

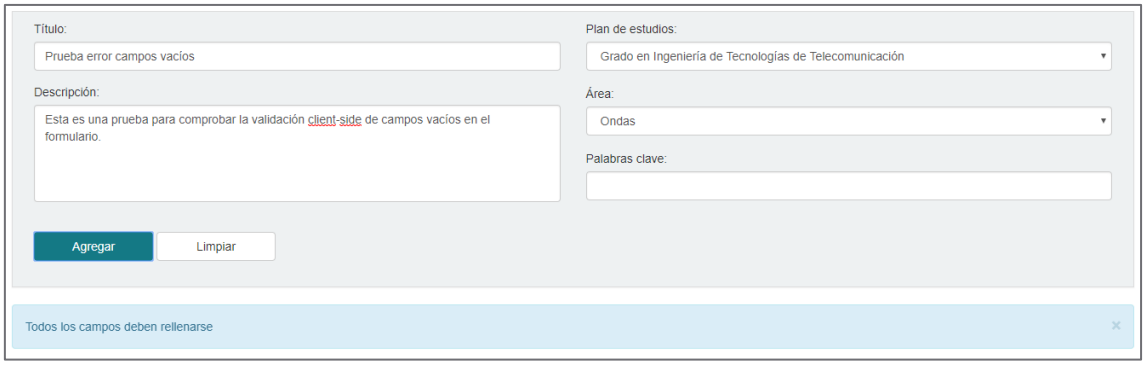

Fig 71. Validación de campos vacíos en el formulario

Los datos se envían al servidor mediante una llamada asíncrona Ajax. Una vez recibidos, en el controlador, se repite la validación de los campos vacíos. Si esta es correcta, se envían los parámetros del Tfg al modelo y este los guarda en la base de datos bajo la entidad *proyectos_global*.

```
//Construimos la query
$sql = sprintf('
        INSERT INTO proyectos_global
            (id_tutor, titulo_proyecto, desc_proyecto, id_plan, area_proyecto,
                keywords_proyecto, estado_proyecto, fecha_oferta_proyecto)
        VALUES (\'%s\', \'%s\', \'%s\', %d, \'%s\', \'%s\', 1, CURDATE());
', $userid, $titulo, $desc, $idplan, $area, $keywords);
try
{
    $success = $this->query($sql); //Ejecutamos query
```

Fig 72. Query para dar de alta un proyecto

Si el Insert en la base de datos se realiza correctamente, devuelve a la vista una respuesta con código 200. Si ha ocurrido un error, devuelve un código 400. La vista evalúa el código devuelto y muestra un mensaje de éxito o error.

Fig 70. Confirmación de que el proyecto ha sido creado

Una vez que el proyecto ha sido registrado en la base de datos, aparecerá en las búsquedas realizadas por los alumnos siempre que estén en el mismo plan de estudios al que va destinado el proyecto e incluyan los filtros adecuados en su búsqueda.

El profesor podrá ver, desde la sección de Proyectos ofertados, el nuevo proyecto que acaba de crear.

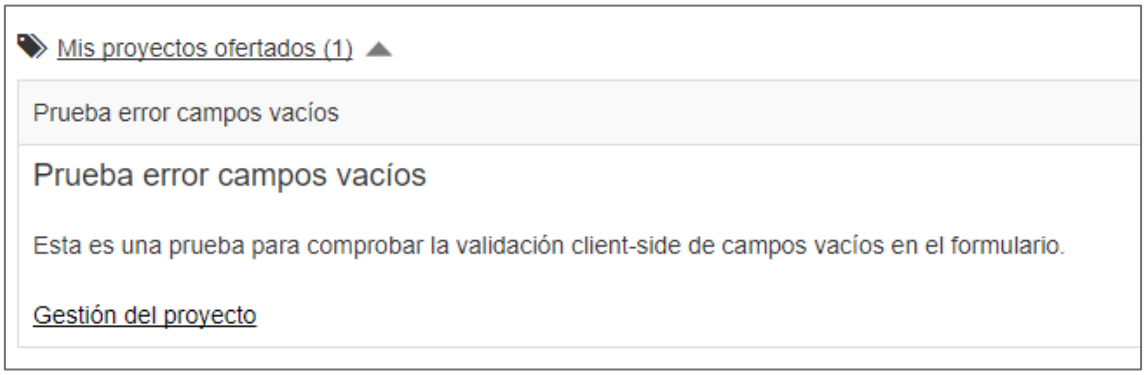

Fig 74. Vista de los proyectos ofertados por el usuario

Mediante el enlace, accederá a la interfaz de gestión del proyecto.

6.3.2 Interfaz de gestión de proyecto

Esta interfaz tiene varias funcionalidades. Permite ver y modificar la información del proyecto, ver los alumnos que han mostrado interés en él, comunicarse con ellos, y llegado el caso, asignar el proyecto a uno. Antes de cargar la vista, el controlador realiza una serie de comprobaciones y validaciones:

- Se filtra el id del proyecto para evitar inyecciones SQL o acciones no deseadas.
- Se comprueba que el estado del proyecto es 1 (ofertado), para evitar modificaciones sobre proyectos en curso.
- Se asegura que el usuario que pretende acceder a la vista es el director y responsable del proyecto.

Se compone de varias secciones:

La primera nos muestra la información del proyecto tal y como se dio de alta en la base de datos, y permite modificarla desde un formulario. La segunda es la relación de alumnos que han marcado el proyecto como favorito. Por último, la tercera indica si un alumno ha solicitado que se le asigne el proyecto, y permite realizar la asignación.

Modificación de la información del proyecto

Esta funcionalidad se implementa mediante un formulario que contiene los campos relativos a los parámetros del proyecto. Para mostrarlo, hay que pulsar en el botón de Editar.

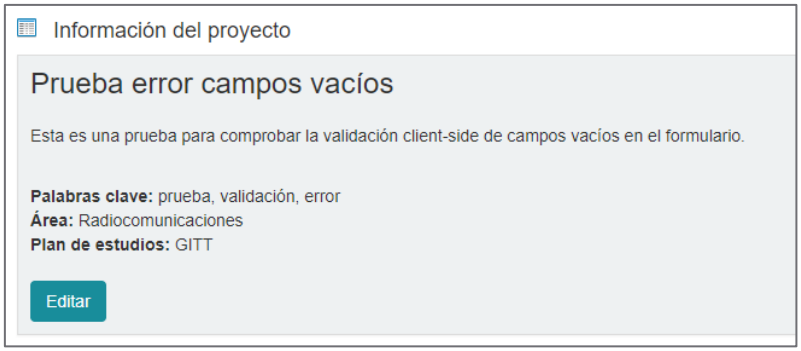

Fig 75. Se muestran los datos actuales del proyecto

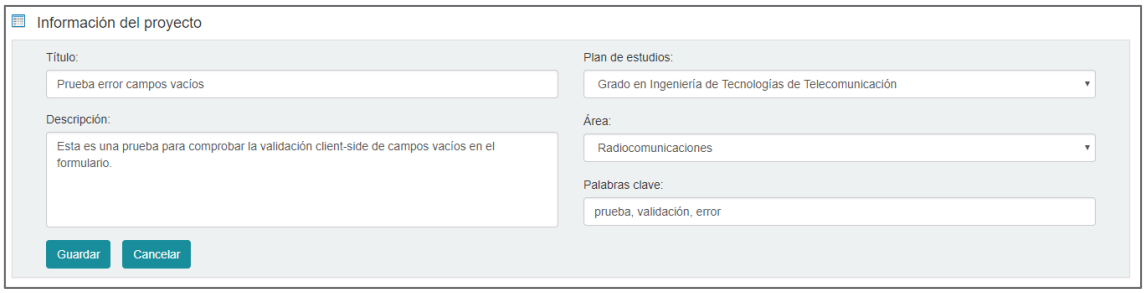

Fig 76. Formulario de edición del proyecto

Al aceptar, se realiza una doble validación (cliente y servidor) para asegurar que todos los campos están rellenos. Si se supera, se vuelve a validar que el usuario tenga permisos para

hacer la modificación, y se envían los datos al modelo que se encarga de ejecutar el cambio en la base de datos.

```
$permisos = $this->getPermisosUser($idProyecto, $oUser);

if($permisos){
    $success = $this->director->editProyecto($oUser, $idProyecto, $titulo, $desc, $plan_estudios, $area, $keywords);
```

Fig 77. Se valida que el usuario que modifica el proyecto sea el director del mismo

```
//Query sql
$sqlquery = sprintf("
            UPDATE proyectos_global
            SET ".$qt."
            WHERE
                id_proyecto = '%s' AND
                id_tutor = '%s'
        ", $idProyecto, $userid);
try
{
    $this->query($sqlquery);
```

Fig 78. Query que modificar el proyecto

Los campos a modificar se insertan parametrizados dentro de la consulta.

Lista de alumnos interesados en el proyecto

Esta funcionalidad se presenta mediante una lista que incluye los nombres de los alumnos que han agregado el proyecto a su lista de favoritos. Incluye la lista de nombres y un enlace para enviar un correo.

Alumnos interesados en el proyecto

Nombre	Acciones
Alumno5 Apellido5 Apellido5	Enviar email

Fig 79. Muestra los alumnos que han mostrado interés en el proyecto

La lista de alumnos se obtiene de la entidad *proyectos_favoritos* que relaciona un identificador de proyecto con uno o varios identificadores de alumnos y se cruza con la entidad *users* para consultar el nombre del usuario y su dirección de correo.

