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

Alumno

Adrián Louro Alonso

Grado en Ingeniería Informática (Ingeniería del Software)

Tutor

Dr. Alexis Quesada Arencibia

Ciencias de la Computación e Inteligencia Artificial

Julio 2017 – Las Palmas de Gran Canaria

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

Adrián Louro Alonso

Título del proyecto

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

Tutor

Dr. Alexis Quesada Arencibia

Agradecimientos

A mi tutor Dr. Alexis Quesada por haber sacado tiempo de donde no tenía para ayudarnos.

A mi compañero Joel por haberme acompañado en este proyecto.

A mi familia y amigos por haberme ayudado a llegar hasta aquí.

Resumen

Hoy en día, muchos colegios se comunican con los padres de sus alumnos mediante el sistema tradicional, es decir, entregando los comunicados impresos a los niños para que se los entreguen a los padres.

Muchas veces, esto es un problema, porque es bastante frecuente que el niño se olvide de entregar el comunicado a sus padres o que, por alguna razón, no le interese entregarlo.

Además de esto, este sistema genera una serie gastos importantes debido a la impresión de los comunicados.

Por ello, se propone un sistema que permita la comunicación con los padres de los alumnos de manera económica, rápida, sencilla y directa. Además, el sistema permitirá la firma de autorizaciones por parte de los padres.

Abstract

Nowadays, many schools communicate with the parents of their students via traditional system, that is, delivering paper-based notices to the children so that they deliver them to their parents.

Often, this is a problem because it is quite common for the child to forget to deliver the notice to his parents or, for some reason, he is not interested in delivering it.

In addition to this, this system generates a significant expense due to the printing of the notices.

Because of this, we propose a system that allows to communicate to students' parents in an economic, fast, simple and direct way. Also, the system will allow the parents to sign the authorizations.

Índice

Índice de ilustraciones

Índice de tablas

Introducción

Para mejorar el sistema de comunicación tradicional entre centros educativos y padres de alumnos, en el que un miembro de secretaría entrega los comunicados impresos al profesor, para que éste los entregue a los alumnos y, por último, lleguen a los padres, hemos decidido, implementar un sistema que agilice todo el proceso, logrando que sea más rápido, sencillo y económico, puesto que no será necesario malgastar recursos en la impresión de los documentos.

Este sistema consiste en una aplicación web que será utilizada por uno o varios administradores designados por el centro. Por otro lado, los padres de los alumnos contarán con una aplicación móvil multiplataforma.

Este TFG tiene como objetivo desarrollar la aplicación web para el centro, desde la que los centros educativos podrán enviar mensajes (autorizaciones de actividades, circulares informativas y encuestas) a los padres de sus alumnos y, además, podrán visualizar las respuestas recibidas por parte de los padres.

La aplicación móvil utilizada por los padres será desarrollada por Joel David Delgado Perdomo 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"*, en la cual los padres podrán visualizar los comunicados enviados por el centro, además de poder firmar autorizaciones para que sus hijos realicen alguna actividad y responder encuestas de interés general acerca de algún tema del centro.

Para que la aplicación móvil pueda comunicarse con la base de datos, hemos decidido desarrollar un servicio web RESTful. Debido a que el *backend* del sistema forma parte de este TFG, se desarrollará una capa de acceso a los datos que permitirá que el acceso a la base de datos sea completamente transparente desde los métodos del servicio web, desarrollados en el TFG encargado de la aplicación móvil.

Estructura del documento

El documento se encuentra estructurado en diferentes capítulos. A continuación, explicaremos los distintos capítulos que hemos desarrollado:

- Capítulo 1: Estado actual y objetivos iniciales

  En este capítulo comenzamos explicando el estado actual del problema, presentando las alternativas existentes y explicando los objetivos que pretendemos alcanzar con el desarrollo del proyecto.

- Capítulo 2: Competencias específicas cubiertas

  En este capítulo enumeramos y justificamos las distintas competencias cubiertas por el proyecto.

- Capítulo 3: Aportaciones

  En este capítulo describimos las distintas aportaciones del proyecto en el entorno socio-económico y a nivel personal.

- Capítulo 4: Normativa y legislación

  En este capítulo presentamos un breve estudio y resumen acerca de la normativa y la legislación vigente relevante sobre este proyecto. También, comentamos las licencias software utilizadas para realizar el proyecto.

- Capítulo 5: Metodología de trabajo y planificación del proyecto

  En este capítulo explicamos y justificamos la metodología de trabajo seguida, así como la planificación inicial del proyecto y los diferentes ajustes realizados sobre ella.

- Capítulo 6: Tecnologías y herramientas utilizadas

  En este capítulo hacemos una breve descripción de las distintas tecnologías y herramientas que han sido utilizadas a lo largo del desarrollo del proyecto.

- Capítulo 7: Análisis

  En este capítulo explicamos la fase de análisis del proyecto. Hablamos sobre los distintos actores que hacen uso de la aplicación y explicamos los requisitos del sistema que hemos identificado, mediante diagramas de casos de uso y la especificación de los casos de uso más relevantes.

- Capítulo 8: Diseño

  En este capítulo explicamos la fase de diseño del proyecto. Explicamos y justificamos la arquitectura del sistema, hablamos sobre el diseño de la base de datos y de la aplicación web y, por último, explicamos los distintos aspectos de diseño que hemos tenido en cuenta a la hora de realizar la interfaz de usuario de la aplicación.

- Capítulo 9: Desarrollo

  En este capítulo explicamos la fase de desarrollo del proyecto. Explicamos y justificamos la elección del *framework* de programación utilizado, el sistema de control de acceso de

la aplicación, las capas de datos y de acceso a datos de la aplicación, la capa de presentación y el sistema de rutas de los controladores.

- Capítulo 10: Resultados, conclusiones y trabajo futuro

  En este capítulo comentamos las distintas pruebas realizadas a lo largo del proyecto para comprobar que el sistema funcionaría correctamente en un entorno real y que las aplicaciones se adaptan correctamente sus usuarios.

- Capítulo 11: Resultados, conclusiones y trabajo futuro

  En este capítulo comentamos los resultados y conclusiones que hemos obtenido tras realizar el proyecto y comentamos posibles mejoras y ampliaciones para el mismo.

- Capítulo 12: Bibliografía

  En este capítulo presentamos la bibliografía sobre la que nos hemos apoyado a lo largo del documento, así como la utilizada a lo largo del desarrollo de la aplicación.

- Anexos

  En este apartado incluiremos información complementaria del proyecto realizado.

  - Anexo I: explicamos de manera clara y sencilla cómo se utiliza la aplicación web.
  - Anexo II: explicamos cómo instalar y desplegar el sistema en un servidor.
  - Anexo III: contiene el script escrito en lenguaje SQL a utilizar para generar la base de datos que utilizan las aplicaciones.

1. Capítulo 1: Estado actual y objetivos iniciales

1.1. Estado actual

A día de hoy, muchos centros educativos siguen utilizando el sistema tradicional para comunicarse con los padres de sus alumnos, el cual consiste en entregar los comunicados impresos a los tutores de cada curso, para que se los entreguen a sus alumnos y que, por último, éstos se los entreguen a sus padres.

Este sistema presenta grandes deficiencias debido a que:

- Genera gastos innecesarios debido a la impresión de los documentos.
- El comunicado puede tardar en llegar al padre debido a que pasan por distintos intermediarios hasta llegar a ellos.
- Es posible que el comunicado no llegue a los padres, puesto que es bastante común que los alumnos no los entreguen a sus padres (ya sea porque se le olvida o porque no le interesa que sus padres lo reciban).
- El control de las autorizaciones es complejo, ya que las autorizaciones firmadas se gestionan manualmente.

Por otro lado, algunos centros ya utilizan aplicaciones informáticas para realizar la gestión del centro y comunicarse con los padres de sus alumnos.

1.1.1. Alternativas existentes

Para ofrecer un producto que tenga éxito y se adapte a las necesidades de sus usuarios, es muy importante estudiar el mercado actual, es decir, qué productos existen para solventar el problema que intentamos resolver.

Actualmente, existen algunas aplicaciones en el mercado que intentan solventar este problema, pero cada una cuenta con características y funcionalidades diferentes. A continuación, mostraremos una comparación de las distintas alternativas existentes en el mercado:

Nombre	Características	Problemas
Educamos [5]	**Colegios** -Concertar, aceptar y rechazar entrevistas y tutorías. -Envío de circulares. -Envío de autorizaciones. **Padres y alumnos** -Ver tareas, exámenes y calificaciones. -Ver incidencias. **Padres** -Justificar incidencias y justificarlas. -Confirmar o denegar autorizaciones. -Seguir la ruta del autobús. -Ver recibos. -Inscribir al hijo/a a las actividades y servicios del colegio.	No permite autoimportar datos. No permite la realización de encuestas.
miColegioApp [6]	**Colegios** -Enviar circulares y adjuntarles archivos.	No permite autoimportar datos.

	Padres -Confirmar citas, firmar autorizaciones.	No permite realizar encuestas.
Escolapp [7]	**Colegios** -Enviar avisos a los padres. -Enviar deberes. -Enviar autorizaciones de excursión. **Padres** -Consultar la actividad del centro. -Comprar uniformes y libros. -Ver calendario escolar. -Ver deberes. -Contratar actividades extraescolares. -Autorizar excursiones.	No permite autoimportar datos. No permite realizar encuestas.
Dinantia [8][9]	**Colegios** -Hacer preguntas a padres y/o alumnos. -Solicitar autorizaciones. -Envío de comunicados. -Pasar lista y notificar las ausencias automáticamente a los padres. -Comunicación interna. **Padres** -Ver calendario de acontecimientos. -Ver el menú escolar. -Seguimiento del aula.	Requiere infraestructura informática en las aulas para aprovechar todo su potencial. No permite autoimportar datos.
TokAppSchool [10][11][12]	**Colegios** -Envíos de circulares. -Solicitud de respuesta en los mensajes. -Los datos del colegio se importan de manera automática. **Padres** -Chat con el centro. -Chat con profesores.	No permite autorizar actividades. Requiere personal capaz de atender a los chats.

Tabla 1: Comparativa de las aplicaciones existentes

1.1.2. Conclusiones

Entre todas las aplicaciones analizadas, todas las funcionalidades propuestas por nuestro sistema están cubiertas. Sin embargo, individualmente, ninguna de ellas las cubre en su totalidad.

La mayoría de aplicaciones abarcan funcionalidades asociadas a la gestión del centro (seguimiento de alumnos, deberes, organización del centro, menús de comedor, transporte escolar, etc.).

Una de las funcionalidades que consideramos que es bastante importante es la importación automática de los datos del centro y, solamente, una de las aplicaciones existentes permite hacer esta gestión.

Por ello, podemos decir que nuestro sistema contiene algunas de las características más interesantes relacionadas, únicamente, con la mensajería.

Al no contar con funcionalidades relacionadas con la gestión del centro, su uso se vuelve muy sencillo, característica que es muy importante si sus usuarios no tienen conocimiento en el uso de aplicaciones informáticas, algo que es bastante frecuente en padres y trabajadores de centros educativos.

Además, nuestro sistema puede ser utilizado en centros con una infraestructura informática mínima, ya que solo necesitan contar un equipo desde el cual enviar los mensajes y ver las respuestas de los padres.

A nivel de personal, es suficiente con tener a un único encargado de la aplicación (pudiendo tener más si fuera necesario).

1.2. Objetivos

El objetivo de este TFG se resume en desarrollar un sistema que permita que el centro se comunique directamente con los padres de los alumnos sin malgastar papel y sin tener que utilizar a profesores y alumnos como intermediarios.

Se podrán enviar circulares informativas, encuestas y autorizaciones de actividades extraescolares. Además, si así lo desea el centro, la importación de datos puede realizarse de forma automática.

Este TFG consistirá en el análisis y diseño del sistema anterior y en el desarrollo de la aplicación web para el centro, la cual será utilizada por uno o varios administradores designados por el centro.

Desde la aplicación web, los administradores del centro llevarán la gestión del centro necesaria para poder enviar los mensajes de manera ordenada a los padres de los alumnos (gestión de cursos, alumnos y padres), además de poder enviar los mensajes y ver las respuestas de los padres.

Hemos intentado que la cantidad de información que se tenga que manejar sea la menor posible. De esta manera, procuramos que el uso de las aplicaciones sea lo más rápido y sencillo posible para sus usuarios. Por ejemplo, eliminamos información innecesaria sobre los alumnos, como pueden ser su dirección o su fecha de nacimiento, ya que estos datos no tienen una utilidad relevante en el sistema de mensajería.

2. Capítulo 2: Competencias específicas cubiertas

2.1. Comunes a la Ingeniería Informática

2.1.1. 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."

A la hora de desarrollar las distintas aplicaciones hemos tenido en cuenta aspectos de seguridad como el control de acceso a la información (no puede ser accedida por alguien que no debe tener acceso a ella).

Las herramientas utilizadas para el desarrollo del sistema fueron estudiadas previamente para asegurarnos de que cumplían con los requisitos de calidad, seguridad y fiabilidad necesarios.

2.1.2. 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 ha sido llevado a cabo por dos alumnos, por lo que la comunicación ha sido un factor clave para estudiar el alcance del proyecto y que las distintas aplicaciones que han sido desarrolladas se integren correctamente.

En las fases de análisis y diseño la negociación y la comunicación fueron llevadas a cabo con mucho cuidado para evitar, en la medida de lo posible, problemas durante el desarrollo del proyecto.

2.1.3. 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."

A lo largo del proyecto, abarcamos las fases de análisis, diseño y desarrollo del sistema. También, hemos tenido que estudiar las distintas tecnologías disponibles y escoger la que nos pareció más adecuada.

Por ello, hemos decidido utilizar el *framework* web Symfony, una tecnología madura y respaldada por una gran comunidad.

2.1.4. 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."

Hemos empleado el tiempo necesario para que el diseño de la base de datos se ajustase lo mejor posible a las necesidades del proyecto. De esta forma, hemos podido utilizar una serie de librerías que nos han permitido tratar los datos como objetos, lo cual nos ha permitido aprovechar al máximo las ventajas de la programación orientada a objetos.

2.1.5. 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 sistema desarrollado cuenta con distintas aplicaciones desarrolladas en distintas tecnologías que funcionan sobre distintas plataformas, por lo que hemos tenido que desarrollar una *API* que permite que dichas aplicaciones puedan comunicarse correctamente.

2.1.6. CII016

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

Hemos utilizado una metodología de desarrollo basada en prototipos, mediante la cual hemos ido desarrollando el proyecto en distintas iteraciones. A lo largo del documento haremos referencia a las distintas fases del ciclo de vida del software, las cuales han sido bien diferenciadas a lo largo del proyecto.

2.1.7. CII017

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

Hemos hecho gran hincapié en la usabilidad de las aplicaciones, puesto que es un factor clave para que los usuarios de las mismas puedan utilizarlas de manera rápida y sencilla. Por ello, hemos puesto en práctica algunos principios de diseño que hemos aprendido durante la carrera.

2.1.8. CII018

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

Se ha realizado un estudio acerca de las normativas más importantes y su impacto sobre este proyecto, así como de las leyes más determinantes sobre él. Por ello, hemos dedicado un capítulo en esta memoria donde explicamos este tema.

2.2. Trabajo Fin de Grado

2.2.1. TFG01

"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 desde cero, a partir de una idea, y nos ha permitido aplicar gran parte de los conocimientos y competencias adquiridas durante la carrera. Una vez concluya su evaluación ante el tribunal correspondiente, esta competencia se dará por satisfecha.

2.3. Ingeniería del Software (IS)

2.3.1. 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 el proyecto, hemos realizado una planificación según una metodología de desarrollo de software, la cual nos ha permitido organizarnos de tal forma que hemos podido cumplir los objetivos marcados, entregando software de calidad que responde a las necesidades

de sus usuarios. Hemos aplicado distintos principios de diseño, patrones de diseño y patrones arquitectónicos para garantizar un sistema robusto y flexible.

2.3.2. IS02

"Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones."

Durante las primeras semanas, dedicamos gran parte de nuestro tiempo en el estudio de las necesidades que tienen nuestros usuarios. A lo largo del proyecto, hemos ido modificando las funcionalidades del sistema para ajustarlas lo máximo posible a las necesidades de los usuarios.

Para especificar los requisitos software hemos realizado diagramas de casos de uso y tablas de especificaciones de casos de uso.

2.3.3. IS03

"Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles."

En este proyecto, en el cual intervienen distintas aplicaciones desarrolladas por distintos alumnos, hemos tenido que utilizar herramientas como GIT (para el control de versiones), para poder trabajar sobre el *backend* del sistema de manera organizada.

De esta forma, hemos conseguido que las distintas aplicaciones compartan un *backend* común.

2.3.4. IS04

"Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales."

Hemos analizado, diseñado e implementado un sistema software poniendo en práctica distintos principios y modelos actuales (especificación de requisitos mediante UML, diseño de la arquitectura del sistema, uso de patrones de diseño) que hemos ido aprendiendo a lo largo de la carrera, de tal forma que siempre buscamos desarrollar software de calidad.

2.4. Tecnologías de la Información

2.4.1. TI06

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

El proyecto consiste en una aplicación web que interacciona con una aplicación móvil desarrollada por otro alumno, la cual se comunica con el resto del sistema mediante un servicio web RESTful que se apoya en el *backend* realizado para la aplicación web.

3. Capítulo 3: Aportaciones

3.1. Entorno socio-económico

El desarrollo de este proyecto ofrecerá tanto a padres como a centros educativos un sistema de comunicación rápido, sencillo y que evitará el malgasto de papel que conlleva el sistema tradicional.

A día de hoy, el malgasto de papel es un tema que está a la orden del día, puesto que su uso deriva en la tala de árboles, la cual tiene una serie de consecuencias negativas como:

- Reducción del medio para convertir el dióxido de carbono en oxígeno, lo que aumenta el efecto invernadero y genera enfermedades respiratorias en la población.
- Cambios en el clima debido a que la falta de árboles reduce la retención de la humedad, lo cual aumenta la posibilidad de sequías.
- Destrucción de los ecosistemas y su consecuente pérdida de la biodiversidad.

Además del impacto positivo en el medio ambiente y de la reducción de gastos que supone para los centros educativos, este sistema eliminará a profesores y alumnos como intermediarios para que los padres reciban comunicados por parte del centro. Esto tendrá una serie de consecuencias positivas en el proceso de comunicación:

- Se vuelve mucho más rápido, debido a que los comunicados llegarán a los padres de manera directa e inmediata.
- Se vuelve mucho más fiable, puesto que ya no existirá la posibilidad de que el profesor no entregue el documento al alumno (el alumno podría no encontrarse en el aula) o de que el alumno no entregue el documento a sus padres (ya sea porque se le olvida o porque no le interesa que sus padres lo reciban).
- El centro podrá controlar fácilmente los listados de alumnos autorizados para realizar una actividad.
- Los padres podrán participar en decisiones importantes del centro de manera rápida y sencilla.

En resumen, podemos decir que el proyecto tiene consecuencias positivas tanto para el medio ambiente (ahorro de papel) como para los centros educativos (ahorro en recursos) y las familias con hijos en edad de escolarización.

3.2. Personal

A nivel personal, este proyecto ha resultado una experiencia muy enriquecedora. Fue llevado a cabo desde cero, por lo que abarcamos las fases del ciclo de vida del software desde el análisis hasta el desarrollo.

Gracias a esto, hemos podido aplicar y reforzar muchos conocimientos adquiridos en la carrera y en la intensificación de Ingeniería del Software. Esto último tiene un gran valor, ya que las fases de análisis y diseño son muy importantes para que cualquier proyecto software tenga éxito y, muchas veces, no se hace demasiado hincapié en ellas.

A nivel técnico, la implementación de la aplicación web me ha servido para aprender tecnologías interesantes, como son Symfony 3, Doctrine y PHP 7.1, para el *backend* de la aplicación y Twig, JQuery, JavaScript y Ajax, para el *frontend* de la misma.

También, el haber tenido que diseñar la arquitectura completa del sistema, el cual cuenta con distintas aplicaciones desarrolladas en distintas tecnologías que interaccionan entre ellas, ha sido una gran oportunidad para poner en práctica los conocimientos adquiridos en arquitectura de sistemas.

Por último, el haber realizado el proyecto conjuntamente con otro alumno, nos ha permitido realizar un análisis y un diseño de mayor calidad. Aunque la implementación de las distintas aplicaciones fue llevada a cabo individualmente, hemos tenido la oportunidad de aprender el uno del otro a la hora de validar conjuntamente el trabajo realizado.

