UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Estudio y desarrollo de un sistema de gestión educativo con tecnologías libres

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Palomares Badenas, Raúl

Tutor: Pelechano Ferragud, Vicente

Curso 2019-2020

Resumen

La irrupción de las tecnologías de la información y la comunicación y la integración de la informática en la sociedad del siglo XXI ha dado lugar a una revolución que afecta ya a todos los aspectos de nuestra vida. La automatización e informatización de los procesos que antaño eran manuales permite ahora trabajar con mayor rapidez y menor esfuerzo. Un área que podría verse profudamente afectada por este cambio es la del sector educativo. El empleo de nuevas herramientas y estrategias de aprendizaje basadas en la informática abre nuevas oportunidades tanto para estudiantes como para profesores, acerca las TIC a los más jóvenes y también ofrece nuevas perspectivas y enfoques con los que modernizar la manera de impartir clase y realizar las tareas de docencia.

De todos quienes podrían verse afectados por este cambio en el paradigma educativo, el profesorado es el colectivo menos habituado a la inclusión de la informática en su trabajo: A pesar de que sus técnicas han ido incorporando nuevas tecnologías en su día a día, el cambio no ha sucedido con la misma celeridad que en otros sectores o grupos de usuarios. Mientras que los alumnos, una población más joven, ha crecido familiarizada con Internet y la informática; parte de los docentes no han tenido el mismo acceso a estos recursos y, por tanto, son más propensos a sufrir las desigualdades provocadas por la brecha tecnológica.

También cabe tener en mente que la educación, un área tan relevante como es la formación de las generaciones venideras, se arraiga profundamente en la enseñanza pública, especialmente en Europa. En este aspecto, la aplicación de tecnologías también libres, sin coste y públicas sería una garantía de la modernización de este estamento, a la vez que se promoverían los valores del bienestar y la cooperación entre docentes, alumnos e instituciones.

Actualmente existen plataformas conocidas como *Course Management Systems* que procuran aplicar estas técnicas y transportan parte del proceso educativo a nuevas aplicaciones como son las páginas web o las *apps*. Explorándolas podemos determinar qué carencias y aciertos han cometido al aproximar las TIC al entorno educativo. Teniendo a mano toda esta información, podríamos evaluar el estado actual de estas herramientas, cómo de próximas y útiles son para quienes las usan y, finalmente, construir una aplicación que persiga estos mismos objetivos y procure aprender de sus errores.

Palabras clave: Tecnologías de la información, educación, profesorado

Resum

La irrupció de les noves tecnologías de la informació i la comunicació i la integració de la informàtica en la societat del siglo XXI ha produït una revolució que afecta ja a tots els aspectes de la nostra vida. La automatització i informatització dels processos que antany eren manuals permiteix ara treballar amb major rapidesa y menor esforç. Una àrea que podria veure's profudament afectada per aquest cambi és la del sector educatiu. L'ús de noves ferramentes i estratègies d'aprenentatge basades en la informàtica obri noves oportunitats tant per als estudiants como per als profesors, apropa les TIC als més joves i també ofereix noves perspectives y enfocaments amb els que modernitzar la manera d'impartir classe i realitzar les tasques de docència.

De tots els qui podrien veure's afectats per aquest cambi en el paradigma educatiu, el professorat és el col·lectiu menys habituat a la inclusió de la informàtica en el seu treball: Encara que les seues tècniques han anat incorporant noves tecnologies en el seu dia a dia, el canvi no ha succeït amb la mateixa celeritat que en altres sectors o grups d'usuaris. Mentre que els alumnes, una població més jove, ha crescut familiaritzada amb Internet i la informàtica; part dels docents no han tingut el mateix accés a aquestos recursos i, per tant, són més propensos a sofrir les desigualtats provocades per la bretxa tecnològica.

També cal tindre en ment que la educació, un àrea tan relevant como és la formació de les generacions esdevenidores, s'arrela profundament en l'ensenyament públic, especialment en Europa. En aquest aspecte, la aplicació de tecnologies també lliures, sense cost i públiques seria una garantia de la modernització d'aquest estament, alhora que es promoverien els valors del benestar i la cooperació entre docents, alumnes i institucions.

Actualment existeixen plataformes conegudes com *Course Management Systems* que procuren aplicar aquestes tècniques i transporten part del procés educatiu a noves aplicacions com són les pàgines web o les apps. Explorant-les podem determinar quines manques i encerts han comés en aproximar les TIC a l'entorn educatiu. Tenint a mà a mano tota aquesta informació, podríem evaluar l'estat actual d'aquestes ferramentes, cóm de pròximes y útils són per als qui les usen i, finalment, construir una aplicació que persiga aquests mateixos objectius i procure aprendre dels seus errors.

Paraules clau: Tecnologies de la informació, educació, profesorat

Abstract

The irruption of information and communication technologies and the integration of computer science in the 21st century society has brought a revolution that already affects all the aspects of our lives. The automatisation and computerisation of formerly manual processes allows for a faster and less demanding workload. The educative system is an area that could be deeply affected by this change. The usage of new tools and learning strategies based on computer science opens new paths both for students and teachers alike, brings ICTs closer to young people and offers new perspectives and approaches that could be used to modernise the way lessons and other teaching tasks are done.

Out of all those who could see themselves affected by this change in the educational paradigm, the teaching staff is the lest used to the inclusion of computer science in their jobs: Although their techniques have been integrating new technologies in their day to day live, this change is taking place at a slower rate when compared to other sectors or user groups. While students, a younger population, is familiarized with the Internet and computer science; the teaching staff did not have the same access to these resources and, as a result, are more proclive to suffer from disparities produced by the so called "technical breach".

We shall also keep in mind that education, an area so relevant that involves the training of the forthcoming generations, is deeply rooted in the public education, specially in Europe. In that regard, the application of *libre*, free and public technologies could ensure the modernisation of this estate, as well as promoting the values of wellbeing and cooperation between teachers, students and institutions.

Course Management Systems are platforms known to apply these techniques nowadays, and they allow for the application of newer tools such as webpages or apps. We could determine their weaknesses and strengths in their approaches to bring the ICTs into the educational environment by exploring them. Knowing all this information, we could evaluate these tools' current situation to determine how close and handy are for those that use them and, finally, we could build an application that follows the same objectives and learns from their predecessors' mistakes.

Key words: Information and computer technologies, education, teaching staff

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

El auge de las TIC en el ámbito educativo ha dado lugar a una manera diferente de concebir la enseñanza. Desde páginas de cursillos online hasta extensas plataformas utilizadas por centros y universidades, ha provocado una revolución sin precedentes que sitúa el acceso al conocimiento a un click de distancia. Este campo se está adaptando lentamente a un cambio social que aproxima la informática a los usuarios de a pie, aunque los últimos tiempos han contribuido a que irrumpa con mucha más fuerza. Se estima que la educación online moverá cerca de 350 Billones de dólares en 2025[2], lo que lo convierte en un posible filón favorecido por las recientes circunstancias desencadenadas por el COVID-19.

Con la intención de poder llevar a cabo esta ardua tarea nacen los *Course Management System* (CMS), sistemas de gestión educativos, herramientas diseñadas para estudiantes y profesores y destinadas a organizar el contenido de los cursos, proveer de medios para evaluar al alumnado y facilitar el acceso a la información y contenidos propios del curso y el centro. A grandes rasgos, los CMS contienen varias utilidades:

- **Organizativas**: Gestión de cursos o asignaturas
- **Comunicativas**: Profesor-alumnos, profesor-padres, sistemas de difusión
- **Evaluativas**: Marcaje de notas, exámenes en la plataforma, correcciones automáticas
- **Informativas**: Datos del curso, horarios
- **De acceso a materiales**: Diapositivas, documentación, etc.

Estas funciones se ponen a disposición tanto de quienes imparten docencia como quienes la reciben, y permiten la transmisión y acceso al conocimiento con mayor facilidad aprovechando la tecnología disponible a día de hoy.

Sin embargo, alojar un CMS no es tarea sencilla: Requiere de cierto capital, de personas capacitadas para mantener el sistema y también de aquellos que puedan adaptarlo a las circunstancias y características de cada curso o centro. Sustentar tales plataformas no es barato, pues el alojamiento de la misma por medios propios o a través de servicios externos puede suponer un importante desembolso económico no siempre asumible por aquellos que podrían beneficiarse de su uso.

Con la finalidad de dar una respuesta a esta creciente demanda han aparecido cientos de proyectos que logran satisfacer estas necesidades en mayor o menor medida – proyectos que o bien funcionan por suscripciones a un servicio o que habilitan la posibilidad de disponer de estos sistemas con un coste muy reducido o incluso de forma gratuita.

Son dignos de mención Sakai (el servicio utilizado por PoliformaT, de la Universitat Politècnica de València) y Moodle, proyectos de renombre empleados en la educación pública en la gran mayoría de universidades; pero también tenemos ejemplos como ITACA[3], autóctono y desarrollado por la Generalitat Valenciana para su uso en centros públicos.

Uno de los inconvenientes que implican estos sistemas es su dificultad de empleo. Sea por su extensión o por sus características, algunos de sus usuarios pueden encontrar obstáculos en su utilización: usuarios con menor experiencia en las TIC, poco acostumbrados a la tecnología; etc.

Es por ello que, utilizando estos modelos como ejemplo, podemos llevar a cabo un análisis que que nos permita determinar sus defectos y virtudes, sus características más relevantes y útiles; de tal forma que se puedan extraer cuáles son las propiedades que les han llevado a ocupar la relevancia que sustentan a día de hoy. También, con esta crítica, podría construirse una plataforma alternativa que procure solventar los defectos que presenten.

Para ello, es crucial analizar cuál es el entorno en el que se desenvuelven sus potenciales usuarios: estudiantes, profesores, miembros de los equipos administrativos... Si bien la aplicación debe cumplir su fin, también debe facilitar su uso y proveer de medios para instruir a quienes operan con ella.

No cabe olvidar que el dinero puede ser una barrera que impida que ciertos miembros de la comunidad tengan acceso a este tipo de herramientas. Es por ello que desarrollar tal sistema tiene que hacerse teniendo en mente que su uso debe ser gratuito, abierto, libre para que cualquier gobierno, institución, individual...; pueda emplearlo para su beneficio y el de sus usuarios.

1.1 Motivación

La Conselleria d'Educació i Esport de la Comunitat Valenciana desarrolló a lo largo de la pasada década un *CMS* conocido como ITACA[3]. La finalidad de su implementación era interconectar los distintos miembros que forman parte del tejido educativo de la comunidad[4], sean alumnos o profesores, y centralizar toda su información en un único lugar. Gracias a ello, los centros pueden transmitir evaluaciones, matriculaciones y otros procesos burocráticos directamente a un servidor central. Beneficiándose de esta centralización, el expediente educativo de un alumno puede ser compartido, transmitido y actualizado entre centros, pudiendo simplificar los procesos de gestión de matriculaciones, registros de asistencias, PGAs y otros menesteres.

Si bien su introducción supuso un notable cambio a mejor, ITACA no está exento de críticas. El sistema cuenta con restricciones de acceso que impiden su uso habitual y relegan a que el profesorado pueda ver limitado su uso a espacios de tiempo concretos, tales como son los fines de trimestre. Teniendo la suerte de ser hijo de docentes, nos ha sido posible observar de primera mano cómo es el uso de la aplicación y también de conocer la opinión sobre la herramienta directamente de los profesores que la emplean a diario. Tal como nos fue descrita, esta plataforma tenía problemas de usabilidad, limitaba el acceso a los profesores y era propensa a producir fallos durante su ejecución.

Siendo ésta una aplicación que marcó un antes y un después y que posee una cierta antigüedad, estas carencias o deficiencias son razonables y comprensibles. Sin embargo, eso no la exime de poder solventar sus defectos, es posible corregirlos o construir sobre esta misma idea. Es por ello que, habiendo dedicado estos años de carrera al aprendizaje y desarrollo software, me veo en la obligación de analizar la herramienta y tratar de cons-

truir, en la medida de lo posible, un CMS que persiga los mismos objetivos que ITACA buscaba en sus inicios, pero corrigiendo sus defectos e incorporando nuevas tecnologías.

También es relevante que ITACA es un proyecto público, pero su código no lo es. En la época en la que ITACA apareció, los movimientos de las comunidades de código libre (*Free/Libre Software*) y código abierto (*Open Source*) no tenían la relevancia y visibilidad de la que gozan a día de hoy.

Si se desea construir un sistema similar a ITACA que pudiera ser empleado en la escuela pública y pagado o financiado también por medio de dinero público, se considera que el código de la aplicación también debería ser público y su uso, gratuito. Hacerlo así respondería a los movimientos que existen actualmente en la Unión Europea, que abogan por la publicación y uso de tecnologías y software de código libre abierto[6], así como de publicar todo software financiado por dinero público a través de licencias *FOSS* (Free, Open Source Software)[7].

Como último detalle, el mero hecho de publicar esta solución bajo licencias de código libre supone que cualquier persona puede observar el código, emplearlo y modificarlo a su gusto. Tal hecho abre la puerta a que un CMS publicado como FOSS pueda ser extendido de manera desinteresada y gratuita por millones de personas en todo el mundo. También, su uso sería gratuito, lo que lo convertiría en una solución no sólo económica, sino también práctica para que cualquier individuo, institución o gobierno sea libre de emplear la herramienta bajo su criterio y satisfaciendo sus fines. Eliminando esta barrera económica se espera que también se pueda acercar el conocimiento y la tecnología aplicando técnicas modernas a países o comunidades donde el uso de alternativas, CMS de pago, fuera imposible.

A su vez, si se plantea que otros desarrolladores puedan contribuir en su creación y construcción, sería crucial y relevante que la estructuración del código facilite y simplifique los cambios a llevar a cabo. Incluir garantías, tales como tests y documentación, permitirán que trabajar en la herramienta y realizar cambios en ella sea una experiencia amena en lugar de hostil.

No cabe olvidar tampoco que si se logra que el software sea mantenible, modificable y adaptable a distintos entornos y circunstancias; más personas podrían emplearlo y, a su vez, aportar su grano de arena al proyecto.

Bajo este pretexto nace **Dagaz**, el resultado de este análisis y del desarrollo de una herramienta que pueda recoger el testigo de ITACA. Entraremos en detalle más adelante.

1.2 Objetivos

ITACA ha sido el detonante y fuente de inspiración para el desarrollo y escritura del siguiente proyecto, es por ello que con este trabajo se pretende estudiar el estado del arte de los CMS y poder detectar cuáles son las características relevantes y deseables en CMS de renombre, utilizando tecnologías libres y presentando su resultado también a través de licencias libres para el beneficio de la comunidad educativa y, en definitiva, de cualquier persona que desee emplear esta pieza de software.

Si bien a día de hoy ya existen CMS que cumplen parte de estas características, la mayoría son demasiado complejos, su mantenimiento es dificultoso por la extensión de su código o su uso dista de ser intuitivo y requiere de entrenamiento y conocimiento previo para utilizar la herramienta. Con **Dagaz** se plantea dar respuesta a este problema con una aplicación más sencilla, más directa y fácil de usar de cara al usuario, pero manteniendo cierto equilibrio y favoreciendo que su código sea alterado y extendido con poco esfuerzo.

Para ello se especificaron las siguientes metas:

- Construir un CMS mediante tecnologías libres
- Lograr que el código sea fácilmente modificable, legible y esté documentado
- Lograr una amplia cobertura de tests que validen la solución
- Contar con interfaces intuitivas y que faciliten el uso a sus usuarios
- Facilitar las tareas del profesorado (marcaje de notas, asistencias, etc.)
- Lograr que el sistema pueda ejecutarse en diferentes entornos sin perder sus capacidades

1.2.1. Impacto esperado

En un primer lugar, este proyecto tiene como público objetivo a los particulares, centros y gobiernos que requieran una solución maleable y simple para gestionar los cursos educativos. Dado que se contempla siempre como una aplicación completamente gratuita, las ganancias económicas no se verían a raíz de lo que esta herramienta generara, sino por los recursos que ahorraría. La ausencia de *royalties* o licencias de uso lo posiciona como una opción competitiva frente a otras soluciones de pago.

Como punto fuerte, su modificabilidad propicia que el dinero que en otras aplicaciones se estaría destinando al pago de tasas pudiera ser dedicado a la personalización y adaptación de la herramienta a las necesidades particulares de quienes la usen, generando puestos de trabajo y permitiendo que los cambios que se produzcan como resultado de ello puedan ser también empleados por otros miembros de la comunidad.

Objetivos para el Desarrollo Sostenible

Siguiendo los Objetivos de Desarrollo Sostenible de la Unión Europea[8], **Dagaz** cumpliría con 6 de las 17 metas:

- **Educación de calidad**, permitiendo el acceso a la educación a comunidades con menos recursos y asentando los nuevos estándares educativos basados en las TIC
- **Trabajo decente y crecimiento económico**, pues el propio diseño de la aplicación propicia a la contratación de desarrolladores que adapten la infraestructura a los casos específicos de cada implementación, generando puestos de empleo
- **Industria, innovación e infraestructura**, aportando nuevas soluciones para los CMS que sean más austeras, pero también flexibles y eficientes
- **Reducción de las desigualdades**, pues el hecho de facilitar el acceso a la educación, así como hacer gratuito su uso, descarga y redistribución; permite eliminar barreras
- **Producción y consumo responsables**, dado que la aplicación es modificable y puede ser preparada y adaptada según la cantidad de docentes y alumnos que se espera para la misma. Igual que puede desplegarse en un servidor de alta capacidad y emplearse en centros de miles de alumnos, también podría ejecutarse en computadores discretos

- **Alianza para lograr los objetivos**, pues al asentarse sobre los principios FOSS, propicia la cooperación entre distintos individuales, incluso centros o gobiernos que pudieran participar activamente en su desarrollo

Tipos de usuarios

Podemos englobar a los potenciales usuarios de la aplicación en 3 grupos:

- **Profesorado/Docentes**, personas que se encargan de impartir clases o que participan directa o indirectamente en ella. Este conjunto se beneficiaría teniendo acceso a herramientas evaluativas, controles de asistencia, registros horarios de sesiones y actividades o pruebas. En última instancia, también podrían ser los encargados de subir contenido a la aplicación para que los alumnos tengan acceso a él.
- **Equipo directivo/administrativo**, que verían satisfechas sus necesidades a través de procesos que faciliten la matriculación de alumnos, el intercambio de expedientes, la creación de cursos, asignaturas e incidencias.
- **Alumnado/Estudiantes**, aquellos a quienes se imparte docencia. Este sector se vería favorecido teniendo acceso a sus evaluaciones, calendario escolar y contenidos del curso a través de la plataforma. Asimismo, sus padres o tutores legales estarían interesados en tener información sobre el curso y su desempeño.

1.2.2. Metodología

Inicialmente, este proyecto se diseñó con la idea de poder llevar a cabo evaluaciones del producto con profesores de centros públicos a través de pruebas de aceptación. Sin embargo, el colectivo para el que va dirigido, el profesorado, se ha visto fuertemente azotado por el efecto del COVID-19, lo que ha impedido que estas pruebas se hayan hecho de forma presencial y han reducido en gran medida la cantidad de personas que han tenido la oportunidad de dar su opinión sobre el uso de la herramienta.

El proyecto se llevó a cabo definiendo dos grandes bloques: Una fase de investigación para recabar información relevante de proyectos de CMS similares y un periodo de desarrollo.

- La **investigación** otorgaría como resultado una lista de tecnologías relevantes para construir una solución de este calibre, así como las propiedades del sistema, las características deseables e implementables que pudieran haberse identificado en otras herramientas del mismo campo, el listado de usuarios y una serie de riesgos que podrían surgir durante su consecución.
- Por otro lado, durante el **desarrollo** se producirían diversos entregables incrementales que serían evaluados por un set de *testers* o usuarios de prueba que utilicen CMS. Éstos llevarían a cabo pruebas de aceptación y aportarían su opinión comparando cada entregable y su funcionalidad con aquella de su CMS de preferencia.

Scrum ha sido el método elegido para la construcción de **Dagaz**. Siendo una aplicación de la filosofía *Agile*, facilita la descripción del proceso de desarrollo, y permite la producción periódica de productos funcionales con una complejidad y funcionalidad que tiende a crecer conforme progresa la herramienta.

El COVID-19 ha afectado con contundencia al grupo de usuarios a quienes destinamos este trabajo, impidiendo que el proceso de desarrollo pudiera involucrarlos en

mayor medida. Es por ello que, a fin de aportar cierta estabilidad y mitigar así las incertidumbres inherentes a la pandemia y el confinamiento, se optó por seccionar la aplicación en dos grandes entregas que sucederían una vez por mes. Se barajó la posibilidad de seccionarlo en *sprints* de 2 semanas, pero la dificultad para reunir a los candidatos que realizarían los ensayos inclinó la balanza en pos de extender este tiempo.

El equipo encargado de llevar a cabo esta tarea se compone únicamente de una persona, con lo que los diferentes roles que Scrum propone están aglutinados en un solo responsable que debe mediar y tomar las decisiones adecuadas para que el proyecto llegue a buen puerto.

Se ha optado por un desarrollo próximo a las directrices del *Test Driven Development*, una estrategia que consiste en describir inicialmente cuál sería el comportamiento esperado de la aplicación a través de tests y, utilizándolo como cimiento, construir el código que atiende a esas características. Estos test actúan de guía y son derivados de los criterios de aceptación de cada tarea. Gracias a ello, se puede certificar que cuando ésta concluye existe una garantía de haber desarrollado una solución que satisface las necesidades indicadas.

De darse cualquier cambio dentro de esos criterios, los test sobre los que se cimienta la solución cambiarían, y con ellos también lo hace el código. El TDD contribuye a verificar que la aplicación funcionará tal y como se desea, y facilita el refactorizado de código a la vez que certifica que se cumple con las características solicitadas. Digno de mención es que esta técnica dota de tests que simplifican el desarrollo posterior y actúan como muro de contención para detectar incidencias o acciones no deseadas dentro de la aplicación, una característica muy deseada en proyectos de código abierto y que atrae a potenciales contribuyentes.

CAPÍTULO 2
Estado del arte

Los CMS nacen en la década de los 90 con la intención de gestionar cursos de forma *online*. La adopción de internet en la computación doméstica y educativa propició la aparición de soluciones que buscaban describir las actividades propias del control y enseñanza de un curso académico. Su objetivo principal es dotar a sus usuarios de herramientas que simplifiquen emplear esta tecnología en la educación.[1]

2.1 Funciones

A grandes rasgos, un CMS ideal sería un espejo de los cursos tradicionales, incorporando en sí aquellas funciones que el profesorado, sus alumnnos o los encargados de definir el sistema y tratar con sus datos; pudieran necesitar para construir y llevar a término un curso académico, pero incorporando facilidades que permitan sortear los obstáculos o dificultades que ésto supone.

Entre sus funciones principales estaría:

- Presentar contenido y documentación
- Permitir la comunicación entre los usuarios (sea docente-alumno, docente-docente...)
- Evaluar y analizar el rendimiento académico de los estudiantes
- Marcar y reportar las evaluaciones de los alumnos
- Gestionar el material académico y las actividades que se lleven a cabo en el curso

Definir sus características es complicado y específico para cada caso. No todos los CMS necesitan una *suite* completa de operaciones para ser útiles, algunos pueden focalizar más en puntos de gestión y tratamiento de datos (evaluaciones, estadística para análisis de rendimiento educativo), de disposición de contenidos (repositorios de material educativo, bibliotecas online) u otros elementos que puedan resultar relevantes de acuerdo a la casuística y el entorno donde se desea aplicar.

Gracias a la aparición de nuevos servicios, los CMS tienden a seguir estándares y adaptarse para incorporar funcionalidades cedidas o proporcionadas por agentes externos, tales como podrían ser integraciones con otras páginas web, *suites* de programas de oficina y demás.

2.2 Coste

El coste de un CMS, al igual que cualquier proyecto de software, viene marcado principalmente por varios factores[9]:

- Las **licencias de uso**, que solo aplica a aquellos casos en los que la utilización del software no sea gratuita. Puede tratarse de un permiso de uso, de un desembolso económico periódico o vinculado al número de usuarios a quienes se dará servicio
- Los **costes asociados a su uso**. Entran en esta categoría el capital destinado al mantenimiento del sistema, así como el equipo que lo desarrolla y gestiona
- El ***hosting*** o **alojamiento**. Dependiendo de la extensión del sistema, las soluciones de autoalojamiento o *self hosting* no son posibles, por lo ciertas compañías fabricantes de CMS ofrecen opciones de *hosting* a precios competitivos, ofertando a la vez soporte y mantenimiento de primera mano
- Los **costes de formación**, dedicados a instruir a los usuarios para que empleen la herramienta
- Los **costes asociados a modificaciones o adaptaciones**, sea para cambiar las interfaces o alterar/extender las funciones de la aplicación

Por su naturaleza, la mayoría de estos costes serán recurrentes (mantenimiento, *hosting*, licencias de uso temporales) mientras que otros serán de un único desembolso (licencia de uso permanente, formación, adquisición de maquinaria para *self-hosting*).

Para aquellos casos donde el CMS sea de uso gratuito, se opte por opciones de *hosting* ajenas a los servicios provistos por las empresas que ofrecen los CMS o se desee gestionar con mayor detalle, pueden existir costes ocultos propiciados por factores como[10]:

- **Las características del entorno** donde se desee implantar. La cantidad de usuarios y patrones de uso pueden implicar un desembolso mayor para disponer de servidores capaces de soportar la carga
- **Cambios futuros o mejoras** que puedan surgir por necesidades de los usuarios tras su uso continuado

De acuerdo al cálculo establecido por Anand Timothy en 2014[9], poniendo como ejemplo el uso de Moodle[12] (CMS gratuito) con *hosting* ajeno y un grupo de 3000 usuarios, el desembolso rondaría los $20000 en costes de un solo pago y cerca de $33000 a los anuales. En definitiva, la integración de un CMS en un sistema educativo o centro podría estar restringido por el coste que tendría su despliegue y uso.

Para particulares o en la educación privada, el principal limitante sería el desembolso que se llevaría a cabo. En el ámbito público, tales como podrían ser las escuelas e institutos, la decisión podría venir dictada por sus respectivas consejerías de educación o gobiernos, con el objetivo de reducir costes y homogeneizar su implementación.

El principal aliciente para este caso sería la implementación de un sistema extendido a una comunidad de estudiantes y docentes en lugar de ceñirse a centros concretos, pues el gasto de disponer de un sistema con poco alcance tendría un coste demasiado alto para el beneficio que aportaría, y si únicamente fuera usado en escuelas concretas podría provocar cierta discriminación.

2.3 Nuevas tecnologías

A diferencia de otros entornos donde la informática ha ejercido su influencia, el mundo del aprendizaje online no ha desarrollado nuevos medios. En su lugar, ha hecho acopio de tecnologías que no fueron diseñadas *ex profeso* para él, pero cuya aplicación supone un importante beneficio y encajan funcionalmente con sus necesidades. Tales son:

- **Plataformas multimedia** con las que compartir contenidos en formato vídeo, que se emplean mayoritariamente como registro de las clases y lecciones impartidas en el curso. Permite que los alumnos puedan visualizar sesiones anteriores, tutoriales, charlas u otro contenido relevante para la formación.
  Ej: VideoApuntes de PoliformaT (integración con Sakai, Universitat Politècnica de València)[13]

- **Repositorios de documentación**, que facilitan el acceso a apuntes, diapositivas u otros archivos relevantes para el curso y sus asignaturas.
  Ej: Módulos de contenido de Sakai[11] y Moodle[12]

- ***Streaming*** o retransmisiones en directo, que permiten impartir clases a distancia y tener comunicación directa con los alumnos que atienden a las sesiones
  Ej: Google Classroom[14]

- ***Big Data***, la aplicación de técnicas de recopilación de datos, estudio estadístico e inteligencia artificial para analizar el rendimiento académico de los alumnos, elaborar perfiles y patrones conductuales.
  Aplicado correctamente, permite encontrar puntos débiles en la docencia y los materiales, así como elaborar cursos adaptados o personalizados para cada estudiante.
  Ej: CENTURY[15]

También son dignas de mención aquellas funciones que ya existían previamente y que han visto sus capacidades mejoradas con las nuevas tecnologías:

- **Utilidades de ofimática**, que se han incorporado en los CMS mediante integraciones
  Ej: *Suite* Microsoft M365[16]

- **Evaluaciones online**, exámenes y pruebas que se llevan a cabo a través del CMS. Pueden incorporar mecanismos de corrección automática que informen a los alumnos de sus notas inmediatamente después de terminar los exámenes.
  Para correcciones sencillas, es posible emplear tests de preguntas múltiples. En situaciones donde la complejidad sea ligeramente mayor se pueden emplear técnicas de *OCR*. Los métodos más complejos llegan al extremo de grabar las cámaras y pantallas de los examinados con la intención de evitar copias.

  La irrupción del COVID-19 ha dejado en evidencia que queda mucho trabajo pendiente en este frente, pues han aflorado soluciones que pretenden garantizar que los estudiantes no copian en las pruebas a través de medidas que violan la privacidad fundamental y resortan a almacenar y contrastar la información biométrica de los examinados. Tal es el caso de Proctoru.[18][19][20]

- **Filtros anticopia**, programas que comparan las entregas de los distintos alumnos para buscar similitudes entre sus respuestas y determinar si han imitado o calcado parte de ellas. Es una de las aplicaciones más comunes en la educación superior, y

las empresas que proveen este tipo de utilidades han visto crecer su mercado con la aparición del *Big Data*. El ejemplo por excelencia en este campo es Turnitin[17], herramienta antiplagio que también comprobará la autenticidad y originalidad de este mismo documento antes de su entrega.

2.4 Crítica al estado del arte

Dado que el aprendizaje a través de la tecnología está en alza, en el mercado encontramos una gran variedad de CMS con diversas capacidades y destinados a un público concreto. Con el desarrollo de Dagaz se pretende aprender de todas ellas y buscar cuáles son los principales factores que les han dotado de fama y éxito, cuáles han sido los elementos que han introducido y facilitado las actividades de sus usuarios. Para ello haremos hincapié en varios elementos: **Aspectos del código** (lenguaje, licencia), **opciones de alojamiento**, **modularidad** y **coste** para un centro, asumiendo una carga de 500 alumnos por curso y 2 ciclos de desarrollo anuales llevados a cabo por un equipo de tamaño variable según la complejidad de la herramienta.

Con respecto a esta última parte, se considera en el total el coste del *hosting* anual junto con el coste de introducir los nuevos requisitos y mejoras en el sistema.

Dado que algunas de estas herramientas poseen extensiones, se procurará contemplar las funcionalidades *out of the box*, esto es, aquellas que vienen por defecto; aunque ciertas integraciones también podrán ser relevantes para la discusión.

2.4.1. Moodle

Lenguaje	Licencia	Tipo de uso	Opciones de alojamiento	Coste anual
PHP	GPLv3	Gratuito	*Self-hosting / Moddle partners*	7811€

Moodle se define como una plataforma de aprendizaje diseñada para que educadores, administradores y estudiantes tengan acceso a un sistema robusto, seguro e integrado para crear entornos de aprendizaje personalizados.

Su característica más relevante es su modularidad, pues permite habilitar o desactivar las funciones que trae por defecto con el fin de adaptarse a la casuística de cada escuela o clase. También cuenta con una nutrida lista de *plugins* y *addons*, extensiones diseñadas por la comunidad que permiten incrementar sus capacidades o interconectarlo con otros sistemas.

Moodle es una opción abierta, con experiencia y renombre dentro de la educación online. Cuenta con soporte de primera mano, documentación para su modificación y adaptación, opciones de gestión para docentes y administradores y una comunidad abierta que invita a la colaboración. Asimismo, también contiene mecanismos con los que gestionar alumnos y cursos.

El proyecto avanza paulatinamente gracias a un grupo desarrolladores internos a tiempo completo que producen grandes actualizaciones cada 6 meses. A su vez, tiene el apoyo de desarrolladores de todo el mundo que contribuyen al proyecto de forma desinteresada o construyen *plugins* para él. En definitiva, contar con actividad de desarrollo tanto de la propia empresa como de personal externo le aporta un valor añadido y permite que la herramienta evolucione.

En sus más de 10 años de actividad ha cosechado cerca de 200 millones de usuarios y se emplea en casi 160 mil entornos. De acuerdo al ratio estudiantes/número de habitantes, España es el país del mundo que más emplea Moodle.

Su arquitectura le permite adaptarse con facilidad a un entorno reducido, como podría ser una clase, o emplearse en entornos más complejos como centros universitarios. En España encontramos casos relevantes como la Universidad Autónoma de Madrid, la Universidad de Oviedo o la Universitat de València.

Entre sus competencias incluye el marcaje de notas, servicios de mensajería, almacenamiento de archivos, calendarios de actividades, definición para cursos y asignaturas, *plugins* anticopia, analíticas, personalización del sitio (esquemas de colores, interfaz)... Además, cuenta con *web services* para poder integrarse con otros sistemas con mayor soltura.

Aunque cuente con opciones para mejorar este aspecto, la interfaz de usuario de Moodle es poco intuitiva y restrictiva. Este factor podría llevar a que la formación de sus usuarios tome más tiempo y recursos, con lo que dominarla requiere esfuerzo y puede resultar incómodo a corto y largo plazo.

A su vez, Moodle no está pensado para interconectar grupos de usuarios. Si un entorno de Moodle fue concebido con la idea de dar servicio a una sola clase, puede resultar complicado o directamente impráctico extenderlo para abarcar un centro completo. De acuerdo a cómo se gestiona la herramienta, su implementación tiende a abarcar el mayor número de usuarios, y posteriormente los agrupa según cursos o grupos.

Integrar una comunidad autónoma, tal y como proponen algunos CMS, puede resultar difícil. Para estos casos, si se contara con una única instancia y se tendiera a la centralización, las cargas por actividad de sus usuarios serían muy altas y concentradas, además de que la personalización sería limitada y común a todos los centros/grupos/-cursos involucrados. Por otro lado, si se tendiera a la descentralización, como podría ser usar una instancia por centro, compartir información entre varios miembros de esta red sería una tarea compleja. Un uso inteligente de los web services podría permitir que varias instancias de Moodle intercambiaran información entre ellas con mayor facilidad.

Si se alojara una plataforma de Moodle a través de sus afiliados, el coste anual rondaría los 611.33€[21]. Considerando que el sueldo medio mensual de un desarrollador de PHP en España está cerca de los 2400€[22], y asumiendo que 3 desarrolladores se encargarán de actualizar y modificar el sistema, mantener un CMS con Moodle costaría cerca de 7811€.

2.4.2. Sakai

Lenguaje	Licencia	Tipo de uso	Opciones de alojamiento	Coste aproximado
Java	ECL v2.0[23]	Gratuito	*Self-hosting*/Afiliados	7666€

Sakai es un CMS arraigado en el ámbito universitario que aporta en utilidades centradas en la enseñanza y aprendizaje.

Sakai divide sus capacidades en 6 grandes grupos[25]:

- **Comunicación y colaboración**, que permiten que la comunidad de usuarios permanezca conectada. Incluye calendarios, chats, foros, *wikis* e incluso email interno.
- **Evaluación**, herramientas orientadas a trabajos, exámenes y marcaje de notas. Los alumnos pueden entregar sus actividades evaluables y recibir información de las

mismas. Los profesores tienen después la posibilidad de realizar la corrección y calificar directamente desde la herramienta. También dispone de áreas donde poder crear rúbricas e incluso exámenes que podrán ser contestados desde la página web de Sakai.

- **Desarrollo y distribución de contenidos**, opciones para el almacenaje y recuperación de archivos en la plataforma. A través de ello es posible crear contenido vinculado a una asignatura concreta, sean apuntes, multimedia o exámenes. Hace gala de integraciones con OneDrive de Microsoft y Google Drive.

- **Gestión de cursos**, que permiten asignar usuarios y contenidos por grupos, asignaturas u otras separaciones arbitrarias; acompañado de un amplio surtido de **herramientas de administración** de sitios, de membresía, control de permisos de usuario, estadísticas...

- **Herramientas construidas por la comunidad**[26]

- **Integración con aplicaciones externas** para incrementar sus capacidades. Se incluyen ejemplos relevantes como iRubric, Wordpress, Turnitin, Zoom...

El enfoque universitario de Sakai limita en gran medida su público objetivo, pero le permite refinar más sus fines. Las comunidades de estudiantes de educación superior requieren de funciones muy específicas y a niveles y estándares de calidad altos. El acceso a contenidos, la separación del alumnado (por grupos, cursos o asignaturas u otras divisiones), la complejidad de las herramientas de corrección, la escalabilidad de Sakai; todo está pensado para alojar a una comunidad completa y que su funcionamiento no se vea coartado o interrumpido.

La configuración de Sakai es compleja y, en la medida de lo posible, debe simplificar al máximo la gestión de un número considerablemente grande de usuarios. Para ello se dota de herramientas de control de recursos del servicio (carga del servidor y estadísticas, *cronjobs* u operaciones periódicas, reportes del estado de la infraestructura que lo forma).

Sakai es una herramienta compleja que también debe progresar con el tiempo, por ello cuenta con desarrollo de "herramientas específicas", *addons* publicados por desarrolladores ajenos[26], similares a los de Moodle pero en un número mucho menor.

No se concibe que Sakai sea empleada en ámbitos extensos como una comunidad autónoma, pero excede las expectativas cuando se utiliza en campus universitarios u otras escuelas superiores. Algunos ejemplos de su uso en España son la Universidad Complutense de Madrid, la Universidad Pública de Navarra, la Universitat de Lleida y la Universitat Politècnica de València (PoliformaT)[27].

En definitiva, Sakai apunta específicamente a un sector de la población educativa y les otorga específicamente aquello que necesitan.