La dirección de envío se extrae de la entidad *users* y el envío se implementa con el método *mailto* de html, que al ser procesado por el navegador web interpreta que debe abrir el cliente de correo electrónico por defecto y preparar un correo a esa dirección.

```
/* Obtenemos los alumnos datos de favoritos*/
$sql = sprintf("
        SELECT
            fav.id_alumno,
            user.nombre,
            user.apellidos,
            user.correo
        FROM proyectos_favoritos AS fav
        INNER JOIN users AS user ON user.userid = fav.id_alumno
        WHERE
            fav.id_proyecto = %d

    ", $idProyecto, $userid);

$res = $this->query($sql);
```

Fig 80. Query que obtiene los alumnos interesados en el proyecto

Lista de solicitudes

En esta sección se mostrarán los alumnos que han solicitado que se les asigne el proyecto. Se visualizan en una tabla, con el nombre del alumno y un botón para asignarle el proyecto, que requiere confirmación en una ventana emergente.

Fig 81. Muestra los alumnos que han solicitado que se les asigne el proyecto

La lista de alumnos se obtiene de la entidad *proyectos*_solicitados que relaciona un identificador de proyecto con uno o varios identificadores de alumnos y se cruza con la entidad *users* para consultar el nombre del alumno.

```
/* Obtenemos los alumnos datos de solicitados*/
$sql = sprintf("
        SELECT
            sol.id_alumno,
            user.nombre,
            user.apellidos,
            user.correo
        FROM proyectos_solicitados AS sol
        INNER JOIN users AS user ON user.userid = sol.id_alumno
        WHERE
            sol.id_proyecto = %d
    ", $idProyecto, $userid);

$res = $this->query($sql);
```

Fig 82. Query que obtiene los alumnos que han solicitado el proyecto

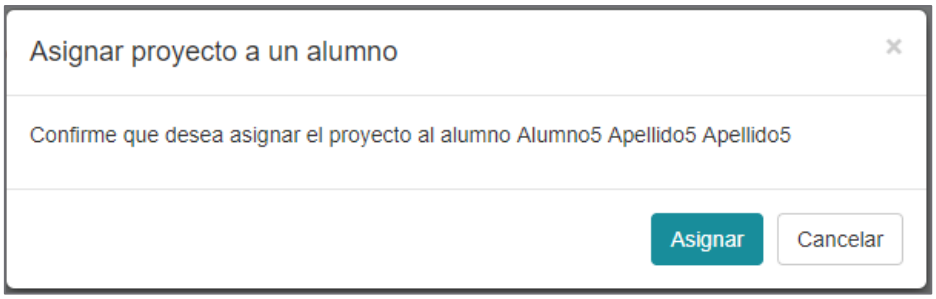

Fig 83. Confirmación para asignar el proyecto a un alumno

Al aceptar la confirmación para asignar el proyecto, se envían al controlador los identificadores del proyecto y del alumno y se realizan una serie de validaciones. Se comprueba:

- Que los identificadores enviados no estén vacíos.
- Que el usuario tenga permisos sobre el proyecto, es decir, que sea el director.
- Que el alumno no tenga otro proyecto en curso actualmente.

```
if(!empty($idAlumno) && !empty($idProyecto)){
    // Comprobar que el usuario tenga permisos sobre el proyecto
    $permisos = $this->getPermisosUser($idProyecto, $oUser);
    if($permisos){
        // Comprobar que el alumno no tenga otro proyecto asignado
        $asignado = $this->checkIfProyectoAsignado($idAlumno);
        if(!$asignado){
            $success = $this->director->asignarProyecto($idProyecto, $idAlumno);
            if($success == "ok"){
                // Borra el resto de solicitudes hechas por el alumno
                $this->director->cancelarSolicitudesAlumno($idAlumno);
```

Fig 84. Validaciones y proceso de asignar proyecto

Al pasar las validaciones, se asigna el proyecto al alumno y se procede a borrar el resto de solicitudes que pudiera tener el alumno en otros proyectos para evitar posibles complicaciones futuras.

El cambio de estado de 'ofertado' a 'en curso' se realiza cambiando el valor del campo *estado_proyecto* en la entidad *proyectos_global*. También se modifica el valor del campo *id_alumno*, antes vacío, con el identificador del alumno en cuestión, y se introduce la fecha actual en el campo *fecha_inicio_proyecto*.

```
$now = date('Y-m-d');
$sql1 = sprintf("
            UPDATE proyectos_global
            SET
                estado_proyecto = 2,
                id_alumno = '%s',
                fecha_inicio_proyecto = '%s'
            WHERE
                id_proyecto = '%s'
", $idAlumno, $now, $idProyecto);

try
{
    $success = $this->query($sql1);
```

Fig 85. Query del método asignarProyecto que da por iniciado el proyecto

Una vez asignado, se notifica por correo tanto al alumno como al tutor de que el proyecto se encuentra en curso.

6.3.3 Interfaz de seguimiento de proyecto

Esta interfaz tiene la funcionalidad de servir como medio de comunicación entre el tutor y el alumno, y permite al tutor seguir el desarrollo del proyecto en función de los datos que vaya añadiendo el alumno.

La vista consta de una sección ocultable en la que se muestra la información del proyecto, otra sección desde la que se enviaría un mail al alumno, y por último una sección en la que se muestran los hitos o logros que el alumno ha registrado, junto con su fecha.

Fig 86. Vista de la interfaz de seguimiento del proyecto

Al cargar la vista, se realiza la comprobación de que el usuario tenga permisos para ver el proyecto, es decir, que sea el director. A continuación, se comprueba que el proyecto en cuestión esté en curso, para evitar que se cuelen proyectos que todavía no están iniciados o que ya hayan sido finalizados.

```
$permisos = $this->checkPermisosUsuario($idProyecto, $oUser);
if($permisos){
    // Comprobar si proyecto activo
    $estado = $this->getEstado($idProyecto);
    if($estado == 2){
```

Fig 87. Validación de permisos del usuario para ver el seguimiento del proyecto

6.3.4 Interfaz de confirmación de trabajos finalizados

Esta interfaz tiene la funcionalidad de aprobar los trabajos que han sido enviados por el alumno. El director del proyecto revisará todos los documentos entregados por el alumno, y dará el visto bueno para que el proyecto pase a estar propuesto para la designación del tribunal que lo evaluará.

La vista consta de una tabla en la que se muestran los proyectos que el profesor puede confirmar o aprobar. Es decir, aquellos proyectos creados por él mismo, que se encuentran en curso, y con el flag de *proyecto_confirmado* con valor 'N', y han sido registrados en la entidad *proyectos_confirmar*. Desde la tabla, el profesor podrá ver los archivos que ha subido el alumno, así como si ha añadido un comentario.

Fig 88. Vista de la interfaz de confirmación de trabajos terminados

Al hacer click en uno de los archivos, se gestionará la descarga según lo explicado en el apartado 5.6.4 de gestión de descargas.

El botón de Confirmar tendrá la función marcar el proyecto como confirmado y pasarlo al siguiente punto del proceso. A nivel de base de datos, se realiza en dos pasos: actualizar el flag *proyecto_confirmado* a 'S' y eliminar el proyecto de la entidad *proyectos_confirmar*, pues ya ha sido confirmado.

Para garantizar la integridad de los datos, se ha empleado una **transacción** que permite controlar que las dos queries se ejecutan correctamente. En caso de que una de las dos falle, se revierte la operación. De esta forma aseguramos que no se produzca ninguna incongruencia en los datos.

```
$db = $this->aQuery();
$db->beginTransaction();

$sql = sprintf('
            UPDATE
                proyectos_global
            SET
                proyecto_confirmado = \'S\'
            WHERE
                id_proyecto = %d AND
                id_tutor = \'%s\'
        ', $idProyecto, $oUser['userid']);

$query = $db->prepare($sql);
$query->execute();

$sql = sprintf('
        DELETE pc
        FROM proyectos_confirmar AS pc
        INNER JOIN proyectos_global AS pg ON pg.id_proyecto = pc.id_proyecto
        WHERE
            pc.id_proyecto = %d AND
            pg.id_tutor = \'%s\'
        ', $idProyecto, $oUser['userid']);

$query = $db->prepare($sql);
$query->execute();

$db->commit();
```

Fig 89. Query de confirmación del trabajo enviado por el alumno

6.3.5 Interfaz de participación en tribunales

Esta interfaz tiene la funcionalidad de mostrar al usuario los tribunales en los que ha sido designado como miembro, y permitir el acceso al panel de calificaciones. Se presenta mediante una tabla que mostrará la información fundamental del tribunal, como el título del proyecto a presentar, el alumno que lo presenta, el director del proyecto, el rol que va a ocupar el usuario (presidente, secretario o vocal) y por último la fecha, hora y lugar de la presentación.