4. Capítulo 4: Normativa y legislación

4.1. Licencias de Software

Una licencia de software establece un contrato entre licenciante, el autor o titular de los derechos de distribución y explotación, y licenciatario, que especifica los términos y condiciones a seguir para el uso del software. Existen distintos tipos de licencia de software, las cuales varían según si el código es abierto, cerrado, de dominio público o según su destinatario.

A continuación, presentaremos las licencias utilizadas para la realización del proyecto:

4.1.1 GPL

La Licencia Pública General de GNU [13] es la licencia de derecho de autor más ampliamente usada en el mundo del software libre y código abierto y 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.

XAMPP utiliza esta licencia. Sin embargo, cada uno de los componentes incluidos tiene su propia licencia y deberán ser consultadas para conocer qué es posible y que no. En el caso de uso comercial se debe consultar las licencias individuales, en particular MySQL.

StarUML [15] utiliza esta licencia, siguiendo dos excepciones:

- Permitir vincular librerías y componentes comerciales específicos.
- Permitir vincular módulos patentados.

El código generado por StarUML no se encuentra sujeto bajo esta licencia, por lo que podemos utilizar el código general por StarUML para realizar código comercial no sujeto a GPL.

MySQL se encuentra desarrollado bajo una licencia dual GPL/Licencia comercial de Oracle Corporation, por lo que los derechos de autor del código están en poder del autor individual. La licencia GNU GPL de MySQL obliga a que la distribución de cualquier producto derivado se haga bajo esa misma licencia.

Si un desarrollador desea incorporar MySQL en su producto, pero desea distribuirlo bajo otra licencia que no sea la GNU GPL, puede adquirir una licencia comercial de MySQL que le permite hacer justamente eso.

4.1.2. Licencia MIT

Esta licencia MIT [14] es una Licencia de software libre permisiva que 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 la GNU General Public License (software con licencia MIT puede integrarse en software con licencia GPL, pero no al contrario). El texto de la licencia no tiene *copyright*, lo que permite su modificación.

Esta licencia permite reutilizar el software así licenciado tanto para ser software libre como para ser software no libre, permitiendo no liberar los cambios realizados al programa original.

También permite licenciar dichos cambios con licencia BSD, GPL u otra cualquiera que sea compatible (es decir, que cumpla las cláusulas de distribución).

Con esta licencia se tiene software libre. Ejemplos en los que podría interesar su aplicación serían las licencias duales, si se pretende difundir un estándar mediante una implementación de referencia, o si simplemente se pretende que el producto sea libre sin mayores consideraciones.

Esta licencia es utilizada por Symfony, Doctrine y JQuery.

4.1.3 Apache License

Apache License [16] 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.

Esta licencia es utilizada por el servidor web Apache.

4.1.4. PHP License

PHP License [17] 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. Está diseñada para incentivar la distribución del código fuente. Se permite la redistribución del contenido licenciado en forma de código fuente o binaria siempre y cuando se cumplan los siguientes requisitos:

- Se incluya la declaración de los derechos de autor de la licencia PHP.
- La palabra "PHP" no se use en el título de las obras derivadas.
- Se incluya el siguiente anuncio bajo cualquier forma en la que se redistribuya el código: *"This product includes PHP software, freely available from http://www.php.net/software/"*.

4.1.5. Microsoft Office: Office 365 Education Plus for students.

Es un conjunto de servicios que te permite colaborar en las tareas escolares y compartirlas. Está disponible de forma gratuita para los profesores que trabajen en una institución académica y para los alumnos que estén inscritos actualmente en centros educativos, en nuestro caso, la universidad.

Este servicio permite que los profesores y los alumnos instalen las aplicaciones completas de Office en un máximo de 5 equipos PC o Mac de forma gratuita.

Esta licencia ha sido utilizada por la aplicación Microsoft Word 2016.

4.1.6. Justinmind

Justinmind [19] o sus proveedores son propietarios de los derechos de propiedad intelectual de todos y cada uno de los componentes protegibles del servicio, incluyendo, ilustraciones y elementos de interfaz de usuario final contenidos en el servicio, muchas de las características individuales y documentación relacionada.

No se permite copiar, modificar, adaptar, reproducir, distribuir, realizar ingeniería inversa, descompilar o disimular cualquier aspecto del servicio que pertenezca a Justinmind o a sus propios proveedores.

Justinmind no reclama derechos de propiedad intelectual sobre el contenido que se suba o proporcione al servicio. Sin embargo, al utilizar el servicio para enviar contenido, se acepta que otros puedan ver y compartir su contenido.

Esta licencia es utilizada por la aplicación Justinmind Prototyper.

4.1.7. PHPStorm: toolbox subscription license agreement for education

Hemos utilizado una licencia de estudiante, a la que tenemos acceso por ser estudiantes de la universidad.

Esta licencia [18] no puede ser vendida o traspasada a terceros y 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 cualquier y cualquier tecnología relacionada o subyacente, y cualquier modificación o trabajo derivado de lo anterior creado por o para JetBrains.

Esta licencia es utilizada por el IDE PHPStorm.

4.2. Seguridad de los datos

La Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal (LOPD) [20], es una Ley Orgánica española que tiene por objeto garantizar y protegerlas libertades públicas y los derechos fundamentales de personas físicas, como el honor, intimidad y privacidad tanto familiar como personal, en todo lo referente al tratamiento de los datos personales.

Según esta Ley, en esta aplicación se debe garantizar la protección de los datos personales que figuren en todos aquellos elementos relacionados, principalmente aquella información guardada en base de datos.

Esta ley afecta a todos los datos que hacen referencia a personas físicas registradas sobre cualquier soporte informático.

En esta aplicación recogemos datos personales de padres y alumnos como:

- Nombres y apellidos.
- Números de teléfono.

Toda esta información pertenece al “Nivel Básico” [21] establecido por la Agencia Española de Protección de Datos (AGPD), por lo que no puede ser accedida por cualquier persona. Por ello, esta información es accesible, únicamente, por los administradores de los centros educativos relacionados con las personas mencionadas anteriormente.

Estos datos se encuentran recogidos en una base de datos protegida bajo contraseña, la cual no podrá ser accedida por personas ajenas a la empresa.

El acceso a la información de los padres por parte de los centros está protegido, es decir, un centro podrá acceder a los datos de un padre, únicamente, si el padre ha dado permiso al centro para que pueda acceder a ellos.

En relación a los datos anteriores, estos se recogerán con fines determinados explícitos y legítimos y no se utilizarán para otros fines. Así mismo, se garantiza que los datos son adecuados, pertinentes y no excesivos en relación con esa finalidad, sin exigir datos de más.

La web se comprometerá a que los datos sólo se conservarán durante el tiempo necesario para las finalidades del tratamiento para las que han sido recogidos y a cancelarlos si ya no son necesarios.

5. Capítulo 5: Metodología de trabajo y planificación del proyecto

5.1. Metodología de trabajo

La metodología de trabajo que hemos utilizado se basa en el modelo de ciclo de vida del software basado en prototipos.

Un prototipo puede ser desde un boceto hasta un programa que funciona, el cual podemos utilizar para validar con los clientes las funcionalidades existentes a medida que se va desarrollando la aplicación. Además, son muy útiles para identificar requisitos que no han sido identificados y para eliminar los que no son realmente necesarios, es decir, los que no cubren una necesidad de los usuarios.

Esta metodología pertenece a los modelos de desarrollo evolutivo, es decir, permite desarrollar versiones de la aplicación que son cada vez más completas y complejas, hasta llegar al objetivo final deseado.

Ilustración 1: Metodología de desarrollo basada en prototipos [1]

Básicamente, consiste en construir prototipos de forma iterativa, enseñárselos a los clientes para validar las funcionalidades implementadas y obtener *feedback* con el fin de obtener información para la siguiente iteración, como puede ser modificar alguna funcionalidad ya implementada, nuevas funcionalidades que se deben añadir a la aplicación o funcionalidades que debemos eliminar.

Hemos decidido escoger esta metodología porque se acomoda a nuestras necesidades, permitiendo que podamos construir el producto de forma incremental, siendo validado por el tutor cada uno de los prototipos realizados.

5.2. Planificación inicial del proyecto

Tras haber hecho un estudio previo del estado del arte, la primera iteración consistió en identificar los distintos actores del sistema, es decir, aquellas personas que utilizarán las aplicaciones y, también, los requisitos funcionales del sistema, es decir, aquellas características que deberían presentar las distintas aplicaciones. Para ello, utilizamos diagramas de casos de uso y tablas de especificación de casos de uso.

Una vez identificados los distintos actores y los requisitos funcionales, utilizamos la herramienta para prototipado de interfaces de usuario Justinmind Prototyper, con la cual realizamos un prototipo de la interfaz de usuario que nos permitió tener una primera visualización de las aplicaciones.

Esta etapa nos ayudó en gran medida a contrastar y definir correctamente las funcionalidades que debería tener el sistema, además de cómo debería ser la interacción de las aplicaciones con sus usuarios.

Tras tener claras las características que deberían tener las aplicaciones, hicimos un estudio de la arquitectura del sistema, así como de las tecnologías existentes para escoger las que mejor se adaptasen a sus características.

Una vez hecho esto, comenzamos la fase de aprendizaje de las nuevas tecnologías, Symfony 3 y Doctrine. Esta fase consistió en el desarrollo de una pequeña aplicación, con el objetivo de obtener experiencia con el *framework* de programación Symfony 3. Para aprender a utilizar las tecnologías, fuimos completando una serie de tutoriales *online* que nos permitieron adquirir destreza con ellas.

Finalizada la fase de aprendizaje de las tecnologías, comenzamos con el desarrollo de las aplicaciones. De acuerdo a la metodología de desarrollo descrita anteriormente, el desarrollo del proyecto fue dividido en distintas iteraciones. Estas iteraciones fueron las siguientes:

- Iteración 1: esta iteración consistió en la configuración básica del proyecto y la realización del sistema de inicio de sesión de la aplicación web.

- Iteración 2: en esta iteración se implementaron las funcionalidades correspondientes a la gestión del centro, como la importación automática de los datos y la gestión de cursos y alumnado.

- Iteración 3: en esta iteración, se implementaron las funcionalidades correspondientes a la gestión de la mensajería, como el envío de autorizaciones, circulares y encuestas, así como la visualización de las respuestas a encuestas y autorizaciones proporcionadas por los padres de los alumnos.

- Iteración 4: tras finalizar la iteración anterior, tuvimos que llevar a cabo una refactorización importante de código para poder utilizar las buenas prácticas propuestas por las APIs REST pragmáticas. De esta forma, logramos organizar mejor las rutas del *backend*, así como eliminar grandes cantidades de código duplicado.

5.3. Ajustes de la planificación del proyecto

En un principio, el alcance del proyecto comenzó siendo mayor al desarrollado. Además de las funcionalidades nombradas anteriormente, comenzamos incluyendo un sistema de tutorías entre los profesores del centro y los padres de los alumnos.

El sistema de tutorías contaba con las siguientes características:

- Administradores del centro:
    - Registrar profesores en el centro.
    - Asociar profesores a un curso.
    - Asociar un tutor a un curso.

    - Establecer asignaturas en el centro.
    - Asociar asignaturas a un profesor.
- Profesores:
    - Establecer un horario de tutorías.
    - Solicitar tutorías a los padres.
    - Ver calendario de tutorías.
    - Confirmar tutorías.
    - Cancelar tutorías.
- Padres:
    - Solicitar tutorías a los profesores de sus hijos.
    - Confirmar asistencia a una tutoría.
    - Cancelar tutorías.

Según fue avanzando el proyecto, nos dimos cuenta de que era demasiado ambicioso y que, como el hecho de tener que llevar a cabo el desarrollo con tecnologías que no habíamos utilizado anteriormente, no contábamos con tiempo suficiente para desarrollar todas las funcionalidades.

Por ello, decidimos implementar solamente las funcionalidades necesarias para la comunicación entre el centro y los padres de alumnos y eliminamos el sistema de tutorías.

6. Capítulo 6: Tecnologías y herramientas utilizadas

6.1. Herramientas

- PHPStorm 2017.1.1: es un IDE para desarrollar en PHP, desarrollado 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.

- StarUML: es una herramienta que soporta diagramas UML.

- 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.

- Justinmind Prototyper: es una herramienta de prototipado para realizar prototipos de la interfaz gráfica de aplicaciones web y aplicaciones móviles. Contiene herramientas *drag and drop* para la colocación de elementos, animaciones, gestión de eventos y enlaces.

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

6.2. 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 y los intérpretes para lenguajes de script PHP y Perl.

- Apache: es un servidor web HTTP de código abierto, multiplataforma, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

6.3. 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 las aplicaciones web basado 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

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.

- Twig: es un motor de plantilla para el lenguaje de programación PHP. Es un producto de código abierto autorizado bajo Licencia BSD. El *framework* Symfony viene con un soporte incluido para Twig como su motor de plantilla por defecto.

- 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).

- JavaScript: es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

- Ajax: es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

- JQuery: es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Es la biblioteca de JavaScript más utilizada. Es software libre y de código abierto, ofrece una serie de funcionalidades basadas en JavaScript, que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

- 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.

7. Capítulo 7: Análisis

7.1. Actores

Hemos identificado dos actores diferentes que utilizarán el sistema. A continuación, se explicará el rol de cada uno dentro del sistema.

7.1.1. Administrador

Será personal del centro designado para la administración de la mensajería del mismo. Sus funciones principales serán las siguientes:

- Gestión de cursos y alumnado del centro: estas funciones no son el objetivo final de la aplicación, pero son necesarias para que el sistema pueda funcionar. Entre estas funciones se encuentran:
    - Importar automáticamente los datos del centro.
    - Crear cursos.
    - Registrar alumnos.
    - Asociar y desasociar alumnos de un curso.
    - Asociar y desasociar padres de un alumno.

- Gestión de mensajería: es el objetivo principal del sistema. Se podrán enviar distintos tipos de mensajes a los padres de los alumnos como:
    - Encuestas para que los padres puedan formar parte de las decisiones importantes del centro. Luego, podrán visualizar los resultados de las mismas.
    - Autorizaciones de actividades de los alumnos para que los padres puedan firmarlas, si les parece bien, desde su *smartphone*. También, se podrá visualizar un listado de los alumnos que han sido autorizados y otro con los alumnos que no han sido autorizados.
    - Circulares informativas para informar a los padres sobre algún acontecimiento relevante relacionado con el centro.

7.1.2. Padre

Son los padres de los alumnos. Podrán realizar las siguientes tareas:

- Ver los mensajes que les ha enviado el centro (circulares informativas, autorizaciones de actividades y encuestas).
- Filtrar mensajes por asunto.
- Firmar autorizaciones, utilizando una contraseña establecida por él/ella.
- Responder encuestas.
- Seleccionar qué centros pueden acceder a sus datos.
- Ver sus hijos asociados.

Además, son ellos quienes se registran en el sistema, ya que, de ser los centros quienes los registrasen, se daría el caso de que, probablemente, estarían enviando mensajes a padres que no cuentan con la aplicación móvil, por lo que no estarían recibiendo los comunicados enviados por el centro.

7.2. Requisitos

Para la representación de los requisitos funcionales del sistema nos hemos apoyado en UML [22] (Lenguaje Unificado de Modelado).

Hemos escogido este lenguaje 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 de la carrera, por lo que tenemos 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 sea 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 titulado *"Análisis, diseño e implementación de una aplicación para dispositivos móviles multiplataforma para la comunicación entre centros educativos y padres de alumnos"*.

7.2.1. Casos de uso

A continuación, mostraremos el diagrama de casos de uso que hemos realizado para tener un esquema claro de los requisitos funcionales que hemos identificado, dividido en distintas imágenes para facilitar su lectura:

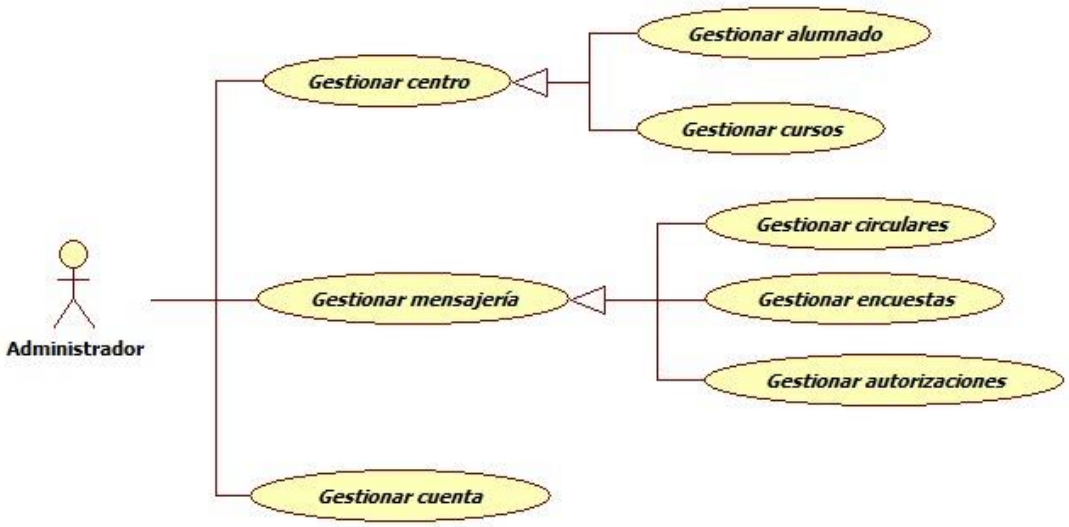

Ilustración 2: Diagrama de Casos de Uso (resumen)

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

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

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

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

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

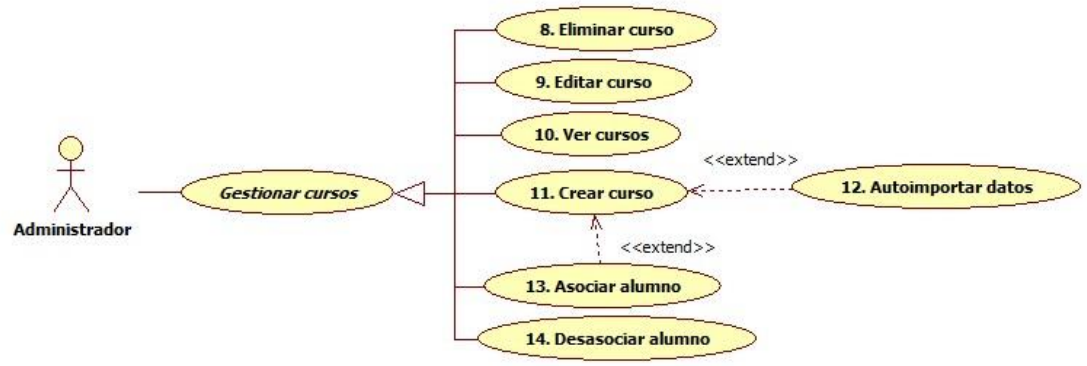

Ilustración 8: Diagrama de Casos de Uso (gestionar cursos)

En la siguiente tabla se muestra una breve descripción de los casos de uso anteriores:

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 padre registrado a un alumno del centro.
7	Administrador	Se desasocia un padre 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 padres 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 padres 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 padres 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 padres 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.

