Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

SIRENO

SISTEMA DE RECOGIDA DE ENCUESTAS ONLINE

Estudiante: García Rodríguez, Urko.

Director/Directora: Villamañe Gironés, Mikel.

Curso: 2022-2023.

Fecha: Bilbao, 26, Julio, 2023.

Resumen

En un mundo donde se valora cada vez más la opinión de las personas, las instituciones buscan comprender y satisfacer las necesidades de sus usuarios. Las encuestas ofrecen una perspectiva externa que ayuda a identificar áreas de mejora. La UPV/EHU ha sido proactiva al utilizar encuestas para recopilar las opiniones de los estudiantes, pero ahora surge la necesidad de avanzar hacia un sistema digital más eficiente.

Los aspectos previamente mencionados confirman el papel fundamental que desempeñan las encuestas en este contexto. Precisamente "papel", es uno de los recursos que se pretende ahorrar con la implementación de **SiREnO** (**S**istema de **R**ecogida de **E**ncuestas **O**nline), un sistema online que busca revolucionar el proceso de valoración, brindando a los alumnos una forma más accesible y efectiva de expresar sus opiniones. Este trabajo abordará los desafíos y beneficios de la transición hacia un sistema telemático.

Laburpena

Jendearen iritziak gero eta gehiago baloratzen diren mundu honetan, erakundeek erabiltzaileen beharrak ulertu eta asetzea bilatzen dute. Inkestek kanpoko ikuspegia eskaintzen dute, hobetu beharreko arloak identifikatzen laguntzen duena. UPV/EHU proaktiboa izan da inkestak erabiliz ikasleen iritziak biltzeko, baina orain sistema digital eraginkorrago batera bilakatu beharra dago.

Aurretik aipatutako alderdiek baieztatzen dute inkestek testuinguru honetan betetzen duten funtsezko zeregina. Hemen sortzen da **SiREnO** proiektua (**S**istema de **R**ecogida de **E**ncuestas **O**nline), hau da, "Online Inkestak Biltzeko Sistema". Online sistema honek ebaluazio-prozesua irauli nahi du, ikasleei beren iritziak adierazteko modu eskuragarriago eta eraginkorragoa eskainiz. Lan honek sistema telematikorako trantsizioaren erronkei eta onurei aurre egingo die.

Abstract

In a world where people's opinions are increasingly valued, institutions seek to understand and satisfy the needs of their users. Surveys offer an outside perspective that helps to identify areas for improvement. The UPV/EHU has been proactive in using surveys to gather student opinions, but now there is a need to move towards a more efficient digital system.

The previously mentioned aspects confirm the fundamental role that surveys play in this context. This is where the **SiREnO** project was born (**S**istema de **R**ecogida de **E**ncuestas **O**nline) which means, "Online Survey Collection System"; an online system that seeks to revolutionize the assessment process, providing students with a more accessible and effective way to express their opinions. This work will address the challenges and benefits of the transition to a telematics system.

Índice general

Índice de tablas

Índice de figuras

1. Introducción

Hoy en día, la mayoría de instituciones que se precien tienen activamente en cuenta las opiniones de sus clientes. El propósito es obtener percepciones subjetivas y niveles de satisfacción por parte de los usuarios en cuestión, con el fin de identificar oportunidades de optimización. Estas encuestas cobran sentido ya que sin hacer uso de un punto de vista externo, identificar las necesidades y problemas de ciertos servicios puede no ser una tarea trivial. Además, las hojas de encuesta otorgan al cliente la sensación de ser escuchado y valorado; la sensación de que su opinión se tiene en cuenta y de que quizás llegue a ser útil en la propia institución para poder mejorar algún área de la misma.

Por ello, la UPV/EHU consta de un sistema de recopilación de opiniones del alumnado mediante encuestas físicas al final de cada cuatrimestre. Estas evalúan aspectos como el trabajo personal de los estudiantes, metodologías utilizadas en la asignatura y la evaluación del docente. El docente entrega las encuestas en formato físico a los alumnos, quienes tienen unos 10 minutos para completarlas. Luego, las hojas son recogidas y almacenadas en un sobre, firmado por el docente y el delegado del aula, antes de entregarlo en secretaría. Una vez entregadas, se organizan y se pasan manualmente, una por una, por un sistema que digitaliza las respuestas; es decir, que aunque todo el proceso sea físico, el resultado final necesita ser digital.

A día de hoy, entre la veintena de centros que componen la UPV/EHU, se estima que el total de alumnos que cursan estudios de grado, master o doctorado ronda los 44.791[1]. Estimando que cada alumno está matriculado de media en unas 10 asignaturas, da como resultado un total de 447.910 encuestas anuales. Casi medio millón de documentos que en caso de una mala gestión, son susceptibles al apilamiento y traspapeleo fácilmente evitable si todo el proceso fuese digital en lugar de únicamente el resultado.

Otro de los problemas derivados de manejar este volumen de documentos es la cantidad de tinta y papel que se utiliza. Más allá del impacto medioambiental que acarrea emplear tal cantidad de recursos, se le suma el coste asociado de estos materiales. Si el que el coste de un folio al por mayor es de aproximadamente 1 céntimo[2] y el coste de una fotocopia es de otros 2 céntimos[3], el coste total aproximado asciende a 13.500€ anuales. Un precio más que considerable que junto con los problemas descritos son el motor para empezar a tomar medidas y buscar alternativas a este sistema de encuestas en papel.

Independientemente de la de adecuación de las preguntas, es obvio que las encuestas tienen problemas intrínsecos y es una realidad que la UPV/EHU quiere cambiar este proceso. Se quiere crear un sistema que permita realizar dichas encuestas de manera telemática para que el procedimiento completo sea puramente online. En este contexto, se introduce **SiREnO** (**Si**stema de **R**ecogida de **En**cuestas **O**nline), un proyecto diseñado específicamente para atender las exigencias de digitalización de encuestas dirigidas al alumnado en la universidad. La digitalización de estos procedimientos evita la necesidad de que el delegado y el docente se desplacen físicamente a la secretaría, optimizando así el uso del tiempo y los recursos involucrados en dicha actividad.

A día de hoy, si un estudiante no pudiera asistir a clase en el momento en que se administra la encuesta, se le privaría del derecho de participar y completar dicha encuesta. Por esta razón, la universidad tiene la intención de ofrecer a los estudiantes el derecho de evaluar la calidad de la enseñanza recibida, incluso si el profesorado no colabora activamente en el proceso. Al realizar estas evaluaciones de forma digital, en el Servicio de Evaluación se dispone de la capacidad de actuar ante la falta de respuesta de un docente. Con esta modalidad en línea, si fuera necesario, se podría extender el plazo para aquellos que no hayan podido participar en la encuesta durante la sesión presencial.

1.1. Razones de la elección del TFG

En primer lugar, mi idea inicial desde que comencé la carrera era hacer un trabajo de fin de grado que pudiera ser útil y/o utilizable en un futuro. Está claro que no voy a poder crear el sistema completo por mí mismo en este plazo de tiempo pero al menos me gustaría crear una base sólida que tenga sentido por sí sola y cuente con las suficientes funcionalidades para que en un futuro, y de manera ideal, pueda ser tomada como raíz de un proyecto más grande. Quiero tener la posibilidad de crear algo propio y como he dicho, me sentiría muy orgulloso si esto siguiese adelante en manos de desarrolladores más expertos y poder decir que el proyecto fue iniciado por mí.

Siendo honestos, el desarrollo web no es uno de mis puntos fuertes puesto que no se profundiza en la carrera y personalmente siempre me he centrado en otras ramas de la informática. Esto me motiva más aún y me plantea un reto junto con la posibilidad de poder generar conocimiento sobre este ámbito que tan demandado está a día de hoy. Otro punto a favor de este trabajo es que las tecnologías a utilizar son de mi elección. Puedo escoger aquellas que crea que mejor se adecuan a la naturaleza de mi solución. Además, me motiva tener que utilizar tecnologías nuevas para mí y así expandir mis conocimientos; algo que considero necesario de cara al futuro, no solo por el hecho de adquirir nuevas competencias, sino también por salir de mi zona de confort y mantener esa capacidad de aprendizaje constante que considero esencial.

[1]https://www.ehu.eus/es/web/gardentasun-ataria/datu-orokorrak
[2]Coste de compra mayorista de papel
[3]Costes fotocopias blanco y negro

2. Planteamiento Inicial

El planteamiento inicial de un proyecto software es un paso crítico para el éxito del proyecto, ya que establece la dirección y los objetivos del mismo, así como los requisitos y las expectativas del usuario, y es la base para todo el proceso de desarrollo posterior.

Este capitulo pretende definir y establecer los objetivos del proyecto, las metas y los requisitos necesarios para su desarrollo exitoso. Además, como parte del planteamiento inicial, se definirán, entre otros, los conceptos usados a lo largo del proyecto, la definición del alcance del proyecto, la selección de la metodología de desarrollo, la planificación de los recursos, etc.

2.1. Definiciones, acrónimos y abreviaturas

En esta sección se listan y explican de manera detallada cada uno de los términos técnicos y los neologismos utilizados en el proyecto.

2.1.1. Términos técnicos

- ***API***: Interfaz de Programación de Aplicaciones, que permite la comunicación e interacción entre diferentes sistemas de software.
- ***REST***: estilo arquitectónico para el desarrollo de servicios web que utiliza los verbos HTTP para realizar operaciones sobre recursos.
- ***API REST***: API que sigue los principios de REST, utilizando HTTP para la comunicación y manipulación de recursos.
- ***JSON***: formato de intercambio de datos ligero y legible por humanos, basado en texto, utilizado para transmitir información estructurada entre sistemas.
- ***Front-end***: parte de un sistema o aplicación web que se encarga de la presentación y la interacción con el usuario final.
- ***Back-end***: parte de un sistema o aplicación web que se encarga del procesamiento y la lógica detrás de la interfaz de usuario, generalmente invisible para el usuario final.
- ***Endpoint***: punto final de una API, que representa una dirección URL específica a la cual se puede enviar una solicitud para obtener o manipular datos.

- ***JSON Web Token***: mecanismo para transmitir información de forma segura entre partes como un objeto JSON compacto y autónomo, utilizado para la autenticación y autorización en aplicaciones web.
- ***Listener***: componente o función que está a la espera de eventos o notificaciones para responder o actuar en consecuencia.
- ***Framework***: entorno o conjunto de herramientas, librerías y componentes predefinidos que facilitan el desarrollo de aplicaciones, proporcionando una estructura y funcionalidades comunes.

2.1.2. Vocabulario relacionado con el proyecto

- ***Formulario***: un formulario se define como un documento, en formato digital, que consta de un conjunto de preguntas específicas destinadas a ser respondidas por los alumnos con el propósito de recopilar sus opiniones.
- ***Encuesta***: una encuesta se puede entender como una entidad que engloba un formulario asociado. Al acceder a una encuesta, se abre el formulario correspondiente, lo cual implica que una encuesta posee, a su vez, un formulario que debe ser completado por los participantes.
- ***Situación docente***: la situación docente es un concepto que agrupa múltiples entidades. Una situación docente engloba la información que establece qué docente está a cargo de qué asignatura, impartida a qué alumnos, en qué curso, en qué grupo, en qué grado y en qué centro educativo. Cada situación docente es única y sirve para reunir estos elementos en un objeto único. Además, cada situación docente está asociada a una encuesta que, a su vez, cuenta con un formulario para que los alumnos de dicha situación docente evalúen tanto la asignatura como al docente correspondiente.
- ***Campaña***: una campaña se refiere a un período de tiempo determinado por la administración, que marca la duración de una situación docente. Durante dicho periodo de duración de la campaña, los alumnos deberían tener la oportunidad de responder a la encuesta asociada. Es importante destacar que, por lo general, las situaciones docentes que pertenecen, por ejemplo, al mismo cuatrimestre comparten la misma campaña, asumiendo que todas comienzan y finalizan al mismo tiempo. Como consecuencia, una única campaña agrupa todas las situaciones docentes deseadas, las cuales comparten el mismo período de validez.
- ***Campaña válida***: una campaña válida se refiere a aquella que se encuentra dentro de su período de vigencia, es decir, que el periodo de vida establecido por la administración ya ha comenzado pero aún no ha expirado. Debido a que una campaña está asociada a una o más situaciones docentes, **la validez de la situación docente la marca la validez de la campaña**. Por ejemplo, de manera hipotética, una campaña podría ser denominada como "2º cuatrimestre del curso 22/23" y abarcar desde el 10 de enero hasta el 15 de junio de 2023. Durante este período, se consideraría que la campaña es válida, lo que implica que las situaciones docentes asociadas a dicha campaña también son válidas.

- ***Abrir encuesta***: durante el período de validez de una campaña, se espera que los alumnos tengan la oportunidad de responder a la encuesta asociada en algún momento. Para que esto sea posible, durante dicho período de validez, tanto el docente como la administración tienen la facultad de abrir la encuesta. Al abrir la encuesta, se establece un período de apertura en el cual los alumnos pueden responder al formulario asociado. Es importante destacar que este período de apertura debe estar dentro del período de validez de la campaña.
- ***Cerrar encuesta***: de manera contraria, existe la posibilidad de cerrar la encuesta, ya sea de forma automática o manual. Cuando la encuesta se cierra, significa que el plazo establecido para responder al formulario asociado ha expirado y, por lo tanto, ya no es posible enviar respuestas a menos que la encuesta sea abierta nuevamente.

2.2. Objetivos

El objetivo principal del proyecto **SiREnO** es crear un sistema online capaz de sustituir al sistema actual de encuestas físicas que se realizan en la universidad al final de cada cuatrimestre. El sistema debe poder permitir: definir las encuestas de opinión, definir las personas que tienen que rellenar dicha encuesta, quién tiene que poder activar/desactivar la posibilidad de responder a la encuesta, establecer quién recibe los resultados, etc. Se quiere ofrecer un diseño intuitivo, limpio y *user friendly* y que además cumpla con los requisitos impuestos por el interesado. Además del principal, existen varios objetivos secundarios.

Simplificar el proceso de gestión de las encuestas en la universidad. Actualmente, la gestión de medio millón de documentos físicos al final de cada curso es tediosa e ineficiente. SiREnO busca mejorar y facilitar la gestión de encuestas de opinión en la UPV/EHU, aprovechando la oportunidad de mejorar algo existente. Ejemplos notables de proyectos exitosos que han reinventado conceptos existentes incluyen Amazon en el comercio electrónico y Uber en el servicio de movilidad.

Crear informes de autoevaluación para docentes. Tratar de desarrollar un sistema que genere informes detallados que permitan a los docentes realizar una autoevaluación e identificar áreas de mejora. Estos informes deben proporcionar información sobre la evolución a lo largo de los años y permitir comparaciones con promedios de referencia para una evaluación más precisa.

Implementar medidas de seguridad y privacidad robustas. El objetivo es asegurar que el sistema SiREnO cumpla con altos estándares de seguridad y privacidad. Esto implica incorporar protocolos y tecnologías que protejan los datos confidenciales y garanticen la integridad de la información recopilada durante las encuestas. Además, se deben establecer políticas de acceso y control de usuarios para asegurar la confidencialidad de los datos.

Optimizar el proceso de apertura y cierre de encuestas. Se quieren desarrollar mecanismos eficientes que simulen el proceso de entrega y recolección de encuestas en papel, pero de manera ágil y automatizada en el entorno online. Esto implica establecer plazos

de apertura y cierre de encuestas dentro del período de validez de la campaña correspondiente, garantizando una experiencia fluida y fácil para los alumnos para responder a los formularios asociados.

Evaluar y demostrar mis aptitudes técnicas. Demostrar como alumno de cuarto curso de ingeniería informática, los conocimientos adquiridos durante mi formación y la capacidad de ponerlos a prueba para diseñar y desarrollar un sistema online de encuestas que sea fácil de usar, accesible y seguro para los usuarios cuyo el objetivo es obtener información valiosa sobre la satisfacción del alumnado con respecto al curso y el profesorado.

Demostrar habilidades de investigación. Como alumno, esta es una oportunidad donde puedo demostrar habilidades de investigación a la hora de desarrollar el sistema. Por un lado, investigación a cerca de cómo diseñar y desarrollar un sistema de estas magnitudes, ya que, personalmente no me he enfrentado a uno de este calibre ni con estas tecnologías. El análisis de los datos recopilados a través de las encuestas con el propósito de generar informes implica un proceso que demuestra mi habilidad para transformar datos crudos en información significativa y relevante. Este proceso me permite desarrollar conocimiento útil a partir de la información subyacente en los datos recopilados. Esto puede ayudar a mejorar la calidad de la enseñanza y ser útil para otros proyectos de investigación en el futuro.

Satisfacción personal y autorrealización. Personalmente me he visto involucrado, al igual que todos mis compañeros, en una gran cantidad de proyectos diferentes y por desgracia para mi, varios de ellos no me han aportado lo que pensaba que harían. Quizás por un mal planteamiento del proyecto o por no haberlos abordado de la manera que se pretendía que lo hiciéremos. Es por ello que quiero comprobar mi capacidad de llevar un proyecto de estas magnitudes de principio a fin de una manera profesional.

Existen otros objetivos detrás del desarrollo e implantación de SiREnO que se podrían llegar a medir una vez terminado por completo el sistema y esté en marcha, cosa que no se pretende con este TFG. Por tanto, no voy a entrar a describirlos en detalle pero por dar un ejemplo, uno de ellos puede ser:

Ahorrar tiempo y mejorar la eficiencia del profesorado. Se quiere agilizar el proceso relacionado con la creación, publicación, respuesta, etc. De las encuestas para que estos trámites no sean razón de pérdida de tiempo. La escuela cuenta con 5704 docentes activos[4]; si cada uno tarda 10 minutos de media en ir a secretaría a por las encuestas y posteriormente regresar a entregar el sobre, nos queda un total de 950 horas totales malgastadas y eso suponiendo que cada docente imparte una sola asignatura en un solo grupo; cosa que generalmente no sucede.

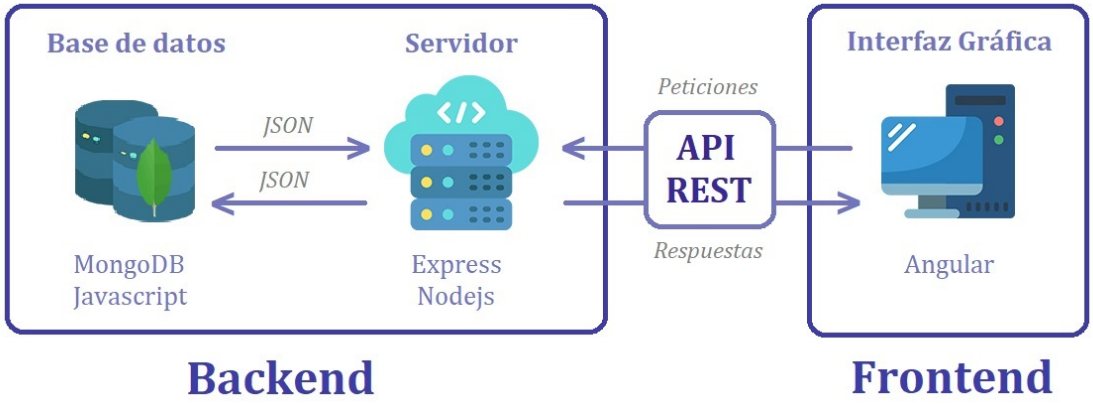
[4]https://www.ehu.eus/docentes

2.3. Arquitectura

Para este proyecto se va a usar una arquitectura basada en el Stack MEAN tal y como se puede ver en la Figura 2.1. MEAN es el acrónimo que referencia las arquitecturas desarrolladas con **M**ongoDB, **E**xpress.js, **A**ngular.js y **N**ode.js. Debido a que los cuatro servicios hacen uso del lenguaje *JavaScript*, la arquitectura MEAN está diseñada para hacer que la creación de aplicaciones web en el lenguaje *JavaScript* y el manejo de JSONs sea bastante más llevaderos.

El sistema se divide en **back-end** y **front-end**. El back-end contiene toda la lógica de negocio del sistema. En ella se almacena todo el código y es la parte encargada del procesamiento de las peticiones, la conexión con la base de datos, etc. El front-end, en cambio, es la sección encargada de interactuar con el usuario, es básicamente la interfaz gráfica o la parte visible SiREnO.

Para comunicar ambas partes, el back-end con el front-end y viceversa, se va a crear una API REST. Las APIs son conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones. Que la API sea REST significa que la información se entrega por medio del protocolo HTTP. Dicho de otro modo, la API REST no es más que un conjunto de URLs que va a tener el servidor para que el cliente pueda pedir recursos. El servidor haciendo uso de la base de datos podrá devolver dichos recursos de la manera esperada al cliente (tal y como se puede ver en el esquema de la Figura 2.1). Por ejemplo, el servidor puede tener una ruta que se llame '*servidor.com/123/encuestas*' y cuando desde el front-end se realice una petición al servidor haciendo uso de dicha URL, el servidor responderá, por ejemplo, con el conjunto de encuestas pendientes para el alumno cuyo id es 123.



2.1. Figura: Arquitectura de SiREnO con stack MEAN.

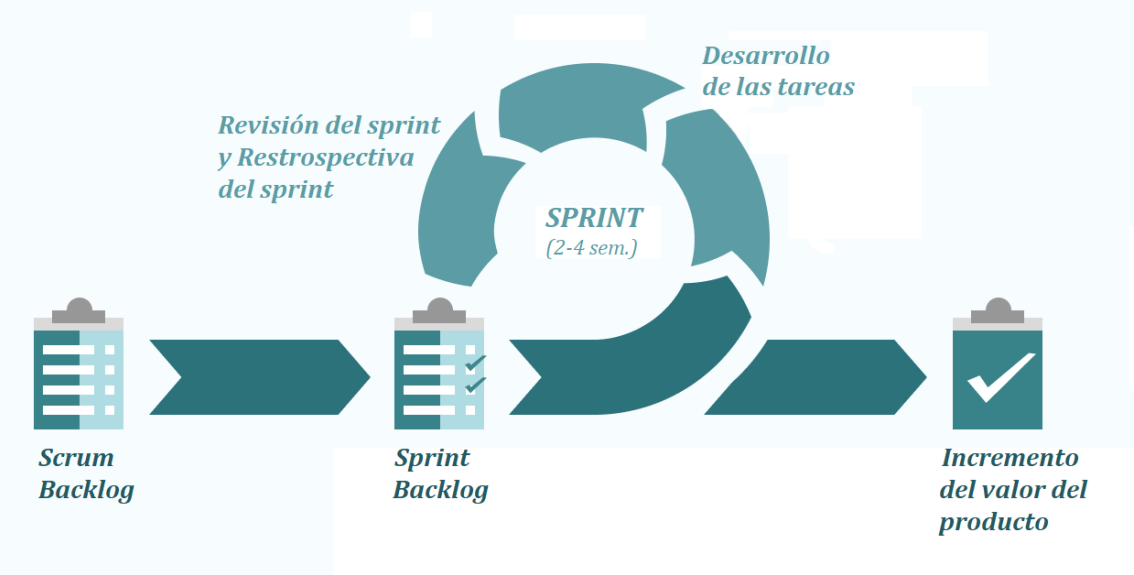
2.4. Alcance

Una metodología de trabajo define las etapas de un proyecto y qué hacer en cada una de ellas, así como qué necesita y genera cada etapa. Debido a esto, es imprescindible definir una correcta metodología de trabajo para facilitar la planificación, el seguimiento y control del proyecto con el fin de evitar terminar ahogándote en el mismo. Planificar el proyecto entero antes de empezarlo no es una tarea sencilla ya que a priori no se sabe cuál es la magnitud real del mismo ni los posibles problemas que pueden llegar a aparecer.

Debido a mi incertidumbre inicial de cómo abordar el trabajo, he escogido emplear metodologías ágiles. Estas metodologías ágiles, a diferencia de las tradicionales, se centran en la planificación adaptativa. En lugar de crear un plan inicial (generalmente bastante rígido) el cual se deba seguir durante el proyecto; se crean pequeñas etapas o fases más reducidas en el tiempo con el objetivo de poder responder ante el cambio.

Se va a utilizar **Scrum** como marco de trabajo para el desarrollo ágil de software basado en un proceso iterativo e incremental. La metodología Scrum se basa en la definición y ejecución de sprints. Un sprint es simplemente un periodo de tiempo en el que se va a desarrollar una tarea o subtarea del proyecto. Los sprints deben tener una duración acotada generalmente de 2 a 4 semanas. El proceso que se va a llevar a cabo en este proyecto es el siguiente:

1. Definir el listado de tareas del proyecto *(Scrum backlog)*.
2. Por cada tarea (sprint) en el *Scrum backlog* se va a realizar el proceso representado en la Figura 2.2 y detallado a continuación:
    a) **Planificación del sprint.** Identificar y organizar las tareas al comienzo de cada sprint (*sprint backlog*). También es una buena práctica planificar de manera aproximada las tareas del sprint en el tiempo.
    b) **Ejecución del sprint.** Desarrollar las tareas definidas en el sprint y concretar reuniones con el cliente si fuera necesario.
    c) **Revisión del sprint.** Al finalizar cada sprint, se lleva a cabo una reunión del equipo (yo) con el cliente (tutor) para presentarle los requisitos completados tras el sprint, recibir impresiones y valorar el producto. En este punto el cliente debe dar a conocer si está en desacuerdo con el desarrollo de alguna funcionalidad para así poder rehacerla a tiempo y evitar que el problema contamine el resto de módulos del proyecto. Si el cliente desea hacer algunos cambios en las funcionalidades se volverá al paso *a)*.
    d) **Retrospectiva del sprint.** Al finalizar el sprint, el equipo (yo) analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad.

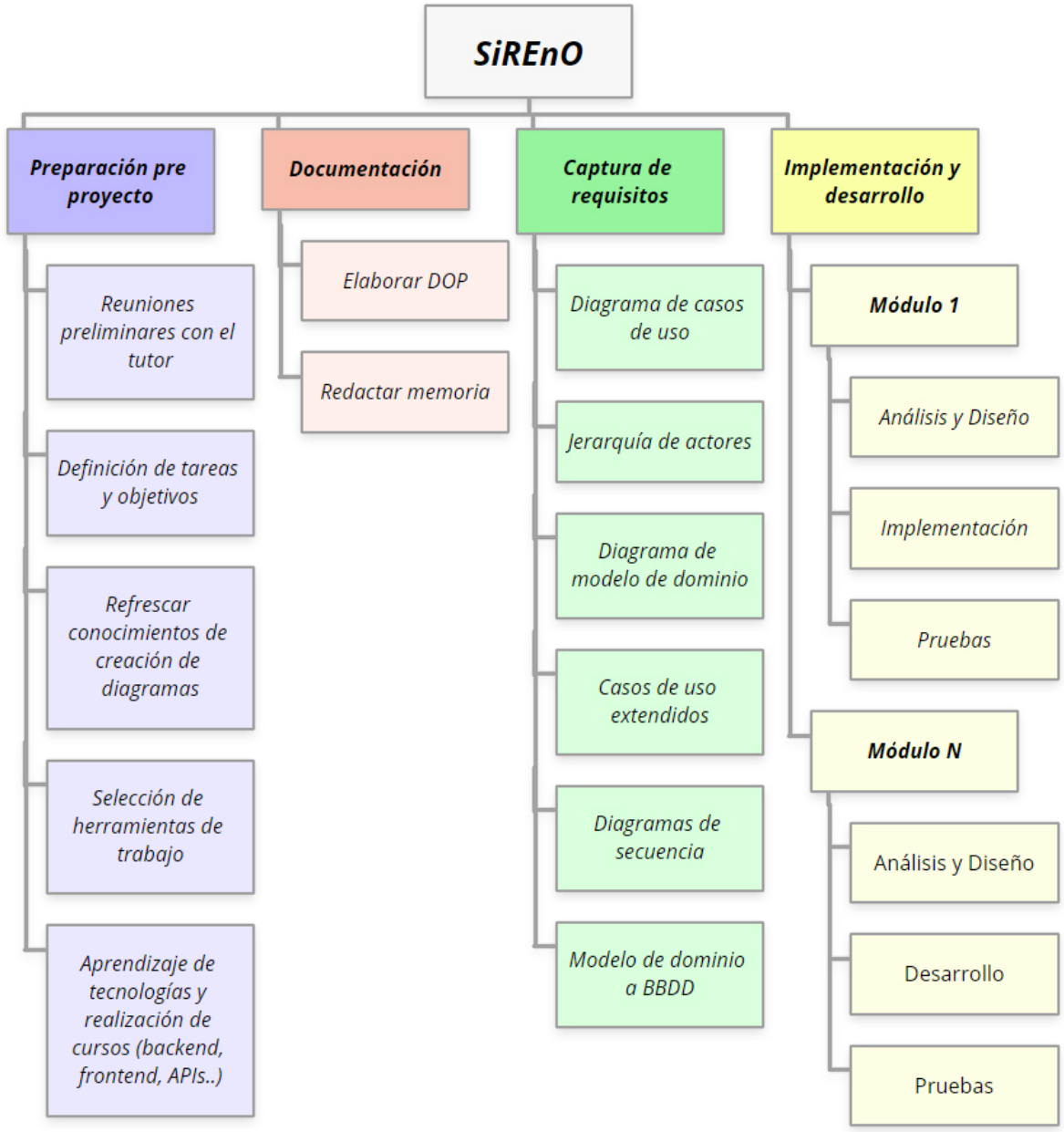


2.2. Figura: Desglose de etapas en la metodología Scrum.

La clave y el sentido de esta metodología es ir añadiendo continuamente más valor al producto final tras la ejecución completa de cada sprint, ya que lo fundamental es el producto y no tanto el proyecto. Gracias a Scrum se consigue que el cliente sea parte activa del proceso puesto que debe retroalimentar cada sprint con sus opiniones al respecto.

Gracias a los sprints y poder recibir *feedback* continuo de parte del cliente no nos vemos tan expuestos al riesgo de crear un producto que no se ajuste a las necesidades del mismo. En un modelo rígido, si durante la fase del desarrollo software se considerase necesario añadir una nueva funcionalidad al producto, sería necesario paralizar el propio desarrollo para volver atrás a la fase anterior y realizar el análisis y diseño de dicha funcionalidad. En Scrum, en cambio, gracias a los sprints es posible llegar a integrar dicho cambio al final del sprint.

El alcance del proyecto se define mediante una Estructura de Descomposición de Trabajo (EDT) en la cual se recogen las tareas y subtareas correspondientes al desarrollo de SiREnO, (Ver Figura 2.3). El EDT cuenta con 4 secciones principales: *preparación pre-proyecto*, *documentación*, *captura de requisitos* e *implementación y desarrollo*.



