UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Desarrollo de módulos de tutorías y gestión de pagos para Dinantia

Facultad de Informática de Barcelona – Q2 Curso 2020-2021

Especialización en Ingeniería del Software

Autor: Marc Simó Guzmán

Director: Cristian Cisa

Ponente: Ernest Teniente López

Fecha: 24/06/2021

Resumen

Dinantia es una aplicación orientada a facilitar, mejorar y concentrar la comunicación y la gestión de escuelas y centros educativos. Los clientes de la aplicación han solicitado la implementación de funcionalidades para la gestión de tutorías y una ampliación de las funcionalidades de pagos. Debido a esta necesidad surge este proyecto, en el que se desarrollará un módulo de gestión y reserva de tutorías y se ampliará el módulo de pagos actual, ampliando sus funcionalidades y utilidad.

Resum

Dinantia és una aplicació orientada a facilitar, millorar i concentrar la comunicació i gestió d'escoles i centres educatius. Els clients de l'aplicació han sol·licitat la implementació de funcionalitats per la gestió de tutories i una ampliació de les funcionalitats de pagaments. A causa d'aquesta necessitat sorgeix aquest projecte, en el qual es desenvoluparà un mòdul de gestió i reserva de tutories i s'ampliarà el mòdul de pagaments actual, ampliant les seves funcionalitats i utilitat.

Abstract

Dinantia is an application aimed at facilitating, improving and concentrating the communication and administration of schools and educational centers. Several clients have requested the implementation of functionalities for the management of tutor meetings and an extension of the payment features. This project arises due to this need, and it will comprise the development of the tutor meetings management and reservation module and the expansion of the current payment module, improving its functionalities and utility.

Índice general

Índice de tablas

Índice de figuras

1 Introducción

Este proyecto es un Trabajo de Fin de Grado en Ingeniería Informática con especialización en Ingeniería del Software en la Facultad de Informática de Barcelona (FIB) perteneciente a la Universidad Politécnica de Cataluña (UPC). Un proyecto cuya finalidad es el desarrollo de dos módulos para una aplicación web y móvil orientada a facilitar, mejorar y concentrar la comunicación y la gestión de escuelas y centros educativos. Estos módulos serán un módulo de gestión de tutorías y un módulo de pagos.

1.1 Contextualización

Desde los primeros centros educativos en la Antigua Grecia al sistema educativo actual, el mundo de la educación ha ido evolucionando y adaptándose, sufriendo una gran cantidad de cambios y convirtiéndose en el sistema tan extenso y organizado que conocemos actualmente, el cual requiere de una compleja gestión.

Por otro lado, la tecnología, que ha existido junto al ser humano desde su inicio, también ha ido evolucionando y ganando importancia, llegando al gran impacto que tiene actualmente en nuestras vidas. Desde la aparición de la imprenta, que supuso una gran revolución, que permitió la rápida difusión de técnicas y conocimientos, y conllevó un incremento del acervo cultural, a los smartphones y ordenadores, que han cambiado por completo la forma en que vivimos y se han ido introduciendo en nuestras vidas de manera exponencial, hasta el punto en que vivir sin un smartphone nos resulta inimaginable.

Con el paso del tiempo los centros educativos se han ido adaptando a estos avances tecnológicos, y prácticamente todos los centros tienen impresoras y están en proceso de digitalizarse, pero a pesar de que algunos centros ya empiezan a utilizar plataformas digitales para la gestión de ciertas tareas, su uso suele estar reducido a ciertos aspectos muy concretos, principalmente subida y descarga de archivos y documentación, sin sacarle partido a la gran cantidad de avances y ventajas que ofrece la tecnología y que podría facilitar enormemente la gestión de prácticamente todos los aspectos del centro.

1.1.1 ¿Qué es Dinantia?

Dinantia es un *software* desarrollado por la empresa Dinantia Mobile SL orientado a escuelas y centros educativos, que tiene como objetivo facilitar y concentrar la comunicación entre el centro y los padres y alumnos en la aplicación, y facilitar la gestión del centro, incluyendo una serie de funcionalidades que facilitan la organización. Los principales clientes de la aplicación son instituciones educativas (colegios, academias, etc.), y tienen clientes tanto en España como en América Latina.

Este *software* está compuesto por una aplicación web y una aplicación móvil, y las diversas funcionalidades de la aplicación se organizan en módulos y pueden ser gestionados de forma independiente por las escuelas, ejemplos de estos módulos serían el módulo de mensajes, el módulo de control de asistencias, el de horario escolar, etc. Todos estos módulos están disponibles desde la aplicación web, mientras que desde la aplicación móvil solo están disponibles aquellos que tiene sentido poder acceder en todo momento, como por ejemplo el módulo de mensajes.

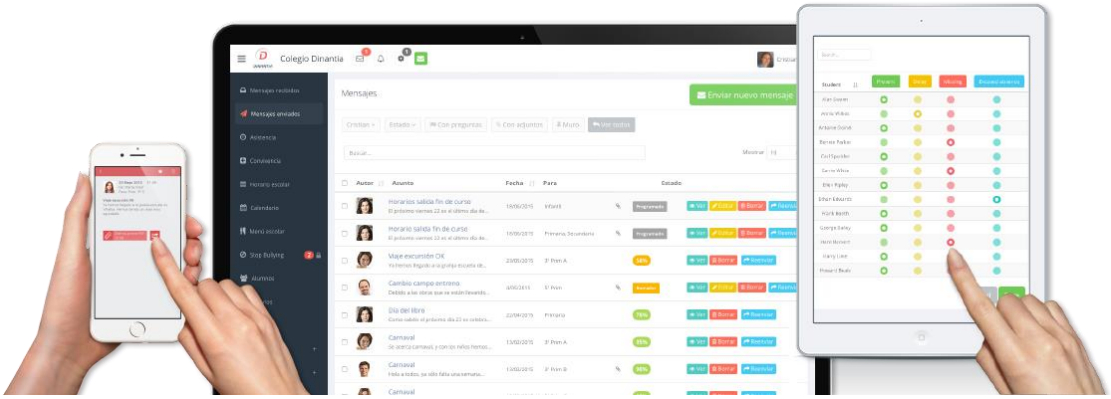


Figura 1: Aplicación Dinantia vista desde diferentes dispositivos. Fuente: Dinantia [1]

Figura 2: Funcionalidades principales de Dinantia. Fuente: Dinantia [1]

1.2 Motivación

La aplicación Dinantia es una aplicación que sigue desarrollando y aumentando sus funcionalidades semestre a semestre para satisfacer las demandas de sus clientes, ampliar su oferta y constituirse como una aplicación más completa. Por lo tanto, surgen constantemente nuevas ideas y módulos que implementar en la aplicación para su crecimiento y mejora, los módulos incluidos en este proyecto son un claro ejemplo de ello.

En los últimos semestres algunos clientes habían solicitado la implementación de un módulo para la reserva y gestión de tutorías, ya que el hecho de tener que notificar a los padres los horarios de las tutorías de cada maestro y gestionar las reservas de la forma tradicional suponía un esfuerzo significativo por su parte, y al ver como la aplicación había revolucionado y facilitado la forma de gestionar la comunicación, los horarios y los cursos entre otras cosas, pensaron que este proceso podía ser optimizado también, por lo que sugirieron este módulo y llegaron a un acuerdo con la empresa para su desarrollo.

Por otro lado, en cuanto al módulo de pagos, la aplicación tiene implementado un módulo que permite solicitar y realizar pagos a través de ella mediante una plataforma TPV, y consultar que alumnos han pagado y cuáles faltan por pagar, pero ahí acaba la cosa, este es un módulo funcional que cumple lo esperado, pero que podría ser mejorado para facilitar la gestión económica y contable por parte de los centros, esta fue la idea que supuso la adición de este módulo al proyecto para su desarrollo y crecimiento.

1.2.1 Factor económico

Al principio, uno de los principales inconvenientes para la adaptación del sector educativo a las nuevas tecnologías ha sido el precio, cuando surgen, todas las tecnologías son caras y suelen pasar por un período de desarrollo y adaptación previo a la democratización y bajada de precios que permite su acceso a una gran cantidad de la población.

Pero este ya no es realmente un problema, puesto que este cambio se ha podido ver claramente en los ordenadores y los smartphones, cuya expansión en las últimas décadas ha sido increíblemente rápida, hasta el punto en que prácticamente todo el mundo tiene uno actualmente.

Debido a esto, la implementación en los centros no tendría inconveniente en la actualidad, puesto que el gasto sería exclusivamente el pago mensual/anual por uso de la aplicación, mientras que, al agilizar y facilitar las tareas de gestión, el personal tendrá más tiempo para dedicar a otras tareas, por este motivo estamos viendo actualmente un boom de aplicaciones dedicadas a la gestión de los centros.

Mientras que, desde el punto de vista del desarrollo, se ha llegado a un punto en el que el desarrollo de una aplicación para la gestión de centros educativos es perfectamente factible y el desarrollo del proyecto podría sostenerse, puesto que la similitud entre los centros educativos, las tareas que se realizan y la gestión necesaria, hacen que sea posible la creación de una aplicación genérica que sea utilizable por la inmensa mayoría de los centros y personalizable en los detalles para situaciones específicas.

1.2.2 Adaptación al cambio

En este sector hay diferentes tipos de centros y personal, mientras que hay centros que siempre buscan ser los mejores en el sector y hacen lo posible por actualizarse a las nuevas tecnologías, hay numerosos centros donde el personal no tiene esta facilidad para adaptarse al cambio, y a pesar estar anticuados, están acostumbrados a los métodos y procedimientos que se han seguido tradicionalmente y preferirían mantenerlos.

Esto representa una dificultad para la implementación de estos tipos de sistemas, puesto que si los usuarios que los utilizan no se adaptan y hacen un uso incorrecto de estos o no los utilizan, su implementación podría acabar teniendo más consecuencias negativas que positivas debido a una mala utilización por parte de los usuarios.

La implementación de estos tipos de sistemas no solo consiste en la obtención del material hardware y software necesario, sino también en una correcta educación y concienciación del personal para que estos tengan una correcta interacción con el nuevo sistema y puedan sacarle el máximo provecho.

Por lo tanto, cualquier implementación de este tipo de sistema deberá ir acompañado de una correcta motivación y adiestramiento de los trabajadores, puesto que por muy buena que sea la plataforma a implementar, si no existe la intención y conocimiento para utilizarla correctamente, esta implementación acabará en fracaso.

1.2.3 Complejidad

En cuanto a la complejidad, debido a la similitud entre centros educativos, que comparten la mayoría de necesidades en cuanto a las tareas de gestión y comunicación, es posible la generación de una app que abarque estas tareas y gestiones de forma genérica permitiendo que cada centro personalice y particularice ciertas características para adaptarlas a su forma de trabajar.

El desarrollo de una aplicación que cubra todos los aspectos de la gestión educativa de un centro podría llegar a ser muy complejo, pero podría perfectamente hacerse por fases, estableciendo unos módulos base indispensables en una aplicación de este tipo, como serían los mensajes y la subida de archivos, e ir abarcando más partes de la gestión escolar a medida que vaya creciendo la aplicación y los centros las vayan requiriendo. En nuestro caso, nos encontramos con una aplicación en una fase bastante avanzada, que ya tiene los módulos básicos e imprescindibles y está entrando en aquellos módulos más avanzados, como son el módulo de pagos y el módulo de tutorías.

2 Estudio de mercado

Con el fin de ser conscientes del potencial de este mercado e investigar las alternativas para obtener un mayor conocimiento del sector, es necesario realizar un estudio del mercado existente, comprobando la presencia de aplicaciones parecidas a Dinantia. A continuación, aparecen algunas de las aplicaciones más importantes que tienen un objetivo o mercado similar al de Dinantia.

El hecho de analizar cada una de ellas permite ver que funcionalidades pueden ser ofrecidas al cliente con tal de innovar, bien porque no existe ninguna herramienta que la ofrezca o porque estas podrían ser mejorables. Con este objetivo se han analizado 5 plataformas similares a Dinantia [2], y finalmente se ha hecho una reflexión global del estado actual del mercado.

2.1 Additio

Disponible en web y móvil, Additio [3] funciona como un cuaderno de notas para que los docentes planifiquen el curso, controlen asistencias y lleven un seguimiento de sus notas. Sus características incluyen: creación de avisos de tutorías, personalización de horarios, visualizar de forma semanal las tareas correspondientes y evaluaciones entre otras.

Es una aplicación muy completa que como Dinantia pretende centralizar todas las tareas de gestión y comunicación del centro, aunque a diferencia de Dinantia está más orientada a un cuaderno de notas que a una herramienta de gestión, esto se percibe principalmente en el caso de los pagos, donde no ofrece la posibilidad de pedir o realizar pagos a través de la aplicación, sino que ofrece un bloc de notas donde apuntar los pagos realizados con ciertas funcionalidades. Así mismo, en cuanto a las tutorías, también ofrece simplemente la posibilidad de anotarlas y avisar a los alumnos involucrados, pero no contiene una funcionalidad completa de gestión de reserva de tutorías.

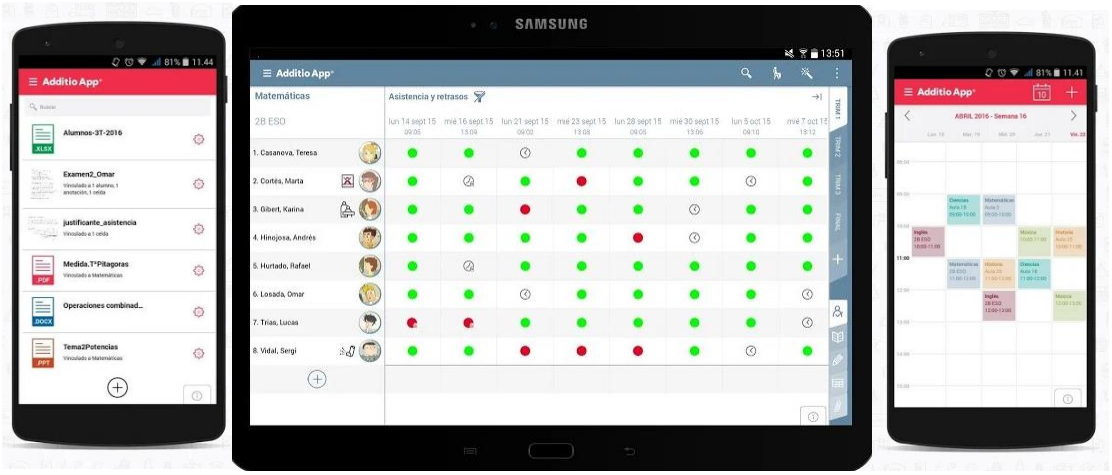


Figura 3: Capturas de pantalla de Additio. Fuente: Additio [3]

2.2 aGora

aGora [4] es una suite de gestión completa e integrada que pretende cubrir todas las necesidades de cualquier centro docente en términos de planificación, gestión y control académico, comercial, económico, etc. El software alega ser escalable, potente y de rápida implantación, permitiendo desde un mismo entorno gestionar alumnos y clientes, definir el *planning* académico, controlar los gastos, emitir facturas o gestionar los cobros.

Esta aplicación ofrece una gran cantidad de utilidades y herramientas de decisión e inteligencia de negocio, y permite la realización de anotaciones de pagos y gastos, agrupación y balance completo. Sin embargo, cuenta con una interfaz bastante obsoleta y los pagos no se pueden realizar a través de la app. En cuanto a las tutorías no parece ofrecer esta opción de forma específica.

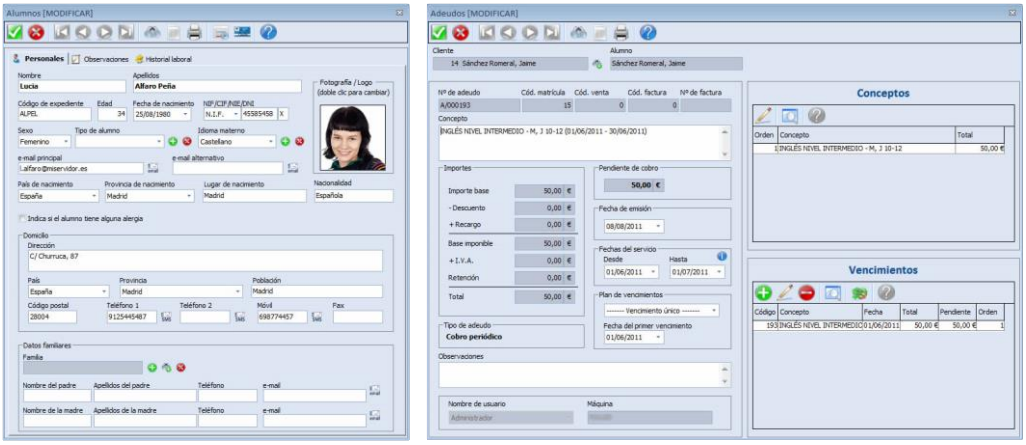


Figura 4: Capturas de pantalla de aGora. Fuente: aGora [4]

2.3 Alexia

La aplicación Alexia [5], desarrollada por Educaria, dispone de una aplicación web y diferentes aplicaciones móviles según el tipo de usuario, teniendo más de 1.200 centros educativos usuarios. Con ella, los centros pueden realizar cualquier gestión académica, económica, de enseñanza y aprendizaje y de comunicación. Dispone de cuadro de mando para extraer indicadores a nivel de centro y agrupación. Se integra con G Suite, Office 365, Moodle, y otros aplicativos, y permite incorporar soluciones específicas de horarios, calidad y aprendizaje, con herramientas para que el centro pueda desarrollar su propia estrategia de contenidos.

En cuanto a los pagos, no parece ofrecer la posibilidad de realizar pagos a través de la app, pero ofrece una potente plataforma de gestión contable, que permite a los centros educativos realizar toda la contabilidad presupuestaria, financiera y analítica desde la misma herramienta que utilizan para su gestión académica y económica. Por lo que respecta a las tutorías, ofrece la posibilidad de gestionar reuniones, y por otra parte ofrece al tutor información sobre el desarrollo de los alumnos obtenida del resto de módulos o funcionalidades, pero no tiene un módulo concreto de reserva y gestión de tutorías.

Figura 5: Capturas de pantalla de Alexia. Fuente: Alexia [5]

2.4 Cifra Educación

Cifra Educación [6] es una plataforma integral de centros de enseñanza que resuelve: las áreas de gestión académicas con un cuaderno del profesor, herramientas para las tutorías y un panel de evaluación; las administrativas, con una secretaría virtual; y las económicas mediante herramientas para llevar al día la facturación. Además, cuenta con una solución especializada para Formación Profesional y también se ha desarrollado en forma de app.

Como en el resto de aplicaciones, Cifra Educación no permite la realización de pagos internamente en la app, pero ofrece una gran cantidad de funcionalidades para la gestión de la facturación, anotación de pagos y gastos y balance. Por otra parte, también ofrece un módulo de tutorías, pero esta está pensada exclusivamente para el uso del personal del centro, por lo que no posee la funcionalidad de gestión de reservas de tutorías, en este simplemente aparece toda la información de los alumnos y un panel de evaluación para la jefatura de estudios y la dirección del centro.

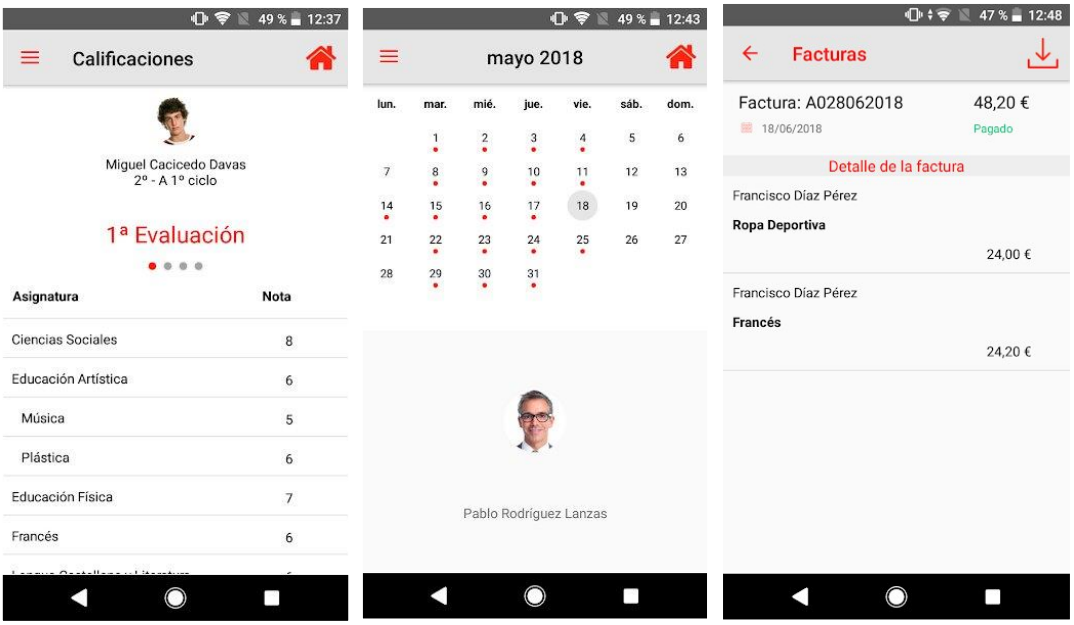


Figura 6: Capturas de pantalla de Cifra Educación. Fuente: Cifra Educación [6]

2.5 Classlife

Classlife [7] es una plataforma que facilita la administración de los centros educativos: desarrollo de los cursos, expedientes académicos, control de contactos y matrículas, gestión de cobros y pagos, asignación de tareas, calendarios y todo el resumen global del centro con reportes, gráficas y estadísticas. Desde su Campus Virtual, es posible conectar con todos los recursos formativos y de comunicación para ofrecer una mejor experiencia de aprendizaje y fomentar la participación de profesores y alumnos. Además, se incluyen informes, encuestas, formularios y automatización de actividades convirtiéndola en una app bastante completa para la gestión del centro.

En cuanto a los pagos, esta aplicación es más completa que el resto en este aspecto, permitiendo realizar y gestionar los pagos a través de la aplicación, ver su estado y mantener un registro y balance de pagos. Sin embargo, en referencia a las tutorías, no tiene ninguna funcionalidad específica, y la reserva se tiene que hacer o vía mensajes dentro de la aplicación o vía medios tradicionales.

Figura 7: Capturas de pantalla de Classlife. Fuente: Classlife [7]

2.6 Conclusiones

Se ha hecho una selección bastante representativa, pero reducida de las aplicaciones orientadas a la gestión de centros en el mercado, con el objetivo de realizar un correcto análisis que permita sacar conclusiones de una forma sencilla y eficaz. Una vez realizada esta selección de potenciales competidores de la aplicación, y comprendido, a grandes rasgos, las funcionalidades y servicios que ofrecen, así como sus objetivos, vamos a realizar un estudio detallado comparándolos con la visión de este proyecto.

Estudiando el mercado se ha podido observar que la mayoría de aplicaciones han tirado por la misma vertiente. En cuanto a los pagos, no permiten realizar pagos a través de la aplicación, pero permiten apuntar los pagos y gastos, y a partir de estos datos obtener balances y gráficas detallados que reflejan el estado actual del centro al respecto. Mientras que por lo que respecta a las tutorías, la mayoría de aplicaciones no las tienen en cuenta como una entidad, se basan de mensajes, anotaciones y reuniones para permitir a los profesores gestionarlas, y las plataformas que las tienen simplemente permiten realizar anotaciones sobre estas por parte del profesor, no permiten realizar reservas o gestionarlas.

En consecuencia, una buena aplicación que permita diferenciarse del resto y brindar utilidad sería una que permita realizar pagos mediante la aplicación, a través de un TPV por ejemplo, permitiendo solicitar y realizar los pagos a través de la aplicación y añadiéndolos automáticamente a la base de datos, sin olvidarse de aquello que ofrece toda la competencia que es la gestión de estos pagos y la visualización de balances, gráficas e información que permitan saber el estado económico del centro, con el extra de que no hay necesidad de incluir los datos de cada pago e ir actualizándolos uno a uno, puesto que, como se realizan a través de la aplicación, ya se tienen los datos.

Mientras que, en el aspecto de las tutorías, nuestra aplicación permitirá a los profesores establecer horas de tutorías y a los alumnos y/o padres reservarlas, pudiendo hacer los profesores anotaciones y especificaciones sobre las tutorías, y cambiar su estado, de forma que simplifique el proceso de gestión de las tutorías, algo que no ofrece prácticamente ninguna otra aplicación en el mercado.

En resumen, aquello que debe ofrecer la aplicación en estos módulos para diferenciarse de los demás es un verdadero funcionamiento interno, que permita diferenciar entre los usuarios y sus capacidades, facilitando que sean los propios usuarios los que hagan los cambios y estos se vayan registrando, informando a los usuarios involucrados y guardando esta información, de forma que estas acciones no se realicen de forma ajena a la aplicación y esta simplemente sea utilizada como un bloc de notas donde apuntar dicha información. Sin prescindir de aquello que ya ofrecen el resto de aplicaciones, que es utilizar esta información introducida para realizar balances y mostrar gráficas que ejemplifiquen y permitan entender estos datos.

3 Definición del alcance

Un proyecto de esta escala, podría tener una gran cantidad de requisitos y funcionalidades, es por ello que es esencial definir el alcance que se quiere abarcar en este proyecto y cuáles son los objetivos.

3.1 Estado actual del sistema

Para definir correctamente el alcance del proyecto, es necesario conocer el estado actual del sistema en el que se va a desarrollar. Actualmente Dinantia está compuesta por dos aplicaciones, una aplicación móvil y una aplicación web. Pero para entender la distribución actual de Dinantia, es necesario saber que, desde hace un año, uno de los clientes más importantes de Dinantia es la Once, una institución que favorece la plena inclusión escolar y social del alumnado con ceguera o deficiencia visual grave. Esto supone que ambas aplicaciones han de cumplir los requisitos de accesibilidad AA de las Directrices de Accesibilidad para el Contenido Web [8], permitiendo que la aplicación sea accesible y utilizable por el máximo número de personas, independientemente de sus conocimientos, capacidades personales o las características técnicas del dispositivo de acceso empleado.

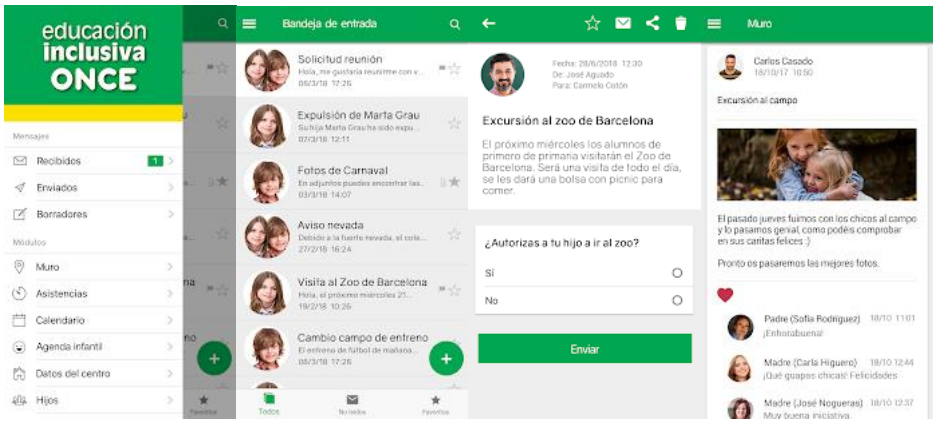


Figura 8: Aplicación móvil Dinantia Once. Fuente: Google Play [9]

En este momento, la aplicación móvil ya cumple los requisitos de accesibilidad AA, y todo el desarrollo realizado en esta también deberá satisfacer estos requisitos. Sin embargo, en cuanto a la web, actualmente existen dos entornos web diferentes que coexistirán en Dinantia como mínimo hasta el verano de este año 2021, cuando está planificado que acabe la migración del entorno antiguo al nuevo. Esto se debe, a que el entorno viejo no garantizaba la accesibilidad, y por ello se ha hecho uno nuevo en el que se ha reutilizado la lógica del entorno antiguo, pero se está recreando el *front*, de forma que garantice la accesibilidad, y actualmente se está realizando la migración de todas las funcionalidades y módulos al nuevo entorno para que, finalmente, este sea el entorno utilizado por todos los clientes.

Actualmente este nuevo entorno no está completo, se están migrando funcionalidades que ya están en el viejo entorno y está siendo utilizado únicamente por La Once, mientras que todos los otros clientes siguen trabajando con el entorno antiguo que no garantiza la accesibilidad, pero contiene todas las funcionalidades y módulos de la aplicación. El objetivo de la empresa es para el verano de 2021 acabar la migración completa de todas las funcionalidades del entorno viejo al entorno nuevo para así poder garantizar la accesibilidad y que todos los clientes utilicen un único entorno, la cual cosa facilitará el desarrollo de la aplicación.

Por este motivo, las funcionalidades que habían sido específicamente solicitadas por los clientes se realizarán primero en el entorno viejo, de forma que pueda salir rápidamente al mercado y esté disponible para los clientes lo más pronto posible, y a continuación, se migrarán las funcionalidades de la web vieja al entorno nuevo, y se continuará el desarrollo en el nuevo entorno, que solo será utilizado por la Once hasta el verano de 2021.

3.2 Objetivos

El proyecto completo consiste en la creación de un módulo de tutorías que permita a los padres y alumnos reservar tutorías con los tutores y a los profesores gestionar sus horarios de tutorías y las diferentes solicitudes, y la ampliación del módulo de pagos ya existente en la aplicación, el cual ya permite la realización de pagos y consultar cierta información de los pagos y transacciones, pero la información puede ser mostrada de forma más representativa y se pueden añadir funcionalidades que mejoren el funcionamiento actual.

La solución contiene muchas funcionalidades, y como la dedicación para el TFG es de unas 500 horas más las 90 horas de GEP (30 horas por crédito) aproximadamente, se priorizarán aquellas funcionalidades base que son indispensables, y una vez realizadas se irán añadiendo más funcionalidades y extendiendo los módulos siguiendo un orden de prioridad basado principalmente en el tiempo, urgencia y beneficio de cada una.

3.3 Stakeholders

Los *stakeholders* son aquellas personas o entidades que de alguna forma afectan o pueden ser afectadas por el desarrollo de un proyecto. En este caso, aquellos que serán afectados por el desarrollo de los módulos son:

- La empresa: Uno de los principales interesados, puesto que el correcto desarrollo de este proyecto ampliará la oferta de su aplicación, mejorando sus prestaciones, su posición en el mercado y sus potenciales clientes.
- El equipo de desarrollo: Aquellas personas que se encargan de desarrollar y mantener la aplicación y que se verán afectadas por la creación y ampliación de módulos, puesto que tendrán que entender su funcionamiento y probablemente utilizarlos o ampliarlos en el futuro para adaptarse a las crecientes necesidades de los clientes.
- Los clientes: Aquellos colegios y academias que verán aumentadas las capacidades de la aplicación que utilizan para comunicación y gestión, facilitando y agilizando su trabajo y la forma de comunicarse con alumnos y padres.
- Secretaría, administración y tesoreros: Estas personas se verán directamente beneficiadas de la ampliación del módulo de pagos, puesto que podrán visualizar y gestionar los pagos de una forma más eficaz, facilitando su trabajo.
- Los profesores: Se beneficiarán de la creación del módulo de tutorías, puesto que no tendrán que pasar por el tedioso proceso actual de ponerse en contacto con los padres mediante notas entregadas a los hijos para la reserva y gestión de tutorías, y podrán hacerlo directamente desde la aplicación.
- Los padres y alumnos: Se beneficiarán de la creación del módulo de tutorías, puesto que les permitirán reservar tutorías de forma sencilla desde la aplicación, estar al día de su estado y mantenerse en contacto con sus tutores.