Sakai también dispone de afiliados para alojar la herramienta, pero el contrato de *hosting* varía según la casuística de cada oferta, así que no se puede establecer un precio fijo. Por similitud, utilizaremos como base el desembolso que se haría para un servidor con Moodle, que ronda los 611€ anuales, y se multiplicará por un ratio de 1.5, pues Sakai incluye funciones que requieren más capacidad de computación y por tanto, el servidor tendrá que disponer de equipamiento lo suficientemente potente. Esto hace un total de 916€ anuales.

Considerando que el sueldo medio mensual de un desarrollador de Java en España está cerca de los 2250€[24], y asumiendo que 3 desarrolladores se encargarán de actualizar y modificar el sistema, mantener un CMS con Sakai costaría cerca de 7666€.

2.4.3. ITACA

Lenguaje	Licencia	Tipo de uso	Opciones de alojamiento	Coste aproximado
Java	Propietaria	Sector público	Privado (GVA)	Desconocido

ITACA es el CMS desarrollado desde 2013 hasta la actualidad por la Conselleria D'Educació i Esport de la Comunitat Valenciana. Los detalles de su implementación e infraestructura no son conocidos, pero puede derivarse de sus páginas que emplea **Java Server Faces** (JSF) y, por tanto, un *frontend* y un *backend* posiblemente construidos con Java.

La llegada de ITACA en 2013 supuso la decomisión de sus dos precursores, GESCEN y ALLEGRO[5]. En sus inicios, la Generalitat participó con centros pioneros para llevar a cabo pruebas con las que determinar cómo integrar este sistema en la docencia y establecer su alcance. Sentó precedente en una época donde la integración de sistemas tecnológicos en la educación era novedoso y revolucionario.

De acuerdo a los datos facilitados en su página web, ITACA se asentó en la comunidad hasta interconectar más de 2.000 centros educativos, 60.000 profesores y a más de 700.000 alumnos[3]. Además, los colegios públicos pueden emplearlo sin coste alguno, lo que lo convierte en un gran incentivo.

Se trata de un sistema compuesto de dos módulos: **Docentes** y **Familia**.

- **Docentes** está pensado para que el profesorado pueda acceder e introducir notas, asistencias, revisar el listado de los alumnos a los que imparte, construir rúbricas y tener acceso a otra información relevante como calendarios de sesiones, actividades, partes, etc.
- Por otra parte, **Familia** se orienta a los padres o tutores legales de los alumnos y aporta detalles de su desempeño en el curso, tales como notas, asistencias a clase, faltas u actividades extraescolares. Incluye un chat desde el cual es posible ponerse en contacto con los profesores. De acuerdo a su web[28]:

  > ITACA dispone de información con la que *conectar* con las familias, ayudándoles a sentirse mejor informados sobre la evolución educativa de sus hijos y permitiéndoles establecer vías de comunicación alternativas con el centro.

Moodle y Sakai abogan por representar tanto cursos como alumnos en la herramienta y, posteriormente, comparten contenido con ellos (sean documentos, evaluaciones o demás) de acuerdo a los grupos o asignaturas a los que pertenencen.

Sin embargo, ITACA se distancia de ello y aporta un nuevo enfoque, pues nació con la finalidad de interconectar los distintos miembros que forman parte del tejido educativo de la comunidad[4], sean alumnos o profesores, y centralizar toda su información en un único lugar. De esta manera, los centros pueden transmitir evaluaciones, matriculaciones y otros procesos burocráticos directamente a un servidor central. Así, el expediente educativo de un alumno podía ser compartido, transmitido y actualizado entre centros, pudiendo simplificar los procesos de gestión de matriculaciones, registros de asistencias, PGAs y otros menesteres.

El código de ITACA no es público y los detalles de su infraestructura tampoco lo son. Dada la sensibilidad de los datos que almacena, ocultar esta información con fines de seguridad puede considerarse un motivo válido. Sin embargo, es esta naturaleza cerrada lo que hace complicado establecer cuál sería su coste, su alcance y la tecnología sobre la

que se cimienta. Dado que su acceso está limitado exclusivamente a los miembros de la comunidad educativa (profesores y padres/tutores legales con hijos matriculados en el curso actual), es también complicado hacer una estimación de cuántas funcionalidades dispone.

Un elemento particular de ITACA es el acceso a la plataforma a través de **Cl@ve**[29], un mecanismo de verificación de indentidad electrónica para la administración empleado por el Gobierno de España.

Teniendo en consideración que lleva en funcionamiento desde 2013 y analizando el contenido de sus páginas de login, las únicas accesibles sin registro, se puede asumir que ITACA emplea tecnología que podría considerarse ya obsoleta, como es *Java Server Faces* (JSF). Como todo proyecto de gran envergadura y cierta antigüedad, puede asumirse también que existirá deuda técnica. Asimismo, dada la ausencia de documentación al respecto, puede intuirse que ITACA no dispone de soporte para adición de módulos.

Encuesta a usuarios de ITACA

ITACA ha sido una importante fuente de inspiración para la elaboración de de este trabajo de investigación y desarrollo. Dada la ausencia de información por medios públicos, se han recabado datos de usabiliad y funcionamiento a través de entrevistas y una encuesta realizadas a profesores de la escuela pública que interactúan con ITACA a diario.

De acuerdo a sus usuarios, ITACA tiene ciertas deficiencias:

- **Las evaluaciones**. ITACA sólo habilita esta función durante unos días cada trimestre, en lugar de permitirlo a lo largo del curso. Cuando se abre el plazo, los servidores se ven sometidos a una extensa carga, pues gran parte de los profesores aprovechan este momento para evaluar a sus alumnos. Como resultado, la aplicación se vuelve inestable, se producen fallos en el acceso y la información introducida no se guarda correctamente.
- **La tolerancia a fallos**. En cursos como Infantil, las evaluaciones se rigen por rúbricas o competencias. Éstas se introducen a través de un campo de texto plano con formatos específicos y separaciones concretas entre cada competencia y su nota, y el sistema valida el *input*. Por desgracia, el *input* es muy específico y puede dar problemas si el formato no es el correcto o se usan acentos, desechando un trabajo muy complejo. Además, no guarda automáticamente la marcación dada.
- **La interfaz**, que no suele considerarse agradable o intuitiva.
- **La evaluación no permite agregar documentos**, tales como podrían ser los exámenes o las actividades corregidas.

Por otro lado, valoran ciertos aspectos como son:

- El área de comunicación profesor-padres/tutores legales, que facilita mucho el intercambio de información entre ambos.
- El apartado de notificaciones para el grupo.
- La facilidad para introducir asistencias.

Entraremos en más detalle en los datos recopilados más adelante.

2.5 Propuesta

En esta exposición se han podido ver dos diferencias relevantes entre los ejemplos descritos: Moodle y Sakai son **CMS orientados a contenidos**, con soporte para almacenamiento de archivos para cursos; y ITACA es un **CMS enfocado a la gestión**, más dirigidos al ámbito burocrático y con herramientas para facilitar el día a día en el registro de sucesos y notas de un curso académico. Moodle y Sakai de ellos fueron creados por empresas o colectivos de desarrolladores y han hecho su código **libre y público**, mientras que ITACA es producto del sector público y su código es **privado**. Moodle se orienta a grupos de usuarios variables, desde clases hasta universidades; Sakai se centra en el público universitario e ITACA abarca toda la Comunitat Valenciana. Esta información se recoge en la siguiente tabla:

CMS	Licencia	Orientado a	Extensible	Cantidad de usuarios
Moodle	GPLv3[36]	Contenidos	Plugins/Integraciones	Variable, adadptable
Sakai	ECL v2.0[23]	Contenidos	*Apps*/Integraciones	Alta, sector universitario
ITACA	Privada	Gestión	No	+760.000 usuarios

Tras su análisis, se han considerado como puntos positivos:

- El uso de código y licencias libres
- La modularidad de Moodle y Sakai
- El apoyo al profesorado que ofrece ITACA
- La diversa cobertura de las aplicaciones

Si se pretende producir un nuevo CMS capaz de aportar novedades y abrirse camino en un sector saturado con posibiliaddes, hará falta razonar adecuadamente cómo cohesionar estos factores y producir una herramienta capaz y útil.

2.5.1. Dagaz

Fuertemente influenciado por ITACA, Dagaz se contempla como un CMS enfocado en la gestión del curso y construido por medio de tecnologías libres. Su publicación bajo la licencia Affero GNU Public License v3[34] asegura que su código siempre será público, favorece que puedan surgir *forks* o variaciones de la misma y permite que otras aplicaciones con licencias compatibles puedan emplear su código para construir nuevas soluciones a problemas en el ámbito educativo.

Este ha sido un movimiento calculado: Haciendo público su código se evita que terceras partes puedan vender la aplicación para su propio beneficio, y contribuye a la creación de un clima de cooperación que incluya a desarrolladores de todo el mundo dispuestos a llevar adelante la herramienta y aportar su grano de arena en este proyecto. Se escoge la AGPLv3 frente a la GPLv3[36] porque así se puede exigir a sus implementadores la liberación del código que se ejecuta en los servidores a los que accederán sus usuarios.

Un objetivo claro es que Dagaz sea un proyecto comunitario y accesible por cualquier persona, independientemente de su país de origen, situación económica o social. Su distribución es gratuita, lo que propicia su uso en la educación pública o privada, y pretende mitigar o minimizar el impacto que las restricciones económicas del bolsillo de cada centro o comunidad podrían tener en la aplicación de nuevas tecnologías en la educación.

Dagaz busca ofrecer soluciones adaptables y escalables independientemente del tamaño de usuarios a los que se dé soporte, y eso implica tener en cuenta conceptos como

la potencia de los dispositivos donde se alojará, el acceso a internet en el territorio donde se vaya a desplegar y también el capital del que se disponga para llevar a cabo este proyecto. Para casos donde exista más inversión se consideran 4 posibles escenarios:

- **Alojamiento centralizado común**, orientado a un gran número de usuarios. En este caso se contempla una sola fuente de datos y un único servidor al que accede toda la comunidad educativa. Todos los centros interactúan con ese servidor, y la personalización se lleva a cabo a nivel comunitario.
- **Autoalojamiento**, pensado para grupos pequeños e intermedios de usuarios. Este modelo supone el uso de un único servidor que se emplearía directamente en el centro donde se use. Dependiendo de cuánta gente haría uso de él, sería posible integrar el servidor en computadores de baja y media potencia. La personalización se llevaría a cabo en cada servidor, pudiéndose adaptar a la casuística concreta de cada grupo.
- **Autoalojamiento con federación**, explicado en mayor detalle en el apartado de Autoalojamiento con federación.
- **Autoalojamiento con federación y servidor central**, similar al caso anterior pero incluyendo un servidor central que recibe datos de todas las instancias y almacena su información con diversos fines (redundancia, análisis, registros, etc.).

Por otro lado, en países con menor nivel de desarrollo o en los casos donde no sea posible invertir tanto dinero, se proponen los siguientes casos:

- **Autoalojamiento** en dispositivos discretos, incluyendo un set menor de funciones para ahorrar potencia. Esto sería ideal para casos donde las escuelas estén alejadas, solo se dispusiera de conexión local o el acceso a la electricidad fuera escaso. Con este fin se planea que Dagaz fuera capaz de ejecutarse en dispositivos de poca capacidad de almacenamiento y procesado, incluso suponiendo casos donde el sistema se encontrara alimentado por paneles solares u otras fuentes eléctricas intermitentes.
- **Alojamiento centralizado común** en aquellos casos donde sea posible. Con esta medida se reduciría el esfuerzo económico que los centros educativos llevarían a cabo para utilizar el sistema, otorgando esa carga a las instituciones públicas.

Tanto si se trata de una única clase en una academia como si se abarcan grandes núcleos poblacionales bajo un mismo sistema, Dagaz pretende adaptarse al mayor número de casuísticas posibles haciendo maleable su código y permitiendo configurar la jerarquía del personal educativo y su alumnado.

Una ventaja primordial en los proyectos de código libre es que cualquier persona es libre de contribuir y utilizar lo presente como punto de partida para una solución alternativa. Tanto si se tratara de extender las funciones ya existentes como de crear nuevas capacidades, Dagaz pretende apelar a ese público para lograr que la herramienta crezca, se vuelva más sólida y alcance a más gente.

Aunque lo idóneo sería incorporar todo el desarrollo directamente en la herramienta, es innegable que seccionarla en pequeñas adiciones permite una mayor granularidad y capacidad de configuración. Para ello, Dagaz se estructuraría en un módulo principal al cual podrían incorporársele *addons*, extensiones que o bien han sido diseñadas por el equipo original de desarrolladores o por terceros.

En cuanto a funciones, se pretendería incorporar parte de las capacidades para representar y modelar los grupos, asignaturas y unidades temporales (trimestres, cuatrimestres, semestres...) de cada curso, tal como hacen los ejemplos expuestos anteriormente. También, a grandes rasgos, se desearía incluir funciones de almacenamiento de datos

asociados a cada uno de ellos, de tal forma que se pudiera crear un repositorio al que los alumnos y profesores pudieran subir o descargar contenido de la plataforma.

No cabe olvidar que Dagaz no deja de ser un concepto, tal vez demasiado ambicioso, y que **abarcarlo en su totalidad en pocos meses y por una única persona es físicamente imposible**. Como tal, el alcance previsto para la elaboración de este documento estará acotado considerablemente. Sin embargo, **quede así constancia de la idea que subyace al construir el proyecto**.

Autoalojamiento con federación

La federación y el autoalojamiento son técnicas tradicionalmente empleadas en redes sociales descentralizadas[30]. Mastodon, Diaspora, Fiendica y otros tantos ejemplos conforman el conocido como Fediverso[31][32] (*Fediverse* en inglés), una conjunción de aplicaciones e instancias de las mismas que utilizan una serie de protocolos comunes para intercambiar información entre ellas. Estableciendo un protocolo común, los datos de la red pueden ser compartidos con cualquiera de sus miembros.

Las posibilidades dejan lugar a la imaginación: Un individuo puede crear su propia plataforma construida sobre un protocolo y establecer contacto con el resto de los miembros de esta red, con lo que estaría creando una aplicación o servicio. Por otra parte, otro individuo puede coger una de las plataformas ya existentes y ejecutarla en su propio servidor, lo que se conoce como instancia.

El concepto parte de tener sistemas independientes, modificables y que pueden perseguir finalidades diferentes, pero con unos mensajes y métodos de comunicación comunes. Es por ello que su aplicación en el campo de los CMS sería revolucionario, pues permitiría gozar de la personalización y los beneficios del autoalojamiento en una instancia con las ventajas de contar con conexión con el resto del sistema educativo, permitiendo crear una red con la que intercambiar no solo contenido educativo, sino también información de matriculaciones, expedientes, GPAs y demás.

Crear un protocolo similar a los utilizados en el Fediverso es extremadamente complejo y, en su estado de desarrollo actual, Dagaz no cuenta con estas funciones. Sin embargo, la propuesta queda relatada para que en un futuro sea posible construir sobre esta idea.

Listado de características

Para llevar a cabo esta decisión y enfocar mejor cuáles serán las características a incorporar, así como su relevancia, se utilizará el método **MoSCoW**. Para ello se incluirán todas las que han sido detectadas en las herramientas investigadas. A través de esta técnica se priorizan las propiedades clasificándolas en 4 tipos:

- ***M**ust have*: Imprescindible para el proyecto o sus *stakeholders*
- ***S**hould have*: Debería incorporarse, es relevante y su inclusión sería muy positiva
- ***C**ould have*: Su incorporación sería beneficiosa pero no tendría tanta relevancia
- ***W**on't have*: No se incorporará, no encaja con los objetivos o carece de importancia

Se utilizará el siguiente esquema de colores para la tabla:

Presente	Ausente	Incompleto	Sin datos	Must	Should	Could	Won't

Características (CAR)	Herramienta			
	Moodle	Sakai	ITACA	Dagaz
CAR1: Información del curso				
1.1: Listado de asistencias	Sí	Sí	Sí	Must
1.2: Listado de notas	Sí	Sí	Sí	Must
1.3: Listado de alumnos	Sí	Sí	Sí	Should
1.4: Listado de cursos	Sí	Sí	Sí	Must
1.5: Listado de grupos	Sí	Sí	Sí	Must
1.6: Listado de sesiones	Sí	Sí	Sí	Must
1.7: Expediente completo del alumno	Solo cursos	Solo cursos	Completo	Could
1.8: Calendario para profesores	Sí	Sí	Sí	Must
1.9: Calendario para alumnos	Sí	Sí	No	Should
1.10: Acceso para profesores	Sí	Sí	Sí	Must
1.11: Acceso para alumnos	Sí	Sí	Limitado	Should
1.12: Acceso para administradores	Sí	Sí	Sí	Must
CAR2: Gestión del curso				
2.1: Matriculación de alumnos	Sí	Sí	Sí	Could
2.2: Marcaje de asistencias	Sí	Sí	Sí	Must
2.3: Creación de cursos	Sí	Sí	Sí	Must
2.4: Creación de grupos	Sí	Sí	Sí	Must
2.5: Creación de asignaturas	Sí	Sí	Sí	Should
2.6: Creación de sesiones	Sí	Sí	No	Must
2.7: Creación de pruebas	Sí	Sí	No	Must
2.8: Creación de rúbricas	No	Sí	No	Won't
CAR3: Modificabilidad e integrabilidad				
3.1: Jerarquía modificable	Limitada	Limitada	No	Should
3.2: Soporte para plugins / extensiones	Sí	Sí	No	Could
3.3: Soporte para personalización	Sí	Sí	No	Must
3.4: API Rest	Sí	Sí	Sin datos	Could
3.4: Webhooks	Sí	Sí	Sin datos	Could
3.5: Federable	No	No	No	Should
CAR4: Evaluaciones				
4.1: Marcaje de notas	Sí	Sí	Trimestral	Must
4.2: Exámenes tipo test	Sí	Sí	No	Won't
4.3: Exámenes en la plataforma	Sí	Sí	No	Won't
4.4: Integración herramientas corrección	Sí	Sí	No	Won't
CAR5: Almacenamiento de archivos				
5.1: Documentos asociados a asignaturas	Sí	Sí	No	Could
5.2: Documentos de corrección de pruebas	Sí	Sí	No	Must
5.3: Subida de archivos (Profesores)	Sí	Sí	No	Could
5.4: Subida de archivos (Estudiantes)	Sí	Sí	No	Could
5.5: Carpeta personal de archivos	Sí	Sí	No	Won't
5.6: Permisos de subida/descarga	Sí	Sí	No	Could
CAR6: Comunicación				
6.1: Foros	Sí	Sí	No	Won't
6.2: Chats	Sí	Sí	No	Won't
6.3: Email interno	Sí	Sí	No	Won't
6.4: Comunicación profesor - padres	No	No	Sí	Could
6.5: Sistemas de avisos	Sí	Sí	Sí	Could
6.6: Internacionalización	Sí	Sí	Sí	Must
CAR7: Alojamiento y capacidad				
7.1: Autoalojable	Sí	Sí	No	Must
7.2: Tamaño de usuarios escalable	S/M/L/XL	L/XL	XXL	Should

CAPÍTULO 3
Análisis del problema

3.1 Identificación y análisis de las soluciones

3.1.1. Información, gestión y evaluaciones del curso

Los CMS de gestión se centran en incluir datos sobre los cursos y quienes participan en él. **Modelar el esquema educativo** (es decir, crear cursos, grupos de alumnos y asignaturas, matricular a los distintos alumnos, crear sesiones y pruebas...) y **acceder a su información** (expedientes, calendarios de eventos, listados de alumnos, notas, cursos...) serán los dos grandes pilares a los que habrá que atender.

Ambos aspectos tienen un denominador común: Existe una gran cantidad de datos que mostrar al usuario y también muchas posibilidades para actuar con ellos. Así pues, podríamos contemplar algunos casos muy comunes:

- Parte de la información de la herramienta puede no ser relevante para un tipo específico de usuario
- Ciertos campos no deberían mostrarse para ciertos grupos de usuarios
- Si un usuario cuenta con demasiadas opciones o campos en una misma vista, podría verse saturado y no saber continuar con su trabajo
- De igual manera, un usuario saturado podría realizar una actividad por error y provocar daños en sus datos
- Podría darse el caso de que algunos usuarios contemplen hacer operaciones malintencionadas para afectar a la información guardada en el sistema

Como posibles soluciones se han propuesto:

- Mostrar únicamente la información relevante para el usuario, agilizando las tareas y haciendo que operar con la herramienta sea más intuitivo.
- Esconder las opciones y complicaciones del sistema con el que se trabaja, pues ello contribuirá a una menor posibilidad de error.
- Restringir el acceso a ciertos recursos para algunos usuarios.
- Impedir la modificación de elementos cruciales para usuarios que no cuenten con las características necesarias para modificarlos.

Alcanzar ese preciado equilibrio entre funcionalidad y simplicidad dará lugar a que la interacción entre usuario y herramienta sea más directa y sencilla.

Se considera también que algunos miembros de la plataforma pueden tener problemas en su uso, sea porque no están acostumbrados a las tecnologías de la información o

porque lo puedan encontrar dificultoso. Con ese fin se procurará que las interfaces sean autoexplicativas y, en su defecto, que incorporen textos que definan brevemente pero con claridad qué operaciones se pueden llevar a cabo y cómo hacerlo correctamente. Además se procurará la aplicación del principio "***un click, una acción***": Las acciones básicas sólo requerirán un click.

La mayor parte de los casos de uso y requisitos funcionales están enfocados en esta misma dirección.

3.1.2. Modificabilidad

Una parte crucial del éxito de los CMS de código libre es su capacidad para poder extenderlos con el menor esfuerzo. Sea mediante la edición del código, la incorporación de nuevos estilos, la integración con otros servicios...; el otorgar nuevas funciones a la plataforma y permitir su configuración será vital para que pueda permanecer relevante y responda a las últimas tendencias educativas.

Modificar la herramienta también puede volverla más inestable o provocar errores que antes no se contemplaban, lo cual es un riesgo que debe minimizarse al máximo. Para ello, la herramienta debe ser tolerante frente a los cambios. La mejor manera de lograrlo es a través de:

- Un estilo de programación defensivo que limite las situaciones imprevistas y devuelva valores seguros que no interrumpan la ejecución.
- Una extensa batería de pruebas con un buen nivel de cobertura que contemple tanto los casos positivos como los extremos que pudieran causar errores.
- El control de las situaciones de errores, de tal forma que cuando sucedan se pueda reconducir al usuario a un punto seguro desde donde poder retomar sus operaciones.

La batería de pruebas sería un valor añadido para el proyecto, pues incitaría a que otros desarrolladores participaran en la herramienta teniendo como garantía que el código que cambien o introduzcan será correcto siempre que pase los tests de los que se dispone. Lograr esto será posible aplicando *frameworks* de *testing* y llevando a cabo pruebas unitarias.

Casi la totalidad de los requisitos no funcionales estarían incorporados en esta sección.

3.1.3. Almacenamiento de archivos

Algunas plataformas, como Moodle o Sakai, tienen un mayor enfoque en el acceso a contenidos, a la subida y bajada de archivos y a su asociación con unidades organizacionales (cursos, asignaturas, etc.). Estos contenidos pueden emplearse durante el tiempo lectivo y también constituyen una fuente de información para los alumnos, pues el uso más común de este almacenamiento son los apuntes, diapositivas y actividades del curso.

Este ha sido el medio de transmisión de contenidos preferido por el profesorado durante los meses de confinamiento por el COVID-19, lo cual lo ha dotado de más relevancia si cabe. Además, todo CMS que se precie incorpora estas funciones, a pesar de que suponen un elevado coste.

Disponer de un servidor con la suficiente capacidad de almacenaje puede ser un limitante para algunos implementadores de Dagaz, especialmente si el número de alumnos

es alto. Además, su complejidad puede incrementar considerablemente si se tomara la decisión de incluir un sistema de federación.

Por estos motivos, dotar a Dagaz de un mecanismo de almacenamiento de archivos está contemplado en la solución, pero pospuesto a un momento posterior.

3.1.4. Alojamiento y capacidad

Como se ha mencionado previamente, es menester que Dagaz esté al alcance de todos sus usuarios, de tal forma que cuenten con la posibilidad de ejecutar la herramienta en su propio servidor y dar servicio a alumnos y profesores. Respondiendo a esta necesidad, será necesario diseñar la herramienta para poder ejecutarse en esta circunstancia.

Otro detalle es la capacidad de usuarios que el sistema puede manejar. Asumiendo que se tendrá una cierta carga de personas trabajando simultáneamente en la herramienta, será menester que las operaciones que lleve a cabo un miembro no afecten a la actividad de los demás. También cabe contemplar cómo se modelarían los cursos en situaciones de pocos y muchos alumnos, y determinar la mejor manera de que ese modelado se pueda hacer de la forma más cómoda posible para todos los involucrados.

3.1.5. Lenguaje y tecnología

Dagaz se planea como una herramienta ejecutable en cualquier entorno, sin importar si se trata de un servidor de alta potencia, un ordenador casero o incluso dispositivos de baja potencia para casos con menor número de alumnos. Si bien la mayoría de lenguajes actuales pueden correr en cualquier procesador, habrá que buscar uno que reduzca las posibles diferencias que habría entre ellos y nos asegure un comportamiento similar para todos estos casos.

Nos decantamos por Java al tratarse de un lenguaje multiplataforma con veteranía, capaz de correr en la mayoría de plataformas y con una plétora de *frameworks* para desarrollar desde aplicaciones móviles (como es el caso de Android) hasta el *backend* y parte del *frontend* de algunos servidores (Spring Framework para el *backend*, Java Server Faces o Java Server Pages para el *frontend*).

Java también se emplea en programas de escritorio (Swing, JavaFX), así que podría ser una opción a considerar en el futuro si se deseara tomar el camino de desarrollar una aplicación. Además, Java cuenta con el *Java Database Connectivity* o JDBC, una API de conexión a bases de datos con un amplio soporte a nivel comercial. Finalmente, Java también dispone de un amplio catálogo de *frameworks* de *testing* que podemos emplear para garantizar el funcionamiento de la herramienta.

Otro detalle a tener en consideración es cómo deberán interactuar los usuarios con el sistema: Para su uso en ordenadores es preferible utilizar una web, pues es más próximo a la usabilidad del resto de aplicaciones de escritorio, mientras que se prefiere el uso de aplicaciones para dispositivos portátiles como *smartphones* o *tablets*. Independientemente de ello, de decantarnos por una aplicación, seguiría siendo necesario crear *endpoints* o direcciones a las que conectarse para comunicarse con el servidor y recuperar o enviar información.

Por experiencia en este frente, se optó por desarrollar una página web. Además, Java es un lenguaje tradicionalmente ligado al *backend* de los servidores, especialmente en el ámbito del *eCommerce*, y cuenta con tecnología que permitiría una construcción rápida y escalable. Como incentivo, parte de la información que se transmite entre el servidor y el navegador podría también ser comunicada a una aplicación mediante APIs, de tal

modo que no se descarta disponer de una web y una *app* simultáneamente en un futuro próximo.

De todas las alternativas disponibles, Spring Framework fue el elegido para edificar el *backend* del servidor. Spring permite crear *webapps* con interfaces que siguen el patrón **modelo-vista-controlador**, emplea JSP para construir el código HTML que se mostrará a los usuarios y, a través de Spring Boot, también abstrae de ciertas complicaciones de la configuración que serían típicas de un combo servidor+base de datos.

Así pues, Spring gestiona automáticamente el servidor sobre el que se ejecuta (Tomcat) y otros detalles de las propiedades de la base de datos y la transformación de la información que almacena. Por último, Spring cuenta con mecanismos de inyección de dependencias para que cualquier persona pueda incorporar o extender el comportamiento por defecto con mínimo esfuerzo, aprovechando el código que se proporciona *out of the box* y configurando las dependencias a emplear en cada una de las clases mediante un archivo de configuración XML.

Como nota adicional, el equipo que construirá el proyecto cuenta con 2 años de experiencia profesional desarrollando soluciones en *Hybris/SAP Commerce*, un *framework* de desarrollo de páginas para *eCommerce* construido directamente sobre Spring Framework, lo que agilizará el proceso de desarrollo y procurará la aplicación de buenas prácticas.

3.1.6. Encuestas y entrevistas con usuarios de ITACA

Con la intención de conocer mejor el colectivo de usuarios que utilizará la herramienta y para construir una solución que se adhiera mejor a sus necesidades, se realizó una encuesta a 42 profesores de la escuela valenciana. Este acto tuvo lugar el mes de Marzo de 2020 tras el inicio del estado de alarma por el COVID-19 en España.

El público al que nos dirigíamos con estas preguntas corresponde a los usuarios actuales de ITACA, el CMS de gestión de la Conselleria d'Educació i Esport de la Comunitat Valenciana. A tavés de una serie de cuestiones fue posible identificar los diversos grupos de usuarios, determinar qué sector poblacional usa esta herramienta y también su familiaridad con las nuevas tecnologías. Puede consultar los resultados aquí.[35]

De los datos recabados se desprenden varias características:

- **Todos los encuestados pertenecen a la escuela pública**, lo cual puede implicar que ITACA no es una opción lo suficientemente relevante en la escuela privada o que directamente no es utilizada en este tipo de centros. Además, puede que algunas de sus opiniones no incluyan a la totalidad de la comunidad educativa o no contemplen casos o situaciones que sí serían presentes en otras modalidades de enseñanza.
- **El 97 %** de los encuestados **han presentado problemas accediendo a la herramienta**.
- **Más del 95 %** de los usuarios utiliza ITACA **para insertar notas**. Más de $\frac{2}{3}$ también lo emplean para registrar asistencias.
- **Más del 90 %** de los encuestados **son profesores de infantil o primaria**. De nuevo, al no incluir a profesores de otros grados como secundaria, bachillerato o universidad, podríamos estar perdiendo información relevante o enfoques distintos.
- Los usuarios muestran una **cierta familiaridad con las nuevas tecnologías**, aunque la velocidad con la que llevan a cabo sus tareas en ITACA y su comodidad empleando la herramienta es baja.
- Se valoran positivamente las funciones de inserción de notas, la introducción de asistencias y la comunicación entre profesores y padres.

- Se identifican dos grupos de usuarios bien marcados: Los **profesores**, que son la mayoría de los encuestados, y un pequeño sector compuesto de **personal de dirección o administrativo** encargado de gestionar las matriculaciones, los cursos de la escuela y demás.

3.2 Solución propuesta

Dada la exposición indicada arriba, se determinó la construcción de Dagaz, un CMS en forma de *webapp*. Se escribirá en Java y hará uso de Spring Framework para controlar el *backend*, el servidor que lo ejecuta, la base de datos que se empleará y también para servir páginas HTML al *frontend*.

Dagaz es un proyecto ambicioso que no puede abarcarse en los 2 meses que atañen a la producción de este trabajo de investigación. Si bien su desarrollo podrá continuar en el futuro, el resultado que se entregará con este análisis es un **mínimo producto viable** que representa una priorización de los elementos que se desea incorporar en la herramienta en un periodo específico de su desarrollo. En esta decisión pueden influir detalles como las limitaciones técnicas, la dependencia entre tareas, las valoraciones recibidas u otros factores.

Es de vital importancia resaltar que ciertos elementos, considerados como imprescindibles para el resultado final, pueden no encontrarse desarrollados en esta entrega. Sin embargo, una vez completado el proyecto, se espera que se incorporen todos aquellos que se han considerado relevantes.

3.2.1. Listado de usuarios

De acuerdo a la investigación llevada a cabo previamente y al análisis de los posibles casos de uso, se identifican 3 tipos distintos de usuarios:

- **Profesores**, que se encargan de tomar la asistencia de los alumnos que acuden a sus clases. Corresponden a un colectivo cuya interacción con la herramienta debe ser rápida y directa, pues todo tiempo extra que dediquen a trabajar con ella es tiempo que pierden de su docencia.

  Poseen cierto conocimiento sobre el funcionamiento de las nuevas tecnologías y tienen cierta familiaridad con herramientas similares, pero en ocasiones pueden encontrar dificultosas las tareas. Por ese motivo habrá que incluir textos que puedan guiarles durante su proceso de adaptación.

- **Administradores**, personal de secretaría y administración de los centros. Son los encargados de la matriculación, la creación de asignaturas, cursos, trimestres y otros menesteres.

  Dado que tratan más a menudo con nuevas tecnologías y con servicios burocráticos asociados a la matriculación y los expedientes de los alumnos, se asume que tienen más destreza técnica. Por este motivo se considera que la dificultad de sus interfaces puede ser ligeramente mayor a fin de que sean más útiles y completas para su labor.

- **Alumnos**, miembros a quienes se les imparte clase y a quienes se evalúa. Tienen familiaridad con las nuevas tecnologías y están acostumbrados a la navegación online. A excepción de permitirles su login, no se les considera para este mínimo producto viable.

3.2.2. Especificación de requisitos funcionales

En los siguientes apartados se tendrán en consideración aquellas características que se hayan marcado como **Must** y parte de las indicadas como **Should**.

RF_1 - Acceso a cuenta

Engloba el proceso de login en la página con distintos tipos de usuarios. También incorpora el proceso de logout.

Requisito (RF)	Características asociadas (CAR)
RF_1 - Acceso a cuenta	$CAR_{1,10}$ - Acceso para profesores $CAR_{1,11}$ - Acceso para estudiantes $CAR_{1,12}$ - Acceso para administradores

RF_2 - Visualización de información del curso

Contiene los procesos por los cuales es posible mostrar al usuario información relativa al curso, sus asignaturas, alumnos y notas. Solo se consideran en este apartado las operaciones de lectura/descarga de estos datos.

Requisito (RF)	Características asociadas (CAR)
RF_2 - Visualización de información del curso	$CAR_{1,1}$ - Listado de asistencias $CAR_{1,2}$ - Listado de notas $CAR_{1,3}$ - Listado de alumnos $CAR_{1,4}$ - Listado de cursos $CAR_{1,5}$ - Listado de grupos $CAR_{1,6}$ - Listado de sesiones $CAR_{1,8}$ - Calendario para profesores $CAR_{5,2}$ - Documentos de corrección de pruebas

RF_3 - Creación de elementos del curso

Incluye las operaciones según las cuales se pueden crear nuevos elementos para el curso o modificar los ya existentes. Puede tratarse de asignaturas, pruebas, evaluaciones... Solo se consideran en este apartado las operaciones de creación y modificación de estos datos.

Requisito (RF)	Características asociadas (CAR)
RF_3 - Creación de elementos del curso	$CAR_{2,2}$ - Marcaje de asistencias $CAR_{2,3}$ - Creación de cursos $CAR_{2,4}$ - Creación de grupos $CAR_{2,5}$ - Creación de asignaturas $CAR_{2,6}$ - Creación de sesiones $CAR_{2,7}$ - Creación de pruebas $CAR_{4,1}$ - Marcaje de notas $CAR_{5,2}$ - Documentos de corrección de pruebas

RF_4 - Cambio de idiomas

Funciones que permiten cambiar el idioma de la página durante la navegación.

Requisito (RF)	Características asociadas (CAR)
RF_4 - Cambio de idiomas	$CAR_{6,6}$ - Internacionalización

3.2.3. Casos de uso

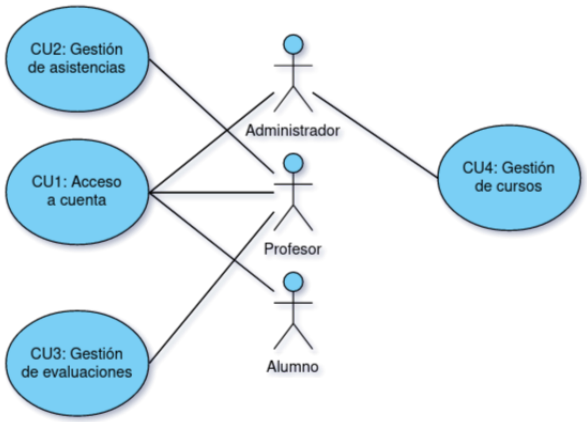

Figura 3.1: Casos de uso generales

Casos de uso (CU)	Requisitos funcionales (RF)	Características (CAR)
CU_1	RF_1 RF_4	$CAR_{1,10}$, $CAR_{1,11}$, $CAR_{1,12}$ $CAR_{6,6}$
CU_2	RF_2 RF_3	$CAR_{1,1}$, $CAR_{1,3}$, $CAR_{1,6}$, $CAR_{1,8}$ $CAR_{2,2}$
CU_3	RF_2 RF_3	$CAR_{1,2}$, $CAR_{1,3}$, $CAR_{1,6}$, $CAR_{1,8}$ $CAR_{2,5}$, $CAR4_{4,1}$, $CAR_{5,2}$
CU_4	RF_2 RF_3	$CAR_{1,6}$, $CAR_{1,8}$ $CAR_{2,6}$, $CAR_{2,7}$
CU_5	RF_2 RF_3	$CAR_{1,4}$, $CAR_{1,5}$ $CAR_{2,3}$, $CAR_{2,4}$, $CAR_{2,5}$, $CAR_{2,6}$

Un detalle relevante de los siguientes casos de uso es que, a excepción del CU_1,1, se asume como precondición global que **los usuarios siempre deben estar logueados**. De lo contrario, no podrán llevar a cabo ninguna de estas acciones y solo tendrán acceso a la página de *login*.

3.2.4. CU_1 - Acceso a cuenta

Se contemplan tres acciones: Cambiar de idioma y *login/logout* del usuario. Todas ellas están asociadas con la cuenta o la sesión de quien accede a la página.