2.3. Figura: Estructura de Descomposición de Trabajo de SiREnO.

La preparación pre-proyecto son todas aquellas tareas que se realizan antes de comenzar o cuando recién se está comenzando el proyecto con el objetivo de sumergirse en él y prepararse para abordarlo. Algunas tareas son: reuniones con el cliente para conocer el alcance del proyecto, descubrir las necesidades del sistema para posteriormente poder definir las tareas y objetivos del proyecto, seleccionar las herramientas de trabajo y adquirir conocimiento de las mismas si fuera necesario, etc.

La documentación a su vez se divide en dos partes. Por un lado, la elaboración del Documento de Objetivos del Proyecto (DOP), donde se define la descripción del proyecto, los objetivos buscados, el alcance del proyecto... Y por otro lado, está la Memoria del Proyecto, digamos la segunda parte de la documentación, en ella se recoge toda la información asociada al proyecto y al desarrollo del mismo.

La Captura de requisitos consiste en saber qué sistema debe construirse y entender el contexto del mismo. Está formado, a su vez, por la realización del diagrama de casos de uso, la jerarquía de actores, el modelo de dominio, los casos de uso extendidos, los diagramas de secuencia y la transformación del modelo de dominio a base de datos.

La implementación y desarrollo se ha dividido en diferentes módulos para poder gestionar mejor cada funcionalidad y trabajarla individualmente. Estos módulos hacen referencia a las funcionalidades que se deben implementar en el sistema. Cada módulo tiene sentido propio y el desarrollo individual de cada uno generará una versión más completa de SiREnO. **El desarrollo de cada módulo será considerado un sprint**. Un módulo consta a su vez de 3 fases: análisis y diseño, implementación y pruebas.

Se debe destacar que el sistema consta de muchas funcionalidades y no se van a poder desarrollar todas. Las funcionalidades a desarrollar no están decididas desde el comienzo del proyecto y éstas se irán concretando a medida que avance el mismo. Por tanto, el último bloque del EDT se deja entreabierto para decidir posteriormente cuáles serán dichos módulos. Además, si se quiere seguir una metodología ágil no tiene mucho sentido predefinir secuencialmente el orden de las tareas ya que estamos cayendo en la rigidez que precisamente se quiere evitar.

La explicación detallada de cada paquete de trabajo se muestra a continuación en forma de tabla.

Descripción del paquete de trabajo
Tarea: Reuniones preliminares con el tutor.
Duración estimada: 4 reuniones de hora y media cada una: 6 horas.
Descripción: Reuniones acordadas con el tutor del Trabajo Fin de Grado para poder concretar cuáles van a ser las funcionalidades y requisitos del sistema a desarrollar.
Entradas: - - -
Salidas/Entregables: Apuntes sobre los puntos expuestos.
Recursos Necesarios: - - -
Precedencias: - - -

2.1. Tabla: Descripción tarea: Reuniones preliminares con el tutor.

Descripción del paquete de trabajo

Tarea: Leer la documentación de referencia y buscar proyectos semejantes.

Duración estimada: 5 horas.

Descripción: Leer y entender otros Trabajos de Fin de Grado realizados por alumnos anteriores con el fin de comprender cómo se debe abordar un proyecto de esta magnitud.

Entradas: Trabajos de Fin de Grado de otros alumnos.

Salidas/Entregables: - - -

Recursos Necesarios: Trabajos de Fin de Grado de otros alumnos.

Precedencias: Reunión con el tutor para definir el tipo de proyecto que quiero hacer.

2.2. Tabla: Descripción tarea: Leer documentación y buscar proyectos semejantes.

Descripción del paquete de trabajo

Tarea: Definición de tareas y objetivos.

Duración estimada: 6 horas.

Descripción: Entender e interiorizar todos los requisitos del sistema para poder redactar un conjunto de tareas a realiza y objetivos a cumplir para suplir todas las necesidades.

Entradas: - - -

Salidas/Entregables: Listado de tareas a realizar con el fin de lograr los objetivos.

Recursos Necesarios: - - -

Precedencias: Reuniones preliminares con el tutor.

2.3. Tabla: Descripción tarea: Definición de tareas y objetivos.

Descripción del paquete de trabajo

Tarea: Refrescar conocimientos de creación de diagramas.

Duración estimada: 10 horas.

Descripción: Refrescar conocimientos obtenidos en la asignatura de ADSI para desarrollar de manera competente todos los diagramas necesarios para mi sistema.

Entradas: - - -

Salidas/Entregables: Conocimiento re-adquirido.

Recursos Necesarios: Apuntes de tercero de carrera.

Precedencias: - - -

2.4. Tabla: Descripción tarea: Refrescar conocimientos de creación de diagramas.

Descripción del paquete de trabajo

Tarea: Selección de herramientas de trabajo y configuración de las mismas.

Duración estimada: 4 horas.

Descripción: Una vez conocidos los requerimientos del proyecto y la naturaleza del mismo se deben escoger las herramientas de trabajo tanto hardware como software que se van a emplear para desarrollarlo y configurarlas en mi entorno de trabajo.

Entradas: - - -

Salidas/Entregables: Listado de herramientas que se van a usar durante el desarrollo del sistema.

Recursos Necesarios: - - -

Precedencias: Definición de tareas y objetivos.

2.5. Tabla: Descripción tarea: Selección y configuración de herramientas de trabajo.

Descripción del paquete de trabajo

Tarea: Aprendizaje de tecnologías y realización de cursos (back-end, front-end, APIs...)

Duración estimada: 40 horas.

Descripción: Leer documentación y realizar cursos online de tecnologías que no manejo pero voy a necesitar emplear para desarrolar una o varias partes del proyecto.

Entradas: - - -

Salidas/Entregables: Conocimiento adquirido.

Recursos Necesarios: - - -

Precedencias: Selección de herramientas de trabajo.

2.6. Tabla: Descripción tarea: Aprendizaje de tecnologías y realización de cursos.

Descripción del paquete de trabajo

Tarea: Elaborar el Documento de Objetivos del Proyecto (DOP).

Duración estimada: 40 horas.

Descripción: Redacción del Documento de Objetivos del Proyecto donde se intenta identificar las intenciones del proyecto.

Entradas: - - -

Salidas/Entregables: Documento de Objetivos del Proyecto (DOP).

Recursos Necesarios: Un ordenador, acceso a internet y Overleaf.

Precedencias: - - -

2.7. Tabla: Descripción tarea: Elaborar el Documento de Objetivos del Proyecto.

Descripción del paquete de trabajo

Tarea: Redactar la memoria del Trabajo Fin de Grado (TFG).

Duración estimada: 55 horas.

Descripción: Redacción de la memoria del proyecto junto con la creación de las tablas, las imágenes y los diagramas necesarios.

Entradas: Documento de Objetivos del Proyecto (DOP).

Salidas/Entregables: Redacción de la memoria del proyecto junto con la creación de las tablas, las imágenes y los diagramas necesarios.

Recursos Necesarios: Un ordenador, acceso a internet, Overleaf, Gloomaps, TeamGantt...

Precedencias: Haber realizado de manera exitosa todas las tareas del EDT.

2.8. Tabla: Descripción tarea: Redactar la memoria del Trabajo Fin de Grado.

Descripción del paquete de trabajo

Tarea: Diagrama de casos de uso.

Duración estimada: 6 horas.

Descripción: Creación del diagrama de comportamiento de las funcionalidades del sistema e interacción de los diferentes actores.

Entradas: - - -

Salidas/Entregables: Diagrama casos de uso.

Recursos Necesarios: Un ordenador con acceso a internet y Gloomaps.

Precedencias: - - -

2.9. Tabla: Descripción tarea: Diagrama de casos de uso.

Descripción del paquete de trabajo

Tarea: Jerarquía de actores.

Duración estimada: 30 minutos.

Descripción: Definición más detallada de los posibles actores del sistema junto con sus roles y funcionalidades asociadas.

Entradas: Diagrama de casos de uso.

Salidas/Entregables: Diagrama jerarquía de actores.

Recursos Necesarios: Un ordenador con acceso a internet y Gloomaps.

Precedencias: Diagrama Casos de Uso.

2.10. Tabla: Descripción tarea: Jerarquía de actores.

Descripción del paquete de trabajo

Tarea: Diagrama de modelo de dominio.

Duración estimada: 10 horas.

Descripción: Creación del diagrama que representa las entidades de los datos que se necesitan almacenar y defunción de interacción entre dichas entidades.

Entradas: - - -

Salidas/Entregables: Diagrama de modelo de dominio.

Recursos Necesarios: Un ordenador con acceso a internet y Gloomaps.

Precedencias: Diagrama de casos de uso.

2.11. Tabla: Descripción tarea: Diagrama de modelo de dominio.

Descripción del paquete de trabajo

Tarea: Casos de uso extendidos.

Duración estimada: 15 horas.

Descripción: Desarrollo más a fondo de cada uno de los casos de usos definidos anteriormente en su diagrama correspondiente. Por cada caso de uso se describe a fondo su funcionalidad y flujo de eventos así como la definición de una interfaz gráfica dedicada en caso de tenerla.

Entradas: Diagrama de casos de uso.

Salidas/Entregables: Diagrama de casos de uso extendidos.

Recursos Necesarios: Un ordenador con acceso a internet, Gloomaps, Photoshop, Paint y Overleaf.

Precedencias: Diagrama de casos de uso.

2.12. Tabla: Descripción tarea: Casos de uso extendidos.

Descripción del paquete de trabajo

Tarea: Diagramas de secuencia.

Duración estimada: 20 horas.

Descripción: Diseño de los diagramas de secuencia necesarios para cada funcionalidad con el objetivo de modelar las interacciones entre objetos en el proyecto.

Entradas: - - -

Salidas/Entregables: conjunto de diagramas de secuencia.

Recursos Necesarios: Un ordenador con acceso a internet y Gloomaps.

Precedencias: - - -

2.13. Tabla: Descripción tarea: Diagramas de secuencia.

Descripción del paquete de trabajo

Tarea: Modelo de dominio a BBDD.

Duración estimada: 10 horas.

Descripción: Convertir el modelo de dominio en una base de datos e implementarlo.

Entradas: Modelo de Dominio.

Salidas/Entregables: Base de datos.

Recursos Necesarios: Un ordenador con acceso a internet, Gloomaps y Visual Studio Code.

Precedencias: Modelo de dominio.

2.14. Tabla: Descripción tarea: Modelo de dominio a BBDD.

Descripción del paquete de trabajo

Tarea: Módulo x:

Duración estimada: 2-4 semanas, 30-60 horas.

Descripción: En primer lugar se debe llevar a cabo el análisis y diseño de la funcionalidad del módulo en cuestión. Una vez hecho, se implementa, es decir, se desarrolla el código de la funcionalidad x. Por último se definen y realizan las pruebas pertinentes.

Entradas: - - -

Salidas/Entregables: El conjunto de funcionalidades desarrolladas.

Recursos Necesarios:

Precedencias: El módulo anterior en caso de haberlo

2.15. Tabla: Descripción tarea: Módulo/Funcionalidad x.

2.5. Planificación Temporal

Tal y como se ha comentado anteriormente, este proyecto sigue una metodología ágil, es por ello que se ha dividido la planificación en sprints. Si se quieren respetar los principios de Scrum, no tendría sentido definir los sprints de antemano ya que se estaría cayendo en la misma dinámica que en una metodología tradicional. Estaríamos definiendo un orden secuencial entre las tareas.

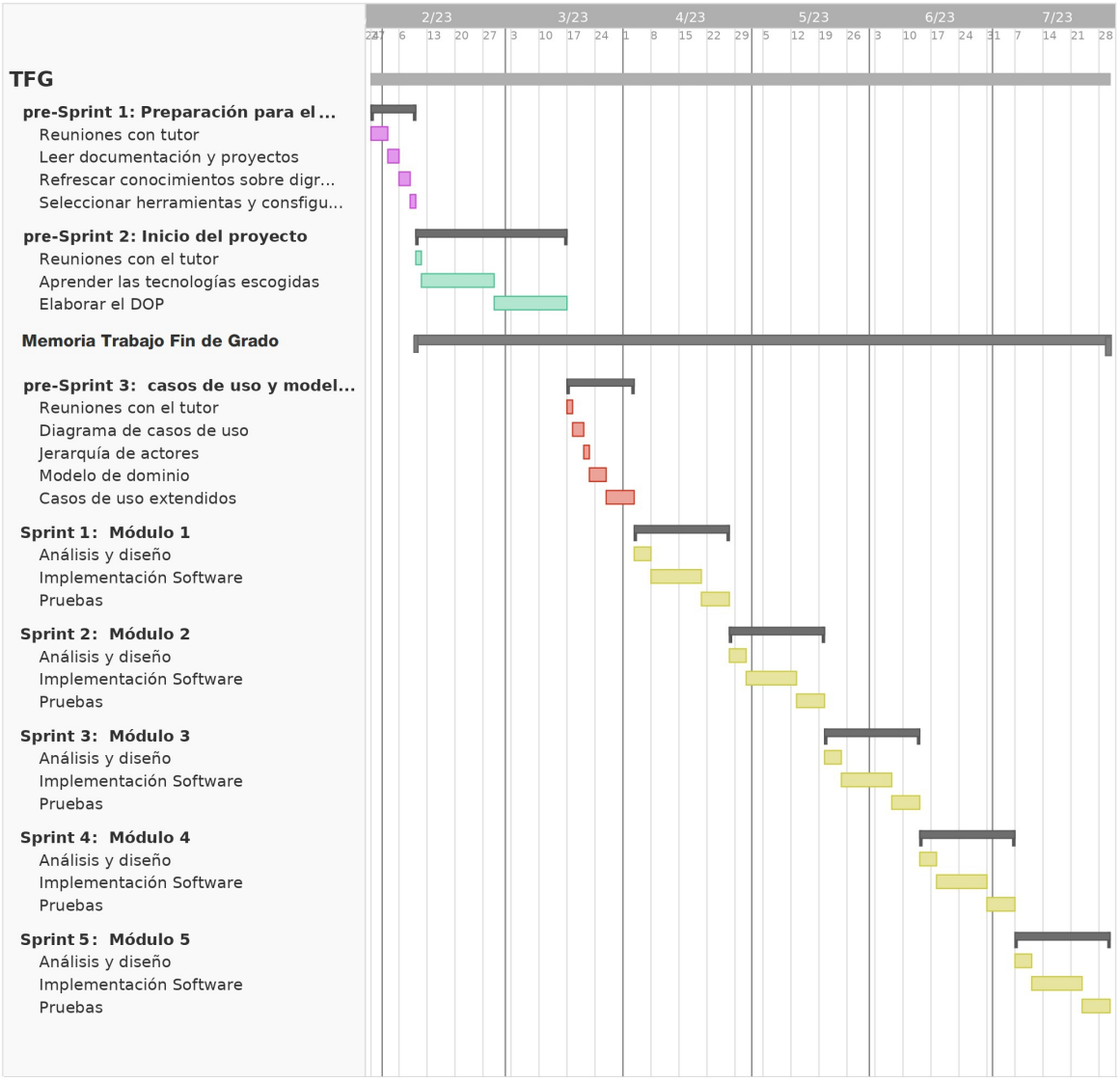
La esencia de la metodología ágil es poder reaccionar de manera eficiente ante cambios o imprevistos y por ello una vez acabado un sprint, y solo entonces, se debería escoger cuáles son las tareas que le suceden. De todos modos, se han definido una serie de sprints. Cada uno de ellos, a su vez, compuesto por un conjunto de tareas (*sprint backlog*). Cada sprint es definido de manera orientativa y nadie garantiza que lo descrito a continuación sea lo que se va a seguir durante el proyecto ya que existe el riesgo de que las circunstancias del mismo cambien. Las tareas se han sacado del diagrama EDT (Figura 2.3) pero las tareas que pertenecen al mismo bloque en el EDT no tienen porqué estar en el mismo sprint.

Una vez definidos los sprints y sus tareas, hay que calcular el tiempo que se necesita para cada sprint en base a la duración descrita en los paquetes de trabajo que se ha hecho tras la definición del EDT. Semanalmente se trabajarán 3 horas diarias en el proyecto durante los días laborables; por tanto la carga de trabajo semanal rondará las 15 horas. Este número puede variar en función de las circunstancias en las que me encuentre en el futuro y puede ser ampliable tanto en el número horas de trabajo diario como en el número de días de en el que se trabaja por semana.

Dado que en el presente momento aún no se ha definido completamente qué funcionalidades del sistema serán implementadas, se mantendrá cierta flexibilidad con los sprints dedicados al desarrollo software del sistema. El número de sprints irá ligado al número de módulos a desarrollar. Tal y como se indica entre las tablas 2.2 y 2.14, la carga horaria de las tareas previas a comenzar a desarrollar los distintos módulos es de aproximadamente 137.5 horas. Si tenemos en cuenta que la carga total de trabajo del TFG deberían ser unas 400 horas, quedan 262.5 horas disponibles para mayormente dedicarlas a desarrollo, pruebas y documentación de los módulos. En la tabla 2.15 se ha estimado que el desarrollo de un módulo dura entre 30 y 60 horas. Pongamos que la duración media es de 50 horas, por tanto se van a poder implementar de manera completa entre 5 y 6 módulos.

Es **importante** destacar que las divisiones y agrupaciones de tareas que preceden al desarrollo, en lugar de considerarse sprints propiamente dichos, se consideran fases iniciales previas a los sprints puros. Estas fases iniciales han sido denominadas "**pre-sprints**". Esto se debe a que en este proyecto, los sprints se refieren a cada uno de los desarrollos de los módulos del sistema. Sin embargo, los pre-sprints se manejarán de manera similar, salvando las diferencias, aunque también se llevarán a cabo reuniones de evaluación con el cliente al final de cada uno. La distinción realizada busca diferenciar entre un pre-sprint que representa la fase previa al desarrollo y un sprint puramente dedicado al desarrollo de software.

El tiempo dedicado a cada tarea y agrupación de tareas (bien en pre-sprints o en sprints) se representa mediante el *diagrama Gantt* de la Figura 2.4 en la siguiente página. Este diagrama parte de la división del conjunto de tareas en sprints. A la izquierda del diagrama se encuentra el listado de las tareas definidas que se deben realizar para llevar a cabo el proyecto. En la misma fila de la tarea, se representa la duración estimada, en semanas.



2.4. Figura: Diagrama Gantt para el tiempo de dedicación previsto.

La tarea llamada *Memoria Trabajo Fin de Grado* abarca desde el inicio hasta el fin del proyecto ya que este documento se irá rellenando a medida que se avance en el desarrollo del sistema. Es una tarea simultanea al resto, ya que, lo común es documentar el desarrollo a medida que va teniendo lugar.

Es relevante considerar que, a pesar de la secuencialidad aparente de las tareas, su ejecución no está condicionada por una dependencia estricta con respecto a la tarea anterior, a menos que se indique específicamente. Esta estructura secuencial se emplea como un medio para representar visualmente la distribución temporal y estimar el tiempo necesario para completar cada tarea de manera individual, dado que el proyecto es llevado a cabo únicamente por un único participante.

En la página siguiente (Ver tabla 2.16) se presenta un desglose detallado de la estructura jerárquica de las tareas agrupadas y las subtareas que las componen, por si en el *diagrama Gantt* no se aprecia bien. El color de cada tarea es el mismo que se ha utilizado en el diagrama de la Figura 2.4. Para los últimos sprints relacionados con los módulos, como se ha dicho, se toma de manera orientativa 50 horas de trabajo. 50 horas de las cuales aproximadamente 15 % del tiempo se dedica en el *análisis y diseño*, el 55 % para *implementación software* y el último 30 % para las *pruebas*

Definición y listado de Sprints

Número Sprint	Duración
pre-sprint 1: Preparación para el proyecto.	**30 horas**
Reuniones con el tutor (describir los objetivos y tareas del proyecto).	12 horas
Leer la documentación de referencia y buscar proyectos semejantes.	5 horas
Refrescar los conocimientos sobre creación de diagramas.	10 horas
Seleccionar las herramientas, tecnologías y bibliotecas que se adecuen al proyecto y realizar la configuración pertinente de las mismas.	3 horas
pre-sprint 2: Inicio del proyecto.	**82 horas**
Reuniones con el tutor (definir concretamente el trabajo y describir las funcionalidades y objetivos del sistema.)	2 horas
Aprender y documentarse sobre las tecnologías escogidas.	40 horas
Elaborar el Documento de Objetivos del Proyecto.	40 horas
pre-sprint 3: Definición y desarrollo; casos de uso y modelo de dominio.	**24.5 horas**
Reuniones con el tutor (definir las funcionalidades concretas de cada rol en el sistema.)	3 horas
Diagrama de casos de uso.	6 horas
Jerarquía de actores.	0.5 horas
Modelo de dominio.	10 horas
Diagrama de casos de uso extendidos.	15 horas
Sprint 1: Módulo 1:	50 horas
Análisis y diseño	7.5 horas
Implementación software	27.5 horas
Pruebas	15 horas

2.16. Tabla: Agrupamiento de tareas del EDT en sprints.

2.6. Herramientas

En esta sección se listan las herramientas que se van a emplear a lo largo del desarrollo y documentación del proyecto. Estas herramientas se pueden dividir, por un lado, en aquellas usadas para documentación, diagramas, elementos gráficos... Y por otro lado, en herramientas de desarrollo de software, entornos, frameworks, lenguajes de programación...

Herramientas para documentación, diagramas y gráficos:

- *Overleaf* es un editor colaborativo de LaTeX que se utiliza para escribir, editar y publicar documentos. Que sea colaborativo ha facilitado el compartir las versiones de la documentación con el tutor a medida que se iba completando la misma.
- *Teamgantt* es un software online de programación de proyectos que facilita mucho la tarea de crear diagramas Gantt.
- *Photoshop*, usado para realizar algunos diagramas para la documentación es conocido por ser uno de los editores, desarrollado por Adobe, de fotografías más utilizado a día de hoy .
- *Paint* junto con *Photoshop*, también se ha usado usado para realizar algunos diagramas para la documentación
- *Gloomaps* es una herramienta de creación de diagrama online simple y rápida. Los de casos de uso, modelo de dominio y los diagramas de secuencia se han hecho con esta herramienta.
- *Visual Paradigm Online* Conjunto de herramientas en línea que incluye, creador de gráficos, creador de diagramas, etc.

Herramientas para el desarrollo de software:

- *Visual Studio Code*, desarrollado por Microsoft para Windows, ha sido usado como un editor de código principal en el transcurso del proyecto.
- *JavaScript* es un lenguaje de programación para páginas web interactivas ejecutado en cliente y en servidor. En el cliente gestiona interacciones con botones, acciones como mover el ratón, modificar el *HTML* y el *CSS*, etc. En el servidor: acceder a bases de datos, cómputos, etc.
- *TypeScript* es un lenguaje de programación superconjunto de *JavaScript*, que esencialmente añade tipos estáticos y objetos basados en clases. Será principalmente usado para el desarrollo del front-end con Angular.
- *HTML*, utilizado principalmente para implementar la interfaz gráfica del sistema, es un lenguaje de marcado para estructurar y presentar contenido en la web y una tecnología fundamental de Internet.

- *CSS* es un lenguaje de estilos ligado al *HTML* que modifica la apariencia de la página web.
- *Nodejs* es un entorno en tiempo de ejecución multiplataforma. Este entorno permite al sistema operativo poder compilar código *JavaScript* como lo haría un navegador por *ejemplo*.
- *Express* es el *framework* de *Nodejs* más popular y utilizado en el mundo de *JavaScript*. *Express* facilita las tareas de creación del back-end así como los enrutamientos, gestiones de sesiones y otras funcionalidades.
- *Angular* es otro *framework* de *Nodejs* desarrollado en *TypeScript*, para crear aplicaciones web modernas de una sola página. Estas son aplicaciones web que interactúan con el usuario para reescribir dinámicamente la página web actual con nuevos datos del servidor web. Realmente solo tienen una página pero al cambiar el contenido de manera dinámica provoca la ilusión de navegar por varias pantallas.
- *GitHub* es un repositorio de código en la nube donde subir las versiones del proyecto a medida que se trabaja en el para poder trabajar incluso desde varios equipos si fuera necesario.
- *Google Drive* es un servicio de alojamiento y sincronización de archivos desarrollado por Google. En él se guarda básicamente todo lo que no es código pero forma parte del proyecto: documentos de contabilidad de horas empleadas, diagramas terminados, guiones para el desarrollo...

2.7. Gestión de Riesgos

En un proyecto de desarrollo software, igual que en cualquier otro, existe la posibilidad de aparición de contratiempos e imprevistos. Una buena práctica para evitar que estos potenciales imprevistos creen un impacto considerable en el proyecto, es tratar de realizar un análisis para prever cuáles pueden ser los riesgos del mismo. Además del análisis, se debe crear un plan de contingencia para cada uno de ellos con el objetivo minimizar su efecto sobre el proyecto. Un plan de contingencia recoge cuáles son los pasos que toma una organización cuando se produce una situación o evento inesperado.

Un riesgo implica, por un lado, **incertidumbre** ya que el acontecimiento que caracteriza al riesgo puede o puede no ocurrir. Y por otro lado, también produce un **impacto** en el proyecto debido a la pérdida potencial, puesto que si el riesgo se convierte en una realidad, ocurrirán consecuencias no deseadas. Por tanto, podemos comprender un riesgo como la combinación entre la incertidumbre del mismo y el impacto que tiene sobre el proyecto si este se llega a dar. Para poder plasmar de manera más objetiva y cuantitativa estas dos magnitudes se han creado las tablas 2.17 y 2.18.

Probabilidad	Porcentaje
Poco probable	0-30 %
Probable	30-80 %
Muy probable	80-100 %

2.17. Tabla: Caracterización de la probabilidad de un riesgo.

Impacto	Retraso
Muy Leve	horas
Leve	1 día
Medio	2-3 días
Grande	3-7 días
Muy grande	+7 días

2.18. Tabla: Caracterización del impacto de un riesgo.

Como mencionado previamente, es fundamental elaborar un plan de prevención y contingencia con el propósito de mitigar el impacto de posibles riesgos. Estos riesgos pueden surgir en diversas etapas del proyecto. Luego de enumerar y evaluar exhaustivamente todos los posibles riesgos del proyecto, se han establecido tres categorías para su agrupación: "Riesgos de planificación", "Riesgos de desarrollo" y "Riesgos personales".

Riesgos de planificación

Conocimiento nulo o escaso en cierta tecnología

Prevención: Documentarse. Investigar a cerca de las tecnologías que se van a utilizar en el desarrollo del proyecto y realizar cursos de las mismas si fuera necesario.

Plan de contingencia: Escoger otras tecnologías similares que puedan suplir a las anteriores y tengan funcionalidades semejantes.

Probabilidad: Probable.

Impacto: Grande.

Elección incorrecta de tecnologías

Prevención: Documentarse. Investigar proyectos similares para analizar el conjunto de tecnologías usadas y verificar si se adecuan a mi proyecto también.

Plan de contingencia: Escoger otras tecnologías que tengan las funcionalidades requeridas por mi proyecto que las anteriores no tenían.

Probabilidad: Probable.

Impacto: Medio.

Requisitos del sistema no tenidos en cuenta

Prevención: En las primeras reuniones con el tutor se tratará de especificar de manera detallada todos los requerimientos esperados en el sistema.

Plan de contingencia: Convocar reuniones extraordinarias para ver cuáles han sido los requisitos no cumplidos y tratar de ajustar el proyecto para evitarlos.

Probabilidad: Probable.

Impacto: Muy Grande.

Cálculo erróneo en la planificación temporal

Prevención: Explorar proyectos similares para tratar de ver la duración que han tenido sus tareas y extrapolar dicha información a mi proyecto y tareas.

Plan de contingencia: Recalcular la planificación temporal para recuperar el tiempo perdido e intentar lo alterar mucho las fechas.

Probabilidad: Probable.

Impacto: Medio.

2.19. Tabla: Riesgos de planificación.

Riesgos durante el desarrollo

Cambio en las especificaciones del proyecto

Prevención: Tratar de concretar desde el principio cuáles son las necesidades que el cliente presenta a cerca del sistema y realizar una captura de requisitos adecuada.

Plan de contingencia: Concretar una reunión extraordinaria con el cliente para entender cuáles son los requisitos no expuestos desde el primer momento.

Probabilidad: Probable.

Impacto: Grande.

Interfaces gráficas poco intuitivas

Prevención: Diseñar prototipos de las interfaces a la hora de hacer los casos de uso extendidos y mostrar estas interfaces al cliente para conocer su opinión de usabilidad e intuitividad al respecto.

Plan de contingencia: Rehacer las interfaces gráficas con el fin de que sean más *user firendly*.

Probabilidad: Poco probable.

Impacto: Medio.

Tamaño de la bbdd muy pequeña
Prevención: Realizar una base de datos del tamaño necesario para la cantidad de usuarios esperados en el sistema **Plan de contingencia:** Hacer una copia de la base de datos para posteriormente ampliar el tamaño y volver a cargar los datos **Probabilidad:** Poco probable. **Impacto:** Grande.
Perdida del proyecto / código
Prevención: Subir periódicamente las versiones a repositorios de cogido en la nube y además guardar una copia de seguridad local por si el servidor falla. **Plan de contingencia:** En caso de pérdida del código y haber fallado tanto la copia en la nube como la copia local, estoy vendido. **Probabilidad:** Poco probable. **Impacto:** Muy grande.
Quedarse sin espacio en el disco
Prevención: Borrar de manera periódica elementos y archivos no utilizados para liberar espacio. **Plan de contingencia:** Comprar un disco duro nuevo, mover todos los archivos del disco principal al disco nuevo y formatear la partición primaria donde se encuentra el sistema operativo. **Probabilidad:** Probable. **Impacto:** Muy grande.

2.20. Tabla: Riesgos de desarrollo.

Riesgos de ámbito personal

Indisposición del desarrollador: exámenes y trabajos

Prevención: Planificar los exámenes y trabajos desde temprano para poder compaginarlos con el desarrollo del TFG.

Plan de contingencia: En caso de no poder compaginarlos tratar de estudiar y trabajar en los más prioritarios y reajustar la planificación temporal del TFG para evitar perder horas de trabajo.

Probabilidad: Muy Probable.

Impacto: Grande.

Indisposición del desarrollador: desmotivación

Prevención: Tratar de trabajar poco a poco y día a día en tareas unitarias planteadas de tal manmera que sean entretenidas de llevar a cabo.

Plan de contingencia: Tomarme unos días de descanso si realmente es necesario.

Probabilidad: Probable.

Impacto: Medio.