Tabla 2: Resumen de casos de uso

7.2.2. Especificación de casos de uso

En este apartado haremos la especificación de los casos de uso que son más relevantes para la aplicación:

CASO DE USO	15	**ENVIAR CIRCULAR**	
Descripción	El centro envía una circular informativa a los padres de un grupo de alumnos del centro.		
Actores	Administrador/a		
Precondiciones	El administrador/a debe haber iniciado sesión en el sistema.		
Flujo normal	**Paso**	**Acción**	
	1	Se establece un asunto para la circular	
	2	Se rellena el cuerpo del mensaje	
	3	Se seleccionan los alumnos cuyos padres deben recibir la circular	
	4	Se envía la circular	
Postcondiciones	Los padres de los alumnos seleccionados reciben la circular.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	1	Se desea adjuntar un archivo a la circular	CU 18: Adjuntar archivo
Excepciones	4	El asunto está vacío	

	4	No se ha seleccionado ningún alumno
Observaciones		

Tabla 3: Especificación de casos de uso (enviar circular)

CASO DE USO	20	**ENVIAR ENCUESTA**	
Descripción		El centro envía una encuesta para preguntar a los padres sobre un tema determinado.	
Actores		Administrador/a	
Precondiciones		El administrador/a debe haber iniciado sesión en el sistema.	
Flujo normal	**Paso**	**Acción**	
	1	Se establece un asunto para la encuesta	
	2	Se establece la fecha límite para responder la encuesta	
	3	Se rellena el cuerpo del mensaje	
	4	Se añaden opciones a elegir en la encuesta	
	5	Se seleccionan los alumnos cuyos padres deben recibir la encuesta	
	6	Se envía la encuesta	
Postcondiciones		Los padres de los alumnos seleccionados reciben la encuesta.	
Variaciones	**Paso**	**Acción**	
	4 A	Se elimina una opción de la encuesta	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	1	Se desea adjuntar un archivo a la encuesta	CU 18: Adjuntar archivo
Excepciones	6	El asunto está vacío	
	6	No se ha indicado la fecha límite	
	6	La fecha límite es anterior a la actual	
	6	No se ha seleccionado ningún alumno	
	6	No se han indicado, al menos, dos opciones de respuesta en la encuesta	
Observaciones			

Tabla 4: Especificación de casos de uso (enviar encuesta)

CASO DE USO	23	VER RESULTADOS		
Descripción		El administrador/a desea ver los resultados de una encuesta.		
Actores		Administrador/a		
Precondiciones		El administrador/a debe haber iniciado sesión en el sistema.		
Flujo normal		**Paso**	**Acción**	
		1	Se selecciona la encuesta deseada	
		2	Se abre un diálogo que contiene el mensaje y los resultados de la encuesta con el número de votos que ha recibido cada opción, destacando la opción que haya recibido más votos	
Postcondiciones		Se cierra el diálogo.		
Variaciones		**Paso**	**Acción**	
Extensiones		**Paso**	**Condición**	**Caso de Uso**
Excepciones				
Observaciones				

Tabla 5: Especificación de casos de uso (ver resultados)

CASO DE USO	25	ENVIAR AUTORIZACIÓN	
Descripción		El centro desea enviar una autorización a un grupo de alumnos para realizar una actividad determinada.	
Actores		Administrador/a	
Precondiciones		El administrador/a debe haber iniciado sesión en el sistema.	
Flujo normal		**Paso**	**Acción**
		1	Se establece un asunto para la autorización
		2	Se establece la fecha límite para autorizar la actividad
		3	Se rellena el cuerpo del mensaje
		4	Se seleccionan los alumnos cuyos padres deben recibir la autorización
		5	Se envía la autorización

Postcondiciones	Los padres de los alumnos seleccionados reciben la autorización.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	1	Se desea adjuntar un archivo a la autorización	CU 18: Adjuntar archivo
Excepciones	5	El asunto está vacío	
	5	No se ha indicado la fecha límite	
	5	La fecha límite es anterior a la actual	
	5	No se ha seleccionado ningún alumno	
Observaciones			

Tabla 6: Especificación de casos de uso (enviar autorización)

CASO DE USO	28	**VER ALUMNOS AUTORIZADOS**	
Descripción	El administrador/a desea ver qué alumnos han sido autorizados y cuáles no para realizar una actividad.		
Actores	Administrador/a		
Precondiciones	El administrador/a debe haber iniciado sesión en el sistema.		
Flujo normal	**Paso**	**Acción**	
	1	Se selecciona la autorización deseada	
	2	Se abre un diálogo que contiene el mensaje y dos listados. Uno de ellos contiene a los alumnos que han sido autorizados, y el otro contiene a los alumnos que no lo han sido	
Postcondiciones	Se cierra el diálogo.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones			
Observaciones			

Tabla 7: Especificación de casos de uso (ver alumnos autorizados)

CASO DE USO	24	EDITAR FECHA LÍMITE	
Descripción	El administrador/a desea modificar la fecha límite para poder responder una autorización o una encuesta.		
Actores	Administrador/a		
Precondiciones	El administrador/a debe haber iniciado sesión en el sistema.		
Flujo normal	**Paso**	**Acción**	
	1	Se selecciona la autorización o encuesta deseada	
	2	Se introduce la nueva fecha límite	
	3	Se modifica la fecha límite	
Postcondiciones	Se cierra el diálogo.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	3	La fecha es anterior a la actual	
Observaciones			

Tabla 8: Especificación de casos de uso (editar fecha límite)

CASO DE USO	12	AUTOIMPORTAR DATOS
Descripción	El administrador/a desea importar automáticamente los datos del centro.	
Actores	Administrador/a	
Precondiciones	El administrador/a debe haber iniciado sesión en el sistema.	
Flujo normal	**Paso**	**Acción**
	1	Se selecciona el archivo que contiene la información necesaria para importar automáticamente todos los datos del centro (cursos, alumnos y relaciones de alumnos con cursos y padres)
	2	Se solicita confirmación
	3	Se importan todos los datos

Postcondiciones	Los datos del fichero se guardan en el sistema.		
Variaciones	**Paso**	**Acción**	
	2 A	El administrador cancela la importación de datos	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	3	El archivo no tiene el formato correcto	
	3	El archivo contiene cursos repetidos	
Observaciones			

Tabla 9: Especificación de casos de uso (autoimportar datos)

A continuación, explicaremos el formato que hemos diseñado y que deberá tener el archivo utilizado para importar los datos del centro (cursos, alumnos, asociaciones entre cursos y alumnos y asociaciones entre alumnos y padres) de forma automática.

Hemos decidido utilizar archivos en formato CSV, los cuales son un tipo de documento sencillo utilizado para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.

El formato que hemos utilizado es el siguiente:

```
[nombre_del_curso1]
[nombre_del_alumno1],[apellidos_del_alumno1],[teléfono_padre1],[
teléfono_padre2],…
[nombre_del_alumno2],[apellidos_del_alumno2],[teléfono_padre1],[
teléfono_padre2],…
…
[nombre_del_curso_2]
…
```

A continuación, mostraremos un pequeño ejemplo de archivo para la importación automática de datos del centro con el objetivo de aclarar su formato:

```
1º A
Juan,Díaz,666666666,600000000
Luís,Santos,611111111
1º B
María,Cáceres,622222222
```

- Tanto el nombre del curso como el nombre y los apellidos de los alumnos son campos obligatorios.
- En el caso de que aún no podamos asociar padres a los alumnos, podrán registrarse los alumnos incluyendo su nombre y sus apellidos, sin ser necesario incluir los números de teléfono de sus padres.
- Si alguno de los padres incluidos no se ha registrado aún en el sistema, no se producirá ningún error.
- Si se incluyen cursos con nombres repetidos, será notificado al administrador.

8. Capítulo 8: Diseño

8.1. Diseño de la arquitectura del sistema

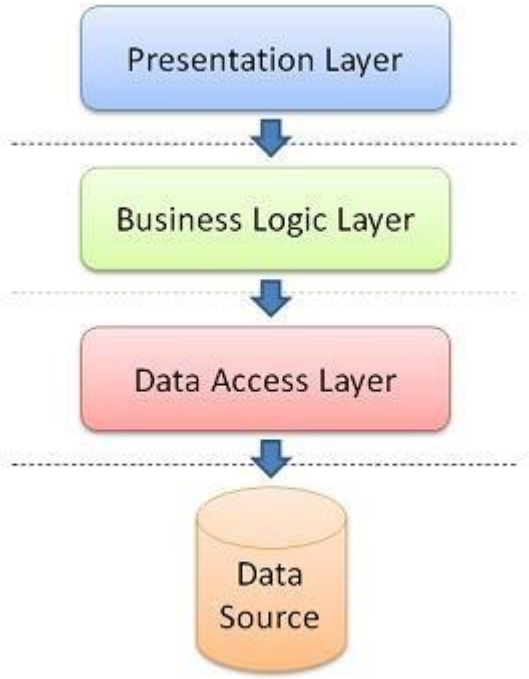

Ilustración 9: Arquitectura Multi-Nivel [2]

Nos hemos decantado por una arquitectura multi-nivel, donde las aplicaciones cliente solo contienen la capa de presentación y una capa de red para conectarse al servidor y alojamos el *backend* de la aplicación web y la API RESTful que utiliza la aplicación móvil en un servidor web (capas de negocio y de acceso a datos y datos). En la misma máquina se aloja la base de datos.

Hemos escogido esta arquitectura puesto que ofrece las siguientes características:

- Es fácilmente escalable: los clientes solo contienen contenido HTML, CSS y JavaScript que puede ser ejecutado desde cualquier navegador, por lo que solo debemos de preocuparnos de los recursos asignados al *backend*.
- Ahorra recursos: como acabamos de comentar, solo debemos de preocuparnos de los recursos asignados al *backend*.
- Es flexible: nos permite organizar mejor las distintas capas de software, lo que hace que sea más sencillo modificar una de ellas.
- Es segura: no es necesario aplicar políticas de seguridad complejas en las aplicaciones del cliente, ya que la lógica de negocio y los datos están alojados en un único servidor.

En el siguiente diagrama se muestra la arquitectura utilizada:

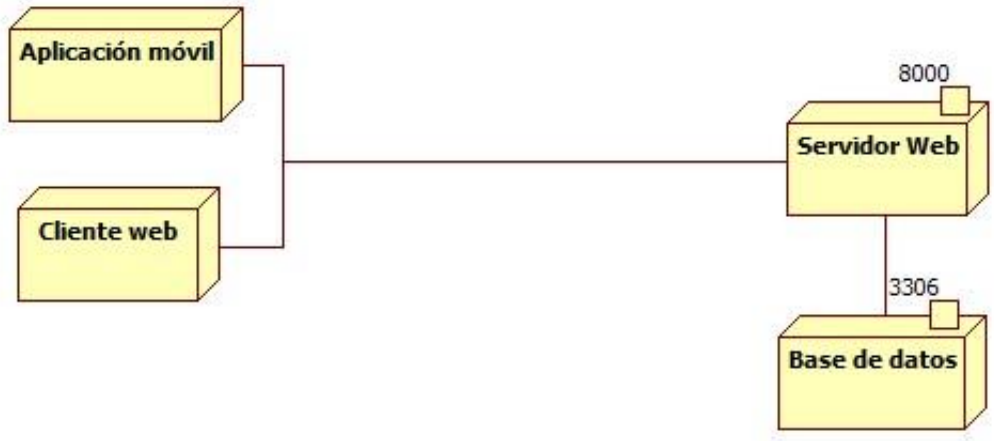

Ilustración 10: Diagrama de despliegue

8.1.1. Frontend

Como hemos comentado anteriormente, el sistema cuenta con dos aplicaciones diferentes:

- Aplicación web: utilizada por el centro. Permite enviar mensajes a los padres de los alumnos y ver los resultados de las encuestas y qué alumnos han sido autorizados para participar en una actividad.
- Aplicación móvil: utilizada por los padres. Permite ver los mensajes enviados por los centros educativos y responder encuestas y autorizaciones.

Hemos decidido que el centro cuente con una aplicación web que será utilizada desde algún equipo del centro. De esta manera, cualquier centro podrá utilizar nuestro sistema, ya que solo es necesario contar con un equipo.

En cambio, los padres cuentan con una aplicación móvil multiplataforma similar a cualquier aplicación de mensajería, la cual puede ser utilizada cómodamente desde un *smartphone*.

8.1.2. Backend

El *backend* se encuentra alojado en un servidor, el cual incluye un servidor web Apache que contiene tanto el *backend* de la aplicación web como la API REST que utiliza la aplicación móvil. En este servidor también se encuentra alojada la base de datos MySQL.

Hemos decidido que tanto el *backend* de la aplicación web como el de la aplicación móvil estén implementados utilizando la misma tecnología (Symfony3, PHP7 y Doctrine), de manera que la capa de acceso a la base de datos pueda ser utilizada por ambas aplicaciones.

8.2. Diseño de la base de datos

Como se ha nombrado anteriormente, hemos utilizado el sistema de gestión de bases de datos MySQL.

A continuación, mostraremos un diagrama en el que se podrán ver las tablas de la base de datos:

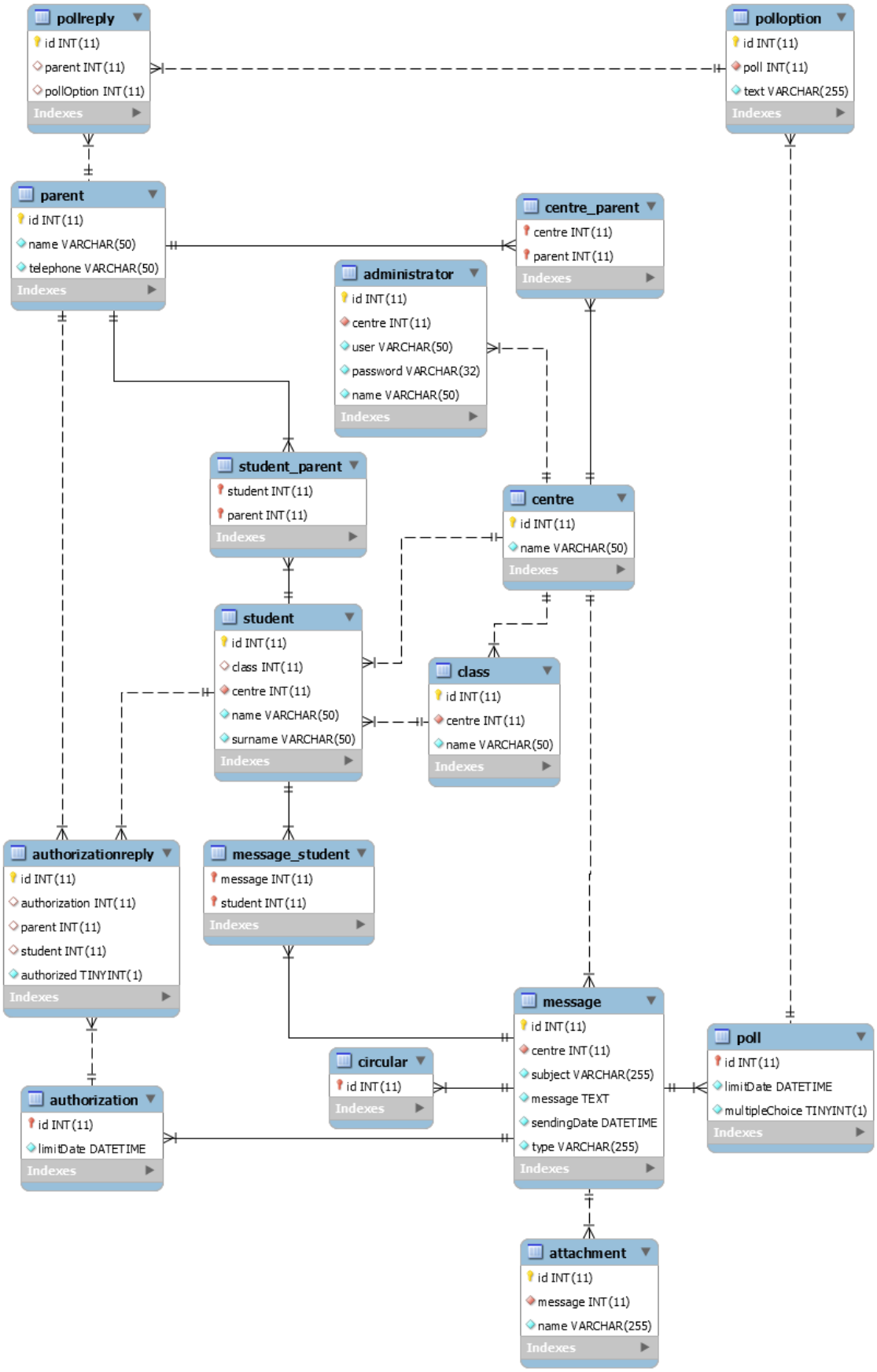

Ilustración 11: Diseño de la base de datos

Cabe destacar que las tablas *circular*, *poll* y *authorization* heredan de la tabla *message*, la cual contiene todos los campos comunes a los distintos tipos de mensaje. Hemos decidido utilizar una herencia para poder separar correctamente las relaciones entre las distintas tablas:

- Un *attachment* puede estar relacionado con cualquier tipo de *message*.
- Un *student* puede estar relacionado con cualquier tipo de *message*.
- Una *pollOption* solo puede estar relacionada con una *poll*.
- Una *authorizationReply* solo puede estar relacionada con una *authorization.*

Aunque actualmente solo puede adjuntarse un único archivo en cada mensaje, hemos decidido separar la tabla *attachment* de *message* para que, en un futuro, puedan adjuntarse varios archivos en cada mensaje.

En la siguiente tabla haremos una pequeña descripción de las distintas tablas de la base de datos que se responden con entidades:

Tabla	Descripción
centre	Tabla que contiene los centros educativos.
administrator	Tabla que contiene los administradores que utilizarán la aplicación web.
class	Tabla que contiene los cursos de cada centro.
student	Tabla que contiene los estudiantes de cada centro.
parent	Tabla que contiene los padres registrados desde la aplicación móvil.
message	Tabla que contiene los distintos mensajes enviados por los centros educativos.
authorization	Tabla que contiene las autorizaciones enviadas por los centros.
circular	Tabla que contiene las circulares enviadas por los centros.
poll	Tabla que contiene las encuestas enviadas por los centros.
pollOption	Tabla que contiene las distintas opciones de cada encuesta.
pollReply	Tabla que contiene las opciones de las encuestas escogidas por los padres.
authorizationReply	Tabla que contiene las respuestas de los padres en las cuales autorizan, o no, a un alumno para realizar una actividad.
attachment	Tabla que contiene el nombre de los archivos adjuntos de los mensajes.

Tabla 10: Explicación de las tablas de entidades de la base de datos

A continuación, mostraremos una tabla con una pequeña descripción para las distintas tablas de la base de datos que se corresponden con tablas que relacionan entidades:

Tabla	Descripción
student_parent	Tabla que relaciona padres con alumnos/hijos.
centre_parent	Tabla que relaciona padres con los centros que podrán acceder a ellos.
message_student	Tabla que relaciona mensajes con alumnos.

Tabla 11: Explicación de las tablas de relaciones de la base de datos

8.3. Almacenamiento de archivos en el servidor

Nos hemos decantado por guardar los archivos adjuntos en el sistema de ficheros en vez de dentro de la base de datos y estas son nuestras razones [23]:

- 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.

Para almacenar los archivos en el servidor en la base de datos guardamos, únicamente, el nombre del archivo y el mensaje al que está asociado. Como puede observarse en la *Ilustración 11*, la tabla *attachment* solo contiene dichos campos, además de un identificador.

Los ficheros se guardan en una carpeta del sistema de ficheros con identificador de la tabla *attachment* como nombre. Para recuperar el archivo del servidor solo necesitamos conocer el identificador del archivo asociado al mensaje y cambiarle el nombre por el original, también guardado en la tabla *attachment*.

8.4. Diseño Arquitectónico

Symfony es un *framework* basado en el patrón arquitectónico Modelo-Vista-Controlador (MVC).

Este patrón tiene tres componentes principales [3]:

- Modelo: es la representación de la información con la cual el sistema opera, por lo que 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.

- Vista: presenta el Modelo (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por lo que requiere de dicho Modelo la información que debe representar como salida.

- 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 comandos a su Vista asociada si se solicita un cambio en la forma en que se presenta el Modelo y, por tanto, se podría decir que el Controlador hace de intermediario entre la Vista y el Modelo.

El siguiente esquema muestra la interacción entre los distintos componentes del patrón MVC:

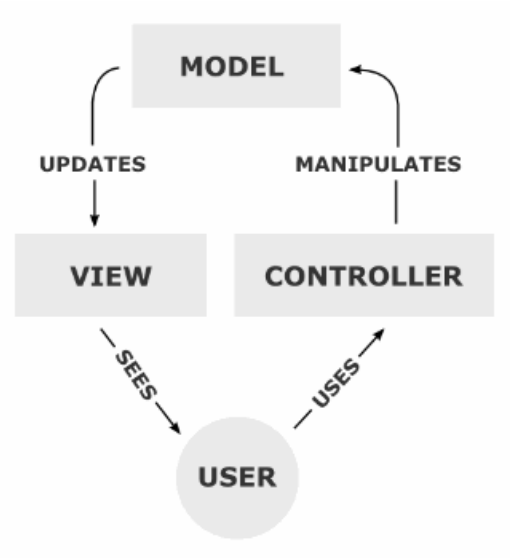

Ilustración 12: Patrón Modelo-Vista-Controlador [3]

8.5. Diseño de la interfaz

La interfaz de usuario es un elemento clave para que una aplicación tenga éxito. Por ello, es muy importante tener en cuenta algunos principios de diseño formulados por especialistas en la materia que hagan que la aplicación sea agradable a la vista y fácil de usar, logrando que el usuario tenga una experiencia agradable con ella.

Para que la interacción con las aplicaciones fuese lo mejor posible nos basamos en algunos de los principios de diseño de interacción del consultor de usabilidad Bruce Tognazzini [24]. A continuación, presentaremos los principios de diseño que hemos tenido en cuenta.

8.5.1. Anticipación

Proporcionar toda la información y las herramientas necesarias para realizar una tarea, evitando cambios de pantalla innecesarios.