- La competencia: Aquellas aplicaciones con un sistema similar, como las analizadas en el estudio de mercado, verán la aparición de una aplicación que pretende mejorar el producto que ellos ofrecen, viéndose amenazada su idea y producto de negocio.

3.4 Requisitos

Los requisitos que han de cumplir los módulos que se van a desarrollar se pueden dividir en dos categorías:

- Requisitos funcionales: Definen las funciones que compondrán el sistema *software*.
- Requisitos no funcionales: Especifican los criterios que ha de cumplir el proyecto desarrollado.

3.4.1 Requisitos funcionales

En cuanto a los requisitos funcionales, el desarrollo de la aplicación se realizará en fases, con el objetivo de priorizar primero el desarrollo de los dos módulos y sus funcionalidades básicas para poder añadirlos y ser utilizados lo más pronto posible. Una vez constituidos los dos módulos se continuará su desarrollo y crecimiento hasta el final del proyecto añadiendo tantas funcionalidades extra como sea posible en el tiempo establecido.

Para llevar esto a cabo, comenzaremos por la fase inicial que abarcará todas las funcionalidades imprescindibles para que el módulo satisfaga los objetivos establecidos para este y se desarrollará en el entorno web viejo, sacando las funcionalidades tan pronto se acabe la fase 1, puesto que las funcionalidades incluidas en esta fase habían sido solicitadas por los usuarios actuales de la aplicación, que están utilizando el entorno viejo, y lo utilizarán hasta el nuevo curso en septiembre de 2021.

Continuaremos con la segunda fase en la que se migrarán los módulos al entorno web nuevo para garantizar la accesibilidad, y se añadirán otras funcionalidades que ampliarán su utilidad y versatilidad, mejorando la vida de los usuarios de la aplicación. Seguida probablemente de más fases realizadas por la empresa una vez finalizado el proyecto que continúen el desarrollo de los módulos según las necesidades futuras de los clientes.

Además estas funcionalidades también las dividiremos en funcionalidades para profesores y funcionalidades para padres y alumnos, puesto que la información y características a las que tendrán acceso dentro de la aplicación son diferentes; y en funcionalidades para la aplicación web y funcionalidades para la aplicación móvil, puesto que no todas las funcionalidades van a estar en ambas, esto se debe a que los centros realizan la mayoría de tareas de gestión únicamente desde la web, mientras que la visualización de datos y actualización de la información se suele realizar desde ambas.

a) Módulo de tutorías

Los requisitos funcionales del nuevo módulo de reserva y gestión de tutorías son los siguientes:

1. Durante la Fase 1, los profesores, desde la web, tendrán la capacidad de crear franjas de tutorías y gestionar las franjas. Mientras que, tanto desde la web como desde el móvil, podrán visualizar sus tutorías, bloquear o desbloquear las tutorías creadas para que se puedan realizar reservas o no, editar una solicitud de tutoría para aprobarla o rechazarla, editar una reserva confirmada para cancelarla y añadir anotaciones públicas y privadas en las tutorías, que podrán ser editadas por el profesor, y que le servirán para añadir anotaciones y recordatorios sobre esta.

Por otro lado, los padres y alumnos no tendrán ninguna funcionalidad en la web durante la fase 1, y desde la aplicación móvil podrán reservar tutorías, seleccionar el tutor con el que realizar la tutoría y, en el caso de los padres, seleccionar el hijo sobre el que se realizará la tutoría, también se podrán visualizar las tutorías reservadas y cancelarlas si fuera necesario.

Finalmente, también se notificará al usuario que hace la reserva y al tutor al que corresponde la franja de tutoría reservada cuando se produzcan actualizaciones sobre la reserva.

La versión web, puesto que es bastante reducida inicialmente, se realizará en el entorno antiguo, para poder garantizar el acceso cuanto antes a los clientes, y posteriormente, en la segunda fase, se migrará al entorno nuevo, que garantizará la accesibilidad.

2. Por lo que respecta a la fase 2, en esta fase se realizará la migración de la parte web del módulo al nuevo entorno que garantizará la accesibilidad, y una vez migrado se continuará el desarrollo en este entorno, donde se añadirán nuevas funcionalidades.

   Concretamente, los padres y alumnos podrán visualizar sus tutorías reservadas, reservar tutorías y cancelar tutorías desde la web. Los profesores podrán solicitar a usuarios que reserven tutorías. Se añadirá la posibilidad de que los padres puedan compartir tutorías cuando así lo deseen. Y las tutorías reservadas se verán en el calendario, si dicho módulo está activo.

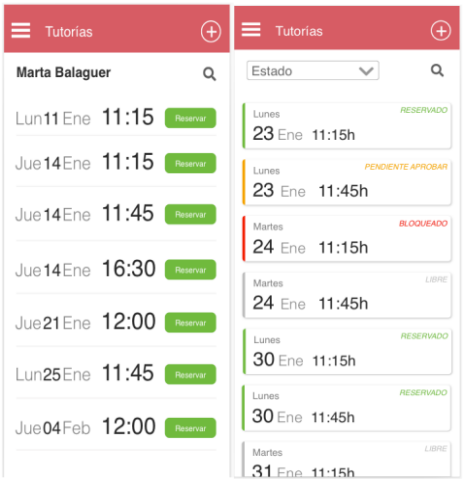


Figura 9: Modelos de pantallas de reserva de tutorías. Fuente: Dinantia [1]

b) Módulo de pagos

La ampliación del módulo de pagos se realizará únicamente en la web, puesto que la gestión de pagos por parte de los centros se gestiona desde la aplicación web, y como con el módulo de tutorías, se iniciará el desarrollo de la fase 1 del módulo en el entorno web viejo de cara a su pronta salida, puesto que algunos usuarios lo habían solicitado. Mientras que en la fase 2 se migrará la parte web al nuevo entorno donde se continuará el desarrollo. Las nuevas funcionalidades que ampliarán y mejorarán el módulo de pagos existente son las siguientes:

1. Durante la fase 1, se incluirá la visualización de pagos por grupos, pudiendo ver si un grupo tiene pagos pendientes dentro de un rango de fechas, ver los pagos de los grupos por cada alumno y los detalles de los pagos, exportar dicha información, editar el estado de los pagos y enviar recordatorios de pagos pendientes.

2. En cuanto a la fase 2, tras la migración se tenía pensado incluir más funcionalidades que se habían barajado, como la inclusión del etiquetado de pagos y el filtro de pagos por etiquetas en las distintas páginas ofrecidas, tareas que no requerían de mucho tiempo debido al apretado calendario, pero que incrementarían en gran medida la utilidad de este módulo. Sin embargo, se decidió centrar los esfuerzos en el módulo de tutorías, por lo que finalmente, solo se realizará la migración de la web vieja al entorno actual en la fase 2.

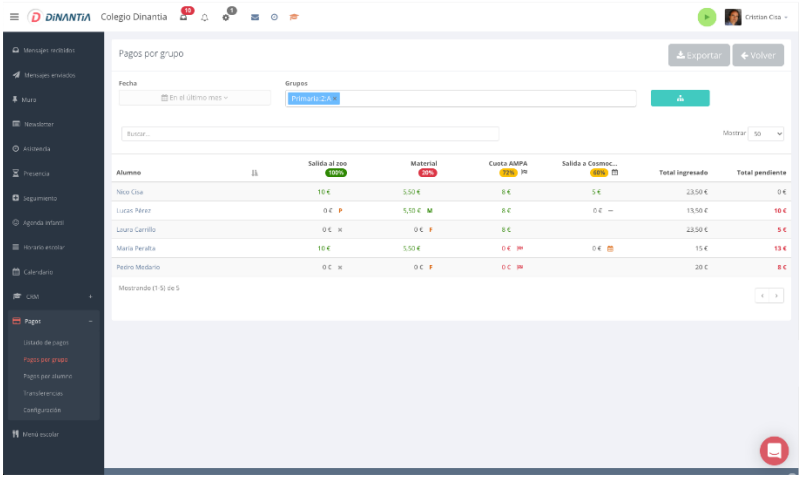


Figura 10: Modelo de la página de organización de pagos por grupos. Fuente: Dinantia [1]

3.4.2 Requisitos no funcionales

En cuanto a los requisitos no funcionales que deberán cumplir los módulos desarrollados, se pueden resumir en los siguientes:

- Apariencia. Se han de mantener los criterios de apariencia definidos por la empresa y utilizados en el resto de módulos.
- Usabilidad. La aplicación ha de ser sencilla a la hora de utilizarla, mostrando menús intuitivos que faciliten a los usuarios navegar a través de ella.
- Asegurar la fiabilidad y disponibilidad. Esta aplicación actualmente está en uso por clientes reales, por lo tanto, ha de estar operativa en todo momento, estando caída solo en momentos necesarios para actualizar bases de datos o código que requiera detener los accesos a la base de datos, pero estos deben de hacerse por la noche que es cuando menos usuarios están activos.
- Mantener la internacionalización. La aplicación actualmente soporta tres idiomas que también deberán soportar los nuevos módulos, español, catalán e inglés, esto se hace mediante el uso de etiquetas para las palabras que se muestran por pantalla, a las cuales se les asigna luego una palabra dependiendo del idioma seleccionado, permitiendo que en cualquier momento se pueda añadir otro idioma si fuera necesario.
- Mantener la compatibilidad. La aplicación web está diseñada y preparada para ser utilizada desde cualquiera de los navegadores actuales. Mientras que la aplicación móvil se desarrolla de forma híbrida refinando de forma nativa algunas características cuando corresponde, permitiendo que la aplicación sea compatible tanto con Android como con IOS y pueda ser utilizada desde cualquier smartphone.

- Asegurar la seguridad y la privacidad de los datos de los clientes siguiendo la legislación establecida. Puesto que la aplicación ya tiene clientes reales y datos confidenciales, se han de seguir las especificaciones de la Ley General de Protección de Datos, de forma que los datos de ningún cliente queden al descubierto. También se deberá tener en cuenta los permisos de los usuarios de cara al acceso a los datos y funcionalidades.
- Garantizar la accesibilidad. Como se ha explicado anteriormente, la aplicación móvil ya cumple el criterio AA de accesibilidad, y la aplicación web está dividida en dos entornos, el entorno antiguo que no cumple los criterios, pero está completo, y el entorno nuevo que no está completo, pero cumple los criterios de accesibilidad. Aquellos cambios y adiciones realizados en la aplicación móvil y en el nuevo entorno de la aplicación web habrán de cumplir estos criterios de accesibilidad.

3.5 Obstáculos y riesgos potenciales

Los riesgos potenciales que pueden afectar al correcto desarrollo del proyecto son los siguientes:

- Inexperiencia: A pesar de haber realizado varios proyectos a lo largo de la carrera y haber trabajado con diferentes lenguajes y entornos, nunca había trabajado con CakePHP, Vue.js o Ionic, que son los principales *frameworks* utilizados en los diferentes repositorios, esto supone que mi conocimiento del entorno no será completo e irá creciendo conforme avance el proyecto, aprendiendo algunas partes a medida que vayan apareciendo.
- *Bugs*: Un problema frecuente en el desarrollo de *software* y que suele suponer la pérdida de una considerable cantidad de tiempo en la realización de depuración para intentar solucionarlos, son los *bugs*, errores o problemas del sistema *software* que desencadenan un resultado indeseado. Y por supuesto como con cualquier otro proyecto *software* está asegurado que vayan a aparecer en este también, por lo que tendrán que ser tratados cuando aparezcan utilizando las diversas herramientas de depuración que nos ofrecen los lenguajes y *frameworks* utilizados.
- Desviaciones de la planificación: En cualquier proyecto se pueden suceder desviaciones de la planificación especificada por diversos motivos: una tarea que es más extensa o complicada de lo esperada, la aparición de un problema que supone la pérdida de una gran cantidad de tiempo, la imposibilidad de reutilizar un complemento diseñado para otras partes de la aplicación y que suponga la creación de un componente específico, etc. Es muy complicado que un proyecto siga la planificación al pie de la letra, por lo que a lo largo del proyecto se ha de ver el estado en que se encuentra y replantear el desarrollo en caso de que sea necesario.
- Posibles problemas derivados de la pandemia: Vivimos en una situación excepcional que ha cambiado la forma en que vivimos, nos comunicamos, nos relacionamos y realizamos nuestro trabajo. Como el trabajo lo realizo desde casa, de forma online, utilizando diferentes plataformas de comunicación, en principio la pandemia no debería suponer un problema o gran riesgo para el desarrollo del proyecto, pero dicho esto, la pandemia podría afectar al desarrollo de forma inesperada, por lo tanto, sigue siendo un factor de riesgo a tener en cuenta, si, por ejemplo, me contagiara y fuera grave, esto supondría un retraso en la realización del proyecto como mínimo.

4 Metodología y rigor

Una vez explicados el alcance, los requisitos y los posibles riesgos del proyecto, pasamos a hablar de las metodologías de trabajo y desarrollo seguidas durante la realización del proyecto, y las herramientas utilizadas.

4.1 Metodología de trabajo

La metodología de trabajo seguida actualmente en el desarrollo de la aplicación es la metodología Scrum [10], perteneciente a la familia de las metodologías ágiles [11], las cuales envuelven todas aquellas metodologías que tienen un enfoque basado en el desarrollo iterativo e incremental de proyectos *software*, donde los requisitos y soluciones evolucionan con el tiempo según las necesidades del proyecto, realizando el trabajo mediante la colaboración de equipos autoorganizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo.

4.1.1 ¿Qué es SCRUM?

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

4.1.2 Proceso de desarrollo basado en SCRUM

En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija llamados *sprints* (iteraciones que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4 semanas), en el caso de mi proyecto son de 4 semanas. Cada iteración tiene que proporcionar un resultado completo, un incremento del producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos o requisitos del producto, que actúa como plan del proyecto. En esta lista el cliente o *Product Owner* prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas.

Las actividades que se llevan a cabo en Scrum son las siguientes:

a) Planificación de la iteración

El primer día de la iteración se realiza la reunión de planificación de la iteración. Tiene dos partes:

1. Selección de requisitos. El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que prevé que podrá completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.
2. Planificación de la iteración. El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos seleccionados. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se autoasignan las tareas, se autoorganizan para trabajar incluso en parejas si es necesario, con el fin de compartir conocimiento o para resolver juntos objetivos especialmente complejos.

b) Ejecución de la iteración

Cada día el equipo realiza una reunión de sincronización o *Daily Scrum*, normalmente delante del tablero con las tareas del *sprint* y su estado. El equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con la previsión de objetivos a mostrar al final de la iteración. En la reunión cada miembro del equipo responde a tres preguntas:

- ¿Qué he hecho desde la última reunión de sincronización para ayudar al equipo a cumplir su objetivo?
- ¿Qué voy a hacer a partir de este momento para ayudar al equipo a cumplir su objetivo?
- ¿Qué impedimentos tengo o voy a tener que nos impidan conseguir nuestro objetivo?

Durante la iteración el *Scrum Master* se encarga de que el equipo pueda mantener el foco para cumplir con sus objetivos.

- Elimina los obstáculos que el equipo no puede resolver por sí mismo.
- Protege al equipo de interrupciones externas que puedan afectar el objetivo de la iteración o su productividad.

Durante la iteración, el cliente junto con el equipo refina la lista de requisitos (para prepararlos para las siguientes iteraciones) y, si es necesario, cambian o replanifican los objetivos del proyecto con el objetivo de maximizar la utilidad de lo que se desarrolla y el retorno de inversión.

c) Inspección y adaptación

El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

1. Revisión/Demostración. El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.
2. Retrospectiva. El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El *Scrum Master* se encargará de eliminar o escalar los obstáculos identificados que estén más allá del ámbito de acción del equipo.

4.1.3 Herramientas utilizadas

Para seguir esta metodología de trabajo y organizar los *sprints* y las tareas se utiliza la herramienta Jira [12], una herramienta en línea para la administración de tareas, el seguimiento de errores e incidencias y para la gestión operativa de proyectos. Jira es utilizado en Dinantia para el desarrollo de *software*, sirviendo de apoyo para la gestión de requisitos, seguimiento del estado de desarrollo y más tarde para la gestión de errores.

Además, Jira es utilizado para la gestión y mejora de los procesos, gracias a sus funciones para la organización de flujos de trabajo que permite la planificación, modificación y seguimiento del *product backlog*, los *sprints* y sus tareas a lo largo del proyecto.

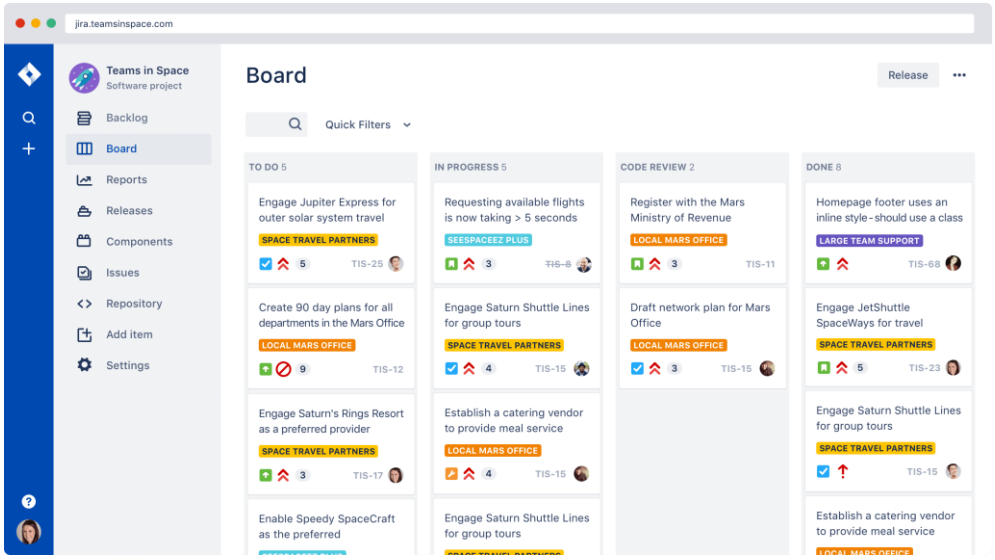


Figura 11: Ejemplo de tablero de tareas de un sprint en Jira. Fuente: Atlassian [12]

4.2 Metodología de desarrollo

Con el objetivo de utilizar los repositorios de código de forma eficiente y efectiva y evitar problemas de incompatibilidad de versiones, la metodología de desarrollo seguida en el desarrollo de la aplicación Dinantia es Gitflow [13], una metodología basada en Git [14], un *software* de control de versiones pensado para mejorar la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos, incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

4.2.1 ¿Qué es Gitflow?

Gitflow es un diseño de flujo de trabajo de Git que define un modelo estricto de ramificación diseñado alrededor de la publicación del proyecto. Proporciona un marco sólido para gestionar proyectos más grandes, siendo ideal para los proyectos que tienen un ciclo de publicación programado.

Este flujo de trabajo no añade ningún concepto ni comando nuevos. En su lugar, asigna funciones muy específicas a las diferentes ramas y define cómo y cuándo deben interactuar. Por supuesto, también te aprovechas de todos los beneficios del flujo de trabajo de rama de función: solicitudes de incorporación de cambios, experimentos aislados y una colaboración más eficaz.

4.2.2 Ramas y funciones

En un proyecto Gitflow se definen 5 tipos de ramas:

- *Master*: En esta rama se almacena el historial de publicación oficial, aquí se encuentra el código de la aplicación que forma parte de una versión publicada de la aplicación, es decir, el código resultante al final de una iteración. Por tanto, no se puede realizar ningún *commit* que provenga directamente del entorno de desarrollo a esta rama.
- *Develop*: Esta rama sirve como rama de integración de funciones, aquí se van incorporando las diferentes funcionalidades que se van desarrollando durante el proyecto, una vez hayan pasado los diferentes controles de calidad establecidos. De forma similar a la rama *master*, no se puede realizar un *commit* directamente a esta rama, sino que hay que pasar antes por otro tipo de rama.
- *Feature*: Estas ramas son las que contienen el código de las nuevas funcionalidades de la aplicación o que solucionan problemas encontrados durante el desarrollo. Estas ramas provienen de la rama *develop* y es donde se realizan todos los *commits* desde el entorno de desarrollo. De esta manera el código nuevo queda aislado del código comprobado y potencialmente funcional de la rama.

  Para poder incorporar estos cambios en *develop*, es decir, realizar la fusión entre las ramas, es necesario realizar un *pull request*, donde se comprobará que se cumplen las funcionalidades específicas para las que se creó la rama y esta no ocasionará problemas de compatibilidad al añadirse a *develop*. El código de esta rama será comprobado por como mínimo un miembro del equipo que no trabajó en su desarrollo para garantizar estos criterios.
- *Release*: Una vez que el desarrollo ha adquirido suficientes funciones para una publicación (o se está acercando una fecha de publicación predeterminada), se bifurca una rama de versión a partir de una de desarrollo. Al crear esta rama, se inicia el siguiente ciclo de publicación, por lo que no se pueden añadir nuevas funciones tras este punto; solo las soluciones de errores, la generación de documentación y otras tareas orientadas a la publicación deben ir en esta rama. Una vez que esté lista para el lanzamiento, la rama de publicación se fusiona en la rama *master*. Además, deberá volver a fusionarse en la de desarrollo, que podría haber progresado desde que se inició la publicación.
- *Hotfix*: Si una vez se ha publicado la versión de la aplicación se detecta algún problema con la aplicación, sobre todo proveniente de las pruebas con usuarios reales, se debe bifurcar una rama llamada *hotfix* desde la rama *master*. En esta rama se solucionarán estos problemas y, acto seguido, se iniciará un *pull request* hacia *master*. Una vez el código haya sido verificado y se haya comprobado que los problemas han sido solventados, se realizará la fusión con la rama *master* y, a continuación, con la rama *develop*.

A continuación, podemos ver las diferentes ramas en un proyecto Gitflow estándar.

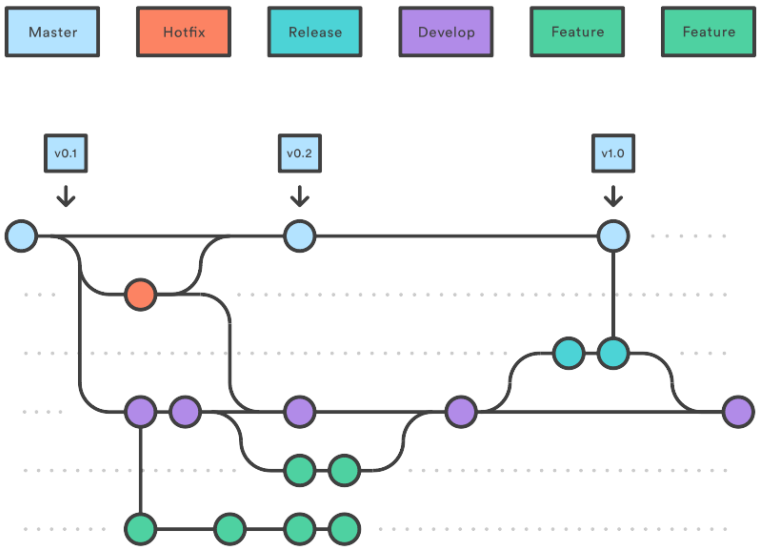


Figura 12: Ramas de un proyecto Gitflow estándar. Fuente: Atlassian [13]

4.2.3 Herramientas utilizadas

Para hospedar los repositorios se utiliza Bitbucket [15], un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de versiones Git, preparado con todas las funcionalidades para seguir una metodología de desarrollo Gitflow de forma eficiente.

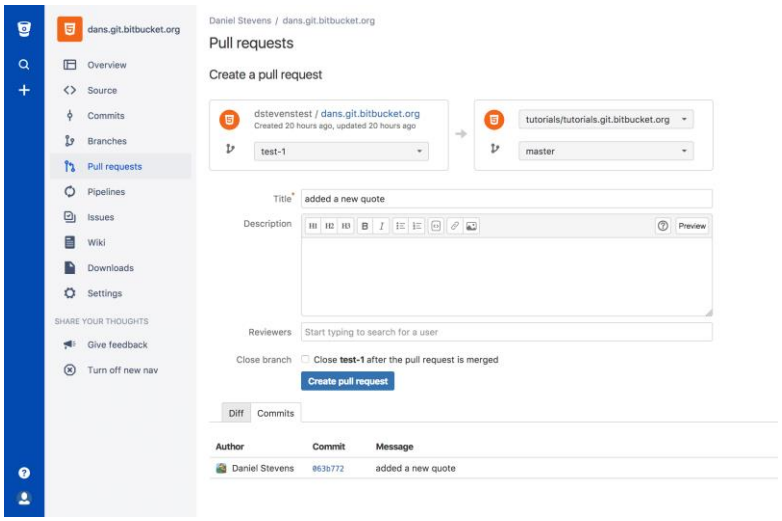


Figura 13: Ejemplo iniciación de un pull request en Bitbucket. Fuente: Atlassian [16]

4.3 Comunicación

Para la comunicación entre los miembros de la empresa utilizamos una combinación de Slack [17], una herramienta de comunicación en equipo a través de la cual podemos realizar cuestiones y acordar soluciones de forma textual, y Google Meet [18], un servicio de videollamadas desarrollado por Google que utilizamos para la realización de reuniones o conversaciones que se realizan de forma más eficiente y efectiva en una videollamada que mediante texto.

5 Especificación

Una vez explicada la motivación detrás de los módulos, haber explicado las soluciones actuales en el mercado, sugerido nuestra alternativa, que mejore las soluciones ya existentes, y haber analizado las funcionalidades y requisitos que ha de satisfacer nuestro sistema, pasamos a explicar las entidades que conformaran el proyecto, y como se podrá interactuar con ellas. Inicialmente, se mostrará el esquema conceptual del sistema y se explicarán cada una de las entidades que lo engloban, y a continuación, se analizarán las diferentes formas en las que el usuario podrá interactuar con el sistema.

Cabe recalcar que solo se tendrán en cuenta aquellas entidades, atributos y acciones pertenecientes a los módulos desarrollados o con las que estos interactúen, puesto que el sistema completo es extremadamente grande, y explicarlo entero simplemente generaría ruido que dificultaría la comprensión de los módulos que nos atañen.

5.1 Esquema conceptual

El esquema conceptual de un sistema es la representación gráfica de los modelos que lo componen. En Ingeniería del Software esto es conocido como diagrama de clases. Primero se explicarán textualmente las diversas entidades que componen el sistema y posteriormente se mostrará el diagrama de clases del que forman parte.

5.1.1 Descripción de las clases

El esquema conceptual del proyecto dispone de una gran cantidad de clases, las cuales se pasan a explicar a continuación.

- *Group*: Un grupo es la entidad que representa los diferentes grupos de cada escuela, estos siguen una estructura basada en nodos, definiendo nodos raíz, y pudiendo ir añadiendo hijos. Los grupos disponen de un identificador (*id*), la abreviatura del nombre del grupo (*code*), la fecha de creación (*created*) y la fecha de última modificación (*modified*). Además, un grupo tiene mínimo un nombre asociado, aunque puede tener un nombre por cada idioma disponible, pertenece a una escuela, está asociado al primer grupo hijo que tiene o a sí mismo, está asociado al último grupo hijo que tiene o al nodo siguiente, puede tener un grupo padre o no y puede tener una serie de miembros, los cuales pueden ser tutores o alumnos del grupo.
- *School*: Una escuela es la entidad que representa los centros que disponen de acceso a la aplicación. Una escuela tiene un *id*, que la identifica, un nombre (*name*), un teléfono de contacto (*phone*), un correo de contacto (*email*), una dirección web (*web*), una dirección postal (*address*), el color utilizado por el centro en la aplicación (*color*), un huso horario (*timezone*), una fecha de creación (*created*) y fecha de última modificación (*modified*). Por otro lado, una escuela tiene administradores, tiene asociados aquellos módulos que están disponibles (algunos módulos se pagan por separado) y si están activos o no, puede tener un logo, tiene asociados los idiomas utilizados en la escuela y su prioridad, estableciendo la traducción que tendrá preferencia en caso de estar definida en diversos idiomas, aquellos ajustes que tienen un valor diferente al establecido por defecto, los grupos que pertenecen a la escuela y un conjunto de cuentas que pertenecen a esta escuela y que pueden tener diferentes roles.

- *Module*: Un módulo representa aquellos módulos que componen la aplicación, como los módulos de tutoría, pagos, asistencia, etc. Un módulo tiene un *id* que lo identifica, un nombre (*name*) y una descripción (*description*). Además, está asociado a aquellas escuelas en las que está disponible, indicando si está activo o no en la escuela.
- *Language*: Un idioma es la entidad que define los idiomas disponibles en la aplicación. Un idioma está identificado por un *id*, tiene un nombre (*name*) y tiene una abreviatura (short). En cuanto a las asociaciones, un idioma está asociado a aquellas escuelas en las que se utiliza y está asociado a diversos grupos, especificando la traducción del nombre del grupo en este idioma.
- *Setting*: Un ajuste es la entidad que identifica las diferentes configuraciones que se pueden establecer en los diferentes módulos. Un ajuste tiene un *id* que lo identifica, un nombre (*name*), si el ajuste es a nivel de escuela o a nivel de cuenta (*scope*), la especificación del tipo de valor que puede tener este ajuste (*data_type*) y el valor por defecto del ajuste (*default_value).* Además, un ajuste puede estar configurado a nivel de escuela o a nivel de cuenta, y cuando el valor del ajuste para una escuela o cuenta se cambia a un valor diferente al valor por defecto, la escuela o cuenta se asocia al ajuste con el valor establecido.
- *Role*: Un rol identifica los roles que una cuenta puede tener en la aplicación y los permisos que esta tiene. Un rol está identificado por un *id* y tiene un nombre (*name*). Los roles están asociados a diversas cuentas, especificando las funciones y permisos que estas cuentas tienen.
- *Account*: Una cuenta es la entidad que identifica a un miembro de una escuela. Una cuenta tiene un identificador (*id*), un nombre (*name*), un correo electrónico (*email*), un teléfono móvil (*phone*), una contraseña (*password*), el género de la cuenta (*gender*), la última fecha de inicio de sesión (*last_login*), la fecha de creación de la cuenta (*created*) y la última fecha de modificación (*modified*). Por otra parte, las cuentas están asociadas a las escuelas a las que pertenecen, y pueden administrar escuelas, a los ajustes de la cuenta, si estos son diferentes al valor por defecto, a los grupos a los que pertenece en dicha escuela y la posición que ocupa en dichos grupos, a los roles que lleva a cabo esta cuenta, a los padres de la cuenta, en caso de que sea alumno y tenga padres definidos en la aplicación, a los mensajes enviados y recibidos por la cuenta, a los pagos y eventos del calendario que involucran a la cuenta, a los dispositivos móviles de la cuenta, a las tutorías y franjas de tutorías en las que forma parte y a la imagen de perfil de la cuenta,
- *Message*: Un mensaje es la entidad que identifica los mensajes enviados entre los usuarios de la aplicación. Un mensaje está compuesto por un *id* que lo identifica, el estado del mensaje (*status*), el ámbito al que pertenece el mensaje (*source*), la última fecha de modificación (*modified*), la fecha de creación (*created*) y la fecha de envío (*sent*). Por otro lado, un mensaje está asociado a la cuenta emisora del mensaje, los destinatarios del mensaje, el mensaje al que responde y el mensaje raíz en caso de ser una respuesta, en caso de ser un mensaje asociado a una tutoría, también está asociado a dicha tutoría y en caso de contener pagos, también está asociado a dichos pagos.
- *TutoringRange*: Una franja de tutorías identifica un rango de fechas en el cual todos los días que coincidan con el día de la semana especificado se ha generado una tutoría a la hora de inicio especificada, su función es permitir la creación, edición y eliminación masiva de tutorías. En cuanto a los atributos, tiene un *id* que lo identifica, una fecha de inicio (*start_date*) y una fecha de finalización (*end_date*) que definen el rango de fechas, un día de la semana (*weekday*) y una hora de inicio (*start_hour*) en las que se crearan tutorías dentro del rango, la fecha de creación de la franja (*created*) y la fecha de última modificación (*modified*). En cuanto a las asociaciones, está asociada a las tutorías definidas dentro del rango y al tutor que imparte las tutorías.

