UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Herramienta avanzada de gestión de Trabajos de Fin de Grado

Rubén Rodríguez Paz
Tutor: Iván Cantador Gutiérrez

Junio 2016

HERRAMIENTA AVANZADA DE GESTIÓN DE TRABAJOS DE FIN DE GRADO

AUTOR: Rubén Rodríguez Paz
TUTOR: Iván Cantador Gutiérrez

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Enero de 2016

Resumen

Tras la implantación del Trabajo de Fin de Grado (TFG) como paso final a la obtención del título en el Plan Bolonia, la Escuela Politécnica Superior (EPS) de la Universidad Autónoma de Madrid ha experimentado un enorme crecimiento en el número de ofertas de trabajos, sus solicitudes de asignación y defensa, y en el número de Comisiones de Evaluación (CCEE) que evalúen esos trabajos. Este hecho ha hecho complicadas la gestión y coordinación de todos los procesos involucrados –desde la propuesta y oferta de trabajos, hasta la publicación de los mismos, pasando por su asignación, modificación, baja y defensa–, originando la necesidad de desarrollar una herramienta software que unifique procedimientos y sus flujos de información de manera integrada.

Este proyecto tiene como objetivo el diseño y desarrollo de tal herramienta. Ésta debe almacenar y permitir la gestión de toda la información referente a los docentes que componen las titulaciones, los datos de TFG propuestos clasificados pos sus tecnologías específicas, los estudiantes matriculados y las CCEE, entre otra. Así mismo, debe mantener y hacer accesible un histórico resultado de la evolución de tal información a lo largo de los diferentes cursos, su tratamiento y posterior almacenamiento como trabajos ya defendidos y comisiones ya realizadas.

Esta herramienta tendrá que unificar toda la información asociada a TFG de la EPS, de manera que tanto estudiantes como profesores puedan ver vía Web los trabajos disponibles y proponer nuevos, y los Coordinadores de TFG puedan gestionar el catálogo de trabajos así como generar de manera asistida las CCEE pertinentes. En este contexto, el procedimiento de generación de comisiones estará automatizado, pero el coordinador podrá realizar ajustes manuales tanto en el número de evaluaciones, como en los TFG y composición de miembros profesores de las mismas.

Palabras clave

Trabajo de Fin de Grado, Comisiones de Evaluación, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Ruby on Rails, Foundation, Javascript, NGINX, Capistrano, GIT

Abstract (English)

After the implantation of the Final Degree Project (FDP) as the final step to obtain a Grade title within the Bologne Process, the Escuela Politécnica Superior (EPS) at Universidad Autónoma de Madrid (UAM) has experienced a huge grow of the number of project proposals, their assignment and defense requests, and the number Evaluation Committees (EECC) to evaluate such works. This fact makes the management and coordination of involved processes –e.g., project proposal and offering, assignment, update, cancelation, and defense– very hard, causing the need to develop a software tool that unifies procedures and information flow in an integrated manner.

This work aims to design and develop such tool. It has to store and allow managing the information regarding the lecturers who belong to the degrees, the data referring to the FDP classified by their specific technologies, the students enrolled in the FDP subject, and the corresponding EECC, among others. Moreover, the tool has to maintain and make accessible history records that emerge from the above information evolution during the different academic years, as well as their processing and further storing as defended projects and created committees.

The tool will have to unify all the information associated to FDP at EPS, so that students and lecturers can access via Web the available projects and propose new ones, and FDP coordinators can manage the project catalog, as well as generate the EECC in a semi-automatic way. In this context, the generation of committees will be done automatically, but the coordinator could further make manual adjustments on both the number of committees, and their corresponding FDP and member composition.

Keywords

Final Degree Project, Evaluation Committees, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Ruby on Rails, Foundation, Javascript, NGINX, Capistrano, GIT

ÍNDICE DE CONTENIDOS

1 Introducción

1.1 Motivación 7

1.2 Objetivos 8

1.3 Organización de la memoria 8

2 Definición y análisis de requisitos

2.1 Definición del proyecto 10

2.2 Funcionalidades 11

2.3 Catálogo de requisitos 12

2.3.1 Requisitos funcionales 13

2.3.2 Requisitos no funcionales 14

3 Diseño

3.1 Arquitectura 16

3.1.1 Arquitectura de la aplicación a nivel de software 16

3.1.2 Arquitectura a nivel de sistemas 17

3.2 Solución aportada 18

3.2.1 Tecnologías y software utilizado 18

3.2.2 Software generado 21

4 Desarrollo

4.1 Hardware 33

4.2 Software 34

4.3 Despliegue 42

5 Integración, pruebas y resultados

5.1 Integración 43

5.2 Pruebas 43

6 Conclusiones y trabajo futuro

6.1 Conclusiones 45

6.2 Trabajo futuro 45

Referencias I

Anexos

A Ruby on Rails. Organización de ficheros III

B Modelo de datos. Diagrama E-R y tablas V

ÍNDICE DE FIGURAS

FIGURA 3-1: DIAGRAMA DE LA ARQUITECTURA DE LA APLICACIÓN A NIVEL DE SOFTWARE 16

FIGURA 3-2: DIAGRAMA DE LA ARQUITECTURA DE LA APLICACIÓN A NIVEL DE SISTEMAS 17

FIGURA 3-3: DIAGRAMA DEL FLUJO DE DESARROLLO USANDO GIT (GIT-FLOW)..... 19

FIGURA 3-4: DIAGRAMA DE LA ARQUITECTURA DE LA APLICACIÓN A NIVEL DE SOFTWARE Y LAS TECNOLOGÍAS UTILIZADAS..... 21

FIGURA 3-5: MAPA DE NAVEGACIÓN DE UN USUARIO SIN LOGUEAR 22

FIGURA 3-6: MAPA DE NAVEGACIÓN DE UN USUARIO LOGUEADO COMO ESTUDIANTE 22

FIGURA 3-7: MAPA DE NAVEGACIÓN DE UN USUARIO COMO PROFESOR..... 23

FIGURA 3-8: MAPA DE NAVEGACIÓN DE UN USUARIO LOGUEADO COMO COORDINADOR 24

FIGURA 3-9: PÁGINA DE INICIO DE LA APLICACIÓN..... 25

FIGURA 3-10: PÁGINA DE LOGIN..... 26

FIGURA 3-11: PÁGINA DEL LISTADO DE TFG DISPONIBLES..... 26

FIGURA 3-12: PROPUESTA DE TFG. PÁGINA INICIAL..... 27

FIGURA 3-13: PROPUESTA DE TFG. SELECCIÓN DE TECNOLOGÍAS 27

FIGURA 3-14: PROPUESTA DE TFG. DATOS DEL PROPONENTE..... 28

FIGURA 3-15: PROPUESTA DE TFG. DATOS DEL TFG 28

FIGURA 3-16: PROPUESTA DE TFG. RESUMEN FINAL..... 29

FIGURA 3-17: ASIGNACIÓN DE TFG. PÁGINA INICIAL..... 30

FIGURA 3-18: ASIGNACIÓN DE TFG. LISTADO DE TFG PARA CAMBIAR ESTADO 30

FIGURA 3-19: GENERACIÓN DE COMISIONES. PÁGINA INICIAL 30

FIGURA 3-20: GENERACIÓN DE COMISIONES. ASIGNACIÓN COMISIÓN-TFG 31

FIGURA 3-21: GENERACIÓN DE COMISIONES. ASIGNACIÓN COMISIÓN-DOCENTES 31

FIGURA 3-22: GENERACIÓN DE COMISIONES. RESUMEN DE COMISIONES GENERADAS 32

FIGURA 4-1: MODELO DE DATOS DEL TFGAPP..... 35

Glosario

API	Application Programming Interface
CCEE	Comisiones de Evaluación
CSS-CSS3	Cascading Style Sheets. Cascading Style Sheets version 3
CVS	Control Versión System
ECTS	European Credit Transfer and Acumulation System
HTML5	HiperText Markup Language. HiperText Markup Language version 5
HTTP	HiperText Transfer Protocol
LEMP	Linux-NGINX-MySQL-PHP
MVC	Modelo-Vista-Controlador
TFG	Trabajo de Fin de Grado
RoR	Ruby on Rails
SASS	Sincactically Awesome Style Sheets
SCM	Software Configuration Management
SCSS	SASS CSS
SQL	Structured Query Language
SSH	Secure Shell
YAML	YAML Ain't Another Markup Language

1 Introducción

Motivación

Un Trabajo de Fin de Grado (TFG) consiste en la elaboración de un proyecto por parte de un estudiante al finalizar sus estudios de grado. Este trabajo supone un paso final para la obtención del título de grado y se realiza bajo la supervisión de un profesor, nombrado como tutor. Como viene especificado en la guía docente de TFG de la Escuela Politécnica Superior (EPS) de la Universidad Autónoma de Madrid (UAM) [1]: "El trabajo de Fin de Grado tiene carácter obligatorio y 12 créditos ECTS. Se realizará al finalizar los estudios de Grado, una vez superadas el resto de asignaturas". Además, un TFG debe ser presentado y defendido ante un tribunal universitario, que en este documento será referenciado como Comisión de Evaluación (CCEE).

En la EPS-UAM el objetivo de un TFG podrá ser el de desarrollar un sistema informático, aplicación software, servicio Web, o proyecto de investigación relacionado los grados impartidos en el centro. A su vez, un TFG puede realizarse en un ámbito profesional, por lo que puede tener un enfoque profesional o de innovación dentro de una empresa del sector de las Tecnologías de la Información y las Telecomunicaciones.

Más específicamente, en la EPS-UAM se realizan TFG para las titulaciones de Grado en Ingeniería Informática (GII), Doble Grado en Ingeniería Informática y Matemáticas (DGII) y Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación (GITST).

Cada grado particular, a su vez, está formado por una serie de tecnologías o ramas. Un TFG puede abordar una o varias tecnologías específicas, dentro de las que se enmarcará el trabajo. Las titulaciones que se imparten en la EPS-UAM se acotan mediante las siguientes tecnologías: Computación, Ingeniería del Software, Sistemas de Información, Ingeniería de Computadores y Tecnologías de la Información para el Grado en Ingeniería Informática, y Sistemas Electrónicos, Radiocomunicaciones, Tratamiento de Señales y Telemática para el Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación..

Tras la elaboración de un TFG, éste será defendido por su autor ante una Comisión de Evaluación, enmarcada dentro de las tecnologías que conforman el trabajo. En la defensa del TFG el autor demostrará sus competencias alcanzadas con el trabajo. La evaluación del TFG de éste estará formada por cinco profesores de la EPS-UAM: un presidente (Catedrático o Profesor Titular de la Universidad), dos vocales y dos suplentes.

Una CCEE tendrá asignadas un máximo de seis TFG que se defiendan en la convocatoria en cuestión y que deberán tener tecnologías o ramas comunes en la medida de lo posible, ya que los miembros docentes de aquella deberán estar especializados en las tecnologías que tengas esos TFG.

Las comisiones se deberán formar tres veces cada curso (uno por convocatoria) y grado. Este hecho junto con el aumento progresivo en el número de propuestas de TFG y estudiantes en disposición de defender un TFG hace muy costosa la gestión de todos los procesos involucrados para coordinadores y profesores, y puede llegar a ser confusa para estudiantes.

Los procesos son gestionados por el coordinador de TFG de cada titulación, que entre otras debe realizar las siguientes tareas:

- Listar todos los TFG ofertados
- Gestionar las propuestas nuevas de TFG
- Realizar las asignaciones de los TFG a los estudiantes atendiendo a sus solicitudes
- Obtener las solicitudes de defensa para cada convocatoria
- Organizar las Comisiones de Evaluación, atendiendo a estas solicitudes.

Todo esto se realiza siguiendo un calendario definido por la escuela y los coordinadores al inicio del curso.

Objetivos

El objetivo de este trabajo es generar un sistema o aplicación informática para centralizar toda la información que atañe a los TFG de la EPS-UAM, permitiendo su gestión y visualización. Debe contener un módulo o funcionalidad para añadir nuevos TFG y nuevas propuestas. A su vez debe facilitar ciertas funcionalidades específicas para los coordinadores puedan generar las CCEE y asignarles docentes.

Todo esto debe organizarse de tal manera que se permita diferenciar entre los distintos perfiles de usuarios que están involucrados en el proceso (coordinadores, docentes y estudiantes), ya que no todos los usuarios acceden a las mismas funcionalidades.

También ha de tenerse en cuenta que debe ser un sistema distribuido para que cada usuario pueda acceder al servicio fácilmente desde sus equipos.

Para cumplir estos objetivos se ha decidido crear una aplicación web que gestione integralmente todo el proceso, teniendo en cuenta la escalabilidad y flexibilidad, ya que este es un problema existente en todas las facultades de la universidad.

Organización de la memoria

El presente documento está compuesto por seis capítulos. Cada uno de ellos se introduce brevemente a continuación.

1. **Capítulo primero: Introducción al documento**. Se describen brevemente los distintos elementos y naturaleza del problema.
2. **Capítulo segundo: Definición y análisis de requisitos**. Se detallan los requerimientos del problema, tanto a nivel funcional, como no funcional. A su vez se detallan requisitos deseables o no fundamentales y posibles mejoras.
3. **Capítulo tercero: Diseño**. Se describen las soluciones aportadas, tanto a nivel de diseño de aplicaciones como a nivel de sistemas, así como el flujo de uso de la aplicación.

4. **Capítulo cuarto: Desarrollo**. Se exponen el modelo de datos, la generación de código, los algoritmos utilizados, los módulos del sistema, y su relación entre todo ello.
5. **Capítulo quinto: Integración, pruebas y resultados**. Se describirá el proceso de creación de pruebas y validación, y se dará un análisis de los resultados obtenidos.
6. **Capítulo sexto: Conclusiones y trabajo futuro**. Se proporcionan las conclusiones extraídas del trabajo realizado, y se resumen posibles acciones futuras.

2 Definición y análisis de requisitos

Definición del proyecto

La aplicación TFGapp se desarrolla con el fin de mejorar el proceso de creación, elección, asignación y gestión de TFG por parte de estudiantes, profesores y coordinadores.

Para ello se ha decidido diseñar una aplicación web mediante la cual estudiantes puedan acceder al listado de TFG disponibles y realizar una solicitud, profesores puedan dar de alta TFG y asignarlos a estudiantes, y coordinadores puedan administrar y generar las comisiones evaluadoras.

Seguidamente se analizará de forma pormenorizada los objetivos específicos de esta aplicación, las condiciones que debe satisfacer, las funcionalidades que se implementarán y los requisitos que debe contener el sistema para cumplir con su cometido.

Para definir las funcionalidades necesarias de la aplicación se ha utilizado como referencia la organización de información utilizada en la EPS y el funcionamiento de los TFG y CCEE en la actualidad.

