Grado en Ingeniería Informática

Trabajo de Final de Grado

Área estadística y de asistencia en la plataforma de eLearning Edueca

Autor:
Almar Abella Moliner

Supervisor:
Héctor Saiz Sánchez
Tutor académico:
Reyes Grangel Seguer

Fecha de lectura: 12 de abril de 2022
Curso académico 2021/2022

Resumen

Este documento describe el desarrollo de un conjunto de funcionalidades en una plataforma de aprendizaje digital. Se trata de un servicio para la compañía Cidet S.L., especializada en soluciones informáticas para cuestiones de enseñanza. La web sobre la que se incorpora el producto, Edueca, tiene como propósito ofrecer un espacio en el que cualquier usuario puede ofrecer cursos (proyectos educativos) a otros. Gracias a la expansión de Edueca resultante de este proyecto, los usuarios que ofrecen cursos pueden acceder a estadísticas y consejos.

El proyecto está realizado con el marco de desarrollo Symfony, que emplea el patrón de diseño modelo-vista-controlador aplicado a plataformas web. Asimismo, una de las mayores cargas de trabajo del proyecto es el control de requisitos realizado bajo las circunstancias típicas de un desarrollo ágil, con unos objetivos definidos a nivel puramente estratégico.

Palabras clave

Symfony, Ágil, MySQL, MVC, Web, eLearning.

Keywords

Symfony, Agile, MySQL, MVC, Web, eLearning.

Índice general

Capítulo 1

Introducción

Se pretende implementar en Edueca, una plataforma de aprendizaje digital (piénsese en Moodle), un área destinada a los profesores que impartan un curso. Dicha área ofrecerá estadísticas con las que evaluar el desarrollo del proyecto educativo en cuestión, así como consejos o nociones que les ayudarán a mejorar el desempaño del curso. Se describen a continuación el contexto del que parte la iniciativa y los objetivos a los que se pretende llegar con ella.

1.1. Contexto y motivación del proyecto

Cidet S.L. (de *Centre for the Innovation and Development of Education and Technology*) es una empresa de tecnologías de la información que ofrece servicios informáticos orientados a la educación. La empresa se presenta aludiendo al valor de la tecnología en cuestiones pedagógicas, con énfasis en proyectos sociales como pueden ser el uso de herramientas informáticas en el ámbito de la tercera edad.

En su plataforma de aprendizaje digital, Edueca, los usuarios pueden adoptar el rol de entidades de enseñanza y ofrecer y gestionar cursos, o apodar el rol de alumnos y acceder a estos contenidos (o ambas). La web consta de diferentes secciones adaptadas a estos dos perfiles: profesorado (o administradores) y alumnado. Se diferencia ante otras plataformas por la sencillez con la que es posible crear proyectos educativos y manipular su contenido, ya sea importado, redirigido, o redactado desde la propia plataforma.

Adicionalmente a la gestión de los cursos, está previsto que Edueca incorpore un área en la que se sintetice visualmente el estado del curso mediante estadísticas y conclusiones derivadas de las mismas. El proyecto cuya propuesta se presenta en este documento consiste en desarrollar e implementar esta área, que deberá encajar en aspecto y a nivel funcional con el resto de la plataforma.

Cabe distinguir el alcance del proyecto dado del de la plataforma que le da soporte. Desde el punto de vista organizativo, existen dos usuarios estereotípicos de Edueca. Por una parte, los alumnos pueden acceder a cursos y su contenido: temario, enlaces a documentos o vídeos,

cuestionarios, etc. Necesitan registrarse, aceptar una invitación o ninguna (acceso público) según la configuración del curso. El área a través de la cual realizan estas actividades es la que comúnmente ve cualquier usuario que entra a un curso (*front-office*).

Como segundo usuario estereotípico y cliente de la plataforma, el profesor o administrador de una entidad de enseñanza puede gestionar su material a impartir. Dicha gestión incluye que los datos o archivos que posea puedan ser almacenados (biblioteca) y alterar la vista del curso de cara a los alumnos, tanto en contenido como en diseño, y organizar eventos virtuales a través de un calendario compartido con el alumnado. Estas acciones pueden ser realizadas desde la cara de la aplicación reservada a los clientes. El proyecto estará contenido en esta *back-office*, por lo que será pertinente que encaje con su entorno en cuanto a formato y tratamiento de las comunicaciones.

Desde el punto de vista funcional, el área a desarrollar debe recopilar datos sobre el curso, sintetizarlos para facilitar su comprensión y proporcionar asistencia pedagógica dinámica, es decir, que se adapte a estos datos específicos del curso. El análisis de datos disponibles, qué información objetiva se puede extraer de estos, y la asistencia en la toma de decisiones pedagógicas que sugiere esta información (siendo este último punto dirigido desde Cidet), estarán incluidos en el desarrollo del proyecto.

Desde el punto de vista informático, el proyecto podrá comunicarse con recursos de visualización gráfica externos o incorporarlos, según sea conveniente. Por lo demás, el entorno tecnológico será en el que ya reside Edueca. La web de Edueca, así como la base de datos MySQL donde se guardan los datos relacionados con los usuarios y cursos, están almacenadas en servidores proporcionados por DigitalOcean.

En cuanto a sistemas externos, Edueca se comunica con interfaces bancarias para gestionar las suscripciones de las entidades de enseñanza y con el Traductor de Google para adaptar los textos a los varios idiomas en los que la página se presenta, ya que está redactada nativamente alternando entre inglés y español. El área a implementar debe incorporar el uso de este servicio.

1.2. Objetivos del proyecto

Actualmente, la *back-office* de Edueca cubre un abanico de funcionalidades estrictamente reactivo; cuando un profesor introduce cambios, la *front-office* de su curso los refleja. Cidet considera que tiene cabida un conjunto de funcionalidades que hagan que Edueca genere y emita información visual, útil y llamativa sobre la actividad de los alumnos en un curso.

Los objetivos tras esta expansión de funcionalidades son estratégicos:

- Aumentar la oferta de herramientas de Edueca.
- Aumentar los ámbitos que Edueca cubre con las herramientas que ofrece, incluyendo la asistencia inteligente.
- Diferenciar Edueca en el mercado ante las plataformas alternativas de gestión de cursos.

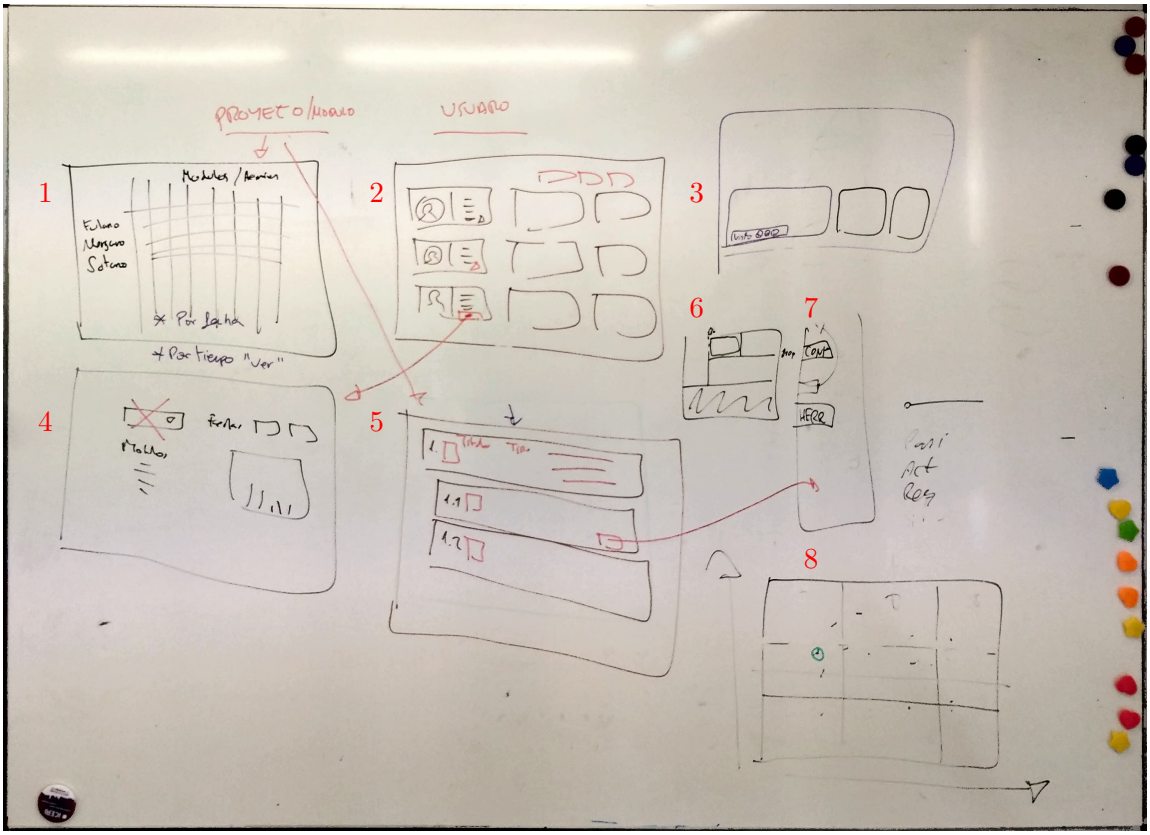

Figura 1.1: Pizarra en Cidet tras la primera reunión

Se pretenden encontrar, dentro del margen de opciones a la hora de implementar un sistema informativo, aspectos que hagan de Edueca una herramienta más necesaria para el cliente. El sistema debe constar de interactividad que aliente la sensación de control del usuario, pero requerir aprendizaje bajo por parte del usuario en comparación a las plataformas alternativas de aprendizaje digital. La primera semana de la estancia en Cidet, se da una reunión a modo de tormenta de ideas en la que el personal partícipe del proyecto propone los siguientes conceptos:

1. **Mapa de calor**. Un mapa de calor de ejes alumno/módulo (un módulo es similar a una lección del temario en el contexto de Edueca), en el que la intensidad de una celda represente la cantidad de accesos del alumno al módulo. Edueca ya incorpora una versión beta de este sistema.

2. **Lista detallada de alumnos**. Una vista en la que el profesor pueda ver representado «qué le pasa» a cada alumno, justificada mediante estadísticas extraídas del uso de la plataforma por parte del alumno.

3. **Sistema de vistos**. Un sistema de «vistos» que, similarmente a cómo Google Drive informa de qué usuarios tienen abierto un documento, muestre al profesor qué alumnos han accedido a un contenido. Este sistema no sería una vista propia, sino una envoltura a una vista ya existente de la plataforma Edueca donde el profesor ve y gestiona el contenido de su curso.

4. **Enlace «Lista detallada de alumnos»/«Auditorías»**. Edueca cuenta con una interfaz

de búsqueda de auditorias en la que el profesor, especificando un rango de fechas y a un alumno, puede ver su actividad en cada sección del aula en ese período. La vista detallada de alumnos podría estar hiperenlazada con una búsqueda específica en esta vista.

5. **Lista detallada de módulos**. Una vista en la que el profesor pueda ver representado «qué le pasa a cada módulo», análogamente a la propuesta 3. Estaría justificada mediante estadísticas sobre los accesos de los alumnos a dicho módulo.

6. **Cambio de estilo**. Modificar la *front-office* de Edueca para que cada bloque de contenido se identifique más separadamente del resto, y facilitar así la futura incorporación de un sistema que destaque visualmente unos bloques sobre otros.

7. **Reorganización**. Reorganizar la navegación de la página, subiendo un nivel jerárquico aquellas secciones relacionadas con las estadísticas o cualquier información no introducida directamente por el profesor.

8. **Análisis de dispersión**. Un diagrama de dispersión de dos o tres dimensiones que represente a todos los alumnos de un curso.

El objetivo de la estancia es estudiar la viabilidad de las propuestas planteadas, definir unos requisitos, e implementarlos. Este fin se puede desglosar en los siguientes subobjetivos, que cubren los aspectos tanto informáticos como de negocio:

- Diseñar una o varias vistas relacionadas entre sí, de carácter llamativo e inspiradas por los conceptos mencionados, que ofrezcan al profesor información no evidente sobre el estado del curso que imparte.

- Diseñar objetos y valores que encapsulen la información que deben proporcionar estas vistas.

- Recopilar la información de la base de datos que puede ser útil para el profesor y que habilite los prototipos de vistas diseñados.

- Diseñar tablas adicionales de la base de datos que harían falta para completar el servicio.

- Diseñar los procesamientos por los que debe pasar esta información para extraerse o guardarse.

- Recopilar los conceptos pedagógicos o de estadística de los que se quiere hacer uso.

- Redactar los términos, descripciones y consejos que permitan que el profesor entienda precisa y fácilmente la situación de un aspecto de su curso.

Además, forma parte del objetivo de la estancia determinar si es conveniente o no adquirir software de terceros para completar el producto. En caso de que sí, la rentabilidad de dicho software deberá ser estudiada junto al equipo de Cidet.

1.3. Descripción del proyecto

Para implementar el sistema, se usarán mayoritariamente tecnologías que ya son de uso frecuente en Cidet cuando se trata de proyectos similares. Para acceder y gestionar las bases de datos, se empleará MySQL Workbench. El sistema está actualmente asociado a tres bases de datos: explotación, desarrollo y emergencia. La base de datos de explotación es volcada periódicamente sobre la de desarrollo, de modo que esta segunda puede ser empleada para analizar los datos de forma estática. La base de datos de explotación puede ser usada para observar el comportamiento de los datos a la hora de realizar cambios en el sistema, preferiblemente a través de la plataforma por razones de seguridad.

El código que se empleará durante el desarrollo está en una rama que se almacena y ejecuta localmente, gestión mediada por el controlador de versiones Git. El IDE con el que se trabajará es PhpStorm de JetBrains. Cualquier navegador permite acceder a la ejecución local del sistema, basta con recargar para visualizar cualquier cambio mientras se edita el código.

El uso de estas tecnologías se rige bajo el patrón de diseño Modelo-Vista-Controlador, cuyo soporte es dado por el *framework* Symfony. Symfony emplea controladores y servicios escritos en PHP, gestiona el paso de información vista/controlador mediante su motor de plantillas Twig (para html) y se comunica con el modelo gracias a las librerías del mapeador objeto-relacional Doctrine.

Finalmente, el hardware local con el que se trabajará es un MacBook Air proporcionado por Cidet.

1.4. Estructura de la memoria

En el resto de la memoria, en primer lugar, se presenta el capítulo de la planificación del proyecto, que detalla por qué ha seguido una metodología ágil y qué metas se han perseguido. Además, se detallarán las tareas llevadas a cabo durante el seguimiento del proyecto.

Más adelante, en el capítulo del análisis del proyecto, se exponen tanto el análisis del área estadística en la que se centra este documento como el de las partes relevantes de la plataforma web en la que se encuentra, Edueca. También se detalla el diseño del proyecto.

El capítulo de implementación y pruebas se describe el funcionamiento a nivel interno del *framework* que da soporte al proyecto, y a continuación se muestra su estructura en diferentes niveles de detalle. Las pruebas que el sistema ha debido superar también están explicadas en este capítulo.

Por último, se da una valoración personal sobre la estancia en prácticas y el proyecto en su totalidad.

Capítulo 2

Planificación del proyecto

El proyecto está planteado como adición a un flujo ya existente de integración continua. Como tal, la plataforma web Edueca está sujeta a constantes actualizaciones por parte de los desarrolladores del equipo de Cidet, lo que conlleva una serie de prácticas de trabajo comunicativas, tanto interna como externamente, y dosificadas en cuanto a volumen y tiempo de entregas.

Esta forma de trabajo y las circunstancias iniciales del proyecto me permiten asentar sus bases metodológicas, que a continuación describo junto a otros detalles de planificación.

2.1. Metodología

Inicialmente, los objetivos del proyecto están definidos a nivel estratégico o de negocio. De ellos se infiere, por un lado, que los requisitos del proyecto deben estar sujetos a la meta final o *Epic* «implementar un sistema que informe sobre un curso en la plataforma Edueca», que puede ser desglosada tal que así:

- El sistema debe preparar información útil para los clientes, potencialmente profesores, sobre el curso que administran.
- El sistema debe sintetizar y exponer esta información de forma naturalmente legible y atractiva
- El sistema debe incluir un componente dinámico de asesoramiento en cuanto a enseñanza, es decir, ofrecer perspectivas derivadas de las circunstancias específicas de un curso.

Por otro lado, que ya que los objetivos del proyecto no definen restrictivamente los requisitos, dicha definición se realiza progresivamente mediante la validación de los resultados obtenidos. Se evalúa si las implementaciones de los requisitos encajan exitosamente en la plataforma, tanto funcionalmente, como en lo que se espera a nivel de negocio. Finalmente, la salida de esta evaluación determina una posible actualización de los requisitos. Ya que tal proceso iterativo

no se puede llevar a cabo en una metodología tradicional, el proyecto se realizará siguiendo una metodología ágil. Otras razones por las que este tipo de metodología es más conveniente en el contexto del proyecto son:

- La forma de obtener los requisitos iniciales, por la cual es vital que puedan ser actualizados a medida que se aclaran restricciones, opciones y necesidades.
- Cómo estas aclaraciones sirven para proponer y poner en práctica nuevas ideas, que a su vez dan lugar a nueva información en un proceso retroalimentativo.
- La expectativa de definir requisitos de cada vez más bajo nivel a medida que se van proponiendo y validando funcionalidades sea parte del desarrollo.
- El hecho de que la comunicación habitual de los avances y las reuniones en grupo ya forman parte del desempeño laboral en Cidet.

Los requisitos iniciales se deben obtener contraponiendo las posibilidades de la plataforma y del proyecto a las propuestas de la primera reunión. De ahí en adelante, las evaluaciones sirven como punto de partida para revisar los requisitos y actualizar las implementaciones en un proceso iterativo en el que, además, se dan otras reuniones periódicas con el equipo de Cidet. En estas iteraciones, según sea necesario:

1. Se investiga qué información almacenada en la base de datos es relevante.
2. Se analiza y documenta el sistema actual, incluyendo las funcionalidades del proyecto y su estado, si su funcionamiento es correcto y si requieren algún cambio.
3. Se plantean, elaboran y actualizan los requisitos del sistema deseado.
4. Se verifican y validan los requisitos en función la información y documentación disponible.
5. Se implementan o actualizan las funciones que habiliten el cumplimiento de los requisitos.
6. Se prototipan las interfaces necesarias correspondientes a estas funciones.
7. Se implementan o actualizan las interfaces.
8. Se elaboran baterías de pruebas necesarias para verificar funciones e interfaces.

2.2. Planificación

Cada iteración durará aproximadamente 2 semanas. Se espera que hayan 8 iteraciones, abarcando desde el 02/11/2021 hasta el 24/02/2021, descartando las pausas por festivos. Cabe tener en cuenta que los primeros días de la estancia están parcialmente destinados a familiarizarse con el supersistema Edueca (tanto a nivel de usuario como de desarrollador) y con el distinto software de trabajo: el IDE PhpStorm, el gestor de bases de datos MySQL Workbench, las DevTools de Chrome, y los distintos lenguajes de programación y técnicas propios del *framework* Symfony. El cuadro 2.1 muestra el periodo correspondiente a cada iteración.

Iteración	Periodo
1	De 02/11/2021 a 15/11/2021
2	De 16/11/2021 a 29/11/2021
3	De 30/11/2021 a 13/12/2021
4	De 14/12/2021 a 23/12/2021, y 10/01/2022
5	De 11/01/2022 a 24/01/2022
6	De 25/01/2022 a 07/02/2022
7	De 08/02/2022 a 21/02/2022
8	De 22/02/2022 hasta acabar la estancia

Cuadro 2.1: Calendario de iteraciones

2.3. Gestión de riesgos

Durante el desarrollo del proyecto y tras su lanzamiento, ciertas medidas deben ser tomadas para remediar los hechos que más adelante describo como impactos de riesgo. Los riesgos descritos lo son en cuanto a que comprometen la calidad final del producto deseado, el sistema estadístico (riesgo de producto) u obstaculizan su desarrollo en lo que probablemente supone una penalización temporal o económica (riesgo de proyecto).