- *Tutoring*: Una tutoría es la entidad que representa una hora de tutoría de un tutor, que puede tener tres estados, disponible/libre, bloqueada o reservada. En cuanto a los atributos tiene un *id* que la identifica, la fecha en que se realizará la tutoría en caso de ser reservada (*date*), la localización donde se hará dicha tutoría (*location*), notas privadas (*private_notes*) y públicas (*public_notes*) del tutor, el estado de la tutoría (*status*), si es compartida con el resto de padres del alumno en caso de haber sido reservada por un padre (*notify_all_parents*), la fecha de creación de la tutoría (*created*) y la fecha de la última modificación (*modified*). En cuanto a las asociaciones, una tutoría está asociada a la franja que la define, al tutor que imparte la tutoría, al alumno y solicitante (padre o alumno que hace la reserva) en caso de estar reservada, a la cuenta que realizo la última modificación, al evento del calendario asociado a la tutoría en caso de estar reservada y a los mensajes de notificación de cambios o solicitud de tutorías enviados en caso de haberlos.
- *Payment*: Un pago es la entidad que identifica los pagos realizados a través de la aplicación. Un pago está compuesto por un *id* que lo identifica, el estado en que se encuentra actualmente (*status*), la cantidad del pago (*amount*), la fecha límite de pago si es que se ha especificado (*limit_date*), si el pago está activo (*active*), la fecha de creación (*created*) y la fecha de última modificación (*modified*). Por otra parte, está asociado a la cuenta que ha creado el pago y la que ha realizado la última modificación, las cuentas a las que va dirigido y los mensajes en los que se ha enviado.
- *Calendarevent*: Esta entidad identifica los eventos del calendario de la aplicación. Está identificado por un *id*, tiene una fecha de inicio (*start*) y finalización (*end*), una localización (*location*), una fecha de notificación (*notification*), si ha sido notificado (*notified*), si se ha de notificar tras su creación (*notified_on_creation*), el color del evento (*color*), la fecha de creación (*created*) y la fecha de última edición (*modified*). Un evento está asociado a la cuenta que lo ha creado, la última cuenta que lo ha modificado, los destinatarios del evento y la tutoría en que consiste el evento, en caso de ser un evento de tutoría.
- *Device*: Esta entidad identifica los dispositivos móviles asociados a las cuentas. Está identificada por un *id,* un atributo *device* que identifica inequívocamente al dispositivo y se utiliza en las llamadas, una gran cantidad de atributos que identifican al dispositivo concreto: versi*ón* (*versión)*, fabricante (*manufacturer)*, modelo (*model)*, sistema operativo (*os)* y versión del sistema operativo (*osversion)*; y la última fecha de inicio de sesión (*last_login*), Un dispositivo está asociado a las cuentas que lo han utilizado para acceder.
- *Uploads*: Esta entidad identifica los archivos que son subidos a la aplicación. Está compuesta por un identificador (*id*), una ruta (*path*), el nombre de archivo (*filename*), la extensión del archivo (*extension*), el tamaño del archivo (*size*) y la fecha de subida (*created*). En cuanto a las asociaciones, las subidas están asociadas a las escuelas, puesto que las escuelas tienen un logo y a las cuentas, puesto que tienen una imagen de perfil.

5.1.2 Diagrama de clases

El diagrama de clases que establece las relaciones entre las distintas clases es el siguiente:

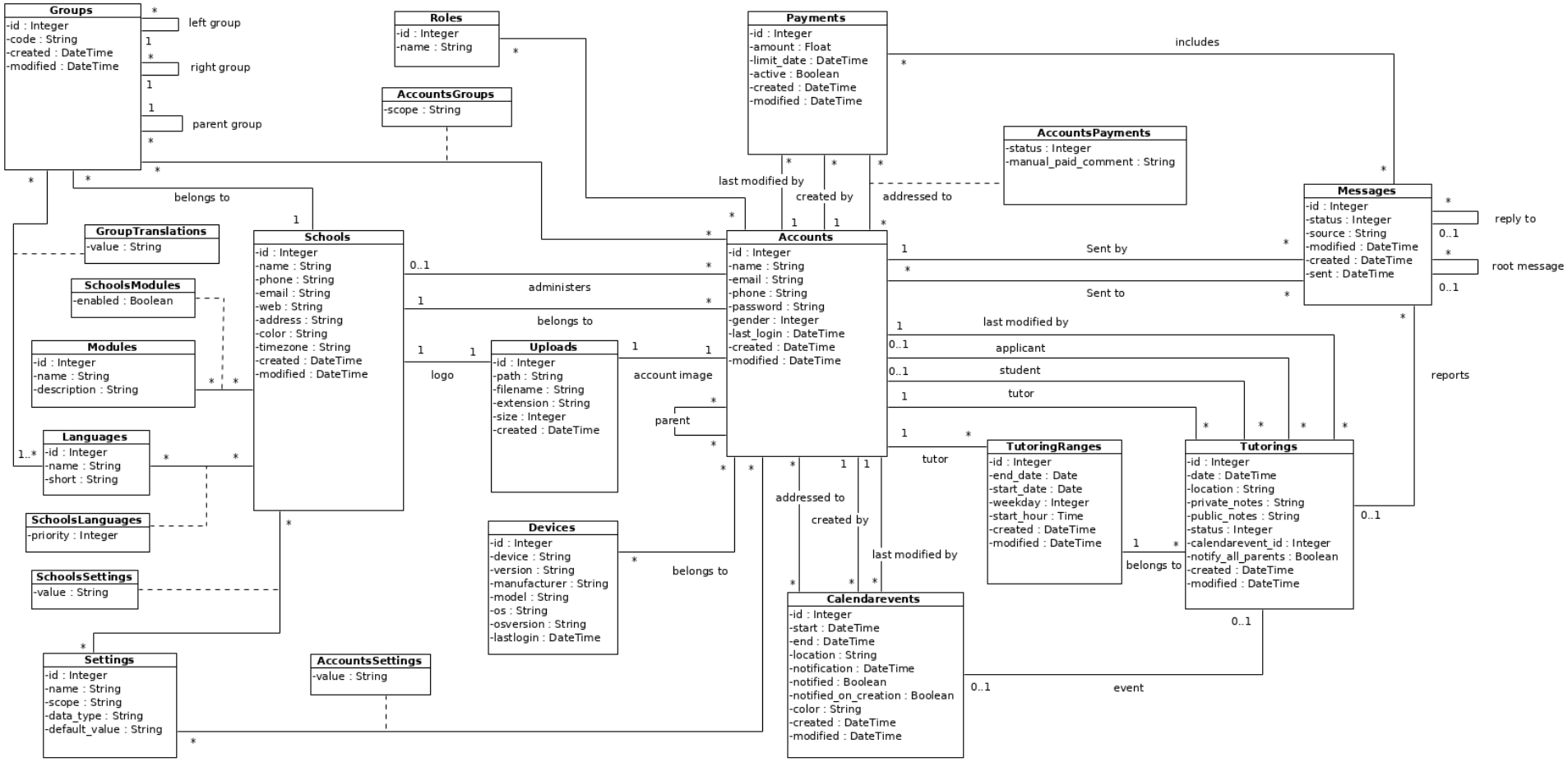


Figura 14: Diagrama de clases. Fuente: Visual Paradigm [19]

5.2 Esquema de comportamiento

Una vez analizadas las clases del modelo es necesario declarar las diversas formas de interactuar con el sistema que tiene el usuario según su rol en la aplicación, a continuación, se muestran y explican los actores que interaccionaran con el sistema, los subsistemas en los que ha sido dividido para una mejor comprensión y los diversos casos de uso establecidos en cada subsistema.

En este caso, se ha decidido agrupar los casos de uso según la clase con la que se interactúa principalmente o la que tiene más importancia en la interacción, resultando en tres subsistemas: Franjas de tutorías, Tutorías y Pagos

5.2.1 Actores

Los actores son aquellas personas o entidades que participarán en el diagrama de casos de uso, en este caso, los actores dependen del rol que tiene la cuenta asociada a la escuela, si es miembro de algún grupo de la escuela y que posición ocupa en este grupo, concretamente, los actores son los siguientes:

- Tutor: Son aquellos usuarios que son tutores de al menos un grupo en la escuela.
- Alumno: Son aquellos usuarios que son alumnos de, como mínimo, un grupo de la escuela.
- Padre: Son aquellos usuarios que tienen al menos un hijo que pertenece a la escuela
- Tesorero: Son los usuarios definidos como tesoreros por el centro.

5.2.2 Subsistema de franjas de tutorías

Los casos de uso del subsistema de franjas de tutorías atañen únicamente a los Tutores, y son los siguientes:

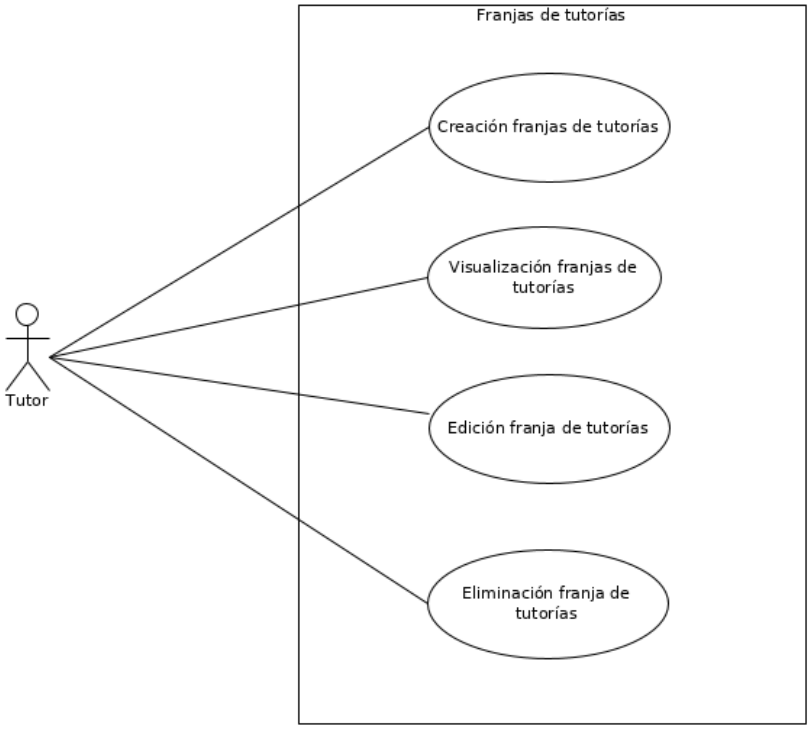


Figura 15: Diagrama de casos de uso del subsistema de franjas de tutorías. Fuente: VisualParadigm [19]

Caso de uso	C1 – Creación franjas de tutorías	
Descripción	Un Tutor puede crear franjas de tutorías, creando por extensión las tutorías correspondientes	
Actores	Tutor	
Flujo principal	1	El Tutor accede a la página de creación de franjas de tutorías.
	2	El Tutor selecciona la fecha de inicio y fecha de finalización de la nueva franja
	3	El Tutor selecciona al menos un día de la semana y una hora
	4	El Tutor guarda los valores seleccionados
	5	Se crea una franja de tutoría por cada día de la semana y hora seleccionados, y las tutorías correspondientes a cada franja de tutorías
Extensiones	La fecha de inicio es posterior a la fecha de finalización	
	2	El Tutor selecciona una fecha de inicio posterior a la fecha de finalización seleccionada.
	5	Se indica al Tutor que la fecha de finalización ha de ser posterior o igual a la fecha de inicio
	Horas repetidas en un mismo día	
	3	El Tutor selecciona horas repetidas en un mismo día de la semana
	5	Se indica al Tutor que no pueden existir horas repetidas en un mismo día
	Algunos campos están vacíos	
	2 - 3	El Tutor rellena los campos, pero se deja alguno vacío
	5	Se indica al Tutor que no puede haber campos vacíos
	La franja de tutorías se solapa con otra ya existente	
	5	Se indica al Tutor que la franja de tutorías se solapa con otra ya existente

Tabla 1: Caso de uso 1 - Creación franjas de tutorías. Fuente: Propia

Caso de uso	C2 – Visualización franjas de tutorías	
Descripción	Un Tutor puede visualizar sus franjas de tutorías	
Actores	Tutor	
Flujo principal	1	El Tutor accede a la página de visualización de franjas de tutorías.
Extensiones	Visualización de franjas de tutorías filtradas	
	2	El Tutor filtra las franjas por fecha de inicio, fecha de finalización, hora de realización de la tutoría y/o día de la semana.
	3	El Tutor visualiza las franjas de tutorías filtradas

Tabla 2: Caso de uso 2 - Visualización franjas de tutorías. Fuente: Propia

Caso de uso	C3 – Edición franja de tutorías	
Descripción	Un Tutor puede editar sus franjas de tutorías	
Actores	Tutor	
Flujo principal	1	El Tutor accede a la página de visualización de franjas de tutorías
	2	El Tutor decide editar una franja de tutorías
	3	El Tutor puede editar los campos de fecha de inicio, fecha de finalización, hora de realización de la tutoría y día de la semana
	4	El Tutor guarda los valores seleccionados
	5	Se actualiza la franja de tutorías y se indica al Tutor que la franja ha sido actualizada con éxito
Extensiones	La fecha de inicio es posterior a la fecha de finalización	
	2	El Tutor selecciona una fecha de inicio posterior a la fecha de finalización seleccionada.
	5	Se indica al Tutor que la fecha de finalización ha de ser posterior o igual a la fecha de inicio
	La franja de tutorías se solapa con otra ya existente	
	5	Se indica al Tutor que la franja de tutorías se solapa con otra ya existente

Tabla 3: Caso de uso 3 - Edición franja de tutorías. Fuente: Propia

Caso de uso	C4 – Eliminación franja de tutorías	
Descripción	Un Tutor puede eliminar sus franjas de tutorías	
Actores	Tutor	
Flujo principal	1	El Tutor accede a la página de visualización de franjas de tutorías
	2	El Tutor decide eliminar una de sus franjas de tutorías
	3	El Tutor confirma la eliminación de la franja de tutorías
	4	La franja de tutorías es eliminada
Extensiones	La franja de tutorías tiene alguna tutoría reservada	
	4	Se le indica al Tutor que no se puede eliminar esta franja de tutorías, puesto que tiene alguna tutoría reservada

Tabla 4: Caso de uso 4 - Eliminación franja de tutorías. Fuente: Propia

5.2.3 Subsistema de tutorías

En cuanto al subsistema de tutorías, los actores involucrados en sus casos de uso son varios, a continuación, se detalla cuáles son y a que funcionalidades tienen acceso.

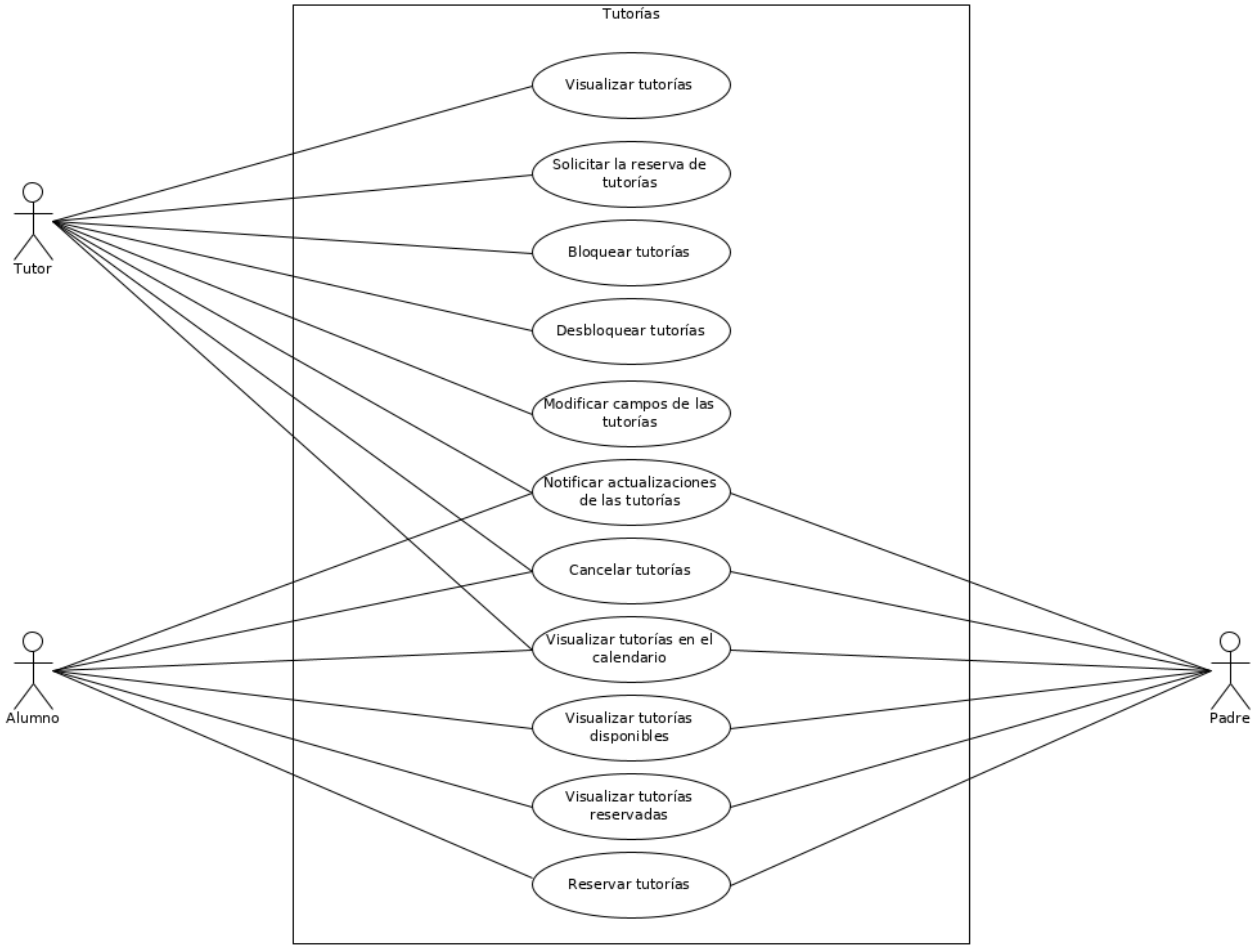


Figura 16: Diagrama de casos de uso del subsistema de tutorías. Fuente: VisualParadigm [21]

Caso de uso	C5 – Visualizar tutorías	
Descripción	Un Tutor puede visualizar sus tutorías	
Actores	Tutor	
Flujo principal	1	El Tutor accede a la página de visualización de tutorías.
Extensiones	Visualización de tutorías filtradas	
	2	El Tutor filtra las tutorías por rango de fechas en que estén las tutorías, estado de las tutorías y/o por nombre del alumno.
	3	El Tutor visualiza las tutorías filtradas

Tabla 5: Caso de uso 5 - Visualizar tutorías. Fuente: Propia

Caso de uso	C6 – Solicitar la reserva de tutorías	
Descripción	Un Tutor puede solicitar la reserva de tutorías a padres y alumnos	
Actores	Tutor	
Flujo principal	1	El Tutor accede a la página de visualización de tutorías
	2	El Tutor indica que quiere solicitar la reserva de tutorías a padres o alumnos
	3	El Tutor escoge los destinatarios y el contenido del mensaje
	4	El Tutor indica que se envíe el mensaje
	5	Se envía el mensaje a través de la aplicación con un link que redirecciona a la página de reserva de tutorías desde donde se puede iniciar la reserva de tutorías
Extensiones	El destinatario no tiene la aplicación móvil o ha configurado los ajustes para recibir los mensajes por correo electrónico siempre	
	6	Se envía un correo electrónico al usuario con el mensaje y un link que redirecciona a la página de reserva de tutorías tras iniciar sesión
	El destinatario tiene la aplicación móvil	
	7	Se envía una notificación *push* al usuario informando que tiene un mensaje nuevo, si el usuario presiona la notificación se abrirá la aplicación y será redireccionado al mensaje

Tabla 6: Caso de uso 6 - Solicitar la reserva de tutorías. Fuente: Propia

Caso de uso	C7 – Bloquear tutorías	
Descripción	Un Tutor puede bloquear sus tutorías libres	
Actores	Tutor	
Flujo principal	1	El Tutor accede a la página de visualización de tutorías
	2	El Tutor selecciona una tutoría libre
	3	El Tutor decide bloquear la tutoría
	4	La tutoría es bloqueada

Tabla 7: Caso de uso 7 - Bloquear tutorías. Fuente: Propia

Caso de uso	C8 – Desbloquear tutorías	
Descripción	Un Tutor puede desbloquear sus tutorías bloqueadas	
Actores	Tutor	
Flujo principal	1	El Tutor accede a la página de visualización de tutorías
	2	El Tutor selecciona una tutoría bloqueada
	3	El Tutor decide desbloquear la tutoría
	4	La tutoría es desbloqueada y pasa a estar disponible

Tabla 8: Caso de uso 8 - Desbloquear tutorías. Fuente: Propia

Caso de uso	C9 – Modificar campos de las tutorías	
Descripción	Un Tutor puede modificar los campos de sus tutorías	
Actores	Tutor	
Flujo principal	1	El Tutor accede a la página de visualización de tutorías
	2	El Tutor selecciona una tutoría reservada
	3	El Tutor visualiza los campos de la tutoría
	4	El Tutor edita la ubicación, las anotaciones privadas y/o las anotaciones públicas
	5	El Tutor guarda los cambios
	6	La tutoría es actualizada
Extensiones	Actualización del evento asociado cuando se modifican los campos de una tutoría reservada	
	7	El evento asociado a la tutoría es actualizado con los nuevos campos, que se muestran en la descripción del evento (ubicación y anotaciones públicas)

Tabla 9: Caso de uso 9 - Modificar campos de las tutorías. Fuente: Propia

Caso de uso	C10 – Notificar actualizaciones de las tutorías	
Descripción	Tutores, Alumnos y Padres son notificados de las actualizaciones en las tutorías que los involucran	
Actores	Tutor, Alumno, Padre	
Flujo principal	1	Una tutoría es actualizada por el Tutor o el solicitante
	2	Se envía un mensaje interno al Tutor y al solicitante, informando de la actualización realizada
	3	Si la tutoría es compartida con el resto de padres, también se informa de la actualización al resto de padres del alumno

Tabla 10: Caso de uso 10 - Notificar actualizaciones de las tutorías. Fuente: Propia

Caso de uso	C11 – Cancelar tutorías	
Descripción	Tutores, Alumnos y Padres pueden cancelar las tutorías que los involucran	
Actores	Tutor, Alumno, Padre	
Flujo principal	1	El usuario accede a la página de visualización de tutorías
	2	El usuario selecciona una tutoría reservada
	3	El usuario decide cancelar la tutoría
	4	La tutoría es cancelada y pasa a estar disponible

Tabla 11: Caso de uso 11 - Cancelar tutorías. Fuente: Propia

Caso de uso	C12 – Visualizar tutorías en el calendario	
Descripción	Tutores, Alumnos y Padres pueden ver las tutorías reservadas que los involucran en el calendario	
Actores	Tutor, Alumno, Padre	
Flujo principal	1	Una tutoría es reservada
	2	Se crea un evento en el calendario con la información de la tutoría, asociado al Tutor y al solicitante
	3	Si la tutoría es compartida con el resto de padres, también se asocia el evento al resto de padres

Tabla 12: Caso de uso 12 - Visualizar tutorías en el calendario. Fuente: Propia

Caso de uso	C13 – Visualizar tutorías disponibles	
Descripción	Los Alumnos pueden visualizar las tutorías disponibles de sus Tutores y los Padres pueden visualizar las tutorías disponibles de los Tutores de sus hijos	
Actores	Alumno, Padre	
Flujo principal	1	El usuario accede a la página de visualización de tutorías.
	2	El usuario quiere ver las tutorías disponibles, por lo que presiona el botón de reservar tutorías
	3	El usuario ve las tutorías disponibles de su Tutor
Extensiones	El usuario accede como Padre y tiene más de un hijo disponible	
	2.1	El usuario selecciona el hijo para el que quiere reservar tutorías
	El usuario es estudiante y tiene más de un Tutor disponible o el usuario es Padre y el hijo seleccionado tiene más de un Tutor disponible	
	2.2	El usuario selecciona el Tutor cuyas tutorías disponibles quiere ver
	Visualización de tutorías filtradas	
	4	El usuario filtra las tutorías disponibles por rango de fechas en que estén las tutorías, día de la semana, a partir de una cierta hora y/o hasta una cierta hora.
	5	El usuario visualiza las tutorías filtradas

Tabla 13: Caso de uso 13 - Visualizar tutorías disponibles. Fuente: Propias

Caso de uso	C14 – Visualizar tutorías reservadas	
Descripción	Alumnos y Padres pueden visualizar las tutorías reservadas que los involucran	
Actores	Alumno, Padre	
Flujo principal	1	El usuario accede a la página de visualización de tutorías reservadas.
Extensiones	Visualización de tutorías filtradas	
	2	El usuario filtra las tutorías por rango de fechas
	3	El usuario visualiza las tutorías filtradas

Tabla 14: Caso de uso 14 - Visualizar tutorías reservadas. Fuente: Propia

Caso de uso	C15 – Reservar tutorías	
Descripción	Los Alumnos pueden reservar las tutorías disponibles de sus Tutores y los Padres pueden reservar las tutorías disponibles de los Tutores de sus hijos	
Actores	Alumno, Padre	
Flujo principal	1	El usuario accede a la página de visualización de tutorías disponibles del Tutor.
	2	El usuario selecciona la tutoría que quiere reservar
	3	El usuario confirma que quiere reservar la tutoría seleccionada
Extensiones	El usuario es Padre, y el hijo seleccionado previamente tiene más de un padre	
	2.1	El usuario indica si quiere compartir la tutoría con el resto de padres del alumno o quiere que sea individual

Tabla 15: Caso de uso 15 - Reservar tutorías. Fuente: Propia

5.2.4 Subsistema de pagos

Finalmente, a las funcionalidades del subsistema de pagos solo tienen acceso los Tesoreros, y a continuación se explica en qué consisten.

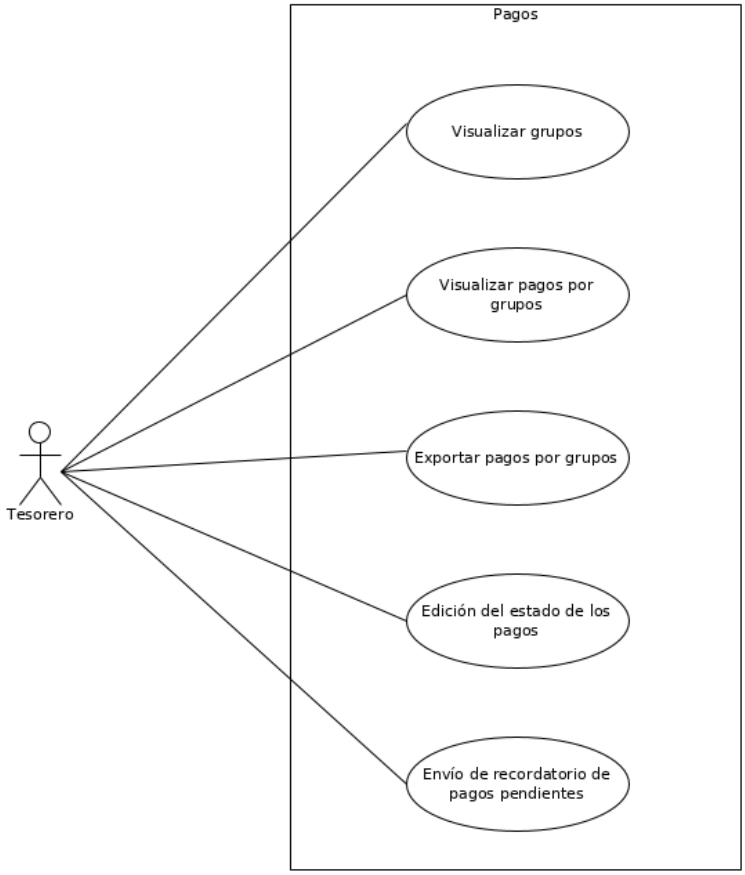


Figura 17: Diagrama de casos de uso del subsistema de pagos. Fuente: VisualParadigm [21]

Caso de uso	C16 – Visualizar grupos	
Descripción	El Tesorero del centro puede visualizar los grupos del centro y si estos tienen pagos pendientes en un rango de fechas	
Actores	Tesorero	
Flujo principal	1	El Tesorero accede a la página del módulo de pagos de visualización de grupos
	2	El Tesorero visualiza los grupos y si estos tienen pagos pendientes en el rango establecido por defecto (Últimos 30 días)
Extensiones	Visualización de grupos filtrados	
	4	El Tesorero modifica el rango de fechas en el que se comprueba si hay pagos pendientes, los grupos que se quieren tener en cuenta y/o si quiere que solo aparezcan los grupos con pagos pendientes o que aparezcan todos
	5	El Tesorero visualiza los grupos filtrados y si estos tienen pagos pendientes

Tabla 16: Caso de uso 16 - Visualizar pagos. Fuente: Propia

Caso de uso	C17 – Visualizar pagos por grupos	
Descripción	El Tesorero del centro puede visualizar los pagos de los grupos del centro y la información respecto al pago de cada alumno del grupo, así como el total ingresado y el total pendiente de cada alumno	
Actores	Tesorero	
Flujo principal	1	El Tesorero accede a la página de visualización de grupos del módulo de pagos y selecciona un grupo
	2	El Tesorero visualiza los pagos del grupo dentro del rango de fechas establecido, la información respecto al pago de cada alumno del grupo, así como el total ingresado y el total pendiente de cada alumno
Extensiones	Visualización de pagos por grupos filtrados	
	2	El Tesorero modifica el rango de fechas en el que se buscan los pagos del grupo, los grupos que se quieren tener en cuenta y/u oculta o vuelve a mostrar los pagos que aparecen y su información correspondiente
	3	El Tesorero visualiza los pagos por grupos filtrados

Tabla 17: Caso de uso 17 - Visualizar pagos por grupos. Fuente: Propia

Caso de uso	C18 – Exportar pagos por grupos	
Descripción	El Tesorero del centro puede exportar los pagos por grupos	
Actores	Tesorero	
Flujo principal	1	El Tesorero accede a la página de visualización de pagos por grupos y establece los filtros que considere oportunos
	2	El Tesorero decide exportar la información mostrada
	3	Se exporta la información de pagos por grupos que se mostraba por pantalla a fichero excel y se inicia la descarga del documento

Tabla 18: Caso de uso 18 - Exportar pagos por grupos. Fuente: Propia

Caso de uso	C19 – Edición del estado de los pagos	
Descripción	El Tesorero del centro puede editar el estado de los pagos de los alumnos de un grupo	
Actores	Tesorero	
Flujo principal	1	El Tesorero accede a la página de visualización de pagos por grupos
	2	El Tesorero selecciona un pago de un alumno
	3	Se muestra la información del pago
	4	El Tesorero decide cambiar el estado del pago a uno de los estados disponibles
	5	Se realiza el cambio de estado
Extensiones	El cambio de estado es de Pendiente a Pagado manualmente	
	4.1	El Tesorero escribe un comentario para explicar por qué este pago está pagado manualmente o deja el campo en blanco
	4.2	El Tesorero confirma que el pago pase a estado Pagado manualmente
Información extra	Cambios de estado posibles: Pendiente ► Pagado manualmente Pendiente ► Descartado Pagado manualmente ► Pendiente Fallido ► Pendiente Descartado ► Pendiente	

Tabla 19: Caso de uso 19 - Edición del estado de los pagos. Fuente: Propia

Caso de uso	C20 – Envío de recordatorio de pagos pendientes	
Descripción	El Tesorero del centro puede enviar un recordatorio de pago a un alumno de un grupo con un pago pendiente	
Actores	Tesorero	
Flujo principal	1	El Tesorero accede a la página de visualización de pagos por grupos
	2	El Tesorero selecciona un pago de un alumno que está en estado pendiente
	3	Se muestra la información del pago
	4	El Tesorero decide enviar una notificación de que el pago aún está pendiente
	5	Se envía dicha notificación al alumno y/o a sus padres según la configuración establecida

Tabla 20: Caso de uso 20 - Envío de recordatorio de pagos pendientes. Fuente: Propia

6 Diseño

Una vez explicado cómo ha de ser la aplicación tanto en requisitos como en objetivos, pasamos a establecer como está estará construida, es decir, cuál ser la arquitectura del sistema, que diseño seguirán la base de datos y el software, y cuál será la interfaz de usuario.