Indisposición del desarrollador: enfermedad

Prevención: Extremar las precauciones en el día a día para evitar caer enfermo

Plan de contingencia: Descansar y seguir las medidas necesarias para recuperarme lo antes posibles y una vez recuperado replanificar el tiempo perdido para recuperarlo

Probabilidad: Probable.

Impacto: Medio.

2.21. Tabla: Riesgos de ámbito personal.

En la Figura 2.22 se ha calculado la valoración de cada riesgo La valoración viene descrita como el producto entre la probabilidad [0-1] y el impacto (en días) del mismo.

Riesgo	Probabilidad	Impacto	Valoración
Riesgos en la planificación			
Conocimiento nulo o escaso en cierta tecnología	Probable (0.7)	Grande (5)	**3.5**
Elección incorrecta de tecnologías	Probable (0.4)	Grande (6)	**2.4**
Requisitos del sistema no tenidos en cuenta	Probable (0.6)	Muy Grande (15)	**9**
Cálculo erróneo en la planificación temporal	Muy probable (0.9)	Medio (2)	**1.8**
Riesgos durante el desarrollo			
Cambio en las especificaciones del proyecto	Probable (0.4)	Grande (7)	**2.8**
Interfaces gráficas poco intuitivas	Poco probable (0.1)	Medio (3)	**0.3**
Tamaño de la bbdd muy pequeña	Poco probable (0.2)	Grande (5)	**1**
Perdida del proyecto / código	Poco probable (0.05)	Muy Grande (100)	**5**
Quedarse sin espacio en el disco	Probable (0.4)	Muy grande (10)	**4**
Riesgos de ámbito personal			
Indisposición del desarrollador: exámenes y trabajos	Muy probable (0.9)	Grande (7)	**6.3**
Indisposición del desarrollador: desmotivación	Probable (0.6)	Medio (3)	**1.8**
Indisposición del desarrollador: enfermedad	Probable (0.5)	Grande (5)	**2.5**

2.22. Tabla: Riesgos de planificación.

Una vez obtenidas las valoraciones de cada riesgo podemos crear tres categorías diferentes en función al valor numérico del mismo.

Valoración	Rango Numérico
Leve	0-2
Normal	2-4
Importante	+4

2.23. Tabla: Agrupación de nominal de las valoraciones.

Por último, para cada riesgo se va a hacer un seguimiento periódico en función de la valoración del mismo. Recordemos que la valoración viene dada por la combinación entre la probabilidad de ocurrencia del riesgo y el impacto que supone en el proyecto en caso de suceder. Tras la clasificación de las valoraciones en tres diferentes grupos, tenemos que éstas pueden ser: leves, normales o importantes. En función de de esta clasificación se realizará un seguimiento más o menos exhaustivo que se describe en la tabla 2.24.

Valoración	Seguimiento del riesgo
Leve	Mensual
Normal	Bisemanal
Importante	Semanal

2.24. Tabla: Periodicidad del seguimiento según la valoración del riesgo.

2.8. Evaluación Económica

Una vez realizada la evaluación temporal, se conoce cual es la duración del proyecto en el tiempo. La valoración económica de cualquier desarrollo a medida está basada en el tiempo de ejecución y en su confiabilidad. También influyen, entre otros, el número de recursos humanos, equipos físicos y software utilizado para el desarrollo del mismo.

Este tipo de cálculo, aunque se sustenta en la incertidumbre, no se realiza a lo loco. La estimación se basa en métricas y medidas extraídas de proyectos semejantes y los costes asociados a los mismos. La valoración preliminar deberá incluir la estimación de todas las fases por las que pasa el proyecto: captura de requisitos, análisis y diseño de

la arquitectura, implementación, pruebas... Para empezar a estimar el coste aproximado del proyecto, se deben conocer cuáles pueden ser los diferentes gastos asociados al mismo:

- *Gastos directos*: gastos en hardware utilizado, en licencias del software empleado, gastos en recursos humanos...
- *Gastos indirectos*: gastos en luz, calefacción...

En cuanto al hardware empleado en el proyecto, se va a utilizar un equipo de sobremesa que junto a los periféricos se valora en unos 1700€. En ocasiones voy a tener que trabajar desde fuera de casa por lo que utilizaré mi equipo portátil "Asus Zenbook UM431D" valorado en 649€[5]. Los cálculos realizados se reflejan en la tabla 2.25.

Descripción	Coste	Amortización	Coste en 6 meses
Equipo sobremesa	1700€	10 años	85€
Portátil Asus Zenbook	649€	5 años	64.9€

2.25. Tabla: Costes equipos hardware.

Respecto al coste de las licencias del software empleado, todo el software empleado es *freeware* (licencia gratuita) excepto *Adobe Photoshop*. En este caso se ha obtenido la licencia anual que asciende a 290 euros y se va a utilizar por 6 meses.

Descripción	Coste	Amortización	Coste en 6 meses
Adobe Photoshop	290€/año	1 año	145€

2.26. Tabla: Costes licencias software.

Las estimaciones de tiempo para las dos primeras fases se encuentran disponibles en la sección 2.5. Por lo tanto, se prevé que la fase previa al desarrollo requerirá un total aproximado de 140 horas, de las cuales 40 horas serán exclusivamente para el DOP. Esto deja un tiempo estimado de 274 horas para el desarrollo de los sprints y la documentación de la memoria. Se estima que se destinarán 55 horas para la documentación de la memoria, lo que implica que de las 274 horas totales, 219 horas se dedicarán exclusivamente al análisis, desarrollo y pruebas de los módulos. El resumen de los cálculos se refleja en la tabla 2.27

[5]https://www.mediamarkt.es/es/product/portatil-asus-zenbook-14-um431d

Tarea	Horas
Documentación tanto del DOP como de la memoria del TFG	95
Captura de requistos, que abarca tanto las reuniones preliminares para conocer los requisitos del sistema como el diseño de las funcionalidades y diagramas asociados (casos de uso, modelo de dominio...).	100
Desarrollo software, todas las tareas relacionadas con la implementación de los diferentes módulos que componen el proyecto.	219

2.27. Tabla: Estimación temporal de cada fase del proyecto.

Se debe tener en cuenta que de la tarea *desarrollo software* está a su vez dividida en distintos módulos. Cada módulo, a su vez, como bien se explicó en la sección 2.4 consta de 3 fases: *análisis y diseño, implementación software y pruebas.* Se estima que el % dedicado a cada fase de un módulo es de:

- 15 % para *análisis y diseño.*
- 55 % para *implementación software.*
- 30 % para las *pruebas.*

Por tanto, tras el desglose de las subtareas tras el desarrollo de los módulos, la estimación temporal final de cada fase del proyecto se encuentra en la tabla 2.27.

Tarea	Horas
Documentación tanto del DOP como de la memoria del TFG	95
Análisis y diseño, que abarca tanto la captura de requisitos de la fase inicial del proyecto como el diseño de las funcionalidades y diagramas asociados (casos de uso, modelo de dominio...) Ahora también se han añadido la suma de las tareas de análisis y diseño pertenecientes a todos lo módulos	133
Implementación software, desarrollo del código de los diferentes módulos que componen el proyecto.	120
Pruebas unitarias de todas las funcionalidades y testeo de los distintos comportamientos de cada módulo ligados a cada una de las posibles acciones.	66

2.28. Tabla: Estimación temporal de cada fase del proyecto.

Una vez se conoce el computo de horas dedicadas a cada sección del proyecto, falta conocer cual es la retribución bruta media de un trabajador novato (menos de 3 años de experiencia) en cada uno de estos ámbitos en España a jornada completa. Para obtener dichos datos se han contrastando las fuentes de talent[6], glassdoor[7] y jobted[8]. Los gastos en desarrollo se pueden ver desglosados por tareas en la tabla 2.29.

- Documentador: 9.7€/hora
- Analista de software: 13.32€/hora
- Desarrollador software: 13.01€/hora
- Ingeniero de pruebas software: 15.23€/hora

Cargo	Horas	Coste/hora	Coste total
Documentador	95 horas	9.7€/hora	921.5€
Analista software	133 horas	13.32€/hora	1768.9€
Desarrollador software	120 horas	13.01€/hora	1561.2€
Ingeniero de Pruebas	66 horas	15.23€/hora	1005.18€

2.29. Tabla: Costes equipos hardware.

A los gastos directos calculados hay que sumarle los gastos indirectos. Aquellos gastos que se producen de manera inevitable durante el transcurso del proyecto: luz, agua, calefacción, etc. Se ha calculado que estos gastos suponen al rededor del 5 % del coste total del proyecto. Por tanto el coste total del proyecto asciende a 5828.84€. (Ver tabla 2.30)

Descripción del gasto	Coste
*G*astos de hardware	149.9€
*G*astos en software	145€
*G*astos de desarrollo	5256.78€
*G*astos indirectos	277.16€
Coste total del proyecto	**5828.84€**

2.30. Tabla: Estimación temporal de cada fase del proyecto.

[6]https://es.talent.com/
[7]https://www.glassdoor.es/
[8]https://www.jobted.es/

Es relevante resaltar que no se obtendrá beneficio económico del proyecto, dado que es parte de un trabajo universitario y se realiza con carácter comunitario y con la finalidad de asistir a la propia universidad. El código estará disponible y será liberado bajo licencia GPLv3, lo que implica que el costo de adquirir el producto final será de 0€.

Es importante considerar que el desarrollo de un sistema de esta envergadura implica costos significativos, y dada la limitación de recursos para un único Trabajo de Fin de Grado, no se implementarán todas las funcionalidades planificadas.

Una vez comprendidos los costos del proyecto, es esencial evaluar cómo se generarán ingresos para recuperar la inversión realizada por la universidad. En este caso, se descarta la opción de monetización mediante anuncios, ya que el sistema pertenece a la universidad. Por consiguiente, la rentabilidad del proyecto radica en la amortización. Como se mencionó en la introducción del proyecto, la universidad destina anualmente 13.500€ en papel y tinta para las encuestas físicas. Si el proyecto implica un costo total de 5828.84€, se estima que en un cuatrimestre estaría completamente amortizado, dejando de incurrir en pérdidas por recursos asociados a encuestas impresas a partir de ese momento.

3. Antecedentes

La utilización de encuestas para evaluar la satisfacción de los alumnos no es una práctica nueva, ya que ha sido utilizada durante unos cuantos años por la universidad para medir la calidad de la educación y la experiencia estudiantil. Sin embargo, el desarrollo de sistemas de encuestas online para este propósito es relativamente reciente.

En la última década, el uso de tecnología en la educación ha aumentado significativamente. Los sistemas de encuestas online permiten a los alumnos responder a las preguntas de manera más rápida y fácil. Además, los resultados se pueden analizar de manera más eficiente que en los métodos tradicionales. A día de hoy, recordemos que para poder analizar los datos subyacentes de las encuestas que realiza la UPV/EHU, primero hay que digitalizar todas ellas.

El uso de encuestas online en instituciones educativas busca mejorar la oferta educativa con el objetivo de atraer y retener alumnos, proporcionando una experiencia educativa satisfactoria y manteniendo una ventaja competitiva en un entorno educativo competente. Los alumnos cada vez tienen más opciones disponibles en cuanto a universidades y programas de estudios. Los sistemas de encuestas online pueden ser una forma efectiva de mejorar la experiencia educativa y, por lo tanto, aumentar la satisfacción de los alumnos.

En la actualidad, las encuestas se pasan en papel al alumnado en el aula. Esta situación puede acarrear ciertas dificultades, tales como la posibilidad de que el número de estudiantes presentes en clase el día de la encuesta sea reducido por pura casualidad, así como la complejidad asociada a la recolección y análisis eficiente de los datos. Para superar estos desafíos, se busca implementar un nuevo sistema que permita al docente decidir cuándo poner la encuesta a disposición del alumnado.

El objetivo consiste en implementar un sistema de encuestas en línea que permita a los docentes abrir la encuesta en el momento oportuno y habilitar la participación de los alumnos en la misma. Con el fin de lograr este objetivo, el docente podrá seleccionar el momento de apertura y el plazo para responder a la encuesta, siempre y cuando entre dentro del plazo establecido por la administración.

Además, con el sistema en línea, se pretende que las encuestas sean completamente anónimas y confidenciales. Se implementarán medidas de seguridad y protección de datos que aseguren la integridad y el resguardo de la información proporcionada por los alumnos. Se implementarán tecnologías de cifrado y protocolos seguros para proteger la transmisión y el almacenamiento de los datos de las encuestas. Esto ayudará a prevenir el acceso no autorizado y salvaguardará la información confidencial de los alumnos.

Además, el nuevo sistema brindará la capacidad de generar nuevos informes de manera más eficiente y ágil; lo cual permitirá obtener información más detallada sobre las calificaciones obtenidas por los estudiantes para los docentes. Además, se ofrecerá la opción de realizar comparativas, ya sea entre un docente y sus propias métricas o en relación a la media de la asignatura, el departamento, el centro, entre otros.

Se ha realizado un estudio de alternativas existentes tratando de descubrir las ventajas y desventajas de cada una con el objetivo de explorar las diferentes posibilidades que tiene la universidad para suplir esta carencia. Un estudio de alternativas es un análisis que se realiza para determinar las diferentes opciones que existen en el mercado y seleccionar la opción más adecuada según las necesidades específicas de la institución.

Durante el estudio, se deben considerar diferentes aspectos, como el costo, la seguridad, la facilidad de uso, etc. Entre las alternativas a evaluar, se encuentran plataformas de encuestas online como *Google Forms*, *SurveyMonkey*, servicios de encuestas personalizados de terceros y el desarrollo de un sistema de encuestas a medida. Cada una de ellas tiene sus propias características y funcionalidades, por lo que es importante conocerlas para determinar cuál se adapta mejor a las necesidades de la universidad. A continuación se muestran las ventajas, desventajas y costes medios estimados para cada alternativa:

3.1. Google Forms

Ventajas:

- Es fácil de usar y no se requieren habilidades técnicas especiales para crear encuestas.
- Permite integrarse con otras herramientas de *Google* como *Google Sheets*.

Desventajas:

- No ofrece muchas opciones de personalización avanzada.
- Algunas funciones requeridas por la universidad no son posibles, como la programación de encuestas.
- Limitado para grandes encuestas o encuestas que requieren complejidad en las preguntas.
- Imposibilidad de gestionar los plazos de apertura y cierre de las encuestas así como la validez

Precio: Gratuito.

3.2. SurveyMonkey

Ventajas:

- Ofrece variedad de plantillas de encuestas para diferentes propósitos y creación de encuestas personalizadas con diferentes tipos de preguntas.
- Ofrece la posibilidad de visualizar los resultados de las encuestas de manera gráfica y realizar análisis avanzados.

Desventajas:

- La versión gratuita tiene limitaciones en cuanto al número de preguntas y respuestas permitidas.
- El costo de las versiones de pago puede ser elevado.
- Se depende en gran medida de la disponibilidad y calidad del servicio proporcionado. Si el proveedor experimenta interrupciones en el servicio o problemas técnicos, esto podría afectar negativamente el flujo de trabajo y los resultados de las encuestas.
- El sistema actual no cuenta con la capacidad de generar los tipos de informes requeridos.

Precio: Desde 25€ por mes para la versión básica y hasta 99€ por mes para la versión empresarial.

3.3. Empresa especializada en sistema de encuestas

Ventajas:

- Ofrece gran variedad de funciones y características para definir las encuestas.
- Ofrece soporte técnico y asesoramiento personalizado.
- Permite la integración con otras herramientas y aplicaciones.

Desventajas:

- El costo puede ser elevado especialmente si se requiere un alto nivel de personalización o funcionalidad adicional.
- La personalización puede ser limitada.
- El uso de un sistema de encuestas online de terceros podría plantear problemas en cuanto a la confidencialidad de la información de las encuestas.

Precio: Desde 100€ por mes hasta 400 € por mes, dependiendo de las características y funciones seleccionadas.

3.4. Sistema de encuestas personalizado

Ventajas:

- Permite la personalización completa del sistema según las necesidades específicas de la universidad.
- Puede ser integrado con otros sistemas de la universidad.
- Permite una mayor flexibilidad en cuanto a la seguridad y privacidad de los datos.
- El control del sistema lo lleva de manera integra la administración de la universidad.
- Es la única alternativa que no requiere de intervención de terceros.

Desventajas:

- Puede requerir una inversión significativa de tiempo y recursos.
- Requiere habilidades técnicas avanzadas para desarrollar y mantener el sistema.
- Puede tener costos adicionales de alojamiento, seguridad y mantenimiento.

Precio: Depende del tiempo y recursos invertidos en el desarrollo del sistema.

En resumen, cada alternativa tiene sus ventajas y desventajas, por lo que es importante evaluar cuidadosamente cada una de ellas según las necesidades específicas de la universidad. *Google Forms* no se considera una solución adecuada debido a sus funcionalidades limitadas y su enfoque en la realización de encuestas simples, no dispone de la capacidad de programar encuestas o realizar los tipos de informes requeridos. Para implementar un sistema complejo y ajustado a la organización de la universidad, se requiere una solución más robusta y personalizable.

SurveyMonkey puede considerarse una opción más avanzada en comparación con *Google Forms*. Sin embargo, aún no logra satisfacer todas las necesidades de organización interna de la universidad, como la gestión de la estructura requerida para asignar quién debe responder cada encuesta y en qué momento. Adicionalmente, *SurveyMonkey* no posee la capacidad de almacenar las respuestas según los requisitos específicos necesitados para garantizar el máximo anonimato, ni tampoco puede generar los informes personalizados que se solicitan. Además, su costo puede ser elevado para poder contar con un servicio de encuestas tan extenso y personalizado como el que demanda la UPV/EHU.

Si se busca una solución personalizada y con soporte técnico, un sistema de encuestas de una empresa especializada puede ser adecuado, pero también puede ser costoso y además se requiere continuamente del soporte de terceros no teniendo libertad absoluta en el sistema. La creación de un sistema altamente personalizado podría implicar costos significativamente elevados, en caso de que se logre encontrar alguna empresa que pueda ofrecer una solución tan altamente adaptada a las necesidades específicas.

Si se dispone de los recursos financieros y técnicos adecuados, la opción de desarrollar un sistema de encuestas personalizado se presenta como una solución más adecuada, ya que brinda la posibilidad de diseñar y modelar la plataforma de acuerdo a las necesidades específicas de la UPV/EHU. Al contar con un sistema hecho a medida, la universidad tiene un mayor control sobre la funcionalidad y el diseño del sistema, lo que permite una integración más ajustada a los procesos internos y requerimientos de la institución.

Además, la autonomía en el proceso de administración es una ventaja significativa. Al tener el control interno del sistema, la universidad no dependerá de organizaciones externas para llevar a cabo la gestión y supervisión del proceso de encuestas. Esto se traduce en una mayor flexibilidad y capacidad de adaptación a cambios futuros en los requisitos y demandas del sistema.

No obstante, es importante mencionar que el desarrollo de un sistema personalizado requerirá una inversión en recursos humanos y financieros, así como un tiempo de desarrollo adecuado para su implementación. Es fundamental llevar a cabo un análisis minucioso de los costos y beneficios, como en este caso, para evaluar si el esfuerzo y la inversión son justificados. Previamente, en la evaluación económica del proyecto, se ha comprobado que efectivamente la inversión en la creación de este sistema se recupera mediante la amortización en un cuatrimestre, al ahorrar los costos asociados al papel y la tinta utilizados en las encuestas impresas. En caso de contar con los medios adecuados, la opción de un sistema personalizado puede ofrecer beneficios a largo plazo para la UPV/EHU.

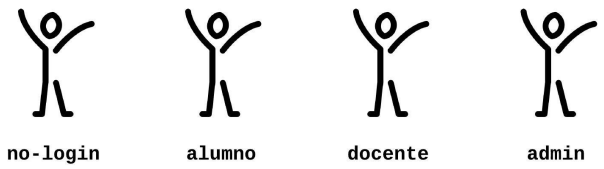
4. Captura de requisitos

En este capitulo se va a llevar a cabo el proceso de entender y definir qué características debe tener y qué funcionalidades debe cumplir el sistema para satisfacer las necesidades de la universidad.

Este proceso es crucial en el desarrollo de software, ya que los requisitos capturados son la base para el diseño, desarrollo, pruebas y entrega del software. La captura de requisitos implica trabajar con los usuarios y las partes interesadas para comprender sus necesidades y expectativas para, posteriormente, traducirlas en requisitos específicos o diagramas con el fin de ser implementados.

4.1. Casos de uso

El modelo de casos casos de uso describe cómo un usuario interactúa con el sistema para lograr un objetivo específico, y se utilizan para modelar los requisitos funcionales de un sistema. A su vez, la jerarquía de actores en los casos de uso se refiere a la clasificación de los usuarios que interactúan con el sistema en diferentes niveles de responsabilidad o autoridad y sus respectivas funciones o tareas. La jerarquía de actores de SiREnO se muestra en la Figura 4.1.

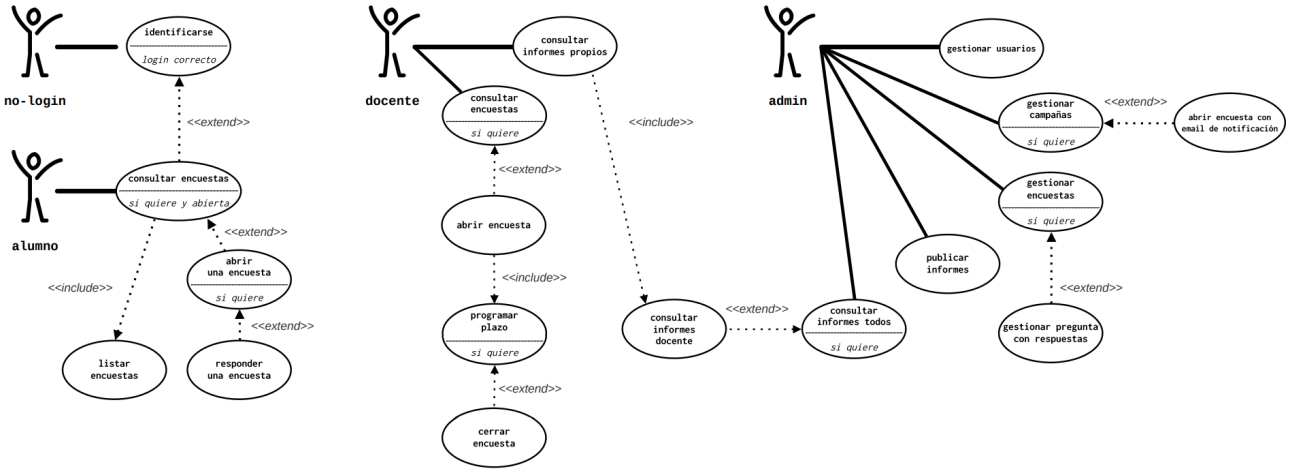


4.1. Figura: Jerarquía de actores de SiREnO.

- ***no-login***: este actor hace referencia a cualquier usuario que accede al sistema de SiREnO y aun no se ha logueado.
- ***alumno***: usuario logueado de manera exitosa. Además, en el proceso de login se le ha asignado el rol de alumno y con ello la capacidad de responder a las encuestas correspondientes.
- ***docente***: usuario logueado de manera exitosa. Además, en el proceso de login se le ha asignado el rol de docente pudiendo acceder así a las funcionalidades del mismo.

- ***admin***: usuario logueado de manera exitosa y además en el proceso de login se le ha asignado el rol de administrador, pudiendo acceder a las funciones de gestión pertinentes.

A continuación se muestra el diagrama de casos de uso de SiREnO, (Ver Figura 4.2).



4.2. Figura: Diagrama de casos de uso de SiREnO.

En esta sección además de definir el diagrama, se proporciona la explicación de cada caso de uso mostrado en la ilustración anterior, ordenado por el actor que lo invoca:

no-login

Estos son los casos de uso de cualquier persona que se conecta a la pantalla inicial del sistema SiREnO y por tanto aun no se ha identificado.

- ***identificarse****:* un usuario cualquiera accede al sistema de SiREnO e introduce su usuario y contraseña. Estas credenciales se comprueban primero contra la base de datos de SiREnO. En caso de una identificación correcta, al actor *no-login* se le asigna uno de los tres roles del sistema: alumno, docente o admin. En caso contrario, se le indica que la identificación ha sido fallida.

alumno

El usuario se ha logueado correctamente con rol de alumno:

- ***Consultar encuestas:*** una vez logueado y asignado el rol de alumno por el sistema, éste es redirigido automáticamente a una página donde se muestran todas las encuestas disponibles para el alumno (de ahí el *<<extend>>* con la condición de 'login correcto').

- ***Listar encuestas:*** este subcaso de uso es simplemente una abstracción para indicar que es automáticamente redirigido (de ahí el <<*include*>>) a una página donde se le listan todas las encuestas disponibles para él, tanto las abiertas como las cerradas.
- ***Abrir una encuesta:*** en caso de que la encuesta esté abierta y el alumno así lo desee, puede acceder al formulario para responder a la misma. Se le listarán las diferentes preguntas con sus respuestas correspondientes que deberá responder en un tiempo limitado.
- ***Responder una encuesta:*** este subcaso de uso hace referencia al proceso de enviar las respuestas de un formulario asociado a una encuesta rellenada por el alumno.

docente

Una vez logueado y asignado el rol de docente por el sistema, éste es redirigido a una página con dos botones, cada botón acciona una funcionalidad diferente: *consultar informes* o *consultar encuestas*.

- ***consultar informes propios:*** este caso de uso hace referencia a la redirección del docente a una página donde se muestran agrupadas por curso escolar las asignaturas que ha impartido y por cada asignatura un conjunto de informes extraídos de las encuestas realizadas pos los alumnos. Nótese observando la Figura 4.2 que el caso de uso llamado *Consultar informes personales* y el *Consultar informes docente* son los mismos y por ello se relacionan con un <<*include*>>. Es simplemente una manera de representar la abstracción.

- ***consultar encuestas:*** el docente ha hecho clic en el botón de "consultar encuestas" y por tanto se le listan todas las encestas correspondientes a las asignaturas que imparte. Las encuestas listadas son aquellas que aun asociadas a campañas válidas; es decir, están en el plazo que administración ha establecido para que puedan ser abiertas.

  - ***abrir encuesta:*** este subcaso se da cuando el docente hace clic en el botón "abrir encuesta" asociado a alguna de las encuestas listadas tras haber accedido a la sección "consultar encuestas". Abrir una encuesta significa activarla para que los alumnos puedan responderla.
  - ***programar plazo:*** este subcaso se ejecuta inmediatamente después del anterior y hace posible que el docente pueda programar un intervalo de tiempo para que los alumnos puedan responder la encuesta recién abierta.
  - ***cerrar encuesta:*** una ver abierta la encuesta y programado el plazo de respuesta de la misma, la encuesta se cierra automáticamente una vez terminado el plazo. Asimismo, como en el menú principal del docente se muestran todas las encuestas propias válidas estén abiertas o no, en el caso de que el docente lo considere necesario, puede cerrar manualmente una encuesta abierta del listado de encuestas mostradas. Esto puede ser útil, por ejemplo, si todos los alumnos han respondido antes del plazo establecido, ya que no tiene sentido mantener la encuesta abierta si nadie mas va a responder.

admin

Por último, tenemos el rol "admin". Una vez logueado en SiREnO y recibido el rol de "admin", éste es redirigido a una pantalla con diferentes botones donde cada uno activa un caso de uso y subcasos asociados.

- ***Gestionar usuarios:*** este caso de uso engloba la gestión integral de los distintos elementos y su organización en SiREnO. Se utiliza el término *gestionar usuarios* como una forma de generalizar y encapsular las diversas gestiones del sistema, tales como centros, departamentos, grados, asignaturas, situaciones docentes, alumnos... Al tratarse de un caso de uso de "gestión" al igual que el resto de gestiones que mencionaré a continuación, se compone de altas, bajas y modificaciones. Por tanto, este caso de uso consta, a su vez, de 3 subcasos de uso:
  - ***añadir usuario***: agregar un nuevo usuario al sistema. De nuevo, recalcar que "usuario" es aplicable para centros, departamentos, grados, grupos, etc.
  - ***modificar usuario***: editar la información de un usuario existente.
  - ***eliminar usuario***: borrar a un usuario existente por completo del sistema.
- ***gestionar campañas:*** el caso de uso tiene como objetivo poder gestionar correctamente cada campaña. Al igual que el anterior, y al tratarse de un caso de uso de "gestión", este también consta de los subcasos: altas, bajas y modificaciones; que deben hacer exactamente lo que su nombre indica.
  - ***finalizar campaña:*** es importante recalcar que en este caso particular, el subcaso de uso "finalizar campaña" (baja) también debe borrar las situaciones docentes del sistema asociadas a dicha campaña para que no quede rastro de qué alumnos eran los que tenían derecho a responder con el objetivo de preservar la confidencialidad. Concretamente, es necesario eliminar los registros de aquellos alumnos que no hayan respondido a la encuesta de la campaña, ya que una vez un alumno haya respondido, su situación docente se elimina automáticamente.
    - ***obtener estadísticas:*** Cuando se cierra la campaña, se quiere saber el porcentaje de respuesta de la misma. Este porcentaje puede ayudar en administración a obtener información de interés respecto a la participación e interés general de la asignatura o también ayudar a detectar irregularidades; por ejemplo, si un docente ha decidido abrir la encuesta a propósito un día que ha acudido poca gente a clase. El porcentaje de respuesta se obtiene, por un lado, sabiendo el número de alumnos que tenían derecho a responder a dicha encuesta (obtenido y guardado al definir la campaña) y por otro lado, sabiendo número de alumnos que han respondido a la encuesta. Este último dato se puede obtener antes de borrar las situaciones docentes una vez finalizado el plazo de la campaña.
- ***gestionar encuestas:*** el caso de uso comienza al hacer clic en el botón "gestionar encuestas" del menú principal de administración. Al hacerlo, se listan todas las encuestas disponibles en el sistema y se pueden realizar acciones sobre ellas.

- ***abrir encuesta con email de notificación**:* en caso de que el docente no haya abierto una encuesta en el plazo establecido de su campaña, administración quiere dar la posibilidad a los alumnos de responderla. Este subcaso sirve para poder abrir las encuestas que no han sido abiertas en el plazo ordinario y además notificar por email de manera automática a los alumnos definidos en la campaña para que se enteren de la apertura de la misma en un plazo extraordinario.
- ***gestionar pregunta con respuestas:*** estando en el menú principal de gestión de encuestas, existe un botón "gestionar pregunta con respuestas" que acciona este subcaso de uso y abre a su vez otra interfaz donde se listan el conjunto de preguntas definidas en el sistema ordenadas por ámbito. Existe la posibilidad de modificar una pregunta junto con su respuestas (de ahí el nombre del subcaso). También se puede definir una nueva pregunta con las posibles respuestas (en caso de no ser una respuesta numérica de escala 0-5). Por último, como en todas las gestiones, se otorga la posibilidad de borrar una pregunta existente del sistema.