En la escuela se imparten las siguientes titulaciones:

- Grado en Ingeniería Informática (GII)
- Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación (GITST)
- Doble Grado en Ingeniería Informática y Matemáticas (DGIM)

Estas titulaciones están gestionadas por los siguientes departamentos:

- Departamento de Ingeniería Informática (II)
- Departamento de Tecnología electrónica y de las Comunicaciones (TEC)

Cada grado y cada departamento tiene unas tecnologías asociadas. Categorías con las que catalogar las disciplinas impartidas en la escuela. Son las siguientes:

- En el Grado en Ingeniería informática:
    - Computación (CO)
    - Ingeniería del Software (IS)
    - Sistemas de la Información (SI)
    - Ingeniería de Computadores (IC)
    - Tecnologías de la Información (TI)
- En el Grado en Tecnología Electrónica y de las Comunicaciones:
    - Sistemas Electrónicos (EL)
    - Radiocomunicaciones (RA)
    - Tratamiento de Señales (SE)
    - Telemática (TE)

En lo relativo a las comisiones de evaluación se distinguen las siguientes diferencias entre los profesores, atendiendo a su categoría profesional:

- Catedrático de Universidad (CU)
- Profesor Titular de Universidad (TU)
- Profesor Contratado Doctor (CDR)
- Profesor Ayudante Doctor (AYD)

Funcionalidades

El objetivo fundamental de la aplicación es centralizar toda la información y automatizar algunos de los procesos involucrados en la administración de TFG en la EPS.

Dicho objetivo resume a alto nivel las pretensiones particulares de la aplicación. Sin embargo, para lograr alcanzarlo, se habrán de centrar las funcionalidades y requisitos en satisfacer una serie de objetivos más básicos, pero que actuarán de puente al objetivo final.

- En primer lugar, facilitar a los estudiantes toda la información sobre los TFG. Su solicitud, y posteriormente su asignación a un tutor y evaluación dentro de una comisión evaluadora.
- En segundo lugar, permitir a los profesores la gestión de sus TFG, dándolos de alta y asignándolos a estudiantes.
- En tercer lugar, permitir a los coordinadores de cada grado, gestionar a nivel global todos los procesos, y crear las comisiones evaluadoras de un modo semi-automático.

Para lograr estos objetivos de la manera más adecuada se han definido unas funcionalidades necesarias en la aplicación clasificadas en los distintos subsistemas que se detallarán posteriormente, cuya leyenda es la siguiente:

GU: Gestión de usuarios
GA: Gestión de estudiantes
GP: Gestión de profesores
GT: Gestión de TFG
GC: Gestión de comisiones

Dichas funcionalidades se podrán resumir en:

- La aplicación presentará una funcionalidad básica de:
    - Entrada de datos e información (GT, GU, GA, GP)
    - Análisis y ejecución de actividades que responden a esta información (GC, GT)

- La información de entrada se conseguirá por medio de cualquier usuario registrado y *login*. Cada usuario podrá hacer uso del sistema para sus objetivos, adecuados a su rol. En función del rol los datos podrán obtenerse tanto por su inclusión uno a uno, como su importación en lotes.
- Esta información se almacenará en el sistema, actualizará los módulos correspondientes a su alcance, como estudiantes, profesores o TFG.

Todas estas funcionalidades irán tal y como se ha comentado con anterioridad, clasificados en distintos subsistemas, abarcando los siguientes aspectos.

Gestión de usuarios

En este subsistema se agrupan las actividades relacionadas con el acceso a la aplicación y los permisos de acceso para cada funcionalidad en función de su rol.

Gestión de estudiantes

Este subsistema se ocupa de la gestión de los estudiantes que pueden realizar un TFG, el grado en el que están matriculados y su relación con los TFG en los que está asignado y comisiones.

Gestión de profesores

Este módulo se ocupa de la gestión de los profesores, sus datos personales y de contacto, su relación con los departamentos y sus tecnologías asociadas.

Gestión de TFG

Este subsistema se ocupa de la creación de TFG su asignación tanto a estudiantes como profesores, la modificación de su estado, su listado y exportación para las posteriores gestiones en administración

Gestión de comisiones

Este subsistema se ocupa de la generación de las comisiones, usando como datos de entrada los TFG propuestos, los datos de profesores y tecnologías asociadas.

Catálogo de requisitos

A continuación se describirán los requisitos que debe cumplir la aplicación para su correcto funcionamiento, y que se referenciarán en el documento mostrando su cumplimiento. Los requisitos se separaran en funcionales y no funcionales. A su vez, los requisitos funcionales se dividirá en subsistemas.

2.3.1 Requisitos funcionales

Gestión de usuarios

- **RF-01** La aplicación tendrá como mínimo tres roles de usuario: estudiante, profesor y coordinador.
- **RF-02** Un usuario sin registrar podrá acceder a la aplicación, pero sólo podrá ver un listado de TFG disponibles, el histórico de TFG o el listado de comisiones.
- **RF-03** Un usuario registrado podrá acceder al sistema y se le mostrará en el menú las opciones disponibles en función de su rol y grado.

Gestión de estudiantes

- **RF-04** La aplicación debe contener un listado de estudiantes matriculados en el TFG para cada grado.
- **RF-05** La aplicación debe permitir cargar el listado de estudiantes en lote mediante la importación de un fichero.

Gestión de profesores

- **RF-06** La aplicación debe contener un listado de profesores para cada grado, y sus tecnologías asociadas.
- **RF-07** La aplicación debe permitir cargar el listado de profesores y tecnologías en lote mediante la importación de un fichero.

Gestión de TFG

- **RF-08** La aplicación debe permitir listar todos los TFG que se encuentran en la BBDD, sus tecnologías asociadas y su estado
- **RF-09** Un usuario, profesor o coordinador debidamente registrado puede dar de alta un TFG en el sistema. En función del rol, el TFG tendrá un estado u otro.
- **RF-10** Un estudiante que dé de alta un TFG se le asignará temporalmente a él como defensor del mismo.
- **RF-11** La aplicación debe permitir asignar tecnologías asociadas en el momento en el que se da de alta.
- **RF-12** La aplicación debe permitir el cambio de estado de TFG por parte de profesores o coordinadores

- **RF-13** La aplicación debe controlar el acceso al alta de TFG, o modificación de su estado en función del calendario del TFG de ese año. Estos datos se deberán importar al sistema al comienzo de cada año lectivo por grado.

Gestión de comisiones

- **RF-14** Un coordinador puede modificar el estado de los TFG para poder generar las comisiones.
- **RF-15** El coordinador puede decidir el número de comisiones en la que se repartirán el grueso de estudiantes
- **RF-16** La aplicación debe poder permitir modificar las asignaciones realizadas automáticamente tanto en estudiantes como en el tribunal que los conforman.
- **RF-17** El algoritmo resultante para la generación de comisiones debe optimizar los recursos docentes tratando de evitar que formen parte de más de una comisión.
- **RF-18** La aplicación debe permitir redistribuir las comisiones eliminando alguna de las generadas si el coordinador lo estima oportuno
- **RF-19** El resultado final de la generación de las comisiones debe ser almacenado y se debe posibilitar su exportación.

2.3.2 Requisitos no funcionales

Usabilidad

- **RNF-01)** El sistema mostrará de forma precisa en qué pantalla se encuentra el usuario.
- **RNF-02)** El sistema dispondrá de una barra flotante en todas las pantallas para facilitar la navegación.
- **RNF-03)** El sistema deberá contar con sistemas de accesibilidad para que usuarios con problemas visuales puedan comprender el contenido.

Seguridad

- **RNF-04** Deberá existir un sistema para encriptar las contraseñas de cada rol almacenadas en la base de datos.
- **RNF-05** Al tratarse de un sistema distribuido y multiusuario es necesario gestionar sesiones de manera segura.

- **RNF-06** Los campos de las contraseñas se ocultarán tras asteriscos para protegerla.
- **RNF-07** Deberá contemplarse un sistema de *logueo* para el cual sólo usuarios con un certificado puedan acceder a zonas restringidas de la aplicación

Rendimiento

- **RNF-08** El sistema tendrá la capacidad de soportar múltiples conexiones simultáneas.
- **RNF-09** El sistema deberá poder gestionar el acceso a los recursos en distintas máquinas (p.e., varios frontales + servidor de base de datos)
- **RNF-10** El sistema deberá tener una respuesta rápida para mostrar los recursos a los usuarios en el menor tiempo posible.

Fiabilidad

- **RNF-11** La aplicación deberá contar con un sistema de notificación para informar al usuario de cualquier fallo que se produzca en el sistema.
- **RNF-12** Para no interrumpir la operatividad del sistema se debe plantear un flujo de trabajo para desarrollar mejoras y reparar errores mientras la aplicación se encuentra online.

Interfaz

- **RNF-13** La interfaz será intuitiva y se basará en un sistema de colores según el diferente mensaje que se le quiera dar al usuario.
- **RNF-14** La barra de herramientas flotante permitirá al usuario tener una vista rápida de las acciones posibles de cada usuario, y de si se encuentra *logueado* o no.
- **RNF-15** Existirá un diseño para la visualización en dispositivos móviles o tablets (*responsive design*).

3 Diseño

Arquitectura

Para explicar la arquitectura del sistema, ésta se considera dividida en dos partes. En la primera parte se definirá la arquitectura necesaria para poner en funcionamiento una aplicación web, la gestión del código y cómo se conectará con la máquina que albergue la aplicación de modo que pueda ser accesible a través de un navegador web. Tras esto se definirá la arquitectura a nivel de sistemas: Todos los componentes necesarios para establecer un flujo de trabajo, desarrollar el código, probarlo, y desplegarlo en el sistema destinado a los usuarios finales.

3.1.1 Arquitectura de la aplicación a nivel de software

A continuación se muestra un diagrama con la estructura básica necesaria para definir una aplicación web a nivel de software:

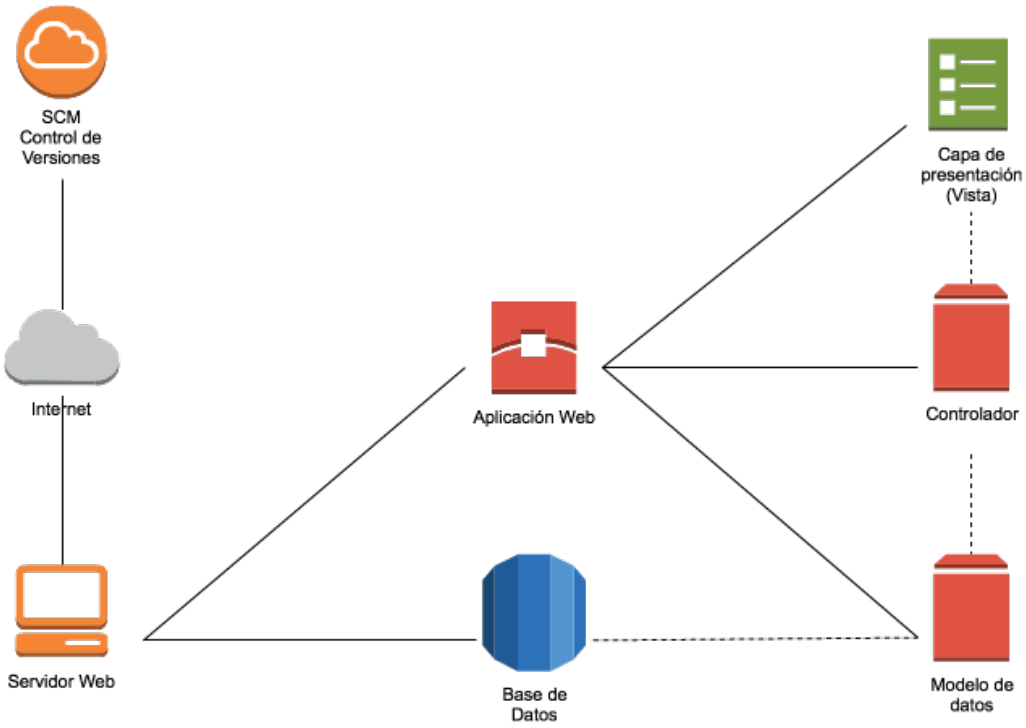


Figura 3-1: Diagrama de la arquitectura de la aplicación a nivel de software

Como se puede ver en la figura 3-1 el sistema está formado por:

- Un servidor web capaz de procesar peticiones de parte de un cliente y ofrecer una respuesta en forma de código HTML.
- Un sistema de control de versiones capaz de almacenar el código del desarrollo, gestionar las versiones.

- Una base de datos conectada con la aplicación que contenga la información que los usuarios añadan mediante su uso..
- Una sistema capaz de gestionar un un patrón de arquitectura MVC.

3.1.2 Arquitectura a nivel de sistemas

A continuación se muestra un diagrama con la estructura básica necesaria para definir una aplicación web a nivel de sistemas:

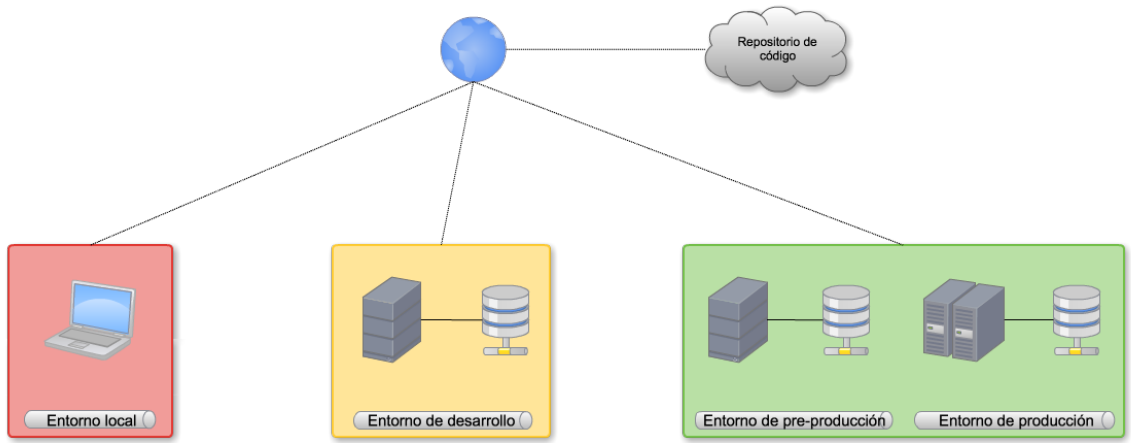


Figura 3-2: Diagrama de la arquitectura de la aplicación a nivel de sistemas

Como se puede ver en la figura 3-2 la arquitectura está formada por:

- Un entorno local donde se desarrollará la aplicación y se realizarán las pruebas más básicas
- Un entorno de desarrollo donde se harán pruebas más exhaustivas. Este entorno deberá parecerse lo más posible al entorno de uso real.
- Un entorno de pre-producción similar al entorno real y con datos reales. Aquí se realizarán las últimas pruebas y se comprobarán las posibles incoherencias en los resultados mediante el uso de datos reales.
- Entorno de producción, que será el que utilizarán los usuarios.