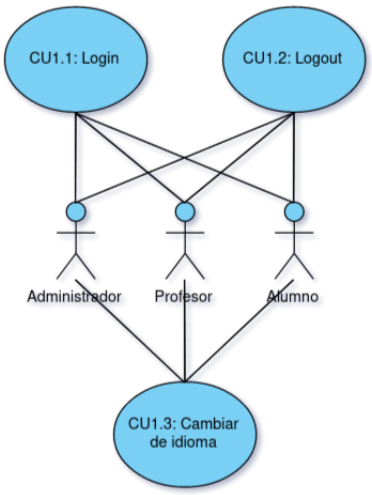

Figura 3.2: Casos de uso del $\mathbf{CU}_1$

$\mathbf{CU}_{1,1}$ - *Login*

Descripción	El usuario accede a su cuenta introduciendo sus credenciales a través de la página de *login*. Según el tipo de usuario que acceda, será redirigido a la página de profesor, administrador o alumno.
Precondición	

$\mathbf{CU}_{1,2}$ - Logout

Descripción	El usuario cierra su sesión y sale de su cuenta. Esta operación se podrá a cabo llevar desde cualquier página.
Precondición	

$\mathbf{CU}_{1,3}$ - Cambiar de idioma

Descripción	El usuario cambia de idioma a cualquiera de los disponibles. El texto de las páginas cambiará al lenguaje seleccionado.
Precondición	

3.2.5. $\mathbf{CU}_2$ - Gestión de asistencias

Incluye tres funciones propias del marcaje de asistencias: Listar las sesiones del profesor para proceder al marcaje, listar la información de dicha sesión y el marcaje de los asistentes.

$\mathbf{CU}_{2,1}$ - Listar sesiones del profesor

Descripción	El profesor visualiza las distintas sesiones que imparte desde el calendario y puede hacer click en sus entradas para acceder a la información de cada una de ellas.
Precondición	

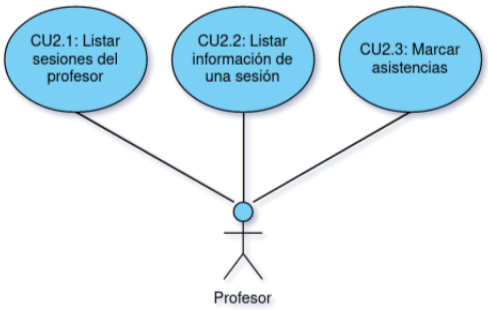

Figura 3.3: Casos de uso del **CU$_2$**

CU$_{2,2}$ - Listar información de una sesión

Descripción	El profesor accede a la página que muestra información de una sesión, incluyendo el curso al que corresponde, el grupo al que imparte, la asignatura y un listado de los alumnos de ese grupo junto con su asistencia (**No indicada**, **Presente**, **Ausente**, **Justificada**).
Precondición	La sesión a la que se accede existe en el sistema.

CU$_{2,3}$ - Marcar asistencias

Descripción	El profesor marca la asistencia de un alumno para una sesión concreta desde la página de la sesión. Podrá marcar 3 posibles tipos de asistencia: **Presente**, **Ausente** y **Justificada**.
Precondición	La sesión y el alumno existen en el sistema

3.2.6. CU$_3$ - Gestión de evaluaciones

Contempla 5 operaciones enfocadas a tratar con la información relativa a pruebas: Listar las pruebas del profesor para proceder a su marcaje, listar los datos de dicha prueba, evaluar a un alumno y la subida y descarga de documentos de corrección de una prueba.

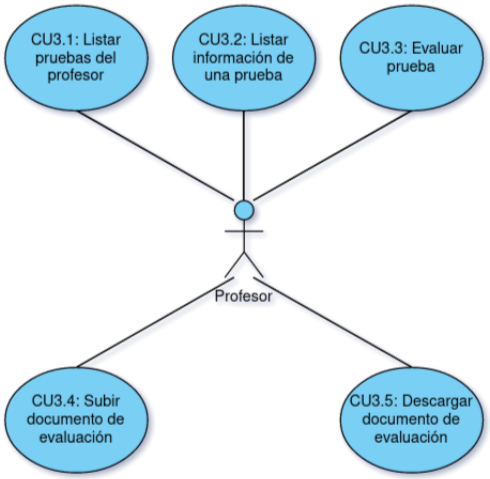

Figura 3.4: Casos de uso del **CU$_3$**

$CU_{3,1}$ - Listar pruebas del profesor

Descripción	El profesor visualiza las distintas pruebas de sus asignaturas desde el calendario y puede hacer click en sus entradas para acceder a la información de cada una de ellas.
Precondición	

$CU_{3,2}$ - Listar información de una prueba

Descripción	El profesor recupera la información correspondiente a una prueba, incluyendo el curso al que corresponde, el grupo que se evalúa, la asignatura a la que pertenece y un listado de los alumnos de ese grupo junto con sus notas, los documentos de sus evaluaciones y el profesor que les ha evaluado.
Precondición	La prueba a la que se accede existe en el sistema.

$CU_{3,3}$ - Evaluar prueba

Descripción	El profesor evalúa a un un alumno en una prueba concreta.
Precondición	La prueba y el alumno existen en el sistema.

$CU_{3,4}$ - Subir documento de evaluación

Descripción	El profesor adjunta un documento a la evaluación de un alumno y la información se almacena en el sistema.
Precondición	La prueba, el alumno y su evaluación existen en el sistema.

$CU_{3,5}$ - Descargar documento de evaluación

Descripción	El profesor descarga en su navegador un documento PDF asociado a la evaluación de un alumno.
Precondición	La prueba, el alumno, su evaluación y el documento existen en el sistema.

3.2.7. CU_4 - Gestión de cursos

Contempla las operaciones que permiten crear los distintos elementos con los que representar el curso y sus unidades: Asignaturas, cursos, grupos, trimestres...

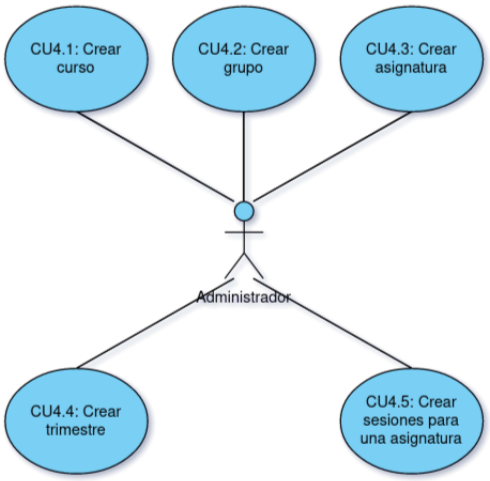

Figura 3.5: Casos de uso del $\mathbf{CU_4}$

$\mathbf{CU_{4,1}}$ - Crear curso

Descripción	El administrador crea un curso indicando su nombre, el año académico en el que tendrá lugar, el centro al que pertenecerá y los grupos asociados a él.
Precondición	El centro debe existir en el sistema.

$\mathbf{CU_{4,2}}$ - Crear grupo

Descripción	El administrador crea un grupo indicando su nombre y el grupo al que estará asociado.
Precondición	El curso debe existir en el sistema.

$\mathbf{CU_{4,3}}$ - Crear asignatura

Descripción	El administrador crea una asignatura indicando su nombre, ID interno, el curso al que pertenece, su tutor, los trimestres que tendrá y sus sesiones.
Precondición	El curso, tutor, trimestres y sesiones deberán existir en el sistema.

$\mathbf{CU_{4,3}}$ - Crear trimestre

Descripción	El administrador crea un trimestre para una asignatura indicando su nombre, fecha de inicio y fin.
Precondición	La asignatura a la que se asociará el trimestre debe existir en el sistema.

$\mathbf{CU_{4,4}}$ - Crear sesiones para una asignatura

Descripción	El administrador crea sesiones para una asignatura indicando el profesor que las impartirá, días de la semana en las que tendrá lugar, horas de inicio y fin y fechas entre las cuales se crearán dichas sesiones.
Precondición	La hora y fecha de inicio deben ser previas a la hora y fecha de fin. El profesor de cada sesión y la asignatura deben existir en el sistema.

3.2.8. Requisitos no funcionales

Los requisitos no funcionales del sistema se han derivado principalmente de tres intenciones:

- Dar a los posibles desarrolladores desinteresados incentivos para participar en el proyecto, tales como una batería de tests y la posibilidad de extender la aplicación con sencillez.
- Facilitar su uso a los potenciales usuarios, haciendo de su interacción un proceso más ameno y simple sin dejar de dado las funcionalidades.
- Permitir que aquellas personas o colectivos que deseen implementar Dagaz puedan hacerlo con facilidad y pudiendo adaptarlo a diversas situaciones y entornos.

RNF$_1$ - Tecnologías libres

Descripción	El sistema debe construirse utilizando tecnologías libres compatibles con GPL. El sistema será licenciado bajo AGPLv3[34].

RNF$_2$ - Sistema modificable

Descripción	El sistema debe permitir su modificación de tal forma que sea posible cambiar su código (servicios, populadores, etc.), sus componentes (base de datos) o su interfaz y estilo.

RNF3 - Cobertura y tests para las áreas críticas de la aplicación

Descripción	Las áreas que otorgan la funcionalidad más relevante de la aplicación deberán contar con una extensa batería de tests y garantizar una cobertura del 100 % de las clases, métodos y las rutas de ejecución.

RNF$_4$ - Interacciones simples

Descripción	Para aquellos casos donde las operaciones a llevar a cabo sean sencillas (marcar asistencia, evaluar a un alumno, subir un documento, etc.), el usuario debe poder realizar su trabajo con un solo click desde la página correspondiente.

RNF$_5$ - Interfaces sencillas

Descripción	Las páginas que componen la interfaz gráfica deben ser autoexplicativas y simples, procurando utilizar colores y guías que permitan al usuario determinar si sus acciones se han llevado a cabo correctamente.

RNF$_6$ - Interfaces con apartado de ayuda

Descripción	Toda página donde se lleven a cabo acciones que afecten a datos debe incorporar una sección de ayuda que guíe al usuario y le ayude a realizar correctamente la operación.

RNF_7 - Restricciones para usuarios no logueados

Descripción	Un usuario que no haya accedido a su cuenta o cuya sesión haya expirado no podrá interactuar con el sistema.

RNF_8 - Interfaces en múltiples idiomas

Descripción	Los textos de las interfaces, alertas, mensajes de error o de éxito; deben estar localizados en español, valenciano e inglés.

RNF_9 - Soporte para múltiples localizaciones

Descripción	El sistema debe permitir la adición de nuevas traducciones a través de archivos de texto *.properties*.

RNF_{10} - Sistema ejecutable en múltiples plataformas

Descripción	El sistema debe poder ejecutarse en varios sistemas operativos, incluyendo Linux, Windows y MacOS.

RNF_{11} - Sistema ejecutable en múltiples arquitecturas

Descripción	El sistema deberá correr en procesadores que empleen sets de instrucciones i386, x86_64 y ARM.

RNF_{12} - Sistema ejecutable en computadores de baja potencia

Descripción	El sistema y su base de datos deberán funcionar en dispositivos poco potentes. Se tomará como punto de referencia una Raspberry Pi 4 Modelo B de 2GB de RAM.

RNF_{13} - Definición programática de las propiedades del sistema

Descripción	La configuración del sistema y sus entidades deben poder definirse de forma programática siempre que sea posible. Se tendrá en cuenta su descripción a través de archivos *.properties* y también la definición de tablas y relaciones de la base de datos mediante anotaciones.

3.3 Plan de trabajo

El plan de acción para Dagaz se dividió en 3 grandes fases:

- Un **periodo inicial de documentación de herramientas**
  Su fin era contar con un listado de tecnologías potencialmente empleables para construir un producto como Dagaz. Puede considerarse como un tiempo dedicado a la formación y exploración de alternativas y un refinamiento de la idea original de este trabajo, mucho antes incluso de la toma de requisitos.

- Una **etapa de investigación**
  Periodo dedicado a recabar documentación de los CMS actuales y, de este modo, extraer potenciales ideas. Simultánemanente, se pretende llevar a cabo encuestas y entrevistas con usuarios de dichos CMS.

- Una **fase de construcción** de la herramienta
  Fundamentada en los datos recabados en los primeros dos tramos del proyecto, abarca el periodo de *setup* de la tecnología a emplear y la definición de su infraestructura, así como del desarrollo de la herramienta. Con el fin de reducir costes y agilizar el desarrollo, el *setup* y la etapa de investigación pueden suceder simultáneamente.

Scrum fue el método elegido para guiar el proceso de distribución de tareas y desarrollo. Esta metodología ágil establece metas temporales, ***Sprints***, para las cuales se desarrollan ciertas funciones del producto. A su término, cada Sprint produce un entregable, y con cada Sprint que pasa se incorporan más funcionalidades al código ya existente. De esta forma, el proceso es iterativo, incremental y aumenta la complejidad del resultado que se presentará.

Preliminarmente se contemplaron varios Sprints de 2-3 semanas de duración, acompañados de pruebas de aceptación de la herramienta y su posterior análisis y retroalimentación. Su fin era que un grupo amplio de docentes de la escuela pública experimentados en el uso de ITACA, uno de los CMS estudiados, pudieran evaluar la usabilidad y las funcionalidades dotadas. Con este análisis sería posible iterar sobre la herramienta y corregir posibles defectos antes de su entrega.

Con la clausura de cada Sprint se haría una retrospectiva, un análisis sobre las metas que se deseaban cumplir y cómo se han cumplido finalmente, así como el desempeño que el equipo ha llevado a cabo con esas metas. La intención es poder encontrar áreas de mejora tanto en el propio proyecto como en la manera de trabajar, y también serviría para poder detectar potenciales mejoras ajenas a los usuarios con quienes se llevarían a cabo las pruebas.

Como más adelante se discutirá, las fases de investigación y construcción se planearon para dar comienzo mucho antes de lo previsto, pero la irrupción del COVID-19 y su impacto en España provocó un cambio de planes en todos los aspectos del plan de trabajo. Se entrará en más detalle en el apartado de Planificación y Organización.

3.4 Presupuesto

Dagaz se planea como un proyecto en tres fases abarcables en un periodo que contempla desde Febrero-Marzo hasta principios de Junio de 2020, compuesto por un equipo formado por un único desarrollador.

Se planea que Dagaz será construido en Java. En el punto 2.4.2, correspondiente al análisis de Sakai (CMS también desarrollado en Java), se aportó una estimación sobre el sueldo medio de un desarrollador Java a tiempo completo (40h semanales) en España, rondando los 2250€[24]. Contemplando que el desarrollador encargado trabajará a tiempo parcial (4h/día entre semana), incluyendo fines de semana (8h/día) a tiempo completo, su capacidad total ronda las 36 horas semanales.

Tomando como base el sueldo indicado previamente y haciendo la proporción horas de trabajo/€, el coste mensual será de 2025€. Multiplicándolo por los 4 de meses de desarrollo, el total destinado sumaría 8100€.

Dada la naturaleza de la licencia con la que se pretende construir Dagaz, las dependencias y librerías que se utilicen en ella deberán ser compatibles con las licencias GPL. Esto implicará que siempre que mantengamos una licencia compatible, no existirán restricciones ni solicitudes de permiso de uso, tasas o costes similares. Su utilización sería gratuita y, por tanto, no sería necesario dedicar dinero a este área del desarrollo. Podrá leerse más al respecto en el apartado Tecnologías libres.

Las licencias de desarrollo y los programas para construir la aplicación podrían suponer un desembolso considerable en la mayoría de proyectos, mas no es el caso para Dagaz. Los entornos de desarrollo a emplear ejecutan una distribución gratuita de GNU/Linux y el IDE escogido, IntelliJ IDEA, dispone de una versión gratuita para la comunidad. Los plugins de los que se goza en dicho IDE, que supondrán una mayor rapidez en el desarrollo de la solución, también son de uso gratuito. Podrá leerse más al respecto en la sección Herramientas utilizadas.

3.5 Análisis de riesgos

Un detalle relevante a mencionar en este apartado y que condicionó el resto del desarrollo de Dagaz es que el impacto del COVID-19 nunca fue contemplado dentro de los riesgos del proyecto. Por tanto, fue necesario un análisis posterior con medidas de contención para mitigar al máximo el impacto. Así pues, se puede dividir el análisis de riesgos en pre-COVID y post-COVID.

3.5.1. Riesgos pre-COVID-19

RI_1 - Librería o dependencia fácilmente reemplazable es incompatible con GPL

Contexto	Se contempla que el elemento afectado puede ser sustituido con facilidad y pertenece a una parte fácilmente intercambiable del programa (como la base de datos o el conector).
Impacto	Impide licenciar el proyecto bajo licencias GPL.
Medidas	Reemplazar la librería o dependencia por una alternativa funcionalmente equivalente.
Tipo	Producto

RI_2 - Librería o dependencia empleada en parte del código es incompatible con GPL

Contexto	Se contempla que el elemento afectado se utiliza en el proyecto y su sustitución es posible, pero puede requerir retrabajo.
Impacto	Impide licenciar el proyecto bajo licencias GPL.
Medidas	Renegociar alcance con el cliente para hacer frente al tiempo a dedicar para introducir ese cambio. Incluir en el backlog una historia de deuda técnica para reemplazar el código afectado. Sustituir la librería o dependencia por una alternativa funcionalmente equivalente y reescribir el código en cuestión.
Tipo	Producto - Proyecto

RI_3 - Librería o dependencia empleada extensamente es incompatible con GPL

Contexto	Se contempla que el elemento afectado se utiliza ampliamente en el proyecto y sustituirlo es inviable económica/temporalmente.
Impacto	Impide licenciar el proyecto bajo licencias GPL.
Medidas	Renegociar licenciado con el cliente. Licenciar el proyecto bajo MIT [53].
Tipo	Producto - Proyecto

RI_4 - Población homogénea entre los usuarios entrevistados

Contexto	El número de docentes entrevistados puede haber sido bajo, dando lugar a opiniones homogéneas o demasiado orientadas a un aspecto concreto de la usabilidad de la herramienta.
Impacto	Toma de requisitos incompleta o enfocada a un público no representativo del posible público objetivo.
Medidas	Aprovechar las redes sociales para emitir entrevistas con otros tipos de usuarios mediante plataformas de encuesta online.
Tipo	Proyecto

RI_5 - Baja aceptación en una prueba con usuarios

Contexto	Tras realizar una prueba con usuarios, sus puntuaciones sobre cierta área de la aplicación son bajas.
Impacto	Posible baja recepción por parte del público objetivo en el momento de la salida del programa.
Medidas	Entrevistar usuarios y recoger su valoración sobre los aspectos criticados en la prueba. Elaborar historias para corregir estas áreas e introducirlas en el próximo Sprint como prioritarias.
Tipo	Producto - Proyecto

RI_6 - Baja aceptación de la interfaz de usuario

Contexto	El equipo no cuenta con diseñadores gráficos ni desarrolladores *frontend*. Al carecer de esa experiencia, es posible que los usuarios pueden encontrar compleja o poco agradable a la interfaz con la que trabajarán en el futuro.
Impacto	Posible baja recepción por parte del público objetivo en el momento de la salida del programa.
Medidas	Contratación de un desarrollador *frontend* para el diseño e implementación de las interfaces afectadas.
Tipo	Producto

3.5.2. Riesgos post-COVID-19

RI_7 - No es posible realizar las pruebas de aceptación in situ

Contexto	Bajo la emergencia sanitaria del COVID-19, la reunión con otros individuos está limitada y puede suponer la expansión del virus.
Impacto	Pérdida del *feedback* asociado a las pruebas de aceptación.
Medidas	Realización de entrevistas y pruebas de aceptación por vía telemática, mediante videollamada o técnicas similares. Reducción del grupo de usuarios de pruebas.
Tipo	Proyecto

RI_8 - Contagio de COVID-19 en el equipo

Contexto	El tiempo de baja por contagio de COVID-19 supone una periodo que puede exceder los 15 días de baja, poniendo en riesgo los Sprints del proyecto. Dado el tamaño del equipo (1 miembro), no se cuenta con sustituciones.
Impacto	El proyecto queda congelado por un periodo de mínimo 15 días y máximo teórico de 1 mes. Los tiempos del proyecto no se cumplirían.
Medidas	No existen medidas para paliar este impacto.
Tipo	Proyecto

RI_9 - El cliente congela el proyecto

Dado que se trata de un trabajo de investigación y desarrollo y no existe una restricción económica real, esta situación es ficticia. Sin embargo, si se tratara de un proyecto llevado a cabo por una empresa real contratada por un cliente, existiría dicha posibilidad. Por este motivo se ha optado por incluirla en el análisis de riesgos.

Contexto	El cliente se ve obligado a posponer los pagos correspondientes al desarrollo de este proyecto, congelando su avance por un tiempo indefinido. Se contempla que la decisión viene tomada por la inestabilidad económica, social y médica.
Impacto	El proyecto queda congelado *sine die* y no es posible hacer frente a los pagos de los sueldos de los desarrolladores.
Medidas	Utilización de mecanismos de protección social para empresas y trabajadores, tales como ERTE u otros fondos sociales dedicados durante el estado de alarma del COVID-19, hasta que el cliente pueda retomar su trabajo.
Tipo	Proyecto

CAPÍTULO 4
Diseño de la solución

4.1 Arquitectura del sistema

Para describir el cuerpo y forma de Dagaz será necesario presentar el sistema a dos niveles distintos: Arquitectura general y funcional.

4.1.1. Arquitectura general

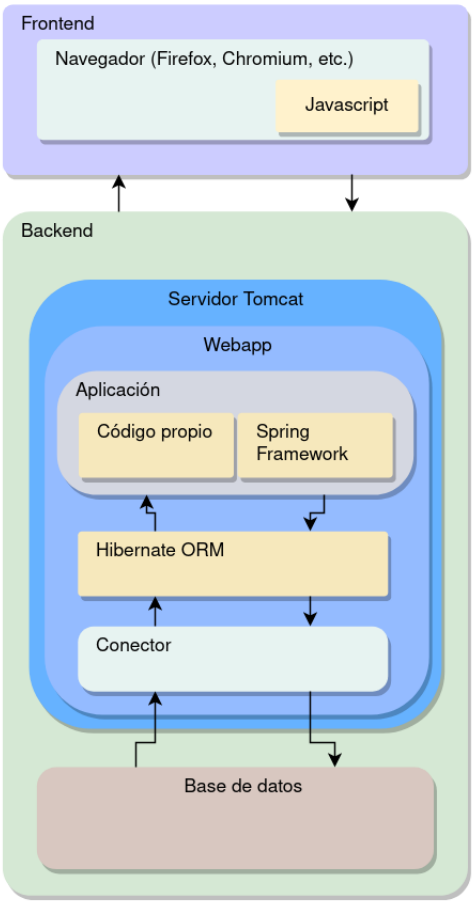

Figura 4.1: Arquitectura general de **Dagaz**

Dagaz es una *webapp*, esto es, una aplicación ejecutada desde un servidor y que es accedida por los usuarios a través de una página web. Por tanto, podemos encontrar dos secciones:

Backend

El *backend* lo conforman un **servidor**, que otorga la funcionalidad de la aplicación, se encarga de servir las páginas al usuario que interactúa con él y de recibir y procesar los datos que se le comunican; y una **base de datos** de la que extrae y almacena datos.

Para desarrollar esta área se han empleado varias herramientas, pero las más relevantes son Spring Framework con Spring Boot y Hibernate. Sus roles, licencias y otros detalles serán explicados en detalle más adelante en la sección Tecnologías libres.

- **Spring Framework** contiene una serie de utilidades diseñadas para la construcción de *webapps*, incorporando además mecanismos de inyección de dependencias, soporte para creación de aplicaciones con el patrón modelo-vista-controlador, mecanismos de seguridad y sesión...

  Spring Boot, a su vez, trae consigo configuración automática para las propiedades del servidor y la interconexión con la base de datos, sirviendo como *boilerplate*. Por defecto, Spring utiliza **Apache Tomcat**[37] para lanzar sus *webapps*.
- **Hibernate** es una herramienta de mapeo relacional de objetos concebida para Java. A través de Hibernate es posible definir los objetos que se almacenarán en la base de datos sin depender de la estructura o características de la misma. Un uso adecuado de Hibernate permite que Dagaz sea una aplicación que funciona **independientemente de la base de datos a emplear**. Otro detalle relevante es que permite el uso de repositorios JPA para extraer o modificar objetos mediante consultas independientes del lenguaje que emplee la base de datos subyacente.

Por motivos de seguridad y pragmatismo, la mayor parte de la lógica se encuentra en este extremo.

La base de datos aporta funciones CRUD. Habiendo la posibilidad de escoger entre varios modelos, se ha optado por emplear bases de datos relacionales por familiaridad y la utilidad que aportan al conjunto de la aplicación. Este elemento puede encontrarse alojado en el mismo dispositivo que el resto del *backend* o en otros computadores.

Gracias a Hibernate es posible utilizar cualquier base de datos (siempre que sea compatible con dicho *framework*). Por tanto, queda a juicio del implementador decidir cuál es la alternativa más adecuada para sus necesidades. La base de datos emplea también un **conector** que le permite comunicarse con el resto del sistema. Durante las fases de desarrollo se utilizó inicialmente MariaDB, aunque posteriormente se pasó a PostgreSQL. Podrá leer más información al respecto en la sección dedicada a PostgreSQL.

Frontend

Engloba la interfaz del usuario y los procesos que suceden en ella. Los usuarios accederán a él a través de un navegador, que renderizará las páginas que recibe del servidor y transmitirá datos al mismo. Toda información emitida por esta fuente y recibida por el servidor se considera inherentemente insegura (los datos podrían no estar validados, o podrían fingirse llamadas desde dispositivos que pretendan vulnerar la seguridad del sistema), por lo que el *input* es interceptado previamente por Spring Security, comprobado por *tokens* y otras verificaciones de sesión y posteriormente validado de acuerdo a mecanismos especificados por el programador.

Entre los datos que se transmiten para mostar cada página se pueden incluir pequeños *scripts* escritos en JavaScript. La intención es facilitar la interacción entre usuario e interfaz y evitar que las acciones que se lleven a cabo causen un refresco completo de la página. Esta sería la única lógica que se ejecutaría fuera del servidor.

4.1.2. Arquitectura funcional

La arquitectura funcional es una descripción a grandes rasgos de la separación que existe entre distintos tipos de clases de acuerdo a sus funciones y los datos con los que tratan.

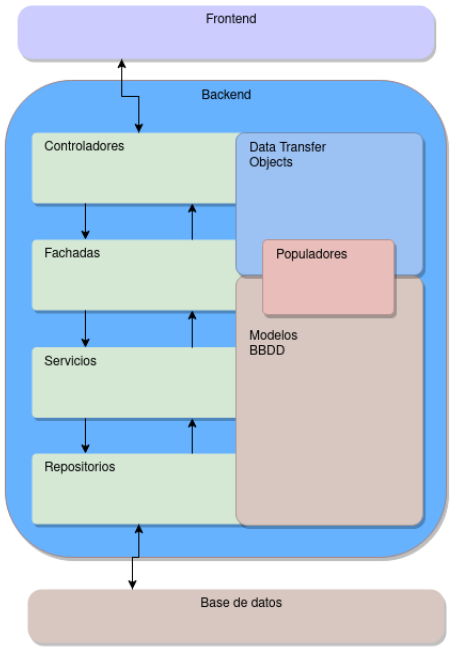

Figura 4.2: Arquitectura de datos de **Dagaz**

Con el fin de ser más específicos en el resto de la exposición sobre su arquitectura, es necesario hablar primero sobre los datos que podemos encontrar en ella:

- **Modelos de base de datos**: Representan un mapeo entre la información almacenada en la base de datos y las clases Java que se emplearán en el resto del código. Incorporan anotaciones con las que describir programáticamente las columnas de cada tabla, las relaciones entre las mismas, etc. A grandes rasgos, son contenedores de datos.
- **DTOs** (por sus siglas en inglés *Data Transfer Objects*, objetos de transferencia de datos): Son POJOs, *plain old Java object*, objetos de Java carentes de lógica que únicamente incorporan campos de datos y sus *getters/setters* asociados. Su finalidad es simplificar los valores de los modelos y preparar ciertos campos para los objetos que posteriormente se usarán en el *frontend* (*e.g.* traducción de campos, cambios de comas decimales, agrupación de valores, eliminación de relaciones, etc.). La finalidad es producir un objeto ligero y de poca complejidad con el que sea fácil interactuar.
- **Modelo de controlador**: Mapa que contiene colecciones de DTOs y otros tipos de datos que serán utilizados para construir una vista.

Con la intención de darle mayor flexibilidad a la aplicación se idearon 5 separaciones entre sus clases de acuerdo a la funcionalidad que aportarían:

- **Repositorios**: Capa de acceso y almacenamiento de datos que permite interactuar con los valores almacenados en la BBDD. Devuelven **modelos**.
- **Servicios**: Aportan funciones con las que trabajar con los modelos. Aglutinan la mayor parte de la lógica de toda la aplicación, e interactúan con los repositorios para crear, recuperar, actualizar y guardar información.
- **Populadores**: Clases diseñadas para convertir datos de un tipo a otro, principalmente para transformar **modelos** en **DTOs**.
- **Fachadas**: Recogen los datos aportados por los servicios y los transforman mediante populadores. A través de ellas es posible ocultar múltiples métodos de los servicios cuyo uso no sería recomendable en los controladores. Incorporan poca lógica, pues relegan la mayor parte del trabajo en los servicios.
- **Controladores**: Constituyen el punto de contacto con el mundo exterior, permiten definir *endpoints* o direcciones con las que el *frontend* (y potencialmente otros sistemas que se integren con Dagaz) puede comunicarse con la plataforma.

  Interactúan con el resto de la aplicación mediante las fachadas y los métodos que éstas hacen visibles. Los controladores devuelven páginas o fragmentos HTML, así como texto con distintos formatos (texto plano, JSON, etc.).

  El **controlador** trabaja conjuntamente con otros dos elementos más para permitir la aplicación del patrón **modelo-vista-controlador**:

  - **Modelo de controlador**: Contiene los datos con los que se construirá la página (DTOs y sus valores, textos u otra información necesaria). Obviamente, corresponde al **modelo**.
  - **Páginas JSP**: Representaciones de páginas en HTML con anotaciones específicas que actúan a modo de lenguaje de *scripting*. Dotándoles de un modelo con información y ejecutando los *scripts* que incorporan, el servidor es capaz de generar dinámicamente una página HTML de acuerdo a los valores indicados. Corresponden a la **vista**.

4.1.3. Modificabilidad de la arquitectura

La decisión de emplear Spring Framework para la construcción de la aplicación tiene muchos beneficios, pero el más significativo de cara a la arquitectura funcional es la capacidad de inyectar dependencias. Spring emplea las llamadas *beans*, clases u objetos que se encuentran gestionados por los mecanismos de *IoC* ("inversión de control" en inglés). A través de ellas es posible indicar cuáles serán las dependencias que tendrá un objeto o clase antes de su creación. De tal manera, es posible indicar qué implementaciones concretas, qué *beans* utilizará una clase.

Definiendo una serie de interfaces que actúan como contrato y permiten decir cuáles serán las funciones que deberá incorporar cualquier clase que las implemente, es posible crear clases alternativas y sustituir aquellas funciones que vienen *out of the box* ("por defecto" en inglés) con la implementación básica de Dagaz por las que resulten más adecuadas para la casuística que el implementador requiera.

Con esta perspectiva, aquellas personas que decidan implementar Dagaz y realizar sus propias adaptaciones podrán reemplazar la funcionalidad por defecto y sustituir las dependencias de una clase como si se tratara de meros bloques de construcción a través de un archivo XML.

Así pues, si se desea modificar las condiciones que permiten determinar si un usuario puede administrar un centro (método `CenterService#canAdministerCenter`), solo se tendrá que:

1. Definir una implementación de `CenterService` sobreescribiendo ese método o cualquier otro que se desee sustituir
2. Crear un *bean* asociado a dicha implementación y asignarle un ID
3. Asociar ese ID con el alias utilizado para definir las dependencias que equivalen a `CenterService`

A partir de ese momento, todas las clases del proyecto podrán utilizar la nueva versión en lugar de la predeterminada, permitiendo mantener el código y modificarlo con el menor impacto y esfuerzo.

4.1.4. Arquitectura del *frontend*

Para construir el *frontend* se ha hecho uso de varias librerías en la que entraremos en más detalle en la sección de Tecnologías libres, entre ellas se ha empleado Bootstrap para la construcción de la mayor parte de las vistas y de JQuery para realizar operaciones sobre los elementos del DOM.

Aunque Spring Framework ofrece distintos tipos de MVC, el que se ha utilizado en esta solución emplea **JSP** (*Java Server Pages*), que permite definir páginas con código HTML acompañado de un lenguaje de *scripting* y un modelo con datos. De acuerdo a los *scripts* y los valores del modelo es posible construir las vistas de forma dinámica.

Con respecto a las páginas y los elementos que las conforman, podemos hacer dos separaciones:

- Las **vistas**, corresponden a los archivos `.jsp`. Son páginas completas compuestas de código HTML y otros fragmentos. Son devueltas por los controladores.
- Las **plantillas**, formadas por archivos `.tag`, son pequeños fragmentos que pueden emplearse en los `.jsp` y permiten reutilizar código.

La conjunción de estas dos partes permite devolver páginas de cierta complejidad, pero de forma fácil y eficiente. Otras alternativas requieren ejecutar JavaScript para la generación de estas interfaces en el extremo del cliente, así que reduciendo el esfuerzo que debe realizarse en el extremo del usuario también incrementamos la velocidad a la que puede acceder a la página.

4.2 Diseño

4.2.1. Diagrama de la BBDD

El diagrama de la base de datos y el de sus clases correspondientes es prácticamente idéntico, salvando la diferencia de aquellas clases donde se producen relaciones * a * (se emplean tablas intermedias), así como aquellas que emplean múltiples claves ajenas como su clave primaria (utilizan clases auxiliares con el prefijo *ID*). Nótese que los nombres utilizados están en inglés y contienen el sufijo *Model*. Su traducción será proporcionada en el área correspondiente a cada uno de ellos.

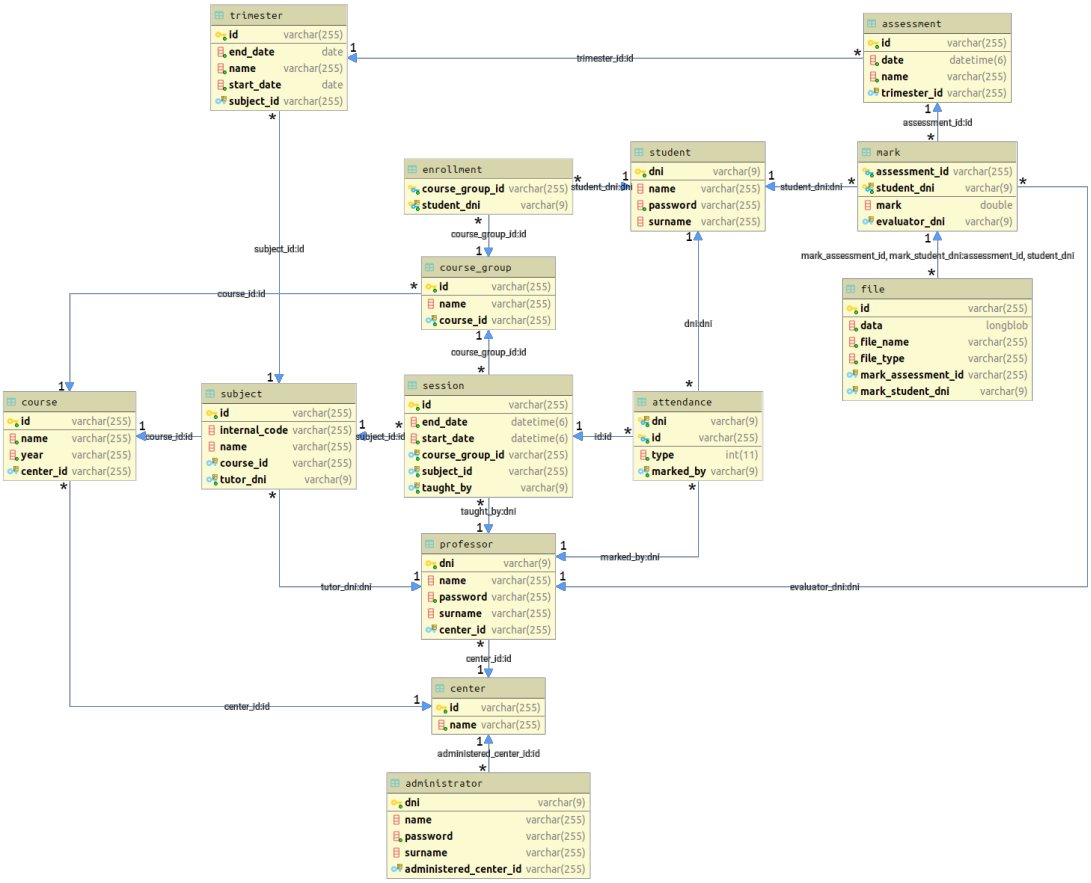

Figura 4.3: Diagrama de clases de la base de datos - Generado con yFiles desde IntelliJ IDEA

Hibernate ha sido la tecnología utilizada para la construcción del diagrama de la base de datos. Gracias a este ORM, es posible definir cuál será la composición de la tabla, sus atributos y relaciones; a través de la escritura de clases Java. Así es posible esbozar cuáles serán los contenedores de datos que se emplearán en el código y cómo se interrelacionan.

Este singluar detalle también permite que la modificación de la base de datos sea mucho más simple. Hibernate incorpora características para crear desde cero, regenerar o actualizar la estructura en base a lo indicado en las clases Java cuando se arranca la aplicación. Por tanto, incluir nuevos atributos o modificar los ya existentes es un trabajo mucho menos tosco, ya que no es necesario hacer uso de SQL para edificar el modelo que será empleado.