La búsqueda en base de datos incluye un filtro temporal, para buscar tribunales que correspondan a fechas con límite en el día anterior. De esta forma, no vemos tribunales que ya han sucedido, pero podemos ver los que han sucedido en el mismo día hace tan solo unas horas.

```
$sql = sprintf('
        SELECT
            t.id_tribunal,
            t.fecha_tribunal,
            t.hora_tribunal,
            t.aula,
            g.titulo_proyecto,
            alumno.nombre AS nombreAlumno,
            alumno.apellidos AS apellidosAlumno,
            tutor.nombre AS nombreTutor,
            tutor.apellidos AS apellidosTutor,
            f.id_archivo,
            f.nombre_archivo,
            f.tamano_archivo
        FROM
            tribunales AS t
        INNER JOIN proyectos_global AS g ON g.id_proyecto = t.id_proyecto
        INNER JOIN users AS alumno ON alumno.userid = g.id_alumno
        INNER JOIN users AS tutor ON tutor.userid = g.id_tutor
        LEFT JOIN miembros_tribunal AS miembros ON miembros.id_tribunal = t.id_tribunal
        LEFT JOIN archivos_proyecto AS f ON f.id_proyecto = t.id_proyecto
        WHERE
            t.fecha_tribunal > CURDATE() - INTERVAL 1 DAY AND
            (t.id_presidente = \'%s\' OR miembros.id_miembro = \'%s\')
        ', $oUser['userid'], $oUser['userid']);
$resProy = $this->query($sql);
```

Fig 90. Query para obtener la lista de tribunales de los que el usuario forma parte

Los cruces con las entidades *miembros_tribunal* y *archivos_proyecto* se realiza con el tipo *LEFT*, para que, en caso de no haber datos relacionados en estas dos entidades, la búsqueda devuelva el resto de resultados.

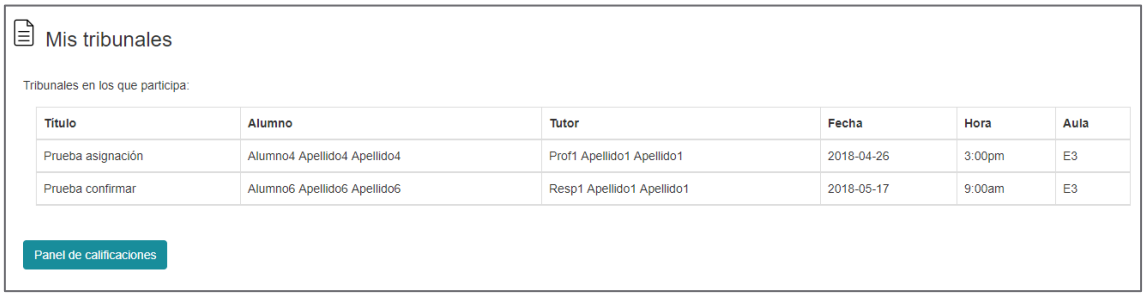

Fig 91. Tabla que muestra los tribunales en los que participa el usuario

Cada fila de la tabla incluye un campo oculto desplegable donde se puede ver una descripción más ampliada del proyecto en cuestión y contiene una lista con los ficheros del proyecto, que previamente habrá cargado el alumno, para que, como miembro del tribunal, pueda disponer de ellos. De esta forma, los miembros del tribunal podrán acceder a todo lo que forme parte del proyecto, como la memoria, documentos técnicos, planos, o código. Esta lista de ficheros se obtiene cruzando la tabla *archivos_proyecto* con la tabla *tribunales* a través del campo *id_proyecto*. Si el usuario pertenece al tribunal de evaluación de un proyecto, la búsqueda devolverá los ficheros asociados.

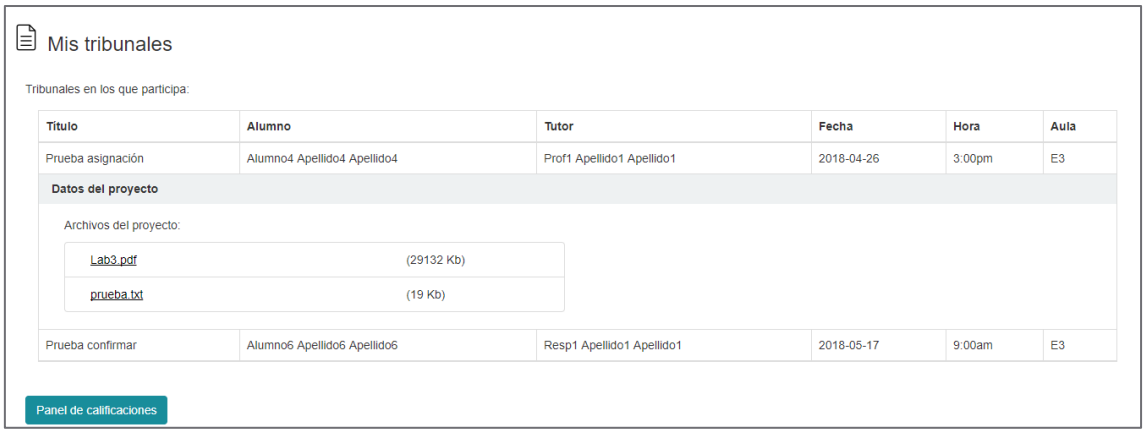

Fig 92. Lista de ficheros a descargar del proyecto que va a evaluarse

La descarga de los archivos subidos se realiza según lo explicado en el apartado 5.6.4.

La función de calificar el proyecto estará disponible solo si el usuario tiene el rol de presidente del tribunal, y durante el mismo día de la presentación. Al hacer click en el enlace se redirecciona hacia la interfaz de calificaciones.

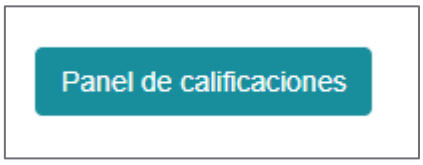

Fig 93. Enlace para acceder a la interfaz de calificaciones

6.3.6 Interfaz de Calificaciones

La funcionalidad de esta interfaz es permitir que el presidente de un tribunal pueda evaluar el proyecto introduciendo la calificación del alumno en la aplicación. La vista consiste en una tabla en la que se muestran los proyectos para los cuales el usuario está autorizado a introducir una calificación. También se aplica un filtro temporal para que solamente se muestren los tribunales con fecha desde los 7 días anteriores al día actual. De esta forma se permite que la calificación del proyecto pueda editarse durante la semana siguiente a la presentación.

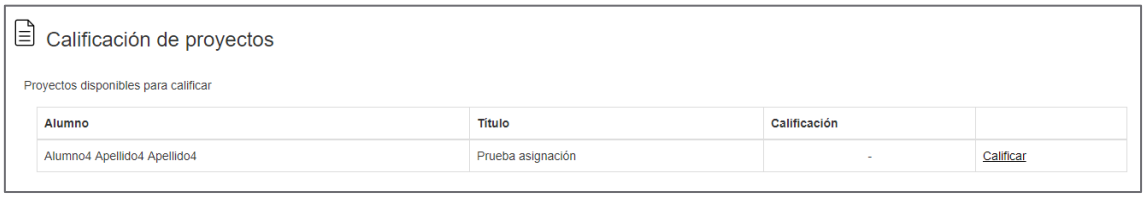

Fig 94. Muestra la lista de proyectos disponibles para calificar

El valor se introduce mediante una ventana emergente con un campo de texto.

Fig 95. Ventana que permite introducir la calificación del proyecto

Al guardar una calificación, se verifica el valor enviado con el validador para números de tipo *float*.

```
if(isset($_POST['calificacion'])){
    $calificacion = filter_var($calificacion, FILTER_SANITIZE_NUMBER_FLOAT);
    $calificacion = str_replace('+', '', $calificacion);
    $calificacion = str_replace('-', '', $calificacion);
}
```

Fig 96. Validación de la calificación introducida

Este método de filtrado permite los signos de adición y substracción (‘+’ y ‘-‘), así que se eliminan de forma manual.

A continuación, y siguiendo la dinámica del resto de la aplicación, se valida que el usuario tenga los permisos necesarios para realizar esa operación. De ser así, se comprueba si se trata de introducir la calificación por primera vez, o si por el contrario se está modificando una calificación que ya existía. Se ejecuta una acción distinta para cada caso.

```
if($permisos){
    // Se trata de añadir o editar la calificación
    $exists = $this->calificar->checkIfExists($idProyecto);
    if($exists == true){
        $res = $this->calificar->editCalificacion($idProyecto, $calificacion);
    } else {
        $res = $this->calificar->addCalificacion($idProyecto, $calificacion);
    }
}
```

Fig 97. Validación de permisos y distinción entre añadir o editar una calificación

Para inserción:

```
$sqlquery = sprintf("
    INSERT INTO calificaciones (id_proyecto, calificacion)
    VALUES(%d, %d)
    ", $idProyecto, $calificacion);

$this->query($sqlquery);
```

Fig 98. Query que crea una calificación para el proyecto

Para edición:

```
$sqlquery = sprintf("
    UPDATE calificaciones
    SET
        calificacion = %s
    WHERE id_proyecto = %s
    ", $calificacion, $idProyecto);
try
{
    $this->query($sqlquery);
```

Fig 99. Query que edita una calificación ya introducida

Cuando la calificación ha sido agregada se considera que el proyecto ha sido finalizado. Se procede a actualizar el flag *estado_proyecto* a 3 (finalizado) y el campo *fecha_fin* con la fecha actual.

```
$sql = sprintf('
        UPDATE
            proyectos_global
        SET
            estado_proyecto = 3,
            fecha_fin = now()
        WHERE
            idProyecto = %d
        ', $idProyecto);
$this->query($sql);
```

Fig 100. Query que establece el proyecto como finalizado

A continuación, se informa al alumno y al director del proyecto de la calificación obtenida mediante un correo electrónico.

