Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE

Diseño e implementación de una aplicación web para mejorar los resultados académicos de los alumnos de una escuela

Estudante: Jorge Becerra Ríos
Dirección: Juan Raposo Santiago

A Coruña, setembro de 2020.

Dedicado a mis padres y a mi hermana

Agradecimientos

En primer lugar quiero agradecer a mis padres el apoyo y los buenos consejos recibidos que me han llevado a estar aquí hoy escribiendo estas líneas.

A mi hermana y amigos por estar siempre a mi lado en los buenos y malos momentos

A el director de este proyecto, Juan Raposo Santiago, por sus consejos, ideas y ayuda recibida tanto en este proyecto como en las asignaturas en las que me ha impartido clase.

Resumen

El objetivo de este proyecto es el diseño e implementación de una aplicación web que de soporte a los profesores y alumnos de una determinada escuela. Para ello contará con una herramienta de generación de cuestionarios tipo Test, un foro de resolución de dudas, un sistema de mensajería, un calendario de eventos y un sistema de clasificación de mejores alumnos por materia y grupo para incentivar a los alumnos a colaborar en el foro.

Para el desarrollo del sistema se usará React para la generación de la interfaz y Spring Boot para la generación del modelo.

Abstract

The objective of this project is to design and implement a web application that supports the teachers and students of a certain school. For this, it will have a tool for generating Test-type questionnaires, a forum for resolving doubts, a messaging system, a calendar of events and a system for classifying the best students by subject and group to encourage students to collaborate in the forum.

For the development of the system, React will be used for the generation of the interface and Spring Boot for the generation of the model.

Palabras clave:

- React
- Redux
- Rest
- Bootstrap
- LMS
- Mensajería
- Foro
- Test
- Ranking
- Calendario

Keywords:

- React
- Redux
- Rest
- Bootstrap
- LMS
- Messaging
- Forum
- Test
- Ranking
- Calendar

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introdución

1.1 Motivación

Son muchas las escuelas que continúan enseñando a través de Métodos obsoletos y que no conectan con los estudiantes actuales. Aportar a los estudiantes herramientas digitales útiles puede ayudar a que estos muestren un mayor interés por el contenido que se imparte, y más aún si se les permite a través de las herramientas elegir de forma anónima parte de los contenidos a dar en clase. A la falta de interés de algunos estudiantes se le suma la falta de recursos efectivos que ayuden a la monitorización temprana e individualizada de los alumnos, siendo ésta indispensable para adaptarse a las necesidades y ritmos de cada estudiante. Dotar a los profesores de herramientas para una mejor comunicación y trato individualizado hacia sus alumnos es básico para tratar de conseguir que ningún alumno se quede atrás.

La capacidad de toma de decisiones, el trabajo en equipo y una buena capacidad de reflexión son valores muy apreciados en el mundo en que vivimos y no siempre es sencillo para los colegios ayudar a los alumnos a la adquisición de estas habilidades. Con herramientas informáticas adecuadas, esta difícil tarea podría resultar algo más sencilla.

Con herramientas digitales y el uso creativo de las mismas por parte de los profesores, se puede conseguir acercar el conocimiento a los alumnos de una manera más entretenida para ellos. Generar interés por el conocimiento es uno de los objetivos principales que debe tratar de cumplir la enseñanza. Para conseguirlo, es importante conocer las inquietudes e intereses de los alumnos y también el grado de satisfacción de los alumnos con respecto a la enseñanza que se les está ofreciendo. Conocer esto podría resultar muy sencillo si se dispusiese de herramientas cómodas y fáciles de utilizar para la realización de cuestionarios tanto anónimos como no anónimos.

1.2 Situación actual

En el mercado existen numerosos sistemas LMS (Learning Content Management). Aunque algunos de ellos ofrecen la oportunidad de ser adaptados a las necesidades de cada centro, a veces es mejor comenzar un desarrollo desde 0 y proporcionar a los usuarios en cuestión, un sistema diferenciador, totalmente nuevo y 100% adaptable a las necesidades actuales y futuras de los usuarios.

1.3 Objetivos

El objetivo de este proyecto ha sido la realización de una aplicación web que ayudase al profesorado y alumnos de una escuela a mejorar los resultados académicos de los alumnos y favorecer una enseñanza en la cual los alumnos pudiesen participar activamente. A través de esta aplicación se ha pretendido conseguir que los alumnos muestren un mayor interés por el aprendizaje, que sean capaces de colaborar los unos con los otros en la resolución de dudas y que se involucren en los contenidos que se imparten, tomando decisiones al respecto.

Para lograrlo, esta aplicación cuenta con 2 tipos de usuarios. Por una parte tenemos a los profesores y por otra a los alumnos.

La aplicación cuenta con un sistema de mensajería a través del cual los profesores pueden comunicarse con alguno de sus alumnos o con la totalidad de los alumnos pertenecientes a uno de sus grupos. Con esto se pretende que los profesores tengan un canal a través del cual poder comunicar calificaciones, recordar eventos, hacer recomendaciones personalizadas o responder a dudas de sus alumnos.

La web también posee una herramienta que les recuerda a los alumnos aquellos acontecimientos importantes tales como exámenes, excursiones o eventos culturales. A través de la aplicación los profesores podrán registrar eventos en un calendario y visualizarlos. También tendrán la posibilidad de borrar alguno de sus eventos si así lo desean. Los alumnos, a través de su calendario, podrán ver los eventos registrados por sus profesores.

La aplicación también cuenta con un foro, un lugar donde los alumnos pueden ayudarse los unos a los otros en la resolución de dudas. Los profesores van a poder moderar la discusión e incentivar a aquellos alumnos que resuelvan correctamente las dudas de sus compañeros, dándoles una puntuación numérica a cada una de las respuestas que ellos consideren. Estas puntuaciones quedarán registradas y con ellas se creará un ranking de alumnos para cada uno de los grupos y para cada una de las asignaturas, donde los alumnos mejor puntuados serán premiados.

Los profesores van a poder registrar en la aplicación los premios (libros, material deportivo, entradas para eventos culturales ...), asociarles un grupo, una asignatura y una fecha

límite, e indicar si los puntos se reiniciarán una vez finalice el concurso. Una vez alcanzada la fecha límite, la aplicación registrará los puntos que cada uno de los alumnos ha obtenido con respecto a su grupo y la asignatura asociada al premio. Tanto alumnos como profesores podrán acceder a este ranking para visualizarlo.

Adicionalmente, la aplicación se encarga de que los alumnos formen parte de la elección del contenido que se impartirá, pudiendo así responder de forma anónima a cuestionarios que el profesor plantee tales como la elección de la novela a leer en la materia de Lengua y literatura, la elección de la pieza musical a interpretar en clase de música o el ganador del concurso de poesía. Estos cuestionarios serán creados por los profesores a través de la propia aplicación web. Los profesores podrán visualizar los resultados del cuestionario cuando lo deseen.

Adicionalmente a los cuestionarios anónimos, la aplicación web ofrece a los profesores la posibilidad de enviar cuestionarios no anónimos tipo test a sus alumnos. Estos deberán responderlos a modo de tareas y serán evaluados por ello de forma automática. Los profesores podrán ver el resultado de los test de sus alumnos a través de la propia aplicación. Los profesores también tendrán acceso a estadísticas sobre la distribución de calificaciones entre los alumnos, pudiendo saber así que tanto por ciento de los alumnos sacó una nota entre el 9 y el 10, que tanto por ciento sacó una nota entre el 8 y el 9 y así sucesivamente.

Resumiendo, los alumnos podrán estar al tanto de comunicados de sus profesores a través del sistema de mensajería y del calendario de eventos. Además, los alumnos podrán plantear dudas en el momento en que les surjan y responder a las de sus compañeros. Esto incrementará las posibilidades de obtener una respuesta rápida y no quedarse atascados en un problema. Con ayuda de los cuestionarios anónimos los profesores podrán amoldar el contenido que imparten a las preferencias de sus alumnos. Con esto se espera que los alumnos muestren una mayor predisposición por aprender. Con los cuestionarios evaluables, los profesores dispondrán de una herramienta muy cómoda de evaluación que les permitirá obtener información acerca del progreso de sus alumnos.

Esta aplicación se ha desarrollado con la intención de que en el futuro se le puedan añadir fácilmente nuevas funcionalidades. Esto es fundamental puesto que un buen sistema de este tipo ha de poder estar abierto a la inclusión de nuevas funcionalidades que sean capaces de cubrir nuevas necesidades de los usuarios. Para conseguirlo, durante el desarrollo se han empleado un conjunto de buenas prácticas y patrones de diseño. En el capítulo 8 de este documento se comentan cuales han sido los principales patrones de diseño empleados durante el desarrollo de la aplicación y en qué consisten.

Capítulo 2

Herramientas y tecnologías utilizadas

Para la elección de las tecnologías a usar se han tenido muy en cuenta las tendencias actuales de diseño y desarrollo. Es por ello que se han decidido utilizar librerías como React o lenguajes como javaScript.

A continuación se muestran los lenguajes de programación empleados en la aplicación:

- JAVA: Ha sido elegido por sus numerosas ventajas, en especial:
  - Su robustez.
  - La orientación a objetos, que nos permite construir aplicaciones modulares y usar patrones de diseño. Esto ayuda a que el software pueda ser modificado de manera más sencilla en el futuro (ampliando sus funcionalidades o modificando las ya existentes).
  - Una gran comunidad con mucho código Open source de terceros listo para ser utilizado.
- JPQL: Ha sido elegido para poder realizar consultas a la base de datos que no son posibles realizar a través de convenciones de nombrado. Además la sintaxis es similar a la de SQL.
- JAVASCRIPT [1]: Se ha escogido puesto que es muy útil para desarrollar páginas dinámicas y posee librerías muy interesantes como React.

A continuación se muestran las tecnologías usadas para la aplicación de estilos en la interfaz de usuario:

- HTML [2]: Ha sido escogido puesto que es un lenguaje de marcado reconocido y admitido por cualquier tipo de explorador y se integra a la perfección con las tecnologías de frontend escogidas.

- CSS [3]: Ha sido elegido debido a su facilidad de uso y la posibilidad de separar cómodamente los estilos de la lógica de la interfaz.
- BOOSTRAP [4]: Se ha decidido utilizar esta librería puesto que permite desarrollar de manera más rápida los elementos visuales de la aplicación tales como formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML Y CSS. Además dispone de un sistema de rejillas bastante fácil de utilizar que permite la reubicación de los elementos gráficos en la pantalla en función del tamaño de la misma, sin comprometer la usabilidad.
- react-star-ratings [5]: Se ha escogido puesto que ofrece componentes muy útiles y atractivos para representar puntuaciones.
- react-toastify [6]: Se ha escogido puesto que ofrece elementos visuales muy útiles a la hora de mostrar por pantalla que una operación se está procesando.
- react-calendar [7]: Se ha escogido puesto que implementa un componente Calendario bastante flexible y fácil de utilizar.
- chart.js: Se ha escogido puesto que posee componentes que ayudan a mostrar resultados en gráficas muy visuales.

Para la base de datos se ha utilizado :

- MYSQL: Ha sido elegido puesto que es una base de datos relacional capaz de integrarse con las demás tecnologías escogidas.

Para el desarrollo del backend se han empleado diversas tecnologías. Para implementar la capa de acceso a datos se ha usado:

- JPA HIBERNATE: Implementa la API de JPA, la cual proporciona anotaciones para mapear entidades a una base de datos relacional y una API para interactuar con la base de datos (operaciones CRUD, lenguajes de consultas, etc).
- SPRING DATA: Librería para implementar ágilmente los DAOs. Se ha decidido trabajar con estas tecnologías puesto que proporcionan una manera rápida y muy legible de operar sobre la base de datos.

Para el desarrollo de las pruebas de integración se ha usado:

- JUNIT [8]: Se ha escogido este framework para la implementación de pruebas automatizadas puesto que nos permite realizar tanto pruebas de unidad como de integración de una manera sencilla y rápida.

Para la construcción de la capa lógica de negocio se ha usado:

- SPRINGBOOT [9]: Ha sido escogido puesto que facilita la creación de aplicaciones basadas en Spring, reduciendo el esfuerzo que tienen que realizar los desarrolladores para configurar la aplicación. Entre los beneficios de usar spring están por ejemplo la inyección de dependencias, el Marshalling JSON o la gestión de transacciones de forma sencilla.

- Librería spring-web: Usada para implementar ágilmente la capa de servicios REST. Gracias a la librería Jackson los DTOs se convierten automáticamente a JSON y viceversa.

Para el desarrollo de la interfaz de usuario (Frontend) se han usado:

- JSX: Gracias a la unión de HTML y javaScript en una sintaxis más compacta y legible el código implementado resulta más fácil de leer y comprender.

- REACT [10]: Se ha decidido usar esta librería javaScript principalmente por la gran acogida que está teniendo actualmente y la gran comunidad que se está generando a su alrededor. Además la librería React se completa con una gran cantidad de librerías de terceros que ofrecen una gran cantidad de funcionalidades. Como principales ventajas del uso de React podemos destacar:

  - DOM virtual, que mejora el renderizado del navegador.
  - Posee una gran comunidad a su alrededor.
  - Las aplicaciones web desarrolladas con React están basadas en componentes reutilizables. Esto facilita que la aplicación sea más escalable y fácil de mantener ya que los errores sucederán en la propia funcionalidad del componente o en la comunicación con los demás.

- REDUX [11]: Se ha decidido utilizar esta librería javaScript debido a que se integra perfectamente con React y se quiere gestionar el estado de la aplicación de forma centralizada y modular. Gracias a Redux conseguimos sacar la mayor parte del estado así como su lógica de modificación de los componentes.

Como herramientas para el desarrollo se han empleado las siguientes:

- MAVEN [12]: Es una herramienta capaz de gestionar un proyecto software completo, desde la etapa en la que se comprueba que el código es correcto, hasta que se despliega la aplicación, pasando por la ejecución de pruebas y generación de informes y documentación. Por otra parte, con Maven la gestión de dependencias entre módulos y distintas versiones de librerías se hace muy sencilla

- ECLIPSE: Ha sido el IDE escogido para este proyecto debido a la gran cantidad de funcionalidades que reúne, entre otras, las siguientes:
  - Permite entender el proyecto en su conjunto de una manera más sencilla.
  - Permite depurar el código.
  - Posee herramientas de auto completado y de validación.
  - Permite refactoring.
  - Existen una gran cantidad de plugins disponibles que se integran a la perfección con el propio IDE.
- NPM [13]: Se ha decidido utilizar este gestor de paquetes porque a través de él se pueden obtener librerías de la comunidad y administrar dependencias de forma más sencilla.
- POSTMAN [14]: Se ha escogido puesto que permite comprobar el funcionamiento de la capa de servicios REST del backend de forma sencilla y rápida. Además ofrece la posibilidad de utilizar Bearer Authentication.
- GIT [15]: Se ha escogido puesto que es una herramienta muy completa de control de versiones.

Capítulo 3

Estado del arte

Son muchos los LMS (del inglés learning management system), sistema de gestión de aprendizaje, que hay disponibles para su uso en el mercado. A continuación se enumeran dos de ellos:

Chamilo [16]: Este sistema destaca por su sencillez de uso. Chamilo es una herramienta muy útil para apoyar al docente en su transición de cursos presenciales a cursos virtuales o semi-virtuales. Posee una gran variedad de funcionalidades: gestión de los cursos, importar o crear documentos de audio, vídeo o imágenes, construir ejercicios y exámenes con calificación automática, crear o importar contenidos (SCORM y AICC), configurar la entrega de trabajos virtuales, comunicarse a través del foro, publicar anuncios, establecer un aula virtual, gestionar las calificaciones, crear encuestas, registrar asistencias etc. En la figura 3.1 (página 9) se muestran algunos de los módulos de esta aplicación y en la figura 3.2 (página 10) se muestra una imagen con un ejemplo de una interacción con el programa, a la hora de añadir una nueva pregunta a un test.

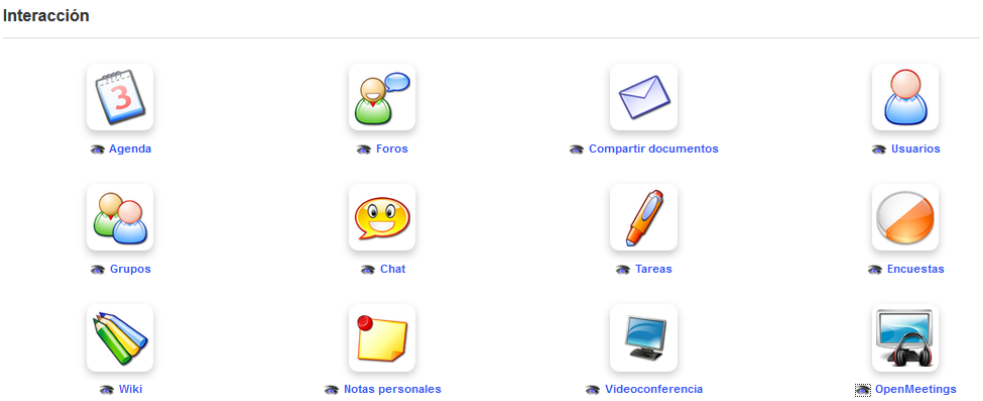


Figura 3.1: Algunos módulos presentes en la aplicación Chamilo

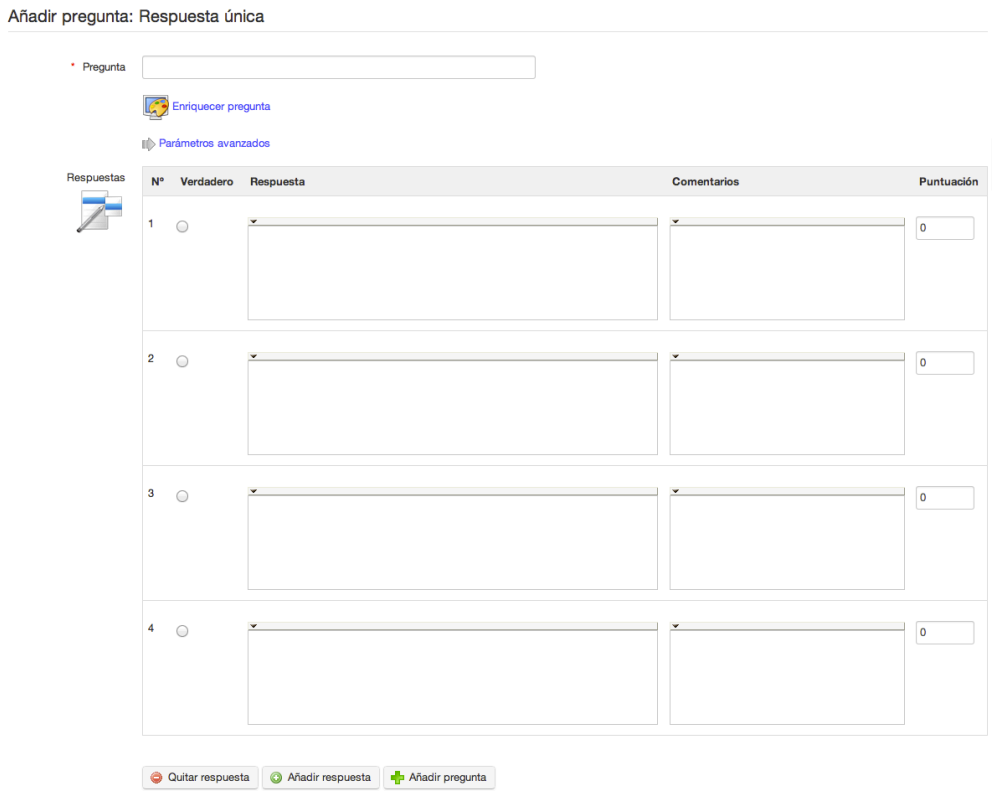


Figura 3.2: Ejemplo de interacción: (añadir una nueva pregunta a un test)

Como podemos ver, a pesar de poseer una interfaz bastante sencilla e intuitiva, es bastante mejorable en el apartado estético, luciendo un poco anticuada para los estándares actuales.

Otro ejemplo de LMS es Fedena [17]. Al igual que la aplicación aquí desarrollada, se centra en escuelas de primaria y secundaria. Ofrece una gran cantidad de funcionalidades. Fedena proporciona acceso de inicio de sesión para maestros, personal administrativo, estudiantes e incluso padres. Los padres pueden discutir el progreso de sus hijos con el maestro y realizar un seguimiento del progreso académico de sus alumnos. Fedena ERP tiene módulos para administrar horarios, asistencia, exámenes, libros de calificaciones, biblioteca, transporte, calendario escolar y eventos. También es una buena herramienta de colaboración que utiliza sus complementos Tarea, Discusión, Encuesta, Blog y Videoconferencia. Hay un sistema de mensajería interno dentro de Fedena, pero también puede integrarse con herramientas externas como correo electrónico y mensajes de texto. En la figura 3.3 (página 11) se muestra uno de los menús que ofrece Fedena.

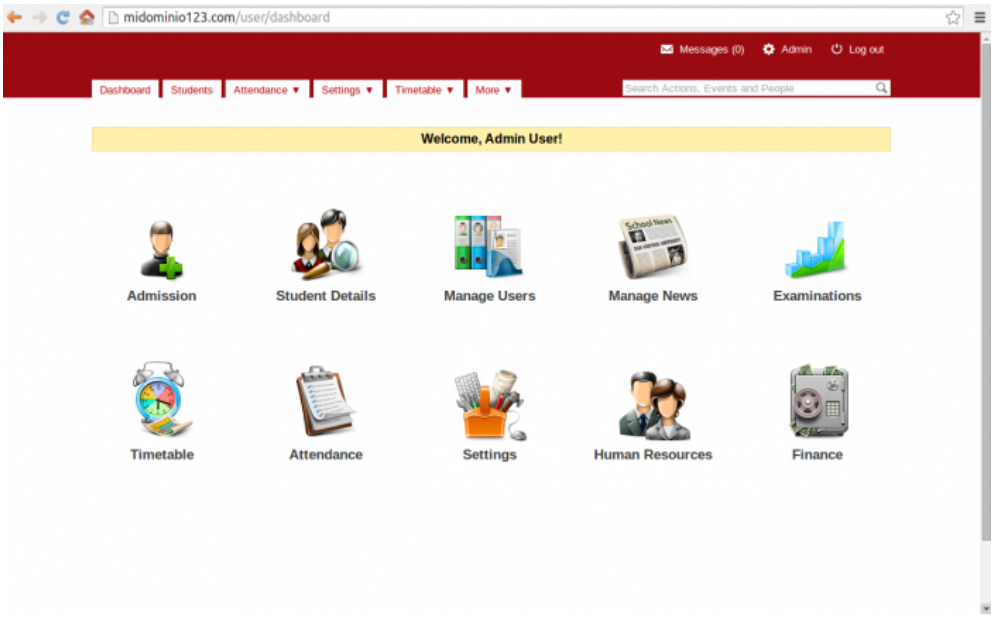


Figura 3.3: Algunos módulos presentes en la aplicación Chamilo

Fedena tiene como principal ventaja la gestión de un montón de procedimientos distintos y la inclusión de los padres de los alumnos en el sistema para que estos puedan realizar un seguimiento de los progresos académicos de sus hijos. Al igual que tiene ventajas, Fedena también tiene alguna desventaja. Quizás debido a la gran cantidad de funcionalidades que posee, muchos usuarios se quejan de que son necesarios demasiados pasos para llegar a ciertas funcionalidades del sistema, y no siempre esos pasos son tan intuitivos como les gustaría.

Estos solo son dos ejemplos de los muchos sistemas LMS que existen. Algunos de ellos, como Fedena, incluso están especializados en escuelas de primaria y secundaria. Estos sistemas ya implementados son capaces de ofrecer una gran cantidad de funcionalidades distintas. Es por ello que con el desarrollo de este proyecto no se pretende crear un sistema capaz de sustituir a los ya existentes, puesto que no se posee ni el tiempo ni los recursos suficientes para hacerlo. Lo que se pretende con este proyecto es sentar las bases de un nuevo desarrollo, con intención de que este sea fácilmente adaptable a las necesidades futuras que pudieran llegar a tener los usuarios del sistema.

Capítulo 4

Análisis de viabilidad

En este apartado se expone como ha sido el estudio de viabilidad del proyecto. El estudio de viabilidad debe preceder siempre a cualquier desarrollo y este ha de ser determinante a la hora de decidir si se lleva a cabo o no el proyecto. Para que el proyecto pueda llevarse a cabo es necesario asegurarse de que el proyecto se puede finalizar con éxito con los medios reservados para él.

En cuanto a la viabilidad económica del proyecto, se ha prestado especial atención a los recursos requeridos y a los ya disponibles. En este caso, los recursos (electricidad, conexión a internet, hardware, software, etc) consumen una cantidad de recursos económicos pequeña, en especial los recursos software ya que se ha decidido utilizar software libre sin costes adicionales. El total estimado de costes de estos recursos ha sido de aproximadamente 200 €, teniendo en cuenta el coste de la luz e internet asociados a los 6 meses de proyecto estimados. En cuanto a los recursos humanos, al tratarse de un trabajo académico no remunerado, no procede su análisis al menos en este apartado. Más adelante se simulan distintos perfiles ficticios asociados a la elaboración del proyecto para hacer una simulación de lo que podría haber sucedido en un entorno real de trabajo.

Capítulo 5

Introducción al desarrollo realizado

5.1 Arquitectura software del sistema

Hoy en día diseñadores, desarrolladores y en general cualquier persona involucrada en el desarrollo software centran su atención en dos aspectos fundamentales: cómo lograr construir mejores aplicaciones en menos tiempo, y cómo utilizar mayor cantidad de estándares en el diseño de las aplicaciones que permitan mayor reutilización del código y mejores mantenimientos de los sistemas desarrollados. Para la realización de este proyecto también se han tenido muy presentes estos dos aspectos, y eso ha conllevado , entre otras cosas, a la decisión de emplear una arquitectura cliente-servidor. Esta arquitectura tiene dos partes claramente diferenciadas, por un lado la parte del servidor y por otro la parte del cliente. Cada una de estas partes constituye una aplicación en si misma.

5.1.1 Backend

El servidor o backend es el encargado de proporcionar servicios al cliente. El backend desarrollado está compuesto de varias capas. Por una parte tenemos la capa modelo, constituida a su vez por la capa de acceso a datos y la capa lógica de negocio. Por otra parte tenemos la capa de servicios REST, que se encarga de recibir las peticiones del cliente, delegarlas en la capa modelo y enviar al cliente las respuestas que la capa modelo le proporciona en un formato adecuado. La capa lógica de negocio es la que se encarga de definir las reglas de negocio (funcionalidades del sistema) y de interactuar con la capa de acceso a datos cuando es necesario.

5.1.2 Frontend

El cliente o frontend se encarga de la parte visual de la aplicación y de la interacción con el usuario. El cliente también se encarga de lanzar las peticiones necesarias al backend y recibir

las respuestas que éste le manda.

La interacción entre frontend y backend está basada en el paso de mensajes, en este caso a través del protocolo HTTP.

Algunas de las ventajas que podemos apreciar en el uso de la arquitectura cliente-servidor son:

- Desacoplamiento: Separación clara entre los componentes de un programa; lo cual permite su implementación por separado e incluso la posibilidad de implementar simultáneamente ambas capas, frontend y backend.
- Encapsulamiento de servicios: Los detalles de la implementación de un servicio son transparentes al cliente.
- Escalabilidad: Horizontal (añadir clientes) y vertical (ampliar potencia de los servidores).
- Permite una mayor especialización de los desarrolladores al permitirles desarrollar una de las capas sin tener que preocuparse por el resto, pudiendo así por ejemplo centrarse en la adquisición de nuevas y mejores habilidades para el desarrollo de la vista (frontend) y dejar a otro u otros programadores el desarrollo del backend.

En la figura 5.1 (página 16) se puede ver un esquema de la arquitectura cliente-servidor.