- ***publicar informes:*** el caso de uso simplemente trata de hacer públicos los informes derivados de una campaña finalizada. Hasta que desde administración no los hagan públicos el docente pertinente no podrá verlos en su menú principal clicando en la sección "consultar informes personales".

- ***consultar informes todos:*** este caso de uso hace referencia a la consulta de todos los posibles informes de cualquier docente almacenado en el sistema. Desde administración deben poder ver el conjunto de todos los informes que pueden ver individualmente cada docente en su menú de informes. La relación con el caso de uso "consultar informes docente" es de <<*extend*>> ya que debe ser capaz, como he dicho, de poder ver los informes de un docente en particular.

4.2. Casos de uso extendidos (Anexo I)

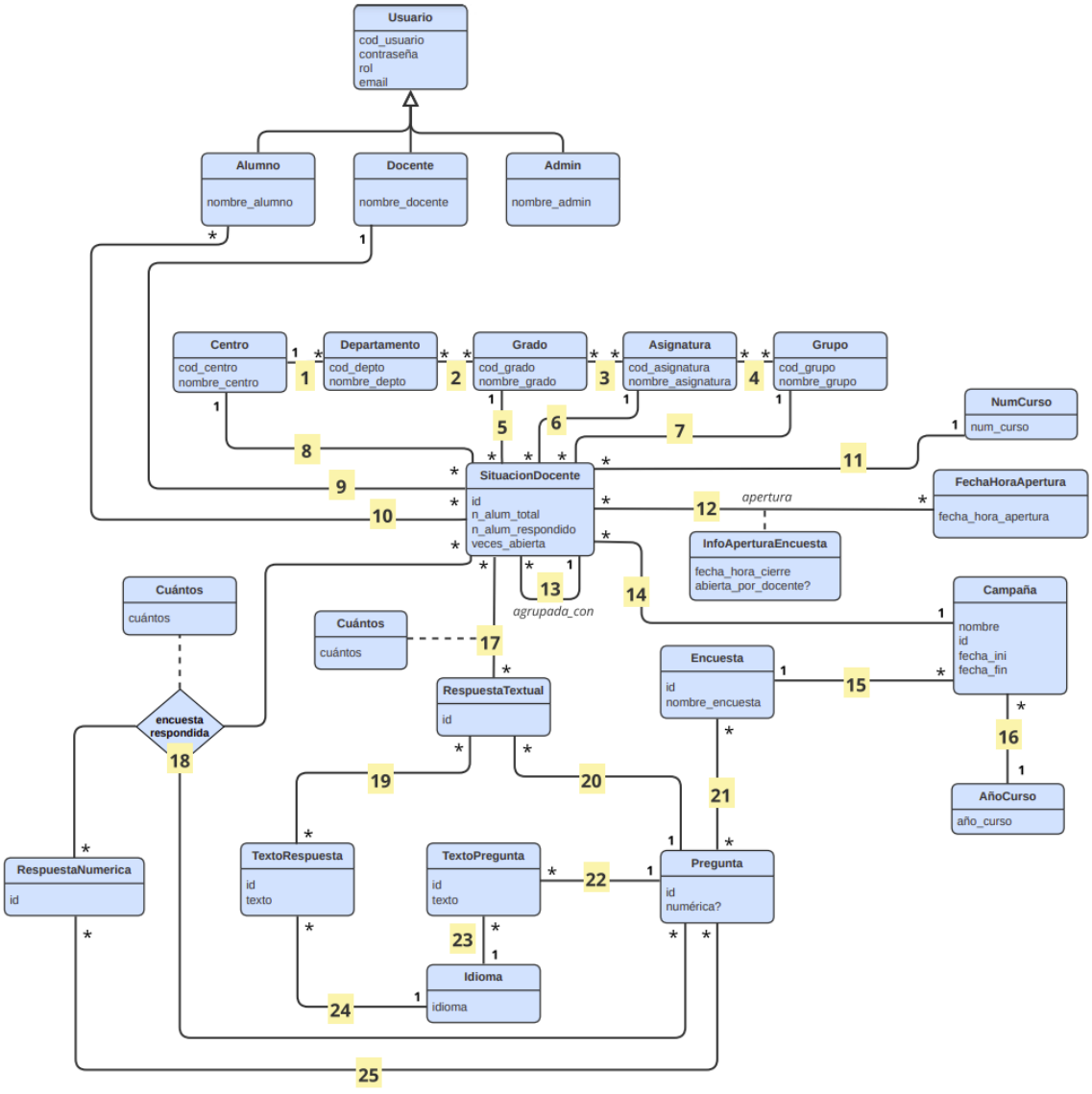
Los casos de uso extendidos son una forma de ampliar y detallar los casos recién explicados. Mientras que los casos de uso tradicionales se centran en describir las interacciones entre un actor y el sistema para lograr un objetivo específico, los casos de uso extendidos proporcionan una visión más detallada de las diferentes etapas y acciones involucradas en el proceso incluyendo información adicional, como las precondiciones, el flujo de eventos, las interfaces gráficas...

Los casos de uso extendidos se encuentran disponibles en el primer anexo de la documentación, al final del documento.

4.3. Modelo de dominio

El modelo de dominio permite realizar la representación conceptual de las entidades, atributos y relaciones que existen en el sistema de SiREnO estableciendo una representación común del dominio del problema.

Es importante entender que aunque no se vaya a desarrollar el sistema por completo, este modelo de dominio debe recopilar todas las entidades y relaciones necesarias que en un futuro pueda necesitar el sistema completo para su funcionamiento. Las entidades del modelo de dominio poseen la información mínima para poder completar el sistema. Si en un futuro interesase añadir más atributos o relaciones para completar el modelo, se podría llevar a cabo modificando la base de datos de manera muy sencilla. El modelo de dominio que refleja los datos del sistema se puede ver a continuación.



4.3. Figura: Modelo de dominio SiREnO.

El modelo de dominio está compuesto por las siguientes entidades:

- ***Usuario***: la entidad usuario representa cualquier persona que esté registrada en SiREnO y guarda la información de todos los usuarios registrados en el sistema.
  - ***Alumno***: representa a los usuarios con rol de alumno inscritos en el sistema. Hereda de la entidad usuario y por tanto además los atributos del usuario, almacena el nombre del alumno.
  - ***Docente***: representa a los usuarios con rol de docente inscritos en el sistema. También hereda de la entidad usuario.
  - ***Admin***: representa a los usuarios con rol de administrador inscritos en el sistema. También hereda de la entidad usuario.
- ***Centro***: representa un centro educativo perteneciente a la UPV/EHU.
- ***Departamento***: representa un departamento o área de estudio perteneciente a su vez a un centro educativo.
- ***Grado***: hace referencia a un nivel de estudio (máster o carrera) perteneciente a un centro educativo.
- ***Asignatura***: entidad utilizada para representar una materia o curso que se imparte en un centro educativo.
- ***Grupo***: un grupo representa un conjunto de alumnos que comparten un mismo horario para tomar clases, en función de, por ejemplo, el idioma en el que se imparten las clases, si las clases son de mañana o de tarde... (Grupo: 01, 02, 16, 31, 32, 46, 61).
- ***SituacionDocente***: la situación docente es la pieza clave para definir las campañas y por tanto las encuestas. Cada situación está ligada a una campaña que define el periodo de inicio y fin de dicha situación docente. Durante lo qué dure ese periodo, se podrá abrir la encuesta de satisfacción asociada a dicha situación docente.

  Una situación docente es única en cada curso puesto que un docente solo puede impartir la misma asignatura al mismo grupo, en el mismo centro... una vez por curso. Es posible conocer si la encuesta asociada a la situación docente ha sido abierta. Que haya sido abierta significa que en algún momento los alumnos pertenecientes a dicha situación docente han tenido la posibilidad de responder a la encuesta asociada.
- ***NumCurso***: entidad que representa el número del curso actual (1,2,3...).
- ***Campaña***: una campaña es el periodo de tiempo que dura un conjunto situaciones docentes. Por ejemplo, todas las situaciones docentes de un cuatrimestre comparten la misma campaña. Además, una campaña relaciona una encuesta con una situación docente; definiendo de este modo: "qué grupo de alumnos deben poder responder qué encuesta sobre qué docente y qué asignatura durante qué periodo de tiempo". Durante dicho periodo, la encuesta asociada a la situación docente puede ser abierta para que así los alumnos puedan responderla. También se puede saber

si una encuesta ha sido abierta en plazo más de una vez puesto que esa puede ser información de interés para la administración.

- ***FechaHoraApertura***: entidad que sirve para llevar el registro de cuándo se ha abierto cierta encuesta asociada a una situación docente. "Abrir" la encuesta de una situación docente, significa abrir un plazo para que los alumnos puedan responder al formulario asociado.
- ***InfoAperturaEncuesta***: entidad que sirve para almacenar la información de interés por cada vez que se abre una encuesta.
- ***Encuesta***: representa una encuesta (conjunto de preguntas y posibles respuestas).
- ***AñoCurso***: entidad que representa el año del curso actual (22/23, 23/24...).
- ***RespuestaTexual***: entidad que almacena las respuestas textuales (no numérica) de una pregunta de una encuesta.
- ***RespuestaNumerica***: entidad que almacena las respuestas numéricas de una pregunta de una encuesta.
- ***TextoRespuesta***: entidad que el texto asociado a una cierta respuesta de una pregunta de una encuesta.
- ***Pregunta***: entidad que almacena y representa una respuesta a una pregunta de una encuesta.
- ***TextoPregunta***: entidad que almacena los textos asociados a cada pregunta de una encuesta.
- ***Idioma***: representa los idiomas disponibles para las encuestas en el sistema.

Las relaciones **enumeradas** en la Figura 4.3 y que completan el modelo de dominio se explican a continuación:

1 Almacena el conjunto de departamentos que existen en un centro.

2 Almacena el conjunto de departamentos involucrados en un grado y el conjunto de grados en los que participa un departamento.

3 Almacena el conjunto de asignaturas de un grado y el conjunto de grados en los que se imparten cierta asignatura.

4 Almacena el conjunto de grupos pertenecientes a una asignatura y el conjunto de asignaturas en las que participa un grupo.

5 Almacena el grado al que pertenece la situación docente concreta y el conjunto de situaciones docentes que contienen cierto grado.

6 Almacena la asignatura a la que pertenece una situación docente y el conjunto de situaciones docentes que contienen cierta asignatura.

7 Almacena el grupo al que pertenece una situación docente y el conjunto de situaciones docentes que contienen cierto grupo.

8 Almacena el centro a la que pertenece una situación docente y el conjunto de situaciones docentes que contienen dicho grupo.

9 Almacena el docente asociado a una situación docente y el conjunto de situaciones docentes a las que pertenece cierto docente.

10 Almacena el conjunto de alumnos pertenecientes una situación docente y el conjunto de situaciones docentes a las que pertenece cierto alumno. Como se puede ver en la Figura 4.3, la entidad "SituacionDocente" tiene un atributo *nAlumTotal*; el valor de este atributo es la suma del número de relaciones entre una entidad "SituacionDocente" y "alumno". Además, el atributo "nAlumRespondido" se inicializa con valor 0 y por cada relación que va desapareciendo se suma 1, ya que, que desaparezca una de estas relaciones indica que un alumno ya a respondido a una encuesta y se ha borrado la situación docente que lo relacionaba.

11 Esta relación enlaza la situación docente con el número de curso al que pertenece.

12 Esta relación (llamada *apertura*) se usa para almacenar todas las veces que una encuesta haya sido abierta bien por el docente o por la administración. Además de almacenar quién la abre, se almacena entre la fecha de apertura y la fecha de cierre.

13 En un aula puede haber alumnos de diferentes situaciones docentes, ya que, en una misma asignatura se pueden juntar alumnos de diferentes grados. A la hora de hacer los informes y mostrar los resultados, al docente se le quieren mostrar de manera agrupada los resultados de los alumnos que están físicamente juntos en el aula, aunque sean de situaciones docentes distintas. Gracias a esta relación podemos almacenar qué situaciones docentes están agrupadas con otras en un mismo espacio físico.

14 Almacena la campaña asociada a cierta situación docente y el conjunto de situaciones docentes a las ha sido asociada dicha campaña.

15 Almacena la encuesta asociada a cierta campaña y el conjunto de campañas a las ha sido asociada dicha encuesta.

16 Almacena el curso al que pertenece cierta campaña y el conjunto de campañas existentes en cierto curso.

17 Esta relación sirve para almacenar las respuestas de los alumnos a las preguntas con respuestas **textuales** de cierta encuesta asociada a una situación docente. De tal manera que cuando una cierta respuesta a una cierta pregunta se da por primera vez en una salutación docente, se crea la relación con el atributo *cuántos* inicializado a 1. Cuando una cierta respuesta a cierta pregunta se repite, no se crea una nueva relación sino que se le suma 1 al valor del atributo *cuántos.*

18 Esta relación es análoga a la anterior pero para las preguntas **numéricas** de cierta encuesta. En esta relación también hace falta conocer a qué pregunta se está respondiendo, ya que, a diferencia de las respuestas textuales que son únicas; las numéricas no lo son. No es lo mismo responder un '1' a la 'satisfacción general en la signatura' que a 'valoración del trabajo personal realizado en la asignatura'.

19 Almacena el texto asociado a un conjunto de respuestas textuales y el conjunto de respuestas textuales a las que se asocia cierto texto.

20 Almacena la pregunta a la que pertenece cierta respuesta textual y el conjunto de respuestas textuales que puede tener una pregunta. Esta relación, combinada con la anterior, hace posible que dos preguntas aun teniendo el mismo texto, sean consideradas diferentes preguntas en función de la pregunta a la que pertenecen

21 Almacena el conjunto de preguntas que puede tener una encuesta y el conjunto de encuestas en las que puede aparecer cierta pregunta.

22 Almacena el texto asociado a una cierta pregunta y el conjunto de preguntas que puede tener asociado cierto texto.

23 Almacena el idioma asociado al texto de una cierta pregunta y el conjunto de textos que puede tener asociado cierto idioma.

24 Análoga a la relación anterior pero para respuestas en lugar de preguntas.

25 Almacena el conjunto de posibles respuestas numéricas asociado a una cierta pregunta y el conjunto de preguntas que puede tener asociado cierta respuesta numérica.

5. Análisis y Diseño

En la fase de análisis y diseño de un proyecto de desarrollo de software, se busca definir claramente el sistema que se va a construir, incluyendo su arquitectura y pautas de funcionamiento. Para ello, se deben analizar y diseñar todos los aspectos importantes del sistema, desde la base de datos hasta las estructuras del back-end y front-end. Es fundamental que el diseño sea lo suficientemente detallado y preciso para permitir la construcción del sistema.

En esta fase se realizan también los diagramas de secuencia más importantes para comprender las funcionalidades del sistema. Estos diagramas son una herramienta valiosa para los desarrolladores, ya que les permiten visualizar de forma clara y detallada cómo se relacionan las diferentes partes del sistema y cómo se llevan a cabo las operaciones. Éstos se pueden encontrar en el anexo correspondiente.

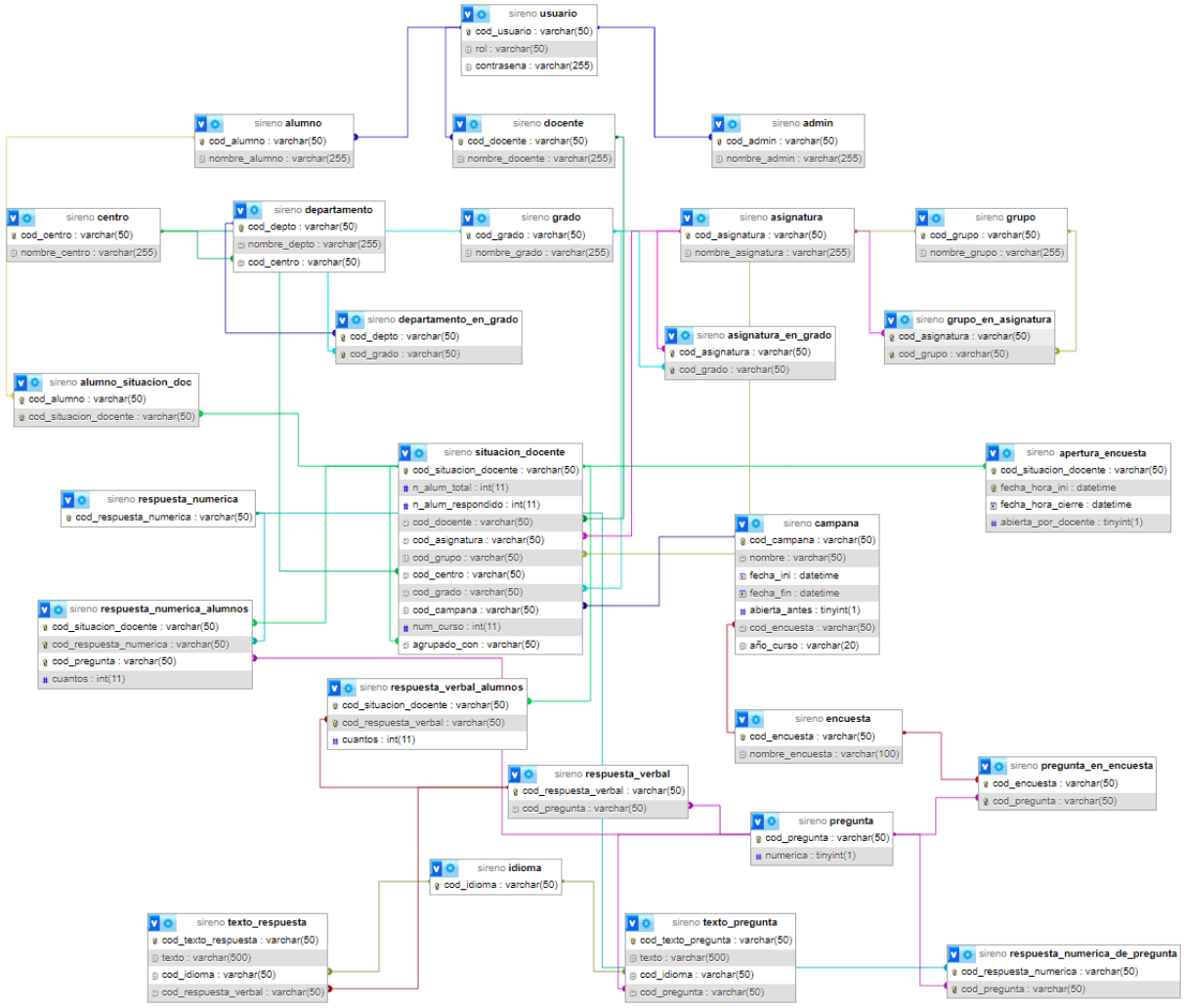
En resumen, la fase de análisis y diseño es crucial para el éxito de un proyecto de desarrollo de software, ya que es la base sobre la cual se construirá el sistema. Un diseño detallado y preciso permitirá trabajar con mayor eficacia en la fase de desarrollo y reducirá los riesgos de errores y fallos en el sistema final.

5.1. Diagrama relacional de la Base de Datos

Gran parte de la responsabilidad de que SiREnO funcione correctamente recae en el correcto análisis e implementación de su base de datos. Para definir e implementar la base de datos, se requiere llevar a cabo una conversión utilizando el modelo de dominio del sistema como base. El proceso de conversión implica convertir las entidades y relaciones del modelo de dominio en tablas en la base de datos correspondiente.

Transformar el modelo de dominio en una base de datos es el proceso de creación de una estructura de datos que refleje el modelo de dominio de la aplicación y que permita el almacenamiento y recuperación de los datos de forma eficiente. Cada tabla en la base de datos representa una entidad en el modelo de dominio de la aplicación, y cada columna representa una propiedad de esa entidad. Las relaciones entre las entidades se representan mediante claves externas o tablas de enlace, dependiendo de la naturaleza de la relación.

En la Figura de a continuación (Ver Figura 5.1) se puede ver la base de datos derivada el modelo de dominio definido en el apartado anterior.



5.1. Figura: Diagrama relacional de la base de datos de SiREnO.

A continuación se presenta una breve explicación sobre el origen de cada tabla que se muestra en la Figura anterior:

Las entidades que forman parte del modelo se transforman en tablas directamente con sus atributos como campos de la misma:

Tablas a partir de las entidades: usuarios, alumnos, docentes, admins, centros, grados, asignaturas, grupos, modalidades, situaciones_docentes, campañas, apertura_encuesta, encuestas, cursos, respuesta_textual, respuesta_numerica, texto_respuestas, preguntas, texto_preguntas, idiomas.

Algunas relaciones se incorporan como atributos en una de las tablas que componen el conjunto, mientras que otras relaciones se transforman en nuevas tablas. A continuación, se explican dichas tablas. Varias de estas tablas se han diseñado para la posterior creación de informes acerca de las encuestas respondidas:

- ***asignatura_en_grado***: relaciona y almacena las asignaturas que se imparten en cada grado (titulación) y los grados en los que se incluye cada asignatura.

- ***grupo_en_asignatura***: relaciona y almacena los grupos de alumnos asignados a cada asignatura.
- ***alumno_situacion_doc***: relaciona el conjunto de alumnos junto con las situaciones docentes a las que pertenecen.
- ***apertura_encuesta***: Almacena la información sobre la apertura de las encuestas. La fecha de inicio y fin de la apertura, si la ha abierto el docente o ha sido administración...
- ***respuesta_textual_alumnos***: almacena las respuestas textuales proporcionadas por los alumnos en un formulario de una encuesta de una situación docente. Relaciona la situación docente del alumno con la respuesta dada a una pregunta específica.
- ***respuesta_numerica_alumnos***: almacena las respuestas numéricas proporcionadas por los alumnos. Es el mismo concepto que el anterior pero para las respuesta numéricas ya que se tratan de manera diferente a las textuales.
- ***pregunta_en_encuesta***: almacena las preguntas incluidas en una encuesta. Para ello relaciona el id de una encuesta con el conjunto de ids de cada respuesta que componen dicha encuesta
- ***respuesta_numerica_de_pregunta***: almacena el conjunto de respuestas numéricas que pueden tener unas preguntas en particular.

5.2. Estructura back-end y API REST

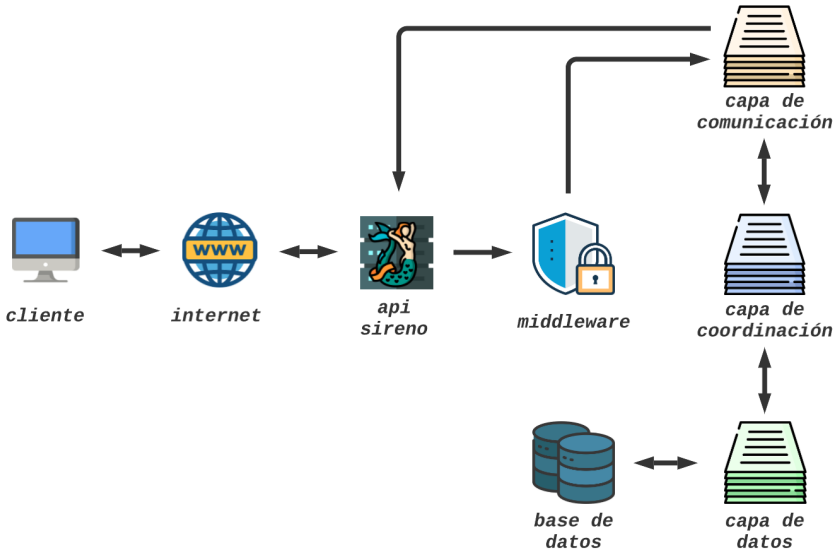
El sistema de recogida de encuestas online, SiREnO, como ya se ha adelantado en la introducción, sigue una arquitectura REST. Se va a desarrollar una API REST para facilitar la comunicación entre el servidor y el cliente. Esta API se divide en tres componentes principales: la parte servidora, la parte cliente y el canal de comunicación para el intercambio de datos.

Por un lado, la parte servidora de SiREnO será responsable de proporcionar los recursos y funcionalidades de la API. Por otro lado, la parte cliente será la encargada de consumir estos recursos y comunicarse con el servidor. Entre ambos componentes, se establecerá un canal de comunicación para enviar los datos necesarios.

Dado que los datos se enviarán a través de redes accesibles para cualquiera, la seguridad es un aspecto fundamental a considerar. Por lo tanto, en el back-end de SiREnO, se implementará un esquema que permita incorporar medidas de seguridad, como la autenticación mediante tokens y la gestión de roles, por ejemplo.

Para construir este sistema en el back-end, se utilizará una estructura de capas que no solo mejora la seguridad y la comunicación entre los recursos, sino que también permite desarrollar un sistema altamente modular. Esto facilitará el mantenimiento y la

escalabilidad de SiREnO, así como la incorporación ordenada y eficiente de nuevas funcionalidades. En la Figura 5.2 se presenta la estructura de capas que seguirá SiREnO.



5.2. Figura: Arquitectura y comunicación entre capas de SiREnO.

Una vez que se han analizado las capas en las que se divide el back-end, es importante profundizar en el proceso de comunicación entre estas capas y los archivos que las componen. Para entender el proceso de comunicación, a continuación se va a explicar siguiendo el esquema de la Figura 5.2:

Un usuario, mediante un cliente, establece una conexión a través de internet utilizando una URL válida proporcionada por SiREnO. Esta URL genera una solicitud de consumo de recursos a la API con el propósito de acceder al back-end del sistema. Una vez que la solicitud alcanza el servidor, el primer paso es que el *middleware* valide dicha solicitud para verificar si proviene de un usuario que ha iniciado sesión, es decir, si posee un token de acceso válido. Si se cumple esta condición, el control se transfiere a la *capa de comunicación*.

La capa de comunicación se encarga de analizar la URL entrante y los parámetros asociados. En caso de que la URL sea válida, se realiza la transferencia de control al módulo correspondiente la capa de coordinación. La capa de coordinación, está compuesta por módulos específicos responsables, cada uno, de gestionar un conjunto URLs entrantes agrupados por temática.

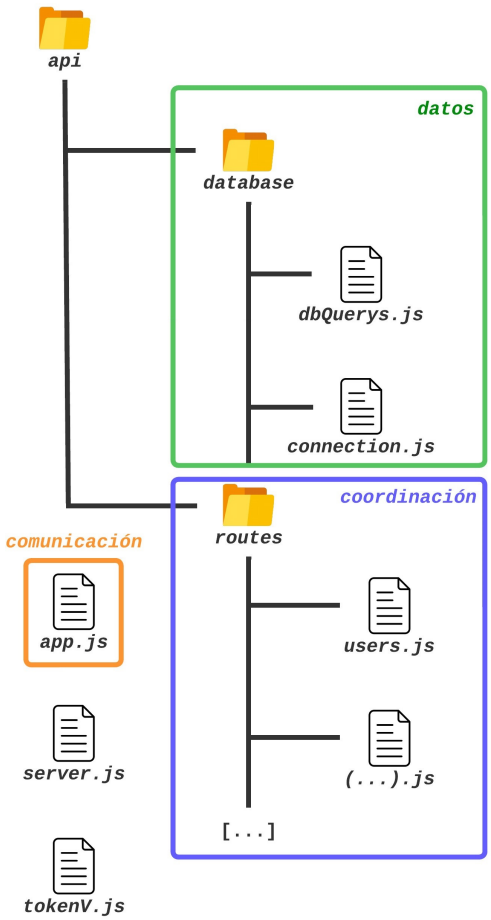
La capa de coordinación desempeña un papel fundamental al tratar individualmente cada una de las rutas entrantes. Como se mencionó anteriormente, esta capa se divide en varios componentes, donde cada uno de ellos se encarga de manejar las URL relacionadas con su temática específica. La capa de coordinación almacena la lógica de negocio de la página web, es decir, define qué acciones realizar y cómo llevarlas a cabo. En muchas ocasiones, para poder proporcionar una respuesta al cliente, es necesario consultar información en la base de datos, y es en este punto donde entra en juego la capa de datos.

La capa de datos asume la responsabilidad de definir los métodos para recuperar y

obtener datos. Esta capa encapsula la obtención de información en funciones que ejecutan sentencias contra la base de datos. El propósito de esta capa es principalmente abstraer el lenguaje de la base de datos de la lógica del sistema. Esto nos permite cambiar el sistema de gestión de bases de datos (SGBD) por otro sin afectar la lógica existente, ya que las llamadas a la base de datos están encapsuladas. Esta solución hace que el sistema sea altamente modular, evitando la necesidad de modificar las funciones que contienen la lógica de negocio en caso de cambios en el SGBD.

Los datos de vuelta se envían en la misma dirección pero sentido contrario. Nótese que el proceso de respuesta no pasa por el middleware, ya que en la petición se ha comprobado la validez de ésta.

En esta Figura se muestra un diagrama que describe la estructura de ficheros principal que conforma el componente del back-end.



5.3. Figura: Arquitectura del back-end de SiREnO.

El fichero “app.js” desempeña principalmente el papel de la capa de comunicación. Todas las solicitudes dirigidas al back-end llegan a este archivo, que redirige cada solicitud al componente correspondiente en la capa de coordinación.

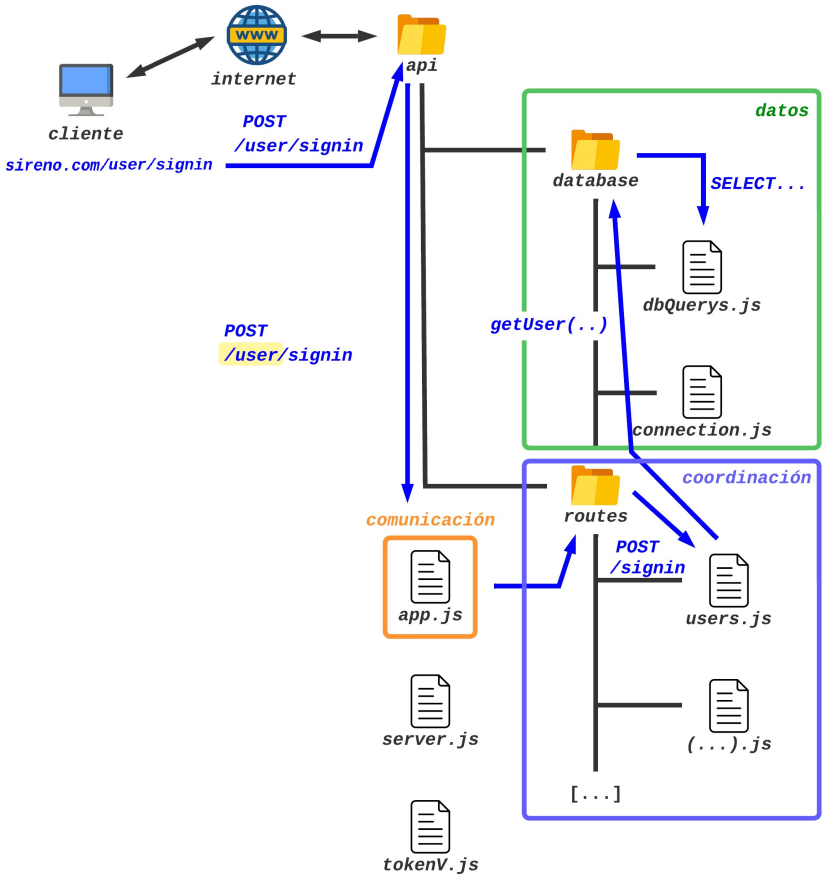
La carpeta denominada "api" contiene dos subcarpetas: "routes" y "database".

