ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN BACKEND PARA LA COMUNICACIÓN ENTRE CENTROS EDUCATIVOS Y PADRES DE ALUMNOS II

Alumno

Daniel Romero Calero

Grado en Ingeniería Informática (Tecnologías de la información)

Tutor

Dr. Alexis Quesada Arencibia

Ciencias de la Computación e Inteligencia Artificial

Ver.12 21/05/18

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D/Dª Daniel Romero Calero autor del Trabajo de Fin de Título Análisis, diseño e implementación de un backend para la comunicación entre centros educativos y padre de alumnos II correspondiente a la titulación Grado en Ingeniería Informática

S O L I C I T A

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

[ ] Se ha iniciado o hay intención de iniciarlo (defensa no pública).
[X] No está previsto.

Y para que así conste firma la presente.

Las Palmas de Gran Canaria, a 12 de Julio de 2018

El estudiante

Fdo.: ROMERO CALERO, DANIEL (FIRMA)

Firmado digitalmente por ROMERO CALERO, DANIEL (FIRMA)
Nombre de reconocimiento (DN): c=ES, serialNumber=45366193G, sn=ROMERO, givenName=DANIEL, cn=ROMERO CALERO, DANIEL (FIRMA)
Fecha: 2018.07.12 19:31:23 +01'00'

A rellenar y firmar **obligatoriamente** por el/los tutor/es

En relación a la presente solicitud, se informa:

[X] Positivamente

[ ] Negativamente
(la justificación en caso de informe negativo deberá incluirse en el TFT05)

QUESADA ARENCIBIA, FRANCISCO ALEXIS (FIRMA)

Fdo.:

Firmado digitalmente por QUESADA ARENCIBIA, FRANCISCO ALEXIS (FIRMA)
Nombre de reconocimiento (DN): c=ES, serialNumber=42870168P, sn=QUESADA, givenName=FRANCISCO ALEXIS, cn=QUESADA ARENCIBIA, FRANCISCO ALEXIS (FIRMA)
Fecha: 2018.07.13 12:43:30 +01'00'

DIRECTOR DE LA ESCUELA DE INGENIERÍA INFORMÁTI

Trabajo de Fin de Grado en Ingeniería Informática (intensificación en Tecnologías de la Información) de la Universidad de Las Palmas de Gran Canaria presentado por el alumno:

Daniel Romero Calero

Título del proyecto

Análisis, diseño e implementación de un *backend* para la comunicación entre centros educativos y padres de alumnos II

Tutor

Dr. Alexis Quesada Arencibia

Agradecimientos

A mi tutor Dr. Alexis Quesada Arencibia por siempre estar disponible ante cualquier duda y ayudarnos en todo lo posible.

A mi compañero Héctor por haberme animado y acompañado en el transcurso del grado.

A mi familia, novia y amigos por darme ánimos y apoyo durante la realización del grado.

Resumen

Este TFT proporciona una herramienta informática para facilitar la comunicación entre centros escolares y los tutores legales de los estudiantes. Actualmente, muchos centros siguen usando métodos convencionales para enviar todo tipo de mensajes informativos y comunicaciones (circulares, encuestas y autorizaciones). El empleo de estos métodos conlleva generalmente un gasto ingente de recursos (papel, tinta, …).

Para evitar los inconvenientes de los métodos tradicionales y agilizar estas comunicaciones, se ha desarrollado el *backend* de un sistema de gestión de comunicaciones del centro que permitirá el intercambio de información de una manera más económica, fácil, segura y rápida. De esta forma, los tutores legales podrán recibir comunicaciones de la secretaría de los centros y de los profesores tutores, por ejemplo, para concretar una tutoría.

Abstract

This TFT offers a tool to facilitate communication between schools and legal guardians of students. Currently, many centers continue to use conventional methods to send all kinds of informative messages and communications (circulars, polls and authorizations). The use of these methods usually involves a huge expenditure of resources (paper, ink, ...).

To avoid the inconveniences of traditional methods and speed up these communications, a backend has been developed for communications management of the center that will allow the exchange of information in a more economical, easy, safe and fast way. Thus, legal guardians can receive communications from the secretariat of the centers and from the tutor teachers, for example, to establish a tutorial.

Índice

Índice de Ilustraciones

Índice de Tablas

Introducción

Este proyecto se ha desarrollado a raíz de otro proyecto denominado “ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN BACKEND PARA LA COMUNICACIÓN ENTRE CENTROS EDUCATIVOS Y PADRES DE ALUMNOS” realizado por D. Adrián Louro Alonso y tutorizado por Dr. Alexis Quesada Arencibia. Los principios de dicho proyecto eran hacer desaparecer los sistemas convencionales de comunicación entre los centros educativos y los padres, evitando así el malgasto de recursos a la hora de enviar mensajes a los padres como pueden ser circulares, encuestas y autorizaciones.

El objetivo que tiene este TFG es proporcionar al centro educativo y a los padres una comunicación directa mediante un sistema de gestión de tutorías, ahorrando tiempo y papel. Para lograr este objetivo se ha tenido que implementar este nuevo sistema de gestión en el *backend* ya desarrollado en el TFG mencionado anteriormente.

Este sistema de tutorías cuenta con una aplicación móvil que es usada por el padre/madre/tutor legal y por los profesores. Esta aplicación se complementa con el desarrollo realizado en paralelo en otro TFG titulado “ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN PARA DISPOSITIVOS MÓVILES MULTIPLATAFORMA PARA LA COMUNICACIÓN ENTRE CENTROS EDUCATIVOS Y PADRES DE ALUMNOS II” realizado por D. Héctor Francisco González Feo. En esta aplicación los padres podrán solicitar citas a los profesores y los profesores podrán definir sus horarios de tutoría y solicitar citar a los padres, a través de una aplicación móvil.

La aplicación móvil es la encargada de gestionar las vistas que conlleva la gestión de tutorías, mientras que este proyecto se centra en desarrollaras los servicios que debe ofrecer el *backend*, y así ofrecer los recursos necesarios para que la aplicación móvil pueda realizar dicha gestión.

Para incluir esta funcionalidad en el sistema se estudiará el estado actual de ambas aplicaciones y luego se definirán e implementarán las actualizaciones necesarias en el sistema para llevar a cabo esta tarea.

A partir de este apartado se hará uso del término “tutor legar” para referirse a la terminología “padre/madre/tutor legal/tutora legal” y el término “profesor” para “profesor/profesora” con el fin de estandarizar estos términos.

Estructura del documento

Esta memoria está constituida por once capítulos. A continuación, se explicará brevemente el contenido de cada uno:

- Capítulo 1 – Estado actual y objetivos iniciales:
  En este capítulo se explica el estado actual del proyecto titulado "Análisis, diseño e implementación de un *backend* para la comunicación entre centros educativos y padres de alumnos". Además, se presenta las alternativas existentes y se explican los objetivos de esta segunda fase del proyecto.

- Capítulo 2 – Justificación de las Competencias Específicas Cubiertas:
  En este capítulo se nombra y se justifican las competencias que cumple este proyecto.

- Capítulo 3 - Aportaciones:
  En este capítulo se exponen algunas aportaciones personales y socio-económicas descubiertas hasta el momento.

- Capítulo 4 – Normativa y legislación:
  En este capítulo se analiza la normativa y legislación vigente para que este proyecto pueda cumplir estos aspectos. Y también se muestra un breve resumen de las licencias utilizadas y las herramientas que hacen uso de esas licencias.

- Capítulo 5 – Metodología de trabajo y planificación del proyecto:
  En este capítulo se define la metodología de trabajo escogida, así como la planificación inicial y los ajustes que hicieron falta.

- Capítulo 6 – Tecnologías y herramientas utilizadas:
  En este capítulo se describen todas las herramientas y tecnologías utilizadas durante el desarrollo del proyecto.

- Capítulo 7 - Análisis:
  En este capítulo se hace un breve resumen de la fase de análisis del proyecto titulado "Análisis, diseño e implementación de un *backend* para la comunicación entre centros educativos y padres de alumnos" y de los objetivos que se cubrieron en dicho proyecto. Además, se explica la nueva fase de análisis explicando los nuevos requisitos requeridos por el sistema, esto se representa mediante casos de uso y sus especificaciones de casos de uso para definir los requerimientos de una manera más clara.

- Capítulo 8 - Diseño:
  En este capítulo se explica la fase de diseño del proyecto. Esta fase está compuesta por la arquitectura del sistema, el diseño de la base de datos y el diseño de la interfaz de la aplicación, en donde se justifican y explican dichos aspectos.

- Capítulo 9 - Desarrollo:
  En este capítulo se explica la fase de desarrollo del proyecto. Se expone y justifica la elección de los *frameworks* de programación utilizados para el desarrollo del proyecto ,además de definir las distintas capas de las que consta el sistema y la seguridad implementada a la hora de realizar peticiones y obtener respuestas de los recursos disponibles.

- Capítulo 10 – Resultados, conclusiones y trabajo futuro:
  En este capítulo se habla de los resultados obtenidos y las conclusiones después de haber finalizado el proyecto, además de definir algunos trabajos futuros vinculados a este proyecto.

- Bibliografía y anexos
  Se termina la memoria con el apartado que recoge las referencias empleadas durante el desarrollo del proyecto y dos anexos:

    - Anexo I: se generará un manual de usuario en donde se explicará detenidamente como se utilizan las distintas opciones del portal web.
    - Anexo II: se generará una pequeña guía para explicar cómo instalar y montar el sistema en un servidor.

Capítulo 1: Estado actual y objetivos iniciales

1.1. Estado actual

Actualmente, la gran mayoría de los centros educativos utilizan un sistema muy deficiente para el envío de mensajes informativos a los tutores legales de los alumnos. Asimismo, la comunicación entre profesores y tutores legales se vuelve tediosa al depender muchas veces del alumno para que este le entregue un papel informativo al tutor legal.

Las deficiencias de los centros que funcionan bajo este sistema son las siguientes:

- Mayor impacto económico respecto a la gran cantidad de impresos informativos y comunicativos que genera el centro.
- El flujo de la comunicación entre el profesor y el tutor legal, dado que el alumno es el intermediario.
- El tiempo que tarde en llegar una cita del profesor al tutor legal no siempre pudiéndole llegar.
- El tiempo que tiene que malgastar el tutor legal asistiendo al centro a pedir una cita con el profesor o el tiempo que puede estar al teléfono intentado localizarlo.

1.1.1. Alternativas existentes

Para que esta aplicación tenga éxito hay que investigar el mercado actual y estudiar las opciones que se ofrecen contrastándolas con la aplicación que va a ser desarrollada en este proyecto, para poder tener un buen producto con el que poder competir.

A continuación, se mostrarán las principales características de las aplicaciones que intentan solventar el problema que ataña este TFG:

- **ApliAula [1]**
  - Centro: permite crear cursos, profesorado, asignaturas, reservas del comedor, enviar mensajes a los tutores legales y a los profesores.
  - Tutor legal: permite ver un histórico de notas de los alumnos, reservas del comedor y recibir mensajería interna del centro.
  - Profesores: permite pasar lista, control de ausencias y control de deberes.
- **Educamos [2]**
  - Centro: permite envío de circulares y autorizaciones.
  - Tutor legal: permite justificar incidencias y calificaciones, confirmar y denegar autorizaciones, seguir la ruta del autobús, ver recibos, inscribir a los hijos/as a las actividades del colegio.
  - Profesores: permite concertar, aceptar y rechazar entrevistas y tutorías.
- **Educanlia [3]**
  - Centro: permite él envió de comunicados (faltas, eventos, autorizaciones), gestión de las aulas, gestión de los horarios de las clases y asignación del profesorado, gestión de asignaturas y profesores en cada aula.
  - Tutor legal: tiene servicio de pago, permite recibir evaluaciones de los hijos/as y firmarlas, información sobre las tareas que tienen que hacer los hijos, gestión de entrevistas con el profesorado, recibir faltas, alertas y comunicados y autorizar notificaciones.
  - Profesores: permite gestión de asistencia de los alumnos, programación de tareas pendientes, gestión de una agenda y gestión de tareas en el aula.

- **Esemtia [4]**
    - Centro: gestión de expedientes, matriculación, facturación, gestión de servicios (extraescolares, comedor, autocar), centro de mensajería (encuestas, circulares, autorizaciones).
    - Tutor legal: permite justificar faltas, información económica, recibir mensaje del centro, información sobre el comedor, revisión de los comentarios y notas de actitud de los hijos/as y recepción de mensajes del profesorado.
    - Profesores: seguimiento pedagógico de los alumnos, gestión del aula y envió de mensajes a los tutores legales.
- **Dinantia [5][6]**
    - Centro: permite realizar preguntas a los padres, solicitar autorizaciones, envío de comunicados, comunicación interna y ofrecer menú escolar.
    - Tutor legal: ver calendario de acontecimientos, ver menú escolar, seguimiento diario del aula.
    - Profesores: pasar la lista y notificar las ausencias automáticamente a los padres.

1.1.2. Conclusiones

Una vez concluido el análisis del mercado actual, se puede observar como todas las funcionalidades propuestas por el sistema que van a ser desarrolladas en este proyecto están cubiertas, aunque también hay que destacar que ninguna aplicación de las mencionadas anteriormente las cubre todas.

Cabe resaltar que ninguna de estas aplicaciones tiene una opción de autoimportar un curso. Esta funcionalidad ha sido desarrollada en la primera fase del proyecto y puede resultar muy útil para el centro.

Para concluir, se ha visto que las aplicaciones estudiadas anteriormente están más enfocadas a la gestión interna del centro y tener un control más exhaustivo de los alumnos con funcionalidades como información sobre las tareas que tienen que hacer los alumnos, control de asistencia de los alumnos, envió de calificaciones de los alumnos entre otros. Por otro lado, la mayoría de estas aplicaciones necesitan de una gran infraestructura para su correcto funcionamiento lo que puede llegar a ser muchas veces un hándicap.

En cambio, la aplicación desarrollada en este proyecto se basa en la sencillez y facilidad con la que el centro puede enviar mensajes a los tutores legales y como estos pueden concretar citas con el profesor de sus hijos de una manera rápida y eficaz.

1.2. Objetivos

Los objetivos de este proyecto son bastante claros. Estos objetivos se basan en aportar al problema una solución sencilla y eficaz. Para ello se partirá del desarrollo implementado en la primera fase del proyecto, donde el sistema permitía al centro comunicarse directamente con los tutores legales de los alumnos sin tener como intermediarios a los alumnos.

El sistema hasta ahora permite enviar circulares informativas, encuestas y autorizaciones de actividades extraescolares y que el centro pueda ver las respuestas de los tutores legales de una manera global por cada mensaje.

En la segunda fase del proyecto el objetivo principal es desarrollar un sistema de gestión de tutorías entre tutores legales y profesores para evitar el malgasto de recursos innecesarios y la pérdida de tiempo que conlleva la solicitud de una cita con el profesor y viceversa.

Además, este proyecto pretende que este sistema se pueda instaurar en centros con pocos recursos dado que el sistema cuenta con una infraestructura mínima para su puesta en marcha.

Capítulo 2: Justificación de las Competencias Específicas Cubiertas

2.1. Trabajo Fin de Grado

Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

Este proyecto ha sido desarrollado a raíz de otro titulado "*Análisis, diseño e implementación de un backend para la comunicación entre centros educativos y padres de alumnos"* por tanto, nos hemos tenido que adaptar al desarrollo de la primera fase y por consiguiente actualizar el sistema de la primera etapa del proyecto añadiendo nuevas funcionalidades y aplicando los conocimientos y competencias adquiridas a lo largo del grado. Esta competencia se adquirirá una vez el tribunal correspondiente evalué satisfactoriamente este trabajo de fin de grado.

2.2. Comunes a la ingeniería informática

CII01

Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

Durante el desarrollo de este proyecto se ha tenido en cuenta la protección de la información y el uso de herramientas adecuadas para asegurar la fiabilidad, seguridad y calidad del producto. Las herramientas utilizadas, fueron estudiadas detenidamente y la información está controlada para que solo pueda tener acceso quien se encuentre autorizado, cumpliendo la legislación y normativa vigente.

CII03

Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.

Este proyecto está vinculado con otro titulado "*Análisis, diseño e implementación de una aplicación para dispositivos móviles multiplataforma para la comunicación entre centros educativos y padres de alumnos II".* Por tanto, durante el desarrollo la comunicación ha sido imprescindible para abarcar la propuesta de desarrollo en esta fase.

CII08

Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.

En la primera fase del proyecto, se decidió usar el *framework* web *Symfony* para desarrollar la propuesta definida en ese momento. En esta segunda fase, se ha respetado *Symfony* como *framework* de desarrollo para el *backend* y se ha escogido una nueva tecnología como es Angular para desarrollar el *frontend*, ya que ambas tecnologías están muy bien documentadas.

CII012

Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos.

Partiendo de un diseño de base de datos propuesto en la primera fase de este proyecto, se ha respetado dicho diseño además de realizar las actualizaciones necesarias para poder desarrollar las nuevas funcionalidades definidas en esta fase. En esta segunda etapa, también se ha decido utilizar una serie de librerías que permiten tratar los datos de la base de datos como objetos, lo que ha facilitado el tratamiento de los datos.

CII013

Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en web.

El proyecto ha sido desarrollado con dos tecnologías totalmente independientes, por tanto, se han tenido que crear una serie de servicios desde el *backend* que serán usados por el *frontend* del portal web y por la aplicación móvil desarrollada paralelamente en otro TFG, con el fin de lograr una comunicación satisfactoria entre ambos.

CII016

Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.

En la primera fase del proyecto se decidió utilizar una metodología de desarrollo basada en prototipos, por tanto, se respetará esta metodología y se definirán las nuevas iteraciones, así como las distintas fases del ciclo de vida del software.

CII017

Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

Durante el desarrollo del *frontend* se ha tenido en cuenta muchos aspectos a la hora de diseñar la interfaz, para facilitar su usabilidad y garantizar la accesibilidad de las acciones que puede realizar el usuario en cada momento dentro de la aplicación.

CII018

Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

Debido a la seguridad y protección de la información que conlleva este proyecto, se ha realizado un estudio acerca la normativa y legislación vigente que se definirá en el capítulo 4.

2.3. Ingeniería del Software

IS01

Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.

Para realizar esta segunda etapa del proyecto, se ha realizado una planificación según una metodología de desarrollo software implantada en la primera fase de este proyecto. Esto ha permitido seguir un orden y cumplir la propuesta realizada, consiguiendo un software de calidad que responde a las necesidades de los usuarios. Además, se han aplicado los principios de diseño, patrones de diseño y arquitectónicos de la primera fase del proyecto para seguir garantizando un sistema robusto y flexible.

2.4. Tecnologías de la Información

TI06

Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.

En este proyecto se lleva a cabo una interacción entre una aplicación web y una aplicación móvil desarrollada por otro alumno, la cual se comunica con el sistema mediante un servicio web RESTful que se alimenta de los servicios que proporciona el *backend* realizado por la aplicación web.

Capítulo 3: Aportaciones

3.1. Entorno socio-económico

Este proyecto hace énfasis en el malgasto de papel de los centros a la hora de enviar circulares, encuestas y autorizaciones a los tutores legales de los alumnos. Además del impacto económico que esto conlleva. Por ello se ha desarrollado un sistema para gestionar los mensajes que enviaba el centro a los tutores legales para evitar las controversias del malgasto de papel a nivel mundial.

Esta segunda fase del proyecto persigue incluir en la plataforma la comunicación entre tutores legales y profesores. Cuando un profesor quiere comunicarse con un tutor legal, la acción se realiza a través de un papel que le entrega el alumno al tutor legal o mediante una llamada telefónica realizada por el centro. Para evitar tanto el gasto que supone una llamada, como el gasto que supone que el alumno le entregue un papel al tutor legal con la solicitud de una tutoría con el profesor, se ha incluido un sistema de gestión de tutorías entre profesores y tutores legales. Con esta implementación en el sistema, además de evitar realizar llamadas y el malgasto de papel, también hay que contemplar un factor muy importante que es el tiempo, con este sistema bastará con que el profesor defina sus horarios de tutoría para que los tutores legales puedan solicitarlas y así evitar que:

- El profesor tenga que solicitar una tutoría entregándole un papel al alumno.
- Tutores legales y profesores se estén llamando unos a otros.
- El padre no localice al profesor o viceversa.
- Los tutores legales tengan que ir al centro para solicitar una cita con el profesor.

3.2. Personal

Este proyecto ha sido una oportunidad para poner a prueba los conocimientos adquiridos a lo largo de la carrera. Hay que destacar que este proyecto ha sido desarrollado a partir de otro, por tanto, se ha tenido que estudiar el estado actual sistema para entender la filosofía y estructura del proyecto y poder mejorar y aumentar su funcionalidad, aprovechando así al máximo el desarrollo realizado en la primera fase.

Por lo que respecta al desarrollo, ha sido bastante interesante estudiar las herramientas con las que se desarrolló la primera fase y debatir si se seguían usando las mismas o se optaba por otras alternativas. En cuanto al *backend* se decidió seguir usando *Symfony 3*, *Doctrine* y *PHP 7.1* para provechar los servicios que estas herramientas ofrecían, por lo que hubo que aprender a utilizar dichas tecnologías. Adicionalmente, el *frontend* fue desarrollado desde cero para que las vistas fueran más dinámicas por lo que se decidió utilizar *Angular*. Aquí también hubo un aprendizaje y un refuerzo de conocimientos aportados en la carrera.

Para concluir, como hemos comentado este proyecto está vinculado con otro y por tanto ha sido un factor muy importante la comunicación entre ambos desarrolladores, lo que ha permitido realizar un análisis y diseño más claro y conciso.

Capítulo 4: Metodología de trabajo y planificación del proyecto

4.1. Metodología de trabajo

En la primera fase de este proyecto, se decidió escoger una metodología de trabajo que se basa en el modelado de ciclo de vida del software basado en prototipos, por tanto, en esta segunda etapa se seguirá utilizando esta metodología por el hecho de ir validando continuamente los prototipos realizados con el tutor.

Esta metodología [18], está enfocada a desarrollar un diseño rápido centrándose en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación, gracias a ésta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo.

A continuación, en la *Ilustración 1* se muestra el flujo de trabajo de la metodología utilizada en este TFG:

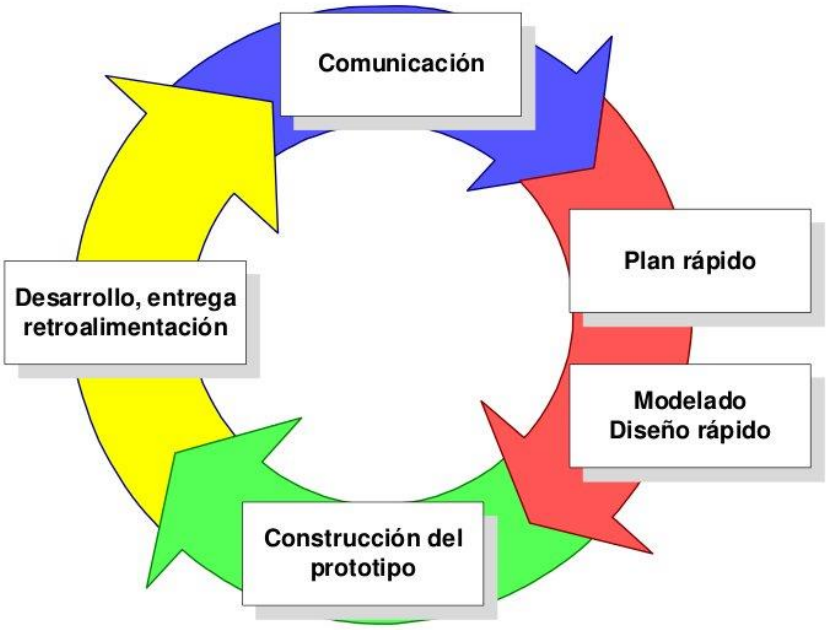


Ilustración 1: Metodología basada en prototipos

4.2. Planificación del proyecto

Inicialmente se realizó una aproximación sobre la planificación del proyecto como se puede observar en la tabla 1:

Estudio previo / Análisis	50	Tarea 1.1: Estudio de las nuevas funcionalidades
		Tarea 1.2: Obtención de requisitos del sistema
		Tarea 1.3: Prototipo de la interfaz de usuario del sistema
Diseño / Desarrollo / Implementación	170	Tarea 2.1: Tarea 2.1: Actualización de la estructura del sistema
		Tarea 2.2: Diseño de la actualización de la base de datos
		Tarea 2.3: Aprendizaje de *PHP7*, *Symfony3* y *Doctrine*
		Tarea 2.4: Implementación de prototipos de la aplicación web
Evaluación / Validación / Prueba	30	Tarea 3.1: Pruebas de funcionamiento
		Tarea 3.2: Pruebas de interacción entre aplicaciones
Documentación / Presentación	50	Tarea 4.1: Realización de la memoria
		Tarea 4.2: Realización de la presentación

Tabla 1: Planificación inicial

Durante el transcurso del proyecto han surgido una serie de contratiempos, dado que se han tenido que estudiar las herramientas y tecnologías usadas en el desarrollo. Además, se ha tenido que dedicar más tiempo en la fase de análisis porque primero se tuvo que estudiar el estado actual de la herramienta para valorar si los objetivos planteados en la propuesta inicial eran coherentes y tenían una buena aceptación por parte del sistema.

4.3. Pre-análisis

Después de estudiar el estado del proyecto desarrollado en la primera fase, se ha decidido estudiar sus trabajos futuros y realizar un análisis de los mismos. En la primera etapa se determinaron tres actores:

- Administrador
- Tutor legal
- Profesor

De estos tres actores, se incluyeron en el sistema, el Administrador del centro y el Tutor legal. Ahora, en esta segunda fase del proyecto se ha decidido añadir la figura del profesor, por tanto, se ha tenido que analizar y estudiar el cambio y la actualización que requiere el sistema. Para ello, la primera iteración de esta fase del proyecto consistió en estudiar los trabajos futuros propuestos en la primera etapa del proyecto. Estos trabajos futuros, añaden las siguientes funcionalidades al sistema:

- Administradores del centro:
    - Registrar profesores en el centro.
    - Asociar asignaturas a un curso.
    - Establecer asignaturas en el centro.
    - Asociar profesor a una asignatura.
- Tutor legal:
    - Solicitar citas a los profesores de sus hijos.

    - Ver solicitudes de cita de un profesor.
    - Cancelar una cita.
- Profesor:
    - Establecer un horario de tutorías.
    - Solicitar citas a los padres.
    - Ver calendario de citas.
    - Confirmar citas.
    - Cancelar citas.

Una vez analizadas estas funcionalidades, se determinó que algunas de las mismas, mencionadas anteriormente, no tenían mucho sentido implementarlas en esta fase del proyecto por el simple hecho de que no aportaban nada al sistema. Se decidió no implementar:

- Administradores del centro:
    - Asociar asignaturas a un curso.
    - Establecer asignaturas en el centro.
    - Asociar profesor a una asignatura.

En cambio, se decidió implementar otras como:

- Administradores del centro:
    - Asociar profesores a cursos.
    - Promocionar alumnos de un curso a otro.

Una vez realizado este análisis preliminar, se definieron los nuevos casos de uso y las nuevas tablas de especificación de casos de uso para representar de una forma más clara los requisitos funcionales del sistema.

Llegado este punto, se decidido aislar completamente el *backend* del *frontend* y realizar un sistema más versátil, en el sentido de que si en algún momento se quieren usar los servicios ofrecidos por el *backend* en otro dispositivo como puede ser un *Smartwatch*, solo bastaría con desarrollar su *frontend* y hacer uso de los servicios que el *backend* ofrece.