Para cumplir con este principio, nos aseguramos de que cada pantalla contiene toda la información necesaria para poder llevar a cabo una acción determinada, de manera que pueda realizarse de la manera más rápida posible.

Por ello, cada acción despliega un diálogo modal que contiene toda la información necesaria para realizar la tarea y que, además, oculta el contenido de la página para no sobrecargar al usuario con información que, en ese momento, le resulte irrelevante.

8.5.2. Consistencia

Los objetos similares deben comportarse de la misma manera (consistentes) y los objetos que no son similares no deben comportarse de la misma manera (inconsistentes).

Es muy importante que todos los objetos que realizan un tipo determinado de acción tengan la misma apariencia y que, además, se comporten de la misma manera, de forma que cuando el usuario va a realizar una acción sobre un objeto habiendo realizado, anteriormente, una acción similar no se encuentre con un comportamiento distinto al que esperaba.

Por ello, hemos separado los botones en distintas categorías (informativo, común, primario y advertencia) de tal forma que, solamente viendo el color, el usuario ya sabe qué tipo de acción realizará el botón.

También, todos los campos que no pueden estar vacíos en un formulario, siempre están acompañados de un asterisco que indica que es obligatorio.

Además, todas las páginas de la web siguen la misma estructura, utilizando siempre las mismas barras de navegación y de comandos. De esta forma, conseguimos que los usuarios no tengan ningún problema al navegar a través de las distintas páginas de la aplicación, aumentando su grado de satisfacción con la aplicación y su rendimiento.

8.5.3. Valores por defecto

Deben poder ser descartados con facilidad y rapidez, si no están claros es mejor no poner nada.

Hemos tenido muy en cuenta que es necesario que la aplicación ofrezca siempre los valores por defecto que consigan que las distintas tareas puedan ser llevadas a cabo de la manera más rápida posible.

Por ello, cada vez que abrimos un diálogo modal, todos los valores de sus distintos campos se reestablecen de tal forma que no es necesario que el usuario tenga que borrar los valores que fueron introducidos anteriormente. Además, hemos tenido en cuenta que la aplicación rellene de forma automática los campos con los datos actuales cuando el usuario desee editar algún recurso.

8.5.4. Uso de metáforas

Deben permitir al usuario comprender los detalles del modelo conceptual, generándoles fuertes conexiones con experiencias pasadas, lo que permite que capten, con rapidez y precisión, las capacidades y limitaciones de su sistema.

Para que la aplicación sea más fácil de entender, hemos intentado que el sistema de mensajería se parezca a un sistema corriente de correo electrónico, cuyo funcionamiento ya ha sido asimilado previamente por la mayoría de usuarios. De esta forma, intentamos conseguir que sea más fácil aprender a utilizar la aplicación.

Además, adaptamos el concepto de correo electrónico a las necesidades de nuestros usuarios, permitiendo enviar distintos tipos de mensaje preestablecidos (circulares, autorizaciones y encuestas) a distintos grupos de padres de manera rápida y sencilla (organización de alumnos por cursos).

8.5.5. Legibilidad

El texto debe tener buen contraste con el fondo y un tamaño de fuente suficientemente grande, la información que sea importante debe mostrarse en un tamaño mayor.

En todas las páginas hemos cuidado el contraste entre las fuentes y el fondo sobre el que se encuentran, utilizando tonos que tengan buen contraste entre ellos (fuentes oscuras sobre fondos claros y viceversa). De esta manera, conseguimos que el texto sea fácil de leer.

También, destacamos información relevante utilizando negritas y fuentes de mayor tamaño.

8.5.6. Simplicidad

Eliminar funcionalidades innecesarias y hacer que sean lo más simple posible.

Es uno de los conceptos que más hemos tenido en cuenta a la hora de desarrollar las aplicaciones, ya que el exceso de información puede frustrar a los usuarios, lo que puede tener como consecuencia que utilicen la aplicación de forma incorrecta o que no se sientan cómodos a la hora de interaccionar con ella.

Por ello, hemos decidido no incluir información que no es relevante para el objetivo de la aplicación (fechas de nacimiento de alumnos, direcciones, DNIs de los padres, etc.), que es que el centro y los padres de los alumnos puedan comunicarse rápida y fácilmente.

8.5.7. Navegación visible

Reducir la navegación al máximo y hacer que sea clara y natural, procurar que parezca que el usuario está siempre en el mismo sitio.

Para que el usuario se sienta a gusto utilizando la aplicación, hemos intentado que todas las funcionalidades relacionadas se encuentren siempre en una misma pantalla, de manera que no es necesario realizar cambios de pantalla innecesarios.

Por ello, todas las acciones se llevan a cabo mediante diálogos modales, los cuales incluyen información suficiente para llevar a cabo una tarea sin tener que realizar búsquedas en otras páginas.

9. Capítulo 9: Desarrollo

9.1. Symfony

"Symfony es un conjunto de componentes de PHP, un framework para aplicaciones web, una filosofía, y una comunidad – Donde todo trabaja en armonía." [25]

Sus principales características, por las cuales nos hemos decantado por utilizarlo, son:

- Su código, y el de todos los componentes y librerías que incluye, se publican bajo la licencia MIT de software libre.

- La documentación del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos.

- Aprender a programar con Symfony te permite acceder a una gran variedad de proyectos: el *framework* Symfony3 para crear aplicaciones complejas, el micro *framework* Silex para sitios web sencillos y los componentes Symfony para otras aplicaciones PHP. Algunos de los *frameworks* de PHP más utilizados (por ejemplo, Laravel) utilizan componentes de Symfony.

- Según GitHub, Symfony es el proyecto PHP más activo, lo que garantiza que nunca te quedarás atrapado en un proyecto sin actividad. Además, el líder del proyecto, Fabien Potencier, es la segunda persona más activa del mundo en GitHub.

- Los componentes de Symfony son tan útiles y están tan probados, que proyectos tan gigantescos como Drupal 8 están construidos con ellos.

- Las versiones actuales de Symfony requieren disponer de PHP 5.3.8 o superior. Así evitas instalar en tus servidores versiones PHP peligrosas llenas de problemas de seguridad y a la vez no es un requisito técnico demasiado exigente.

- En producción, las aplicaciones Symfony solamente necesitan permiso de escritura en dos directorios internos de la propia aplicación. Además, Symfony incluye varias herramientas gráficas y de consola para depurar fácilmente los errores que se produzcan en las aplicaciones.

- Para evitar el uso de contraseñas en archivos de configuración, Symfony permite establecer los parámetros de configuración de las aplicaciones a través de variables de entorno del propio servidor.

- La seguridad es tan importante para el proyecto Symfony, que antes de su lanzamiento, se encargó una auditoría de seguridad a una empresa.

- Su licencia de tipo MIT permite crear gratuitamente incluso aplicaciones comerciales.

- Dispone de un plan de lanzamientos predecible, con versiones estables mantenidas durante tres años.

- Su primera versión se publicó en el año 2005, por lo que es un proyecto maduro que ha sido probado en decenas de miles de sitios y aplicaciones web.

- En su desarrollo participan cientos de programadores de todo el mundo, pero las decisiones técnicas importantes siempre las toma un reducido grupo de líderes, lo que garantiza que se mantenga la visión del proyecto y evita la descoordinación.

9.2. Estructura del proyecto

Una característica de Symfony es que cuenta con una estructura de directorios bien definida, lo que permite que, si utilizamos la estructura recomendada, podremos movernos fácilmente por proyectos que hayan sido desarrollados por cualquier equipo de desarrollo.

Los directorios principales son los siguientes:

- /app/: ficheros de configuración de la aplicación, plantillas y traductores.
- /bin/: ficheros ejecutables (por ejemplo, la consola).
- /src/: el código PHP del proyecto (controladores, entidades, etc.).
- /tests/: tests.
- /var/: ficheros generados (cache, logs, etc.).
- /vendor/: dependencias.
- /web/: directorio raíz de la aplicación web. Aquí incluimos imágenes, archivos CSS y JavaScript, entre otros.

9.3. Control de acceso a la aplicación web

Como ya hemos comentado, Symfony está formado por un conjunto de componentes, que son librerías reutilizables de PHP. Para controlar el acceso a la aplicación nos hemos apoyado en el componente Security [27], que proporciona una infraestructura para sistemas de autorización sofisticados.

Gracias a él, hemos conseguido implementar un sistema de autenticación para acceder a la aplicación web en pocas líneas de código.

Nuestro sistema de acceso a la aplicación web es bastante sencillo. Simplemente, hay que cumplir una única condición para poder acceder a ella, haber iniciado sesión como administrador.

Para ello, hemos realizado las siguientes configuraciones:

9.3.1. security.yml

El sistema de seguridad se configura en el fichero *app/config/security.yml*.

La palabra clave *firewall* es el corazón de la configuración de nuestro sistema de seguridad.

La entrada *access_control* permite indicar qué roles de usuario pueden acceder a ciertas rutas en la aplicación. En nuestro caso, queremos que cualquier página, excepto la de inicio de sesión, requiera que se haya iniciado sesión como administrador. Para ello, añadimos las siguientes líneas:

```
access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/, roles: ROLE_USER }
```

Ahora, debemos indicar de dónde obtenemos los usuarios. En nuestro caso, será de la tabla *administrador* de la base de datos.

Para ello, añadimos las siguientes líneas:

```
providers:
    db_provider:
        entity: {class: AppBundle:Administrator, property: user}
```

Una vez hecho esto, debemos indicar qué algoritmo hemos utilizado para codificar las contraseñas.

En nuestro caso, creamos las cuentas de administrador directamente en la base de datos, codificando sus contraseñas mediante el algoritmo MD5, cuya característica principal es que reduce cualquier ristra de caracteres en otra ristra de tamaño fijo (32 caracteres).

Para ello, añadimos las siguientes líneas:

```
encoders:
    AppBundle\Entity\Administrator:
        algorithm: md5
        encode_as_base64: false
        iterations: 0
        ignore_case: false
```

El problema principal que presenta este algoritmo, es que es el tamaño del *hash* es lo suficientemente pequeño como para que resulte vulnerable frente a ataques de fuerza bruta, por lo que es muy sencillo descifrar el contenido original.

Somos conscientes de que este algoritmo no es adecuado para una aplicación en entorno de explotación, puesto que supone un problema importante de seguridad. Sin embargo, es sencillo de utilizar y es ideal para realizar un prototipo donde no existe información sensible.

Por último, configuramos el *firewall* para indicarle el *provider* la ruta de *login* y la ruta de *logout*. Para ello, añadimos las siguientes líneas:

```
secured_area:
    anonymous: ~
    provider: db_provider
    form_login:
        login_path: login
        check_path: login_check
    logout:
        path: /logout
        target: /login
```

9.3.2. routing.yml

En este fichero creamos las rutas necesarias para que Symfony inicie y cierre la sesión del usuario. Para ello, añadimos las siguientes líneas:

```
login:
    path:   /login
    defaults:  { _controller: AppBundle:Security:login }
```

```
login_check:
    path:   /login_check

logout:
    path:   /logout
```

9.3.3. SecurityController

Este controlador contiene las acciones de inicio y cierre de sesión.

En caso de que el usuario introduzca unas credenciales incorrectas en el formulario de inicio de sesión o se produzca algún error, la acción de inicio de sesión devolverá un mensaje de error, además del nombre de la cuenta de usuario introducida por el usuario que no tenga que volver a introducirla en el formulario.

9.4. Ajax

Para evitar recargar innecesariamente las distintas páginas de la aplicación tras realizar una acción, lo cual mejora la interactividad, velocidad y usabilidad en la misma, todas las acciones se realizan mediante peticiones Ajax, las cuales funcionan de la siguiente forma:

1. Ocurre un evento en el navegador y se ejecuta una función JavaScript.
2. La función JavaScript envía una petición Ajax al servidor.
3. El servidor procesa la petición y devuelve una respuesta, generalmente, en formato JSON.
4. El navegador ejecuta el *callback* de la petición Ajax y, si es necesario, se actualiza el contenido de la página.

A continuación, mostraremos un esquema que explica el funcionamiento de las peticiones Ajax de manera visual:

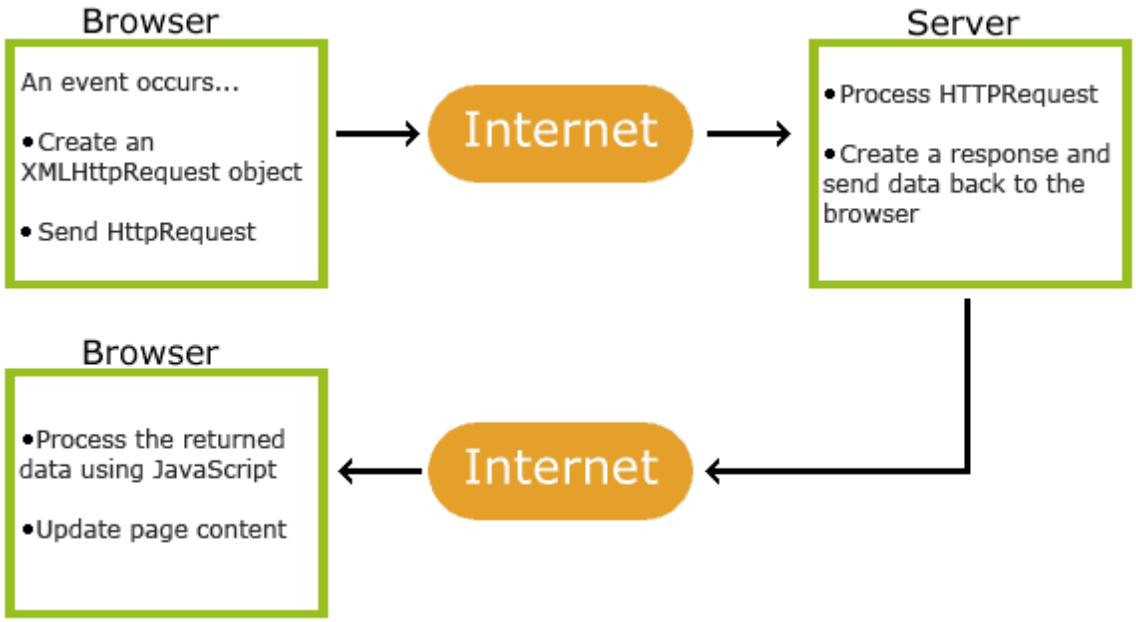

Ilustración 13: Esquema de peticiones Ajax [4]

Para procesar las respuestas devueltas por el servidor, una buena práctica es que todas sigan el mismo formato, de tal forma que sea más sencillo extraer información de ellas desde el cliente web.

Para conseguir esto, hemos utilizado un patrón de diseño llamado Simple Factory, el cual consiste en definir una clase que se encarga de crear nuevas instancias de objetos.

Esta clase, la cual hemos denominado *ResponseFactory*, contiene una función estática que recibe dos parámetros, que son el tipo de respuesta y el contenido de la misma.

El tipo de respuesta permite a la función saber si debe crear una respuesta de tipo *success* o una respuesta de tipo *error*.

Para cada tipo de respuesta, su formato correspondiente es:

- Success: `{'success':true, 'content':[…]}`
- Error: `{'success':false, 'mensaje de error'}`

La fábrica de respuestas también se utiliza en el servicio web, de forma que sus respuestas también siguen un formato estándar.

9.5. Enrutamiento

El enrutamiento es un factor importante, tanto en el *backend* de la aplicación web como en el servicio web que utiliza la aplicación móvil.

Las peticiones Ajax realizadas desde la aplicación web al servidor web hacen uso de las rutas establecidas en los controladores del *backend*. Para que las rutas estén bien organizadas y sea sencillo acceder a ellas desde la aplicación web, es muy importante seguir una estructura eficiente y utilice siempre el mismo formato.

Por ello, hemos seguido una serie de prácticas recomendadas para realizar APIs RESTful pragmáticas [29] para implementar el *backend* de las aplicaciones, de forma que la API:

- Utiliza estándares web.
- Es amigable para el desarrollador y es explorable mediante una barra de direcciones del navegador.
- Es sencilla, intuitiva y consistente.
- Es flexible, para potenciar la interfaz de usuario con los desarrolladores.
- Es eficiente, manteniendo el equilibro con los demás requisitos.

Para desarrollar una API RESTful pragmática hemos utilizado los siguientes practicas:

- Separamos la API en recursos lógicos. Los recursos son sustantivos que tienen sentido desde una perspectiva del consumidor de la API.
- Escribimos los recursos en plural para mantener URLs consistentes. De esta forma, evitamos lidiar con plurales irregulares y hacemos más simple la vida del consumidor de la API.
- Utilizamos peticiones HTTP donde cada método tiene un significado específico para manejar acciones CRUD (crear, leer, actualizar y borrar) utilizando métodos HTTP de la siguiente forma:
    - GET: para devolver un recurso.
    - POST: para crear un recurso.
    - PUT: para actualizar un recurso.
    - PATCH: para actualizar parcialmente un recurso.
    - DELETE: para eliminar un recurso.

- Filtramos recursos utilizando un único parámetro de consulta por cada campo que implemente el filtro.

A continuación, mostraremos una serie de ejemplos para aclarar la creación de URLs que siguen las técnicas explicadas anteriormente:

- *GET /students*- Devuelve una lista de alumnos.
- *GET /students?name=Juan*- Devuelve una lista de alumnos llamados Juan.
- *GET /students/12*- Devuelve el alumno #12.
- *POST /students*- Crea un nuevo alumno.
- *PUT /students/12*- Actualiza el alumno #12.
- *PATCH /students/12*- Actualiza parcialmente el alumno #12.
- *DELETE /students/12*- Elimina el alumno #12.
- *GET /classes/12/students* - Devuelve una lista de alumnos para la clase #12.
- *POST /classes/12/students/5* - Asocia al alumno #5 a la clase #12.
- *DELETE /classes/12/students/5* - Borra el alumno #5 de la clase #12.

9.6. Twig

Para representar las vistas, Symfony se apoya en un motor de plantillas llamado Twig, para evitar que el programador tenga que utilizar código en PHP en la capa de presentación. Las características más importantes de Twig son:

- Es conciso: al contrario que PHP, utiliza una sintaxis muy concisa para hacer plantillas mucho más legibles.
- Utiliza una sintaxis orientada a plantillas: tiene atajos sencillos para patrones sencillos, de manera que podemos realizar acciones complejas escribiendo muy poco.
- Es completo: con tiene todo tipo de funcionalidades para construir plantillas complejas de manera sencilla (herencia múltiple, bloques…).
- Es fácil de aprender: su sintaxis es muy sencilla, lo que hace que sea cómodo trabajar con él.
- Está bien documentado: esto hace que sea muy sencillo aprender a utilizarlo.
- Muestra mensajes de error claros: cuando aparece un error de sintaxis, se muestran mensajes útiles que indican dónde ocurrió.
- Es rápido: compila las plantillas a código en PHP optimizado.

Para que las plantillas sean lo más simple posible, hemos hecho uso de herencia entre plantillas creando una plantilla base de la que heredan las distintas páginas, de tal forma que evitamos duplicar código en ellas.

Así, además de ahorrar tiempo al hacerlas, si una sección de una página (por ejemplo, encabezado o barra de navegación) es común a varias páginas, cuando necesitamos hacer una modificación en alguna de estas secciones, con modificar la plantilla base todas las páginas son modificadas automáticamente.

9.7. Doctrine

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 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:

```
database_host: 127.0.0.1
database_port: 3306
database_name: hermerest
database_user: hermerest
database_password: hermerest
```

Como se puede apreciar, solamente es necesario indicar el *host* de la base de datos, el puerto a través del cual accederemos a ella y el nombre, usuario y contraseña de la base de datos.

Una vez hecho esto, podemos generar clases de entidad a partir de una base de datos existente o generar una base de datos a partir de unas clases de entidad implementadas por nosotros.

En nuestro caso, hemos decidido optar por la segunda opción, ya que Doctrine no es capaz de generar las clases de identidad correctamente a partir de una base de datos ya hecha si contiene ciertos tipos de relaciones entre tablas.

Por ello, tras configurar los distintos parámetros de conexión de la base de datos en el fichero *parameters.yml*, creamos en directorio */src/AppBundle/Controller/Entity*, donde creamos todas las clases de entidad que se corresponden con la capa de Modelo en la arquitectura MVC.

Para crear las clases de entidad, Doctrine utiliza un sistema de etiquetas que permiten el mapeo entre objetos de PHP y registros de la base de datos. Tan solo tenemos que crear una clase de PHP y añadir las etiquetas necesarias para realizar el mapeo.