- La carpeta "routes" representa la capa de coordinación. En esta carpeta se encuentran varios archivos considerados componentes. Cada componente almacena un conjunto de rutas entrantes y la lógica de negocio asociada a cada una de ellas, encapsulada en métodos individuales.
- La carpeta "database" representa la capa de datos del sistema. Aquí se gestionan todas las operaciones relacionadas con la conexión y consultas a la base de datos. El archivo "dbQuerys.js" contiene todos los métodos necesarios para la comunicación, recuperación, inserción, etc., de información en la base de datos. Por otro lado, el archivo "connection.js" se encarga de establecer la conexión con la base de datos remota y crear una pool de conexiones, explicado más adelante.

Al mismo nivel que la carpeta "api" se encuentran otros 2 módulos:

- El módulo "server.js" contiene la configuración del servidor.
- El módulo "tokenVerifier.js" es un *middleware* encargado de verificar la validez y el rol asociado al token antes de pasar la petición de la capa de comunicación a la capa de coordinación para su tratamiento.

La Figura 5.4, ilustra sobre la Figura anterior el proceso de comunicación seguido por la parte del servidor.



5.4. Figura: Proceso de comunicación en el back-end.

En este ejemplo específico, un usuario intenta iniciar sesión a través de un cliente en la página de inicio de sesión de SiREnO, la cual se encuentra en la URL "*sireno.com/user/signing*". Esta petición es enviada al back-end mediante la API previamente descrita utilizando el método POST a la URL "*/user/signing*". La capa de comunicación es la responsable de recibir y gestionar todas las solicitudes entrantes, por lo que recibe esta petición entrante y verifica su validez, tras lo cual, transfiere el control a la capa de coordinación.

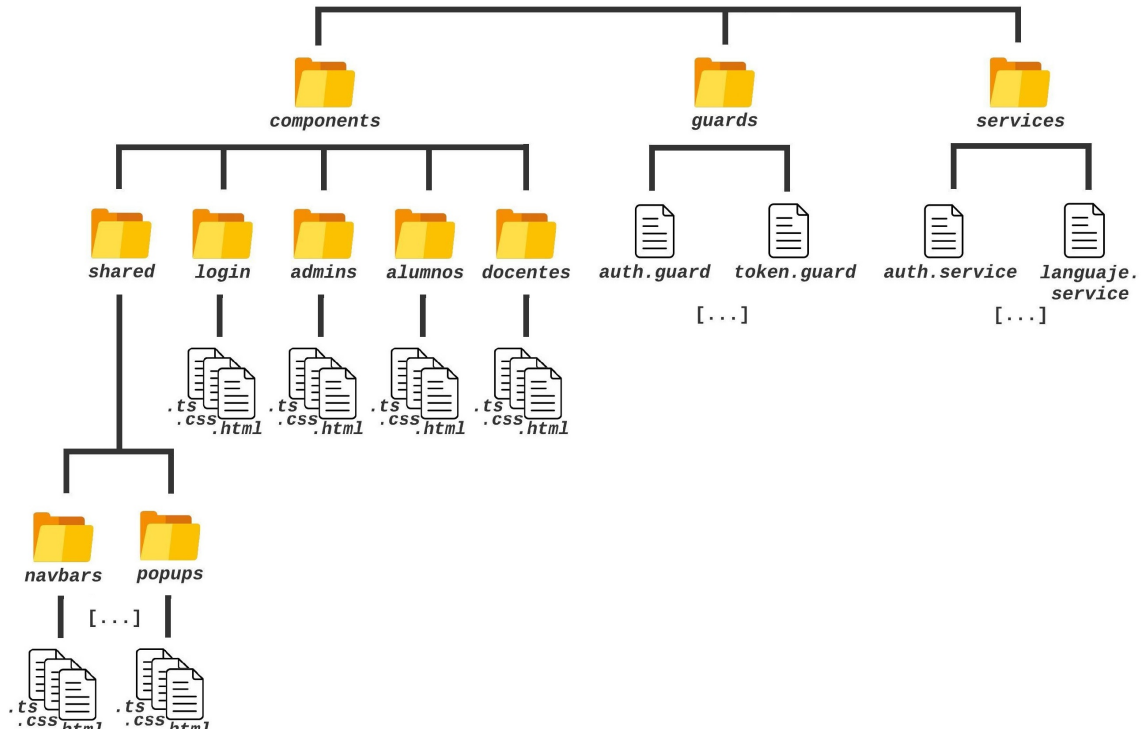
Al encontrarse la petición en el directorio "user" ("*sireno.com/**user**/signing*"), el componente específico en la capa de coordinación encargado de manejar esta solicitud es "user.js". Dicho componente ejecuta la lógica de negocio asociada a la ruta "*/signing*" de este módulo, que en este caso implica recuperar el usuario y la contraseña de la base de datos para verificar si el inicio de sesión es válido. Para lograr esto, la capa de coordinación transfiere el control a la capa de datos utilizando el método "getUser(user)".

A su vez, este método realiza una consulta a la base de datos mediante una instrucción SELECT. Si las credenciales son correctas, el usuario se autentica y se le otorga un token de acceso con una fecha de expiración. En caso contrario, se muestra un mensaje en pantalla indicando que el inicio de sesión no fue exitoso.

5.3. Estructura front-end

Tras la explicación detallada del back-end del proyecto, toca adentrarse en la parte visible del sistema, es decir, el front-end. Para el desarrollo del front-end, se ha utilizado el framework Angular. Angular facilita el inicio de un proyecto proporcionando una estructura básica de código, lo que simplifica la construcción de una arquitectura que los navegadores puedan interpretar correctamente y de manera eficiente.

Angular es un framework basado en componentes. Los componentes son fragmentos de código reutilizables en otros proyectos de Angular, lo que permite un desarrollo de aplicaciones más ágil. Un componente controla tanto la estructura como el estilo y funcionalidad de una parte de la interfaz gráfica. Por ejemplo, la barra de navegación del sitio web con las opciones de cambio de idioma o cerrar la sesión y que se repite en todas las pantallas, debe ser un componente.



5.5. Figura: Arquitectura del front-end de SiREnO.

La Figura 5.5 muestra la solución para el desarrollo del front-end. Cada componente representa una pantalla de la interfaz, aunque algunos componentes pueden ser elementos más pequeños, por ejemplo, una gráfica, que se incrusta dentro de un componente principal. Todos estos componentes se organizan y almacenan en la carpeta denominada "components".

Siguiendo un orden de izquierda a derecha, en primer lugar tenemos la carpeta "shared" donde se almacenan todos los componentes más pequeños recién mencionados, que van a ser utilizados en múltiples pantallas. Estos están agrupados por categorías, y a medida que se vayan creando más, se crearán nuevas categorías. Por ejemplo, en la Figura 5.5 se observan las subcarpetas "navbars" y "pop-ups" que contendrán todas las barras de navegación y todos los diálogos emergentes del front-end.

La carpeta denominada "login" almacena el componente correspondiente al inicio de sesión, el cual está compuesto por un formulario para facilitar dicho proceso. Las otras tres carpetas hacen referencia al conjunto restante de pantallas de la interfaz. Estas pantallas están agrupadas de acuerdo al rol al cual están destinadas: administradores, alumnos y docentes. Por ejemplo, el componente "indexAlumnos" que se encuentra ubicado dentro de la carpeta "alumnos", representa la pantalla principal para los alumnos, a la cual son redirigidos después de un inicio de sesión exitoso. Cada componente en Angular está compuesto por tres archivos distintos: un archivo *.html*, un archivo .css, y un archivo .ts.

- El archivo *.html* se encarga de definir la estructura de la página y los contenidos que se mostrarán en ella. Los elementos HTML se organizan en una jerarquía adecuada para representar la interfaz de usuario del componente.

- El archivo *.css* se utiliza para proporcionar estilos adicionales a la página como colores, fuentes, tamaños, márgenes y otros atributos que afectan la apariencia y presentación del componente.
- Finalmente, el archivo *.ts* contiene la lógica detrás de cada elemento del componente. En este archivo se define la funcionalidad y el comportamiento del componente, como la gestión de eventos, la manipulación de datos y la interacción con otros componentes y servicios.

SiREnO hace uso de servicios de Angular para proporcionar funcionalidades adicionales. Los servicios son clases de Angular que se utilizan para organizar y proporcionar funcionalidades compartidas en toda la aplicación. Estos servicios encapsulan la lógica de negocio y se utilizan, por ejemplo, para conectarse y obtener datos de la API descrita en el apartado anterior, gestionar la traducción completa de los textos de la página web o insertar de manera automática cabeceras en cada llamada a la API para no tener que hacerlo de manera manual. Los servicios se crean una vez y se pueden inyectar en múltiples componentes o incluso en otros servicios. Esto promueve la reutilización de código, la separación de responsabilidades y la creación de aplicaciones más modulares y mantenibles.

Por último, el sistema también utiliza *guards* de Angular para gestionar la seguridad y el control de acceso. Estas clases se implementan para proteger rutas y componentes, y permiten tomar decisiones sobre si un usuario tiene permisos suficientes para acceder a determinadas partes de la aplicación. Los *guards* se ejecutan antes de que se active una ruta y controlan si se permite o se bloquea el acceso al recurso solicitado. Por ejemplo, se puede utilizar un *guard* para verificar si un usuario está autenticado antes de permitirle acceder a una página protegida.

5.4. Diagramas de secuencia (Anexo II)

Los diagramas de secuencia son una herramienta de modelado muy utilizada para representar y analizar la comunicación entre los distintos elementos de un sistema. Estos diagramas permiten visualizar la secuencia de mensajes que se intercambian entre los objetos y cómo estos colaboran para llevar a cabo una funcionalidad específica. Su aplicación facilita una mejor comprensión y análisis de las relaciones y procesos que ocurren en el sistema en cuestión.

Los diagramas de secuencia de SiREnO se encuentran disponibles en el segundo anexo de la documentación, al final del documento.

6. Desarrollo

En este capítulo titulado "Desarrollo", se expondrá en detalle el proceso de creación del sistema SiREnO, así como los pasos realizados durante su implementación. El desarrollo se organiza en **módulos**, y se seguirá el enfoque del método Scrum para llevar a cabo la implementación de cada uno en forma de **sprints**. El sistema web consta de dos componentes diferentes, back-end y front-end. En cada sprint se va a implementar tanto la parte del servidor como la del cliente, por tanto, en la explicación de cada módulo se expondrá el desarrollo de ambas.

6.1. Inicializacion del back-end y front-end

En esta sección se aborda la inicialización del back-end y del front-end. Este proceso trata de preparar y configurar los componentes correspondientes de una aplicación.

La inicialización del back-end implica configurar el servidor y los servicios relacionados. Esto incluye la instalación y configuración del software del servidor, la configuración de la base de datos, la puesta a punto del sistema de ficheros, etc. También implica planificar las rutas y *endpoints* necesarios para que el servidor pueda recibir y responder a las solicitudes de los clientes.

Por otro lado, la inicialización del front-end se refiere a la configuración de la parte del cliente de la aplicación. Esto implica la configuración del entorno de desarrollo, la instalación y configuración de las dependencias y bibliotecas necesarias. También incluye la planificación y agrupación de las rutas de navegación junto con la integración del back-end para realizar solicitudes y recibir respuestas.

6.1.1. Inicializacion del back-end

El primer paso es crear un nuevo proyecto vacío en *Visual Studio Code*. Una vez abierto, debemos debemos crear el archivo "package.json". Éste es un archivo de configuración utilizado para administrar las dependencias y configuraciones del proyecto. Para crear este archivo debemos ejecutar el siguiente comando en la consola:

```
npm init -y
```

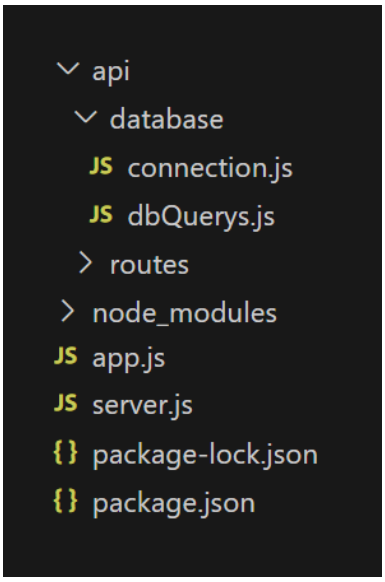
Una vez creado el paquete "package.json", hay que instalar las dependencias. Primero, instalaremos *Express*. También instalaremos *MySQL* y *JSON Web Token*; este último define un formato compacto y seguro para transmitir información entre partes como un

objeto JSON. Instalaremos *CORS* para poder hacer consultas desde nuestro front-end. Además, se incluirá la librería *nodemon* para permitir la ejecución continua del servidor.

```
npm install express mysql jsonwebtocken cors nodemon
```

Mientras se instalan las dependencias, vamos a crear la jerarquía de carpetas para el back-end. Esta jerarquía es una solución, pero hay muchas otras formas de crear servidores *Express*. La estructura que se va a crear es la que se ha presentado en la Figura 5.3.

En esta estructura, en el primer nivel está el archivo "app.js" y otro llamado "server.js", cuyas funcionalidades se explicaron previamente. También se ha creado la carpeta llamada "api". Dentro de la carpeta "api" habrá dos carpetas: "connection" y "routes". Dentro de "connection", habrá un archivo de conexión a la base de datos y otro archivo encargado de ejecutar las sentencias. Dentro de "routes", recordemos, guardaremos un archivo por cada agrupación de rutas que queramos crear. Esta es la estructura resultante:



6.1. Figura: Estructura de ficheros del back-end.

En la carpeta "database", en el archivo"connectión" se va a configurar la conexión con la base de datos, para ello se debe especificar el usuario y contraseña del administrador así como la ruta y puerto de la misma. Una conexión a la base de datos tiene un tiempo de vida limitado, tras el cual caduca y se vuelve inválida. Esto implica que no podemos utilizar una misma conexión *SQL* de forma permanente en el back-end, ya que eventualmente caducará.

Una alternativa sería crear manualmente una nueva conexión por cada sentencia ejecutada contra la base de datos, pero si hay numerosas llamadas a la base de datos, esto generaría una cantidad desproporcionada de conexiones y no se aprovecharía adecuadamente el tiempo de vida de cada una. En este escenario es donde entra en juego el concepto de los *pool* de *SQL*. Un *pool* de *SQL* es una técnica que administra y reutiliza conexiones a una base de datos en entornos de aplicaciones web. En lugar de abrir y

cerrar una conexión a la base de datos cada vez que se requiere realizar una consulta o transacción, el *pool* de *SQL* mantiene un conjunto de conexiones preestablecidas y listas para su uso.

```
const mysqlConnection = mysql.createPool({
  host:'localhost',
  user: 'root',
  password: '',
  database: 'sireno',
  port: '3306',
});

mysqlConnection.getConnection((err, connection) => {
  if (err) {
    if (err.code === 'PROTOCOL_CONNECTION_LOST') {
      console.error('Database connection was closed.')
    }
    if (err.code === 'ER_CON_COUNT_ERROR') {
      console.error('Database has too many connections.')
    }
    if (err.code === 'ECONNREFUSED') {
      console.error('Database connection was refused.')
    }
  }
  if (connection) {
    connection.release()
  }
  return
})
```

En este código se define un conjunto inicial de conexiones de base de datos en la *pool* con sus respectivos parámetros de conexión, como la dirección IP, el nombre de usuario, la contraseña y el nombre de la base de datos. Cuando la aplicación necesita realizar una operación en la base de datos, solicita una conexión de la *pool*. La *pool* busca una conexión disponible y la asigna a la aplicación. Una vez que la aplicación ha terminado de usar la conexión, la devuelve a la pool en lugar de cerrarla. La conexión se restablece y se marca como disponible para su reutilización.

Dentro de la función de devolución de llamada, se verifica si se produjo un error al establecer la conexión. Si hay un error, se realizan algunas comprobaciones adicionales para determinar la causa del error. Si el código de error coincide con alguno de los valores específicos, se imprime un mensaje de error correspondiente. Si no hay errores y se ha establecido la conexión correctamente, se libera la conexión llamando al método *"release()"* en el objeto de conexión.

Siguiendo con la estructura del proyecto, en la misma carpeta "database" se ha definido un segundo archivo llamado "dbQuerys.js" que almacena y encapsula todas las llamadas a la base de datos para así abstraer el SGBD usado en esta solución. Si el back-end necesitase, por ejemplo, el listado de usuarios del sistema, en lugar de ejecutar una llamada a la base de datos desde donde se requiera dicho listado haciendo uso de un *SELECT*; se llama a la función *"getUsers()"* de "dbQuerys.js", esta función es la encargada de ejecutar el *SELECT*.

Esta implementación proporciona una mayor flexibilidad para realizar cambios rápidos en el sistema de gestión de bases de datos. En lugar de tener que realizar modificaciones en múltiples secciones del código, solo se requiere ajustar las llamadas a la base de datos en el archivo "dbQuerys.js". Esto simplifica y agiliza el proceso de cambio de sistema de gestión de bases de datos.

El módulo "app.js" es el que va a recibir todas las rutas entrantes. Simplemente vamos a dejar importado el módulo de *Express* y *CORS* y habilitamos el *middleware* para el manejo del body en formato URL-encoded y JSON. Por último, exportamos la clase.

En el módulo "server.js", requeriremos el módulo HTTP e importaremos la aplicación que acabamos de exportar ("app.js"). El módulo "server.js" se encarga de definir el puerto de escucha del servidor y finalmente arrancarlo.

Con estos pasos, se ha completado la configuración de la estructura del back-end y se encuentra listo para iniciar la implementación de las funcionalidades requeridas.

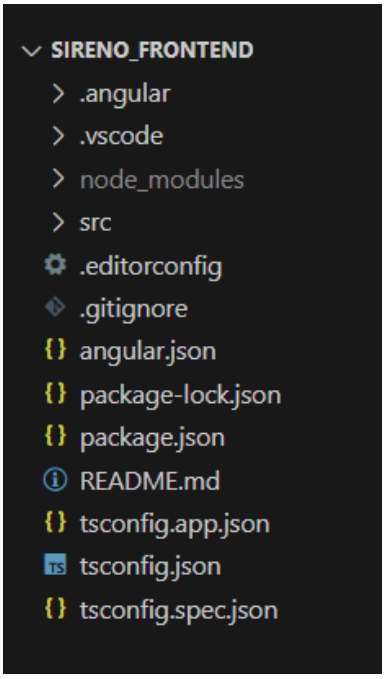
6.1.2. Inicializacion del front-end

Ahora que ya está el back-end creado, debemos crear otro proyecto en *Visual Studio Code* que será la raíz del del front-end. Como he adelantado, para el desarrollo del front-end de la aplicación web, se emplea Angular, un *framework* basado en *TypeScript*. Para iniciar un proyecto en Angular, se ejecutan los siguientes comandos en la consola. El primer comando instala el *framework* Angular en el entorno de trabajo. El segundo, crea el proyecto del front-end, generando la estructura deseada para el mismo.

```
npm install -g @angular/cli

ng new sireno_front-end
```

Después de ejecutar los dos comandos, se ha generado la estructura inicial que sirve como base para desarrollar la parte del front-end de SiREnO. Esta estructura base puede se puede ver en la Figura 6.2.



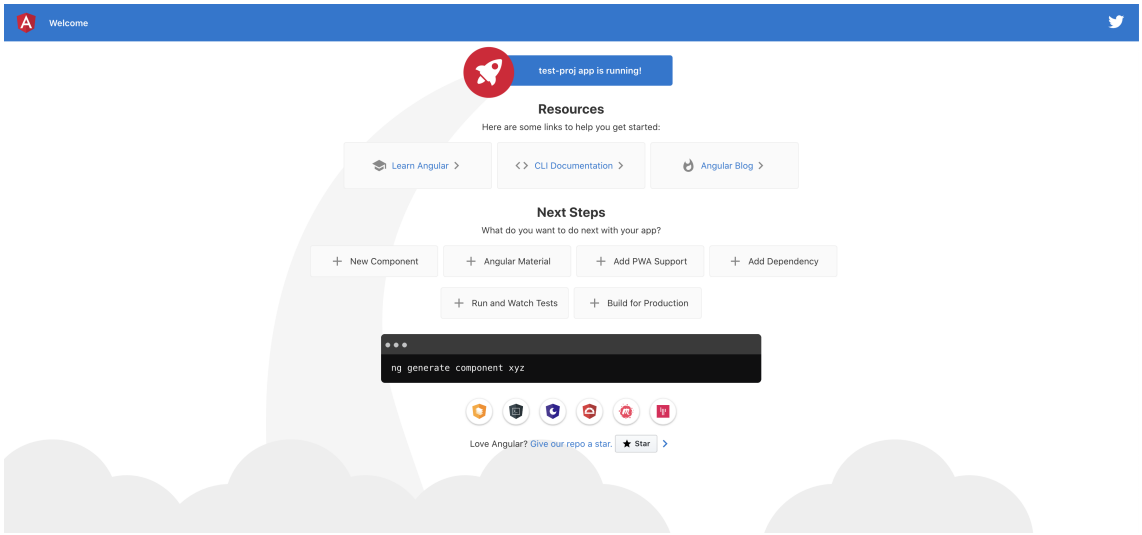
6.2. Figura: Estructura de ficheros del front-end.

El último paso para finalizar la estructura del front-end es crear la propia estructura del proyecto. Esto implica generar todas las carpetas mencionadas previamente y mostradas en la Figura 5.5.

Para el desarrollo de los elementos gráficos se va a utilizar *boostrap*. *Bootstrap* es un popular *framework* de código abierto utilizado para desarrollar interfaces de usuario (UI) y aplicaciones web. Incluye plantillas de una amplia gama de componentes, como botones, formularios, etc.

Para poder hacer uso de *boostrap* en nuestro proyecto, debemos copiar la etiqueta que proporciona *boostrap* en su página y pegarla en nuestro "index.html". Esta etiqueta de script carga un archivo *JavaScript* de *Bootstrap* desde una URL externa.

Debido a que el front-end está desplegado en local en el puerto 4200, cuando nos conectemos a la ruta "localhost:4200" podremos ver la página inicial de Angular indicativa de que la instalación ha sido exitosa (ver Figura 6.3).



6.3. Figura: Página por defecto de Angular.

6.2. Sprint 1: Login

El objetivo de este sistema de login es garantizar la seguridad y la gestión adecuada de los usuarios que interactúan con SiREnO. Mediante la asignación de roles específicos, se busca otorgar diferentes niveles de acceso y funcionalidades a los usuarios según sus responsabilidades y requerimientos en el contexto del sistema.

Desarrollar un sistema de login con roles implica la implementación de diversas funcionalidades, como la autenticación de usuarios, la gestión de sesiones, el control de acceso basado en roles, etc. Todas las funcionalidades descritas se llevarán a cabo mediante la metodología ágil SCRUM, como se ha mencionado en el alcance del proyecto. La implementación del sistema de inicio de sesión se considerará en su totalidad como un sprint.

6.2.1. Análisis y diseño

El sistema de inicio de sesión se desarrollará de forma simultánea tanto para el back-end como para el front-end. En primer lugar, a nivel de servidor, se implementará la lógica completa del sistema y se definirán las rutas necesarias en la API, las cuales serán posteriormente consumidas en el front-end. Mientras se implementa la consumición de estas rutas a nivel de front-end, se va llevar a cabo el diseño gráfico del mismo.

En el back-end, se debe contar con un método que reciba un usuario compuesto por un identificador y una contraseña y valide las credenciales. Para ello, este método, a su vez, deberá invocar a otro método ubicado en la capa de datos, el cual devolverá el rol asociado a dicho usuario en caso de existir. En el propio back-end, será necesario generar un token basado en la información del usuario autenticado, el cual será devuelto al front-end.

En el front-end, además de la creación de los componentes necesarios y la implementación visual de los mismos con ayuda de *boostrap*. También deberemos implementar algo

de lógica del lado del cliente. Es crucial proteger ciertas rutas en función del rol de acceso para que por ejemplo, un usuario logueado en el sistema con rol de alumno no pueda acceder a las funciones de un administrador.

6.2.1.1. Sprint backlog

Las tareas que se deben llevar a cabo para la implementación del login en orden secuencial son las siguientes:

Back-end:

- Definición de las rutas respectivas a los usuarios no logueados e implementación de la lógica de las mismas.
- Definición de métodos en "dbQuerys.js" para recuperar la información necesaria de la base de datos.

Front-end:

- Creación y desarrollo del componente "login".
- Creación de los tres componentes referentes a los índices. El usuario debe ser redirigido a uno de esos componentes según el rol al que pertenece (docente, alumno o admin)
- Crear un servicio que consuma la API del back-end para la autenticación.
- Configurar los guards de Angular para restringir el acceso a las rutas en función del rol subyacente al token.

6.2.2. Implementación

Back-end:

En la carpeta "routes", deberemos crear un archivo "user.js" que almacenará el conjunto de rutas pertenecientes a los usuarios no logueados. En él se comenzarán a crear varias rutas. La primera ruta (*"/signIn"*) recibe dos parámetros: un nombre de usuario y una contraseña. Estos parámetros se utilizan para buscar dicho usuario en la base de datos. Si se encuentra un usuario que coincide con los parámetros proporcionados, se devuelve un token. Este token contiene información adicional, como el *username* y el rol asociado.

Es importante recordar que la lógica de negocio es independiente de la implementación de la base de datos. Por lo tanto, la llamada a la base de datos no se realiza directamente dentro del método de la lógica. En su lugar, se debe crear un método llamado "*getUser(user,pass)*" en el archivo "dbQuerys.js". Este método se encargará de realizar la búsqueda correspondiente en la base de datos y será invocado por el método mencionado anteriormente cuando se deseen recuperar usuarios. El código del *endpoint* POST asociado la ruta "*/signin*" se puede ver en el siguiente cuadro:

```
router.post('/signin', (req, res) => {
   const { user, pass } = req.body;
   dbQuery.getUser(user, pass, (err, userData) => {
      if (!err) {
         const secretKey = config.secretKey;
         const token = jwt.sign(userData, secretKey);
         res.json(token);
      } else {
         res.json(err);
      }
   });
});
```

Este fragmento de código se activa cuando se realiza una solicitud a la ruta *"/signIn"*. Esta ruta invoca el método correspondiente de la capa de datos para buscar un usuario que coincida con los parámetros de entrada. Si se encuentra un usuario coincidente, se genera un token de acceso y se envía al front-end en formato JSON.

Front-end:

El primer paso en el desarrollo del front-end es la creación de las barras de navegación (navbars). Cada navbar es un componente y por tanto deber ir dentro de la carpeta "components", y a su vez, dentro de la carpeta "navbars". Se van a generar cuatro barras de navegación; una por cada rol y otra para la pantalla de inicio. Cada una deberá implementar la lógica de negocio correspondiente. Para generar un componente que represente un navbar, se debe ejecutar el siguiente comando:

```
ng generate component nabvar-inicio
```

Posteriormente y con el mismo comando vamos a generar los componentes "index-docentes", "index-alumnos", "index-admins y "login" en la carpeta "components". De momento solo vamos a trabajar en el de "login". Este componente de login consta de un formulario con los campos de usuario y contraseña. Al pulsar el botón del formulario, mediante la función "*logIn()*" se van a mandar estos datos al servidor y el servidor nos debe devolver el token. Para comunicarnos con el servidor (back-end) debemos consumir la API. Para ello, tenemos que crear un servicio dentro de la carpeta "services":

```
ng generate service auth
```

En este servicio, se creará una función llamada "*sigin(user)*" encargada del proceso de inicio de sesión. Esta función recibirá un objeto de usuario como parámetro y se encargará de consumir la URL "*/user/sigin*" de la API previamente desarrollada en el back-end. Al hacer la solicitud a esta URL, si las credenciales de usuario y contraseña son válidas, la API responderá con un token de acceso.

Dependiendo del valor del token recibido, el usuario será redirigido a la página de inicio correspondiente. A partir de este punto, el token de acceso será requerido para realizar cualquier petición al back-end, por lo que debe ser incluido en la cabecera de cada solicitud.