Tras decidir este cambio en el sistema, se escogieron las tecnologías para desarrollar cada parte del mismo. Para el *backend* se escogió *Symfony 3* dado que este fue usado en la primera etapa del proyecto y se pudieron aprovechar todos los servicios implementados hasta el momento, aunque ahora el *backend* se configuro como *API REST,* lo que permite un desacoplamiento total del *backend* y el *frontend*. Por otro lado, se ha desarrollado completamente desde cero el *frontend* con Angular, dado que antes estaba desarrollado en Twig que es el motor de plantillas que viene integrado con Symfony.

Una vez elegidas las herramientas de desarrollo, comenzó la fase de aprendizaje. Este aprendizaje consistió en leer mucha documentación de ambas tecnologías y realizar algunos tutoriales para aprender a desarrollar los servicios que ofrece el *backend* y cómo hacer uso de esos servicios desde el *frontend*.

Concluidas las fases definidas anteriormente, se empezó a desarrollar el nuevo sistema incorporando las funcionalidades que ya estaban implementadas en el sistema anterior. Siguiendo las bases de la metodología por prototipos, se definieron las siguientes iteraciones:

- Iteración 1: en esta iteración se comenzó con la configuración básica del proyecto (tanto el *backend* como el *frontend*). Una vez realizada la configuración básica del *backend*, se importaron al mismo los servicios desarrollados en la primera fase.
- Iteración 2: esta iteración consistió en desarrollar el nuevo *frontend* y hacer uso de los servicios ofrecidos por el *backend*.
- Iteración 3: en esta iteración se desarrollaron los nuevos servicios que ofrece el backend y que fueron mencionados anteriormente.
- Iteración 4: por último, en esta iteración, se realizó un segundo desarrollo del *frontend* de modo que pudiera integrar los nuevos servicios implementados en el *backend*.

Capítulo 5: Tecnologías y herramientas utilizadas

5.1. Herramientas

- *PHPStorm 2018.1*: es un *IDE* para desarrollar en *PHP*, aunque también está preparado para desarrollar en lenguajes como *HTML* y *JavaScript*. Esta herramienta está desarrollada en la plataforma *JetBrains IntelliJ IDEA*. Cuenta con un editor para *PHP*, *HTML y JavaScript*, un depurador y refactorización automática para código *PHP* y *JavaScript* y autocompletado de código.

- *WebStorm 2018.1*: es un *IDE* para desarrollar en *HTML, JavaScript, CSS, SASS, TypeScript* entre otros. Esta herramienta está desarrollada en la plataforma *JetBrains IntelliJ IDEA*. Cuenta con un editor para las tecnologías mencionadas anteriormente, un depurador y refactorización automática para código *HTML, JavaScript, CSS, SASS, TypeScrip*t y autocompletado de código.

- *MySQL Workbench*: es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos *MySQL*.

- *Google Chrome*: navegador web. Cuenta con herramientas de desarrollo para desarrollar y depurar el *frontend* de aplicaciones web.

- *GitHub*: es una plataforma de desarrollo colaborativo de software utilizada para alojar proyectos utilizando el sistema de control de versiones *Git*.

- *StarUML*: es una herramienta que soporta diagramas *UML*.

5.2. Servidores

- *XAMPP*: es un paquete de instalación independiente de plataforma, software libre, que consiste principalmente en el sistema de gestión de bases de datos *MySQL*, el servidor web *Apache* que nos aporta un sitio virtual donde poder alojar nuestro proyecto y ver los cambios en tiempo real que vamos realizando en el desarrollo, además cuenta con los intérpretes para lenguajes de script *PHP* y *Perl*.

5.3. Tecnologías

- *MySQL*: es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual *GPL*/Licencia comercial por *Oracle Corporation* y está considerada como la base datos de código abierto más popular del mundo, sobre todo para entornos de desarrollo web.

- *Symfony 3*: es un *framework* diseñado para optimizar el desarrollo de aplicaciones web basadas en el patrón Modelo-Vista-Controlador. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

- *PHP 7*: es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los 20 primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento *HTML* en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de *PHP* que genera la página web resultante. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

- *Doctrine*: es un mapeador de objetos-relacional (*ORM*) escrito en *PHP* que proporciona una capa de persistencia para objetos *PHP*. Es una capa de abstracción que se sitúa justo encima de un *SGBD* (sistema de gestión de bases de datos).

- *Angular 6*: es un *framework* para aplicaciones web desarrollado en *TypeScript*, de código abierto, mantenido por *Google*, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo-Vista-Controlador, en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

- *TypeScript*: es un lenguaje de programación de código abierto desarrollado por Microsoft, el cual cuenta con herramientas de programación orientada a objetos, muy recomendable su uso si se tienen proyectos grandes. Anders Hejlsberg, arquitecto principal del desarrollo del lenguaje de programación C#, es el principal participante en el desarrollo de este lenguaje. *TypeScript* convierte su código en *Javascript* común. Es llamado también *Superset de Javascript*, lo que significa que, si el navegador está basado en *Javascript*, este nunca llegará a saber que el código original fue realizado con *TypeScript* y ejecutará el *Javascript* como lenguaje original.

- *CSS*: es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de las páginas web, e interfaces de usuario escritas en *HTML*.

- *HTML*: es un lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código *HTML*) para la definición de contenido de una página web. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la *World Wide Web (WWW)*. Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

- *Git*: es un software de control de versiones, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Gestiona eficientemente el desarrollo distribuido, ofreciendo a cada programador una copia local del historial de desarrollo. Además, permite que los almacenes de información se puedan publicar vía *HTTP*, *FTP*...

- *Xdebug*: es una extensión de *PHP* para poder depurar con herramientas de depuración tradicionales, desde el editor, tal como se hace en lenguajes de programación clásicos. Además de permitir analizar el contenido de las variables, nos ayuda a realizar un seguimiento del flujo de ejecución, para saber qué es lo que realmente está ocurriendo cuando algo no funciona como se esperaba.

Capítulo 6: Normativa y legislación

6.1. Licencia de Software

Una licencia de software [7] es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribución) y el licenciatario (usuario consumidor, profesional o empresa) del programa informático, para utilizarlo cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas, es decir, es un conjunto de permisos que un desarrollador le puede otorgar a un usuario en los que tiene la posibilidad de distribuir, usar o modificar el producto bajo una licencia determinada. Además, se suelen definir los plazos de duración, el territorio donde se aplica la licencia (ya que la licencia se soporta en las leyes particulares de cada país o región), entre otros.

Posteriormente, se nombrarán las licencias según las herramientas utilizadas a lo largo del desarrollo de este proyecto:

6.1.1. WebStorm, PHPStorm: toolbox subscription license agreement for education

Se han utilizado estas dos herramientas de desarrollo de programas informáticos, bajo una licencia de pago proporcionada por la universidad.

Esta licencia [8] no puede ser vendida o traspasada a terceros ni utilizar sus productos con propósitos económicos. *JetBrains* tiene y retiene todo derecho, título e interés, incluyendo todos los derechos de propiedad intelectual sobre los productos y cualquier tecnología relacionada o subyacente y modificaciones o trabajos derivados de lo anterior creado por o para *JetBrains*.

6.1.2. Licencia pública general de GNU

Licencia [9] de derecho de autor ampliamente usada en el mundo del software libre y código abierto que garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software. Su propósito es doble: declarar que el software cubierto por esta licencia es libre, y protegerlo (mediante una práctica conocida como *copyleft*) de intentos de apropiación que restrinjan esas libertades a nuevos usuarios cada vez que la obra es distribuida, modificada o ampliada.

Esta licencia es utilizada por: *MySQL, XAMPP* y *Doctrine*.

XAMPP [10] hace uso de otras licencias, dado que está constituida por otros componentes, los cuales tienen sus propias licencias. Por tanto, habría que estudiar cada una de las licencias de dichos componentes para saber cuáles son los límites de esta herramienta en su conjunto.

6.1.3. Licencia comercial

Esta licencia [11] se conoce como software propietario o, dicho correctamente, privativo. Este tipo de licencia tiene como finalidad omitir la posibilidad de acceder de forma libre a su código fuente, el cual solo se encuentra a disposición de su desarrollador y no se permite su libre modificación, adaptación o incluso lectura por parte de terceros.

Esta licencia es utilizada por: *MySQL* y *StarUML*.

6.1.4. Freeware

Este tipo de licencia [12] se define en un tipo de software que se distribuye sin costo, disponible para su uso, pero que mantiene el *copyright*, por lo que no se puede modificar o utilizar libremente como ocurre con el software libre.

Esta licencia es utilizada por: *Google Chrome*.

6.1.5. MIT

Es una licencia [13] de software libre permisiva lo que significa que impone muy pocas limitaciones en la reutilización y por tanto posee una excelente compatibilidad de licencia. La licencia *MIT* permite reutilizar software dentro de Software propietario. Por otro lado, la licencia *MIT* es compatible con muchas licencias *copyleft*, como de *GNU*.

Esta licencia es utilizada por: *Symfony* y *Angular*.

6.1.6. Licencia PHP

La licencia *PHP* [14] es la licencia bajo la que se publica el lenguaje de programación *PHP*. De acuerdo a la *Free Software Foundation* es una licencia de software libre no *copyleft* y una licencia de código abierto según la *Open Source Initiative*. Debido a la restricción en el uso del término "PHP", no es compatible con la licencia *GPL*.

6.1.7. Licencia Apache

La licencia *Apache* [15] es una licencia de software libre permisiva creada por la *Apache Software Foundation (ASF).* La licencia *Apache* requiere la conservación del aviso de derecho de autor y el descargo de responsabilidad, pero no es una licencia *copyleft*, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.

6.2. Seguridad de los Datos

El Reglamento General de Protección de Datos [16] es una legislación elaborada para reforzar y estandarizar la protección de la privacidad de los datos de los habitantes de la Unión Europea. El RGPD viene a sustituir la Directiva de Protección de Datos de la Unión Europea, que se creó en 1995 y representa la última legislación importante de la UE en lo que se refiere a la privacidad de los datos personales. El cumplimiento del RGPD es fundamental para todos aquellos que realicen negocios en los países de la UE.

Los principales objetivos de la RGPD son los siguientes:

- Intenta proteger a cada persona europea y que ellos tomen el control sobre sus datos personales.
- Reforzar los derechos de cada persona (información, accesos, eliminación de datos, modificaciones, y mucho más).
- Crea una ley que se tiene que cumplir por toda la UE por igual.
- Modificar la manera en la que se autorregulará y verificará el tratamiento de datos.

Teniendo en cuenta los aspectos mencionados anteriormente, para que la aplicación desarrollada cumpla la ley, hay que asegurar la protección de los datos.

Esta aplicación hace uso de datos personales, los cuales son:

- Nombre y apellidos (Tutores legales, Alumnos y Profesores)
- Teléfono (Tutores legales)

Por tanto, basándonos en estos datos hay que asegurar que los datos no puedan ser accedidos por terceros, para ello se seguirán las siguientes pautas:

- La base de datos estará protegida bajo contraseña y solo el rol que llamaremos “Super Admin” tendrá acceso.
- Los recursos estarán controlados mediante un *token* por lo que nadie externo a la plataforma podrá acceder a dichos recursos.
- Los datos de los alumnos y los tutores legales se alojarán en la plataforma siempre y cuando los tutores legales den su consentimiento.
- El sistema garantiza que los datos sólo serán alojados en la base de datos durante el tiempo necesario para cumplimentar la finalidad en la que fueron cedidos a los centros. En caso de no ser necesarios en cierto punto, o que el tutor legal solicite la baja en el sistema, los datos serán eliminados de la base de datos.
- Los datos que utilizará el sistema serán los estrictamente necesarios para el correcto funcionamiento del mismo, nunca se pedirá un dato innecesario.

Capítulo 7: Análisis

7.1. Actores

Inicialmente se detectaron dos actores en el sistema, sin embargo, se ha determinado la necesidad de incorporar un nuevo rol al sistema denominado “profesor”. A continuación, se explicará la función de cada uno de ellos en el sistema.

7.1.1. Administrador

El administrador tendrá que ser alguna persona del centro destinada a esta labor y se encargará de llevar a cabo las siguientes funciones:

- **Gestión de los cursos y alumnos del centro:** el sistema tendrá que llevar a cabo una serie de gestiones necesarias para que pueda funcionar, aunque estas no son el objetivo real de la aplicación. Estas gestiones serán las siguientes:
    - Importar automáticamente los datos del curso.
    - Crear cursos.
    - Registrar alumnos.
    - Asociar y desasociar alumnos a un curso.
    - Asociar y desasociar tutores legales de un alumno.

- **Gestión de mensajería:** este fue el objetivo al que se quería llegar en la primera fase del proyecto, a través del cual se podían enviar distintos tipos de mensajes como se verá a continuación:
    - Autorizaciones de actividades: el centro podrá emitir este tipo de mensaje a los tutores legales de los alumnos para que puedan fírmalas mediante un dispositivo móvil que tenga instalada la aplicación móvil. Luego, se podrá ver el listado de los alumnos que quedan autorizados y cuáles no.
    - Encuestas: los tutores legales puedan ser partícipes de las decisiones importantes del centro. Una vez contestadas, se podrá visualizar el resultado de las respuestas.
    - Circulares informativas: los tutores legales podrán estar informados de los acontecimientos importantes relacionados con el centro.

Estas funcionalidades estaban definidas en la primera fase, por tanto, al entrar en la segunda fase se han tenido que añadir las siguientes funcionalidades:

- **Gestión de profesores y tutorías:** dado que se ha introducido en el sistema el rol del profesor, se ha tenido que añadir una serie de nuevas funcionalidades asociadas al administrador para que el sistema pueda agregar profesores y horarios de tutorías. Dichas funcionalidades son las siguientes:
    - Registrar profesores en el centro.
    - Asociar profesores a cursos.

- **Gestión de los alumnos de un curso**: se ha implementado una funcionalidad para poder promocionar los alumnos de curso de una manera muy sencilla e intuitiva. La importancia de esta funcionalidad es poder realizar el cambio de los alumnos de

curso de la manera más eficaz posible, ahorrando mucho tiempo al administrador del centro cuando comienza un nuevo curso académico.

7.1.2. Tutor legal

El tutor legal, accederá a las distintas funcionalidades a través de la aplicación móvil. Aquí, es donde los tutores legales se registran, esto se ha planteado de esta forma para evitar que el centro registre a los tutores legales de los alumnos y envíen mensajes innecesarios a los mismos, dado que puede darse el caso que los tutores legales no cuenten con la aplicación, por lo que nunca estarían informados de los mensajes enviados por el centro.

Los tutores legales son los responsables de los alumnos, los cuales podrán realizar las siguientes tareas:

- Ver los mensajes que les ha enviado el centro (autorizaciones, encuestas y circulares).
- Filtrar mensajes por asunto.
- Firmar autorizaciones, utilizando una contraseña establecida por los responsables legales de los alumnos.
- Responder los mensajes enviados por el centro (autorizaciones, encuestas y circulares).
- Ver sus hijos asociados.
- Seleccionar los centros de sus hijos.

Hasta este punto, es donde llega la primera fase de esta aplicación. A continuación, se indicarán las nuevas funcionalidades implementadas:

- Solicitar citas a los profesores de sus hijos.
- Ver solicitudes de cita de un profesor.
- Cancelar una cita.

Se han añadido esta serie de funcionalidades porque creemos que pueden mejorar sustancialmente la comunicación entre tutores legales y profesores.

7.1.3. Profesor

El profesor, es quien mantendrá contacto directo con los padres, para que estos puedan tener un mayor control de sus hijos en el centro. Las tareas que podrá realizar el profesor son las siguientes:

- Establecer un horario de tutorías.
- Solicitar citas a los padres.
- Ver calendario de citas.
- Confirmar citas.
- Cancelar citas.

7.2. Requisitos

Para representar los requisitos funcionales del sistema se han utilizado diagramas UML (Lenguaje Unificado de Modelado).

Se ha escogido esta opción debido a que es el lenguaje de modelado de sistemas de software más utilizado en la actualidad y, además, lo hemos utilizado a lo largo del grado, por tanto se tiene experiencia con él.

Una de las características más interesantes de UML es que es un estándar aprobado por la ISO (Organización Internacional de Normalización), lo que hace que cualquier diagrama creado con UML pueda ser interpretado por cualquier persona que conozca el estándar.

Como el objetivo de este TFG solo incluye el desarrollo de la aplicación web utilizada por la administración del centro, nos centraremos únicamente en los casos de uso del administrador del centro.

Los casos de uso relacionados con los padres de los alumnos serán explicados en el TFG paralelo a este titulado "Análisis, diseño e implementación de una aplicación para dispositivos móviles multiplataforma para la comunicación entre centros educativos y padres de alumnos II".

7.2.1. Casos de uso

En las *Ilustraciones 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 y 13* se mostrarán los diagramas de casos de uso para tener una idea más clara de los requisitos funcionales adheridos al sistema, dividido en distintas imágenes como se propuso en la primera fase del proyecto. En esta segunda fase se respetará esta pauta para facilitar su lectura, además se destacarán en otro color (naranja) las nuevas implementaciones que han surgido:

7.2.1.1. Administrador

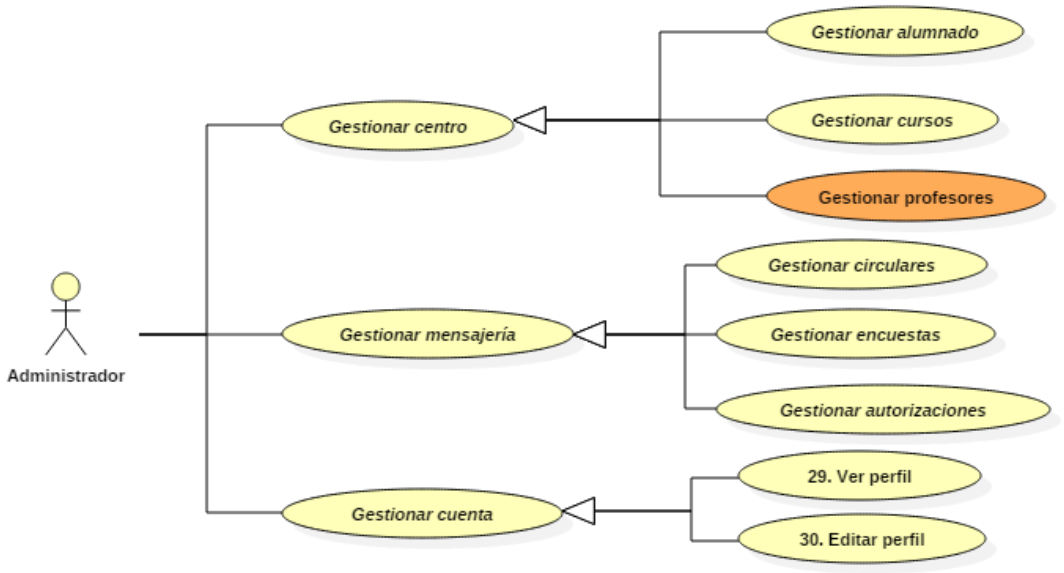


Ilustración 2: Diagrama de Casos de Uso (resumen)

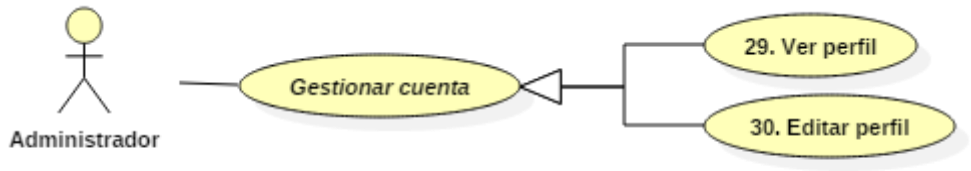


Ilustración 3: Diagrama de Casos de Uso (gestionar cuenta)

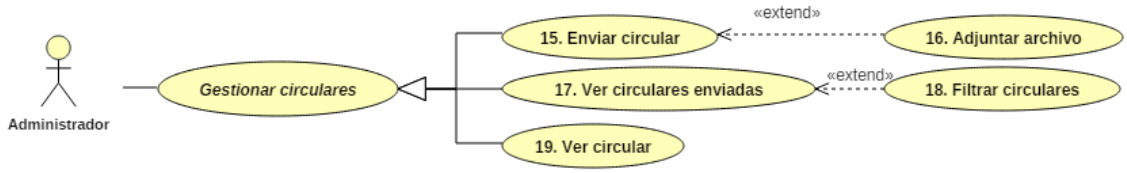


Ilustración 4: Diagrama de Casos de Uso (gestionar circulares)

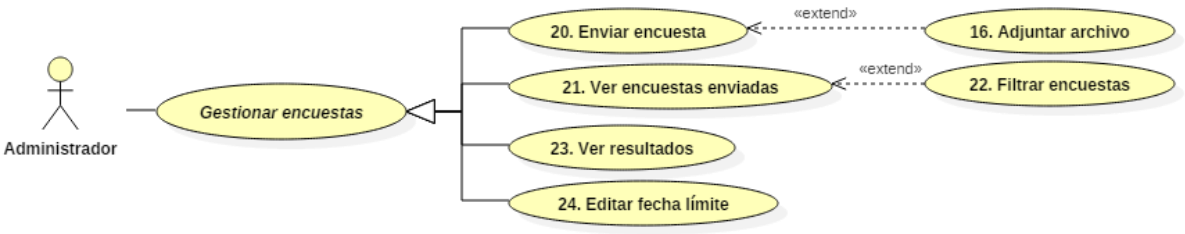


Ilustración 5: Diagrama de Casos de Uso (gestionar encuestas)

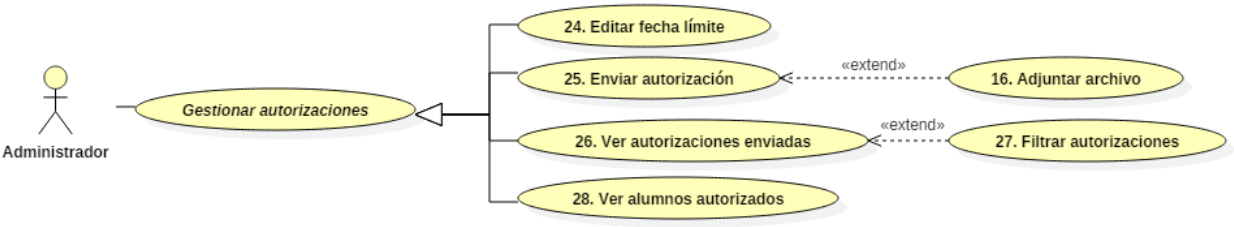


Ilustración 6: Diagrama de Casos de Uso (gestionar autorizaciones)

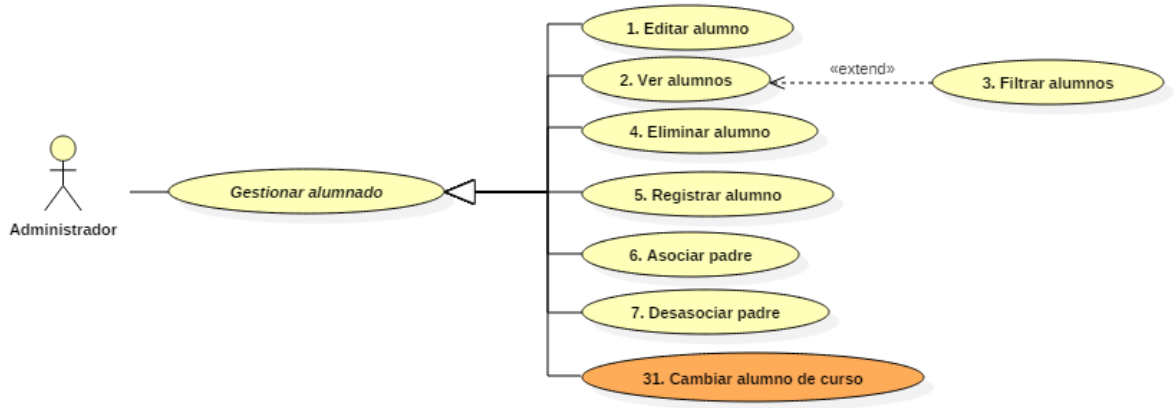


Ilustración 7: Diagrama de Casos de Uso (gestionar alumnado)

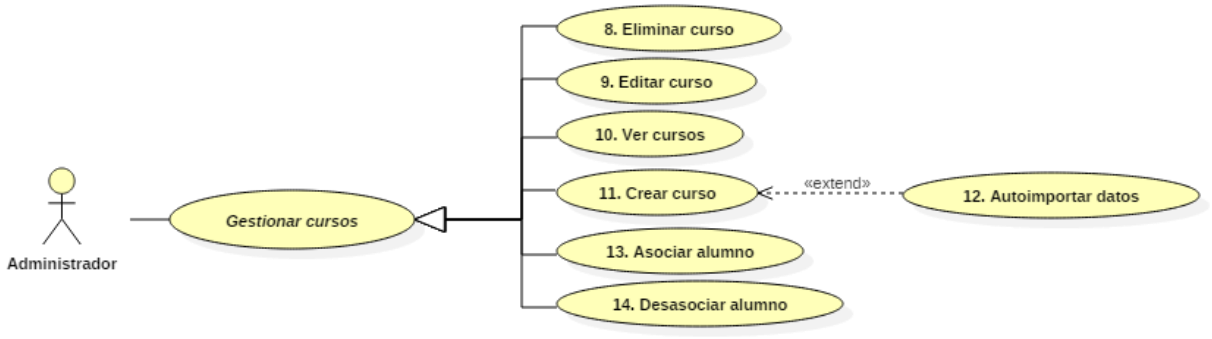


Ilustración 8: Diagrama de Casos de Uso (gestionar cursos)

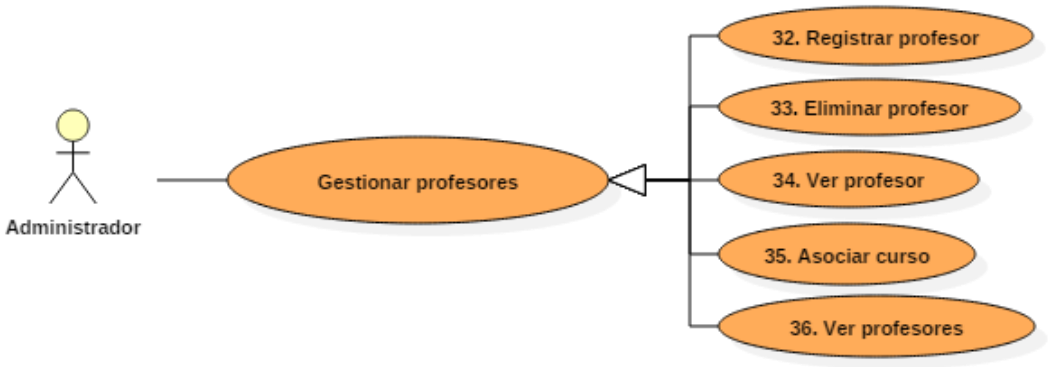


Ilustración 9: Diagrama de Casos de Uso (gestionar profesores)

A continuación, en la *tabla 2* se muestra una descripción resumida de los casos de uso vinculados al administrador:

ID	Actor Principal	Casos de Uso
1	Administrador	Se edita la información conveniente del alumno (nombre, curso).
2	Administrador	Se muestra un listado de los alumnos del centro.
3	Administrador	Se muestran los alumnos que coinciden con los filtros establecidos (curso, nombre).
4	Administrador	Se elimina un alumno del sistema.
5	Administrador	Se registra un alumno en el sistema asociado al centro.
6	Administrador	Se asocia un tutor legal registrado a un alumno del centro.
7	Administrador	Se desasocia un tutor legal de un alumno.
8	Administrador	Se elimina un curso del sistema.
9	Administrador	Se edita la información conveniente de un curso (nombre).
10	Administrador	Se muestra un listado con los cursos del centro.
11	Administrador	Se registra un curso en el sistema asociado al centro y, si se desea, se le asocian alumnos existentes.
12	Administrador	Se selecciona un archivo CSV que permite autoimportar los datos del centro. Se crean cursos, se registran alumnos, se

		asocian los alumnos al curso, se asocian tutores legales a los alumnos.
13	Administrador	Se asocia un alumno registrado al curso.
14	Administrador	Se desasocia un alumno del curso.
15	Administrador	Se envía una circular a los tutores legales de los alumnos seleccionados.
16	Administrador	Se adjunta un archivo en formato PDF al mensaje.
17	Administrador	Se muestran las circulares enviadas.
18	Administrador	Se muestran las circulares que coinciden con el filtro (fecha de envío, asunto).
19	Administrador	Se muestra la circular seleccionada en detalle.
20	Administrador	Se envía una encuesta a los tutores legales de los alumnos seleccionados.
21	Administrador	Se muestran las encuestas enviadas.
22	Administrador	Se muestran las encuestas que coinciden con el filtro (fecha de envío, asunto, estado).
23	Administrador	Se muestran los resultados obtenidos de las encuestas, destacando la opción que ha obtenido más votos.
24	Administrador	Se modifica la fecha límite establecida para poder responder una autorización o una encuesta.
25	Administrador	Se envía una autorización a los tutores legales de los alumnos seleccionados.
26	Administrador	Se muestran las autorizaciones enviadas.
27	Administrador	Se muestran las autorizaciones que coinciden con el filtro (fecha de envío, asunto, estado).
28	Administrador	Se muestra qué alumnos han sido autorizados y cuáles no.
29	Administrador	Se muestra la información correspondiente al administrador que está conectado en la aplicación.
30	Administrador	Se modifica la información del administrador (nombre, usuario o contraseña). Es necesario que introduzca su contraseña actual.
31	Administrador	Se muestran dos listas de cursos seleccionados para realizar el cambio de curso de los alumnos.
32	Administrador	Se registra un profesor en el sistema.
33	Administrador	Se elimina un profesor del sistema.
34	Administrador	Se muestra una breve información del profesor.
35	Administrador	Se asocia un profesor a un curso.
36	Administrador	Se muestra una lista con los profesores registrados en el sistema.