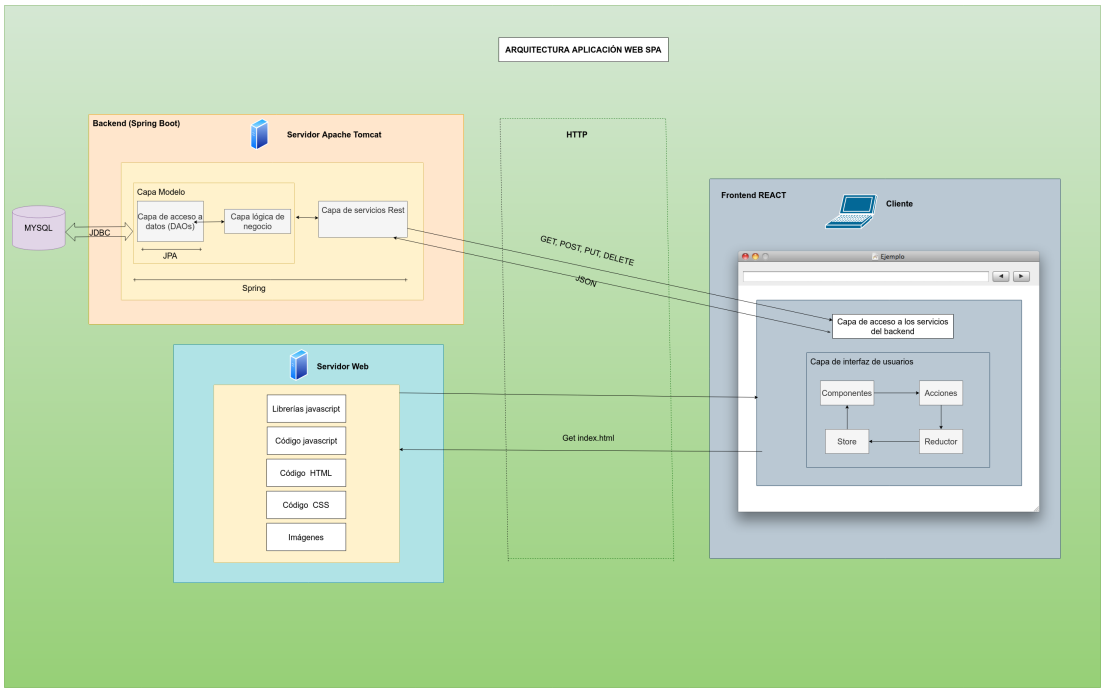


Figura 5.1: Arquitectura cliente-servidor

5.2 Metodología e iteraciones

Se ha optado por seguir una metodología basada en Proceso Unificado de Desarrollo Software. Se aplica el patrón "divide y vencerás", dividiendo el proyecto en subproyectos más pequeños. Siguiendo el esquema iterativo e incremental, las distintas iteraciones van aumentando la funcionalidad del sistema hasta llegar a la implementación completa. En cada iteración del proyecto se realiza análisis, diseño, implementación y pruebas, a excepción de en la primera iteración, en la que no se realiza software todavía. Como marca PUD, el desarrollo ha de ser dirigido por los casos de uso. Es por ello que en la primera iteración se realizan los diagramas de casos de uso y la correspondiente especificación de requisitos software. Al tratarse de una metodología basada en iteraciones e incrementos, se ha optado por construir la aplicación en 7 iteraciones. La primera de ellas, la iteración 0, es la reservada para la elaboración de la especificación de requisitos software, el análisis de tecnologías y la elaboración de la planificación inicial. En la especificación de requisitos se detallan los casos de uso que se encargarán de dirigir el desarrollo. La funcionalidad a implementar en cada iteración ha sido escogida en función de la relevancia de los requisitos. Las iteraciones que siguen a la primera iteración de especificación de requisitos son:

- Iteración 1 Análisis, diseño, implementación y pruebas del subsistema de usuarios
- Iteración 2: Análisis, diseño, implementación y pruebas del subsistema de mensajes
- Iteración 3: Análisis, diseño, implementación y pruebas del subsistema de calendario
- Iteración 4: Análisis, diseño, implementación y pruebas del subsistema de foro
- Iteración 5: Análisis, diseño, implementación y pruebas del subsistema de ranking
- Iteración 6: Análisis, diseño, implementación y pruebas del subsistema de tests

Además, como fases principales del trabajo podemos destacar las siguientes:

- Definición de objetivos
- Especificación de requisitos software
- Diseño del software
- Implementación del software
- Pruebas
- Realización de la memoria

Capítulo 6

Planificación y evaluación de costes

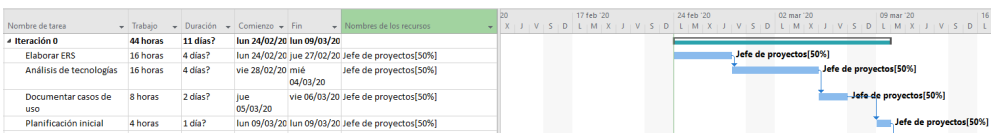
PARA evitar que un proyecto finalice sin éxito o sea cancelado antes de su finalización es necesario hacer una buena gestión de proyectos. Para ello es necesario una buena estimación de los parámetros esfuerzo, tiempo y coste y el posterior seguimiento y control de los mismos. En este apartado se van a simular la existencia de 3 perfiles distintos encarnados por 3 personas distintas.Son los siguientes:

- Jefe de proyectos: Es el encargado de capturar los requisitos, gestionar los recursos y realizar la planificación, seguimiento y control del proyecto.

- Analista: Diseña la solución al problema y le entrega al programador los artefactos necesarios para que este sepa cómo implementar la solución.

- Programador: Realiza la implementación y las pruebas del sistema.

Como ya hemos visto, el proyecto se divide en iteraciones. Además, cada una de ellas, es dividida por el jefe de proyectos en subtareas más pequeñas a las cuales les asigna una duración en horas-hombre y una cierta cantidad de recursos estimados. Esto da como resultado un calendario que muestra las fechas estimadas de inicio y finalización de cada una de las tareas. En las figuras 6.1 (página 19), 6.2 (página 20), 6.3 (página 20), 6.4 (página 20), 6.5 (página 20), 6.6 (página 20) y 6.7 (página 20) se muestra la planificación de cada una de las iteraciones.



Nombre de tarea	Trabajo	Duración	Comienzo	Fin	Nombres de los recursos
Iteración 0	**44 horas**	**11 días?**	**lun 24/02/20**	**lun 09/03/20**	
Elaborar ERS	16 horas	4 días?	lun 24/02/20	jue 27/02/20	Jefe de proyectos[50%]
Análisis de tecnologías	16 horas	4 días?	vie 28/02/20	mié 04/03/20	Jefe de proyectos[50%]
Documentar casos de uso	8 horas	2 días?	jue 05/03/20	vie 06/03/20	Jefe de proyectos[50%]
Planificación inicial	4 horas	1 día?	lun 09/03/20	lun 09/03/20	Jefe de proyectos[50%]

Figura 6.1: Planificación para la iteración 0

Iteración 1 Subsistema de usuarios	70 horas	17,5 días?	mar 10/03/20	jue 02/04/20	
Planificación y seguimiento	1 hora	0,25 días?	mar 10/03/20	mar 10/03/20	Jefe de proyectos[50%]
Análisis	2 horas	0,5 días?	mar 10/03/2	mar 10/03/2	Analista[50%]
Diseño	3 horas	0,75 días?	mar 10/03/2	mié 11/03/2	Analista[50%]
Implementación	48 horas	12 días?	mié 11/03/2	vie 27/03/20	Programador[50%]
Pruebas	16 horas	4 días?	vie 27/03/20	jue 02/04/20	Programador[50%]

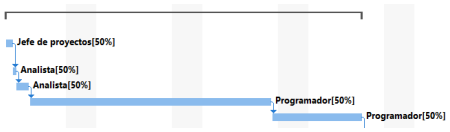


Figura 6.2: Planificación para la iteración 1

Iteración 2 Subsistema de mensajes	52 horas	13 días?	jue 02/04/20	mar 21/04/20	
Planificación y seguimiento	1 hora	0,25 días?	jue 02/04/20	jue 02/04/20	Jefe de proyectos[50%]
Análisis	1 hora	0,25 días?	jue 02/04/20	jue 02/04/20	Analista[50%]
Diseño	2 horas	0,5 días?	vie 03/04/20	vie 03/04/20	Analista[50%]
Implementación	40 horas	10 días?	vie 03/04/20	vie 17/04/20	Programador[50%]
Pruebas	8 horas	2 días?	vie 17/04/20	mar 21/04/2	Programador[50%]

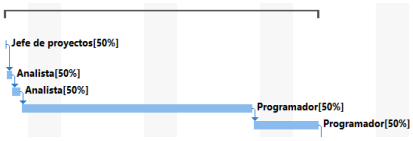


Figura 6.3: Planificación para la iteración 2

Iteración 3 Subsistema de calendario	59 horas	14,63 días?	mar 21/04/20	mar 12/05/20	
Planificación y seguimiento	1 hora	0,13 días?	mar 21/04/20	mar 21/04/20	Jefe de proyectos
Análisis	1 hora	0,25 días?	mar 21/04/2	mar 21/04/2	Analista[50%]
Diseño	1 hora	0,25 días?	mar 21/04/2	mié 22/04/2	Analista[50%]
Implementación	48 horas	12 días?	mié 22/04/2	vie 08/05/20	Programador[50%]
Pruebas	8 horas	2 días?	vie 08/05/20	mar 12/05/2	Programador[50%]

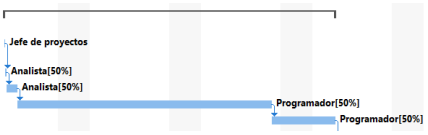


Figura 6.4: Planificación para la iteración 3

Iteración 4 Subsistema de foro	62 horas	15,38 días?	mar 12/05/20	mar 02/06/20	
Planificación y seguimiento	1 hora	0,13 días?	mar 12/05/20	mar 12/05/20	Jefe de proyectos
Análisis	1 hora	0,25 días?	mar 12/05/2	mar 12/05/2	Analista[50%]
Diseño	2 horas	0,5 días?	mar 12/05/2	mar 12/05/2	Analista[50%]
Implementación	48 horas	12 días?	mié 13/05/2	jue 28/05/20	Programador[50%]
Pruebas	10 horas	2,5 días?	vie 29/05/20	mar 02/06/2	Programador[50%]

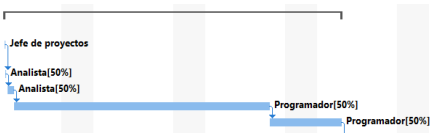


Figura 6.5: Planificación para la iteración 4

Iteración 5 Subsistema de ranking	79 horas	19,63 días?	mar 02/06/20	mar 30/06/20	
Planificación y seguimiento	1 hora	0,13 días?	mar 02/06/20	mar 02/06/20	Jefe de proyectos
Análisis	2 horas	0,5 días?	mar 02/06/2	mié 03/06/2	Analista[50%]
Diseño	4 horas	1 día?	mié 03/06/2	jue 04/06/20	Analista[50%]
Implementación	56 horas	14 días?	jue 04/06/20	mié 24/06/2	Programador[50%]
Pruebas	16 horas	4 días?	mié 24/06/2	mar 30/06/2	Programador[50%]

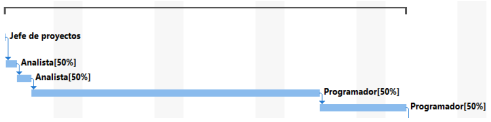


Figura 6.6: Planificación para la iteración 5

Iteración 6 Subsistema de tests	93 horas	23,13 días?	mar 30/06/20	vie 31/07/20	
Planificación y seguimiento	1 hora	0,13 días?	mar 30/06/20	mar 30/06/20	Jefe de proyectos
Análisis	1 hora	0,25 días?	mar 30/06/2	mar 30/06/2	Analista[50%]
Diseño	3 horas	0,75 días?	mar 30/06/2	mié 01/07/2	Analista[50%]
Implementación	64 horas	16 días?	mié 01/07/2	jue 23/07/20	Programador[50%]
Pruebas	24 horas	6 días?	jue 23/07/20	vie 31/07/20	Programador[50%]

Figura 6.7: Planificación para la iteración 6

En la figura 6.8 (página 21) se pueden ver las fecha de comienzo y fin del proyecto estimadas, y el coste total estimado de los recursos humanos del proyecto.

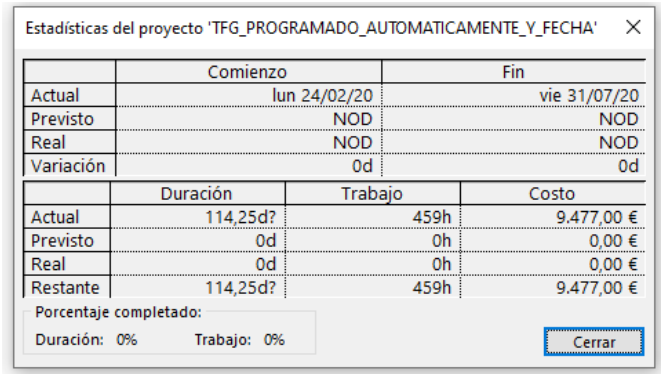


Figura 6.8

En la siguiente tabla se muestra el desglose de los costes estimados asociados al proyecto.

Perfil	**Precio/Hora**	**Horas**	**Precio x Horas**
Programador	18 €	386	6948
Analista	23 €	23	529
Jefe de proyectos	40 €	50	2000
Coste Recursos humanos			9477 €
Coste recursos técnicos			200 €
IVA (21%)			2032.17 €
Total			**11709.17 €**

Tabla 6.1: Desglose de costes estimados asociados al proyecto

Capítulo 7

Requisitos del sistema

Como ya hemos visto, en la primera iteración, iteración 0, se realizan entre otras cosas, el análisis de requisitos software y su correspondiente documentación de los casos de uso. La especificación de requisitos es fundamental para un correcto entendimiento del sistema a realizar. Los requisitos del sistema son utilizados por los ingenieros de software como punto de partida para el diseño del producto. Este tipo de requisitos agregan detalle y explican cómo el producto debe proporcionar los requisitos de usuario. Por una parte tenemos los requisitos funcionales, que son aquellos que describen lo que el sistema debe hacer. Éstos describen la interacción entre el sistema y su entorno. Para describir el entorno que interactúa con el sistema se han definido los siguientes actores:

- Usuario no identificado.
- Alumno.
- Profesor.

Cada uno de ellos tiene acceso a diferentes funcionalidades.

Por otra parte tenemos los requisitos no funcionales que representan las cualidades que debe tener el producto para que pueda resultar de utilidad para los usuarios. Por la naturaleza del problema se ha prestado especial atención a los siguientes requisitos no funcionales:

- Usabilidad: Es muy importante que el sistema sea fácil de usar tanto para usuarios con conocimientos tecnológicos como para los que no. El sistema ha de poder ser utilizado de manera cómoda por usuarios de todas las edades, desde los alumnos más pequeños, hasta los profesores más mayores y que no han tenido tanto contacto con herramientas digitales.
- Fiabilidad: El sistema ha de ser probado de forma minuciosa para que se pueda garantizar lo más posible que el sistema funciona correctamente de acuerdo a los requisitos del usuario.

- Seguridad: Este sistema mantiene información confidencial de alumnos, por ejemplo sus calificaciones. Estas deben ser accedidas sólo por el propio alumno y sus profesores. Además, el profesor tiene acceso a los cuestionarios resueltos. Estos no deberían ser en ningún caso accesibles para los alumnos, puesto que se vería comprometido el sistema de evaluación y por consiguiente no sería posible conocer si el alumno ha adquirido o no los conocimientos esperados.

En la especificación de requisitos se introducen los diagramas de los casos de uso y su interacción con los actores. En las figuras 7.1 (página 24), 7.2 (página 25), 7.3 (página 25) y 7.4 (página 26) se muestran las relaciones entre los distintos actores y los casos de uso.

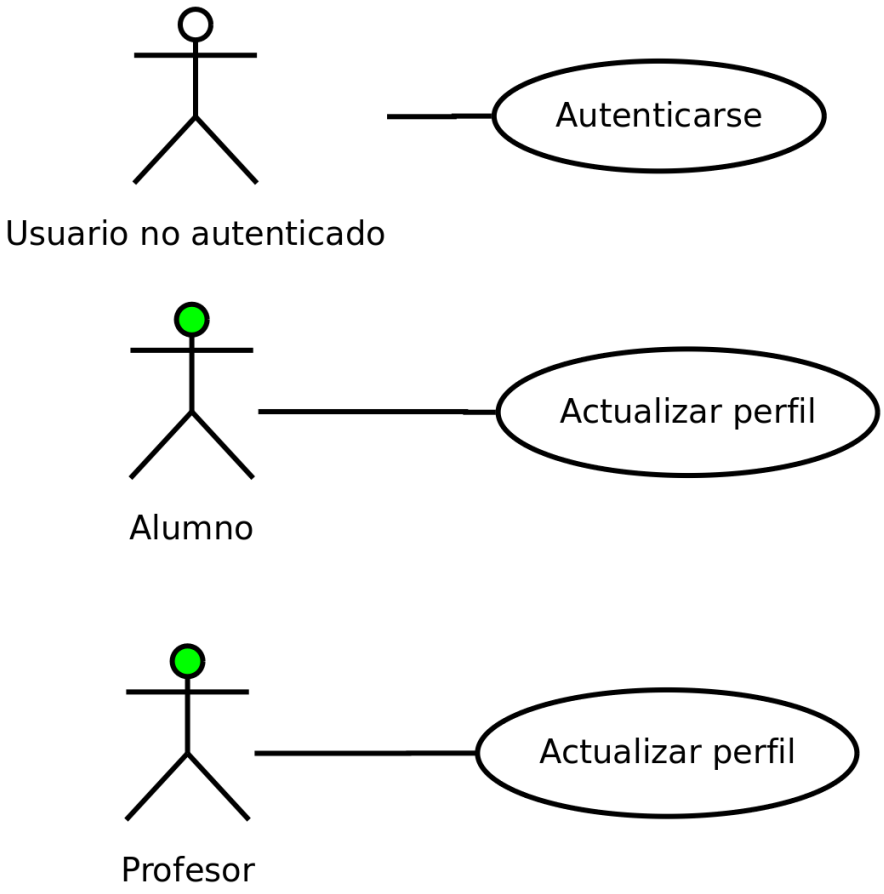


Figura 7.1: Diagrama de casos de uso (subsistema Usuarios)

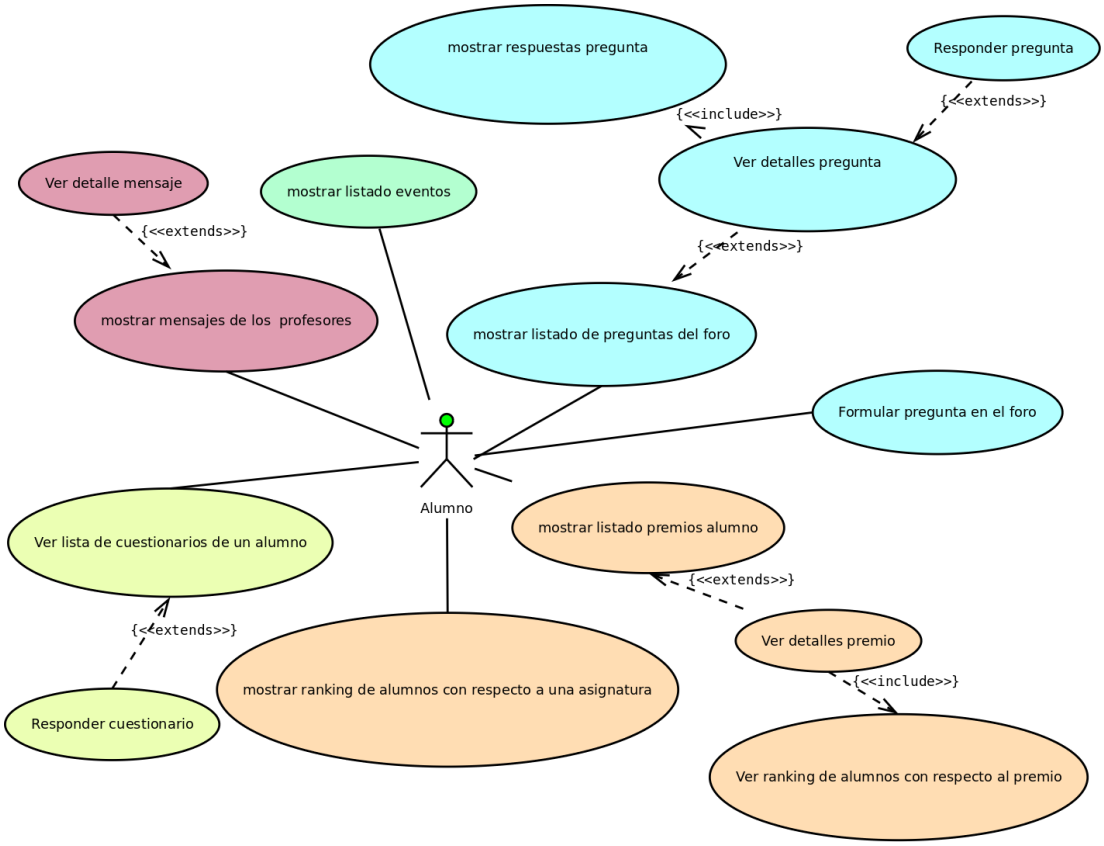


Figura 7.2: Diagrama de casos de uso Alumno

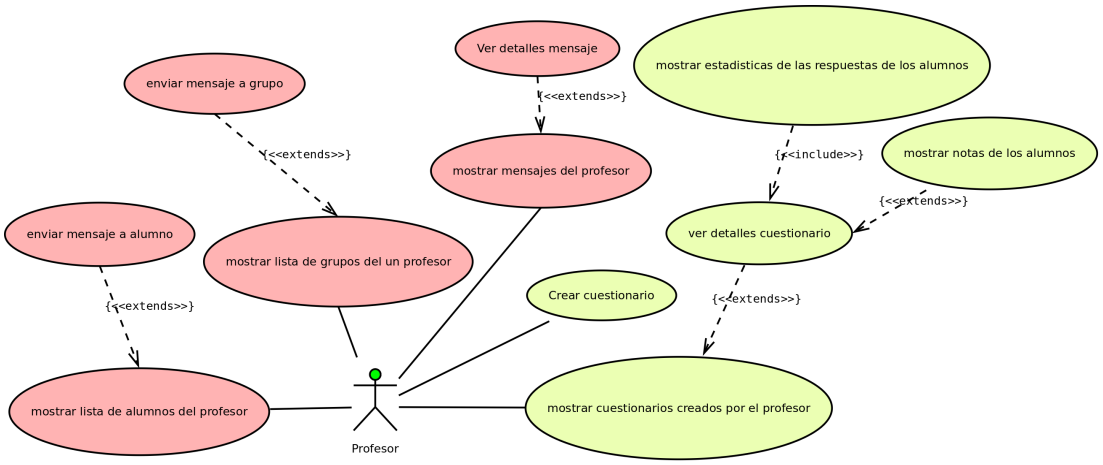


Figura 7.3: Diagrama de casos de uso Profesor (subsistemas Mensajes y Tests)

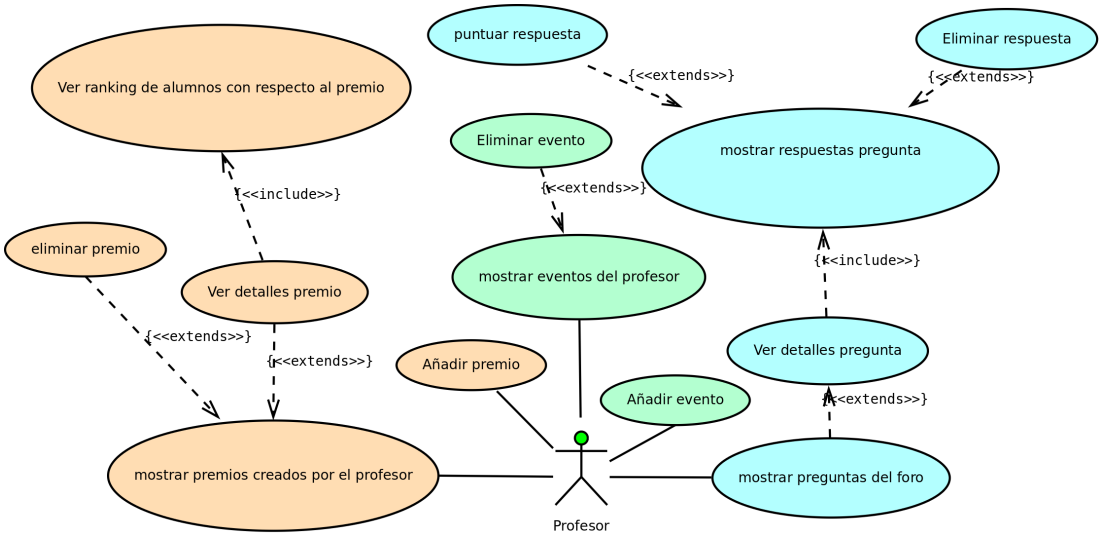


Figura 7.4: Diagrama de casos de uso Profesor (subsistemas Calendario, Foro y Ranking)

Capítulo 8

Diseño de la aplicación

En este capítulo se muestran las principales decisiones de diseño que se han tomado y los principales patrones de diseño que se han seguido. Las decisiones tomadas en cuanto al diseño proporcionan grandes beneficios al software generado, entre ellas las siguientes:

- Facilidad de cambio: El software puede ser fácilmente ampliable para satisfacer nuevas funcionalidades y corregir bugs existentes.
- Robustez: Gracias a principios como el de Inversión del control los cambios en una clase no se propagan al resto de clases del sistema, consiguiendo así un código más robusto y adaptable.
- Reusabilidad del software creado para éste u otros proyectos.
- Buena legibilidad del código gracias al nivel de cohesión alto de los componentes y el bajo acoplamiento.

8.1 Patrones y principios de diseño empleados

Los patrones de diseño son aquellos que expresan esquemas para definir estructuras o micro-arquitecturas de diseño o sus relaciones en los componentes, es decir, definen los detalles de cómo está construido un sistema de software mediante la colaboración de clases y objetos para resolver un problema general de diseño en un contexto particular. A continuación se explican los principales patrones y principios de diseño empleados:

- Patrón Data Access Object (DAO): Es un patrón de tipo estructural que abstrae el dominio (o capa lógica de negocio) de la persistencia, con la intención de ocultar la implementación de las operaciones sobre la base de datos a las capas superiores. Como ventajas de la utilización de este patrón podemos destacar:

  - Disposición de una interfaz de comunicación predecible, ya que están muy bien definidas las entradas y salidas de las funciones.
  - Sigue el beneficio de programar orientado a interfaces, la implementación es fácilmente reemplazable.
  - Las operaciones estarán encapsuladas en un único sitio, por lo que el código es reusable.
  - La implementación tiene un único propósito. No habrá lógica de negocio en los DAOs, sólo operaciones sobre la capa de persistencia.

- Paginación: Este patrón ayuda a la implementación de interfaces más eficientes en lo que a rendimiento se refiere. Además también mejora la usabilidad de la interfaz ya que reduce el número de clics que ha de hacer el usuario para navegar por el contenido. Ayuda a presentar de una manera más cómoda para el usuario una lista larga de elementos, proporcionando un número adecuado de los mismos y haciendo posible la navegación hacia adelante y hacia atrás para obtener el resto de elementos. El uso de la paginación es considerablemente más eficiente en cuanto al uso de los recursos, por ejemplo el uso de la memoria.

- Inversión del control: Es un principio de diseño de software en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales. Un ejemplo de inversión de control es por ejemplo la inyección de dependencias explicada a continuación.

- Inyección de Dependencias: También llamada DI (Dependency Injection). La inyección de dependencias es un patrón de diseño que nos permite construir software con poco acoplamiento. Básicamente, el patrón funciona cono un objeto que se encarga de construir las dependencias que una clase necesita y se las suministra. Esto implica que la clase ya no crea directamente los objetos que necesita, sino que los recibe de otra clase. Así se evita el uso de dependencias explícitas en el código de cada componente.