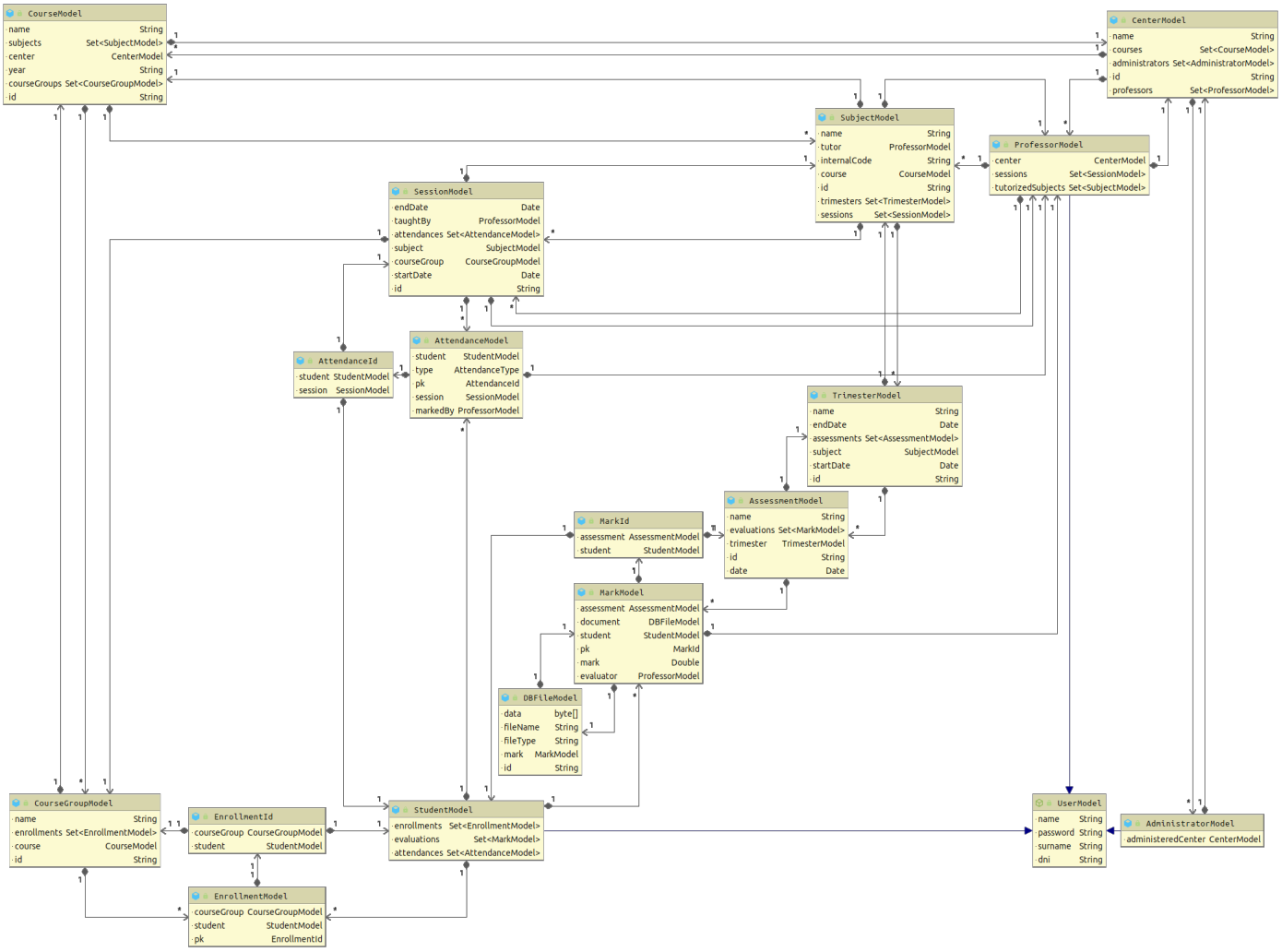

Figura 4.4: Diagrama de clases en Java - Generado con yFiles desde IntelliJ IDEA

Los elementos que pueden encontrarse se clasifican en 3 grupos: Curso, asignatura y usuarios. Además, existe un caso específico para describir los archivos guardados en la base de datos.

Entidades del curso

Abarca las entidades que permiten la descripción de las unidades organizacionales del curso: Centro, curso, grupos y matriculaciones.

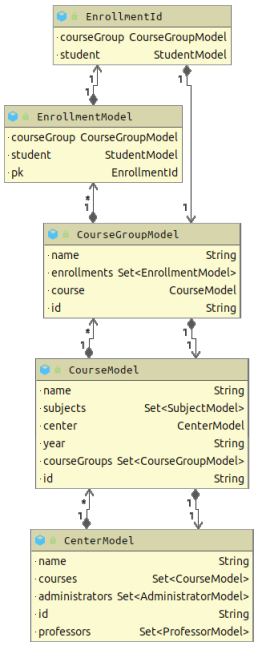

Figura 4.5: Diagrama de las entidades del curso - Generado con yFiles desde IntelliJ IDEA

- **Center** (Centro)

  Describe un centro donde se imparte docencia. El centro englobaría a profesores, docentes, cursos, etc. Todo centro tiene una serie de usuarios administradores que permiten la gestión de matriculaciones, la creación de cursos, grupos, asignaturas y otros menesteres.

  Ejemplo: Universidad Politècnica de València.

- **Course** (Curso)

  Corresponde a un curso académico creado para un año concreto. Está compuesto por un conjunto de grupos y asignaturas comunes y pertenece a un centro.

  Ejemplo: 4º del Grado en Ingeniería Informática de 2020.

- **Course group** (Grupo de curso)

  Son agrupaciones de alumnos para un curso concreto. Cada grupo tendrá sus sesiones y horarios específicos para cada asignatura.

  Ejemplo: 4ºA - Grupo de mañanas.

- **Enrollment** (Matriculación)

  Entidades lógicas que permiten indicar la pertenencia de un estudiante a un grupo concreto.

Entidades de asignaturas

Incluye aquellas entidades que definen los elementos para modelar las asignaturas de un curso concreto, tales como son sus trimestres, sus sesiones, las asistencias a dichas sesiones, las pruebas evaluativas y también las notas de las mismas.

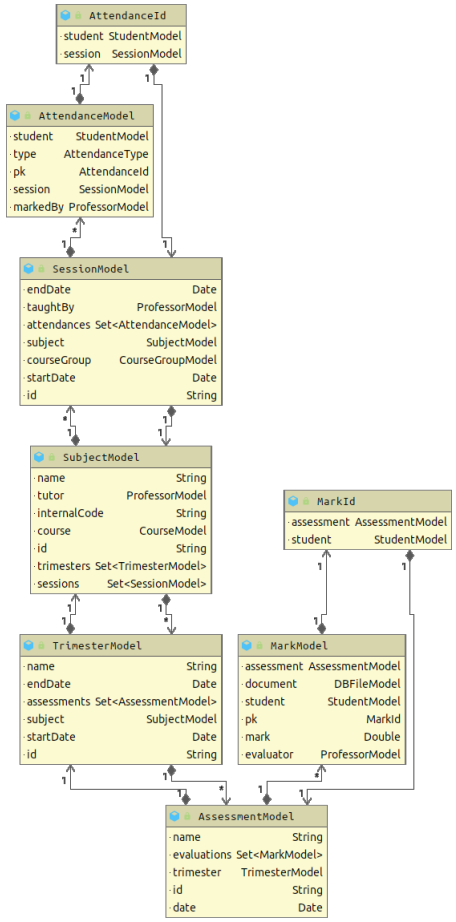

Figura 4.6: Diagrama de las entidades asociadas a asignaturas - Generado con yFiles desde IntelliJ IDEA

- **Subject** (Asignatura)

  Objeto que modela una asignatura. Toda asignatura tendrá un profesor tutor, estará asociada a un curso y la conformarán uno o más trimestres. A su vez, cada asignatura tendrá una serie de sesiones.

  Ejemplo: Arquitectura de computadores.

- **Trimester** (Trimestre)

  Unidades de tiempo asociadas a una asignatura. A pesar de su nombre, no refiere necesariamente a un periodo de 3 meses, pues la longitud de los mismos es configurable y adaptable. Cada trimestre tendrá una fecha de inicio y final, así como un listado de las pruebas ejecutadas durante el mismo.

- **Session** (Sesión)

  Acto destinado a representar una clase o actividad a la que deben asistir ciertos alumnos. Una sesión tendrá una asignatura asociada, así como un grupo de curso y fechas de inicio y fin. Todos los alumnos que pertenezcan a dicho grupo deberán asistir a la sesión.

Ejemplo: Sesión de arquitectura de computadores - Lunes 23 de Marzo [10:00 - 11:00].

- **Attendance** (Asistencia)

  Indica la presencia de un alumno en una sesión. Por motivos de seguridad, se registra también qué profesor ha marcado la asistencia o ausencia de cada alumno.

  Se contemplan 4 posibles modos:

  - **No indicado** - Estado por defecto cuando todavía no se ha marcado la asistencia
  - **Presente** - Utilizado cuando el alumno ha acudido a la sesión
  - **Ausente** - Empleado en aquellos casos en los que el alumno se ausenta sin causas justificadas
  - **Justificada** - Empleado para ausencias justificadas

- **Assessment** (Prueba)

  Las pruebas son una forma de modelar tanto exámenes/tests como entregables u actividades evaluables del curso. Toda prueba tendrá una fecha y pertenecerá a un trimestre concreto para una asignatura. A las pruebas se presentarán todos los alumnos que pertenezcan a una asignatura concreta.

  Ejemplo: Examen de arquitectura de computadores - Lunes 28 de Marzo [10:00].

- **Mark** (Nota)

  Las notas son la evaluación dada por un profesor a una prueba concreta. Cada alumno que se presente a una prueba tendrá su propia nota. Por motivos de seguridad, se indicará qué profesor ha evaluado a qué alumno. Es posible agregarle un documento, tal como podría ser la corrección del alumno.

Entidades de usuarios

Parten todas ellas de la misma base "Usuario", y equivalen a los tres tipos de actores que Dagaz contempla a día de hoy: Profesores, administradores y alumnos.

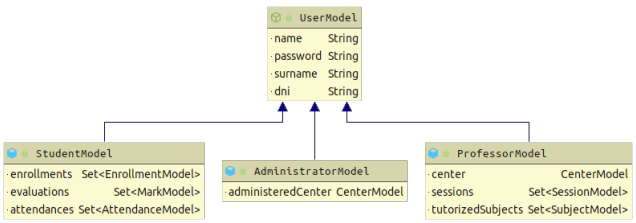

Figura 4.7: Diagrama de las entidades asociadas a usuarios - Generado con yFiles desde IntelliJ IDEA

El **User** o "Usuario" es la base común de la que parten. Tiene nombre, apellidos, un DNI que constituye un identificador único y una contraseña de acceso. Las clases que derivan de él son:

- **Professor** (Profesor)

  Persona que pertenece a un centro concreto, imparte sesiones, marca asistencias y evalúa pruebas. Puede darse el caso de que además sea el tutor de una o más asignaturas.

  Este tipo de usuarios acceden a la perspectiva del profesor una vez entran en la herramienta.

- **Administrator** (Administrador)

  Usuario encargado de la creación de cursos, grupos, asignaturas y trimestres. Ejerce un rol de gestión.

  Los administradores acceden a la perspectiva del administrador cuando entran a la herramienta.

- **Student** (Alumno)

  Estudiante al cual se imparte docencia, está matriculado en un grupo concreto, asiste a unas sesiones y es evaluado por medio de pruebas.

  Este tipo de usuarios accedería a la perspectiva de alumno. Sin embargo, tal vista no está disponible para esta entrega.

Almacenaje de archivos en la base de datos

Con la intención de guardar documentos asociados a las notas de los alumnos se concibió una entidad nombrada *DBFile* (Archivo de base de datos), una tabla sencilla que almacena nombre, un *array* de *bytes* y el tipo de dato del archivo. Actualmente solo puede existir asociado a una nota, aunque teóricamente se podría extender con otras finalidades.

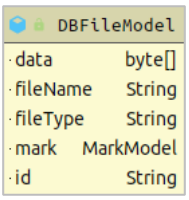

Figura 4.8: Clase *DBFileModel* - Generado con yFiles desde IntelliJ IDEA

4.2.2. Clases funcionales del proyecto

Repositorios

Los repositorios JPA constituyen el punto de contacto entre la base de datos y el resto de la aplicación. A través de ellos es posible buscar objetos concretos de acuerdo a sus claves primarias, así como generar consultas a la base de datos a través de los mecanismos habilitados por Hibernate (HQL, consultas mediante nombres de métodos, etc.).

Los repositorios constituyen la capa de acceso a datos y son extensamente empleados en los servicios. En parte, su simplicidad es también su virtud, pues contribuyen a una facilidad de uso mucho mayor. La interacción con el resto de la base de datos queda oculta gracias a ellos.

Podrá evidenciarse en el nombre de los métodos cómo se ha hecho uso de las capacidades de Hibernate para generar consultas automáticamente. Su sintaxis emplea los campos de las entidades como puntos de referencia, e incorpora operaciones genéricas propias de las bases de datos (ordenación, fechas, etc.). Las uniones de las distintas tablas en el momento de la consulta son controladas completamente por Hibernate.

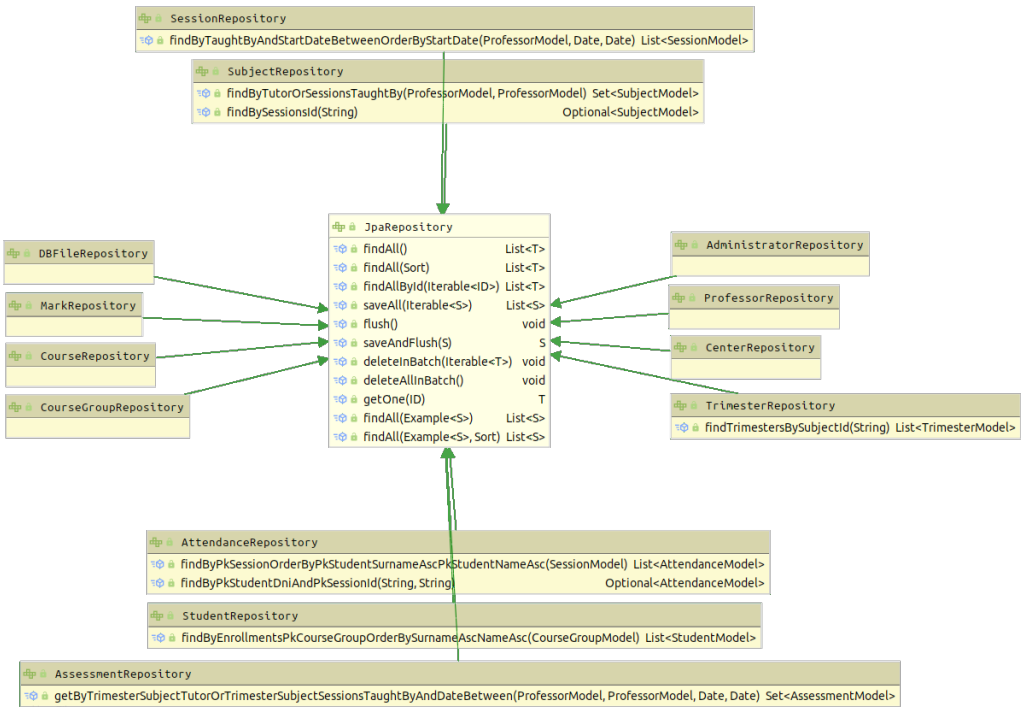

Figura 4.9: Repositorios de Dagaz - Generado con yFiles desde IntelliJ IDEA

Nota: En los diagramas que se mostrarán a continuación se verán asociaciones entre clases implementadas (denotadas por el prefijo *Default*).

En la implementación real del programa, una clase implementada **siempre** se relaciona con interfaces en lugar de sus implementaciones, pues esto permite inyectar dependencias sin incorporar las restricciones que podría suponer emplear una clase implementada, además de facilitar su sustitución.

Dado que estas dependencias pueden ser modificadas si se desea mediante la creación de nuevas *beans*, se han tomado las clases por defecto para la generación de los diagramas.

Servicios

Los servicios aglutinan la mayor parte de la lógica del código. A grandes rasgos, tienen dependencias entre ellos y también entre los repositorios de los que extraen información.

Los servicios tienen implementaciones *Default*, las que se consideran como la versión original o por defecto de las mismas. Pueden visualizarse sus dependencias en la siguiente imagen.

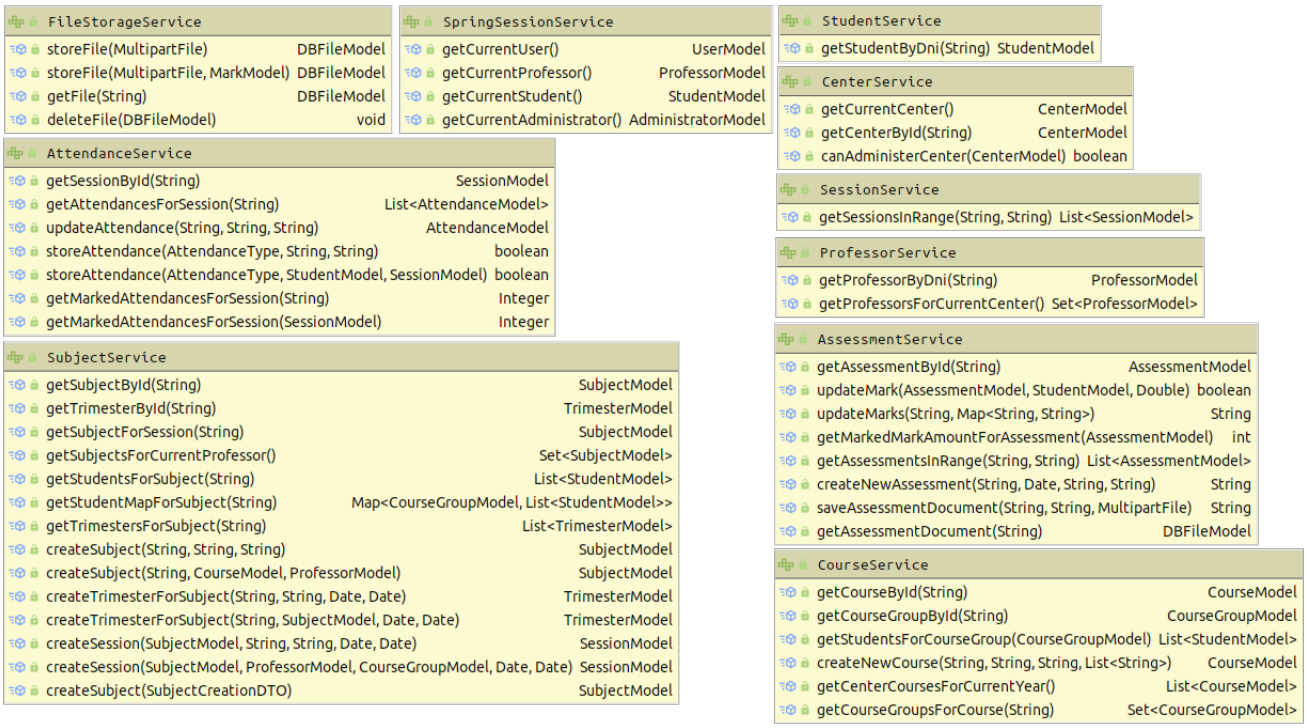

Figura 4.10: Interfaces de servicios - Generado con yFiles desde IntelliJ IDEA

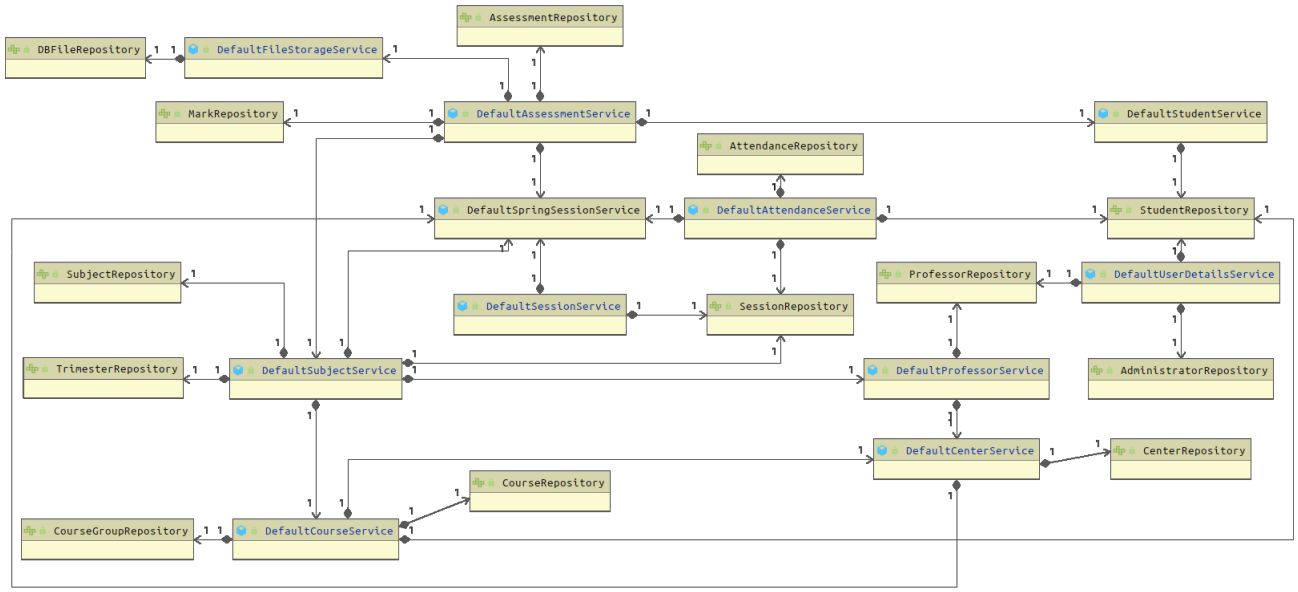

Figura 4.11: Servicios y sus dependencias - Generado con yFiles desde IntelliJ IDEA

Populadores

Los populadores son los encargados de convertir datos de un tipo a otro distinto. Todos ellos parten de una clase abstracta, `AbstractPopulator`, que sirve como base para indicar cuáles serían las operaciones a llevar a cabo.

Todo populador está preparado para convertir objetos sea de forma individual o en colecciones. Pueden depender de servicios u otros populadores para llevar a cabo las transformaciones.

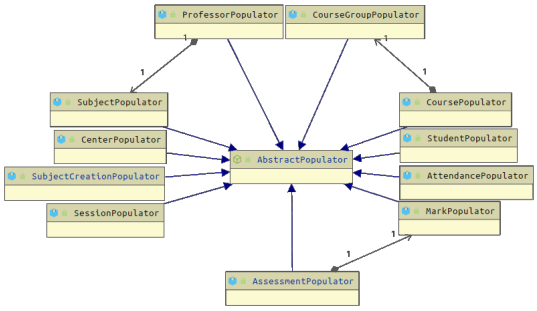

Figura 4.12: Populadores y sus dependencias - Generado con yFiles desde IntelliJ IDEA

Fachadas

Transforman datos de *Model* a *DTO* (o viceversa) utilizando los servicios para guardar/recuperar datos y los populadores para modificarlos.

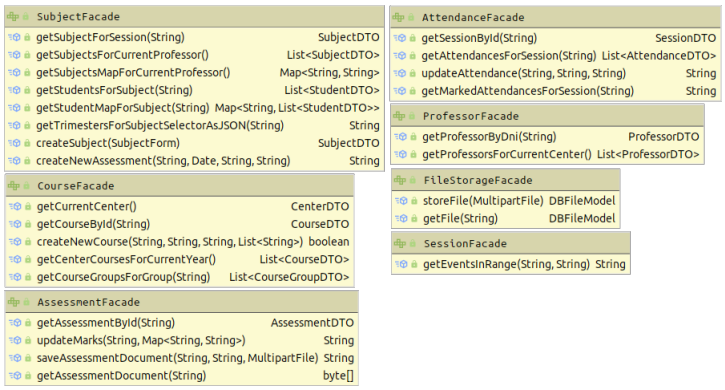

Figura 4.13: Interfaces de fachadas - Generado con yFiles desde IntelliJ IDEA

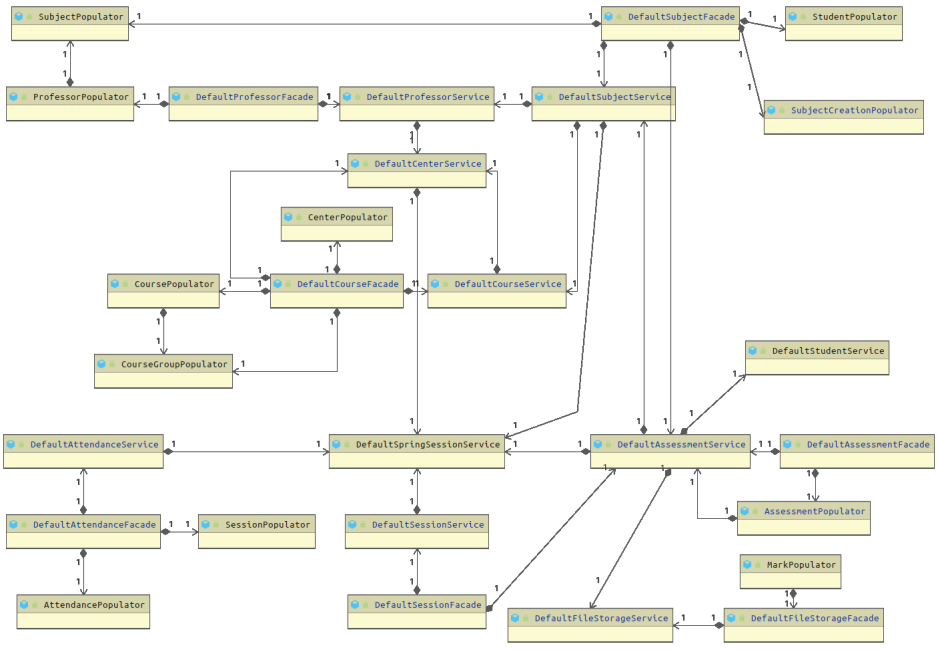

Figura 4.14: Fachadas y sus dependencias - Generado con yFiles desde IntelliJ IDEA

Controladores

Con los controladores da comienzo la capa de presentación. Podemos encontrar 5 controladores con funciones específicas:

- `LoginController`, controlador responsable de las tareas de login y redirección a la página principal de cada uno de los usuarios.
- `ProfessorController`, agrupa las operaciones que puede llevar a cabo un profesor.
- `AdministratorController`, similar al caso anterior pero para administradores.
- `ApiController`, un controlador que implementa una API REST básica con la que retornar ciertos datos al *frontend*. La intención es extenderlo de cara al futuro si así fuera necesario para integrar a través de él las distintas llamadas que podrían ser necesarias para que otras aplicaciones puedan comunicarse con Dagaz.
- `AssessmentPageController`, controlador específico para la página de evaluaciones y que permite la subida y descarga de archivos asociados a las notas de cada alumno.

Los controladores se comunican con las diversas fachadas, así como con dos clases utilitarias:

- `AlertMessages`, una clase diseñada para ser extensible y ayudar en la construcción de mensajes que serán mostrados en las alertas de las páginas. Acompañado de un fragmento en el *frontend*, muestra información de forma unificada en las distintas vistas.
- `BreadcrumbUtils`, similar a `AlertMessages` pero concebida para controlar los *breadcrumbs*, pequeñas guías que ayudan al usuario a saber en qué página de qué área se encuentra en cada momento.

Roles Gracias a las utilidades provistas por Spring Framework, es posible indicar "roles" a los distintos usuarios que interactúan con la aplicación y restringir de esta manera su acceso a ciertos recursos o direcciones dentro de la página. Así pues, cada uno de estos controladores solo será accesible por uno de los 4 roles disponibles:

- `PROFESSOR`, empleado para profesores. Se gasta en los controladores de `ProfessorController` y `AssessmentPageController`.
- `ADMINISTRATOR`, equivalente del caso anterior pero pensado para administradores. Únicamente se emplea en `AdministratorController`.
- `STUDENT`, diseñado para ser usado por los estudiantes, pero sin funcionalidad en el estado actual de la herramienta.
- `API_USER`, otorgado a aquellos usuarios a quienes se les desea dar la oportunidad de emplear la API existente en la herramienta.

  Este rol existe con una previsión de futuro, pues podría emplearse para identificar a las herramientas que se integren con Dagaz.

  A través de él se podría limitar acceso a los métodos o controladores que se decidieran, incrementando la seguridad de la herramienta y permitiendo su compartimentalización.

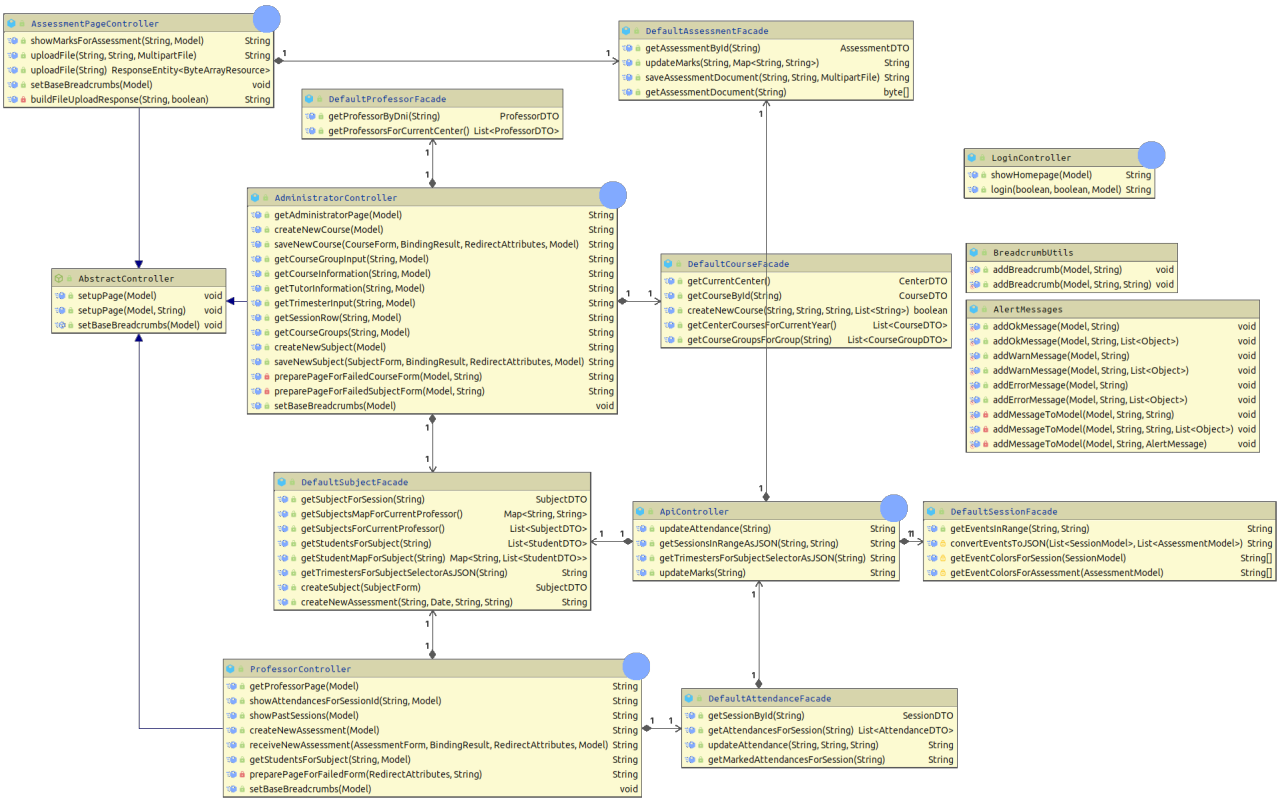

Figura 4.15: Controladores, sus dependencias y clases asociadas - Generado con yFiles desde IntelliJ IDEA
Los controladores están identificados con un marcador azul

4.3 Comportamiento del sistema

A continuación se describirán mediante diagramas de secuencia las funciones provistas por el sistema para las funcionalidades contempladas en el análisis de la aplicación. Se han tomado las siguientes decisiones con respecto a su representación:

- De ser posible, **se utilizará el nivel de servicio como punto de partida**. Dado que los populadores, fachadas y controladores tienen un rol de control de vistas y preparación de datos para mostrar en el *frontend*, su representación en estos esquemas resta importancia y dificulta su lectura.
- Aquellas características que vengan introducidas por defecto en Spring Framework, donde el caso más notable es el login, serán excluidas de este análisis al tratarse de código del *framework* y, por tanto, no ser producto de este trabajo.
- Es digno de mención que buena parte de los métodos de la aplicación emplean **lambdas** y ***streams***, funciones inspiradas en la programación funcional. Dado que su representación es poco factible en un diagrama de secuencia, para mostrarlas se emplearán bucles y condicionales como si se tratara de programación convencional en Java.
- A lo largo del proyecto existen varios métodos dedicados a devolver objetos de un repositorio concreto en base a su clave primaria. La operación será omitida. A grandes rasgos, su funcionamiento se puede describir con el siguiente diagrama genérico:

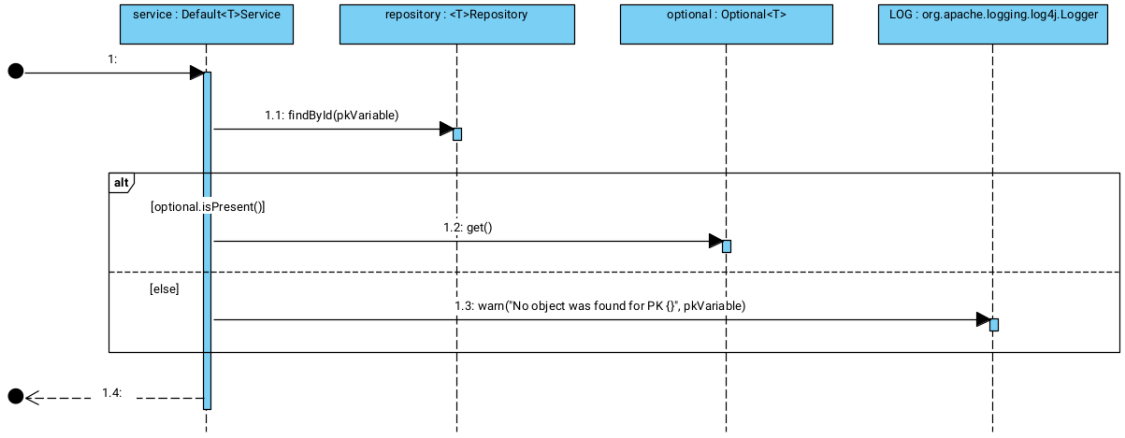

Figura 4.16: Extracción de modelos por clave primaria - Generado con VisualParadigm

4.3.1. $CAR_{1,1}$ - Listado de asistencias

La siguiente característica se manifiesta en la página del listado de asistencias, donde es posible consultar qué alumnos han acudido. Para recuperar las asistencias de una sesión será necesario indicar su ID.

Cuando un profesor busque por primera vez la sesión se generarán todas las asistencias de un alumno, preparadas con un estado de asistencia `NOT_SET`. En este proceso pueden existir incidentes, tal como que se produzca un error al almacenar el objeto en la base de datos, en cuyo caso se borran todas las asistencias que se hubieran generado para dicha sesión.

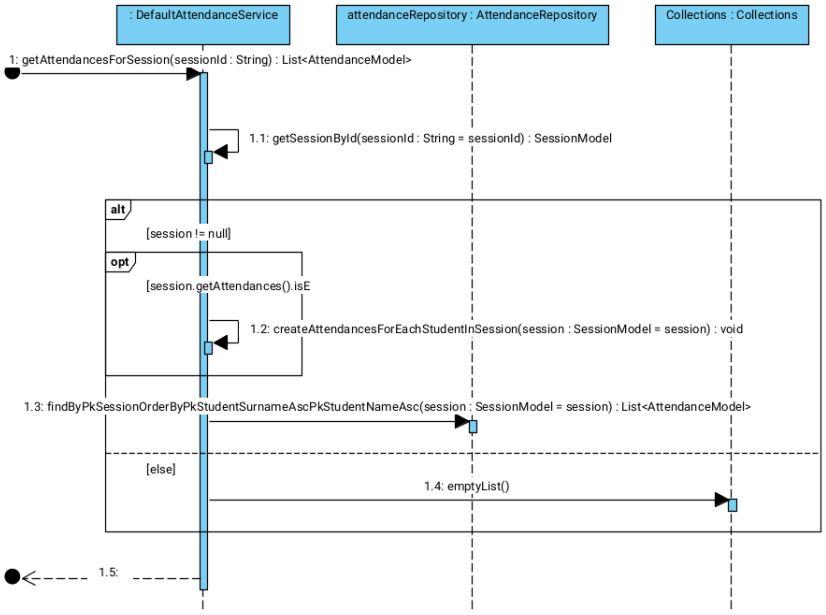

Figura 4.17: $CAR_{1,1}$ - Listado de asistencias - Generado con VisualParadigm

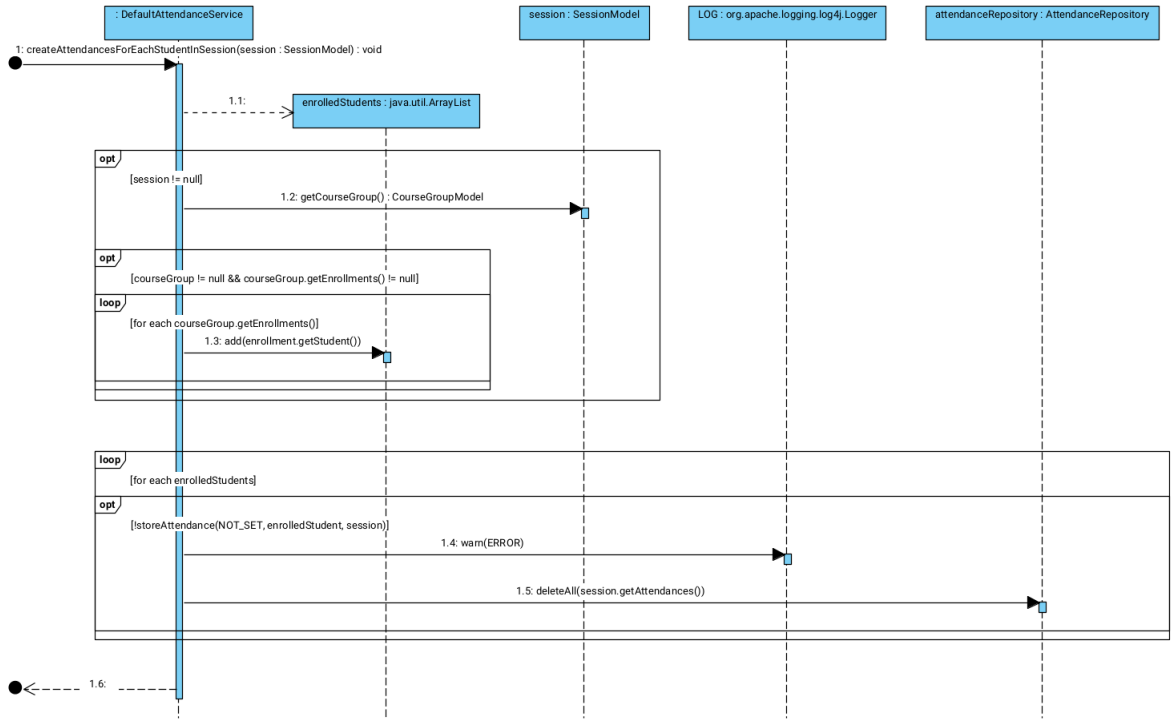

Figura 4.18: $CAR_{1,1}$ - Creación de asistencias para sesión - Generado con VisualParadigm

4.3.2. $CAR_{1,2}$ - Listado de notas

De acuerdo a la implementación actual de Dagaz, las notas están asignadas a pruebas y alumnos. Puesto que no existe aún la perspectiva del alumno, la única manera de acceder a dichas notas es a través de las páginas de notas de la vista del profesor. Por tanto, para recuperar las notas de una prueba bastará con recuperar la prueba que las contiene. Para obtenerlas solo sería necesario utilizar el repositorio correspondiente, `AssessmentRepository`, indicarle el ID de la prueba y luego recuperar sus evaluaciones.