El análisis FODA del cuadro 2.2 describe la situación de la que parte el proyecto. En él, las fortalezas y debilidades se refieren a la internalidad del proyecto: sus recursos y la situación del desarrollador; mientras tanto, las oportunidades y amenazas hacen referencia al entorno que, tanto durante el desarrollo como en la explotación, afectará al proyecto [1].

Dicho análisis contiene las claves de los riesgos a los que se enfrenta el proyecto, inferidos en el cuadro 2.3, así como de los tratamientos que cada uno debe recibir, incluido en el cuadro 2.4.

Id	Descripción	Tipo
01	Fallo al emplear las tecnologías	Proyecto
02	Empleo poco convincente de recursos visuales	Producto
03	Incompatibilidad con los cambios de otros desarrolladores	Proyecto
04	Funcionamiento inesperado en explotación	Producto y proyecto
05	Impedimento de trabajo por causas externas	Proyecto
06	Falta de utilidad encontrada por los usuarios	Producto
07	Dificultad de los usuarios para emplear el sistema	Producto
08	Decisiones no acertadas de diseño	Producto

Cuadro 2.3: Identificación de riesgos

Fortalezas

- Tiempo: el proyecto abarca un periodo de desarrollo mayor del que cuesta en el mejor de los casos implementar un producto mínimo de estas características.
- Base: el proyecto reside en una plataforma que lleva años adentrada en su flujo de integración continua. Está provista de ejemplos extensivos y robustos de las tecnologías afines a Symfony con las que el proyecto debe realizarse.
- Framework: Symfony se encuentra entre los frameworks PHP mejor valorados, y consta de numerosos casos estudiados de éxito [2].
- Documentación general: además, este *framework* ofrece extensiva documentación gratuita [3].

Debilidades

- Documentación específica: no se dispone de documentación acerca del supersistema dentro del cual el producto debe integrarse.
- Curva de aprendizaje del *framework*: la pericia en el *framework* queda más allá del alcance temporal del proyecto.
- Experiencia del desarrollador: poca en ciertos ámbitos como desarrollar sistemas reactivos a entradas masivas de datos.

Oportunidades

- Público: la plataforma Edueca ya consta de un público provisto de clientes a nivel internacional y en aumento.
- *Feedback*: Cidet es receptivo ante las experiencias de los usuarios en la plataforma y su diseño se beneficia del conocimiento de las mismas.
- Proveedores: la posible demanda de bibliotecas y recursos de software de representación gráfica está cubierta por numerosos proveedores en línea, tales como Hightcharts [4].

Amenazas

- Edad del público: una parte sustancial de los proyectos educativos en la plataforma están enfocados a enseñanza de adultos o mayores, cuyo fallo a la hora de emplear herramientas informáticas tiende a ser mayor.
- Situación pandémica: el proyecto está comprendido en el periodo [2021 - 2022], en el que una situación de pandemia reduce la estabilidad y certidumbre de ciertos factores humanos y laborales.
- Competencia: las plataformas de aprendizaje ganan cada vez más popularidad en el mercado en linea, generando un público menos impresionable. Algunas de ellas, como Moodle, tienen una posición estable en el mercado.

Cuadro 2.2: Análisis FODA del proyecto

Id	Impacto	Prevención	Respuesta	Tipo
01	Las interfaces implementadas no se comporten como deben o los datos que muestran no son los esperados.	Leer detenidamente la documentación de Symfony sobre cada aspecto del proyecto y consultar al supervisor si es necesario.	Reservar un margen de una o más iteraciones para enmendar el problema.	M
02	La forma en que los datos se muestran o manipulan a través de los recursos visuales no es adecuada o intuitiva.	Ver demostraciones profesionales de diferentes tipos de recursos visuales estadísticos, ya sea de los proveedores originales o en otras plataformas de aprendizaje digital.	Adquirir e implementar dichos recursos, estudiando la rentabilidad que esto conlleva y adaptando los requisitos del proyecto a sus posibilidades.	MT
03	Hay superposición de código alterado entre dos o más desarrolladores, o la ejecución de dicho código provoca inconsistencias en los datos.	Descargar frecuentemente las actualizaciones realizadas por otros desarrolladores y dedicarse periódicamente a resolver conflictos emergidos.	Estudiar e implementar los mejores compromisos a los que se puede llegar entre las funcionalidades conflictivas.	ME
04	La variedad o el tamaño de los datos generados diariamente desde la plataforma destapa errores del sistema o problemas de rendimiento hasta entonces indetectados.	Intentar maximizar los escenarios cubiertos por las pruebas del sistema y minimizar el orden espacial y temporal de las funciones implementadas.	Según la gravedad del fallo, podría asumirse; si los datos no reflejan al menos orientativamente la realidad o los tiempos de carga son inadmisibles en virtud de la experiencia del usuario, se deberá invertir tiempo arreglando los problemas.	MA
05	El desarrollador está en condiciones de trabajar, pero no se puede asistir al lugar de trabajo por necesidad de aislamiento o otras causas médicas.	Respetar las medidas de seguridad vigentes durante el estado de alarma.	Habilitar el teletrabajo y emplear unos días a adaptar el desarrollo a las nuevas restricciones.	M
06	No se detecta uso de las herramientas por parte de los usuarios.	Diseñar el sistema de forma que las herramientas estén hiperenlazadas allá donde sea más útil.	Parametrizar las herramientas para que otros desarrolladores puedan mejorar su utilizad una vez integradas en el proyecto y su uso general proporcione conclusiones.	T

Id	Impacto	Prevención	Respuesta	Tipo
07	Los usuarios no logran sacar partido de las funcionalidades del sistema que requieren input de su parte.	Limitar la complejidad de las interacciones que el sistema puede llegar a requerir por parte del usuario, y proveerlo de una configuración por defecto para que, en el mejor de los casos, pueda beneficiarse de él sin hacer nada.	Diseñar un sistema con bajo acoplamiento para que, de ser problemática o redundante alguna funcionalidad, esta pudiese ser retirada o reemplazada sin vulnerarlo.	ET
08	El sistema no es efectivo a nivel pedagógico; no está diseñado con nociones correctas de negocio.	Pedir frecuentemente el visto bueno de los supervisores o remediar los problemas que detectan como expertos de negocio.	Dada la prevención, se entiende que de haber inconveniencias en el sistema final, estas serían mínimas, y por tanto asumibles o fácilmente remediables.	CA

Cuadro 2.4: Consecuencias, prevención y respuesta a cada riesgo

En dicho análisis se muestra que cada riesgo merece, según su naturaleza, una respuesta de diferente tipo. La siguiente leyenda los describe [5]:

- M - Mitigar: el caso más frecuente. Generalmente, se trata de invertir tiempo en arreglar o contener el problema una vez se ha manifestado. Para que sea viable, se cuenta con margen de tiempo al final del proyecto destinado a este tipo de remedios.
- T - Transferir: preparar el sistema para que otros desarrolladores, en circunstancias de más información disponible, puedan solucionar el problema, o gestionar la adquisición de código manufacturado por terceros en virtud de los requisitos del proyecto.
- E - Evitar: descartar una funcionalidad si su coste de reparación resulta superar su potencial beneficio, y su fallo actual se considera inadmisible.
- A - Aceptar: conservar una funcionalidad problemática si su coste de reparación supera su potencial beneficio, y su fallo actual se considera admisible.

2.4. Estimación de recursos según primera concreción de metas

Los tipos de recursos invertidos en el proyecto son: de esfuerzo humano, de hardware, y de software. A continuación se desarrolla la estimación de costes para cada uno de estos recursos.

Para estimar los costes de esfuerzo humano, se aplica la técnica COCOMO (de *Constructive Cost Model*) en su variante básica. La premisa de COCOMO es que los diferentes costes (humano y temporal) de un proyecto de software crecen en función de su tipo y envergadura de forma

generalmente similar. Como resultado de un análisis estadístico realizado sobre 63 proyectos de software del mundo real, COCOMO proporciona estas funciones, y las variables y constantes que incluyen.

En primer lugar, el esfuerzo necesario en personal-meses para llevar a cabo un proyecto de software tiene la forma $E(K) = a \times K^b$, donde K son los miles de líneas de código a ser redactados y a y b son constantes. A su vez, el tiempo de desarrollo en función del esfuerzo tiene la forma $T(E) = c \times E^d$, donde las constantes son c y d.

Estas constantes son unas u otras dependiendo del tipo de proyecto, que puede ser orgánico, semiencajado o empotrado. El proyecto descrito en esta memoria es de tipo orgánico, esto es: un producto de negocio sencillo, realizado por pocos programadores experimentados en otros proyectos del mismo estilo. El cuadro 2.5 muestra las constantes de un proyecto de tipo orgánico.

Constante	a	b	c	d
Valor	2,4	1,05	2,5	0,38

Cuadro 2.5: Constantes COCOMO en un proyecto orgánico

La cantidad de miles de líneas de código escritas en el desarrollo de proyecto depende de cuántas de las funcionalidades planteadas se implementen. Observando el código de otros módulos ya implementados del supersistema Edueca, se puede concluir que para cada una de ellas, hacen falta al menos:

- Un controlador PHP (de aproximadamente 100 líneas).
- Una vista Twig (de aproximadamente 500 líneas).
- Un test PHP (de aproximadamente 100 líneas).

Algunas funcionalidades requieren componentes adicionales:

- Entidades del modelo PHP (de aproximadamente 100 líneas): para encapsular información necesaria para dar un servicio (como almacenar configuraciones del usuario) que no se encuentra actualmente en la base de datos.
- Un DAO (objeto de acceso a datos) PHP/MySQL (de aproximadamente 100 líneas): necesario si los datos a extraer o guardar requieren consultas complejas, y redundante en cualquier otro caso gracias al mapeador objeto-relacional Doctrine, incorporado en el *framework* Symfony.

Como se ha explicado anteriormente, es parte del proyecto analizar el supersistema Edueca para determinar requisitos habiendo comprobado su viabilidad. Este análisis se detalla en el seguimiento del proyecto.

Para el final de la tercera iteración, 3 funcionalidades planteadas al principio de la estancia (ver figura 1.1 y consiguiente lista) se consideran viables y aptas para empezar proceso de implementación. Se trata de las ideas 2, 3 y 8, reiteradas a continuación:

- Una lista de alumnos, su estado, y valores estimadores sobre el alumno que avalen dicho estado. Requiere de una entidad del modelo que almacene estos estados, y un DAO donde estuviesen los cálculos necesarios para obtener los estimadores.
- Un diagrama de dispersión donde ver representados a los alumnos. Requiere de una entidad del modelo que almacene configuraciones asociadas al diagrama, y un DAO donde estuviesen los cálculos necesarios para posicionar a los alumnos.
- Una extensión de la actual vista *front-office* del curso que permita ver qué alumnos han visitado recientemente cada sección. No debería requerir entidades ni objetos de acceso, ya que emplearía un sistema de auditorías ya incorporado en la base de datos de Edueca.

En suma, el cuadro 2.6 muestra que la envergadura del proyecto está sobre las 2.200 líneas de código. Si aplicamos COCOMO para calcular el esfuerzo en personal-meses, obtenemos:

$$E(2{,}2) = 2{,}4 \times 2{,}2^{1{,}05} \approx 5{,}5pm$$

Obtengo así un esfuerzo de 5,5 personal-meses, que puede ser equivalente a una persona durante 5 meses y medio o, según el ajuste de COCOMO:

$$T(5{,}5) = 2{,}5 \times 5{,}5^{0{,}38} \approx 4{,}7m$$
$$\frac{5,5pm}{4,7m} \approx 1,15p$$

1,15 personas (o una persona asistida por otra) en 4,7 meses. En un caso no limitado a las 300 horas de la estancia en prácticas, esto tendría el coste de dicha fracción de un salario anual de desarrollador *full stack* junior (21.532 €), es decir, 8.433€ [6].

Ante la limitación de tiempo a unas 300 horas, distribuidas en unos 3 meses con semanas de 4 días laborales, el tiempo necesario para alcanzar todos los requisitos podría exceder el disponible. Esto es válido debido a la flexibilidad de objetivos de bajo nivel de la que parte el proyecto, siempre que el producto final no quede comprometido (funcionalidades a medias).

Para evitar esto, las funcionalidades del proyecto deben implementarse secuencialmente, de forma que cada entrega garantice un producto entero y expansible por futuras entregas. En las últimas dos iteraciones, cualquier funcionalidad no empezada se descarta en virtud de optimizar el producto existente a partir del *feedback* recibido y el conocimiento de los fallos destapados por pruebas.

La secuencia escogida para implementar las funcionalidades, por motivos que se explicarán en el seguimiento del proyecto, es la siguiente:

1. Análisis de dispersión.
2. Lista detallada de alumnos.

Componente	Cantidad	Líneas de código
Controlador	3	300
Vista	3	1.500
Entidades	2	200
DAOs	2	200
Total		**2.200**

Cuadro 2.6: Desglose de líneas de código estimadas

Nombre	Descripción
Droplets	Máquinas virtuales
App Platform	Plataforma como servicio
Spaces Object Storage	Gestión de archivos
Load Balancers	Distribuidor de carga entre servidores
Managed Kubernetes	Plataforma de contenedores y microservicios
Managed Databases	Bases de datos
Volumes	Memoria en disco
Container Registry	Imágenes de código ejecutable

Cuadro 2.7: Planes o productos de DigitalOcean

3. Sistema de vistos.

En un caso remunerado bajo las mismas circunstancias de flexibilidad de objetivos, el coste correspondería al salario de 3 meses con cuatro días por semana, equivalente a $3 \times \frac{4}{5} = 2{,}4$ salarios con semanas completas. Partiendo nuevamente del salario anual de un desarrollador *full stack*, la cifra queda en 4.306€.

El resto de costes están derivados del equipo empleado para realizar el proyecto (software y hardware de desarrollo) y del servidor online.

Todo el software empleado es de código abierto. PhpStorm, como otros programas del repertorio de JetBrains, puede ser descargado y empleado gratuitamente [7]. Sucede similarmente con el gestor de bases de datos MySQL Workbench y con los navegadores empleados: primariamente Chrome, Opera y Firefox. El *framework* Symfony es de código abierto y sus diferentes versiones son de acceso público, siendo el único requisito instalar las librerías que el *framework* proporciona.

DigitalOcean, el servicio contratado para almacenar la base de datos y la lógica del servidor, ofrece productos en la nube de diferentes índoles [8], los cuales están mostrados en el cuadro 2.7. En el caso de Edueca, se emplean los productos mostrados en el cuadro 2.8

Para estimar qué ponderación de ese precio corresponde al área estadística, es necesario calcular qué fracción de cada recurso es empleada.

En los Droplets reside la lógica del servidor. El propio código tiene un tamaño ínfimo com-

[1]Tres droplets cuya función tenga relación con el área estadística, puesto que sirven como CPUs de propósito general en la plataforma. Hay cinco en total, siendo los dos adicionales para gestión de videollamadas y correo.

Nombre	Cantidad	Precio mensual
Managed Databases	1	280€
Load balancers	1	30€
Volumes	1	10€
Droplets	3[1]	120€
Total		**440€**

Cuadro 2.8: Productos de DigitalOceans empleados por Edueca

Concepto	Coste	Porción	Coste ponderado	Frecuencia
Coste humano	4.306€	1	4.306€	Una vez
Droplets	120€	1/26	5€	Cada mes
Load balancers	30€	3/130	1€	Cada mes
Managed databases	280€	1/64	4€	Cada mes
Hardware	1.129€	1/20	56€	Una vez

Cuadro 2.9: Detalle de costes del proyecto

parado con el del resto del supersistema, de modo que el recurso asociado al almacenamiento de dicho código se puede desestimar. El tiempo de CPU que consume la ejecución de dicho código, por otro lado, puede asociarse a la cantidad de peticiones que recibe. Dado que todas las peticiones se resuelven en un orden temporal similar (el de milisegundos), una métrica válida son las veces que el usuario entra a consultar el área estadística, en relación al resto de áreas. En primer lugar, el área estadística forma parte de la *back-office*, que representa solamente un 50 % de la carga en peticiones de la página. De esta *back-office*, el área estadística planeada ocupa dos secciones de un total de trece. Considerando un uso homogéneo de la plataforma, esto conlleva un 26avo de las peticiones del cliente.

Esta fracción es la misma que el tiempo de CPU que los balanceadores consumen atendiendo a los tres droplets contabilizados. Los balanceadores se encargan de distribuir la carga entre servidores para que no se saturen, y todos las actividades pueden detonar la necesidad de que este servicio se active. Por tanto, contabilizando los cinco droplets, el área estadística consume $\frac{3}{5\times 26} = \frac{3}{130}$ partes del tiempo de CPU de los balanceadores.

En la base de datos, el área estadística requeriría de dos entidades (o tablas), contra 127 ya existentes. En otras palabras, el área estadística representa como máximo un 64avo del almacenamiento en base de datos. Este cálculo asume que el tamaño de las tablas pertenecientes al área estadística (correspondientes a las configuraciones del usuario en cada funcionalidad) difícilmente superaría el tamaño promedio de una tabla de la base de datos de Edueca, ya que por defecto, el sistema estadístico no requiere una configuración expresamente introducida por el usuario para funcionar. Por tanto, para la gran mayoría de usuarios no habrían filas existentes en las tablas correspondientes al área estadística.

En cuanto al hardware empleado, se trata de un MacBook Air, cuyo precio de mercado es de 1.129€ [9]. Teniendo estos dispositivos una durabilidad media de 5 años [10], y habiéndose empleado para el desarrollo del sistema estadístico durante 3 meses, la fracción de su coste correspondiente al desarrollo del proyecto sería de $\frac{3}{5\times 12} = \frac{1}{20}$.

El cuadro 2.9 muestra el resumen de estos costes.

Frecuencia	Total
Una vez	4.362€
Cada mes	10€

Cuadro 2.10: Costes totales

En el cuadro 2.10, estos costes están agrupados por frecuencia. Teniendo en cuenta la duración del proyecto, 3 meses, el coste total es de 4.392€.

2.5. Seguimiento del proyecto

El proyecto ha seguido un desarrollo basado en 8 iteraciones, el periodo de cada una de las cuales está mostrado en el cuadro 2.1 de la planificación. Asimismo, en el apartado de la metodología se especifica una secuenciación de ocho tipos de tarea. El fin de cada iteración supone un punto en el que revisar si se resuelven satisfactoriamente y en orden, o si alguno es motivo de atoramiento, requiere especial atención o ser saltado, etc. Estos tipos de tarea están resumidos en el cuadro 2.11.

Id	Tipo de tarea
1	Investigación de la base de datos
2	Documentación del sistema
3	Identificación de requisitos
4	Verificación y validación de requisitos
5	Implementación de funciones del controlador
6	Prototipado de vistas
7	Implementación de vistas
8	Elaboración de pruebas

Cuadro 2.11: Resumen de tipos de tarea

A continuación se resumen los hitos más importantes que han tenido lugar en cada iteración.

2.5.1. Iteración 1

Uno de los objetivos principales de la primera iteración fue familiarizarse con los procesos de trabajo desempeñados en Cidet. Tras recibir nociones sobre el sistema de control de versiones empleado, se contribuyó en el proceso de correcta instalación de las herramientas y librerías de software necesarias para trabajar sobre el *framework* Symfony. Estas debían ser compatibles con una rama actual del proyecto y con el sistema operativo empleado en la empresa, macOS.