- DTO : Un DTO es un objeto que transporta datos entre procesos. La motivación de su uso tiene relación con el hecho de que la comunicación entre procesos se realiza generalmente mediante interfaces remotas (por ejemplo, servicios web), donde cada llamada es una operación costosa. Como la mayor parte del coste de cada llamada está relacionado con la comunicación de ida y vuelta entre el cliente y servidor, una forma de reducir el número de llamadas es usando un objeto, (el DTO), que agrega los datos que habrían sido transferidos en varias llamada en una sola llamada. Además un DTO sirve para enviar lo necesario y nada más que eso, reduciendo así el tamaño de la comunicación.

- Principio KISS: La simplicidad debe ser mantenida como un objetivo clave del diseño y cualquier complejidad innecesaria debe ser evitada.

- Principio YAGNI "You Aren't Gonna Need It": Predecir el futuro es una tarea complicada, así que se deben implementar los cambios solo cuando realmente se necesiten, no solo cuando se prevea que se necesitarán.

- DRY: El principio DRY (Don't Repeat Yourself) es una filosofía de definición de procesos que promueve la reducción de la duplicación. Según este principio toda pieza de información nunca debería ser duplicada debido a que la duplicación incrementa la dificultad en los cambios y evolución posterior, puede perjudicar la claridad y crear un espacio para posibles inconsistencias.

- Patrón observador: El patrón observador, también conocido como patrón de publicación/suscripción, es un patrón de diseño donde un objeto (llamado sujeto u observable), mantendrá una lista de "dependientes" llamados observadores. Ante un cambio de estado, este sujeto notificará automáticamente a los observadores. Un observador es un tipo de dependencia de uno a muchos entre objetos. Eso significa que un observador podría tener muchos suscriptores. Redux, que administra el estado en el frontend de forma centralizada y modular, funciona bajo este patrón.

- Patrón inmutable: En Redux se trata cada propiedad del estado como un objeto inmutable. La inmutabilidad en React mejora la eficiencia de la aplicación puesto que el virtual DOM de React posee mecanismos de optimización de renderizado que se benefician de la inmutabilidad del estado. La idea es que se puede provocar que un componente sólo se vuelva a renderizar si nuevas propiedades del nuevo estado son distintas a la última vez. Como test de igualdad se utiliza la igualdad referencial (shallow comparison).

- Principio Hollywood "No nos llame , ya le llamaremos": Consiste en desacoplar elementos de alto nivel y de bajo nivel que pueden trabajar juntos. El elemento de alto nivel mantiene el control y es el encargado de llamar a los elementos de bajo nivel cuando lo considera necesario. Los elementos de bajo nivel nunca llaman a los de alto nivel directamente. Es una forma de inversión del control como también lo es la inyección de dependencias. Un ejemplo de este principio lo podemos ver en el funcionamiento de los métodos sort del API de colecciones de Java utilizados en el proyecto, que delegan la parte de comparación del algoritmo en los métodos compareTo de los propios objetos.

8.2 Arquitectura escogida

Como ya se ha dicho anteriormente, la arquitectura escogida ha sido la arquitectura cliente-servidor. Esta arquitectura separa la aplicación en 2 componentes bien diferenciados, el backend y el frontend. En los siguientes apartados se muestra el diseño realizado de ambos componentes por separado.

8.3 Diseño del Backend

El servidor o backend es el encargado de proporcionar servicios al cliente. El backend desarrollado está compuesto de varias capas. Por una parte tenemos la capa modelo, constituida a su vez por la capa de acceso a datos y la capa lógica de negocio. La capa lógica de negocio es la que se encarga de definir las reglas de negocio (funcionalidades del sistema) y de interactuar con la capa de acceso a datos cuando es necesario. Por otra parte tenemos la capa de servicios REST, que se encarga de recibir las peticiones del cliente (o los clientes), delegarlas en la capa modelo y enviar las respuestas que la capa modelo le proporciona al cliente en un formato adecuado.

Para explicar el diseño del componente backend dividimos el backend en las siguientes partes:

- Datos
- Capa de acceso a datos
- Capa lógica de negocio
- Servicios

8.3.1 Datos

En las figuras 8.1 (página 31), 8.2 (página 31), 8.3 (página 32), 8.4 (página 32) y 8.5 (página 33) se muestran los diagramas de entidades de la capa lógica de negocio

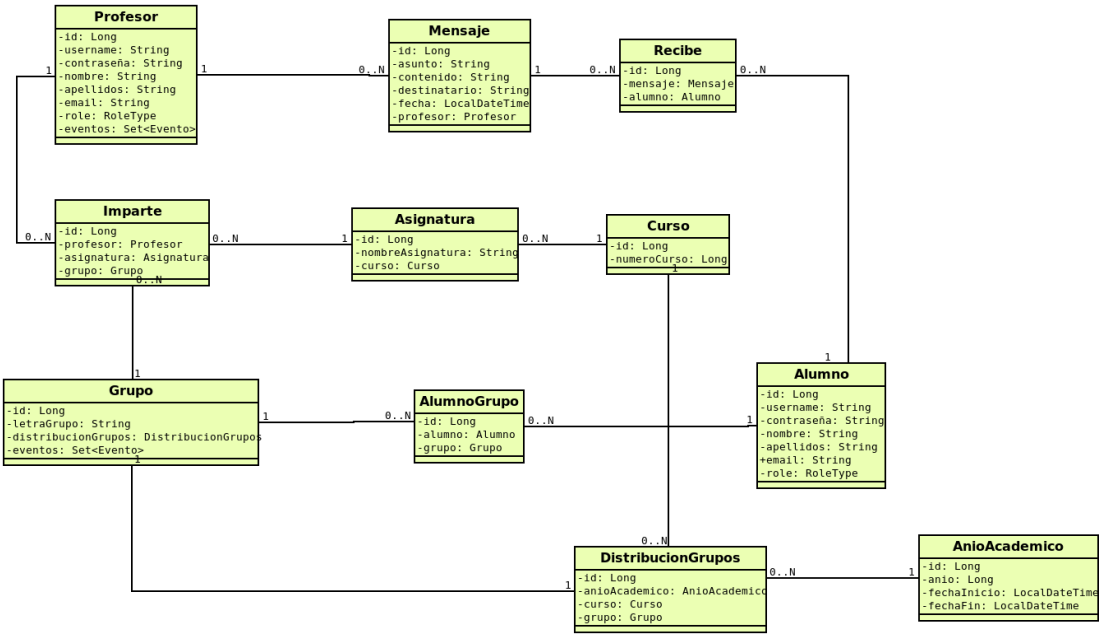


Figura 8.1: Entidades de los subsistemas Usuarios y Mensajes

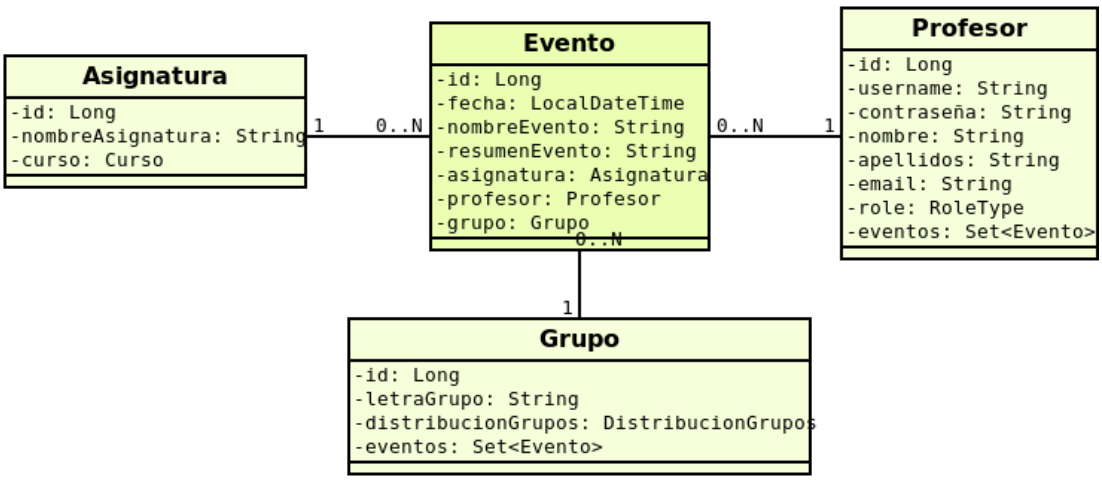


Figura 8.2: Entidades del subsistema Calendario

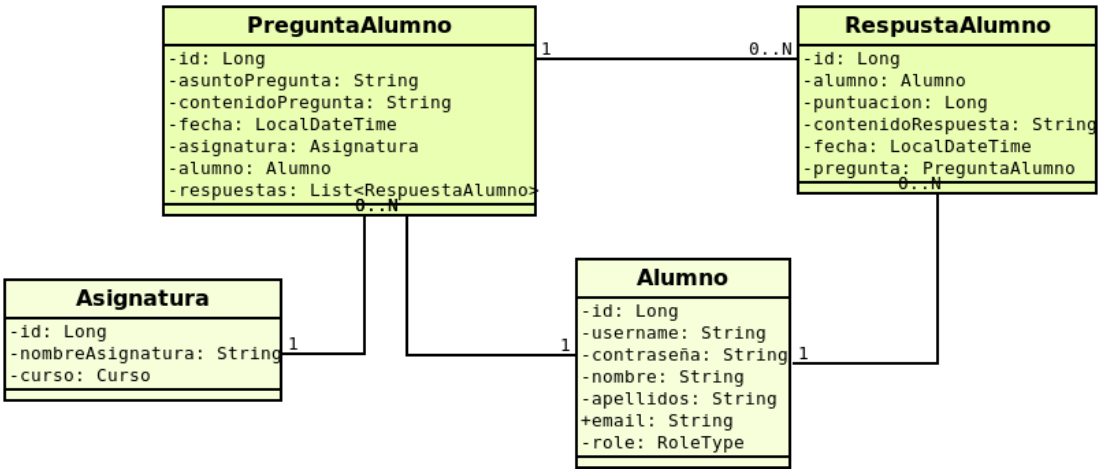


Figura 8.3: Entidades del subsistema Foro

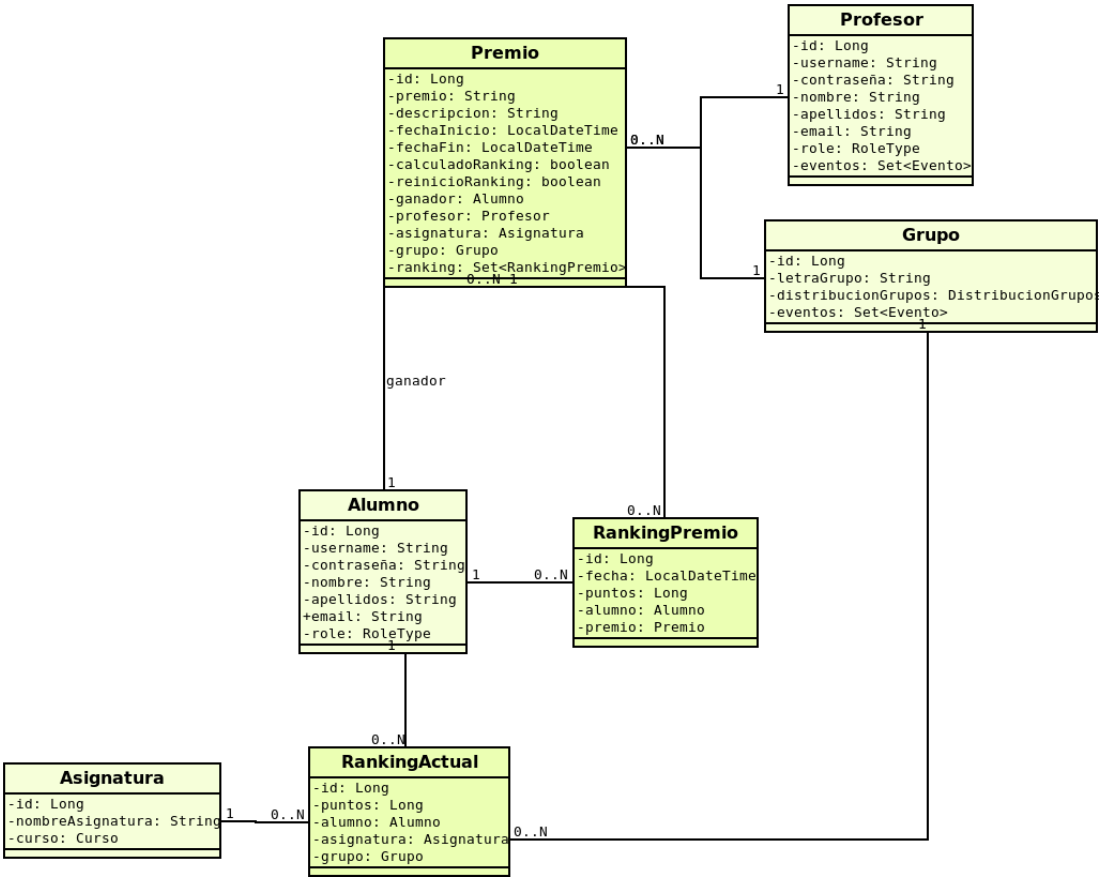


Figura 8.4: Entidades del subsistema Ranking

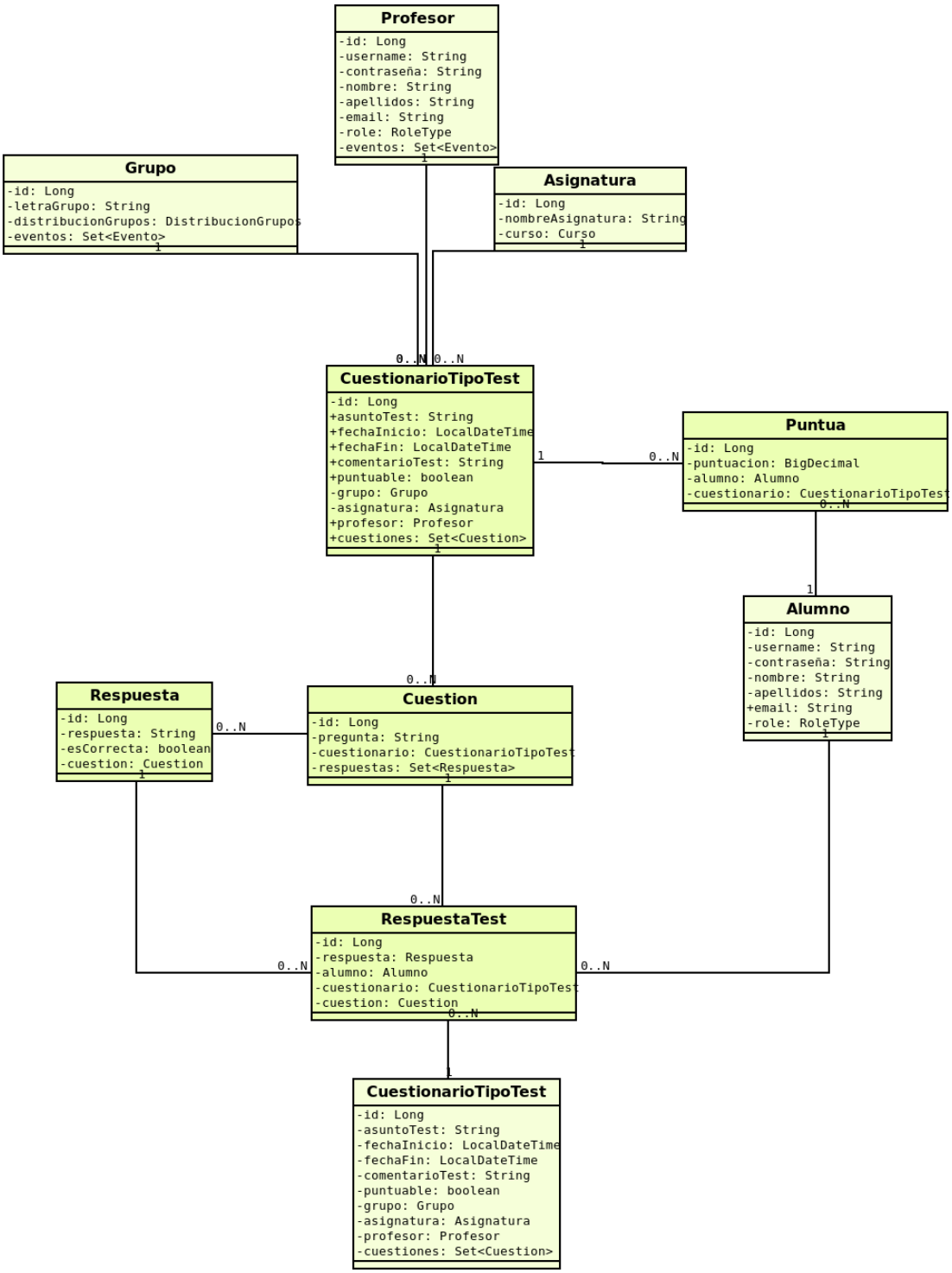


Figura 8.5: Entidades del subsistema Tests

A continuación se definen cada una de las entidades y su propósito:

- Profesor: Contiene los datos de un profesor.
- Mensaje: Contiene los datos de un mensaje enviado por un profesor a un alumno o a uno de los grupos en los que imparte o ha impartido clase.
- Recibe: Entidad intermedia que relaciona Mensaje y Alumno (quien recibe el mensaje).
- Imparte: Entidad intermedia que representa que un Profesor imparte una Asignatura en un Grupo determinado.
- Asignatura: Contiene los datos de una asignatura.
- Curso: Contiene los datos de un curso.
- Grupo: Contiene los datos de un grupo y este es único en todo el sistema.
- AlumnoGrupo: Entidad intermedia que representa que un Alumno pertenece a un Grupo determinado.
- Alumno: Contiene los datos de un alumno.
- DistribucionGrupos: Entidad intermedia que representa que un Grupo está asociado a un determinado Curso y a un determinado AnioAcademico.
- AnioAcademico: Contiene los datos de un año académico.
- Evento: Contiene los datos de un evento del calendario.
- PreguntaAlumno: Contiene los datos de una pregunta realizada por un alumno en el foro.
- RespuestaAlumno: Contiene los datos de una respuesta de un alumno a una pregunta del foro.
- Premio: Contiene la información de un premio.
- RankingPremio: Contiene el ranking de alumnos final para un premio determinado. Se calcula automáticamente una vez ha finalizado la fecha de expiración del premio.
- RankingActual: Contiene los puntos de un alumno con respecto a un determinado grupo y asignatura por las respuestas que ha realizado en el foro y han sido valoradas positivamente por el profesor.
- CuestionarioTipoTest: Contiene la información de un cuestionario tipo test elaborado por un profesor.

- Cuestion: Contiene la información sobre una cuestión de un determinado cuestionario tipo Test.
- Respuesta: Contiene la información sobre una respuesta posible (correcta o no) correspondiente a una determinada cuestión de un determinado cuestionario.
- RespuestaTest: Representa la respuesta marcada como correcta por un alumno a una determinada cuestión de un determinado cuestionario tipo Test.
- Puntua: Entidad que almacena la puntuación alcanzada por un alumno en un determinado cuestionario. Gracias a esta entidad, cada vez que se quieran conocer las notas de los alumnos con respecto a un determinado cuestionario, solo será necesita consultar la tabla Puntúa. Así se evita la generación de cálculos repetitivos e innecesarios, no teniendo así que calcular la nota cada vez que se necesite conocer su valor.

8.3.2 Capa de acceso a datos

En las figuras 8.6 (página 36), 8.7 (página 37) y 8.8 (página 38) se muestran los DAOs y las operaciones que estos incluyen a mayores de las CRUD, que todos ellos ofrecen por extender de la clase PagingAndSortingRepository. Además también se incluyen las razones de utilización de las diferentes operaciones.

AlumnoDao	Optional<Alumno> findByUserName(String userName)
	Utilizado por la operación login para localizar el usuario que se loguea en base de datos
AlumnoGrupoDao	List<AlumnoGrupo> findByGrupoId(Long grupoId)
	Utilizado para encontrar los alumnos de un grupo determinado sin paginar
AlumnoGrupoDao	Slice<AlumnoGrupo> findByGrupoIdOrderByAlumnoApellidos (Long grupoId,Pageable pageable)
	Utilizado para encontrar los alumnos de un grupo determinado ordenados y paginados
AlumnoGrupoDao	List<AlumnoGrupo> findByAlumnoId (Long alumnoId)
	Utilizado para encontrar los grupos a los que ha pertenecido o sigue perteneciendo un alumno
AlumnoGrupoDao	AlumnoGrupo findByAlumnoIdAndGrupoDistribucionGruposAnioAcademicoId(Long alumnoId, Long anioId)
	Utilizado para encontrar el grupo al que pertenece o ha pertenecido un alumno en un determinado año
AnioAcademicoDao	List<AnioAcademico> findAllByOrderByFechaInicioDesc ()
	Utilizado para encontrar la lista de años académicos registrados en el sistema ordenados. Así se puede obtener por ejemplo el año más reciente registrado en el sistema
AsignaturaDao	
CuestionDao	
CuestionarioTipoTestDao	Slice<CuestionarioTipoTest> findByProfesorIdOrderByFechaInicio(Long profesorId, Pageable pageable)
	Utilizado para encontrar los cuestionarios formulados por un profesor paginados y ordenados por fecha de inicio descendente
CuestionarioTipoTestDao	List<CuestionarioTipoTest> findByGrupoId(Long grupoId)
	Utilizado para encontrar los cuestionarios pertenecientes a un grupo determinado
CursoDao	
CustomizedCuestionarioDao	Slice<CuestionarioTipoTest> findCuestionariosOfGrupos(List <Long> grupoIds, int page, int size)
	Utilizado para encontrar los cuestionarios correspondientes a los grupos introducidos por parámetro en la lista grupoIds
DistribucionGruposDao	
EventoDao	
GrupoDao	

Figura 8.6: Daos

ImparteDao	List<Imparte> findByProfesorIdOrderByGrupoDistribucionGruposAnioAcademicoAnioDescGrupoDistribucionGruposCursoNumeroCursoAscGrupoLetraGrupoAsc(Long profesorId)
	Utilizado para encontrar los grupos de un determinado profesor ordenados
ImparteDao	List <Imparte> findByGrupoId(Long grupoId)
	Utilizado para encontrar las asignaturas impartidas en un grupo determinado
MensajeDao	Slice<Mensaje> findByProfesorIdOrderByFechaDesc(Long profesorId,Pageable pageable)
	Utilizado para encontrar mensajes recibidos por un determinado alumno paginados y ordenados por fecha de manera descendente
PreguntaDao	Slice<PreguntaAlumno> findByAsignaturaIdOrderByFechaDesc(Long asignaturaId, Pageable pageable)
	Utilizado para encontrar las preguntas realizadas en el foro de alumnos sobre una determinada asignatura paginadas y ordenadas
PremioDao	findByProfesorIdOrderByFechaInicioDesc(Long profesorId)
	Utilizado para encontrar los premios de un profesor ordenados por fecha de inicio de manera descendente
PremioDao	List<Premio> findByGrupoIdOrderByFechaInicioDesc(Long grupoId)
	Utilizado para encontrar los premios correspondientes a un grupo determinado ordenados por fecha de inicio de manera descendiente
PremioDao	List<Premio> findByCalculadoRanking(boolean calculadoRanking)
	Utilizado para encontrar los premios para los cuales no se ha calculado el ranking de premios todavía
ProfesorDao	Optional<Profesor> findByUserName(String userName)
	Utilizado por la operación login para localizar el usuario que se loguea en base de datos
PuntuaDao	Optional<Puntua> findByAlumnoIdAndCuestionarioId(Long alumnoId, Long cuestionarioId)
	Utilizado para encontrar la nota que ha sacado un alumno en un determinado cuestionario
PuntuaDao	List <Puntua> findByCuestionarioIdOrderByAlumnoApellidosAsc(Long cuestionarioId)
	Utilizado para encontrar las notas de los alumnos con respecto a un determinado cuestionario
RankingActualDao	List <RankingActual> findByGrupoIdAndAsignaturaIdOrderByPuntosDesc(Long grupoId, Long asignaturaId)
	Utilizado para encontrar el ranking de alumnos con sus respectivas notas en relación a un grupo y asignatura determinados

Figura 8.7: Daos

RankingPremioDao	List <RankingPremio> findByPremioIdOrderByPuntosDesc(Long premioId)
	Utilizado para encontrar el ranking de alumnos con respecto a un premio determinado ordenados por puntuación de manera descendente
RankingPremioDao	RankingPremio findByPremioIdAndAlumnoId(Long premioId, Long alumnoId)
	Utilizado para encontrar la entidad con los puntos que un determinado alumno ha conseguido con respecto a un premio determinado
RecibeDao	Slice<Recibe> findByAlumnoIdOrderByMensajeFechaDesc(Long alumnoId,Pageable pageable)
	Utilizado para encontrar los mensajes recibidos por un determinado alumno paginados y ordenados por fecha de manera descendente (los más recientes primero)
RecibeDao	Recibe findByAlumnoIdAndMensajeId(Long alumnoId,Long mensajeId)
	Utilizado para encontrar un mensaje determinado enviado a un alumno y poder asegurar que ha sido enviado a ese alumno en cuestión
RespuestaAlumnoDao	List<RespuestaAlumno> findByPreguntaIdOrderByPuntuacionDesc(Long preguntaId)
	Utilizado para encontrar las respuestas a una determinada pregunta del foro ordenadas por puntuación de manera descendente
RespuestaAlumnoDao	List<RespuestaAlumno> findByPreguntaIdOrderByFechaDesc(Long preguntaId)
	Utilizado para encontrar las respuestas a una determinada pregunta del foro ordenadas por fecha de manera descendente
RespuestaDao	
RespuestaTestDao	Long countByRespuestaId(Long respuestaId)
	Utilizado para contar el número de alumnos que marcaron una determinada respuesta al realizar el cuestionario (con motivos estadísticos)
RespuestaTestDao	Long countByCuestionId(Long cuestionId)
	Utilizado para contar el número de alumnos que contestaron a una pregunta (con motivos estadísticos)

Figura 8.8: Daos

8.3.3 Capa lógica de negocio

Las operaciones que expone la capa modelo han sido divididas en clases distintas (servicios distintos) en función del subsistema al que dan soporte.

- Las operaciones relacionadas con el login, cambiar contraseña y modificar perfil de alumnos y profesores se encuentran en el servicio UsuarioService.

- Las operaciones relacionadas con los mensajes se encuentran en MensajeService, incluido encontrar grupos de un profesor y encontrar alumnos de un grupo. Esto es así porque para enviar un mensajes a un grupo el profesor debe poder conocer los grupos en los que imparte o ha impartido clase, y para enviar un mensaje a un alumno, el profesor ha de poder conocer los alumnos que pertenecen a un determinado grupo para poder así mandarle un mensaje a alguno de ellos.

- Las operaciones relacionadas con los eventos (crearlos, buscarlos, eliminarlos) se encuentran en CalendarioService.

- Las operaciones relacionadas con el foro se encuentran en el ForoService, incluido encontrar asignaturas de un alumno. Esto es así porque para la búsqueda de preguntas en el foro, el alumno tiene a su disposición un desplegable en el cual puede seleccionar la asignatura que desee de entre todas las asignaturas que cursa o ha cursado. Se ha decidido que los alumnos puedan seguir accediendo a los foros de las asignaturas que ya han cursado y aprobado, puesto que se considera importante que puedan seguir aprendiendo acerca de cada una de ellas. Además, estos alumnos pueden ser de gran ayuda en la resolución de las dudas relacionadas con dichas asignaturas. En el ForoService también se incluye la operación encontrar asignaturas de un profesor. En este caso el profesor solo recibe las asignaturas que imparte en el curso actual. Esto es así porque se ha estimado oportuno que cada profesor supervise solo los foros de las asignaturas que imparte en la actualidad. Al final la labor del profesor en el foro es supervisarlo y puntuar las respuestas. Las asignaturas que haya impartido en el pasado ya son impartidas por otros profesores en el presente, y serán estos últimos los que se ocupen de supervisar los foros de dichas asignaturas.