Una vez implementadas las clases de entidad, abrimos la consola del sistema en el directorio raíz del proyecto y ejecutamos el siguiente comando para validar si los ficheros se han implementado correctamente:

```
php bin/console doctrine:schema:validate
```

Si los ficheros son correctos, ejecutamos el siguiente comando para actualizar la base de datos:

```
php bin/console doctrine:schema:update --force
```

Por último, si se desea, podemos ejecutar el siguiente comando para generar los *getters* y los *setters* de cada entidad:

```
php bin/console doctrine:generate:entities
AppBundle/Entity/EntityName
```

A partir de ahora, ya podemos tratar los datos de la base de datos como objetos y sacar el máximo partido de la programación orientada a objetos.

9.8. Clases de Entidad

Como explicamos en el apartado anterior, para generar las tablas de la base de datos tuvimos que implementar las clases de entidad. Estas clases equivalen a las clases de la capa de Modelo en el patrón arquitectónico Modelo-Vista-Controlador, es decir, las clases que contienen los datos que guardaremos en la base de datos.

A continuación, presentaremos un diagrama de clases con las clases de entidad que hemos implementado:

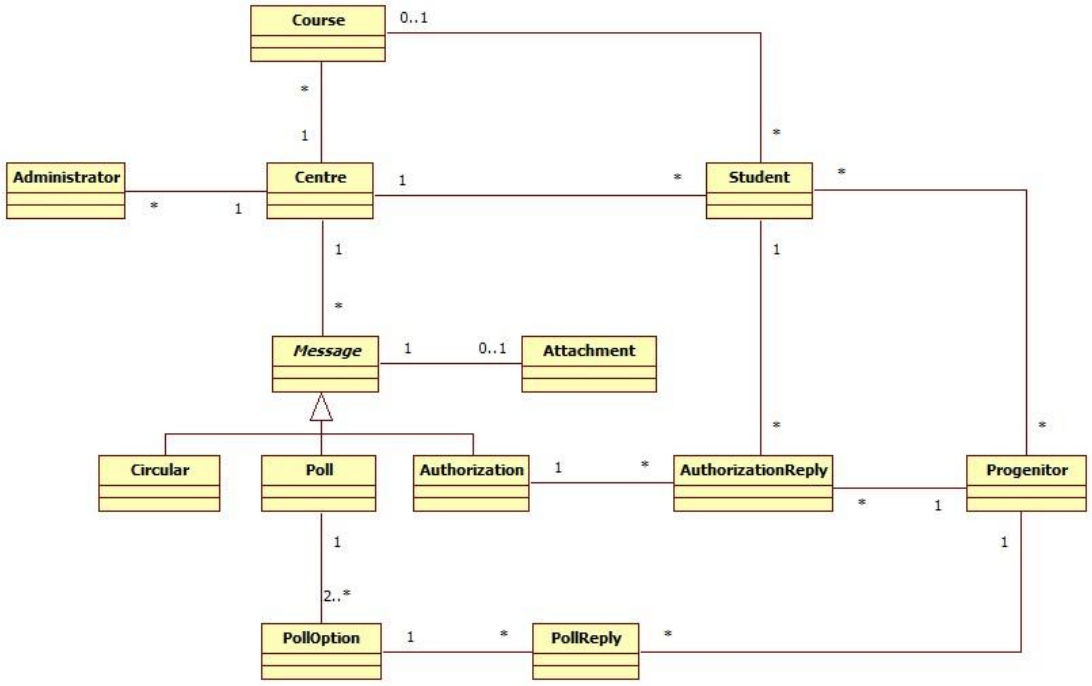

Ilustración 14: Diagrama de clases de entidad

Cabe destacar que las clases del diagrama anterior coinciden con las tablas de la base de datos presentadas en la *Tabla 10* (en la que podemos ver una descripción de las mismas), salvo por las siguientes diferencias:

- La tabla *parent* equivale a la clase *Progenitor*.
- La tabla *class* equivale a la clase *Course*.

Esto se debe a que *class* y *parent* son palabras reservadas en PHP, por lo que no pueden ser utilizadas para nombrar a una clase.

9.9. Capa de acceso a base de datos

Para que la interacción entre los controladores de la aplicación y la base de datos sea lo más sencillo posible, hemos utilizado el patrón estructural Facade. Este patrón consiste en crear una "fachada" que simplifique el acceso a la base de datos desde los controladores de la aplicación.

De esta manera, conseguimos que la interacción con la base de datos sea lo más simple posible en cualquier controlador de la aplicación web, al no tener que tratar con ella directamente.

Además, el uso de este patrón permite que el desarrollo del servicio web que utiliza la aplicación móvil, el cual será implementado por otro alumno, no requiera del aprendizaje del funcionamiento de Doctrine para poder interactuar con la base de datos.

En el siguiente diagrama se muestra un pequeño ejemplo del uso del patrón Facade en el proyecto:

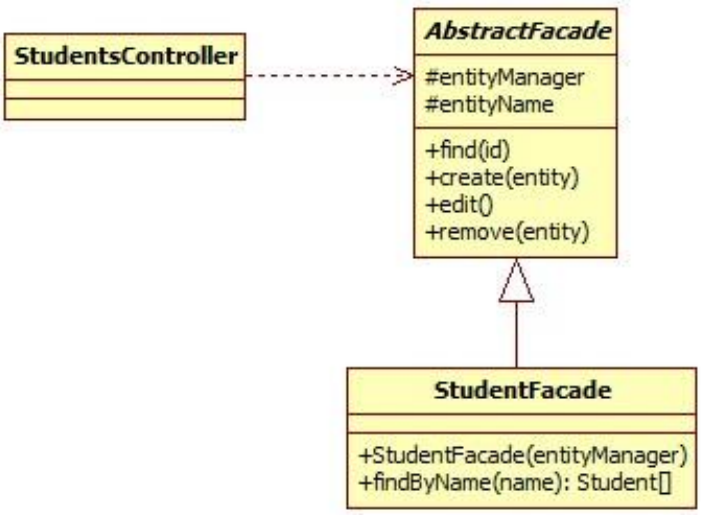

Ilustración 15: Patrón Facade

A continuación, haremos una serie de explicaciones acerca de la implementación del patrón que hemos realizado:

- El controlador crea una instancia de *StudentFacade* proporcionándole el *entityManager* (objeto encargado para leer y escribir en la base de datos).
- Para que Doctrine sepa sobre qué entidad debe trabajar la instancia del *Facade* creada, *StudentFacade* proporciona el nombre de dicha entidad llamando al constructor de *AbstractFacade* con ese nombre.
- Todos los *Facades* heredan de *AbstractFacade*.
- En *AbstractFacade* tenemos métodos estándar que son útiles para cualquier entidad, como son los necesarios para las operaciones CRUD (crear, eliminar, actualizar y borrar).
- Cada subclase de *AbstractFacade* tiene sus métodos específicos para acceder a la base de datos. Por ejemplo, en el caso de los alumnos, tenemos un método que nos permite buscarlos según su nombre.

En resumen, los controladores nunca accederán directamente a la base de datos, sino que crearán una instancia de algún *Facade* y ejecutarán algunos de sus métodos, los cuales se encargarán de la interacción con la base de datos.

9.10. Capa de acceso al sistema de ficheros

Al igual que con la base de datos, también es recomendable que la interacción con el sistema ficheros sea transparente desde un controlador.

Una de las razones que apoyan el enunciado anterior es que los sistemas de ficheros son distintos según el sistema operativo en el que se encuentre instalada la aplicación.

Por ello, hemos decidido crear una clase encargada de gestionar toda la interacción entre la aplicación con el sistema de ficheros, la cual hemos denominado *AttachmentManager*.

En nuestro caso, utilizamos el sistema de ficheros para guardar los archivos adjuntos de los mensajes en el servidor.

Para que esta acción se lleve a cabo de manera sencilla desde los controladores de la aplicación, incluimos una función en la clase anterior a la que proporcionamos el nombre del archivo y su contenido para que lo guarde en el sistema de ficheros.

Gracias a esto, los controladores nunca incluirán código que se corresponda con la interacción de la aplicación con el sistema de ficheros, permitiendo que, desde su punto de vista, sea una tarea sencilla y que no depende del sistema de ficheros utilizado.

10. Capítulo 10: Pruebas

10.1. Pruebas de integración

Aunque no disponemos de un servidor que esté siempre operativo donde poder desplegar el sistema, hemos comprobado que las distintas aplicaciones funcionarían correctamente en un entorno en el que el *backend* y la base de datos estuvieran alojados en un servidor y las aplicaciones cliente funcionaran desde una red externa.

Para probar esto:

- Desplegamos la base de datos y el *backend* de las aplicaciones en uno de nuestros equipos para que funcionase como servidor.
- Utilizamos la aplicación web en un equipo de una red externa a la del servidor y comprobamos que funcionaba correctamente.
- Instalamos la aplicación móvil en un *smartphone* y comprobamos que funcionaba correctamente desde una red externa a la del servidor, tanto en IOS como en Android.

Una vez hecho esto, comprobamos que las acciones realizadas en una de las aplicaciones tenían un efecto inmediato en la otra aplicación, de manera que simulamos un escenario real en el que el sistema debería responder correctamente. Por ejemplo:

- Si registramos a un padre desde la aplicación móvil y lo asociamos a un centro, comprobamos que un administrador de ese centro puede acceder al padre desde la aplicación web.
- Si enviamos un mensaje a algún hijo del padre registrado desde la aplicación web, se puede ver el mensaje desde la aplicación móvil.
- Si se responde un mensaje desde la aplicación móvil se puede ver el resultado de la respuesta en la aplicación web.

10.2. Pruebas de usabilidad

El objetivo de cualquier aplicación es lograr que sus usuarios solventen algún problema lo más eficientemente posible. Por ello, es muy importante comprobar que se sienten cómodos con la aplicación y que su uso les permite realizar las tareas con facilidad.

Para asegurarnos de que cada versión del prototipo se adecúa a las características de sus usuarios, hemos realizado distintas demostraciones a medida que avanzábamos con el desarrollo. De esta manera, hemos ido ajustando las funcionalidades implementadas para que resolvieran los problemas de los usuarios de la mejor manera posible.

Durante las demostraciones, comprobamos que los usuarios:

- Navegaban sin problema por las distintas páginas de la aplicación.
- Entendían la estructura de las páginas.
- Percibían un orden lógico en los distintos pasos de cada tarea.
- Eran capaces de realizar las distintas tareas sin ayuda.
- Tenían toda la información necesaria en los distintos pasos de cada tarea.
- No se sentían molestos debido a los tiempos de carga.
- Entendían los mensajes de error mostrados cuando introducían datos incorrectos y les ayudaban a corregirlos.

Es un error muy común culpar a los usuarios por no utilizar correctamente una aplicación informática. Sin embargo, lo correcto es detectar los errores que cometen y hacer lo posible para que los cometan con la menor frecuencia posible.

A la hora de estudiar la interacción de los usuarios con las distintas versiones del prototipo, utilizamos una filosofía en la que, si un usuario cometía el mismo error repetidamente, o si detectábamos que varios usuarios cometían el mismo error, los responsables éramos los desarrolladores y no los usuarios.

Así, cuando detectábamos que la información proporcionada no resultaba clara o que la realización de alguna tarea era confusa, anotábamos el problema y lo corregíamos en la siguiente iteración.

11. Capítulo 11: Resultados, conclusiones y trabajo futuro

11.1. Resultados y conclusiones

La realización de este proyecto ha resultado una experiencia muy enriquecedora. Hemos podido aplicar y reforzar muchos conocimientos adquiridos en la carrera y en la intensificación de Ingeniería del Software. Además de reforzar conocimientos ya adquiridos, nos ha servido para aprender a desarrollar en nuevas tecnologías, como Symfony 3, Doctrine y PHP 7.1, Twig, entre otras.

Por otro lado, es la primera vez que realizamos un proyecto que cuenta con distintas aplicaciones desarrolladas en distintas tecnologías, lo que nos ha ayudado vivir de primera mano la experiencia de tener que diseñar e implementar una arquitectura de estas características, además de tener que estudiar las tecnologías existentes para escoger la que mejor se adapte a nuestras necesidades.

También, debemos destacar que nos sentimos muy satisfechos sabiendo que este tipo de proyectos que buscan eliminar el uso del papel, ayudan con el cuidado del medio ambiente disminuyendo considerablemente la tala de árboles y sus consecuencias negativas para el planeta.

Una de las conclusiones más importantes que hemos sacado tras haber terminado el proyecto es que nos hemos dado cuenta de que en estos años en la universidad no nos han enseñado únicamente a desarrollar en determinadas tecnologías, sino que hemos desarrollado una capacidad para poder arrancar un proyecto capaz de resolver problemas existentes desde cero, incluso utilizando tecnologías que desconocíamos sin depender de un experto para resolver nuestros problemas. Este aspecto tiene un gran valor para nosotros, puesto que nos ha dado confianza de cara a nuestro futuro.

Ahora sabemos que con esfuerzo y dedicación somos capaces de resolver cualquier obstáculo que se nos presente en nuestra vida profesional. Es cierto que en ciertas etapas del proyecto hemos tenido momentos en los que parecía que no avanzábamos, pero con paciencia y esfuerzo hemos ido solventando todos los problemas que nos han ido surgiendo.

El resultado que hemos obtenido ha sido muy gratificante, puesto que consideramos que hemos escogido un tema de gran importancia en la actualidad, como es la transición de la mensajería tradicional de los centros educativos hacia la mensajería mediante la informática. Además, las aplicaciones desarrolladas solventan el problema de tal forma que cualquier centro pueda utilizar nuestro sistema, aunque cuente con recursos informáticos escasos.

11.2. Trabajo Futuro

De cara a realizar una versión completa de este proyecto, podemos decir que tiene un largo camino por delante. Además de algunas mejoras sobre las características ya implementadas, podríamos implementar un gran número de funcionalidades que ampliarían la utilidad del sistema.

Como ya hemos comentado anteriormente, en un principio estudiamos la posibilidad de incluir un sistema de tutorías, para que la comunicación entre profesores y padres de alumnos también pudiese hacerse mediante la aplicación móvil. Sin embargo, nos dimos cuenta de que no iba a ser posible implementarlo en 300 horas, por lo que decidimos dejarlo para más adelante.

A continuación, describiremos algunas mejoras posibles sobre las funcionalidades ya implementadas y funcionalidades nuevas que serían útiles para nuestros usuarios:

- Reemplazar el algoritmo MD5 que utilizamos para cifrar las contraseñas de los administradores por otro más seguro.
- Refactorizar el código JavaScript para evitar duplicidad de código y, también, lograr que sea más limpio.
- Implementar un sistema de notificaciones *push* para que los padres reciban notificaciones a tiempo real cuando se les envíe un mensaje.
- Permitir que las encuestas tengan múltiples preguntas.
- Mejorar la estética de la interfaz de usuario.
- Aunque la aplicación no esté pensada para ser utilizada desde un dispositivo móvil, se podría hacer que la interfaz gráfica fuera *responsive*.
- Permitir que el centro pueda ver estadísticas acerca de la lectura de los mensajes, para controlar si los padres están utilizando la aplicación móvil o no.
- Permitir actualizar rápidamente todos los datos del centro a la hora de comenzar un nuevo año académico.
- Permitir adjuntar múltiples archivos a la hora de enviar un mensaje.
- Utilizar el protocolo HTTPS para que las conexiones sean seguras.

Por último, describiremos brevemente las características del sistema de tutorías que no pudimos implementar por falta de tiempo:

- Aplicación web
    - Administradores del centro:
        - Registrar profesores en el centro.
        - Asociar profesores a un curso.
        - Asociar un tutor a un curso.
        - Establecer asignaturas en el centro.
        - Asociar asignaturas a un profesor.
- Aplicación móvil
    - Profesores:
        - Establecer un horario de tutorías.
        - Solicitar tutorías a los padres.
        - Ver calendario de tutorías.
        - Confirmar tutorías.
        - Cancelar tutorías.
    - Padres:
        - Solicitar tutorías a los profesores de sus hijos.
        - Confirmar asistencia a una tutoría.
        - Cancelar tutorías.

12. Capítulo 12: Bibliografía

[1] Aruquipa, Oscar. “Análisis comparativos de los distintos ciclos de vida en el software”, [en línea]. 04 de octubre de 2012. Disponible en: http://oscararuquipacolque.blogspot.com.es/2012_10_01_archive.html

[2] Normén, Fredrik. “Using Web Services in a 3-tier architecture”, [en línea]. 5 de noviembre de 2008. Disponible en: https://weblogs.asp.net/fredriknormen/using-web-services-in-a-3-tier-architecture

[3] Wikipedia. “Modelo-vista-controlador”, [en línea]. 17 de junio de 2017. Disponible en: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[4] W3Schools. “Ajax Introduction”, [en línea]. Disponible en: https://www.w3schools.com/xml/ajax_intro.asp

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

[6] creaTáctil S.L. “Descubre las Ventajas de miColegioApp”, [en línea]. Disponible en: http://micolegioapp.com/wordpress/

[7] Escolapp. “Cómo funciona”, [en línea]. Disponible en: http://escolapp.es/como-funciona/

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

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

[10] TokApp OnLine S.L. “Centros Educativos”, [en línea]. Disponible en: https://www.tokappschool.com/welcome/centros

[11] TokApp OnLine S.L. “Padres Alumnos”, [en línea]. Disponible en: https://www.tokappschool.com/welcome/padres

[12] TokApp OnLine S.L. “Profesores”, [en línea]. Disponible en: https://www.tokappschool.com/welcome/profesores

[13] Wikipedia. “GNU General Public License”, [en línea]. 17 de mayo de 2017. Disponible en: https://es.wikipedia.org/wiki/GNU_General_Public_License

[14] Wikipedia. “Licencia MIT”, [en línea]. 17 de junio de 2017. Disponible en: https://es.wikipedia.org/wiki/Licencia_MIT

[15] StarUML. “About StarUML”, [en línea]. Disponible en: http://staruml.sourceforge.net/v1/license.php

[16] Wikipedia. “Apache License”, [en línea]. 4 de junio de 2017. Disponible en: https://es.wikipedia.org/wiki/Apache_License

[17] Wikipedia. “Licencia PHP”, [en línea]. 29 de octubre de 2016. Disponible en: https://es.wikipedia.org/wiki/Licencia_PHP

[18] JetBrains. “TOOLBOX SUBSCRIPTION LICENSE AGREEMENT FOR EDUCATION”, [en línea]. 9 de octubre de 2016. Disponible en: https://www.jetbrains.com/student/license_educational.html

[19] Justinmind. “Terms of Use”, [en línea]. Disponible en: https://www.justinmind.com/terms

[20] “Ley Orgánica de Protección de Datos de Carácter Personal”, [en línea]. 13 de diciembre de 1999. Disponible en: https://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750

[21] Agencia Española de Protección de Datos. “Guía de Seguridad de Datos”, [en línea]. Disponible en: https://www.agpd.es/portalwebAGPD/canaldocumentacion/publicaciones/common/Guias/GUIA_SEGURIDAD_2010.pdf

[22] Wikipedia. “Lenguaje unificado de modelado”, [en línea]. 20 de junio de 2017. Disponible en: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

[23] Software Engineering Stack Exchange, “Storing files in the database”, [en línea], 31 de mayo de 2012. Disponible en: https://softwareengineering.stackexchange.com/questions/150669/is-it-a-bad-practice-to-store-large-files-10-mb-in-a-database

[24] “First Principles of Interaction Design”, [en línea], 5 de marzo de 2014. Disponible en: http://asktog.com/atc/principles-of-interaction-design/

[25] “Qué es Symfony”, [en línea]. Disponible en: http://symfony.es/pagina/que-es-symfony/

[26] “El patrón MVC”, [en línea]. Disponible en: https://librosweb.es/libro/symfony_1_2/capitulo_2/el_patron_mvc.html

[27] “Security”, [en línea]. Disponible en: https://symfony.com/doc/current/security.html

[28] Microsoft. “Patrones de Fabricación: Fábricas de Objetos”, [en línea]. Disponible en: https://msdn.microsoft.com/es-es/library/bb972258.aspx

[29] Armentano, Lucila. “Buenas prácticas para el Diseño de una API RESTful Pragmática”, [en línea]. 12 de febrero de 2017. Disponible en: https://elbauldelprogramador.com/buenas-practicas-para-el-diseno-de-una-api-restful-pragmatica/

[30] SensioLabs. “Twig Documentation”, [en línea]. Disponible en: https://twig.sensiolabs.org/doc/2.x/

[31] “Patrones de Diseño (XI): Patrones Estructurales – Facade”, [en línea]. Disponible en: http://programacion.net/articulo/patrones_de_diseno_xi_patrones_estructurales_facade_1014

[32] Doctrine Team. “Doctrine 2 ORM’s documentation”, [en línea]. Disponible en: http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/index.html

[33] “Symfony Documentation”, [en línea]. Disponible en: http://symfony.com/doc/current/index.html

[34] Weaver, Ryan; Pelham, Leanna. “Joyful Development with Symfony 3”, [en línea]. Disponible en: http://knpuniversity.com/screencast/symfony

[35] W3Schools. “JavaScript Tutorial”, [en línea]. Disponible en: https://www.w3schools.com/js/

[36] W3Schools. “jQuery API”, [en línea]. Disponible en: http://api.jquery.com/

[37] PHP Group. “Manual de PHP”, [en línea]. Disponible en: http://php.net/manual/es/

[38] Oracle Corporation. “MySQL 5.7 Reference Manual”, [en línea]. Disponible en: https://dev.mysql.com/doc/refman/5.7/en/

[39] Profesora Eugenia. “¿Cómo citar una Página Web?”, [en línea]. 25 de julio de 2006. Disponible en: http://profesoraeugenia.blogspot.com.es/2006/07/cmo-citar-una-pgina-web.html

[40] Potencier, Fabien; Weaver, Ryan; Eguiluz, Javier. “Buenas prácticas oficiales de Symfony”, [en línea]. Disponible en: https://librosweb.es/libro/buenas_practicas_symfony/

Anexo I: Manual de usuario

Introducción

La finalidad de este manual es explicar, de forma clara y sencilla las funcionalidades de la aplicación web desarrollada para la administración del centro. Por ello, realizaremos una breve explicación de cada funcionalidad de la aplicación web, las cuales estarán acompañadas por una captura de pantalla de la aplicación.