Tabla 2: Resumen de Casos de Uso (Administrador)

7.2.1.2. Profesor

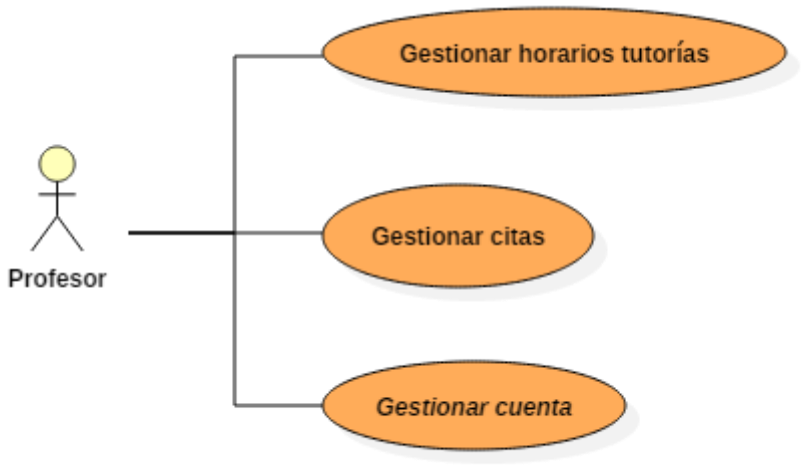


Ilustración 10: Diagrama de Casos de Uso (resumen)

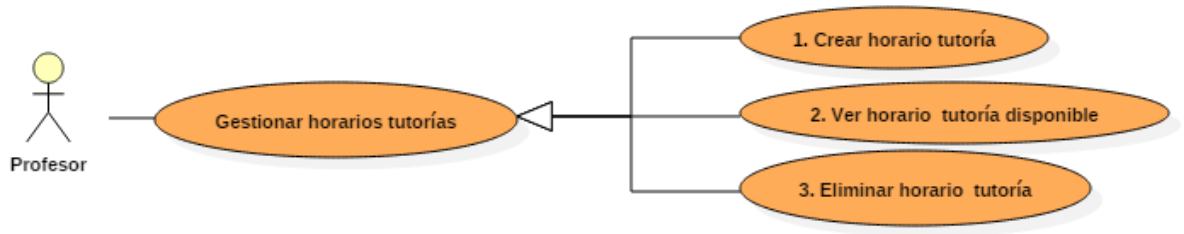


Ilustración 11: Diagrama de Casos de Uso (gestionar horarios tutorías)

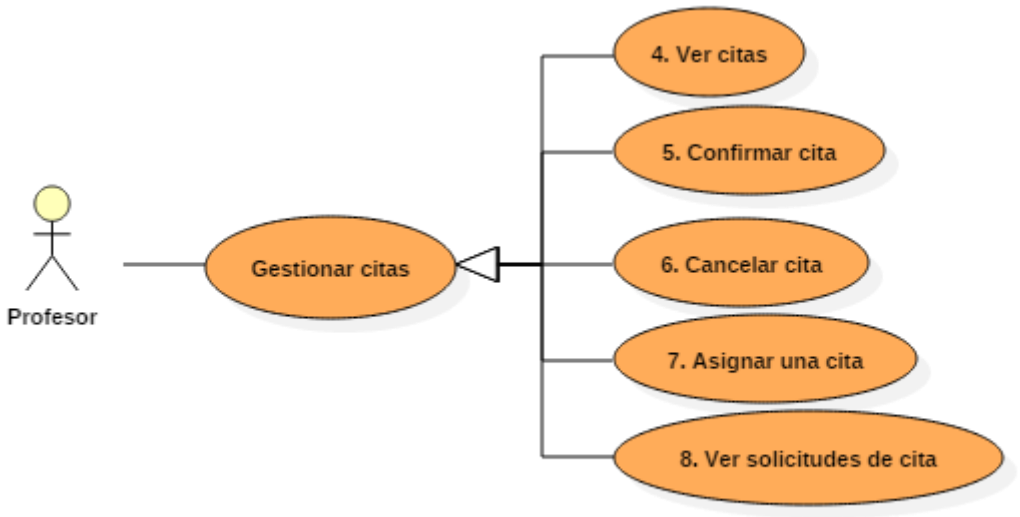


Ilustración 12: Diagrama de Casos de Uso (gestionar citas)

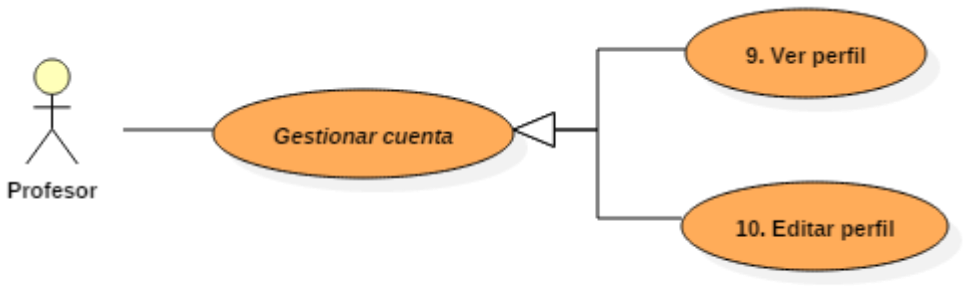


Ilustración 13: Diagrama de Casos de Uso (gestionar cuenta)

En la *tabla 3*se muestra una descripción resumida de los casos de uso propios del profesor:

ID	Actor Principal	Casos de Uso
1	Profesor	Se crea un horario de tutoría
2	Profesor	Se muestra un listado con los horarios de tutoría disponibles del profesor.
3	Profesor	Se elimina un horario de tutoría del sistema
4	Profesor	Se muestra un listado con las citas del profesor.
5	Profesor	Se confirma una cita con un tutor legal.
6	Profesor	Se cancela una cita con un tutor legal.
7	Profesor	Se asigna una cita a un tutor legal.
8	Profesor	Se muestra una lista con las solicitudes de cita.
9	Profesor	Se muestra la información correspondiente al profesor que está conectado en la aplicación
10	Profesor	Se modifica la información del profesor (nombre, usuario o contraseña). Es necesario que introduzca su contraseña actual.

Tabla 3: Resumen de Casos de Uso (Profesor)

7.2.2. Especificación de casos de uso

La especificación de los casos de uso se refiere a la descripción de cada una de las partes definidas para lograr su descripción completa. En esta segunda fase han surgido nuevos casos de uso, por tanto, se ha tenido que realizar la especificación de aquellos que han resultado ser los más relevantes para la aplicación como se muestran en las *tablas 4, 5, 6, 7 y 8*:

CASO DE USO	31	**CAMBIAR ALUMNOS DE CURSO**	
Descripción		El administrador/a desea cambiar un alumno o alumnos de un curso a otro.	
Actores		Administrador/a	
Precondiciones		El administrador/a debe haber iniciado sesión en el sistema.	
Flujo normal		**Paso**	**Acción**

	1	Se muestran dos selectores con los cursos disponibles para elegir el curso origen y el curso objetivo.	
	2	Se cambia el alumno o alumnos de curso.	
Postcondiciones	El alumno o los alumnos han sido promocionados de curso.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones			
Observaciones			

Tabla 4: Especificación de casos de uso (cambiar alumnos de curso)

CASO DE USO	32	**REGISTRAR PROFESOR**	
Descripción	El administrador/a desea registrar un profesor en su centro		
Actores	Administrador/a		
Precondiciones	El administrador/a debe haber iniciado sesión en el sistema.		
Flujo normal	**Paso**	**Acción**	
	1	Se abre un diálogo para crear un profesor.	
	2	Se crea el profesor.	
	3	Se envía un correo al profesor con una contraseña aleatoria.	
Postcondiciones	Se cierra el diálogo.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones			
	2	Se registra el profesor en el sistema.	
Observaciones			

Tabla 5: Especificación de casos de uso (registrar profesor)

CASO DE USO	33	**ELIMINAR PROFESOR**	
Descripción		El administrador/a desea eliminar un profesor del sistema.	
Actores		Administrador/a	
Precondiciones		El administrador/a debe haber iniciado sesión en el sistema.	
Flujo normal		**Paso**	**Acción**
		1	Se abre un diálogo para confirmar la eliminación de un profesor.
		2	Se elimina el profesor.
Postcondiciones		Se cierra el diálogo.	
Variaciones		**Paso**	**Acción**
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones			
Observaciones			

Tabla 6: Especificación de casos de uso (eliminar profesor)

CASO DE USO	35	**ASOCIAR PROFESOR**	
Descripción		El administrador/a desea asociar un profesor a un curso.	
Actores		Administrador/a	
Precondiciones		El administrador/a debe haber iniciado sesión en el sistema. Tiene que haber un curso creado.	
Flujo normal		**Paso**	**Acción**
		1	Se abre un diálogo para asociar el profesor a un curso.
		2	Se asocia el profesor al curso.
Postcondiciones			
Variaciones		**Paso**	**Acción**
Extensiones		**Paso**	**Condición**	**Caso de Uso**

Excepciones			
Observaciones			

Tabla 7: Especificación de casos de uso (asociar profesor)

CASO DE USO	36	**VER PROFESORES**		
Descripción		El administrador/a desea ver una lista de los profesores que hay registrados en el sistema.		
Actores		Administrador/a		
Precondiciones		El administrador/a debe haber iniciado sesión en el sistema.		
Flujo normal		**Paso**	**Acción**	
		1	Se muestra una tabla con un listado de profesores registrados en el sistema.	
Postcondiciones		Se muestran los profesores registrados en el sistema.		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones				

Tabla 8: Especificación de casos de uso (ver profesores)

En cuanto a los casos de uso propios de los profesores, no se ha realizado ninguna especificación de caso de uso dado que este Trabajo de Fin de Grado es el encargado de desarrollar y ofrecer los servicios necesarios para que se puedan desarrollar las vistas desde la aplicación móvil, desarrollada paralelamente a este proyecto.

Capítulo 8: Diseño

8.1. Diseño de la arquitectura del sistema

Se ha decidido usar una arquitectura *API REST* [19][20][21] porque permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda *HTTP*, ya que es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar *HTTP*. Esta arquitectura es ideal para el sistema dado que desde el *backend* se desarrollan una serie de recursos que son aprovechados por el *frontend* (portal web y la aplicación móvil). *REST* proporciona las siguientes ventajas:

- Es un protocolo sin estado, debido a que no se guarda la información en el servidor. Es decir, toda la información será enviada por el cliente en cada mensaje *HTTP*, consiguiendo un ahorro en variables de sesión y almacenamiento interno del servidor.
- Presenta un conjunto de operaciones bien definidas, siendo las más importantes *GET*, *POST*, *PUT* y *DELETE*, que se emplea en todos los recursos.
- Utiliza URIs únicas siguiendo una sintaxis universal.
- Emplea *hipermedios* para representar la información, que suelen ser *HTML, XML o JSON*.

En la *Ilustración 14*, se muestra un diagrama con la arquitectura del sistema:

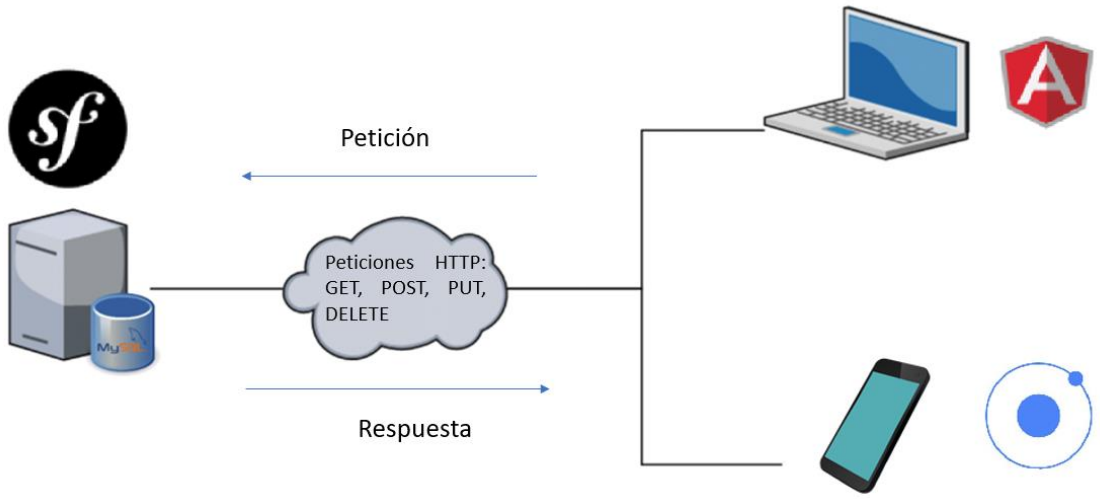


Ilustración 14: Arquitectura del sistema (API REST)

8.1.1. Frontend

El sistema cuenta con dos interfaces totalmente diferenciadas:

- El portal web desarrollado en *Angular*. Este es utilizado por el administrador del centro para gestionar labores propias del centro, como, por ejemplo: enviar todo tipo de mensajes a los tutores legales (circulares, encuestas y autorizaciones), cambiar los alumnos de un curso a otro, registrar profesores etc.
- La aplicación móvil desarrollada en *Ionic 3*, encargada de gestionar la lectura de las circulares, así como de la contestación de encuestas y autorizaciones. Además de realizar una gestión de los horarios de tutoría del profesor y las citas con el tutor legal.

8.1.2. Backend

El *backend* ha sido desarrollado con el *framework PHP* denominado *Symfony* que a su vez hace uso de *PHP 7* y *Doctrine*. Este se encuentra alojado en un servidor web *Apache*. Aquí también está alojada la base de datos *MySQL*.

Este *backend* es usado por ambas aplicaciones (portal web y aplicación móvil), de manera que el portal web y la aplicación móvil hacen uso de la misma base de datos.

8.2. Diseño de la base de datos

En la primera fase del proyecto, se decidió utilizar el sistema de gestión de base de datos MySQL. Por tanto, en esta segunda fase se seguirá utilizado MySQL para gestionar la base de datos, ya que así se puede reutilizar la base de datos de la primera fase.

A continuación, se mostrará la estructura de la base de datos generada en la primera fase del proyecto y las actualizaciones pertinentes (remarcadas con un borde rojo) que se tuvieron que realizar para añadir al sistema las nuevas funcionalidades.

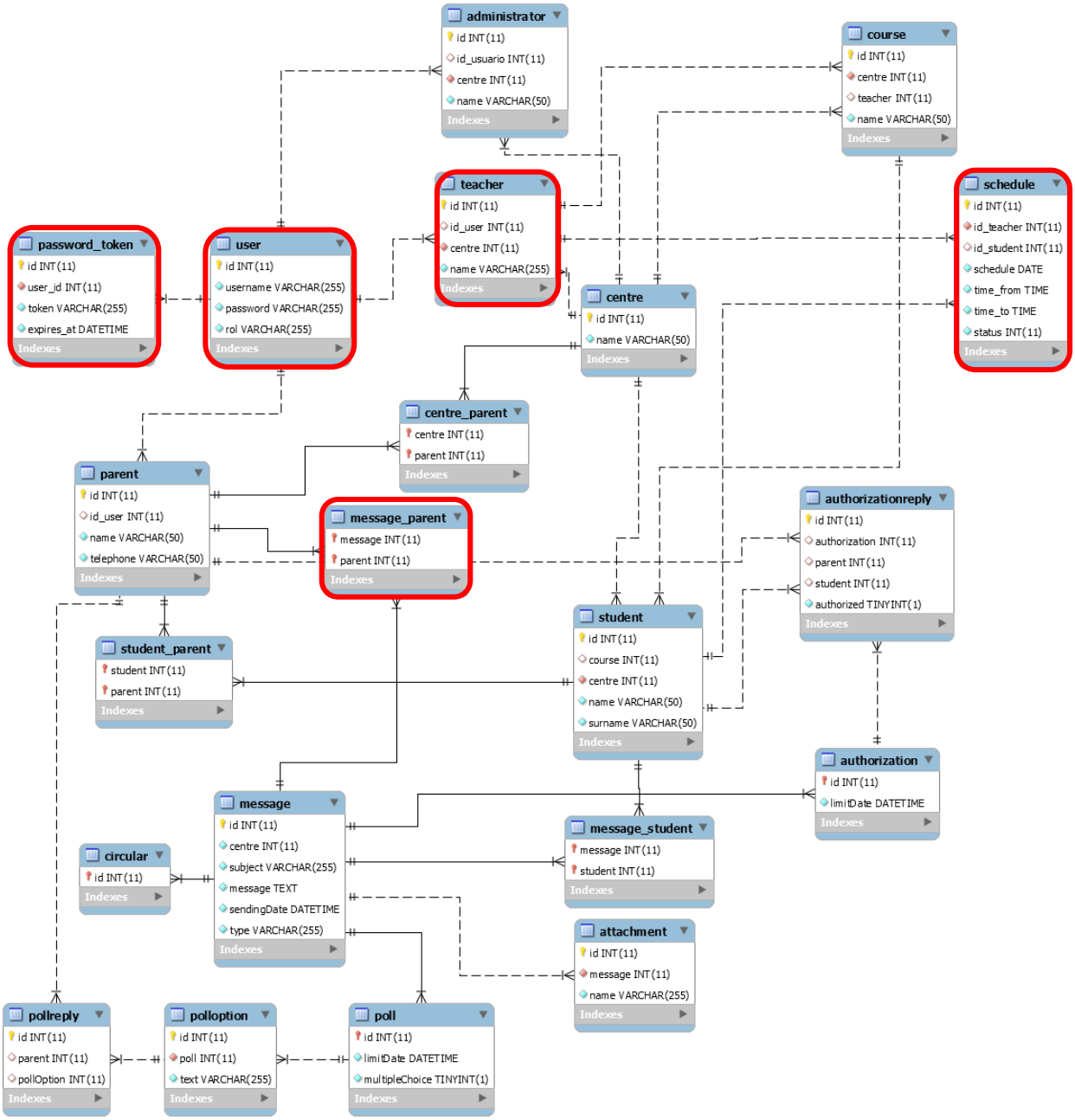


Ilustración 15: Esquema base de datos

Como se puede observar en la *Ilustración 15*, la actualización de la base de datos ha consistido en añadir cinco nuevas tablas, cuatro de ellas están relacionadas con entidades y la otra es una relación de entidades.

La tabla definida como `user`, ha sido necesaria unificar las credenciales de inicio de sesión de cada usuario y así evitar tener los campos `username` y `password` en las tablas de `administrator`, `parent` y `teacher`.

La tabla definida como `password_token`, ha sido necesaria para implementar un sistema que permita al profesor restablecer su contraseña en caso de olvidarse de la misma.

La tabla definida como `teacher`, ha sido necesaria para introducir en el sistema el rol del profesor.

La tabla definida como `schedules`, ha sido necesaria para que el profesor pueda gestionar sus horarios de tutoría.

Por otra parte, se ha añadido una nueva tabla de relación de entidades denominada `message_parent`. Esta tabla ha sido añadida para tener un registro de que los tutores legales han leído los mensajes enviados por el centro.

8.3. Almacenamiento de ficheros en el servidor.

En la primera fase del proyecto se decidió que los ficheros adjuntos a los mensajes (circulares, encuestas y autorizaciones) se guardarán en el sistema de ficheros por las siguientes razones:

- El nivel de conocimiento necesario para mantener una base de datos es proporcional al tamaño de la base de datos, ya que:
  - Dificulta las migraciones.
  - Dificulta las copias de seguridad.
- Guardar los ficheros en la base de datos complica el código encargado de guardar los ficheros en la base de datos, puesto que podemos guardarlos en distintos tipos de datos y tenemos que crear una capa de software distinta para cada tipo de datos.
- Es más complicado acceder a los ficheros guardados en una base de datos desde una aplicación web.
- No podemos aprovechar la potencia del almacenamiento en la nube si guardamos los ficheros en la base de datos.

Por tanto, en esta segunda fase del proyecto se ha decidido respetar esta decisión ya que es una buena práctica no guardar los ficheros directamente en la base de datos.

8.4. Diseño arquitectónico

En este proyecto, se ha decido usar dos *frameworks* de desarrollo. Symfony como *framework* PHP para desarrollar el *backend* y Angular como *framework* JavaScript para desarrollar el *frontend*. Hay que destacar que Symfony está basado en el patrón arquitectónico Modelo-Vista-Controlador (MVC) y la arquitectura de Angular está basada en módulos.

8.4.1. *Symfony*

De manera genérica, los componentes de MVC [22] se podrían definir como sigue:

- El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.
- El Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar órdenes a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o *scroll* por un documento o por los diferentes registros de una base de

datos), por tanto, se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

- La Vista: Presenta el 'modelo' (información y *lógica de negocio*) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por tanto, requiere de dicho 'modelo' la información que debe representar como salida.

En la siguiente Ilustración se muestran los distintos componentes y el flujo del patrón MVC:

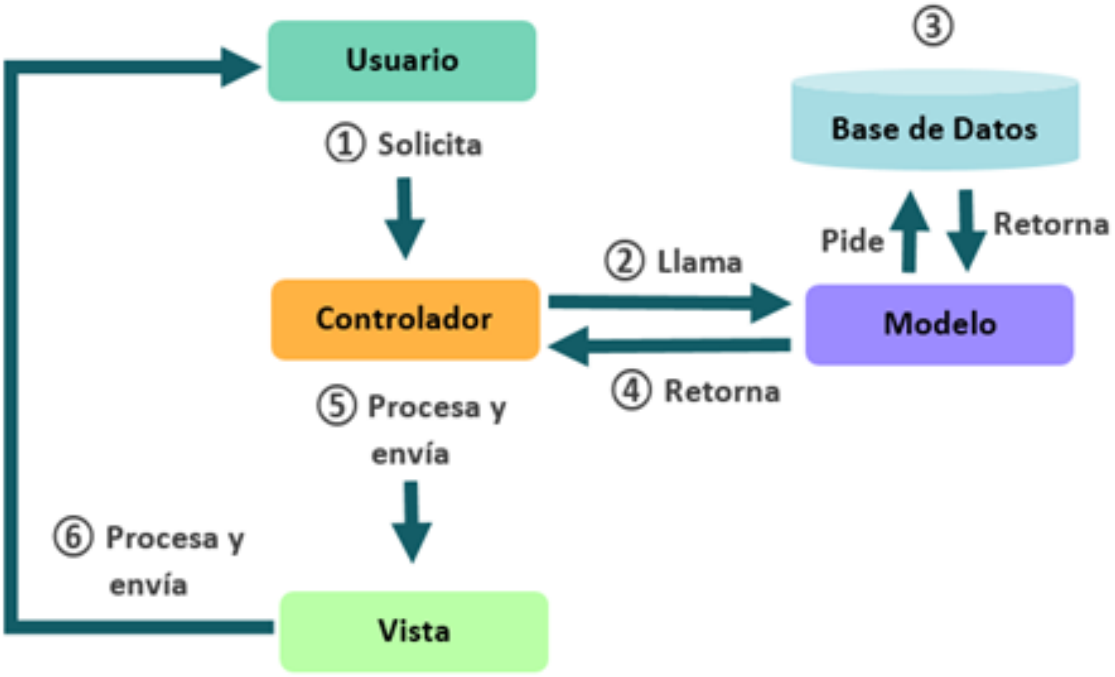


Ilustración 16: Patrón Modelo-Vista-Controlador

8.4.2. *Angular*

Una de las grandes diferencias entre un *framework* y una biblioteca, es que el *framework* es un conjunto grande de funcionalidades "genéricas" preparadas para desarrollar funcionalidades específicas. En cambio, una biblioteca se utiliza para hacer uso de una única función genérica. Partiendo de esta definición se puede decir que un *framework* consiste en varias bibliotecas escritas para manejarse todas juntas. Manteniendo este planteamiento, se puede decir que Angular preparó todo para que una aplicación solo utilice los módulos (o bibliotecas) que se van a necesitar para desarrollar una web o aplicación móvil.

Teniendo en cuenta los aspectos mencionados anteriormente, cuando se crea un proyecto Angular, este va a tener el módulo principal llamado "core". Aquí es donde se puede ejecutar la aplicación y donde se podrán definir cada uno de los componentes que van a formar parte de la ella.

Por otra parte, si la aplicación necesita generar rutas o si se necesitan agregar formularios, Angular tiene los módulos necesarios para poder hacer uso de estos elementos.

Como se ha visto hasta ahora, Angular en su interior está provisto de un gran conjunto de módulos que pueden ser usados dependiendo de las necesidades de cada uno. Para usar un módulo solo bastaría con importarlo ya se podría hacer uso de ese modulo en el proyecto.

Las aplicaciones desarrolladas en Angular [23][24] están compuestas por:

- Componentes: son los elementos que definen la información o contenido que queremos mostrar en un segmento la pantalla. En los componentes también se define cómo se debe responder a distintos eventos.
- Templates HTML: definen cómo se despliegan los componentes en una vista.
- Directivas: atributos especiales que usamos en las plantillas para poder transformar el DOM (Modelo de Objetos del Documento) de nuestra vista de manera dinámica (Ej: ngIf, ngFor).
- Servicios: tienen un propósito específico y bien definido que se presta a ser encapsulado para ser reutilizado por distintos componentes en nuestra aplicación.
- El "inyector": es un mecanismo que se encarga de proveer a los componentes con instancias de los servicios que necesitan.