También es posible devolver la nota para un alumno y prueba concretos, tal como se indica en la siguiente figura:

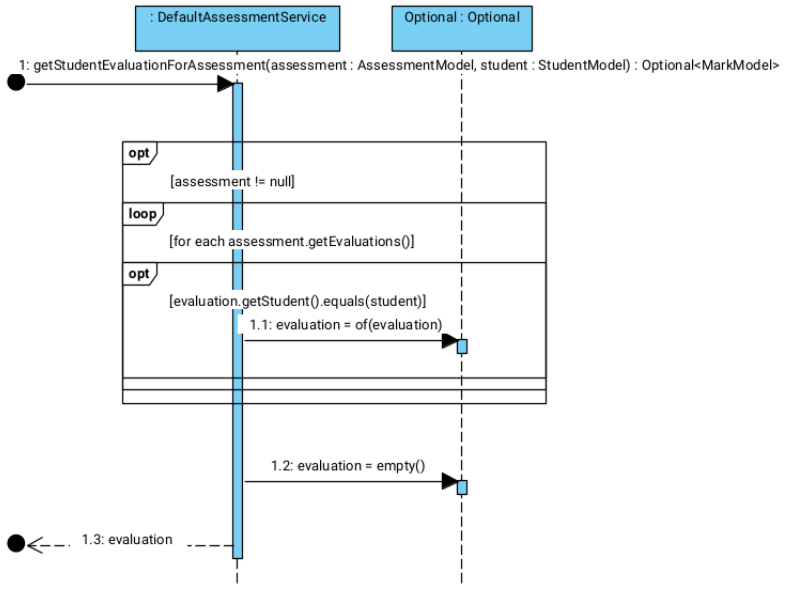

Figura 4.19: $CAR_{1,2}$ - Listar nota de un alumno para una prueba - Generado con VisualParadigm

4.3.3. $CAR_{1,3}$ - Listado de alumnos

El listado de alumnos puede venir de tres formas distintas para varias situaciones:

$CAR_{1,3}$ - Listado de alumnos de un grupo

La idea de obtener los alumnos de un grupo es saber cuáles son los estudiantes que pertenecen a un grupo concreto. En los grupos existen los objetos de matriculación `EnrollmentModel`, que serán la base para realizar este listado.

Dado que el repositorio es capaz de hacer la mayor parte del esfuerzo de este método, asumiendo el filtrado y ordenado de los datos que queremos obtener, el resultado es una función sencilla que consiste en una simple condición.

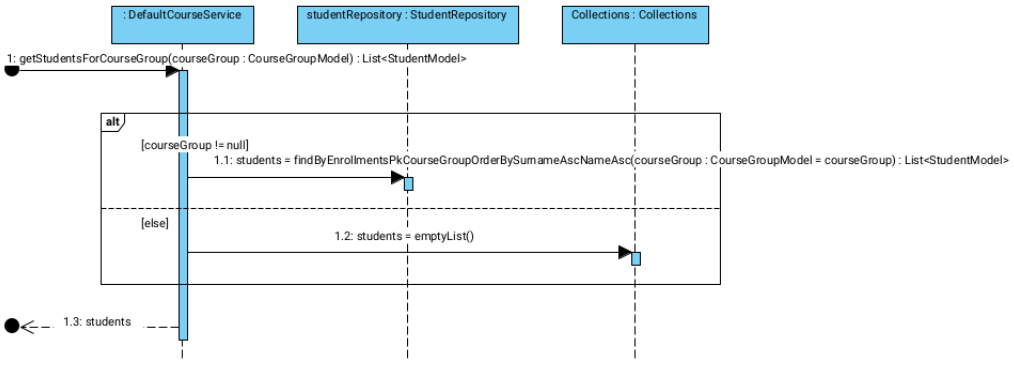

Figura 4.20: $CAR_{1,3}$ - Listar alumnos de un grupo - Generado con VisualParadigm

$CAR_{1,3}$ - Listado de alumnos de una asignatura

Si lo que se desea es obtener los alumnos de una asignatura, el proceso es similar, aunque en esta ocasión se hace uso del método descrito en el punto anterior. Cada asignatura tiene un listado de cursos, así que la operación consistirá en recoger cada uno de esos cursos y extraer sus alumnos.

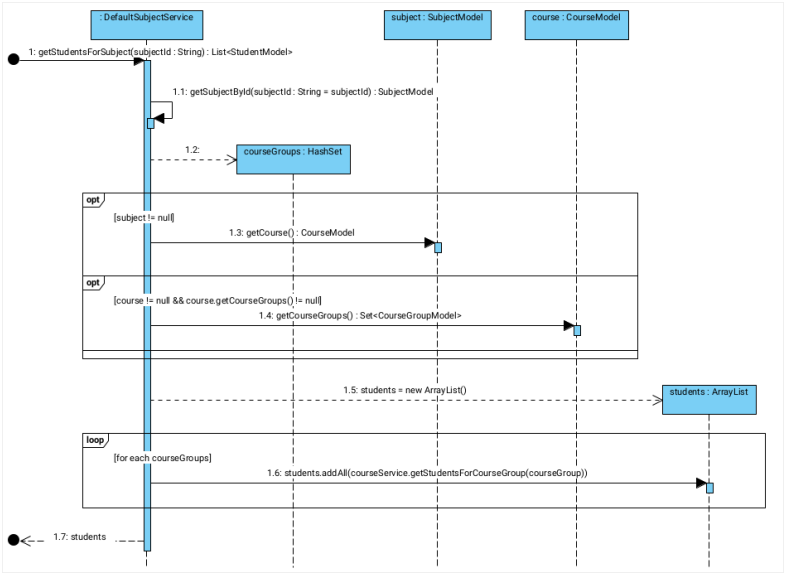

Figura 4.21: CAR1.3 - Listar alumnos de una asignatura - Generado con VisualParadigm

$CAR_{1,3}$ - Listado de alumnos por grupo para una asignatura

Finalmente, en algunas situaciones tal vez es relevante retornar esta información en forma de mapa, de tal manera que se tenga cada grupo de una asignatura junto con la lista de alumnos que pertenecen a él.

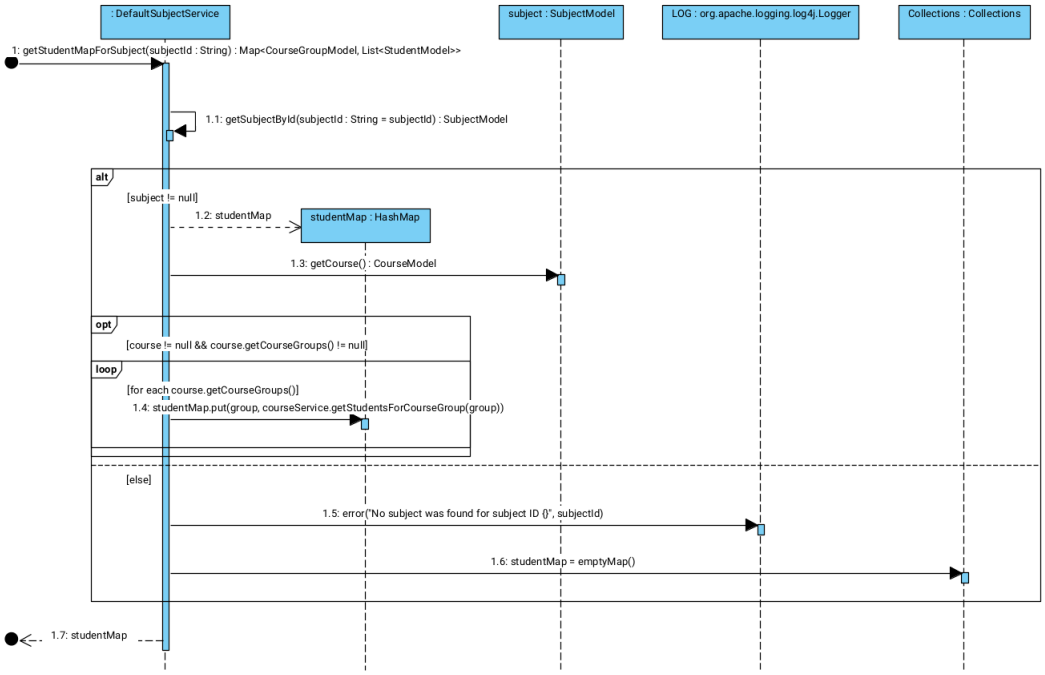

Figura 4.22: $CAR_{1,3}$ - Mapa de alumnos de una asignatura por grupos - Generado con VisualParadigm

4.3.4. $CAR_{1,4}$ - Listado de cursos

De acuerdo al modelado de Dagaz, todos los cursos están asignados a un centro. Asimismo, cada curso tiene su propio año, de tal forma que sea posible filtrarlos si fuera necesario.

La primera forma de la que se puede lograr esto con el código actual es recuperando todos los cursos de un centro. Ésta es una operación trivial, se puede lograr indicando el ID de dicho centro, recuperándolo a través del repositorio tal y como se había explicado anteriormente y, finalmente, ejecutando la operación `getCourses()` sobre el objeto devuelto.

El listado de cursos solo es utilizado por los usuarios de administración. De tal forma, todos ellos tendrán un centro administrado. Si se pretende que la aplicación sea útil, en ese caso solo se deberán devolver aquellos cursos que sean relevantes. Se consideran relevantes los cursos que pertenecen al año educativo actual.

Dado que solo sería menester devolver aquellos cursos que estén asociados con el centro al que pertenece el usuario, se ha concebido el siguiente método:

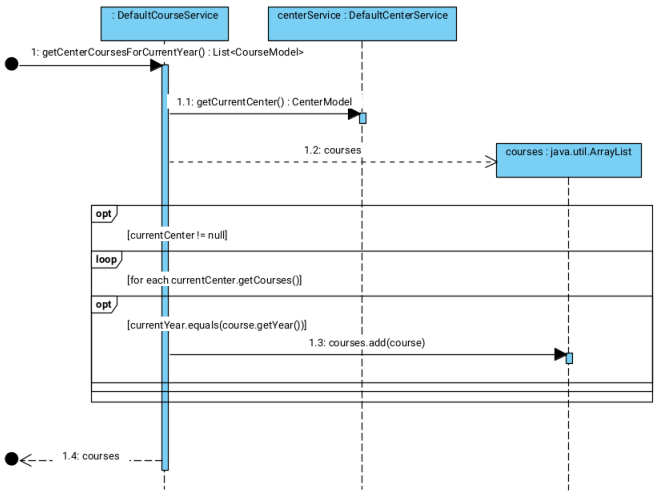

Figura 4.23: $CAR_{1,4}$ - Listado de cursos para el año actual y centro del usuario actual - Generado con VisualParadigm

También es digno de mención el método que obtiene el centro del usuario actual:

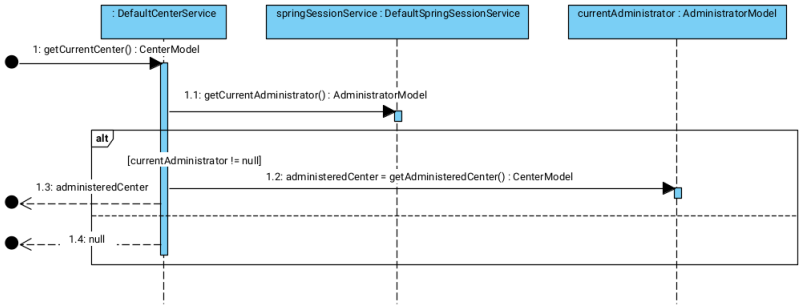

Figura 4.24: $CAR_{1,4}$ - Obtención del centro del usuario actual - Generado con VisualParadigm

4.3.5. $CAR_{1,5}$ - Listado de grupos

Los grupos están relacionados con un curso concreto. Si recuperamos ese curso solo tendremos que devolver el resultado de su `getCourseGroups()`.

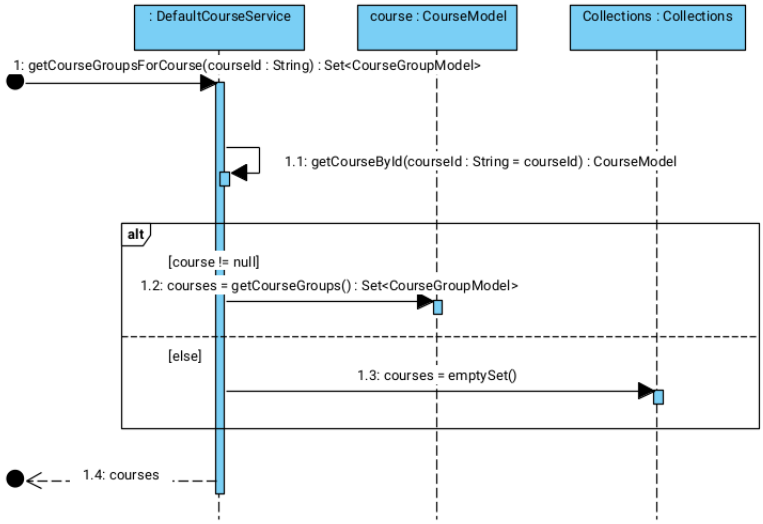

Figura 4.25: $CAR_{1,5}$ - Listado de grupos para un curso - Generado con VisualParadigm

4.3.6. $CAR_{1,6}$, $CAR_{5,2}$ - Listado de sesiones / Calendario para profesores

El calendario de profesores (**$CAR_{5,2}$**) está compuesto a grandes rasgos de tres pasos:

1. Extraer el listado de sesiones impartidas por el profesor que realiza la consulta que tienen lugar entre dos fechas (que es la finalidad del **$CAR_{1,6}$**)
2. Extraer el listado de pruebas impartidas o tutorizadas por el profesor que realiza la consulta que tienen lugar entre dos fechas
3. Realizar una transformación para mostrar estos datos en forma de JSON, de tal forma que el *frontend* pueda mostrarlo en la página del calendario

Por comodidad, ambas características se mostrarán juntas. Damos comienzo por la lista de sesiones, cuyo proceso consiste en transformar dos String recibidos, fechas de inicio y final, en objetos `Date` de Java con los que hacer la búsqueda. Asimismo, para recuperar las fechas del usuario que se encuentra en sesión será necesario utilizar el `SpringSessionService`.

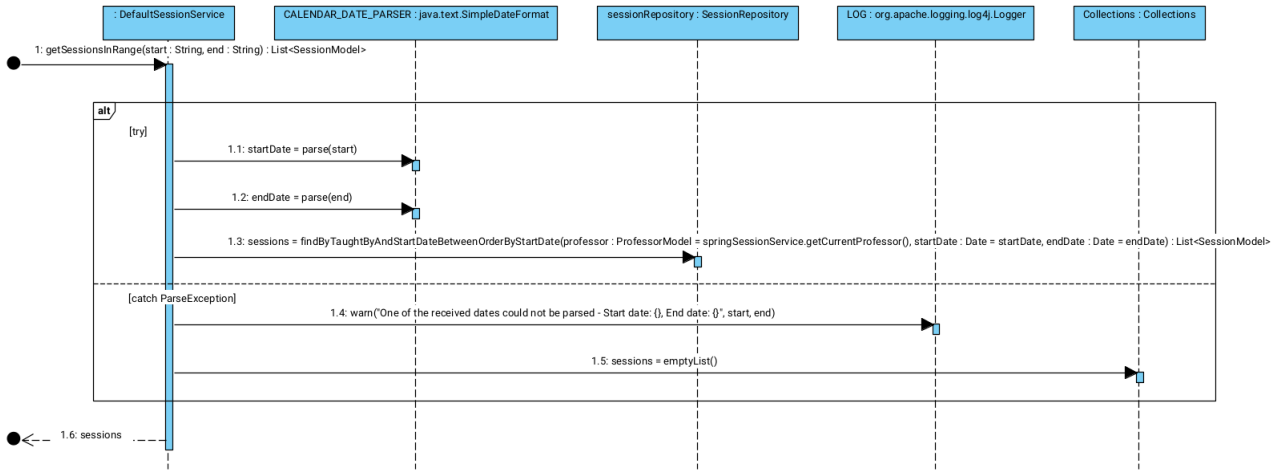

Figura 4.26: $CAR_{1,6}$ - Obtener las sesiones del usuario contenidas entre un rango de fechas - Generado con VisualParadigm

Nótese la extensión del nombre del método al que se llama de `SessionRepository`. Dicho método es el encargado de hacer el filtrado por fechas y por profesor.

El proceso para obtener las pruebas es muy similar: Se obtiene cuál es el profesor actual, se transforman los Strings que denotan las fechas de inicio y final y se relega la operación al repositorio.

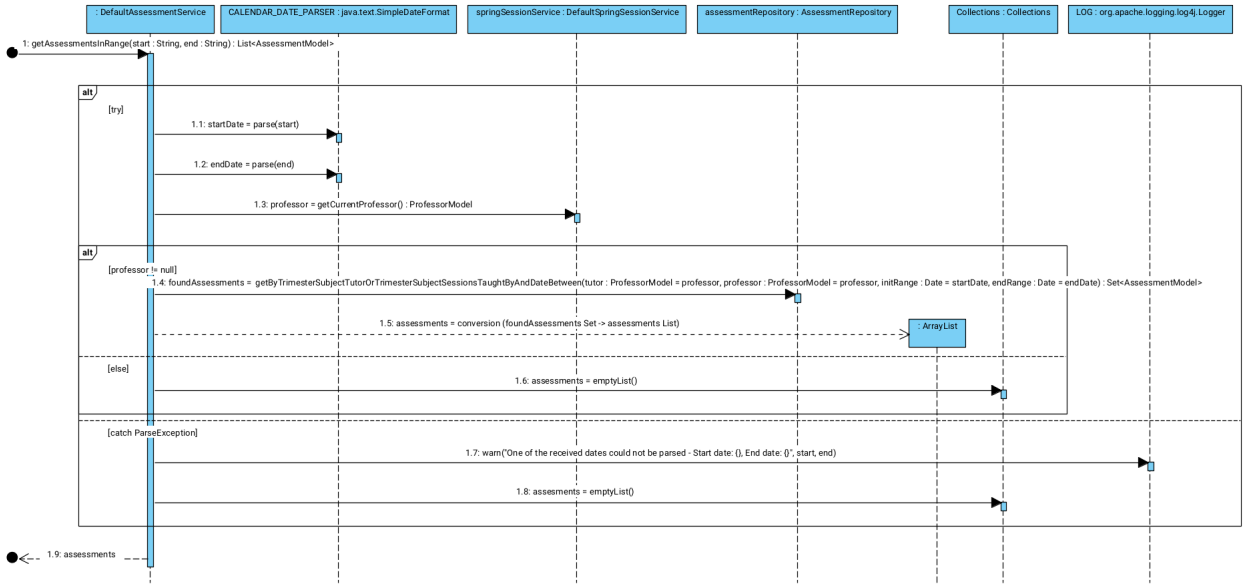

Figura 4.27: $CAR_{1,6}$ - Obtener las pruebas del usuario contenidas entre un rango de fechas - Generado con VisualParadigm

Finalmente, tanto las sesiones como las pruebas son transformadas al formato JSON y devueltas en forma de String para que el controlador pueda transmitirlas al *frontend*. Se logra de la siguiente manera:

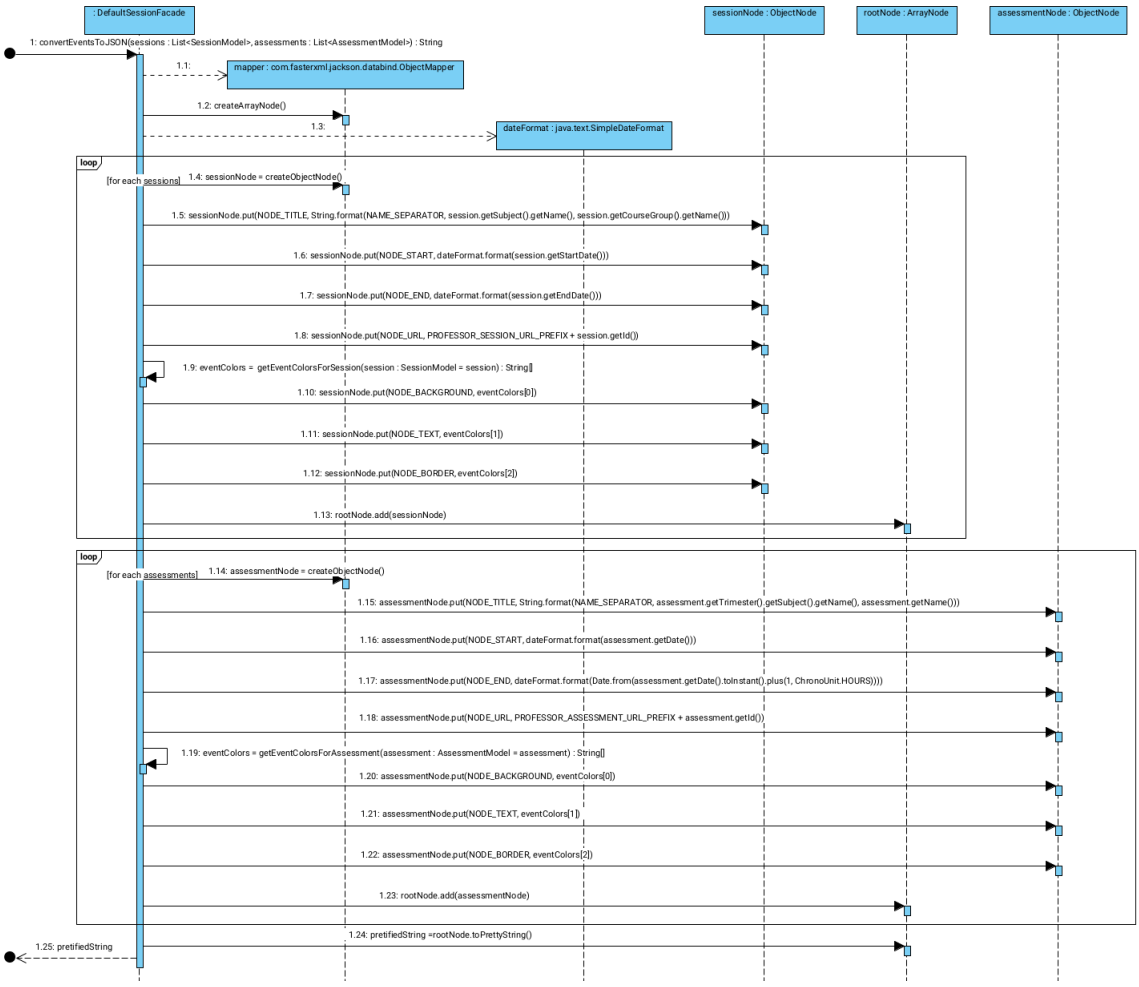

Figura 4.28: $CAR_{1,6}$ - Obtener los eventos del usuario entre un rango de fechas - Generado con VisualParadigm

4.3.7. $CAR_{1,10}$, $CAR_{1,11}$, $CAR_{1,12}$ - Acceso para profesores / estudiantes / administradores

El proceso de *login* es una de las funciones que vienen incorporadas de forma predeterminada en Spring Framework. Buena parte de su lógica es, por tanto, irrelevante para este análisis.

Sin embargo, para poder permitir que Spring busque a posibles usuarios dentro de la base de datos sí ha sido necesario introducir ciertas modificaciones. Spring cuenta con un servicio basado en la interfaz `UserDetailsService` que permite reconocer a los usuarios e insertarlos adecuadamente en sesión. Tan solo es necesario sobreescribir el bean asociado a ese servicio, introducir nuestros cambios y pasarle el identificador único del usuario (su DNI en este caso).

Estos cambios también permiten introducir los roles de los usuarios (llamados `Authorities` en Spring) en el momento del *login*.

Dado que el login es idéntico para todos los usuarios, lo único que tendría que tenerse en cuenta es la detección del tipo de usuario. Por ello, primero se busca la cuenta como usuario de tipo `Professor`, después de tipo `Administrator` y finalmente, `Student`. Si no se encuentra, el login es fallido.

Así pues, necesitaremos 3 pasos: La obtención del usuario, la obtención de su rol y el paso del usuario logueado al resto de Spring para que el proceso pueda seguir su curso de acuerdo a la lógica del *framework*.

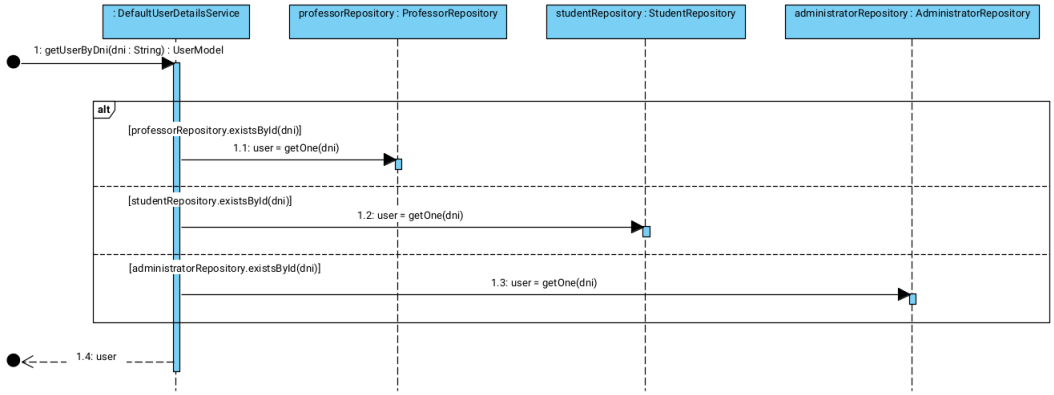

Figura 4.29: $CAR_{1,10}$, $CAR_{1,11}$, $CAR_{1,12}$ - Obtención del usuario durante el login - Generado con VisualParadigm

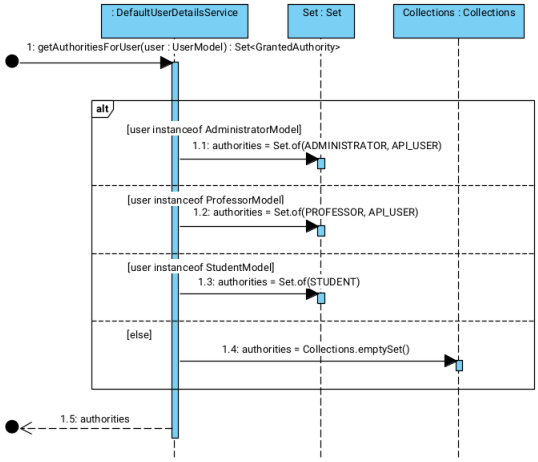

Figura 4.30: $CAR_{1,10}$, $CAR_{1,11}$, $CAR_{1,12}$ - Obtención del rol del usuario - Generado con VisualParadigm

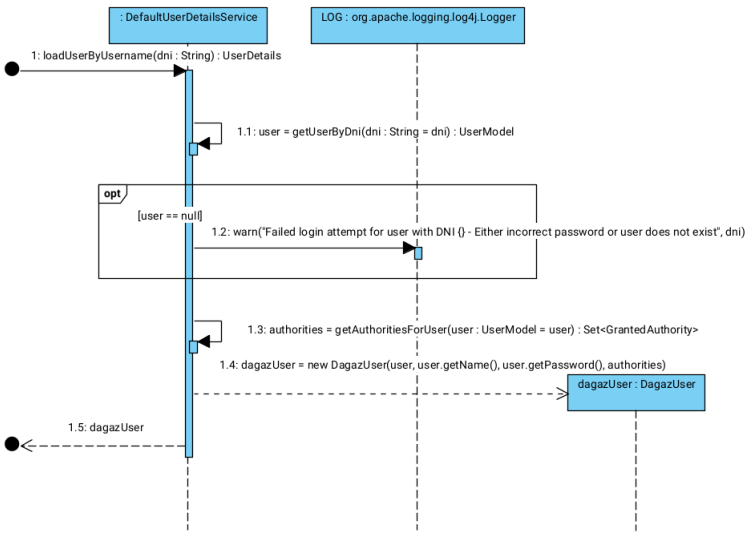

Figura 4.31: $CAR_{1,10}$, $CAR_{1,11}$, $CAR_{1,12}$ - Creación del objeto de login DagazUser - Generado con VisualParadigm

4.3.8. $CAR_{2,2}$ - Marcaje de asistencias

Para marcar una asistencia serán necesarios 3 parámetros: El tipo de la asistencia, el ID de la sesión y el DNI del alumno cuya asistencia se va a marcar. Así pues, el proceso es sencillo:

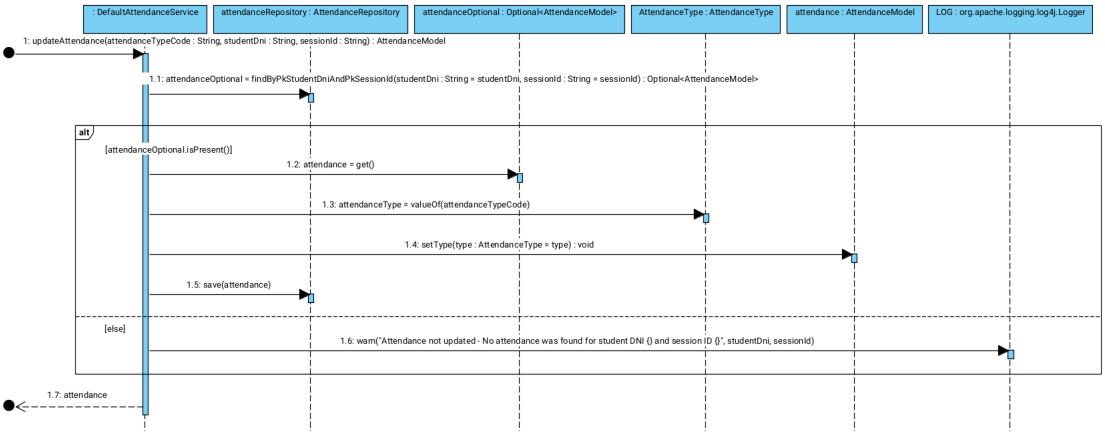

Figura 4.32: $CAR_{2,2}$ - Marcaje de asistencias - Generado con VisualParadigm

4.3.9. $CAR_{2,3}$, $CAR_{2,4}$ - Creación de cursos / grupos

La creación de grupos solo puede llevarse a cabo por un administrador. La operación toma el ID del centro, el año en el que el curso tendrá lugar, el nombre que se le va a dar y, a su vez, el nombre de los grupos que lo conformarán.

Dado que los grupos siempre estarán asociados a un curso, no tiene sentido crearlos de forma independiente. Por tanto, dentro de Dagaz, los grupos siempre se generan asociados a un curso.

Como característica singular, un administrador no podrá crear un curso ni sus grupos si no administra el centro indicado. La intención de esta comprobación es añadir un nivel extra de seguridad para impedir que usuarios malintencionados puedan afectar a las jerarquías de otros centros.

De tal modo, la operación para construir un curso y sus grupos es la siguiente:

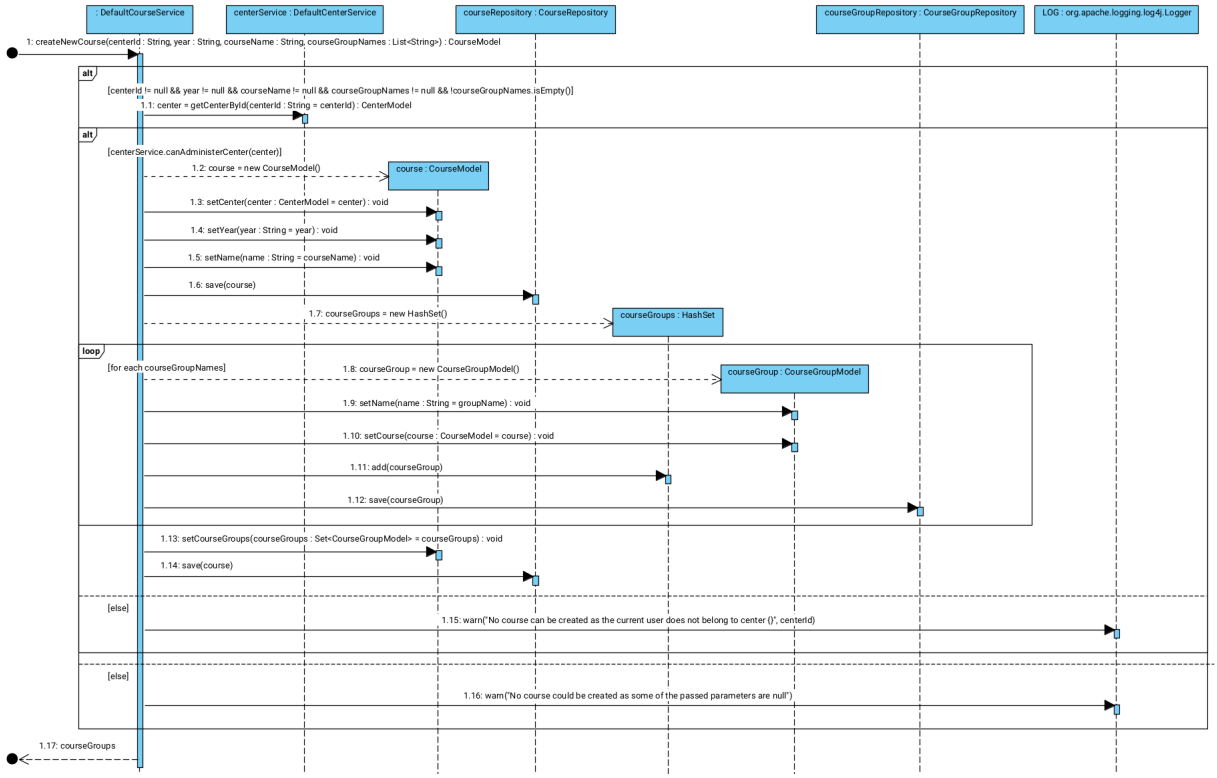

Figura 4.33: $CAR_{2,3}$, $CAR_{2,4}$ - Creación de cursos y grupos - Generado con VisualParadigm

4.3.10. $CAR_{2,5}$ - Creación de asignaturas

Una asignatura es uno de los ejemplos más complejos de objetos dentro de Dagaz. Una asignatura tiene trimestres, sesiones asociadas, un curso y un profesor tutor. A su vez, las sesiones forman parte de un grupo y tienen a un docente que las imparte.

En Dagaz, la creación de una asignatura es, sin lugar a dudas, la operación más complicada y compleja que se ha desarrollado. Esta dificultad se debe a la imperiosa necesidad de hacer más útil la página de creación de asignaturas, pues la creación de una sola asignatura es en realidad la conjunción de muchos métodos complejos:

1. Se recuperan los datos desde el *frontend*, se validan y se tratan
2. Se transforman en DTOs (`SubjectCreationDTO`, `SessionRowDTO`, `TrimesterRowDTO`) para facilitar el tratamiendo que se le da a estos datos
3. Se crean las sesiones asociadas a la asignatura (Que se podrá analizar en la sección *$CAR_{2,6}$ - Creación de sesiones*)
4. Se crean los trimestres asociados a la asignatura
5. Se crea, finalmente, la asignatura

Dada su singularidad, este caso será estudiado más a fondo incorporando la recepción de datos desde el *frontend*, su transformación y la creación de sus objetos asociados.