Hay que destacar que para realizar las conexiones y copias de código en cada entorno será necesario utilizar o desarrollar un sistema para copiar los ficheros de manera segura, evitando caídas del sistema explotando al máximo las ventajas del uso de versionado y control de código.

Solución aportada

3.2.1 Tecnologías y software utilizado

Para la solución del problema se ha desarrollado una solución integral utilizando e integrando distintas tecnologías existentes y realizando modificaciones sobre éstas.

Para el desarrollo de la **aplicación web** se ha decidido utilizar Ruby on Rails[1] (RoR) versión 4.2, un *framework* para el desarrollo de aplicaciones web basado en el lenguaje Ruby y que sigue el paradigma de la arquitectura MVC. A este *framework* se le han añadido distintos módulos para agregar funcionalidad extra.

Ruby on Rails permite agregar módulos con funcionalidades extra fácilmente. Esto evita invertir tiempo en el desarrollo de componentes muy básicos que se utilizan en la mayoría de aplicaciones web. En Ruby on Rails (requisitos no funcionales RNF-04, RNF-05, RNF-11) estos módulos se llaman gemas. Las gemas más destacadas utilizadas en este proyecto son las siguientes:

- **Devise**: Gema con módulos para la gestión de sistemas con **autenticación** (RF-03, RNF-07).
- **CanCan**: Gema con módulos para la gestión de sistemas con **roles de usuario** (RF-4).
- **Passenger**: Gema para la ejecución de aplicaciones Rails en servidores NGINX [2](se detalla más adelante)

Para el **almacenamiento y gestión de los datos** se ha decidido utilizar MySQL2[3], un sistema para la gestión de bases de datos escalable, multihilo y multirelacional (RNF-09).

Para el desarrollo del **apartado visual de la aplicación** se ha utilizado un compendio de tecnologías descritas a continuación.

- Como armazón sobre la que se construirá todo el *front-end* se ha decidido utilizar Foundation[4], un *framework* de maquetación basado en HTML5, CSS3 y Javascript con funcionalidades para la generación de diseños *responsive.*(RNF-02, RNF-03, RNF-06, RNF-13, RNF-14, RNF-15)
- Para la generación de estilos CSS se ha utilizado SASS[5], un lenguaje de hojas de estilos, que permite el uso de variables, funciones, etc., y que tras compilarlo genera un código CSS válido para ser visualizado por un navegador.
- Para las modificaciones de estilos asíncronas y las peticiones AJAX se ha utilizado Javascript y jQuery, una biblioteca de Javascript que permite interactuar con documentos HTML.

[1] Web oficial de RoR: http://rubyonrails.org

[2] Web oficial de NGINX http://nginx.org

[3] Web oficial de MySql: https://www.mysql.com

[4] Web de Foundation: http://foundation.zurb.com

[5] Web oficial de SASS: http://sass-lang.com

Para la **gestión del código** generado, mejoras, y despliegue se ha utilizado GIT, como software de control de versiones. Todo el código del proyecto, su evolución, ramas y tareas se han guardado en un repositorio privado almacenado en BitBucket [6](RNF-12).

Para el **despliegue y gestión de entornos** de producción, *staging*, y desarrollo se ha utilizado Capistrano, un gestor de tareas especialmente desarrollado para despliegues de aplicaciones web con código gestionado por un sistema de control de versiones (RNF-09).

Para la **ejecución** de la aplicación se ha decidido utilizar un entorno LEMP básico, ya que el rendimiento de NGINX (RNF-08) es mayor respecto a Apache en cuanto a la gestión de muchos usuarios concurrentes, como se da el caso en esta aplicación.

Un entorno LEMP es un máquina que actúa como servidor web y que tiene instalado Linux (L) como sistema operativo, NGINX (E) como servidor HTTP, MySql (M) como servidor de bases de datos y Php (P) como intérprete de código.

Debido a que NGINX o Apache sólo sirven páginas HTML (o PHP si se activa) es necesario un servidor haga de puente y convierta el código generado por RoR en código que pueda servir NGINX. Para esto se unas la gema Passenger.

Para el desarrollo de esta aplicación y tras su posterior puesta en producción se ha decidido contar con un **flujo de trabajo** basado en tres grandes ramas de desarrollo, que a su vez corresponden con ramas en Git y con los distintos entornos de producción definidos.

En la figura 3-3 se esquematiza el flujo de trabajo utilizando Git como sistema de versionado de código.

Figura 3-3: Diagrama del flujo de desarrollo usando Git (Git-flow)

[6] Web oficial de Bitbucket: https://bitbucket.org

En la figura se puede ver como para realizar una mejora (Feature) se saca una rama (branch, una copia del código) de la rama *Master* que corresponde a la rama del entorno de producción, y se realiza la mejora definida.

Tras realizar la mejora se integra con la rama de desarrollo (Develop) para realizar las primeras pruebas y test.

Tras validar los test, esta rama se integra con la rama de integración (Staging) para verificar que esta mejora no genera inconsistencias con el sistema real de producción. Tras validar que todo está correcto se integra la mejora con la rama de producción (Master).

Dev: Tras el desarrollo de una mejora en local se sube al servidor de desarrollo utilizando Capistrano y se comprueba si los resultados esperados son correctos. En el desarrollo del trabajo se ha utilizado un VPS hospedado en OVH, ya que no se contaba con los entornos finales de la EPS de pre-producción y producción.

Staging: Tras la validación en el entorno de dev se añade esta funcionalidad al entorno de *Staging* o pre-producción, un entorno igual al de producción con una copia de la información de la base de datos de producción.

Producción: Entorno final usado por los estudiantes y profesores, en los que se interactúa con datos reales.

El resultado de las tecnologías utilizadas se pueden ver en la figura 3-4 de la arquitectura de la aplicación:

- El servidor LEMP obtiene la versión del código fuente del repositorio Git alojado en Bitbucket usando Capistrano como gestor de tareas para servidores.
- Este código se ejecuta en un servidor RoR, que se conecta al servidor web NGINX mediante Passenger
- Los datos de la aplicación se almacenan usando MySQL y se conectan mediante un socket a la aplicación RoR
- Para mostrar la información correctamente se integra en la vista Foundation como framework y se utiliza SCSS como lenguaje para generar el código CSS

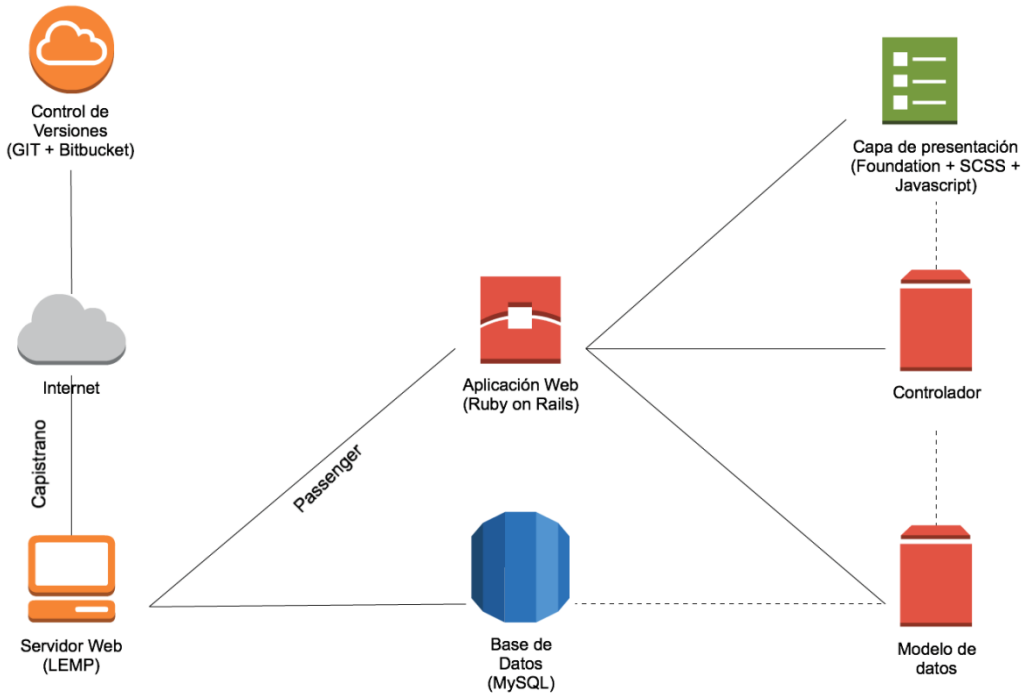


Figura 3-4: Diagrama de la arquitectura de la aplicación a nivel de software y las tecnologías utilizadas

3.2.2 Software generado

El software generado en este TFG consta de una aplicación web basada en RoR con un modo de funcionamiento similar a un CMS, pero con un objetivo específico y definido. El conjunto de funcionalidades que el usuario dispone varía en función de su rol, definido anteriormente en los requisitos funcionales. El resultado de la relación funcionalidades-rol se puede visualizar en los mapas de navegación que siguen a continuación. Como se podrá ver, son incrementales. Es decir, los usuarios tendrán un nivel mayor de funcionalidades que el usuario de nivel "inferior", no distintas.

Los tipos de usuarios según el número de funcionalidades a las que pueden acceder (de menor a mayor) son los siguientes:

- Usuario sin *loguear*
- Usuario *logueado* con rol estudiante
- Usuario *logueado* con rol profesor
- Usuario *logueado* con rol coordinador

A continuación se detallan los mapas de navegación en función de su rol.

Usuario sin loguear

En la figura 3-5 se puede ver el mapa de navegación en la aplicación de un usuario sin loguear. Este tiene como posibles acciones tras llegar a la *Home* loguearse, o visualizar los distintos listados disponibles para todos los usuarios (TFG disponibles, asignaciones, comisiones y TFG anteriores). Si el usuario se encuentra en el listado de TFG disponibles, comisiones o TFG anteriores, tiene la posibilidad de acceder al detalle de cada elemento del listado. En esta página detalle se muestra información extra relevante que por criterios de usabilidad no era necesaria incluir en el listado.

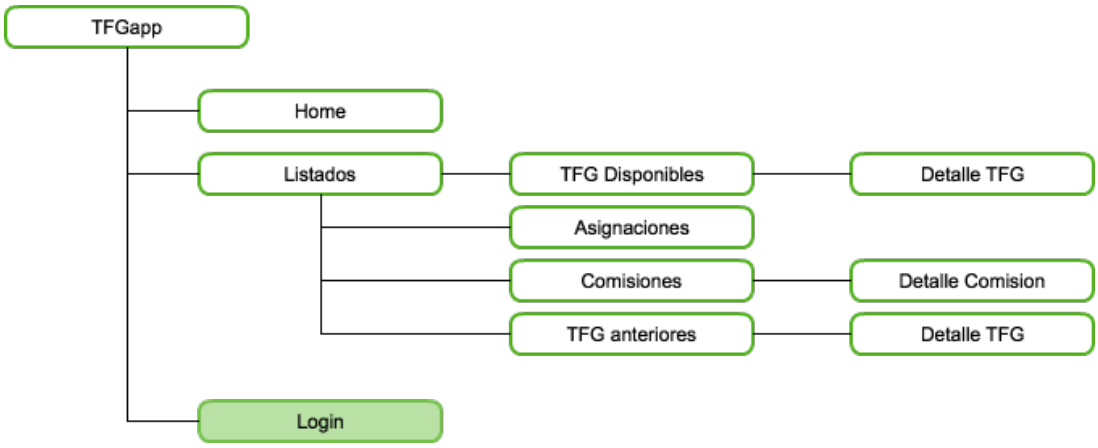


Figura 3-5: Mapa de navegación de un usuario sin loguear

Usuario logueado como estudiante

En la figura 3-6 se detalla el mapa de navegación de un usuario logueado como estudiante. A las posibles acciones de un usuario sin loguear se le añade la posibilidad de proponer un TFG. Este proceso se divide en 5 pasos: Eleccion de Grado, elección de tecnologías, datos del proponente, datos del TFG y resumen de la solicitud.

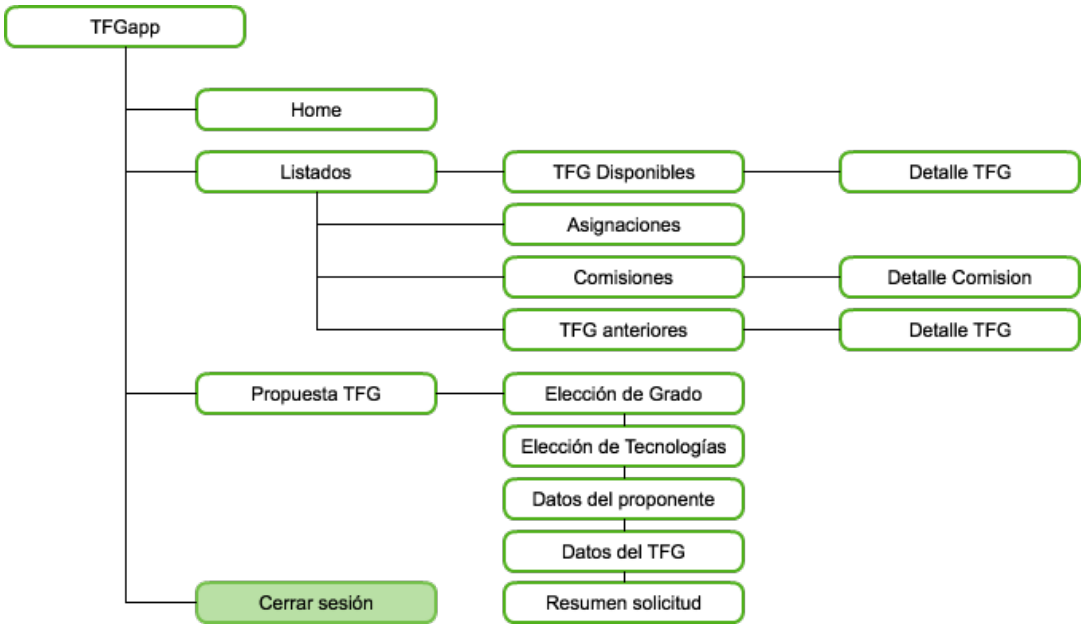


Figura 3-6: Mapa de navegación de un usuario logueado como estudiante

Usuario logueado como profesor

En la siguiente figura se muestra el mapa de navegación de un usuario logueado como profesor. A nivel de navegación es similar al rol de usuario. Las diferencias se encuentran en la información que se le solicita al usuario en el proceso de propuesta de TFG. Para todos los perfiles el proceso se divide en 5 pasos.