Por tanto, el siguiente paso es almacenar el token de acceso en el *Local Storage* del navegador para permitir que la aplicación web lo retenga de manera persistente. A continuación, se muestra el método "*logIn()*" dentro del componente "inicio". Este método se invoca al presionar el botón del formulario de inicio de sesión y a su vez, llama al servicio recién creado para realizar el proceso de inicio de sesión.

```
login() {
  this.authService.getRole(this.user).subscribe((res: any) => {
    localStorage.setItem(‘‘token’’, res.token);
    const decoded = decode(token);
    const { role } = decoded;
    switch (role) {
      case ‘docente’:
        this.router.navigate([’indexDocentes’]);
        break;
      case ‘alumno’:
        this.router.navigate([’indexAlumnos’]);
        break;
      case ‘admin’:
        this.router.navigate([’indexAdmins’]);
        break;
    }
  });
}
```

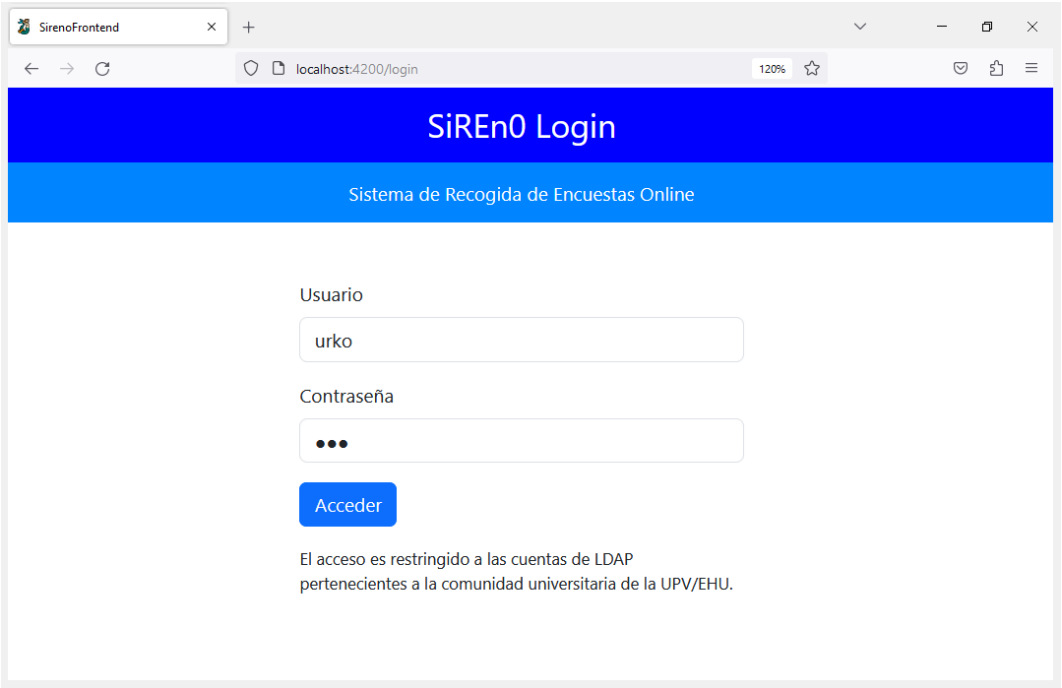
Vamos a crear una serie de guards dentro de la carpeta "guards" para establecer la protección y accesibilidad a las rutas. Para crear los guards necesarios se debe ejecutar el siguiente comando:

```
ng generate guards auth

ng generate guards role
```

El primer *guard* verifica la existencia y validez de un token de usuario almacenado. Si ambas condiciones se cumplen, el *guard* permite el acceso a la página protegida. El segundo *guard*, además de consumir el primero, verifica si el usuario que intenta acceder a la URL tiene el rol necesario. Para esto, se decodifica el token y se verifica el valor de los metadatos subyacentes, el que nos interesa es el rol. A su vez, cada página debe requerir un "*ExpectedRole*" que indica "con qué rol puedo conectarme a esta interfaz". Si el "*ExpectedRole*" y el rol del usuario coinciden, el *guard* permite el acceso a la página protegida.

Por último, se debe implementar el *.html* del componente "login" para añadir un formulario y poder iniciar sesión. Haremos uso de *boostrap* para crearlo. El resultado se puede observar en la siguiente Figura 6.4.



6.4. Figura: Formulario de login del front-end.

6.2.3. Revisión del sprint

Una vez terminado el sprint, se realiza una reunión en la que se le muestra al cliente el trabajo completado durante el sprint. El objetivo principal de la revisión del sprint es obtener retroalimentación y validar si se han cumplido los objetivos establecidos. Tras la revisión de las funcionalidades, las implementadas eran correctas pero surgieron algunas necesidades extra:

- La aplicación debe ser multiidioma y soportar la adición de un idioma de manera sencilla en el futuro, por si fuera necesario.
- Un usuario puede tener más de un rol, es decir, un docente puede ser a su vez alumno o administrador. Por tanto, en el proceso de login, en caso de tener varios roles, se le debe preguntar al usuario con cual de ellos desea acceder.
- Una vez se está logueado no se debe poder acceder a la pantalla de login. Además, debe crearse un mecanismo para poder cerrar la sesión.

La **primera funcionalidad adicional** consiste en adaptar las interfaces para proporcionar soporte a una aplicación multilingüe, permitiendo la adición de más idiomas en el futuro. Para lograr esto, se deben crear diferentes archivos de *strings*, uno para cada idioma deseado. En este caso, se han creado tres: castellano, euskera e inglés. Para que el

usuario pueda cambiar el idioma, se deben añadir tres botones en la barra de navegación cada uno de ellos respectivo a un idioma.

Se ha implementado un nuevo servicio de Angular llamado "LanguageService'", el cual se encarga de gestionar los idiomas a nivel global en el front-end. Cada componente que requiera traducciones debe suscribirse a los métodos proporcionados por este servicio.

La función "*ngOnInit()*" se ejecuta automáticamente al iniciar un componente en Angular. Por lo tanto, dentro de esta función se realiza la gestión del idioma para cada componente. En primer lugar, se establece el idioma predeterminado al iniciar la aplicación. Luego, se suscribe al flujo de cambios del idioma actual a través de la propiedad "currentLanguage" del servicio "LanguageService". Cuando se produce un cambio en el idioma actual, se cargan los *strings* correspondientes al nuevo idioma utilizando el método "*loadStrings(lang)*" del servicio "LanguageService":

```
ngOnInit() {
   const lang = 'es';
   this.languageService.currentLanguage$.subscribe(lang => {
      this.languageService.loadStrings(lang).subscribe(
         data => {
         },
         this.strings = data;
         error => {
         }
      });
   });
}
```

La **segunda ampliación** es crear una verificación multirrol. Para realizar esta verificación, es necesario realizar un cambio en la nomenclatura de los roles. En lugar de utilizar *strings*, se utilizarán identificadores numéricos. Cada identificador representará de manera única una combinación de roles. A continuación se muestra la asociación de los identificadores numéricos y los valores nominales de los roles:

1. Docente.
2. Alumno.
3. Administrador.
4. Docente y alumno.
5. Docente y administrador.
6. Docente, alumno y administrador.

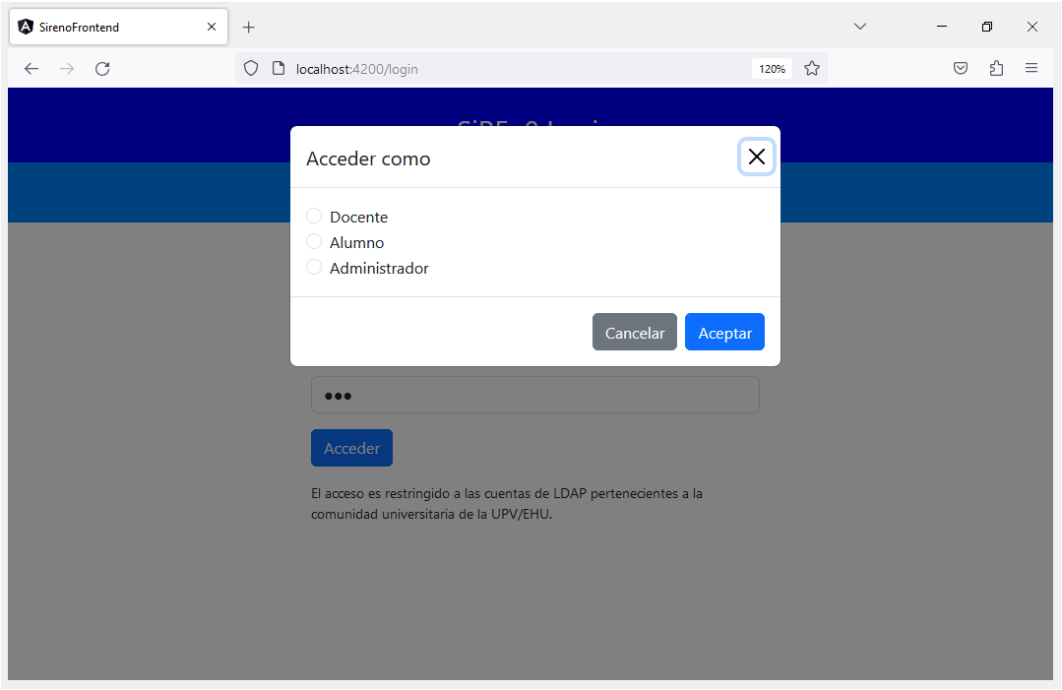
Es necesario realizar una modificación en el proceso de inicio de sesión, de manera que se obtenga el rol del usuario que desea acceder antes de verificar las credenciales. Esto se

lleva a cabo con el propósito de permitir al usuario seleccionar, si es necesario, mediante un diálogo emergente, con cuál de sus roles desea acceder. Para ello, en el back-end, en el módulo "users.js" vamos a crear una ruta adicional (*"/getRoles"*) que recibe dos parámetros: un nombre de usuario y una contraseña. Si se encuentra un usuario que coincide con los parámetros proporcionados, se devuelve el id del rol asociado.

En el desarrollo del front-end, es necesario implementar múltiples *pop-ups* (diálogos emergentes) que se le mostrarán al usuario para que pueda seleccionar el rol con el que desea identificarse en caso de disponer de varios roles asociados. Estos *pop-ups* se implementarán como componentes de Angular y se almacenarán en la carpeta "components/shared/popups".

Además, se requerirá modificar el proceso de inicio de sesión en el front-end. Después de que el usuario ingrese su nombre de usuario y contraseña y haga clic en el botón de acceso, se enviará una solicitud al back-end para verificar el rol asociado. Para realizar esta verificación, se utilizará el servicio previamente configurado, consumiendo el recurso "/getRoles" de la API y pasando como parámetro el usuario actual.

Si el rol devuelto por el back-end es 0, 1 o 2, se ejecutará el proceso de inicio de sesión descrito en la subsección anterior. En caso de que el rol del usuario actual sea 3, 4 o 5, significa que se trata de un caso de multirrol. En este escenario, se mostrará el *pop-up* correspondiente al usuario, y una vez que elija el rol pertinente, se le otorgará acceso al sistema bajo ese rol. En la Figura 6.5 se muestra el ejemplo de un usuario con los roles "docente", "alumno" y "admin" intentando acceder al sistema:



6.5. Figura: Proceso de elección de roles al iniciar sesión.

La **tercera y última funcionalidad adicional** discutida durante la revisión del sprint implica que, una vez que un usuario se haya autenticado, no se le permitirá acceder a la pantalla de inicio de sesión. Para que el usuario pueda acceder a la pantalla de login, éste no debe disponer de ninguna sesión abierta. Por ello, es necesario implementar también un mecanismo para cerrar la sesión.

Por un lado, para evitar el acceso al formulario de roles una vez que los usuarios hayan iniciado sesión, es necesario agregar un *guard* al componente de inicio de sesión. La lógica de este *guard* es opuesta a la explicada anteriormente. En lugar de verificar si el usuario tiene un token válido para permitir el acceso al recurso solicitado, en este caso debemos hacer lo contrario. El *guard* verifica la no existencia de ningún token almacenado en el almacenamiento local. Si se cumple esta condición, se otorga acceso para acceder a la página de inicio de sesión. En caso contrario, se cierra la sesión existente para poder iniciar sesión con otra cuenta.

Por otro lado, se ha implementado un mecanismo para cerrar la sesión. Una vez que el usuario ha iniciado sesión en el sistema, se muestra la opción de "Cerrar sesión" en la barra de navegación. Al hacer clic en este botón, se invoca el método "*onLogoutClic()*" del componente correspondiente, el cual a su vez llama al servicio "auth" para eliminar el token del almacenamiento local. Una vez completada esta operación, se redirige al usuario a la página principal de inicio de sesión. Este mecanismo garantiza que el usuario pueda finalizar su sesión de manera segura y que se elimine correctamente la información de autenticación almacenada en el *Local Storage*.

6.2.4. Pruebas

Para las pruebas realizadas al sistema de login por roles de SiREnO, se han tenido en cuenta las diferentes acciones que se pueden realizar sobre él. A continuación, se presentan las tablas con las pruebas realizadas:

Prueba	Rol	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.1. Login con usuario y pass incorrectas.	*indiferente*	Se debe mostrar por pantalla el mensaje "Usuario o clave incorrectos".	Se muestar el mensaje por pantalla.	Sí
1.2. Login con usuario incorrecto y pass correcta.	*indiferente*	Se debe mostrar por pantalla el mensaje "Usuario o clave incorrectos".	Se muestar el mensaje por pantalla.	Sí

Prueba	Rol	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.3. Login con usuario correcto y pass incorrecta.	*indiferente*	Se debe mostrar por pantalla el mensaje "Usuario o clave incorrectos".	Se muestar el mensaje por pantalla.	Sí
. 1.4. Login con usuario y pass correcta	0: Docente	Acceder al índice de docentes con un mensaje (temporal y mientras dure el desarrollo) de: "Bienvenido 'rol' ".	Acceso correcto y mensaje "Bienvenido docente".	Sí
	1: Alumno	Acceder al índice de alumnos con un mensaje (temporal y mientras dure el desarrollo) de: "Bienvenido 'rol' ".	Acceso correcto y mensaje "Bienvenido alumno".	Sí
	2: Administrador.	Acceder al índice de admins con un mensaje (temporal y mientras dure el desarrollo) de: "Bienvenido 'rol' ".	Acceso correcto y mensaje "Bienvenido admin".	Sí
	3: Docente y alumno	Aparición de *pop-up* con opciones: "docente". y "alumno"	Aparición de *pop-up* con opciones: "docente" y "alumno".	Sí
	4: Docente y administrador	Aparición de *pop-up* con opciones: "docente" y "admin".	Aparición de *pop-up* con opciones: "docente" y "admin".	Sí

Prueba	Rol	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.4. Login con usuario y pass correcta	5: Docente, alumno y administrador	Aparición de *pop-up* con opciones: "docente", "alumno" y "admin".	Aparición de *pop-up* con opciones: "docente", "alumno" y "admin".	Sí
1.5. Acceder a la pantalla de login con sesión abierta.	*indiferente*	No debe redireccionar.	No redirige a login ya que el *guard* funciona pero hace una redirección a la pantalla inicial (localhost:4200) y el navegador se queda en blanco ya que no puede acceder a ninguna ruta.	**No**
2.1. Desde una pantalla con verificación de rol, borrarlo manualmente del almacenamiento local y refrescar la página.	*indiferente*	Debe redirigir al login.	Redirige al login.	Sí
2.2. Cerrar sesión.	*indiferente*	Debe borrar el token del almacenamiento local y redirigir al login.	Borra el token y redirige al login.	Sí
2.3. Con la sesión cerrada, intentar acceder a alguna ruta protegida con guard.	*indiferente*	No debe permitir el acceso a la página deseada y debe mostrar un mensaje (temporal) por consola.	No redirecciona al recurso solicitado y se muestra el mensaje: "La sesión ha expirado".	Sí

Prueba	Rol	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
3.1. Cambiar el idioma al Inglés.	*indiferente*	Se deben traducir todos los componentes de la pantalla actual, no solo la barra de navegación.	Se traducen todos los componentes.	Sí
3.2. Cambiar el idioma al Euskera.	*indiferente*	Se deben traducir todos los componentes de la pantalla actual, no solo la barra de navegación.	Se traducen todos los componentes.	Sí
3.3. Cambiar el idioma al Castellano.	*indiferente*	Se deben traducir todos los componentes de la pantalla actual, no solo la barra de navegación.	Se traducen todos los componentes.	Sí

6.3. Sprint 2: Listar encuestas a alumnos

Este segundo sprint aborda la funcionalidad de listar las encuestas válidas o las actualmente abiertas a los alumnos. Al iniciar sesión en SiREnO con el rol de alumno, éstos serán redirigidos a su índice principal, donde obtendrán un listado de las encuestas disponibles acorde a las situaciones docentes a las que pertenecen. Este listado presenta tanto las las encuestas cerradas pendientes de ser abiertas por el docente, como las encuestas abiertas, es decir, aquellas con plazo vigente para poder responder al formulario asociado. Es recomendable refrescar los diferentes conceptos relacionados con el vocabulario específico del proyecto explicados en el apartado 2.1.2.

En el caso de las encuestas abiertas, se muestra el tiempo restante hasta que el plazo para poder responder al formulario asociado finalice. Además de las encuestas abiertas, también se muestran las encuestas asociadas a una campaña válida que aun están cerradas, esto se hace con la finalidad de notificar al alumno sobre el derecho de responder en algún momento al formulario asociado.

6.3.1. Análisis y diseño

El desarrollo se realizará primero a nivel de back-end. Se quiere que, al acceder a una ruta específica de la API proporcionando el token en la cabecera de la petición, se pueda **obtener el listado de encuestas asociadas a una situación docente válida del alumno**, sin requerir ningún otro parámetro adicional. Recordemos que:

- Una situación docente tiene una fecha de inicio y una fecha de fin que marca la validez de la misma. Estas fechas de inicio y fin son las que le dan sentido al concepto “campaña”.
- Dentro de ese período de tiempo, es posible establecer plazos para responder a la encuesta asociada a esa campaña. A esta acción se le llama “abrir la encuesta”.

Como se mencionó previamente, al alumno se le mostrará el conjunto de encuestas asociadas a una campaña válida. Sin embargo, además de esta condición, **se deben cumplir uno de estos dos criterios adicionales**. El primero, es que la encuesta debe estar actualmente abierta, es decir, que el plazo para responder al formulario aún sea vigente. El segundo, es que la encuesta no se haya abierto nunca. Recalcar que al alumno no se le van a mostrar aquellas encuestas válidas que ya han sido abiertas al menos una vez pero el plazo haya finalizado, las haya respondido o no.

El mostrarle las encuestas asociadas a situaciones docentes válidas nunca abiertas tiene como objetivo informar al alumno que, aunque dicha encuesta aún no haya sido abierta, debería ser abierta al menos una vez dentro del plazo correspondiente, para así brindar el derecho de responder al formulario de satisfacción asociado.

Las encuestas se devolverán en formato JSON y cada una estará compuesta por unos datos de interés. Este es un ejemplo de una encuesta:

```
{
    ‘‘cod_campana’’: ‘‘camp002’’,
    ‘‘nombre_campana’’: ‘‘Segundo Cuatri. 22/23’’,
    ‘‘fecha_fin’’: ‘‘2023-07-21 21:59:59’’,
    ‘‘abierta_antes’’: 1,
    ‘‘cod_encuesta’’: ‘‘enc002’’,
    ‘‘cod_situacion_docente’’: ‘‘sd002’’,
    ‘‘cod_asignatura’’: ‘‘asig002’’,
    ‘‘nombre_asignatura’’: ‘‘Derecho Civil’’,
    ‘‘cod_docente’’: ‘‘doc002’’,
    ‘‘nombre_docente’’: ‘‘Marta Gonzalez’’,
    ‘‘num_curso’’: 2,
    ‘‘año_curso’’: ‘‘22/23’’,
    ‘‘fecha_fin_activacion’’: ‘‘2023-06-17T13:00:00.000Z’’
}
```

A nivel de front-end, una vez consumido el recurso de la API para obtener el conjunto de encuestas del usuario, se deben mostrar de manera atractiva. Se va a usar un componente “card” de *boostrap* para hacer la presentación de cada encuesta.

En el ámbito del front-end se requiere gestionar la visibilidad de las encuestas, ya que cuando un alumno realiza una solicitud para obtener las encuestas, es posible que en ese momento algunas de ellas tengan una fecha de expiración próxima. Por lo tanto, cuando esa fecha llegue, dichas encuestas deben dejar de mostrarse. Gestionar esto a nivel de front-end nos evita tener que hacer llamadas periódicas al back-end para obtener las encuestas a mostrar.

6.3.1.1. Sprint backlog

Las tareas que se deben llevar a cabo para la implementación del sprint son las siguientes:

Back-end:

- Definición de las rutas respectivas a los alumnos e implementación de la lógica des la mismas.
- Creación de un *middleware* para verificar validez e identidad del token entrante.
- Definición de los métodos necesarios en "dbQuerys.js" para recuperar las encuestas asociadas al usuario.

Front-end:

- Creación y desarrollo de los componentes de las encuestas.
- Configuración de un servicio que consuma la API del back-end para obtener el conjunto de encuestas válidas del alumno.
- Creación dinámica de los componentes de las encuestas según los datos recuperados del back-end.
- Gestión de la validez de las encuestas.

6.3.2. Implementación

Back-end:

En primer lugar, es necesario implementar un *middleware* llamado "tokenVerifier.js" que intercepte el token presente en la cabecera de la petición y analice su validez. Cuando se realice una llamada a cualquier ruta del back-end que requiera autenticación, esta deberá pasar inicialmente por el *middleware* y, posteriormente, redirigirse hacia el recurso solicitado.

El primer paso que realiza el *middleware* es verificar la existencia de una cabecera denominada *authorization*, ya que en dicha cabecera se incluye el token del usuario. En caso de que esta cabecera exista, se procede a comprobar la validez del token. Si el token resulta válido, se decodifica la metainformación (usuario y rol) y se inserta en el cuerpo de la petición. Por último, se continúa con la llamada al recurso solicitado.

En la carpeta "routes", se va a crear otro módulo llamado "alumno.js". Este es análogo al "user.js". En "alumno.js" se van a almacenar el conjunto de rutas pertenecientes a los usuarios identificados como alumnos. La ruta *"/getCampanas"* recibe en el cuerpo de la petición, el usuario y el rol del que proviene la llamada a la API (insertados en el

cuerpo de la petición por el *middleware*). Este método se encarga de llamar la capa de datos mediante la función *"getEncuestas(user)"*.

El back-end devuelve el listado de encuestas asociado al alumno en formato JSON de manera ordenada. Las encuestas abiertas figuran en primer lugar, ordenadas a su vez según la fecha de finalización de su apertura. Después de las encuestas abiertas, se presentan las encuestas cerradas que nunca han sido abiertas pero que aún se pueden abrir.

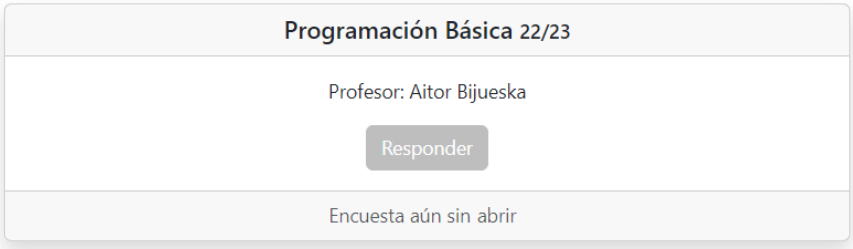
Front-end:

El primer paso es crear dos componentes de Angular usando el comando visto en el Sprint 1. Estos componentes hacen referencia a las encuestas abiertas y cerradas, respectivamente.

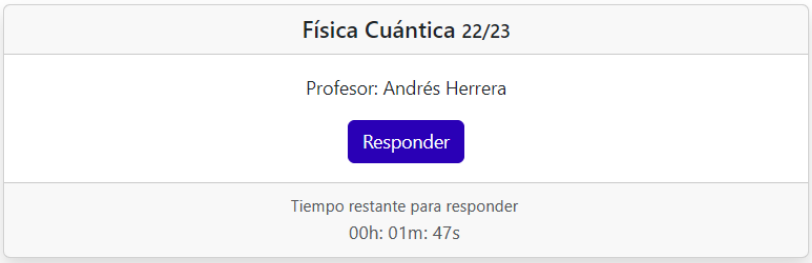
En el componente previamente desarrollado en el sprint anterior, denominado "index-alumnos", el cual representa la página de inicio de los alumnos, se desea mostrar el conjunto de encuestas asociadas al usuario actual. Para lograr esto, es necesario implementar una función que consuma el recurso creado en el back-end.

Una vez obtenido el JSON que contiene todas las encuestas del alumno, se procederá a iterar sobre dicho conjunto y crear un componente individual para cada encuesta. Asimismo, se deberán asignar los atributos correspondientes del objeto "encuesta" con la información obtenida del back-end.

En función de si la encuesta está o no abierta, se creará el componente correspondiente: "encuesta-cerrada" (Figura 6.6) o "encuesta-abierta" (Figura 6.7), respectivamente. La diferencia entre ambos componentes radica en que, en la versión cerrada, el botón de respuesta estará deshabilitado y no se mostrará el tiempo restante para la finalización de la encuesta. En su lugar, se mostrará un mensaje que indica que la encuesta aun no se ha abierto.



6.6. Figura: Componente respectivo a encuesta cerrada.



6.7. Figura: Componente respectivo a encuesta abierta.

Para el componente que representa una encuesta abierta, es necesario calcular el tiempo restante hasta su cierre. Al crear el componente, se realiza una operación para determinar la diferencia entre la "fecha_fin_activación" y la fecha actual. Este tiempo se formatea en "hh:mm:ss" y se muestra en el componente. Sin embargo, se presenta un problema: este tiempo restante no se actualiza a tiempo real en el navegador, ya que se calcula solo una vez; al crear el componente. Para visualizar el tiempo actualizado, sería necesario recargar la página.

Para solucionar este problema, en el método "*ngOnInit()*", que se ejecuta una vez que se ha inicializado un componente, se lleva a cabo una operación para actualizar el tiempo restante de las encuestas utilizando un observable. Cada segundo, se ejecuta la función "*getTiempoCierre()*" para obtener el tiempo restante de las encuestas y se actualiza el valor de 'tiempoRestante' con el resultado obtenido:

```
ngOnInit() {
     this.tiempoRestante = interval(1000).pipe(
          map(() => this.getTiempoCierre())
     );
}
```

Dentro del elemento *div* para mostrar el valor de "tiempoRestante" se usa el filtro *async*. El filtro *async* es una funcionalidad de Angular que permite suscribirse y mostrar los valores de un observable de forma reactiva en la plantilla:

```
<div class=''card-footer text-muted''>
     <small>Tiempo restante</small> <br />
     {{ tiempoRestante | async }}
</div>
```

Cuando el tiempo restante de la encuesta llega a cero, resulta incongruente seguir mostrando la encuesta. Por lo tanto, se actualiza el valor del atributo "mostrarEncuesta" a *false*. Gracias a la cláusula '*ngIf="mostrarEncuesta"' en el *.html*, la encuesta dejará de ser visible para el alumno.

6.3.3. Revisión del sprint

Una vez finalizado el sprint, se lleva a cabo una reunión en la que se le muestra al cliente el trabajo completado durante el periodo de desarrollo. El propósito fundamental de la revisión del sprint es obtener retroalimentación y validar si se han cumplido los objetivos establecidos del mismo. Durante esta reunión, se presentaron las funcionalidades implementadas, las cuales fueron evaluadas como correctas y satisfactorias por el cliente.

Durante la revisión, surgieron algunas pequeñas sugerencias por parte del cliente, ajustes estéticos o cambios menores en las funcionalidades existentes. La implementación era correcta y no se han identificado necesidades extra que pudieran tener un impacto significativo en el proyecto. Esto indica que el desarrollo se ha llevado a cabo de manera satisfactoria y se ha logrado cumplir con los objetivos establecidos para el sprint en cuestión.

6.3.4. Pruebas

Para las pruebas realizadas, se han tenido en cuenta los diferentes posibles escenarios que se pueden dar según los estados posibles de cada de encuesta. A continuación, se presentan las tablas con las pruebas realizadas:

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1. El alumno no tiene ninguna encuesta asociada a campañas válidas.	El alumno obtiene un mensaje indicándole que no hay encuestas disponibles en ese momento.	Se muestra correctamente el mensaje “No tienes encuestas disponibles en este momento.”	Sí
2.1. **encuestas de campañas no válidas**: encuestas asociadas a campañas con fecha de inicio y fin posteriores a la fecha actual.	Las encuestas no válidas no se deben mostrar al alumno.	Las encuestas no válidas no se le muestran al alumno.	Sí
2.2. **encuestas de campañas no válidas**: encuestas asociadas a campañas con fecha de inicio y fin anteriores a la fecha actual.	Las encuestas no válidas no se deben mostrar al alumno.	Las encuestas no válidas no se le muestran al alumno.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
2.3. **encuestas de campañas no válidas**: encuestas asociadas a campañas con fecha de inicio y fin correctas pero ya han sido abiertas previamente y el plazo ha expirado.	Las encuestas no válidas no se deben mostrar al alumno.	Las encuestas no válidas no se le muestran al alumno.	Sí
3. **encuestas de campañas válidas** pero **no abierta**: encuestas asociadas a campañas cuyas fechas de inicio y fin están dentro de la fecha actual y además dicha encuesta no ha sido abierta nunca.	Las encuestas al ser válidas y nunca haberse activado antes, se deben mostrar como un componente “encuesta-cerrada” con el botón ‘responder’ deshabilitado e indicando que la encuesta asociada está cerrada aún.	Las encuestas se muestran como un componente ‘’encuesta-cerrada”.	Sí
4.1. **encuestas de campañas válidas**: encuestas asociadas campañas con fecha de inicio y fin son correctas pero el plazo de apertura no incluye fecha actual.	Las encuestas no se deben mostrar porque el plazo de apertura aun no ha llegado. Este caso en el funcionamiento normal del sistema no se va a poder dar.	Las encuestas no se muestran.	Sí
4.2. **encuestas de campañas válidas**: Encuestas de campañas cuyas fechas de activación y cierre están dentro de la fecha actual. La encuesta tiene un plazo de apertura vigente.	Las encuestas se generan como componentes ‘encuesta-abierta’. Se inicializan todos sus atributos. El tiempo restante para el cierre se actualiza cada segundo.	Las encuestas se muestran correctamente y ordenadas mostrando primero las que tienen un tiempo restante inferior.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
4.3. **encuestas abiertas**: encuestas **abiertas una o más veces** y con fechas inicio y fin de la apertura actual correctas.	Se muestran las encuestas y los datos de cada encuesta son los correspondientes a la última apertura de la misma. Se generan como componentes "encuesta-abierta".	Las encuestas se muestran correctamente y ordenadas. Se muestran los datos correspondientes a la última apertura de la misma.	Sí
5. **mix de encuestas**: Se ha generado un alumno que tiene al menos una encuesta de cada tipo de explicada previamente.	Se deben mostrar las encuestas asociadas a una campaña válida nunca abiertas o las abiertas actualmente. Las encuestas se deben mostrar ordenadas.	Se muestran las encuestas correctas. Las encuestas mostradas están ordenadas, primero todas las abiertas y al final las cerradas. Las abiertas a su vez se muestran ordenadas según el tiempo restante para ser cerradas.	Sí

6.4. Sprint 3: Mostrar formulario de encuesta

El tercer sprint tiene como objetivo presentar al alumno el formulario correspondiente a una encuesta abierta en la cual ha clicado. Este formulario está compuesto por un conjunto de preguntas previamente definidas que el alumno deberá contestar para evaluar al docente y la asignatura; en relación a la situación docente asociada a la encuesta.