A continuación, se muestra un diagrama de la arquitectura de Angular:

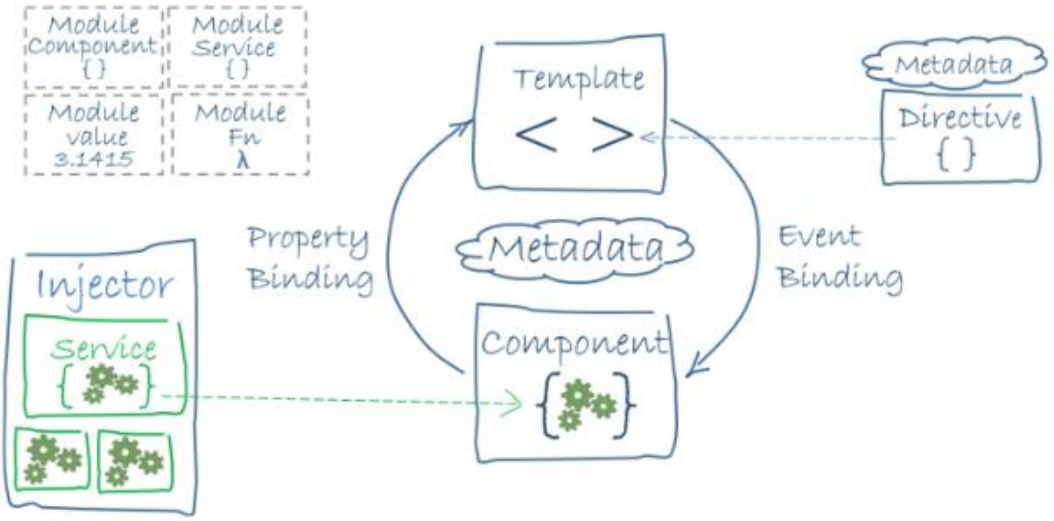


Ilustración 17: Diagrama de la arquitectura de Angular

8.5. Diseño de la interfaz

En la fase de diseño de cualquier proyecto web es muy importante pensar en su usabilidad. Entendemos usabilidad como la facilidad que tienen los usuarios de relacionarse con la interfaz de nuestra página y de navegar en ella. Una buena usabilidad aportará un aumento de la eficiencia de nuestra página web, una reducción de costes y un aumento de la fidelización de los usuarios. Por tanto, se han seguido una serie de principios definidos por Jakob Nielsen [25][26] (Doctor en diseño de interfaces de usuario y ciencias de la computación en la Universidad Técnica de Dinamarca) para desarrollar la interfaz.

A continuación, se describirán cada uno de los principios de usabilidad definidos por Jakob Nielsen y su papel en el sistema desarrollado en este proyecto:

1. **Visibilidad del estado del sistema:** La web o aplicación debe mostrar en todo momento al usuario qué está pasando y en qué punto de la navegación se encuentra.

   A la hora de enviar formularios, por ejemplo, en la creación de un curso, siempre se le va a mostrar al usuario un pequeño mensaje situado en la parte inferior derecha de la pantalla diciéndole el estado de la acción que acaba de realizar y diferenciado por

colores, color verde si la acción se ha realizado satisfactoriamente y de color rojo si la acción no se ha finalizado correctamente.

Por otro lado, siempre se le muestra al usuario en que punto de la navegación esta mediante un menú lateral, en el que se queda marcado en diferente color el apartado del menú en el que se encuentra.

2. **Adecuación entre el sistema y el mundo real**. El sistema debe hablar en el mismo lenguaje que los usuarios.

   Para conseguir este principio, la información mostrada en el portal tiene que tener un orden lógico, por ejemplo, cuando se crea un curso, se mantiene un flujo.

   Primero se crea el curso, se le asocia un profesor y por último se le asocian los alumnos. Además, se muestran iconos para aclarar las acciones que puede realizar el usuario sin darle la posibilidad de equivocarse. Con esto, se consigue que la interacción con el usuario sea natural y no le cueste navegar por la interfaz.

3. **Libertad y control por el usuario**. Los usuarios deben poder volver fácilmente a un estado anterior. Es conveniente dar las opciones de “deshacer” y “rehacer”.

   En la mayoría de las acciones que puede realizar el usuario se le ofrece la posibilidad de subsanar el error. Por ejemplo, si el usuario crea un curso y se equivocó a la hora de definir el nombre, se le permite editarlo. En cambio, cuando el administrador del centro decide enviar un mensaje a los tutores legales, ya sea una circular, encuesta o autorización no se le ofrece la posibilidad de editar el mensaje porque puede que ya haya tutores legales que hayan leído el mensaje y hayan aceptado el contenido del mensaje. Por tanto sería incongruente que el administrador pudiera modificar el mensaje y cambiar sus términos y condiciones ya que el tutor legal podría no volver a leer el mensaje al haberlo leído anteriormente.

4. **Consistencia y estándares**. Es conveniente seguir y repetir algunos patrones para no confundir a los usuarios.

   Para adoptar este principio, el sistema ha sido diseñado utilizando tres colores (azul, blanco y rojo) vinculado cada uno de ellos a distintas acciones que pueden ser realizadas. El azul y el blanco indican que se pueden realizar acciones de añadir o editar contenido al portal y el rojo para ejecuta la acción de eliminar algún ítem.

5. **Prevención de errores**. Es mejor prevenir los errores que generar mensajes una vez se produzcan.

   Siempre hay que facilitar a los usuarios los errores al instante, por tanto, todos los formularios están provistos de validadores interactivos que muestran mensajes de error al usuario instantáneamente, así se evita que el usuario rellene un formulario completo y que al finalizar se encuentre con la sorpresa de que ha cometido errores en pasos anteriores del formulario.

6. **Reconocer mejor que recordar.** Hay que intentar en la medida de lo posible mostrar objetos, acciones y opciones para minimizar el uso de memoria del usuario.

Este principio es recogido en el sistema mediante el uso de nombres e iconos descriptivos vinculados a cada acción para así evitar que el usuario haga uso de su memoria, sino que reconozca las acciones que puede hacer al instante.

7. **Flexibilidad y eficiencia de uso**. Es importante personalizar las acciones frecuentes. A veces hay que crear aceleradores o atajos para mejorar la usabilidad para los usuarios más expertos.

   Este principio no está recogido en el sistema porque no tiene vistas extremadamente complicadas que necesiten ayudas o atajos predefinidos para realizar alguna acción.

8. **Estética y diseño minimalista**. Intentar simplificar, eliminar el contenido irrelevante para que el usuario sólo se fije en lo realmente importante.

   El diseño del portal está basado en tablas que muestran cada uno de los ítems que forman parte del apartado seleccionado, por ejemplo, si se quieren ver todos los profesores del centro, se mostrará una tabla con los profesores registrados en el centro y así también con alumnos, cursos y mensajes. Por otra parte, tenemos los botones necesarios para poder ejecutar cualquier acción permitida por el sistema en cada apartado del portal.

9. **Ayudar a los usuarios a reconocer, diagnosticar y solucionar los errores**. Los mensajes de error deben expresar claramente cuál ha sido la causa del problema.

   Como se ha mencionado en el punto cinco, los formularios están controlados para evitar que el usuario introduzca datos erróneos. Estos formularios muestran mensajes de error en caso de haberlos y al finalizar una acción como. Por ejemplo, al registrar un alumno en el sistema, al usuario se le muestra un pequeño mensaje que le informa del estado de la transacción de la acción.

10. **Ayuda y documentación**. En algunos casos puede ser necesario que el usuario necesite ayuda. Es necesario que ésta sea fácil de encontrar, útil, y si puede ser no demasiado extensa.

    Como en toda aplicación, la documentación de ayuda es muy importante, por tanto, se ha desarrollado un manual de usuario que contiene toda la información necesaria para guiar a los distintos usuarios en la ejecución de las distintas acciones que se pueden realizar en la plataforma en función de su rol.

Capítulo 9: Desarrollo

En este apartado del documento se explicará porque se ha decido escoger el *framework* de PHP Symfony para desarrollar el *backend* y el *framework* de JavaScript denominado Angular para desarrollar el frontend. Además, se definirán las estructuras de cada *framework*.

Por último, se explicará la seguridad que se ha implementado en el sistema, para realizar el inicio de sesión tanto en la aplicación web como la aplicación móvil y la obtención de recursos del *backend* de estas dos aplicaciones.

9.1. *Symfony*

Se ha escogido *Symfony* [27], debido a que ofrece las siguientes ventajas:

- Actualmente *Symfony 3* es el *framework* más utilizado y mejor valorado para programar en *PHP*.
- Es fácil de instalar y se adapta a cualquier plataforma.
- Permite elegir el gestor de bases de datos que se desee.
- El *framework Symfony 3* sigue la mayoría de mejores prácticas de diseño para aplicaciones web.
- Un independiente sistema gestor de bases de datos nos da una capa de abstracción con el uso de su *ORM Doctrine*. Ideal para gestionar todas las consultas y peticiones al servidor, así como mostrar las respuestas que la lógica de negocio requiera.
- Utiliza programación orientada a objetos tanto en la capa de negocio, como para gestionar el modelo de datos, lo que es una gran ventaja.
- Se basa en la arquitectura de diseño de software *MVC (Model View Controller)*. Esta característica permite una comunicación entre las tres capas independientes con gran eficiencia.
- Facilidad para gestionar las rutas *URL* inteligentes, que permite configurar direcciones amigables en las páginas de la aplicación. El uso de controladores que gestionan estas rutas es ideal para crear aplicaciones web complejas a medida, de forma muy ordenada y eficiente.
- Permite su integración con las bibliotecas de otros fabricantes y así aprovechar las ventajas en la parte del *frontend* de nuevas librerías tanto en *JavaScript* tipo *Angular*, *Vue.js, Jquery*, etc.

9.1.1. Estructura del proyecto

Una de las principales características de la estructura de directorios en un proyecto de *Symfony*, es que tiene una estructura bien definida donde cada uno de sus directorios cumple una función concreta, además de ser estándar para todo desarrollador.

La estructura de directorios de *Symfony* es la siguiente:

- `/app/`: aquí se encuentran los ficheros de configuración de la aplicación, plantillas y traductores (no se hará uso del motor de plantillas que ofrece *Symfony* ya que el *frontend* está desarrollado con Angular).
- `/bin/`: aquí se encuentran los ficheros ejecutables, este directorio permite ejecutar órdenes como por ejemplo para crear una nueva entidad.

- `/src/`: aquí es donde están los controladores, entidades y servicios creados en la aplicación.
- `/var/`: aquí es donde se encuentran los ficheros generados (cache, logs, sessions, etc.).
- `/vendor/`: aquí es donde se encuentran las dependencias del proyecto.
- `/web/`: este es el directorio raíz de la aplicación web. Aquí es donde se incluyen las imágenes, archivos *CSS* y *JavaScript*, entre otros. Este directorio no se usará por el mismo motivo definido en el primer punto.

9.1.2. Enrutamiento

El enrutamiento en *Symfony* se puede implementar de distintas maneras, en este proyecto se ha decidido utilizar las anotaciones ya que ha resultado ser la manera más fácil y entendible de definir las rutas. Este apartado equivale al Controlador en el patrón arquitectónico *MVC*, que es donde se desarrollaran cada uno de los recursos que ofrece el *backend*.

Para hacer uso de las anotaciones, estas son definidas en el controlador, indicando encima del método a desarrollar el nombre de la ruta y el método HTTP utilizado.

Los métodos *HTTP* utilizados son los siguientes:

- *GET*: devolver un recurso.
- *POST*: crear un recurso.
- *PUT*: actualizar un recurso.
- *DELETE*: eliminar un recurso.

```
/**
 * @Route("/{id}/courses", name="listarCursosDelCentro")
 * @Method("GET")
 */
public function getCoursesAction(Request $request, $id)
{
}
```

A continuación, se mostrará una serie de ejemplos de las *URLs* definidas utilizando los métodos *HTTP* definidos anteriormente:

- `listarHorariosDeTutoriaProfesor: GET /teachers/{id}/schedules`
- `crearHorarioDeTutoria: POST /schedules`
- `asociarAlumnoAHorarioDeTutoria:POST /schedules/{idSchedule}/students/{idStudent}`
- `editarHorarioDeTutoria: PUT /schedules/{id}`
- `eliminarHorarioDeTutoria: DELETE /schedules/{id}`

9.1.3. *Doctrine*

Para acceder a la información de la base de datos se ha utilizado *Doctrine*. *Doctrine* es un mapeador de objetos-relacional (ORM) que proporciona una capa de persistencia para objetos *PHP*. Es una capa de abstracción que se sitúa justo encima de un sistema de gestión de bases de datos.

Una característica de *Doctrine* es el bajo nivel de configuración que necesita para empezar un proyecto. En un proyecto *Symfony*, donde *Doctrine* viene integrado por defecto, basta con

establecer una serie de campos en el fichero `/app/config/parameters.yml` que, en nuestro caso, serán los siguientes:

```
parameters:
    database_host: localhost
    database_port: 3306
    database_name: hermerest
    database_user: root
    database_password: null
```

Una vez realizada esta configuración se podrá generar la base de datos a partir de las clases de entidad desarrolladas.

Es importante saber que *Doctrine* tiene un conjunto de órdenes que facilitan muchas tareas como la creación de un esquema de base de datos, creación de entidades, ejecutar *SQLs*, entre otras.

Las órdenes [28] más utilizadas en el proyecto han sido los siguientes:

- `php bin/console generate:doctrine:entity` (Crear una entidad)
- `php bin/console doctrine:schema:update --dump-sql` (Alter tables)
- `php bin/console doctrine:schema:update –force` (Actualizar BD, solo agrega columnas nuevas)
- `php bin/console cache:clear --env=prod` (Borrar caché de producción)
- `php bin/console cache:clear --env=dev` (Borrar caché de desarrollo)

9.1.4. Entidades

Para generar la base de datos se tuvieron que implementar las entidades con sus respectivas relaciones. Estas entidades equivalen a la capa de Modelo en el patrón arquitectónico MVC, es decir, las clases que contienen los datos que guardaremos en la base de datos.

En el siguiente diagrama se muestran las relaciones entre las entidades, cabe destacar que el diagrama ha sido actualizado para introducir las nuevas funcionalidades en el sistema que abarca esta segunda fase del proyecto. Las actualizaciones se representarán en color naranja para ver con más claridad las nuevas entidades y relaciones:

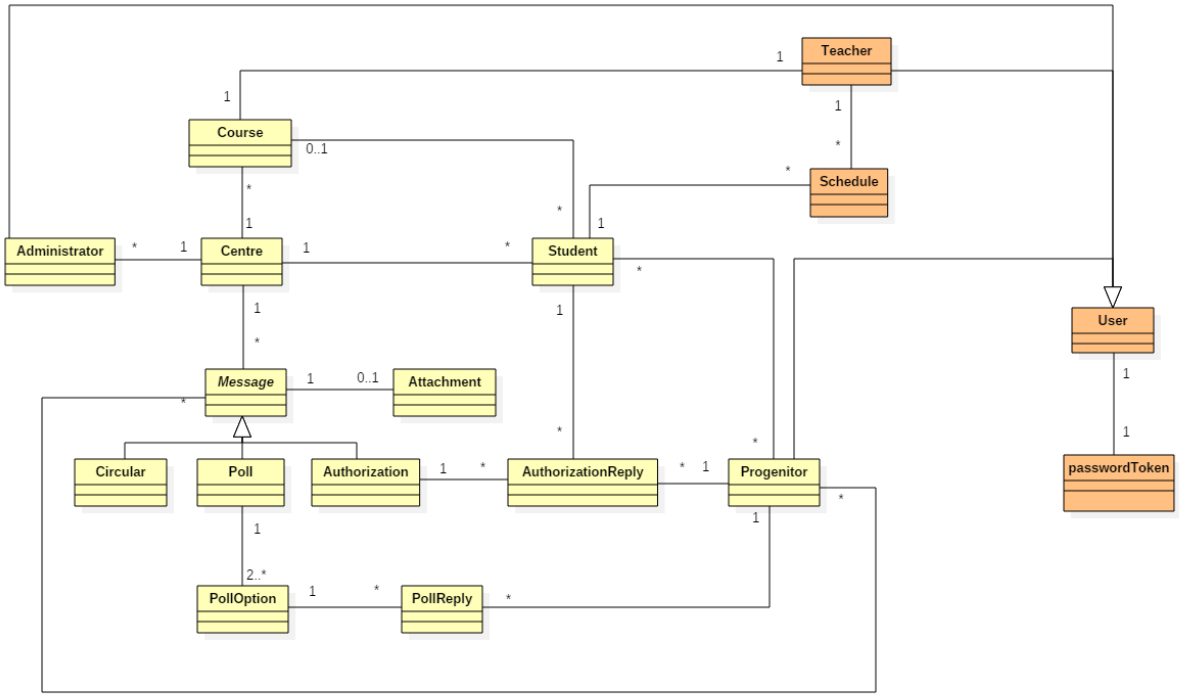


Ilustración 18: Diagrama de entidades

9.1.5. Acceso a la base de datos

En la primera fase del proyecto, para que la interacción entre los controladores de la aplicación y la base de datos fuese lo más sencilla posible, se utilizó el patrón estructural `Facade`. Este patrón consiste en crear una "fachada" que simplifique el acceso a la base de datos desde el controlador.

Por tanto, en esta segunda fase se ha decidido seguir utilizando este patrón de diseño para aprovechar las fachadas desarrolladas en la primera fase.

Otra de las razones de porque se ha seguido utilizando el patrón de diseño `Facade` es por la limpieza en el código que ofrece, dado que todos los `Facades` heredan de `AbstractFacade` en donde están todos los métodos estándar que utiliza cualquiera de las entidades para realizar las operaciones *CRUD* (crear, eliminar, actualizar y borrar). La explicación del uso del patrón de diseño `Facade` en este proyecto se puede encontrar en el Trabajo de Fin de Grado denominado "ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN BACKEND PARA LA COMUNICACIÓN ENTRE CENTROS EDUCATIVOS Y PADRES DE ALUMNOS".

9.1.6. Representación de los recursos

Cuando el cliente realiza peticiones al servidor, este responde ofreciéndole el recurso en cuestión. Para representar estos recursos se ha utilizado la interfaz `Normalizer` lo que permite serializar la información (en formato *JSON* o *XML*) que ofrece el servidor cuando se realiza una petición sobre un recurso.

```php
class CourseNormalizer implements NormalizerInterface
{

    public function normalize($object, $format = null, array $context = array())
    {
        return [
            'id' => $object->getId(),
            'name' => $object->getName(),
            'centre' => (new CentreNormalizer())->normalize($object->getCentre()),
            'numberOfStudents' => count($object->getStudents()),
        ];
    }


    public function supportsNormalization($data, $format = null)
    {
        return $data instanceof Course;
    }
}
```

9.2. *Angular*

Se ha escogido *Angular* [29][30], debido a que ofrece las siguientes ventajas:

- Al ser un *framework*, *Angular* nos ofrece por defecto más funcionalidades que una simple biblioteca.
- Uso de *TypeScript* que aporta las siguientes ventajas:
    - Consistencia en la documentación: Cuando buscamos documentación de *TypeScript* toda la sintaxis y la manera de desarrollar es la misma, lo que añade coherencia a la información y a la forma de leer el código.
    - Mejor y más fácil mantenimiento de las aplicaciones.
    - Esta consistencia ayuda a evitar la confusión y la sobrecarga en la toma de decisiones derivadas de empezar a trabajar con Angular.
    - Utilizar *TypeScript* y *Angular* ayuda a comprender mejor cómo funciona internamente JavaScript y hace que el código sea mejor. Y, sobre todo, facilita las labores de *testing*, y mejora la comprensión y calidad del software en general.
- Crear componentes web: Un componente en Angular es una porción de código que es posible reutilizar en otros proyectos de Angular sin apenas esfuerzo, lo que permite un desarrollo de aplicaciones mucho más ágil, pasando de un "costoso" MVC a un juego de puzles con nuestros componentes.
- La búsqueda de errores de *runtime* en JavaScript puede ser una tarea imposible. TypeScript proporciona detección temprana de errores (en tiempo de compilación), y tipado fuerte de clases, métodos, así como de objetos y APIs JavaScript ya existentes.
- Angular es respaldado por Google y una gran y creciente comunidad.
- Son proyectos totalmente *opensource*, publicados en GitHub y abiertos a contribuciones.

9.2.1. Estructura del proyecto

En un proyecto de *Angular*, al igual que en *Symfony*, la estructura del directorio está bien definida y sigue unos estándares que se muestran a continuación:

- `/e2e/`: en esta carpeta se colocan los archivos para la realización de las pruebas "end to end".
- `/node_modules/`: en esta carpeta es donde *npm* (gestor de paquetes) va colocando todas las dependencias del proyecto.

- */src/*: esta carpeta es donde están las fuentes del proyecto. Esta carpeta es la que se usará para desarrollar la aplicación donde se irán colocando los componentes.
- `/src/app/`: esta carpeta contiene el código del componente principal, que está dividido en varios archivos.
- `/src/assets/`: aquí es donde se suelen guardar las imágenes que se utilizan en la aplicación.
- `/src/environments/`: aquí es donde se suelen definir las variables que van a ser usadas en distintas partes de la aplicación.

Al crear un proyecto de Angular también se crean archivos sueltos en la carpeta raíz del proyecto. Los más importantes son los siguientes:

- `index.html`: archivo que contiene información básica del proyecto recién creado.
- `angular-cli.json`: archivo oculto en el que se almacenan configuraciones del CLI de Angular.
- `tslint.json`: este archivo sirve para configurar el `linter`, el programa que nos alertará cuando tengamos problemas de estilo en el código.
- `.editorconfig`: este es un archivo que sirve para definir la configuración para el editor de código que estemos utilizando. Permite centralizar la configuración, de modo que sea común para todos los desarrolladores que vayan a trabajar en el proyecto.

9.2.2. Obtener recursos del *backend*

Para obtener los recursos que necesita el *frontend* se ha creado un servicio definiendo los métodos *HTTP* que van a ser usados.

```
get(endpoint: string) {
  return this.http.get(this.baseUrl + endpoint, this.headers);
}

post(endpoint: string, json) {
  return this.http.post(this.baseUrl + endpoint, this.bodyToString(json), this.headers);
}

put(endpoint: string, json) {
  return this.http.put(this.baseUrl + endpoint, this.bodyToString(json), this.headers);
}

delete(endpoint: string) {
  return this.http.delete(this.baseUrl + endpoint,  this.headers);
}
```

La comunicación entre el frontend y el backend se realiza mediante peticiones y respuestas. Estas peticiones y respuestas se envían y reciben en formato JSON.

Todas las respuestas del servidor vendrán definas de la siguiente manera:

- Respuesta satisfactoria: {success: true, content: [contenido del recurso solicitado]}
- Respuesta no satisfactoria: {success: false, content: 'mensaje de error'}
- Petición no autorizada: {success: false, content: 'el token no existe o es incorrecto'}

9.3. Seguridad

Para aumentar la seguridad en él envió de datos entre servidor-cliente (Symfony-Angular) se ha utilizado JSON [31] Web Token (JWT). Este es un estándar abierto basado en JSON para crear un

token que sirva para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros.

Ahora que ya se ha definido qué es JWT, vamos a describir cómo funciona el proceso completo con un diagrama de uso:

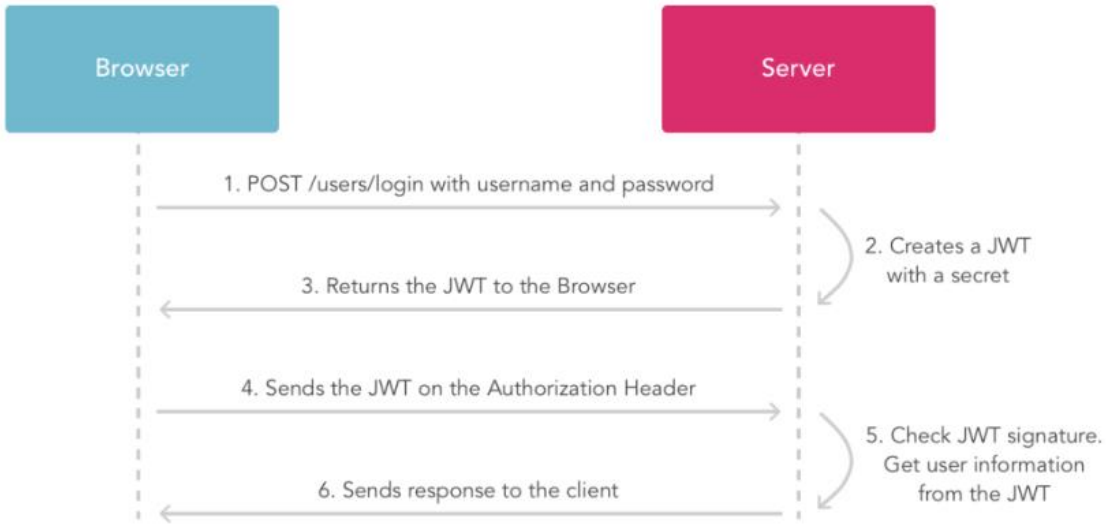


Ilustración 19: Diagrama de JWT

El proceso completo del JWT consta de estos pasos:

1. El usuario de una aplicación web/móvil/desktop hace login con sus credenciales en el servidor donde esta publicada el API.
2. El usuario es validado en el servidor y se crea un nuevo Token JWT (usando el "secret-key") para entregárselo al usuario.
3. El servidor retorna el JWT firmado que contiene los datos referentes al usuario y caducidad del Token.
4. El cliente/browser almacena el JWT para su uso y lo envía en cada petición mediante "Authorization: Bearer.
5. El servidor verifica la firma del Token, su caducidad y comprueba si usuario tiene permisos para acceder al recurso leyendo los datos del payload (información de los privilegios del token).
6. El servidor responde al cliente la petición una vez ha confirmado el Token y comprobando que los permisos del usuario son correctos.

Capítulo 10: Resultados, conclusiones y trabajo futuro

10.1. Resultados y conclusiones

Al concluir este TFG, podemos decir que se han cumplido todos los objetivos propuestos ya que se han llevado a cabo satisfactoriamente. Se ha logrado el propósito que se perseguía, el cual consistía en implementar una mejor gestión entre los profesores del centro y los tutores legales.

Como consecuencia del trabajo de fin de grado realizado, hay que destacar que ha sido un reto poder llevar a cabo todas las tareas que fueron definidas en la fase de análisis. Asimismo, este TFG ha sido desarrollado a partir de otro, por tanto, se ha partido de una idea ya desarrollada. Esto ha supuesto un estudio bastante completo de la herramienta actual, por tanto, hemos tenido que dedicarle tiempo a entender el funcionamiento y el estado de la implementación realizada hasta el momento.

Gracias al uso de las buenas prácticas llevadas a cabo en el desarrollo de la primera fase de la herramienta, ha sido bastante fácil entender la estructura del proyecto. Además, hemos aprendido a ver el punto de vista que tienen otros programadores y en algunos aspectos mejorar y refinar la herramienta.

El paso por el grado nos ha ayudado en gran medida a realizar este proyecto. Es verdad que en el grado no vimos ninguna de las herramientas utilizadas en profundidad, pero gracias a que nos formaron para entender la base de toda herramienta o sistema, se pudo desarrollar satisfactoriamente la herramienta desarrollada en este TFG. Asimismo, el grado nos ha ayudado a saber que tecnología de desarrollo elegir en cada momento y como hacer uso de esta de la manera más correcta.

También, nos hemos dado cuenta que cada asignatura de la carrera aporta su granito de arena en la realización de un proyecto. Ya sea a la hora detectar los tipos de usuarios que harán uso del sistema, así como las acciones que realizarán cada uno de ellos, saber que infraestructura es necesaria para la puesta en marcha del sistema, las tecnologías y herramientas que forman parte del desarrollo, etc.