Recepción de los datos desde el *frontend*

Los datos del *frontend* se reciben a través de un objeto formulario `SubjectForm`. Estos datos se encuentran validados por medio de Javax y el sistema de validación incorporado en los `BindingResult` de Spring. Si el formulario es válido,

entonces se procede a llamar a la fachada `SubjectFacade` y, si todo funciona correctamente, devolver la asignatura recién creada. Si la asignatura se genera adecuadamente, el usuario es redirigido a la página de administración con un mensaje indicando que la asignatura se ha credao con éxito. En caso contrario se le devuelve a la página de creación de asignaturas y se le notifica la incidencia.

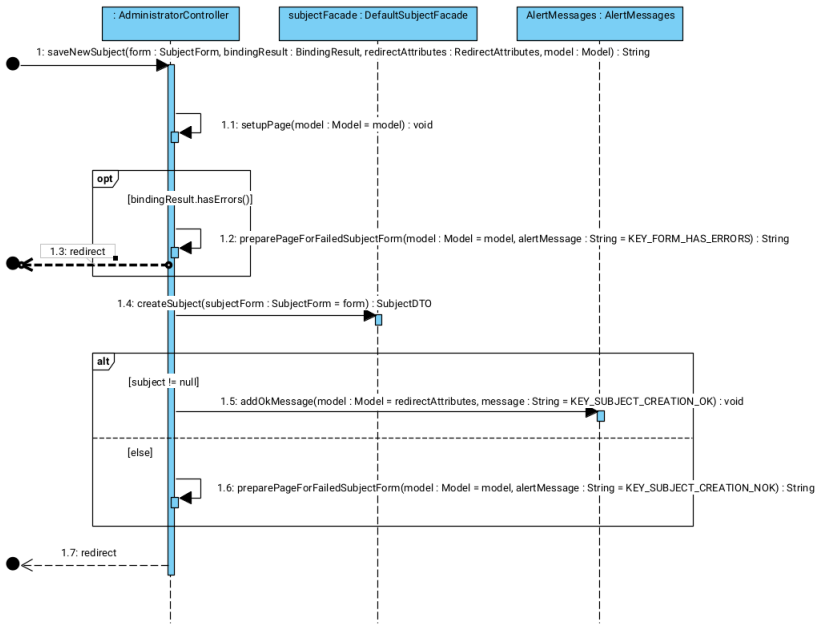

Figura 4.34: $CAR_{2,5}$ - Recepción del formulario de creación de asignaturas - Generado con VisualParadigm

Conversión del formulario de creación de asignatura

Posteriormente, en la fachada, se procede a convertir los datos del formulario en DTOs que puedan ser utilizables por la aplicación. Para ello se utiliza un populador especial, `SubjectCreationPopulator`, para realizar la transformación pertinente. Acto seguido se le pasa el DTO al servicio de `SubjectService` para que genere la nueva asignatura, y cuando éste la devuelve, se transforma de `SubjectModel` a `SubjectDTO`.

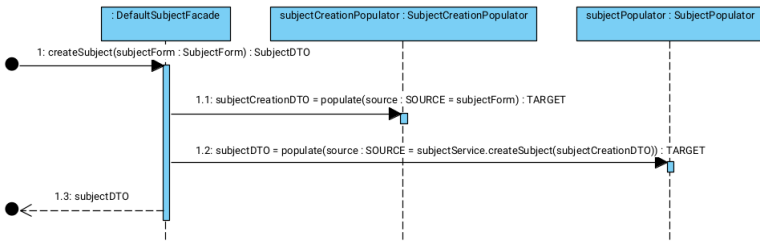

Figura 4.35: $CAR_{2,5}$ - Fachada *SubjectFacade* - Generado con VisualParadigm

Acto seguido, desde el populador se obtienen los distintos campos necesarios para crear el DTO de `SubjectCreationDTO`, entre ellos tanto los `TrimesterRowDTO` que representan los trimestres que se guardarán como los `SessionRowDTO` que corresponde a las sesiones de la asignatura.

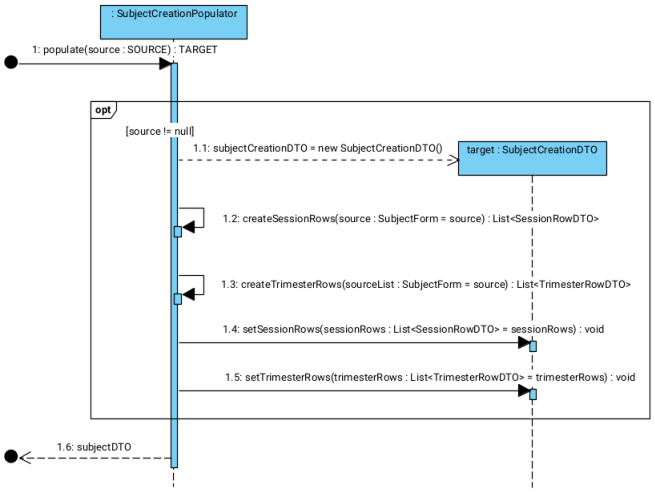

Figura 4.36: $CAR_{2,5}$ - *Populator* para *SubjectCreationDTO* - Generado con VisualParadigm

El método `createSessionRows` hace lo siguiente:

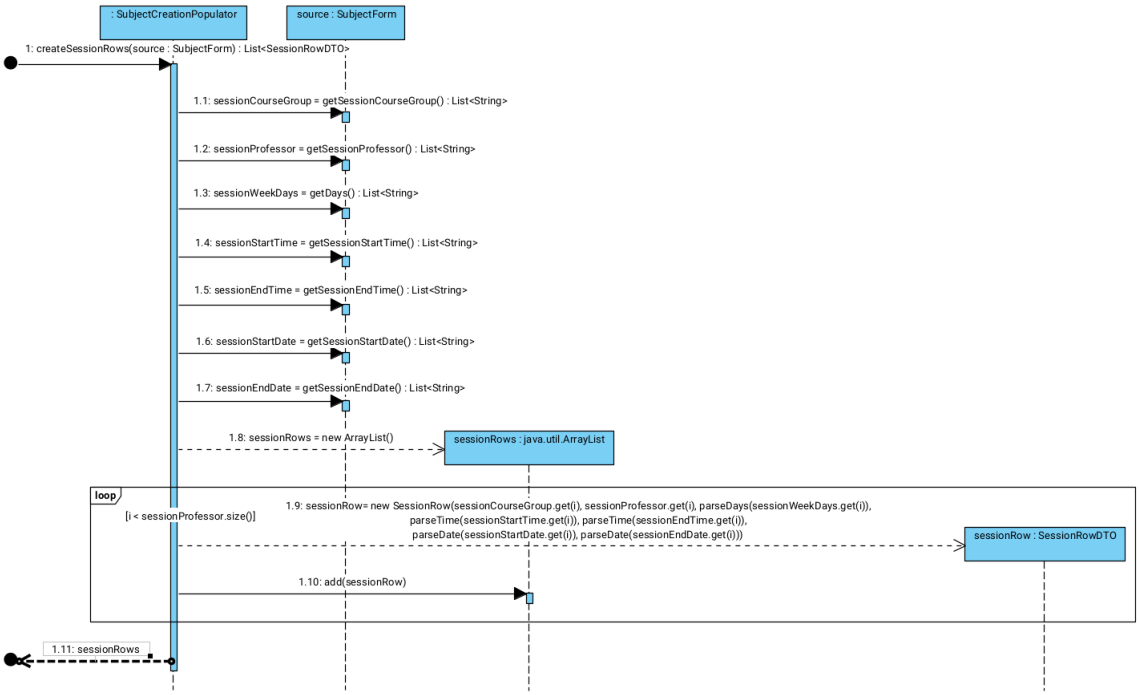

Figura 4.37: $CAR_{2,5}$ - Método *createSessionRows* - Generado con VisualParadigm

Por otra parte, `createTrimesterRows` desempeña las siguientes funciones:

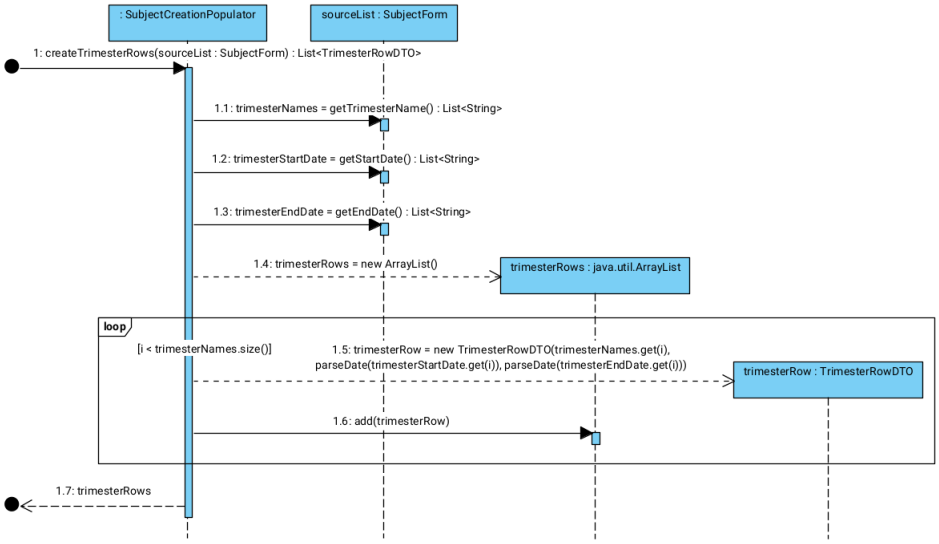

Figura 4.38: $CAR_{2,5}$ - Método *createTrimesterRows* - Generado con VisualParadigm

Creación de la asignatura en el servicio

El `SubjectDTO` contiene los datos validados y convertidos para trabajar con ellos con el menor esfuerzo posible. A partir de este punto se tendrá que:

1. Crear el objeto asignatura e incorporar parte de sus datos
2. Crear las sesiones de la asignatura (cubierto en la sección *$CAR_{2,6}$ - Creación de sesiones*)
3. Crear los trimestres para la asignatura
4. Asociar trimestres y asignaturas y guardar la asignatura en la base de datos

Primeramente, la operación completa tiene la siguiente forma:

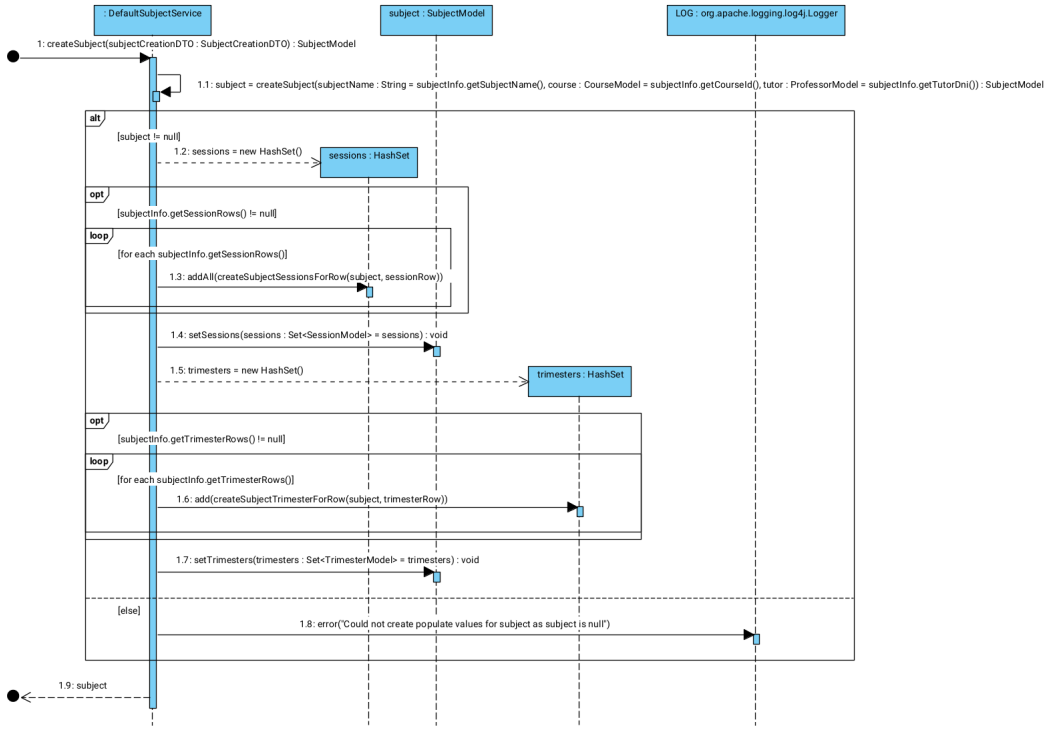

Figura 4.39: $CAR_{2,5}$ - Método *createSubject* para *SubjectCreationDTO* - Generado con VisualParadigm

Nótese la llamada a `createSubject` con una gran cantidad de valores. Así se creará la asignatura que necesitaremos y se realizarán las primeras asociaciones (curso, tutor, etc.)

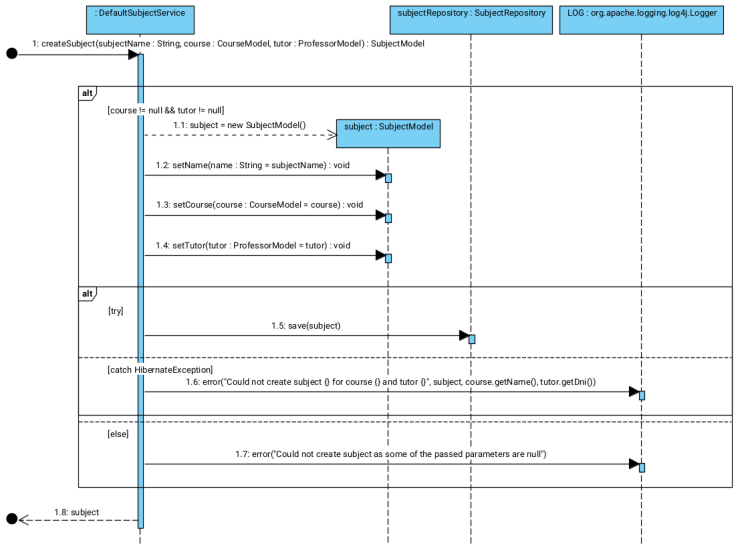

Figura 4.40: $CAR_{2,5}$ - Método *createSubject* para crear asignaturas - Generado con VisualParadigm

Puesto que la creación de sesiones se tratará próximamente, es preferible omitirla aquí. Así pues, solo quedaría describir el proceso para generar los trimestres, `createSubjectTrimesterForRow`:

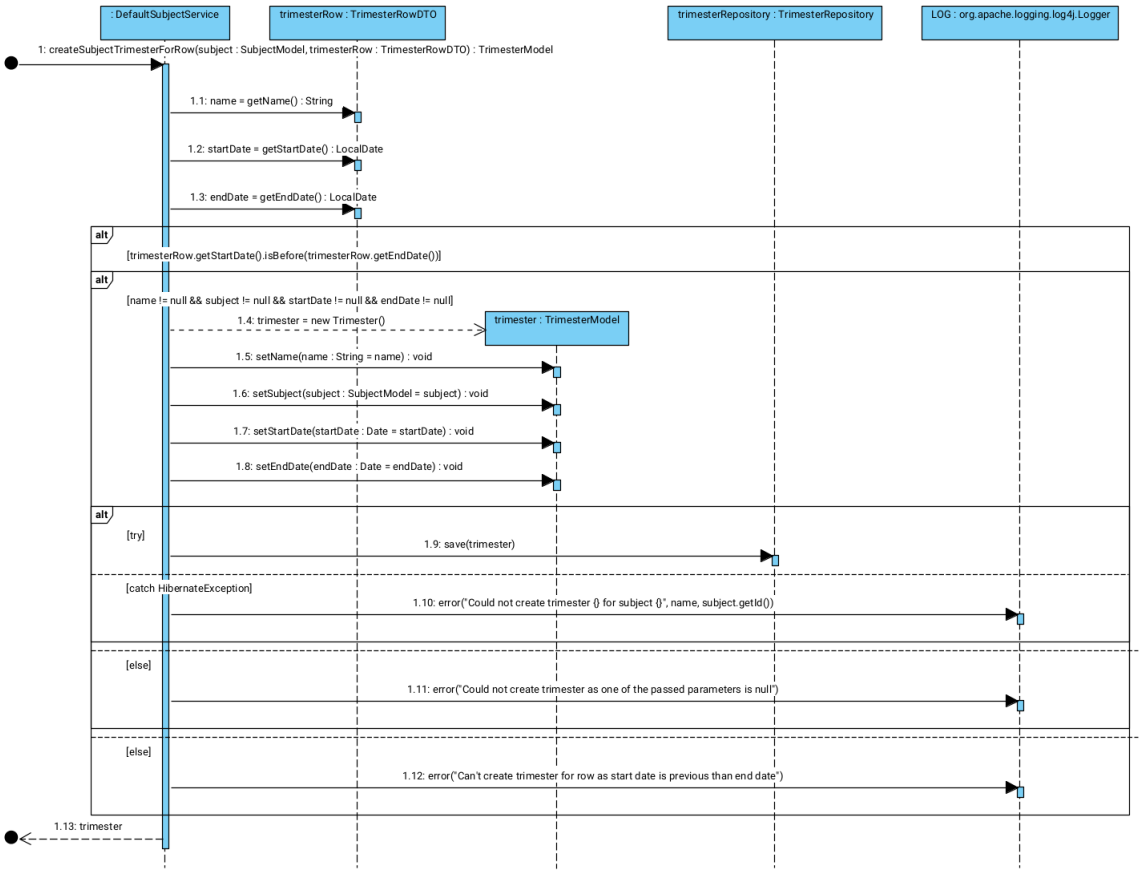

Figura 4.41: $CAR_{2,5}$ - Método *createSubjectTrimesterForRow* - Generado con VisualParadigm

4.3.11. $CAR_{2,6}$ - Creación de sesiones

Como antes se mencionaba, las sesiones se generan siempre como parte de una asignatura. Así pues, utilizando lo expuesto en el punto *$CAR_{2,5}$ - Creación de asignaturas*, analizaremos el método `createSubjectSessionsForRow`.

El funcionamiento es el siguiente:

- Se indican los días de la semana en los que se tendrá cada sesión
- Se indica una hora de inicio y fin para las sesión que se tendrá
- Se indican una fecha de inicio y final que especificarán el rango de tiempo en el cual se generarán sesiones de forma periódica

Así pues, si se indica que se tendrán clases cada lunes de 9 a 10 de la mañana entre los días 7 y 21, el programa analizará cada día entre la fecha de inicio y, si coincide con uno de los días de la semana especificados, generará la entrada correspondiente, generando 2 sesiones correspondientes a los 2 lunes comprendidos en ese rango de fechas.

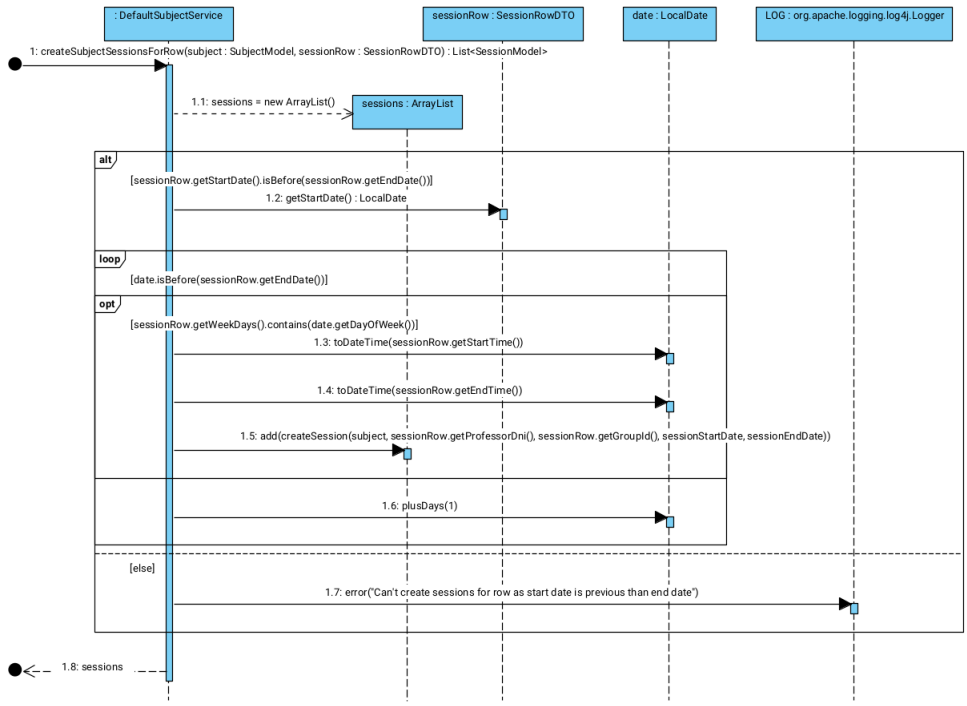

Figura 4.42: $CAR_{2,6}$ - Método *createSubjectSessionsForRow* - Generado con VisualParadigm

Por último, el método `createSession` genera la sesión con los datos necesarios.

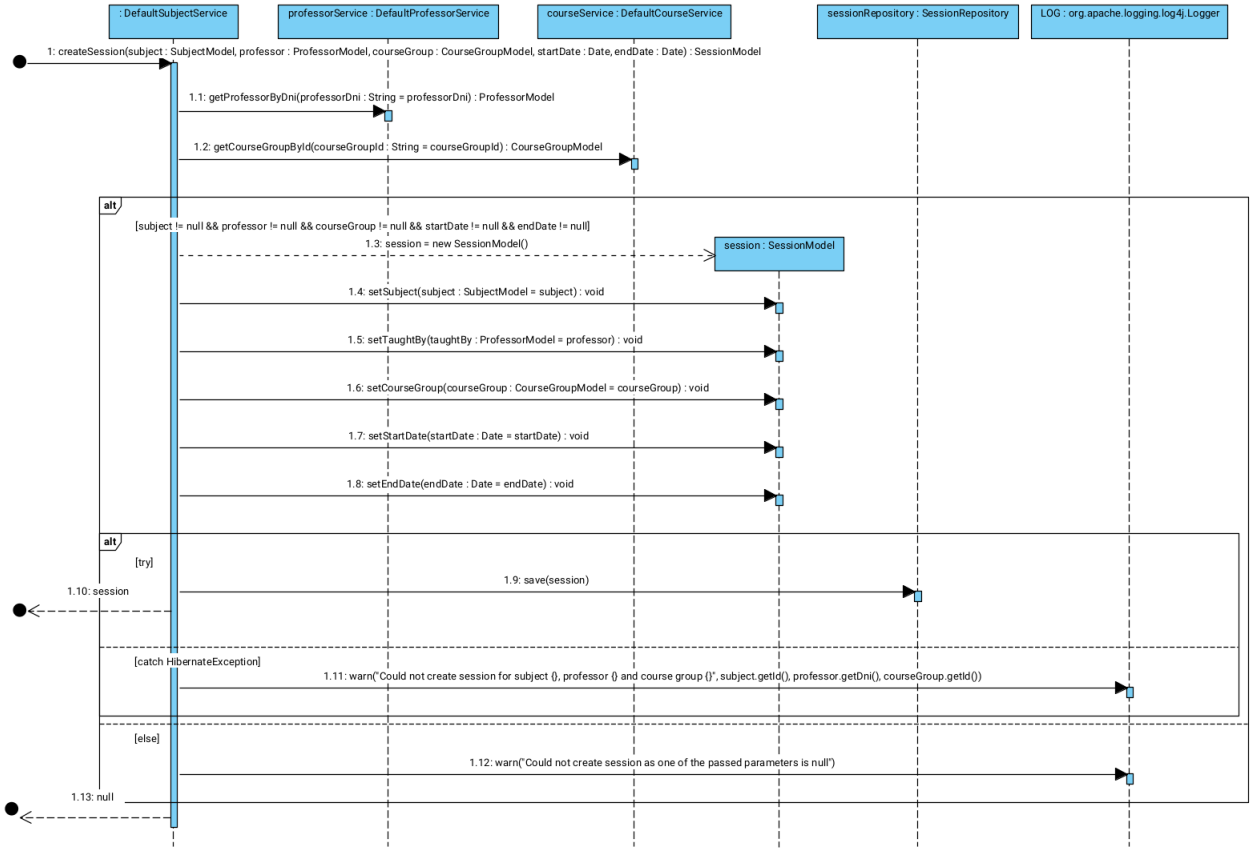

Figura 4.43: $CAR_{2,6}$ - Método *createSession* - Generado con VisualParadigm

4.3.12. $CAR_{2,7}$ - Creación de pruebas

La creación de pruebas es una tarea correspondiente a los profesores. Para generar una será necesario indicar su nombre, la fecha y hora en la que dará comienzo, la asignatura a la que corresponde y el trimestre al que pertenece.

Cuando se genera una prueba también se generan las entradas correspondientes a las notas de cada uno de los alumnos que pertenecen a una asignatura, es decir, están matriculados en un grupo de dicha asignatura. Cuando esto se haga, las notas no tendrán marcado ningún evaluador y su puntuación por defecto será de 0. Así pues, el método encargado de todo esto es `createNewAssessment`.

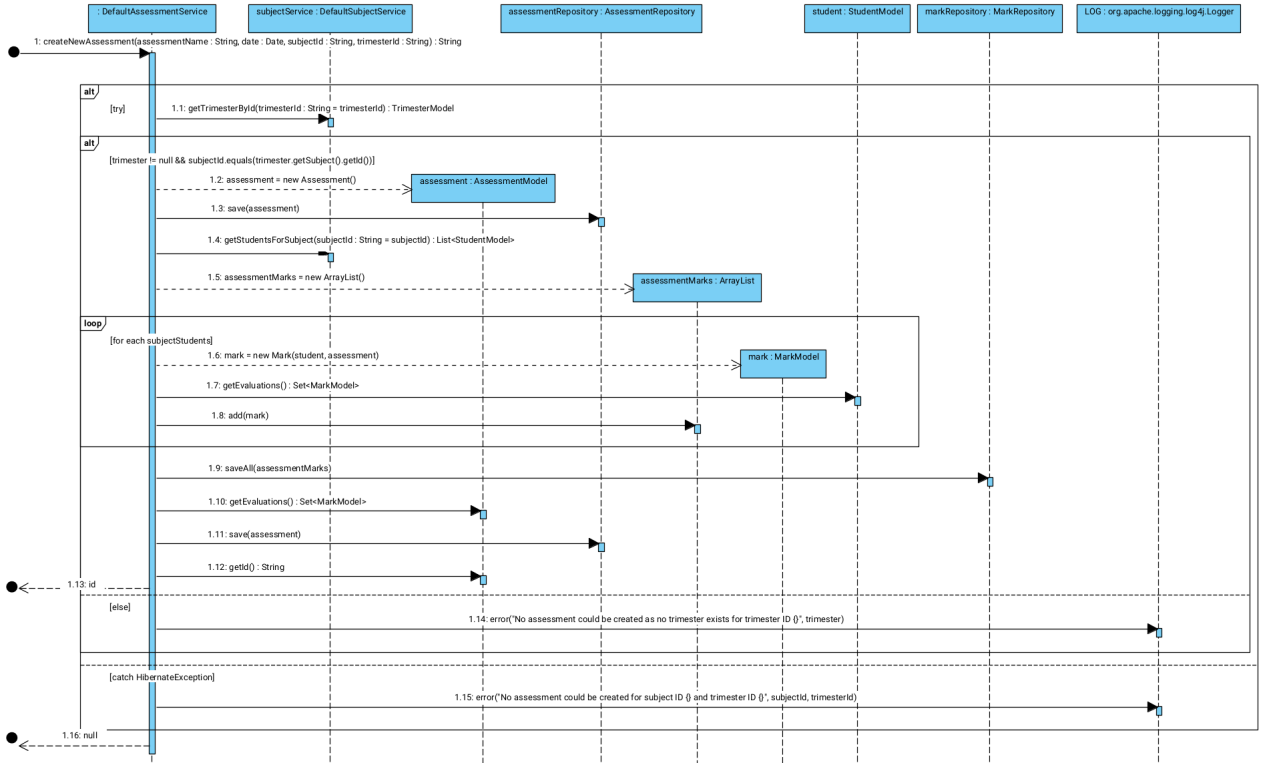

Figura 4.44: $CAR_{2,7}$ - Creación de pruebas - Generado con VisualParadigm

4.3.13. $CAR_{4,1}$ - Marcaje de notas

El marcaje de notas es otra operación llevada a cabo por profesores. Para poner una nota a un alumno solo será necesario contar con el ID de la prueba donde se evalúa, el DNI del alumno y, por supuesto, la nota que se le va a poner. Si la actualizacón se ha llevado a cabo con éxito, el programa devuelve `false`; en el caso contrario se retorna `true`. El método está en `updateMark`.

Huelga mencionar que existe otro método, `updateMarks`, que permite la actualización simultánea de múltiples notas y permite devolver cierta información al *frontend* en forma de JSON. Dicha función hace uso de `updateMark`, pero dado que no estamos abarcando los detalles de la interfaz en este caso, se excluye de este análisis.

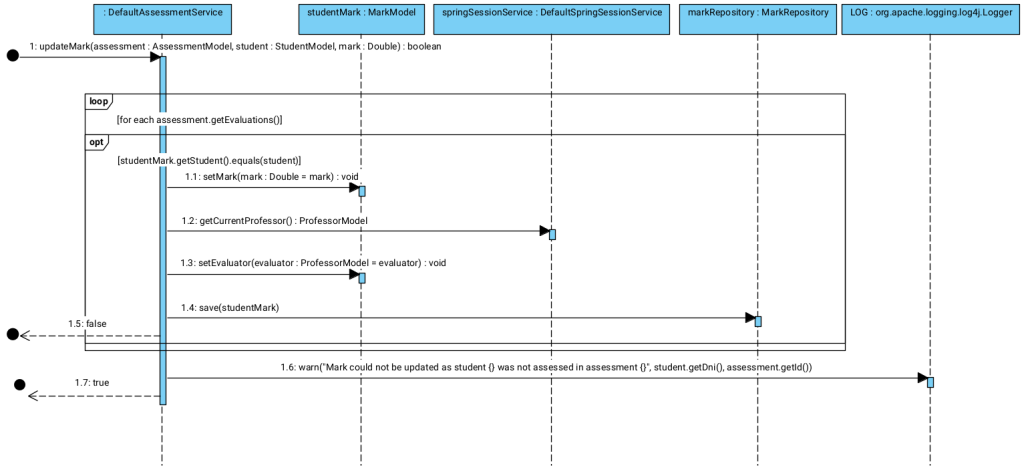

Figura 4.45: $CAR_{4,1}$ - Marcaje de notas - Generado con VisualParadigm

4.3.14. $CAR_{5,2}$ - Documentos de corrección de pruebas

Las operaciones asociadas a documentos de corrección de pruebas pueden ser de dos tipos: De subida y de bajada. Obviamente, para que se pueda descargar el archivo será necesario que exista previamente.

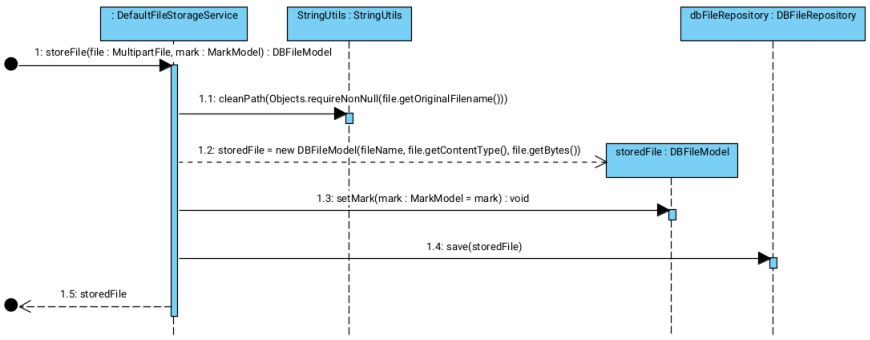

Figura 4.46: $CAR_{5,2}$ - Subida de documentos de corrección de pruebas - Generado con VisualParadigm

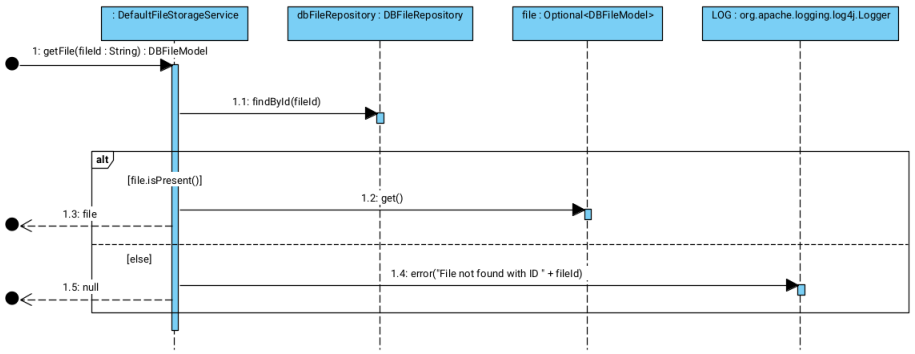

Figura 4.47: $CAR_{5,2}$ - Descarga de documentos de corrección de pruebas - Generado con VisualParadigm

CAPÍTULO 5
Desarrollo de la solución

5.1 Planificación y organización

Su inicio se programó para Marzo de 2020, pero la irrupción del COVID-19 afectó a la toma de requisitos y a la disponibilidad del equipo y los entrevistados, provocando que su inicio fuera retrasado a Abril del mismo año.

El impacto del COVID-19 en la población educativa implicó un cambio de planes y el descarte de buena parte de este planteamiento inicial. A causa del confinamiento que tuvo lugar en España desde principios de Marzo hasta finales de Junio de 2020, la realización de pruebas presenciales no era posible, y las pruebas a distancia se vieron restringidas en gran medida por las limitaciones técnicas de las redes disponibles para el equipo de desarrollo durante los primeros meses de la cuarentena.

A propósito de dotar de más solidez frente a los cambios y postergar al máximo las pruebas de aceptación, se optó por fragmentar el desarrollo en 2 Sprints de 1 mes, dando comienzo el 1 de Abril de 2020 y clausurando el desarrollo el 1 de Junio de 2020. Ante la imposibilidad de realizar las pruebas de aceptación presencialmente, éstas serían retrasadas a fechas en las cuales sí fuera posible reunirse con miembros del equipo educativo. También, considerando las medidas de salud y seguridad, el grupo de *testers* también se reduciría a fin de limitar una posible expansión del virus.

5.1.1. Sprint 1

El Sprint 1 ocurrió entre el 1 de Abril y el 1 de Mayo del 2020. Su finalidad era la creación de la base del sistema, la adición del sistema de localizaciones y la perspectiva del usuario **Profesor**. Específicamente, incluyó los siguientes elementos:

- ***Setup* técnico**
  - Desarrollo del esquema preliminar de la base de datos
  - *Setup* de Spring Boot
  - Preparación de la arquitectura del *backend* de la aplicación (BBDD, estructura de clases, etc.)
  - Preparación de la arquitectura del *frontend* (fragmentos, páginas, etc.)
- **Introducción de las primeras páginas**
  - Página de login
  - Página de calendario
  - Listado de asistencias

  - Vista de notas
- **Funcionalidades para profesores**
  - Login para profesores
  - Marcar/ver asistencias
  - Calendario con sesiones y pruebas
  - Introducción de soporte para localización

Durante la retrospectiva producida al final del Sprint se determinaron varios aspectos a positivos, detalles a mejorar y nuevos enfoques potencialmente beneficiosos de cara a la siguiente fase de desarrollo:

Aspectos positivos

Se valoró gratamente el esfuerzo llevado a cabo para preparar la mayor parte de la infraestructura en un tiempo récord. El análisis preliminar de las herramientas a utilizar fue fructífero y permitió dar con la manera correcta de sentar las bases de lo que luego terminaría siendo la arquitectura tanto funcional como de la base de datos.

Nuevos enfoques

Este primer sprint procuraba usar JavaScript lo mínimo posible en las interfaces de la página, lo cual implicó que algunas de las vistas fueran poco intuitivas y criticadas en parte por los usuarios. Para poder solventarlo sería necesario tomar conciencia sobre las utilidades que herramientas como JQuery podían aportar para realizar actualizaciones en el DOM.

Asimismo, el código hasta el momento se había construido utilizando los mecanismos de autoresolución de dependencias de Spring, de tal forma que la implementación de una interfaz utilizada en una clase era resuelta por Spring en lugar de indicarse explícitamente. Esto podía afectar a su modularidad, así que sería necesario introducir una historia asociada a esta deuda técnica.

Por último, dado que hasta el momento se consideraba una única perspectiva, no había soporte para distinguir entre los usuarios que podrían acceder a la página, así que sería necesario incorporar una forma de restringir su acceso y permitir que pudieran hacer *login* en la herramienta.

5.1.2. Sprint 2

El Sprint 2 tuvo lugar entre el 1 de Mayo y el 1 de Junio del 2020. A grandes rasgos, su intención era entregar más calidad que contenido, mejorando las interfaces ya existentes e incluyendo la perspectiva del **Administrador**.

El motivo por el que se optó por dar prioridad a la perspectiva del administrador frente al desarrollo de la perspectiva de los alumnos era simple: Los administradores serían quienes aportarían contenido a la página y permitirían agrupar a los alumnos, así que sería necesario que éstos existieran antes para poder simular mejor el funcionamiento de la página.