Este sprint también se centra en garantizar el correcto acceso de los alumnos al formulario correspondiente de la encuesta abierta que han clicado. El objetivo es evitar que aquellos alumnos que no deben responder una encuesta concreta sean capaces de si quiera visualizar el formulario asociado, así como también prevenir el acceso manual a formularios cuya campañas asociada ha expirado. Además, se busca evitar que, por ejemplo, los alumnos respondan encuestas abiertas en nombre de otros alumnos mediante la modificación de ciertos parámetros del sistema a la hora de cargar el formulario.

6.4.1. Análisis y diseño

Al igual que en el sprint anterior, se seguirá el proceso de diseño e implementación comenzando por el back-end y posteriormente se implementará el front-end para mostrar los recursos consumidos a través de la API. El objetivo es permitir que los alumnos, al hacer clic en una encuesta abierta, puedan visualizar el formulario asociado.

En la pantalla donde se muestra el conjunto de encuestas (sprint 2), se almacenan los códigos de los formularios asociadas a cada encuesta. El proceso de recuperación de información se llevará a cabo utilizando dicho código junto con el token de acceso personal del alumno. En primer lugar, se desea obtener las preguntas relativas al formulario de la encuesta seleccionada. Para cada pregunta en la lista de preguntas de la encuesta, se buscará obtener el texto de la pregunta en el idioma actual.

Cada pregunta está vinculada a una serie de posibles respuestas. Estas respuestas pueden ser numéricas, por ejemplo, un rango del 1 al 5; o pueden ser textuales, por ejemplo, para evaluar el nivel de interés en una asignatura (bajo, alto, muy alto, etc.). Dependiendo del tipo de respuesta, es necesario recuperarla desde el lugar de almacenamiento correspondiente. Además, en el caso de respuestas textuales, se deberá asegurar que la recuperación del texto de la respuesta se realice en el idioma actual.

Una vez se haya obtenido el conjunto de preguntas y respuestas de la encuesta seleccionada, se procederá a generar el componente del formulario en el front-end, permitiendo así al alumno visualizarlo. Asimismo, en el nivel del front-end, se implementarán medidas de seguridad con el objetivo de asegurar que el alumno actualmente autenticado tenga el derecho de acceder, únicamente, a los formularios de las encuestas específicas que deba responder. Así, se quiere evitar que cualquier alumno pueda visualizar cualquier formulario del sistema.

6.4.1.1. Sprint backlog

Las tareas que se deben llevar a cabo para la implementación del sprint son las siguientes:

Back-end:

- Definición de las rutas pertinentes para recuperar los datos de la encuesta solicitada.
- Definición de los métodos necesarios en "dbQuerys.js" para recuperar los datos necesarios de cada encuesta.

Front-end:

- Configuración del servicio existente de comunicación con la API para obtener el conjunto de datos de la encuesta.
- Creación y desarrollo del componente del formulario.
- Generación dinámica del formulario, incluyendo sus preguntas y posibles respuestas.
- Implementación de la lógica y medidas de seguridad para el componente:
  - Actualización automática del formulario al cambiar el idioma.
  - Validación de que todas las preguntas hayan sido respondidas antes de permitir continuar.
  - Control del tiempo de finalización de la encuesta, expulsar a los alumnos que estuvieran respondiendo al formulario en el momento de finalización del plazo.
  - Restricción de acceso a los formularios asociados a las encuestas únicamente para los alumnos autorizados, evitando el acceso no autorizado.

6.4.2. Implementación

Back-end:

En el back-end, se debe crear una nueva ruta para los alumnos (*"/alumno/getFormulario"*) que reciba como parámetros del código del formulario y el idioma deseado. Esta llamada debe devolver el conjunto de preguntas y respuestas correspondientes a dicho formulario. Para ello, este método a su vez, debe invocar otro de la capa de datos encargado de recuperar de la base de datos la información solicitada.

Por tanto, en la capa de datos, se debe crear un método que dado código del formulario y el idioma deseado, devuelva el conjunto de preguntas y respuestas. En la llamada a la base de datos, se recorren todas las preguntas asociadas al código del formulario. Por cada pregunta, se obtiene el código de la pregunta y el texto asociado en el idioma correspondiente. A su vez, por cada pregunta se quieren obtener el conjunto de posibles

respuestas. Esto se lleva a cabo con promesas de *JavaScript*.

Si el conjunto de respuestas a una pregunta es numérica, se deben obtener dichas respuestas de la tabla "respuesta_numerica_de_pregunta". Al ser numéricas no hace falta traducción ya que el propio código de la respuesta es el texto de la respuesta (concretamente, el número).

En caso de que el conjunto de respuestas sea textual, se debe obtener el código de cada respuesta, y ahora sí, el texto asociado a cada una de ellas en el idioma correspondiente; ya que en este caso el id de la respuesta no coincide con el texto de la respuesta. Al finalizar de recorrer las preguntas del formulario, se resuelven todas las promesas pendientes y se devuelve el arreglo con el conjunto de preguntas y respuestas.

Front-end:

El primer paso consiste en desarrollar un componente de Angular que contenga tanto la vista como la lógica del formulario en el front-end. Desde la pantalla anterior, donde se muestra el listado de encuestas disponibles para el alumno, al hacer clic en el botón "Responder" de una encuesta abierta, se debe navegar hasta la vista asociada al componente del formulario correspondiente. Para lograr esto, es necesario pasarle los parámetros requeridos, como el código del formulario, la situación docente, la fecha de finalización de la apertura, etc. Estos parámetros pueden ser pasados durante la navegación utilizando, a priori, los argumentos del objeto *Router*.

Durante la inicialización del componente del formulario, es necesario realizar una llamada a la API con el objetivo de consumir el método recién creado. Esta llamada se hace utilizando el servicio configurado en el front-end. Mediante la llamada se obtiene el conjunto de preguntas y respuestas asociadas al formulario que se desea mostrar, considerando el idioma actual de la página. A partir de esta información, se generará dinámicamente la vista del formulario recorriendo del listado de preguntas y respuestas.

Al traducir la página de SiREnO, es necesario asegurarse de que el formulario también se traduzca. Para lograr esto, se realiza una llamada a la API cada vez que se cambia el idioma, lo que permite obtener nuevamente los textos de las preguntas y respuestas en el idioma actual. Otro aspecto importante es que el formulario no puede considerarse finalizado hasta que se hayan respondido todas las preguntas. Para abordar esta situación, se implementa un *listener* en el botón "Enviar" del formulario. Al hacer clic en dicho botón, se verifica si todas las preguntas tienen una respuesta. Si hay alguna pregunta sin respuesta, se muestra un diálogo emergente con un mensaje indicando que el formulario está incompleto. Por otro lado, si todas las preguntas han sido respondidas, se procedería al envío del formulario.

Además, es necesario gestionar el tiempo restante para responder a las preguntas. Esta gestión se realiza de manera similar a las encuestas. Cuando se agota el tiempo establecido, la vista del formulario finaliza automáticamente y el alumno es redirigido nuevamente al índice principal, donde se muestran su conjunto de encuestas disponibles.

Para garantizar la adecuada gestión de accesos a los formularios, es fundamental implementar un mecanismo de autorización y autenticación que valide la identidad del alumno para acceder a un formulario específico. Es crucial evitar situaciones en las que un alumno sin encuestas abiertas acceda manualmente a un un formulario asociado mediante la URL.

En este sentido, debemos tener en cuenta que los formularios cuentan con metadatos, como el código de la encuesta asociada o el código de la situación docente a la que hacen referencia. Por lo tanto, se ha generado un *guard* intermedio que verificará si el alumno actual (el registrado en el token de acceso) pertenece a la situación docente asociada a la encuesta del formulario que se pretende visualizar. El *guard* también verifica si la encuesta asociada a la situación docente actual está abierta. La implementación del *guard* es la siguiente:

```
if (this.cod_situacion_docente && this.nombreAsignatura && this.nombre_docente &&
this.fecha_fin_activacion && this.cod_encuesta) {

   // Obtener la lista de cod_situaciones del servicio
      return this.authService.getSDsAlumno().pipe(
         map((situaciones_docentes: Object) => {

            // Verificar si cod_situacion_docente está en la lista
            if (Object.values(situaciones_docentes).includes(cod_situacion_docente)) {

               // Verificar si la campaña de la situación docente es activa
               const isAbierta = this.authService.isAbierta(this.cod_situacion_docente);
               if (isAbierta)
                  return true;
               }else
                  this.router.navigate([‘indexAlumnos’]);
                  return false;
               }
            } else {
               // Si cod_situacion_docente no está en la lista, redirigir al usuario
               this.router.navigate([‘indexAlumnos’]);
               return false;
            }
         }),
         take(1) // Tomar solo el primer valor y completar el Observable
      );
   }

   this.router.navigate([‘indexAlumnos’]);
   return false;
```

De esta manera, al implementar este guardia intermedio, se asegura que solo los alumnos autorizados puedan acceder al formulario de las encuestas asociadas a su situación docente, evitando accesos no autorizados y garantizando la confidencialidad y validez de los datos recopilados en el proceso de evaluación.

Gracias al conjunto de alumnos relacionados con las situaciones docentes, se puede conocer qué alumnos deben responder a la encuesta de dicha situación docente. Por tanto, una vez que el alumno haya completado el formulario de la encuesta, se eliminará la

relación correspondiente.

El último problema a gestionar es la persistencia de los metadatos del formulario para garantizar que no puedan ser alterados por el usuario. Como se ha mencionado anteriormente, en la navegación se pueden pasar los argumentos utilizando el objeto *Router*. Sin embargo, esto genera una URL con parámetros, y si el alumno modifica dicha URL, se alteran los valores internos del sistema. Por ejemplo, si modifica el tiempo restante de la apertura de la encuesta en la URL, puede llegar a tener tiempo ilimitado para responder al formulario asociado, o al menos para visualizar el formulario, ya que en el momento de enviar el formulario, si la apertura de la encuesta ha caducado, dicho formulario no se debería guardar.

Existen diferentes enfoques para solucionar este problema. Uno de ellos es validar y persistir todos los datos para evitar que sean modificados al alterar la URL. Otra opción más sencilla es utilizar un servicio para pasar los parámetros en lugar del objeto *Router*. A través de los métodos *"setData()"* y *"getData()"*, podemos transmitir internamente los parámetros necesarios de un componente a otro.

Al emplear este enfoque, nos aseguramos de que los metadatos se mantengan intactos y protegidos contra cualquier intento de manipulación por parte del usuario. Además, se simplifica el proceso de comunicación de parámetros entre componentes, lo que contribuye a una mayor seguridad y confiabilidad en la aplicación.

6.4.3. Revisión del sprint

Una vez finalizado el sprint, como siempre, se lleva a cabo una reunión en la que se le muestra al cliente el trabajo completado durante el periodo de desarrollo. Tras las revisión del sprint surgieron algunos cambios menores debido a una captura de requisitos que no fue perfectamente detallada. Los cambios a realizar fueron:

- El formulario debe permitir su envío incluso si no se han respondido todas las preguntas. Se debe eliminar la condición de que todas las preguntas estén respondidas antes de proceder a enviar el conjunto de respuestas. Sin embargo, se debe mostrar una advertencia en forma de diálogo si existen preguntas sin responder al momento de enviar el formulario. Si el alumno pulsa el botón "Aceptar" del diálogo, se procederá al envío de las respuestas.
- Se debe agregar un botón de cancelación que interrumpa el proceso de respuesta al formulario. Al hacer clic en él, aparecerá un diálogo indicando que abandonar el formulario resultará en la pérdida del progreso de las respuestas. Si se pulsa en el botón "Aceptar" del diálogo, la respuesta al formulario en curso se cancelará y el alumno será redirigido a su pantalla de inicio.
- Se debe reemplazar la palabra "profesor" por la palabra "docente" para asegurar que la palabra carezca de género.

6.4.4. Pruebas

Estas pruebas cubren los casos críticos y garantizan la correcta generación de las encuestas, restringiendo su visualización únicamente a los alumnos correspondientes.

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1. Abrir el formulario asociado a encuesta cerrada.	No debe ocurrir nada.	No ocurre nada, no se abre el formulario porque el botón está deshabilitado.	Sí
2.1. Pinchar en una encuesta abierta asociada al tipo de formulario nº1.	Se debería abrir el tipo de formulario nº1 (formulario de grado) con todos los datos cargados.	Se abre el formulario de grado con todos los datos cargados.	Sí
2.2. Pinchar en una encuesta abierta asociada al tipo de formulario nº2.	Se debería abrir el tipo de formulario nº2 (formulario de máster) con todos los datos cargados.	Se abre el formulario de máster con todos los datos cargados.	Sí
3.1. Intentar enviar el formulario sin responder ninguna pregunta.	Debería saltar el *pop-up* indicando la situación.	Salta el *pop-up* dando del aviso.	Sí
3.2. Intentar enviar el formulario con alguna pregunta sin responder	Debería saltar el *pop-up* indicando la situación.	Salta el *pop-up* dando del aviso.	Sí
3.3. Intentar enviar el formulario con todas las preguntas respondidas	Debería aparecer por consola un mensaje simulando que el formulario se ha enviado (el envío aun no está implementado).	Aparece el mensaje en consola indicando el correcto envío del formulario.	Sí
4. Comprobar que se cierre el formulario al terminar el plazo.	Al terminar el plazo el formulario se debe cerrar, no se debe enviar nada y le debe redirigir al inicio.	El formulario se cierra, no se envía y la redirección es exitosa.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
5.1. Intentar acceder a un formulario sin haberse logueado.	No se debería poder acceder al formulario y nos debería mantener en la pantalla de login. Ya que acceder al formulario pertenece a la ruta del alumno está protegida por el *guard* que verifica los roles.	Se prohíbe el acceso sin autenticación al formulario y nos mantiene en la pantalla de login.	Sí
5.2. Intentar acceder a un formulario asociado a la encuesta de otro usuario a la cual NO pertenezco.	No se debería poder acceder al formulario y nos debería mantener en la pantalla de login. Ya que está protegida por el *guard* que verifica la validez de los datos y compara la situación docente del alumno y el de la encuesta del formulario.	Se prohíbe el acceso al formulario y nos mantiene en la pantalla actual.	Sí
5.3. Intentar acceder a un formulario perteneciente a otro usuario, a la cual también estoy vinculado mediante la situación docente de la encuesta del formulario.	No se debería poder acceder al formulario y nos debería mantener en la pantalla de login. Ya que está protegida por el *guard* que verifica la validez de los datos	Se prohíbe el acceso al formulario y nos mantiene en la pantalla actual.	Sí
5.4. Intentar acceder manualmente a un formulario de una situación docente propia pero cuya encuesta no está abierta	No se debería poder acceder al formulario, ya que está protegida por el *guard* que verifica la validez de los datos	Se prohíbe el acceso al formulario.	Sí

6.5. Sprint 4: Responder a un formulario

En el anterior sprint se ha llevado a cabo el proceso de mostrar un formulario. En este cuarto sprint se va a implementar el proceso de responder al formulario mostrado. Cuando el alumno seleccione las respuestas pertinentes a las preguntas del formulario y pulse el botón "Enviar" las respuestas a cada pregunta deben ser persistidas y almacenadas en la base de datos.

Además de la importancia de registrar de manera adecuada cada respuesta en relación con la situación docente correspondiente, resulta crucial asegurar el anonimato de las respuestas. Se busca asegurar la privacidad y confidencialidad de las respuestas, evitando que se pueda rastrear o asociar directamente una respuesta con un alumno específico. Esto es especialmente relevante en el contexto de encuestas y cuestionarios, donde se recopilan datos sensibles sobre la experiencia y las opiniones de los alumnos.

En este sentido, en este sprint también se implementarán medidas de seguridad en el servidor para garantizar que solo los alumnos autorizados, con situaciones docentes activas, puedan acceder e iniciar el proceso de almacenamiento permanente de las respuestas en la base de datos a través de la API REST. Para proteger el anonimato de los alumnos encuestados, una vez se responda a un formulario asociado a cierta encuesta, se debe eliminar cualquier información que pudiera identificar al alumno que debía responder a la encuesta de esa situación docente en particular.

6.5.1. Análisis y diseño

Como se ha mencionado anteriormente, cuando un alumno pulse el botón "Enviar" en un formulario, es necesario guardar las respuestas seleccionadas en la base de datos. Con el fin de lograr esto, desde el front-end, que es el punto de inicio del proceso de comunicación, se debe recopilar el conjunto de preguntas y respuestas seleccionadas por el alumno en formato JSON. Una vez que el JSON se ha creado, debe enviarse al back-end junto con el código correspondiente a la situación docente a la cual hace referencia la encuesta del formulario actual.

En el back-end, se recibe el JSON de respuestas junto con el código de la situación docente. Utilizando estos datos, se inicia un proceso de validación y persistencia con el objetivo de almacenar un conjunto de enviadas por un alumno. Una vez se ha verificado la autenticidad del alumno y se ha validado la situación docente obtenida desde el front-end, se procede a verificar si el alumno tiene pendiente responder dicha situación docente y si la encuesta asociada a dicha situación docente está abierta.

Después de verificar que todos los procesos de validación son correctos, se procede a recorrer el conjunto de respuestas seleccionadas por el alumno y se insertan en la base de datos. Una vez finalizado este proceso, se realiza la eliminación de la la relación que vincula la propia situación docente con el alumno. Esta eliminación indica que el alumno ha respondido al formulario asociado a la encuesta de dicha situación docente y, por lo tanto, no debe volver a responderla. Al eliminar la situación docente del conjunto de situaciones

pendientes del alumno, se logra proporcionar anonimato al evitar la trazabilidad de quién ha respondido, al menos exclusivamente a través de los datos en SiREnO. En el caso de que solo un alumno haya respondido, alguien de administración con acceso tanto a la base de datos de SiREnO como a GAUR podría verificar qué único alumno matriculado no aparece en la lista de situaciones pendientes y deducir de quién es la respuesta. Aunque esta medida no garantiza una imposibilidad del 100 % para determinar la identidad, sí dificulta mucho dicha tarea.

6.5.1.1. Sprint backlog

Las tareas que se deben llevar a cabo para la implementación del sprint son las siguientes:

Back-end:

- Definición de las rutas pertinentes para recoger las preguntas y respuestas del usuario enviadas desde el front-end.
- Implementación de medidas de seguridad previas a la persistencia de las respuestas:
  - Verificación de la apertura de la encuesta asociada.
  - Autenticación y validación de la situación docente asociada al usuario actual.
- Definición de los métodos necesarios en "dbQuerys.js" para persistir las respuestas de un formulario.

Front-end:

- Configuración del servicio existente de comunicación con la API para enviar al back-end el conjunto de respuestas del formulario.
- Implementación de la lógica para el envío de las respuestas

6.5.2. Implementación

Back-end:

A nivel de servidor se va a crear un nuevo método en la capa de coordinación del alumno, encargado de validar y persistir las respuestas de un formulario que se recibe desde el front-end. Para ello se va a crear una nueva ruta llamada *"/setRespuestas"* que obtiene como parámetros el conjunto de respuestas, el token del alumno que acaba de enviar dichas respuestas y la situación asociada a la encuesta del formulario.

El primer paso consiste en validar si el alumno del token tiene pendiente de responder la encuesta asociada a la situación docente proporcionada como parámetro. Para lograr esto, se llama al método existente de la capa de datos y se obtienen todas las situaciones pendientes del alumno. Por último, se verifica si la situación docente está presente en esa

lista. En caso de ratificar esta condición se pasa al segundo nivel de validación.

El segundo paso implica verificar si la encuesta asociada a la situación docente se encuentra abierta. Si se confirma su apertura vigente, se supera la segunda y última medida de autenticación, lo cual da paso al proceso de persistencia de las respuestas en la base de datos.

Para llevar a cabo la persistencia de las respuestas en la base de datos, se requiere realizar una llamada adicional a la capa de datos, específicamente al método *"setRespuestas()"*. Este método debe recibir como parámetros el alumno asociado al token de acceso vigente y la situación docente de la encuesta asociada al formulario actual. En este método, se realiza un recorrido por el conjunto de respuestas para cada pregunta. Todas las consultas quedan pendientes de resolución para ser persistidas en la base de datos; al finalizar el recorrido, se ejecutan en serie utilizando una función recursiva llamada *"executeQueries()"*.

Finalmente, es imprescindible eliminar de la base de datos la entrada que vincula al alumno con la situación docente que acaba de responder haciendo uso del formulario asociado a la encuesta de la propia situación docente. Al eliminar esta relación, se indica que el alumno ya no tiene pendiente responder dicha encuesta.

Front-end:

Desde la interfaz de usuario en el front-end, deben enviar las respuestas seleccionadas por el alumno en el formulario. El primer paso es estructurar las preguntas respondidas junto con la respuestas seleccionadas. Para ello, se realiza un proceso iterativo sobre el propio formulario con el fin de generar un objeto JSON que agrupe cada pregunta respondida con su respectiva respuesta.

Haciendo uso del servicio que comunica el front-end con el back-end, se va a enviar el JSON recién creado al método recién creado usando la URL "*/alumno/setRespuestas*". Con esta llamada, invocamos al método del back-end encargado de validar y almacenar las respuestas de manera persistente.

En caso de que el envío de las respuestas devuelva un código de respuesta exitoso, se redirige al alumno a la página principal y se muestra un cuadro de diálogo emergente que indica que las respuestas del formulario ha sido enviada satisfactoriamente. Sin embargo, si se produce algún error durante el proceso de envío de las respuestas, se muestra otro cuadro de diálogo que señala la situación desfavorable, y se mantiene al alumno en la vista del formulario para que pueda intentar enviarlo nuevamente.

En este sprint no se ha tenido que crear ningún componente o interfaz adicional. El envío de la encuesta se realiza desde la interfaz creada en el sprint 3.

6.5.3. Revisión del sprint

El sprint recién implementado ha logrado satisfacer todos los requisitos funcionales establecidos previamente. Durante la etapa de planificación, se llevó a cabo una cuidadosa identificación y documentación de los requisitos del proyecto. Esto proporcionó una base sólida para el desarrollo del sprint y permitió tener una visión clara de los objetivos a alcanzar.

Gracias a esta planificación, análisis y diseño inicial sólidos, el sprint pudo implementarse exitosamente y cumplir con todos los requisitos acordados. La buena comprensión de los requisitos y una cuidadosa planificación permitieron minimizar los riesgos y maximizar la eficiencia del desarrollo.

6.5.4. Pruebas

El objetivo de estas pruebas es garantizar el correcto envío de las respuestas asociadas los formularios. Durante estas pruebas, se abordarán todos los escenarios críticos para verificar su funcionamiento adecuado.

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.1. Enviar respuestas sin token de autenticación.	El *middleware*no debe dejar proceder.	El middleware bloquea el acceso al método con un mensaje de “No autorizado”.	Sí
1.2. Enviar respuestas para una encuesta de una situación docente que no debo responder.	No se deben insertar las respuestas y se debe notificar con un mensaje de error.	No se insertan las respuestas y se notifica con un código de error y un mensaje: “Alumno no autorizado”.	Sí
1.3. Enviar respuestas para una encuesta de una encuesta cerrada.	No se deben insertar las respuestas y se debe notificar con un mensaje de error.	No se insertan las respuestas y se notifica con un código de error y un mensaje: “La encuesta no está abierta”.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.4. Enviar correctamente las respuestas de una encuesta abierta.	El proceso debe ser exitoso. Las respuestas se deben almacenar en la base de datos.	Las respuestas se persisten en la base de datos de manera correcta. Se obtiene un código de respuesta exitoso.	Sí
1.5. Enviar un formulario sin respuestas seleccionadas.	El proceso debe ser exitoso. No se deben insertar respuestas en la base de datos porque la lista es vacía.	Se obtiene un código de respuesta exitoso. No hay nuevas respuestas en la base de datos.	Sí

6.6. Sprint 5: Consultar encuestas docente

En este quinto sprint, el cual marca el inicio de las funcionalidades relacionadas con los docentes, el objetivo principal es generar una lista que contenga el conjunto de encuestas asociadas a cada situación docente a la cual el pertenece el docente actualmente autenticado en el sistema.

Al momento de iniciar sesión, el docente es redirigido a su índice principal, donde se encuentra con dos botones: "Consultar encuestas" y "Consultar informes". Cada uno de estos botones inicia una funcionalidad de SiREnO. El primer botón, denominado "Consultar encuestas", tiene como objetivo listar las encuestas asociadas a las campañas válidas del docente.

Es importante recordar que una situación docente la componen, entre otros, el grupo, la asignatura, el grado, el docente... En el ámbito universitario, es posible que dos estudiantes se encuentren simultáneamente en el mismo aula, en la misma asignatura y con el mismo docente, pero pertenezcan a grados y/o cursos diferentes. Por ejemplo, la asignatura de robótica puede ser una asignatura troncal en tercer curso de ingeniería electrónica, pero también puede ser una asignatura optativa de cuarto curso para los estudiantes de ingeniería informática. Esto implica que, en un mismo momento y aula, pueden coincidir estudiantes de tercer curso de ingeniería electrónica y estudiantes de cuarto curso de ingeniería informática, es decir, estudiantes con diferentes situaciones docentes.

Tomando en consideración este factor, al mostrar las encuestas asociadas a cada situación docente, se desea agrupar a los estudiantes que se encuentran físicamente juntos en el aula, incluso si pertenecen a situaciones docentes distintas. Ya que, desde la perspectiva del docente, es "casi invisible" que sus estudiantes sean de grados o cursos diferentes. Para el docente, todos ellos conforman un único grupo de estudiantes y eso es suficiente.

Por último, recalcar que este sprint se enfoca exclusivamente en recuperar el conjunto de encuestas asociadas a las campañas válidas actuales del docente y mostrarlas. Abrir la encuesta con un plazo predefinido, junto con el proceso de cierre de la misma, se implementará más adelante, en el próximo sprint.

6.6.1. Análisis y diseño

El desarrollo de este sprint sigue un enfoque similar al del sprint 2, donde se presentaba el conjunto de encuestas para el alumno autenticado. El proceso de desarrollo se comenzará a implementar en el back-end. El objetivo final es mostrar en el front-end el conjunto de encuestas válidas para el docente. La obtención de la lista en el front-end se realizará utilizando la API, permitiendo que el back-end recupere las encuestas desde la base de datos, como se ha hecho anteriormente. Se busca lograr este proceso sin necesidad de requerir ningún parámetro adicional, aparte del token incluido en la cabecera de la solicitud.

Además, como se ha comentado, una situación docente puede estar agrupado con otras situaciones docentes, de este modo:

cod_situacion_docente	agrupada_con
sd001	
sd004	sd001
sd005	sd001

6.1. Tabla: Ejemplo de un conjunto de situaciones docentes agrupadas.

En este ejemplo concreto, se pretenden agrupar las tres situaciones docentes en una única entidad, dado que para el docente este grupo de tres situaciones docentes se percibe como una sola y, por consiguiente, se espera que abra la encuesta de manera simultánea para las tres.

Las encuestas deben incluir un registro de las veces que ha sido abiertas dentro del plazo de duración de su campaña, establecido por la administración. Es importante almacenar el recuento total de respuestas recibidas del cuestionario asociado en todas las ocasiones en que la encuesta ha sido abierta. Por lo tanto, la estructura de la encuesta debe incluir esta información.

```
{
    ‘‘cod_situacion_docente’’: ‘‘sd001’’,
    ‘‘n_alum_total’’: 35,
    ‘‘n_alum_respondido’’: 10,
    ‘‘veces_activada’’: 1,
    ‘‘nombre_asignatura’’: ‘‘Derecho Civil’’,
    ‘‘num_curso’’: 2,
    ‘‘año_curso’’: ‘‘22/23’’,
    ‘‘fecha_fin’’: ‘‘2023-10-17T13:00:00.000Z’’
    ‘‘agrupado_con’’: [‘‘sd004’’, ‘‘sd005’’]
}
```

Estructura que define una encuesta

Esta encuesta refleja la agrupación de las situaciones docentes de la tabla 6.1. En este ejemplo, la encuesta ha sido activada anteriormente en una ocasión, y durante dicha activación, solo 10 de los 35 alumnos esperados han respondido al formulario asociado.

El docente recupera sus encuestas al hacer clic en “Consultar Encuestas” en su menú principal. Se mostrará una pantalla con encuestas vinculadas a campañas válidas en las que participa como docente. El front-end solicitará a la API las situaciones docentes asociadas a encuestas válidas del docente actual. Posteriormente, se generarán componentes para cada encuesta, siguiendo el enfoque utilizado en el sprint 2 para los alumnos.

6.6.1.1. Sprint backlog

Las tareas que se deben llevar a cabo para la implementación del sprint son las siguientes:

Back-end:

- Implementación de un nuevo módulo en la capa de coordinación destinado a gestionar las rutas relacionadas con los docentes.
- Definición de las rutas pertinentes para obtener el conjunto de encuestas válidas del docente.
- Definición de los métodos necesarios en "dbQuerys.js'.

Front-end:

- Creación y desarrollo de los componentes de las encuestas.
- Configuración de un servicio que consuma la API del back-end para obtener el conjunto de encuestas válidas del docente.
- Creación dinámica de los componentes de las encuestas según los datos recuperados del back-end.
- Gestión de la validez de las encuesta.

6.6.2. Implementación

Back-end:

El primer paso es crear un nuevo módulo en la capa de coordinación que haga referencia a las rutas respectivas de los docentes. A su vez, en la capa de comunicación hay que incluir este nuevo módulo y asignarle una ruta. La ruta asignada es *"/docente"*, es decir, todas las URL que comiencen con este preámbulo, serán redirigidas al módulo de coordinación del docente.

El siguiente paso consiste en crear e implementar la nueva ruta *"/docente/getEncuestas"*. Esta ruta será publicada por la API y se utilizará para obtener el conjunto de encuestas pertenecientes a situaciones válidas para el docente que realiza la solicitud. Tras recibir la solicitud en el back-end, éste llama al método *"getEncuestasValidasDocente()"* de la capa de datos. A este método se le debe pasar el identificador del docente extraído del token de la cabecera que realiza la solicitud. Finalmente, tras ejecutarlo, se obtiene el conjunto ordenado de encuestas asociadas a situaciones válidas del docente actual. Las encuestas del conjunto se ordenarán de forma ascendente según el número de veces que hayan sido abiertas, priorizando aquellas que hayan sido abiertas menos veces. En caso de empate, se mostrarán primero aquellas que tengan una fecha de expiración próxima y ordenándolas de forma ascendente según dicha fecha.