De forma simultánea a este proceso, que abarcó la primera semana, se realizaron numerosos recorridos a través de la plataforma Edueca, probando las opciones disponibles, entendiendo los resultados y compartiendo impresiones a nivel de experiencia de usuario con el equipo. Durante este proceso, se comprendió lo que Cidet buscaba a un nivel elevado de funcionalidad.

Tuvo lugar el primer *brainstorming*, cuyo resultado detallan la figura 1.1 y la posterior lista, donde se elaboraron junto al equipo de Cidet diferentes propuestas de sistemas con los que afrontar las metas a nivel estratégico del proyecto.

Por último, se aprendieron las bases del *framework* Symfony, sobre el que la plataforma Edueca está construido, y las herramientas que Cidet emplea para interactuar con él: el IDE PhpStorm, el gestor de bases de datos MySQL Workbench, y las funciones de análisis que el propio *framework* Symfony empotra en el navegador cuando se ejecuta una versión de desarrollo de la plataforma (el *profiler*).

Id	**Tipo de tarea**
1	Investigación de la base de datos

Cuadro 2.12: Tipos de tarea de la primera iteración

2.5.2. Iteración 2

Se decidió realizar una *spike* con el objetivo de poner en práctica y ampliar el conocimiento adquirido sobre el *framework* Symfony y el entorno Edueca. Para ello, se conceptualizaron e implementaron requisitos consistentes en implementar un sistema que mostrara las notas media y máxima de los cuestionarios de un curso de Edueca, y que permitiera filtrar por lección.

La *spike* fue terminada satisfactoriamente, y durante la investigación del supersistema, se elaboró documentación en forma de diversos diagramas de clases sobre la base de datos de Edueca. Estos diagramas abarcaban las tablas accedidas por funciones relacionadas con el proyecto final.

Posteriormente, se escogieron junto con el equipo de Cidet las propuestas a partir de las cuales empezaría el desarrollo iterativo del proyecto. Dichas fueron la 2, 3 y 8, tal como están representadas en los objetivos (ver figura 1.1 y posterior lista):

- **2. Lista detallada de alumnos**. Una vista en la que el profesor pueda ver representado «qué le pasa» a cada alumno, justificada mediante estadísticas extraídas del uso de la plataforma por parte del alumno.
- **3. Sistema de vistos**. Un sistema de «vistos» que, similarmente a cómo Google Drive informa de qué usuarios tienen abierto un documento, muestre al profesor qué alumnos han accedido a un contenido.
- **8. Análisis de dispersión**. Un diagrama de dispersión de dos o tres dimensiones que represente a todos los alumnos de un curso.

Se decidió que, para facilitar el proceso de prueba de las siguientes funcionalidades implementadas, se elaboraría además en la siguiente iteración una vista de tipo formulario para insertar objetos de cualquier clase. Esta decisión fue tomada debido a las formas alternativas de insertar datos en el sistema; a través de su interfaz o a través de la base de datos, ambas teniendo inconvenientes:

- A través de la interfaz, el proceso de generación de entidades era inadmisiblemente lento, incluyendo registrar nuevos usuarios en algunas ocasiones.
- A través de la base de datos, el resultado podía ser inconsistente, ya que podían estar no respetándose algunas reglas de integridad definidas a nivel del servidor (por el mapeador objeto-relacional Doctrine).

Id	Tipo de tarea
1	Investigación de la base de datos
2	Documentación del sistema
3	Identificación de requisitos

Cuadro 2.13: Tipos de tarea de la segunda iteración

Observaciones

Antes de definir los requisitos del proyecto final, se definieron los requisitos relacionados con la *spike* y el formulario de inserción de datos.

2.5.3. Iteración 3

Durante la tercera iteración, se definieron los requisitos del proyecto. Fueron concretados intentando que el sistema resultante aprovechara las características de las tres propuestas escogidas de forma cohesiva. A continuación, se listan acompañados de la propuesta a la que cada uno hace referencia:

- El sistema obtendrá a partir de dos aspectos de cada alumno (su rango de notas, de entre n rangos existentes, y su rango de participación en la plataforma, de entre p rangos existentes) una clasificación de entre $n \times p$ posibilidades **(2 y 8)**.
- Estos aspectos serán representados mediante un diagrama de dispersión de dos dimensiones, que contendrá la representación de cada alumno y los rangos correspondientes a cada clasificación **(8)**.
- El profesor podrá ver a todos los alumnos de un curso representados por cartas. Estas cartas contendrán, de cada alumno: imagen de perfil, nombre, un calificativo asignado automáticamente según la clasificación y el desglose de los dos aspectos que la justifican **(2)**.
- Desde el diagrama de dispersión, el profesor podrá añadir, eliminar o editar los rangos que delimitan las clasificaciones **(8)**.
- El sistema tendrá preparadas varias descripciones en lenguaje natural para lo que sucede en el diagrama. Por ejemplo: alta correlación, baja correlación, notas generalmente altas, participación generalmente baja, etc., y las implicaciones que estos casos conllevan **(8)**.

- El profesor podrá editar las descripciones o calificativos asignados a cada clasificación **(8)**.
- Las interfaces «Análisis de dispersión», «Vista detallada de alumnos» y «Sistema de vistos» estarán mutuamente hiperenlazadas **(2, 3 y 8)**.

Tras esta definición de requisitos, la vista «Análisis de dispersión» quedaba ampliada a dos apartados: en primer lugar, el diagrama de dispersión, y en segundo lugar, los bloques de texto que expondrían la situación del curso, y donde Cidet insertaría nociones y consejos apropiados para diferentes circunstancias.

Ante ciertas ambigüedades que planteaban estos requisitos, surgieron las siguientes preguntas de validación, las cuales deberían ser afrontadas en la siguiente iteración:

- ¿Son adecuadas las dimensiones planteadas para medir el estado de un alumno?
- ¿Qué dominios debería tener cada dimensión de la dispersión?
- ¿Es conveniente que los dominios se ajusten a cada curso?
- ¿Cómo debería funcionar el ajuste de rangos?
- ¿Qué forma de calcular los valores que describen al alumno es útil para el profesor?
- ¿Cómo se debería medir la participación?
- ¿Cuándo es relevante que un alumno haya 'participado' en una sección?
- ¿Qué 'síntomas' de un hecho sobre el alumnado de un curso pueden verse reflejados en una dispersión?
- ¿Cómo es práctico delimitar estos síntomas?

Se implementó, tanto a nivel de controlador como visual, el formulario de pruebas planteado en la anterior iteración (ver figura 2.1). Dicho formulario recibe en su constructor una lista de las entidades para las que debe presentar campos de inserción. Así, su uso es dinámico; si se quiere trabajar objetos de una nueva entidad, generalmente basta con añadirlo en este constructor y recargar.

Por último, se realizó el primer prototipado de las vistas correspondientes a los requisitos planteados. La figura 2.2 muestra el prototipo de la vista «Análisis de dispersión».

Id	**Tipo de tarea**
3	Identificación de requisitos
4	Verificación y validación de requisitos
5	Implementación de funciones del controlador
6	Prototipado de vistas

Cuadro 2.14: Tipos de tarea de la tercera iteración

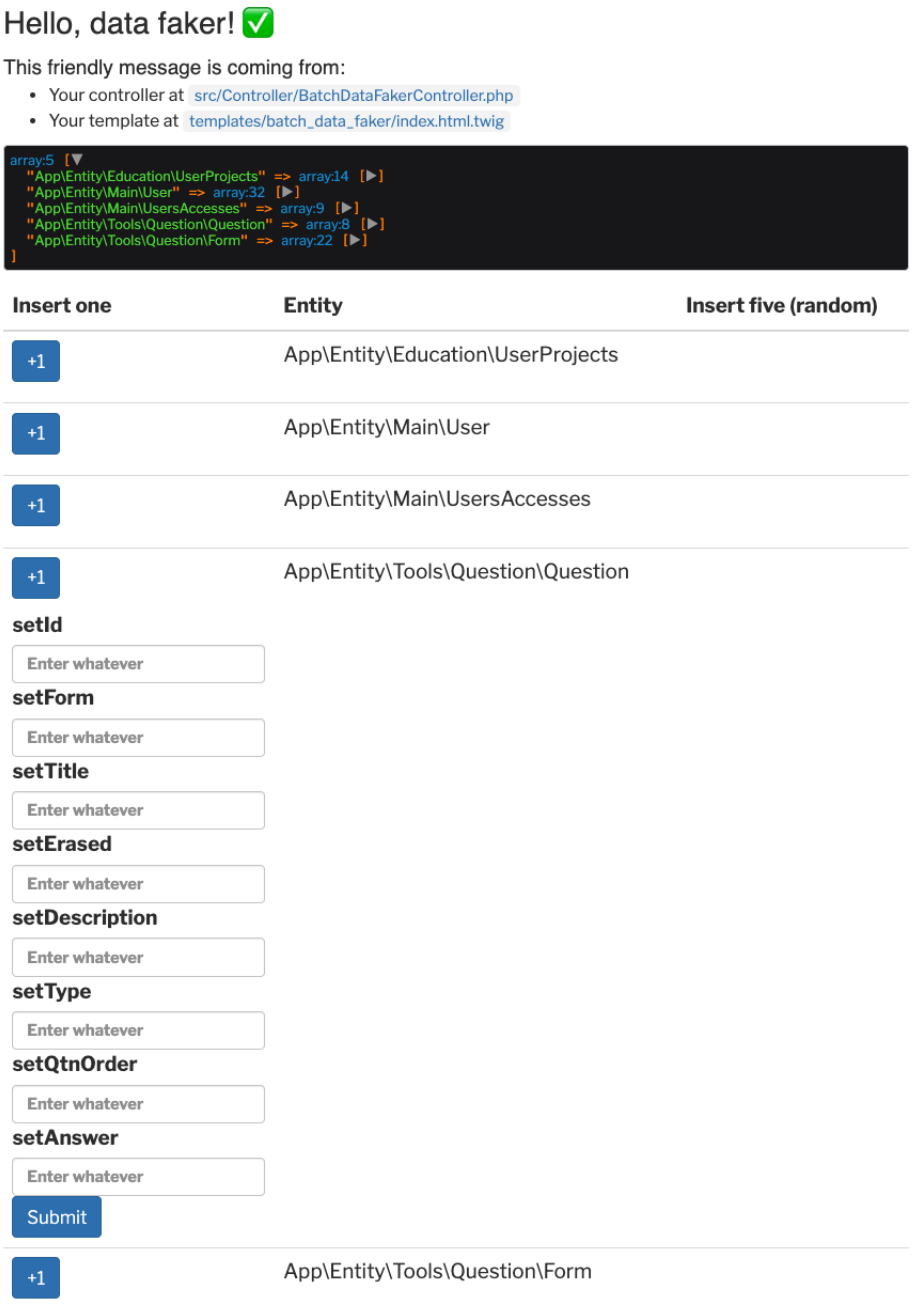

Figura 2.1: Formulario para crear objetos de prueba para desarrolladores

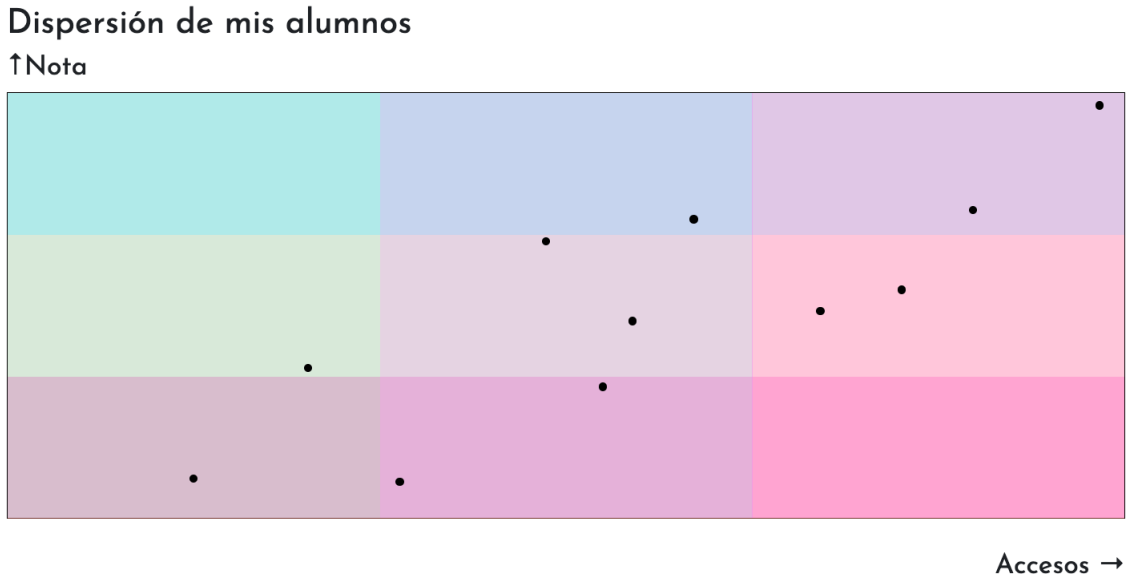

Figura 2.2: Concepto de la vista «Análisis de dispersión»

Observaciones

Antes de definir los requisitos del proyecto final, se definieron los requisitos relacionados con el formulario de inserción de datos.

En cuanto a las cuestiones de validación, se decidió que su resolución se prolongaría hasta la siguiente iteración, dedicando más tiempo a estudiar el sistema Edueca a nivel de negocio.

2.5.4. Iteración 4

En la cuarta iteración, se respondieron las cuestiones de validación planteadas, aclarando así aspectos del sistema que la definición de requisitos dejaba al descubierto.

- **¿Son adecuadas las dimensiones planteadas para medir el estado de un alumno?**
  El eje y, la nota, es un estimador habitual para evaluar al alumno en cualquier proyecto educativo. Ya que Edueca ya implementa un sistema de cálculo de notas basado en cuestionarios, emplearlo aquí aumenta la cohesión de la plataforma.

  El eje x, la participación, refleja de forma aproximada la dedicación que el alumno invierte en el curso. Su cálculo debe ser parametrizable y, por tanto, fácil de calibrar desde el controlador. Así queda facilitada la transferencia del sistema a otros desarrolladores que lo mantengan teniendo mayor conocimiento del uso que se le dé a la herramienta (ver cuadro 2.4 de la gestión de riesgos, riesgo 6).
- **¿Qué dominios debería tener cada dimensión de la dispersión?**
  Se han planteado dos dominios porcentuales, es decir, de 0 a 100. Tanto la nota como la participación tienen un valor máximo en relación al cual se obtiene el porcentaje.

- **¿Es conveniente que los dominios se ajusten a cada curso?**
  En el caso de las notas, no. Es común que la nota máxima sea 10 o 100, según el sistema, y no es prioritario que esto sea personalizable.

  En el caso de la participación, sí, ya que el contenido del aula virtual de cada curso afecta directamente a cómo el alumno puede participar en ella.

- **¿Qué tipo de ajuste de los rangos es conveniente (continuo, discreto)?**
  En principio, uno continuo, ya que es:

  1. Más fácil de implementar.
  2. Más capaz de adoptar dominios precisos para clasificar a los alumnos, si así lo desea el profesor.

  En cualquier caso, no se descarta el estudio de un ajuste discreto de rangos si el ajuste continuo diese problemas o resultase propenso a errores.

- **¿Qué forma de calcular los valores que describen al alumno es útil para el profesor?**
  Una que no conlleve una excesiva carga mental. La intención es dar una idea orientativa del rendimiento de cada alumno, lo cual se obtiene con dos variables fácilmente visualizables.

- **¿Cuándo es relevante que un alumno haya 'participado' en una sección?**
  Solo en ciertos casos. Un curso típico de Edueca consta de bloques que poseen diversas características en cuanto a interactividad:

Bloque	**Acciones realizables**
Foros	Añadir una entrada, contestar una entrada existente
Texto	Solo ver
Blog	Solo ver
Índice	Acceder al temario
Perfil	No procede
Comunidad	Acceder a comentarios que realizan otros miembros de grupos concretos, más allá de «alumnos de un curso»
Cuestionarios	Responder a las preguntas
Tareas	Entregar

Cuadro 2.15: Qué permite hacer cada herramienta

- **¿Cómo se debería medir la participación?**

  Mediante el siguiente algoritmo:

  - Un alumno empieza con un total de 0 puntos.
  - Obtiene w_0 por cada lección consultada.
  - Obtiene w_3 por responder al menos una vez a una entrada de foro creada por un profesor.
  - Obtiene w_4 por crear al menos una entrada de foro en un foro.
  - Obtiene w_5 por responder un cuestionario.

- Obtiene w_6 por realizar una tarea.

Este cálculo precisa de variables mágicas, los pesos w. Su valor por defecto será 1.

- **¿Qué 'síntomas' de un hecho sobre el alumnado de un curso pueden verse reflejados en una dispersión?**

Los siguientes:

Síntoma	**Hecho**
Hay una correlación lineal	Los contenidos favorecen el desempeño de los de los alumnos
No hay correlación lineal	Los contenidos no favorecen el desempeño de los alumnos
La dispersión se concentra arriba	Los alumnos tienen muy bien desempeño, o los cuestionarios son fáciles
La dispersión se concentra a la izquierda	Los alumnos prestan muy poca atención a los contenidos
La dispersión se concentra abajo	Los alumnos tienen mal desempeño, o los cuestionarios son difíciles
La dispersión se concentra a la derecha	Los alumnos dedican mucho a los contenidos

Cuadro 2.16: Conclusiones que pueden emerger del diagrama

- **¿Cómo es práctico delimitarlos?**

De forma sencilla; lo importante es que el profesor pueda personalizar estos límites. Por ejemplo, las notas podrían estar separadas en tres rangos:

- $[0, 5[$ (suspendido).
- $[5, 8[$ (aprobado).
- $[8, 10]$ (excelente).

Si se da el caso de que un profesor quiere distinguir entre aquellos alumnos que tienen nota inferior e igual o superior a 3, puede crear una nueva franja y situar su límite en el valor 3, o desplazar una franja existente.

Por último, comenzó la implementación de la vista «Análisis de dispersión», tanto a nivel de controlador como visual.

Id	**Tipo de tarea**
4	Verificación y validación de requisitos
5	Implementación de funciones del controlador
7	Implementación de vistas

Cuadro 2.17: Tipos de tarea de la cuarta iteración

Observaciones

Pese a que las cuestiones de validación se acabaron de resolver en esta iteración, los prototipos diseñados en la anterior eran vigentes, y por tanto, se conservaron. Como resultado, en esta iteración se pasó directamente de implementar los controladores a implementar las vistas.

2.5.5. Iteración 5

Durante la quinta iteración, se avanzó en la implementación de las vistas «Análisis de dispersión» y «Lista detallada de alumnos», habiendo cumplido aproximadamente la mitad de los requisitos.

Id	Tipo de tarea
5	Implementación de funciones del controlador
7	Implementación de vistas

Cuadro 2.18: Tipos de tarea de la quinta iteración

Observaciones

Esta iteración constituyó el grueso de código implementado. Durante la implementación de las vistas, se modificaron los controladores bajo ciertas circunstancias:

- Cuando la implementación de la vista demandó un formato más óptimo para los datos.
- Cuando el uso de las interfaces destapó errores provenientes del controlador.

2.5.6. Iteración 6

Durante la sexta iteración, se detectaron, mediante el uso de las interfaces implementadas, algunos problemas. La iteración fue dedicada enteramente a resolverlos, postergando la implementación de la vista «Sistema de vistos».

En primer lugar, las participaciones de algunos alumnos en un curso se registraban con valores incongruentes, mayores al 100 %. Esta observación se dio tras volcar una auditoría real de accesos a la plataforma sobre la base de datos de desarrollo, haciendo que el sistema diese resultados a partir de ella.