Entre sus historias se encuentran:

- **Introducción de soporte para múltiples tipos de usuarios**
  - Sistema de roles (*Authorities*) para distintos tipos de usuarios
  - Restricción de acceso a recursos según el rol
  - *Login* para profesores, alumnos y administradores

- **Configuración del *backend***
  - Introducción de configuración de dependencias de Spring por XML
  - Refactorización del diseño de los populadores
- **Extensión de la página de notas**
  - Marcaje de notas
  - Restricción y sanitización del input de notas
  - Documentos adjuntos a notas
- **Introducción de nuevas páginas**
  - Página de creación de curso y grupos
  - Página de creación de asignaturas, sesiones y trimestres
- **Mejoras en las interfaces existentes**
  - Actualización de la página de notas
  - Actualización de la página de creación de pruebas
  - Adición de nuevas configuraciones para permitir que todas las páginas existentes tengan un comportamiento *responsive*
- **Funcionalidades para administradores**
  - Crear curso
  - Crear grupos asociados un cursos
  - Crear asignatura
  - Crear trimestres asociados a una asignatura
  - Crear múltiples sesiones asociadas a una asignatura

Aunque no se contemplaba inicialmente, fue en este Sprint cuando se determinó que la base de datos que se había utilizado hasta ahora en el desarrollo, MariaDB, tenía una licencia incompatible con aquella que se pretendía utilizar en el proyecto. Por ello, se tuvo que llevar a cabo la instalación de una base de datos con una licencia válida, así como reconfigurar Dagaz para dejar de utilizar MariaDB y empezar a usar su sustituta. Se optó por utilizar PostgreSQL.

En su correspondiente retrospectiva se llevó a cabo el siguiente análisis:

Aspectos positivos

El cambio de base de datos fue un evento que se contempló como un riesgo durante la concepción del proyecto. Dado que ya se había previsto que tal situación podría manifestarse, existían guías sobre qué hacer para paliar sus efectos. Sin embargo, su impacto no fue tan grave como inicialmente se esperaba.

Gracias a la utilización de Hibernate, configurar Dagaz para que empleara una nueva base de datos fue una cuestión sencilla, únicamente fue necesario cambiar las direcciones que apuntaban a la base de datos y la generación de tablas correría a cargo del ORM. La estructura de la base de datos autogenerada por Hibernate era idéntica a la de MariaDB, así que no fue necesario cambiar ningún aspecto más del código. La introducción de Hibernate se asume ahora como una de las mejores decisiones de todo el proyecto.

En este Sprint también se empezó a tolerar el uso de JavaScript de una forma más extendida, pero manteniéndolo siempre en unos márgenes que no afectaran al rendimiento del servidor o de las máquinas que visualizarán la página. Al hacer que ciertas vistas incorporaran actualizaciones del DOM, algunos elementos del *frontend* pudieron tratarse de forma distinta y abarcarse como información que podría entregarse como si de una API se tratara.

De cara al usuario, las páginas ahora serían más intuitivas y permitirían tener información más relevante sobre los elementos con los que tratan, pero mejorando su experiencia de uso. Esto tuvo su recompensa en las pruebas de aceptación con usuarios, que valoraron gratamente el refresco de la información en tiempo real, así como la mejorada responsividad de la página.
Echando la vista atrás, considerando las limtiaciones encontradas, el esfuerzo dedicado, el resultado obtenido...; el equipo valora positivamente el trabajo realizado.

Nuevos enfoques

Siendo este el último Sprint, se procuró zanjar la mayoría de apartados abiertos para que Dagaz pudiera empezar su desarrollo de cara al público, pero también deja algunas cuestiones en el tintero.
Añadir soporte para configuración por XML permitía ahora que otros desarrolladores pudieran cambiar las clases que se utilizaban y hacer uso de sus propias implementaciones en lugar de emplear las que el programa trae por defecto. Sin embargo, pueden quedar algunos resquicios de deuda técnica, tales como *beans* específicos definidos en código Java en lugar de a través del XML (tales como el bean asociado a la encriptación de contraseñas o de la localización de la página) que deberán ser atendidos en un futuro próximo.
Además, quedó pendiente una última función para los administradores: la matriculación de alumnos a un curso concreto. De haber desarrollado esta última vista y su funcionalidad, la perspectiva del administrador podría considerarse como terminada. Esto se abarcaría en un posible 3er Sprint.

5.1.3. Consideraciones para posteriores desarrollos

Dagaz no es una herramienta terminada, pero ha conseguido satisfacer un MVP en un tiempo considerable y bajo unas circunstancias singulares causadas por la irrupción del COVID-19. De su proceso de desarrollo se pueden extraer múltiples lecciones de acciones que se habrían podido tomar y que no fueron posibles por la situación en la que dio comienzo su construcción.

- **El grupo de usuarios de pruebas deberá ser mayor**
  Para futuras versiones, contar con un mayor grupo de usuarios sería vital y muy relevante. Contar con una población más variada, con experiencia en otros CMS y que correspondan a sectores educativos distintos a los de los entrevistados podría aportar nuevos puntos de vista que ahora podrían no estar siendo considerados.
- **Será necesario publicar la herramienta al mundo**
  La licencia empleada, AGPLv3, exige la publicación del código fuente de la aplicación. La herramienta aún se está construyendo y no está siendo empleada por ninguna persona, pero en un futuro será necesario darla a conocer y hacerla accesible al mundo.
- **Deberá apelar más a los desarrolladores voluntarios**
  Dagaz cuenta con documentación, tests y aspectos configurables para que modificarla sea un proceso mucho más sencillo. A pesar de ello, nunca se podrá aprovechar si no se le da un buen uso y se encuentra a gente dispuesta a colaborar en el proyecto. Será menester buscar a programadores a través de internet, publicitar el proyecto en redes sociales y tratar de darle cierto eco a lo que se pretende construir para lograr que más personas accedan a trabajar en Dagaz.

5.2 Tecnologías libres

Cuando hablamos de tecnologías libres nos referimos a proyectos o apliacciones categorizadas como **software libre**[33]. A su vez, el Software Libre se define de acuerdo a las llamadas **Cuatro libertades**:

- Libertad 0, para ejecutar el código con cualquier fin
- Libertad 1, para estudiar el funcionamiento del programa y modificarlo para cumplir un objetivo deseado
- Libertad 2, para redistribuir copias del programa y compartirlas con la comunidad
- Libertad 3, para cambiar el programa y redistribuir sus versiones modificadas

El Software Libre no entran en conflicto con aspectos económicos, solo referencian cuán accesible es el código que forma la aplicación y cómo puede redistribuirse.

Dagaz, siendo una aplicación diseñada por y para la comunidad educativa, se diseñó desde cero con la idea de ser libre de ser usada, estudiada, alterada y compartida con cualquier fin. Haciendo el software más accesible y publicándolo libre de coste se permite que las restricciones económicas, si no desaparecen completamente por la propia naturaleza de la aplicación (*hosting*, conexión a internet, etc.), sí las mitiga o reduce. Además, usar licencias libres permite que personas interesadas por la causa de la herramientan puedan participar en ella libremente, otorgándole mayores funciones y dando lugar a un clima de colaboración y cooperación. Por este motivo, Dagaz se encuentra licenciada bajo **AGPLv3**[34], caracterizada por ser **FOSS** y ***Copyleft***.

Esta no es una decisión arbitraria, pues el hecho de tratarse de una licencia *Copyleft* implica que todo proyecto que parta de Dagaz deberá distribuirse bajo la misma licencia o aquellas que sean compatibles, lo que impide:

- El uso privativo del código, es decir, código no publicado o que no cumpla las 4 libertades .
- Que todos los trabajos derivativos serán públicos y, por tanto, su código podrá ser usado libremente tanto en Dagaz como en cualquier otro que parta de él.

AGPLv3 permite la comercialización del proyecto o sus derivados, pero dado que su código y el de sus variantes siempre será público, y por tanto de acceso gratuito, se espera que los usuarios opten por descargarlo, compilarlo o modificarlo gratuitamente. Evitando que el personal educativo pueda verse limitado o restringido económicamente se fomenta uno de los principios de la aplicación: contribuir a la educación erradicando las barreras económicas que pudiera haber, haciendo libre el acceso y uso de Dagaz. De igual manera también se previene la compraventa de este software a terceros.

Nacida a hombros de gigantes, Dagaz emplea las siguientes adiciones y *frameworks*.

Spring Framework y Spring Boot

Spring Framework[38] es un *framework* para construir *webapps* diseñado para Java. Fue diseñado por Spring Foundation y hace uso de la licencia Apache License 2.0[39].

Se trata del elemento más complejo y versátil de todo el proyecto. Spring hace gala de la **injección de dependencias**, un mecanismo extensamente empleado en la solución propuesta, según el cual es posible que una clase defina cuáles serían los servicios de los que haría uso, facilitando el desacoplamiento del código. Si se emplea correctamente, esta inversión de dependencias también simplifica la modificación y extensión del programa, motivo que todavía lo hace más relevante para el fin que buscamos. Con su uso logramos dar un paso adelante en el seguimiento en todos los frentes de los principios SOLID.

Spring también cuenta con mecanismos de validación de *input*, internacionalización, extensiones específicas para uso en tests, compatibilidad con ORMs y repositorios y Spring MVC, que otorga una funciones y estructuras con las que construir aplicaciones web siguiendo el patrón de Modelo-Vista-Controlador.

Spring Boot, por otro lado, es una opción que contiene buena parte de las funciones de Spring Framework y las simplifica en gran medida. Siendo compatible con todas las herramientas que el *framework* ofrece por defecto, aporta configuración automática para estos servicios y permite abstraer al programador, obviando detalles como la configuración del servidor sobre el que se ejecuta, o la configuración de ejecución. Por supuesto, sigue siendo posible sobreescribir este *preset* de propiedades, dejando abierta la posibilidad de alterar estos valores y configurar al detalle toda la infraestructura para casos más específicos o complejos.

Hibernate ORM

Hibernate es una *suite* de herramientas para el tratamiento y almacenamiento de datos. En este proyecto se ha hecho uso de su ORM (*object relational mapping*) para Java[40]. Fue diseñado por Red Hat y emplea la licencia LGPL[41]. Hibernate se convierte en una pieza clave para cumplir tres objetivos en este proyecto:

- El código debe ser independiente de la base de datos empleada
- Las entidades que formarán parte de la aplicación deben definirse de forma programática y sencilla a través de Java
- El almacenamiento y recuperación de información debe ser sencillo e independiente de la base de datos empleada

Hibernate dispone de "dialectos", que permiten traducir sus comandos a los que emplearía una BBDD concreta. Gracias a ello, cualquier potencial implementador de Dagaz es libre de usar aquella que se adecue a sus necesidades, solo tiene que indicar el dialecto correspondiente y podrá aprovechar todos los beneficios que aporta este ORM. Lograr esta flexibilidad se encontraba entre los objetivos iniciales del proyecto, incluso en su fase de prototipado.

Mediante las anotaciones que proporciona, es posible construir POJOs, objetos de Java carentes de funciones, y especificar de forma programática cuál será la estructura de las tablas en las que se almacenará esta información en una BBDD. Hibernate se encarga de construir, adaptar y formar las relaciones entre estas tablas, contribuyendo a trabajar a niveles de abstracción más altos y con mayor sencillez.

Hibernate es compatible con los repositorios JPA, mecanismos que hacen transparente la extracción y guardado de datos y que eliminan la necesidad de lidiar con los detalles particulares del almacenamiento de datos al que se conecta.

También, para poder realizar consultas a la BBDD, se puede hacer uso de la Hibernate Query Language, un lenguaje inspirado en SQL que funciona sobre

entidades (a diferencia de SQL, que está orientado a tablas) y que es traducido específicamente para la BBDD que se emplee. De esta forma, una consulta diseñada en HQL sería tan válida en MariaDB como en SQLite, abriendo posibilidades para configuraciones acordes a las necesidades de cada usuario.

Por último, Hibernate también permite formular consultas HQL en los repositorios JPA mediante lenguaje natural, convirtiendo un trabajo que en ocasiones podría ser tedioso en una experiencia mucho más amena y fácil de trabajar.

PostgreSQL

Durante el desarrollo se optó por hacer uso de PostgreSQL[43], una base de datos relacional de uso gratuito. Se trata de la pieza más antigua de este proyecto, pues el desarrollo de esta BBDD comenzó hace aproximadamente 30 años. Se trata de un trabajo licenciado bajo PostgreSQL License [44]. Por otra parte, el conector que permite que Dagaz se comunique con ella utiliza una BSD de 2 cláusulas[45]. Ambas son compatibles con GPL.

Gracias a la introducción de Hibernate, cualquier base de datos compatible podría ser usada para ejecutar Dagaz y todo usuario es libre de emplear cualquier base de datos y redistribuir la configuración de la misma.

Si se deseara sustituir PostgreSQL por otras alternativas, podrían producirse problemas legales si el conector o la base de datos no fueran compatibles con GPL. Puede leer más en el apartado Licencias y bases de datos incompatibles.

Otras opciones barajadas durante las deciciones arquitecturales incluyen:

Base de datos	Licencia	Página oficial
MariaDB	GPLv2[42] incompatible con GPLv3	[48]
MySQL	Propietaria + GPLv2 + BSD [49]	[50]
HSQLDB	HSQL License, basada en BSD [51]	[52]

Bootstrap

Bootstrap es *framework* que permite construir páginas web simplificando parte del proceso de diseño y aportando cohesión al resultado. A través de su sintáxis es posible definir el comportamiento y disposición de los elementos de la página mediante HTML. Cuenta con un sistema de separación del contenido por columnas, concebido para la producción de páginas *responsive*, esto es, vistas que se adaptan al tamaño de la pantalla del dispositivo que las visualiza. Emplea la licencia MIT[53].

El uso de Bootstrap viene motivado por dos razones: La carencia de experiencia en el desarrollo *frontend* y el potencial que aporta para construir un diseño moderna e intuitiva con facilidad.

Actualmente, el equipo de desarrollo de Dagaz no incluye *frontends*, por lo que aprovechar este *framework* fue crucial para alcanzar las fechas establecidas y lograr una interfaz usable y agradable.

Además, dado que Bootstrap es empleado en proyectos en todo el planeta, existe un gran mercado de *frontends* con experiencia en su uso, por lo que hay disponibilidad de mano de obra que podría adaptar y diseñar nuevas interfaces para Dagaz.

Librerías del *frontend*: JQuery, MomentJS, FullCalendar, Tempus Dominus

Librería	Licencia	Página oficial
JQuery	MIT[53]	[54]
MomentJS	MIT[53]	[55]
FullCalendar	MIT[53]	[56]
Tempus Dominus	MIT[53]	[57]

Conforme se diseñaba la interfaz, uno de los detalles que se convirtió en evidente era la dificultad que suponía interactuar con ella. Para lograr que su uso fuera intuitivo, debía existir una cierta familiaridad, de tal manera que un nuevo usuario pudiera entender o descubrir por similitud cómo navegar entre las páginas y los datos que se le presentan.

JQuery contribuyó a dar una notable fluidez a la página, refrescar los elementos mostrados sin necesidad de recargar la pestaña del navegador y también alterar el contenido del DOM. Mediante AJAX y un uso inteligente de los recursos a mostar, la navegación pasó a ser más sencilla, rápida y también modificable.

Abusar de estas tecnologías puede tener sus inconvenientes. Si un servidor envía grandes cantidades de archivos para controlar la interfaz, los usuarios con conexiones más lentas podrían verse interrumpidos o sufrir de largos periodos de carga. Por ese motivo se decidió acotar el alcance de los cambios y limitar el uso de Javascript y otras herramientas al mínimo necesario. A menor carga en el lado del cliente, la página se mostrará en menos tiempo, y el servidor también se vería más liberado al no tener que servir tantos datos.

De acuerdo a las necesidades descubiertas, el profesorado requiere de un lugar donde consultar sus sesiones y pruebas, tanto futuras como pasadas. Originalmente, esta información se iba a mostrar a través de páginas estáticas con diseños crudos y sencillos. Afortunadamente, se decidió buscar otros medios más acertados de visualizar los datos. Con FullCalendar y su cómoda integración se pudo cubrir esta frente, descartando los diseños obsoletos e incómodos. Con él, el usuario cuenta con 3 modos distintos de visualización (semanal, mensual o agenda), más intuitivos que la solución original.

FullCalendar emplea MomentJS, una librería de fechas con la que trabajar con tiempo y zonas horarias. Si se emplea adecuadamente, permite adaptar Dagaz visualizar las mismas en distintos husos con un simple click.

También cabe mencionar que la página hace uso de selectores de fechas y horas en los formularios. Para incluir esta función se presentaban dos alternativas:

- Utilizar el selector de fechas por defecto incluido en cada navegador
- Buscar librerías en JavaScript que permitieran recoger campos de fecha y hora, que serían más extensibles y gozarían de otras funciones (fechas límites o lógica de zonas horarias)

Contemplando que ciertos navegadores no ofrecen los mencionados selectores y que todos los usuarios deberían tener la misma visualización de una página sin importar su navegador, se optó por integrar Tempus Dominus, un *date-time picker* también afianzado sobre MomentJS. Aunque se disponía de otras alternativas, ésta era la más completa que se encontró en su momento.

Librerías de *testing*: JUnit, Mockito, AssertJ y Powermock

Librería	Licencia	Página oficial
JUnit	EPL v2.0 [59]	[58]
Mockito	MIT[53]	[60]
AssertJ	Apache 2.0[39]	[61]
Powermock	Apache 2.0[39]	[62]

El TDD especifica que la primera parte del desarrollo de una funcionalidad o cambio son los tests que garantizan que su comportamiento cumple con los criterios de aceptación, relegando la implementación del código a un segundo lugar. Puesto que en este estadio del desarrollo es posible que no se cuente con el código definitivo, es necesario emplear objetos y clases simuladas, llamados *mocks* en inglés.

Un objeto o clase *mock* simula la devolución de datos o la realización de ciertas operaciones sobre los mismos. Con ellos es posible describir qué pasos se cumplirán en el código producido, y servirá como guía en el posterior desarrollo. Especificando el comportamiento y también las circunstancias que podrían darse al trabajar con esta información, podemos forjar una batería de pruebas que cubran buena parte o la totalidad de los caminos que podrían darse durante una ejecución.

JUnit es la *suite* de *testing* unitario por excelencia en Java, permite definir y probar tests y recibir información de sus resultados. Por desgracia, entre sus capacidades no está definir objetos mock, ni tampoco la inyección de dependencias. Por ello se emplea Mockito, que actúa como complemento y sí contempla estos escenarios. A su vez, Mockito también incorpora instrucciones de verificación con las que determinar si se ha interactuado con una clase o método, los valores que se le han pasado e incluso el orden de llamada de los mismos.

JUnit trae consigo asertos con los que comprobar que los valores de los datos resultantes coinciden con los que se desea obtener, pero son limitados y emplearlos en colecciones de objetos resulta tedioso y verboso. Para hacer el proceso más cómodo y aumentar la complejidad de las pruebas se incluyó AssertJ.

Por último, Mockito puede simular de prácticamente cualquier clase, pero es incapaz de tratar con aquellas que son finales o estáticas. Powermock lo complementa y da la posibilidad de crear mocks para estos casos.

Gracias a ello, la *suite* de tests que abarca el paquete *core* de Dagaz cuenta con una cobertura del código del 100 % en todas sus clases, un objetivo remarcable para un proyecto que busca atraer a nuevos desarrolladores.

5.2.1. Licencias y bases de datos incompatibles

De acuerdo a las reglas establecidas por la AGPLv3, la integración de código no compatible con GPL requiere permiso explícito de quienes lo desarrollaron o lo gestionan actualmente[46].

Un área legalmente debatible sería el empleo de una base de datos con licencia incompatible. De acuerdo al reglamento citado en el apartado de Mere Agregation de la GPL [47], aquellos programas que comparten información a través de ciertos medios que sirven como *input/ouput* del sistema podrían ser totalmente válidos y compatibles con su uso en un proyecto bajo GPL. Si se considera que Dagaz y su base de datos son entidades completamente diferentes, encajaría dentro del reglamento.

Otro apartado que se menciona es la complejidad de los mensajes que se transmiten entre ambas partes: Si se trata de estructuras de datos complejas, su uso no sería aceptable. Por otro lado, la comunicación entre ambas partes se hace mediante un lenguaje común, la API de JDBC (*Java Database Connectivity*), intermediado por Hibernate (que sí es compatible con la GPL).

Estas cuestiones supusieron un inconveniente en el desarrollo. Originalmente se planteó utilizar MariaDB, un fork FOSS de MySQL mantenido por MariaDB Foundation, pero su licencia (GPLv2[42] estricta) la volvía incompatible con la GPLv3 del proyecto. Habitualmente, los programas licenciados bajo GPLv2 incluyen una cláusula adicional que permite su uso para posteriores ediciones de la GPL. MariaDB emplea únicamente GPLv2, que es intrínsecamente incompatible con GPLv3.

De haber aceptado el uso de MariaDB, el proyecto habría pasado a emplear GPLv2. Por otro lado, la Apache License 2.0 y otras tantas más usadas en el proyecto son también incompatibles con esta opción.

Para evitar incidentes legales y aprovechando otros proyectos más alineados legalmente, se decidió desechar MariaDB por PostgreSQL, que sí cumple con todas las necesidades, su uso es válido con la licencia actual y evita potenciales quebraderos de cabeza.

Por las razones expuestas, huelga decir que esta cuestión es relativa y podría analizarse desde un punto de vista legislativo. Aprovechando que el propio reglamento de la GPL deja resquicios que permitirían el uso de bases de datos con licencias incompatibles, cualquier usuario podría hacer públicas sus modificaciones de Dagaz incorporando la base de datos de su preferencia, pero podría ser sancionado si un tribunal lo considerara inválido.

Otra potencial alternativa es presentar modificaciones de este proyecto omitiendo tanto el conector como la base de datos que se emplearía en su listado de dependencias. De esta forma se dejaría que el usuario fuera libre de emplear la base de datos que considere oportuna. Así, el proyecto sería libre y compatible con GPL y nos abstraeríamos de las cuestiones de la base de datos con la que se opere.

5.3 Herramientas empleadas durante el desarrollo

Dagaz es una herramienta construida a hombros de gigantes y que jamás se podría haber llevado a cabo sin contar con la tecnología adecuada en el momento correcto. Aunque ya se ha descrito cuáles han sido las librerías y dependencias utilizadas en la propia herramienta, este apartado servirá para indicar cuáles han sido los programas y utilidades de los que se ha hecho uso durante el proceso de desarrollo. Al igual que en la aplicación, se ha procurado hacer uso de software *FOSS*, salvo con algunas excepciones expresamente identificadas.

Linux

Es el entorno de preferencia del equipo de desarrollo que ha construido Dagaz, y ha facilitado mucho la instalación de los distintos elementos que lo conforman (base de datos) y de otros programas relevantes para la redacción de este mismo documento. A propósito, Arch Linux es la distribución minimalista GNU/Linux empleada de entre todas las disponibles, ya que goza de un modelo de actualización *rolling release* (actualizaciones contínuas de sus paquetes), un amplio soporte y documentación en forma de *wiki* con la que configurar y

extender el sistema y sus capacidades. Parte de esa información ha sido necesaria para preparar las bases de datos del sistema.

Mozilla Firefox

Desarrollado por Mozilla Foundation, Firefox ha sido el navegador *FOSS* escogido para la mayor parte de las pruebas que se han llevado a cabo en la página. Sus herramientas de *debugging* de JavaScript, el editor de DOM del que dispone y otras funciones para la revisión de los datos y las vistas provistas por Dagaz han simplificado en gran medida el trabajo a llevar a cabo. No ha sido el único navegador en el que se ha probado la aplicación (también se ha empleado Chromium, proyecto sobre el que se basa Google Chrome y otras alternativas en el mercado), poero sí ha sido el escogido para realizar el análisis y las correcciones pertinentes en la página.

GNU Image Manipulation Program

También conocido como *GIMP* por sus siglas, este editor de imágenes obra de GNU Software Foundation ha servido para modificar las figuras que se adjuntan en este documento. Sus herramientas de tratado de color y edición rápida han permitido una utilización eficiente del tiempo disponible.

Git y GitLab

La herramienta por excelencia para control de versiones ha sido Git, desarrollado por Linus Torvalds. Git es rápido, versátil y ha permitido mantener bajo control la gran cantidad de archivos que existen en el repositorio del proyecto con gran simplicidad. Por redundancia y seguridad, se ha contado con GitLab como repositorio online. Se planea que en un futuro se realice la distribución de Dagaz a través de este medio.

IntelliJ IDEA

En cuanto a desarrollo de código en Java, IntelliJ es el IDE (*Integrated Development Environment*, entorno integrado de desarrollo) favorito del equipo de desarrollo. Su aparente simplicidad, la variedad de opciones y configuraciones de las que dispone y su modificabilidad lo convierten en un programa imprescindible para construir Dagaz. Cuenta con una versión de pago (con licencia propietaria) y otra comunitaria (FOSS), que es la utilizada en este caso.

IntelliJ cuenta con diversos añadidos auxiliares que han permitido trabajar a mayor velocidad y con más confianza, entre ellos su editor de bases de datos, el *plugin* de Spring Framework y el visualizador de diagramas de clases y bases de datos.

Gummi

Gummi es el editor de documentos LaTeX utilizado para redactar este escrito. Utiliza la librería gráfica GTK e incorpora un visualizador en tiempo real que muestra el resultado del documento de acuerdo a los cambios que se hacen. Es rápido, ligero y dispone de identificación de sintaxis y otras herramientas genéricas para facilitar la escritura con este lenguaje.

VisualParadigm

VisualParadigm es una herramienta de edición de esquemas que dispone de versiones online y de escritorio. Aunque es un programa de licencia propietaria, ha sido de gran utilidad a la hora de construir buena parte de los diagramas que se adjuntan en este documento. Además, incorpora utilidades que permiten analizar código escrito en Java para crear representaciones del mismo, aunque con ciertas deficiencias que han sido corregidas para la exposición de este proyecto.

CAPÍTULO 6
Pruebas y documentación

6.1 Pruebas unitarias

Uno de los objetivos marcados para Dagaz era la incorporación de tests que permitan garantizar que su funcionamiento es el esperado. Para lograrlo, se optó por utilizar TDD (*Test Driven Development* o desarrollo dirigido por tests), según el cual se construyen inicialmente los tests que describen el comportamiento que se espera de la herramienta para después escribir código que los pase con las condiciones establecidas.

Como resultado, se producirá una amplia batería de tests con los que trabajar y que permitirán detectar regresiones u otros incidentes que puedan darse en el código mientras éste se modifique.

Como ya se había mencionado anteriormente, se hizo uso de varias librerías de *testing*: JUnit, Mockito, AssertJ y Powermock. De entre todas ellas, Mockito es la más relevante dado que permite crear objetos simulados y facilitar el proceso de *testing*, permite falsificar los valores devueltos por objetos y funciones, verificar llamadas a ciertos objetos y los parámetros con los que suceden y, en definitiva, desarrollar tests unitarios más independientes y complejos.

Dagaz cuenta con un paquete *core* de clases, aquellas que aportan la funcionalidad básica y por defecto con la que se provee la aplicación, compuesto por servicios, populadores y fachadas. El área que más debería verificarse es, sin lugar a dudas, la que corresponde a este código.

Con la batería de tests se pretende alcanzar una cobertura de clases, métodos y líneas del 100 %. Efectivamente, cumpliendo estas tres características, se podría decir que la aplicación está completamente probada para cualquier posible camino de ejecución, y para ellos existirán pruebas que verificarán su comportamiento en todos y cada uno de los casos.

En total, la batería de tests está compuesta de **244 tests** que cubren **31 clases distintas**, sumando cerca de **3400 líneas de código Java**. Comparándolo con el código dedicado a incluir la funcionalidad del proyecto y la documentación del mismo, que ocupa aproximadamente 4440 líneas de código Java, **casi el 43 % del código escrito corresponde a tests**.

Element	Class, %	Method, %	Line, %
facade	100% (7/7)	100% (40/40)	100% (131/131)
populators	100% (12/12)	100% (23/23)	100% (166/166)
repository	100% (0/0)	100% (0/0)	100% (0/0)
service	100% (11/11)	100% (104/104)	100% (486/486)
util	100% (1/1)	100% (3/3)	100% (5/5)

Figura 6.1: Cobertura de la suite de tests para el paquete *core*

6.2 Pruebas de aceptación

Los efectos del COVID-19 en España durante los meses de confinamiento de 2020 impidieron que las pruebas de aceptación se llevaran a cabo con un grupo mayor de usuarios de test. Sin embargo, fue posible reunir a varios individuos, todos ellos profesores de la escuela pública valenciana con relativa experiencia en el uso del CMS ITACA. Para mantener su privacidad, sus nombres serán omitidos de este documento.
Este colectivo estaba compuesto de 6 individuos, a todos ellos se les pidió que evaluaran su experiencia en las nuevas tecnologías mediante una puntuación de entre 0 y 10. Gracias a ello se pudo determinar que había una cierta variedad y se pudo identificar patrones de comportamiento con respecto a cómo empleaban la herramienta. Contar con esta variedad es relevante, pues permite obtener distintos puntos de vista y usabilidad, críticos para una herramienta que está construyéndose utilizando a un público similar como objetivo y contemplando también a usuarios menos experimentados.
Se realizaron dos sets de pruebas, una por cada Sprint finalizado, y se tomó nota de varios aspectos de la herramienta y la satisfacción del usuario durante su uso, así como posibles mejoras o comentarios que podrían convertirse en nuevos requisitos en un futuro.

6.2.1. Sprint 1

Para la prueba del Sprint 1 se indicó a los usuarios el objetivo que se pretendía cumplir con Dagaz y se les introdujo ligeramente a la misma. Inicialmente se les dejó navegar por la herramienta y hacer click en los elementos que consideraran, a modo de proceso explorativo con el que determinar los puntos a los que dedicaban mayor atención.
Al comenzar la prueba se les dio una serie de objetivos idénticos y se observó su comportamiento en cada uno de ellos. Se proporcionó asistencia a aquellos usuarios que requirieron ayuda o algún tipo de guía durante el proceso. De ella se desprendieron los siguientes datos:

Usuario	**US1**	**US2**	**US3**	**US4**	**US5**	**US6**
Experiencia con TIC	8	4	6	7	7	10
Rapidez	9	9,5	8,5	10	8,6	8
Intuitividad	9	6,5	8	9	8	9
Facilidad de uso	8,5	7	9	8,2	7,3	8,6
Interfaz	8	7	8	8,5	8,4	8,7
Satisfacción	10	7,5	8	8,3	7,8	8

- De todos los encuestados, US2 mostró muchas dudas con la interfaz y presentó dificultad para completar las tareas sin asistencia externa.
- US2 y US5 reportaron cierta confusión al navegar a través de los elementos de la *navbar* (barra de navegación) de la aplicación.
- US1, US3, US4 y US6 valoraron muy positivamente la facilidad del marcado de asistencias en la herramienta.

Tras investigar con ellos los detalles que determinaron incómodos o poco intuitivos, se determinó introducir cambios a la interfaz de algunas páginas para facilitar su navegación, así como la adición de apartados de ayuda con mayor detalle con los que guiar a los usuarios y mayor responsividad para las pantallas que empleaban.

6.2.2. Sprint 2

El Sprint 2 se repitió con los mismos usuarios, repitiendo el mismo proceso que en la última ocasión, explicándole los cambios introducidos y dejando al usuario navegar libremente mientras descubría la aplicación y sus novedades.

Se les pidió a todos ellos que repitieran las mismas actividades que llevaron a cabo en la primera prueba, de tal forma que se pudiera determinar si la interfaz previa había sido lo suficientemente memorable como para agilizar el proceso al retomarlo tras un tiempo. Esto permitiría determinar si la familiaridad con la aplicación había incrementado, así como asegurarnos de qué elementos de su interfaz se habían recordado de la anterior ocasión.

Usuario	**US1**	**US2**	**US3**	**US4**	**US5**	**US6**
Experiencia con TIC	8	4	6	7	7	10
Rapidez	9	10	9,5	10	9,2	10
Intuitividad	9	8	8,5	10	9	10
Facilidad de uso	9	7	9	8,2	8,6	8
Interfaz	8,5	9	9	9,3	10	8,6
Satisfacción	10	8,5	9	8,3	8,7	9

De los resultados se extrae que la aplicación mejoró en todos los ámbitos estudiados, evidenciando que los cambios introducidos para mejorar la experiencia de uso tuvieron su recompensa.

Se recogieron algunas opiniones de los usuarios, tales como:

- US1 valoró positivamente que en el apartado de notas se pudiera hacer uso tanto de comas como puntos decimales
- US2 mostró un comportamiento menos errático y aprovechó los botones y mensajes de ayuda para guiarse a través de la aplicación
- US3 solicitó un cambio en la creación de asignaturas para evitar que el identificador interno de las mismas se pudiera utilizar simultáneamente en distintas asignaturas
- US4 y US6 elogiaron la facilidad de introducción de notas y que se pudieran asociar documentos a las evaluaciones de las pruebas, respectivamente
- US1 solicitó que se incluyera un modo oscuro para la interfaz

En definitiva, las pruebas con usuarios resultaron exitosas y permitieron recuperar datos relevantes para adaptar la herramienta al público, y la variedad de entrevistados con los que se contaba permitió que, a pesar de tratarse de un grupo muy reducido, fuera una experiencia enriquecedora para ambas partes.

6.3 Documentación

Con la intención de dar a entender qué funciones desempeña Dagaz para que otros programadores puedan trabajar en ella, se ha provisto a la aplicación de documentación a través de *Javadocs*, el estándar para documentación de código en Java. De las cerca de 4440 líneas que componen la aplicación, casi $\frac{1}{4}$ corresponden a este tipo de comentarios.

La finalidad de contar con una referencia con la que poder entender el funcionamiento del sistema y su arquitectura es crucial para que otras personas sean capaces de modificarlo. Por ello se destinó tiempo y esfuerzo a proveer de esta

información para los potenciales desarrolladores que apoyarían la herramienta.

Por supuesto, no toda la información tendría por qué encontrarse únicamente en el código, pues hay aspectos (decisiones tecnológicas, arquitecturales o de otra índole) que pueden no desprenderse directamente con la lectura de los archivos del proyecto. Por eso mismo, este documento tiene también la intención de convertirse en una referencia que dice cuáles son:

- Los detonantes que llevaron a la construcción de la herramienta, así como las alternativas que han servido de inspiración para hacerla.
- Los objetivos del proyecto, para qué se ha construido y a quién va dirigida la herramienta.
- Las licencias con las que se distribuye y el razonamiento detrás de ello.
- La arquitectura que lo compone, así como la funcionalidad de cada una de sus partes.
- Una descripción a grandes rasgos de las capacidades de la herramienta.
- Su visión de futuro.

Así pues, este documento cumple un doble papel como trabajo de final de grado y también de documentación de la herramienta desde un punto de vista distinto al que se proporcionaría en la descripción de las funciones que componen Dagaz.

CAPÍTULO 7

Ejemplo de uso

Un usuario puede acceder a Dagaz a través de su navegador, indicando la dirección IP donde se aloje el servidor. Será recibido en una página de *login* desde la que podrá introducir su DNI y su contraseña para acceder a su cuenta.

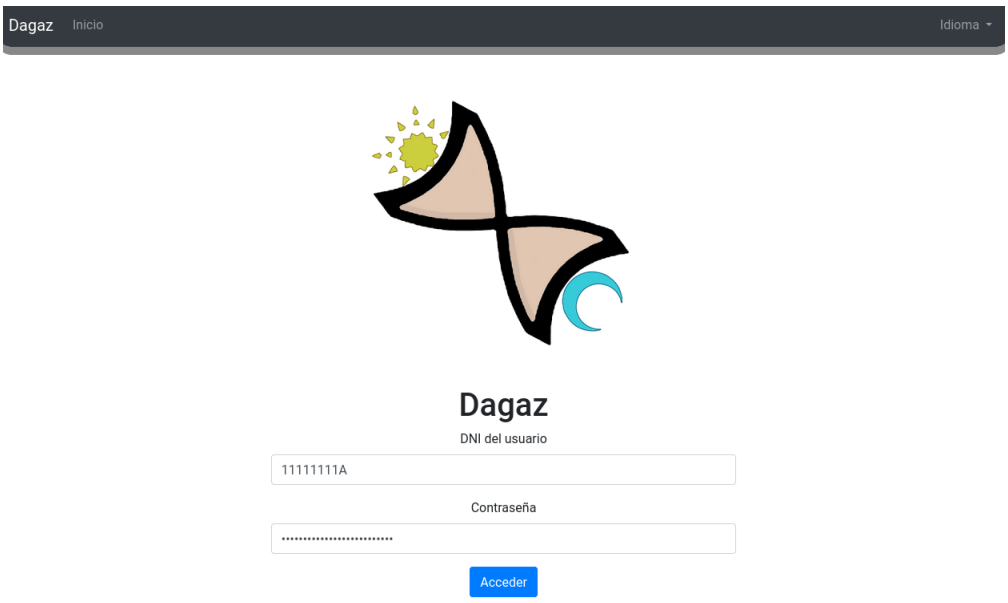

Figura 7.1: Página de *login* de Dagaz

Actualmente, Dagaz cuenta con dos perspectivas distintas: La de **profesor** y la de **administrador**. Cuando un usuario acceda a la página se determinará a cuál de estos grupos pertenece y podrá visualizar la página de inicio correspondiente a su rol.