Inicio de sesión

Para poder acceder a la aplicación web, el centro debe haber solicitado cuentas de administrador para acceder a la aplicación.

Una vez hecho esto, cualquier administrador del centro podrá acceder a la aplicación utilizando una de las cuentas proporcionadas. Si así lo desea, podrá modificar sus credenciales posteriormente desde la misma aplicación.

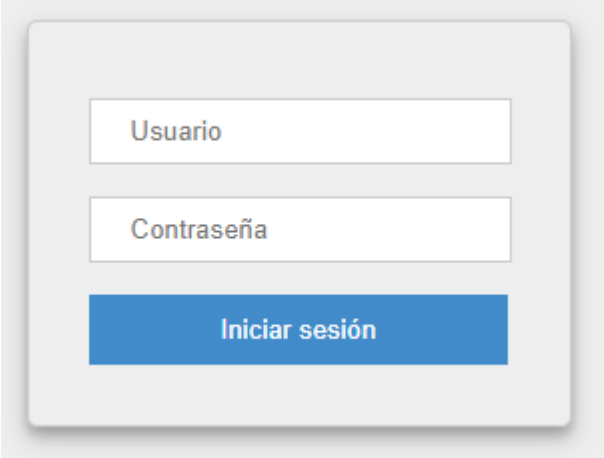

Ilustración 16: Formulario de inicio de sesión

Estructura de las páginas

El menú de navegación (remarcado en azul oscuro en la *Ilustración 17*) está dividido en distintas secciones:

- Centro: se encuentran las páginas relacionadas con la gestión del centro (cursos, alumnos, padres…).
- Mensajería: se encuentran las páginas relacionadas con la comunicación entre el centro y los padres (envío de mensajes y visualización de respuestas).

El menú de acciones (remarcado en azul claro en la *Ilustración 17*) contará con distintos botones que permitirán realizar acciones específicas.

En el encabezado de cada página se permitirá cerrar la sesión actual.

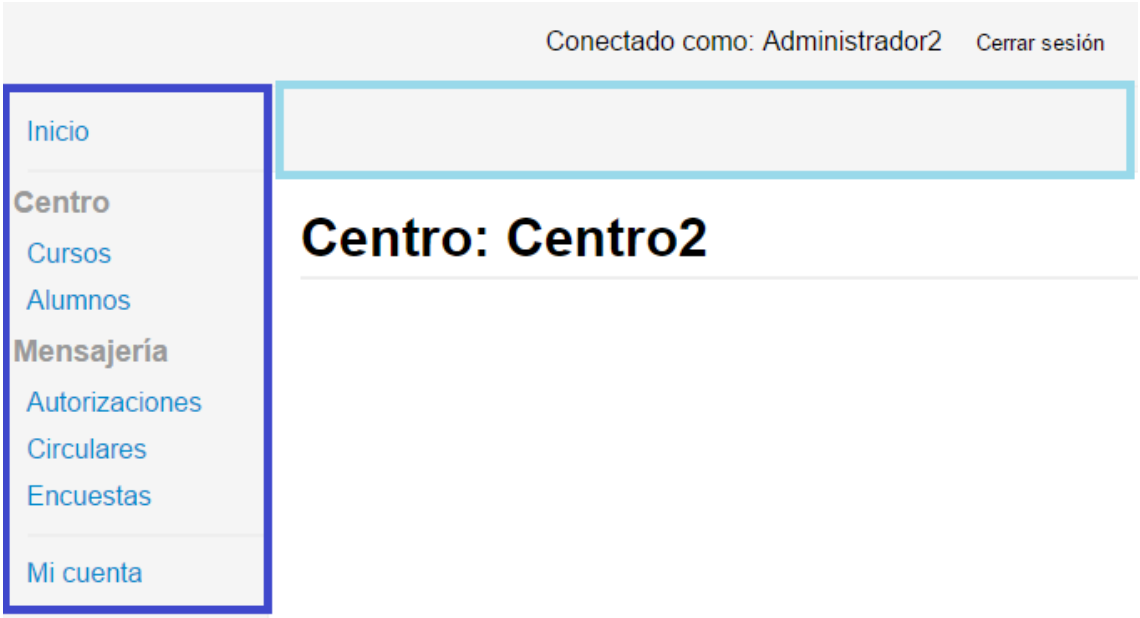

Ilustración 17: Estructura de la página

Cursos

En esta página se podrá:

- Visualizar un listado de los cursos existentes.
- Crear un curso nuevo.
- Importar los datos del centro automáticamente, seleccionando el archivo CSV correspondiente y confirmando la acción.
- Haciendo *click* en el botón "Ver" de un curso se mostrará la página de dicho curso.

Ilustración 18: Página Cursos

Al hacer *click* en el botón “Nuevo curso”, se mostrará un diálogo en el que podremos elegir el nombre del curso y podremos añadirle alumnos ya registrados en el centro. Podemos filtrar los alumnos introduciendo su nombre en el segundo campo.

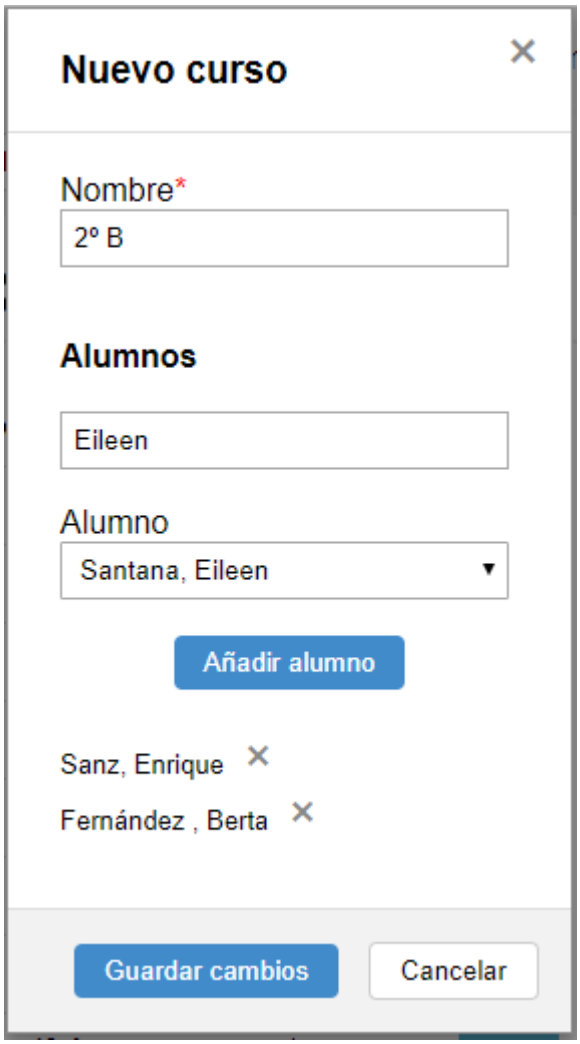

Ilustración 19: Nuevo curso

Alumnos

En esta página podremos:

- Registrar nuevos alumnos en el centro.
- Visualizar los alumnos ya registrados.
- Filtrar los alumnos mostrados según el curso en el que se encuentren y su nombre.
- En el botón “Ver” de cada alumno, podremos acceder al perfil de dicho alumno.

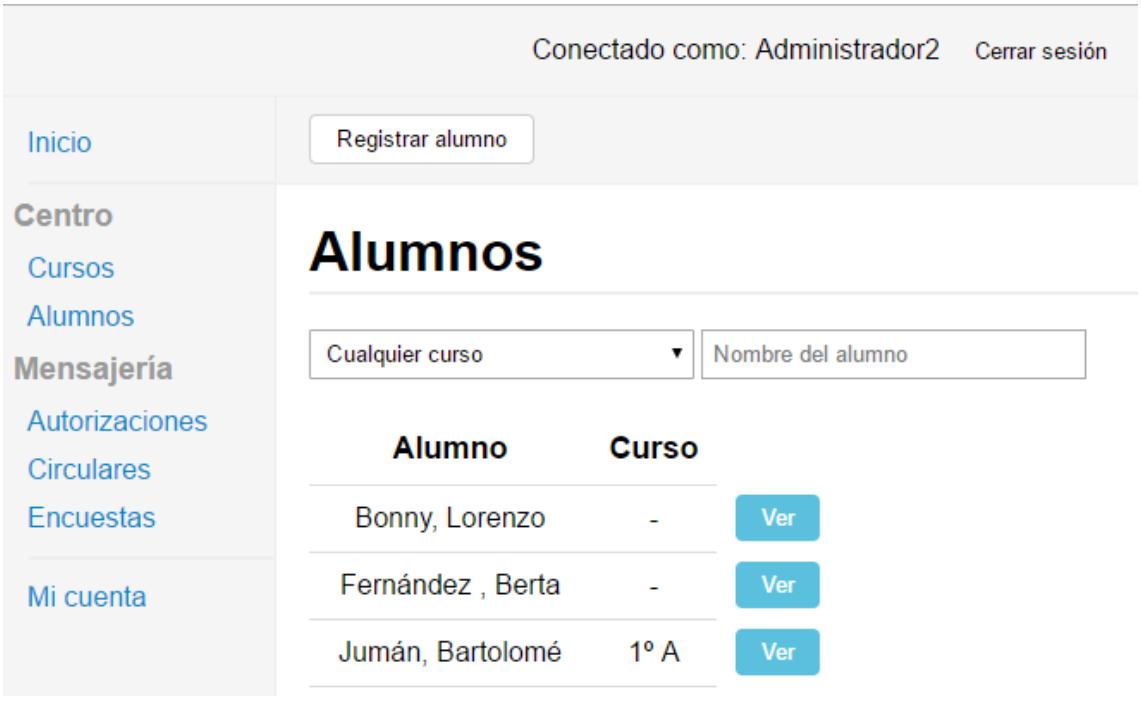

Ilustración 20: Página Alumnos

Al hacer *click* en el botón "Registrar alumno" se mostrará un formulario en el que tendremos que introducir su nombre, sus apellidos y el curso en el que se encuentra. Por último, podremos asociar padres al alumno, introduciendo el número de teléfono del padre. Se permitirá añadir al padre, únicamente, si se ha registrado en el sistema y el centro se encuentra entre sus escogidos.

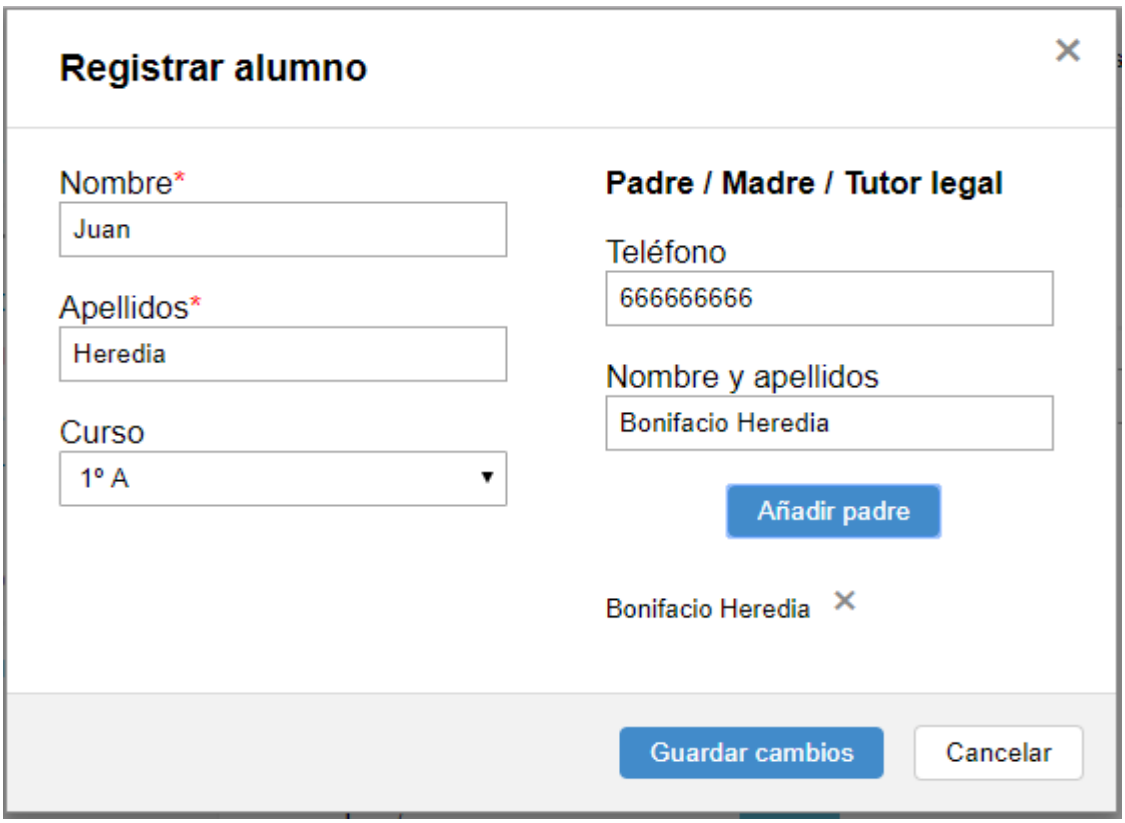

Ilustración 21: Registrar alumno

Perfil del alumno

En esta página podremos:

- Ver todos los datos del alumno.
- Ver los padres asociados al alumno.
- Eliminar al alumno del sistema.
- Asociarle padres al alumno.
- Desasociarle un padre al alumno.

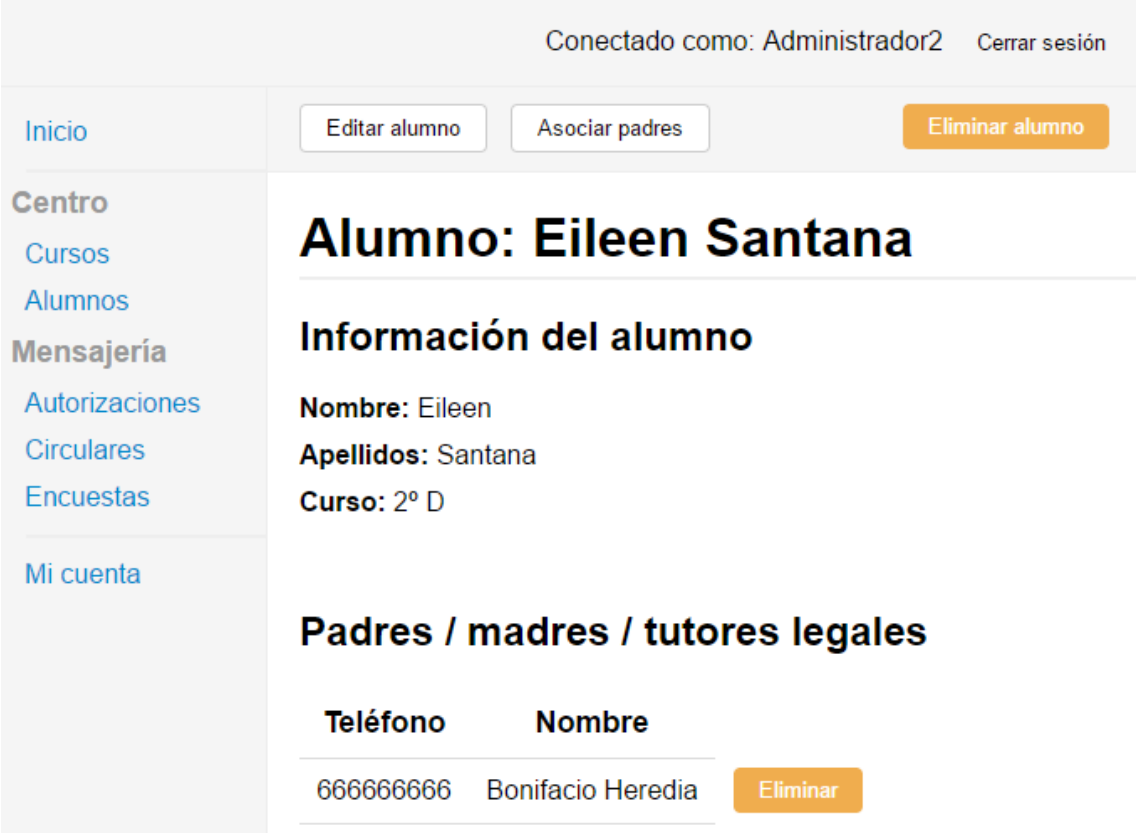

Ilustración 22: Página Alumno

Al hacer *click* en el botón “Editar Alumno” se mostrará un formulario en el que podremos editar los datos del alumno (nombre, apellidos y curso).

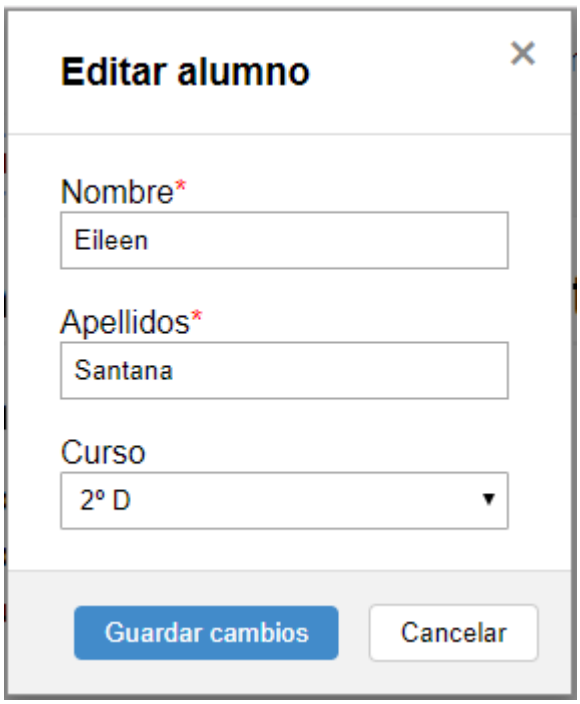

Ilustración 23: Editar alumno

Al hacer *click* en el botón “Asociar padres” se mostrará un formulario en el que podremos asociar nuevos padres al alumno, introduciendo su número de teléfono.

Al igual que en el registro del alumno, solamente se podrá asociar al padre si se ha registrado en el sistema y el centro se encuentra entre sus escogidos.

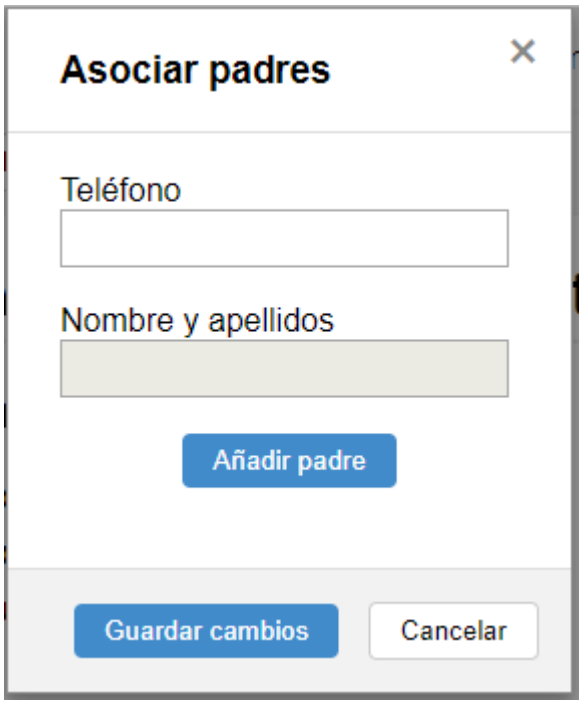

Ilustración 24: Asociar padres a un alumno

Autorizaciones

En esta página podremos:

- Enviar autorizaciones a los padres de los alumnos.
- Ver las autorizaciones enviadas.
- Editar la fecha límite para responder a la autorización.
- Filtrar las autorizaciones según su asunto, el mes en el que fueron enviadas o su estado (en curso o finalizadas, según su fecha límite). Las autorizaciones finalizadas mostrarán la fecha límite en color rojo.