El problema resultó ser el siguiente: Edueca permite distribuir las herramientas de un curso (cuestionarios, enlaces, foros, etc.) en dos lugares: los módulos (lecciones del temario) y en la página frontal del curso. Si una herramienta es eliminada por el profesor, un valor asociado a ella llamado `erased`, nulo por defecto, debería pasar a contener la fecha de eliminación. Esto se cumplía para las herramientas pertenecientes a los módulos, pero no en aquellas pertenecientes a

la página frontal del curso. Por tanto, el sistema estaba dando por válidos accesos a herramientas ya no existentes. Este fallo quedaba fuera del alcance del proyecto en desarrollo, por lo que se informó de ello a los supervisores y estuvo arreglado poco tiempo después.

Otro problema encontrado fue que la aparente desaparición de algunas constantes durante el ciclo de vida de una página cargada. Algunas funcionalidades implementadas requerían que el controlador preservase ciertas constantes una vez acabada la comunicación con la vista, a la espera de ser empleadas bajo demanda tras la llegada de nuevas instrucciones. Se averiguó que en Symfony, dicho sistema es vulnerable, ya que los controladores se pueden reiniciar (y conservar solamente la lógica) una vez la página ha sido satisfactoriamente cargada. Por tanto, la información que no se puede solicitar a la base de datos, como los identificadores de las entidades, debe quedarse en el cliente del sistema.

Se trabajó durante unos días en el tratamiento de los datos para ajustarse a este requisito tecnológico. Este cambio, una vez realizado, presentaba otra necesidad: implantar medidas de seguridad que imposibilitasen un acceso indebido a la base de datos. El nuevo formato de las peticiones permitía que un cliente modificado con malas intenciones solicitase información a la que no debía tener acceso, por lo que era necesario añadir una capa adicional de seguridad en los controladores.

Id	**Tipo de tarea**
5	Implementación de funciones del controlador

Cuadro 2.19: Tipos de tarea de la sexta iteración

2.5.7. Iteración 7

En la séptima iteración, se arreglaron los problemas de seguridad mencionados. Para ello, se estudió una clase propia del sistema Edueca, `Security`, que el equipo de Cidet había desarrollado para solventar incidencias pasadas manifestadas por la misma causa.

Dicha clase se encarga de monitorizar el origen de cada petición y las características de la misma para asegurar que concuerdan. Cuando no lo hacen, el sistema emite una excepción y la solicitud no procede.

Tras realizar los cambios adecuados para hacer uso de la clase `Security`, comenzó la implementación de la parte de la vista «Análisis de dispersión» correspondiente a los bloques de análisis.

Dada la prolongación de la implementación de los sistemas «Análisis de dispersión» y «Lista detallada de alumnos», se reestudió la viabilidad de la funcionalidad restante, «Sistema de vistos».

Para ello, se estudió la página de Edueca sobre la que debía implementarse la envoltura «Sistema de vistos», la sección «Contenido» del *dashboard*. Debido al alto acoplamiento entre el controlador y la vista de dicha página, se estimó que implementar esta envoltura requeriría un tiempo superior al disponible en el resto de la estancia en prácticas. Por tanto, esta funcionalidad se descartó en virtud de emplear el tiempo restante para finalizar los dos sistemas que se

encontraban en avanzado estado de desarrollo, mejorarlos en función del *feedback* recibido, y realizar pruebas.

Id	Tipo de tarea
3	Identificación de requisitos
4	Verificación y validación de requisitos
5	Implementación de funciones del controlador

Cuadro 2.20: Tipos de tarea de la séptima iteración

Observaciones

En esta iteración hubo un cambio en los requisitos, pero al tratarse de una reducción de los mismos, no supuso un contratiempo en el proceso de implementación de funciones.

2.5.8. Iteración 8

En la iteración final, se terminó de implementar la parte de la vista «Análisis estadístico» correspondiente a los bloques de análisis.

El sistema recibió *feedback* por parte del equipo de Cidet, el cual queda sintetizado en los siguientes puntos:

- Las dos vistas creadas requerían un buscador o filtrador.
- La vista «Análisis de dispersión», además de las opciones que presentaba, debía contar con varias plantillas de configuraciones por defecto para que los usuarios pudieran establecer configuraciones con más facilidad.

Una vez estos requisitos fueron implementados, se dedicó el resto de la iteración a la elaboración de pruebas y depuración del sistema, hasta que todas las pruebas pasaron con éxito.

Id	Tipo de tarea
3	Identificación de requisitos
4	Verificación y validación de requisitos
5	Implementación de funciones del controlador
6	Prototipado de vistas
7	Implementación de vistas
8	Elaboración de pruebas

Cuadro 2.21: Tipos de tarea de la octava iteración

Observaciones

Para solventar la expansión de requisitos, esta iteración pasó de tener una duración acortada (desde el 22 de febrero hasta el 24 de febrero) a una duración de dos semanas como el resto de iteraciones. Esto permitió atravesar las etapas necesarias para que los nuevos requisitos fuesen satisfactoriamente cumplidos y las pruebas, superadas.

Capítulo 3

Análisis y diseño del sistema

En este capítulo, se realiza un modelado del área estadística, así como un modelado parcial del supersistema que la contiene, Edueca, que facilite la comprensión de las decisiones tomadas. Posteriormente, se detalla el diseño del área estadística y las decisiones tomadas durante él.

3.1. Proceso de definición de requisitos

El proceso de definición de requisitos, como se ha expuesto en el seguimiento, se realizó en las tres primeras iteraciones del proyecto. Para ello, se tuvieron en vista los objetivos a nivel de negocio y se plantearon y descartaron, según su viabilidad o conveniencia, requisitos basados en las propuestas del *brainstorming* inicial.

Uno de los objetivos era crear un sistema que contribuyese a la diferenciación entre Edueca y otras plataformas de aprendizaje digital. Desde Cidet se consideró que la propuesta «Lista detallada de alumnos» era la que mejor cumplía este propósito. El sistema no podía limitarse a exponer estimadores (valores escalares) sobre el desempeño del alumno, ya que esto no cubriría la necesidad a nivel de negocio de que el sistema ofreciese asistencia inteligente. Por tanto, tenía que proporcionar como mínimo un descriptor en lenguaje natural para cada alumno.

Se trató de idear sistemas que asociasen la propuesta elegida del *brainstorming* con las propuestas restantes. En cada caso, era necesario esclarecer con qué estimadores evaluar a los alumnos y qué categorías podían emerger de estos estimadores. A continuación se resumen los estimadores valorados:

- **Mapa de calor:** categorizar a los alumnos según el número de clics que hacen en las diferentes secciones del aula. Este sistema aportaría poca información; sería una versión compacta del ya existente mapa de calor y no proporcionaría un estimador relevante del desempeño del alumno.
- **Sistema de vistos / auditorías:** categorizar a los alumnos por el número de secciones a las que acceden. Este sistema aportaría de un vistazo información relevante que ya se

puede encontrar en Edueca, así como en la mayoría de plataformas de aprendizaje digital, de forma menos directa: la actividad desglosada de cada alumno.

- **Análisis de dispersión:** categorizar a los alumnos por la región a la que pertenecen en una dispersión bidimensional, que a su vez se obtiene mediante otros estimadores. Esta opción tiene varias ventajas que la hacen la más conveniente. Es potencialmente la que más información aporta, pudiendo incluir una de las magnitudes anteriormente citadas como eje; facilita la justificación de las categorías asignadas de cara al usuario e incluso habilita la configuración de las mismas de una forma visual, incentivando a los profesores a crear sus propias clasificaciones.

Ante las ventajas de la categorización por región en una dispersión bidimensional, se decidió definir los requisitos con base en tal propuesta, «Análisis de dispersión», la ya explicada propuesta «Lista detallada de alumnos», y la propuesta «Sistema de vistos».

En cuanto a los ejes de la dispersión, se decidió pronto que uno de ellos sería la nota media del alumno. Se valoró que el número de accesos al aula por parte del alumno fuera el otro eje, pero presentaba algunos inconvenientes:

- Dominio infinito: todos los números naturales. Difícil de representar.
- Volatilidad: puede aumentar rápidamente para un alumno tan solo con que algún problema técnico le obligue a recargar.
- Dificultad para probar en desarrollo: en la base de datos, la auditoría que contiene dichos accesos es una vista (una consulta de alta complejidad), no una tabla física o entidad del modelo. Por tanto, no se pueden insertar valores, ni directamente ni a través del formulario de inserción desarrollado durante las primeras iteraciones.

Así se introdujo el concepto de 'participación', una consulta que agruparía las distintas interacciones que puede haber entre un alumno y una sección del aula. Esta magnitud representaría el otro eje de la dispersión. Faltaba por cumplir el objetivo de negocio de que el sistema ofreciese consejos derivados de las circunstancias específicas de un curso. Para ello, el concepto de la vista «Análisis de dispersión» fue ampliado, incluyendo desde entonces un conjunto de bloques de texto donde Cidet podría plasmar sus conocimientos en el negocio en forma de consejos para los usuarios. El cuerpo de estos textos cambiaría en función de estadísticas obtenidas a partir del diagrama de dispersión.

Los requisitos finales, incluyendo las actualizaciones que recibieron durante el seguimiento del proyecto, se listan a continuación:

- El sistema obtendrá a partir de dos aspectos de cada alumno (su rango de notas, de entre n rangos existentes, y su rango de participación en la plataforma, de entre p rangos existentes) una clasificación de entre $n \times p$ posibilidades.
- Estos aspectos serán representados mediante un diagrama de dispersión de dos dimensiones, que contendrá la representación de cada alumno y los rangos correspondientes a cada clasificación.

- El profesor podrá ver a todos los alumnos de un curso representados por cartas. Estas cartas contendrán, de cada alumno: imagen de perfil, nombre, un calificativo asignado automáticamente según la clasificación y el desglose de los dos aspectos que la justifican.

- Desde el diagrama de dispersión, el profesor podrá añadir, eliminar o editar los rangos que delimitan las clasificaciones.

- El sistema tendrá preparadas varias descripciones en lenguaje natural para lo que sucede en el diagrama. Por ejemplo: alta correlación, baja correlación, notas generalmente altas, participación generalmente baja, etc., y las implicaciones que estos casos conllevan.

- Las vistas involucradas requieren un buscador o filtrador de alumnos.

- El sistema debe proporcionar plantillas de configuraciones por defecto. Una configuración contiene n rangos de notas, p rangos de participación, y $n \times p$ descriptores para las clasificaciones resultantes.

3.2. Análisis del sistema

Es conveniente analizar el área estadística teniendo en cuenta su entorno: el sistema Educa. Siendo una arquitectura un subconjunto de la otra, el esquema del sistema mayor ha afectado a las decisiones tomadas durante el desarrollo del proyecto en mano. A continuación se explica la arquitectura del área estadística a ambos niveles.

3.2.1. Análisis del supersistema: Edueca

Edueca, o los sistemas que contiene que conciernen al caso, es desglosada como sistema en esta sección. El desglose se centra en las opciones que ofrece y cómo afectan al modelo (base de datos), dejando de lado los aspectos de alto nivel como la navegación de la página, que es estudiada más a fondo en los detalles de la implementación.

En Edueca, todos los usuarios participan en uno o varios proyectos educativos, también referidos como cursos. Cuando un usuario se registra por primera vez, es por defecto Profesor de un curso creado expresamente. Puede ser asociado a cualquier otro curso de la plataforma si se registra, solicita acceso, o es invitado a dicho curso (cada curso determina cuáles de estas opciones ofrece).

Un Profesor es esencialmente un Participante con privilegios dentro del contexto de un curso. Puede realizar todas las acciones que están a disposición del resto de participantes, como los alumnos, y adicionalmente tiene acceso a la *back-office* del curso, restringida a él y otros profesores, donde puede modificar el contenido del curso.

Existen más niveles internos de autoridad, como la diferencia entre Profesor y Administrador, pero no afectan a la parte del modelo relevante para el área estadística: el contenido de las herramientas y la auditoría de accesos de los alumnos.

La figura 3.1 muestra el diagrama de casos de uso que incluye las funcionalidades que los alumnos pueden llevar a cabo dentro de un curso (es decir, sin incluir la creación o borrado de cursos, la adquisición de planes de suscripción, ajustes del perfil, u otras opciones superiores en la jerarquía de navegación).

En el diagrama se emplea el subsistema «Herramienta». Solo está representado una vez en el diagrama, pero en el sistema real, dicho subsistema se repite por cada herramienta. Una herramienta es todo aquello que el Profesor puede presentar ante el alumno en el aula: temario en forma de texto e imágenes, ficheros, enlaces, cuestionarios, foros, etc. La figura muestra cómo el sistema ofrece la inserción de algunas de ellas en el aula.

El subsistema «Herramienta» sirve para traer atención sobre el hecho de que todas las actividades realizables por alumnos dentro del contexto de un curso lo son a través de herramientas. «Ver Portada» presenta la página inicial de la *front-office* del curso en cuestión, donde el Profesor ha dispuesto las herramientas a su criterio, y «Acceder» y «Responder» implican que el alumno ha dejado constancia de su interacción con la herramienta de formas que se explican a continuación. Por tanto, es necesario entender cómo se guarda en el modelo el contenido de las herramientas, que en la base de datos son todas aquellas tablas que constan del prefijo «`Herramienta`» procedido por otro prefijo que en cada caso indica un tipo concreto de herramienta.

El diagrama de objetos de la figura 3.3 muestra estas entidades del modelo, y cómo están asociadas a los usuarios y a los cursos. A continuación se analiza el contenido del diagrama.

El tronco del diagrama está formado por los objetos `Usuario`, `Proyecto` y `Herramienta_nombre`. El último representa cualquier tabla de nombre prefijado por `Herramienta`, es decir, todas las herramientas. El tronco del diagrama muestra lo que cabe esperar: muchos alumnos participan en muchos cursos. A su vez, cada curso puede incluir, para cada herramienta, muchas de su tipo.

La rama derecha del diagrama muestra el contenido de las herramientas. Cada herramienta consta de un esquema similar mediante el que guardar su contenido: `Herramienta_nombre` es la raíz o cabecera, y cada nodo que hace referencia a ella representa una entidad dentro del contexto de la herramienta que, a su vez, puede ser referenciada por otros nodos de un nivel aún inferior. Esta sucesión puede seguir indefinidamente, dependiendo de la herramienta.

La figura 3.4 muestra un ejemplo: los foros tienen entradas, las entradas tienen respuestas, y dichas tienen «Me gusta».

Todo el contenido que un alumno puede generar directamente a través de una herramienta está en este esquema de contenido. Las acciones que incrementan este contenido son las que en la figura 3.1 se abstraen como «Responder», ya que en la mayoría de casos, es una acción literal de respuesta: a una entrada de foro, un cuestionario o una tarea.

Hay herramientas, sin embargo, cuyo contenido no será alterado por nada que haga el