- Las operaciones relacionadas con los premios (crearlos, consultarlos, eliminarlos) y con el ranking (consultar ranking de premios y ranking de asignaturas) se encuentran en el RankingService. También se incluye en este servicio la operación puntuar respuesta del foro, ya que lo que ocurra en esta operación repercute directamente sobre el ranking.

- Las operaciones relacionados con los cuestionarios (crearlos, buscarlos, responder a un

cuestionario, ver estadísticas de un cuestionario y ver notas de los alumnos) se encuentran en el servicio TestService puesto que dan soporte al subsistema de tests.

En las figuras 8.9 (página 40), 8.10 (página 40), 8.11 (página 40), 8.12 (página 40), 8.13 (página 41) y 8.14 (página 41) se muestran las interfaces de los servicios de la capa modelo.

UsuarioService

```
+loginProfesor(userName:String,contraseña:String): Profesor
+loginAlumno(userName:String,contraseña:String): Alumno
+actualizarProfesor(id:Long,email:String): Profesor
+actualizarAlumno(id:Long,email:String): Alumno
+cambiarContraseñaAlumno(id:Long,contraseñaAntigua:String,contraseñaNueva:String): void
+cambiarContraseñaProfesor(id:Long,contraseñaAntigua:String,contraseñaNueva:String): void
```

Figura 8.9: Servicio de Usuarios

MensajeService

```
+encontrarGruposDeUnProfesor(profesorId:Long): List<Imparte>
+encontrarAlumnosDeUnGrupo(profesorId:Long,grupoId:Long,page:int,size:int): Block<Alumno>
+encontrarMensajesDeUnAlumno(alumnoId:Long,page:Long,size:Long): Block<Mensaje>
+verDetallesMnsajeAlumno(alumnoId:Long,mensajeId): Mensaje
+verDetallesMnsajeProfesor(profesorId:Long,mensajeId:Long): Mensaje
+encontrarMensajesDeUnProfesor(profesorId:Long,page:Long,size:Long): Block<Mensaje>
+enviarMensajeAlumno(asunto:String,contenido:String,profesorId:Long,alumnoId:Long,page:int,size:int): Block<Mensaje>
+enviarMensajeAGrupo(asunto:String,contenido:String,profesorId:Long,grupoId:Long,page:int,size:int): Block<Mensaje>
```

Figura 8.10: Servicio de Mensajes

CalendarioService

```
+encontrarEventosAlumno(alumnoId:Long,mes:int,anio:int): List<Evento>
+encontrarEventosProfesor(profesorId:Long,mes:int,anio:int): List<Evento>
+crearEvento(profesorId:Long,grupoId:Long,asignaturaId:Long,nombreEvento:String,resumenEvento:String,
             fecha:LocalDateTime): List<Evento>
+eliminarEvento(profesorId:Long,calendarioId:Long): List<Evento>
+encontrarEventosAlumno(alumnoId:Long,mes:int,anio:int,dia:int): List<Evento>
+encontrarEventosProfesor(profesorId:Long,mes:int,anio:int,dia:int): List<Evento>
```

Figura 8.11: Servicio del Calendario

ForoService

```
+encontrarAsignaturasDeUnAlumno(alumnoId:Long): List<Asignatura>
+crearPregunta(alumnoId:Long,asignaturaId:Long,contenidoPregunta:String,asuntoPregunta:String): Block<PreguntaAlumno>
+crearRespuesta(alumnoId:Long,preguntaId:Long,contenidoRespuesta:String): PreguntaAlumno
+encontrarRespuestasAlumno(preguntaId:Long): PreguntaAlumno
+encontrarRespuestasProfesor(preguntaId:Long): PreguntaAlumno
+encontrarPreguntasDeUnaAsignaturaAlumno(alumnoId:Long,asignaturaId:Long,page:int,size:int): Block<PreguntaAlumno>
+encontrarPreguntasDeUnaAsignaturaProfesor(profesorId:Long,asignaturaId:Long,page:int,size:int): Block<PreguntaAlumno>
+encontrarAsignaturasDeUnProfesor(profesorId:Long): List<Asignatura>
```

Figura 8.12: Servicio de Foro

RankingService

```
+crearPremio(premio:String,descripcion:String,profesorId:Long,asignaturaId:Long,grupoId:Long,fechaFin:LocalDateTime,
             reinicioRanking:boolean): List<Premio>
+eliminarPremio(profesorId:Long,premioId:Long): List<Premio>
+encontrarPremiosProfesor(profesorId:Long): List<Premio>
+encontrarPremiosAlumno(alumnoId:Long): List<Premio>
+encontrarRankingAlumnosDeUnPremio(alumnoId:Long,premioId:Long): List <RankingPremio>
+encontrarRankingAlumnosDeUnaAsignatura(alumnoId:Long,asignaturaId:Long): List <RankingActual>
+puntuarRespuesta(respuestaId:Long,puntuacion:Long): PreguntaAlumno
+encontrarAsignaturasAlumnoAnioActual(alumnoId:Long): List<Asignatura>
+findPremio(premioId:Long): Premio
```

Figura 8.13: Servicio de Ranking

TestService

```
+crearCuestionario(grupoId:Long,asignaturaId:Long,profesorId:Long,asuntoTest:String,fechaFin:LocalDateTime,
                   comentarioTest:String,puntuable:boolean,cuestiones:List <Cuestion>,page:int,size:int): Block<CuestionarioTipoTest>
+encontrarCuestionariosProfesor(profesorId:Long,page:int,size:int): Block<CuestionarioTipoTest>
+verDetallesCuestionario(cuestionarioId:Long): CuestionarioTipoTest
+encontrarCuestionariosAlumno(alumnoId:Long,page:int,size:int): Block<CuestionarioTipoTestAlumnoDto>
+responderCuestionario(cuestionarioId:Long,alumnoId:Long,cuestiones:List <Cuestion>): BigDecimal
+encontrarPuntuacionesAlumnos(profesorId:Long,cuestionarioId:Long): List <Puntua>
+encontrarEstadisticasCuestionario(cuestionarioId:Long): CuestionarioConEstadisticasDto
```

Figura 8.14: Servicio de Tests

En la figura 8.15 (página 41) se muestra el diagrama completo de diseño de uno de los servicios de la capa modelo, el MensajeService.

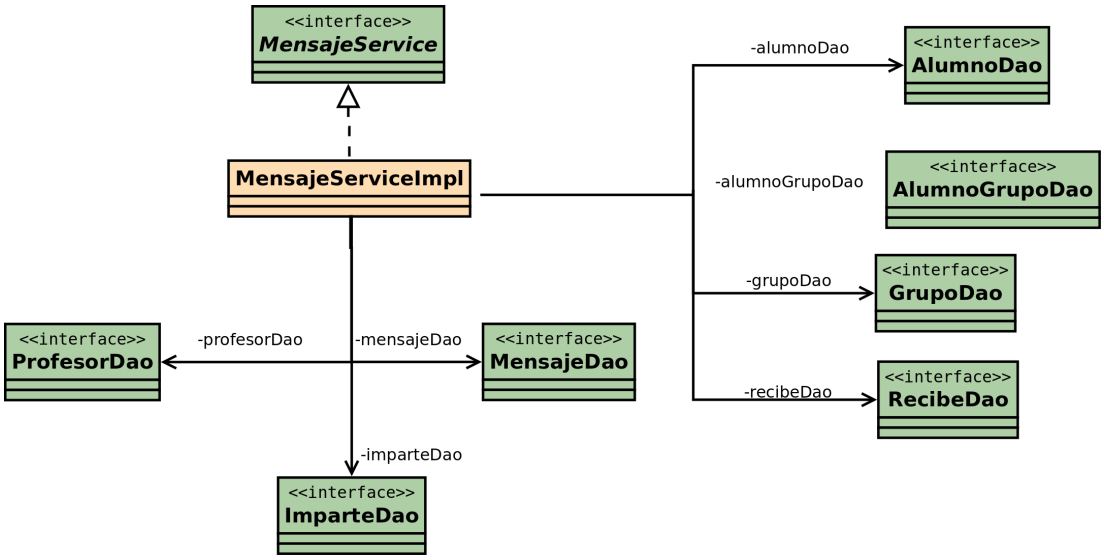


Figura 8.15: Diseño del servicio de Mensajes

8.4 Servicios REST

Los servicios REST son los encargados de recibir las peticiones HTTP y delegarlas en los servicios de la Capa Modelo. Las operaciones que ofrecen estos servicios son repartidas en diferentes clases (controladores) según la funcionalidad que ofrecen. En este caso cada

uno de los controladores definidos se ocupa de mapear las peticiones que recaen sobre uno de los servicios de la capa modelo, es decir: el UsuarioController se ocupa de mapear todas las operaciones que el UsuarioService ofrece, el MensajeController las del MensajeService, el CalendarioController las del CalendarioService, el ForoController las del ForoService, el RankingController las del RankingService y el TestController las del TestService.

Cada operación del controlador tiene una ruta asociada a la cual tiene que responder en caso de llamada. Para las comunicaciones entre backend y frontend se utilizan los DTOs y no las entidades de la capa modelo. Tanto los parámetros de entrada como los datos de salida de una determinada operación del controlador deben ser encapsulados en objetos. Por ejemplo, en la operación crearEvento del controlador, se utiliza el DTO CreateEventoParams para encapsular los parámetros que llegan en el cuerpo de la petición, y se utiliza el DTO EventoDto, para encapsular los datos de salida de la petición. Como se puede apreciar en los diagramas UML de las clases, los atributos de EventoDto y Evento no son iguales. Podemos observar, por ejemplo, que el atributo fecha de tipo LocalDateTime es representado en el DTO por un Long, representando así la fecha en milisegundos. Si se hubiese modelado como un LocalDateTime, la fecha aparecería como un String en el JSON en formato ISO-8601, lo que dificultaría su conversión en el frontend.

Evento
id: Long
fecha: LocalDateTime
nombreEvento: String
resumenEvento: String
asignatura: Asignatura
profesor: Profesor
grupo: Grupo

EventoDto
id: Long
fecha: Long
nombreEvento: String
resumenEvento: String
nombreAsignatura: String
nombreProfesor: String

Tabla 8.1: Diferencia entre los atributos de una clase del modelo y su correspondiente DTO

En la figura 8.16 (página 43) se muestra la interfaz operacional que ofrece uno de los servicios Rest, el MensajeController.

Recurso	Método	Entrada	Salida
/mensajes/sendMensajeToGrupo	Post	userId [jwt], sendMensajeToGrupoParams, page [param]	BlockDto <MensajeSummaryDto> [body]
/mensajes/sendMensajeToAlumno	Post	userId [jwt], sendMensajeToAlumnoParams, page [param]	BlockDto<MensajeSummaryDto> [body]
/mensajes/mensajesAlumno	Get	userId [jwt], page [param]	BlockDto<MensajeSummaryDto> [body]
/mensajes/mensajesProfesor	Get	userId [jwt], page [param]	BlockDto<MensajeSummaryDto> [body]
/mensajes/gruposProfesor	Get	userId[jwt]	List<ChooseGrupoDto> [body]
/mensajes/alumnosGrupo	Get	userId[jwt], grupoId [param], page [param]	BlockDto<UserDto> [body]
/mensajes/mensajeAlumno/{mensajeId}	Get	userId[jwt], mensajeId [PathVariable]	MensajeDto [body]
/mensajes/mensajeProfesor/{mensajeId}	Get	userId[jwt], mensajeId [PathVariable]	MensajeDto [body]

Figura 8.16: Interfaz operacional del MnesajeController

8.5 Diseño del Frontend

El cliente o frontend se encarga de la parte visual de la aplicación y de la interacción del usuario con esta. El cliente también se encarga de lanzar las peticiones necesarias al backend y recibir las respuestas que éste le manda para poder así mostrarlas como resultado por pantalla.

Para explicar el diseño del componente frontend, dividimos el frontend en las siguientes partes:

- Capa de acceso a servicios
- Capa de interfaz de usuario

8.5.1 Capa de acceso a servicios

Dentro de src/backend existe un fichero javascript por cada controlador REST existente en el backend. Define tantas funciones como casos de uso expone el controlador REST. La capa de acceso a servicios es la encargada de enviar las peticiones que se generan en el frontend a la aplicación backend. Después de su envío la capa de acceso a los servicios se queda a la espera de las respuestas del servidor .

Las peticiones a realizar al backend se estructuran de la siguiente forma:

- En usuarioService se realizan las peticiones dirigidas al UsuarioController del backend.
- En mensajeService se realizan las peticiones dirigidas al MensajeController del backend.
- En calendarioService se realizan las peticiones dirigidas al CalendarioController del backend.

- En foroService se realizan las peticiones dirigidas al ForoController del backend.
- En rankingService se realizan las peticiones dirigidas al RankingController del backend.
- En cuestionarioService se realizan las peticiones dirigidas al TestController del backend.

Las peticiones que se lanzan desde esta capa utilizan el protocolo HTTP para las comunicaciones, encapsulando los datos enviados en las peticiones POST y PUT en el formato JSON.

8.5.2 Capa de interfaz de usuario

Para facilitar la gestión del código, la capa de interfaz de usuario está organizada en módulos. Cada uno de estos módulos contiene el código de la interfaz de usuario correspondiente a los casos de uso expuestos por uno de los servicios Rest del backend. Los módulos existentes dentro de la carpeta modules del frontend son usuario, mensaje, calendario, foro, ranking y cuestionario, que contienen el código de la IU correspondiente a los casos de uso expuestos por los servicios Rest UsuarioController, MensajeController, CalendarioController, ForoController, RankingController y TestController respectivamente.

Además la carpeta modules del frontend contiene otros 2 módulos:

- app (layout de la interfaz de usuario)
- common, que contiene componentes reusables entre módulos, como es el caso del componente Pager, que muestra por pantalla los botones de paginación atrás y adelante.

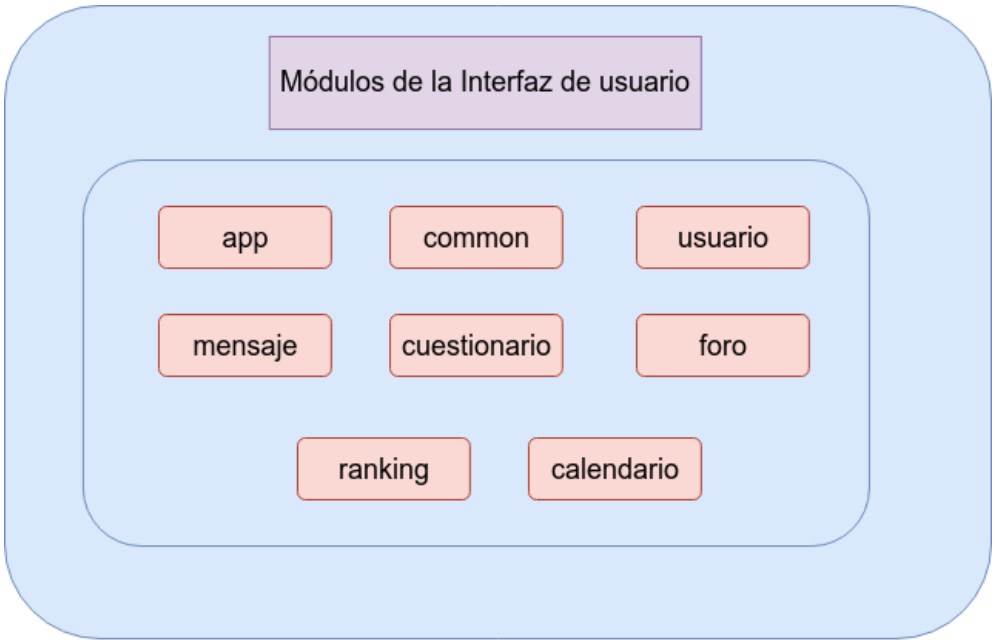


Figura 8.17: Módulos de la Interfaz de usuario

Cada módulo está compuesto por diferentes componentes. El módulo App está compuesto por los componentes App (componente padre de la aplicación), Header, Body y Home.

Para el diseño de los componentes más relevantes de la interfaz se han elaborado maquetas. En estas maquetas se posicionan los componentes que irán dentro de una determinada pantalla. En la figura 8.18 (página 45) se puede visualizar la maqueta que ha sido diseñada para el componente App.

En esta maqueta se puede apreciar que el componente App contiene otros componentes en su interior, estos son Header y Body. A su vez Body contiene otros componentes que se renderizan en función de la URL del navegador. En este caso se muestra la primera pantalla que un profesor ve nada mas loguearse.

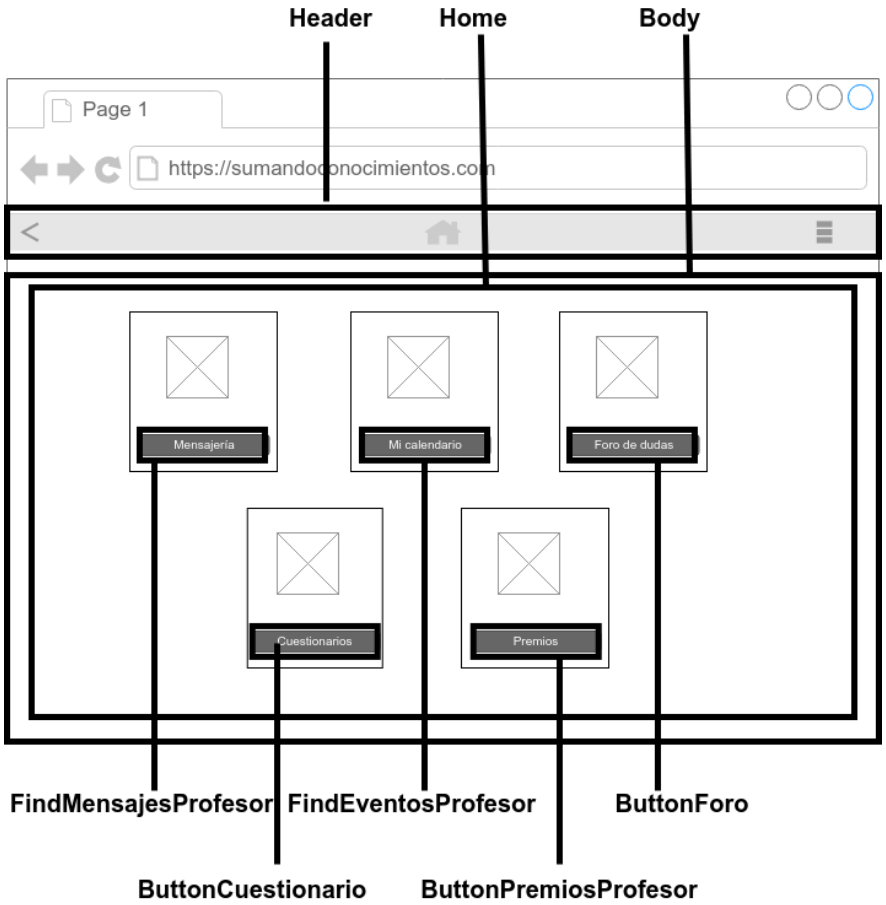


Figura 8.18: Componentes del módulo App

En la figura 8.19 (página 46) se muestra otra maqueta, correspondiente a una interacción del usuario profesor con la interfaz (cuando el profesor da clic sobre el botón Mensajería)

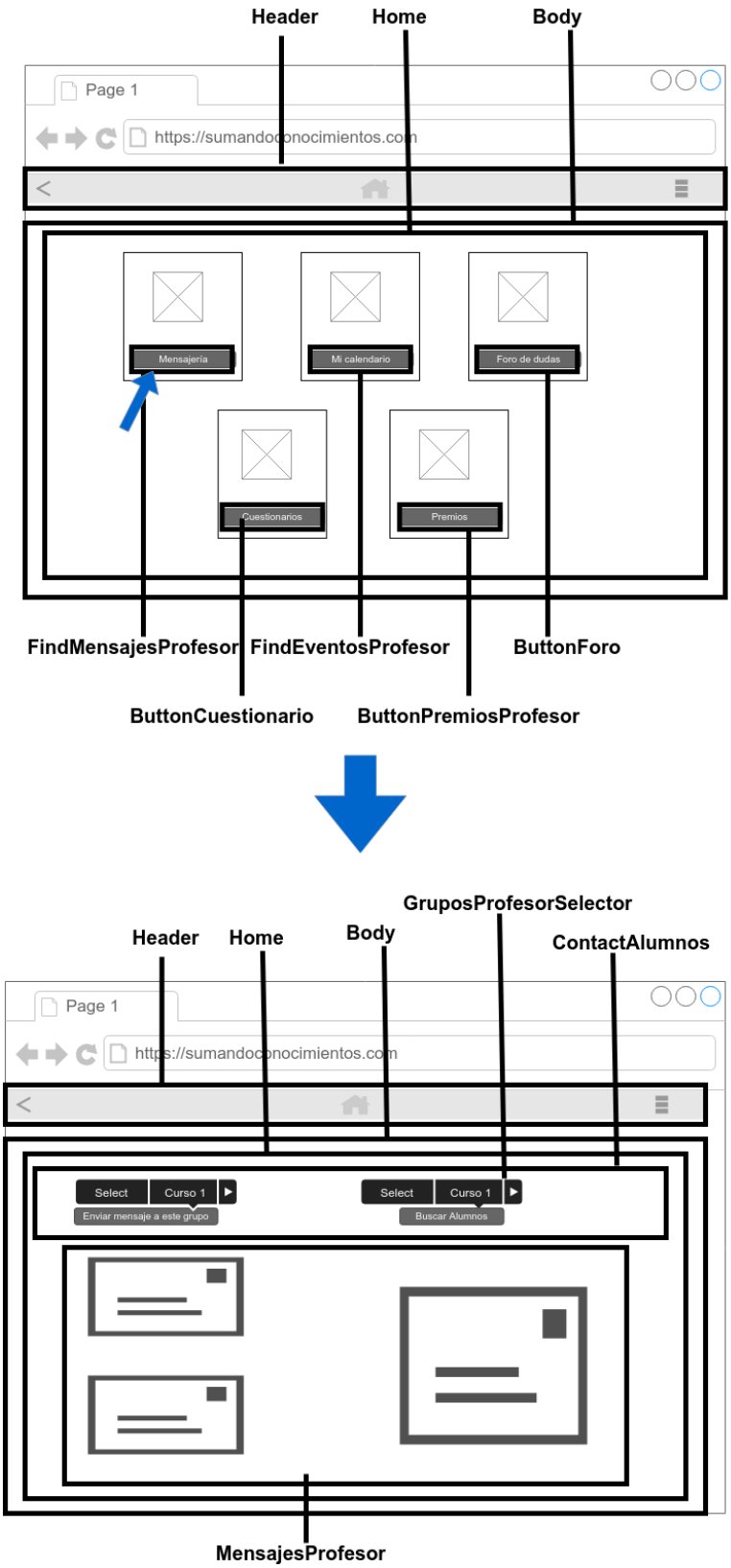


Figura 8.19: Interacción del usuario con la interfaz (dar clic sobre el botón Mensajería)

Gracias a la creación de las maquetas ha resultado mucho más sencillo y rápido crear los componentes de la interfaz.

También han resultado de mucha utilidad los diagramas secuencia realizados. Estos ilustran los intercambios de información entre el backend y el frontend, con especial interés en el flujo de información entre los distintos componentes del frontend.

A continuación se muestra un diagrama secuencia, figura 8.20 (página 48), que nos sirve para ver de forma simplificada el intercambio de información entre los componentes del frontend y del backend para una de las interacciones del usuario. Esta interacción del usuario se corresponde con la mostrada en la figura anterior (dar clic sobre el botón Mensajería) 8.19 (página 46)

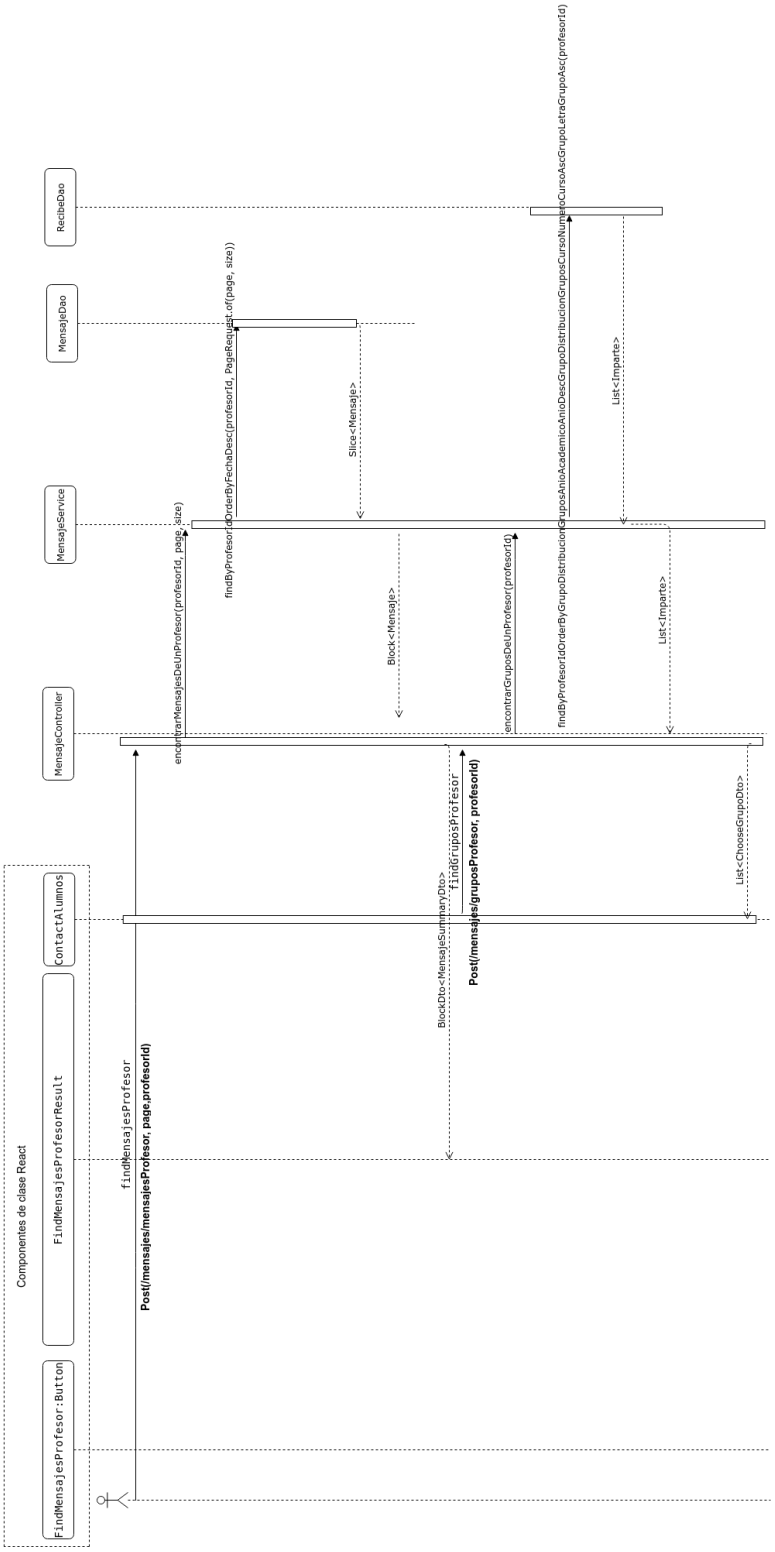


Figura 8.20: Diagrama secuencia de las comunicaciones entre backend y frontend al hacer clic sobre el botón Mensajería

Capítulo 9

Implementación

La fase de implementación ha sido con diferencia la fase que más tiempo ha consumido. En este capítulo se muestran los detalles más relevantes de la implementación de la aplicación, tanto del backend (implementación de las entidades, los DAOS, los servicios de la capa modelo, los controladores y la autenticación y control de acceso) como del frontend (gestión del estado, enrutamiento y control de acceso).

9.1 Implementación del backend

En esta sección se verán los detalles de implementación más relevantes de la aplicación backend.