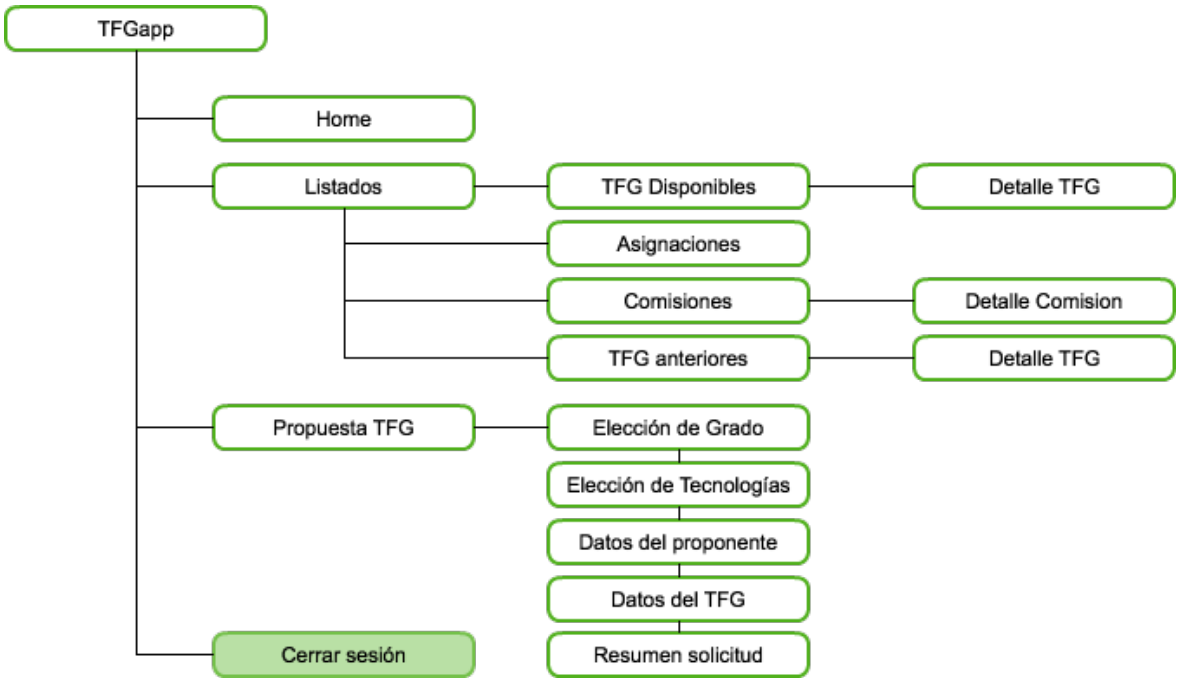


Figura 3-7: Mapa de navegación de un usuario como profesor

Usuario logueado como coordinador

En la figura 3-8 se muestra el mapa de navegación de un usuario logueado como coordinador. A las acciones posibles para un profesor y un alumno se le añaden las relativas a las comisiones. Una vez logueado, el coordinador puede acceder al formulario para modificar las asignaciones de los TFG o generar las Comisiones de Evaluación. Este proceso se realiza en 4 pasos: Selección de parámetros, modificación TFG-comisiones, asignación de profesores y resumen final.
A su vez el coordinador puede acceder a la sección de administración de la aplicación, donde puede dar de alta nuevos usuarios, fijar las fechas para realizar cada plazo, o realizar una importación de TFG "en masa". También tiene acceso a una página con estadísticas relativas a los TFG, tecnologías que más se aplican, profesores con más TFG asignados, etc...

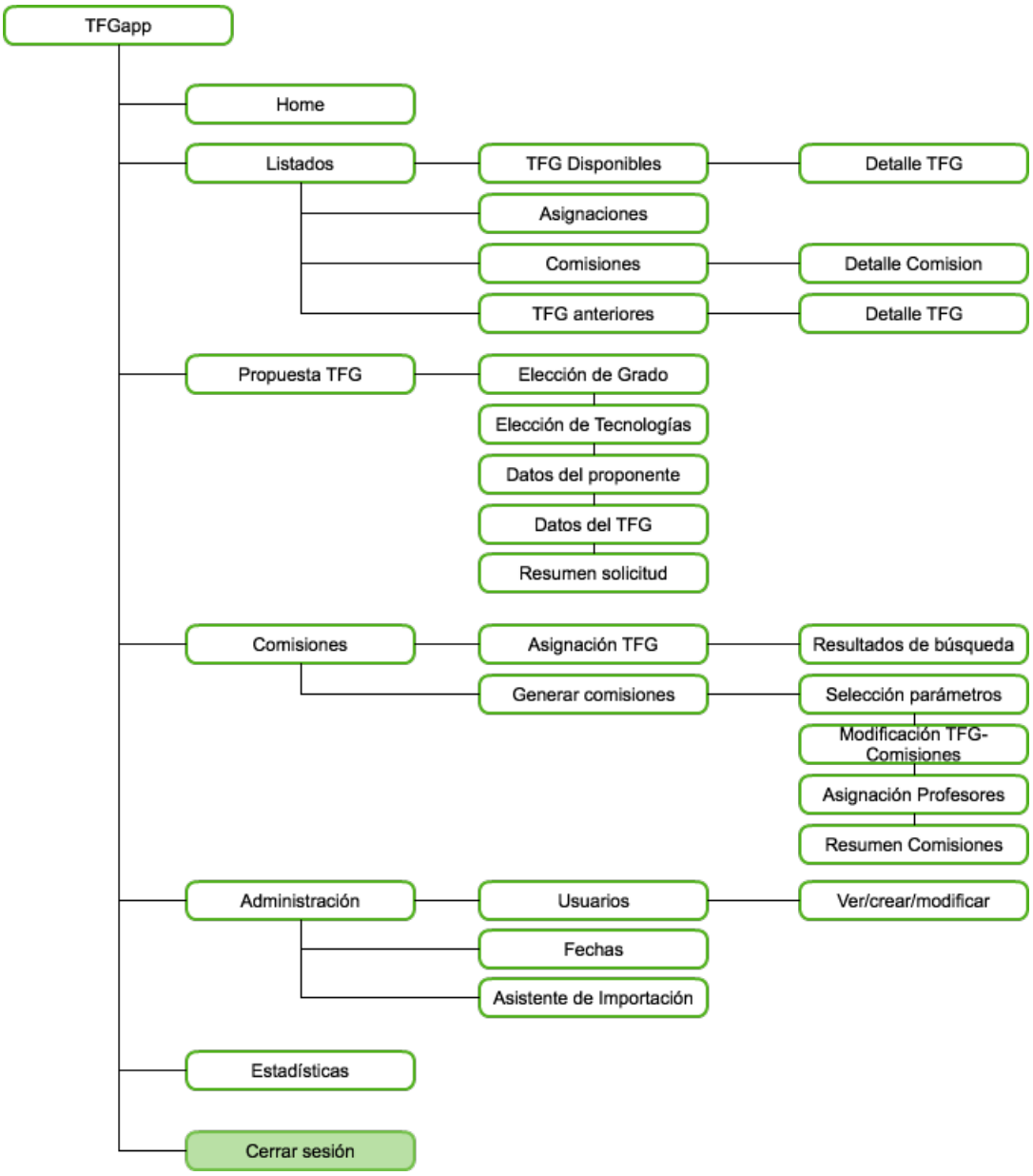


Figura 3-8: Mapa de navegación de un usuario logueado como coordinador

Navegación por la aplicación

A continuación se describirá la funcionalidad general de la aplicación, utilizando pantallazos para explicar las funciones de cada página y cada proceso.

Inicio

En la imagen inferior se puede ver la página principal de la aplicación. Consta de dos partes claramente diferenciadas:

- **Menú de navegación**: Se mantiene en todo el sitio. Contiene todas las posibles acciones del usuario dentro de su rol (RF-03)
- **Zona de contenido**: Se mostrará la información referida a la funcionalidad o módulo a la que se ha accedido. En la página principal puede mostrar una introducción de la herramienta e información relevante para los usuarios. Actualmente muestra un listado de los cambios realizados entre cada versión y su despliegue en el servidor de desarrollo.

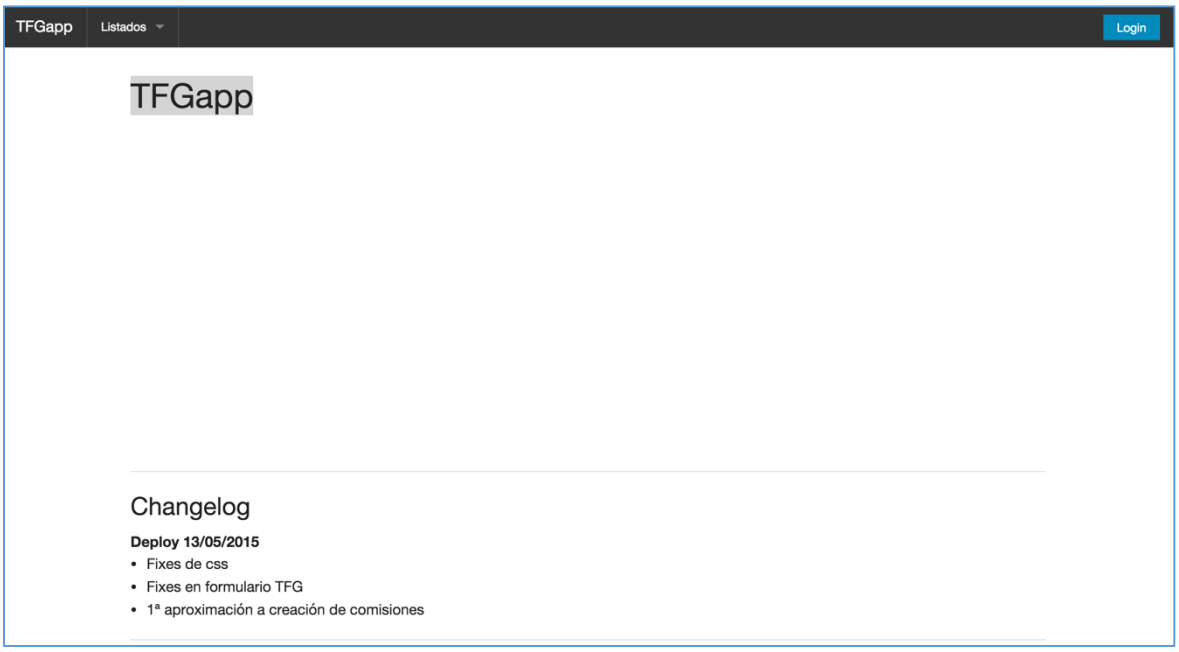


Figura 3-9: Página de inicio de la aplicación

Login

En la figura 3-10 se muestra la vista de la sección de acceso. Tras hacer click en el enlace de login se dirige a la sección de acceso, la cual tras introducir las credenciales se almacena el perfil en la aplicación, para gestionar las restricciones de acceso (RF-01, RF-02).

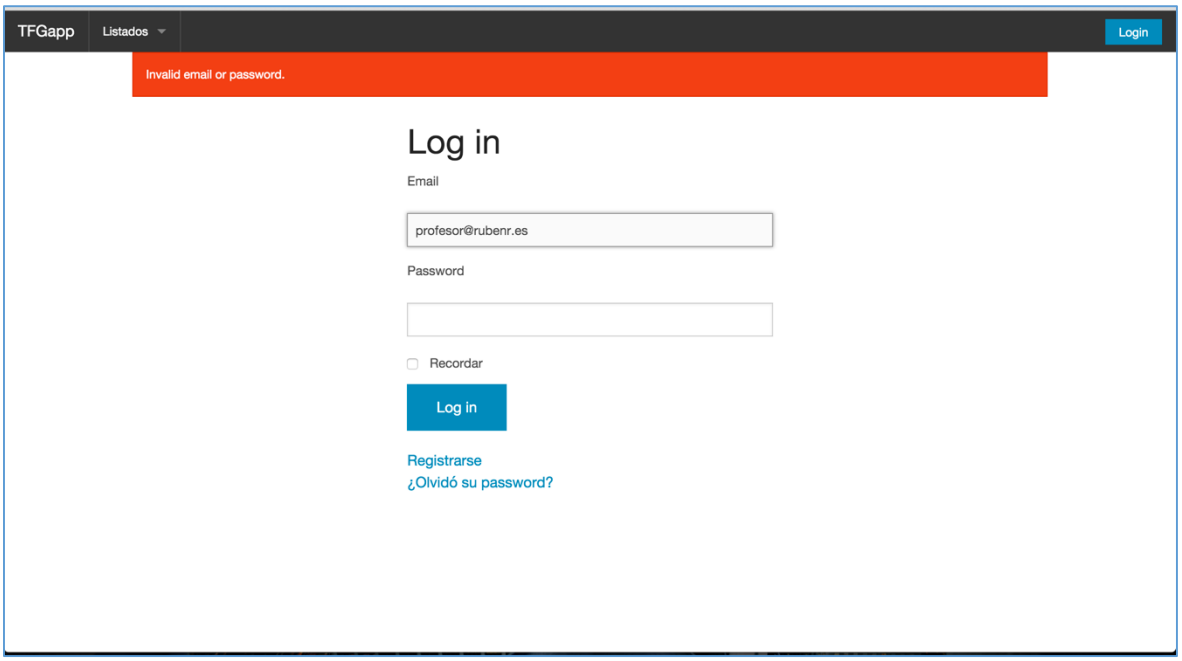


Figura 3-10: Página de login

Listados

En la figura inferior se muestra la vista de detalle de los listados. En esta sección los usuarios pueden ver los TFG disponibles, comisiones y TFG de años anteriores, con un botón para acceder al detalle de cada elemento. Esta página de detalle contiene la información relevante sobre esta (nombre, código, tutor, descripción…). El coordinador a su vez podrá editar y eliminar solicitudes (RF-8).

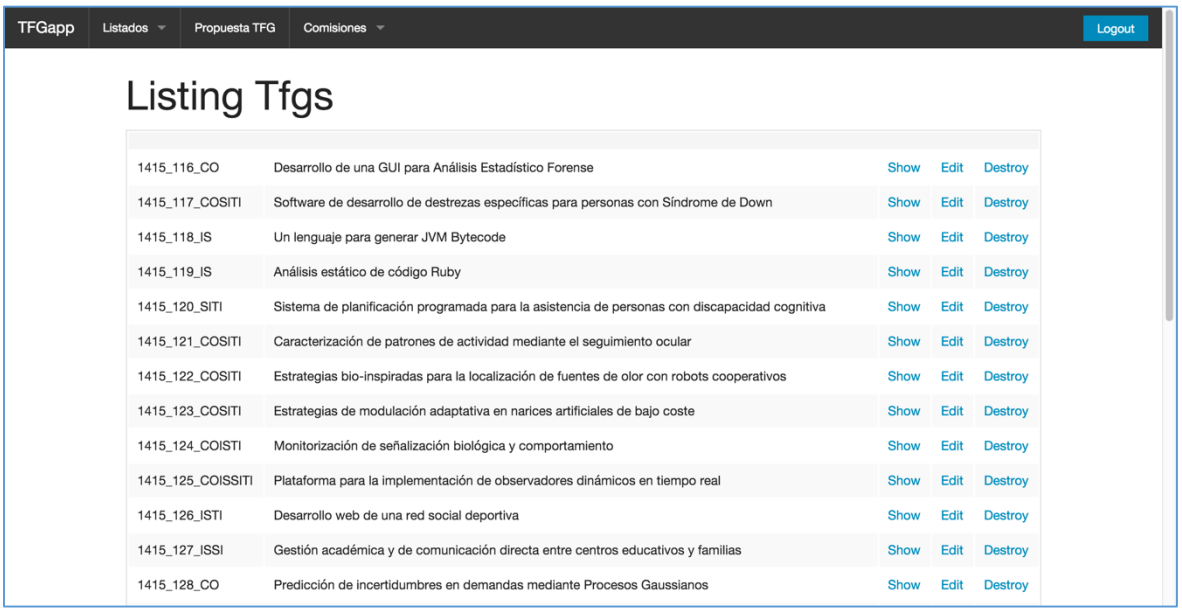


Figura 3-11: Página del listado de TFG disponibles

Propuesta de TFG

La generación de propuestas de TFG se ha divido en cuatro pasos, para facilitar la comprensividad por parte del usuario y para no tener que gestionar muchas llamadas a métodos y helpers usando peticiones AJAX (RF-9).