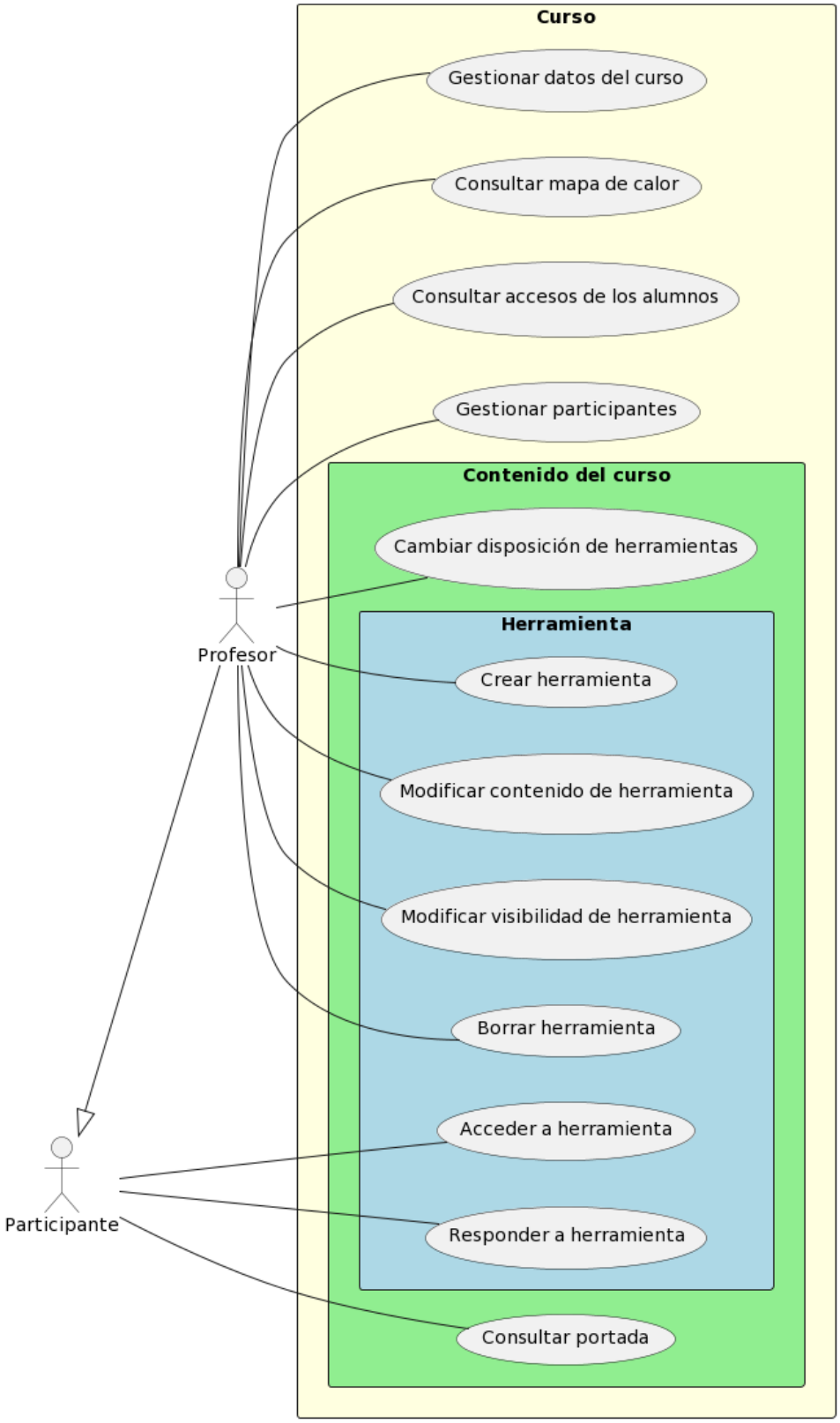

Figura 3.1: Diagrama de casos de uso de Edueca: contexto de un curso

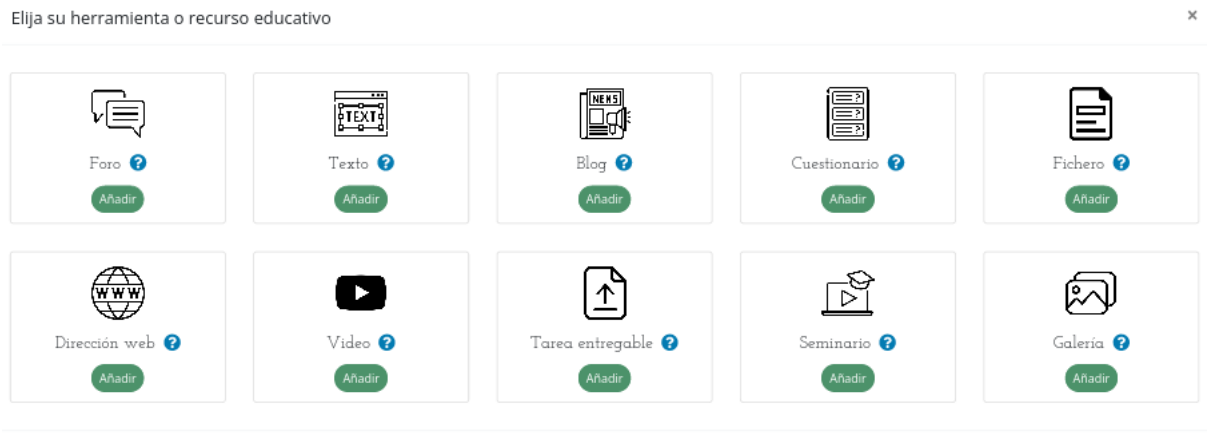

Figura 3.2: Algunas herramientas que ofrece Edueca

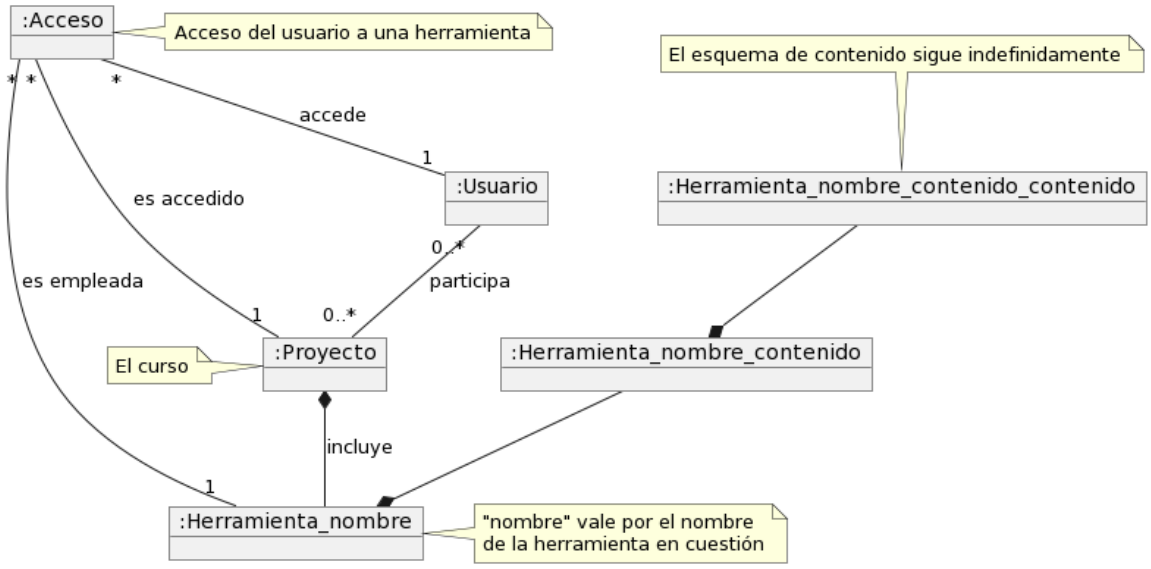

Figura 3.3: Diagrama de objetos de Edueca: familia de las herramientas

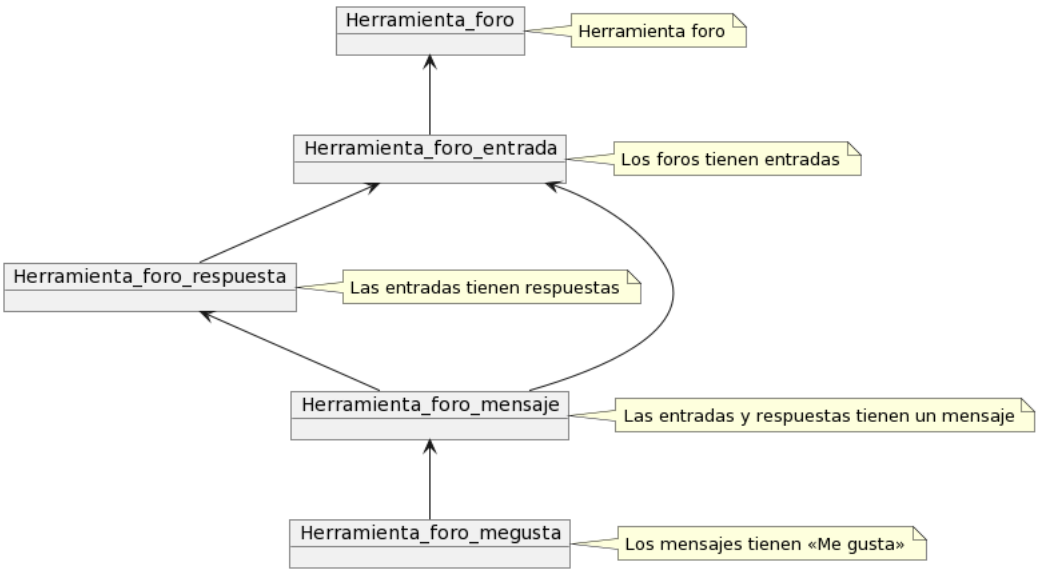

Figura 3.4: Diagrama de objetos de la herramienta «foro»

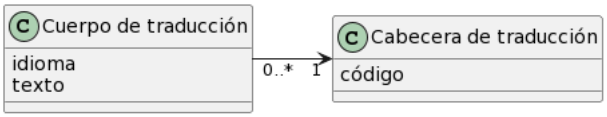

Figura 3.5: Diagrama de clases del análisis de Trans

alumno: las que son de comunicación unidireccional: enlaces, bloques de texto e imágenes, blogs, etc.

En este caso, el único rastro que deja el alumno es su acceso a la herramienta, guardado en la auditoría. El funcionamiento de la auditoría está en la rama izquierda del diagrama de objetos de la figura 3.3: la clase `Accesos` (que en el modelo no es una tabla física, sino una vista) se asocia al usuario que accede, la herramienta a la que accede, y el proyecto que la contiene.

El hecho de que un alumno haga clic en una herramienta para ver su contenido en totalidad provoca un registro de esta acción en la auditoría, y es la acción referida en la figura 3.1 como «Acceder».

Edueca cuenta con una clase propia, `Trans`, que se encarga de traducir y almacenar textos automáticamente. La primera vez que se solicita un texto en un idioma, si su traducción no ha sido introducida en la base de datos manualmente, emplea la API del Traductor de Google y almacena la respuesta para ahorrarse futuras llamadas.

Esta se emplea para traducir textos no personalizables por el usuario final. Para emplearla, es necesario introducir el texto llano que se desea traducido en la base de datos, en la tabla correspondiente al cuerpo de la traducción, indicando el idioma. Esta fila hará referencia a un código en la otra tabla, la cabecera de la traducción, indicando así que otras filas con el mismo código corresponden al mismo texto en diferentes idiomas. Su uso está incorporado en el área estadística y se detalla en su análisis, a continuación. Este detalle hace referencia a las

ocasiones en las que `Trans` es empleado para textos que provienen del controlador, no para aquellos que provienen de la vista (donde también es posible emplear este sistema), ya que lo segundo involucra individualmente cabeceras, títulos, botones, etc.

3.2.2. Análisis del área estadística

Habiendo expuesto el análisis del entorno Edueca, a continuación se expone el análisis del área estadística.

El área estadística expande las funcionalidades que el Profesor encuentra en la *back-office* de Edueca, en el contexto de su curso. La figura 3.6 muestra el diagrama de casos de uso correspondiente a esta expansión de funcionalidad, dividida por subsistemas correspondientes a las dos vistas.

Los nueve casos adicionales tienen secuencias de acción similares, ya que sus respectivas vistas están a la misma altura en la navegación de la página. Los cuadros 3.1 y 3.2 muestran, respectivamente, las especificaciones de los casos de uso CU1 y CU2.

Especificación del caso de uso CU1	
Identificador	CU1
Nombre	Consultar dispersión de alumnos
Autores	Almar Abella Moliner
Fuentes	Almar Abella Moliner & Cidet
Descripción	El sistema debe mostrar a todos los participantes de un curso como puntos en un diagrama de dispersión que tenga como ejes la nota media y la tasa de participación. Los puntos deben ser identificables como el participante que representan, y el diagrama debe estar segmentado por regiones etiquetadas con descriptores.
Alcance	El sistema debe realizar consultas a la base de datos y representar visualmente los valores obtenidos.
Nivel	Tarea principal
Actor principal	Profesor
Actores secundarios	Administrador de curso
Relaciones	CU2 Aplicar plantilla, CU3 Cambiar nombre de clasificación,CU4 Cambiar rango de franja, CU6 Filtrar alumnos por nombre.
Precondición	El usuario ha de ser Profesor o Administrador de un curso y hallarse en la Configuración de dicho curso.
Condición de final exitoso	El sistema mostrará datos veraces de los participantes del curso.
Condición de final fallido	El sistema no mostrará datos veraces de los participantes del curso.
Trigger	El profesor accede a «Análisis de dispersión» desde la Configuración de su curso.
Secuencia normal	**Acción**
1	El usuario se autentica en el sistema.
2	El usuario accede a la vista de cursos de los que es participante.
3	El usuario accede a un curso del que es Profesor o Administrador.
4	El usuario accede a la configuración de dicho curso.
5	El usuario accede a la vista «Análisis de dispersión».
6	El sistema muestra satisfactoriamente el análisis de dispersión.
Frecuencia esperada	Una vez por semana
Importancia	Vital
Prioridad	Medio plazo

Cuadro 3.1: Especificación del caso de uso «Consultar dispersión de alumnos»

Especificación del caso de uso CU2	
Identificador	CU2
Nombre	Aplicar plantilla
Autores	Almar Abella Moliner
Fuentes	Almar Abella Moliner & Cidet
Descripción	El sistema ha de ofrecer un conjunto de plantillas consistentes en: diferentes formas de segmentar el diagrama de dispersión, etiquetas para las regiones resultantes, y una breve descripción de lo que representa cada plantilla. Si un usuario aplica una, dicha pasa a definir la forma en que se clasifican los alumnos de su curso.
Alcance	Actualizar la base de datos. Cada plantilla que proporciona el sistema debe definirse manualmente.
Nivel	Tarea principal
Actor principal	Profesor
Actores secundarios	Administrador de curso
Relaciones	CU1 Ver dispersión de alumnos
Precondición	El usuario ha de ser Profesor o Administrador de un curso y hallarse en la vista «Análisis de dispersión» de dicho curso.
Condición de final exitoso	La plantilla se aplica correctamente y la página se recarga.
Condición de final fallido	La plantilla no se aplica correctamente y la página alerta del fallo.
Trigger	El profesor selecciona una plantilla desde la vista «Análisis de dispersión».
Secuencia normal	**Acción**
1	El usuario se autentica en el sistema.
2	El usuario accede a la vista de cursos de los que es participante.
3	El usuario accede a un curso del que es Profesor o Administrador.
4	El usuario accede a la configuración de dicho curso.
5	El usuario accede a la vista «Análisis de dispersión».
6	El usuario abre la configuración de la vista.
7	El usuario selecciona una plantilla.
8	El sistema recarga la página con el diagrama de dispersión actualizado satisfactoriamente.
Frecuencia esperada	Una vez al mes.
Importancia	Deseable
Prioridad	Medio plazo

Cuadro 3.2: Especificación del caso de uso «Aplicar plantilla»

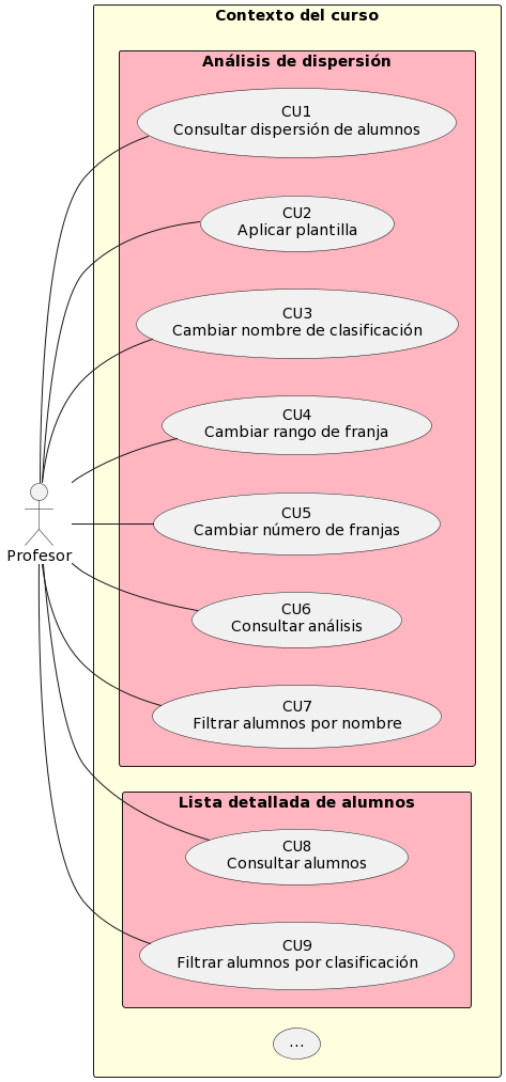

Figura 3.6: Diagrama de casos de uso del área estadística

El flujo de acciones que permite realizar la vista «Análisis de dispersión» está mostrado por el diagrama de actividades de la figura 3.7. La página carga y presenta el diagrama de dispersión antes que nada, así que esa es siempre la primera actividad. Las acciones realizables a continuación son: consultar el análisis estadístico, y alterar la configuración del diagrama de diferentes formas.

«Configurar» representa las acciones que se realizan en la vista desde un panel desplegable. Dicho panel incluye todas las formas en que al usuario le es posible configurar el diagrama de dispersión excepto una, «Cambiar rango de franja». Esta acción, si no se hace mediante el aplicado de una plantilla, se debe realizar sobre el propio diagrama. La razón es que se espera que en algunos casos, el usuario decida dónde delimitar las clasificaciones en función de la dispersión y no pese a esta. Estos casos incluyen:

- Si observa cúmulos o *clusters* de alumnos y quiere tenerlos identificados.

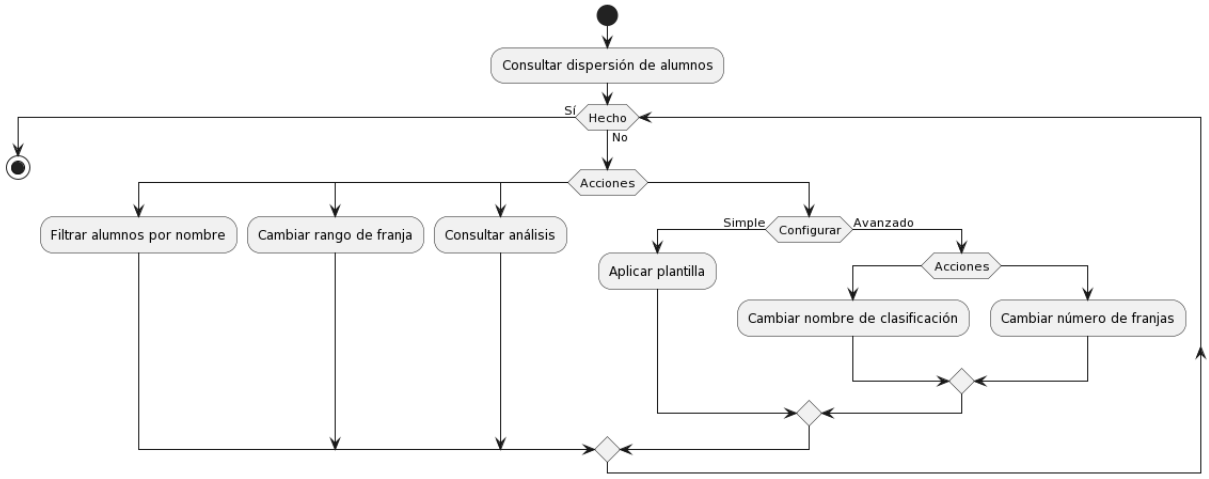

Figura 3.7: Diagrama de actividades de la vista «Análisis de dispersión»

- Si observa que hay alumnos que oscilan sobre una constante concreta de nota o participación y quiere saber cuándo si por encima o por debajo.
- Si quiere seccionar el diagrama dejando a n alumnos en un lado. Por ejemplo, 'los cinco alumnos con mejores notas'.

Para estos casos, no es práctico tantear introduciendo valores escalares, sino realizar un ajustado visual. Los otros tipos de configuraciones, en cambio, tratan con valores discretos o cualitativos, para los cuales un formulario es válido:

- Las plantillas tienen un valor determinista para todos los parámetros del diagrama.
- Los nombres o etiquetas de las clasificaciones son calificadores en lenguaje natural.
- El número de franjas, verticales u horizontales, es natural.

En el diagrama de actividades de la vista «Lista detallada de alumnos» (figura 3.8), nuevamente, todas cartas que representan a los alumnos se cargan en cuanto la vista es iniciada, y la acción realizable entonces es filtrarlos por clasificación.

La figura 3.9 muestra el diagrama de clases del área estadística. A continuación se describen las clases, agrupadas por los subsistemas a los que sirven. El diagrama obvia las referencias a la tabla 'Curso' (`edu_projects`) por simplicidad, ya que se entiende que su funcionamiento es siempre interno a un curso.

Diagrama de dispersión

- **Participación:** relación ente Participante y Participable. No tiene guarda fecha, ya que solo se puede instanciar una vez por Participante y Participable únicos.

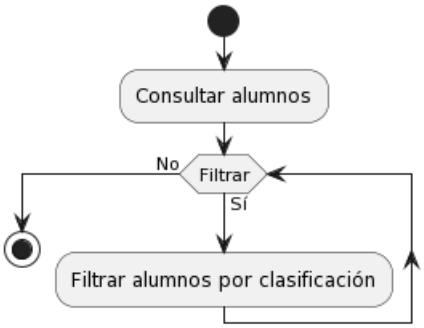

Figura 3.8: Diagrama de actividades de la vista «Lista detallada de alumnos»

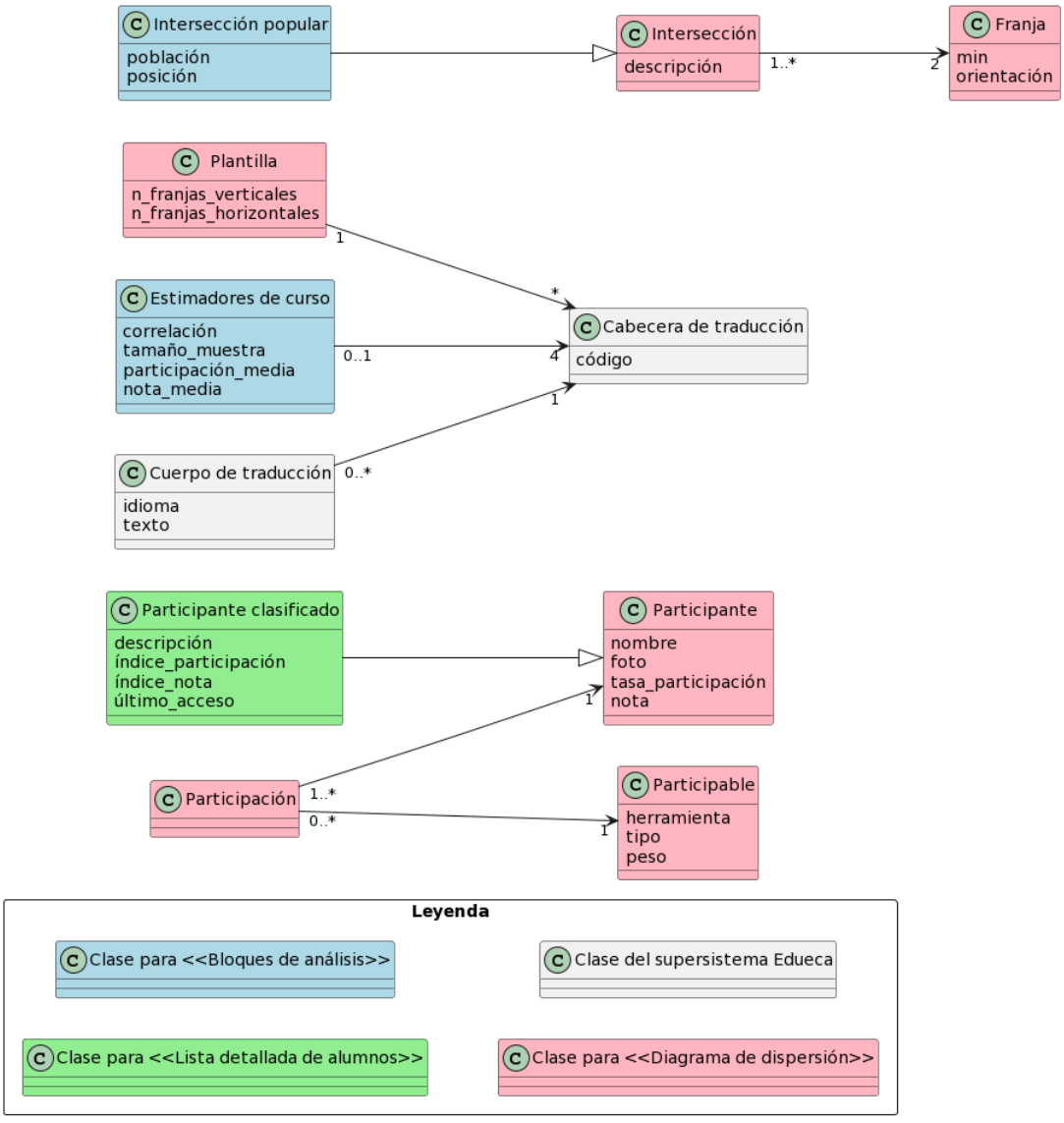

Figura 3.9: Diagrama de clases del área estadística a nivel del análisis

- **Participable:** referencia a una herramienta existente del curso.
- **Intersección:** señala a dos franjas mediante sus posiciones relativas (1a, 2a, 3a, etc.) o índices, y asigna una descripción a la intersección entre ellas.
- **Participante:** cualquier usuario que haya realizado al menos una participación en un curso.
- **Plantilla:** una configuración por defecto. Incluye número de franjas verticales y horizontales, y señala a las cabeceras de traducción asociadas a: la descripción de la plantilla, su título, y las descripciones de sus intersecciones.
- **Franja:** definida por su valor mínimo y su orientación. Su valor máximo no se especifica; debe obtenerse lógicamente como el mínimo de la franja con misma orientación y siguiente índice.