9.1.1 Implementación de las entidades

Para la implementación de las entidades se usa JPA/Hibernate, que nos permite mapear las entidades de nuestra base de datos a entidades del dominio de la aplicación y viceversa. Para hacerlo posible nos proporciona una serie de anotaciones que se colocan sobre las entidades del dominio.

Cada una de las entidades del dominio se anota con @Entity, para que pueda así Hibernate reconocerlas. Hibernate mapea por defecto cada entidad a una tabla con el mismo nombre que la clase de la entidad. Lo mismo ocurre con los atributos de la entidad, que se mapean a columnas con el mismo nombre. Si no fuera así, existen las etiquetas @Table y @Column para indicar el nombre de la tabla o la columna a la que se quiere mapear.

El atributo clave correspondiente con la columna de la clave primaria en BBDD se ha anotado con @Id y se ha usado conjuntamente con la etiqueta @GenerateValue para que la clave se genere automáticamente.

Para modelar las relaciones entre entidades también se hace uso de las anotaciones, colocadas esta vez sobre los métodos get. Como se puede ver a continuación, se anota con @One-

ToMany cuando se navega de la entidad con cardinalidad 1 a la entidad con cardinalidad muchos y @ManyToOne cuando se hace al revés. También se puede ver que se usa optional = false cuando se quiere indicar que el método no puede devolver null.

Para evitar que el mapeador objeto-relacional recupere entidades relacionadas con una entidad dada cada vez que ésta se recupera, se indica sobre el método get que las recupera, la política LAZY, haciendo uso del parámetro fetch. Esto se puede ver a continuación, donde vemos que el método getCuestiones no recuperará las cuestiones cuando se recupera una instancia de la clase CuestionarioTipoTest.

Mediante la política Lazy las instancias de las entidades relacionadas no se recuperan en un primer momento, lo único que se recupera de esas instancias son los proxies, que inicialmente solo contienen la clave primaria de la entidad.

También hay en casos en los que la política Lazy no hará que se recuperen más rápido los datos que se requieren, y es cuando la mayoría de veces que se inicializa el proxy hay que recuperar igualmente los atributos faltantes después. Para evitar esto se usa la política EAGER.

Como podemos ver, hemos indicado que el grupo , la asignatura y el profesor relacionados con la instancia de la clase CuestionarioTipoTest, se recuperen siempre. Esto se debe a que las operaciones que hacen uso de la clase CuestionarioTipoTest casi siempre requieren estas entidades. Así evitamos llamadas innecesarias a BBDD para recuperar los atributos faltantes.

```
1
2  @Entity
3  public class CuestionarioTipoTest {
4    private Long id;
5    ...
6    private Grupo grupo;
7    private Asignatura asignatura;
8    private Profesor profesor;
9    private Set<Cuestion> cuestiones=new HashSet<>();
10
11     ...
12
13   @Id
14   @GeneratedValue(strategy = GenerationType.IDENTITY)
15   public Long getId() {
16     return id;
17   }
18
19
20
21   @OneToMany(mappedBy="cuestionario", fetch=FetchType.LAZY)
22   public Set<Cuestion> getCuestiones() {
```

```
23      return cuestiones;
24    }
25
26    @ManyToOne(optional=false, fetch=FetchType.EAGER)
27    @JoinColumn(name= "grupoId")
28    public Grupo getGrupo() {
29      return grupo;
30    }
31
32
33    @ManyToOne(optional=false, fetch=FetchType.EAGER)
34    @JoinColumn(name= "asignaturaId")
35    public Asignatura getAsignatura() {
36      return asignatura;
37    }
38
39    @ManyToOne(optional=false, fetch=FetchType.EAGER)
40    @JoinColumn(name= "profesorId")
41    public Profesor getProfesor() {
42      return profesor;
43    }
44    ...
45  }
46
47  }
```

9.1.2 Implementación de los DAOs

Para el desarrollo más ágil de los DAOs se ha utilizado Spring Data para JPA. Las interfaces de los DAOs no revelan ningún detalle de implementación. En la mayor parte de los casos Spring Data implementa automáticamente los métodos de los DAOs en tiempo de ejecución. Todos los DAOs definidos en la aplicación extienden de PagingAndSortingRepository. Gracias a esto las operaciones CRUD ya están definidas. Muchas de las operaciones de búsqueda que necesitamos se pueden realizar mediante convenciones de nombrado como las siguientes.

```
1
2  public interface CuestionarioTipoTestDao extends
       PagingAndSortingRepository<CuestionarioTipoTest, Long>,
       CustomizedCuestionarioDao{
3
4    Slice<CuestionarioTipoTest>
       findByProfesorIdOrderByFechaInicio(Long profesorId, Pageable
       pageable);
5
6    List<CuestionarioTipoTest> findByGrupoId(Long grupoId);
```

```
}
```

Hay veces en que las convenciones de nombrado no pueden expresar la condición de búsqueda. Para esos casos se puede usar Spring Data JPQL (Java Persistence Query Language).

La sintaxis es similar a SQL pero se usan nombres de entidades y atributos en lugar de tablas y columnas. La consulta ”encontrar los cuestionarios de varios grupos” se ha implementado usando JPQL.

Para construir esta operación hay que definirla en una interfaz adicional, en este caso CustomizedCuestionarioDao. Hay que implementarla en CustomizedCuestionarioDaoImpl, usando la API de JPA para lanzar la consulta y usando JPQL para implementarla. Por último hay que hacer que la interfaz del Dao (CuestionarioTipoTestDao) extienda de la interfaz adicional generada (CustomizedCuestionarioDao).

A continuación se muestra la consulta realizada en el lenguaje JPQL.

```
String queryString = "SELECT c FROM CuestionarioTipoTest c";

  if (grupoIds.size() > 0) {
    queryString += " WHERE c.grupo.id = :item0";
  }


  if (grupoIds.size() > 1) {
    queryString += " OR ";


    for (int i = 1; i<grupoIds.size()-1; i++) {
      queryString += "c.grupo.id = :item" + i + " OR ";
    }

    queryString += "c.grupo.id = :item" + (grupoIds.size()-1);

  }

  queryString += " ORDER BY c.fechaInicio Desc";

  Query query =
  entityManager.createQuery(queryString).setFirstResult(page*size)
                      .setMaxResults(size+1);



  if (grupoIds.size() != 0) {
    for (int i = 0; i<grupoIds.size(); i++) {
      query.setParameter("item" + i, grupoIds.get(i));
```

```
30        }
31
32      }
33
34      List<CuestionarioTipoTest> cuestionarios =
        query.getResultList();
```

9.1.3 Implementación de los servicios de la Capa Modelo

A continuación se explican los detalles más relevantes de la implementación de los servicios de la capa modelo, la inyección de dependencias y la transaccionalidad.

9.1.4 Inyección de dependencias

Para implementar los servicios es necesario que cada uno pueda tener acceso a los DAOs que necesite (a ser posible sin necesidad de conocer las clases de implementación).

Para implementar este comportamiento se ha hecho uso de las anotaciones que provee Spring para realizar la inyección de dependencias.

Se coloca @Service sobre las clases de los servicios y @Repository sobre los DAOs, para que estos puedan ser gestionados por el contenedor de objetos. El contenedor de objetos es el encargado de crear los objetos e inyectarles las referencias de los objetos de los que dependen.

Durante el arranque del backend el contenedor de objetos de Spring inspecciona las clases anotadas con @Repository y @Service y crea una instancia de cada una de ellas (bean).

Para cada bean creado inspecciona los atributos anotados con @Autowired y los inicializa con un bean de este tipo. Por defecto, inyecta un bean que implementa la interfaz de cada uno de los atributos marcados con @Autowired.

9.1.5 Transaccionalidad

Spring permite implemntar la transaccionalidad de los métodos de los servicios locales a través del uso de la anotación @Transactional.

Esta anotación, colocada encima de cada método o a nivel de clase, permite que cuando se llama a una operación del modelo desde la capa de servicios REST, se invoque una nueva operación y a esta se enganchen las siguientes operaciones que se invoquen. Por tanto, las operaciones invocadas de los Daos se enganchan a la transacción actualmente creada.

Se han anotado con @Transactional(readOnly=true) algunos métodos como el siguiente, ya que sólo ejecutan operaciones de lectura. Así permitimos que el gestor de transacciones realice optimizaciones al respecto.

```
1  //busca el mensaje y comprueba que el profesor indicado es quien
     lo envío
```

```
    @Transactional(readOnly=true)
    @Override
    public Mensaje verDetallesMnsajeProfesor(Long profesorId,Long
    mensajeId) throws InstanceNotFoundException {
      Optional<Mensaje> mensaje =mensajeDao.findById(mensajeId);

      if (!mensaje.isPresent() ||
    mensaje.get().getProfesor().getId() != profesorId) {
        throw new
    InstanceNotFoundException("project.entities.profesor",
    mensajeId);
      }
      return mensaje.get();
    }
```

9.1.6 Implementación de los controladores/servicios REST

Se ha definido una clase controlador por cada servicio local. Cada uno de estos controladores se anota con @RestController para que pueda ser gestionado por el contenedor de objetos de Spring y poder así inyectarle el servicio local asociado marcado con la anotación @AutoWired en el propio controlador.

Además, encima de cada clase controlador se coloca la anotación @RequestMapping indicando el nombre de la ruta a la que responderá cada controlador. Cada operación de los controladores completa la ruta del propio controlador para especificar la petición exacta a la que responderá. Esto se hace a través de la anotación (@GetMapping/@PostMapping/@PutMapping)("/ruta"). A continuación se muestra como ejemplo la implementación de una de las operaciones del controlador TestController.

```
  //Devuelve los cuestionarios tipo test creados por un profesor
  @GetMapping("/cuestionariosProfesor")
  public BlockDto<CuestionarioTipoTestDto>
    encontrarCuestionariosProfesor(
      @RequestAttribute Long userId,
      @RequestParam(defaultValue="0") int page) throws
    InstanceNotFoundException {
    Block<CuestionarioTipoTest> cuestionarios =
    testService.encontrarCuestionariosProfesor(userId, page, 10);
    return new
    BlockDto<>(toCuestionarioDtos(cuestionarios.getItems()),
    cuestionarios.getExistMoreItems());
  }
```

9.1.7 Implementación de la autenticación y el control de acceso

Algunas peticiones requieren conocer el id del usuario. Es necesario emplear un esquema que permita enviar el id del usuario desde el frontend al backend de forma segura e inyectar el id del usuario en el parámetro userId de los métodos de los controladores.

Es necesario poder identificar de forma unívoca a cada uno de los usuarios y darle los permisos de acceso pertinentes. Para ello se opta por aplicar el estándar JWT (JSON Web token), gracias al cual es posible a través de un token identificar a un usuario, su rol y por tanto sus privilegios. Es importante destacar que se ha definido un TINYINT en base de datos que representa el rol de cada una de las entidades usuario (con valor 1 para Profesor y valor 0 para Alumno). Esto se representa en las entidades del backend como un enum (enum RoleType ALUMNO, PROFESOR) y será gracias a esto y a Spring Security que se podrán dar los privilegios de acceso pertinentes a los usuarios.

Además, la contraseña de los usuarios (profesores y alumnos) se guarda encriptada en BBDD. Para poder validar la contraseña recibida de un usuario lo que se hace es encriptarla y compararla con la persistida en base de datos.

Para la implementación del estándar JWT se han utilizado las siguientes clases pertenecientes al paquete es.udc.tfgproject.backend.rest.common.

- JwtFilter: Encargado de validar el token (comprueba firma válida + token no caducado). Una vez validado el token, lee el userId y habilita inyecciońn del parámetro userId en los métodos de los controladores. Además lee el rol e informa a Spring Security del rol del usuario.
- JwtGenerator: Es la interfaz que define las operaciones de generar token y recuperar información de un token
- JwtGeneratorImpl: Implementa las funciones de la interfaz :
  - String generate(JwtInfo info): A partir del id, del username y del rol genera un token válido, representado por un String. (tarea de encriptado)
  - JwtInfo getInfo(String token): A partir del token recupera el id, el username y el rol. (tarea de desencriptado)
- JwtInfo: Es una clase que representa parte de la información de un usuario, válido tanto para el profesor como para el alumno. Posee los atributos necesarios para identificar a un usuario: id, username y rol.
- SecurityConfig: Clase que extiende de WebSecurityConfigurerAdapter y donde se puede configurar el control de acceso a los recursos por parte de los distintos roles. En la figura 9.1 (página 56) se muestran los permisos de acceso definidos.

Además, para completar el control de acceso, en la capa lógica de negocio se implementa código de control de acceso a los recursos. Este es el caso, por ejemplo, de la operación verDetallesMnsajeAlumno (Long alumnoId,Long mensajeId), que antes de devolver el mensaje, comprueba si el mensaje en cuestión ha sido enviado al alumno que se indica en el parámetro alumnoId. En caso de que el mensaje no haya sido enviado a ese alumno se lanza una excepción.

```
23
24          http.cors().and().csrf().disable()
25              .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
26              .addFilter(new JwtFilter(authenticationManager(), jwtGenerator))
27              .authorizeRequests()
28              .antMatchers("/users/signUp").permitAll()
29              .antMatchers("/users/login").permitAll()
30              .antMatchers("/users/loginProfesor").permitAll()
31              .antMatchers("/users/loginAlumno").permitAll()
32              .antMatchers("/users/loginFromServiceToken").permitAll()
33              .antMatchers("/mensajes/sendMensaje").hasRole("PROFESOR")
34              .antMatchers("/mensajes/sendMensajeToGrupo").hasRole("PROFESOR")
35              .antMatchers("/mensajes/sendMensajeToAlumno").hasRole("PROFESOR")
36              .antMatchers("/mensajes/mensajesAlumno").hasRole("ALUMNO")
37              .antMatchers("/mensajes/mensajesProfesor").hasRole("PROFESOR")
38              .antMatchers("/mensajes/mensajeAlumno/{mensajeId}").hasRole("ALUMNO")
39              .antMatchers("/mensajes/mensajeProfesor/{mensajeId}").hasRole("PROFESOR")
40              .antMatchers("/mensajes/gruposProfesor").permitAll()
41              .antMatchers("/mensajes/alumnosGrupo").permitAll()
42              .antMatchers("/calendario/createEvento").hasRole("PROFESOR")
43              .antMatchers("/calendario/deleteEvento").hasRole("PROFESOR")
44              .antMatchers("/calendario/eventosProfesor").hasRole("PROFESOR")
45              .antMatchers("/calendario/eventosAlumno").hasRole("ALUMNO")
46              .antMatchers("/calendario/eventosProfesorForADay").hasRole("PROFESOR")
47              .antMatchers("/calendario/eventosAlumnoForADay").hasRole("ALUMNO")
48              .antMatchers("/foro/preguntasAlumno").hasRole("ALUMNO")
49              .antMatchers("/foro/createPregunta").hasRole("ALUMNO")
50              .antMatchers("/foro/preguntasProfesor").hasRole("PROFESOR")
51              .antMatchers("/foro/createRespuesta").hasRole("ALUMNO")
52              .antMatchers("/foro/AsignaturasAlumno").hasRole("ALUMNO")
53              .antMatchers("/foro/RepuestasAlumno").hasRole("ALUMNO")
54              .antMatchers("/foro/RepuestasProfesor").hasRole("PROFESOR")
55              .antMatchers("/foro/AsignaturasProfesor").hasRole("PROFESOR")
56              .antMatchers("/ranking/createPremio").hasRole("PROFESOR")
57              .antMatchers("/ranking/findPremiosProfesor").hasRole("PROFESOR")
58              .antMatchers("/ranking/deletePremio").hasRole("PROFESOR")
59              .antMatchers("/ranking/findPremiosAlumno").hasRole("ALUMNO")
60              .antMatchers("/ranking/AsignaturasActualesAlumno").hasRole("ALUMNO")
61              .antMatchers("/ranking/findRankingAlumnoByAsignaturaId").hasRole("ALUMNO")
62              .antMatchers("/ranking/premio").permitAll()
63              .antMatchers("/ranking/puntuarRespuesta").hasRole("PROFESOR")
64              .antMatchers("/test/testDetails").permitAll()
65              .antMatchers("/test/estadisticasCuestionario").permitAll()
66              .antMatchers("/test/createCuestionario").hasRole("PROFESOR")
67              .antMatchers("/test/cuestionariosProfesor").hasRole("PROFESOR")
68              .antMatchers("/test/cuestionariosAlumno").hasRole("ALUMNO")
69              .antMatchers("/test/responderCuestionario").hasRole("ALUMNO")
70              .antMatchers("/test/puntuacionesAlumnos").hasRole("PROFESOR")
71              .antMatchers("/**").hasAnyRole("PROFESOR" ,"ALUMNO");
72
```

Figura 9.1: Permisos de acceso a los recursos para los distintos roles

9.2 Implementación del frontend

A continuación se muestran los detalles más relevantes sobre la implementación de la aplicación frontend

9.2.1 Gestión del estado (Redux)

Para gestionar el estado de la aplicación frontend de forma centralizada y modular, se ha utilizado la librería Redux. Con Redux se consigue sacar la mayor parte del estado y su lógica

de modificación de los componentes. Para hacerlo es necesario implementar los siguientes elementos:

- Store: Es un objeto de Redux que almacena el estado de la aplicación.
- Acciones: Cada vez que se quiere modificar el estado (normalmente como consecuencia de una interacción del usuario), hay que enviar un objeto acción a store. La acción indica su tipo y los datos necesarios.
- Reductor: Es una función encargada de producir un nuevo estado ante una acción. Devuelve el nuevo estado a partir del estado anterior y la acción.

De manera simplificada se puede ver en la figura 9.2 (página 57) el flujo de información en Redux una vez se lanza una petición.

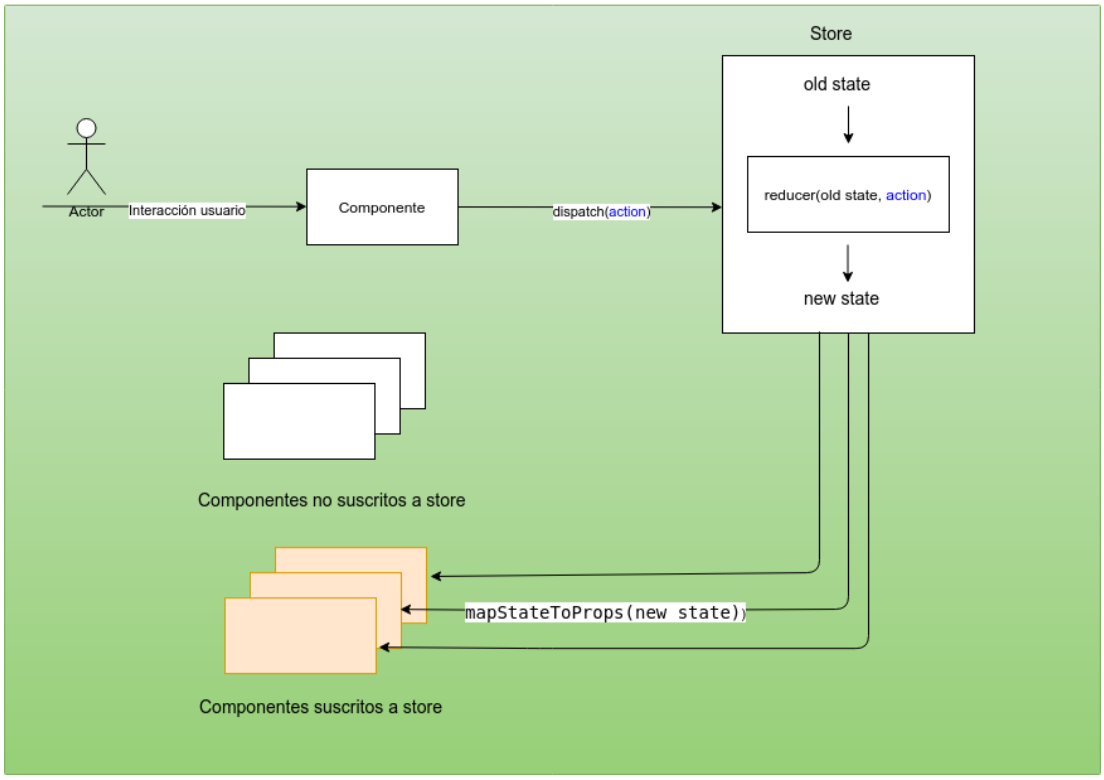


Figura 9.2: Flujo Redux

En la aplicación existen componentes que necesitan conocer el estado de Redux. Estos componentes se subscriben al store para poder recibir el estado cuando este cambia. Para que los componentes puedan suscribirse al store, en index.js (lugar donde se renderiza el componente padre (App)), se coloca lo siguiente:

```
    ReactDOM.render(
    <Provider store={store}>
        <IntlProvider locale={locale} messages={messages}>
            <App/>
        </IntlProvider>
    </Provider>,
    document.getElementById('root'));
```

El componente Provider permite pasar el store a todos los componentes en el árbol de componentes.

En los componentes que necesitan conocer el estado se coloca lo siguiente:

```
const mapStateToProps = (state) => ({
    mensajesProfesorSearch:
    selectors.getMensajesProfesorSearch(state)
});

const mapDispatchToProps = {
    previousFindMensajesProfesorResultPage:
    actions.previousFindMensajesProfesorResultPage,
    nextFindMensajesProfesorResultPage:
    actions.nextFindMensajesProfesorResultPage
}

export default connect(mapStateToProps,
    mapDispatchToProps)(FindMensajesProfesorResult);


```

Connect recibe dos argumentos de tipo función:

- mapStateToProps: Para crear un objeto que mapee el estado a las propiedades del componente.
- mapDispatchToProps: para crear un objeto con funciones que pueden hacer un dispatch de acciones a store.

Gracias a connect el componente FindMensajesProfesorResult queda subscrito a store.
Para la actualización del estado se implementa el reductor:

```
 ...
const mensajesAlumnoSearch = (state =
    initialState.mensajesAlumnoSearch, action) => {

    switch (action.type) {

```

```
        case actionTypes.FIND_MENSAJES_ALUMNO_COMPLETED:
            return action.mensajesAlumnoSearch;

        case actionTypes.CLEAR_MENSAJES_ALUMNO_SEARCH:
            return initialState.mensajesAlumnoSearch;

        default:
            return state;

    }
...
const reducer = combineReducers({
    mensajesAlumnoSearch,
    mensaje,
    gruposProfesor,
    alumnosGrupoSearch
});

export default reducer;
}


```

En store/rootReducer.js se encuentra el reductor raíz, que combina los reductores de todos los módulos. Gracias a él el estado es global a toda la aplicación y puede ser accedido a través de los selectores.

```

const rootReducer = combineReducers({
    app: app.reducer,
    users: users.reducer,
    mensaje: mensaje.reducer,
    calendario: calendario.reducer,
    foro: foro.reducer,
    ranking: ranking.reducer,
    cuestionario: cuestionario.reducer,
});
export default rootReducer;

```

A continuación se muestra parte de los selectores del módulo de mensajes. Estos sirven para ocultar la estructura interna del estado:

```
 const getModuleState = state => state.mensaje;

export const getMensajesAlumnoSearch = state =>
    getModuleState(state).mensajesAlumnoSearch;
```

```
    ...

```

9.2.2 Gestión del enrutado

Para poder navegar de unas pantallas a otras de la aplicación se ha utilizado la librería React Router. Esta librería proporciona el componente Router, que cumple una función similar al componente Provider de Redux. Debe contener a los componentes que usen enlaces a pantallas o provoquen programáticamente un cambio de pantalla. A estos componentes contenidos les inyecta propiedades relativas al enrutamiento.

```
render() {

        return (
            <Router>
                <div>
                    <Header/>
                    <Body/>

                </div>
            </Router>
        );

    }

```

Cuando se provoca un cambio de pantalla, internamente se notifica al componente Router, Router se vuelve a renderizar y en consecuencia también los componentes que contiene, en este caso Header y Body como podemos ver en el código anterior. Además estos componentes deben estar encapsulados por el HOC withRouter. withRouter inyecta varias propiedades al componente que encapsula, como lo son match, location y history.

A continuación se ve como el componente Body es encapsulado por withRouter.

```
export default withRouter(connect(mapStateToProps)(Body));

```

Cuando se produce un cambio de pantalla, cambia la propiedad location, que contiene entre otras cosas el nuevo path seleccionado. Esto hace que se renderice el Body y consecuentemente el componente correspondiente a la ruta seleccionada.

```
const Body = ({user}) => (

    <div className="container col-lg-10">
        <Route path="/" component={AppGlobalComponents}/>
```

```
5           <Switch>
6               <Route exact path="/" component={Home}/>
7               {user && esAlumno(user.role) && <Route exact
    path="/mensajes/find-mensajes-alumno-result"
    component={FindMensajesAlumnoResult}/>}
8               {user && esProfesor(user.role) && <Route exact
    path="/mensajes/find-mensajes-profesor-result"
    component={FindMensajesProfesorResult}/>}
9               {user && esProfesor(user.role) && <Route exact
    path="/mensajes/contact-alumnos" component={ContactAlumnos}/>}
10              ...
11
```

Como vemos, en función del path se renderiza un componente u otro dentro del Body.

9.2.3 Control de acceso

Aunque en el backend ya se realiza un control de acceso a los recursos, en el frontend también se colabora al respecto, para conseguir así una aplicación más segura y robusta.