6.4 Desde el punto de vista de un jefe de estudios

Un usuario con *groupid* 30, es decir, jefe de estudios o responsable de titulación, puede acceder a las mismas interfaces que un usuario con *groupid* 20 (docentes). Además de estas, también tiene acceso a unas interfaces que solamente están disponibles para ese tipo de usuarios. Estas interfaces son:

6.4.1 Interfaz de designación de tribunal

Esta interfaz tiene la funcionalidad de permitir al responsable de la titulación designar un tribunal para cada proyecto, esto es, seleccionar los distintos miembros y otros datos como la fecha, hora y lugar.

La composición de un tribunal depende del plan de estudios, y el encargado de designar dicho tribunal deberá ceñirse a lo dictado por la normativa de su titulación. Se presupone que cada usuario responsable de una titulación está al tanto de la normativa que debe aplicar. Aun así, la aplicación controla que los datos que se introducen se ajusten a la normativa estipulada. Para esto es necesario tener una tabla de parametrización de los tribunales en función de cada plan de estudio.

Ante la diversidad de normativas diferentes, nos hemos centrado en la de la titulación de Grado en Ingeniería de Tecnologías de Telecomunicación. Esta normativa ha sufrido cambios recientemente. Según la normativa anterior, el secretario sería el director del TFG, el presidente sería designado por el alumno y el vocal se seleccionaría por orden.

Con la nueva normativa, se designan por un lado los miembros y después se asignan los roles en función de su rango y edad. El miembro de mayor rango y edad tendría el rol de presidente. El miembro con menor rango y edad ocuparía el papel de secretario. El miembro restante, sería el vocal.

Esta interfaz de designación de tribunales está basada en la normativa anterior, cuando cada miembro se seleccionaba según un criterio. En el apartado 7.6 de desarrollos futuros se explica el plan de adaptación a la normativa vigente y los requerimientos técnicos que supondría.

La vista consta de una sección que sirve de filtro para realizar búsquedas, y de una sección con una tabla que mostrará los resultados de dichas búsquedas.

Fig 101. Formulario de búsqueda de proyectos para la designación de un tribunal

Se permite filtrar por los siguientes parámetros:

- Título del proyecto: Campo de texto libre.
- Nombre del alumno que lo realiza: Campo de texto libre.
- Nombre del tutor: Campo de texto libre.
- Plan de estudios: Combo seleccionable que carga desde la entidad *planes_resp*.
- Tribunal asignado: Combo seleccionable con valores S/N.

Al realizar una búsqueda, la vista envía al controlador todos los parámetros seleccionados en el filtro, y el controlador realiza un filtrado general para evitar posibles inyecciones SQL, como se hace en el resto de la aplicación.

```
if(isset($_POST['tituloProyecto'])){
    $tituloProyecto = filter_var($_POST['tituloProyecto'], FILTER_SANITIZE_STRING);
} else {
    $tituloProyecto = "";
}
```

Fig 102. Filtrado de los datos introducidos en el buscador

Una vez en el modelo, se construye la query SQL incluyendo los campos que se han usado en el filtro y excluyendo aquellos que se hayan dejado vacíos.

```
if(!empty($nombreTutor)){
    $sqlquery .= " AND concat_ws(' ', tutor.nombre, tutor.apellidos) LIKE '%".$nombreTutor."%'";
}
if(!empty($nombreAlumno)){
    $sqlquery .= " AND concat_ws(' ', alumno.nombre, alumno.apellidos) LIKE '%".$nombreAlumno."%'";
}
```

Fig 103. Creación de la query que buscará solo a partir de los filtros introducidos

En la tabla podemos ver los campos relativos al título del proyecto, nombre del alumno, nombre del director, plan de estudios, tribunal asignado (s/n) y fecha de presentación.

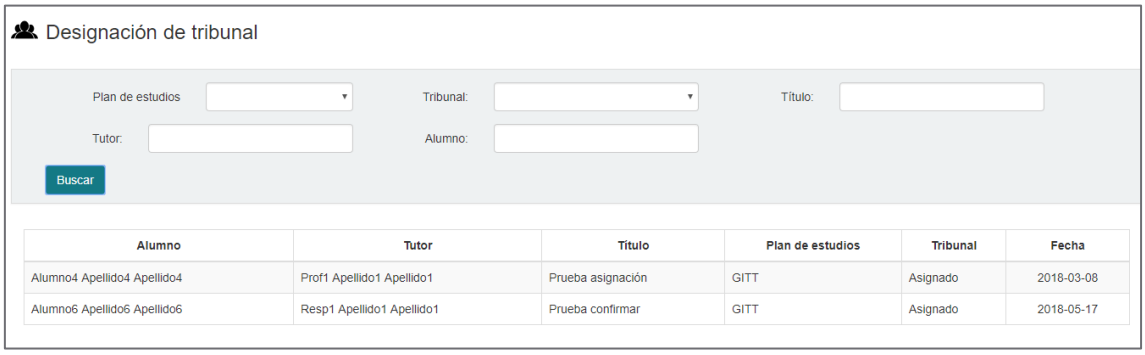

Fig 104. Buscador con su tabla de resultados

Cada fila contendrá un cuadro desplegable en el que se mostrará la información detallada. En concreto, indicará los miembros que componen el tribunal, es decir, el presidente, el secretario y los vocales. También incluye la fecha, hora y lugar de presentación. Todos estos campos partirán de un estado vacío y serán configurables por el jefe de estudios.

Fig 105. Formulario para la designación de un tribunal, sus miembros, lugar, fecha y hora

La composición de un tribunal depende del plan de estudios, y el encargado de designar dicho tribunal deberá ceñirse a lo que dicta la normativa de su titulación. Se presupone que cada usuario responsable de una titulación está al tanto de la normativa que debe aplicar. Aun así, la aplicación controla que los datos que se introducen se ajusten a la normativa estipulada. Para esto es necesario tener una tabla de parametrización de los tribunales en función de cada plan de estudio.

Tabla de parametrización

Se utiliza la entidad *parametros_tribunales* para establecer la parametrización que permitirá a la aplicación realizar las acciones y validaciones necesarias para cada plan de estudios.

Recordamos la estructura de la entidad:

- **Id_plan**: Identifica al plan de estudios.
- **Num_vocales**: Número de vocales en el tribunal.
- **Presidente**: Parámetro de modo de selección de presidente.
- **Secretario**: Parámetro de modo de selección de secretario.
- **Vocal**: Parámetro de modo de selección de vocales.

Esto es, para cada plan de estudios, especificamos cómo se van a elegir el presidente, el secretario y los vocales y, además, indicamos la cantidad de éstos últimos.

Los modos de selección que se han contemplado son los siguientes:

Director:

Significa que el director o tutor del proyecto ocupará un determinado puesto del tribunal. Al abrir el modal de selección, solo podremos ver el nombre del director del proyecto, ya que será el único usuario que podamos seleccionar.

Se obtendrá de la base de datos, bajo la entidad *proyectos_global* el identificador del tutor asociado al proyecto y se cruzará con la entidad *users* para obtener el nombre completo.

```
$sqlIdDirector = sprintf('
        SELECT
            id_tutor AS userId,
            CONCAT(users.nombre, \' \', users.apellidos) AS nombreCompleto
        FROM
            proyectos_global
        INNER JOIN users AS users ON users.userid = proyectos_global.id_tutor
        WHERE
            id_proyecto = %d
        ', $idProyecto);
$idDirector = $this->query($sqlIdDirector);
```

Fig 106. Query para la selección del director del proyecto

Orden:

La aplicación registra cada vez que un usuario participa en un tribunal. Al usar este modo de selección, se designará al usuario que corresponda según un orden FIFO.

Esta funcionalidad requiere de su propia entidad de control, llamada *orden_seleccion_miembros*. Recordamos su estructura:

- **index**: Índice numérico que permite llevar el control de cuando se ha registrado cada usuario en la entidad.
- **id_plan**: Identificador del plan de estudios al que pertenece.
- **userid**: Identificador del usuario.