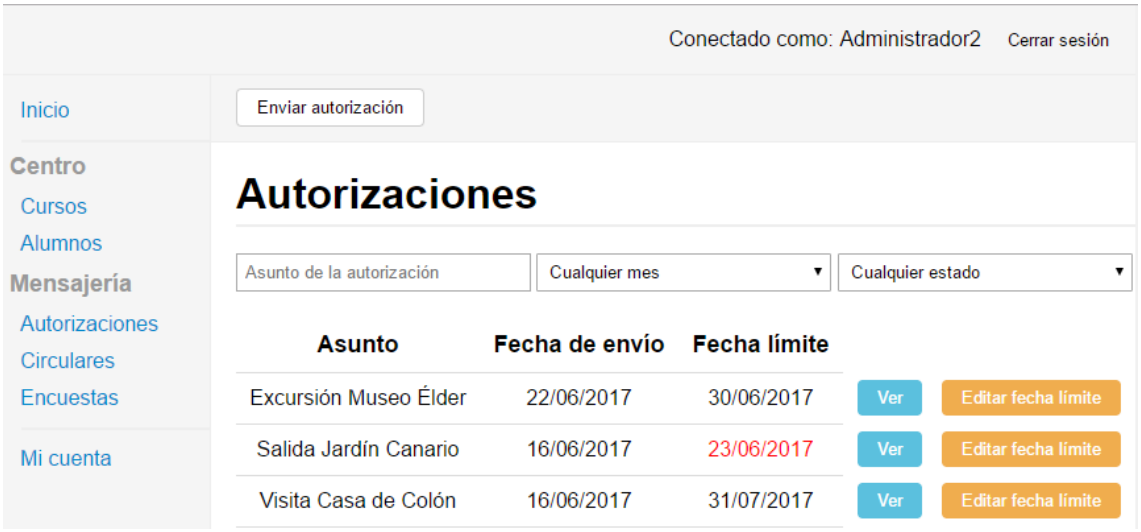

Ilustración 25: Página Autorizaciones

Al hacer *click* en el botón "Editar fecha límite" de una autorización, se mostrará un formulario que permitirá modificar la fecha límite para responder dicha autorización.

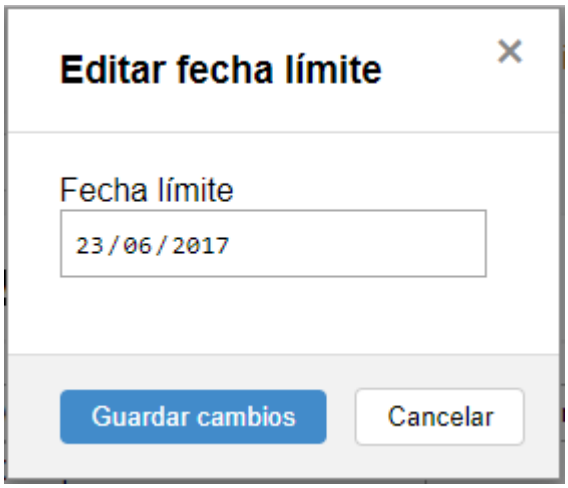

Ilustración 26: Editar fecha límite

Al hacer *click* en el botón "Ver" de una autorización, se mostrará un formulario con la información correspondiente a la autorización (asunto, fecha de envío y fecha límite, archivos adjuntos, mensaje, listado de alumnos autorizados y listado de alumnos no autorizados).

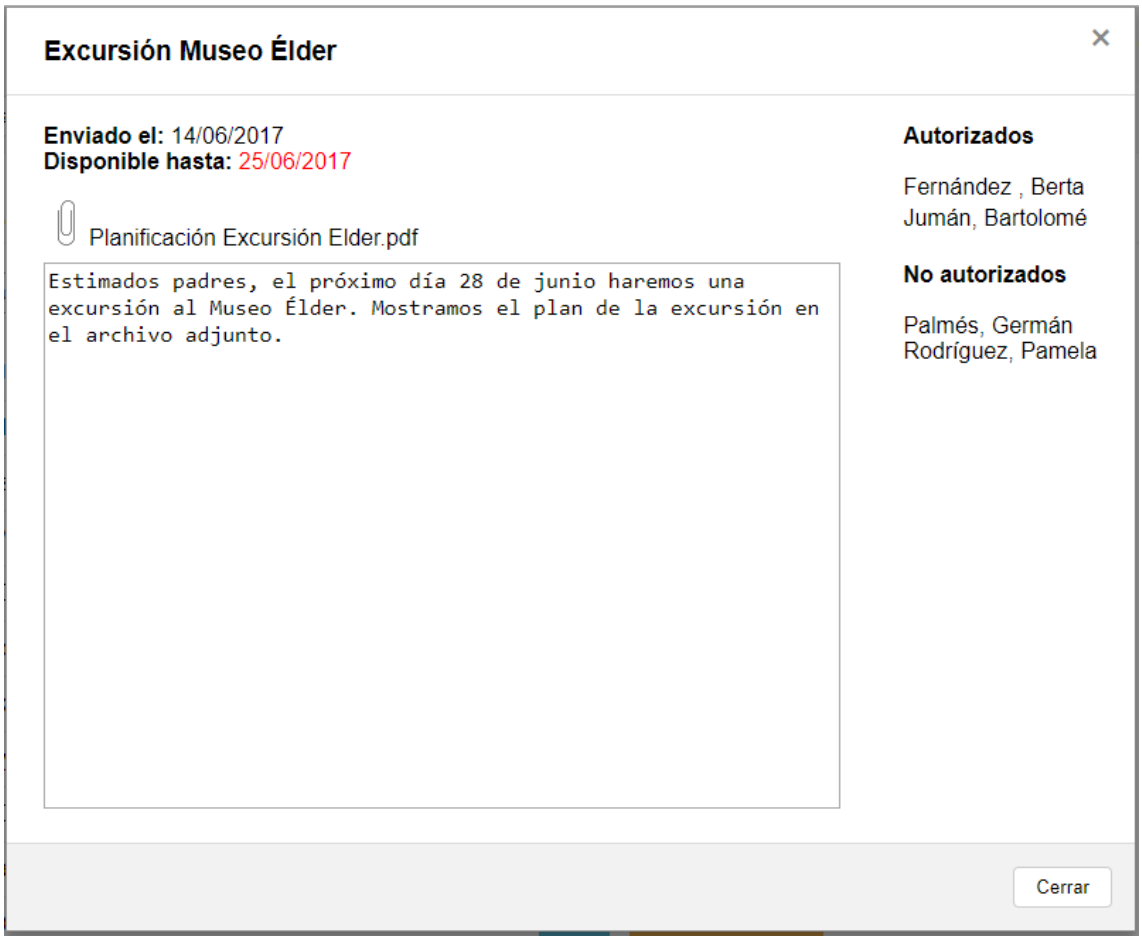

Ilustración 27: Ver autorización

Al hacer *click* en el botón "Enviar autorización", se mostrará un formulario en el que introduciremos el asunto de la autorización, su fecha límite para ser respondida, el mensaje de la autorización y, si es necesario, podremos adjuntarle un archivo en formato PDF.

Además, podremos seleccionar individualmente o por grupos, los alumnos que queremos que sean autorizados para realizar la actividad.

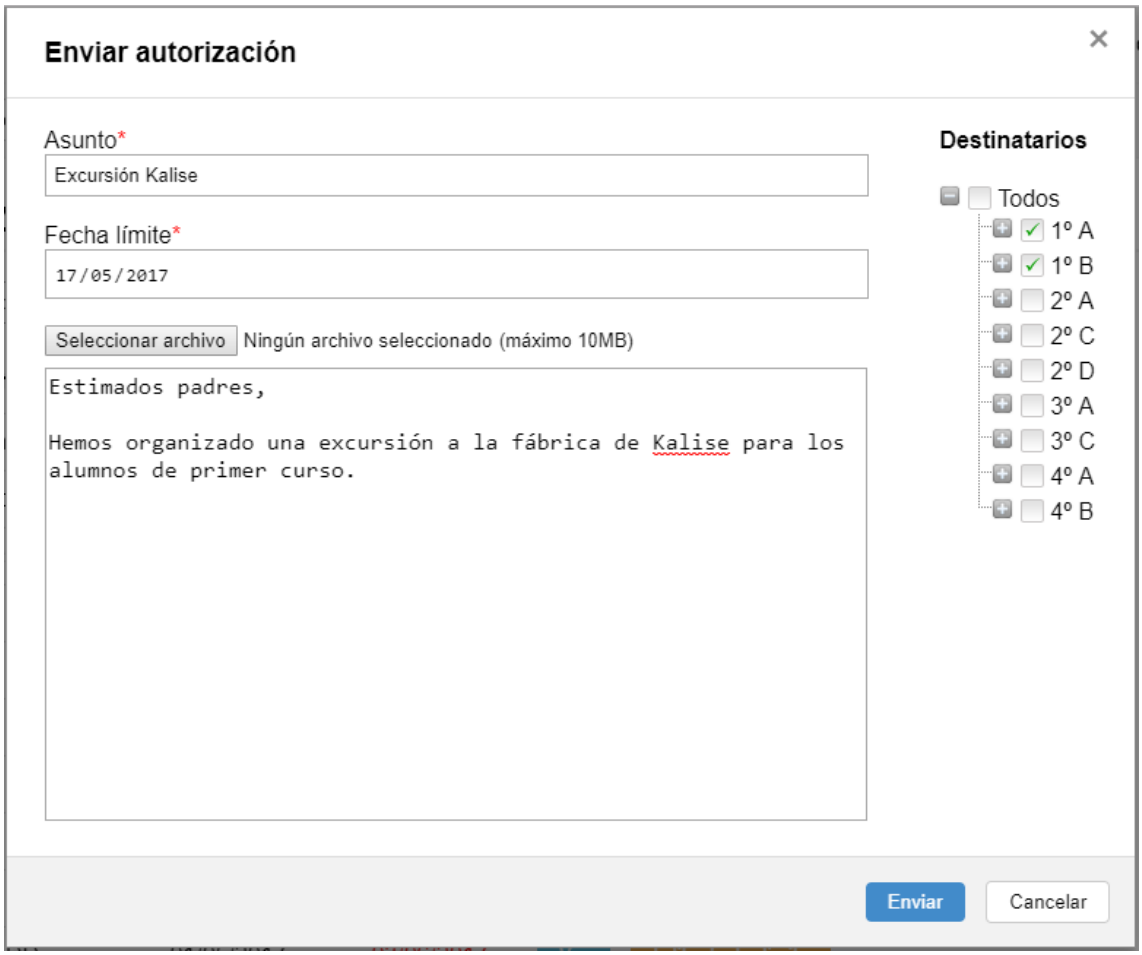

Ilustración 28: Enviar autorización

Circulares

En esta página podremos:

- Enviar circulares a los padres de los alumnos.
- Ver las circulares enviadas.
- Filtrar las circulares según su asunto o el mes en el que fueron enviadas.

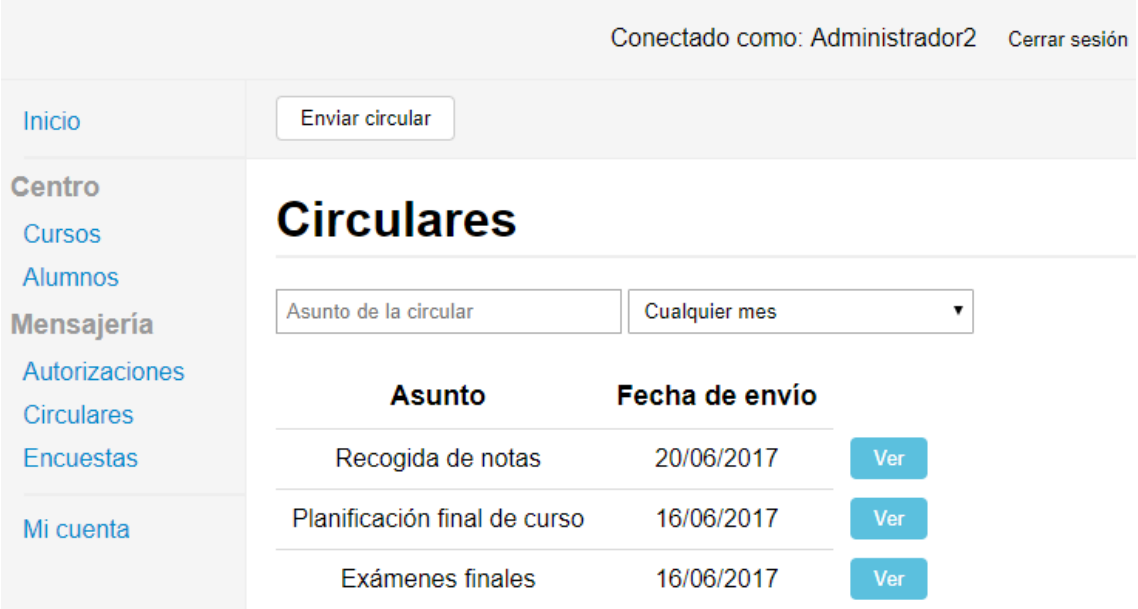

Ilustración 29: Página Circulares

Al hacer *click* en el botón "Ver" de una circular, se mostrará un formulario con la información correspondiente a la circular (asunto, fecha de envío, archivos adjuntos y mensaje).

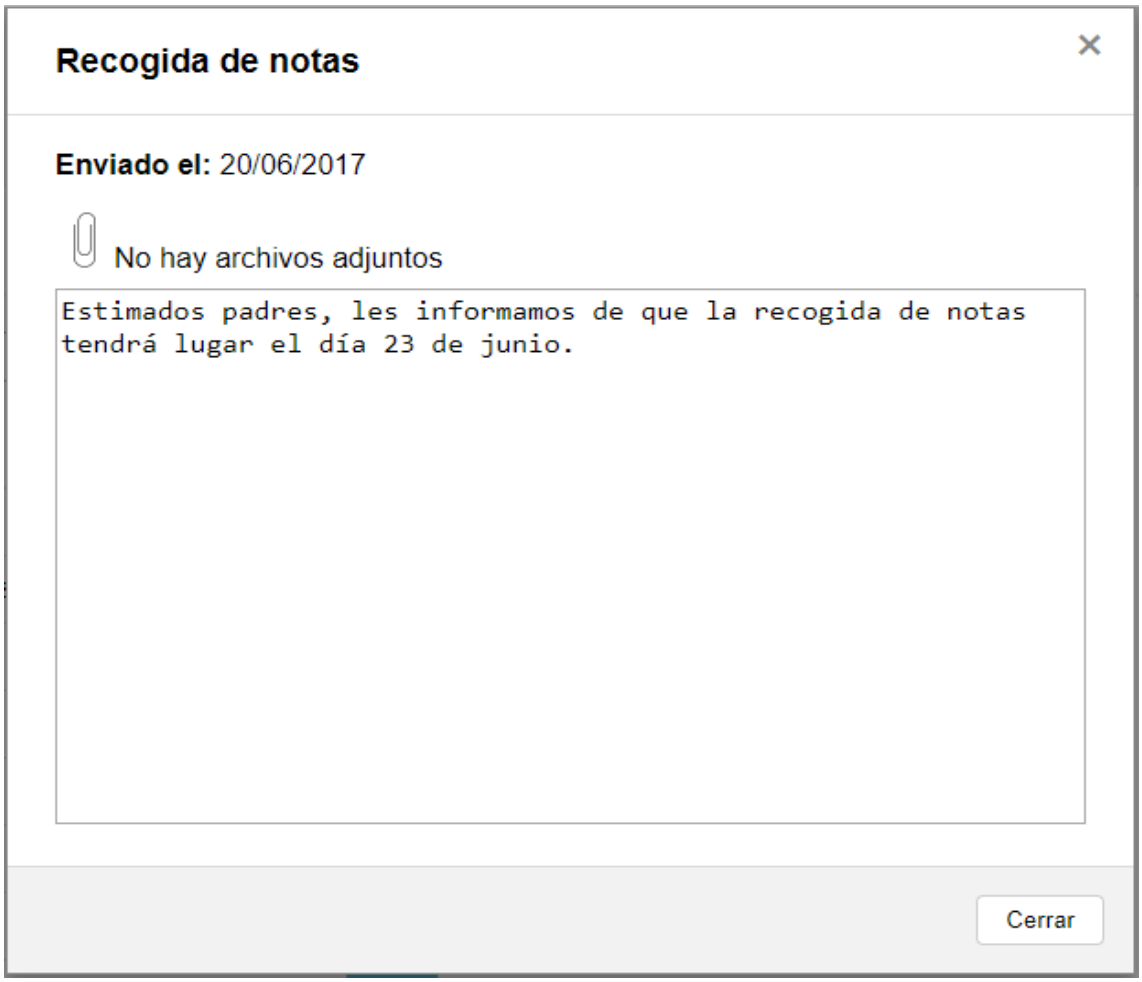

Ilustración 30: Ver circular

Al hacer *click* en el botón “Enviar circular”, se mostrará un formulario en el que introduciremos el asunto de la circular, el mensaje de la circular y, si es necesario, podremos adjuntarle un archivo en formato PDF.

Además, podremos seleccionar individualmente o por grupos, los alumnos cuyos padres queremos que reciban la circular.

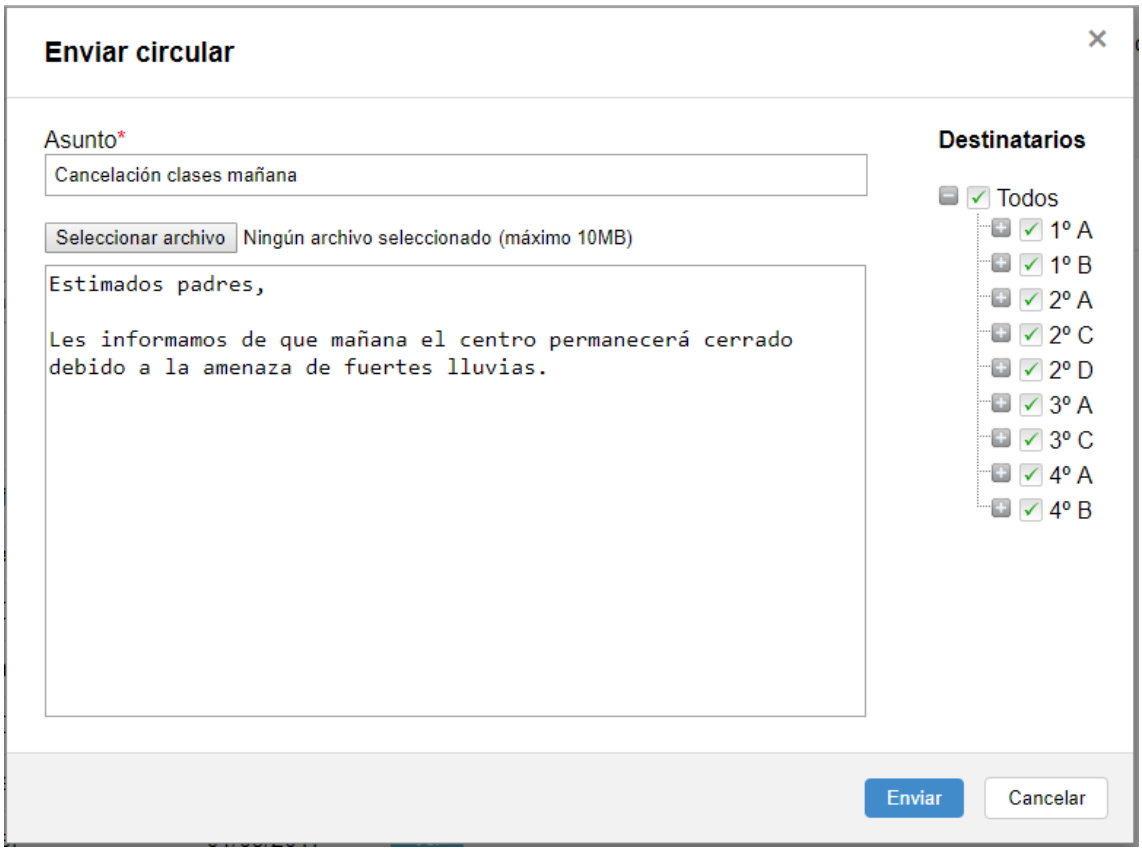

Ilustración 31: Enviar circular

Encuestas

En esta página podremos

- Enviar encuestas a los padres de los alumnos.
- Ver las encuestas enviadas y editar la fecha límite para responder a la encuesta.
- Filtrar las encuestas según el asunto, el mes en el que fueron enviadas o su estado (en curso o finalizadas, según su fecha límite). Las encuestas finalizadas mostrarán la fecha límite en color rojo.