Por último, gracias a que se han utilizado las versiones más actualizadas de las tecnologías usadas en este proyecto, hemos conseguido hacer que la herramienta desarrollada en este TFG sea más fácil de actualizar y así hacer posible que otras personas puedan seguir desarrollando nuevas funcionalidades en el futuro.

A modo de conclusión, podemos decir que se ha realizado un trabajo bastante satisfactorio, dado que hemos cumplido con los objetivos propuestos. Además de servirnos de motivación y superación al darnos cuenta que pese a los problemas que han ido surgiendo durante el desarrollo, siempre se ha logrado obtener una solución a base de dos pilares fundamentes como son la constancia y el esfuerzo.

Trabajos futuros

Concluido el proyecto se han detectado una serie de mejoras y actualizaciones que podrían implementarse en la herramienta desarrollada, dichas mejoras son las siguientes:

- Implementar un sistema de notificaciones *push* para recibir notificaciones de los mensajes emitidos por el centro y también para el control de las citas con los profesores.
- Implementar una funcionalidad para exportar los datos según la nueva normativa de la RGPD.
- Implementar un sistema de estadísticas para poder controlar que tutores legales han leído los mensajes enviados por el centro y así saber cuántos hacen uso de la aplicación móvil.
- Poder adjuntar más de un fichero a los mensajes enviados por el centro.

Bibliografía

[1]. Apliaula. “Plataforma de gestión escolar, funciones”, [en línea]. Disponible en: http://www.apliaula.com/index/funciones

[2]. Colegio La Cuesta. “Guía básica”, [en línea]. Disponible en: http://pmaria-lacuesta.org/pdf/Appeducamos.pdf

[3]. Educanlia. “Padres y tutores”, [en línea]. Disponible en: http://educanlia.es/funcionalidades/padres-y-tutores/

[4]. Esemtia | school | esemtia – grupo edebé. Disponible en: https://www.esemtia.com/index.php/esemtia-school/

[5]. Dinantia. “Funcionalidades para el centro”, [en línea]. Disponible en: https://www.dinantia.com/es/funcionalidades/centro

[6]. Dinantia. “Funcionalidades para las familias”, [en línea]. Disponible en: https://www.dinantia.com/es/funcionalidades/familias

[7]. Wikipedia. “Licencia de software”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Licencia_de_software

[8]. JetBrains. “TOOLBOX SUBSCRIPTION LICENSE AGREEMENT FOR EDUCATION”, [en línea]. Disponible en: https://www.jetbrains.com/student/license_educational.html

[9]. Wikipedia. “GNU General Public Licence”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/GNU_General_Public_License

[10]. Wikipedia. “XAMPP”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/XAMPP

[11]. Wikipedia. “Software propietario”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Software_propietario

[12]. Wikipedia. “Software gratis”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Software_gratis

[13]. Wikipedia. “Licencia MIT”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Licencia_MIT

[14]. Wikipedia. “Licencia PHP”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Licencia_PHP

[15]. Wikipedia. “Apache License”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Apache_License

[16]. Merlos. “Que objetivos quiere conseguir el nuevo Reglamento General de Protección de Datos”, [en línea]. Disponible en: http://www.merlos.net/2018/04/25/que-objetivos-quiere-conseguir-el-nuevo-reglamento-general-proteccion-datos-rgpd/

[17]. Wikis. “Niveles de seguridad de los ficheros”, [en línea]. Disponible en: http://wikis.fdi.ucm.es/ELP/Niveles_de_seguridad_de_los_ficheros

[18]. Wikipedia. “Modelo de prototipos”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Modelo_de_prototipos

[19]. TSGroup. “La Arquitectura REST”, [en línea]. Disponible en: http://www.tsgroup.com.co/wps/portal/tsg/blog/detalle-blog/la-arquitectura-rest

[20]. GitBooks. “Arquitectura de una API REST”, [en línea]. Disponible en: https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html

[21]. GaussWebApp. “Arquitectura REST: Concepto y fundamentos”, [en línea]. Disponible en: https://gausswebapp.com/arquitectura-rest.html

[22]. Wikipedia. “Modelo-Vista-Controlador”, [en línea]. Disponible en: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[23]. SG Buzz. “Angular: Mucho más que un framework”, [en línea]. Disponible en: https://sg.com.mx/revista/56/angular
[24]. Angular. “Architectura overview”, [en línea]. Disponible en: https://angular.io/guide/architecture
[25]. Grafix. “Los 10 principios de usabilidad de Jakob Nielsen”, [en línea]. Disponible en: http://www.grafix.es/los-10-principios-de-usabilidad-de-jakob-nielsen
[26]. SEMrush. “Principios de usabilidad web de Jakob Nielsen”, [en línea]. Disponible en: https://es.semrush.com/blog/usabilidad-web-principios-jakob-nielsen
[27]. Offing. “11 razones para utilizar el framework Symfony 3”, [en línea]. Disponible en: https://www.offing.es/11-razones-para-utilizar-symfony-3
[28]. Github. “Symfony: Comandos comunes en consola”, [en línea]. Disponible en: https://gist.github.com/javierdaza/4866065f344ecaff0f1f
[29]. El blog de Aitana. “Ventajas de utilizar Angular, un framework JavaScript”, [en línea]. Disponible en: https://blog.aitana.es/2018/04/10/ventajas-de-utilizar-angular/
[30]. CampusMVP. “Las 5 principales ventajas de usar Angular para crear aplicaciones web”, [en línea]. Disponible en: https://www.campusmvp.es/recursos/post/las-5-principales-ventajas-de-usar-angular-para-crear-aplicaciones-web.aspx
[31]. EnMiLocalFunciona. “Contruyendo una Web API REST segura con JSON Web Token”, [en línea]. Disponible en: http://enmilocalfunciona.io/construyendo-una-web-api-rest-segura-con-json-web-token-en-net-parte-i/

Anexo I: Manual de usuario

Introducción

Este manual de usuario ha sido creado con el fin de mostrar al usuario de una manera clara y sencilla las distintas acciones que puede realizar en la web destinada a la administración del centro. Cada explicación vendrá acompañada de imágenes para lograr un mejor entendimiento de las funcionalidades del sistema.

Inicio de sesión

Para acceder al portal web se necesitan unas credenciales de administrador que tendrá que ser solicitadas con anterioridad.

Teniendo las credenciales de administrador podremos ingresar en el portal introduciendo el nombre de usuario y la contraseña facilitada como se puede ver en la *Ilustración 20*.

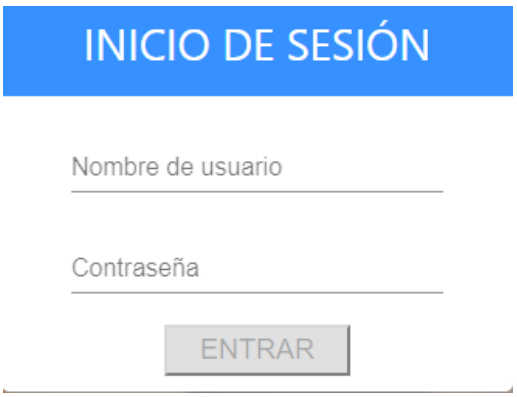


Ilustración 20: Formulario de inicio de sesión

Estructura del portal web

El portal está divido en tres partes (ver *Ilustración 21*):

- Menú de navegación lateral (representado con el color naranja), este es el encargado de facilitar la navegación del portal. Aquí se diferencian dos subapartados:
    - Apartado denominado “Centro”: aquí es donde se encuentran las páginas relacionadas con la gestión del centro (cursos, alumnos, padres, profesores y actualización de los alumnos de un curso).
    - Apartado denominado “Mensajería”: aquí es donde se encuentras las páginas relacionadas con la comunicación entre el centro y los padres (envío de mensajes y visualización de respuestas).
- Barra de navegación superior (representado con el color verde), encargada de ocultar el menú lateral, informar que usuario está conectado en la plataforma y contiene un menú desplegable. En este menú desplegable el administrador del centro podrá editar sus datos, así como restablecer su contraseña y cerrar sesión.
- Por último, centralizado en la página está el contenido que hay en cada sección de la web (representado con el color azul).

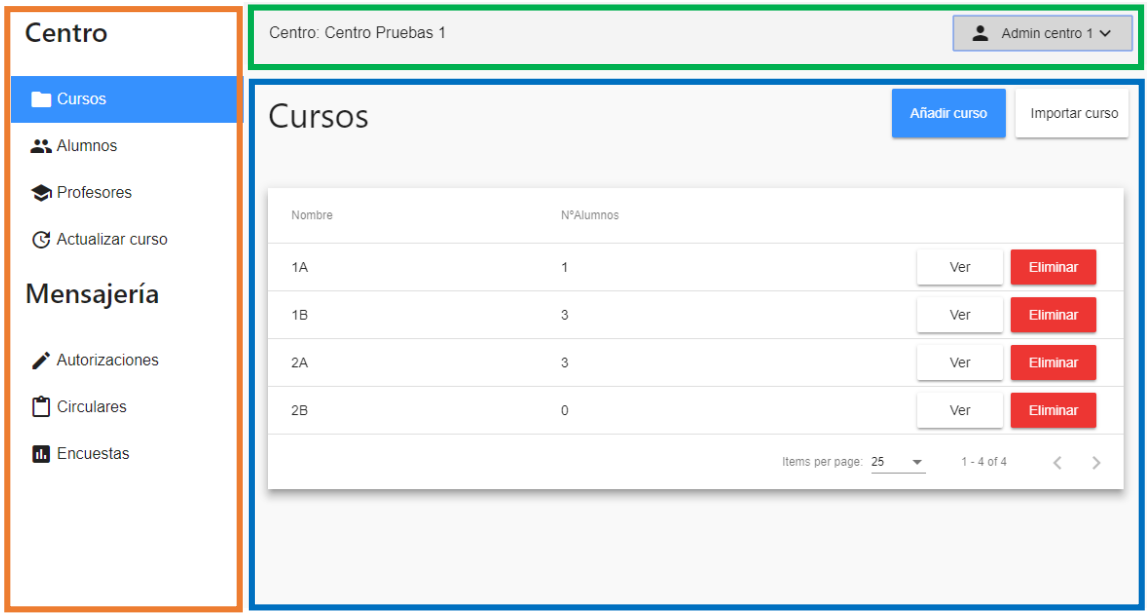


Ilustración 21: Estructura del portal

Cursos

Esta es la página predefinida cuando se inicia sesión en el portal como se muestra en la *Ilustración 22*. Aquí se podrán realizar las siguientes acciones:

- Ver una lista con los cursos registrados en el sistema con el nombre y número de alumnos que hay en el mismo.
- Añadir un nuevo curso asociándole un profesor.
- Importar los datos del centro con todas las asociaciones que ello conlleva (nombre del curso, profesor del curso, alumnos del curso y padres de los alumnos). El fichero que se va a importar tiene que está en formato CSV.
- Ver una lista de los alumnos que forman parte de un curso.
- Eliminar un curso.

En cada nueva página habrá una serie de botones en los que se podrán realizar acciones. Se explicará cada acción posible de la manera más clara posible.

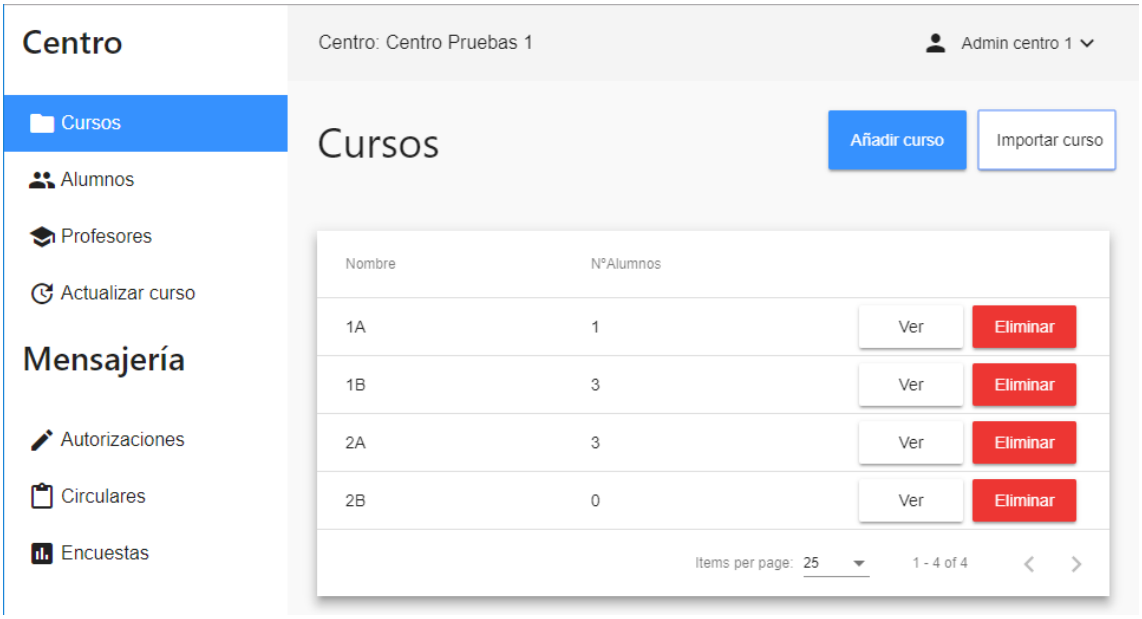


Ilustración 22: Página de cursos

Haciendo *click* en el botón "Añadir curso" se desplegará un modal como se puede observar en la *Ilustración 23* en el que mostrará un formulario para definir el nombre del curso y un campo *autocomplete* (representado con un borde naranja) para escoger a uno de los profesores registrados en el sistema, este es quién será el tutor de curso.

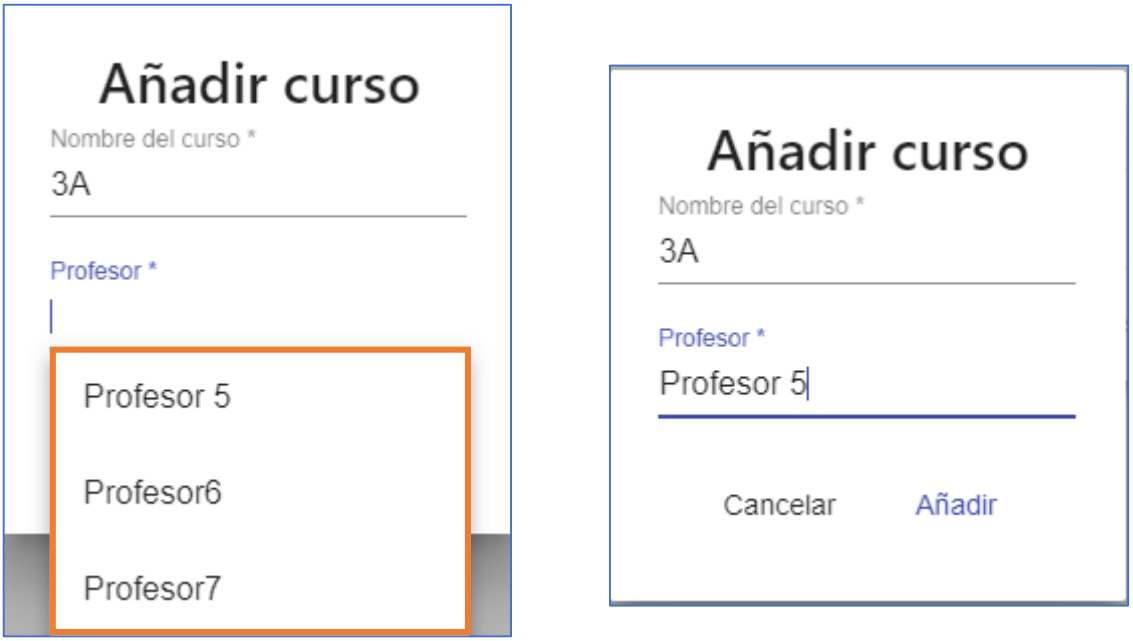


Ilustración 23: Diálogo añadir curso

Haciendo *click* en el botón "Importar curso", se desplegará un diálogo del sistema de ficheros como se puede ver en la Ilustración 24, donde se podrá escoger un fichero .csv con los datos del centro.

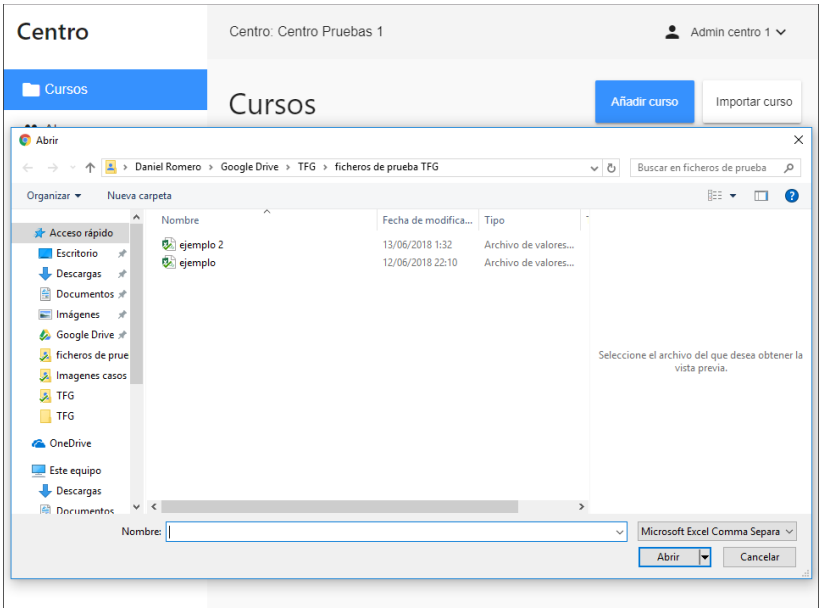


Ilustración 24: Diálogo sistema de ficheros (Importar curso)

Haciendo *click* en el botón "Eliminar" asociado a cada curso, se desplegará un diálogo donde saldrá una advertencia como se puede observar en la *Ilustración 25* dado que se trata de una acción irreversible.

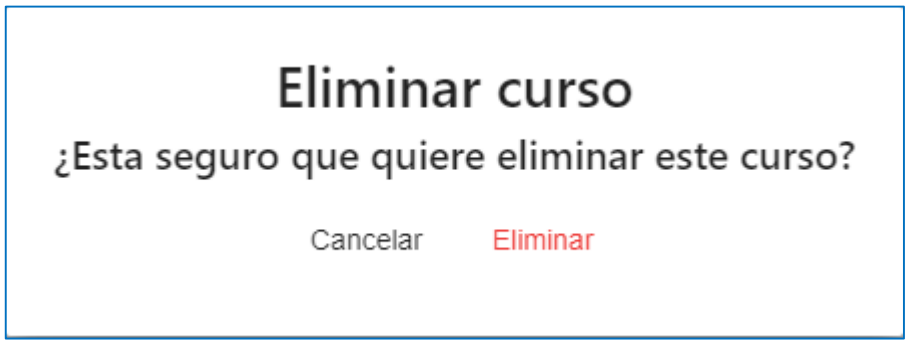


Ilustración 25: Diálogo eliminar curso

Haciendo *click* en el botón "Ver" asociado a cada curso, se navegará a otra página del portal donde se muestra una lista con los alumnos que hay en el curso como se puede observar en la *Ilustración 26*.

Aquí se muestra una lista con los alumnos que hay registrados en un curso. En esta vista se pueden realizar las siguientes opciones:

- Añadir un alumno al curso.
- Ver la información de un alumno.
- Desasociar un alumno del curso.
- Asociar profesor al curso.

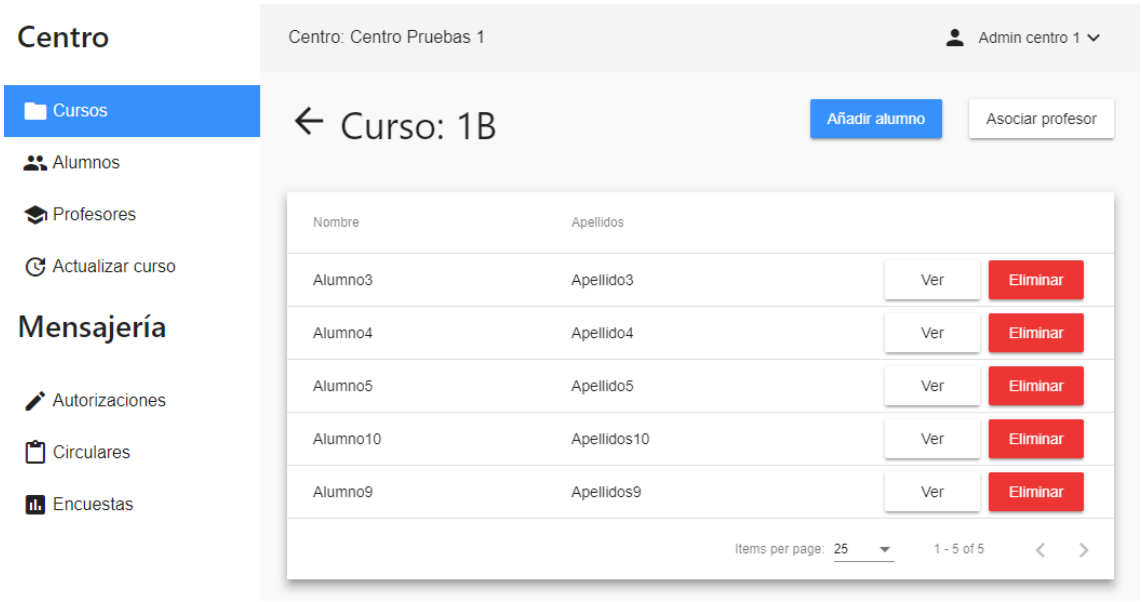


Ilustración 26: Página de un curso

Haciendo *click* en el botón "Añadir alumno" se desplegará un diálogo como se puede observar en la *Ilustración 27,* donde habrá un campo *autocomplete* (representado con un borde naranja) para seleccionar uno de los alumnos registrados en el sistema que no estén asociados a ningún curso todavía.

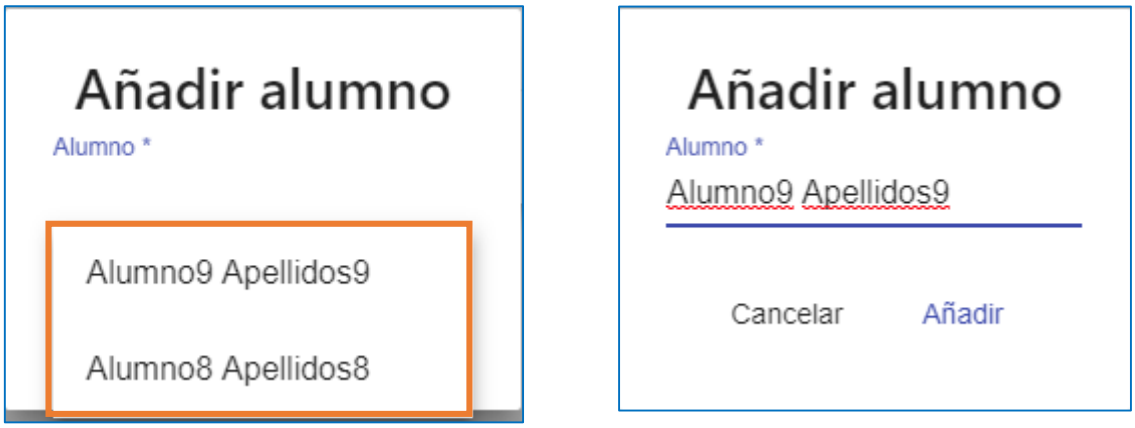


Ilustración 27: Diálogo añadir alumno

Haciendo *click* en el botón "Asociar profesor", se desplegará un diálogo como se puede observar en la *Ilustración 28,* donde habrá un campo *autocomplete* (representado con un borde naranja) para seleccionar uno de los profesores registrados en el sistema que no estén asociados a ningún curso todavía. Cabe destacar que, si el curso ya tiene un profesor asociado, y se asocia otro, el curso tendrá el último profesor que haya sido asociado.

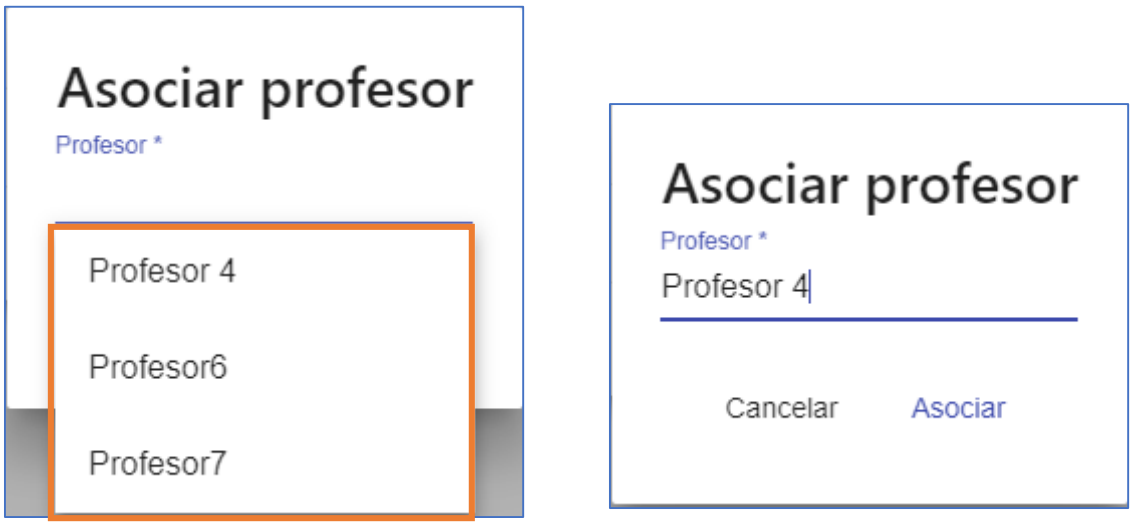


Ilustración 28: Diálogo asociar profesor

Haciendo *click* en el botón "Ver" asociado a cada alumno de la tabla, se navegará a otra página del portal donde se muestra la información registrada en el sistema del alumno como se puede observar en la *Ilustración 29*. Las acciones que pueden ser realizadas en esta página se explicarán más adelante en el apartado de alumnos.

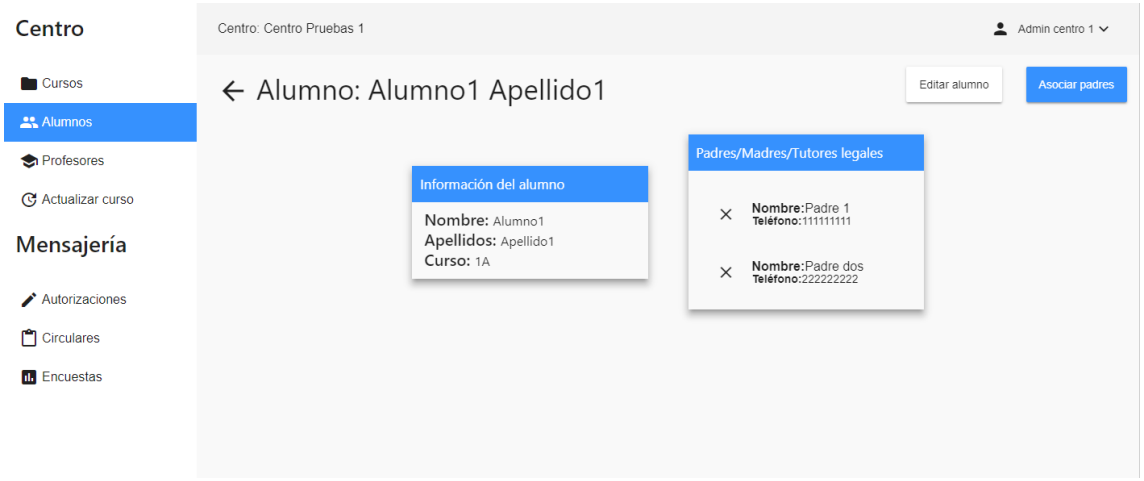


Ilustración 29: Información del alumno

Haciendo *click* en el botón "Eliminar" asociado a cada alumno de la tabla, se desplegará un diálogo como se puede observar en la *Ilustración 30*, donde saldrá una advertencia dado que se trata de una acción irreversible.