Cuando se designan los componentes de un tribunal, se borra de esta tabla a cada participante, y se les vuelve a añadir incrementando en uno el valor del mayor *index*. Cuando se realiza la designación de un miembro según este modo, se realiza una búsqueda en la tabla de control filtrando por plan de estudios y escogiendo el *index* más bajo que encontremos. De esa forma se obtiene el identificador del primer usuario que fue insertado en la lista, que será al que le corresponde participar en el tribunal.

```
/* Se actualiza la tabla orden_seleccion_miembros para la selección automática por orden */
$idPlan = $this->getIdPlanByProject($idProyecto);
$this->tribunal->updateTablaOrden($idPlan, $listaVocales);
```

Fig 107. Al seleccionar miembros para el tribunal, se actualiza la tabla que los registra por orden

```
$sqlOrden = sprintf('
        SELECT
            users.userid,
            CONCAT(users.nombre, \' \', users.apellidos) AS nombreCompleto
        FROM
            users
        INNER JOIN planes_impartidos AS pi ON pi.userid = users.userid
        INNER JOIN proyectos_global AS proy ON proy.id_plan = pi.id_plan
        INNER JOIN orden_seleccion_miembros AS orden ON orden.userid = users.userid
                AND orden.id_plan = proy.id_plan
        WHERE
            proy.id_proyecto = %d AND
        ORDER BY orden.index ASC
        LIMIT 1

        ', $idProyecto);
$listaIds = $this->query($sqlOrden);
```

Fig 108. Query que selecciona a los miembros del tribunal según un orden FIFO

Aleatorio:

En este modo, se selecciona cada vez un usuario al azar que cumpla el criterio de impartir docencia en ese plan de estudios.

```
// Aleatorio. Se escoge al azar entre los miembros del plan de estudios
$sqlAleatorio = sprintf('
        SELECT
            users.userid,
            CONCAT(users.nombre, \' \', users.apellidos) AS nombreCompleto
        FROM
            users
        INNER JOIN planes_impartidos AS pi ON pi.userid = users.userid
        INNER JOIN proyectos_global AS proy ON proy.id_plan = pi.id_plan

        WHERE
            id_proyecto = %d
        ', $idProyecto);
$oListaCompleta = $this->query($sqlAleatorio);
$rand = 's';
```

Fig 109. Query para la selección de miembros de forma aleatoria

Se obtiene la lista de posibles candidatos y se asigna un valor al flag *rand* que luego se usará para escoger uno de los candidatos aleatoriamente. El motivo por el que la aleatoriedad no se aplica en este momento es que la lista de candidatos debe ser filtrada para evitar que haya usuarios que no deban aparecer porque ocupan otro rol dentro del tribunal, y esta comprobación se hace en tiempo real mientras el usuario selecciona los miembros. Mediante el flag *rand* indicamos al front-end que la selección será aleatoria.

Elegible:

Este modo significa que el miembro es elegido por el alumno, que comunicará su decisión al jefe de estudios y éste será el encargado de introducirlo en el tribunal. Al poder seleccionar a cualquier docente de la titulación, la búsqueda a realizar consiste en cruzar la entidad *users* con la entidad *planes_impartidos* para el identificador de plan que corresponda.

```
$sqlElegible = sprintf('
        SELECT
            users.userid,
            CONCAT(users.nombre, \' \', users.apellidos) AS nombreCompleto
        FROM
            users
        INNER JOIN planes_impartidos AS pi ON pi.userid = users.userid
        INNER JOIN proyectos_global AS proy ON proy.id_plan = pi.id_plan

        WHERE
            id_proyecto = %d
        ', $idProyecto);
$listaIds = $this->query($sqlElegible);
```

Fig 110. Query para la selección de modales elegible por el alumno

Manual:

En este caso, se permite al jefe de estudios designar al usuario que desee, siempre que esté pertenezca al plan de estudios. Una vez más, consiste en un cruce de la entidad *users* con la entidad *planes_impartidos*.

```
$sqlListaCompleta = sprintf('
        SELECT
            users.userid,
            CONCAT(users.nombre, \' \', users.apellidos) AS nombreCompleto
        FROM
            users
        INNER JOIN planes_impartidos AS pi ON pi.userid = users.userid
        INNER JOIN proyectos_global AS proy ON proy.id_plan = pi.id_plan

        WHERE
            id_proyecto = %d
        ', $idProyecto);
$listaIds = $this->query($sqlListaCompleta);
```

Fig 111. Query para la selección de miembros de forma manual

En cualquiera de los casos, después de obtener la lista de ids disponibles, se comprobará que la lista no incluya usuarios reservados para un rol en concreto. Por ejemplo, si en una titulación el director del proyecto hace de presidente del tribunal, su nombre no podrá aparecer entre los posibles secretarios o vocales. El objetivo es que todos los datos que muestre la aplicación sean correctos y no haya ninguna incongruencia entre lo que el usuario puede ver y lo que la aplicación permite.

```
// Se comprueba que no haya Ids que no deban estar, por estar reservados para otro rol
if($oParams[0]['presidente'] == 'director' || $oParams[0]['secretario'] == 'director'){
    $idDirector = $this->getTutorByProyecto($idProyecto);
    foreach($listaIds as $key=>$id){
        if($id['userId'] == $idDirector){
            array_splice($listaIds, $key, 1);
        }
    }
}
```

Fig 112. Validación de miembros en roles reservados

Selección de miembros

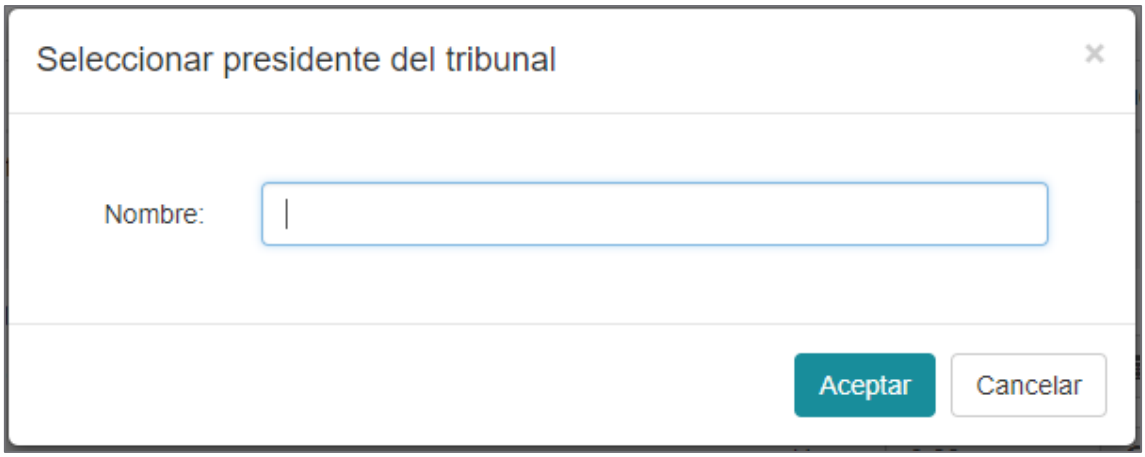

Fig 113. Ventana de selección de miembros para el tribunal

La selección de miembros se realiza desde la ventana mostrada en la figura 113, que consta de un campo de texto libre con función de auto relleno y sugerencias, es decir, según vamos escribiendo nos va mostrando los nombres que podemos seleccionar a partir del texto que tenemos escrito.

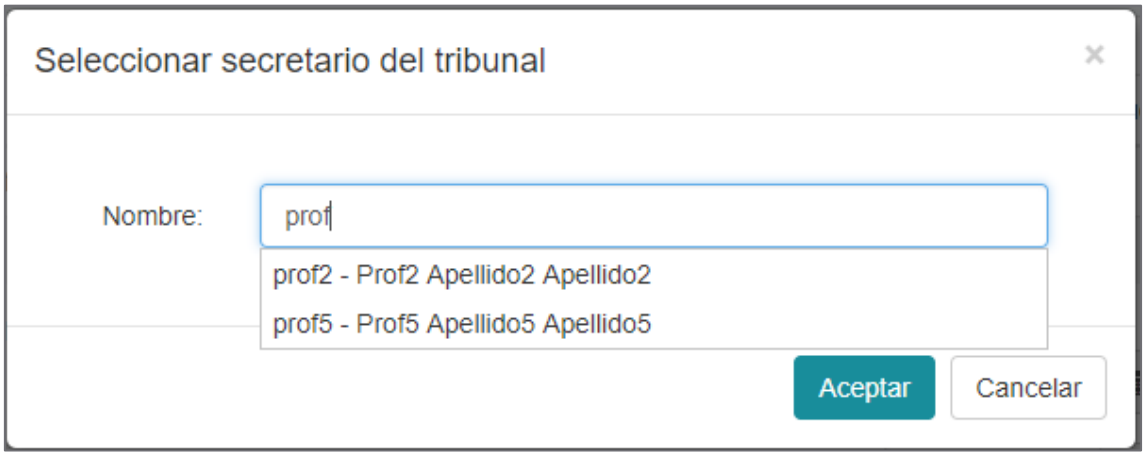

Fig 114. Ventana de selección de miembros con función de autocompletar

En los casos en los que la selección es fija como, por ejemplo, la designación por orden o cuando el miembro debe ser el director del TFG, la lista mostrará la única opción disponible para elegir y no permitirá seleccionar ninguna otra, ni escribiendo manualmente el nombre completo. En este caso se mostrará un mensaje de error (figura 115)