Bloques de análisis

- **Intersección popular:** extiende Intersección añadiendo su posición en el *ranking* de Intersecciones con más participantes, y la cuenta de participantes que tiene (población).
- **Estimadores de curso:** diversos cálculos obtenidos a partir de la dispersión; correlación entre notas y participación, nota media general, participación media general, y cantidad de datos (o alumnos) disponibles, el tamaño de muestra. Hace referencia a la cabecera de traducción de los cuatro bloques de texto que han de exponerse dependiendo de sendas magnitudes.

Lista detallada de alumnos

- **Participante clasificado:** una extensión de Participante que encapsula valores adicionales para facilitar su paso a la vista detallada; los índices de sus respectivas franjas, la descripción por clasificación, y la fecha del último acceso a la plataforma.

3.3. Diseño de la arquitectura del sistema

La figura 3.10 muestra el diagrama de clases del diseño del área estadística. Hay ciertos atributos precedidos por el carácter '/', lo cual indica que son derivados o calculados. Esto es posible gracias a que la mayoría de tablas del área estadística son de tipo lógico y no físico, por lo cual se generan cada vez a partir del estado inequívoco de la base de datos.

El tratamiento de dichas tablas lógicas como si fuesen físicas se debe a la versatilidad de MySQL, que permite que el resultado de una consulta sea recibido por otra indistintamente a cómo lo haría con una tabla 'cruda'.

La mayoría de valores calculados o derivados tienen un nombre suficientemente descriptivo, pero es esclarecedor ver exactamente con qué valores de la base de datos están relacionados

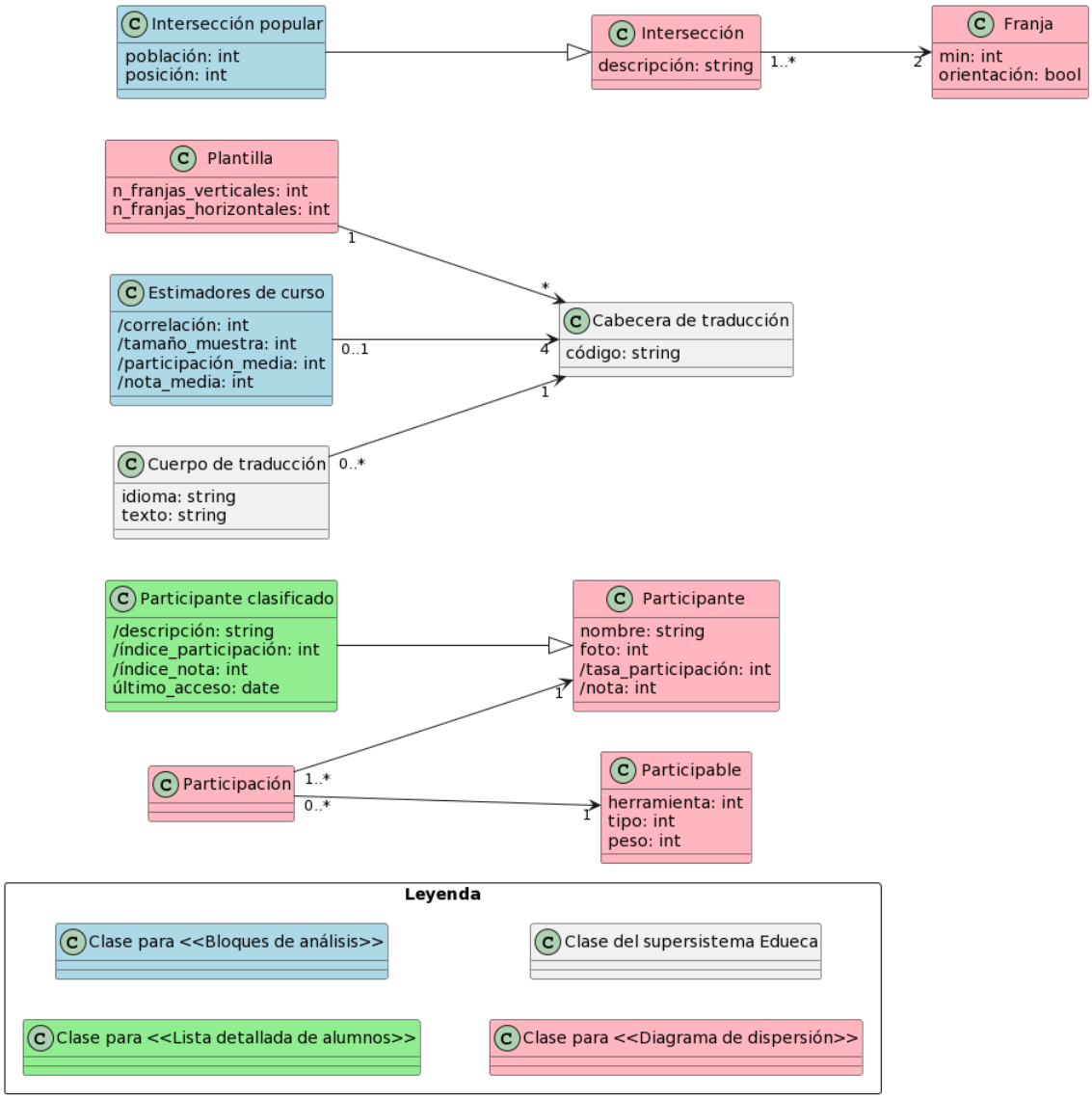

Figura 3.10: Diagrama de clases del área estadística a nivel de diseño

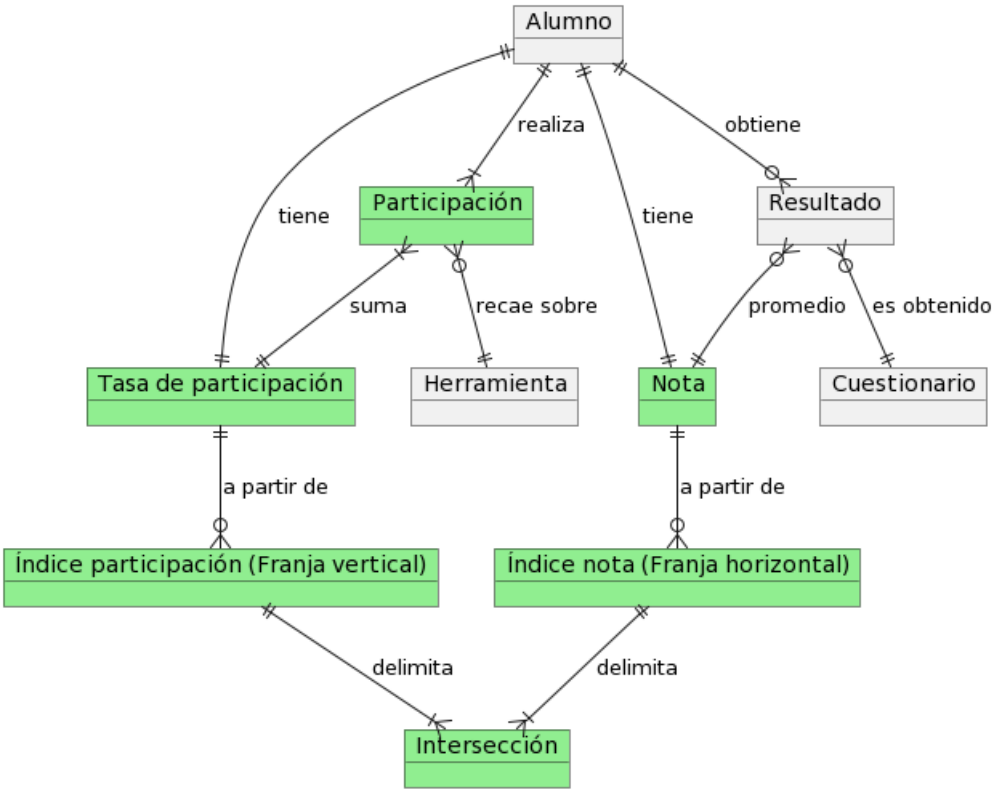

Figura 3.11: Modelo E/R centrado en alumno

mediante un modelo entidad/relación. Si bien el modelo no consta de notación matemática, se puede emplear para reflejar las uniones posibles entre entidades mediante consultas SQL.

Por sencillez, el modelo se ha dividido en dos esquemas: el modelo de la figura 3.11, centrado en el Alumno, y el modelo de la figura 3.12, centrado en el curso. En ambos, las entidades pertenecientes al modelo del área estadística están coloreadas.

Más allá de las relaciones explícitamente figuradas en el modelo, existen otras de carácter implícito, como consecuencia de las multiplicidades de otras relaciones. Por ejemplo, existe una relación muchos a uno entre Alumno e Intersección, ya que:

- Un alumno está asociado a los estimadores `Tasa de participación` y `Nota`.
- Dichos estimadores solamente pueden estar asociados a una franja cada uno (de tipo vertical y horizontal, respectivamente), puesto que las franjas son mutuamente excluyentes. Esto se debe a que, como se ha explicado en el análisis, el máximo valor de una franja se define necesariamente como el valor mínimo de la siguiente.
- Una Franja vertical y una Franja horizontal solamente tienen una Intersección.

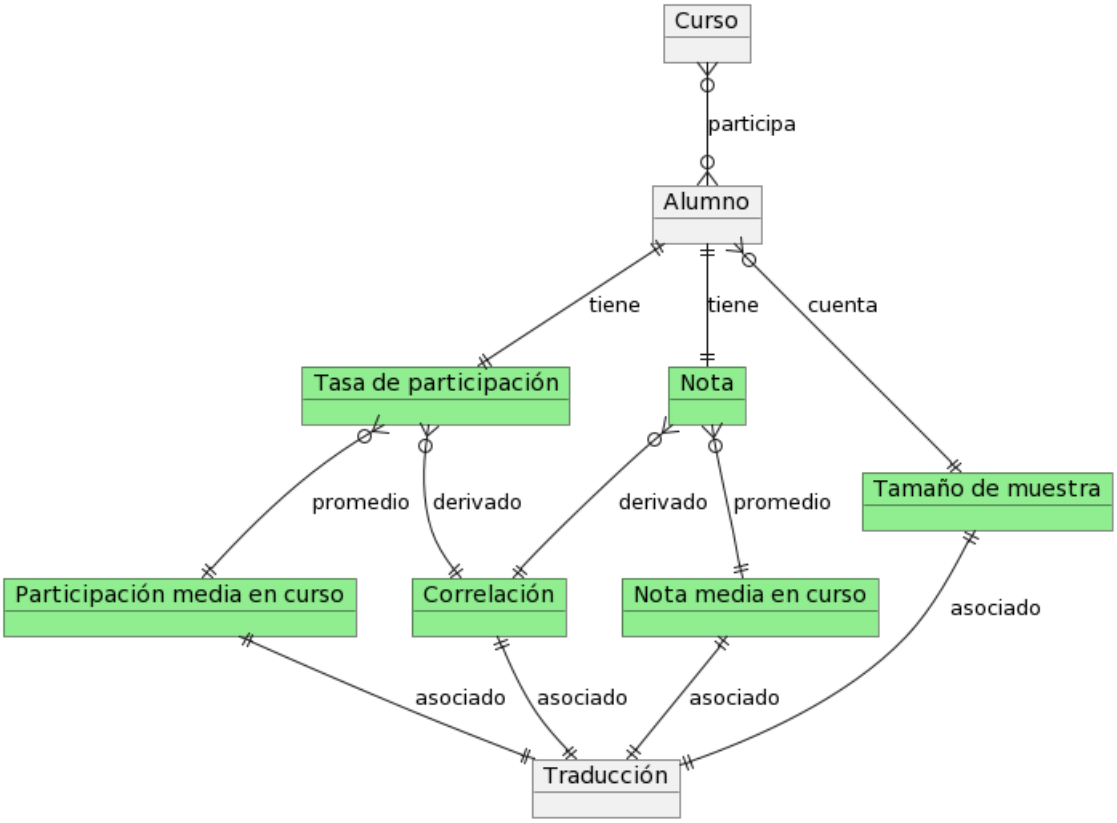

Figura 3.12: Modelo E/R centrado en curso

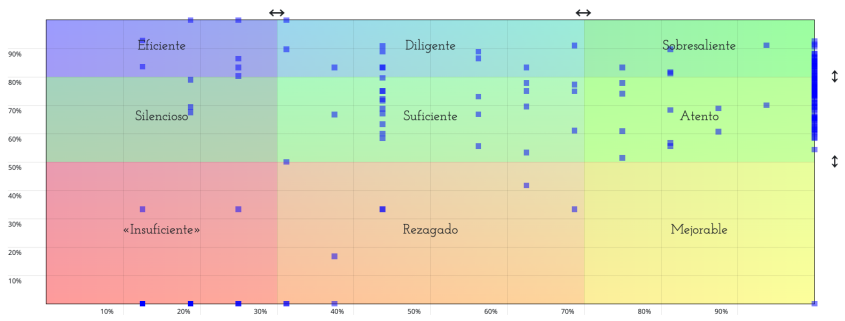

Figura 3.13: Vista del diagrama de dispersión

3.4. Diseño de la interfaz

En la plataforma Edueca, el estilo por defecto de los elementos HMTL más populares (encabezados, textos, botones, cuadros, espaciados, etc.) está sobrescrito con uno propio, así que simplemente se emplean dichos elementos durante la implementación del área estadística y el resultado adquiere el estilo que se eligió para la web. Para elementos reactivos comunes, coincidiendo con el resto de la página, se emplea Bootstrap.

Los únicos elementos diseñados expresamente para el área estadística son: el diagrama de dispersión (ver figura 3.13) y las cartas de los alumnos (ver figura 3.14). En ambos casos, se ha tratado de minimizar el diseño para que la información se encuentre fácilmente. El color tiene un rol muy importante en estas vistas. Ya que el color sobre el cual un alumno está en la dispersión coincide con el de su carta en la lista detallada, el profesor puede hacerse idea rápidamente de cuántos alumnos hay en cada categoría (las capturas tomadas no corresponden a la misma configuración).

Figura 3.14: Vista de la lista detallada de alumnos

Capítulo 4

Implementación y pruebas

En este capítulo se describe la implementación del diseño expuesto. Para explicar los detalles del *framework* Symfony, se habla también de la *spike* al principio de la estancia.

4.1. Spike

Como figura en el seguimiento del proyecto, las primeras iteraciones fueron dedicadas a la creación de una *spike*. En proyectos de software, se entiende por *spike* un programa pequeño construido previamente a la realización de un proyecto, que emplea de forma mínima todas las tecnologías de las que constará el proyecto, bajo condiciones similares, y hecho con fines de estudiar, prototipar, o probar su funcionamiento.

Ya que el producto final debe residir en un entorno de software no documentado, la web Edueca, la *spike* sirve también como primer contacto con dicho entorno. La realización de la *spike* conlleva estudiar el funcionamiento de Edueca, tanto a nivel externo como programático, entender cómo está organizada y de qué funciones está provista.

Concretamente, la *spike* debe constar de:

- Un controlador PHP que consulte información de la base de datos de desarrollo y los pase a la vista. La consulta debe involucrar algún cálculo.
- Una vista con código HTML+CSS+JS que dibuje un gráfico de algún tipo con los datos recibidos.
- Un método, llamado desde la vista y gestionado por el controlador, que provoque una respuesta con nueva información desde la base de datos.
- Cualquier otro requisito que demuestre ser esencial para el funcionamiento de un modulo en Edueca.

Los requisitos planteados son los siguientes:

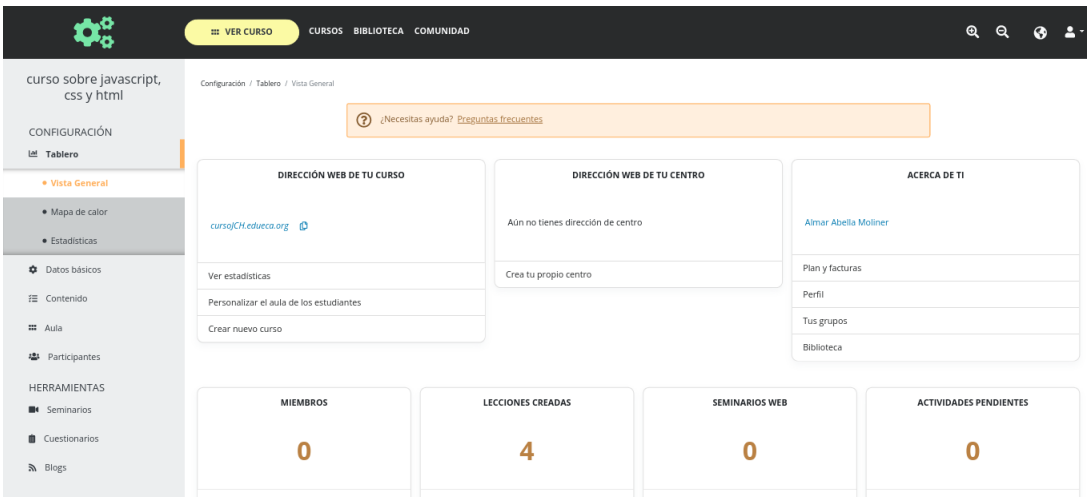

Figura 4.1: Inicio de la *back-office* de Edueca

- El sistema debe mostrar un diagrama de barras. Para cada cuestionario de un curso, el diagrama debe mostrar las notas media y máxima obtenidas por los alumnos.
- El sistema debe permitir al usuario escoger una lección del temario, de modo que se solicitarán desde la base de datos y mostrarán solamente las notas media y máxima de los cuestionarios pertenecientes a dicha lección.