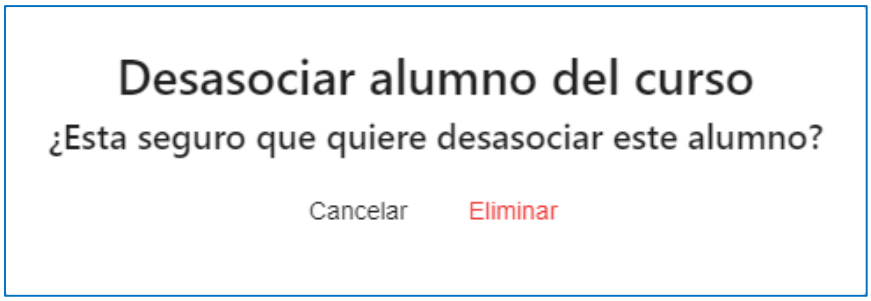


Ilustración 30: Diálogo desasociar alumno de un curso

Alumnos

Esta sección del portal está destinada a los alumnos registrados en el centro (ver *Ilustración 31*). En este apartado se pueden realizar las siguientes acciones:

- Ver una tabla con los alumnos registrados en el sistema.
- Filtrar los alumnos registrados en el sistema (por curso y nombre o apellidos).
- Registrar un alumno en el sistema.
- Ver la información del alumno.
- Eliminar un alumno del sistema.

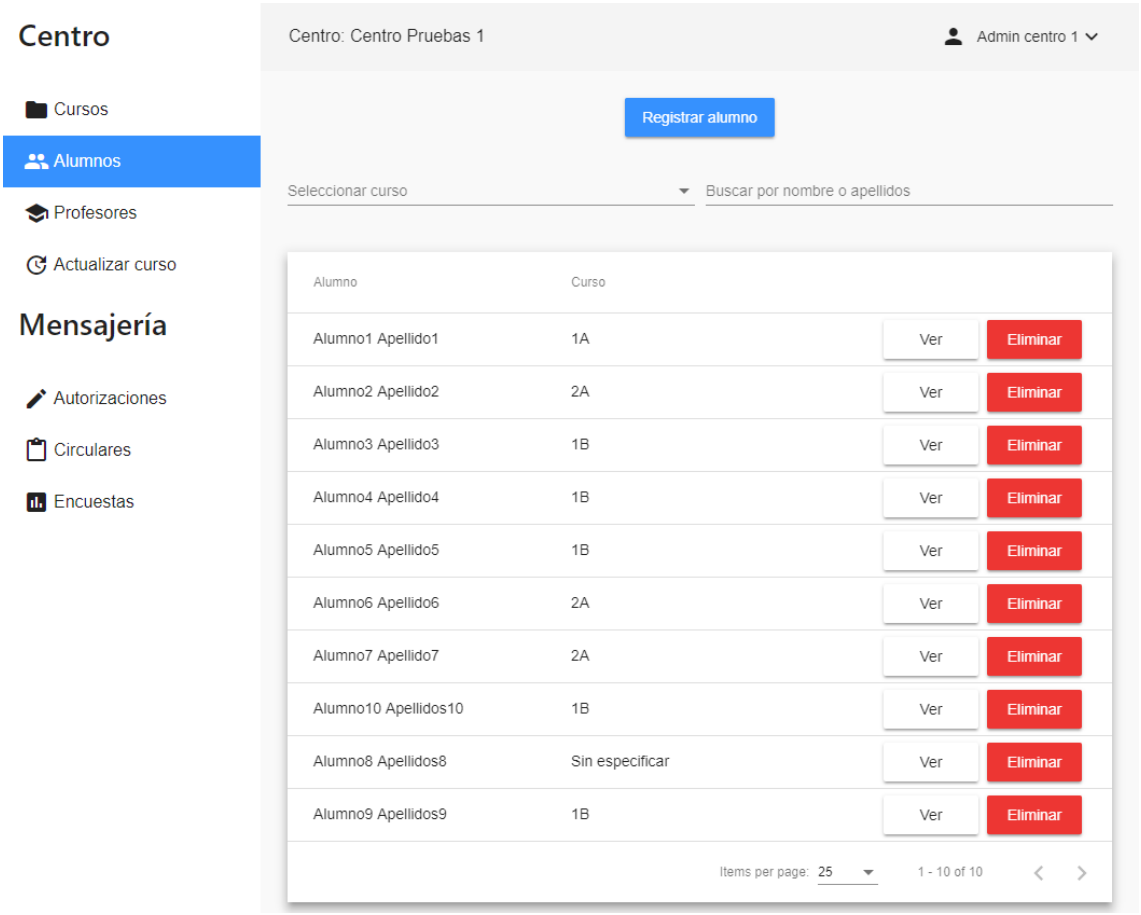


Ilustración 31: Página de alumnos

Haciendo *click* en el botón “Registrar alumno” se abrirá un diálogo como se puede observar en la *Ilustración 32*, donde se podrá registrar un nuevo alumno en el sistema. Este diálogo permite introducir el nombre y apellidos del estudiante además de dar la opción de asociarlo a un curso y asociarle los tutores legales.

Para asociar tutores legales desde este diálogo lo único que hay que hacer es introducir el teléfono del padre/madre/tutor legal y si este se ha registrado desde la aplicación saldrá su nombre automáticamente en el apartado nombre y apellidos.

Una vez que salga el nombre y apellidos del tutor legal bastaría con hacer *click* en el botón “Añadir padre”, ahora saldrá una lista debajo de este botón con los nombres de los tutores legales añadidos con la posibilidad de borrar el legal de la lista previsualizada haciendo *click* en el icono de la papelera.

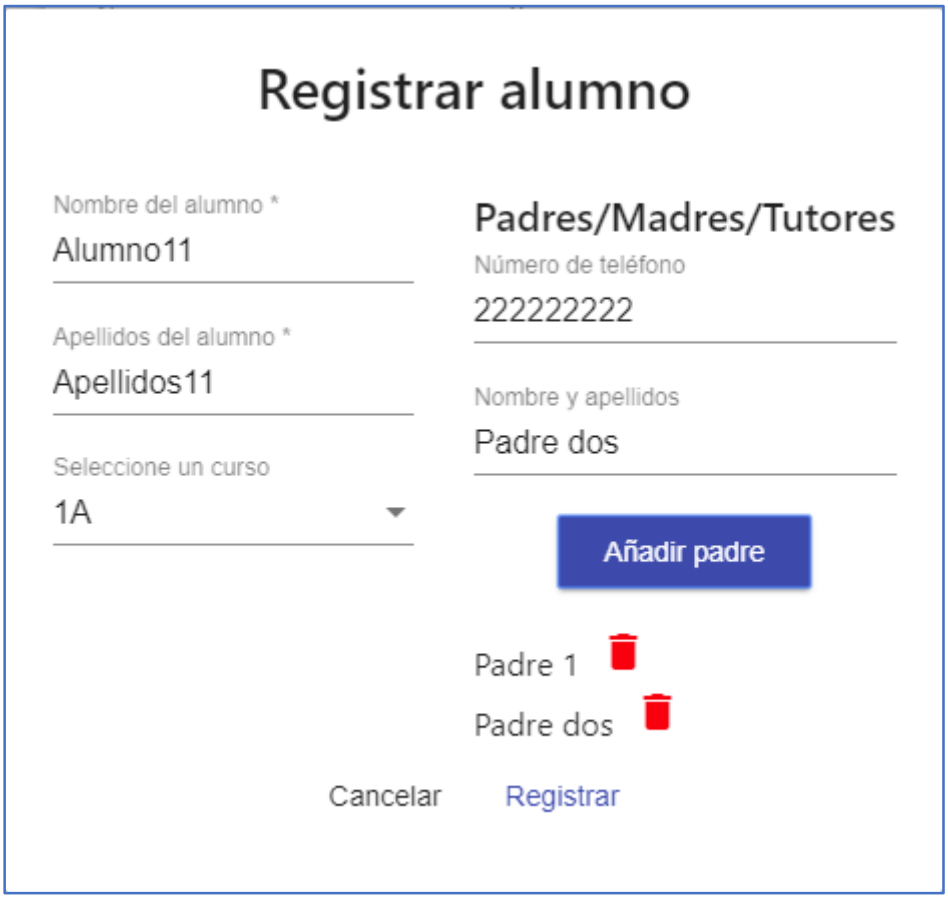


Ilustración 32: Diálogo registrar alumno

Para hacer uso del filtro por curso bastaría con hacer *click* en el desplegable definido con el nombre “Seleccionar curso” como se puede observar en la *Ilustración 33*.

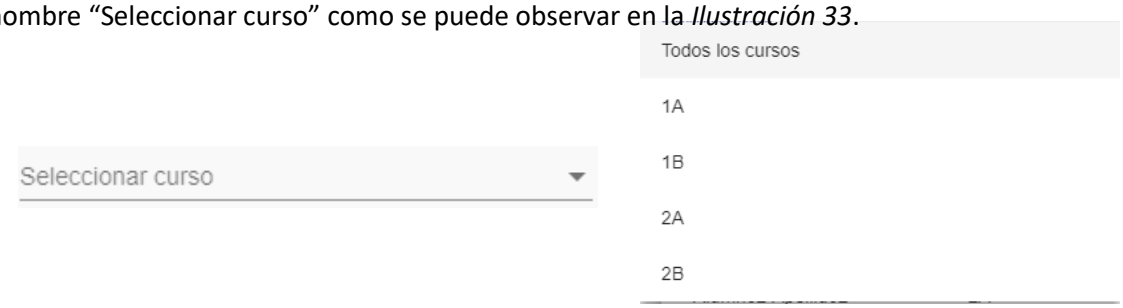


Ilustración 33: Filtro por curso

Para buscar por nombre o apellidos del estudiante bastaría con hacer *click* encima del input definido con el nombre "Buscar por nombre o apellidos" como se puede observar en la *Ilustración 34.*

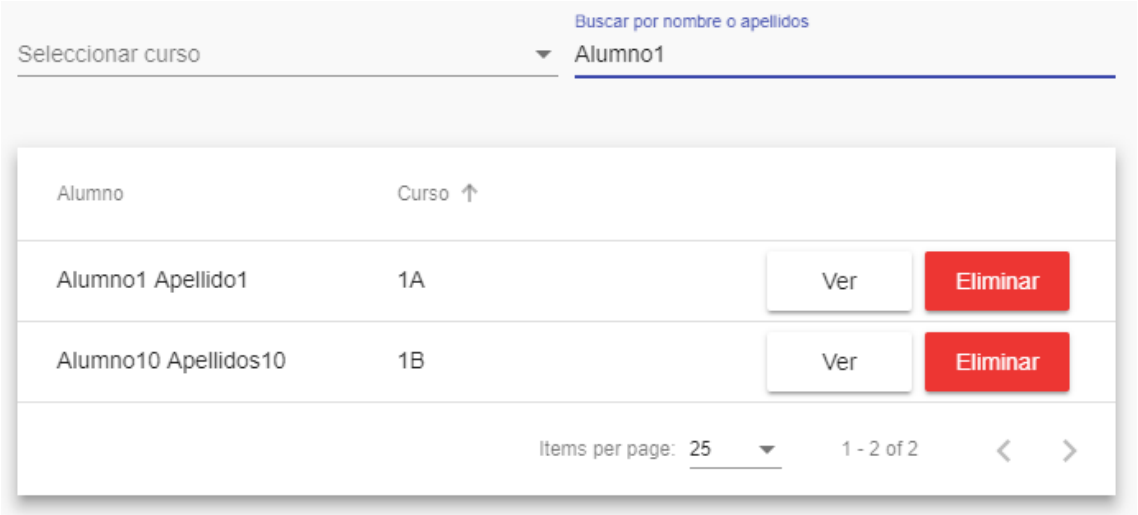


Ilustración 34: Filtro por nombre o apellidos del alumno

Haciendo *click* en el botón ver se navegará hasta la página que se muestra en la *Ilustración 29*. En esta página se podrá editar la información del demandante y asociar y desasociar tutores legales.

Al hacer *click* en el botón "Editar alumno", se abrirá un diálogo como se puede observar en la *Ilustración 35*, para editar el nombre, apellidos del alumno y el curso al que pertenece.

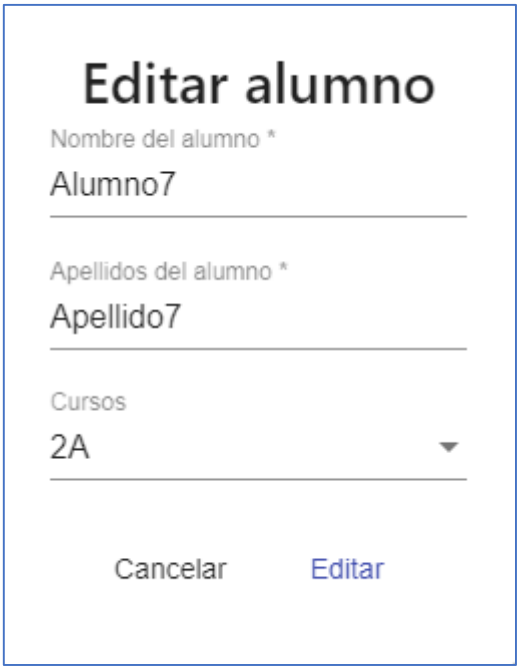


Ilustración 35: Diálogo para editar un alumno

Al hacer *click* en el botón “Asociar padres”, se abrirá un diálogo para asociar tutores legales a un alumno como se puede observar en la *Ilustración 36*. El funcionamiento es igual que en el diálogo de registrar un alumno (*Ilustración 32*) excepto que en este diálogo los tutores legales se asocian de uno en uno.

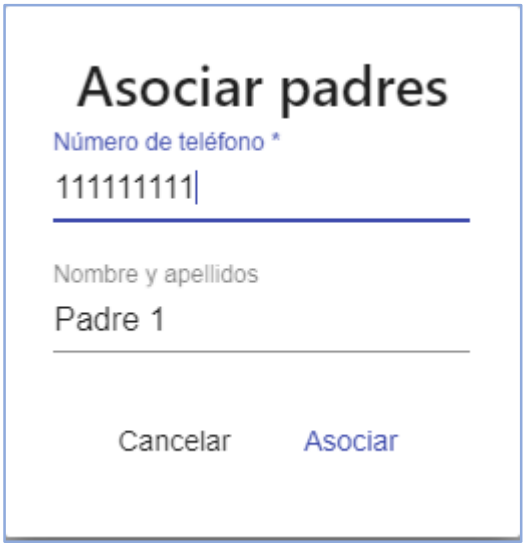


Ilustración 36: Diálogo para asociar tutores legales a un alumno

Al hacer *click* en el icono con el símbolo X se desasocia el padre/madre/tutor legal correspondiente como se puede observar en la *Ilustración 37*.

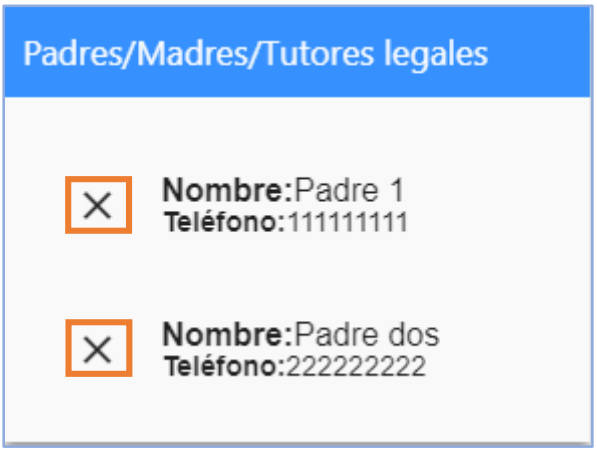


Ilustración 37: Diálogo para asociar tutores legales a un alumno

Volviendo a la vista general de los alumnos registrados en el sistema, queda la opción de eliminar un alumno del sistema, esto se consigue haciendo *click* en el botón “Eliminar” el cual está asociado a cada alumno de la tabla. Una vez hecho *click* en dicho botón se desplegará un diálogo donde saldrá una advertencia dado que se trata de una acción irreversible.

Profesores

Esta sección del portal está destinada a los profesores registrados en el centro como se puede observar en la *Ilustración 38*. En este apartado se pueden realizar las siguientes acciones:

- Ver una tabla de los profesores registrados en el sistema.
- Registrar un profesor en el sistema.

- Eliminar un profesor del sistema.

La tabla que muestra los profesores registrados en el sistema contiene el nombre y correo electrónico del profesor, además también se muestra el nombre del curso en el que está el profesor como tutor. Si el profesor no estuviera asociado a ningún curso, en la columna "Curso" de la tabla se mostraría el mensaje "No tiene asociado un curso".

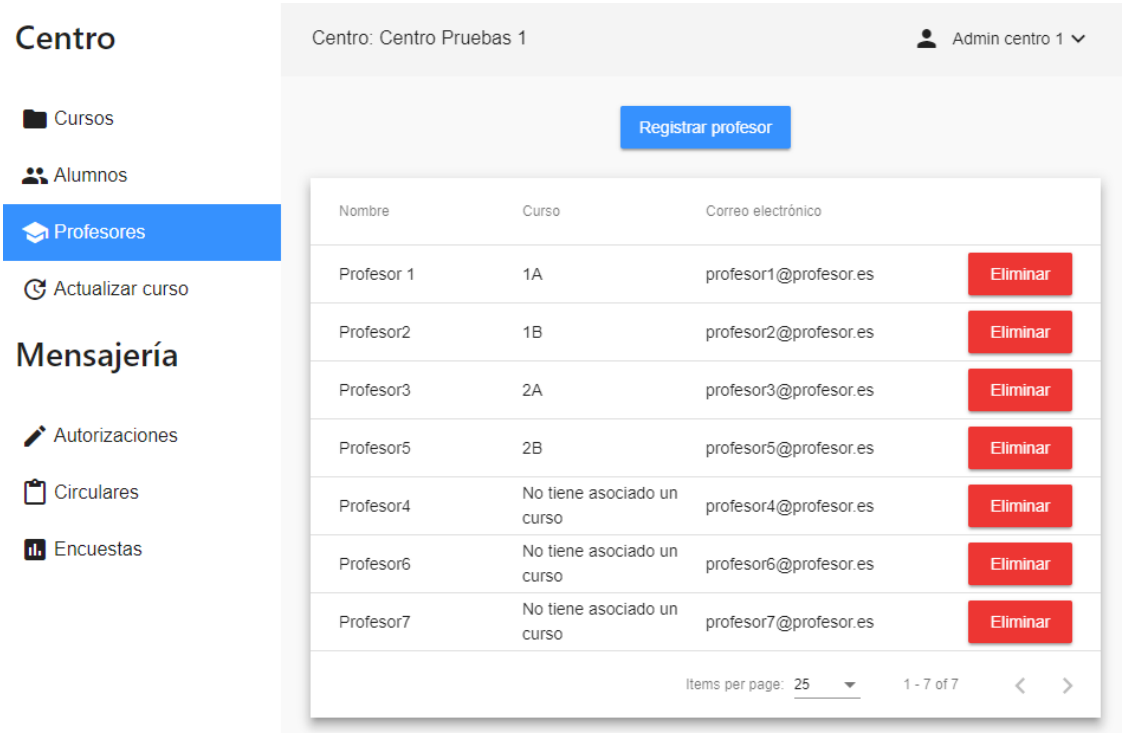


Ilustración 38: Página de profesores

Haciendo *click* en el botón "Registrar profesor" se abrirá un diálogo para registrar un profesor en el sistema como se puede observar en la *Ilustración 39*. Los campos que hay que rellenar en este diálogo son el nombre y apellidos del profesor y su correo electrónico.

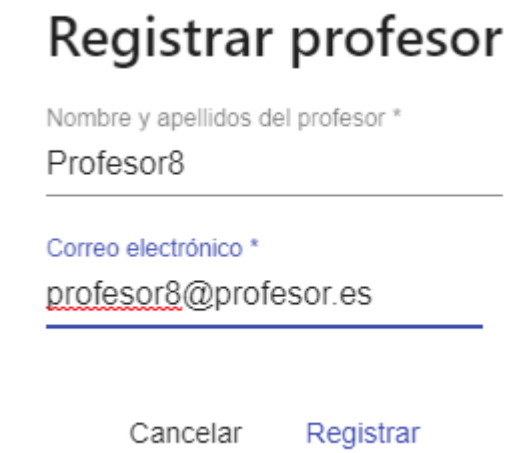
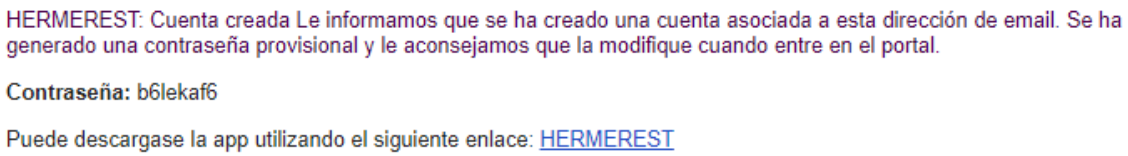


Ilustración 39: Diálogo registrar profesor

Una vez finalizado el registro anterior, al profesor se le enviará un correo con una contraseña generada aleatoriamente para que pueda realizar el inicio de sesión desde la aplicación móvil como se puede observar en la *Ilustración 40, d*esarrollada en el TFG titulado "ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN PARA DISPOSITIVOS MÓVILES MULTIPLATAFORMA PARA LA COMUNICACIÓN ENTRE CENTROS EDUCATIVOS Y PADRES DE ALUMNOS II".


HERMEREST: Cuenta creada Le informamos que se ha creado una cuenta asociada a esta dirección de email. Se ha generado una contraseña provisional y le aconsejamos que la modifique cuando entre en el portal.

Contraseña: b6lekaf6

Puede descargase la app utilizando el siguiente enlace: HERMEREST

Ilustración 40: Correo electrónico cuenta creada

Por último, en esta sección tenemos el botón "Eliminar" asociado a cada profesor registrado en el sistema. Al hacer *click* en el botón "Eliminar" se desplegará un diálogo como se puede observar en la *Ilustración 41,* donde saldrá una advertencia dado que se trata de una acción irreversible.

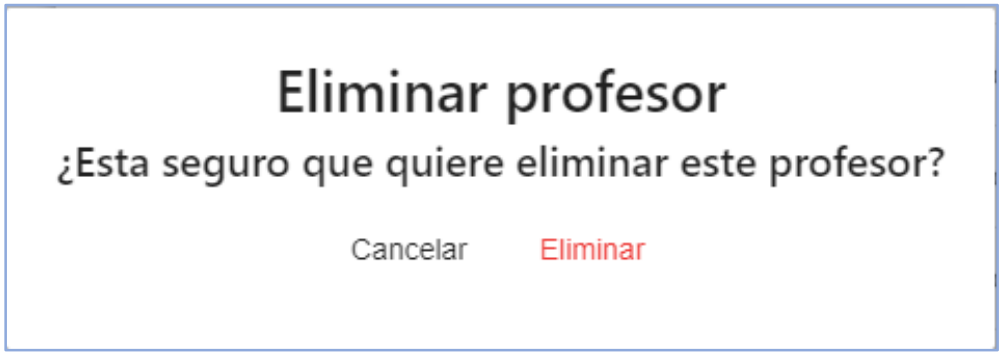


Ilustración 41: Diálogo eliminar profesor

Actualizar curso

En esta sección del portal se ha desarrollado una utilidad *drag&drop* para facilitar el cambio de alumnos de un curso a otro.

Esta utilidad permite cambiar los alumnos de un curso a otro arrastrando y soltando elementos. Este cambio puede realizarse individualmente o por bloque, para ello, lo primero es seleccionar un curso de origen y un curso destino.

Una vez seleccionado el curso origen y el curso objetivo, se podrán pasar los alumnos de uno en uno, solo bastaría con arrastrar el alumno de un curso a otro. Por otra parte, si se quisiera pasar un bloque de alumnos de un curso a otro, bastaría con hacer *click* (al realizar la acción en el fondo del alumno se marcará con un color azul) en aquellos alumnos que quieran ser promocionados de curso y una vez seleccionados los alumnos que se desean cambiar de curso bastaría con arrastrar el bloque de alumnos de un curso a otro.