6.1 Arquitectura del sistema

Como se ha explicado anteriormente, actualmente Dinantia está compuesta por dos aplicaciones, una aplicación móvil y una aplicación web, y como, desde hace un año, uno de los clientes más importantes de Dinantia es la Once, una institución que favorece la plena inclusión escolar y social del alumnado con ceguera o deficiencia visual grave. Esto supone que ambas aplicaciones han de cumplir los requisitos de accesibilidad AA de las Directrices de Accesibilidad para el Contenido Web [8], permitiendo que la aplicación sea accesible y utilizable por el máximo número de personas, independientemente de sus conocimientos, capacidades personales o las características técnicas del dispositivo de acceso empleado.

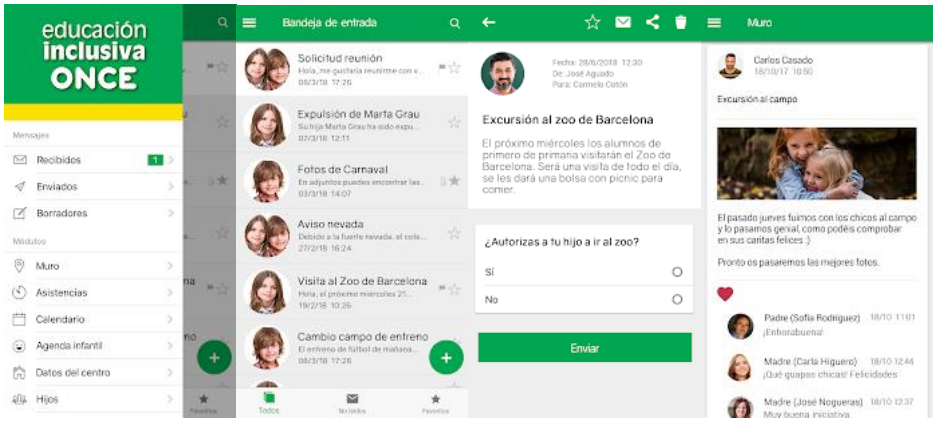


Figura 18: Aplicación móvil Dinantia Once. Fuente: Google Play [9]

La aplicación móvil se desarrolla utilizando Angular + Ionic [20] [21], un kit de desarrollo *software* (SDK) completo de código abierto para el desarrollo de aplicaciones móviles híbridas que permiten desarrollar aplicaciones para Android y IOS de la misma forma que desarrollarías una aplicación web. Gracias a la facilidad de adaptación de Ionic, la aplicación móvil ya está adaptada y cumple los requisitos de accesibilidad AA.

En cuanto a la versión web, actualmente existen dos entornos web diferentes que coexistirán en Dinantia como mínimo hasta el verano de este año 2021, cuando está planificado que acabe la migración del entorno antiguo al nuevo. Ambos entornos están basados en CakePHP, pero el entorno nuevo tiene incorporado Vue.js que permitirá cumplir los requisitos de accesibilidad de cara a la interacción del usuario con la web. Este fue el motivo que supuso la construcción del entorno nuevo con las mismas funcionalidades que el entorno viejo, pero garantizando la accesibilidad, y actualmente se está realizando la migración de todas las funcionalidades y módulos al nuevo entorno para que finalmente este sea el entorno utilizado por todos los clientes.

Actualmente este nuevo entorno no está completo, se están migrando funcionalidades que ya están en el viejo entorno y está siendo utilizado únicamente por La Once, mientras que todos los otros clientes siguen trabajando con el entorno antiguo que no garantiza la accesibilidad, pero contiene todas las funcionalidades y módulos de la aplicación. El objetivo de la empresa es para el verano de 2021 acabar la migración completa de todas las funcionalidades del entorno viejo al entorno nuevo para así poder garantizar la accesibilidad y que todos los clientes utilicen un único entorno, la cual cosa facilitará el desarrollo de la aplicación.

Por este motivo, la parte web correspondiente a la fase 1 del desarrollo se realizará inicialmente en la web antigua, de forma que pueda salir rápidamente al mercado y esté disponible para los clientes lo más pronto posible, puesto que las funcionalidades de la fase 1 habían sido específicamente solicitadas, y en la fase 2 se migrarán las funcionalidades de la fase 1 a la web nueva, y se continuará el desarrollo en el nuevo entorno, que será utilizado únicamente por la Once hasta el verano de 2021.

La siguiente es una representación de la arquitectura seguida en Dinantia:

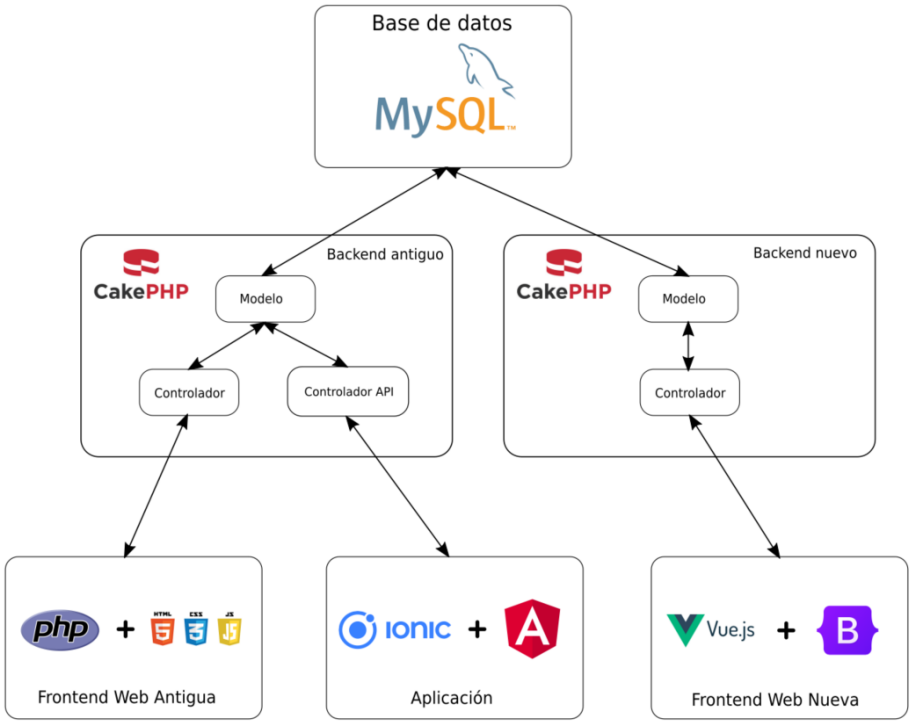


Figura 19: Diagrama de la arquitectura actual de Dinantia. Fuente: Propia

6.2 Patrones de diseño

A continuación, se explicarán en detalle los patrones principales se han utilizado en el desarrollo del sistema y dónde son utilizados.

6.2.1 Patrón Modelo-Vista-Controlador (MVC)

Modelo-Vista-Controlador (MVC) [22] es un patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado, define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

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

- El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la vista aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al modelo a través del controlador.

- El Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su vista asociada si se solicita un cambio en la forma en que se presenta el modelo (por ejemplo, desplazamiento o *scroll* por un documento o por los diferentes registros de una base de datos), por tanto, se podría decir que el controlador hace de intermediario entre la vista y el modelo.

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

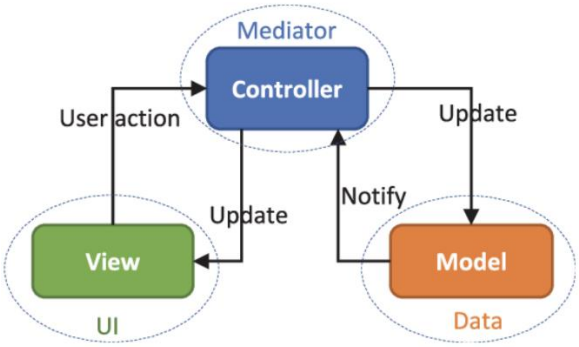


Figura 20: Patrón Modelo-Vista-Controlador. Fuente: Wikipedia [22]

Este patrón es el utilizado en la arquitectura base de la aplicación, la cual está basada en CakePHP.

6.2.2 Patrón Modelo-Vista-Modelo de vista (MVVM)

Modelo-Vista-Modelo de vista (MVVM) [23] es un patrón de arquitectura de software que facilita la separación del desarrollo de la interfaz gráfica de usuario (la vista), ya sea a través de un lenguaje de marcado o código GUI, del desarrollo de la lógica del *backend* (el modelo) para que la vista no dependa de ninguna plataforma de modelo específica. El modelo de vista de MVVM es un convertidor de valor, lo que significa que el modelo de vista es responsable de exponer (convertir) los objetos del modelo de tal manera que los objetos se administran y presentan fácilmente. En este sentido, el modelo de vista es más modelo que vista, y maneja la mayoría, si no toda, la lógica de visualización de la vista. El modelo de vista puede implementar un patrón de mediador, organizando el acceso a la lógica de *backend* en torno al conjunto de casos de uso admitidos por la vista.

De forma genérica, los componentes de MVVM se podrían definir de la siguiente manera:

- Modelo: El modelo se refiere a un modelo de dominio, que representa contenido de estado real (un enfoque orientado a objetos), o a la capa de acceso a datos, que representa contenido (un enfoque centrado en datos).
- Vista: Como en los patrones modelo-vista-controlador (MVC) y modelo-vista-presentador (MVP), la vista es la estructura, diseño y apariencia de lo que un usuario ve en la pantalla. Muestra una representación del modelo y recibe la interacción del usuario con la vista (clics del mouse, entrada de teclado, gestos de toque de pantalla, etc.), y reenvía el manejo de estos al modelo de vista a través del enlace de datos (propiedades, devoluciones de llamada de eventos, etc.) que se define para vincular la vista y el modelo de vista.
- Modelo de vista: El modelo de vista es una abstracción de la vista que expone las propiedades y los comandos públicos. En lugar del controlador del patrón MVC, o el presentador del patrón MVP, MVVM tiene una carpeta, que automatiza la comunicación entre la vista y sus propiedades enlazadas en el modelo de vista. El modelo de vista se ha descrito como un estado de los datos en el modelo

  La principal diferencia entre el modelo de vista y el Presentador en el patrón MVP es que el presentador tiene una referencia a una vista, mientras que el modelo de vista no. En cambio, una vista se vincula directamente a las propiedades del modelo de vista para enviar y recibir actualizaciones. Para funcionar de manera eficiente, esto requiere una tecnología de vinculación o generar un código repetitivo para realizar la vinculación.

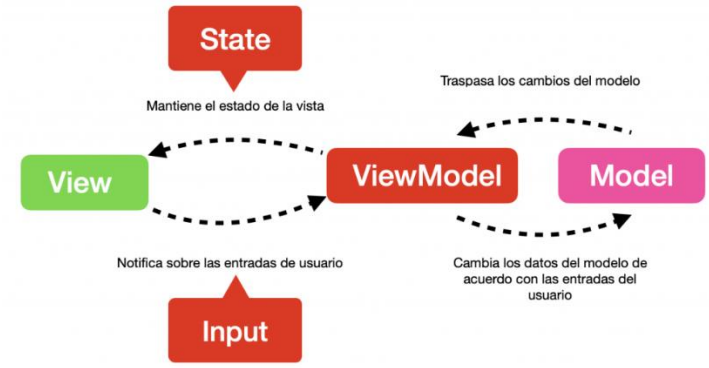


Figura 21: Patrón Modelo-Vista-Modelo de vista. Fuente: Wikipedia [23]

Este patrón es empleado en el *frontend* de la web nueva, donde se utiliza Vue.js, un *framework* que utiliza este patrón.

6.2.3 API REST

Una API [24] es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita del consumidor (la llamada) y el que requiere el productor (la respuesta). En otras palabras, cuando se desea interactuar con una computadora o un sistema para obtener datos o ejecutar una función, las API permiten comunicar lo que desea al sistema, para que este comprenda la solicitud y la cumpla.

Por otro lado, el término REST (*Representational State Transfer*) se refiere a un conjunto de restricciones a tener en cuenta en la arquitectura software para crear aplicaciones web respetando HTTP. Concretamente, las restricciones que definen a un sistema RESTful son:

- Cliente-servidor: El servidor se encarga de controlar los datos mientras que el cliente se encarga de manejar las interacciones del usuario. Esta restricción mantiene al cliente y al servidor débilmente acoplados, el cliente no necesita conocer los detalles de implementación del servidor y el servidor no necesita saber cómo son usados los datos que envía al cliente.
- Sin estado: Cada petición que recibe el servidor debe ser independiente y contener todo lo necesario para ser procesada.
- Cacheable: Debe admitir un sistema de almacenamiento en caché. Este almacenamiento evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso.
- Interfaz uniforme: Define una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme, lo cual simplifica y separa la arquitectura. Esta restricción indica que cada recurso del servicio REST debe tener una única dirección o URI (*Uniform Resource Identifier*).
- Sistema de capas: El servidor puede disponer de varias capas en su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad del sistema.

Hoy en día API REST es considerado un estándar lógico y eficiente, y es utilizado por la mayoría de las empresas para crear servicios web.

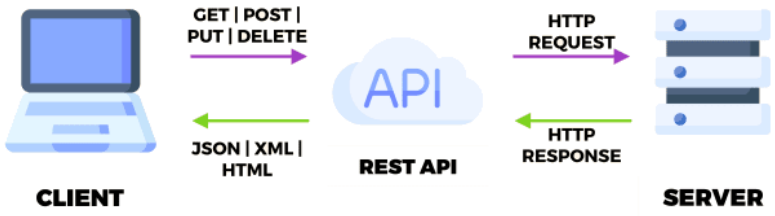


Figura 22: Patrón API REST. Fuente: API REST [24]

Este es el patrón utilizado en las llamadas entre *frontend* de la web y *backend,* y en las llamadas entre la aplicación y el controlador API.

6.2.4 Patrón observador

El patrón observador [25] es un patrón de diseño utilizado en programación para observar el estado de un objeto. Está relacionado con el principio de invocación implícita.

Este patrón se utiliza principalmente para implementar sistemas de tratamiento de eventos distribuidos. En algunos lenguajes de programación, los problemas tratados por este patrón, son tratados en la sintaxis de tratamiento de eventos nativa. Esta es una funcionalidad muy interesante en términos de desarrollo de aplicaciones en tiempo real.

La esencia de este patrón es que uno o más objetos (llamados observadores) son registrados (o se registran ellos mismos) para observar un evento que puede ser lanzado por el objeto observado (el sujeto). El objeto que puede lanzar un evento generalmente mantiene una colección de observadores.

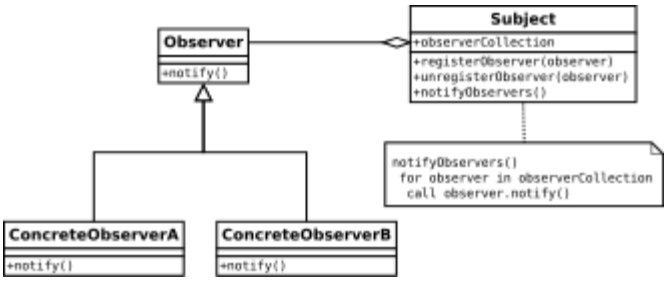


Figura 23: Patrón observador. Fuente: Wikipedia [25]

Este es el patrón utilizado en *frontend* de la web para generar respuestas a los eventos que suceden en los diferentes elementos de la página.

6.3 Diseño de la interfaz

Como se ha mencionado anteriormente, en cuanto a la interfaz, se seguirá el estilo y los criterios marcados por la aplicación en el resto de módulos, los cuales ya están enfocados a facilitar la usabilidad haciendo que la aplicación sea sencilla de utilizar, mostrando menús y elementos intuitivos que faciliten a los usuarios la navegación y el uso de la aplicación. Además, en la aplicación móvil y en la web nueva, la interfaz también está enfocada a garantizar una accesibilidad de nivel AA, de manera que todas las personas puedan acceder a la aplicación en igualdad de condiciones.

A continuación, se muestran los diseños de interfaz seguidos en los diferentes entornos:

6.3.1 Web antigua

Estos son algunos ejemplos de la interfaz seguida en la web antigua, donde esencialmente tenemos 3 páginas de tutorías: visualización de franjas de tutorías, creación de franjas de tutorías y visualización de tutorías, a las que solo pueden acceder los tutores; y, 2 páginas de pagos por grupos: visualización de grupos y visualización de pagos por grupos, a las que solo pueden acceder los tesoreros. Todas con diversas funcionalidades, filtros y modales. Estas páginas contienen todas las características establecidas para la fase 1.

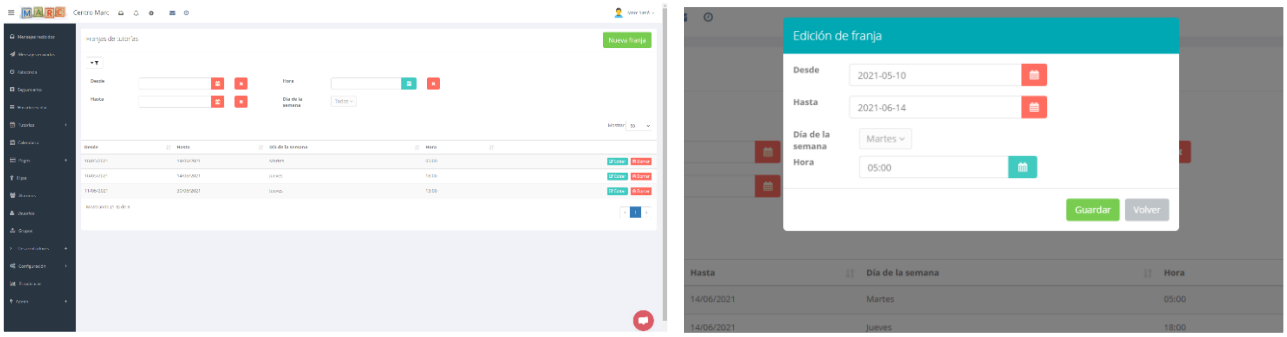


Figura 24: Capturas de pantalla de Dinantia (web antigua): Visualización franjas de tutorías. Fuente: Dinantia [1]

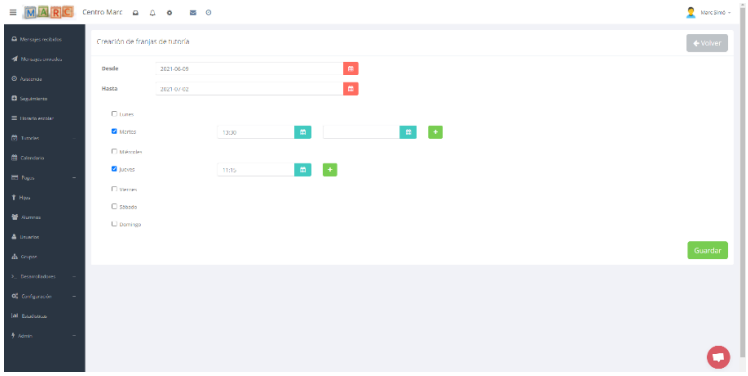


Figura 25: Capturas de pantalla de Dinantia (web antigua): Creación franjas de tutorías. Fuente: Dinantia [1]

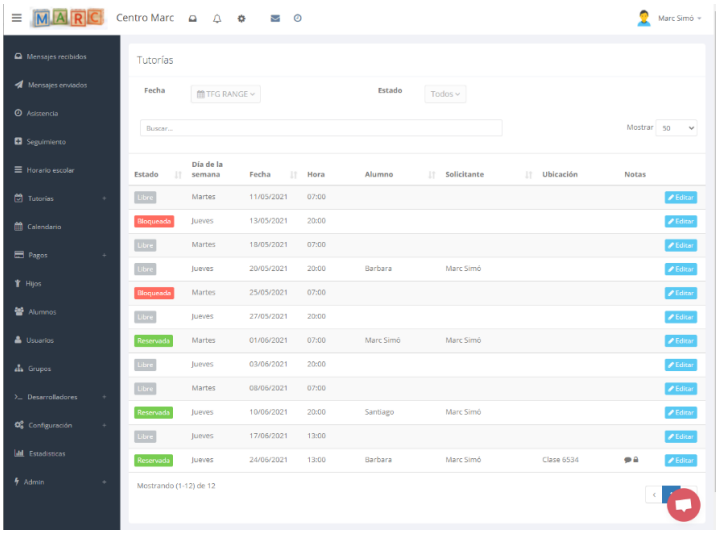


Figura 26: Capturas de pantalla de Dinantia (web antigua): Visualización tutorías. Fuente: Dinantia [1]

Figura 27: Capturas de pantalla de Dinantia (web antigua): Modales de bloqueo y desbloqueo de tutorías. Fuente: Dinantia [1]

Figura 28: Capturas de pantalla de Dinantia (web antigua): Modal de edición tutoría reservada. Fuente: Dinantia [1]

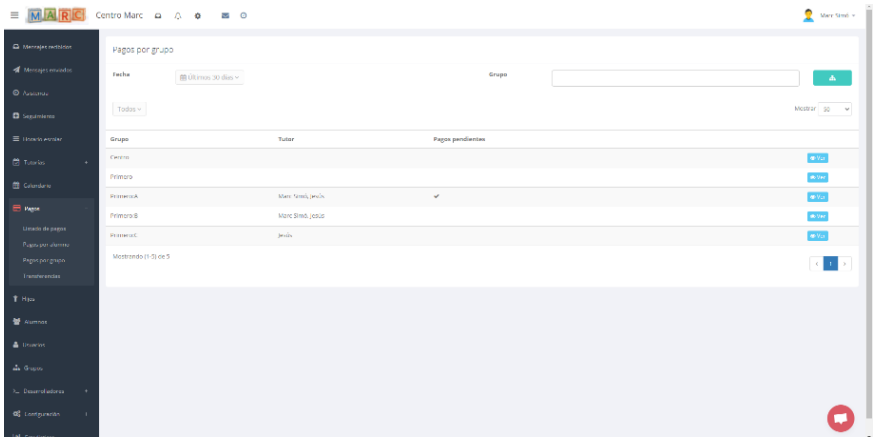


Figura 29: Capturas de pantalla de Dinantia (web antigua): Visualización grupos. Fuente: Dinantia [1]

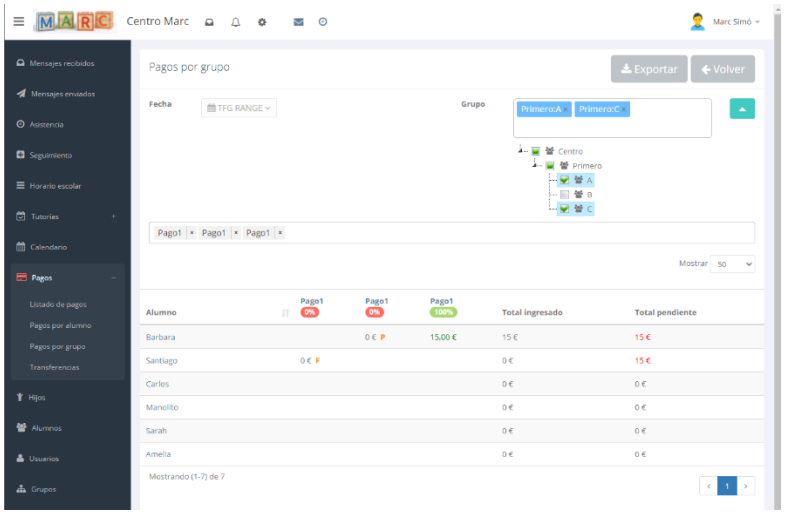


Figura 30: Capturas de pantalla de Dinantia (web antigua): Visualización pagos por grupos. Fuente: Dinantia [1]

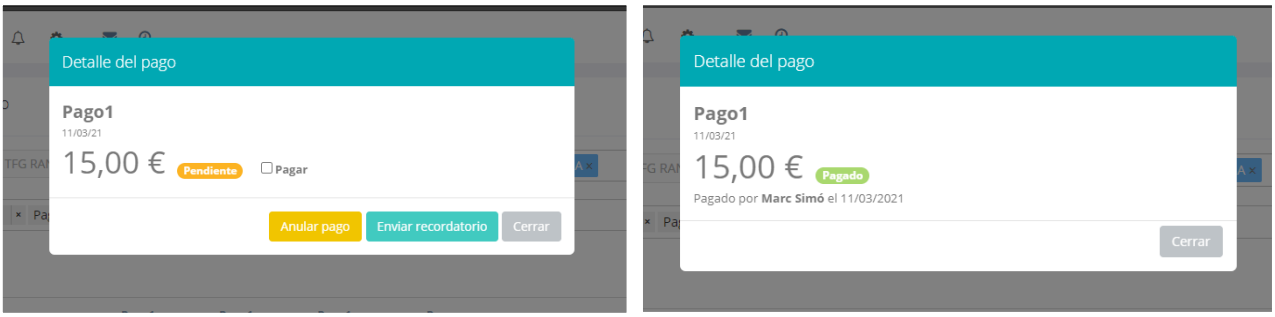
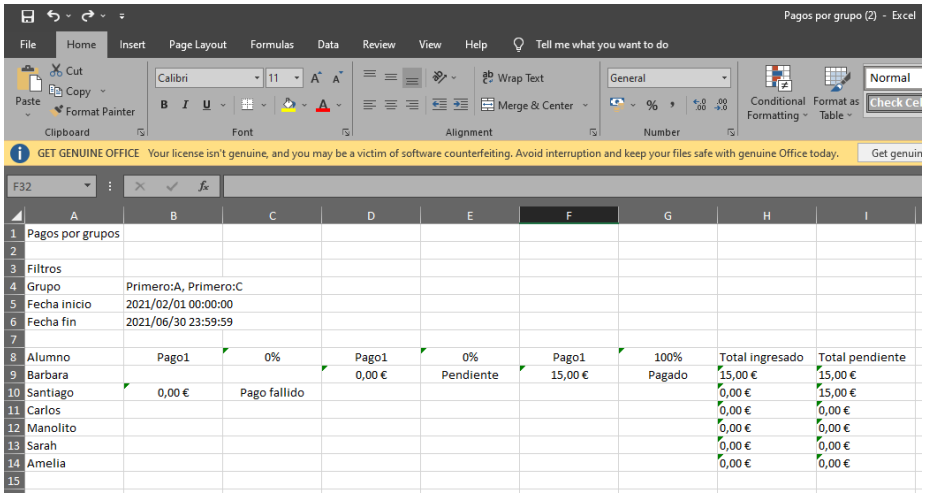


Figura 31: Capturas de pantalla de Dinantia (web antigua): Modales de edición de pagos. Fuente: Dinantia [1]



	A	B	C	D	E	F	G	H	I
1	Pagos por grupos								
2									
3	Filtros								
4	Grupo	Primero:A, Primero:C							
5	Fecha inicio	2021/02/01 00:00:00							
6	Fecha fin	2021/06/30 23:59:59							
7									
8	Alumno	Pago1	0%	Pago1	0%	Pago1	100%	Total ingresado	Total pendiente
9	Barbara			0,00 €	Pendiente	15,00 €	Pagado	15,00 €	15,00 €
10	Santiago	0,00 €	Pago fallido					0,00 €	15,00 €
11	Carlos							0,00 €	0,00 €
12	Manolito							0,00 €	0,00 €
13	Sarah							0,00 €	0,00 €
14	Amelia							0,00 €	0,00 €
15									

Figura 32: Capturas de pantalla de Dinantia (web antigua): Fichero de pagos por grupos exportado. Fuente: Dinantia [1]

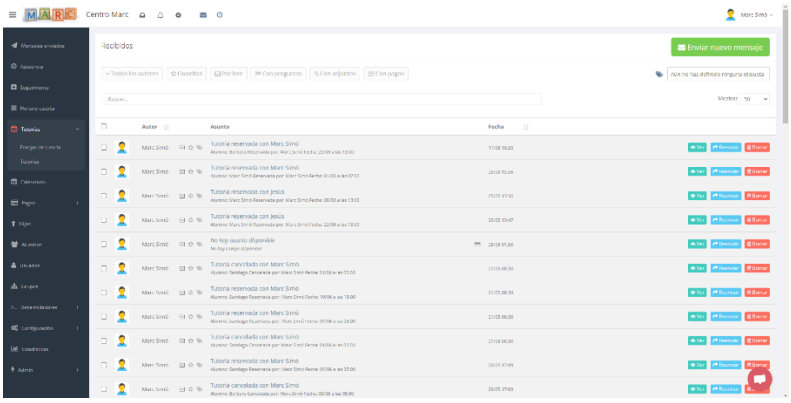


Figura 33: Capturas de pantalla de Dinantia (web antigua): Visualización mensajes. Fuente: Dinantia [1]

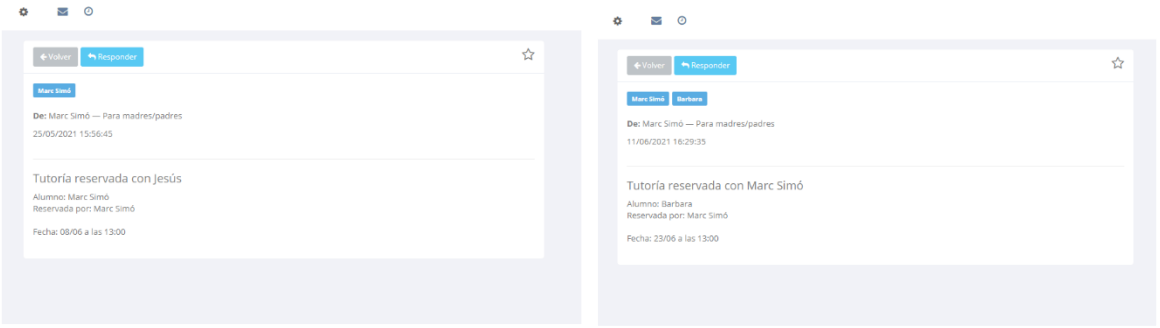


Figura 34: Capturas de pantalla de Dinantia (web antigua): Mensajes de notificación de actualización de tutorías. Fuente: Dinantia [1]

6.3.2 Web nueva

En la web nueva se han migrado todas las funcionalidades desarrolladas en la web antigua, con una apariencia muy similar, pero con elementos que garantizan la accesibilidad, mediante un mayor contraste, un correcto movimiento del foco, títulos y descripciones ARIA que facilitan el uso de la página mediante lectores para invidentes, y otras funcionalidades. Aparte de la migración de páginas antiguas, también hay 2 nuevas páginas: la de ver tutorías reservadas y la de reservar tutorías, a las que solo tendrán acceso padres y alumnos; y algunas funcionalidades nuevas como la de ver tutorías en el calendario y poder compartir las tutorías.