A continuación se describe la implementación de la spike y los conocimientos sobre el supersistema Edueca y el *framework* Symfony que se adquirieron en el proceso.

En la *back-office* de Edueca, la *spike* debe estar a la misma altura que un conjunto de secciones a las que se accede desde el *dashboard*. El *dashboard* es el menú principal (lateral derecho en la figura 4.1), y permite a cualquier administrador de un proyecto educativo navegar las diferentes opciones de Edueca. Está dividido por dos agrupadores de secciones: Configuración y Herramientas.

La *spike* debe desarrollarse bajo el agrupador Configuración, al igual que el proyecto final. Este agrupador ya contiene actualmente dos secciones que tienen como propósito informar a un administrador de curso sobre la actividad que los alumnos realizan:

- «Mapa de calor», como su nombre indica, es un mapa de calor que emplea los nombres de los alumnos como primer eje, las diferentes herramientas y lecciones de un curso como segundo eje, y el número de clics que cada alumno realiza sobre una herramienta o lección como intensidad de calor.
- «Estadísticas» es un buscador de auditorías: introduciendo fechas y el nombre de un alumno, muestra a qué partes del curso ha accedido dicho alumno en el periodo indicado.

Mediante las funciones automatizadas de Symfony, que se acceden a través de la terminal con el prefijo `php/console` , se puede crear un controlador PHP llamado MyStatisticsController:

```
php/console make:controller MyStatisticsController
```

Dicho comando también genera una vista basada en el motor de plantillas Twig, ya conectada con el controlador, y que por defecto recibe el nombre del controlador como parámetro. Twig funciona simulando que dos niveles del patrón de diseño MVC, la vista y el controlador, son el mismo. Para ello, proporciona una serie de comandos y estructuradores lógicos que se escriben en la plantilla, junto al código HTML+CSS+JS, pero que internamente se traducen a código PHP y se ejecutan antes del que la misma sea procesada visualmente.

A continuación se presentan unos ejemplos de esta mecánica. En ellos, el uso de los delimitadores {{ }} o {%%} indica el cambio de semántica Javascript a Twig.

Sea el siguiente seudocódigo:

```
Dado un vector en mi entorno PHP
Si dicho vector tiene al menos un elemento
  Guardo el primer elemento en mi entorno Javascript
```

Una implementación inválida podría tener la siguiente forma:

```
1 const arrayLength = {{ array|length }}
2 if (arrayLength > 0){
3   const firstElement = "{{ array.0 }}"
4 }
```

Este código falla en la línea 3 si el vector está vacío. Para el motor Twig, las estructuras de control pertenecientes a otros lenguajes son invisibles. Por tanto, '{{ array.0 }}' (línea 3) se ejecutará siempre durante la traducción de la plantilla, provocando una excepción de acceso nulo.

Una solución es emplear las estructuras de control propias de Twig:

```
1 {% if array|length > 0 %}
2   const firstElement = "{{ array.0 }}"
3 {% endif %}
```

Estas estructuras de control siempre se ejecutan primero: su ubicación indica dónde va a estar en la plantilla resultante el código HTML+CSS+JS generado por el comando Twig, y el contenido suele contener valores provenientes del controlador.

Una vez entendido esto, comienza la implementación específica de la *spike* conceptualizada.

El código HTML+CSS+JS insertado en la plantilla recibe datos en forma de cadenas JSON y pinta un diagrama de barras con dichos datos. Las cadenas deben ser fabricadas desde el controlador. Para el caso de estudio, pasar desde el controlador una serie de valores de ejemplo demuestra que la comunicación controlador-vista funciona correctamente.

La siguiente parte de la *spike* consiste en que los datos mostrados sean reales y provenientes de la base de datos. Para esto, cabe hablar de la otra componente de Symfony que gestiona el patrón de diseño MVC, el mapeador objeto-relacional Doctrine.

Doctrine está presente por defecto en cualquier controlador de Symfony (que se define como cualquier clase PHP que extienda la clase propia `AbstractControler`). Puede ser solicitado así:

```
1 $this->getDoctrine->getManager()
```

Este comando devuelve un objeto encapsulado que permite manipular las clases propias del proyecto en forma de objetos. Doctrine se encarga de forma automática de representar estos objetos en la base de datos, actualizándola cuando son eliminados, creados o modificados desde el controlador.

Existen casos donde no se trata con objetos de la base de datos en su forma original, sino con valores obtenidos mediante consultas complejas. Para estos casos, Doctrine permite ejecutar sentencias, ya sean de lectura o escritura, y devuelve el resultado mediante un array PHP.

Este es el caso al que nos enfrentamos en la *spike*, ya que se debe recoger el valor promedio obtenido por los alumnos en cada test. Este valor no es reflejado por ninguna entidad del sistema Edueca, por lo que es necesario calcularlo mediante una sentencia.

Desde el software MySQL Workbench, la base de datos puede ser estudiada para posteriormente saber qué forma debe tener la sentencia, y qué tablas debe incluir. El programa permite comprobar los nombres de todas las tablas disponibles, su definición, y las filas que contienen.

Para observar qué tablas están siendo manipuladas por los controladores que se pretende estudiar, resulta útil consultar el *profiler* de Symfony. El *profiler* es un panel de control desplegable que proporciona información sobre cualquier vista de la aplicación que se esté ejecutando en su versión de desarrollo: nombre del controlador que proporciona datos a dicha vista, peticiones que se están realizando, respuestas, avisos de incidencias, etc.

En este caso, la vista de interés es la herramienta «cuestionario», ya que es donde se originan las notas de los alumnos, y la información que proporciona el *profiler* es el nombre del controlador y el de los métodos que están siendo llamados. Realizando una traza de estos métodos, obtenemos:

- Las entidades del modelo involucradas en la gestión de las notas.
- Las tablas de la base de datos que reflejan físicamente estas entidades.

Comprobando entonces cómo estas tablas se relacionan en la base de datos, es posible elaborar documentación sobre el supersistema Edueca que será útil tanto en el transcurso de este proyecto, como para futuras referencias. La figura 4.2 muestra el diagrama de clases de la familia «cuestionario», es decir, asociadas a dicha herramienta.

Una vez implementada la consulta a la base de datos desde el controlador, la *spike* muestra, mediante un diagrama de barras, las notas promedio reales de cada cuestionario de un curso.

Lo único restante para completar la *spike* es que el sistema tenga comunicación bidireccional, es decir, que el cliente pueda realizar una acción que provoque una respuesta desde la base de

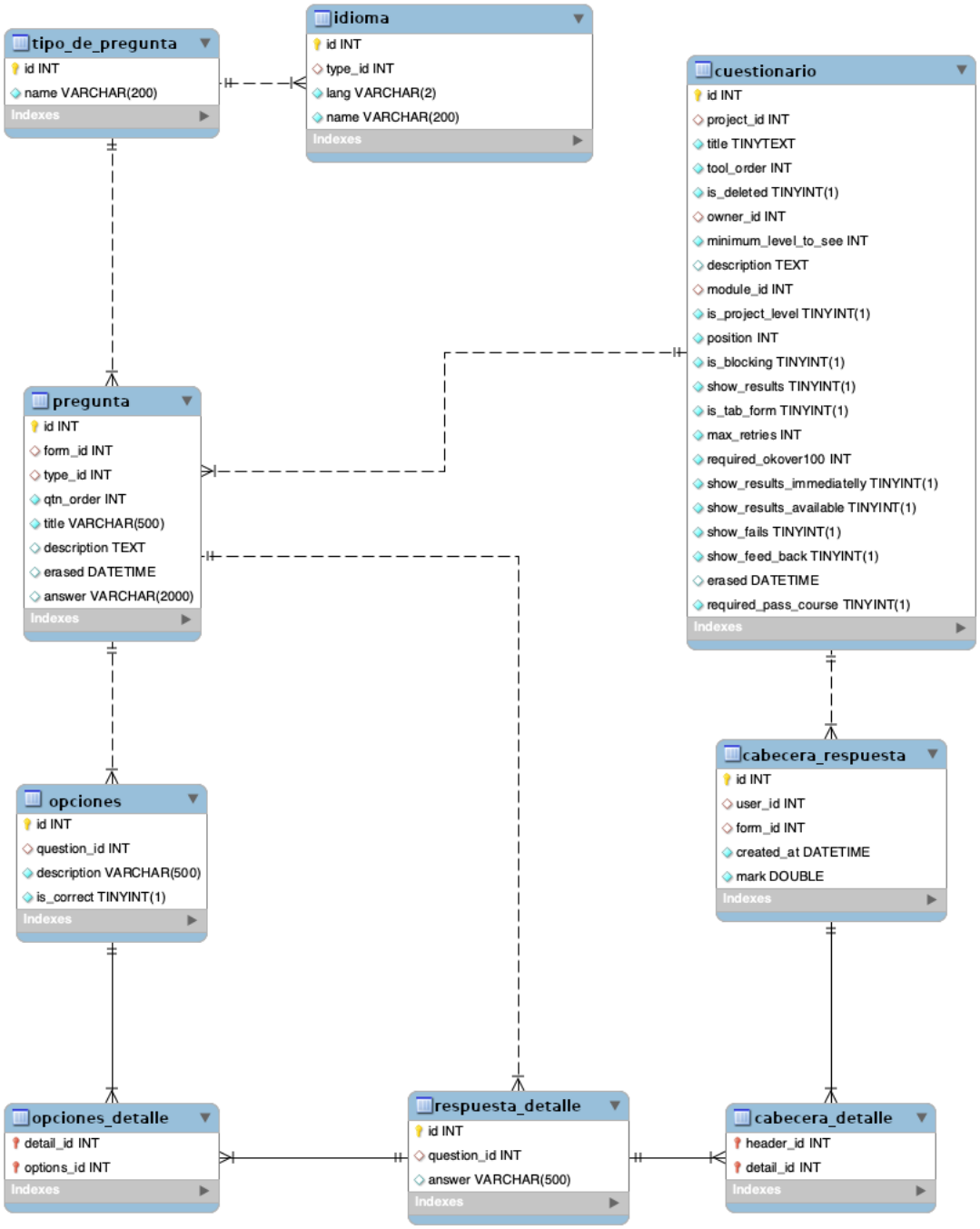

Figura 4.2: Tablas (o clases) asociadas a la herramienta «cuestionario»

datos. En este caso, mediante un filtro, se debe poder escoger una lección del temario, y la base de datos deberá devolver solamente las notas promedio de los cuestionarios pertenecientes a dicha lección.

Para que una acción que realiza el usuario sea percibida desde el servidor, este deberá ofrecer una ruta a la que el cliente pueda llamar al realizar la acción. En Symfony, el rutado se realiza a través del controlador mediante los comentarios PHPDocs:

```
/**
 * @Route("/my_filtered_statistics/{module}", name="my_filtered_statistics")
 * @Route("{_locale}/my_filtered_statistics/{module}", name="
    my_filtered_statistics")
 * @param Security $sec
 * @param $module
 * @return JsonResponse
 */
public function filterMarksMean(Security $sec, $module): JsonResponse {...}
```

El modificador `@Route` le indica al *framework* que a continuación está la ruta que será llamada cuando se quiera ejecutar la función `filterMarksMean`.

4.2. Área estadística

Una vez mostrado el funcionamiento del *framework*, se procede a explicar cómo está implementada el área estadística.

El área estadística consta de dos vistas, dos controladores PHP, un servicio PHP, y dos entidades del modelo. En la figura 4.3 se expone cómo están mapeadas las comunicaciones: el servicio `LearnersPerformanceService`, inyectado a los controladores, es el que posee la mayor parte de la lógica del área estadística, y el que se comunica con la base de datos. Este servicio gestiona todas las clases del diseño, y las proporciona a los controladores mediante los `Métodos fachada`. Estos deben su nombre al patrón de diseño fachada empleado en el servicio `LearnersPerformanceService`, como se detalla más adelante.

Solo dos de las clases del diseño son físicas (existen en el modelo de Symfony y en la base de datos), `Stripe` e `Intersection`. Estas clases almacenan la información relativa a la configuración del diagrama de dispersión: las franjas y sus intersecciones. Es necesario que existan físicamente, ya que la información que contienen varía en cada curso y es personalizable por sus administradores.

El motivo por el que las otras clases son lógicas es que toda la información que contienen se obtiene a partir de otros datos ya existentes en la base de datos. Almacenar estas clases requeriría de una infraestructura que revisase el estado consistente de la base de datos tras cada modificación, un proceso que tendría un coste similar a simplemente calcularlos cada vez y desecharlos después. Así pues, se opta por eliminar la redundancia.

Estas clases tampoco podrían existir de forma normal en el modelo de Symfony, ya que si

se definen como objetos del modelo, el gestor objeto-relacional Doctrine lanzaría una excepción al comprobar que no existen en la base de datos.

Los controladores `MyStatisticsController` y `LearnersCardsController` encapsulan la información de una forma óptima en la que ser recogida desde la vista, y pasan dicha información. La vista, como se ha visto en la explicación de la *spike*, gestiona los datos mediante Twig.

El servicio `LearnerPerformanceService` trata las respuestas a consultas MySQL como clases lógicas. Por ejemplo, los objetos de la clase Participable del diseño corresponden a cada fila de la tabla resultante de ejecutar esta consulta:

```
1     protected $globalParticipationSentence = "
2        (select 'lessons' as kind, 11 as entity_id, 1 as weight, id as ref from
      'lessons' where project_id = %1$d and erased is null) union
3        (select 'forums' as kind, 14 as entity_id, 1 as weight, id as ref from
     forums where project_id = %1$d and erased is null) union
4        (select 'lesson_contents' as kind, 15 as entity_id, 1 as weight, id as
     ref from lesson_contents where project_id = %1$d and erased is null) union
5        (select 'forum_entries' as kind, 16 as entity_id, 1 as weight, topics.
     id as ref from forum_entries topics join forums on (topics.forum_id = forums
     .id)
6           join suscription up using(user_id) where forums.project_id = %1$d and
      forums.erased is null and up.role>=50) union
7        (select 'questionary' as kind, 17 as entity_id, 1 as weight, id as ref
     from questionary where project_id = %1$d and erased is null) union
8        (select 'uploadable' as kind, 19 as entity_id, 1 as weight, id as ref
     from uploadable where project_id = %1$d and erased is null)
9    ";
```

En ella, se une mediante el comando MySQL `union` un conjunto de subconsultas en las que se solicitan los atributos (peso, tipo e identificador) de diferentes herramientas. El atributo tipo está presente de forma redundante, donde `entity_id` es el identificador absoluto del tipo, y `kind` es un descriptor incluido para facilitar la depuración. Nótese que el peso `weight` de todas está fijado a 1, ya que como se explicó en la validación de los requisitos, se trata de un valor parametrizable para cada herramienta que ahora posee su valor por defecto.

4.2.1. Estructura del servicio LearnerPerformanceService

La lógica del servicio `LearnerPerformanceService` se separa en varios esquemas. A continuación se presenta cada uno, donde los métodos fachada que ven los controladores están estilizados de azul, y las tablas físicas de la base de datos, en rojo.

Para obtener la posición de cada alumno en el diagrama de dispersión, es decir, su nota media y su participación, el servicio obtiene la clase lógica Participantes mediante los métodos mostrados en la figura 4.4, empezando por el método fachada `getLearnersPerformance`.

Para obtener las clases físicas Franja e Intersección, agrupadas e indexadas para facilitar su tratamiento desde la vista, se ejecutan los siguientes métodos, empezando por el método `getStripes`.

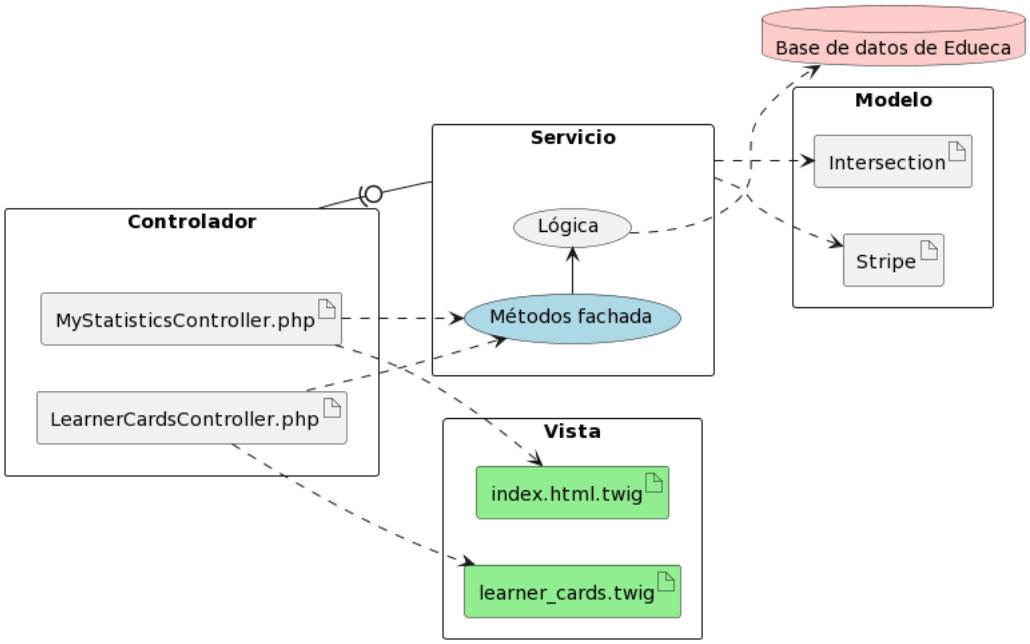

Figura 4.3: Arquitectura simplificada del área estadística

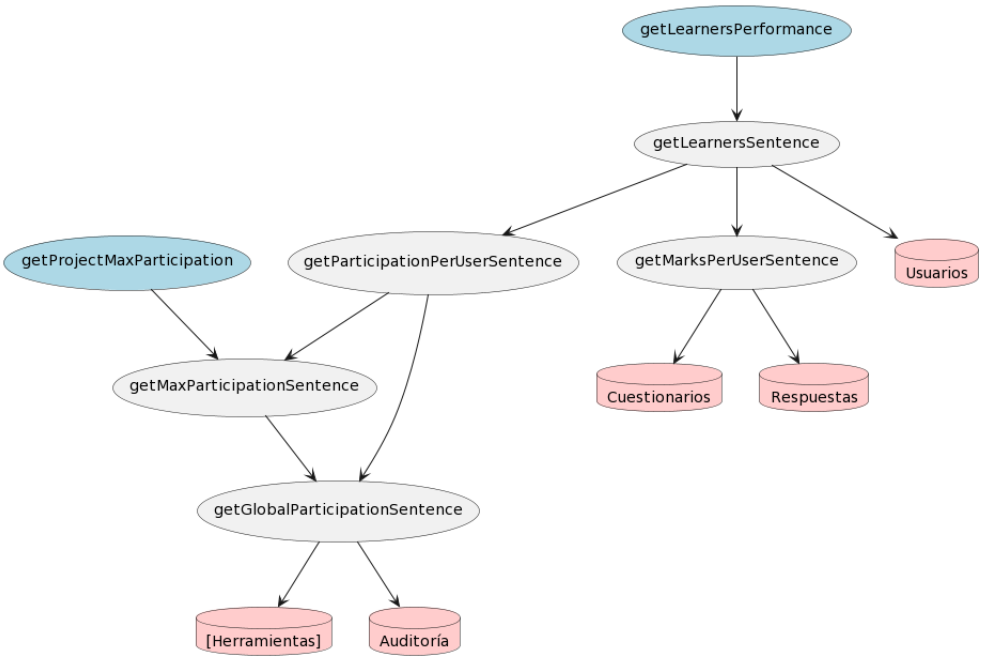

Figura 4.4: Métodos que obtienen los datos de los participantes para el diagrama de dispersión

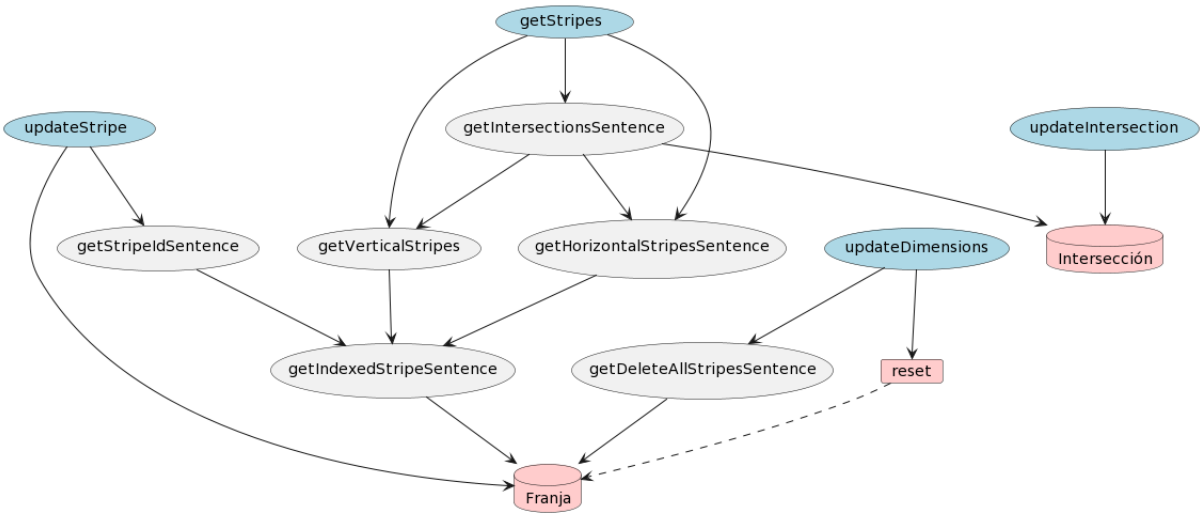

Figura 4.5: Métodos que obtienen la configuración del diagrama de dispersión

Además, este esquema abarca la gestión de la configuración: el actualizado de franjas, el de número de franjas (las dimensiones del diagrama) y la actualización de intersecciones, desde los métodos `updateStripe`, `updateDimensions` y `updateIntersection`.