A continuación, se mostrarán con una serie de imágenes los pasos a seguir para pasar individualmente alumnos de un curso a otro:

1. **Escoger curso de origen y curso destino (ver *Ilustración 42*)**

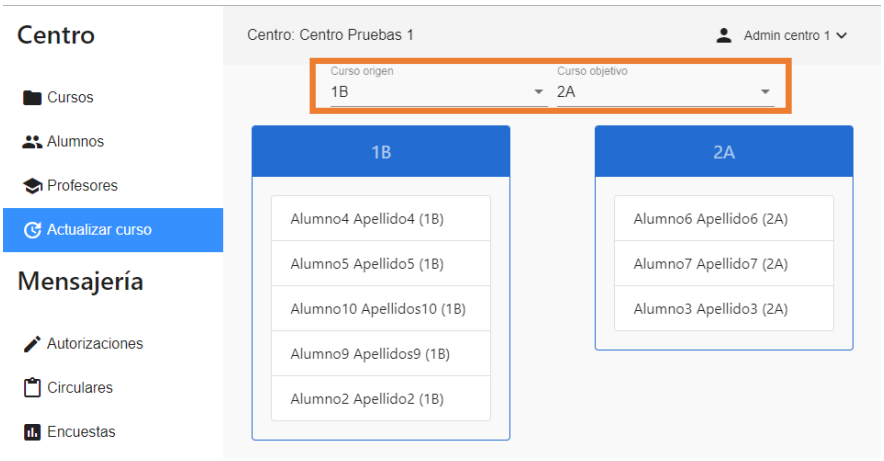


Ilustración 42: Elección de curso origen y curso objetivo

2. **Seleccionar y arrastrar un alumno del curso origen al curso destino o viceversa (ver *Ilustración 43*)**

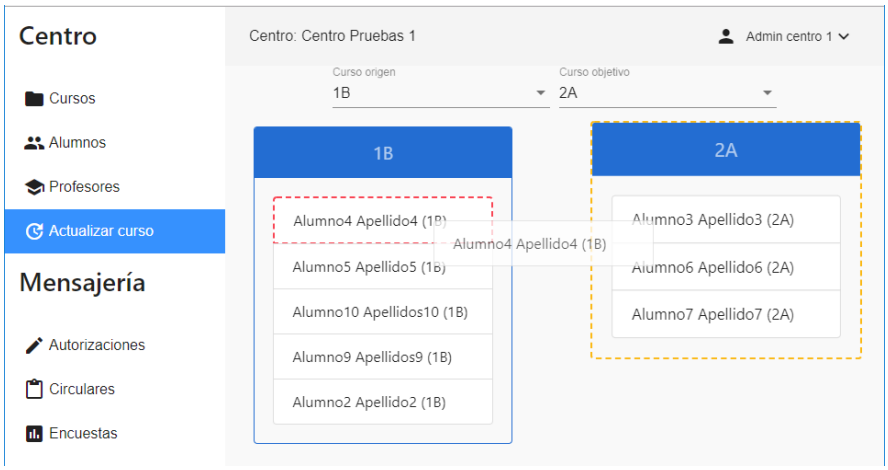


Ilustración 43: Coger y arrastrar alumno

3. **Soltar el alumno en el curso objetivo o viceversa (ver *Ilustración 44*)**

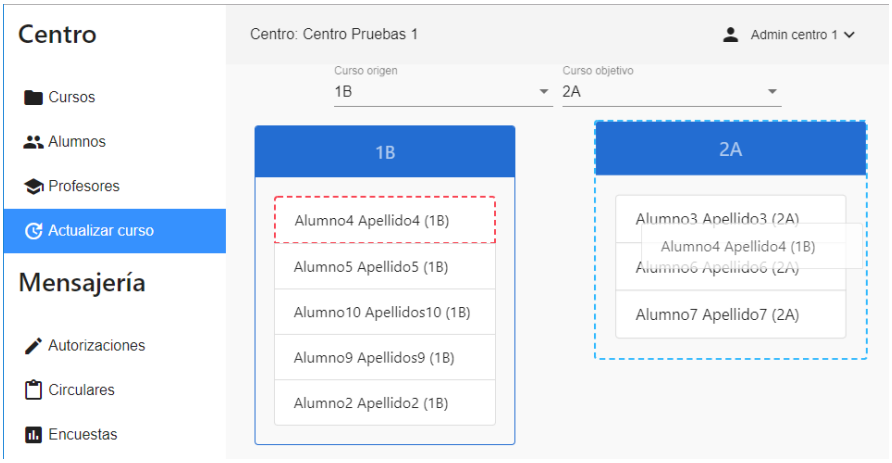


Ilustración 44: Soltar alumno en curso objetivo

4. **Alumno cambiado de curso correctamente (ver Ilustración 45)**

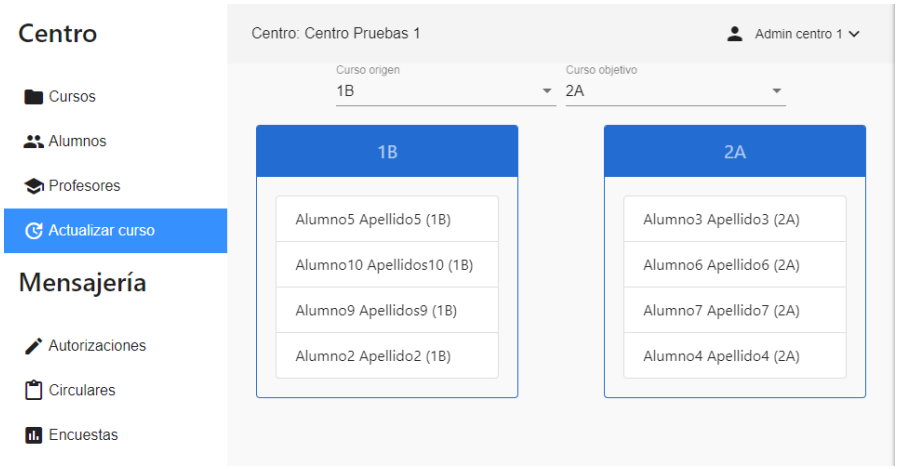


Ilustración 45: Alumno cambiado correctamente de curso

Ahora se explicará cómo cambiar un bloque de alumnos de un curso a otro. Para no repetir todos los pasos anteriores solo redefinirán el paso 2 y 3 mencionados anteriormente dado que el funcionamiento es igual que el anterior, pero con un pequeño matiz.

2. **Hacer *click* en los alumnos que se quieran cambiar de curso (ver *Ilustración 46*)**

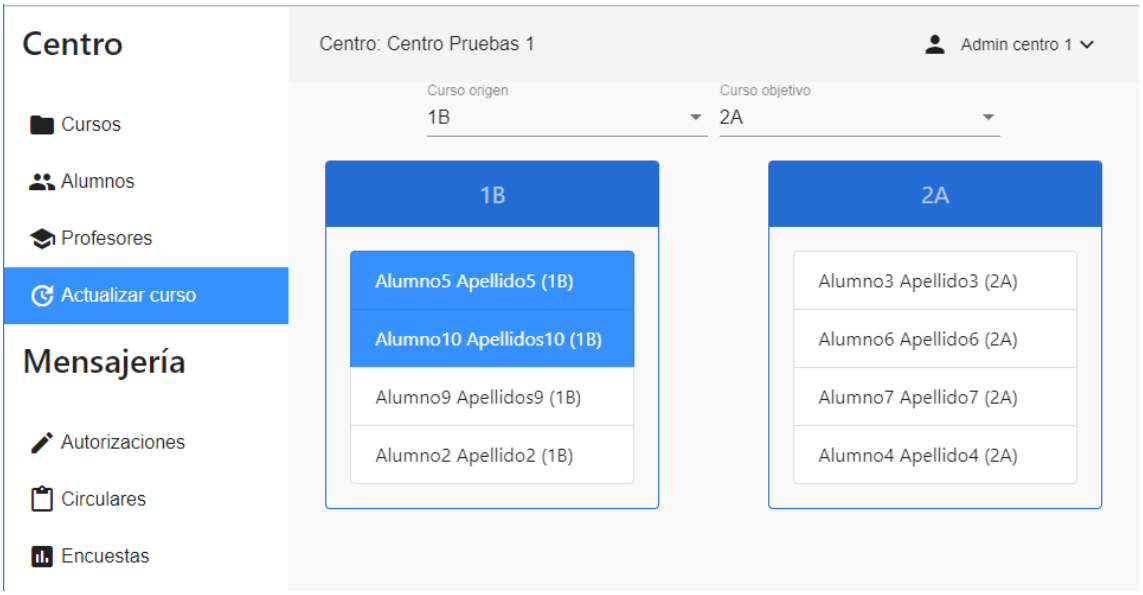


Ilustración 46: Selección de varios alumnos

3. **Coger y arrastrar bloque de alumnos del curso origen al destino y viceversa (ver *Ilustración 47*)**

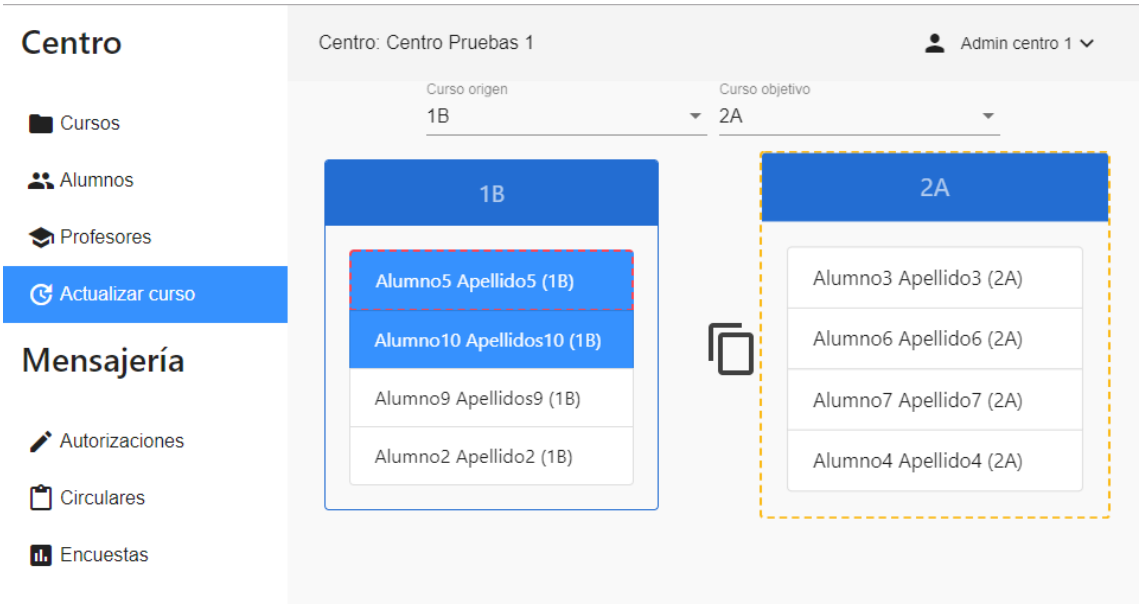


Ilustración 47: Coger y arrastrar bloque de alumnos

Mensajería

Este apartado del portal web está destinado al envío de mensajes por parte del centro para informar a los padres de los alumnos.

Aquí se pueden enviar tres tipos de mensajes:

- Autorizaciones
- Circulares
- Encuestas

A continuación, se explicará cómo se pueden enviar cada uno de los tipos de mensajes nombrados anteriormente.

Autorizaciones

Esta sección proporciona al centro el envió de autorizaciones a los padres de los alumnos como se puede observar en la *Ilustración 48*.

Aquí se pueden realizar las siguientes acciones:

- Enviar una autorización a los padres de los alumnos del centro.
- Filtrar las autorizaciones enviadas.
- Ver una autorización.
- Editar la fecha límite para responder la autorización.

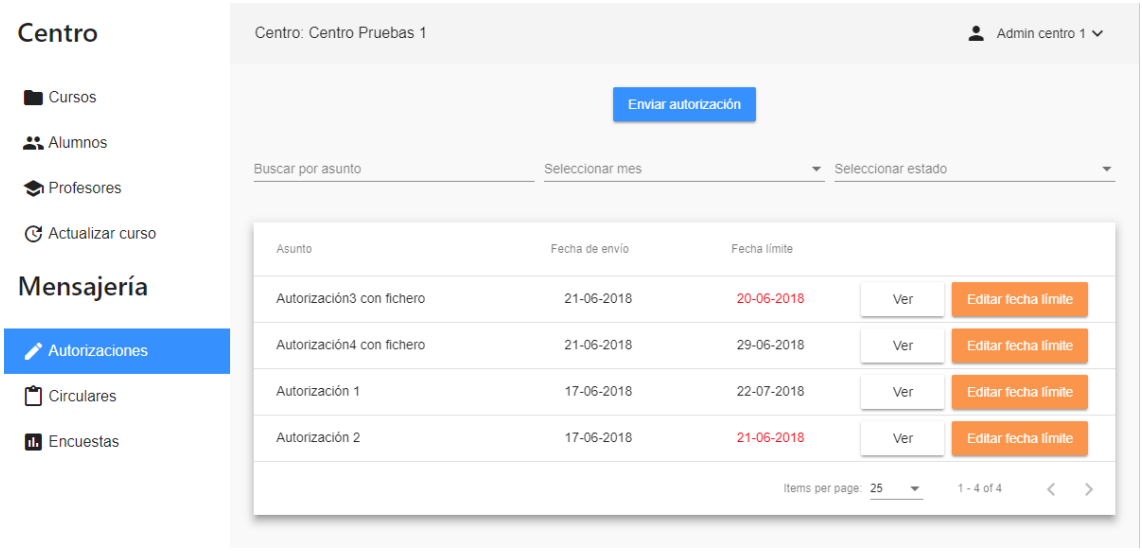


Ilustración 48: Página de autorizaciones

Haciendo *click* en el botón "Enviar autorización" se abrirá un diálogo como se puede observar en la *Ilustración 49,* en el que habrá un formulario a rellenar con los siguientes campos:

- Asunto de la autorización.
- Fecha límite para ser respondida.
- Un campo para añadir un mensaje si fuese necesario.
- Un botón con la posibilidad de adjuntar un fichero.
- Un árbol de *checkboxs* para seleccionar el alumno o los alumnos que quieren ser notificados con el mensaje.

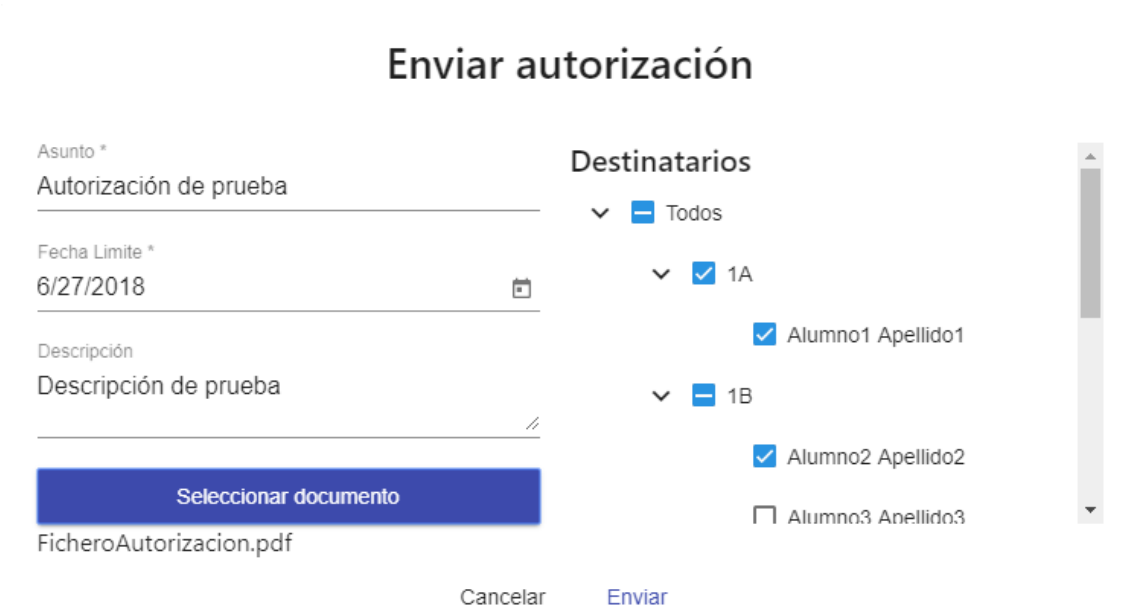


Ilustración 49: Diálogo enviar autorización

Para buscar por asunto de la autorización bastaría con hacer *click* encima del input definido con el nombre "Buscar por asunto" y escribir la cadena de texto por la que se quiere buscar como se puede observar en la *Ilustración 50*.

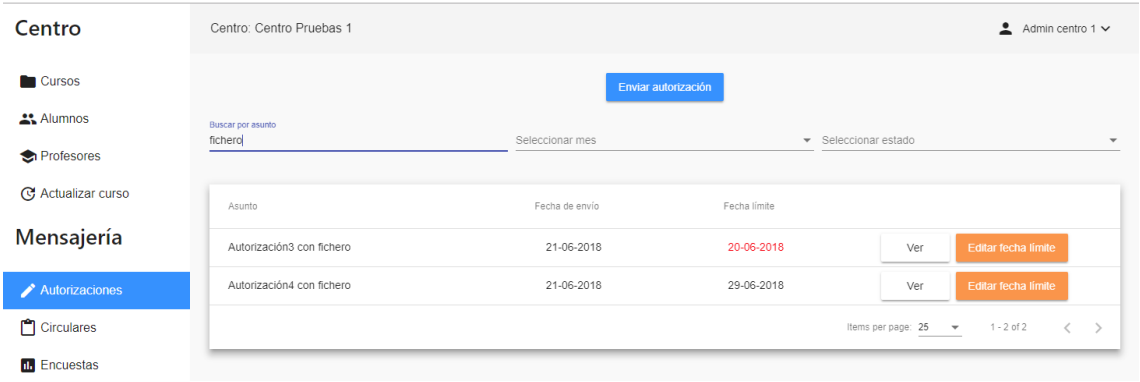


Ilustración 50: Buscar por asunto

Para hacer uso del filtro por mes de envío bastaría con hacer *click* en el desplegable definido con el nombre "Seleccionar mes" y seleccionar una de las opciones como se puede observar en la *Ilustración 51*.

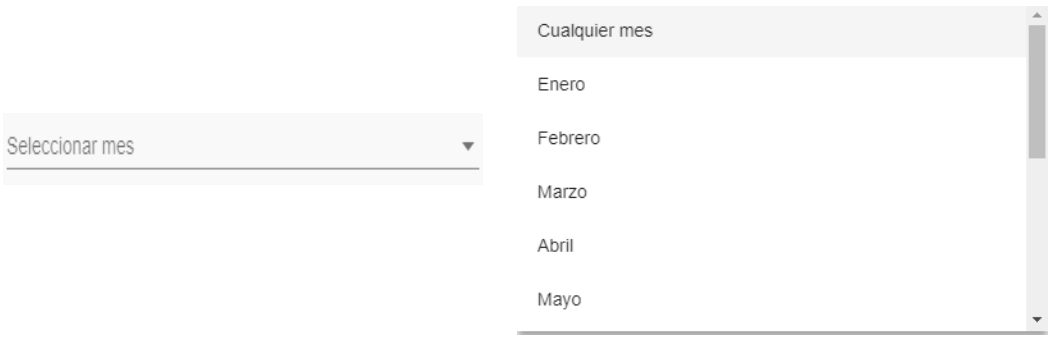


Ilustración 51: Filtrar por mes de envío

Para hacer uso del filtro estado bastaría con hacer *click* en el desplegable definido con el nombre "Seleccionar estado" y seleccionar una de las opciones como se puede observar en la *Ilustración 52*.

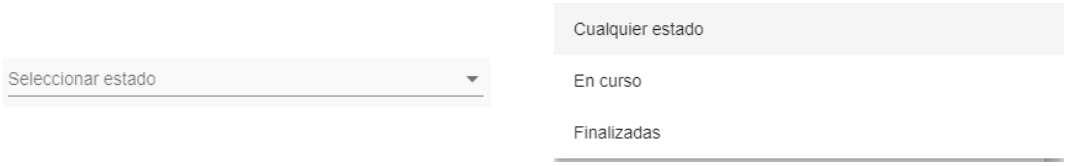


Ilustración 52: Filtrar por estado

Haciendo *click* en el botón "Ver" asociado a cada autorización se abrirá un diálogo mostrando la siguiente información (ver *Ilustración 53*):

- Asunto de la autorización.
- Fecha de envío.
- Fecha límite para ser respondida.
- Archivo adjunto con posibilidad de descargarlo.
- Mensaje de la autorización.
- Listado diferenciado de los alumnos que han sido autorizados y no autorizados.

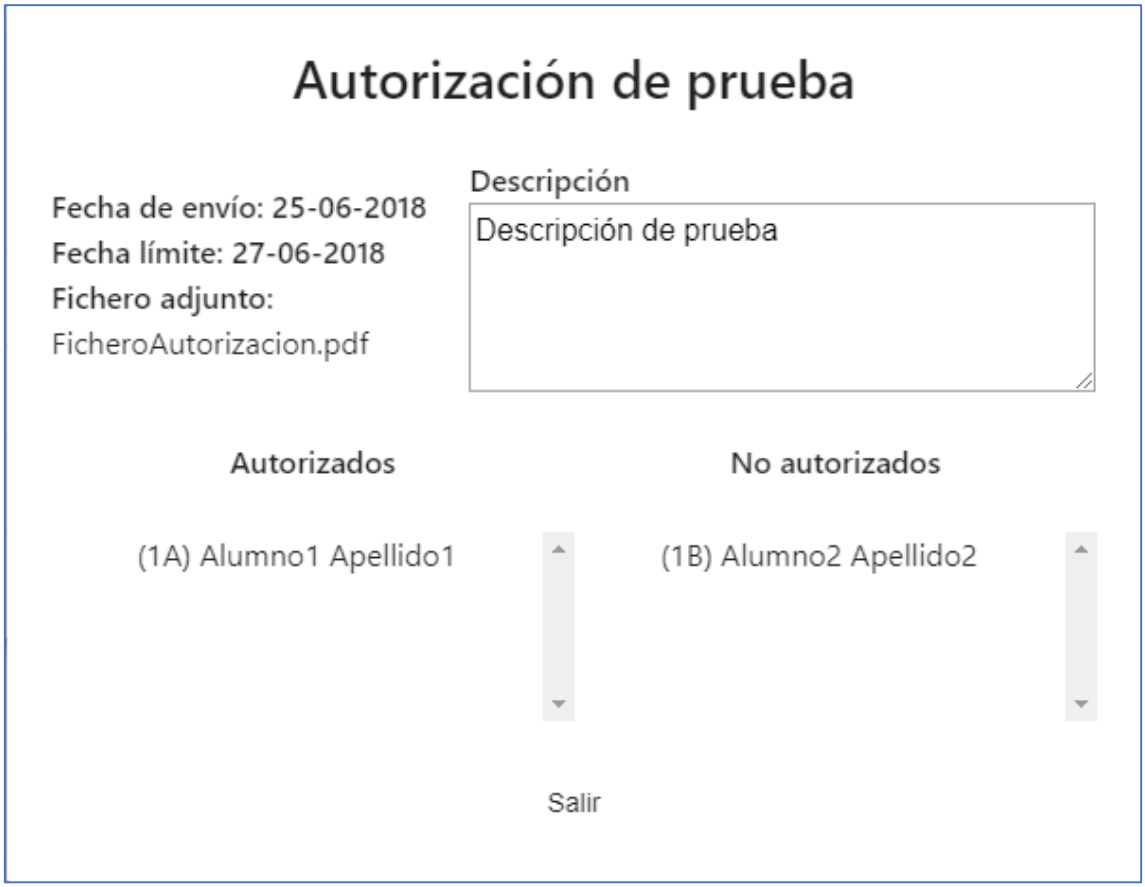


Ilustración 53: Diálogo ver autorización

Haciendo *click* en el botón "Editar fecha límite" se abrirá un modal como se puede observar en la *Ilustración 54* para cambiar la fecha límite por una posterior. Cuando se abre el diálogo, por

defecto sale la última fecha limite definida, si se quiere editar esta fecha bastaría con hacer *click* en el icono del calendario y seleccionar una nueva fecha límite.

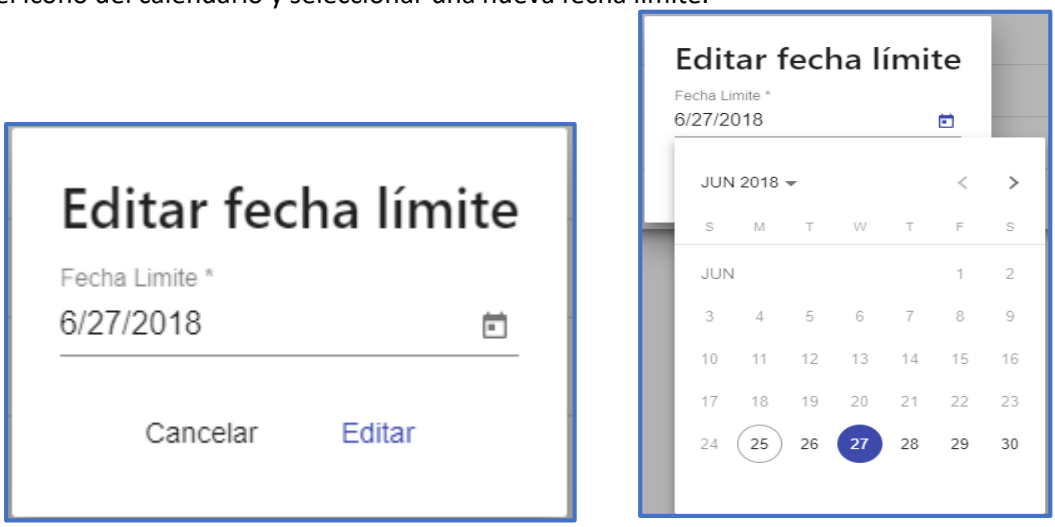


Ilustración 54: Diálogo editar fecha límite

Circulares

Esta sección proporciona al centro él envió de circulares informativas a los padres de los alumnos como se puede observar en la *Ilustración 55*.

Aquí se pueden realizar las siguientes acciones:

- Enviar una circular a los padres de los alumnos del centro.
- Filtrar las circulares enviadas.
- Ver una circular.

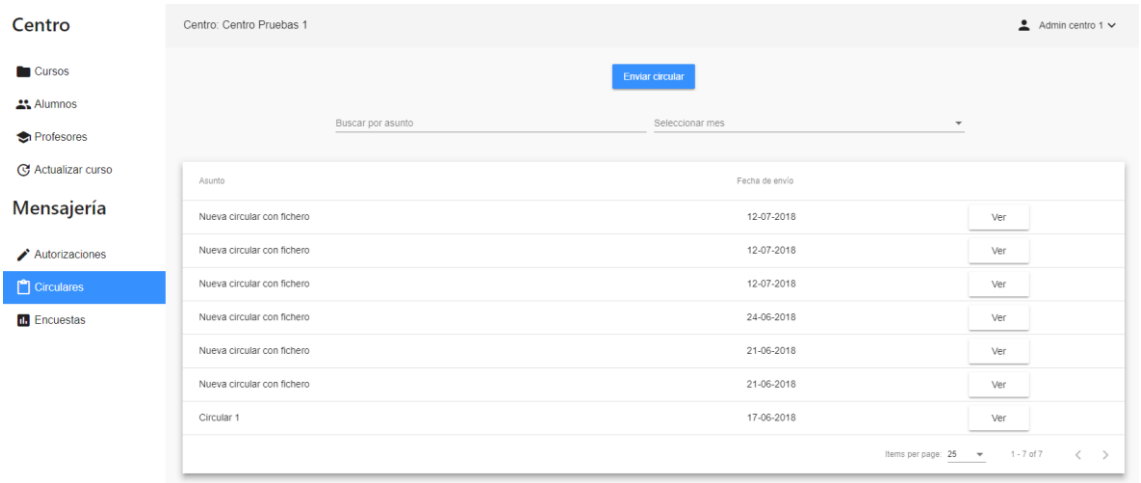


Ilustración 55: Página de circulares

Haciendo *click* en el botón “Enviar circular” se abrirá un diálogo como se puede observar en la *Ilustración 55,* en el que habrá un formulario a rellenar con los siguientes campos:

- Asunto de la circular.
- Un campo para añadir un mensaje si fuese necesario.

- Un botón con la posibilidad de adjuntar un fichero.
- Un árbol de *checkboxs* para seleccionar el alumno o los alumnos que quieren ser notificados con el mensaje.

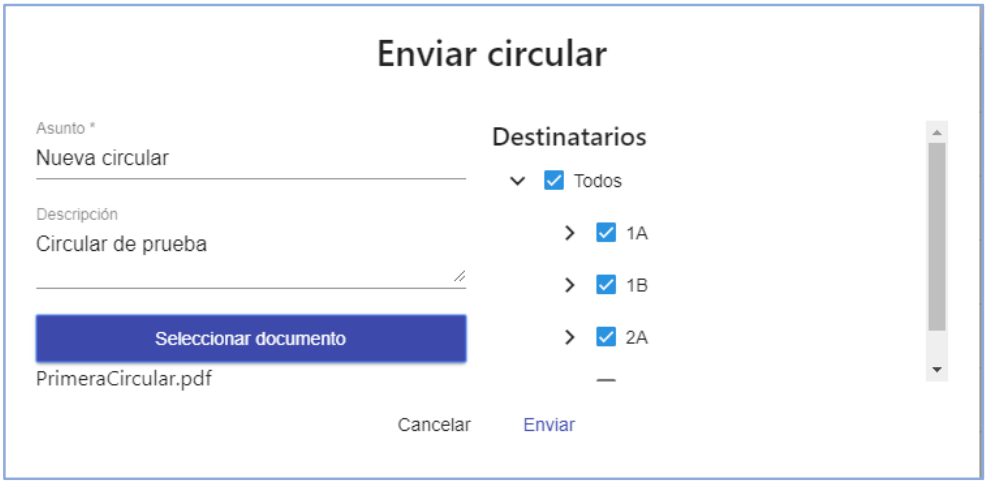


Ilustración 56: Diálogo enviar circular

Para buscar por asunto de la circular bastaría con hacer *click* encima del *input* definido con el nombre “Buscar por asunto” y escribir la cadena de texto por la que se quiere buscar cómo se puede observar en la *Ilustración 57*.