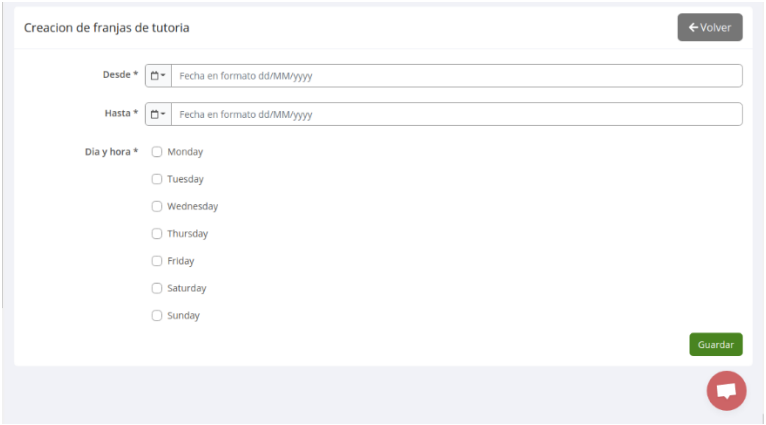


Figura 35: Capturas de pantalla de Dinantia (web nueva): Creación franjas de tutorías. Fuente: Dinantia [1]

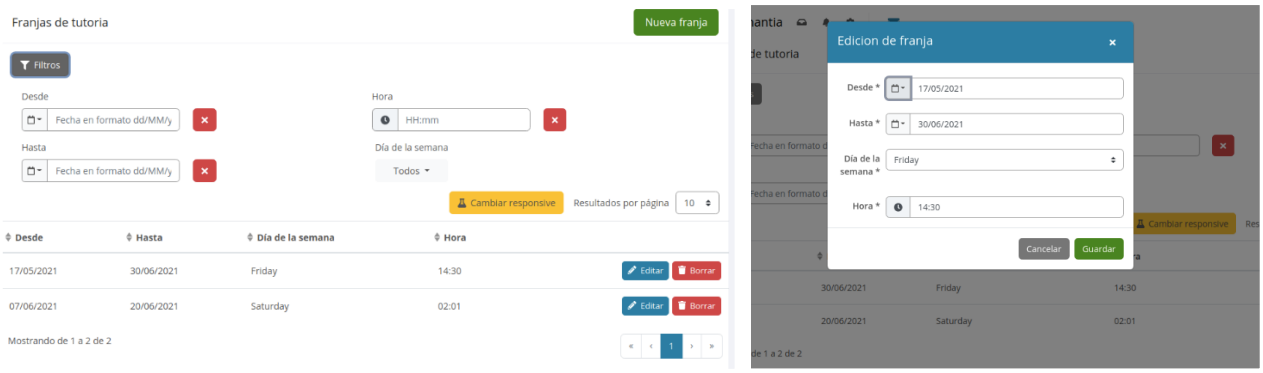


Figura 36: Capturas de pantalla de Dinantia (web nueva): Visualización y edición franjas de tutorías. Fuente: Dinantia [1]

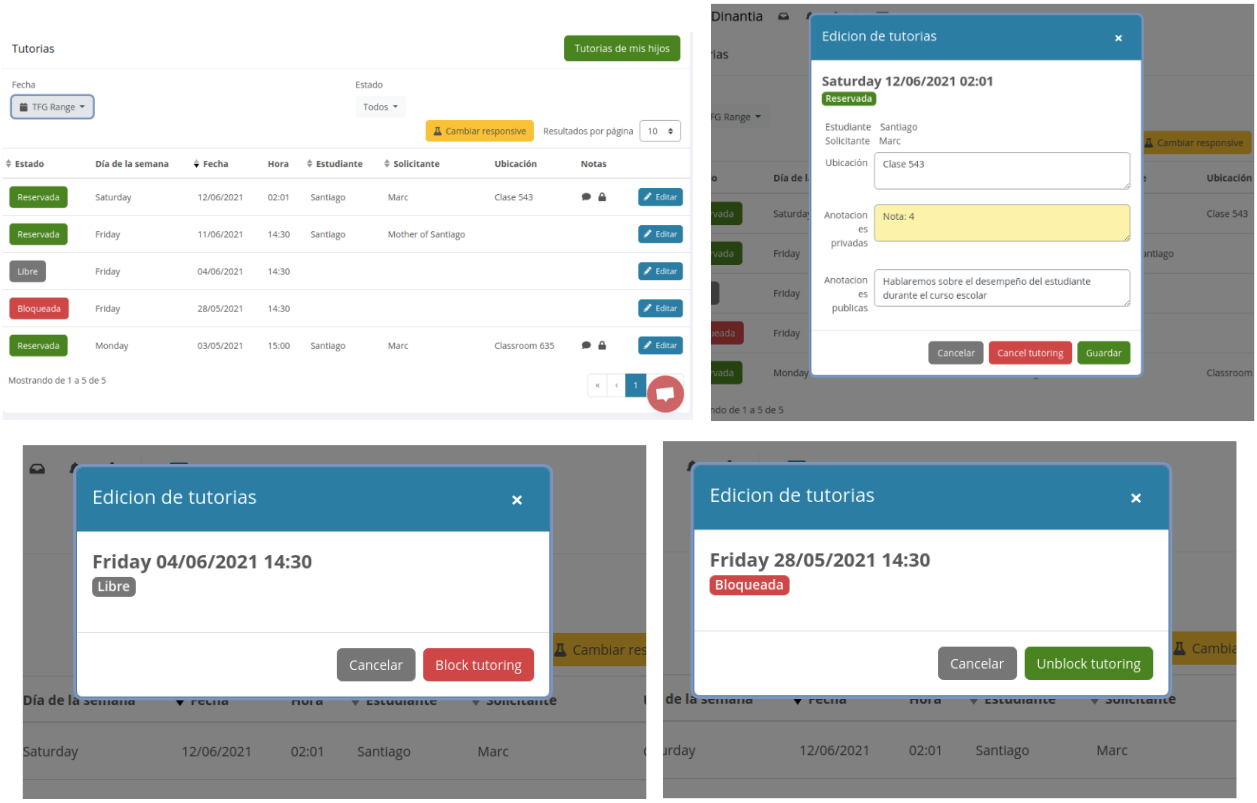


Figura 37: Capturas de pantalla de Dinantia (web nueva): Visualización y edición tutorías del tutor. Fuente: Dinantia [1]

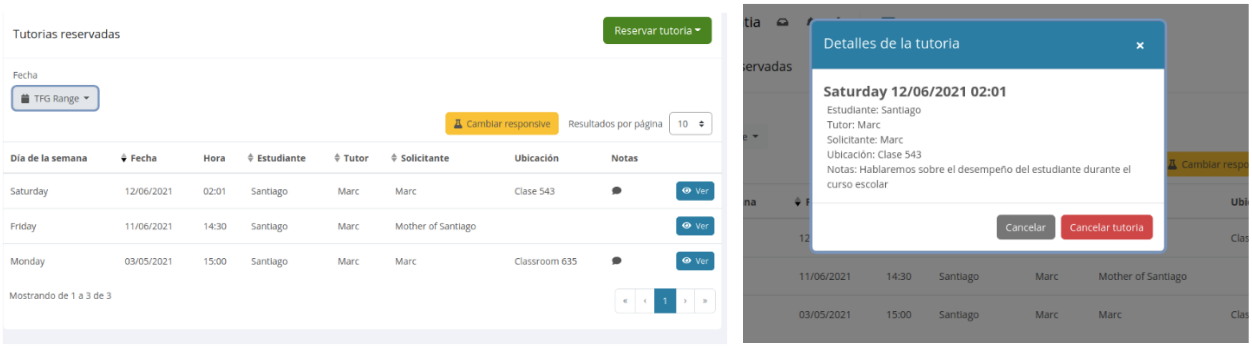


Figura 38: Capturas de pantalla de Dinantia (web nueva): Visualización tutorías reservadas. Fuente: Dinantia [1]

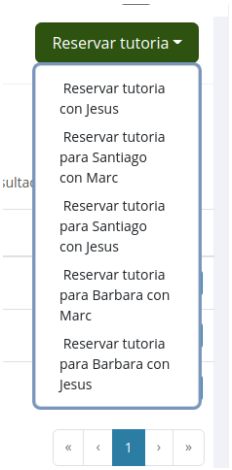


Figura 39: Capturas de pantalla de Dinantia (web nueva): Modal selección de hijo y tutor para reservar tutoría. Fuente: Dinantia [1]

Figura 40: Capturas de pantalla de Dinantia (web nueva): Reserva de tutorías y Modal de notificación sobre la reserva. Fuente: Dinantia [1]

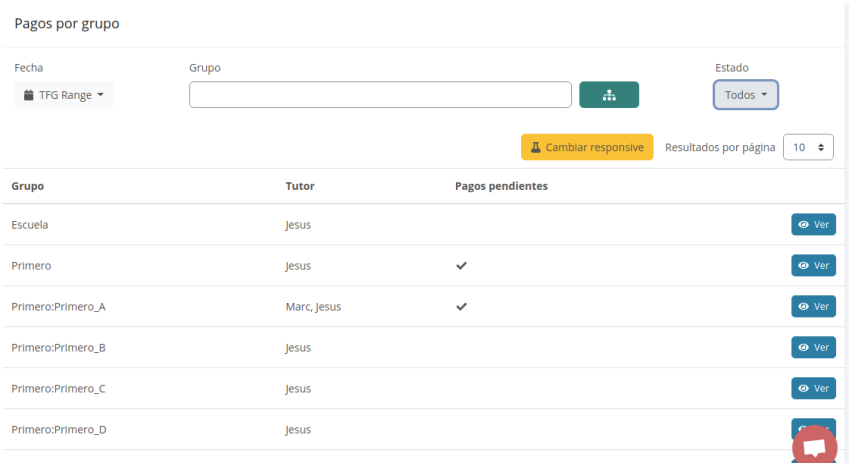


Figura 41: Capturas de pantalla de Dinantia (web nueva): Visualización grupos. Fuente: Dinantia [1]

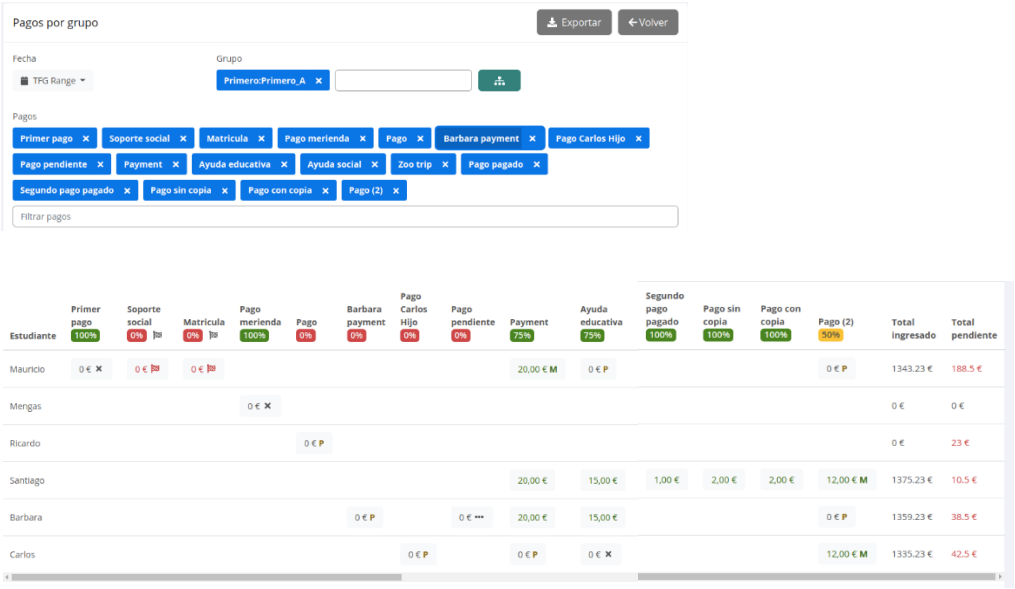


Figura 42: Capturas de pantalla de Dinantia (web nueva): Visualización pagos por grupos. Fuente: Dinantia [1]

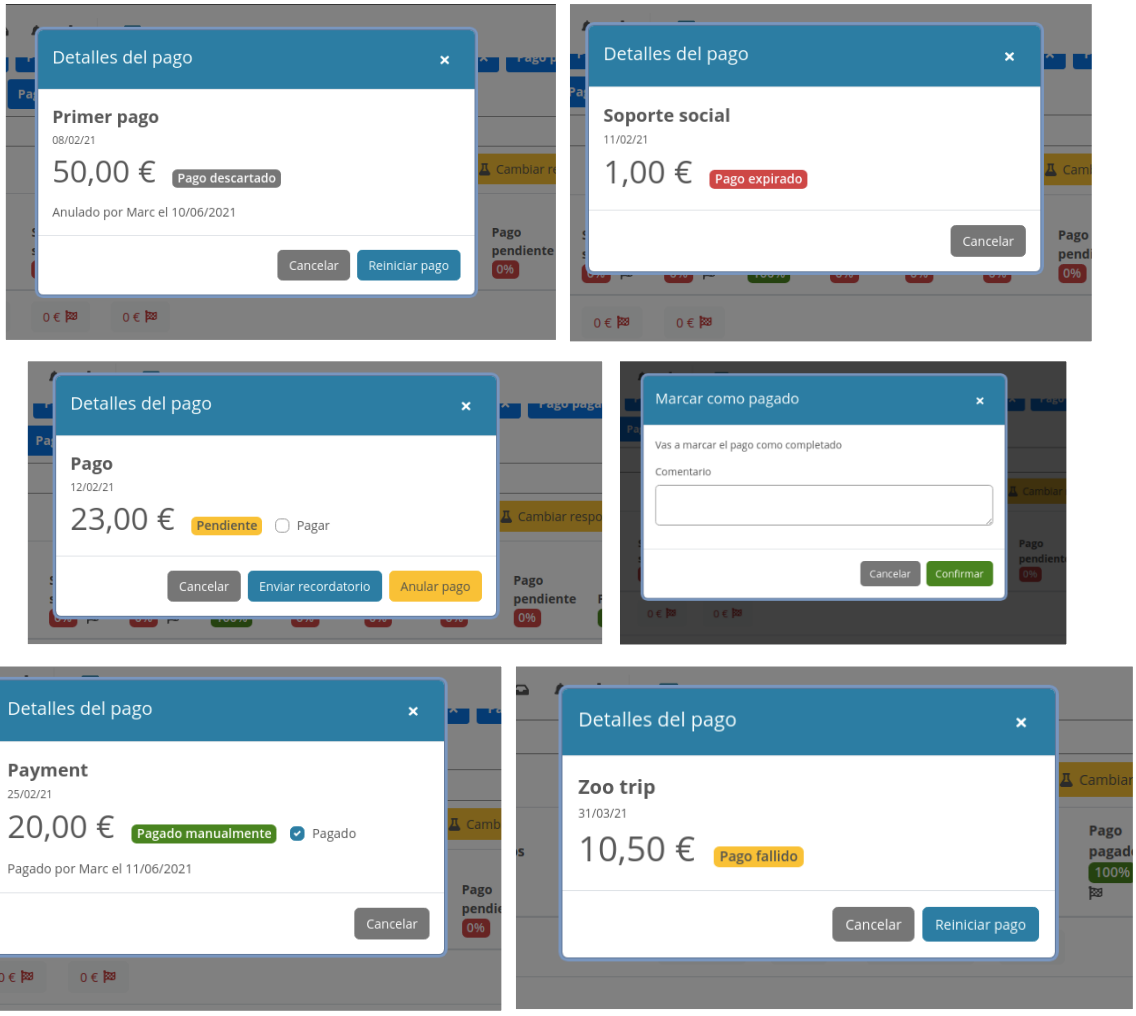


Figura 43: Capturas de pantalla de Dinantia (web nueva): Modales de edición del estado de los pagos. Fuente: Dinantia [1]

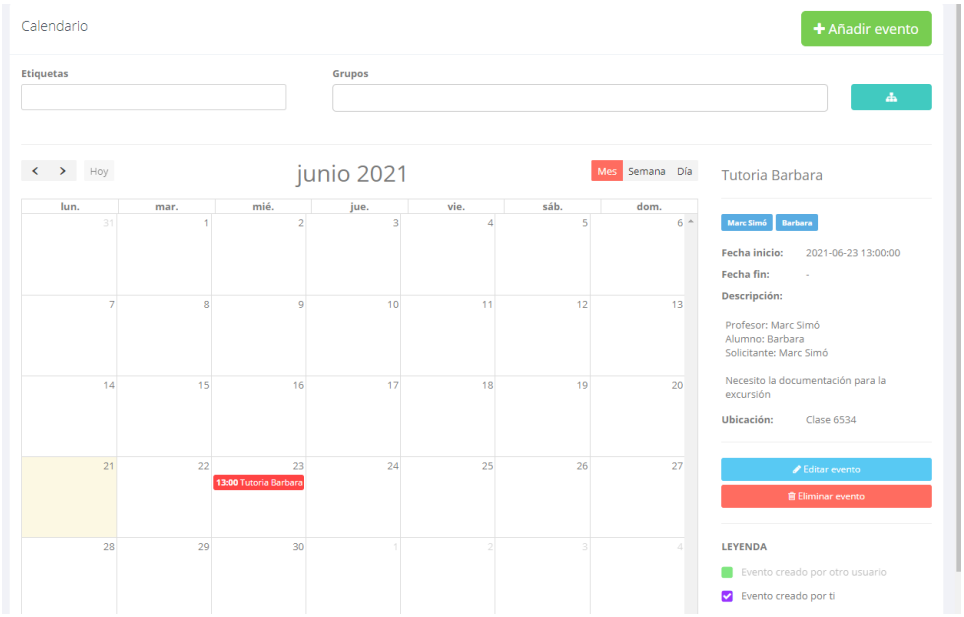


Figura 44: Capturas de pantalla de Dinantia (web nueva): Visualización tutorías en el calendario. Fuente: Dinantia [1]

6.3.3 Aplicación móvil

En el caso de la aplicación móvil, también se garantiza una accesibilidad AA, y se incluyen prácticamente todas las funcionalidades del subsistema de tutorías.

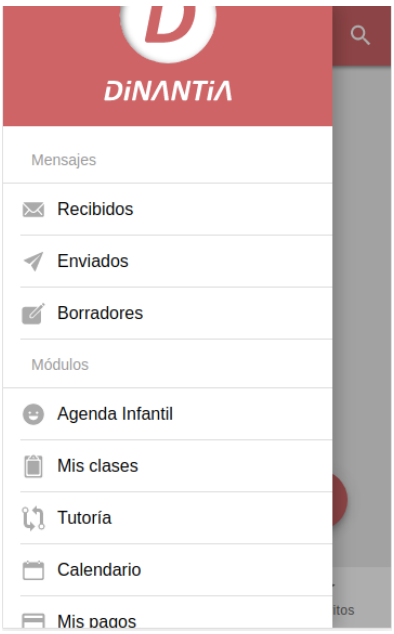


Figura 45: Capturas de pantalla de Dinantia (app): Barra lateral. Fuente: Dinantia [1]

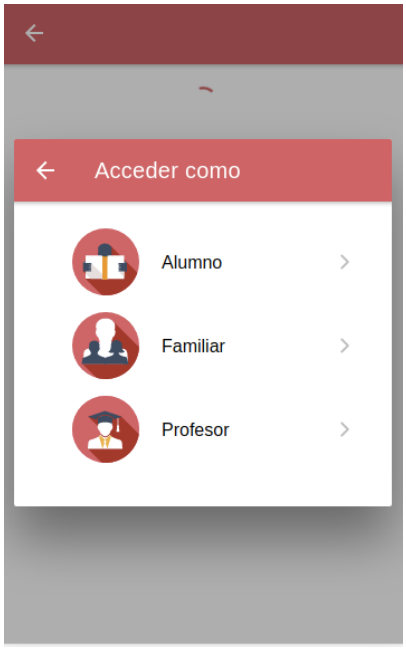


Figura 46: Capturas de pantalla de Dinantia (app): Modal de selección de rol (En caso de tener más de uno). Fuente: Dinantia [1]

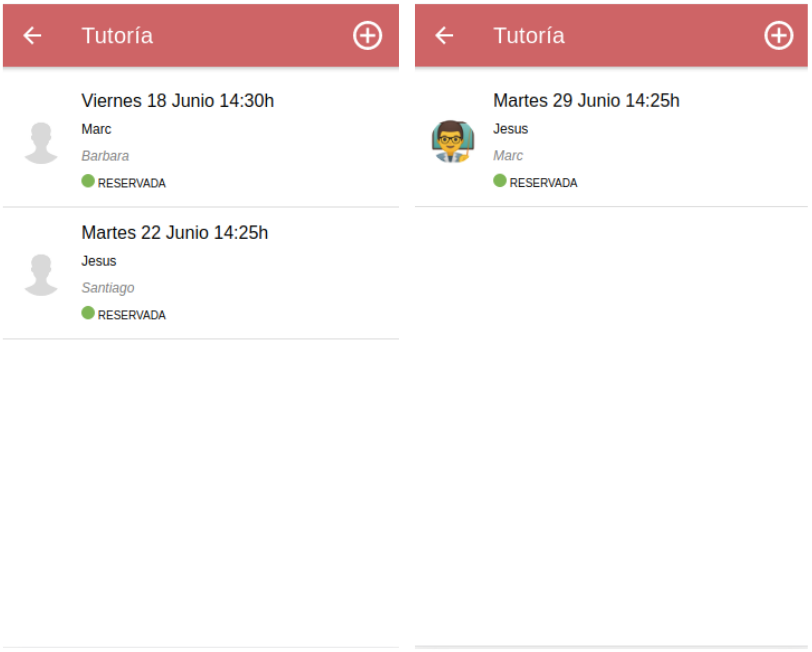


Figura 47: Capturas de pantalla de Dinantia (app): Visualización de tutorías reservadas. Fuente: Dinantia [1]

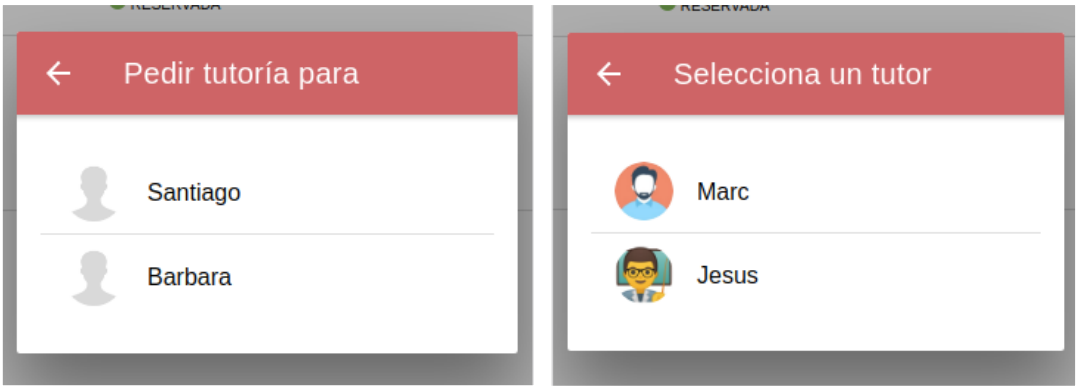


Figura 48: Capturas de pantalla de Dinantia (app): Modal de selección de hijo y tutor para la reserva de tutorías. Fuente: Dinantia [1]

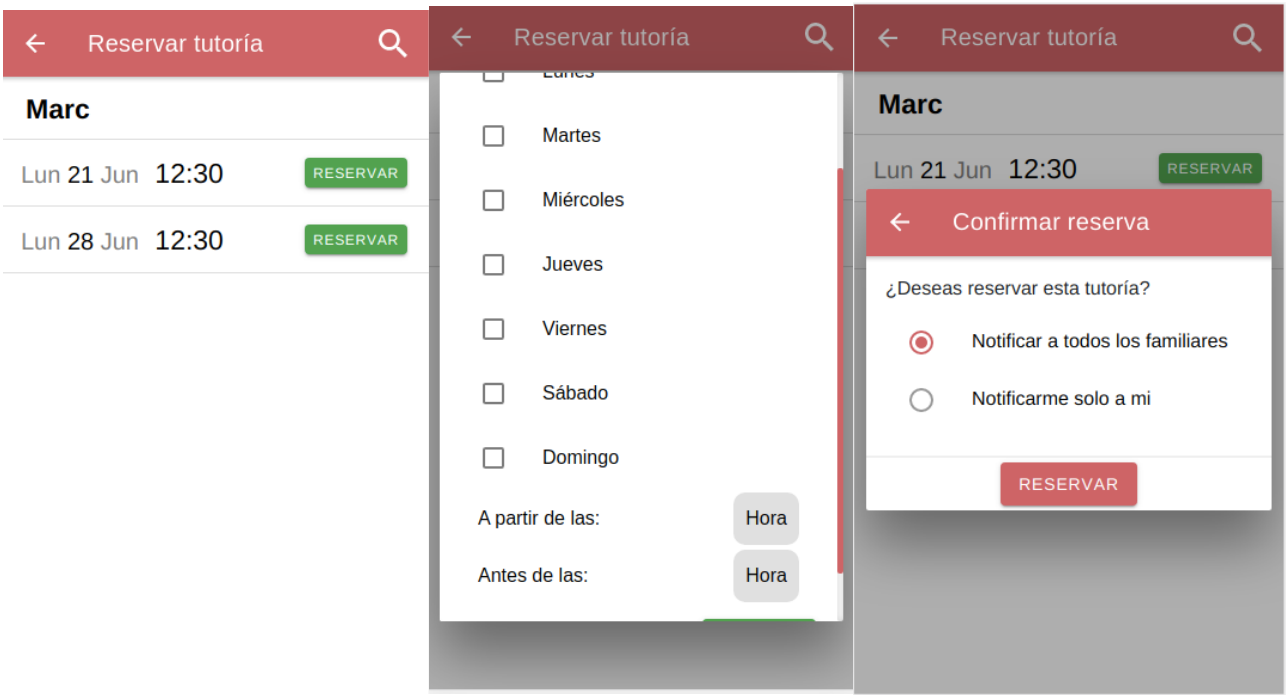


Figura 49: Capturas de pantalla de Dinantia (app): Página de reserva de tutorías y Modales de filtro y notificación. Fuente: Dinantia [1]

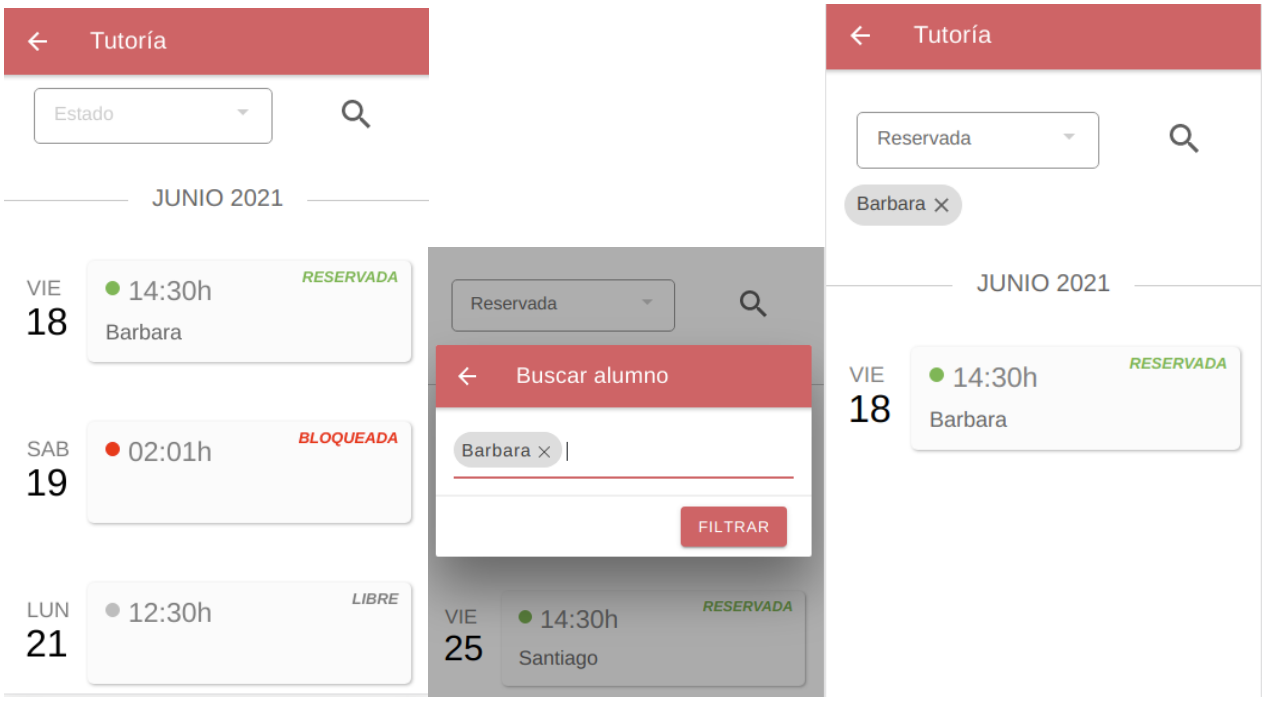


Figura 50: Capturas de pantalla de Dinantia (app): Visualización y filtrado de tutorías. Fuente: Dinantia [1]

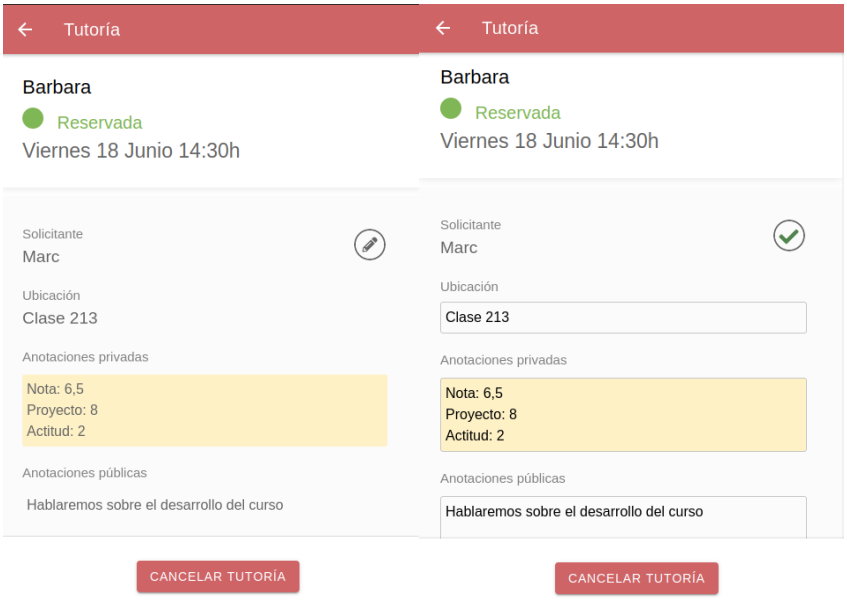


Figura 51: Capturas de pantalla de Dinantia (app): Visualización y edición tutoría. Fuente: Dinantia [1]

Figura 52: Capturas de pantalla de Dinantia (app): Visualización tutoría (Alumno y Padre). Fuente: Dinantia [1]

7 Implementación

Una vez establecido como será el sistema, es decir, que requisitos, objetivos y alcance tendrá, y el diseño que será utilizado, el siguiente paso es hablar en detalle de la implementación del sistema, explicando las tecnologías y herramientas utilizadas durante el desarrollo.