Por último, el esquema incluye `reset`, un procedimiento local almacenado en la base de datos de Edueca, escrito en código MySQL. Este procedimiento elimina todas las franjas de un curso para que a continuación, otro proceso como el aplicado de una plantilla las restaure. El controlador aplica las plantillas llamando a los métodos `updateDimensions` y `updateIntersection`.

Por último, los datos necesarios para mostrar las cartas en la lista detallada de alumnos y los datos necesarios para los bloques de análisis comparten prácticamente el mismo esquema: el de la figura 4.6.

La clase lógica «Participantes clasificados», que contiene la información textual y visual que se muestra en cada carta (incluyendo los rangos de nota y participación a los que pertenece, para determinar el color), se obtiene mediante el método `getClassifiedLearners`.

De cara al controlador, los bloques de análisis solo usan información de las clases lógicas Estimadores e «Intersección popular». Internamente, estas son un análisis de todos los participantes de un curso, por lo que emplean llaman al método `getClassifiedLearners` para realizar el cálculo.

4.3. Pruebas

El área estadística ha sido sometida a un conjunto de pruebas, cuyos escenarios han sido escritos siguiendo el criterio de clases de equivalencia.

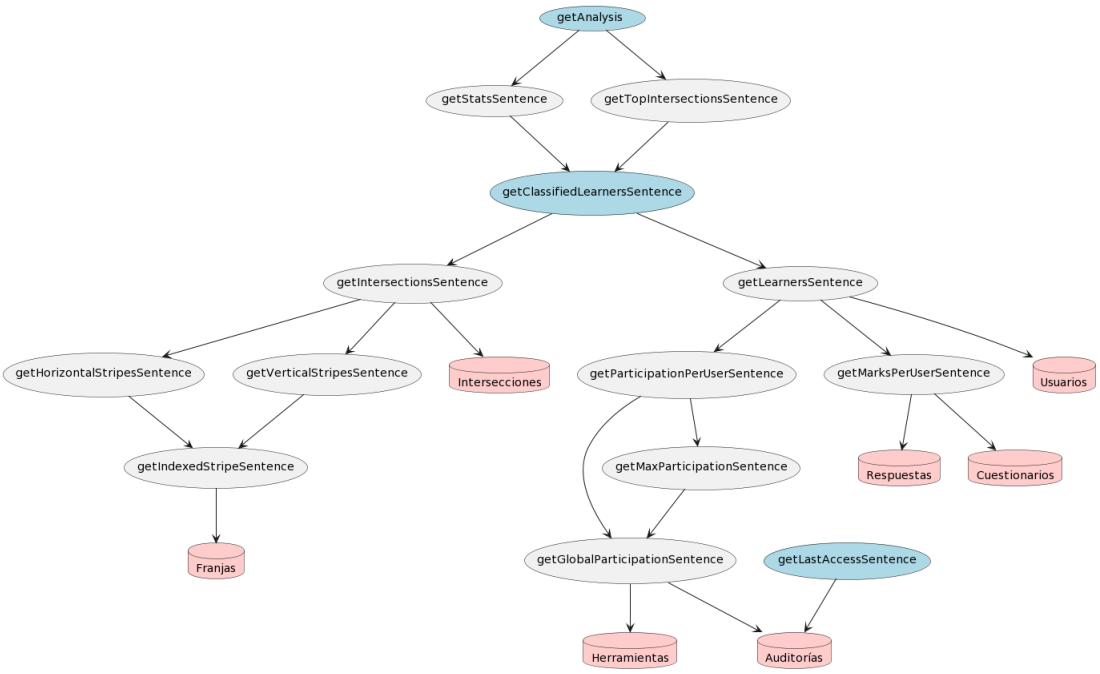

Figura 4.6: Métodos que obtienen la información de los bloques de análisis y cartas de alumnos

La parte con mayor complejidad de código del proyecto es la elaboración y gestión de las clases lógicas. Por tanto, es conveniente explorar cómo funciona esta gestión cuando se presentan datos de entrada de distinta índole desde la base de datos, en busca de fallos.

Para ello, se ha elaborado un *stub* del servicio `LearnerPerformanceService` llamado `LearnerPerformanceServiceStub`. Dicho servicio extiende a `LearnerPerformanceService` y, por tanto, puede ser inyectado a los controladores (ver la figura 4.7).

En este stub, todos los métodos que solicitan información de la base de datos son modificados ligeramente para obtener la información de otro lugar: una consulta que proviene de otro método con su mismo nombre procedido del sufijo *Stub*. Por ejemplo, el diagrama de la figura 4.4 quedaría modificado con la inyección del *Stub* de la forma representada en el diagrama de la figura 4.8.

Estos métodos adicionales devuelven una consulta cuyas características coinciden la de la original: número de columnas, y nombre y tipo de las mismas. En el *stub*, en cualquier caso, estos datos son rígidos: se obtienen de variables estáticas almacenadas en el propio *stub*, y cuyo valor se puede modificar mediante un *setter* en pruebas. Como ejemplo, a continuación se presenta la sentencia MySQL que obtiene los datos de los alumnos en el método real `getLearnersSentence`, comparada a cómo se construye dicha sentencia en `getLearnersStub`.

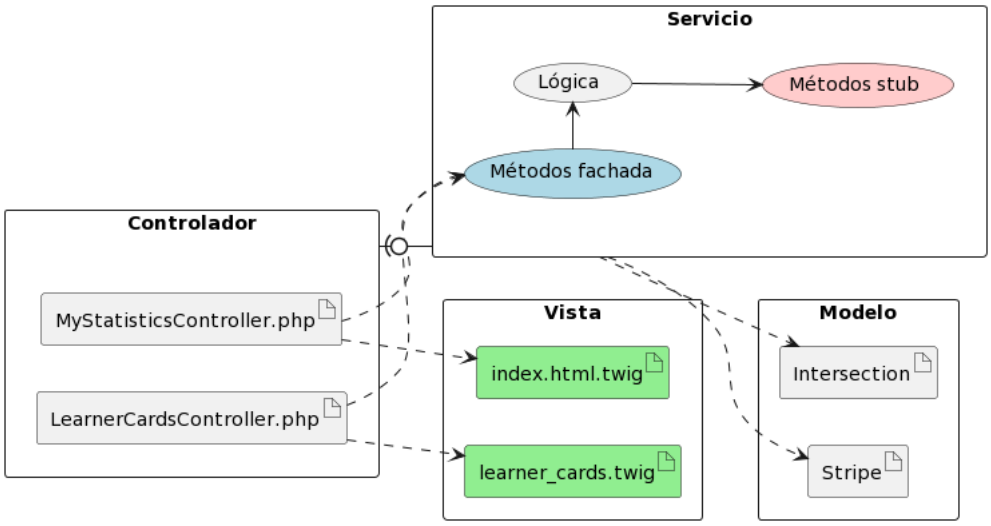

Figura 4.7: Arquitectura simplificada del área estadística, con el *stub* conectado

Figura 4.8: Métodos que obtienen los datos de los participantes, con inyección de *stub*

getLearnersSentence

```
1     protected $learnersSentence = '
2         select concat(u.name, \' \', u.familyname) name, user_id, amount, mark,
      image_file_id pic
3         from (%s) participation left join (%s) marks using(user_id)
4         right join users u on (u.id = user_id)
5         where lower(name) like lower(\'%s\')
6     ';
```

getLearnersStub

```
1
2         $users = 'select null id, null name, null pic';
3
4         if (sizeof(self::$learners) > 0){
5             $learners_array = [];
6             for ($i = 0; $i < sizeof(self::$learners); $i++){
7                 $learners_array[] =
8                     '(select '.self::$learners[$i]['id'].' id,\''
9                     .self::$learners[$i]['name'].'\' name, 706 pic)';
10
11            }
12            $users = join(' union ', $learners_array);
13        }
14
15        return '
16        select name, user_id, amount, mark, pic
17        from (%s) participation left join (%s) marks using(user_id)
18        right join ('.$users.') u on (u.id = user_id)
19        where lower(name) like lower(\'%s\')
20        ';
```

Ambas presentan las mismas columnas: `name`, `user_id`, `acount`, `mark` y `pic`.

La diferencia primordial es que `getLearnersSentence` obtiene datos de la tabla de usuarios `sec_users` (línea 4), mientras que `getLearnersStub` obtiene datos otra sentencia fabricada en el propio método y guardada en la variable `$users`.

Una vez introducido el funcionamiento del *stub*, se procede a explicar la elaboración de las pruebas. El criterio de clases de equivalencia busca elaborar el menor número de escenarios lo más representativos que sea posible. Para ello, se especifica primeramente qué datos de entrada regirán las clases de equivalencia, y bajo qué condiciones. El cuadro 4.1 recoge esta información.

Como se puede observar, se ha definido como condición general el tamaño de cada conjunto de datos. En el cuadro 4.2 se definen varios escenarios que cubran todas las clases de equivalencia.

A continuación se muestra como ejemplo la prueba del escenario E2. Después de introducir la batería de datos en el *stub*, se realizan solicitudes a las dos vistas que conforman el proyecto y se comprueba que han sido renderizadas éxito. Las cinco pruebas restantes implementan sus respectivos escenarios, y todas ellas son superadas (ver figura 4.9).

Id	Datos	Clases de equivalencia
V	Franjas verticales	0 - ninguna, 1 - pocas, 2 - muchas
H	Franjas horizontales	0 - ninguna, 1 - pocas, 2 - muchas
I	Intersecciones	0 - ninguna, 1 - pocas, 21 - muchas de una misma franja horizontal, 22 - muchas de una misma franja vertical, 23 - muchas repartidas
B	Participables	0 - ninguno, 1 - pocos, 21 - muchos de un tipo, 22 - muchos de distintos tipos
P	Participaciones	0 - ninguna, 1 - pocas, 21 - muchas de mismo alumno y actividad, 22 - muchas de mismo alumno en varias actividades, 23 - muchas de varios alumnos en una actividad, 24 - muchas de varios alumnos en varias actividades
N	Notas	0 - ninguna, 1 - pocas, 2 - muchas
L	Alumnos	0 - ninguno, 1 - pocos, 2 - muchos

Cuadro 4.1: Clases de equivalencia del área estadística

Id	Escenario
E0	V0, H0, I0, B0, P0, N0, L0
E1	V1, H2, I1, B0, P1, N1, L1
E2	V2, H1, I21, B1, P21, N2, L2
E3	V2, H2, I22, B21, P22, N0, L2
E4	V2, H2, I23, B22, P23, N1, L2
E5	V2, H2, I23, B22, P24, N2, L1

Cuadro 4.2: Escenarios de las pruebas

```
        $client->followRedirects();
        LearnerPerformanceServiceStub::setValues(
    //muchas franjas verticales (el número marca sus valores mínimos)
            [32,54,75,2,6,54,34,74,45,23,64],
        //pocas franjas horizontales
            [1],
    //muchas intersecciones en la misma franja horizontal
            [
                ['x'=>'5', 'y'=>'1', 'tag'=>'Inaudito'],
                ['x'=>'4', 'y'=>'1', 'tag'=>'Plusquamperfecto'],
                ['x'=>'2', 'y'=>'1', 'tag'=>'Tony Stark'],
                ['x'=>'9', 'y'=>'1', 'tag'=>''],
                ['x'=>'2', 'y'=>'1', 'tag'=>'Sans'],
                ['x'=>'7', 'y'=>'1', 'tag'=>'<html>'],
            ],
        //pocos participables
            [['kind'=>'lesson', 'entity_id'=>3, 'weight'=>100, 'ref'=>15]],
    //muchas participaciones del mismo alumno en la misma actividad
            [
                ['user_id'=>3434, 'ref'=>0, 'entity_id'=>1, 'action'=>'task'],
                ['user_id'=>3434, 'ref'=>0, 'entity_id'=>1, 'action'=>'task'],
                ['user_id'=>3434, 'ref'=>0, 'entity_id'=>1, 'action'=>'task'],
                ['user_id'=>3434, 'ref'=>0, 'entity_id'=>1, 'action'=>'task'],
                ['user_id'=>3434, 'ref'=>0, 'entity_id'=>1, 'action'=>'task'],
                ['user_id'=>3434, 'ref'=>0, 'entity_id'=>1, 'action'=>'task'],
            ],
        //pocas notas
            [['user_id'=>33, 'mark'=>33]],
    //pocos alumnos
            [['id'=>33, 'name'=>'El increíble botjo con gafas']]
        );
        $client->request('GET', 'http://edueca.local/es/dashboard/7095/1d');
        $this->assertResponseIsSuccessful();
        $this->assertSelectorTextContains('h1', 'Dispersión de mis alumnos');

        $client->request('GET', 'http://edueca.local/es/dashboard/7095/1e');
        $this->assertResponseIsSuccessful();
        $this->assertSelectorTextContains('h1', 'Mis alumnos');
```

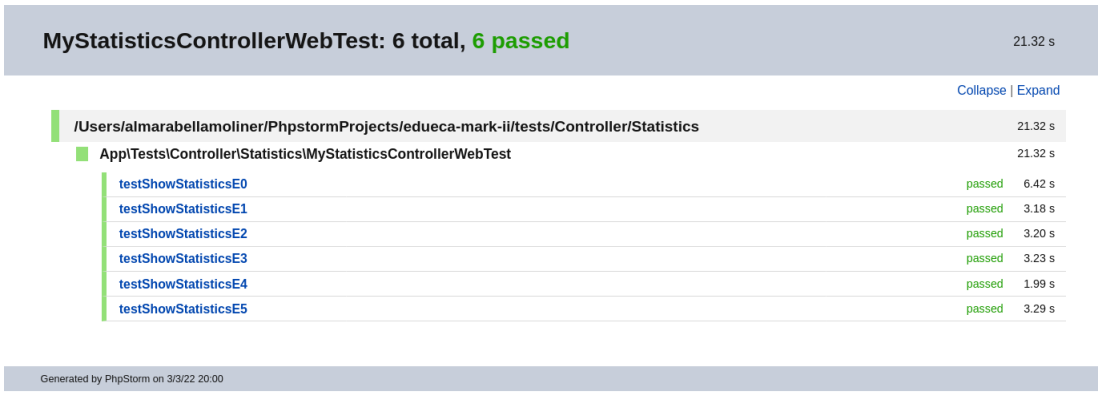

Figura 4.9: Pruebas de Symfony superadas

Capítulo 5

Conclusiones

El proyecto ha sido finalizado con éxito y estoy satisfecho con el trabajo. Este proyecto, por sus características, me ha permitido explorar diferentes áreas de la informática en mucha mayor profundidad que la que había podido hasta ahora. Especialmente, mi conocimiento en manejo de vistas con elementos nativos (HTML+CSS+JS) se ha incrementado en gran medida. En el apartado de bases de datos, también he adquirido mucho conocimiento útil, en parte gracias al IDE PhpStorm, que convenientemente sugiere formas más eficientes de implementar aquello que estás intentando.

La estancia en Cidet ha sido altamente enriquecedora a nivel personal. El ambiente laboral ha sido espléndido, lo cual siempre ayuda. No tenía muy claro en qué áreas de la informática quería centrarme saliendo de la carrera, y mis experiencias anteriores con el desarrollo *full stack* no me habían apasionado tanto. Quizá debido a la relativa libertad que he tenido para explorar soluciones a problemas de distintas índoles (estéticos, visuales, lógicas, etc.) me he dado cuenta de que me gustan los proyectos donde se tiene contacto con una sección casi vertical del proyecto, que incluye todas sus capas.

Una de las pocas cosas que no he necesitado hacer en última instancia, pero a las que hubiera estado dispuesto, es a realizar un proceso de obtención de requisitos mediante entrevistas a los usuarios finales. No ha sido necesario ya que el *brainstorming* inicial vino cargado de ideas que de por sí ya sentaban bases, y tras las cuales se había un razonamiento extendido.

Bibliografía

[1] Manuel Pérez. Análisis foda o dafo. https://yebesvaldeluz.wpcomstaging.com/. [Consulta: 8 de Enero de 2022].

[2] Case studies. https://symfony.com/blog/category/case-studies. [Consulta: 8 de Enero de 2022].

[3] Symfony documentation. https://symfony.com/doc/current/index.html. [Consulta: 8 de Enero de 2022].

[4] Highcharts demos. https://www.highcharts.com/demo. [Consulta: 8 de Enero de 2022].

[5] Guía del pmbok. https://www.sadamweb.com.ar/news/2016_08Agosto/Guia_Fundamentos_para_la_Direccion_de_Proyectos-4ta_Edicion.pdf?PMBOX=http://www.sadamweb.com.ar/news/2016_08Agosto/Guia_Fundam. [Consulta: 1 de Febrero de 2022].

[6] Sueldo: Desarrollador full stack junior. https://www.glassdoor.es/Sueldos/desarrollador-full-stack-junior-sueldo-SRCH_KO0,31.htm. [Consulta: 26 de Febrero de 2022].

[7] The lightning-smart php ide. https://www.jetbrains.com/phpstorm/. [Consulta: 7 de Marzo de 2022].

[8] Pricing overview. https://www.digitalocean.com/pricing. [Consulta: 7 de Marzo de 2022].

[9] https://www.apple.com/es/shop/buy-mac/macbook-air. https://www.apple.com/es/shop/buy-mac/macbook-air. [Consulta: 7 de Marzo de 2022].

[10] This is how long macs and macbooks last. https://www.macworld.co.uk/news/how-long-macbooks-last-3783678/. [Consulta: 7 de Marzo de 2022].