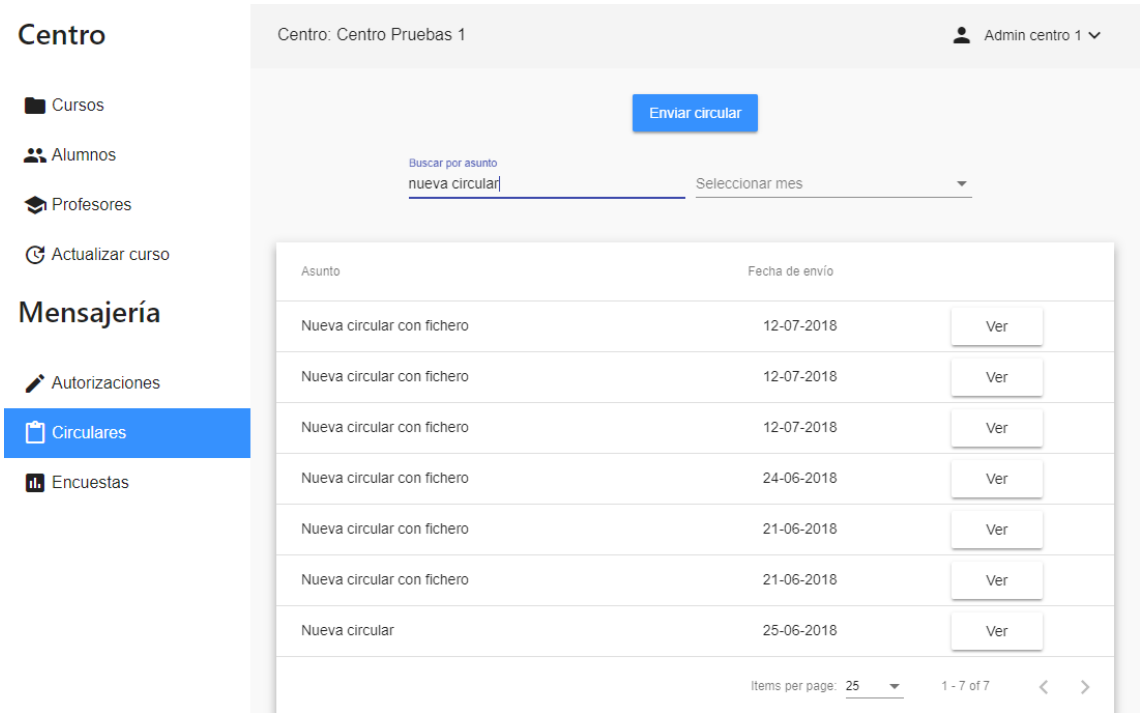


Ilustración 57: Buscar por asunto

Para hacer uso del filtro por mes de envío bastaría con hacer *click* en el desplegable definido con el nombre “Seleccionar mes” y seleccionar una de las opciones como se puede observar en la *Ilustración 58*.

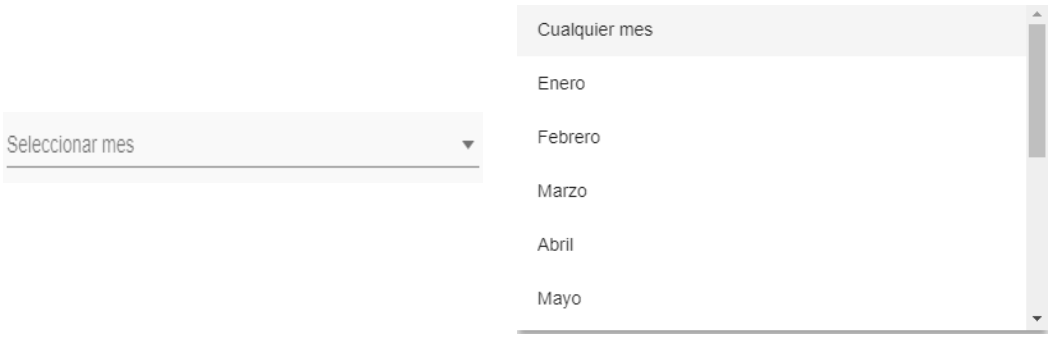


Ilustración 58: Filtrar por mes de envío

Haciendo *click* en el botón “Ver” asociado a cada circular se abrirá un diálogo mostrando la siguiente información (ver *Ilustración 59*):

- Asunto de la circular.
- Fecha de envío.
- Archivo adjunto con posibilidad de descargarlo.
- Mensaje de la circular.

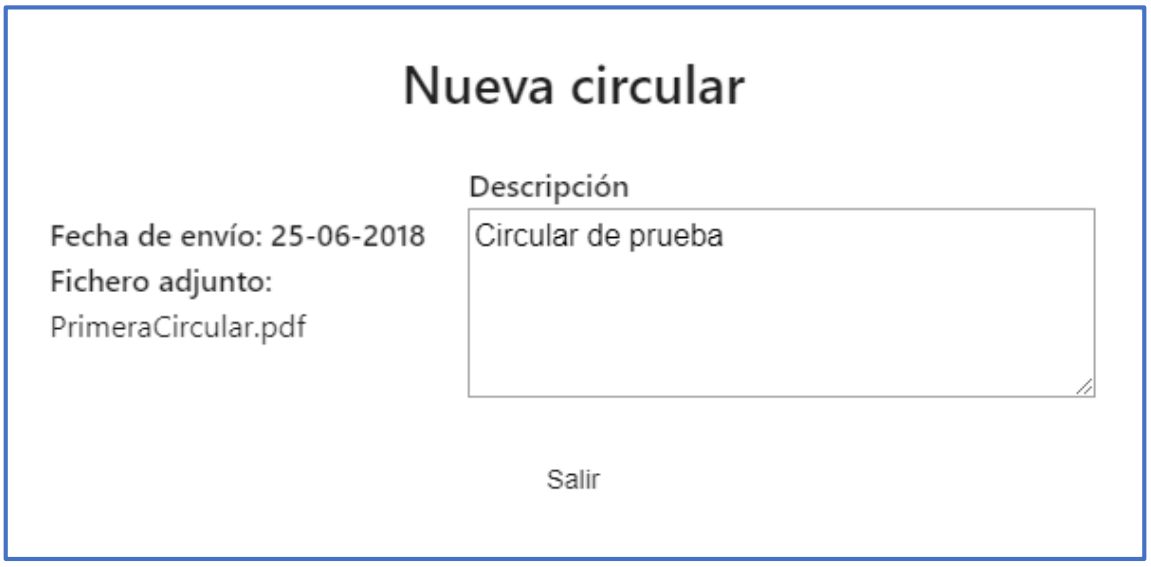


Ilustración 59: Diálogo ver circular

Encuestas

Esta sección proporciona al centro el envió de encuestas a los padres de los alumnos como se puede observar en la *Ilustración 60*.

Aquí se pueden realizar las siguientes acciones:

- Enviar una encuesta a los padres de los alumnos del centro.
- Filtrar las encuestas enviadas.

- Ver una encuesta.
- Editar la fecha límite para responder la encuesta.

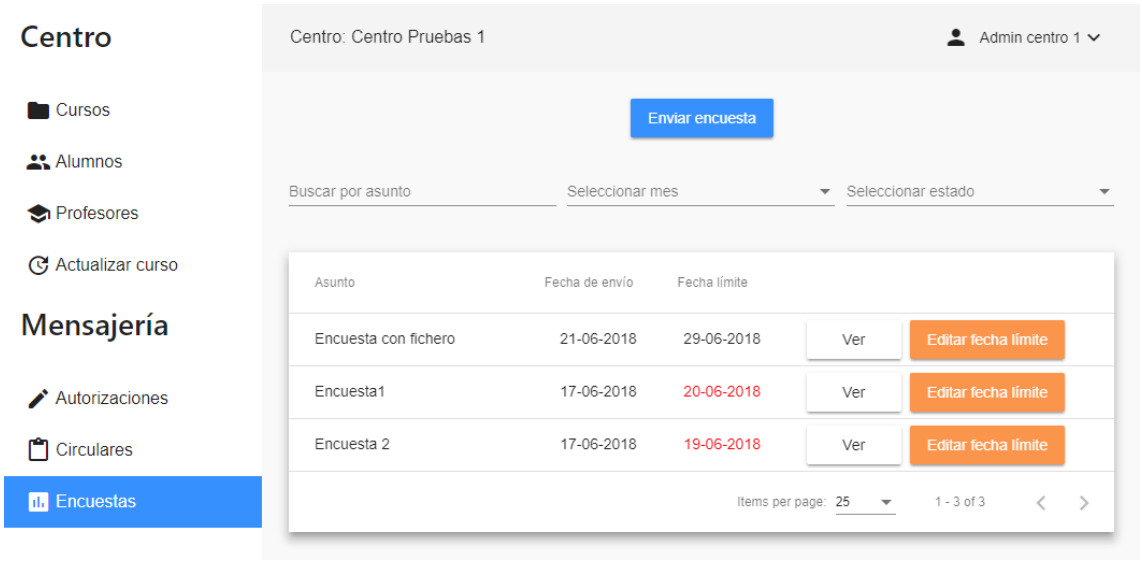


Ilustración 60: Página de encuestas

Haciendo *click* en el botón “Enviar encuesta” se abrirá un diálogo como se puede observar en las *Ilustraciones 61, 62 y 63* en el que habrá un formulario compuesto por tres pasos donde en cada paso se pueden rellenar con los siguientes campos:

- Paso 1
  - Asunto de la encuesta (en este caso este campo es equivalente a la pregunta de la encuesta).
  - Fecha límite para ser respondida.
  - Un campo para añadir un mensaje si fuese necesario.
  - Un botón con la posibilidad de adjuntar un fichero.
- Paso 2
  - Un apartado para añadir las opciones de la encuesta.
  - Permitir múltiple elección de opciones.
- Paso 3
  - Un árbol de *checkboxs* para seleccionar el alumno o los alumnos que quieren ser notificados con el mensaje.

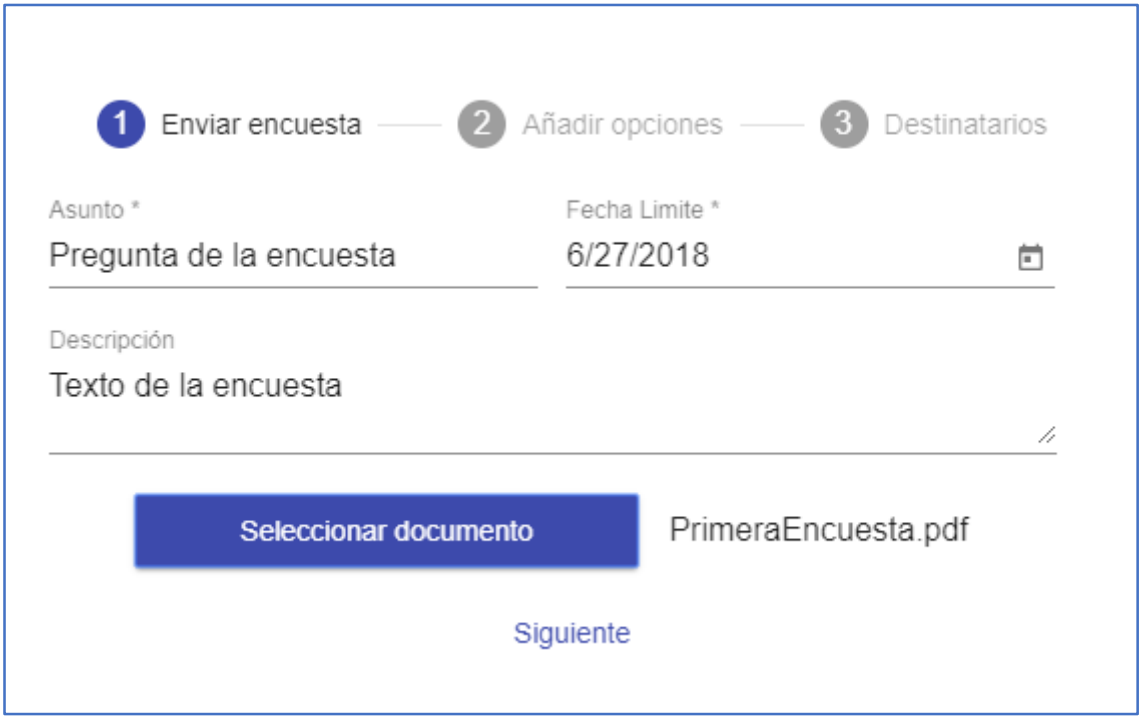


Ilustración 61: Diálogo enviar encuesta (Paso1)

Ilustración 62: Diálogo enviar encuesta (Paso2)

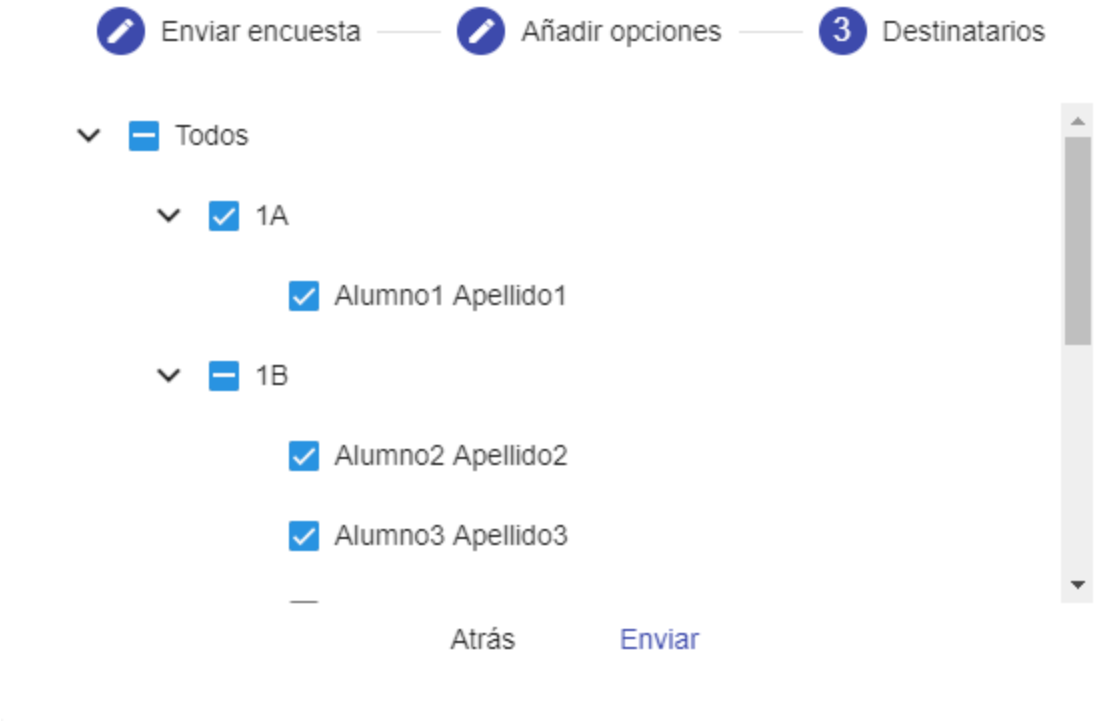


Ilustración 63: Diálogo enviar encuesta (Paso3)

Para buscar por asunto de la encuesta bastaría con hacer *click* encima del *input* definido con el nombre “Buscar por asunto” y escribir la cadena de texto por la que se quiere buscar cómo se puede observar en la *Ilustración 64*.

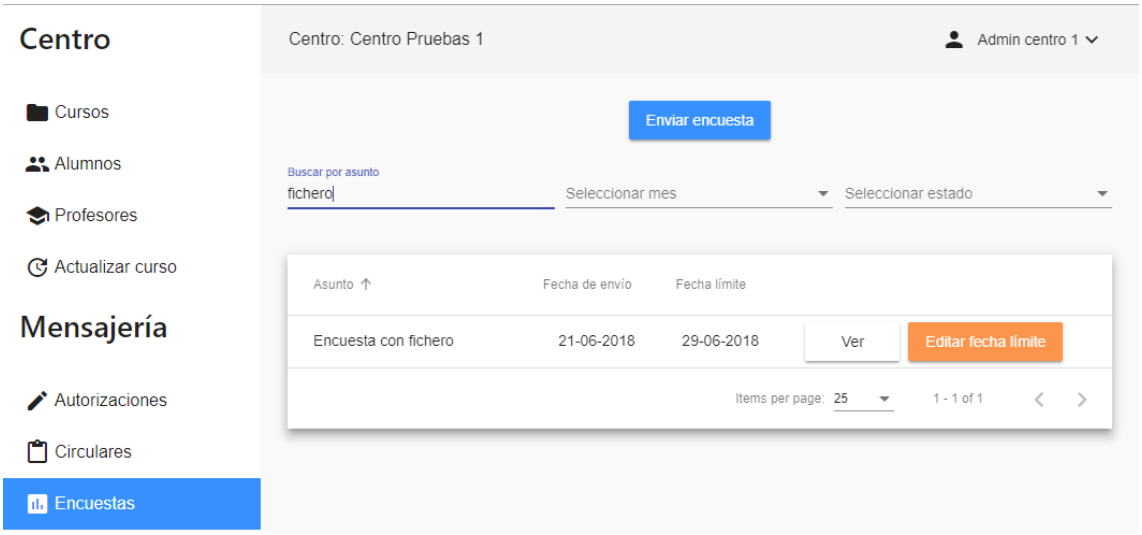


Ilustración 64: Buscar por asunto

Para hacer uso del filtro por mes de envío bastaría con hacer *click* en el desplegable definido con el nombre "Seleccionar mes" y seleccionar una de las opciones como se puede observar en la *Ilustración 65*.

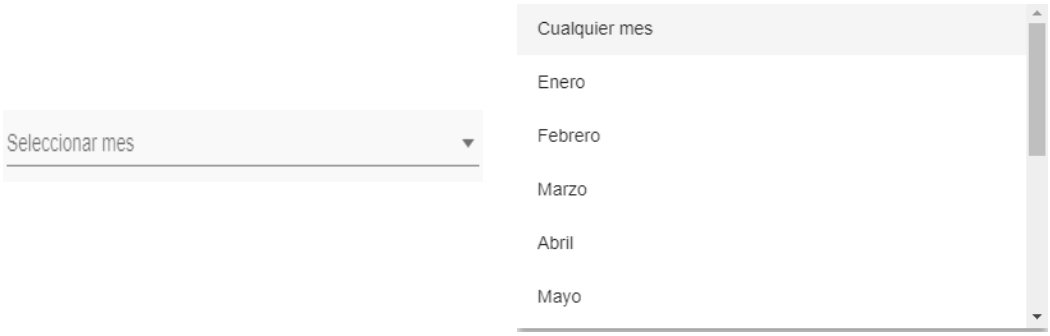


Ilustración 65: Filtrar por mes de envío

Para hacer uso del filtro estado bastaría con hacer *click* en el desplegable definido con el nombre "Seleccionar estado" y seleccionar una de las opciones como se puede observar en la *Ilustración 66*.

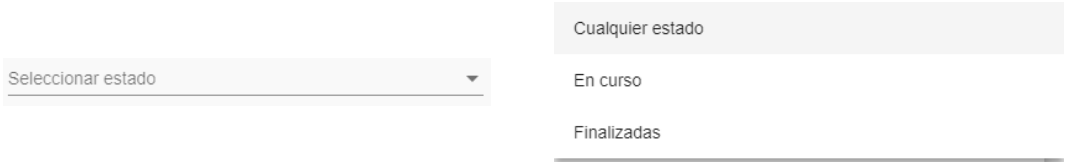


Ilustración 66: Filtrar por estado

Haciendo *click* en el botón “Ver” asociado a cada encuesta se abrirá un diálogo mostrando la siguiente información (ver *Ilustración 67*):

- Asunto de la encuesta (en este caso es la pregunta de la encuesta).
- Fecha de envío.
- Fecha límite para ser respondida.
- Archivo adjunto con posibilidad de descargarlo.
- Mensaje de la encuesta.
- Resultados de las opciones de la encuesta.

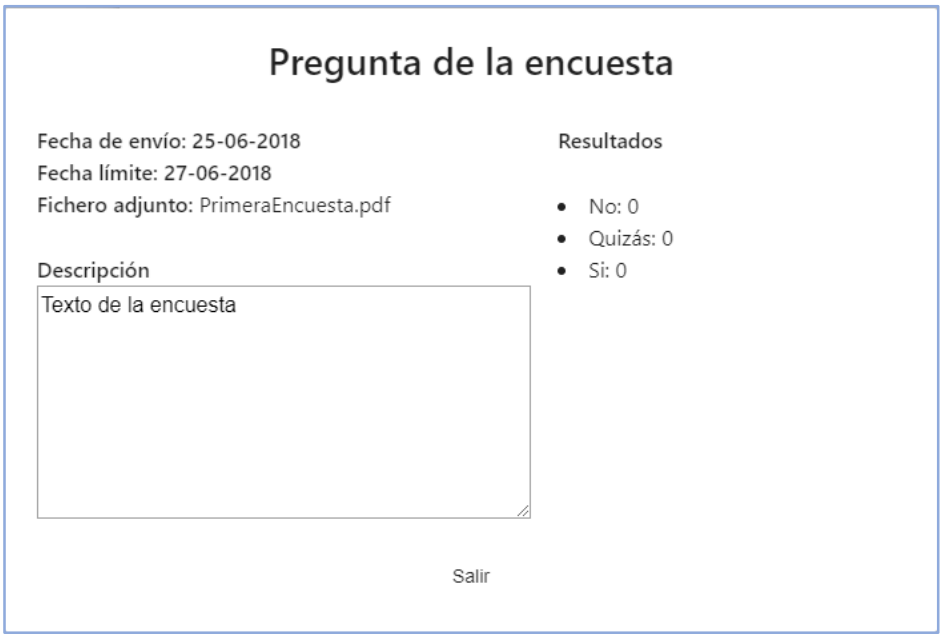


Ilustración 67: Diálogo ver encuesta

Haciendo *click* en el botón “Editar fecha límite” se abrirá un modal para cambiar la fecha límite por una posterior como se puede observar en la *Ilustración 68*. Cuando se abre el diálogo, por defecto sale la última fecha limite definida, si se quiere editar esta fecha bastaría con hacer *click* en el icono del calendario y seleccionar una nueva fecha límite.

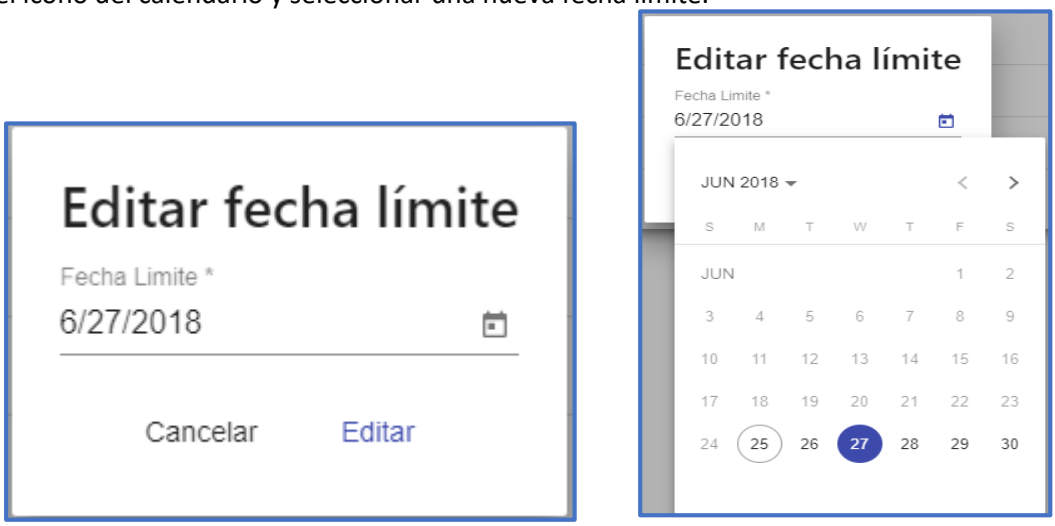


Ilustración 68: Diálogo editar fecha límite de encuesta

Anexo II: Manual de instalación

Este manual de instalación ha sido creado para ayudar al usuario en la instalación de la infraestructura y así poder probar y seguir desarrollando más funcionalidades de este proyecto.

A continuación, se definirán los requisitos previos para montar todo el sistema:

- Servidor Apache 2.0 o superior.
- MySQL 5.0 o superior.
- Intérprete PHP 7.1 o superior.
- Node 8.9.2 o superior.
- Npm 5.5.1 o superior.
- Herramienta Git 2.11 o superior.

En este proyecto se ha utilizado XAMPP. Esta es una herramienta multiplataforma (Windows, Linux, MacOS) que permite desplegar aplicaciones web desarrolladas en PHP. Se ha escogido esta herramienta porque tiene los tres primeros requisitos mencionados anteriormente que son los necesarios para ejecutar el *backend* desarrollado en Symfony.

Por otra parte, se ha tenido que instalar Node.js y *npm* para poder ejecutar el *frontend* dado que está desarrollado en Angular y este necesita de estos dos requisitos previos.

Ahora una vez instaladas las herramientas anteriormente se procederá a desplegar el *backend* y luego el *frontend*.

Para desplegar el *backend* primero hay que dirigirse a la carpeta C:\xampp\htdocs, una vez ahí se abre un terminal de la herramienta Git y se clona el proyecto denominado "hermerest_backend" del link https://github.com/danix7501/hermerest_backend.git.

Llegados a este punto se arranca el servidor Apache y MySQL desde XAMPP. Una vez hecho esto se accede a la URL http://localhost/phpmyadmin/ , desde aquí se creará la base de datos que va a ser usada por el *backend*. En este caso la base de datos ha sido llamada "hermerest".

Ahora ya creada la base de datos es la hora de ejecutar un intérprete de comandos php o abrir un IDE como puede ser PHPStorm. Esto es necesario para ejecutar la orden "composer install", esta orden instalará las dependencias necesarias para que el proyecto, además de mostrar una serie de pasos para configurar la base de datos. En uno de estos pasos se le pedirá al usuario el nombre de la base de datos que tiene que coincidir con el nombre de la base de datos creada anteriormente desde phpMyAdmin.

Siguiendo estos pasos, el proyecto de Symfony ya tendrá una base de datos vinculada. El único paso que falta es crear automáticamente las tablas de la base de datos, para ello, hay que abrir otra vez el intérprete de órdenes en el directorio raíz del proyecto o desde el IDE PHPStorm y ejecutar la orden "php bin/console doctrine:schema:update –force", esta orden se encargará crear las tablas de la base de datos y las relaciones que tienen entre ellas si es que las hubiese.

Por último, se desplegará el *frontend*. Para ello hay que dirigirse al link https://github.com/danix7501/hermerest_frontend.git y clonar el proyecto en cualquier directorio. Una vez clonado el proyecto bastará con abrirlo en un IDE como WebStorm y abrir el terminal o ejecutar directamente un terminal en el directorio raíz del proyecto. Ahí se tendrá

que ejecutar la orden “npm install” para instalar todas las dependencias del proyecto, este paso es de vital importancia para poder arrancar el proyecto.

Para finalizar, ejecutamos un terminal en el directorio raíz del proyecto o desde el IDE WebStorm. Aquí hay que ejecutar la orden “ng serve” y el proyecto empezara a compilar. Una vez compilado habrá que acceder a la URL http://localhost:4200 y ya se podrá hacer uso de proyecto.

En el proyecto se ha creado un servicio llamado http.service.ts, el mismo ha sido creado para realizar peticiones al *backend* y poder obtener los recursos que este ofrece. Ahora mismo está predefinido para hacer peticiones al proyecto situado en la URL http://localhost/hermerest_backend/web si algún día se quisiera cambiar la ubicación del *backend*, bastaría con definir la nueva ruta del *backend* en el servicio del *frontend*.