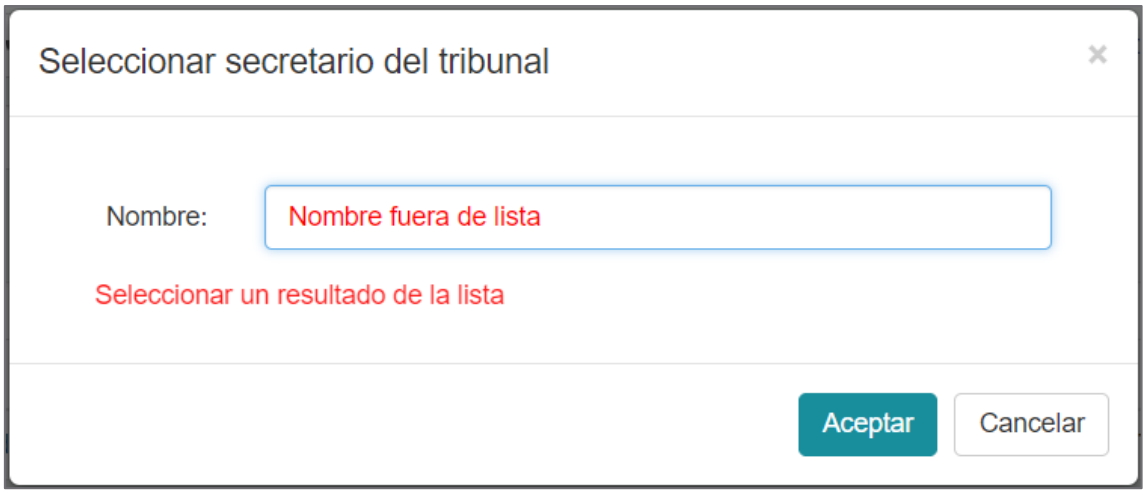

Fig 115. Error al tratar de seleccionar un miembro no disponible

Selección de momento y lugar

Para la designación de la fecha y la hora, los miembros seleccionados del tribunal deberán ponerse de acuerdo entre ellos y comunicar su elección al jefe de estudios. Como los tribunales son editables hasta la fecha de presentación, el jefe de estudios podrá modificar la fecha y hora elegidas para cuadrar los horarios de todos los miembros.

En la sección derecha del desplegable encontramos los campos para establecer la fecha, hora y el lugar de la presentación. La selección de fecha y hora se ha resuelto mediante un componente *datepicker* y un *timepicker*.

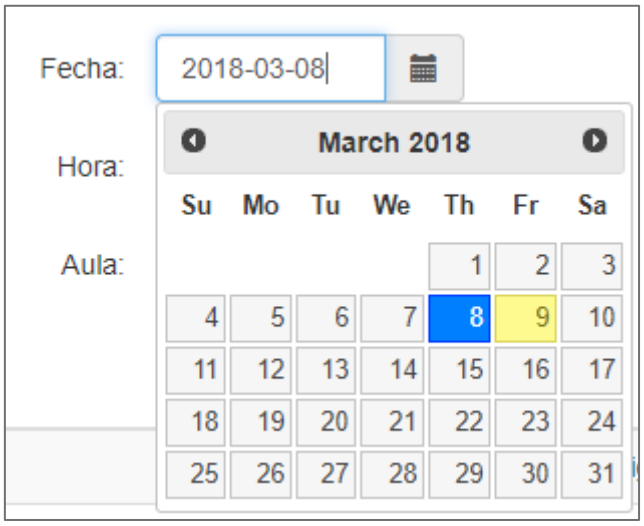

Fig 116. Componente 'datepicker' para seleccionar la fecha

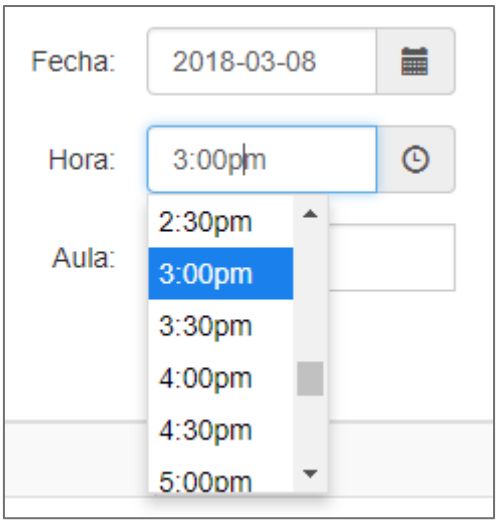

Fig 117. Componente 'timepicker' para seleccionar la hora

Envío de datos y validaciones

Para poder enviar los datos al servidor es necesario pasar una serie de validaciones para mantener la integridad de datos en relación a la parametrización del plan de estudios.

La primera de todas, una validación 'en caliente', comprueba que todos los campos estén rellenos y que el número de vocales sea el adecuado. Si faltan miembros para cumplir el requisito del tribunal, no se permitirá usar el botón de aceptar. La validación 'en caliente' significa que se realiza de forma dinámica, sin tener que realizar la acción de guardar los datos. Por este motivo, generalmente se realiza de forma local utilizando un lenguaje de *front*. En este caso, se ha creado el método *controlSaveBtn* que será invocado cada vez que uno de los campos sea modificado, y se encargará de las comprobaciones de cada campo. En función del resultado de dichas comprobaciones, activa o desactiva el botón de Aceptar tribunal.

Fig 118. Diferentes estados del botón de aceptar según el resultado de la validación dinámica

Cuando el usuario intenta guardar los datos mediante el botón de aceptar, se realiza una doble validación *front end/back end*. La validación en el *front-end* es sencilla y se encarga de que no se envíen campos vacíos o incongruencias graves. Es el *back-end* el que se encarga de que la integridad de los datos se mantenga y no se introduzcan datos que no cumplan los requisitos de la normativa del plan de estudios.

```
$idProyecto = filter_var($data['idProyecto'], FILTER_SANITIZE_NUMBER_INT);
$idPresidente = filter_var($data['presidente'], FILTER_SANITIZE_STRING);
$idSecretario = filter_var($data['secretario'], FILTER_SANITIZE_STRING);
$listaVocales = filter_var_array($data['vocales'], FILTER_SANITIZE_STRING);
$fecha = filter_var($data['fecha'], FILTER_SANITIZE_STRING);
$hora = filter_var($data['hora'], FILTER_SANITIZE_STRING);
$aula = filter_var($data['aula'], FILTER_SANITIZE_STRING);
```

Fig 119. Filtrado de datos introducidos para evitar las inyecciones de SQL

La validación específica se realiza en tres partes. Validación del presidente del tribunal, validación del secretario y validación de los vocales. Para los dos primeros casos, se comprueba si ese puesto debía ser ocupado por el director del proyecto, y posteriormente, si el usuario que ocupa el puesto pertenece al plan de estudios.

```
// Validar presidente
if($oCriterios[0]['presidente'] == 'director'){
    // Validamos que el presidente sea el director del proyecto
    $director = $this->getTutorByProyecto($idProyecto);
    if($director['id'] != $idPresidente){
        $validacion = false;
    }
} else {
    // Validamos que el presidente pertenezca al plan de estudios
    $pertenece = $this->checkPlanImpartidoByProyecto($idPresidente, $idProyecto);
    if(!$pertenece){
        $validacion = false;
    }
}
```

Fig 120. Validación del presidente

```
// Validar secretario
if($oCriterios[0]['secretario'] == 'director'){
    // Validamos que el secretario sea el director del proyecto
    $director = $this->getTutorByProyecto($idProyecto);
    if($director['id'] != $idSecretario){
        $validacion = false;
    }
} else {
    // Validamos que el secretario pertenezca al plan de estudios
    $pertenece = $this->checkPlanImpartidoByProyecto($idSecretario, $idProyecto);
    if(!$pertenece){
        $validacion = false;
    }
}
```

Fig 121. Validación del secretario

```php
// Validar vocales
// Numero de vocales
$numVocales = $oCriterios[0]['num_vocales'];
if($numVocales != sizeof($listaVocales)){
    $validacion = false;
}
// Validamos que los vocales pertenezcan al plan de estudios
foreach($listaVocales as $vocal){
    $pertenece = $this->checkPlanImpartidoByProyecto($vocal, $idProyecto);
    if(!$pertenece){
        $validacion = false;
    }
}
```

Fig 122. Validación de vocales

Si todos los parámetros son correctos, procedemos a guardar los datos. Si el tribunal ya estaba formado y se ha realizado un cambio, solo se actualiza.

```php
// El tribunal ya existía, hacemos un Update
$idTribunal = $exists;
$a = $this->tribunal->updateTribunal($idProyecto, $idTribunal, $idPresidente, $idSecretario, $fecha, $hora, $aula);
$b = $this->tribunal->updateVocales($idTribunal, $listaVocales);
```

Fig 123. Actualización del tribunal

```php
$sql = sprintf('
        UPDATE tribunales
        SET
            id_presidente = \'%s\',
            id_secretario = \'%s\',
            fecha_tribunal = \'%s\',
            hora_tribunal = \'%s\',
            aula = \'%s\'
        WHERE id_tribunal = %d AND id_proyecto = %d
        ', $id_presidente, $id_secretario, $fecha, $hora, $aula, $id_tribunal, $idProyecto);
$this->query($sql);
```

Fig 124. Query de actualización del tribunal

Si no existía previamente, se inserta en la base de datos.

```php
$idTribunal = $this->tribunal->getMaxIdTribunal();
// Se guardan los datos del tribunal en las tablas tribunales y vocales
$a = $this->tribunal->crearTribunal($idProyecto, $idTribunal, $idPresidente, $idSecretario, $fecha, $hora, $aula);
$b = $this->tribunal->asignarVocales($idTribunal, $listaVocales);
```

Fig 125. Creación del tribunal

```
$sql = sprintf('
        INSERT INTO tribunales
            (id_tribunal, id_proyecto, fecha_tribunal, hora_tribunal, id_presidente, aula, id_secretario)
        VALUES
            (%d, %d, \'%s\', \'%s\', \'%s\', \'%s\', \'%s\')
        ', $id_tribunal, $idProyecto, $fecha, $hora, $id_presidente, $aula, $id_secretario);
$this->query($sql);
```

Fig 126. Query de creación del tribunal

La relación de vocales asignados al tribunal se guarda en la entidad *vocales*.

```
// Insert
foreach($listaVocales as $vocal){
    $sqlInsert = sprintf('
        INSERT INTO vocales (id_tribunal, id_vocal)
        VALUES (%d, \'%s\')
        ', $id_tribunal, $vocal);
    $this->query($sqlInsert);
}
```

Fig 127. Query que registra los vocales del tribunal

Si el proceso de guardado concluye con éxito, se informará por correo a los miembros del tribunal, al alumno y al director.