7.1 Tecnologías utilizadas

Durante implementación del proyecto se han utilizado una gran cantidad de tecnologías diferentes, con el objetivo de adecuarse a las necesidades de cada sistema. A continuación, se explicarán en detalle las diferentes tecnologías utilizadas.

7.1.1 Lenguajes de programación

a) PHP

PHP (*Hypertext Preprocessor*) [26] es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML, el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente original.

Concretamente, en el proyecto, debido al *framework* utilizado para gestionar la lógica del servidor (CakePHP), se utiliza PHP para generar el *front* en la web antigua, y para gestionar la lógica en todo el servidor.

Figura 53: Logo PHP. Fuente: PHP [26]

b) HTML

El Lenguaje de Marcado de Hipertexto (HTML) [27] es el código que se utiliza para estructurar y desplegar una página web y sus contenidos. HTML nos provee etiquetas para describir los diferentes tipos de contenidos (elementos) de nuestra web. Gracias a ello, el navegador podrá comprender el contenido enviado por el servidor y representarlo en pantalla.

Es utilizado junto a CSS/SCSS y Javascript/Typescript en las páginas tanto de la app como de la web.

Figura 54: Logo HTML. Fuente: Wikipedia [28]

c) CSS

CSS (*Cascading Style Sheets*) [27] es un lenguaje de hojas de estilo, es decir, te permite aplicar estilos de manera selectiva a elementos en documentos HTML. Con CSS asignamos fuentes y color a textos o cajas, modificamos tamaños, añadimos imágenes de fondo, definimos márgenes o incluso podemos cambiar completamente la apariencia de un elemento HTML como una lista para convertirla en una barra o menú de navegación.

Gracias a CSS también podemos hacer que nuestra página web se vea correctamente en otros dispositivos como móviles o tabletas. Es lo que se conoce como diseño web adaptativo o *responsive*, y es utilizado en todas las páginas.

Figura 55: Logo CSS. Fuente: SeekLogo [29]

d) SCSS

Las hojas de estilo a medida que van creciendo se van transformando en archivos complejos y difíciles de mantener. Los preprocesadores vienen al rescate, entregándonos características que nos permiten escribir estilos de manera amigable y estructurada.

Un preprocesador CSS es una herramienta que nos permite generar, de manera automática, hojas de estilo, añadiéndoles características que no tiene CSS, y que son propias de los lenguajes de programación, como pueden ser variables, funciones, selectores anidados, herencia, etc.

Estas características de los procesadores nos permiten que el CSS generado sea más fácil de mantener y más reutilizable. Concretamente, en este proyecto utilizaremos Sass [30], el lenguaje de extensión CSS más maduro, estable y potente actualmente, utilizando la extensión más popular y utilizada de este, conocida como SCSS (*Sassy* CSS), es muy similar a la sintaxis nativa de CSS, tanto así que nos permite importar hojas de estilos CSS (copiar y pegar) directamente en un archivo SCSS y obtener un resultado válido.

Figura 56: Logo Sass. Fuente: SeekLogo [29]

e) Javascript

JavaScript [27] es un lenguaje de programación del lado del cliente que permite implementar dinamismo y funcionalidad a nuestra página web. Además del contenido estático ofrecido por HTML y la estética mejorada con CSS, con JavaScript podremos:

- Mostrar actualizaciones de contenido.
- Vincular eventos dinámicos a elementos HTML (clic en botones, accesos a menús, filtros en formularios…).
- Almacenar datos en variables.
- Usar funciones complementarias como gráficos o mapas mediante APIS y librerías de terceros.
- Acceder a conjuntos de datos públicos o privados.
- Muchas otras funcionalidades que mejoran la forma de programar e interactuar con la página web.

Figura 57: Logo Javascript. Fuente: SeekLogo [29]

f) Typescript

TypeScript [31] es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. TypeScript es usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor, o extensiones para programas.

TypeScript extiende la sintaxis de JavaScript, por tanto, cualquier código JavaScript existente debería funcionar sin problemas en TypeScript. Está pensado para grandes proyectos, los cuales a través de un compilador de TypeScript se traducen a código JavaScript original. Es utilizado en la web nueva y en la aplicación, puesto que permite generar código más marcado, estructurado y sencillo de depurar.

Figura 58: Logo Typescript. Fuente: SeekLogo [29]

7.1.2 Frameworks

Los *frameworks* son plantillas o esquemas, que simplifican la elaboración de una tarea, ya que solo es necesario complementarlo de acuerdo a lo que se quiere realizar. En el caso de la programación, la mayoría de *frameworks* más completos suelen ofrecer una estructura base con una serie de funcionalidades y librerías que permiten realizar un proyecto de forma sencilla, centrándote únicamente en las adiciones específicas de tu proyecto y no aquella base que todos los proyectos de este estilo han de tener. A continuación, se explican los *frameworks* utilizados, sus características y el motivo de su uso.

a) CakePHP

CakePHP [32] es un *framework* de desarrollo rápido para PHP, libre y de código abierto, se trata de una estructura que sirve de base a los programadores para que se puedan crear aplicaciones Web, estableciendo el núcleo de la aplicación y aportando una gran cantidad de funcionalidades que facilitan al programador trabajar de forma estructurada y rápida, sin pérdida de flexibilidad, permitiendo centrarse en la lógica específica de la aplicación.

Las aplicaciones CakePHP siguen el patrón de diseño de software MVC (Modelo-Vista-Controlador), separando la aplicación en tres partes principales. El modelo representa los datos de la aplicación, la vista hace una presentación del modelo de datos, y el controlador maneja y enruta las peticiones hechas por los usuarios.

Se escogió utilizar el *framework* CakePHP en el servidor porque proporciona una base robusta y muy completa para el sistema, siendo capaz de manejar cualquier aspecto, desde la solicitud inicial del usuario hasta el renderizado final de la página web. Además, como el *framework* sigue los principios MVC, puedes fácilmente personalizar y extender muchos aspectos de la aplicación.

El *framework* también proporciona una estructura de organización básica, desde los nombres de los archivos hasta los de las tablas de la base de datos, manteniendo toda la aplicación consistente y lógica. Este aspecto es simple pero poderoso, puesto que si se siguen las convenciones siempre se sabrá donde esta cada cosa y como está organizada, y cualquier desarrollador que conozca CakePHP será capaza de añadirse al proyecto entendiendo más fácilmente el funcionamiento de la aplicación.

CakePHP proporciona clases para el Controlador, Modelo y la Vista, pero adicionalmente incluye algunas clases más para mejorar el tratamiento a cada capa y hacer el desarrollo más sencillo y rápido. Componentes, Comportamientos y *Helpers* son clases que proveen de extensibilidad y reusabilidad a las clases base del MVC.

Además, CakePHP te permite la generación de la base de datos a partir de migraciones, facilitando el desarrollo en grupo mediante git, puesto que generas las migraciones, y las ejecutas para generar las tablas, de forma que una vez subidas las migraciones al repositorio, para que el resto de desarrolladores actualicen su base de datos a la más actual, simplemente deben obtener la última versión de la rama y ejecutar las migraciones, actualizando así la base de datos, modificando las tablas, atributos y especificaciones definidas en las migraciones.

También facilita la creación de APIs REST, al ofrecer todas las herramientas para su funcionamiento, facilitando la creación de controladores REST que permitan interactuar con la capa lógica de la aplicación a través de llamadas a la API y que se pueden generar de forma muy sencilla.

Otra gran ventaja de CakePHP, es el testeo, CakePHP viene con un completo soporte de testeo integrado. Concretamente, CakePHP viene con integración para PHPUnit y además de las funciones que ofrece PHPUnit, CakePHP ofrece algunas funciones adicionales para facilitar las pruebas.

Como ha quedado claro, CakePHP es un *framework* muy completo y tiene una gran cantidad de características que facilitan el desarrollo de aplicaciones web, convirtiéndolo en un *framework* ideal para el desarrollo de un servidor que tendrá funcionalidades muy diversas como el nuestro.

Figura 59: Logo CakePHP. Fuente: SeekLogo [29]

b) Vue.js

Como su nombre indica Vue.js [33] es un *framework* javascript, es decir, un conjunto de herramientas y funciones que permiten desarrollar páginas web de una manera más cómoda. Vue nace con la necesidad de no tener que escribir tanto código javascript y sobre todo con la idea de ahorrar tiempo al programador.

VueJS es una librería javascript pensada para tener un *framework* con el que desarrollar páginas web. Con Vue puedes crear todas las vistas de tu página web, puedes hacerlas dinámicas, puedes conectarla a un servidor para tener datos dinámicos de una base de datos, etc. En definitiva, Vue es un *framework* completo pensado para los programadores web y que se puede usar en todo tipo de webs.

Además, Vue sigue una estructura de MVVM (Modelo-Vista-Model de vista) obteniendo todos los beneficios que utilizarla supone, y que se han explicado anteriormente.

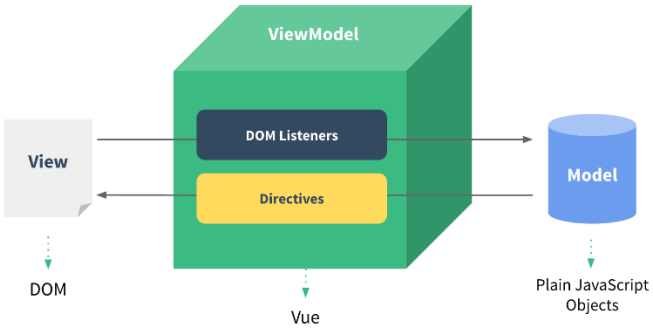


Figura 60: Vue Modelo-Vista-Modelo de vista. Fuente: Vue.js [34]

Características de Vue

Vue ofrece todo un conjunto de características y funcionalidades que permiten crear aplicaciones web completas, el conjunto de todas estas es el motivo que nos ha llevado a la implementación de Vue.js en la nueva web. Como es un *framework*, va más allá de simplemente ofrecer funciones y utilidades para Javascript, es una nueva forma de programar páginas web, así que vamos a ver lo que puede ofrecer.

- Modularidad: Lo primero que salta a la vista de Vue es que es completamente modular. A diferencia de otros *frameworks* que está pensados para ofrecerte todo lo que puedas necesitar, aunque no lo llegues a utilizar, en Vue tienes más libertad para usar ciertas características, Vue ofrece lo básico para que puedas elegir si instalar a posteriori las utilidades que necesites.
- Reactividad: Vue tiene propiedades reactivas, eso quiere decir que, si cambia una variable en una parte de la vista de la página, Vue actualizará su nuevo valor sin necesidad de que lo hagas manualmente, facilitando la programación al no tener que implementar *listeners* constantemente.
- Componentes web: Vue se basa en componentes web. Un componente web es una parte de una web que puede ser reutilizada y que normalmente tiene estilos y una funcionalidad aislada. Los componentes web básicamente, permiten crear tus propias etiquetas HTML personalizables, es decir, imagina que necesitas crear un calendario y que lo vas a usar en varias vistas de la web. Creando un calendario en forma de componente web, puedes crear calendarios simplemente llamando a la etiqueta que tú mismo definas para el HTML. Esto permite una gran reutilización de código.
- Virtual DOM: Si hay que hacer un cambio en la vista, en lugar de sustituir directamente los nuevos valores en la vista, Vue creará una especie de réplica del DOM, es decir, de los elementos de la página web, para que a la hora de hacer cambios en la vista se hagan de forma más óptima.
- Eventos: Permite reaccionar a eventos que se producen en el DOM de forma más sencilla, por ejemplo, cuando el usuario hace clic en un elemento, cuando lo mueve, cuando escribe, etc.
- Transiciones y animaciones: El sistema de transiciones y animaciones de Vue es muy sencillo de usar y a la vez muy completo. Las animaciones son las que ya se conocen de CSS por lo que no es necesario aprender sistemas nuevos.
- *Mixins*: Los *mixins* son funciones y lógica de los componentes que se pueden reutilizar y reusar en otros componentes web. Con los *mixins* puedes compartir cualquier clase de opción que le puedes pasar a los componentes
- Accesibilidad: Vue ofrece una serie de herramientas para poder gestionar partes importantes de la accesibilidad como gestionar el foco, el ARIA (*Accessible Rich Internet Application*) y el título de los elementos, entre otras funcionalidades. Además, la reusabilidad permite que sea más sencillo el manejo de la accesibilidad de los elementos, puesto que puedes crear tus propias etiquetas a partir del elemento original, y añadirle ciertas propiedades que pueden ser incluidas en el momento de la declaración y otras automáticas para convertirlo en accesible y facilitar su uso en la aplicación.

En conclusión, la decisión de utilizar Vue en la nueva página web fue sencilla, puesto que además de ofrecerte las herramientas necesarias para que una página web sea accesible, la facilidad de reutilización de código hace que sea sencillo definir tus propios elementos accesibles y utilizarlos a lo largo de la página, además de facilitar la programación en general.

Figura 61: Logo Vue. Fuente: SeekLogo [29]

c) Angular

Angular [35] es un *framework* para aplicaciones web desarrollado en TypeScript, de código abierto y mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

Características de Angular

- Velocidad y rendimiento
    - Generación de código: Angular convierte tus plantillas en código altamente optimizado para las máquinas virtuales de JavaScript de hoy en día, ofreciéndote todas las ventajas del código escrito a mano con la productividad de un *framework*.
    - Universal: Ejecuta la primera vista de tu aplicación en node.js, .NET, PHP, y otros servidores para renderizado de forma casi instantánea obteniendo solo HTML y CSS. También abre posibilidades para la optimización del SEO del sitio, incluyendo configuración.
    - División del código: Las aplicaciones de Angular se cargan rápidamente gracias al enrutador de componentes. Este ofrece una división automática de códigos para que los usuarios solo carguen el código necesario para procesar la vista que solicitan.
- Productividad
    - Plantillas: Permite crear rápidamente vistas de interfaz de usuario con una sintaxis de plantilla simple y potente.
    - Angular CLI: Las herramientas de línea de comandos permiten empezar a desarrollar rápidamente, añadir componentes y realizar test, así como previsualizar de forma instantánea la aplicación.
    - IDEs: Obtén sugerencias de código inteligente, detección de errores y otros comentarios en la mayoría de los editores populares e IDEs.
- Historia completa del desarrollo
    - *Testing*: Utiliza Karma para realizar pruebas unitarias, y Protractor para realizar pruebas de extremo a extremo de forma rápida, estable y eficaz.
    - Animación: Permite crear animaciones complejas y de alto rendimiento con muy poco código a través de la intuitiva API de Angular.
    - Accesibilidad: Posee características para crear aplicaciones accesibles con los componentes disponibles para ARIA.

Figura 62: Logo Angular. Fuente: SeekLogo [29]

Como podemos ver Angular es un *framework* muy completo, pero se decidió utilizar CakePHP en la web, puesto que las herramientas que ofrece se adecuan más a lo que necesitamos en el servidor de este proyecto y nos permite garantizar la accesibilidad en el *front* al combinarlo con Vue. Sin embargo, muchas de estas cualidades serían muy útiles en una aplicación, facilitando enormemente el desarrollo y permitiendo la generación de una aplicación accesible, que es uno de los requisitos del proyecto. Pero para que esto sea posible necesitamos combinarlo con el siguiente *framework* que vamos a explicar, Ionic, que nos permitirá desarrollar una aplicación híbrida que se podrá utilizar tanto en IOS como en Android, estará basada en Angular y garantizará un nivel de accesibilidad AA.

d) Ionic

Ionic Framework [36] es un SDK de *frontend* de código abierto para desarrollar aplicaciones híbridas basado en tecnologías web (HTML, CSS y JS). Es decir, un *framework* que nos permite desarrollar aplicaciones para iOS nativo, Android y la web, desde una única base de código. Su compatibilidad y, la implementación de Cordova y Ionic Native, hacen posible trabajar con componentes híbridos. Se integra con los principales *frameworks* de *frontend*, como Angular, React y Vue, aunque también se puede usar Vanilla JavaScript.

Características de Ionic

Estas son algunas razones por las que decantarse por Ionic a la hora de desarrollar aplicaciones móviles híbridas:

- Componentes UI de Ionic: La facilidad que Ionic ofrece para el diseño de interfaces es uno de sus puntos fuertes y lo consigue gracias a sus componentes. Los Componentes de Ionic son bloques de construcción de alto nivel que nos ayudan a construir de forma rápida la interfaz de usuario de nuestra aplicación.

- Numerosas integraciones y *plugins*: Ionic se integra con los *frameworks* con los que habitualmente se trabaja, Angular, React y Vue. Además, se integra también con numerosas herramientas y dispone de numerosos *plugins*. Esto es importante, puesto que lo integraremos con Angular para desarrollar nuestra aplicación.

- Más productividad y menos costes: Ionic favorece una mayor productividad de los desarrolladores y reduce los costes de desarrollo de la aplicación. Desarrollar aplicaciones híbridas en un único código propicia un menor tiempo de desarrollo y hace que su mantenimiento y escalado sea más sencillo, resultando en un desarrollo menos costos que una aplicación nativa

- Diseño de interfaces sencillo: Ionic hace más sencillo y rápido el diseño de interfaces de usuario para los desarrolladores. Pueden ir eligiendo elementos UI predeterminados de su librería de componentes en vez de tener que ir codificando uno a uno.

- Buena documentación y respaldo de la comunidad: Ionic Framework es un proyecto de código abierto, muy bien documentado y con una comunidad muy activa, lo cual siempre se agradece y facilita el aprendizaje y desarrollo.

No obstante, Ionic también presenta algunas desventajas:

- Peor rendimiento que las aplicaciones nativas: Si el objetivo principal es el rendimiento de nuestra aplicación, como podría ser el caso de un videojuego, se deberían contemplar otras opciones, que permiten desarrollar aplicaciones nativas, que tienen un mejor rendimiento que las aplicaciones híbridas.

- Dependencia con los *plugins*: Cada vez que necesitemos acceder a funcionalidad nativa deberemos recurrir a un plugin. Normalmente encontraremos un plugin para implementar la funcionalidad que necesitemos, pero en algunas ocasiones muy concretas podemos tener que crearlo nosotros mismos.
- Aplicaciones más pesadas que las nativas: Crear nuestra aplicación usando HTML, CSS y JavaScript implica escribir mucho código y agregar librerías, complementos y dependencias que harán que nuestra aplicación sea más pesada que una aplicación nativa.

En el caso de nuestra aplicación las ventajas superan con creces las desventajas, que suelen ser de rendimiento y peso de la aplicación, puesto que nuestra aplicación no requiere un alto rendimiento y a grandes rasgos funciona como un *front* que se comunica con la API Rest del servidor para obtener la mayoría de los datos.

Figura 63: Logo Ionic. Fuente: SeekLogo [29]

e) Bootstrap

Bootstrap [37] es el *framework* de *frontend* de código abierto más popular actualmente. Bootstrap es una colección de clases CSS y funciones de JavaScript y se utiliza para el diseño *responsive*. Por lo general, funciona en un sistema de cuadrícula para crear diseños de página con la ayuda de filas y columnas y es compatible con todos los navegadores para crear sitios web receptivos.

Este es utilizado junto a Vue en el *frontend* de la nueva web para mejorar el diseño de los elementos y facilitar la accesibilidad.

Figura 64: Logo Bootstrap. Fuente: SeekLogo [29]

7.1.3 Bases de datos

En programación es prácticamente inevitable trabajar con algún tipo de sistema de gestión de bases de datos. Cualquier programa que imaginemos tarde o temprano necesitará almacenar datos en algún lugar, como mínimo para poder almacenar la lista de usuarios autorizados junto a sus permisos y propiedades, y a medida que crece un sistema también lo hace la cantidad de datos almacenados.

MySQL [38] es el sistema de gestión de bases de datos relacional más extendido en la actualidad, y esto se debe a que es de código abierto y presenta algunas ventajas que lo hacen muy interesante. La más evidente es que trabaja con bases de datos relacionales, es decir, utiliza tablas múltiples que se interconectan entre sí para almacenar la información y organizarla correctamente, que serán las utilizadas en este proyecto.

a) Características principales de MySQL

Las características que hacen MySQL un sistema de gestión de bases de datos tan popular, y los motivos de su uso en este proyecto son los siguientes:

- Arquitectura Cliente y Servidor: MySQL basa su funcionamiento en un modelo cliente y servidor. Es decir, clientes y servidores se comunican entre sí de manera diferenciada para un mejor rendimiento.
- Compatibilidad con SQL: SQL es el lenguaje generalizado dentro de la industria y al ser un estándar MySQL ofrece plena compatibilidad.
- Vistas: MySQL ofrece compatibilidad para poder configurar vistas personalizadas del mismo modo que podemos hacerlo en otras bases de datos SQL. En bases de datos de gran tamaño las vistas se hacen un recurso imprescindible.
- Procedimientos almacenados. MySQL posee la característica de no procesar las tablas directamente, sino que a través de procedimientos almacenados es posible incrementar la eficacia de nuestra implementación.
- Desencadenantes. MySQL permite además poder automatizar ciertas tareas dentro de nuestra base de datos. En el momento que se produce un evento, otro es lanzado para actualizar registros u optimizar su funcionalidad.
- Transacciones. Una transacción representa la actuación de diversas operaciones en la base de datos como un dispositivo. El sistema de base de registros avala que todos los procedimientos se establezcan correctamente o ninguno de ellos. En caso por ejemplo de una falla de energía, cuando el monitor falla u ocurre algún otro inconveniente, el sistema opta por preservar la integridad de la base de datos, resguardando la información.

Con todas estas funcionalidades, MySQL satisface los requisitos necesarios para ser el sistema de gestión de base de datos de nuestro proyecto.

Figura 65: Logo MySQL. Fuente: SeekLogo [29]

7.1.4 Librerías externas

El objetivo de las librerías es poder evitar la necesidad de reinventar la rueda cuando existe una librería detallada que te ofrece las funcionalidades necesarias, ha sido ampliamente testada y funciona de forma correcta, rápida y eficiente, siendo altamente versátil y configurable para casos específicos, o al menos para los necesarios.

La mayoría de funciones externas necesarias son aportadas por los propios *frameworks*, que ya incluyen una cantidad importante de librerías. Sin embargo, hay algunas librerías bastante potentes que si han sido incluidas en algunos puntos, facilitando el desarrollo y aportando funcionalidades complejas y con gran versatilidad. Ambas son librerías de Javascript y son las siguientes:

a) DataTables

La librería DataTables [39] está enfocada al renderizado de tablas a partir de un set de datos. Permite obtener una tabla a partir de un conjunto de datos obtenidos generalmente de la base de datos y mostrarlo en la página que se va a renderizar, resultando en una tabla con todos los datos organizados en filas y columnas. Además, permite la ordenación por columnas, admite la búsqueda de datos en tiempo real y la paginación de la tabla cuando supera una cierta cantidad de resultados.

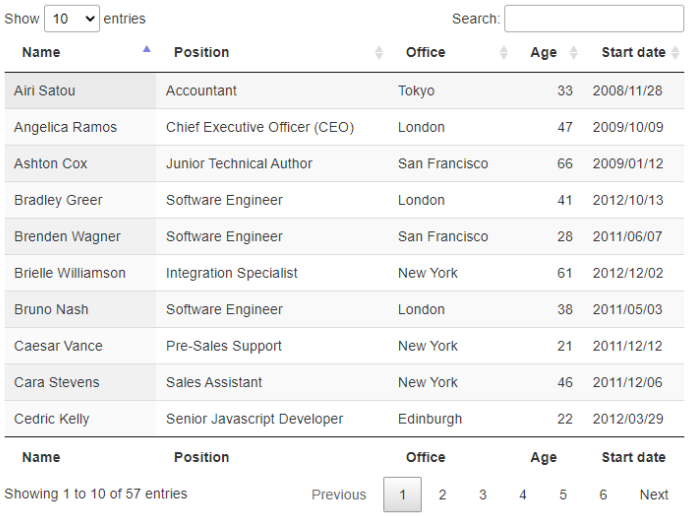


Figura 66: Tabla generada utilizando la librería DataTables. Fuente: DataTables [39]

b) date-fns

La librería date-fns [40] [41] está enfocada a facilitar la interacción con fechas en Javascript, puesto que trabajar con fechas en JavaScript suele ser un fastidio. El método de la fecha local suele ser largo, ocasionalmente inconsistente y, por tanto, propenso a errores. Pero lo gratificante está a la vista. Hay varias bibliotecas que se han dedicado a ofrecer funcionalidades que agrupen las necesidades más comunes del programador. Un claro ejemplo son las siguientes imágenes, donde se muestra cómo obtener el último día de este mes. En la primera imagen apreciamos la dificultad de legibilidad del código aceptado en Javascript para obtener este valor, mientras que en la segunda observamos la sencillez del método ofrecido por date-fns.

```
1 var t = new Date();
2 alert( new Date(t.getFullYear(), t.getMonth() + 1, 0, 23, 59, 59) );
```

Figura 67: Código Javascript para obtener el último dia del mes sin la librería date-fns. Fuente: ProgrammerClick [41]

```
1 const today = new Date();
2 console.log( lastDayOfMonth(today) );
```

Figura 68: Código Javascript para obtener el último dia del mes con la librería date-fns. Fuente: ProgrammerClick [41]

8 Evaluación del sistema

Todo proyecto ha de ser validado y evaluado, para asegurarse que cumpla las directrices y restricciones necesarias para ser considerado finalizado. En este apartado nos centraremos en cómo se han realizado dichas pruebas y que herramientas se han utilizado. Concretamente, estas pruebas se han realizado a lo largo del desarrollo y una vez finalizado el proyecto, garantizando su correcto funcionamiento.

8.1 *Tests*

CakePHP viene con un completo soporte de prueba integrado, concretamente, viene con integración para PHPUnit, y además de las funciones que ofrece PHPUnit, CakePHP ofrece algunas funciones adicionales para facilitar las pruebas. Durante el desarrollo se elaboraron *tests* de las funciones de los modelos y controladores del *backend* para comprobar que estos funcionaban de la forma esperada y para lo que fueron diseñados.

Para realizar estos *tests*, además del código que ejecute los métodos y la comprobación de los resultados obtenidos se utilizaron *fixtures*, los *fixtures* te permiten generar tablas de datos temporales cargadas con datos de muestra para ser utilizadas por los *tests*, en lugar de la base de datos de la aplicación. El beneficio de usar *fixtures* es que su prueba no tiene ninguna posibilidad de interrumpir o editar los datos de la aplicación en vivo, y no se ve afectada por los cambios producidos en la aplicación, los datos de testeo y los de la aplicación son completamente independientes. Además, se puede comenzar a probar el código antes de desarrollar contenido en la aplicación.

8.2 Proceso de testeo

A continuación, se aclarará cuál es el proceso completo de testeo. Como se ha explicado previamente, se ha seguido una metodología SCRUM y un desarrollo basado en git, en el que se ha hecho uso de la funcionalidad de *pull request*, y, concretamente, el proceso de testeo ha sido el siguiente:

1. Se selecciona una tarea y se clona una rama a partir de *develop* para desarrollar la tarea
2. Se desarrolla la tarea hasta que se considere que se satisfacen todos los requisitos necesarios establecidos para esta tarea y se elaboran *tests* que comprueben su correcto funcionamiento.
3. Una vez considerado que esta rama contiene todas las funcionalidades de dicha tarea, estas funcionan correctamente y todos los *tests* pasan, se abre un *pull request* para hacer *merge* con la rama padre.
4. La rama es evaluada por los compañeros desarrolladores de la empresa previamente a su fusión con la rama de desarrollo para asegurarse que funcione correctamente y cumpla los requisitos funcionales y no funcionales establecidos. En caso de no pasar, se especifican las faltas encontradas y se vuelve al paso 2 para su corrección.
5. Llegados a este punto se considera que la PR esta lista y se hace *merge* en *develop*, del cual se pasa a hacer *deploy* en el servidor beta de testeo para que las nuevas funcionalidades sean probadas por los *Product Owners*, que se asegurarán que las funcionalidades satisfacen aquello que se había pedido y, si durante el testeo se detecta algún error o posible mejore que había pasado la PR, se indica en la tarea y se vuelve a abrir la rama volviendo al paso 2.

6. Si los testeos han ido correctamente, la funcionalidad es considerada finalizada. Generalmente, cuando se acerca el final del sprint y se tienen suficientes funcionalidades nuevas, se abre una rama *release* de *develop*, obteniendo una nueva versión lista para salir al mercado.
7. Se inician los procesos necesarios para publicar la nueva versión de la app, si los cambios son en la app, y si son en la web, se actualiza la versión del servidor. Primero se saca la versión solo para algunos centros, que prueban la aplicación y comprueban que funciona correctamente, aportando *feedback* sobre errores, posibles mejoras y peticiones, a cambio de tener acceso anticipado. Una vez ha pasado un tiempo y no se han detectado fallos, se procede a sacar la nueva versión para todos los centros. Si por el contrario, se detectan errores, se vuelve al paso 2 para corregirlos.
8. Como la app está actualmente en uso por una gran cantidad de usuarios, si algún usuario detecta algún error o mejora puede ponerse en contacto con la empresa, y si se considera válido y necesario se crea una nueva tarea o se reabre una ya existente para implementar la mejora o solucionar el error.

Principalmente se han realizado cuatro grandes *releases* de versiones, que serían la versión de pagos por grupos en la web antigua, la versión de la fase 1 de tutorías tanto en web antigua como en móvil, la versión de pagos por grupos en la web nueva y la versión de la fase 2 de tutorías tanto en móvil como en la nueva web. Aparte de estas, se ha sacado alguna nueva versión debido a la realización de *hotfix* o la mejora de algún detalle menor.