El acceso a las diferentes rutas está controlado de forma que usuarios con el rol de Alumno no obtengan ningún tipo de respuesta por parte de enlaces a recursos solo habilitados para los profesores. Para implementar este comportamiento es necesario conocer si hay un usuario autenticado, y si lo hay, conocer su rol. Esta información se puede obtener a través del estado de Redux.

```
1  const mapStateToProps = (state, ownProps) => ({
2      user: users.selectors.getUser(state)
3  });
4
5  function esProfesor(role) {
6    return role === "PROFESOR";
7  }
8  function esAlumno(role) {
9      return role === "ALUMNO";
10 }
11
```

Como podemos ver a continuación, en el body, cuando se machea la ruta seleccionada con la de los componentes, también se comprueba si el rol de usuario es el que debería ser o no. Esta comprobación se puede hacer, por ejemplo, a través de las funciones definidas esAlumno y esProfesor.

```
1       <Switch>
2               <Route exact path="/" component={Home}/>
```

```
            {user && esAlumno(user.role) && <Route exact
    path="/mensajes/find-mensajes-alumno-result"
    component={FindMensajesAlumnoResult}/>}
    component={FindMensajesAlumnoResult}/>}
}

```

9.3 Software requerido

En este apartado se especifica el software que ha sido utilizado para la realización de este proyecto. Para el desarrollo del backend se ha utilizado lo siguiente:

- **Eclipse IDE:** en la versión Photon Release (4.8.0)
- **JDK** en la versión 1.8.0.181
- **Apache Maven** enn la versión 3.5.4
- **MySql** en la versión 14.14 Distrib 5.7.29
- **Spring Boot** en la versión 2.1.2
- **Postman** en la versión 7.30.1

Para el desarrollo del frontend se ha utilizado lo siguiente:

- **Visual Studio Code** en la versión 1.44.0
- **Node.js** en la versión 8.10.0
- **NPM** en la versión 3.5.2
- **React** en la versión 16.4.1
- **Boostrap** en la versión 4.1.1

Además se han utilizado git en la versión 2.17.1 como herramienta de control de versiones

9.4 Estructura

En la figura 9.3 (página 63) se muestra la estructura del proyecto maven

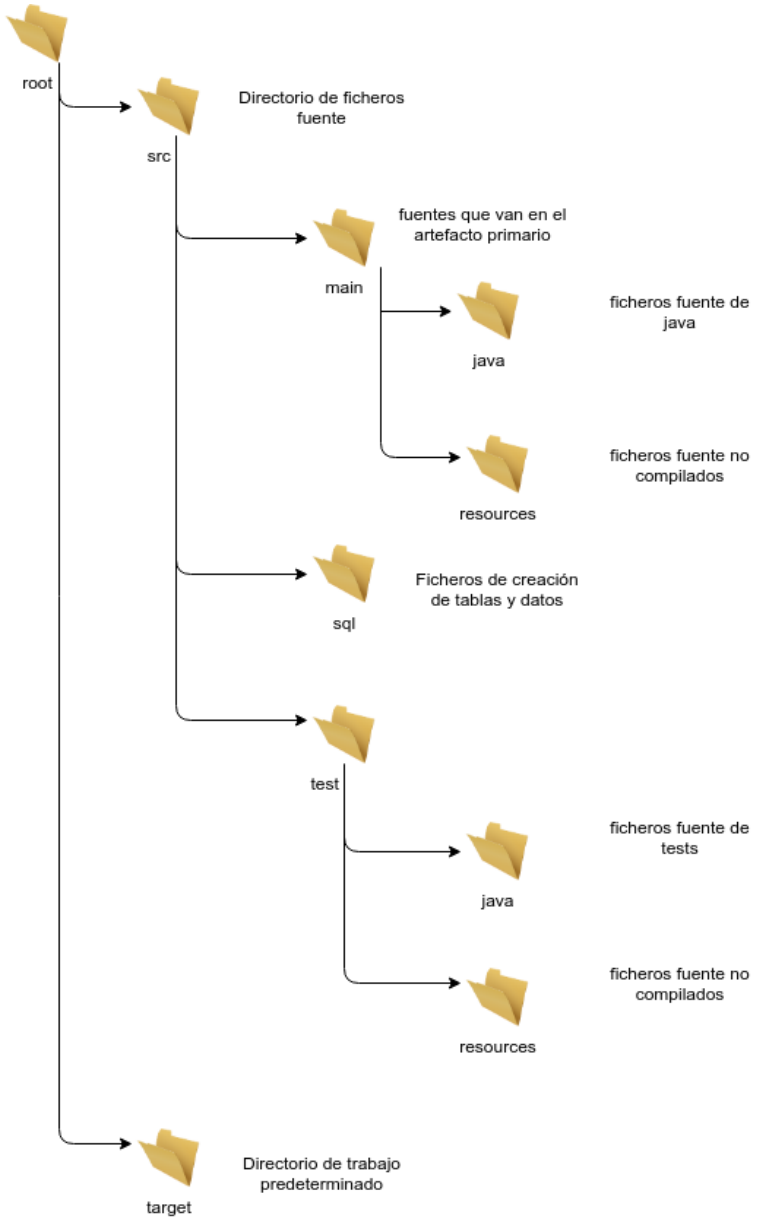


Figura 9.3: Estructura de directorios Maven

En la figura 9.4 (página 64) se muestra la estructura del proyecto frontend

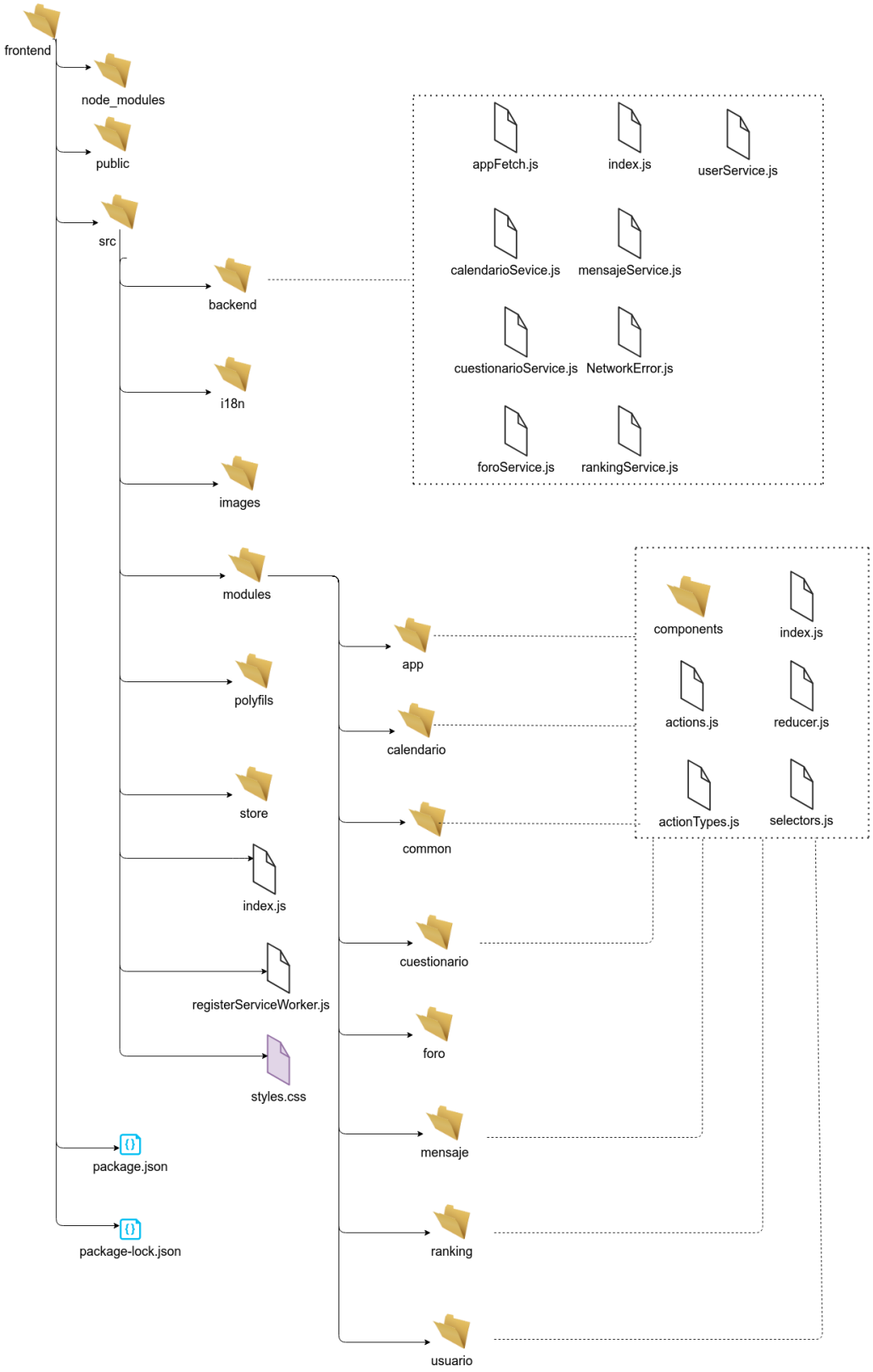


Figura 9.4: Estructura de directorios frontend

Capítulo 10

Pruebas

En este proyecto se ha realizado una fase de pruebas posterior a la implementación de cada uno de los subsistemas elaborados. Esta fase es muy importante ya que nos permite aumentar la confianza que tenemos sobre el código realizado, reduciendo considerablemente la probabilidad de encontrar errores en un futuro. Existen varios tipos de pruebas diferentes, pruebas de unidad, de integración, de sistema y de aceptación.

En este proyecto se han realizado pruebas de integración sobre la capa modelo, pudiendo así validar el funcionamiento de la capa de acceso a datos y la capa de servicios. También se han realizado pruebas sobre los controladores del API REST, pudiendo así validar por ejemplo el funcionamiento del control de acceso o la devolución de los datos por parte del servicio en formato JSON.

10.1 Pruebas de integración sobre la capa modelo

Las pruebas de integración se ocupan de verificar el correcto ensamblaje entre los distintos componentes con el fin de comprobar que interactúan correctamente a través de sus interfaces y cubren la funcionalidad establecida. Además las pruebas automatizadas de integración son de mucha ayuda cada vez que se cambia parte del software. Así cuando se introduce un cambio en el sistema se puede comprobar de forma sencilla si otras unidades del código se han visto afectadas.

Para las pruebas de integración el sistema cuenta con una base de datos distinta a la que se utiliza para probar la aplicación como desarrollador. Esta base de datos distinta, que sirve para ejecutar las pruebas automatizadas de integración, se llama tfgprojecttest. Las funcionalidades de cada uno de los servicios son probadas en la clase cuyo nombre es el nombre del servicio seguido de la palabra Test. Estas clases inyectan los servicios y los DAOs que necesitan para ser ejecutadas.

Se ha usado la librería spring-test para implementar ágilmente las clases de prueba de los

servicios. Cada una de estas clases de prueba utiliza la anotación @RunWith para especificar que las pruebas se ejecutan con SpringRunner (spring-test). Esto facilita tareas como la inyección de dependencias o el tratamiento individualizado de cada uno de los tests. A través de la anotación @Transactional cada caso de prueba termina con un rolback, deshaciendo así los cambios que haya hecho en base de datos.

La estructura de cada caso de pruebas es la siguiente:

- Inicialización de los datos.
- Llamada a funciones.
- Aserción, donde se compara el resultado esperado con el obtenido a través del assertEquals proporcionado por el framework Junit.

Una vez validadas todas las pruebas realizadas se ha utilizado la herramienta jacoco para medir la cobertura. Los resultados de la medición se pueden ver en la figura 10.1 (página 66)

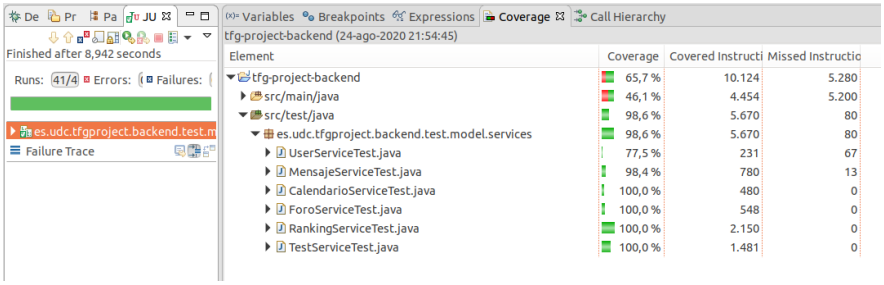


Figura 10.1: Cobertura de las pruebas

10.2 Pruebas sobre la API REST

Para probar el comportamiento de los controladores se ha usado la herramienta Postman. Esta herramienta permite lanzar peticiones http (GET/POST/PUT/DELETE) al servidor y visualizar las respuestas que devuelve. Además nos permite utilizar el token para identificarnos en el sistema y poder probar todas las operaciones, tanto las que son restringidas a un determinado rol como las que no. En la figura 10.2 (página 67) se muestra un ejemplo de petición a la API REST realizada desde Postman. En la petición, un profesor trata de enviar un mensaje a sus alumnos. Puede verse que la operación ha sido ejecutada con éxito y devuelve el mensaje generado junto con otros mensajes más para completar el número de mensajes que se quieren mostrar por pantalla, determinado por el atributo size.

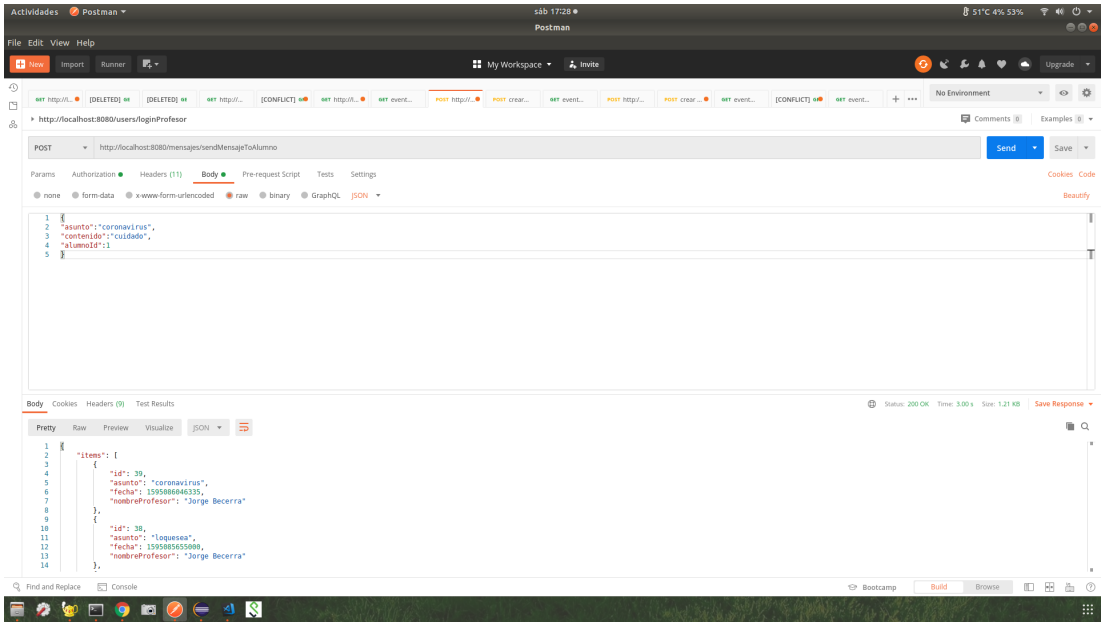


Figura 10.2: Ejemplo de prueba de la API REST a través de Postman

Capítulo 11

Conclusiones y futuras líneas de trabajo

En este capítulo se analiza el trabajo realizado y se reflexiona sobre los objetivos cumplidos. El proyecto desarrollado no habría sido posible sin la formación obtenida durante estos años de formación. A través de este capítulo se analizan las competencias adquiridas durante la titulación y que han servido para poder concluir el trabajo de una manera satisfactoria. Son muchos los conocimientos adquiridos durante la titulación y por ello no es posible reflejar todos ellos en un solo proyecto. Dada la naturaleza de este proyecto, los conocimientos adquiridos que más se pueden ver reflejados en el proyecto están relacionados con los impartidos en las materias de los últimos dos años de carrera (en especial las materias de la mención Ingeniería del software).

11.1 Conclusiones

Una vez terminado el proyecto, toca echar la vista atrás para ver si se han cumplido los objetivos planteados en un inicio. Por una parte se analizarań los objetivos cumplidos relacionados con el software desarrollado y por otra los objetivos personales cumplidos.

En lo relativo a la aplicación, todas las funcionalidades requeridas han sido implementadas. El sistema desarrollado cuenta con un módulo de mensajería, permitiendo así la comunicación de los profesores con los alumnos; un módulo de eventos, donde los profesores de forma cómoda pueden ver los eventos introducidos en su calendario e introducir nuevos eventos relacionados con alguno de sus grupos para que así los alumnos puedan visualizarlos de igual manera; un foro, donde los alumnos pueden plantear preguntas, responder a las de sus compañeros e incluso obtener puntos por ello; un módulo de ranking, donde los alumnos pueden ganar premios y estar al tanto de su clasificación en el ranking de alumnos; y un importante sistema de tests, donde los alumnos podrán ser evaluados o responder anónimamente a cues-

tionarios, y los profesores podrán crear dichos cuestionarios y obtener estadísticas acerca de los mismos.

Además hay que prestar atención a la calidad del software desarrollado, puesto que uno de los principales objetivos era poder generar un producto abierto que facilitase la incorporación de nuevos cambios y de nuevas funcionalidades. En este sentido se puede decir que se ha cumplido. Esto ha sido gracias a:

- El tiempo invertido en los inicios de cada iteración en el diseño de los componentes del sistema. No solo los diagramas de la fase de diseño han servido para diseñar la solución, también son una fuente de información muy buena para consultar cuando se necesite, ya sea para añadir nuevas funcionalidades como para modificar las ya existentes.
- La aplicación de numerosos patrones y principios de diseño
- La documentación del código, en especial la relativa al backend y los subsistemas más complejos (el subsistema de ranking y el subsistema de tests).
- La elección de tecnologías, como React, que permiten que la incorporación de cambios futuros se realice de forma más sencilla y rápida.

Este proyecto ha sido realizado con fines académicos, con la intención de afianzar conocimientos ya adquiridos e incluso adquirir algunos nuevos. Es por ello que se puede decir que a nivel personal los objetivos también han sido cumplidos. He podido trabajar con tecnologías que actualmente son bastante valoradas en el ámbito laboral; he conseguido aplicar los conocimientos adquiridos durante estos últimos años y me he podido dar cuenta de la importancia de las horas invertidas en las fases de análisis y diseño. Ahora soy más consciente que antes de la importancia de construir un software de manera responsable y ordenada y además, he logrado perderle el miedo a algunas tecnologías, en especial a las relacionadas con el desarrollo del frontend.

A continuación se enumeran las competencias adquiridas que más han servido para la realización de este proyecto:

- Poder capturar los requisitos de un sistema de forma correcta: Proceso software e Ingeniería de requisitos
- Construcción del modelo de datos: Bases de datos
- Conocer como funciona un sistema gestor de base de datos y como se debe interactuar con el: Bases de datos avanzadas
- La capacidad de diseñar una solución a un problema complejo abstrayendose de los detalles innecesarios: En general todas las asignaturas relacionadas con el diseño e implementación de un sistema software y en particular la asignatura Diseño software

- El dominio técnico de nuevas tecnologías, en especial javascript y React: Programación avanzada

11.2 Futuras líneas de trabajo

Como ya se ha dicho, con este proyecto no se ha buscado obtener una implementación completa de un sistema LMS. Son muchas las funcionalidades interesantes que podrían ser implementadas en un futuro y algunas de ellas serían muy necesarias para que el sistema pudiese ser utilizado de una manera mucho más cómoda por los usuarios.

Sería interesante crear una herramienta para administrar de forma sencilla los alumnos, los grupos, los cursos, las asignaturas etc, pudiendo por ejemplo introducir en el sistema toda esta información a través de archivos CSV. Estos archivos podrían ser introducidos por un nuevo usuario con el rol de administrador a través de la propia web.

También sería muy interesante la implementación de una aplicación móvil que contase con parte de los módulos de la aplicación web, en especial aquellos relacionados con la comunicación profesor-alumno (módulo de mensajería y módulo de eventos). Los alumnos podrían, por ejemplo, recibir notificaciones emergentes en el móvil cada vez que uno de sus profesores introdujese un nuevo evento en el calendario o un nuevo cuestionario.

Además para poder apreciar el potencial que tiene la generación de un nuevo sistema como este, 100 % adaptable a las necesidades actuales y futuras de los usuarios, se ilustran a continuación algunos ejemplos de funcionalidades que podrían ser incorporadas con relativa facilidad y que no están disponibles en la mayoría de sistemas LMS actuales.

- Se podría añadir un módulo de Revista escolar en la cual, de forma sencilla, un profesor podría ocuparse de la publicación mensual de artículos de diversa temática (publicación de actividades realizadas por el centro , publicación de tareas/dibujos/relatos realizados por los alumnos, etc).

- Podría crearse un módulo de Competiciones deportivas, donde alumnos y profesores de otros colegios pudiesen acceder para la construcción de torneos deportivos entre colegios.

- Se podría añadir también un módulo para comunicar a profesores y padres. A través de este módulo, los padres y los profesores podrían acordar citas para hablar sobre el progreso del alumno o intercambiar información vía chat.

Apéndices

Apéndice A

Manual de usuario

En este apartado se muestra una pequeña guía de usuario en la cual se exponen las principales funcionalidades que ofrece el sistema.

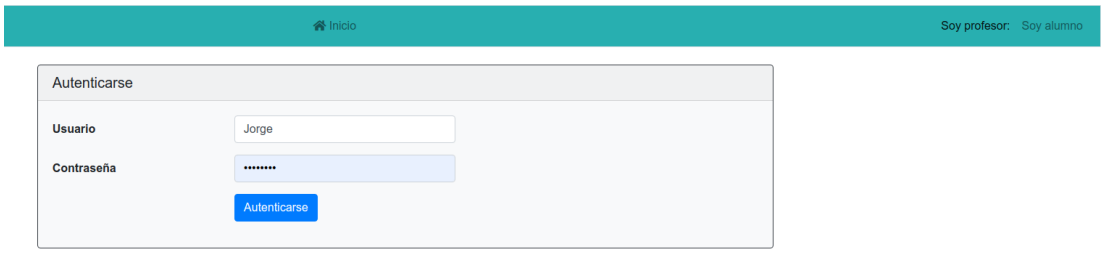


Figura A.1: Login de profesor

Para poder loguearse en el sistema, el usuario debe identificarse a través de la selección de su perfil en la esquina superior derecha de la pantalla.

A.1 Perfil profesor

Presionando sobre ”Soy profesor”, el usuario accede al formulario de login que se muestra en la figura A.1 (página 75). Una vez rellenado correctamente, el usuario accede al menú principal de la aplicación, en este caso al menú correspondiente al perfil profesor.

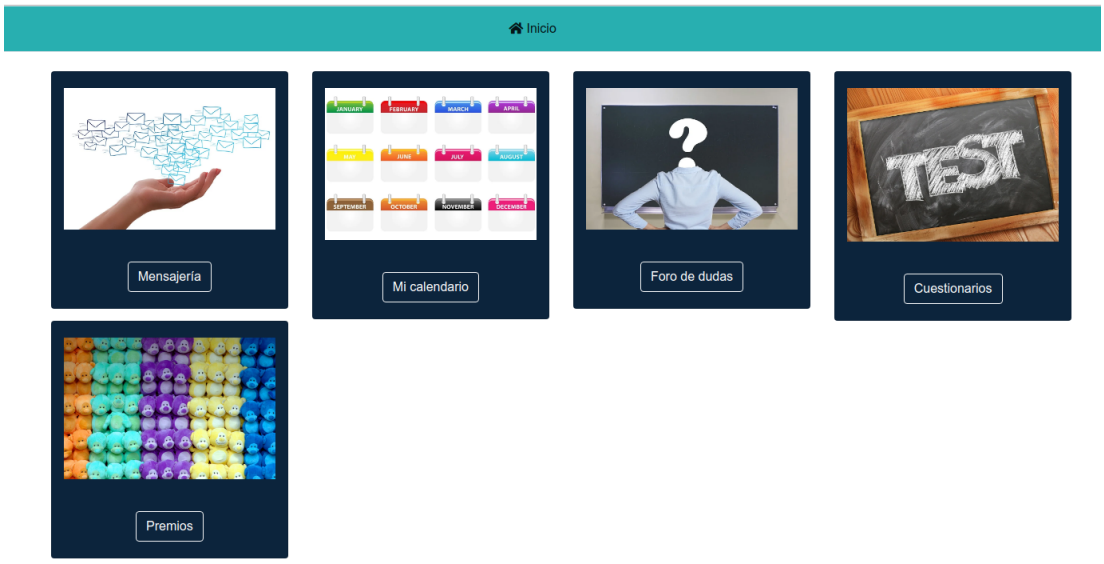


Figura A.2: Menú principal del profesor

En la figura A.2 (página 76) se muestra el menú principal que ve un profesor nada más loguearse. Desde él puede acceder a cada uno de los bloques de funcionalidades: ”Mensajería”, ”Mi Calendario”, ”Foro de dudas”, ”Cuestionarios” y ”Premios”.

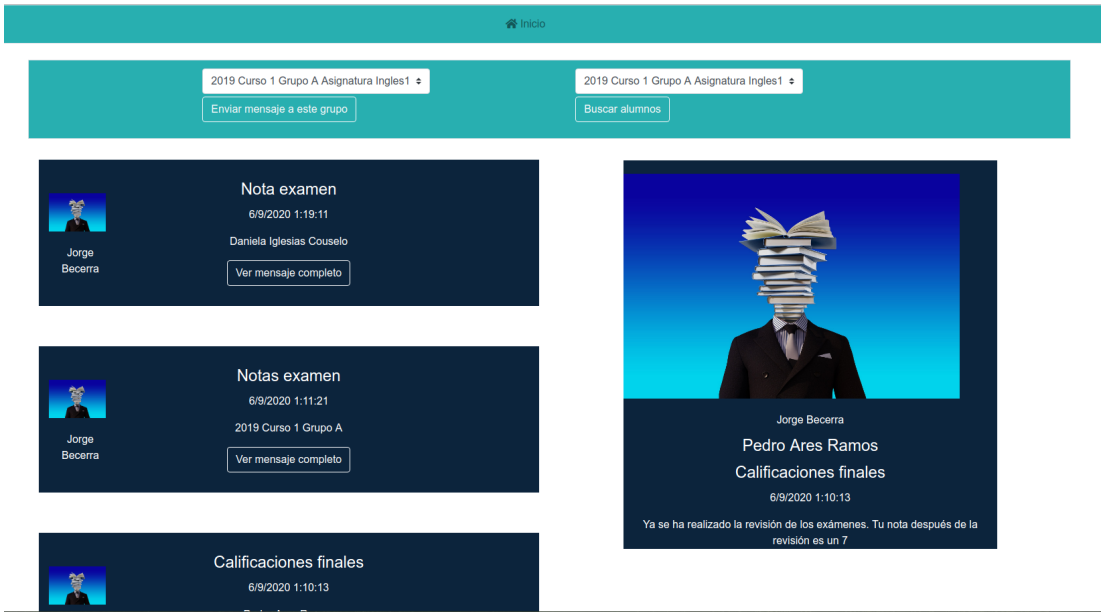


Figura A.3: Mensajes enviados por el profesor

Clicando sobre el bloque ”Mensajería” (del menú A.2 (página 76)), el profesor accede a sus

mensajes enviados. Como se muestra en la figura A.3 (página 76), el profesor puede visualizar la lista de mensajes enviados y acceder a los detalles de alguno de ellos presionando sobre el botón ”Ver mensaje completo”.

Para enviar un mensaje, el profesor tiene a su disposición los siguientes botones: ”Enviar mensaje a este grupo” y ”Buscar alumnos”. Estos botones se encuentran en la barra superior de la pantalla mostrada en la figura A.3 (página 76).

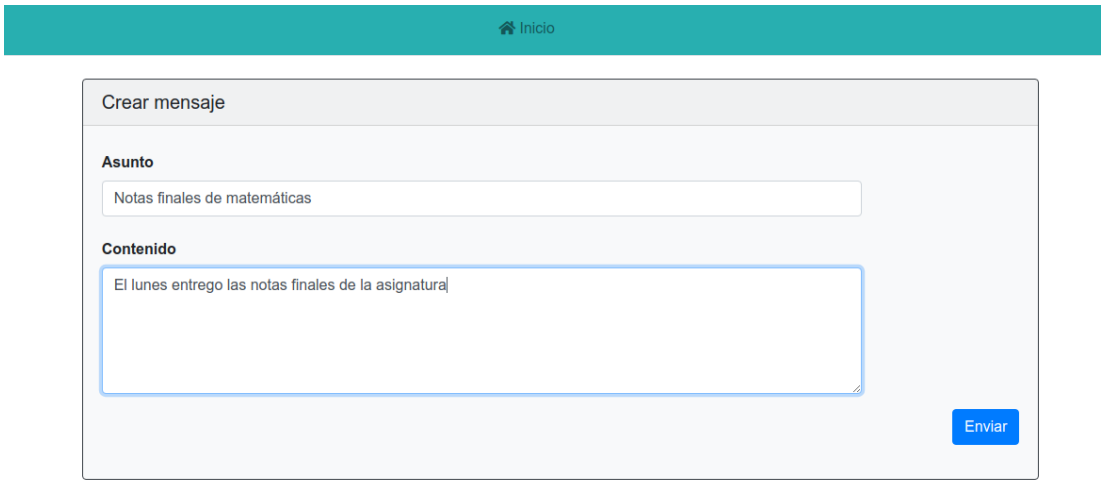


Figura A.4: Enviar mensaje

Para enviar un mensaje a un grupo completo, el profesor debe seleccionar en el selector uno de sus grupos y posteriormente clicar sobre el botón ”Enviar mensaje a este grupo”. Haciendo esto se accede al formulario que aparece en la figura A.4 (página 77). Una vez rellenado se puede clicar sobre el botón ”Enviar” y se enviará el mensaje.