6.4.2 Interfaz de Vista general

Esta interfaz tiene la funcionalidad de mostrar a un usuario de grupo 30 (jefes de estudio o responsables de titulación) un histórico con los proyectos que se han llevado a cabo desde la introducción de esta aplicación, o que se encuentran actualmente en curso. El objetivo es ofrecer una visión más ampliada sobre el desarrollo de los TFGs en una titulación y poder obtener métricas como cuántos proyectos hay actualmente en curso, cuántos proyectos tratan determinada temática, cuántos proyectos dirige un docente en concreto, etc.

La vista consiste en dos secciones, siendo la primera una interfaz de búsqueda en la que se aplican los filtros para acotar la información que queremos obtener, y la segunda una tabla de resultados donde se mostrará la respuesta de la base de datos.

En el cuadro de búsqueda aparecen los filtros disponibles. Si no se emplean, es decir, se dejan como vienen por defecto, la aplicación muestra todos los resultados posibles, manteniendo siempre el criterio de permisos según usuario y plan de estudios. Los filtros disponibles son:

- **Plan de estudios:** Combo seleccionable que carga desde la entidad *planes_resp*.
- **Estado:** Combo seleccionable que carga desde la entidad *estados_proyecto*.
- **Tutor:** Campo de texto libre que corresponde al tutor o director del proyecto.
- **Alumno:** Campo de texto libre que corresponde al alumno que realiza el proyecto.
- **Título:** Campo de texto libre que corresponde al título del proyecto.
- **Área:** Combo seleccionable que carga desde la entidad *areas*.

- **Fecha desde:** Campo Datepicker para acotar temporalmente la búsqueda.
- **Fecha hasta:** Campo Datepicker para acotar temporalmente la búsqueda.

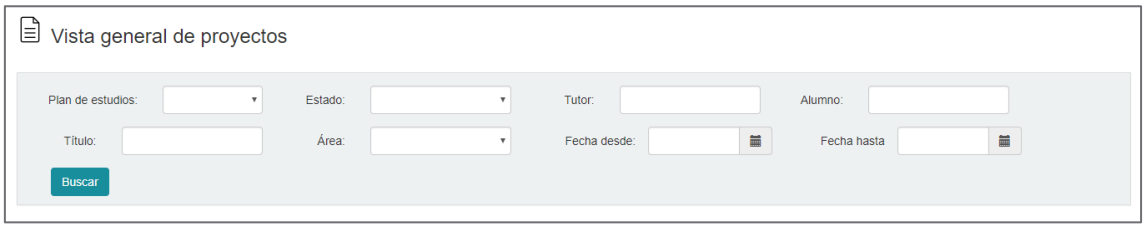

Fig 128. Formulario de búsqueda para la vista general de proyectos

La tabla de resultados sigue la tónica habitual de la aplicación y se compone de filas en las que se muestra la información principal, y un cuadro desplegable donde se amplía. Podemos encontrar los siguientes campos en la cabecera de la tabla:

- Plan de estudios
- Alumno
- Título
- Tutor
- Fecha inicio
- Fecha fin
- Calificación
- Estado

Al desplegar el contenido oculto podremos ver la información sobre los miembros del tribunal y los archivos adjuntos al proyecto.

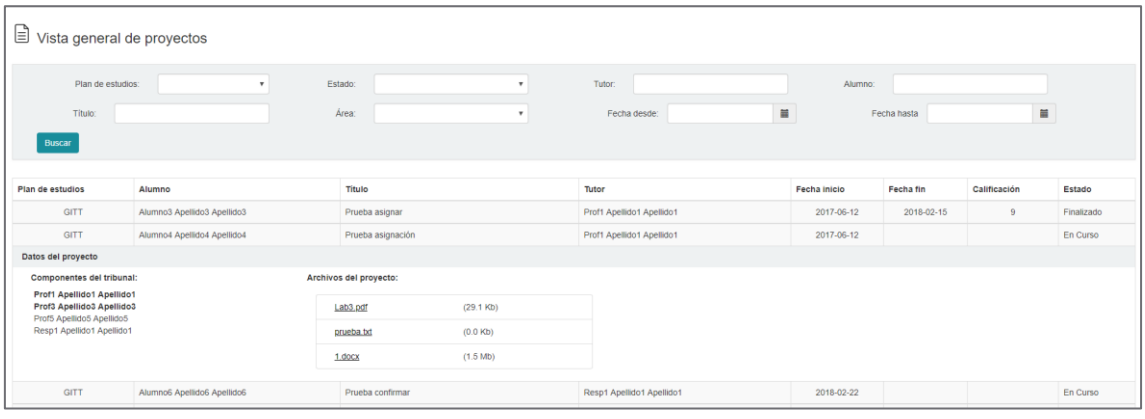

Fig 129. Vista de la interfaz con el desplegable que contiene la información del proyecto

Funcionamiento de la búsqueda

Al realizar una búsqueda, los parámetros del filtro son enviados al servidor y recibidos por el controlador, que realiza un filtrado para evitar inyecciones SQL y procede a enviarlos al modelo, que elabora la consulta SQL.

```
$oListaProyectos = $this->vista->buscarProyectos($plan_estudios, $estado,
        $tutor, $alumno, $titulo, $area, $fechaDesde, $fechaHasta);
```

Fig 130. El controlador recoge los datos y los reenvía al modelo

El modelo elabora la consulta.

```
SELECT
    global.id_proyecto,
    global.id_plan AS plan,
    global.tribunal_asignado AS asignado,
    alumno.nombre AS anombre,
    alumno.apellidos AS aapellidos,
    tutor.nombre AS tnombre,
    tutor.apellidos AS tapellidos,
    global.titulo_proyecto AS titulo,
    global.fecha_oferta_proyecto AS fechaoferta,
    global.fecha_inicio_proyecto AS fechainicio,
    global.fecha_fin_proyecto AS fechafin,
    calificaciones.calificacion AS calificacion,
    ep.desc_estado AS estado,
    miembros.nombre AS mnombre,
    miembros.apellidos AS mapellidos,
    calificaciones.comentario_tribunal AS comentario,
    presi.nombre as pnombre,
    presi.apellidos as papellidos,
    secre.nombre as snombre,
    secre.apellidos as sapellidos

FROM proyectos_global AS global

    INNER JOIN users AS alumno ON global.id_alumno = alumno.userid
    INNER JOIN users AS tutor ON global.id_tutor = tutor.userid
    LEFT JOIN tribunales AS tribunal ON tribunal.id_proyecto = global.id_proyecto
    LEFT JOIN miembros_tribunal AS miembrostribunal ON miembrostribunal.id_tribunal = tribunal.id_tribunal
    INNER JOIN users AS miembros ON miembrostribunal.id_miembro = miembros.userid
    INNER JOIN calificaciones AS calificaciones ON global.id_proyecto = calificaciones.id_proyecto
    LEFT JOIN users AS presi ON tribunal.id_presidente = presi.userid
    LEFT JOIN users AS secre ON tribunal.id_secretario = secre.userid
    INNER JOIN estados_proyecto AS ep ON ep.id_estado = global.estado_proyecto
```

Fig 131. Query de búsqueda de proyectos

Si el usuario ha especificado algún filtro para la búsqueda se añadirá a continuación en la cláusula *WHERE*. Una segunda consulta obtiene los archivos que han sido entregados en cada TFG.

```
// Busco los archivos de cada proyecto
$queryArchivos = sprintf("SELECT id_archivo, id_proyecto, nombre_archivo, tamano_archivo
                    FROM archivos_proyecto
                    WHERE id_proyecto IN ('%s')
", implode("','", $listaIdsProyecto));
```

Fig 132. Query que consulta los archivos subidos de cada proyecto

Posibles desarrollos futuros

Esta interfaz tiene un gran potencial para incorporar nuevas funcionalidades, tales como visualización de estadísticas avanzadas o representación de gráficas. Se explican más detalladamente en el apartado 7.3.

7. Posibles líneas de desarrollo futuro

7.1 Selección de idiomas y multilenguaje

En la cabecera de la página se incluye una botonera que muestra la opción de visualizar la aplicación en inglés o en castellano. Actualmente solo es visible en castellano, porque, aunque la infraestructura esté creada, no se han traducido los textos al inglés.

Se propone, aparte de llevar a cabo las traducciones al inglés, incluir otros idiomas basándose en la afluencia de alumnos extranjeros según nacionalidad o idioma natal. De agregar más idiomas a la aplicación, habría una necesidad de modificar la interfaz del selector de idiomas, probablemente con un menú desplegable que permita mostrar todos los idiomas disponibles, o disminuyendo el tamaño de los botones usados actualmente.