9 Planificación temporal

A continuación, se explicarán la planificación inicial que se hizo, la planificación en el momento de la reunión de seguimiento, y la planificación seguida finalmente, tratando las desviaciones ocurridas, los cambios realizados y que evolución ha seguido el proyecto.

9.1 Calendario

La duración del trabajo de fin de grado se extiende desde el 15 de febrero, con el inicio del cuatrimestre, hasta entre el 28 de junio y 2 de julio, con la presentación del TFG ante el tribunal, teniendo una duración de aproximadamente cuatro meses y medio, unas 19 semanas aproximadamente, teniendo en cuenta que la memoria se ha de entregar con una semana de antelación a la presentación para su previa lectura y análisis por parte del ponente, director y tribunal que evaluará el TFG.

Por otra parte, el convenio con la empresa empezó el día 8 de febrero y acabó el 8 de junio, desde el inicio del convenio hasta el comienzo del proyecto estuve trabajando en ambas aplicaciones (web y móvil), realizando tareas independientes del proyecto, tanto en el nuevo como en el viejo repositorio web, ampliando mis conocimientos sobre el funcionamiento y distribución actual de la aplicación, de forma que me fuera más sencillo el desarrollo de los dos módulos una vez el proyecto comenzará.

El desarrollo del proyecto comenzó el día 29 de marzo, y acabó el 11 de junio debido a los términos establecidos en el convenio con la empresa. Este desarrollo duró un total de 11 semanas trabajando a jornada completa, es decir, 5 días por semana, 8 horas por día, excepto 2 días de las 11 semanas que eran fiesta en Cataluña, disponiendo de un total de 424 horas para el desarrollo del proyecto más la gestión *agile* del mismo, que aproximadamente supone 2 horas cada semana, unas 22 horas en total de las 424 de desarrollo. A estas horas se han de añadir las invertidas en tareas de documentación, tanto durante la realización de GEP como a posteriori, y de preparación de la presentación, aproximadamente 200 horas. Por lo tanto, el tiempo invertido en este proyecto ha sido de unas 624 horas aproximadamente.

9.2 Recursos

Los recursos que se han utilizado durante la realización de este proyecto han sido los siguientes:

9.2.1 Recursos humanos

El conjunto de individuos que colaboraron en la realización del proyecto está formado por:

- YO => Yo: He sido el único desarrollador del proyecto, y me he encargado de llevar a cabo todas las tareas.
- DT => Equipo de desarrollo de la aplicación: Han revisado las *pull request* que he ido realizando, asegurándose que la implementación realizada se adhiera a los estándares de la aplicación, además de proporcionarme soporte y resolver mis dudas sobre ciertos aspectos de la aplicación.
- PO => *Product Owners*: Han intervenido en las reuniones, comprobando el correcto desarrollo del proyecto y ayudando en la priorización de objetivos, balanceando el valor que aportan respecto al coste.
- DTFG => Director del TFG: Mi director, Cristian Cisa, ha proporcionado soporte y supervisado el progreso a lo largo del proyecto, puesto que constituye uno de los *Product Owners*.

- PTFG => Ponente del TFG: Mi ponente, Ernest Teniente, ha proporcionado soporte y ayuda sobre el desarrollo del TFG cuando ha sido necesario.

9.2.2 Recursos Materiales

Las herramientas que han sido utilizadas para el desarrollo del proyecto son las siguientes:

- VSC => Visual Studio Code: IDE utilizada para el desarrollo en los diferentes entornos y lenguajes.
- DB => DBeaver: Herramienta de administración de bases de datos utilizada durante el desarrollo.
- GB => Git y Bitbucket: Herramientas utilizadas para el control de versiones y gestión de repositorios.
- D => Docker: Herramienta de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de *software*. Esta será utilizada durante el desarrollo de la aplicación web.
- J => Jira: Herramienta utilizada para el seguimiento de la metodología Scrum a lo largo del proyecto.
- GM => Google Meet: Aplicación de videollamadas utilizada para la realización de reuniones.
- S => Slack: Herramienta de comunicación en equipo a través de la cual se pueden realizar cuestiones, hablar con los compañeros y acordar soluciones de forma textual.
- GC => Google Chrome: Navegador utilizado para el desarrollo e inspección de la aplicación.
- A => Atenea: Página web de la universidad donde se encuentran los recursos referentes a la redacción y desarrollo del proyecto.
- W => Word: Herramienta utilizada para la redacción de toda la documentación del proyecto.
- E => Excel: Utilizado para la elaboración de ciertas tablas que requerían cálculos y los diagramas de Gantt.
- P => PowerPoint: Herramienta utilizada para la elaboración y presentación de diapositivas.
- VP => VisualParadigm: Herramienta utilizada para la elaboración del diagrama de clases y los diagramas de casos de uso.

9.3 Tareas

Al seguir la metodología Scrum, que supone el trabajo por *sprints*, las tareas han ido quedando claramente definidas a lo largo del proyecto, puesto que previo a su implementación fueron debatidas e incluidas en el *sprint*. Sin embargo, como el proyecto aún no había empezado durante la elaboración de la planificación inicial, algunas tareas aún no estaban claramente definidas y/o suficientemente detalladas, además, al seguir una metodología *agile* estas tareas fueron sufriendo cambios a lo largo del proyecto.

A continuación, se explicarán las tareas que se han realizado a lo largo del proyecto. Cabe recalcar que en el tiempo dedicado a estas tareas se incluye la realización de *tests* para garantizar su correcto funcionamiento.

9.3.1 Tareas de gestión

Estas tareas engloban la documentación, planificación, presentación y defensa del proyecto y no han sufrido ningún cambio durante la realización del proyecto, salvo movimientos en el calendario.

- TG1 => Alcance (25h): Elaboración de la primera entrega de gestión del proyecto, esta contiene el contexto, justificación, alcance y metodología utilizada del proyecto.

- TG2 => Planificación (15h): Redacción de la segunda entrega de gestión del proyecto, en esta se incluyen la planificación de tareas, el diagrama de Gantt y la gestión de riesgos.
- TG3 => Gestión económica y sostenibilidad (15h): Preparación de la tercera entrega de gestión del proyecto, donde se especifican los presupuestos y el informe de sostenibilidad.
- TG4 => Documento final de la fase inicial (15h): Se combinan las tres entregas realizadas anteriormente referentes a la gestión del proyecto, aplicando los cambios especificados.
- TG5 => Reunión y documentación de la fase de seguimiento (30h): Elaboración de un informe que recoja el progreso y estado actual del proyecto, y reunión con el ponente, para tratar el estado actual y seguimiento del proyecto.
- TG6 => Documentación de la fase final (60h): Elaboración de la memoria del TFG conteniendo, adaptando y ampliando los documentos entregados con anterioridad, que deberá ser entregada a un tribunal de la especialidad.
- TG7 => Preparación presentación del TFG (40h): Preparación de la defensa del TFG que deberá ser presentado en una sesión pública delante de un tribunal de la especialidad.
- TG8 => Gestión *agile* (22h): Esta tarea engloba todas las tareas y rutinas de gestión y seguimiento del proyecto establecidas en la metodología Scrum, estimando el tiempo total dedicado a la suma de *Sprint Planning*, *Daily Scrum* y demás rutinas *agile*.

9.3.2 Tareas de implementación del módulo de tutorías

Estas tareas son aquellas referentes al desarrollo del módulo de tutorías, en un principio se anotaron tareas muy grandes y genéricas, que a lo largo del proyecto se han ido desglosando en diversas subtareas y además se han añadido nuevas tareas, también cabe recalcar que muchas de estas tareas se han de hacer tanto en la web como en la app, y por tanto tienen dos estimaciones temporales.

- TT1 => Creación del módulo de tutorías (6h): Creación del módulo de tutorías en la página web y las tablas en la base de datos, preparando el entorno para iniciar el desarrollo de las páginas y diversas funcionalidades.
- TT2 => Creación de franjas de tutoría (30h): Los tutores podrán desde la web crear franjas de tutoría, que constituyen un rango, de fecha inicio a fecha fin, en el cual se especifica un día de la semana y una hora, y dentro de ese rango se genera una tutoría para cada día de la semana que coincida, a esa hora especificada.
- TT3 => Visualización y filtrado de franjas de tutoría (22h): Los profesores podrán desde la web visualizar las franjas de tutoría creadas y filtrarlas dentro de un rango, por hora y por día de la semana.
- TT4 => Edición de franjas de tutoría (16h): Los profesores podrán desde la web editar las franjas de tutorías creadas, editando así las tutorías asociadas. Se puede modificar la fecha de inicio del rango, la fecha de fin, el día de la semana y la hora.
- TT5 => Eliminación de franjas de tutoría (6h): Los profesores podrán desde la web eliminar franjas de tutorías que no tengan ninguna tutoría asociada reservada.
- TT6 => Visualización y filtrado de tutorías (web – 24h) (app – 30h): Los profesores podrán ver sus tutorías filtradas por rango, estado y nombre del estudiante.

- TT7 => Cambio del estado de las tutorías (web – 16h) (app – 8h): Los profesores podrán cambiar el estado de las tutorías, bloqueando, desbloqueando y cancelando tutorías.
- TT8 => Adición de campos extra a las tutorías (1h): Se añadirán tres campos extra a las tutorías: ubicación, anotaciones privadas y anotaciones públicas. Todos campos de texto editables por el profesor y visibles por el solicitante, excepto las anotaciones privadas que solo serán visibles por el profesor
- TT9 => Visualización y edición de los campos de las tutorías (web – 7h) (app – 8h): El profesor podrá editar los campos de las tutorías.
- TT10 => Visualización y filtrado de tutorías disponibles (web – 8h) (app – 22h): Los solicitantes podrán ver las tutorías disponibles de sus tutores o tutores de sus hijos, filtrándolas por día de la semana, a partir de una cierta hora y hasta una cierta hora.
- TT11 => Reserva de tutorías (web – 4h) (app – 8h): Los solicitantes podrán reservar las tutorías disponibles de sus tutores o tutores de sus hijos.
- TT12 => Visualización de tutorías reservadas (web – 8h) (app – 22h): Los solicitantes podrán ver sus tutorías reservadas y la información asociada a ellas.
- TT13 => Cancelación de tutorías reservadas (web – 4h) (app – 8h): Los solicitantes podrán cancelar sus tutorías reservadas.
- TT14 => Notificación de actualizaciones en la reserva (8h): Cuando se produzcan cambios en el estado de una tutoría en la que un usuario está involucrado, este será notificado del cambio del estado mediante notificaciones *push* o correo electrónico, según este configurado.
- TT15 => Migración del módulo de tutorías (22h): Migración del módulo de tutorías, sus pantallas y todas las funcionalidades desarrolladas hasta el momento en el repositorio viejo al nuevo repositorio web.
- TT16 => Solicitación de reservas de tutorías (22h): Los tutores podrán enviar una solicitud a sus alumnos para que reserven una de sus horas de tutoría.
- TT17 => Compartir tutorías (web – 6h) (app – 2h): Cuando un padre reserve una tutoría podrá decidir si quiere compartir esta tutoría con el resto de padres del estudiante o quiere que sea individual.
- TT18 => Ver tutorías reservadas en el calendario (16h): Los usuarios involucrados en una tutoría reservada podrán verla en el calendario y serán notificados una hora antes de su inicio. Además, desde el calendario se podrá ver la información de la tutoría y los campos públicos, que se actualizarán si se actualiza la tutoría.

9.3.3 Tareas de implementación del módulo de pagos

Estas tareas son aquellas referentes al desarrollo del módulo de pagos, las cuales han sufrido el mayor cambio, puesto que se decidió priorizar el módulo de tutorías, reduciendo el número de tareas y añadiendo menos funcionalidades. A pesar de esto, las funcionalidades añadidas son muy potentes y mejoran en gran medida la utilidad del módulo.

- TP1 => Visualización y filtrado de grupos por pagos pendientes (12h): El tesorero del centro podrá ver que grupos del centro tienen pagos pendientes y filtrar esta búsqueda por rango de fechas entre las que se crearon los pagos, grupos, y solo aquellos con pagos pendientes.

- TP2 => Visualización y filtrado de pagos por grupo (16h): El tesorero del centro podrá ver los pagos creados y dirigidos a alumnos de los grupos seleccionados y el estado de estos pagos, esta búsqueda podrá estar filtrada por un rango de fechas, grupos seleccionados y se podrán filtrar fuera pagos que no se quieran tener en cuenta. Se mostrará también el total ingresado y pendiente por alumno teniendo en cuenta los pagos visualizados.
- TP3 => Exportación de pagos por grupo (4h): Los datos mostrados en la pantalla de visualización y filtrado de pagos por grupo podrán ser exportados.
- TP4 => Edición del estado de los pagos (13h): Si se selecciona un pago de un alumno en la pantalla de visualización y filtrado de pagos por grupo, se mostrará la información del pago y se podrá modificar su estado.
- TP5 => Envío de recordatorio de pagos pendientes (1h): En la pantalla de visualización y filtrado de pagos por grupo, se podrá enviar un recordatorio de que un pago está pendiente cuando un pago de un alumno este pendiente de pago.
- TP6 => Migración del módulo de pagos (22h): Migración de las nuevas pantallas del módulo de pagos y todas las funcionalidades desarrolladas hasta el momento en el repositorio viejo al nuevo repositorio web.

9.4 Estimaciones y Gantt

La siguiente es una tabla con las estimaciones de las diferentes tareas seguidas en el proyecto:

Código	Tareas	Duración (h)	Parte web (h)	Parte app (h)	Dependencias	Recursos humanos	Recursos Materiales
TG1	Alcance	25				YO, DTFG, PTFG	A, W, GM
TG2	Planificación	15			TG1	YO, DTFG	A, W, E, GM, G
TG3	Gestión económica y sostenibilidad	15			TG2	YO, DTFG	A, W, E, GM
TG4	Documento final de la fase inicial	15			TG3	YO, DTFG, PTFG	A, W, E
TG5	Reunión y documentación de la fase de seguimiento	30			TG4	YO, PTFG	W, E, GM
TG6	Documentación de la fase final	60			TG5	YO, DTFG, PTFG	W, E, GM, VP
TG7	Preparación presentación del TFG	40			TG6	YO	W, E, P, GM
TG8	Gestión agile	22				YO, DT, PO	GM, J
	TOTAL: Tareas de Gestión	222	0	0			
TT1	Creación del módulo de tutorías	6	6			YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT2	Creación de franjas de tutoría	30	30		TT1	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT3	Visualización y filtrado de franjas de tutoría	22	22		TT2	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT4	Edición de franjas de tutoría	16	16		TT3	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT5	Eliminación de franjas de tutoría	6	6		TT3	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT6	Visualización y filtrado de tutorías	54	24	30	TT2	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT7	Cambio el estado de las tutorías	24	16	8	TT6	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT8	Adición de campos extra a las tutorías	1	1		TT1	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT9	Visualización y edición de los campos de las tutorías	15	7	8	TT8	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT10	Visualización y filtrado de tutorías disponibles	30	8	22	TT2	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT11	Reserva de tutorías	12	4	8	TT10	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT12	Visualización de tutorías reservadas	30	8	22	TT11	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT13	Cancelación de tutorías reservadas	12	4	8	TT12	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT14	Notificación de actualizaciones en la reserva	8	8		TT1	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT15	Migración del módulo de tutorías	22	22		TT1 - TT14	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT16	Solicitación de reservas de tutorías	22	22		TT15	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT17	Notificación a usuarios extra	8	6	2	TT14	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TT18	Ver tutorías reservadas en el calendario	16	16		TT17	YO, DT, PO	VSC, DB, GB, D, J, S, GC
	TOTAL: Tareas Módulo Tutorías	334	226	108			
TP1	Visualización y filtrado de grupos por pagos pendientes	12	12			YO, DT, PO	VSC, DB, GB, D, J, S, GC
TP2	Visualización y filtrado de pagos por grupo	16	16			YO, DT, PO	VSC, DB, GB, D, J, S, GC
TP3	Exportación de pagos por grupo	4	4		TP2	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TP4	Edición del estado de los pagos	13	13		TP2	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TP5	Envío de recordatorio de pagos pendientes	1	1		TP2	YO, DT, PO	VSC, DB, GB, D, J, S, GC
TP6	Migración del módulo de pagos	22	22		TP1 - TP5	YO, DT, PO	VSC, DB, GB, D, J, S, GC
	TOTAL: Tareas Módulo Pagos	68	68	0			
	TOTAL	624	294	108			

Tabla 21: Estimación de duración, dependencias y recursos por tarea. Fuente: Propia

El siguiente es el diagrama de Gantt de la planificación temporal seguida en el proyecto, donde se establecen los plazos delimitados para la realización de cada tarea y la distribución de estas a lo largo del proyecto:

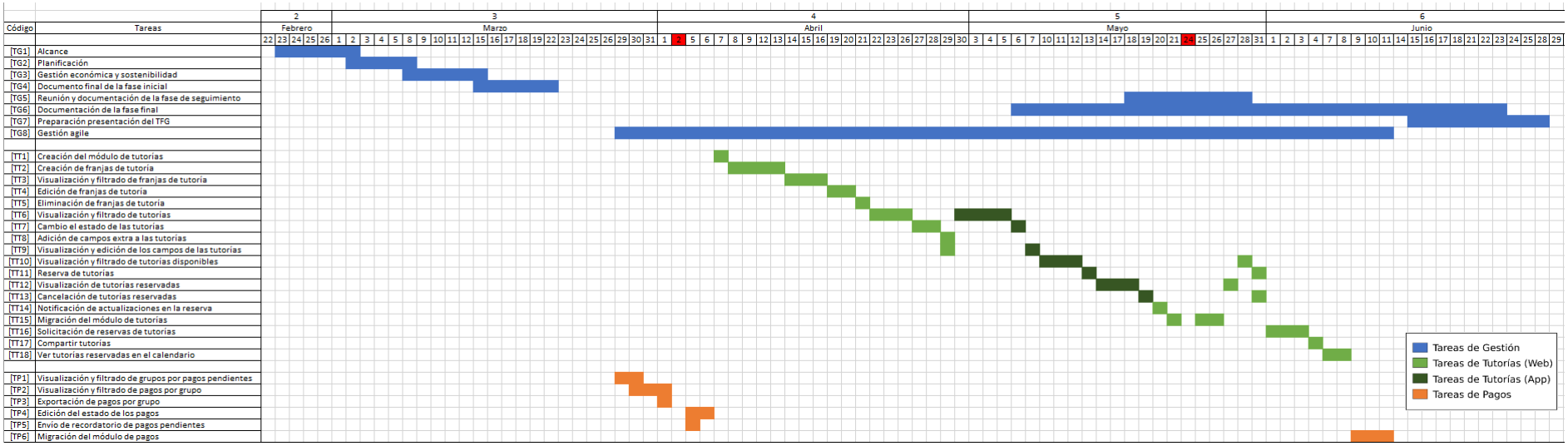


Figura 69: Diagrama de Gantt de la planificación seguida. Fuente: Propia

9.4.1 Cambios en la planificación

Inicialmente, la visión del proyecto era la creación de un módulo de tutorías funcional y la ampliación del módulo de pagos aumentando su utilidad, teniendo ambos igual prioridad, y centrándose en ambos por igual, con una distribución del tiempo de desarrollo prácticamente equitativa entre ambos módulos.

Esta visión no ha cambiado a lo largo del proyecto, pero la prioridad y reparto del tiempo de desarrollo entre los módulos sí. A medida que fue avanzando el proyecto, se vio que las tareas eran más extensas y tenían más detalles que los pensados originalmente, dificultando su realización y ampliando su duración, y que para tener un módulo de tutorías que fuera realmente funcional y que estuviera lo suficientemente completo como para salir al mercado, sería necesario añadir más tareas y dedicar más tiempo al módulo.

Con estas observaciones, y teniendo en cuenta que el módulo de pagos era un módulo ya existente y funcional, simplemente se querían mejorar ciertas partes que habían solicitadas por los clientes, y de paso ampliar sus funcionalidades para hacerlo más completo. Se decidió dar más prioridad al módulo de tutorías, resultando en un módulo de tutorías completamente funcional y un módulo de pagos con menos ampliaciones, pero mejoras potentes que aportan gran funcionalidad y utilidad al módulo. Acabando con una dedicación del 17% del tiempo de desarrollo al módulo de pagos y un 83% del tiempo de desarrollo al módulo de tutorías.

A continuación, explicaremos brevemente las planificaciones anteriores y mostraremos el diagrama de Gantt correspondiente a cada una de ellas.

a) Planificación inicial

La planificación inicial se realizó durante la realización del curso de GEP, prácticamente un mes antes del inicio del proyecto, en este momento, se añadieron las tareas que se consideraron necesarias para tener un módulo de tutorías, y una cantidad similar de tareas del módulo de pagos.

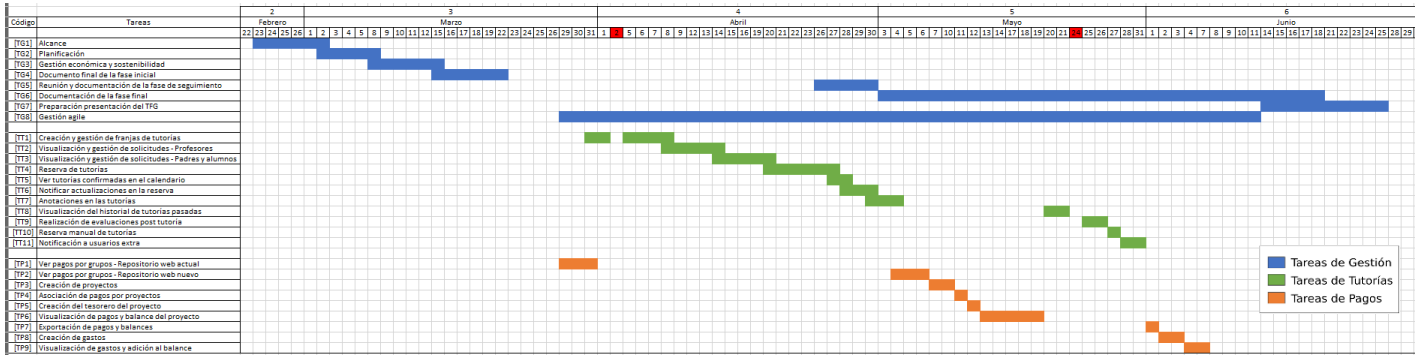


Figura 70: Diagrama de Gantt de la planificación inicial. Fuente: Propia

b) Planificación en el momento de la reunión de seguimiento

La reunión de seguimiento se realizó el día 25 de mayo, a tres semanas de finalizar el desarrollo del proyecto, y con un gran cambio respecto a la planificación inicial, disminuyendo la importancia del módulo de pagos y añadiendo una gran cantidad de tareas al módulo de tutorías.

Figura 71: Diagrama de Gantt de la planificación de seguimiento. Fuente: Propia

9.5 Gestión del riesgo

En cuanto a los posibles riesgos y obstáculos que pueden afectar al correcto desarrollo del proyecto, su gestión y los planes alternativos propuestos han sido los siguientes:

- Inexperiencia: Este es un obstáculo que se ha tenido en cuenta desde el principio, yo empecé a trabajar en la empresa un mes y medio antes del inicio del proyecto y a lo largo de este mes y medio he estado trabajando tanto en la aplicación web como en la aplicación móvil, tocando todos los aspectos posibles para adaptarme al entorno y facilitar el desarrollo del proyecto. En cuanto al resto de inconvenientes que puedan surgir debido a mi inexperiencia, siempre puedo consultarlos con el equipo de desarrolladores e investigar para obtener la solución más óptima, por lo que este obstáculo se ha tratado a lo largo del proyecto.
- Bugs: Prácticamente todos los proyectos los tienen o han tenido, siempre acaban surgiendo y se intentan tratar o mitigar lo más pronto posible. Los bugs están destinados a aparecer en cualquier proyecto de envergadura considerable, pero lo que es importante tener un plan de acción que los haga difíciles de aparecer y fáciles de detectar, y la mejor forma de realizar esto es mediante *tests* que comprueben que el código se comporta de la forma esperada y si en algún momento se detecta un comportamiento fuera de lo normal se utilizan herramientas de depuración para tratar el problema. A lo largo del desarrollo, se han elaborado *tests* de forma regular para garantizar el correcto funcionamiento de los distintos elementos que conforman la aplicación.
- Desviación de la planificación: Es muy complicado que un proyecto siga la planificación al pie de la letra, por lo que a lo largo del proyecto se ha de ver el estado en que se encuentra y replantear el desarrollo en caso de que sea necesario. Esto está previsto en la metodología utilizada, Scrum. La realización de *sprints*, reuniones diarias y semanales y elaboración de tareas concretas en forma de historias de usuario permite detallar el estado actual del proyecto, el porcentaje realizado y el restante, y adaptar en todo momento el desarrollo en base a los datos obtenidos para obtener el producto esperado con todos los detalles posibles en el término establecido. Según las desviaciones reportadas a lo largo del proyecto se debatieron y tomaron las medidas necesarias para su correcto desarrollo. En caso de ser necesario porque la desviación se hubiera descontrolado, se podría haber llegado a utilizar la ayuda de los compañeros del equipo de desarrollo en ciertas tareas, pero no llegó a ser necesario.
- Posibles problemas derivados de la pandemia: La pandemia ha cambiado la forma en que vivimos, nos comunicamos, nos relacionamos y realizamos nuestro trabajo, y esta situación excepcional se debería tener en cuenta. En principio, la pandemia no debería haber supuesto un gran problema para el desarrollo del proyecto, puesto que la totalidad del desarrollo del proyecto se realizó de forma online, lo cual supone una disminución del riesgo de contagio, y la posibilidad de seguir trabajando desde casa en caso de confinamiento. Sin embargo, no se relajaron las medidas de seguridad, ya que el contagio hubiera supuesto, con total seguridad, un retraso del proyecto y un riesgo para la salud. Si el contagio provocara un retraso excepcional, también se podría haber recurrido a la ayuda de los compañeros del equipo de desarrollo, pero no acabo siendo necesario.

10 Presupuesto

10.1 Identificación de los costes

Con el objetivo de identificar correctamente los costes del desarrollo del proyecto se han de tener en cuenta los diferentes tipos de coste involucrados en el desarrollo. Estos costes derivados del desarrollo de un proyecto se pueden dividir en: Costes de Personal por Actividad (CPA), Costes imputados Genéricamente (CG), costes de riesgos y gastos de contingencia. A continuación, se explican en detalle cada uno de ellos, en qué consisten, como calcularlos y su estimación.

10.1.1 Costes de Personal por Actividad (CPA)

Los recursos humanos son el conjunto de empleados y colaboradores que participarán en el proyecto desempeñando diferentes roles y realizando diferentes tareas. Con el objetivo de planificar los costes derivados del tiempo empleado por estas personas se realizará una estimación del salario medio cobrado según su posición. Las estimaciones de salario se han obtenido de la página web Glassdoor [42], especificando la zona de Barcelona y asumiendo 1.720 horas efectivas reales de trabajo anuales como se establece en el BOE [43].

$$Salario\ por\ hora = \frac{Salario\ anual}{Horas\ anuales}$$

Posición	Salario/hora (bruto)	Salario/hora + Seguridad Social (x1,3)
Product Owner [44]	26,05€	33,87€
Desarrollador *full-stack* [45]	20,35€	26,45€
Desarrollador becario	9€	11,70€

Tabla 22: Salario por hora según posición. Fuente: Propia

Una vez obtenidas las estimaciones de salario para los diferentes empleados y aplicado el coste derivado de la Seguridad Social, podemos estimar el coste humano derivado de cada tarea, aplicando al tiempo que tendrán que emplear en las tareas que tienen asignadas según el diagrama de Gantt el salario obtenido. Para ello hemos de tener en cuenta que el equipo de desarrolladores está compuesto por dos desarrolladores *full-stack*, pero que las revisiones serán realizadas por un solo *full-stack developer*, y que las horas de los *Product Owners* estimadas, son las totales combinadas de todos los *Product Owners* participantes en la tarea. Una vez aplicado obtenemos la siguiente tabla.