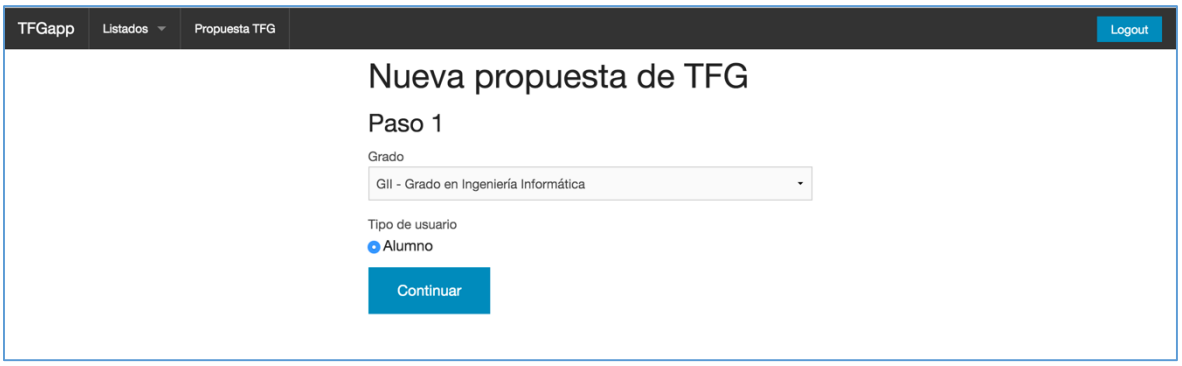


Figura 3-12: Propuesta de TFG. Página inicial

Paso 1: Selección del grado en el que se desea añadir la solicitud y el tipo de usuario (sólo para coordinadores)

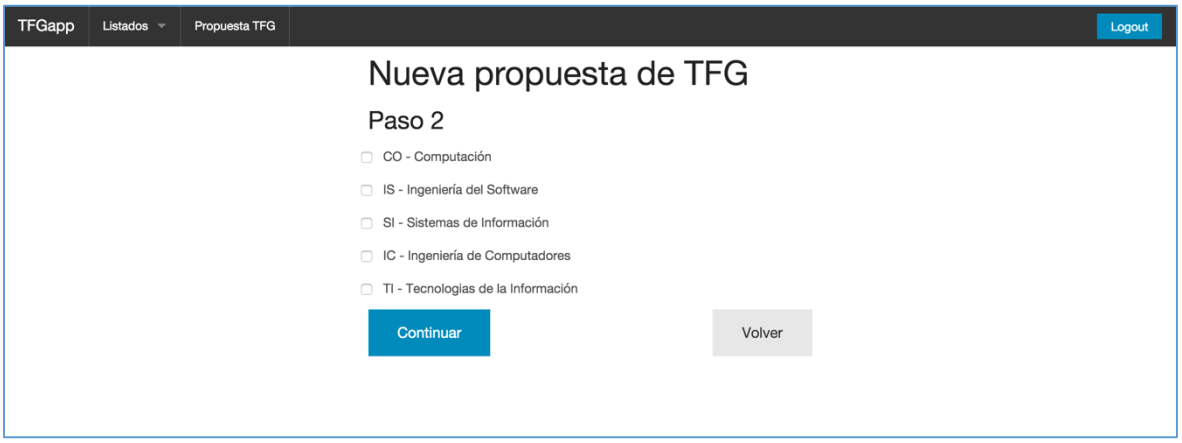


Figura 3-13: Propuesta de TFG. Selección de tecnologías

Paso 2: Selección de las tecnologías posibles para el grado seleccionado. Se puede dar el caso que el grado no tenga tecnologías asociadas. En este caso aparecerá un checkbox preseleccionado indicando esta situación.

Figura 3-14: Propuesta de TFG. Datos del proponente

Paso 3: Inserción de los datos del proponente, nombre y correo.

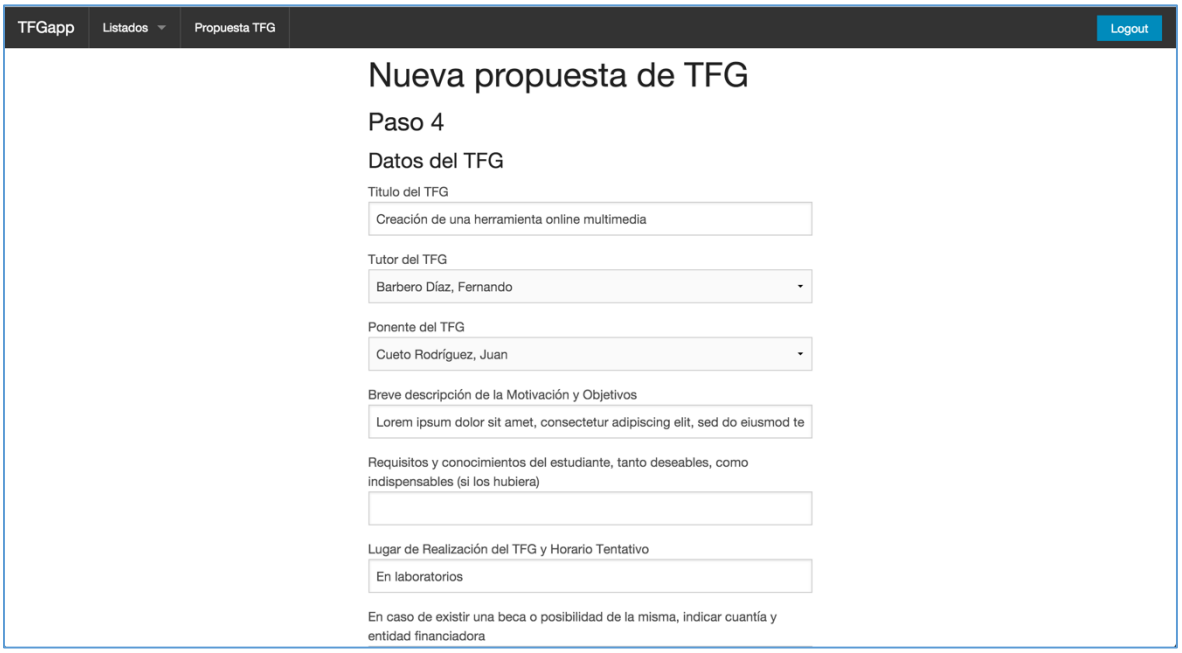


Figura 3-15: Propuesta de TFG. Datos del TFG

Paso 4. Inserción de los datos del TFG, título, tutor, ponente (si aplica), descripción, requisitos, etc…El formulario finaliza con un checkbox obligatorio para aceptar un consentimiento.

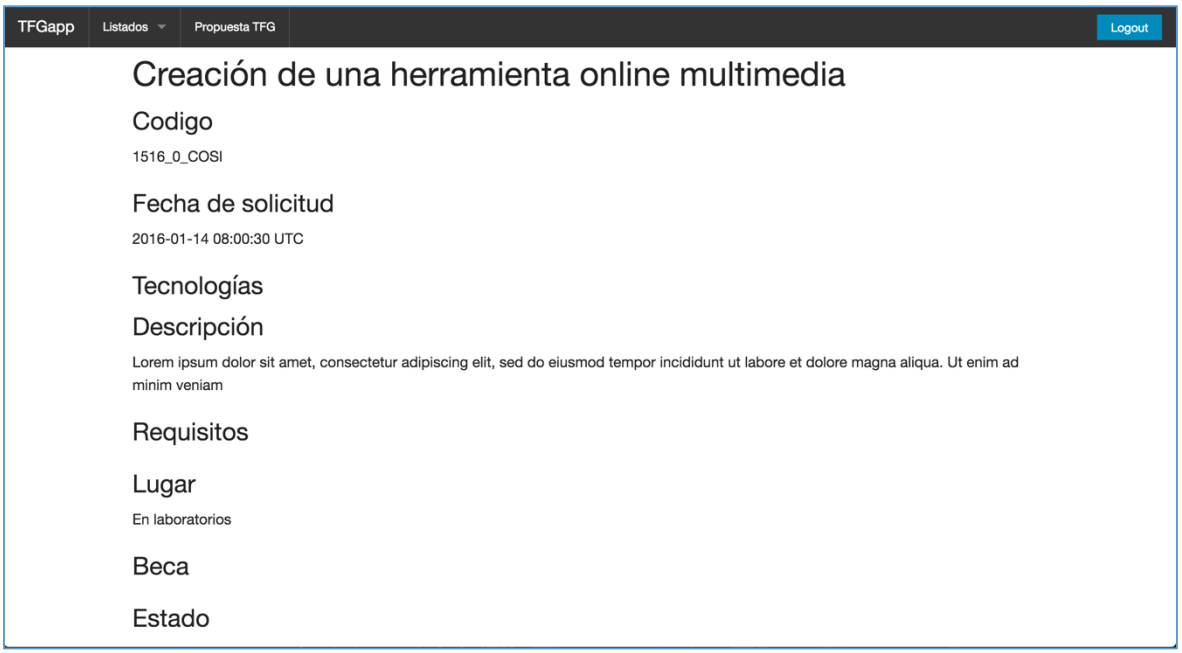


Figura 3-16: Propuesta de TFG. Resumen final

Resumen final. Vista detalle de la solicitud creada

Asignación de TFG

Este apartado será usado por los coordinadores para cambiar los estados de los TFG. Un TFG puede pasar pos los siguientes estados (RF-11):

- Ofertado: Estado inicial cuando un profesor o coordinador añade un TFG
- Propuesto: Estado inicial cuando un estudiante propone un TFG
- Asignado: Estado seleccionado cuando el coordinador recibe el listado de asignaciones de TFG de secretaría.
- A comisión: Estado seleccionado por el coordinador cuando recibe el listado de solicitudes de defensa para la siguiente convocatoria de defensas.
- En comisión: Estado modificado tras la creación de comisiones en la cual dicho TFG forma parte.
- Defendido: Estado final del ciclo. El TFG pasa a formar parte del listado de TFG de años anteriores salvo que se suspenda y vuelva a pasar al estado Asignado, o se abandone y vuelva al estado Ofertado.

Figura 3-17: Asignación de TFG. Página inicial

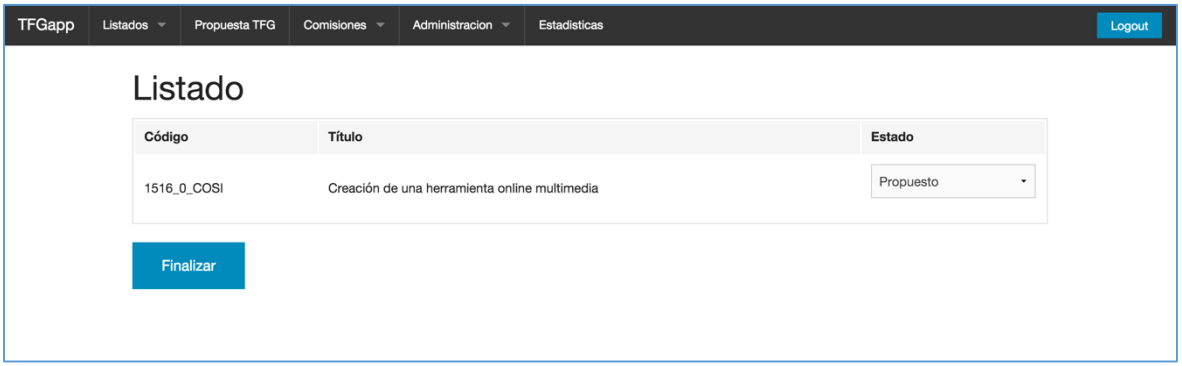


Figura 3-18: Asignación de TFG. Listado de TFG para cambiar estado

Generación de comisiones

La generación de Comisiones de evaluación se ha divido en 3 pasos como muestran 3-19, 3-20 y 3-21 las fig. El el proceso más complejo de la aplicación, ya que la creación de comisiones requiere recorrer el listado de TFG que se pretenden asignar a las comisiones múltiples veces.