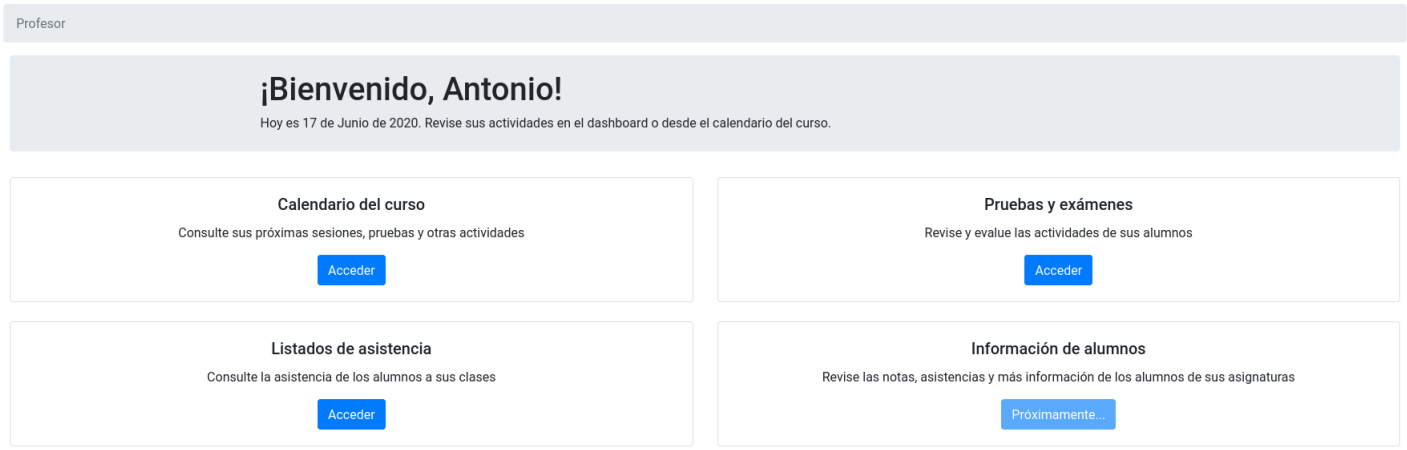

Figura 7.2: Ejemplo de página de inicio del Profesor

7.1 Vistas del profesor

7.1.1. Calendario del curso

El profesor podrá revisar las sesiones y pruebas tanto pasadas como presentes desde el calendario del curso. Tendrá disponibles 3 opciones de visualización: Semana, agenda del día y vista del mes. Cada evento vendrá indicado con un código de color para que sea fácil determinar cuándo tendrá lugar.

Podrá hacerse click en cualquiera de los eventos del calendario para navegar hasta ellos. Los que correspondan a sesiones redirigirán a la página de asistencia, mientras que las pruebas enviarán al usuario a la página de notas.

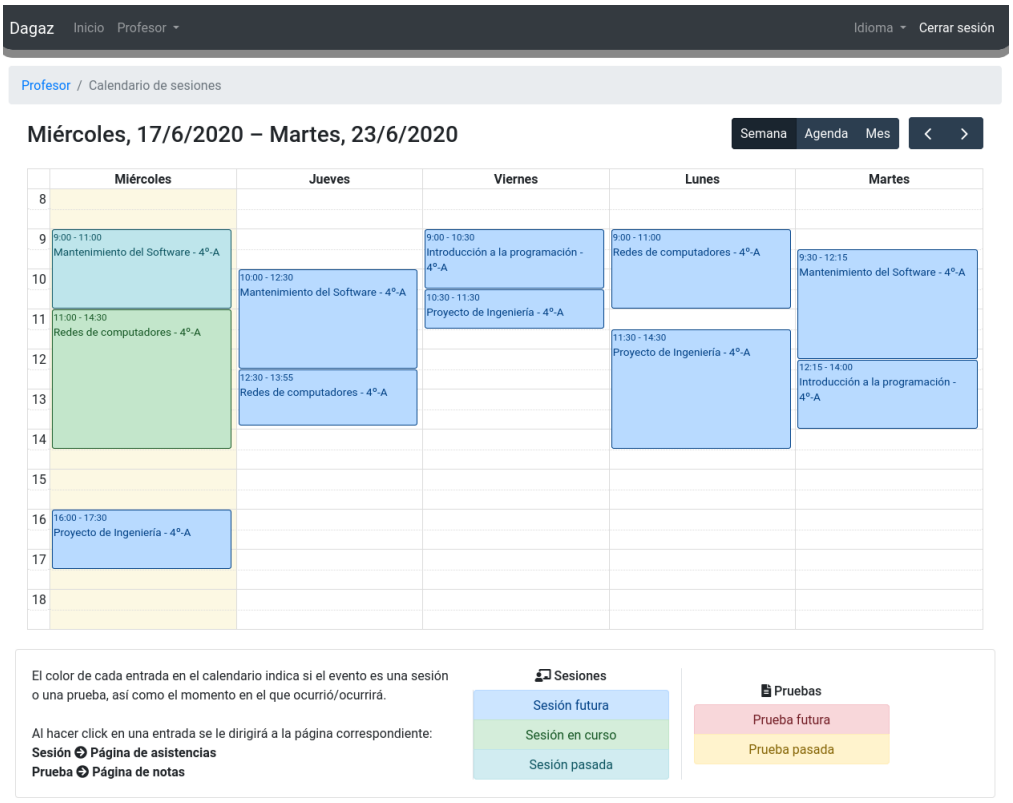

Figura 7.3: Calendario del curso

7.1.2. Página de asistencias

En la página de asistencias se tendrá al acceso toda la información de la sesión para la que se están indicando la presencia o ausencia de los alumnos. Se contará con un listado de los alumnos que participarían en dicha sesión y una tabla marcada con códigos de color que permiten que el usuario pueda determinar con un simple vistazo si se han presentado o no.

La interfaz está pensada para que el usuario pueda, haciendo click en uno de los botones que se muestran en el lado derecho de la tabla, cambiar el estado de la asistencia de sus alumnos con facilidad. El color de cada fila cambia según la opción escogida, y los botones se difuminan para hacer más evidente si cabe cuál es el estado del alumno.

En la parte superior de la misma se cuenta con dos bloques: **Asistencias marcadas** y **Ayuda**. La primera es un contador que indica cuántos alumnos han sido marcados con alguna asistencia en esta sesión, mientras que el botón de ayuda despliega un pequeño apartado con una breve descripción que guía al usuario en el caso de que éste no supiera cómo continuar el marcaje.

Utilizando el mismo esquema de colores en múltiples puntos de la pantalla como si se tratara de un apoyo visual, el marcaje de notas es una experiencia rápida y memorable.

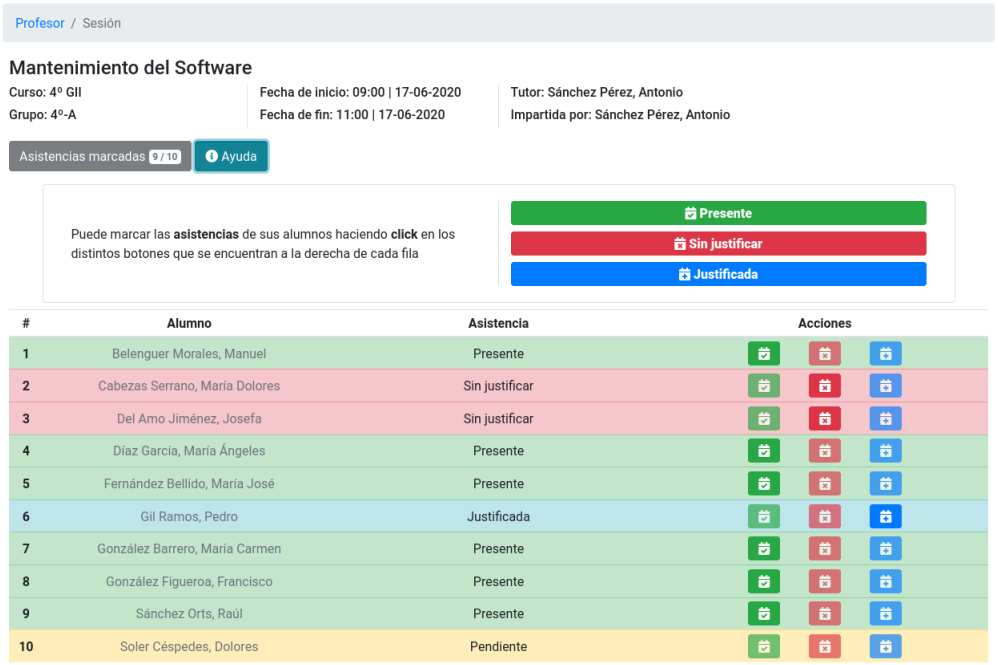

Figura 7.4: Página de asistencias

7.2 Página de creación de pruebas

Un profesor puede crear una prueba para todos los alumnos de un curso concreto. Especificando una hora y una fecha concretas, así como el grupo que será evaluado; podrá crearse la prueba.

En el lado dercho se mostrará un acordeón que el usuario podrá abrir para ver cuáles son los alumnos que se presentarán a la prueba. Además, también se contará con un apartado de ayuda que sirva como guía de los pasos a seguir para crearla correctamente.

Si un usuario introdujera valores incorrectos, el programa le indicaría cuáles han sido los campos inválidos y mostraría una pequeña ayuda para que los corrija. Cuando la prueba se crea con éxito, el usuario es redirigido a la página de marcaje de notas.

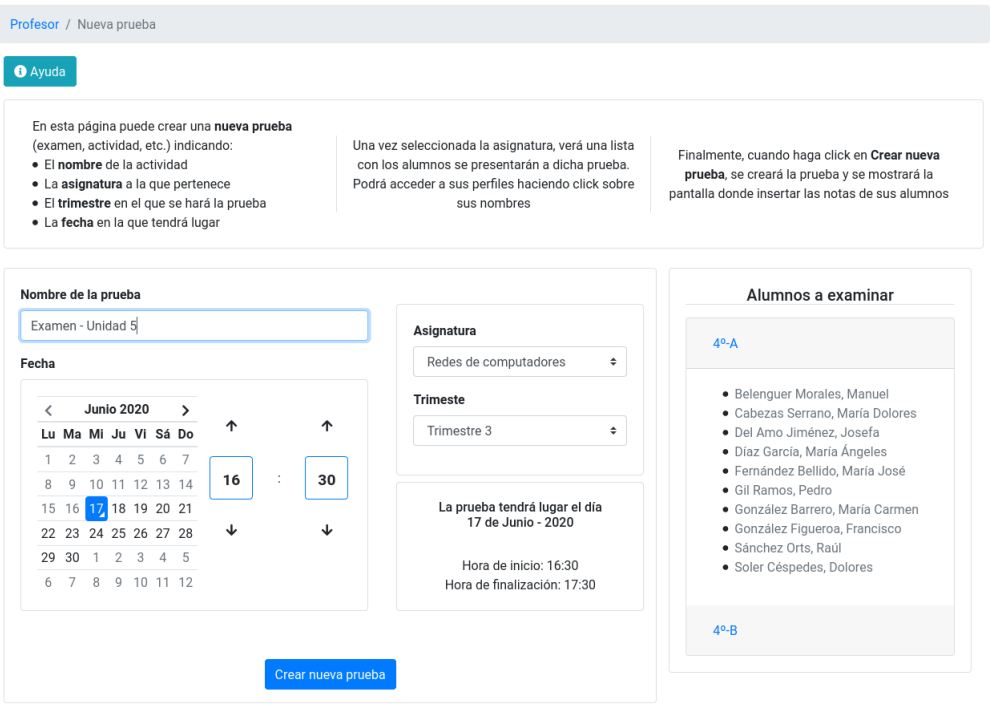

Figura 7.5: Página de creación de pruebas, vista de escritorio

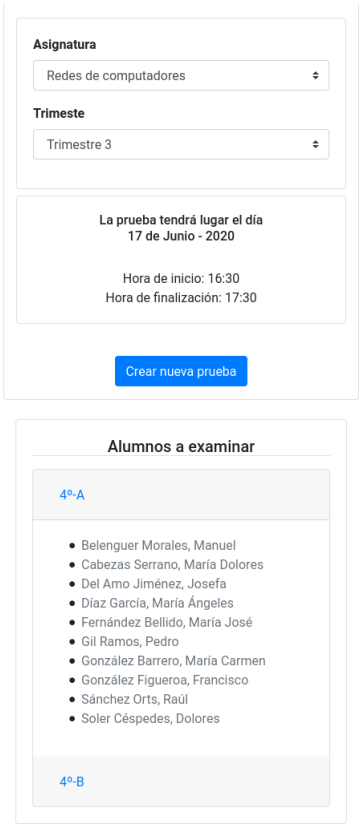

Figura 7.6: Fragmento de la página de creación de pruebas, vista de móvil

7.2.1. Página de marcaje de notas

Cuando se accede a una prueba se obtiene la siguiente vista para introducir las notas de los alumnos y subir documentos de sus correcciones. Si el usuario es redirigido a esta página tras haber creado una prueba, será recibido con un mensaje que le permita saber que su prueba se ha creado con éxito, reforzando positivamente su experiencia con la herramienta.

De manera similar al caso expuesto antes en la página de asistencias, también se contará con un apartado de **Notas guardadas**, para indicar cuántos de los alumnos han sido evaluados, y un botón de ayuda que despliega texto con el que aconsejar al usuario.

Para evitar que se realicen demasiadas llamadas al servidor, la actualización de notas se envía en bloques. Cuando el usuario está modificando una nota, el contador se reinicia, permitiendo que se envíen el mayor número de notas por llamada para reducir la carga en el extremo del servidor.

Las notas se pueden introducir como texto. Cuando el profesor inserte un valor, el campo aparecerá en azul si es válido y se mostrará un *spinner* para indicar que se está enviando al servidor. Por motivos de seguridad, se marcará la identidad del profesor que evalúa a cada alumno.

El *input* de notas se valida de tal forma que se puedan emplear comas decimales o puntos decimales. Toda nota estará entre un rango de 0 y 10, y podrá utilizar hasta 2 decimales. Si una nota no cumple con estas condiciones, su campo se tornará rojo para dar a entender que el valor es inválido, y se mostrará un mensaje para que el usuario pueda corregirlo.

Finalmente, a mano derecha se contará con dos botones de subida y bajada de documentos de corrección. Los usuarios podrán adjuntar archivos PDF de no más de 10MB.

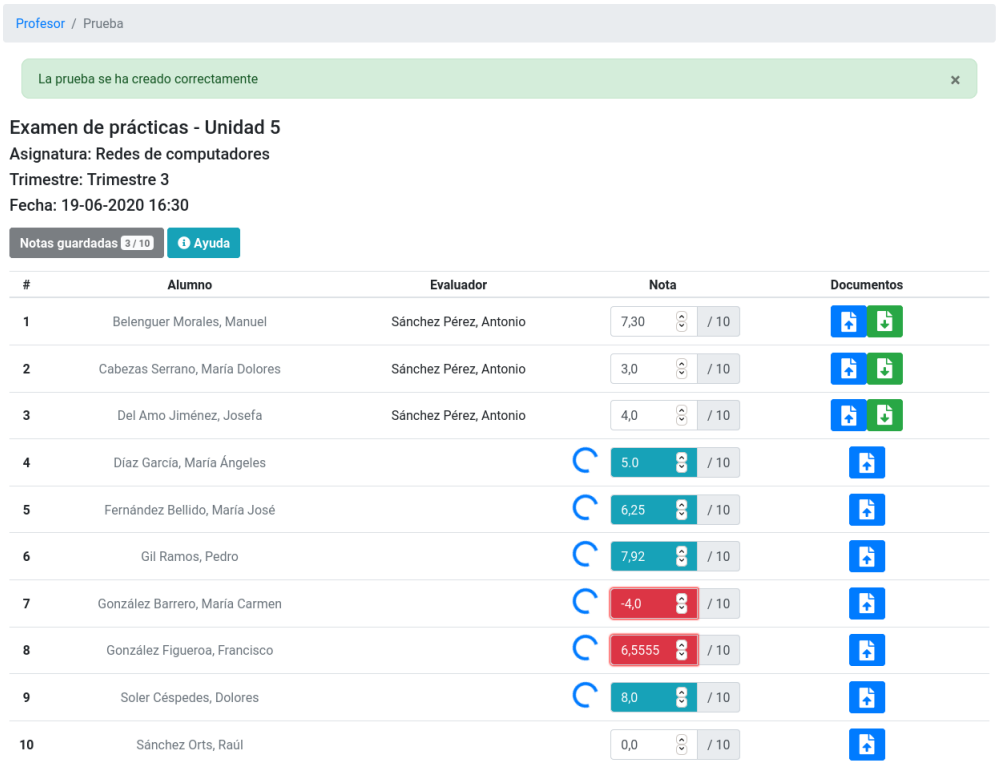

Figura 7.7: Página de marcaje de notas

7.3 Vistas del administrador

7.3.1. Página de creación de curso

Los cursos podrán ser creados por el administrador únicamente para el centro que administra y el curso actual. Haciendo click en el botón azul con el símbolo plus se podrán añadir nuevos cursos.

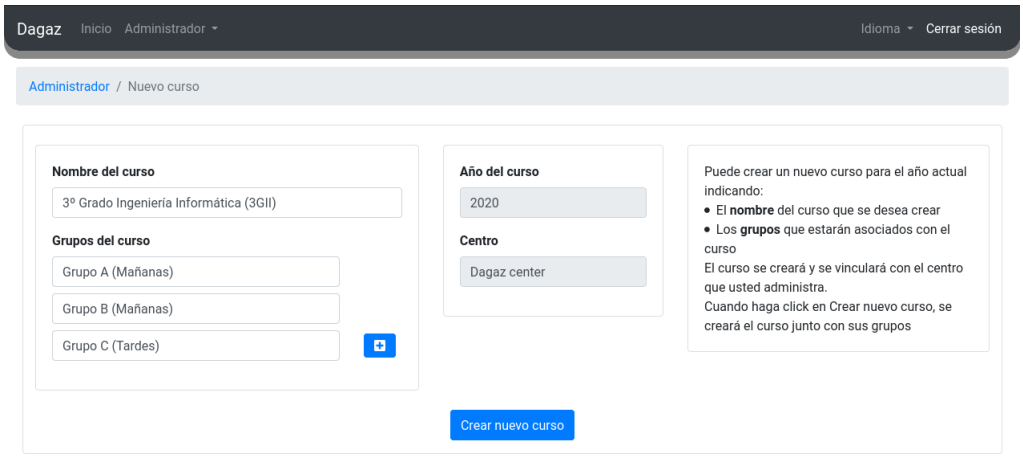

Figura 7.8: Página de creación de curso

7.3.2. Página de creación de asignaturas

La página de asignaturas es el aspecto más complejo de la aplicación e incluye una interfaz especialmente diseñada para facilitar el uso ante la cantidad de datos con los que se tratan.

En primer lugar, el administrador podrá indicar el nombre de la asignatura, así como el identificador interno que se utilizará en ella. Acto seguido podrá indicar a qué curso pertenecerá, así como el tutor que se encargará de gestionarla.

Tras seleccionar un curso o un tutor, se cargará información asociada a ellos. Estos datos permiten aclarar con un simple vistazo si la opción seleccionada es realmente la adecuada.

En el lado derecho, el administrador podrá crear los trimestres que tendrá la asignatura, marcando su nombre y fechas de inicio y fin. Si se desea añadir nuevos trimestres se podrá hacer click en el botón con el símbolo de plus.

Finalmente, bajo todos estos datos, se encuentra la tabla de creación de sesiones. En ella se podrá indicar el grupo que tendrá la sesión, el profesor que la impartirá, los días de la semana en los que tendrá lugar y también la hora de inicio y fin de dicha sesión.

Nótese los campos de fecha de inicio y fecha de fin que se encuentran a la derecha. Éstos serán utilizados para indicar entre qué periodos se crearán las sesiones. Así pues, con indicar las fechas de inicio y fin a modo de rango, se generarán automáticamente sesiones para los días de la semana indicados utilizando las horas de inicio y fin seleccionadas.

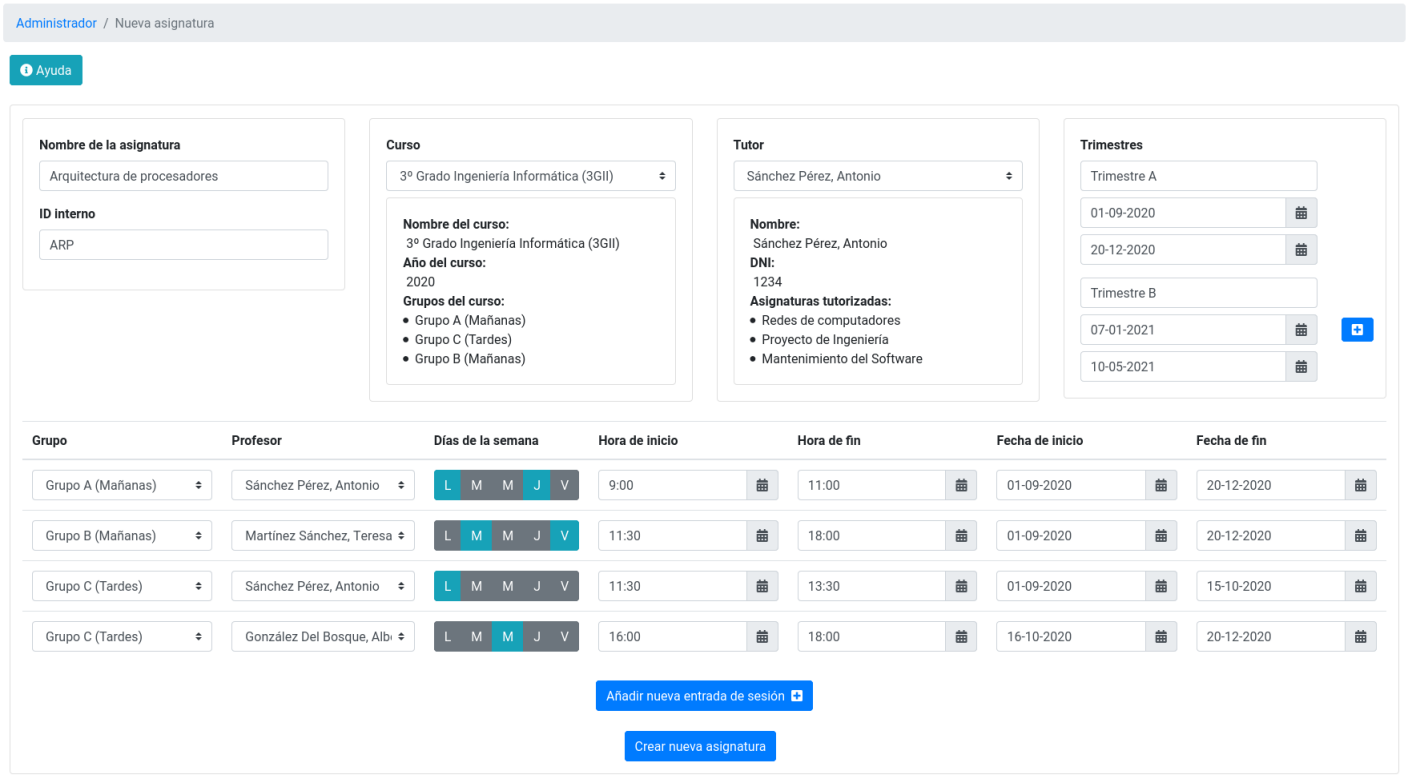

Figura 7.9: Página de creación de asignatura

CAPÍTULO 8
Conclusiones

Dagaz ha sido un proyecto en el que se han invertido muchas horas de trabajo y que ha resultado una experiencia enriquecedora y agotadora. Dado que dio comienzo en una situación de máxima inestabilidad provocada por el COVID-19 y su impacto en España, ha contado con muchos factores en contra y riesgos que se han materializado, afectando al curso original del proyecto.

Por ello, valorando el resultado final y considerando las vicisitudes por las que se ha pasado para lograr su construcción, el proyecto se valora positivamente.

Para la construcción de Dagaz no se contaba con programadores frontend y el equipo de desarrollo se componía únicamente de un desarrollador backend con muy poca experiencia en interfaces y otros aspectos de experiencia de usuario (*UX*), con lo que el esfuerzo ha sido doble en este ámbito. Sin embargo, es grato aprender nuevas tecnologías y obtener algo más de destreza.

La experiencia profesional con la que se contaba en Spring Framework estaba ligeramente alejada de la perspectiva que Spring Boot toma para la configuración del entorno. Utilizando como base el bagaje del que se disponía, se pudieron descubrir nuevos aspectos del framework y aplicaciones que podrían emplearse tanto a nivel personal en el desarrollo de Dagaz como en el ámbito profesional.

Dentro de las capacidades que originalmente se planearon para Dagaz, se valora que el alcance original se ha visto recortado. No se contemplaba que la dificultad de construir una herramienta de estas características pudiera ser tal, pero es positivo saber que la solución aportada se ha construido procurando aplicar buenas prácticas y teniendo en cuenta que esta aplicación podría tener un uso real, podría servir a la comunidad y alejarse de ser un proyecto meramente teórico. Por ello, por valorarse como una herramienta útil y que pudiera servir, se han tomado muchísimas decisiones que pudieran acercar Dagaz al mundo y a su uso en entornos reales.

En su estado actual, Dagaz no difiere mucho de otros CMS, y se asume que es razonable su parecido. Sin embargo, la base con la que se ha edificado es diferente, cuenta con un enfoque distinto que tal vez no es todavía palpable en el punto de desarrollo en el que se encuentra. Sin embargo, hay ideas y conceptos que en un futuro sí podrían desarrollarse y que podrían aportar nuevas perspectivas al mundo de la educación, su gestión y sus contenidos educativos.

CAPÍTULO 9
Trabajos futuros

Como tareas pendientes y que podrían tener gran relevancia una vez Dagaz madure o, quién sabe, cuente con más apoyo tanto de desarrolladores como tal vez económico, se contempla la inclusión de nuevas funcione y la finalización de algunas ya existentes

- Finalización de la perspectiva de administración incluyendo la función de matriculación de alumnos.
- Construcción de un componente base que sirva como punto de partida para extensiones u otras adiciones. modulares, de tal forma que el componente sea el estándar *de facto* de una implementación básica de Dagaz.
- Inclusión de la perspectiva de alumnos, con opciones tales como
  - Visualizar notas y expediente educativo, incorporando funciones de evaluación de rendimiento y análisis académico.
  - Comunicación profesor-alumno directamente desde la herramienta.
  - Adición de un módulo de comunicación profesor-padres/tutores del alumno.
- Creación y mejora de la API básica existente para extenderla y favorecer la integración de otros sistemas con Dagaz.
- Publicitar Dagaz y buscar financiación presentando el proyecto a diversas instituciones europeas.
- Mejoras en la interfaz para lograr que todavía sea más responsive.
- Construcción de una aplicación para *smartphones* y *tablets* con la que acceder a Dagaz y sus funciones
- Exploración y construcción de un protocolo de comunicación que permita la federación entre instancias de Dagaz
- Adición de un apartado de configuración de la interfaz para simplificar la personalización de cada instancia
-

CAPÍTULO 10
Agradecimientos

Quiero dedicar este trabajo en especial **a mis padres Teresa y José**, ambos profesores de profesión y de vocación, que han llevado su oficio en el alma y han dedicado toda su vida a la docencia con paciencia y esmero. A ellos, que me han enseñado a aprender, de quienes he heredado su esfuerzo y sus ganas de seguir adelante; son los primeros a quienes quiero elogiar en este fragmento final.

También **a mi hermano gemelo Rafael**, que ha sabido levantarme incluso en los momentos más bajos, de quien aprendo tanto cada día, que me ha ayudado a ver el mundo con otros ojos y tiene la paciencia para aguantarme cuando me convierto en un manojo de nervios. Ha escuchado mil y una veces este escrito y no ha rechistado ni una vez, solo eso sería ya suficiente para construirle un monumento... Sin él, ni este texto ni yo estaríamos aquí, y no hay suficientes páginas en este texto para dedicarle su labor.

A mi familia, pero en especial **a mi tía Mª Carmen**, que siempre me han acompañado en los momentos duros y me han apoyado en todas las decisiones que he tomado, que no han dejado de apostar por mí ni un solo momento. También son poseedores de una paciencia infinita y consiguen lo imposible, incluso sacarme una sonrisa en los días grises.

A mis amigos, que también son mi gran apoyo, que siempre han estado ahí para ayudarme en lo que hiciera falta, con quienes comparto dudas, inquietudes y trastadas, los que son ya una parte más de mi familia. Mención especial para **Daniel O.**, autor del logo que adorna la página principal de Dagaz y que ha tenido la inventiva para diseñar esa pieza; y **Javier P.**, que leyó hasta las comas de este trabajo y me ayudó muchísimo con él.

A Vicente Pelechano, mi tutor para este trabajo, que aceptó mi proposición de TFG sin dudarlo un minuto, apoyó en todo momento la idea y ha valorado mucho su construcción y consecución. Sin él, este texto jamás se habría escrito y Dagaz no sería más que una idea esbozada en un papel. Tardé meses en encontrar un tutor, pero estoy agradecido de haber dado con el adecuado.

No puedo olvidar **a la comunidad que lucha activamente por promulgar el Software Libre** allá a donde va, para quienes la informática es una herramienta para cambiar el mundo más allá de la pantalla. También a todos aquellos que defienden la libertad de uso de la tecnología y producen herramientas increí-

bles que cambian la faz de la tierra para el bien de todos.

A todos mis compañeros de trabajo, de quienes he aprendido tantísimo a nivel profesional y personal, que han sido una gran fuente de inspiración y me han hecho crecer en muchos aspectos tanto programador como humano, que me han enseñado a desenvolverme en un mundo que apenas hace unos años era ajeno a mí.

A toda la comunidad educativa, en especial a los profesores de la escuela pública que luchan día a día por enseñar a sus alumnos, a los que trabajan de sol a sol y llevan su profesión en el corazón. Extiendo esta dedicatoria a todos los profesores de los que he tenido la suerte de ser alumno, aquellos que fueron capaces de hacerme aprender más allá de un aula, que se esforzaron por inclucarnos a todos sus estudiantes el interés por el mundo, la ciencia y, en definitiva, por ayudarnos a descubrir y preguntarnos qué hay más allá.

En definitiva, gracias a todos los que habéis hecho posible que Dagaz exista.

Bibliografía

[1] Glenda Morgan *Faculty Use of Course Management Systems*. University of Wisconsin System, volumen 2, 2003 - Investigación de EDUCAUSE Center for Applied Research

[2] Online Education Market & Global Forecast, by End User, Learning Mode (Self-Paced, Instructor Led), Technology, Country, Company - Research and Markets
Consultado en https://www.researchandmarkets.com/reports/4876815/online-education-market-and-global-forecast-by

[3] Página web oficial de ITACA, Generalitat Valenciana - Conselleria d'educació, cultura i esport
Consultado en https://www.ceice.gva.es/webitaca/es/

[4] Portal de ITACA, Generalitat Valenciana - Conselleria d'educació, cultura i esport
Consultado en https://portal.edu.gva.es/cvtic/es/herramientas/itaca/

[5] Sección Centros - Página web oficial de ITACA, Generalitat Valenciana - Conselleria d'educació, cultura i esport
Consultado en https://www.ceice.gva.es/webitaca/es/centres.htm

[6] Open Source Strategy - Página web de la Comisión Europea
Consultado en https://ec.europa.eu/info/departments/informatics/open-source-software-strategy_en

[7] Public Money, Public Code - Carta abierta a la Comisión Europea
Consultado en https://publiccode.eu/

[8] Objetivos de Desarrollo Sostenible - Programa de las Naciones Unidas para el Desarrollo
Consultado en https://www.undp.org/content/undp/es/home/sustainable-development-goals.html

[9] Reality check - Open Source LMS: a costly affair
Consultado en https://elearningindustry.com/reality-check-open-source-lms-costly-affair

[10] The true cost of a learning management system - eLearning Industry
Consultado en https://elearningindustry.com/true-cost-of-a-learning-management-system

[11] Página oficial de Sakai
Consultado en https://www.sakailms.org/

[12] About Moodle - Página oficial de Moodle
Consultado en https://docs.moodle.org/38/en/About_Moodle.

[13] Servicio de VideoApuntes - Universitat Politècnica de Valencia
Consultado en https://videoapuntes.upv.es/

[14] Google Classroom - Página oficial
Consultado en https://edu.google.com/intl/es/products/classroom

[15] CENTURY, an AI-powered teaching and learning platform - UNESCO
Consultado en https://en.unesco.org/news/century-ai-powered-teaching-and-learning-platform

[16] Microsoft M365 - Página oficial
Consultado en https://www.microsoft.com/en-us/education/buy-license/microsoft365/default.aspx

[17] Turnitin - Página oficial
Consultado en https://www.turnitin.com/

[18] 'You're being watched and recorded, every breath': Students unsettled by exam software - The Sydney Morning Herald
Consultado en https://www.smh.com.au/national/nsw/you-re-being-watched-and-recorded-every-breath
html

[19] 'Big Brother' software commodifies and endangers the private information of students - Sky News Australia
Consultado en https://www.skynews.com.au/details/_6152560663001

[20] UQ students complain about exam monitoring software - Brisbane times
Consultado en https://www.brisbanetimes.com.au/national/queensland/uq-students-complain-about-exam-monitoring-software-20200615-p552qm.html

[21] Tabla de precios de *hosting* - Página oficial de Moodle
Consultado en https://moodlecloud.com/app/en/

[22] Sueldo medio de un desarrollador PHP - Glassdoor
Consultado en https://www.glassdoor.es/Sueldos/php-developer-sueldo-SRCH_KO0,13.htm

[23] Educational Community License, Version 2.0 (ECL-2.0) - Open Source Initiative
Consultado en https://opensource.org/licenses/ECL-2.0

[24] Sueldo medio de un desarrollador Java - Glassdoor
Consultado en https://www.glassdoor.es/Sueldos/java-developer-sueldo-SRCH_KO0,14.htm

[25] Feature details - Página oficial de Sakai
Consultado en https://www.sakailms.org/feature-details

[26] Repositorio de Sakai Contrib Tools - Github
Consultado en https://github.com/sakaicontrib

[27] Sakai Community Members - Página oficial de Sakai
Consultado en https://www.sakailms.org/sakai-community-members

[28] ITACA: Web Familia - Página oficial de la Conselleria D'Educació i Esport de la Generalitat Valenciana
Consultado en http://www.ceice.gva.es/webitaca/es/pares_alumnes.htm

[29] ¿Qué es? - Cl@ve - Página oficial de Cl@ve, Gobierno de España
Consultado en https://clave.gob.es/clave_Home/es/clave/queEs.html

[30] Decentralization, federation, and self-hosting - Página oficial de Free Software Foundation
Consultado en https://www.fsf.org/campaigns/priority-projects/decentralization-federation

[31] What on Earth is the fediverse and why does it matter? - New Atlas
Consultado en https://newatlas.com/what-is-the-fediverse/56385/

[32] Fediverse - Wikipedia
Consultado en https://en.wikipedia.org/wiki/Fediverse

[33] What is free software? - Página oficial de Free Software Foundation
Consultado en https://www.gnu.org/philosophy/free-sw.html

[34] Affero GNU General Public License v.3 - Página oficial de Free Software Foundation
Consultado en https://www.gnu.org/licenses/agpl-3.0.en.html

[35] Encuesta a usuarios de ITACA - Elaboración propia
Consultado en https://forms.office.com/Pages/AnalysisPage.aspx?id=DQSIkWdsW0yxEjajBLZtrQAAAAAAAAAAAAO_&_du6tcBUMDhBTENBRlQyQzVISzNVSFFFNjVFMklQOC4u&AnalyzerToken=bXv0sCAQDjom6QZckL82x2vSeQIHwDuT

[36] GNU General Public License v.3 - Página oficial de Free Software Foundation
Consultado en https://www.gnu.org/licenses/gpl-3.0.en.html

[37] Apache Tomcat - Página oficial de Apache Software Foundation
Consultado en https://tomcat.apache.org/

[38] Spring Framework - Página oficial de Spring
Consultado en https://spring.io/projects/spring-framework

[39] Apache License 2.0 - Página oficial de Apache Software Foundation
Consultado en https://www.apache.org/licenses/LICENSE-2.0

[40] Hibernate ORM - Página oficial de Hibernate
Consultado en http://hibernate.org/orm

[41] GNU Lesser General Public License - Página oficial de Free Software Foundation - Consultado en https://www.gnu.org/licenses/lgpl-3.0.html

[42] GNU General Public License v.2 - Página oficial de Free Software Foundation
Consultado en https://www.gnu.org/licenses/gpl-2.0.en.html

[43] About - Página oficial de PostgreSQL
Consultado en https://www.postgresql.org/about

[44] PostgreSQL License - Página oficial de PostgreSQL
Consultado en https://www.postgresql.org/about/licence

[45] The 2-Clause BSD License - Open Source initiative
Consultado en https://opensource.org/licenses/BSD-2-Clause

[46] Frequently Asked Questions - Sección GPL Incompatible Libs - Página oficial de Free Software Foundation
Consultado en https://www.gnu.org/licenses/gpl-faq.en.html#GPLIncompatibleLibs

[47] Frequently Asked Questions - Sección MereAggregation - Página oficial de Free Software Foundation
Consultado en https://www.gnu.org/licenses/gpl-faq.html#MereAggregation

[48] About MariaDB Server - Página oficial de MariaDB
Consultado en https://mariadb.org/about/

[49] Licencia de MySQL Server - Repositorio oficial de MysqlServer - GitHub
Consultado en https://github.com/mysql/mysql-server/blob/8.0/LICENSE

[50] Documentación de MySQL - Página oficial de MySQL
Consultado en https://dev.mysql.com/doc
[51] HSQL License - Página oficial de HSQLDB
Consultado en https://hsqldb.org/web/hsqlLicense.html
[52] Página oficial de HSQLDB
Consultado en https://hsqldb.org
[53] The MIT License - Open Source Initiative
Consultado en https://opensource.org/licenses/MIT
[54] Página oficial de JQuery
Consultado en https://jquery.com/
[55] Página oficial de MomentJS
Consultado en https://momentjs.com
[56] Página oficial de FullCalendar
Consultado en https://fullcalendar.io/
[57] Proyecto Tempus Dominus - GitHub
Consultado en https://github.com/tempusdominus/bootstrap-4
[58] Página oficial de JUnit 5
Consultado en https://junit.org/junit5
[59] Eclipse Public License v2.0 - Página oficial de Eclipse Foundation
Consultado en https://www.eclipse.org/legal/epl-2.0
[60] Página oficial de Mockito
Consultado en https://site.mockito.org
[61] Página oficial de AsserJ
Consultado en https://assertj.github.io/doc/
[62] Página oficial de Powermock
Consultado en http://powermock.github.io