Código	Tareas	Duración (h)	Horas por posición			Coste total (€)
			YO	DT	PO	
TG1	Alcance	25	25	0	2	360
TG2	Planificación	15	15	0	2	243
TG3	Gestión económica y sostenibilidad	15	15	0	2	243
TG4	Documento final de la fase inicial	15	15	0	2	243
TG5	Reunión y documentación de la fase de seguimiento	30	30	0	0	351
TG6	Documentación de la fase final	60	60	0	2	770
TG7	Preparación presentación del TFG	40	40	0	0	468
TG8	Gestión agile	22	22	15	18	1264
	TOTAL: Tareas de Gestión	222	222	15	28	3943
TT1	Creación del módulo de tutorías	6	6	0,5	0	83
TT2	Creación de franjas de tutoría	30	30	2	0	404
TT3	Visualización y filtrado de franjas de tutoría	22	22	1,5	0	297
TT4	Edición de franjas de tutoría	16	16	1	0	214
TT5	Eliminación de franjas de tutoría	6	6	0,5	0	83
TT6	Visualización y filtrado de tutorías	54	54	3,5	0	724
TT7	Cambio el estado de las tutorías	24	24	1,5	0	320
TT8	Adición de campos extra a las tutorías	1	1	0,25	0	18
TT9	Visualización y edición de los campos de las tutorías	15	15	1	0	202
TT10	Visualización y filtrado de tutorías disponibles	30	30	2	0	404
TT11	Reserva de tutorías	12	12	1	0	167
TT12	Visualización de tutorías reservadas	30	30	2	0	404
TT13	Cancelación de tutorías reservadas	12	12	1	0	167
TT14	Notificación de actualizaciones en la reserva	8	8	0,5	0	107
TT15	Migración del módulo de tutorías	22	22	2	0	310
TT16	Solicitación de reservas de tutorías	22	22	1,5	0	297
TT17	Notificación a usuarios extra	8	8	0,5	0	107
TT18	Notificación a usuarios extra	16	16	1	0	214
	TOTAL: Tareas Módulo Tutorías	334	334	23,25	0	4523
TP1	Visualización y filtrado de grupos por pagos pendientes	12	12	0,75	0	160
TP2	Visualización y filtrado de pagos por grupo	16	16	1	0	214
TP3	Exportación de pagos por grupo	4	4	0,25	0	53
TP4	Edición del estado de los pagos	13	13	0,75	0	172
TP5	Envío de recordatorio de pagos pendientes	1	1	0,2	0	17
TP6	Migración del módulo de pagos	22	22	2	0	310
	TOTAL: Tareas Módulo Pagos	68	68	4,95	0	927
	TOTAL	624	624	43,2	28	9392

Tabla 23: Costes de Personal por Actividad (CPA). Fuente: Propia

10.1.2 Costes imputados Genéricamente (CG),

Estos incluyen todos aquellos gastos que se calcularán de forma global sin especificar en qué tarea se utilizan, y se pueden dividir en gastos de *hardware*, gastos de *software* y gastos generales.

a) Hardware

Para la realización de este proyecto será necesaria la utilización de cierto equipamiento *hardware*. Este equipamiento se deberá añadir a los costes del proyecto, teniendo en cuenta, como está estipulado por Hacienda, que el *hardware* será amortizado en 4 años, por lo que solo aplicaremos el precio relativo a la duración del proyecto respecto a estos 4 años. Teniendo en cuenta que las horas efectivas reales de trabajo anuales establecidas en el BOE son 1720h, cada año tiene 215 días laborables de 8 horas máximo, como el desarrollo del proyecto se realizará en 53 días laborables de desarrollo y 24 días de gestión, con una dedicación diaria de 8 horas, la fórmula aplicada para calcular las amortizaciones será la siguiente:

$$Amortización = \left(\frac{Coste\ Hardware}{4\ años * 215\ días\ laborables}\right) * 77\ días\ laborables\ de\ proyecto$$

Hardware	Coste (€)	Amortización (€)
MSI GL63 8RD-407XES (Ordenador portátil)	1098,99	98
Samsung LF22T450FQU (Monitor)	146,1	13
Logitech G203 (Ratón)	39,99	4
TOTAL		115

Tabla 24: Costes de Hardware. Fuente: Propia

b) Software

A lo largo del proyecto también se utilizará *software* diverso para el desarrollo, gestión y documentación del proyecto, y para la comunicación entre compañeros. Debido al tipo de licencias de estos productos (en su mayoría, anuales), no se tendrá en cuenta ninguna amortización de Hacienda, y simplemente se tendrá en cuenta el coste derivado del tiempo o cantidad utilizada durante la realización del proyecto. Para obtener estos costes aplicamos la siguiente fórmula:

$$Amortización = \left(\frac{Coste\ anual\ Software}{215\ días\ laborables}\right) * 77\ días\ laborables\ de\ proyecto$$

Software	Coste anual (€)	Amortización (€)
Visual Studio Code	Gratuito	0
Dbeaver	Gratuito	0
Git	Gratuito	0
Bitbucket [46]	150,88	54
Docker	Gratuito	0
Jira	Gratuito	0
Google Meet	Gratuito	0
Slack	Gratuito	0
Google Chrome	Gratuito	0
Atenea	Gratuito	0
Gantt Project	Gratuito	0
Microsoft Office 365 [47]	69	25
TOTAL		79

Tabla 25: Costes de Software. Fuente: Propia

c) Gastos generales

Finalmente, se han de tener en cuenta aquellos costes derivados de otros elementos, como pueden ser la luz, Internet y el material de oficina necesario para el desarrollo del proyecto. En cuanto al material de oficina, cabe recalcar que Hacienda permite una amortización de 20 años, que se tendrá en cuenta en el cálculo. Para calcular los gastos de electricidad se utilizarán los watts empleados por los aparatos utilizados. En este caso el portátil consume 150W y la pantalla auxiliar 28W. Finalmente para calcular el coste de Internet se aplicará al precio mensual los meses de uso, que en este caso serán 4. Las fórmulas utilizadas para los cálculos complejos son las siguientes:

$$Amortización\ material\ oficina = \left(\frac{Coste\ material\ oficina}{20\ años * 215\ días\ laborables}\right) * 77\ días\ laborables\ de\ proyecto$$

$$Amortización\ electricidad = Suma\ Potencia * 8h\ al\ dia * 77\ días\ laborables * Precio\ Unidad\ Potencia$$

Gastos generales	Coste (€)	Amortización (€)
Mesa oficina - LEVIRA	104,9	2
Songmics OBG22B - Silla de Oficina	114,99	2
Electricidad	0,1199€/KWh	13
Internet	30€/mes	120
TOTAL		137

Tabla 26: Gastos generales. Fuente: Propia

10.1.3 Costes de riesgos

En cuanto a los diferentes riesgos involucrados en el proyecto, se especificará la probabilidad de que ocurran y se multiplicará por el coste que supondría el riesgo si llegará a ocurrir. En caso de no llegar a ocurrir, el tiempo y recursos sobrantes que iría destinado a estos riesgos se dedicará a ampliar y mejorar las funcionalidades de los módulos.

Riesgo	Probabilidad (%)	Horas	Coste/hora (€/h)	Coste riesgo (€)	Coste a tener en cuenta (€)
Inexperiencia	70	15	9	135	95
Bugs	50	15	9	135	68
Desviación de la planificación	20	10	20	200	40
Problemas derivados de la pandemia	5	40	15	600	30
TOTAL					232

Tabla 27: Costes de riesgos. Fuente: Propia

10.1.4 Gastos de contingencia

Para finalizar, solo queda tener en cuenta los gastos de contingencia para imprevistos y riesgos no anticipados durante la estimación de los Costes de Personal por Actividad (CPA) y la estimación de los Costes imputados Genéricamente (CG), teniendo en cuenta que en el sector de la informática es de un 15%. A continuación, se especifican los detalles de este cálculo.

Gastos Contingencia	Precio estimado (€)	Porcentaje contingencia (%)	Gastos contingencia (€)
CPA	9392	15	1409
CG	331	15	50
TOTAL			1458

Tabla 28: Gastos de contingencia. Fuente: Propia

10.2 Presupuesto final

Finalmente, a partir de todas las estimaciones de costes realizadas obtendremos el presupuesto establecido para el desarrollo del proyecto, que se ha estimado de 11413€.

Tipo de gastos	Presupuesto estimado
CPA	9392
CG	331
Riesgos	232
Contingencia	1458
TOTAL	11413

Tabla 29: Presupuesto final estimado. Fuente: Propia

10.3 Control de gestión

Simplemente predecir los costes derivados de un proyecto y asumir que este será el coste final, no sirve para conseguir un correcto desarrollo, a lo largo del proyecto se ha de poder saber si las previsiones establecidas se siguen. Con el objetivo de saber si estas predicciones se van cumpliendo a lo largo del proyecto o se están produciendo desviaciones que podrían afectar al correcto desarrollo de este, los costes y horas previstos se compararán con los costes y horas reales de desarrollo, sirviendo estas comparaciones como indicadores que se irán consultando y actualizando para poder observar y ser consciente en todo momento del estado del desarrollo del proyecto, pudiendo tomar acciones y responder a estas desviaciones de la forma más oportuna si llegará a ser necesario. Estos indicadores son los siguientes:

Indicador	Método de cálculo
Desviación de tiempo por tarea	Horas estimadas – Horas reales
Desviación en el coste por tarea	Coste estimado tarea – Coste real tarea
Desviación en el CPA	CPA estimado – CPA real
Desviación en el CG	CG estimado – CG real
Desviación en los gastos de riesgos	Gastos de riesgo estimados – Gastos de riesgo reales
Desviación en los gastos de contingencia	Gastos de contingencia estimados – Gastos de contingencia reales
Desviación general del presupuesto	Presupuesto estimado – Presupuesto real

Tabla 30: Indicadores del estado del proyecto. Fuente: Propia

A lo largo del proyecto se han utilizado los indicadores, y no se han producido grandes cambios entre el coste establecido y el real, la diferencia se ha mantenido en todo momento dentro de un margen aceptable, puesto que la duración y las horas dedicadas al proyecto fueron establecidas previamente y se han mantenido.

11 Leyes y regulaciones

Con el paso de los años han ido surgiendo leyes para garantizar la seguridad de los usuarios y sus datos cuando utilizan dispositivos electrónicos con acceso a internet.

Hay que tener en cuenta el reglamento general de protección de datos que se aprobó en el parlamento europeo en 2016 [48]. Este reglamento especifica los derechos de los usuarios en lo que se refiere al tratamiento de los datos de carácter personal. Según este reglamento, las personas físicas deben dar el consentimiento explícito para la recogida y el procesamiento de información personal. También determina que los usuarios tienen derecho de actualizar y rectificar los datos de carácter personal, suprimirlos cuando lo deseen y exigir que no se utilicen sus datos en cualquier momento. Esta ley establece nuevos derechos de los usuarios. El derecho al olvido es un nuevo concepto que incorpora esta ley que determina que cualquier persona tiene derecho a la supresión de sus datos personales e impedir la difusión, también existe el derecho a la limitación del tratamiento de los datos, es decir, el usuario puede decidir qué se puede hacer y qué no con sus datos personales.

Los módulos desarrollados para la aplicación no utilizan los datos del usuario, ni piden ningún tipo de datos personal, se utiliza un identificador interno para identificar al usuario durante las conexiones a la base de datos y las peticiones a la API, que se obtiene al iniciar sesión con el usuario y contraseña. Sin embargo, aunque los módulos no traten datos personales, el módulo de pagos permite la realización de pagos a través de la aplicación, la cual cosa podría generar vulnerabilidades, pero para ello se basa del uso de un TPV virtual de Redsys que garantiza la seguridad de las operaciones, de forma que el único dato recogido es si la operación se ha realizado con éxito o han surgido inconvenientes, garantizando la confidencialidad de los datos bancarios de los usuarios.

12 Informe de sostenibilidad

La sostenibilidad en un proyecto *software* es muy importante, puesto que nos permite estimar los efectos del proyecto y su desarrollo en la sociedad que nos rodea, siendo conscientes de las consecuencias de nuestros actos. Por ello, a continuación, se explicarán los impactos derivados del desarrollo del proyecto y su uso en 3 ámbitos diferentes: ambiental, económico y social

Respecto a la dimensión económica, esta es la que más se había tenido en cuenta en el proyecto, puesto que la aplicación ha de generar rentabilidad económica de manera responsable y a largo plazo, continuando operativa una vez realizado el proyecto y con los módulos totalmente funcionales y ampliables.

En cuanto a la dimensión ambiental, puesto que se trata de una aplicación móvil se había visto por encima, suponiendo que mejora la situación actual porque reducirá en gran cantidad el uso de papel, tóner y otros recursos más tradicionales, pero se han de tener presentes el impacto energético que tiene su uso y desarrollo, puesto que esta energía es necesaria para proporcionar acceso a los dispositivos que la utilizan y esto conlleva un impacto ambiental indirecto derivado de las emisiones contaminantes generadas durante su producción.

Por lo que respecta a la dimensión social, el proyecto ha visualizado en todo momento una mejora de los escenarios y contextos sociales actuales relacionados con el tema tratado. La aplicación permitirá tener acceso constante a la información del centro, interactuar entre profesores, alumnos y padres, y gestionar el curso estudiantil de una forma más sencilla, teniendo en cuenta también que la aplicación sea accesible, de forma que personas con discapacidades sean capaces de utilizarla con las herramientas apropiadas.

Una vez resumidos los impactos derivados del desarrollo del proyecto y su uso, pasamos a una explicación en detalle de los mismos.

12.1 Impacto ambiental

En cuanto al beneficio ambiental que supondrá la realización de este proyecto, claramente, su realización tiene como beneficio ambiental que disminuirá el uso de tóner, papel y otros recursos más tradicionales en los centros, esto supone un gran avance, que disminuye en gran cantidad la contaminación producida por los centros. Puesto que el tóner está creado a base de numerosos compuestos químicos que, si no se reciclan adecuadamente, pueden contaminar la tierra o el agua. Además, la industria del papel es una de las mayores contaminantes del agua y el aire debido a las grandes cantidades de agua y energía consumidas durante la fabricación, y al extensivo uso de medios de transporte, emitiendo alrededor de 3,3 kg de CO2 por cada kilo de papel.

Por otro lado, el impacto ambiental negativo, principalmente, consiste en la energía necesaria para desarrollar el proyecto y acceder y utilizar la aplicación. Respecto a la reducción del impacto negativo, este no es fácilmente reducible, puesto que es derivado de la electricidad consumida para el desarrollo y utilización de la aplicación y las emisiones contaminantes derivadas de la producción de electricidad.

Posibles soluciones al impacto ambiental negativo serían la utilización de energías renovables, y posibles agravantes serían el uso de energías aún más contaminantes. Pero esto está fuera del alcance del proyecto.

En conclusión, el balance del impacto ambiental es altamente positivo, puesto que la cantidad de reducción del uso de recursos tradicionales contaminantes supera con creces el impacto negativo derivado de la producción y transporte de la electricidad necesaria para el desarrollo y uso de la aplicación, y se prevé que el impacto ambiental negativo decrezca con el tiempo, debido a la extensión del uso de energías renovables que está siendo promovida en los últimos años en muchos países, para disminuir los efectos negativos de la constante contaminación de nuestro planeta.

12.2 Impacto económico

El coste de la realización del proyecto, que ha sido calculado en apartados anteriores, es adecuado para un proyecto de esta escala, siendo un proyecto orientado de forma que sea sencillo de mantener y ampliar para un mayor rendimiento, y rentable, puesto que ha sido solicitado por los clientes, teniendo un mercado asegurado.

Actualmente las tutorías son gestionadas a través de notas, con los alumnos como intermediarios, o mediante llamadas, esto supone un mayor coste de contacto que el que supondría el uso de la aplicación, y no solo debido al coste de las llamadas o el papel y el tóner, sino también al tiempo extra que deben utilizar los empleados en realizar estas tareas y que podrían estar dedicando a otras más importantes.

Por otra parte, el módulo de pagos facilitará la gestión económica del centro, puesto que como actualmente solo está implementado de una forma básica, los tesoreros tienen que estar constantemente extrayendo la información de distintas páginas para hacer los cálculos y una implementación más extensa facilitará el trabajo de los contables y la gestión correcta de los pagos a través de la aplicación. Disminuyendo, una vez más el tiempo requerido para la gestión de estas tareas.

El sistema no debería tener un gran coste durante su vida útil aparte del necesario para mantener la aplicación en el servidor, Play Store y App Store para que pueda ser utilizado por los clientes, y estos costes serán ampliamente compensados por el pago de los centros para utilizar la aplicación.

Además, pocos escenarios que perjudicasen la viabilidad del proyecto podrían sucederse, puesto que la empresa está en constante contacto con los centros y usuarios de la aplicación, obteniendo *feedback* rápido y constante, y mejorando la aplicación cuando surge la oportunidad, teniendo una clientela bastante fiel, y que actualmente está en aumento. Por lo que antes de llegar a un punto en que no haya clientes de la aplicación, se deberían haber tomado las medidas necesarias para impedir esta situación.

Por otra parte, los módulos han sido diseñados para ser fácilmente mantenibles, escalables y ampliables, y se ha mantenido al tanto al resto de miembros del equipo de desarrollo de su funcionamiento, para que sea posible la continuación de su desarrollo una vez finalizado el proyecto.

Resumiendo, el hecho de no tener que utilizar papel y tóner o *software* de terceros para desempeñar las tareas que permiten realizar estos módulos que serán incluidos en la aplicación, la cual ya se estaba utilizando para la gestión del centro, permitirá reducir los costes desde el punto de vista de los clientes, y hará la aplicación más atractiva y completa, extendiendo el rango de potenciales clientes y aumentando los beneficios.

12.3 Impacto social

En cuanto a la dimensión social, este proyecto se ha desarrollado de forma que constituya una aplicación accesible. Estas son aquellas aplicaciones en las que todos los elementos que conforman la interfaz con la que interactúan los usuarios están preparados para garantizar su correcto acceso y uso a personas con diferentes capacidades. Y tiene como objetivo facilitar la gestión de las tareas tradicionales, adaptándolas a las nuevas tecnologías y facilitando y acelerando su desempeño.

Actualmente las tutorías son gestionadas a través de notas, con los alumnos como intermediarios, o mediante llamadas, esto dificulta la correcta comunicación y gestión entre padres y tutores, puesto que las notas no siempre llegan, y ponerse en contacto con los padres puede llegar a ser complicado. Mediante la aplicación, todo esto estará centralizado y será fácilmente accesible y gestionado, facilitando la tarea de los tutores y la comunicación con los padres.

Mientras que el módulo de pagos facilitará la gestión económica del centro, puesto que como actualmente solo está implementado de una forma básica, tienen que estar constantemente extrayendo la información de distintas páginas para hacer los cálculos y una implementación más extensa facilitará el trabajo de los contables y una gestión apropiada de los pagos a través de la aplicación.

Además, el hecho de poder centralizar la gestión del centro en una aplicación permitirá una mayor accesibilidad, comunicación y transparencia entre el centro y padres y alumnos.

En cuanto a la existencia de sectores que puedan ser perjudicados, estas son aquellas personas que no dispongan de dispositivos que permitan el acceso a la aplicación, puesto que estarán en desventaja respecto al resto de alumnos al no tener acceso a la aplicación, desde donde se gestiona toda la interacción con el centro.

En muchos sitios se está incluyendo la posibilidad de que los profesores hagan los trámites por el alumno cuando no tiene acceso, por ejemplo, los pagos se pueden marcar como pagados manualmente si el alumno le da el dinero al profesor, sin necesidad de utilizar la aplicación, y en el futuro está previsto que el tutor pueda reservar manualmente sus tutorías con alumnos y padres. Sin embargo, esto está pensado para casos especiales, ya que esta, es una aplicación que tiene pensado facilitar la gestión del centro, y no es recomendado su uso en centros en los que alumnos y padres no tienen acceso a los dispositivos necesarios para usar la aplicación, puesto que generaría situaciones de desigualdad y dificultaría la gestión del centro.

Para concluir, reiterar que este proyecto tiene una necesidad real, puesto que su desarrollo ha sido exigido por parte de los clientes actuales de la aplicación, alegando que el desarrollo de estos módulos les facilitará el trabajo, ahorrando parte del tiempo empleado en la gestión de tutorías y pagos, y pudiendo dedicarlo a otras tareas.

13 Conclusiones

Una vez explicados todos los aspectos del proyecto, comenzando por la formulación del problema, continuando con el diseño de la solución, prosiguiendo con la implementación de esta y acabando con la definición de costes temporales y económicos, y la sostenibilidad de la aplicación, llegamos a la finalización del trabajo de fin de grado. En este último apartado se detallarán las competencias genéricas y técnicas trabajadas, los resultados obtenidos, la perspectiva de futuro de los módulos y el trabajo desempeñado.

13.1 Competencias genéricas

Durante la realización de las prácticas se han trabajado una serie de competencias genéricas, que han sido necesarias para la correcta realización del proyecto. Las principales son:

- Comunicación eficaz oral y escrita: Al trabajar en equipo, y siguiendo la metodología SCRUM, había reuniones a diario y una comunicación constante con el resto del equipo de desarrollo, en la que era necesario comunicar de forma oral y escrita con otras personas conocimientos, procedimientos, resultados e ideas. Debatiendo la mejor solución a los problemas propuestos.

- Trabajo en equipo: En este proyecto he formado parte del equipo de desarrollo de Dinantia, y he contribuido de forma activa al desarrollo de la aplicación, aportando ideas y contribuyendo en todos los ámbitos de la aplicación, incluso aquellos que no afectaban directamente al TFG.

- Uso solvente de los recursos de información y aprendizaje autónomo: Como he explicado anteriormente, muchos de los conocimientos necesarios para desarrollar este proyecto no los poseía previamente al inicio de las prácticas, por lo que, aparte del soporte ofrecido por el equipo de desarrollo, me ha sido necesario gestionar la adquisición, la estructuración, el análisis y la visualización de datos y de información para obtener nuevos conocimientos necesarios en el desarrollo, ampliando mis conocimientos, aprendiendo nuevos métodos y tecnologías, y adaptándome a nuevas situaciones.

- Actitud adecuada ante el trabajo: Esta competencia engloba una gran cantidad de aspectos que son necesarios para un correcto desarrollo en el mundo laboral y que he me han sido necesarios durante el proyecto. Algunos ejemplos son: tener motivación para la realización profesional y para afrontar nuevos retos, sentirse motivado por la calidad y la mejora continua, actuar con rigor en el desarrollo profesional, tener una gran capacidad de adaptación, etc.

13.2 Competencias técnicas

Además de las competencias genéricas, también ha habido una serie de competencias técnicas, que han sido una parte imprescindible del correcto desarrollo del proyecto. Estas son:

- CES1.1 - Desarrollar, mantener y evaluar sistemas y servicios software complejos y/o críticos: El proyecto en sí ha consistido en la evaluación del sistema actual de Dinantia, y los requisitos de los clientes, para extender el sistema actual con el desarrollo de un módulo nuevo, y la ampliación de uno ya existente, que satisfagan las necesidades de los clientes, siendo desarrollados de forma que sean fáciles de mantener y ampliar.

- CES1.2 - Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles: Teniendo en cuenta la situación actual de Dinantia ha sido necesario evaluar la distribución que tiene actualmente, para aportar las mejores soluciones a los problemas propuestos, usando las estrategias, estándares y tecnologías disponibles.
- CES1.7 - Controlar la calidad y diseñar pruebas en la producción de software: Para las funcionalidades desarrolladas se han elaborado casos de prueba para que sea posible comprobar en cualquier momento que su funcionamiento es el esperado, y facilitar la detección de errores.
- CES2.1 - Definir y gestionar los requisitos de un sistema software: Previamente al desarrollo del proyecto ha sido necesario definir y gestionar los requisitos de los clientes y los *Product Owners*, para elaborar una solución centrada en satisfacer estos requisitos de la mejor manera posible, y durante el desarrollo se ha adaptado el producto a los nuevos requisitos que han ido apareciendo.
- CES3.1 - Desarrollar servicios y aplicaciones multimedia: El proyecto desarrollado consiste en una aplicación web y móvil con una gran diversidad de funcionalidades, constituyendo una aplicación multimedia.

13.3 Resultado

En la fase inicial se concretaron el alcance que tendría el proyecto, y que se quería conseguir como objetivo para la fase final. Actualmente, con el desarrollo del proyecto finalizado, puedo confirmar que, a pesar de los cambios de planificación y de enfoque del proyecto, se ha alcanzado el objetivo establecido, obteniendo un módulo de tutorías completamente funcional y muy completo, y ampliando el módulo de pagos mejorando su utilidad.

Ambos módulos ya están en el mercado y siendo utilizados por los clientes de Dinantia, ambos cumplen con los objetivos establecidos previamente, y de cara al proyecto pueden considerarse completos. Aun así, esto no quiere decir que no haya margen para su crecimiento, al fin y al cabo, ambos módulos seguirán evolucionando junto a las necesidades de los clientes, adquiriendo más funcionalidades.

13.4 Futuro

A lo largo del proyecto, se han debatido una gran cantidad de funcionalidades, de las cuales se ha hecho una priorización teniendo en cuenta diversos factores. Concretamente, se ha priorizado la obtención de un módulo de tutorías totalmente funcional, con todas aquellas características que eran imprescindibles, y con el tiempo restante se han añadido algunas que no eran imprescindibles, pero que aportaban una gran utilidad. Por otro lado, en el módulo de pagos se han añadido funcionalidades que permitieran mostrar la información de los pagos de una forma más representativa y permitiera la interacción con estos, pero al haberse priorizado el módulo de tutorías, muchas de las ideas debatidas no se han podido implementar y se han reservado para más adelante.

Algunas de las mejoras para ambos módulos que se pensaron, pero no se llegaron a implementar son:

- Reserva manual de tutorías: El profesor será capaz de realizar la reserva manual de tutorías para un alumno o sus padres, facilitando la reserva en caso de que, por cualquier motivo, estos no puedan o quieran realizar el trámite a través de la aplicación.
- Establecer el tipo de tutoría: El tutor podrá definir franjas de tutorías que sean online, presenciales o a escoger, permitiendo a los alumnos seleccionar aquellas que se adapten a sus necesidades, y a los tutores establecer como se realizarán las tutorías.

- Etiquetado de pagos: Se podrán crear etiquetas y asignarlas a los pagos, asociando múltiples pagos a un mismo tema y permitiendo una asociación de pagos por diversas categorías muy potente.
- Adición de filtros de pagos por etiqueta: En las pantallas en las que se muestre información sobre los pagos se añadirá la posibilidad de utilizar un filtro por etiquetas, mostrando solo los pagos que tengan alguna de las etiquetas seleccionadas.
- Creación de gastos: Los tesoreros podrán crear gastos y asignarles etiquetas, permitiendo obtener un balance más completo del estado económico del centro.

Estas fueron comentadas he incluso creadas en Jira, la herramienta utilizada para gestionar las tareas, para su desarrollo en el futuro.

13.5 Conclusión final

Mi experiencia personal del proyecto ha sido muy positiva. Me ha sido posible aprender una gran cantidad de nuevas tecnologías y, siendo mi primera experiencia en un entorno laboral real, he podido disfrutar el poder trabajar junto a un equipo de desarrolladores, con *Product Owners* y clientes reales, que tienen requisitos y especificaciones reales, para las que el proyecto ha de aportar una solución, manteniéndome motivado durante todo el proyecto.

El proyecto, aparte de ampliar mi conocimiento, me ha permitido aprender a organizarme mejor y hacer estimaciones más precisas de las tareas de desarrollo, esto se puede apreciar en la definición de tareas y en las planificaciones temporales del proyecto, que inicialmente eran más imprecisas, con tareas más abiertas, y a medida que ha ido avanzando el proyecto se han definido tareas muy concretas y se ha estimado el tiempo con gran precisión, esto ha sido posible gracias al contacto con las tecnologías utilizadas, puesto que me han aportado la información y experiencia necesarias para estimar el coste real de las tareas.

Finalmente, me gustaría recalcar que la realización de este proyecto, me ha permitido ver de primera mano, que los informáticos son capaces de utilizar los conocimientos obtenidos a lo largo de su carrera, para facilitar y mejorar la vida y las tareas de las personas, cambiando el mundo que nos rodea. Algo que me parece maravilloso y me ha motivado más allá de este proyecto.

14 Bibliografía

[1] Dinantia. URL: https://www.dinantia.com/. [Último acceso: 20 03 2021].

[2] Educación 3.0. URL: https://www.educaciontrespuntocero.com/recursos/plataformas-gestion-escolar/. [Último acceso: 06 03 2021].

[3] Additio. URL: https://www.additioapp.com/es. [Último acceso: 06 03 2021].

[4] aGora. URL: https://www.agora-erp.com/. [Último acceso: 06 03 2021].

[5] Alexia. URL: https://www.alexiaeducaria.com/. [Último acceso: 06 03 2021].

[6] Cifra Educación. URL: https://www.cifraeducacion.com/. [Último acceso: 06 03 2021].

[7] Classlife. URL: https://www.classlife.education/. [Último acceso: 06 03 2021].

[8] Wikipedia: Accesibilidad Web. URL: https://es.wikipedia.org/wiki/Accesibilidad_web. [Último acceso: 27 02 2021].

[9] Google Play: Educación Once Familias. URL: https://play.google.com/store/apps/details?id=com.dinantia.once&hl=es_419&gl=US. [Último acceso: 21 03 2021].

[10] Proyectos Agiles: Scrum. URL: https://proyectosagiles.org/que-es-scrum/. [Último acceso: 28 02 2021].

[11] Wikipedia: Desarrollo Ágil. URL: https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software. [Último acceso: 28 02 2021].

[12] Wikipedia: Jira. URL: https://es.wikipedia.org/wiki/Jira. [Último acceso: 28 02 2021].

[13] Atlassian: Gitflow. URL: https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow. [Último acceso: 28 02 2021].

[14] Wikipedia: Git. URL: https://es.wikipedia.org/wiki/Git. [Último acceso: 28 02 2021].

[15] Wikipedia: Bitbucket. URL: https://es.wikipedia.org/wiki/Bitbucket. [Último acceso: 01 03 2021].

[16] Atlassian: Bitbucket tutorial. URL: https://www.atlassian.com/es/git/tutorials/learn-branching-with-bitbucket-cloud. [Último acceso: 19 03 2021].

[17] Wikipedia: Slack. URL: https://es.wikipedia.org/wiki/Slack_(software). [Último acceso: 01 03 2021].

[18] Wikipedia: Google Meet. URL: https://es.wikipedia.org/wiki/Google_Meet. [Último acceso: 01 03 2021].

[19] Visual Paradigm. URL: https://www.visual-paradigm.com/. [Último acceso: 20 06 2021].

[20] Wikipedia: Ionic. URL: https://en.wikipedia.org/wiki/Ionic_(mobile_app_framework). [Último acceso: 27 02 2021].

[21] Ionic Framework. URL: https://ionicframework.com/. [Último acceso: 27 02 2021].

[22] Wikipedia: Patrón Model-Vista-Controlador. URL: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador. [Último acceso: 25 05 2021].

[23] Wikipedia: Patrón Modelo-Vista-Modelo de vista. URL: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel. [Último acceso: 25 5 2021].

[24] API REST. URL: https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/. [Último acceso: 25 05 2021].

[25] Wikipedia: Patrón observador. URL: https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o). [Último acceso: 26 5 2021].

[26] PHP. URL: https://www.php.net/manual/es/intro-whatis.php. [Último acceso: 13 6 2021].

[27] HTML, CSS y Javascript. URL: https://geoinnova.org/blog-territorio/html-css-y-javascript-lenguajes-para-el-desarrollo-de-paginas-web/. [Último acceso: 13 5 2021].

[28] Wikipedia: HTML. URL: https://es.wikipedia.org/wiki/HTML. [Último acceso: 15 6 2021].

[29] SeekLogo. URL: https://seeklogo.com/. [Último acceso: 15 6 2021].

[30] Sass. URL: https://styde.net/aprende-sass-caracteristicas-principales/. [Último acceso: 14 6 2021].

[31] Wikipedia: Typescript. URL: https://es.wikipedia.org/wiki/TypeScript. [Último acceso: 14 5 2021].

[32] CakePHP. URL: https://book.cakephp.org/. [Último acceso: 14 5 2021].

[33] Vue. URL: https://codingpotions.com/que-es-vue. [Último acceso: 14 05 2021].

[34] Vue.js. URL: https://012.vuejs.org/guide/. [Último acceso: 15 06 2021].

[35] Wikipedia: Angular. URL: https://es.wikipedia.org/wiki/Angular_(framework). [Último acceso: 15 06 2021].

[36] Ionic. URL: https://profile.es/blog/que-es-ionic/. [Último acceso: 15 05 2021].

[37] Bootstrap. URL: https://getbootstrap.com/. [Último acceso: 15 6 2021].

[38] OpenWebinars: MySQL. URL: https://openwebinars.net/blog/que-es-mysql/. [Último acceso: 16 6 2021].

[39] Datatables. URL: https://datatables.net/. [Último acceso: 16 6 2021].

[40] date-fns. URL: https://date-fns.org/. [Último acceso: 16 6 2021].

[41] ProgrammerClick: date-fns. URL: https://programmerclick.com/article/18921553684/. [Último acceso: 16 6 2021].

[42] Glassdoor. URL: https://www.glassdoor.es. [Último acceso: 11 03 2021].

[43] BOE (Boletín Oficial del Estado). URL: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2020-1626#:~:text=Duraci%C3%B3n.,horas%20efectivas%20reales%20de%20trabajo. [Último acceso: 11 03 2021].

[44] Glassdoor: Salario Product Owner. URL: https://www.glassdoor.es/Sueldos/barcelona-product-owner-sueldo-SRCH_IL.0,9_IM1015_KO10,23.htm. [Último acceso: 13 03 2021].

[45] Glassdoor: Salario full-stack developer. URL: https://www.glassdoor.es/Sueldos/barcelona-full-stack-developer-sueldo-SRCH_IL.0,9_IM1015_KO10,30.htm. [Último acceso: 13 03 2021].

[46] Bitbucket: Precios. URL: https://www.atlassian.com/es/software/bitbucket/pricing. [Último acceso: 12 03 2021].

[47] Microsoft 365: Precios. URL: https://www.microsoft.com/es-es/microsoft-365/buy/compare-all-microsoft-365-products. [Último acceso: 12 03 2021].

[48] Wikipedia: Reglamento General de Protección de Datos. URL: https://es.wikipedia.org/wiki/Reglamento_General_de_Protecci%C3%B3n_de_Datos. [Último acceso: 22 05 2021].

[49] RedHat: API. URL: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces. [Último acceso: 23 05 2021].

[50] Atlassian: Jira. URL: https://www.atlassian.com/es/software/jira. [Último acceso: 19 03 2021].