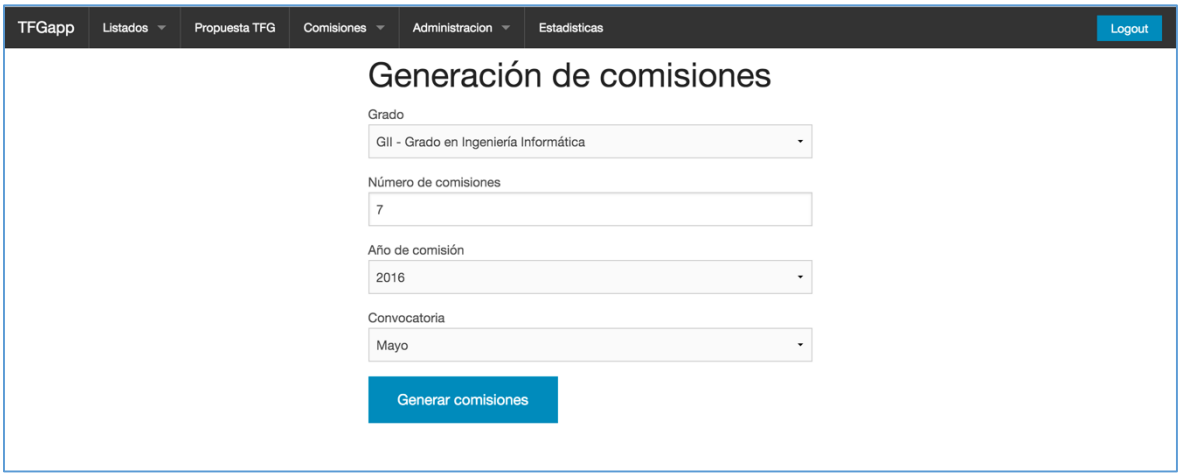


Figura 3-19: Generación de comisiones. Página inicial

Paso 1. Parte inicial del proceso de generación de comisiones. Se fija el grado, si el coordinador coordina más de un grado, el número de comisiones, el año y la convocatoria (RF-15)

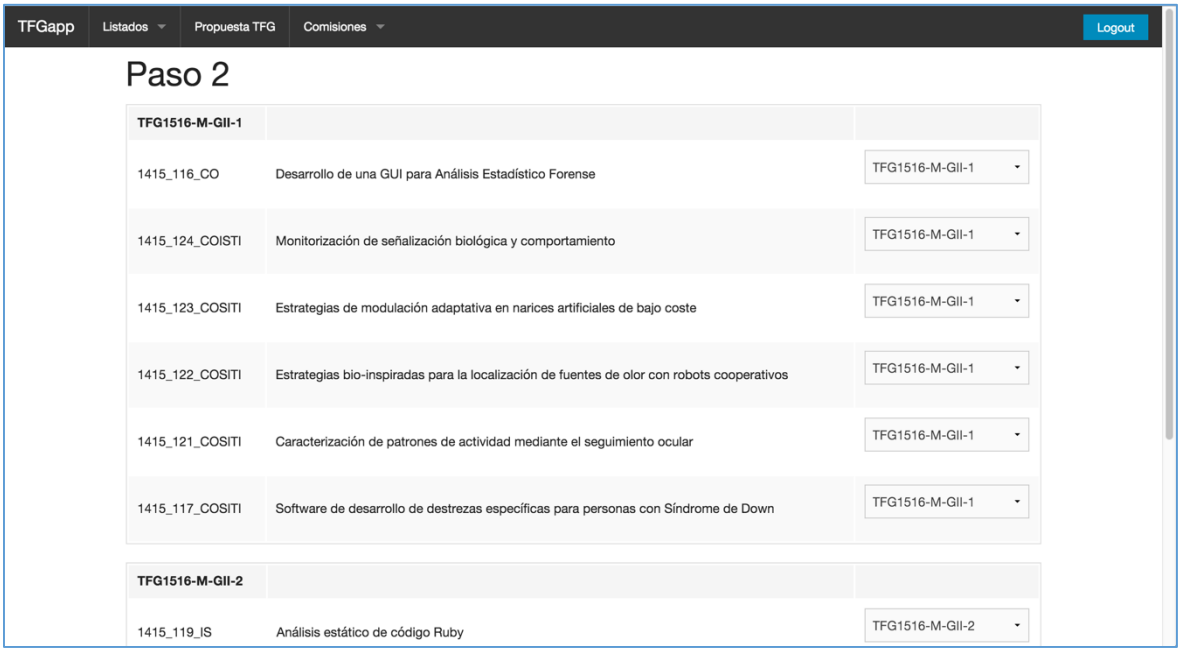


Figura 3-20: Generación de comisiones. Asignación Comisión-TFG

Paso 2. En este paso se presentan las comisiones organizadas por comisiones utilizando el algoritmo explicado posteriormente. También permite modificar la comisión a la que pertenece cada TFG mediante un selector. Cuando se modifica la comisión se cambia automáticamente la posición, y se coloca en la tabla elegida (RF-16, RF-17).

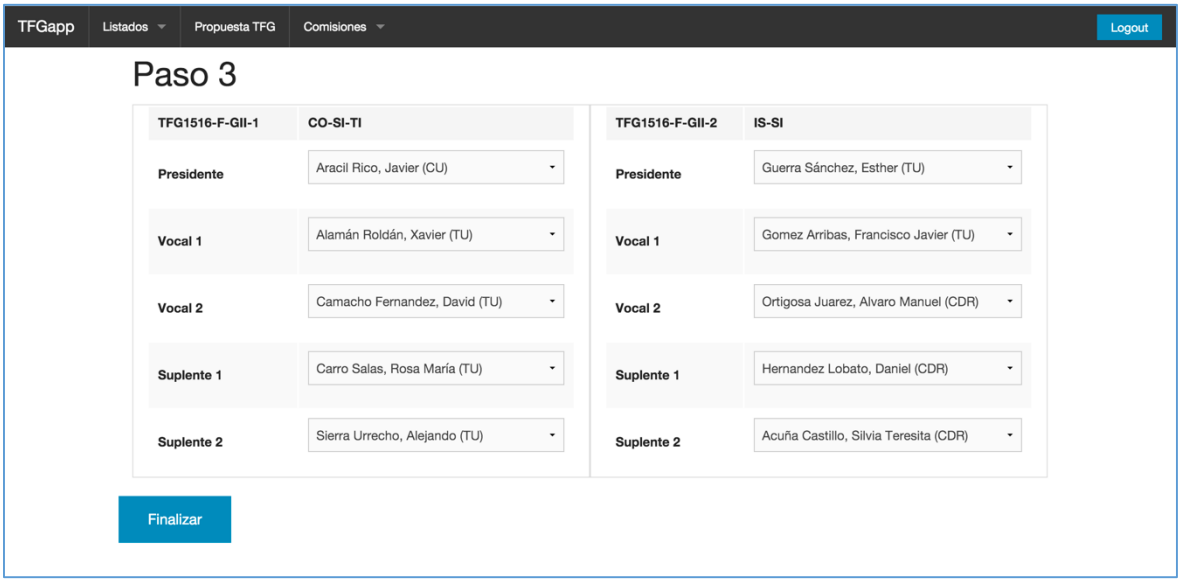


Figura 3-21: Generación de comisiones. Asignación Comisión-Docentes

Paso 3. En este apartado se eligen los docentes que conforman cada comisión. Estas asignaciones se generan automáticamente, pero se puede modificar el docente en cada puesto mediante el uso de un selector (RF-18).

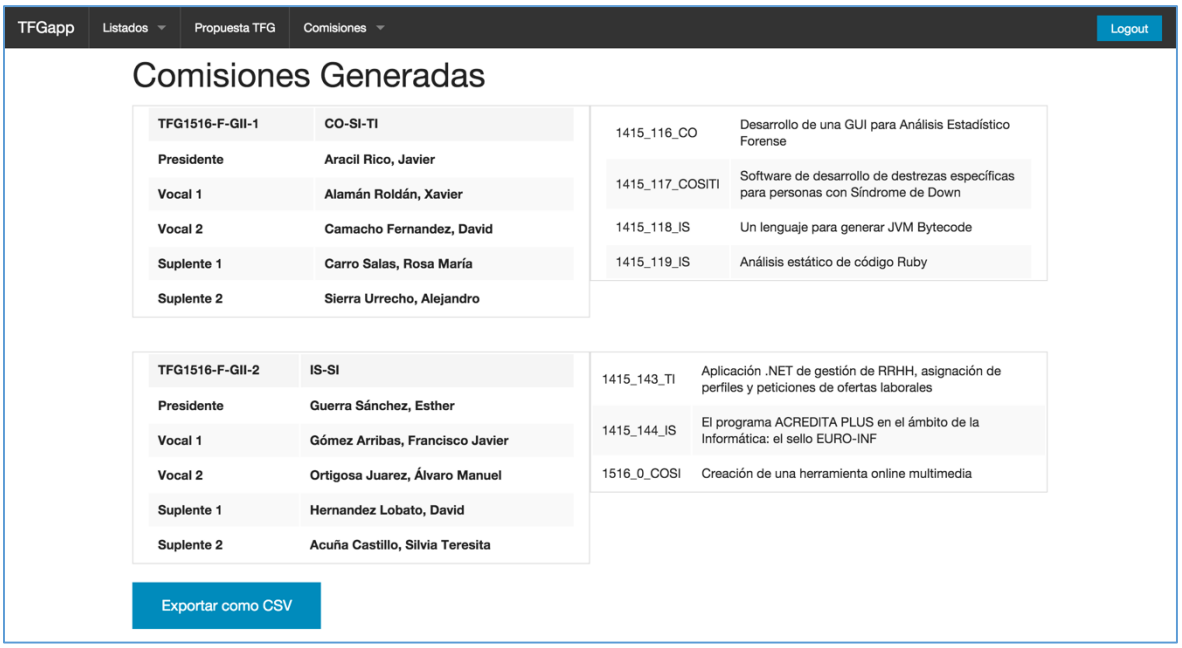


Figura 3-22: Generación de comisiones. Resumen de comisiones generadas

Finalmente se listan las comisiones generadas y los TFG que lo conforman. También existe la posibilidad de los datos exportando los valores de las tablas a un fichero .csv, utilizando una de las funcionalidades nativas de RoR (RF-19).

4 Desarrollo

Hardware

A continuación se describirán las acciones emprendidas a nivel de hardware para poder ejecutar la aplicación. Todas estas configuraciones se han realizado en un servidor VPS. Este tipo de servidores emulan el comportamiento de un servidor físico mediante software., dividiendo una máquina en múltiples servidores virtuales, abaratando el coste de contratación y posibilitando una rápida escalabilidad.

Configuración del VPS

Para probar el software con una configuración de hardware más realista mientras se realizaba el desarrollo de la aplicación se ha optado por usar un servidor VPS alojado en OVH. Una vez creada la instancia del VPS por parte de la empresa de hosting se han realizado las siguientes acciones:

- Instalar Ubuntu server 14.04. Se ha decidido utilizar este SO y no CentOS por la sencillez para hacer modificaciones en el SO y la experiencia de haberlo utilizado a lo largo del grado.
- Instalar MySQL y crear una base de datos nueva para alojar la información de TFGapp
- Instalar rbenv, un sistema para gestionar múltiples instancias y versiones de Ruby en un mismo SO, y posteriormente poder instalar el compilador de Ruby.
- Instalar la gema de RoR y Passenger
- Instalar NGINX.

Tras esto hemos pasado a la configuración del servidor web

Configuración de NGINX

Tras la instalación de NGINX hemos creado un un fichero de configuración para que el servidor apunte a la futura posición de la aplicación y pueda cargar el socket de Passenger. Como se puede ver también existe un flag que activa la ejecución de Passenger en el servidor de desarrollo.

```
server {
  listen 80;
  server_name localhost;
  passenger_enabled on;


  root /home/deploy/tfgapp/public;

  try_files $uri/index.html $uri @app;

  location @app {
    proxy_pass http://app;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_redirect off;
  }

  error_page 500 502 503 504 /500.html;
  client_max_body_size 4G;
  keepalive_timeout 10;
}
```

Software

Para generar el proyecto en Ruby on Rail sólo es necesario usar el comando `rails new TFGapp`. Este comando crea toda la estructura de ficheros y las configuraciones básicas para poder ejecutar la una aplicación RoR en un entorno local. La estructura de los ficheros se describirá en detalle en uno de los anexos.

Modelo de datos

En RoR el diseño del modelo de datos está íntimamente relacionado con el número de módulos y sus enlaces entre sí. Cuando se genera un módulo puedes decidir que cuente con un modelo de datos y cuáles son sus campos. Para ello usa un sistema llamado Active Record.

En el modelo del módulo se pueden añadir relaciones y restricciones para que estas sean pasadas posteriormente al modelo de MySQL, aunque lo correcto es gestionar las restricciones y las relaciones usando los métodos de Active Record, para que la gestión de la base de datos sea transparente al desarrollador. De este modo, si es necesario cambiar de sistema de gestión de bases de datos, sólo es necesario añadir la gema del nuevo sistema y modificar la ubicación del socket de la base de datos. Una vez terminado de definir los campos y relaciones de un módulo sólo es necesario migrar los cambios a la base de datos mediante el comando `rake db migrate` y reiniciar el servidor Rails para que se actualicen los cambios en la base de datos.

A su vez, se ha utilizado un script para poblar la base de datos para poblar de contenido esta y poder realizar pruebas.

La estructura de módulos y base de datos utilizada en el proyecto es la diagrama UML. El diagrama E-R completo y la relación y descripción de las tablas se puede ver en detalle en el Anexo C:

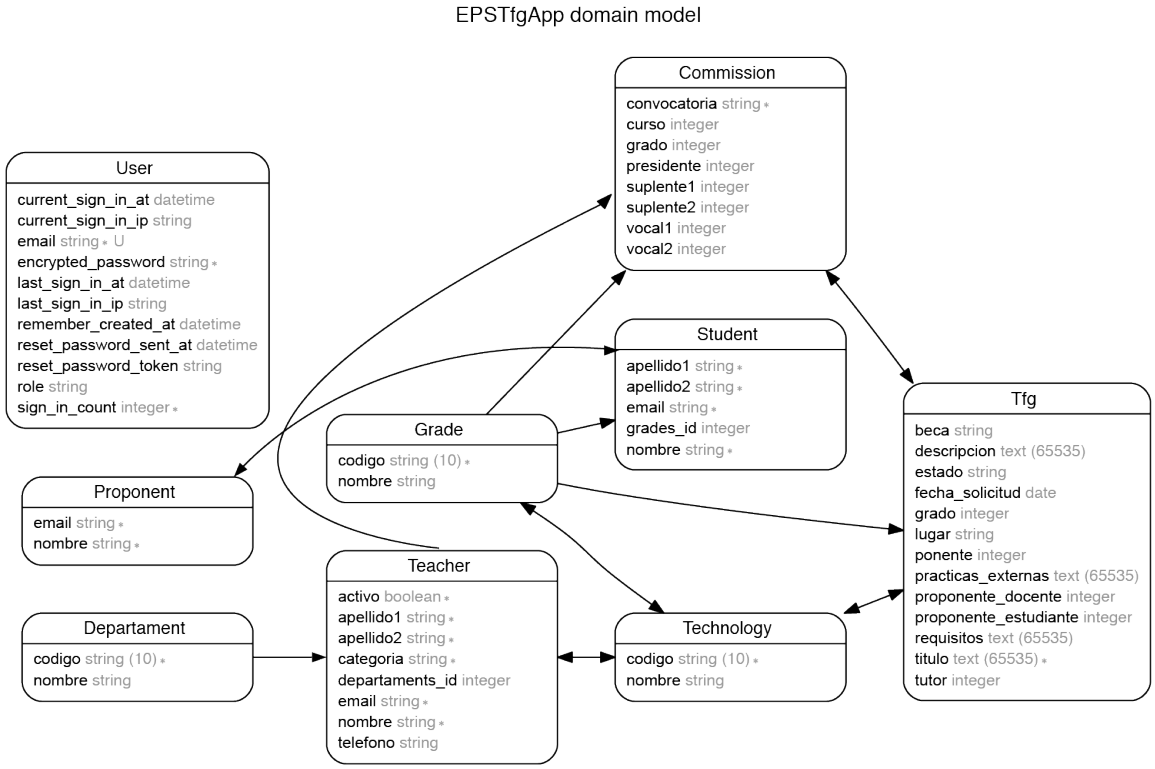


Figura 4-1: Diagrama UML de la aplicación

Generación de módulos

RoR contiene un sistema de generación dinámico de módulos, en los que al utilizar un comando genera automáticamente los siguientes ficheros:

- Modificación del fichero de rutas añadiendo este nuevo módulo
- Modelo de datos
- Controlador
- Vista con inicio, listado, modificación y vista en detalle
- Tests unitarios
- Ficheros JavaScript, y SCSS específicos de este módulo
- Fichero de migración de base de datos en el que se le añaden las nuevas tablas

Por ejemplo, si queremos generar el módulo de proponente el comando sería:

```
$ rails generate scaffold Proponent email:string nombre:string
```

Enrutado

Para que se puedan cargar las vistas y su contenido en el navegador es necesario hacer uso del fichero de rutas de RoR. Este fichero define las acciones posibles al hacer una petición HTTP y de qué tipo.

```
get 'tfg_assignment/index'
resources :tfgs_assignment do
  collection do
    post 'list'
    post 'finish'
    put 'update_assignment'
  end
end
```

En este fragmento de código se puede ver las acciones posibles para la vista de asignación de comisiones (modificación de estado).