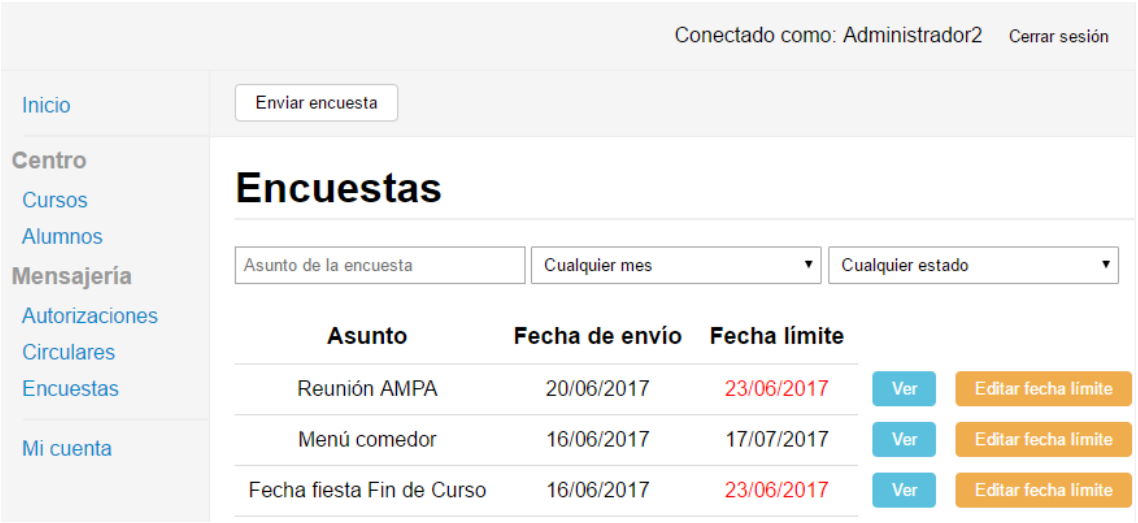

Ilustración 32: Página Encuestas

Al hacer *click* en el botón "Editar fecha límite" de una encuesta, se mostrará un formulario que permitirá modificar la fecha límite para responder dicha encuesta.

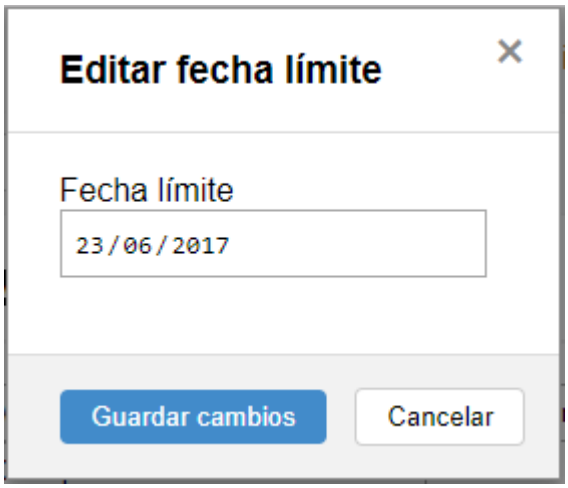

Ilustración 33: Editar fecha límite

Al hacer *click* en el botón “Ver” de una encuesta, se mostrará un formulario con la información correspondiente a la encuesta (asunto, fechas de envío y límite, archivos adjuntos, mensaje, y resultados de la encuesta). La opción que haya recibido más votos se destacará en color verde.

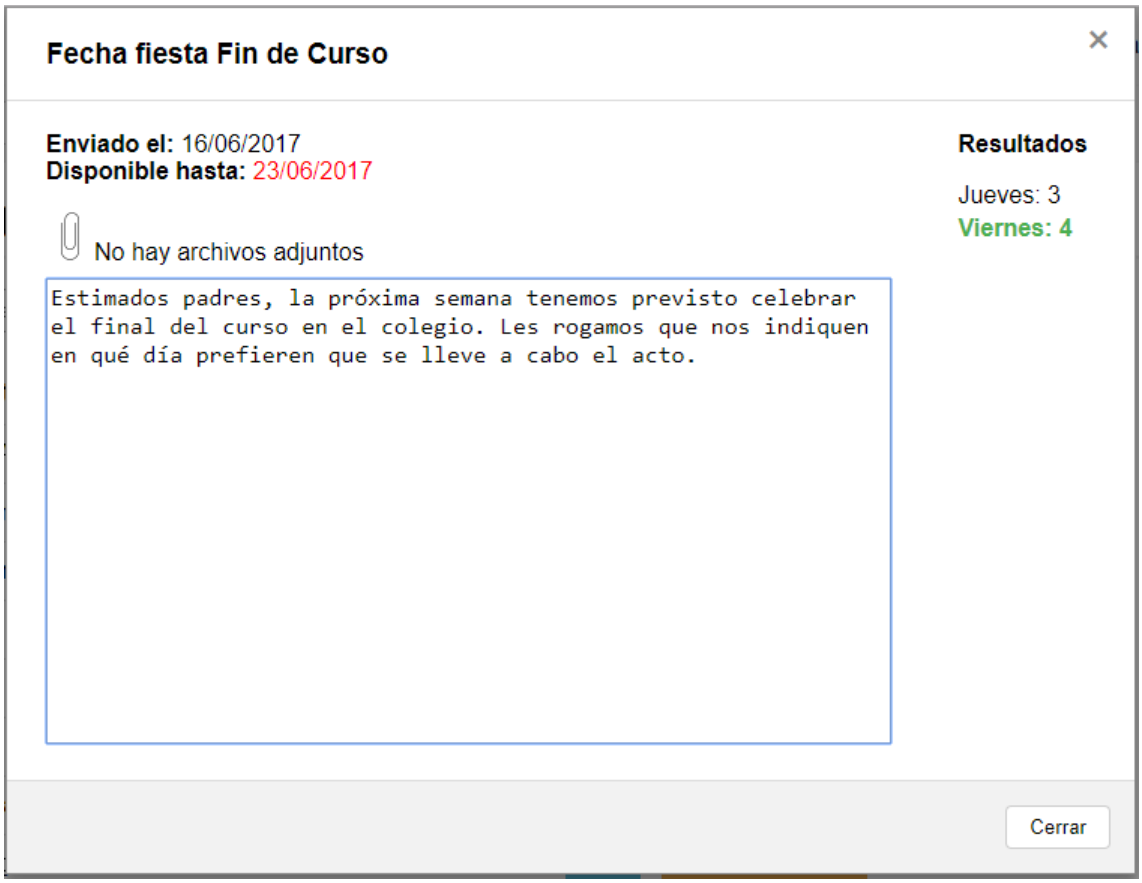

Ilustración 34: Ver encuesta

Al hacer *click* en el botón “Enviar encuesta”, se mostrará un formulario en el que introduciremos el asunto de la encuesta, su fecha límite para ser respondida, el mensaje de la encuesta y, si es necesario, podremos adjuntarle un archivo en formato PDF.

Además, podremos establecer si la encuesta permitirá selección de múltiples opciones a la hora de ser respondida.

También, podemos añadir distintas opciones posibles a la encuesta. Por último, podremos seleccionar individualmente o por grupos, los alumnos cuyos padres queremos que respondan la encuesta.

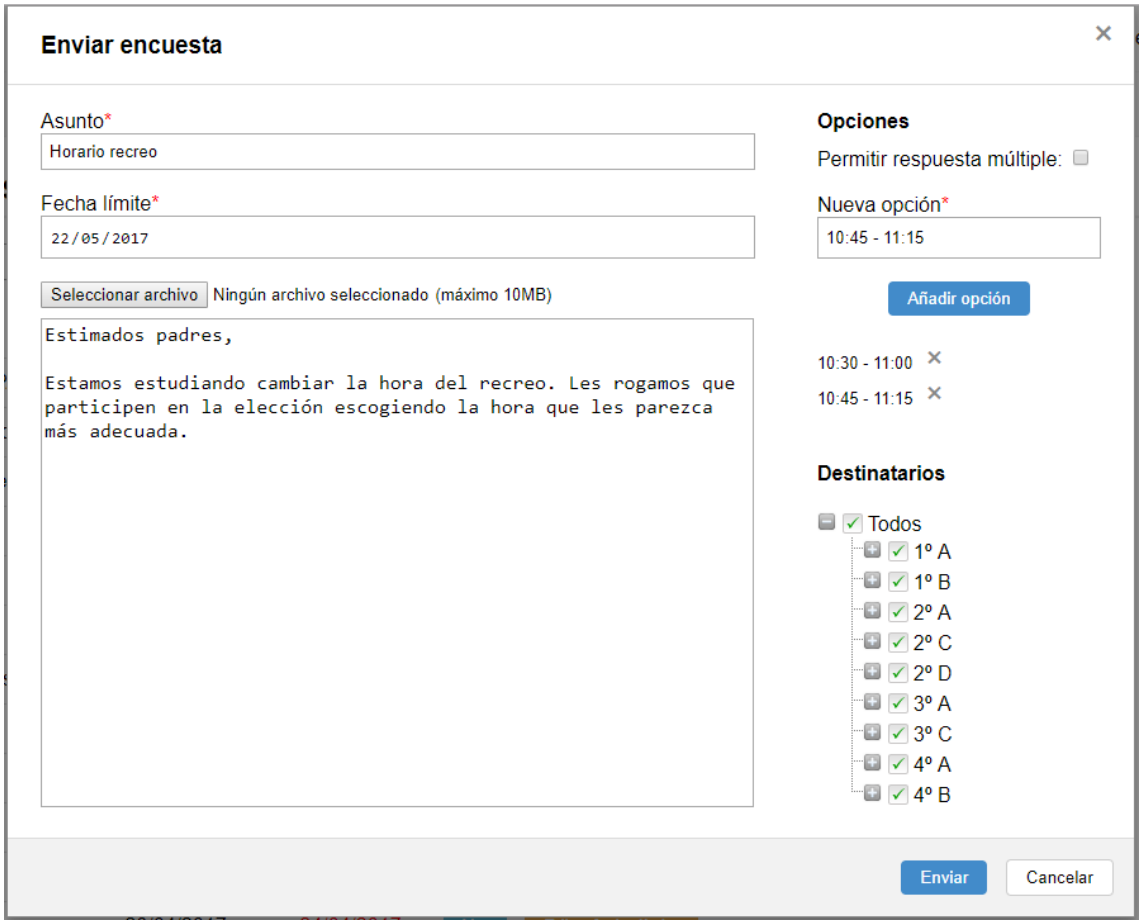

Ilustración 35: Enviar encuesta

Mi cuenta

En esta página, el administrador del centro podrá:

- Ver información sobre su cuenta, como su nombre y su usuario.
- Editar su nombre y sus credenciales de acceso.

Ilustración 36: Página Mi cuenta

Al hacer *click* en el botón "Editar cuenta", se mostrará un formulario en el cual se podrán establecer un nuevo nombre, usuario o contraseña. Para poder realizar cualquier cambio, es necesario introducir la contraseña actual por motivos de seguridad.

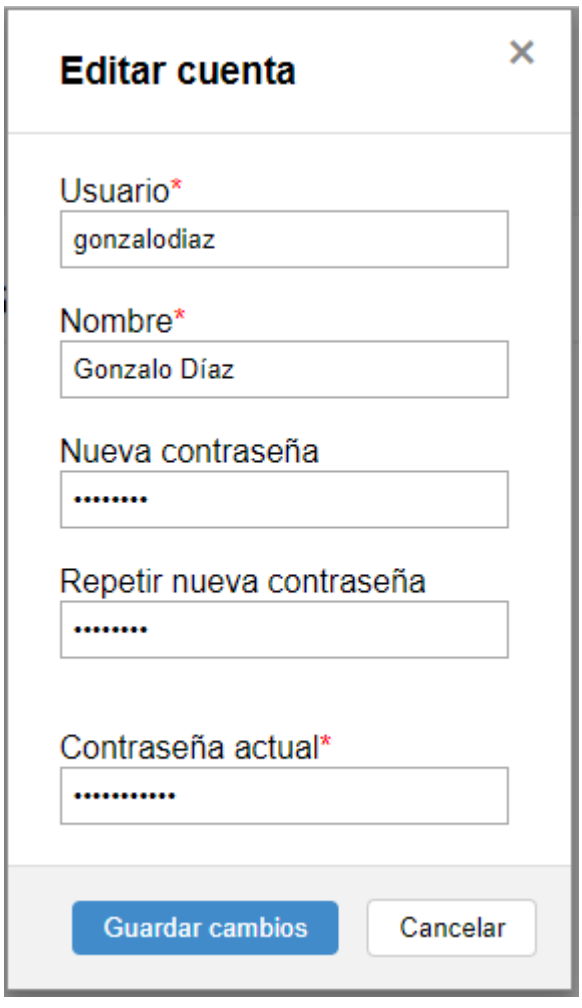

Ilustración 37: Editar cuenta

Anexo II: Manual de instalación

A continuación, explicaremos los requisitos necesarios para poder desplegar la aplicación en un servidor:

- Servidor Apache 2.0 o superior.
- MySQL 5.0 o superior.
- Intérprete PHP 7.1 o superior.

En nuestro caso, hemos utilizado XAMPP, una herramienta multiplataforma (Windows, Linux, MacOS) que nos permite desplegar aplicaciones web desarrolladas en PHP de manera rápida y sencilla. Esta herramienta contiene todos los requisitos mencionados anteriormente, por lo que, tras instalarla, ya tendremos todo lo necesario.

Para desplegar la aplicación en el servidor web Apache, es tan sencillo como ubicar la carpeta del proyecto bajo el directorio *xampp/htdocs*.

Además, creamos el directorio *xampp/htdocs/Hermerest_attachments* para poder guardar todos los archivos adjuntos que se suben en la aplicación.

Una vez hecho esto, mediante la herramienta PHPMyAdmin, a la cual podemos acceder desde el navegador (accediendo a *localhost*), creamos una base de datos y un usuario que tenga acceso a ella y, en el proyecto web, establecemos los parámetros *database_name*, *database_user* y *database_password* en el fichero */app/config/parameters.yml*.

Para crear automáticamente las tablas de la base de datos, abrimos la consola de sistema en el directorio raíz del proyecto y ejecutamos el siguiente comando:

```
php bin/console doctrine:schema:update --force
```

Para poder ejecutar el comando, si utilizamos un sistema operativo Windows, debemos añadir la ruta al intérprete de PHP, situado en *xampp/php*, en la variable de sistema *PATH*.

Otra alternativa para crear la base de datos (con sus tablas y su usuario) es ejecutar el script presentado en el *ANEXO III* en la herramienta PHPMyAdmin.

Por último, para que la aplicación funcione correctamente, nos ubicaremos en el directorio raíz del proyecto y ejecutaremos el siguiente comando desde la consola de comandos del sistema para instalar la herramienta Composer (gestor de paquetes) y descargar todas las dependencias del proyecto:

```
php composer.phar install
```

Anexo III: Script para la base de datos

```
DROP DATABASE IF EXISTS hermerest;
CREATE DATABASE hermerest
      DEFAULT CHARACTER SET utf8
      DEFAULT COLLATE utf8_general_ci;

CREATE USER IF NOT EXISTS hermerest;
GRANT ALL ON hermerest.* to 'hermerest'@'localhost' IDENTIFIED BY
'hermerest';

CREATE TABLE centre(
      id int AUTO_INCREMENT,
      name VARCHAR(255) NOT NULL,
      PRIMARY KEY(id)
);

CREATE TABLE administrator(
      id int AUTO_INCREMENT,
      user VARCHAR(255) NOT NULL UNIQUE,
      password VARCHAR(32) NOT NULL,
      name VARCHAR(255) NOT NULL,
      centre int NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(centre) REFERENCES centre(id) ON DELETE CASCADE
);

CREATE TABLE class(
      id int AUTO_INCREMENT,
      name VARCHAR(255) NOT NULL UNIQUE,
      centre int NOT NULL,
      PRIMARY KEY(id),
      UNIQUE (name,centre),
      FOREIGN KEY(centre) REFERENCES centre(id) ON DELETE CASCADE
);

CREATE TABLE student(
      id int AUTO_INCREMENT,
      name VARCHAR(255) NOT NULL,
      surname VARCHAR(255) NOT NULL,
      class int,
      centre int NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(class) REFERENCES class(id) ON DELETE SET NULL
      FOREIGN KEY(centre) REFERENCES centre(id) ON DELETE CASCADE
);

CREATE TABLE parent(
      id int AUTO_INCREMENT,
      name VARCHAR(255) NOT NULL,
      telephone VARCHAR(255) NOT NULL UNIQUE,
```

```
      PRIMARY KEY(id)
);

CREATE TABLE student_parent(
      student int NOT NULL,
      parent int NOT NULL,
      PRIMARY KEY(student, parent),
      FOREIGN KEY(student) REFERENCES student(id) ON DELETE CASCADE,
      FOREIGN KEY(parent) REFERENCES parent(id) ON DELETE CASCADE
);

CREATE TABLE centre_parent(
      centre int NOT NULL,
      parent int NOT NULL,
      PRIMARY KEY(centre, parent),
      FOREIGN KEY(centre) REFERENCES centre(id) ON DELETE CASCADE,
      FOREIGN KEY(parent) REFERENCES parent(id) ON DELETE CASCADE
);

CREATE TABLE message(
      id int AUTO_INCREMENT,
      subject VARCHAR(255) NOT NULL,
      message TEXT NOT NULL,
      sendingDate TIMESTAMP NOT NULL,
      centre int NOT NULL,
      type VARCHAR(255) NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(centre) REFERENCES centre(id) ON DELETE CASCADE
);

CREATE TABLE authorization(
      id int,
      limitDate TIMESTAMP NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(id) REFERENCES message(id) ON DELETE CASCADE
);

CREATE TABLE circular(
      id int,
      PRIMARY KEY(id),
      FOREIGN KEY(id) REFERENCES message(id) ON DELETE CASCADE
);

CREATE TABLE poll(
      id int,
      limitDate TIMESTAMP NOT NULL,
      multipleChoice TINYINT(1) NOT NULL;
      PRIMARY KEY(id),
      FOREIGN KEY(id) REFERENCES message(id) ON DELETE CASCADE
);

CREATE TABLE pollOption(
      id int AUTO_INCREMENT,
```

```
      text VARCHAR(255) NOT NULL,
      poll int NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(poll) REFERENCES poll(id) ON DELETE CASCADE
);

CREATE TABLE message_student(
      message int NOT NULL,
      student int NOT NULL,
      PRIMARY KEY(message, student),
      FOREIGN KEY(student) REFERENCES student(id) ON DELETE CASCADE,
      FOREIGN KEY(message) REFERENCES message(id) ON DELETE CASCADE
);

CREATE TABLE pollReply(
      id int AUTO_INCREMENT,
      parent int,
      pollOption int,
      PRIMARY KEY(id),
      UNIQUE(parent, pollOption),
      FOREIGN KEY(parent) REFERENCES parent(id) ON DELETE CASCADE,
      FOREIGN  KEY(pollOption)  REFERENCES  pollOption(id)  ON  DELETE
CASCADE
);

CREATE TABLE authorizationReply(
      id int AUTO_INCREMENT,
      authorization int,
      parent int,
      student int,
      authorized boolean NOT NULL,
      PRIMARY KEY(id),
      UNIQUE(parent, authorization, student),
      FOREIGN KEY(authorization) REFERENCES authorization(id) ON DELETE
CASCADE
      FOREIGN KEY(parent) REFERENCES parent(id) ON DELETE CASCADE,
      FOREIGN KEY(student) REFERENCES student(id) ON DELETE CASCADE,
);

CREATE TABLE attachment(
      id int AUTO_INCREMENT,
      name VARCHAR(255) NOT NULL,
      message int NOT NULL,
      PRIMARY KEY(id),
      FOREIGN KEY(message) REFERENCES message(id) ON DELETE CASCADE
);
```