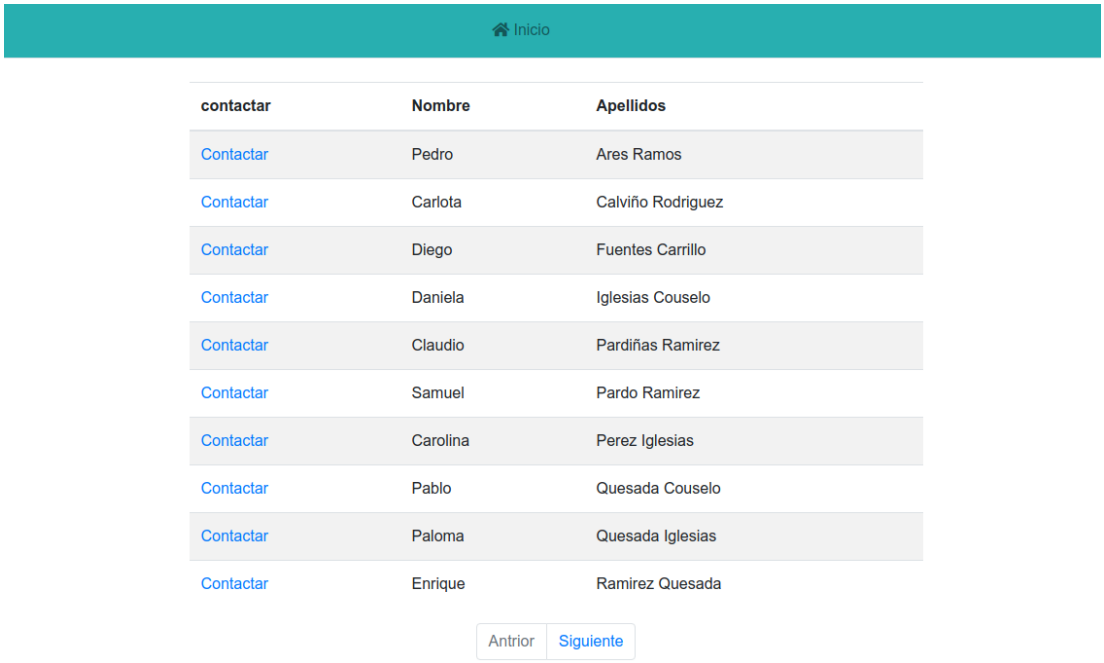


Figura A.5: Alumnos de un determinado grupo

Para enviar un mensaje a un alumno en específico, el profesor debe seleccionar el grupo al que pertenece dicho alumno en el selector y clicar sobre el botón ”Buscar alumnos”. Haciendo esto se llega a la pantalla mostrada en la figura A.5 (página 78), donde el profesor puede seleccionar el alumno al que le quiere enviar el mensaje dando clic sobre el botón ”Contactar”.

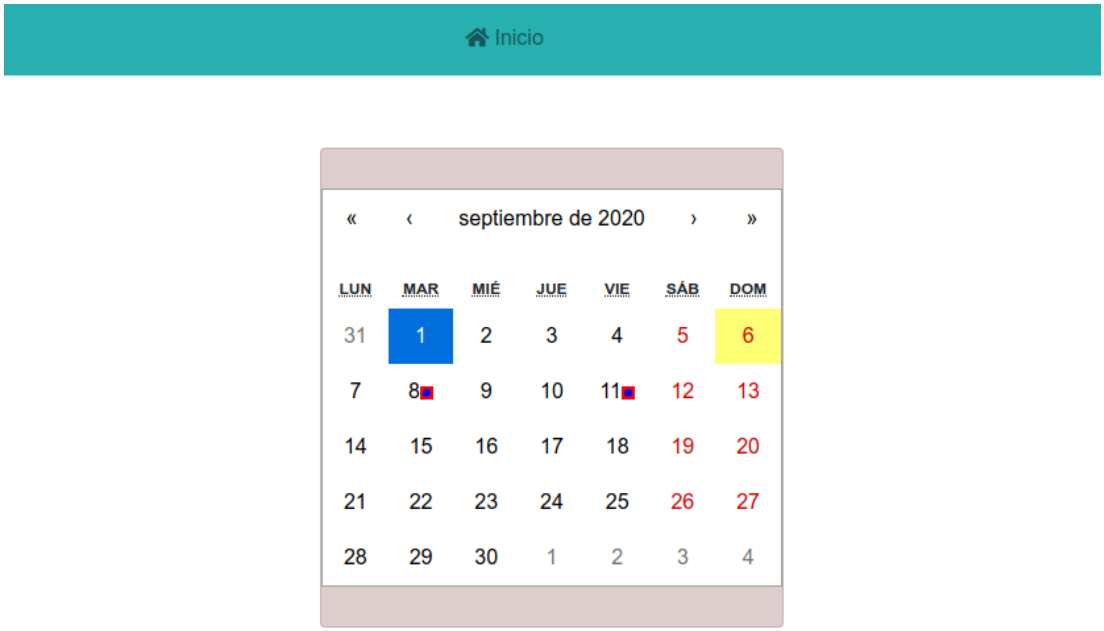


Figura A.6: Calendario con eventos

Clicando sobre el bloque ”Mi calendario” (del menú A.2 (página 76)) , el profesor accede a la pantalla mostrada en la figura A.6 (página 79). Como se puede ver, el profesor puede navegar entre los diferentes meses del calendario y seleccionar la fecha que desee. Las fechas en las cuales ya se haya insertado algún evento aparecen marcadas con el símbolo rojo. Clicando sobre una de las fechas se accede a la pantalla mostrada en la figura A.7 (página 80).

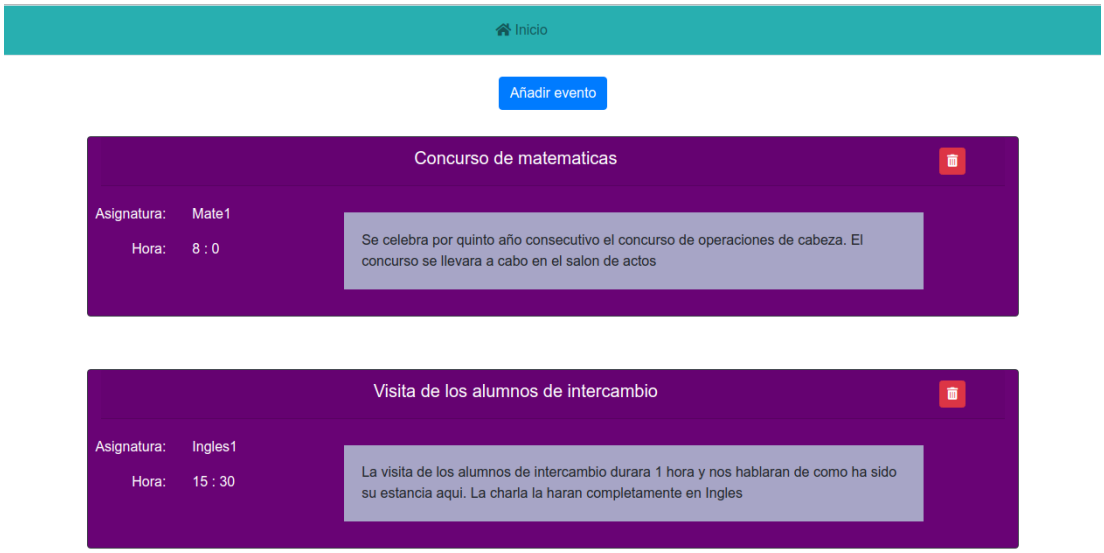


Figura A.7: Eventos creados

En esta pantalla, el profesor puede visualizar sus eventos creados e incluso añadir uno nuevo. También puede borrar un evento dando clic sobre el icono de la basura asociado al evento que quiere eliminar.

Figura A.8: Crear evento

Para añadir un nuevo evento, el profesor debe clicar sobre el botón ”Añadir evento”. Esto le

llevará al formulario mostrado en la figura A.8 (página 80), donde puede rellenar los campos asociados al nuevo evento y crearlo presionando sobre el botón ”Crear evento”.

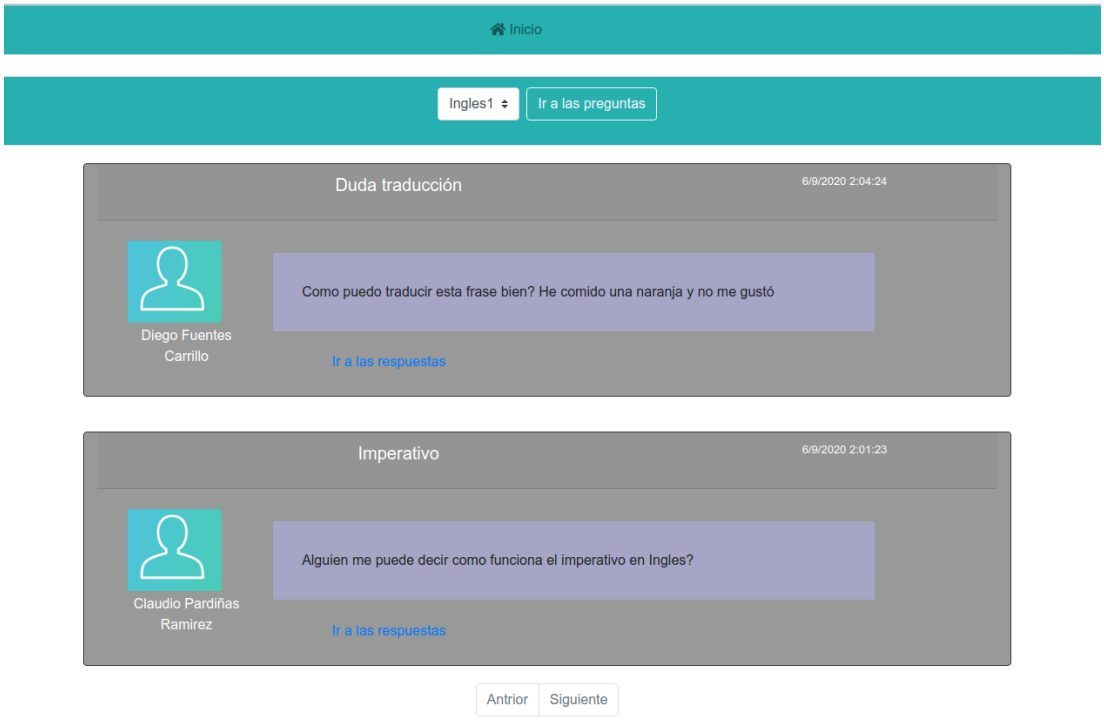


Figura A.9: Preguntas en el foro de una determinada asignatura

Clicando sobre el bloque ”Foro de dudas” (del menú A.2 (página 76)), el profesor accede a la pantalla mostrada en la figura A.9 (página 81). En ella se puede seleccionar, sobre el selector que se encuentra en la parte superior de la pantalla, una de las asignaturas para así buscar las preguntas que han realizado los alumnos frente a esa asignatura.

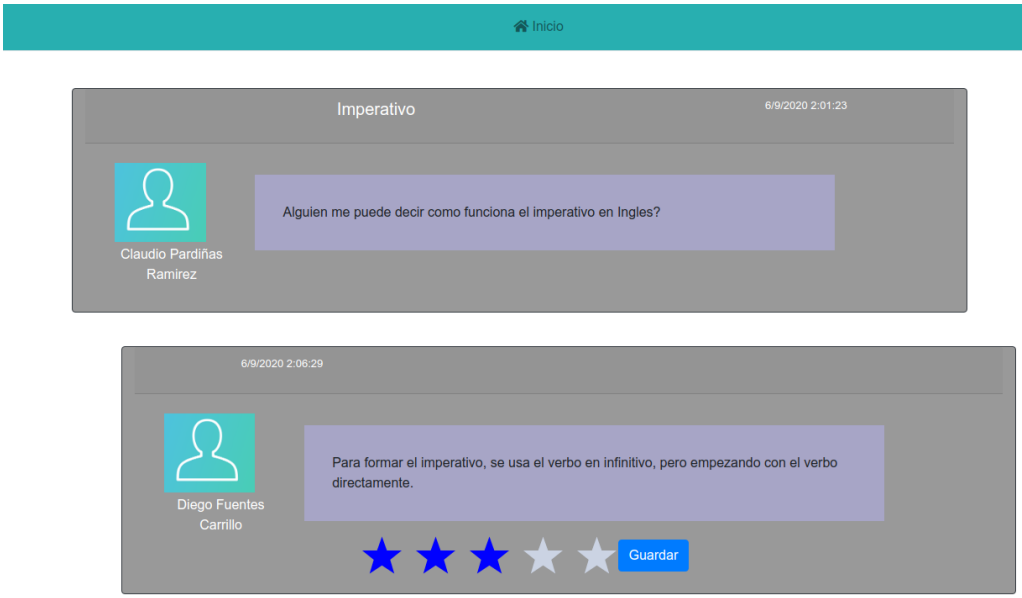


Figura A.10: Pregunta y sus respuestas

Para dirigirse a las respuestas de una determinada pregunta el profesor debe hacer clic sobre el enlace ”Ir a respuestas”. Con esto se accede a la pantalla mostrada en la figura A.10 (página 82). Como se puede apreciar, el profesor puede puntuar cada una de las respuestas asignando un número de estrellas entre 0 y 5 y guardar los cambios.

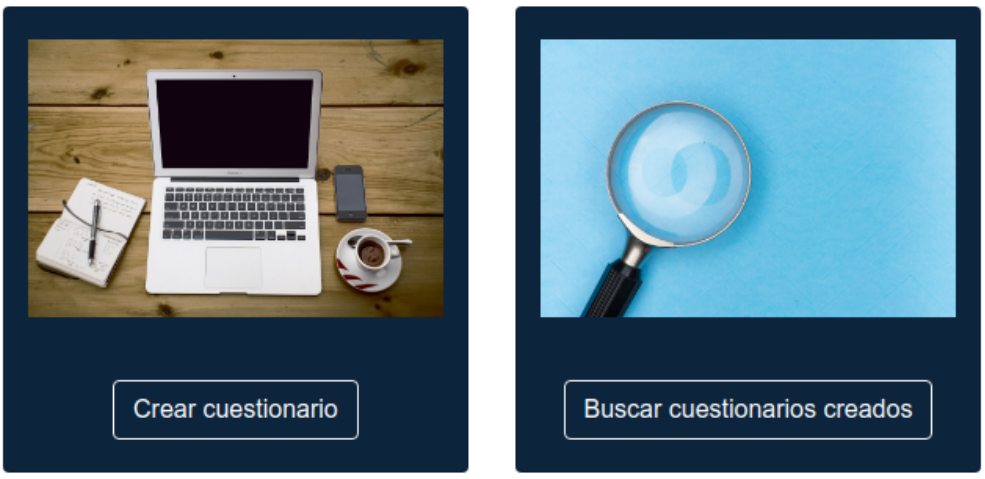


Figura A.11: Menú cuestionarios profesor

Clicando sobre el bloque ”Cuestionarios” (del menú A.2 (página 76)), el profesor accede

a la pantalla mostrada en la figura A.11 (página 82). Como se puede ver, esta pantalla ofrece 2 funcionalidades: ”Crear cuestionario” y ”Buscar cuestionarios creados”.

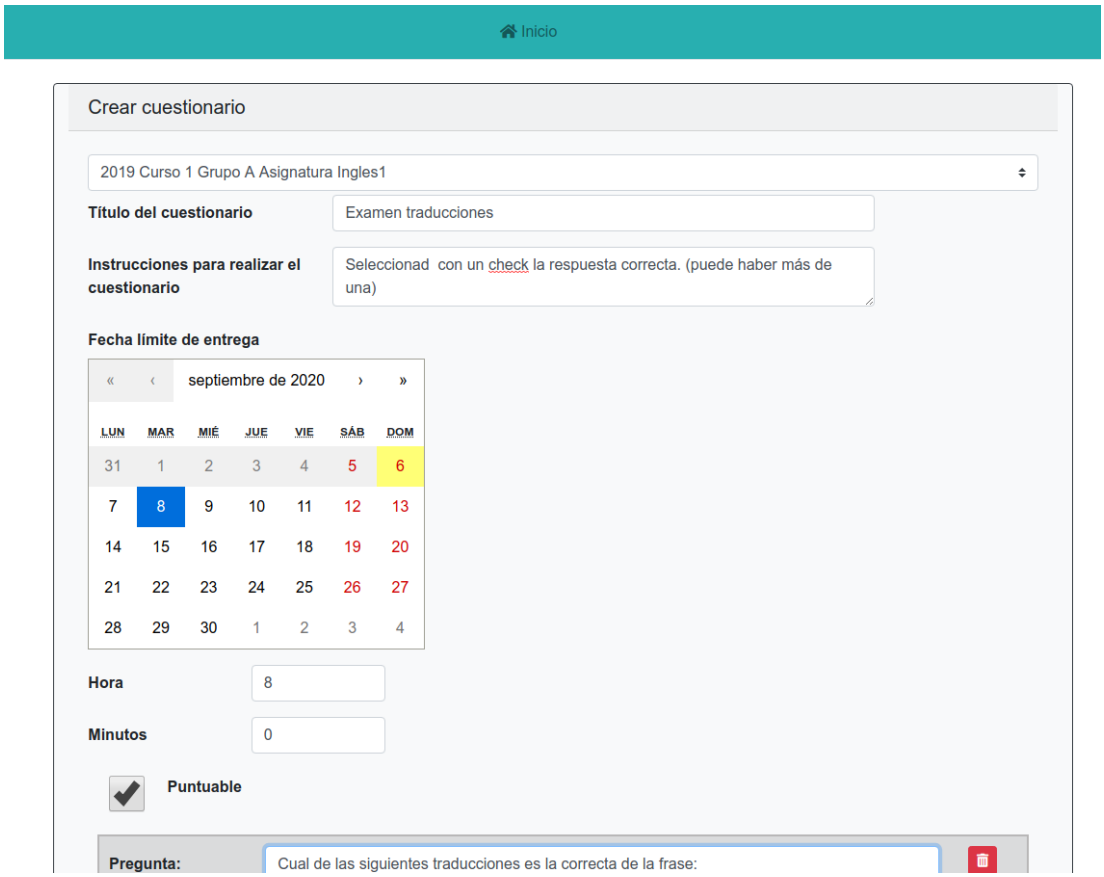


Figura A.12: Formulario crear cuestionario (datos generales)

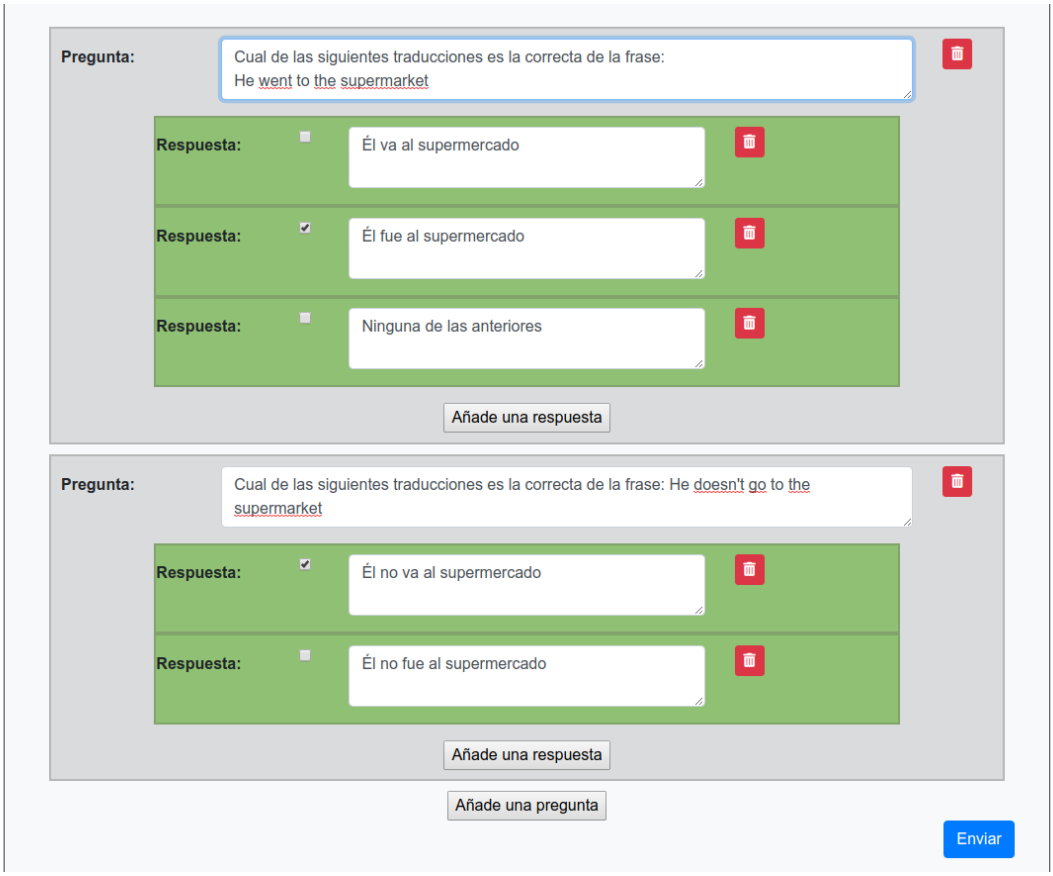


Figura A.13: Formulario crear cuestionario (crear preguntas y respuestas)

Dando clic sobre el botón ”Crear cuestionario”, el profesor accede al formulario mostrado en las figuras A.12 (página 83) y A.13 (página 84). En esta pantalla el usuario puede crear el número de preguntas que desee. Clicando sobre el botón ”Añadir una nueva pregunta”, el profesor puede crear una nueva. El profesor también puede crear el número de respuestas que desee clicando sobre el botón ”Añadir una respuesta”. Haciendo clic sobre el checkbox asociado a una respuesta, el profesor puede marcarla como verdadera. El profesor puede crear dos tipos de cuestionarios distintos, cuestionarios evaluables y cuestionarios no evaluables. Para indicar que el cuestionario es evaluable, el profesor debe marcar la casilla correspondiente al campo ”Puntuable”. Una vez rellenado el formulario con las preguntas y respuestas deseadas, el profesor puede enviar el cuestionario clicando sobre el botón ”Enviar”.

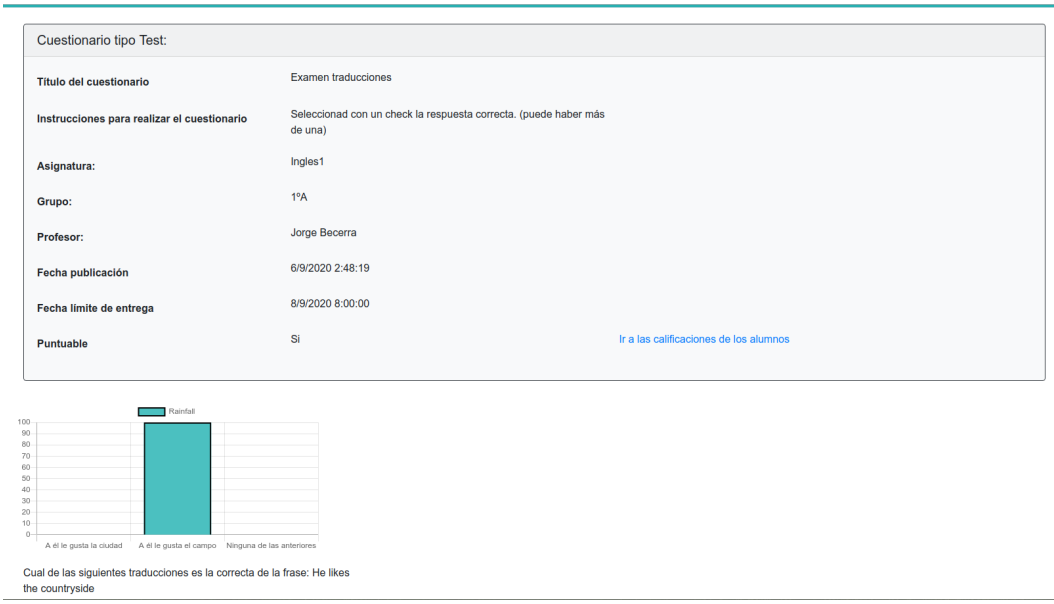


Figura A.14: Ver un determinado cuestionario (con los porcentajes de elección de cada una de las respuestas)

Dando clic sobre el botón ”Buscar cuestionarios creados” de la pantalla mostrada en la figura A.11 (página 82), el profesor accede a los cuestionarios que han sido creados por él. Accediendo a uno de ellos, el profesor puede visualizar las estadísticas sobre dicho cuestionario. En la figura A.14 (página 85) se ve uno de los cuestionarios y sus estadísticas. En esta pantalla se visualiza un diagrama de barras por cada una de las preguntas existentes en el cuestionario. En cada uno de los diagramas se representa una de las preguntas del test junto con el porcentaje de personas que han seleccionado cada una de las respuestas.

Figura A.15: Ver calificaciones alumnos y estadísticas de calificaciones

Si se quieren obtener las notas de los alumnos, el profesor debe clicar sobre el enlace ”Ir a las calificaciones de los alumnos” que aparece sobre los detalles del cuestionario en la pantalla mostrada en la figura A.14 (página 85). En la figura A.15 (página 86) se pueden ver las notas de los alumnos y un gráfico que muestra el porcentaje de personas que ha obtenido una determinada calificación.

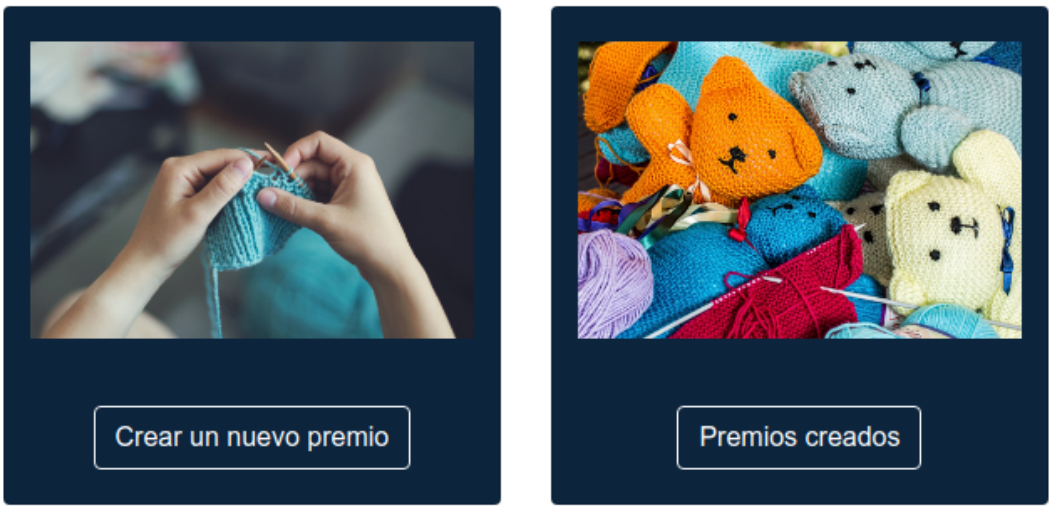


Figura A.16: Menú premios de un profesor

Clicando sobre el bloque ”Premios” (del menú A.2 (página 76)), el profesor accede a la

pantalla mostrada en la figura A.16 (página 86). Como se puede ver, esta pantalla ofrece 2 funcionalidades: ”Crear un nuevo premio” y ”Premios creados”.