- Acceso a la página inicial de cambio de estado de TFG.
- Peticiones de tipo POST para listar los TFG y para terminar el procedimiento.
- Peticiones de tipo PUT para ejecutar el cambio de estado mediante AJAX.

Modelo

Aunque la generación dinámica de RoR facilita el trabajo, es necesario realizar modificaciones en el modelo para realizar las relaciones entre módulos.
Por ejemplo, este es el resultado del modelo TFG

```
class Tfg < ActiveRecord::Base

  self.primary_key = "codigo"
  has_and_belongs_to_many :technologies
  has_and_belongs_to_many :commissions
  belongs_to :grade
  has_one :proponent, class_name: "proponente_estudiante", foreign_key:
"students_id"
  has_one :student, class_name: "asignado_estudiante", foreign_key:
"students_id"
  has_one :teacher, class_name: "proponente_docente", foreign_key:
"teachers_id"
  has_one :teacher, class_name: "ponente", foreign_key: "teachers_id"

  attr_accessor :tipo_usuario
  attr_accessor :proponente
  attr_accessor :email
  attr_accessor :pec
  attr_accessor :departamento
  attr_accessor :consentimiento
  attr_accessor :tecs

  OFERTADO = 'Ofertado'
  PROPUESTO = 'Propuesto'
  ASIGNADO = 'Asignado'
  DEFENDIDO = 'Defendido'
  A_COMISION = 'A Comision'
  EN_COMISION = 'En Comision'
```

```
end
```

Como se puede ver, en RoR no se añaden los atributos de una entidad desde el modelo, sólo las relaciones y las claves. Los atributos se añaden en el fichero schema.rb al crear un módulo. Este fichero es el que se toma de referencia al generar una migración o poblar con datos de prueba las tablas, para revisar la estructura de la base de datos, y si no es equivalente, modificarla.

Controlador

Los controladores hacen de nexo de unión entre el modelo de datos y la vista, capturando las peticiones realizadas en la vista, accediendo al modelo para obtener los datos que la petición solicita, y devolverlos a la vista. Debe constar (como mínimo) de tantos métodos como acciones estén enrutadas en el fichero de rutas de RoR. Por ejemplo. El controlador TFG contiene como mínimo un método para listar los TFG, mostrar uno en detalle, eliminar y editar.

Los controladores también se encargan de "masajear la información" para añadirla al modelo de datos. Especial mención tiene el controlador de generación de comisiones, que se encarga de generar las comisiones utilizando los controladores y *helpers* de profesores y TFG.

Vista

Ficheros ERB

RoR genera las vistas de usuario mediante el uso de ficheros de tipo .erb. Estos ficheros contienen código HTML y scripts de Ruby, para "pintar" la información rescatada del modelo o del controlador. Este proceso se realiza mediante la inserción de etiquetas <%= %>

```
<% if can? :manage, Tfg %>
    <li><%= link_to "Propuesta TFG", new_tfg_path %></li>
    <li class="divider"></li>
<% end %>
```

En esta porción de código se puede ver un ejemplo del menú de navegación, en el que si el usuario tiene un rol que pueda gestionar objetos de tipo TFG se pinta un elemento del menú para proponer TFG

JavaScript

Por otra parte también se hace uso de JavaScript y jQuery para recolocar elementos en las tablas en el proceso de generación de comisiones.

```
function update_status(code, status)
{
    data = "code=" + code + "&status=" + status;

    $.ajax({
        type: "PUT",
        url: "/tfgs_assignment/update",
        data: data
    })
}
```

Foundation y SCSS

Para la generación de los estilos de las vistas hacemos uso de Foundation. Este framework contiene un set de estilos pre-definidos para formularios, textos, menús y un sistema de grids para posicionar la información y que esta no se deforme en función del tamaño de la pantala o el tipo de dispositivo.

```
<div class="row">
  <div class="column small-12 large-6 large-centered">
    <h2>Nueva propuesta de TFG</h2>
    <%= form_for @tfg do |f| %>

        <% if @tfg.errors.any? %>
            <div data-alert class="alert-box alert">
              <h2><%= pluralize(@tfg.errors.count, "error") %>
prohibited this tfg from being saved:</h2>

              <ul>
                <% @tfg.errors.full_messages.each do |message| %>
                    <li><%= message %></li>
                <% end %>
              </ul>
            </div>
        <% end %>
        <%= render "#{@tfg.current_step}_step", :f => f %>
        <%= f.submit "Continuar", class:"button left" %>
        <%= f.submit "Volver", class:"button right secondary", :name =>
"back_button" unless @tfg.first_step? %>
    <% end %>
  </div>
</div>
```

En esta porción de código se puede ver como funciona el sistema de grids.

- La clase row genera una nueva subdivisión del espacio en columnas.
- La clase column da inicio a un nuevo set de columnas. En el caso del ejemplo el contenido de ese div se muestra con todo el ancho posible para dispositivos de ancho de hasta 1.080px (móvil y tablet) y del 50% y centrado para tamaños mayores.

De serie, los rows de Foundation se dividen en 12 columnas con un padding de 10px. Todos estos valores son modificables mediante un fichero de configuración SCSS. También se pueden sobreescribir utilizando un fichero SCSS una capa "por encima" del framework Foundation.

Para la generación de estilos se utilizan ficheros SCSS. Este lenguaje de hojas de estilo permite el uso de variables, funciones, mixins e indentación para generar código CSS, de manera que esta parte del desarrollo sea más sencilla de manejar.

```
$color-white : #FFF;
$color-dark  : #000;
$font-montserrat : "Montserrat";
.nav {
    &-tabs {
        > li {
            > a {
                -webkit-border-radius: 0;
                -moz-border-radius: 0;
                border-radius: 0;
                font-family: $font-montserrat;
                color: $color-white;
                @include transition(all, 0.3s);
                border: none;
                &:hover {
                    background-color: $color-dark;
                    //border-color: $color-dark;
                    @include transition(all, 0.3s);
                    border: none;
                }
            }
            &.active {
                > a {
                    color: $color-whine;
                }
            }
        }
    }
}
```

En este ejemplo de código del menú se puede ver como se definen 3 variables (2 de color, y una para definir la tipografía) y el código necesario para dar estilos al menú.
Este código genera las siguientes clases:

- Nav-tabs > li > a
- Nav-tabs > li > a:hover
- Nav-tabs > li:active > a

Como se puede ver, este sistema de *indentación* permite ver más fácilmente el alcance de los estilos, y permite generar con poco código muchas más clases y reglas que utilizando un fichero CSS standard.

RoR, como parte de los procesos para lanzar una aplicación hace uso de la gema Sass para compilar los ficheros SCSS en un CSS situado en la carpeta /assets que se añade en la cabecera de los documentos HTML.

Algoritmos

Algoritmo de generación de comisiones

Para generar las comisiones es necesario el uso de un algoritmo que reparta los TFG y los agrupe por tecnologías de la mejor manera para que todas las comisiones queden equilibradas.

Primero se crea cada comisión generando un código que lo identifique unívocamente (TFGaño_curso_convocatoria_númeroDeComisión

Tras generar las comisiones en las que se insertarán los TFG se obtienen todos los TFG cuyo estado es A Comisión. Esta consulta devuelve un listado ordenado por id de TFG. Para realizar las asignaciones necesitamos ordenar los datos por número de tecnologías implicadas. Esto posibilitará que se puedan repartir los TFG correctamente.

Por ejemplo, si tenemos un grupo de TFG con las siguientes tecnologías:

```
{2} {1,3} {1} {2,4,5} {2,3} {5} {1,2} {2,3,4}
```

El resultado sería el siguiente array:

```
{1} {2} {5} {1,2} {1,3} {2,3} {2,3,4} {2,4,5}
```

para ello utilizamos el siguiente algoritmo:

```
tfgs.each do
  tfgs_disordered.push(i)
end

temporal = tfgs_disordered.pop
tfgs_ordered.push(temporal)
posicion = 0
tfgs_disordered.each do |i|
  tfgs_ordered.size.times do |j|
    resultado = compare_to(tfgs_ordered[j], i)
    if resultado == -1
      posicion = j
    else
      posicion = j+1
    end
  end
  tfgs_ordered.insertar(posicion, i)
end
```

Tras esto ya se pueden asignar los TFG a las comisiones. Para ello se recorren las comisiones y se comprueba el número de coincidencias entre cada TFG contenido y el

TFG que se pretende insertar. Posteriormente se inserta en la comisión que tiene más coincidencias y no está completa.

```
foreach @tfg
  foreach @commision
    numero_coincidencias = buscar_coincidencias(@comision, @tfg)
    if (numero_coincidencias > @comision_anterior)
      @comision_posible = @comision
    end
  endforeach
  asignar_tfg(comision, tfg)
endforeach
```

Algoritmo de asignación de profesores a comisiones

Tras la asignación de los TFG a las comisiones se pasa a asignar profesores a las comisiones. Para ello es necesario tener en cuenta que los presidentes de las comisiones sólo pueden ser Catedráticos o Profesores Titulares. Para ello se añaden todos los profesores posibles en una pila, y se comprueban uno a uno que el docente posee tecnologías contenidas en la comisión. Si las contiene, se guarda el candidato. Si no, se añade a la lista de otros candidatos al final.

El procedimiento para asignar profesores es el siguiente:

```
foreach commission
  president_candidates.shuffle
  other_candidates

  while !president
  candidate = president_candidates.pop
    if technologies_included(commision, candidate)
      presidente = candidate
    else
      other_candidates.push_last(candidate)
    end
  end
  other_candidates.push(president_candidates)
  while !vocal1
    candidate = other_candidates.pop
    if technologies_included(commision, candidate)
      vocal1 = candidate
    else
      other_candidates.push_last(candidate)
    end
    ...
  end
end
```

El algoritmo también contempla la posibilidad en la que se la pila de profesores se quede vacía, por lo que se vuelve a añadir todos los profesores

Despliegue

Para realizar los despliegues en los entornos de desarrollo, *staging* y desarrollo se ha utilizado la herramienta Capistrano, un gestor de tareas para que, tras configurarlo permite realizar tareas de despliegue.

El comando ejecutado es:

```
cap production deploy
```

El procedimiento que realiza tras la configuración el el siguiente:

- Inicia una conexión SSH al servidor de destino.
- Accede al repositorio y clona el contenido de la rama en una carpeta con la fecha y hora del despliegue.
- Compila los assets de la aplicación y minifica los ficheros javascript y css.
- En la versión a desplegar, crea enlaces simbólicos para las carpetas que no se almacenan en el repositorio (estáticos, imágenes, datos específicos del usuario).
- Modifica el enlace simbólico a la versión actual.
- Elimina las cachés de la aplicación.
- Cierra la conexión SSH.

5 Integración, pruebas y resultados

Integración

Para realizar la integración, y una vez configurado el servidor de desarrollo, sólo es necesario ejecutar el siguiente comando desde la raíz del proyecto:

```
cap production deploy | cap staging deploy | cap development deploy
```

Este comando ejecuta la tarea de Capistrano para el despliegue en producción, copia el código guardado en la rama *master, staging* o *dev*, cambia el enlace simbólico de la versión a mostrar y borra las cachés cd RoR.

Pruebas

El plan de pruebas de la aplicación se ha dividido en 3 bloque claramente diferenciados, ya que el tipo y el ámbito de estas son distintas.

Pruebas locales de integración

Tras la codificación de una mejora o módulo se han realizado pruebas a nivel local, comprobando que la funcionalidad desarrollada realizaba las acciones deseadas a nivel general, y con datos de prueba.

Pruebas unitarias

Ruby on Rails tiene integrado un sistema de pruebas unitarias, por lo que se pueden generar tests tras añadir un módulo con relativa facilidad. En estas pruebas se han probado las inserciones y modificaciones en la base de datos usando el modelo. La comparación de resultados obtenidos tras realizar una consulta y la comparación de resultados mostrados con los resultados teóricos. A su vez las pruebas unitarias testean los posibles valores y límites de cada campo de cada modelo de la aplicación. Estos tests se generan en la carpeta /test y están compuestos por

- Fichero YAML que contiene los datos que se utilizarán para ejecutar los tests. Un fichero YAML es un formato de serialización de datos, al estilo de XML.
- Fichero.rb con el script para realizar las pruebas. Este fichero obtiene los datos del fichero YAML y los inserta, modifica o elimina, en función del test.

Para realizar los test sólo es necesario ejecutar el siguiente comando:

```
$ rake test test/models/tfg_test.rb
```

Pruebas de integración

Tras realizar una mejora, testearla en local y subirla al servidor de desarrollo se han explorado todas las posibilidades acciones a nivel de usuario, explorando cada página, cada posible ruta, y todos los procedimientos posibles. En este apartado se han desarrollado las pruebas de seguridad, en las que se ha tratado de acceder a funcionalidades restringidas para los distintos roles de usuario.

6 Conclusiones y trabajo futuro

Conclusiones

En este proyecto se ha conseguido alcanzar el objetivo básico que se perseguía: Desarrollar una herramienta online capaz de gestionar la información relacionada con los TFG de la EPS-UAM de manera autónoma y accesible desde cualquier lugar, tanto en un entorno local como en el entorno de desarrollo.

Se ha invertido una gran parte del tiempo en investigar las tecnologías utilizadas en entornos reales. La decisión de utilizar las tecnologías seleccionadas ha exigido la formación del autor en dichas tecnologías ya que no se tenían conocimientos previos sobre ellas, realizando múltiples pruebas antes de aplicarlo al proyecto. El aprendizaje de un framework tan completo y complejo como RoR ha consumido gran parte del tiempo del trabajo.

El framework ha resultado complejo al principio, aunque con el tiempo se ha presentado como muy cómodo y útil. Esto ha hecho que haya ciertas funcionalidades contempladas en un principio que no se han podido desarrollar –p.e., generación de estadísticas e informes sobre TFG. En cambio se decidió por construir un sistema robusto y flexible que pueda utilizarse en la práctica y extensible en un futuro, en vez de realizar un desarrollo completo, pero que no conlleve un análisis realista del problema con garantías.