Dado que la aplicación se ha diseñado con suporte multi idioma, añadir nuevos idiomas sería tan trivial como incluir un fichero con las traducciones de todos los textos y el sistema será capaz de mostrarlos correctamente según el idioma seleccionado por el usuario.

7.2 Mejoras en seguridad

7.2.1 Perfilado de accesos a BBDD

Una mejora importante en el ámbito de la ciberseguridad es incrementar la protección de la aplicación mediante la creación de múltiples usuarios para la base de datos con diferentes perfiles de privilegios. Actualmente la base de datos tiene un único usuario con permisos de administrador y es el que se usa desde toda la aplicación. El riesgo potencial de esta configuración es que de producirse un ataque de inyección de código SQL, las sentencias introducidas se ejecutarían con permisos de administrador, lo que implica acceso total a la base de datos.

La mejora propuesta, basada en las metodologías de desarrollo seguro, consiste en la creación de nuevos usuarios de base de datos que podrían alinearse con los grupos definidos dentro de la aplicación. Cada usuario de la base de datos tendría permisos de lectura sobre determinadas tablas, y de escritura solamente sobre las que fuera necesario. Con esta configuración, aunque un usuario malicioso consiguiera ejecutar sentencias SQL, estaría limitado por los permisos de la base de datos.

7.2.2 Control de intentos de acceso al login

Actualmente el login de la aplicación no tiene ninguna seguridad añadida más allá de validar los datos introducidos por el usuario para controlar que no haya inyecciones de código SQL. A pesar de esto, el login sigue siendo vulnerable a ataques como, por ejemplo, los ataques por reiteración de intentos de inicio de sesión.

Estos consisten en intentar acceder probando una gran cantidad de contraseñas para un mismo o varios usuarios. Estas contraseñas pueden salir de un diccionario o ser generadas por un software específico para recorrer todas las combinaciones posibles. Es lo que se conoce como un ataque por fuerza bruta.

Hay muchas formas de protegerse de estos ataques. Una de las más frecuentes últimamente es la inclusión de un *captcha* o código que es necesario introducir para poder iniciar sesión y

que requiere la interacción del usuario. También hay formas más complejas que pasan por controlar el número de intentos incorrectos para bloquear una cuenta, el número de intentos incorrectos para bloquear una dirección IP o incluso controlar el volumen de peticiones originadas en una misma dirección IP.

En nuestro caso se considerará el uso del plugin de Google para generar *captchas* dinámicos, reCaptcha (https://www.google.com/recaptcha)

7.3 Mejora en interfaz de Vista general

La interfaz de visión general de trabajos es una de las partes de la aplicación con mayor potencial de desarrollo. Actualmente solo muestra una pequeña cantidad de datos en comparación con lo que podría ofrecer. La principal idea de desarrollo consiste en usar un plugin de representación gráfica de datos. Se barajan tanto Google Charts (https://developers.google.com/chart/) como la librería D3js (https://d3js.org/), aunque este último ofrece unas opciones muy potentes que probablemente no serían aprovechadas correctamente.

Algunos ejemplos de las gráficas más sencillas que se pueden obtener usando Google Charts:

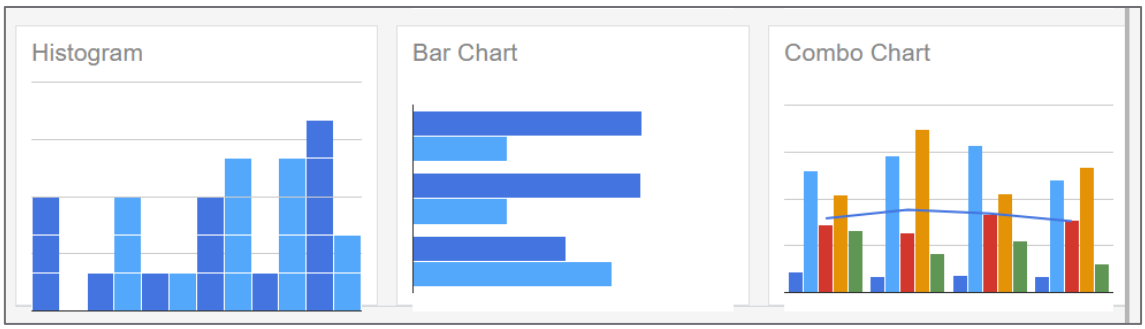

Fig 133. Ejemplos de gráficas aplicables a la interfaz de Vista General

Partiendo de estos ejemplos, se podrían representar de una forma visualmente atractiva al usuario las estadísticas que se consideren apropiadas. Por ejemplo, la duración media de un TFG en función del plan de estudios o del tutor, estadísticas sobre las clasificaciones obtenidas, número de trabajos presentados en cada año, y un largo etc.

La integración de Google Charts es muy sencilla. Las librerías necesarias se encuentran subidas en la CDN (*Content Delivery Network*) de Google, un servicio que permite tener la versión más actual de un software sin tener que actualizarlo manualmente en la aplicación, dado que se enlaza a una URL externa en la que siempre se encuentra actualizado. De este servicio se benefician numerosos plugins o librerías, como, por ejemplo, *jQuery*.

Para cargar las librerías desde el servidor de Google bastaría con incluir las siguientes líneas de Javascript:

```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
  google.charts.load('current', {packages: ['corechart']});
  google.charts.setOnLoadCallback(drawChart);
  ...
</script>
```

Fig 134. Carga de librerías dinámicas desde CDN

7.4 Calendario de eventos futuros

Otra de las mejoras que se quiere incluir es un calendario que indique los eventos que tiene próximamente el usuario. La fuente de datos será la entidad *tribunales*, de donde se extraerán las fechas de cada evento. La representación gráfica sería mediante un calendario maquetado con Html5 y Css3, y en cada día que tuviera un evento señalado se indicaría dicho evento mediante un estilo distinto y un mensaje contextual. También se barajaría la posibilidad de utilizar algún plugin que facilite la construcción del calendario y la representación de las fechas.

7.5 Configurar un servidor de envío de correos

Actualmente el método desde el que se envían los correos está desactivado. En su lugar se guarda un log de cada correo que se debe enviar, con su destinatario, su título y su contenido. Esto se debe a que, durante el desarrollo, se han usado algunas direcciones personales a modo de prueba para comprobar que el envío funciona adecuadamente, pero la mayoría de las direcciones presentes en la base de datos son *dummies*, no existen realmente. Para evitar el "spam" se ha optado por desactivar esta función hasta que se disponga de un servidor de correo dedicado y una base de datos con información real de personas.

7.6 Ampliación en la parametrización de los tribunales

Según lo comentado en el punto 6.4.1, cada plan de estudios tiene su propia normativa en cuanto a la composición de los tribunales de evaluación. Por este motivo, para que la designación automática o guiada sea compatible con todos los planes, es necesario emplear varias tablas para conseguir parametrizar todo.

En el caso del Grado en Ingeniería de Tecnologías de Telecomunicación, la designación de roles se hace a partir del rango dentro de la universidad y la edad de cada miembro. Esto implica tres necesidades que actualmente no están cubiertas:

La primera es la fecha de nacimiento de cada usuario, que iría en un campo de tipo *Date* en la entidad *users* junto con el resto de datos de cada usuario.

La segunda es el rango de cada profesor, que se relacionaría con cada profesor a través de una nueva entidad *rango_profesores*. Esta entidad tendría la siguiente estructura:

Nombre	Tipo	Referencia	Descripción
Userid	String	Users	PK – Identificador único de usuario
Id_rango	Int	Rangos	Identificador del rango

Adicionalmente, sería necesaria una segunda entidad *rangos* que contendría todos los valores posibles, y les asignaría un valor o peso que permitiera compararlos. Tendría la siguiente estructura:

Nombre	Tipo	Referencia	Descripción
Id_rango	Int		PK – Identificador único de rango
Nombre_rango	String		Nombre del rango
Peso	Int		Indica el peso del rango para compararlo con otros

Para poder parametrizar correctamente otras titulaciones de otras escuelas o facultades, se tendría que aplicar una solución particularizada a cada caso, con el consiguiente estudio de requisitos y posible ampliación del modelo de datos.

8. Anexos

La aplicación incluye tres manuales de uso según el perfil del usuario. Se envían junto con el código de la aplicación.

- Manual_usuario_ALUMNOS.pdf
- Manual_usuario_PROFESORES.pdf
- Manual_usuario_RESPONSABLES.pdf
- Software.zip

9. Referencias

1. *Información general de la ETSIIT sobre la realización de trabajos de fin de grado.*
   https://web.unican.es/centros/etsiit/Paginas/TFG.aspx
2. *Tutorial: Introduction to PHP*
   https://www.w3schools.com/php/default.asp
3. *Relational Database Design*
   https://www.ntu.edu.sg/home/ehchua/programming/sql/Relational_Database_Design.html
4. *Patrón de arquitectura Modelo Vista Controlador*
   https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf
5. *Xampp: Apache + SQL + PHP*
   https://www.apachefriends.org/
6. *MySqlWorkbench*
   https://www.mysql.com/products/workbench/
7. *Plataforma Git de GitHub*
   https://github.com/