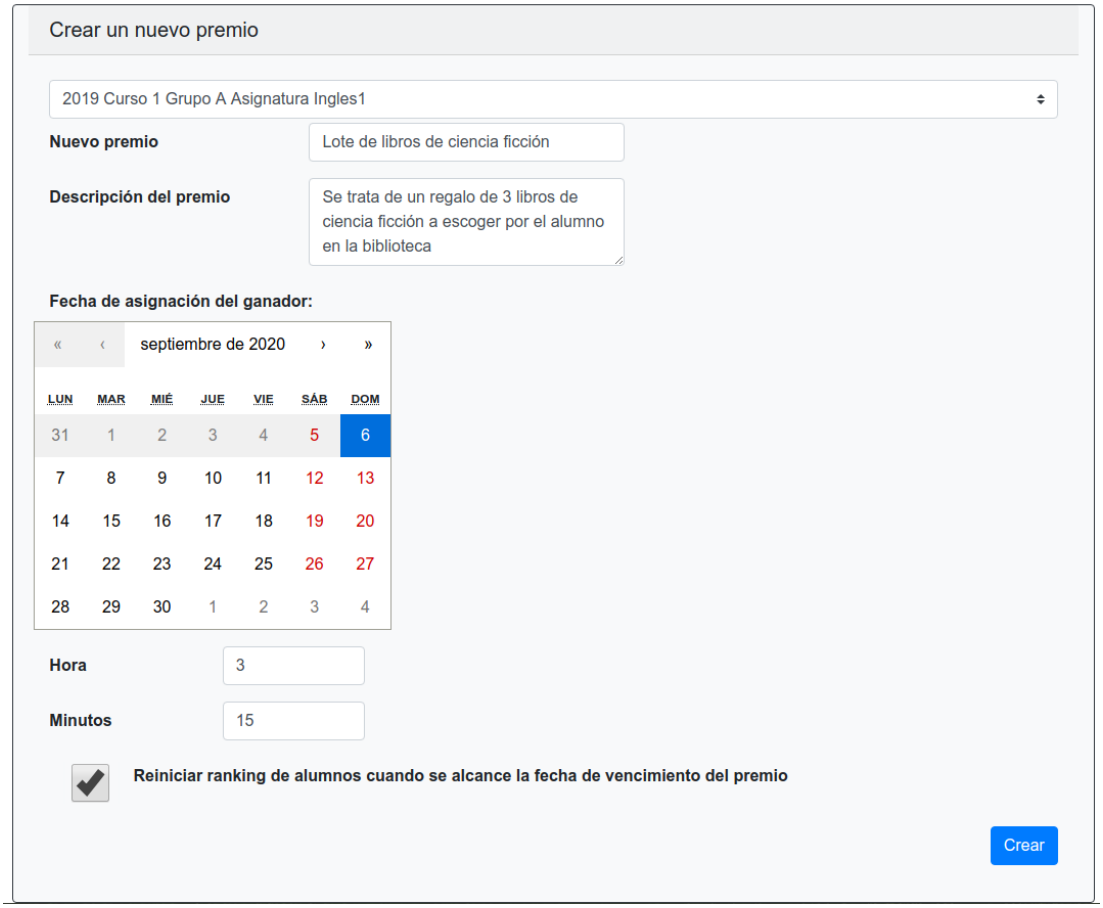
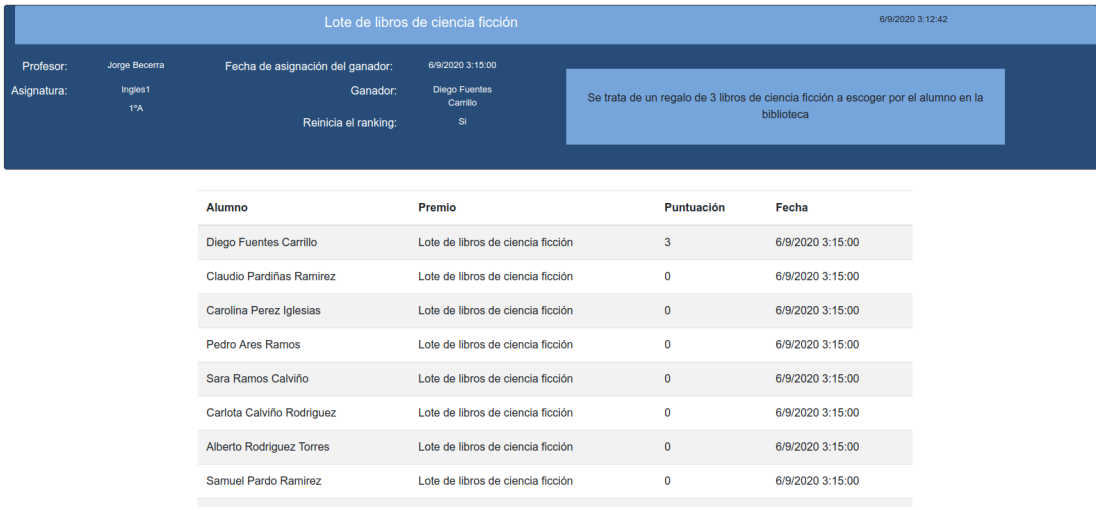


Figura A.17: Crear un nuevo premio

Clicando sobre el botón ”Crear un nuevo premio”, el profesor accede al formulario que se muestra en la figura A.17 (página 87). La casilla ”Reiniciar ranking de alumnos cuando se alcance la fecha de vencimiento del premio” puede ser marcada por el profesor si quiere que los alumnos vuelvan a tener el marcador de puntos a 0 una vez se alcance la fecha en la cual se asigna un ganador.

Figura A.18: Premios creados por el profesor

Clicando sobre el botón ”Premios creados” (del menú de premios A.16 (página 86)), el profesor accede a la pantalla mostrada en la figura A.18 (página 88). En ella se muestran los premios que ha creado. Como se puede ver, si un premio ya tiene asignado un ganador, éste es mostrado.



Alumno	Premio	Puntuación	Fecha
Diego Fuentes Carrillo	Lote de libros de ciencia ficción	3	6/9/2020 3:15:00
Claudio Pardiñas Ramirez	Lote de libros de ciencia ficción	0	6/9/2020 3:15:00
Carolina Perez Iglesias	Lote de libros de ciencia ficción	0	6/9/2020 3:15:00
Pedro Ares Ramos	Lote de libros de ciencia ficción	0	6/9/2020 3:15:00
Sara Ramos Calviño	Lote de libros de ciencia ficción	0	6/9/2020 3:15:00
Carlota Calviño Rodriguez	Lote de libros de ciencia ficción	0	6/9/2020 3:15:00
Alberto Rodriguez Torres	Lote de libros de ciencia ficción	0	6/9/2020 3:15:00
Samuel Pardo Ramirez	Lote de libros de ciencia ficción	0	6/9/2020 3:15:00

Figura A.19: Detalles de un premio y su ranking

Clicando sobre el nombre de un premio, se accede a los detalles de dicho premio. Esto se muestra en la figura A.19 (página 88). Como se puede ver, junto con los detalles del premio, se muestra el ranking de alumnos para dicho premio.

A.2 Perfil alumno

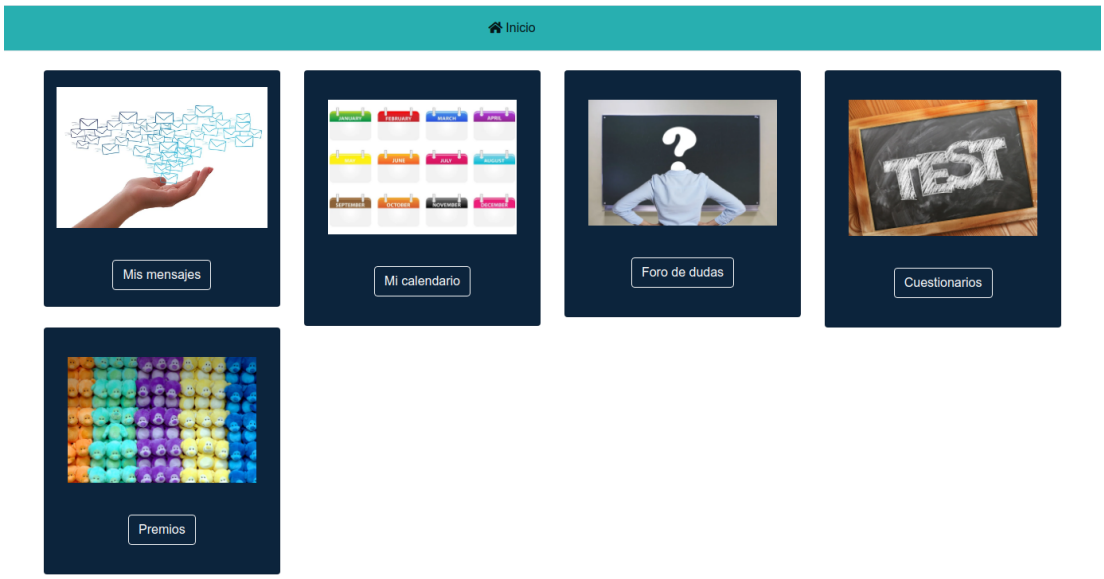


Figura A.20: Menú principal del alumno

En la figura A.20 (página 89) se muestra el menú principal que ve un alumno nada más loguearse. Desde él puede acceder a cada uno de los bloques de funcionalidades: ”Mis mensajes”, ”Mi Calendario”, ”Foro de dudas”, ”Cuestionarios” y ”Premios”.

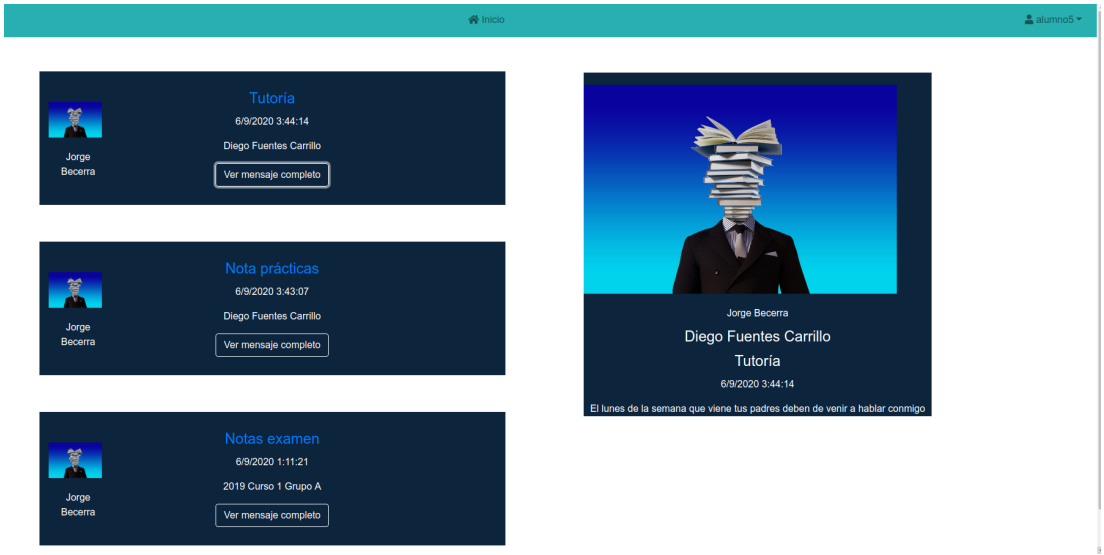


Figura A.21: Mensajes recibidos por un alumno

Clicando sobre el bloque ”Mis mensajes” (del menú A.20 (página 89)), el alumno accede a los mensajes que ha recibido, tanto los dirigidos para él, como los dirigidos para algún grupo al que pertenece o ha pertenecido. Como se muestra en la figura A.21 (página 89), el alumno puede visualizar la lista de mensajes recibidos y acceder a los detalles de alguno de ellos presionando sobre el botón ”Ver mensaje completo”.

Clicando sobre el bloque ”Mi Calendario” (del menú A.20 (página 89)), el alumno accede a un calendario idéntico al de un profesor (mostrado en la figura A.6 (página 79)). A través de este, puede acceder a los eventos de un determinado día clicando sobre una de las fechas.

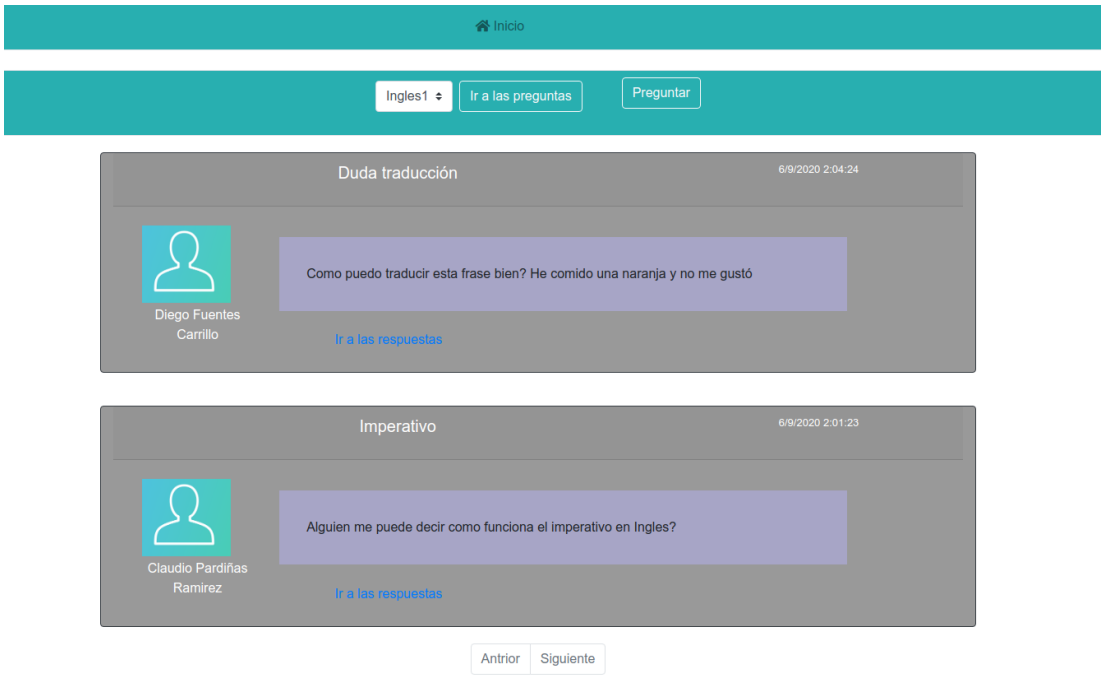


Figura A.22: Mensajes recibidos por un alumno

Clicando sobre el bloque ”Foro de dudas” (del menú A.20 (página 89)), el alumno accede a la pantalla mostrada en la figura A.22 (página 90). Como se puede ver, en la parte superior de la pantalla hay una barra. Esta barra nos ofrece la posibilidad de buscar los foros de las diferentes asignaturas y crear una nueva pregunta. Clicando sobre el botón ”Preguntar”, se accede al formulario mostrado en la figura A.23 (página 91)

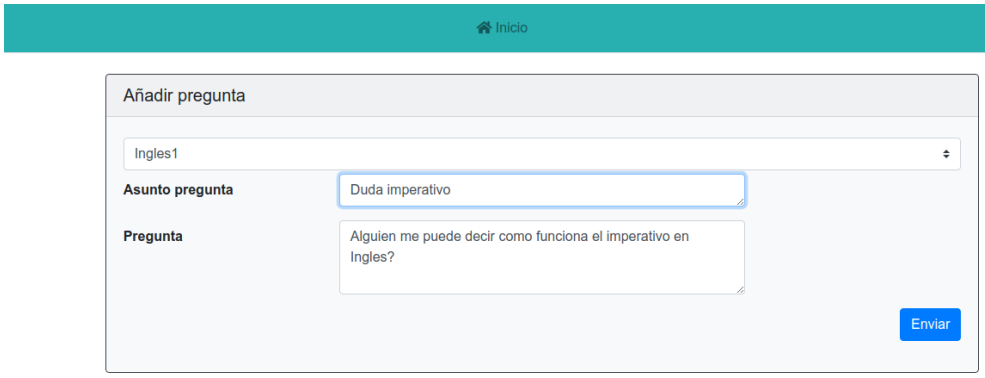


Figura A.23: Formulario para realizar una pregunta en el foro

Figura A.24: Respuestas a una determinada pregunta del foro

Clicando sobre ”Ir a las respuestas”, se accede a la pantalla mostrada en la figura A.24 (página 91). Desde esta pantalla se puede responder a la pregunta, clicando sobre el botón ”Responder”.

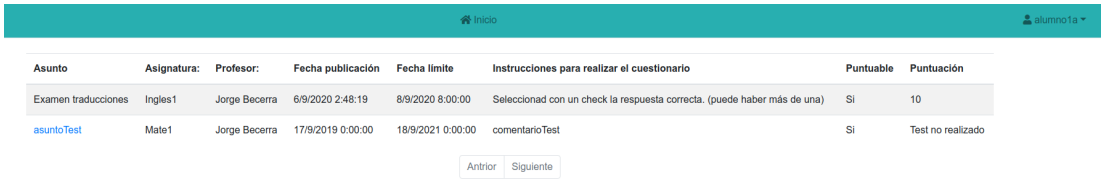


Figura A.25: Cuestionarios recibidos por un alumno

Clicando sobre el bloque ”Cuestionarios” (del menú A.20 (página 89)), el alumno accede a la pantalla mostrada en la figura A.25 (página 92). Desde esta pantalla, el alumno puede ver los cuestionarios recibidos de sus profesores y acceder a alguno de ellos clicando sobre el nombre del cuestionario. Una vez hace clic, el alumno accede a la pantalla mostrada en la figura A.26 (página 92). En esta pantalla, el alumno deberá responder a las preguntas y pulsar en ”Enviar cuestionario”.

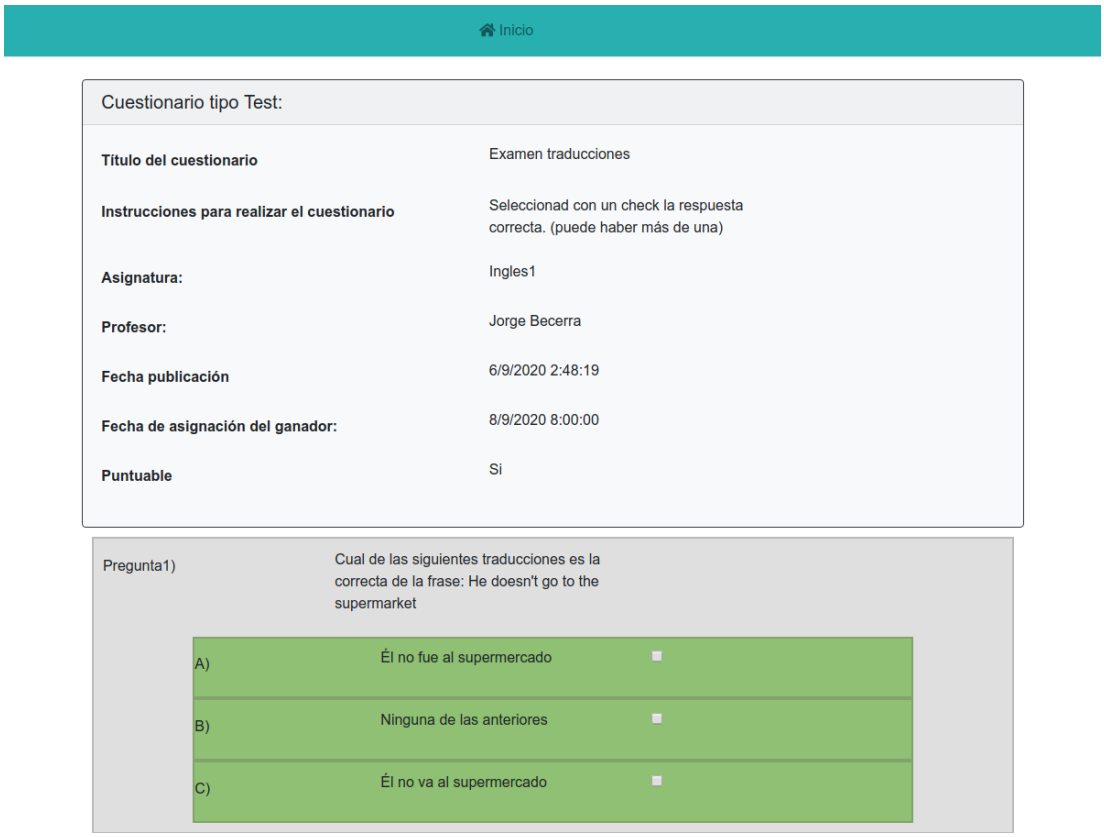


Figura A.26: Cuestionario para ser rellenado por el alumno

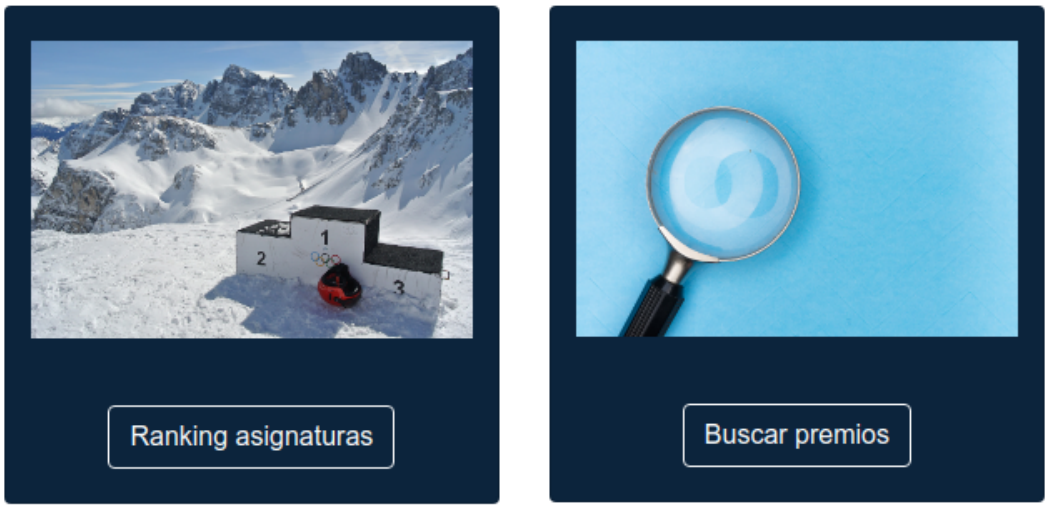
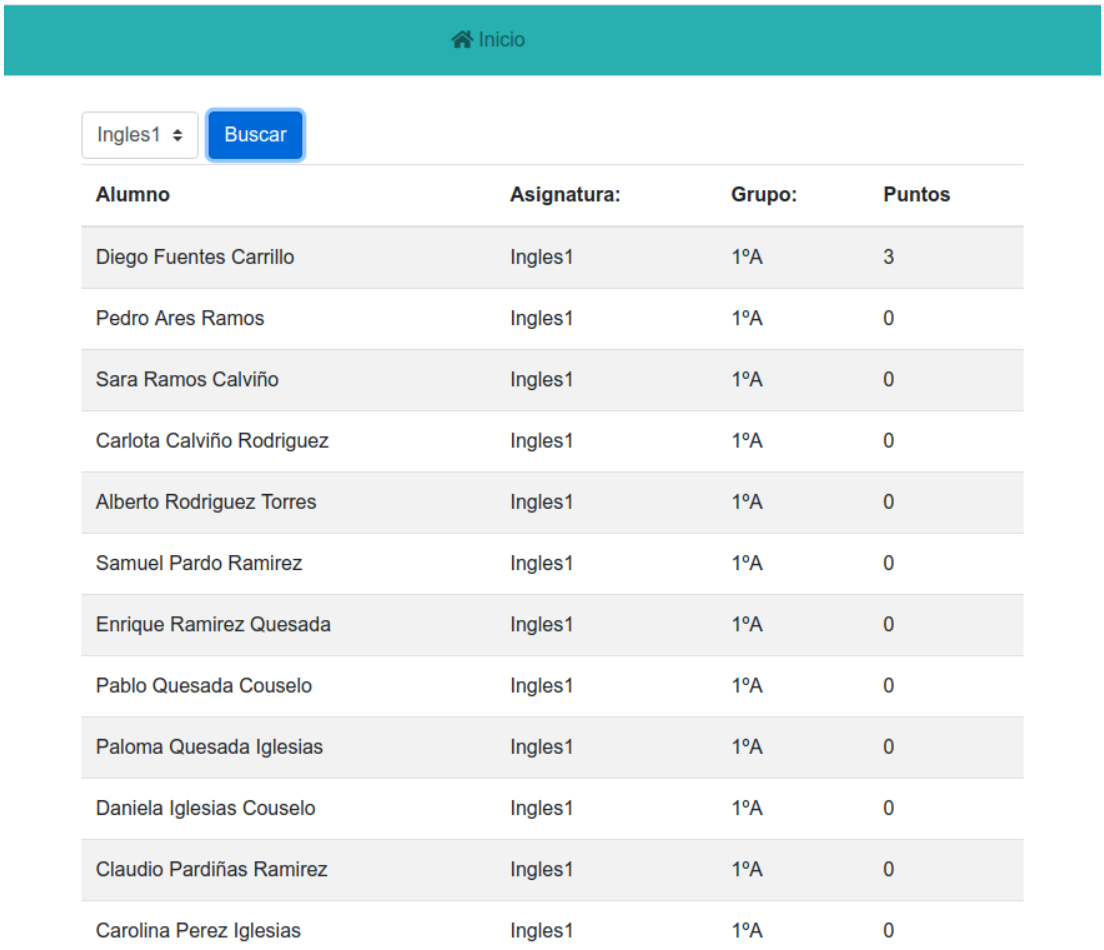


Figura A.27: Menú premios del alumno

Clicando sobre el bloque ”Premios” (del menú A.20 (página 89)), el alumno accede al menú mostrado en la figura A.27 (página 93). Este menu’ofrece dos funcionalidades: ”Buscar premios” y ”Ranking asignaturas”.

Dando clic sobre ”Buscar premios”, el alumno accede a una pantalla muy similar a la mostrada en la figura A.18 (página 88). En ella se mostrarían los premios asociados con alguno de los grupos a los que pertenece el alumno.



Alumno	Asignatura:	Grupo:	Puntos
Diego Fuentes Carrillo	Ingles1	1ºA	3
Pedro Ares Ramos	Ingles1	1ºA	0
Sara Ramos Calviño	Ingles1	1ºA	0
Carlota Calviño Rodriguez	Ingles1	1ºA	0
Alberto Rodriguez Torres	Ingles1	1ºA	0
Samuel Pardo Ramirez	Ingles1	1ºA	0
Enrique Ramirez Quesada	Ingles1	1ºA	0
Pablo Quesada Couselo	Ingles1	1ºA	0
Paloma Quesada Iglesias	Ingles1	1ºA	0
Daniela Iglesias Couselo	Ingles1	1ºA	0
Claudio Pardiñas Ramirez	Ingles1	1ºA	0
Carolina Perez Iglesias	Ingles1	1ºA	0

Figura A.28: Ranking de alumnos para una determinada asignatura

Dando clic sobre ”Ranking asignaturas”, el alumno accede a la pantalla mostrada en la figura A.28 (página 94). En la parte superior de esta pantalla hay un selector donde el alumno puede escoger una de sus asignaturas y asi’visualizar por pantalla el ranking de esa asignatura.

Lista de acrónimos

API Application Programming Interface. 6

CSS Cascading Style Sheets. 6

DAO Data Access Object. 27

DTO Data Transfer Object. 28

HTML HyperText Markup Language. 5

HTTP Hypertext Transfer Protocol. 16

IDE Integrated Development Environment. 8

JPA Java Persistence API. 6

JSON JavaScript Object Notation. 7

LMS Learning Management Systems. 9

REST REpresentational State Transfer. 7

SQL Structured Query Language. 52

Bibliografía

[1] "Javascript." [En línea]. Disponible en: https://javascript.info/

[2] "Html." [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTML

[3] "Css." [En línea]. Disponible en: https://www.w3schools.com/css/

[4] "Boostrap." [En línea]. Disponible en: https://www.w3schools.com/bootstrap4/default.asp

[5] "react-star-ratings." [En línea]. Disponible en: https://www.npmjs.com/package/react-star-ratings

[6] "react-toastify." [En línea]. Disponible en: https://www.npmjs.com/package/react-toastify

[7] "react-calendar." [En línea]. Disponible en: https://www.npmjs.com/package/react-calendar

[8] "Junit." [En línea]. Disponible en: https://junit.org/junit5/

[9] "Spring-boot." [En línea]. Disponible en: https://spring.io/projects/spring-boot

[10] "React." [En línea]. Disponible en: https://es.reactjs.org/

[11] "Redux." [En línea]. Disponible en: https://es.redux.js.org/

[12] "Maven." [En línea]. Disponible en: https://maven.apache.org/

[13] "Npm." [En línea]. Disponible en: https://www.npmjs.com/

[14] "Postman." [En línea]. Disponible en: https://www.postman.com/

[15] "Git." [En línea]. Disponible en: https://git-scm.com/

[16] "Chamilo." [En línea]. Disponible en: https://campus.chamilo.org/index.php?include=Manuales_spanish.html

[17] "Fedena." [En línea]. Disponible en: https://fedena.com/