Otra parte del tiempo se ha invertido en la configuración del servidor VPS; la conexión del servidor RoR con NGINX ha sido bastante compleja, y ha obligado a cambiar de Unicorn a Passenger como conector entre servidores. Unicorn era demasiado complejo y está enfocado a grandes sistemas distribuidos, y el aprendizaje exigía demasiado tiempo, cuando no hemos podido formalizar el acceso a máquinas de producción (lo cual imposibilitaba las pruebas).

Las pruebas de despliegue con Capistrano, para futuros despliegues en entornos de producción ha resultado bastante complejo. A pesar de que Capistrano está especialmente pensado para aplicaciones RoR hay ciertas configuraciones y detalles que hacen que la configuración sea un poco engorrosa.

Trabajo futuro

Como trabajo futuro queda el despliegue a servidores reales de producción, que aunque la aplicación ya está preparada para ello, será necesario replicar las configuraciones aplicadas en el VPS de OVH, y realizar un despliegue inicial mediante Capistrano.

Tras el despliegue en producción se pueden crear nuevas funcionalidades en el sistema, como la visualización de estadísticas (muy útiles para los coordinadores y ya contemplado

como requisito futuro prioritario), el almacenamiento y relación de TFG de años anteriores y la conexión con la base de usuarios de la UAM, de modo que los usuarios puedan acceder con sus credenciales de la universidad.

También se puede refinar el aspecto visual del sitio, o mostrar ciertos aspectos en el proceso de generación de comisiones, como por ejemplo los estudiante asignados a los TFG a defender/defendidos.

Debido al carácter modular y escalable de la aplicación, se podría ampliar el uso de ésta a otra escuela o facultad desplegando múltiples máquinas para cada centro, o utilizando una sola instancia de la aplicación como sistema global para la gestión de los TFG en la universidad.

Referencias

[1] "Trabajo de Fin de Grado. Guía Docente", Universidad Autónoma de Madrid http://www.uam.es/ss/Satellite?blobcol=urldata&blobheader=application%2Fpdf&blobheadername1=Content-disposition&blobheadername2=pragma&blobheadervalue1=attachment%3B+filename%3D17846_TFG.pdf&blobheadervalue2=public&blobkey=id&blobtable=MungoBlobs&blobwhere=1242773730909&ssbinary=true

[2] "Ruby on Rails Guides", Ruby on Rails http://guides.rubyonrails.org/

[3] "UBUNTU, NGINX, UNICORN, RAILS, CAPISTRANO, RBENV, OH MY!", The Flying Developer, http://theflyingdeveloper.com/server-setup-ubuntu-nginx-unicorn-capistrano-postgres/

[4] "How to deploy a Rails App", Digital Ocean https://www.digitalocean.com/community/tutorials/how-to-deploy-a-rails-app-with-passenger-and-nginx-on-ubuntu-14-04

[5] "Managing responsive breakpoint with SASS", Sitepoint http://www.sitepoint.com/managing-responsive-breakpoints-sass/

[6] "Foundation for sites", Zurb Foundation http://foundation.zurb.com/sites/docs/

[7] "Active Record", The Odin Project http://www.theodinproject.com/ruby-on-rails/active-record-basics

[8] "Sass Basics", Sass Lang http://sass-lang.com/guide

[9] "Deploying with Capistrano", Bean Salk http://guides.beanstalkapp.com/deployments/deploy-with-capistrano.html

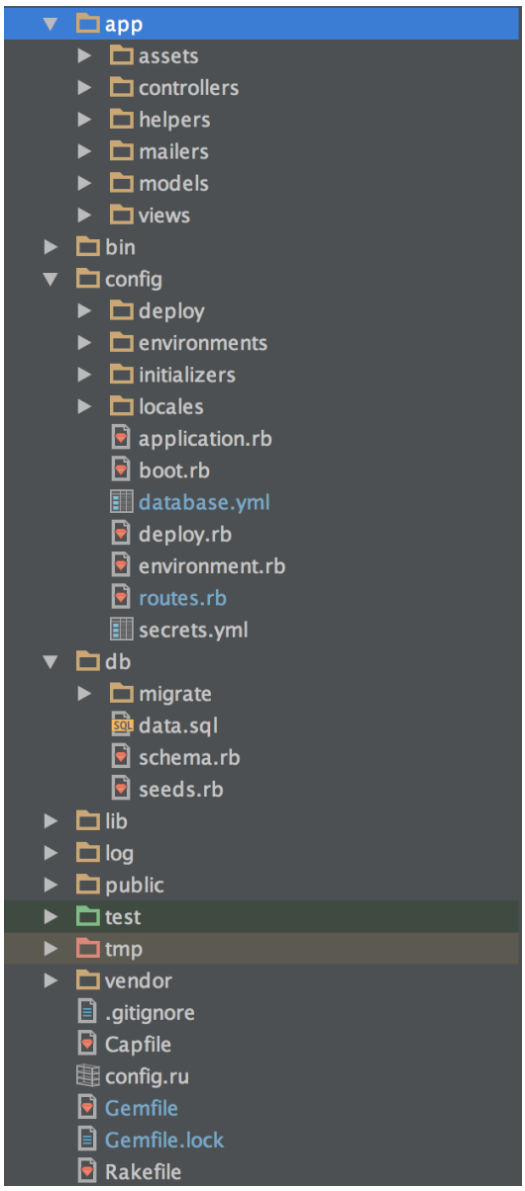
[10] "Getting Started", Capistranorb http://capistranorb.com/

[11] "Authentication with CanCan and Devise", Hibbard http://hibbard.eu/authentication-with-devise-and-cancancan-in-rails-4-2/

Anexos

A Ruby on Rails. Organización de ficheros

Los proyectos de una aplicación RoR deben tener una estructura específica para poder funcionar correctamente. En la siguiente figura se detallará la estructura de ficheros de un proyecto RoR básico:

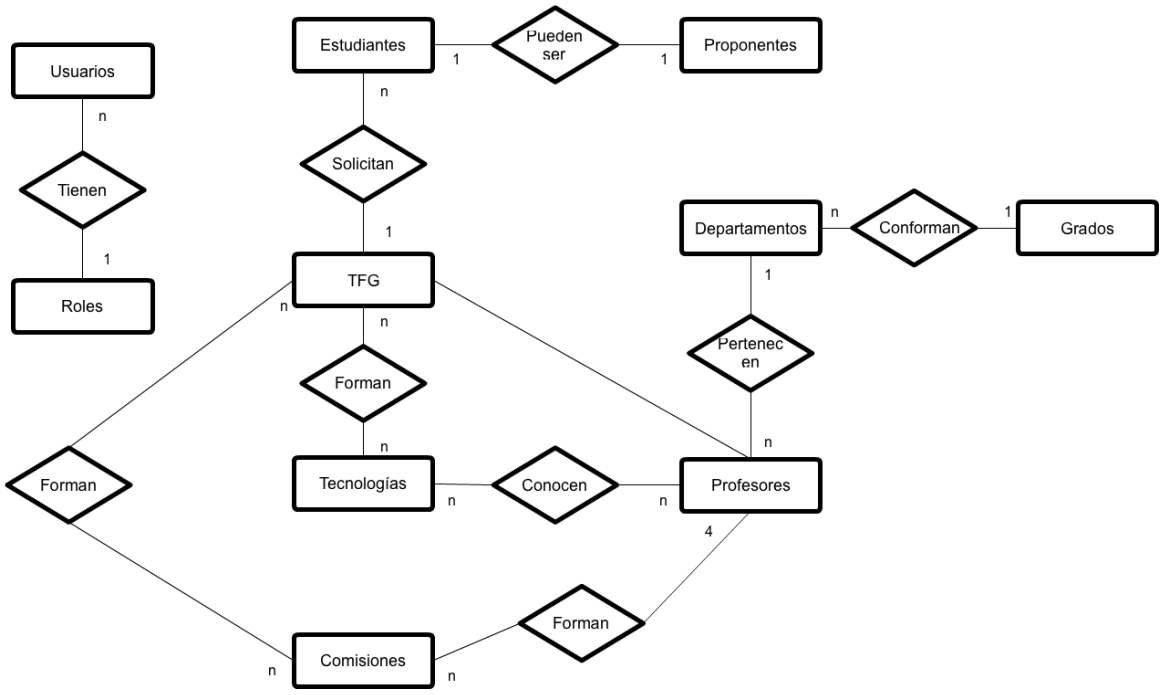


- **App**: Contiene todo el código relacionado con el núcleo de la aplicación. Controladores, modelos y vistas se deben situar aquí junto con los ficheros JavaScript o SCSS.
- **Config**: En esta carpeta se encuentran todos los ficheros necesarios para configurar el servidor RoR y base de datos. Cabe destacar los siguientes ficheros.

    - Database.yml: En este fichero se definen los sockets, bases de datos y usuarios de todos los entornos.
    - Routes.rb: Definen las rutas accesibles desde la aplicación y para cada rol.
    - Secrets.yml: Este fichero definen las contraseñas o hashes necesarios para configurar la aplicación en entornos de producción.
- **Db**: Esta carpeta contiene toda la información necesaria para generar la base de datos y el contenido.
    - Schema.rb: define la estructura de la base de datos
    - Seeds.rb: Fichero para poblar la base de datos
- **Test**: Contiene todos los test unitarios. También se pueden automatizar los test de integración. Se ubican en la misma carpeta.
- **Vendor**: Carpeta con todas las librerías necesarias para ejecutar la aplicación.
- **Capfile**: Fichero de configuración de Capistrano.
- **Gemfile**: Fichero en el que se incluyen todas las gemas necesarias para cargar la aplicación.

B Modelo de datos. Diagrama E-R y tablas

Diagrama E-R



Tablas

Comisiones

Campo	Tipo	Longitud	Comment
codigo	VARCHAR	255	Clave primaria. Código de comision
convocatoria	VARCHAR	255	
created_at	DATETIME		
updated_at	DATETIME		
presidente	INT	11	Id del presidente
vocal1	INT	11	Id del vocal
vocal2	INT	11	Id del vocal 2
suplente1	INT	11	Id del suplente 1
suplente2	INT	11	Id del suplente 2
grado	INT	11	Id del grado al que pertenece
curso	INT	11	

Esta tabla almacena toda la información relativa a las comisiones generadas, referencia a las ids de los docentes que la componen y el grado al que pertenece.

Comisiones_TFG

Campo	Tipo	Longitud	Comment
commission_id	VARCHAR	255	Id de comisión
tfg_id	VARCHAR	255	Id de TFG
sala	VARCHAR	50	
fecha	DATE		
hora	TIME		

Esta tabla relaciona las comisiones con los TFG que la componen. A su vez está contemplado que almacenen la fecha y hora de la defensa.

Departamentos

Campo	Tipo	Longitud	Comment
id	INT	11	
codigo	VARCHAR	10	
nombre	VARCHAR	255	
created_at	DATETIME		
updated_at	DATETIME		

Esta tabla almacena la información relativa a los departamentos a los que pertenecen los docentes

Grados

Campo	Tipo	Longitud	Comment
id	INT	11	
codigo	VARCHAR	10	
nombre	VARCHAR	255	
created_at	DATETIME		
updated_at	DATETIME		

Esta tabla almacena el listado de grados que se pueden administrar con la aplicación, su código y nombre.

Grados_Tecnologias

Campo	Tipo	Longitud	Comment
grade_id	INT	11	
technology_id	INT	11	

Esta tabla relaciona los grados, con las tecnologías que la conforman

Proponentes

Campo	Tipo	Longitud	Comment
id	INT	11	
nombre	VARCHAR	255	
email	VARCHAR	255	
created_at	DATETIME		
updated_at	DATETIME		

Esta tabla define la entidad proponente mediante un nombre y un email.

Roles

Campo	Tipo	Longitud	Comment
id	INT	11	
name	VARCHAR	255	
created_at	DATETIME		
updated_at	DATETIME		

Esta tabla define la entidad rol para ser usada posteriormente por los usuarios

Estudiantes

Campo	Tipo	Longitud	Comment
id	INT	11	
nombre	VARCHAR	255	
apellido1	VARCHAR	255	
apellido2	VARCHAR	255	
email	VARCHAR	255	
created_at	DATETIME		
updated_at	DATETIME		
grades_id	INT	11	

Esta tabla define la entidad estudiante, y la relaciona con los grados en los que está matriculado.

Profesores

Campo	Tipo	Longitud	Comment
id	INT	11	
nombre	VARCHAR	255	
apellido1	VARCHAR	255	
apellido2	VARCHAR	255	
email	VARCHAR	255	
telefono	VARCHAR	255	
categoria	VARCHAR	255	
activo	TINYINT	1	
created_at	DATETIME		
updated_at	DATETIME		
departaments_id	INT	11	

Profesores_Tecnologias

Campo	Tipo	Longitud	Comment
teacher_id	INT	11	
technology_id	INT	11	

Esta tabla relaciona los profesores con las tecnologías en las que está formado

Tecnologías

Campo	Tipo	Longitud	Comment
id	INT	11	
codigo	VARCHAR	10	
nombre	VARCHAR	255	
created_at	DATETIME		
updated_at	DATETIME		

La tabla tecnologías define la entidad tecnología, mediante un código y un nombre

Tecnologias_TFG

Campo	Tipo	Longitud	Comment
tfg_id	VARCHAR	20	
technology_id	INT	11	

Esta tabla relaciona un TFG con las tecnologías que lo componen

TFGs

Campo	Tipo	Longitud	Comment
codigo	VARCHAR	20	
titulo	TEXT		
fecha_solicitud	DATE		
descripcion	TEXT		
requisitos	TEXT		
lugar	VARCHAR	255	
beca	VARCHAR	255	
practicas_externas	TEXT		
estado	VARCHAR	255	
created_at	DATETIME		
updated_at	DATETIME		
proponente_estu...	INT	11	
proponente_docente	INT	11	
ponente	INT	11	
tutor	INT	11	
grado	INT	11	

La tabla TFG define la entidad TFG, toda la información relativa, y la relaciona con la tabla de proponentes, y docentes.

Usuarios

Campo	Tipo	Longitud	Comment
id	INT	11	
email	VARCHAR	255	
encrypted_passw...	VARCHAR	255	
reset_password_t...	VARCHAR	255	
reset_password_s...	DATETIME		
remember_create...	DATETIME		
sign_in_count	INT	11	
current_sign_in_at	DATETIME		
last_sign_in_at	DATETIME		
current_sign_in_ip	VARCHAR	255	
last_sign_in_ip	VARCHAR	255	
created_at	DATETIME		
updated_at	DATETIME		
role	VARCHAR	255	

Esta tabla define la entidad usuarios, almacena su password encriptado e información relativa al acceso.