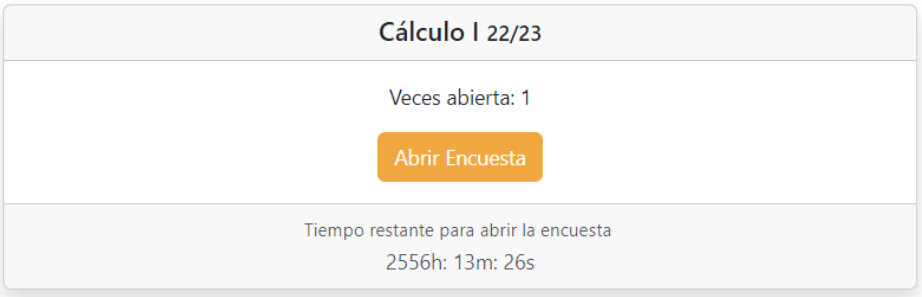
Front-end:

El primer paso es crear el componente que reflejará visualmente cada encuesta de la lista de encuestas obtenida desde el back-end. Estos componentes se insertarán de manera dinámica en la pantalla de visualización de encuestas, uno por cada encuesta presente en la lista.

Tras haber creado el componente, es necesario implementar, como siempre, la llamada a la API con el fin de obtener la lista de encuestas. Después de obtener el JSON que incluye todas las encuestas válidas del docente, se llevará a cabo un recorrido por dicho conjunto y se creará un componente individual para cada encuesta. Además, se asignarán los atributos correspondientes del objeto "encuesta" utilizando la información obtenida del back-end.

Para cada componente, es necesario calcular el tiempo restante hasta su fin, es decir, calcular el tiempo restante hasta que la encuesta deje de ser válida, es decir, que la campaña asociada expire. Al crear el componente, se realiza un cálculo para determinar la diferencia entre la "fecha fin" de la campaña asociada y la fecha actual. Este tiempo se formatea en "hh:mm:ss" y se muestra en el componente. De nuevo, igual que en del segundo sprint, el tiempo restante no se actualiza a tiempo real en el navegador, ya que se calcula solo una vez, al crear el componente.

Para ello, cada segundo, se ejecuta la función *"getTiempoCierre()"* obteniendo el tiempo restante de cada encuesta. En la vista del componente se muestran los valores del observable. Cuando el tiempo restante de la validez campaña asociado a la encuesta llega a cero, no tiene sentido seguir mostrando la encuesta. Por lo tanto, cuando la campaña asociada a una encuesta expira, se oculta dicha encuesta dejando de ser visible para el docente. La Figura 6.8 muestra el aspecto visual de un componente correspondiente a una encuesta en la vista del docente.



6.8. Figura: Componente respectivo a una encuesta de un docente.

6.6.3. Revisión del sprint

Una vez finalizado el sprint, se lleva a cabo la reunión correspondiente en la que se presenta al cliente el trabajo completado durante el proceso de desarrollo. Tras una exhaustiva revisión de las funcionalidades implementadas, se constató que el desarrollo era favorable a los requisitos. No obstante, surgieron algunas necesidades adicionales que requieren atención. Estas necesidades han surgido como resultado de una combinación de la falta de una especificación detallada de los requisitos funcionales en el presente sprint, junto con un análisis y diseño que no han sido suficientemente exhaustivos.

Las expansiones solicitadas fueron las siguientes:

- El componente visual respectivo a una encuesta debe mostrar la relación entre el total de alumnos que deben responder a la encuesta asociada y el número de alumnos que la han respondido hasta el momento.
- Es necesario distinguir entre las encuestas cerradas y las encuestas abiertas. Asimismo, se debe crear un componente específico para las encuestas abiertas del docente, el cual incluirá, entre otros, el tiempo restante hasta la finalización de la apertura. Este tiempo restante hasta la finalización de la apertura sustituirá al tiempo restante hasta la finalización de la encuesta que se muestra en el componente de una encuesta cerrada.
- Es necesario sustituir la expresión "alumnos respondidos" por otra que sea neutral en cuanto al género, como por ejemplo, "respuestas recibidas".

Para llevar a cabo la primera ampliación, se requiere realizar una modificación en la consulta a la base de datos que obtiene el conjunto de encuestas del docente. En este sentido, es necesario solicitar los atributos "n_alum_total" y "n_alum_respondido" como parte de la consulta.

Esta aproximación funciona adecuadamente para las situaciones docentes convencionales; sin embargo, surgen dificultades cuando se trata de situaciones docentes agrupadas como las de la tabla 6.1. En el caso de obtener un conjunto de situaciones docentes agrupadas, los valores correspondientes a "n_alum_total" y "n_alum_respondido" se refieren únicamente a la situación docente principal, sin considerar los alumnos totales y los alumnos que han respondido dentro del conjunto de situaciones docentes agrupadas.

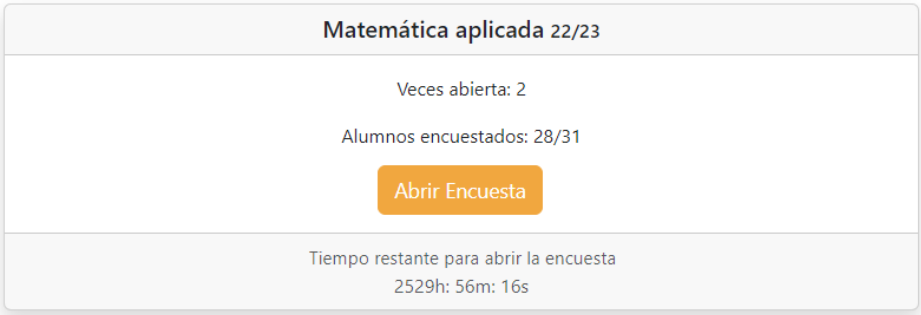
Para abordar esta problemática, es necesario incluir el número de alumnos totales y respondidos del conjunto de situaciones docentes agrupadas. Para lograrlo, una vez recuperado el conjunto de situaciones docentes agrupadas, se procede a iterar individualmente por cada una de ellas, añadiendo los valores de "n_alum_total" y "n_alum_respondido" a las situaciones docentes agrupadas correspondientes.

Finalmente, al generar cada componente de una encuesta desde el front-end, se realiza un recorrido por todas las situaciones docentes agrupadas y se suman los alumnos totales y respondidos respectivamente:

```
if (encuesta.agrupado_con && encuesta.agrupado_con.length > 0) {
     let nAlumTotalSum = encuesta.n_alum_total;
     let nAlumRespondidoSum = encuesta.n_alum_respondido;
     encuesta.agrupado_con.forEach((agrupado: any) => {
          nAlumTotalSum += agrupado.n_alum_total;
          nAlumRespondidoSum += agrupado.n_alum_respondido;
     });
     [...]
}
```

Segmento de código que contabiliza el total de respuestas.

La apariencia visual resultante del componente que hace referencia a las encuestas del docente después de la modificación se muestra en la Figura 6.9.



6.9. Figura: Componente respectivo a una encuesta de un docente.

Con respecto a la segunda ampliación, es necesario diferenciar entre las encuestas cerradas y las encuestas abiertas del docente. Por tanto, se debe desarrollar un componente que haga referencia a una encuesta abierta.

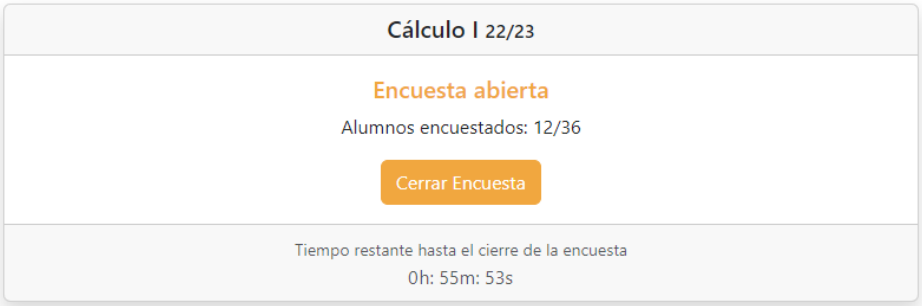
Para lograr esto, es necesario realizar una modificación adicional en la consulta a la base de datos para verificar si encuesta se encuentra actualmente abierta. En caso de estar abierta, se deberá obtener la fecha de finalización de la apertura. Si la encuesta no está actualmente abierta, se retornará una fecha de finalización de activación con un valor nulo, indicando esta condición.

Los criterios de ordenación de las encuestas han sido modificados. En primer lugar, se dará prioridad a las encuestas abiertas. Dentro de las encuestas abiertas, se realizará una ordenación ascendente basada en la fecha y hora de cierre. Mostrando primero aquellas con una fecha de cierre más próxima. En caso de empate en la fecha y hora de cierre, se considerará el número de veces que han sido abiertas, mostrando primero aquellas que hayan sido abiertas menos veces.

Por otro lado, para las encuestas cerradas, es decir, aquellas que no cuenten con una fecha y hora de cierre definida, se aplicarán dos criterios de ordenación. En primer lugar,

se ordenarán de forma ascendente según el número de veces que han sido abiertas, priorizando aquellas que hayan sido abiertas menos veces. En caso de empate en el número de veces abiertas, se considerará la fecha de finalización de la campaña asociada a la encuesta, mostrando primero aquellas que tengan una fecha de expiración próxima y ordenándolas de forma ascendente según dicha fecha.

El componente correspondiente a una encuesta abierta es muy similar al componente existente para una encuesta cerrada (Figura 6.9). La distinción radica en que, en el caso de una encuesta abierta, se indica el hecho de que está actualmente abierta y se muestra el tiempo restante hasta la el cierre en lugar de la fecha de expiración de la campaña asociada. El aspecto visual del componente de una encuesta abierta se muestra en la Figura 6.10.



6.10. Figura: Componente respectivo a una encuesta abierta de un docente.

Finalmente, es necesario gestionar el momento en el que una encuesta deja de estar abierta, es decir, termina el plazo de apertura. En este caso, el componente de la encuesta debe ser transformado en un componente asociado a una encuesta cerrada (Figura 6.9). Es importante resaltar que la lógica subyacente para abrir y cerrar encuestas se implementará en el próximo sprint. La implementación actual se limita únicamente al aspecto visual del listado de encuestas.

Para lograr esto, sabiendo que los componentes se encuentran integrados en la vista que muestra el listado de encuestas, si un componente refresca la vista en la que es contenido, se genera nuevamente la pantalla que incluye todas los componentes de las encuestas. Este refresco implica realizar una llamada a la base de datos para volver a obtener la información de actualizada de todas las encuestas en tiempo real. Por tanto, cuando una encuesta se cierra, refresca la vista, haciendo que se vuelva a obtener el listado de encuestas actualizadas desde el back-end, una vez obtenido se crea de nuevo la vista actualizada.

6.6.4. Pruebas

El propósito de estas pruebas es asegurar la correcta recuperación de encuestas y grupos de encuestas por parte de cada docente, abordando escenarios críticos para verificar su funcionamiento adecuado.

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.1. El docente no tiene encuestas asociadas a campañas válidas.	Debe mostrarse un mensaje que indique la situación.	No se muestran encuestas para el docente actual. Solo el mensaje.	Sí
1.2 El docente tiene encuestas asociadas a campañas válidas.	Las encuestas se deben mostrar ordenadas por veces abiertas y, en caso de empate, por fecha próxima a su expiración.	Las encuestas se ordenan correctamente siguiendo ambas condiciones	Sí
1.3. El docente tiene encuestas respectivas a situaciones sin agrupar.	Cada encuesta se debería mostrar como un componente independiente.	Se muestran tantos componentes independientes como situaciones sin agrupar tiene el docente.	Sí
1.4. El docente tiene encuestas respectivas a situaciones agrupadas.	Cada grupo de encuestas se debería mostrar como un componente que agrupe el conjunto.	Se muestra una sola encuesta que hace referencia a todas las situaciones agrupadas.	Sí
1.5. El docente tiene tanto encuestas asociadas a situaciones sin agrupar como agrupadas.	Las situaciones no agrupadas se deberían mostrar cada una como una encuesta independiente y Las agrupadas deben mostrar una sola encuesta por agrupación.	Ambas se muestran de como un componente individual. Las agrupadas hacen referencia al conjunto de situaciones.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
2.1. El docente tiene encuestas abiertas y cerradas.	Las abiertas se deberían mostrar arriba. Todas deben estar ordenadas, cada una por su criterio de ordenación.	Al principio se listan las encuestas abiertas seguidas de las cerradas, ordenadas según el criterio	Sí
2.2 Una encuesta abierta llega a su fin de apertura.	Cuando el contador llega a 00:00:00 se debería refrescar la vista y la encuesta se debería mostrar como cerrada.	La vista se refresca y la encuesta abierta pasa a estar cerrada.	Sí
3. Un usuario no docente trata de ver las encuestas de un docente.	No se debería permitir el acceso a dicha página.	Acceso no autorizado y redirección al login.	Sí
4. Mostrar encuestas de campañas no válidas (expiradas).	Las encuestas de campañas cuyo plazo ha terminado no se deberían mostrar.	Las encuestas de campañas expiradas no se le muestran al docente.	Sí
4.1. La encuesta deja de ser visible al expirar la campaña asociada.	Cuando termine el plazo de la campaña asociado a la encuesta actual definido por la administración, la propia encuesta debe dejar de ser visible para el docente.	Cuando la campaña asociada termina, es decir, cuando el contador que indica el tiempo restante para abrir la encuesta alcanza 00:00:00, la encuesta se elimina automáticamente de la pantalla de encuestas listadas para el docente.	Sí

6.7. Sprint 6: Abrir, programar y cerrar encuestas

El objetivo principal del sexto sprint es desarrollar las funcionalidades del sistema que le permiten al docente abrir, programar y cerrar una encuesta, brindándole a los alumnos asociados la posibilidad de responderla.

El primer paso consiste en implementar la funcionalidad que permita al docente abrir una encuesta concreta estableciendo un plazo finito de tiempo para la apertura. Este período de tiempo limitado, se encuentra dentro del intervalo de validez de la campaña asociada a la encuesta. El periodo de apertura de la encuesta es el tiempo durante el cual los alumnos participantes en la situación docente podrán responder al formulario de opinión asociado. En caso de que el docente quiera cerrar la encuesta antes de que termine el plazo inicialmente establecido, existirá una opción para poder hacerlo.

En el momento en que el docente seleccione la opción "Abrir Encuesta" para una encuesta específica en el menú que muestra sus encuestas vigentes, se le solicita indicar una fecha de finalización para la apertura que se va a realizar. Tras la confirmación por parte del docente, la encuesta en cuestión será abierta. En consecuencia, cuando un alumno, que forme parte de la situación docente de la encuesta recién abierta, acceda a su menú principal, se le mostrará dicha encuesta como abierta. Como resultado del nuevo estado de apertura, el alumno podrá responder al formulario asociado.

En el caso de que el docente tome la decisión de cerrar una encuesta de manera manual antes del plazo de finalización previamente establecido, se procederá a cerrar la encuesta correspondiente. Cuando una encuesta se cierre, de manera manual o automática, el sistema actuará y expulsará al alumno que se encuentre en el proceso de respuesta del formulario, sin requerir una confirmación adicional.

Adicionalmente, es necesario considerar varias restricciones de seguridad. Al abrir una encuesta, se realizarán verificaciones para asegurarse de que no esté abierta previamente y que la campaña asociada no haya expirado, entre otras condiciones. Al programar el plazo de la encuesta, entre otros factores, se verificará que éste no exceda la fecha validez de la campaña y que la fecha introducida tampoco sea anterior al momento actual. Finalmente, al cerrar la encuesta, se gestionará la expulsión de los alumnos que estén respondiendo al formulario asociado. Asimismo, si la encuesta está cerrada pero el listado de encuestas abiertas del índice del alumno no ha sido actualizado, se impedirá abrir el formulario asociado al pulsar el botón "Responder".

6.7.1. Análisis y diseño

El proceso de apertura y establecimiento de un plazo comienza en la interfaz de usuario (front-end). Desde el menú que lista las encuestas del docente autenticado, cuando éste seleccione el botón "Abrir Encuesta" (ver Figura 6.9), está indicando su intención de abrir un nuevo plazo para que los alumnos que pertenezcan a esa situación docente puedan responder al formulario de la encuesta correspondiente.

Como resultado de que el docente presione el botón "Abrir Encuesta", se mostrará un diálogo emergente que solicitará ingresar una fecha límite para la apertura de la encuesta. Después de validar la fecha ingresada, cuando el docente confirme su selección, se abrirá la encuesta hasta la fecha indicada. Finalmente, tras abrir la encuesta y establecer el plazo correspondiente, también es necesario actualizar el contador de veces que la encuesta ha sido abierta. Finalmente, cuando un alumno responda al formulario de una encuesta, el contador de respuestas deberá actualizarse sin necesidad de refrescar la página.

En cuanto al cierre de la encuesta, existen dos escenarios a considerar. Si se alcanza la fecha de finalización establecida en la activación de la encuesta, ésta se cerrará de manera automática. Sin embargo, si el docente quisiera cerrar el plazo antes de su finalización ordinaria, se le proporcionará la opción de hacerlo. Para ello, en el componente referente a una encuesta abierta del docente, el botón "Abrir Encuesta" se reemplazará por un botón similar llamado "Cerrar Encuesta".

Si el docente selecciona el botón de cierre, se abrirá un diálogo emergente solicitando confirmación adicional por seguridad. Tras la confirmación, la encuesta se cerrará y los alumnos que respondían al formulario serán expulsados sin guardar sus respuestas. La encuesta no desaparecerá inmediatamente del índice de alumnos, pero al actualizar la página, deberá desaparecer. Aunque al presionar el botón "Responder" de una encuesta cerrada, no se permitirá acceder al formulario gracias a una medida de seguridad adicional, ya que, para generar el componente del formulario al pulsar el botón de "Responder", se debe obtener nuevamente la fecha de finalización de la apertura y hacer una comprobación.

6.7.1.1. Sprint backlog

Los pasos necesarios para ejecutar el sprint incluyen:

Back-end:

- Establecimiento de las rutas necesarias para obtener el conjunto de encuestas válidas del docente.
- Definición e implementación de los métodos requeridos en "dbQuerys.js" para abrir y cerrar las encuestas.
- Implementación de las verificaciones y medidas de seguridad correspondientes en el back-end.
    - Verificación de la validez de la campaña asociada a la encuesta antes de su activación.
    - Gestión de la validez de las encuestas abiertas y cerradas, incluyendo acciones como marcar la encuesta como visible al abrir y expulsar a los alumnos al cerrar.
    - Impedir el acceso a la encuesta cerrada al presionar el botón "Responder Encuesta", aunque la encuesta no deaparezca hasta recargar la página.

Front-end:

- Creación y desarrollo de los diferentes tipos de diálogos emergentes.
- Configuración de un servicio para consumir la API del back-end y obtener el conjunto de encuestas válidas del docente.
- Implementación de las verificaciones y medidas de seguridad pertinentes en el front-end.
    - Asegurarse de que se abran correctamente todas encuestas asociadas a las situaciones docentes agrupadas.
    - Permitir la apertura de dos encuestas en serie y gestionar la visibilidad y recarga de los componentes para que la previamente abierta se muestre correctamente.
    - Actualizar automáticamente el contador del componente por cada alumno que responda a una encuesta de la encuesta activada.

6.7.2. Implementación

Back-end:

En primer lugar, es necesario habilitar nuevas rutas en el módulo de coordinación del docente. Estas rutas adicionales en la API serán responsables de recibir las llamadas externas y responder con la información solicitada. La primera ruta que se habilitará es *"/docente/abrirEncuesta"*, la cual recibirá como parámetros el conjunto de situaciones docentes asociadas a la encuesta que se desea abrir y la fecha de finalización de la apertura. Se pasa el conjunto de situaciones, ya que, recordemos que una situación docente puede estar agrupada con otras y a la hora de abrir la encuesta de la situación principal se deben también abrir las encuestas del grupo de situaciones asociadas. El método asociado a la ruta se encargará de crear una nueva apertura para la encuesta.

Antes de abrir la encuesta, se lleva a cabo una verificación a nivel de back-end para determinar si la campaña asociada es válida; esto implica verificar si la campaña está actualmente vigente y no ha expirado. Se debe también confirmar si la encuesta no está actualmente abierta, ya que sería redundante abrirla por segunda vez. Si ambas condiciones se cumplen, se procede a abrir la encuesta. Para lograr esto, se realiza una nueva llamada a la capa de datos con el propósito de agregar una nueva entrada en la tabla de apertura de las encuestas. Antes de abrirse, se comprueba que dicha fecha de finalización se encuentre entre la fecha actual y la fecha de fin de la campaña asociada. Además, es necesario incrementar en 1 el contador de veces que se ha abierto la encuesta recién abierta.

Para cerrar una encuesta, se sigue un proceso similar. Se ha implementado una segunda ruta denominada *'/docente/cerrarEncuesta"* encargada de llevar a cabo esta operación. Esta ruta también recibe un conjunto de situaciones docentes, ya que se debe contemplar la posibilidad de tener varias situaciones agrupadas. En primer lugar, se verifica si la

encuesta está ya abierta. No es necesario comprobar si es válida, ya que se realiza dicha validación al abrir la encuesta. Por lo tanto, si una encuesta está abierta, implica que también es válida. En caso de estar abierta, se procede a cerrar la encuesta. Para lograrlo, se realiza una llamada al método *"cerrarEncuesta()"* de la capa de datos pasándole los parámetros necesarios obtenidos desde el front-end. Este método se encarga de modificar la fecha de finalización de la apertura establecida al inicio, utilizando la fecha actual como la nueva fecha del fin de apertura.

Front-end:

En el front-end, el proceso se inicia cuando el docente hace clic en el botón "Abrir Encuesta" de una de sus encuestas. Al hacer clic en dicho botón, se muestra un diálogo emergente que permite al docente seleccionar la fecha de finalización de la apertura. Una vez que se ha elegido la fecha, se realiza una validación en el servidor, y si todo es correcto, la encuesta se abre. Como resultado, es necesario actualizar completamente la página para obtener nuevamente, desde el back-end, el listado actualizado de encuestas del docente, con el fin de actualizar los componentes asociados.

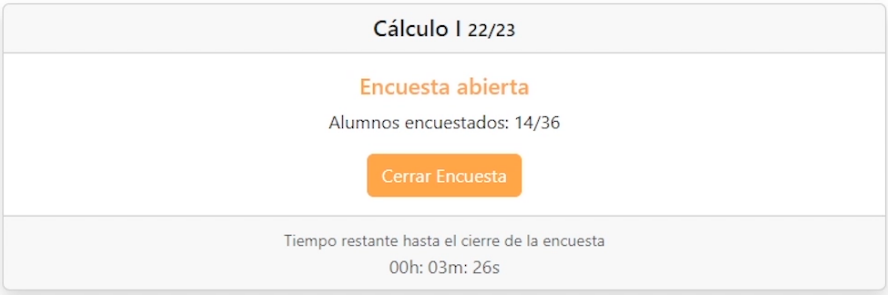
El componente correspondiente a la encuesta abierta se muestra en la Figura 6.10. Este componente mantendrá su apariencia hasta que la fecha de apertura expire, es decir, cuando el contador llegue a 00:00:00. En ese momento, la página que contiene el listado de encuestas del docente se actualizará automáticamente. Como resultado de la expiración de la fecha de apertura, la encuesta se clasificará nuevamente como cerrada (ver Figura 6.9) y se mostrará como un componente de encuesta cerrada pero se seguirá mostrando hasta que la campaña asociada expire. Además, se recogerá el valor actualizado de las veces que ha sido abierta teniendo en cuenta esta última apertura.

Mientras la encuesta esté abierta, los alumnos pueden responder al formulario asociado. Cuando un alumno responda, se debe actualizar de manera automática el contador de alumnos respondidos en la vista de la encuesta del docente (Figura 6.11). Para ello, desde el front-end y cada segundo, se realiza una solicitud al back-end para recuperar el número de respuestas recibidas para la situación docente asociada a la encuesta actual:

```
interval(1000).subscribe(() => {
    authService.getRespondidos(s_doc).subscribe((total_resp:any) => {
        if(this.n_alum_resp){
            if (total_resp['suma'] > this.n_alum_resp){
                this.n_alum_resp = total_respondidos['suma'];
                this.parpadeo = true;
                setTimeout(() => {
                    this.parpadeo = false;
                }, 1000);
            }
        }
    });
});
```

Segmento de código que actualiza cada segundo el valor de num_alum_resp.

Este código se encarga de realizar tareas repetitivas, obtener información y actualizar variables y efectos visuales, para ello, utiliza el operador *interval* emitiendo valores cada 1 segundo. Verifica las situaciones y obtiene la cantidad de alumnos respondidos. Si el total es mayor que "n_alum_respondido", actualiza la variable correspondiente y activa/-desactiva un efecto de parpadeo. Utilizando un visor compatible con formatos GIF, como *Adobe Acrobat Reader*, es posible visualizar la animación referente a un nuevo registro de respuesta mostrado en formato GIF en la Figura 6.11.



6.11. Figura: Animación cuando se registra una nueva respuesta.

Antes de que la apertura expire automáticamente, el docente tiene la opción de cerrar manualmente la encuesta. Para ello, en el componente de una encuesta abierta, se encuentra un botón llamado "Cerrar Encuesta" (ver Figura 6.11) que tiene la función de finalizar inmediatamente la apertura de la encuesta correspondiente. Después de hacer clic en el botón "Aceptar" del diálogo de confirmación, la encuesta pasa a estar cerrada. Como resultado del cierre, se debe recargar automáticamente el listado de encuestas de la página principal del docente y la encuesta que acaba de cerrarse se mostrará cerrada siempre y cuando la campaña asociada cumpla la condición de validez establecida.

Cuando se cierra una encuesta, es importante que deje de ser visible para los alumnos asociados a la situación docente correspondiente. Este aspecto se abordó durante la creación del listado de encuestas para los alumnos en el sprint 1. Cuando una encuesta válida se abre y posteriormente se cierra, ningún alumno debería poder verla como pendiente en su índice principal. Al cerrar la encuesta, si un alumno refresca su índice de encuestas disponibles, la encuesta cerrada ya no se mostrará más (a no ser que sea abierta de nuevo). Gracias a la correcta implementación de esta característica en el sprint 1, se logra que la encuesta desaparezca del índice de encuestas de los alumnos una vez que el docente la cierra, siempre y cuando el alumno refresque la vista.

Por último, es necesario gestionar qué sucede si el docente cierra una encuesta antes de que termine su el plazo ordinario de apertura inicial mientras un alumno está respondiendo al formulario asociado. En ese caso, el alumno debe ser expulsado de la encuesta en curso, de manera similar a si el plazo ordinario hubiera expirado. Para lograr esto, se ha implementado un *listener* que verifica cada segundo si la encuesta asociada al formulario actual está abierta. Cuando el docente cierra la encuesta, este *listener* detectará que la encuesta ya no está abierta y redirigirá al alumno a su índice principal sin previo aviso.

Después de la redirección, se notificará al alumno sobre la situación extraordinaria que acaba de ocurrir mediante un diálogo emergente.

6.7.3. Revisión del sprint

Tras finalizar el sprint, se lleva a cabo una reunión donde se le muestra al cliente el trabajo completado durante el período de desarrollo. El propósito principal de la revisión del sprint es obtener retroalimentación y validar si se han cumplido los objetivos establecidos del mismo. Durante esta reunión, se presentaron las funcionalidades implementadas, las cuales fueron evaluadas como correctas y satisfactorias por el cliente.

En la revisión, no se identificaron cambios por parte del cliente. No se plantearon ajustes estéticos ni modificaciones en las funcionalidades existentes. La implementación se consideró correcta y no se identificaron necesidades adicionales. Esto indica que el desarrollo se ha llevado a cabo de manera satisfactoria y se ha logrado cumplir con los objetivos establecidos para el sprint en cuestión Todo ello gracias a una correcta captura de requisitos junto con una excelente ejecución del analisis y diseño del sprint.

6.7.4. Pruebas

El objetivo de estas pruebas es garantizar la adecuada apertura, programación y posterior cierre de las encuestas, abordando situaciones críticas para comprobar su correcto funcionamiento.

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.1. Apertura de encuesta de **campaña no válida**.	El back-end devuelve un código de error, la encuesta no debe abrirse y un diálogo indica la situación desfavorable.	La encuesta no se abre puesto que no es valida, se recibe el error correspondiente y salta el diálogo.	Sí
1.2. Apertura de encuesta **abierta**.	El back-end devuelve un código de error, la encuesta no debe abrirse y un diálogo indica la situación desfavorable.	La encuesta no se abre puesto que ya está abierta, se recibe el error correspondiente y salta el diálogo.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.3 Apertura de encuesta de campaña válida con fecha inicial anterior a la actual.	El back-end devuelve un código de error, la encuesta no debe abrirse y un diálogo indica la situación desfavorable.	La encuesta no se abre puesto que las fechas son erróneas, se recibe el error correspondiente y salta el diálogo.	Sí
1.4. Apertura de encuesta de campaña válida con fecha fin posterior al plazo de validación.	El back-end devuelve un código de error, la encuesta no debe abrirse y un diálogo indica la situación desfavorable.	La encuesta no se abre puesto que las fechas son erróneas, se recibe el error correspondiente y salta el diálogo.	Sí
1.5. Apertura de encuesta de campaña válida con plazos dentro del margen de duración de la campaña asociada.	La encuesta se abre de manera satisfactoria y un diálogo indica la situación favorable.	La encuesta se abre, se refresca la vista, se muestra el componente de encuesta abre y salta el diálogo.	Sí
1.6. Incremento de veces abierta de la situación docente.	El contador de veces abierta de la encuesta recién abierta debe incrementarse en 1.	Se suma 1 al contador de veces abierta y se refleja en la vista del componente.	Sí
2. Incremento automático del contador de alumnos que han respondido.	Cuando un alumno envía las respuestas de la encuesta asociada a la situación docente abierta, se debe refrescar automáticamente a tiempo real el numero de respuestas recibidas.	El contador de respuestas se refresca de manera automática a tiempo real y cada vez que un alumno responde se ejecuta una animación en el contador.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
3. Apertura de una segunda encuesta mientras la primera sigue abierta.	Se deben poder abrir múltiples encuestas a la vez y poder visualizar y gestionar todos los componentes de manera independiente.	Se pueden abrir varias encuestas a la vez, sin conflictos en la vista ni en la lógica y la gestión es independiente y correcta.	Sí
4. Confirmación requerida al cerrar manualmente una encuesta.	Antes de cerrar la encuesta debe saltar un diálogo de confirmación.	Salta un diálogo de confirmación y hasta no aceptarlo no se cierra la encuesta.	Sí
4.1. Cierre de una encuesta cerrada.	El back-end devuelve un código de error, la encuesta no debe cerrarse ya que no está abierta.	La encuesta no se cierra puesto que no hay una apertura en curso. Se devuelve un error.	Sí
4.2. Cierre de una encuesta abierta.	La encuesta se cierra de manera satisfactoria y un diálogo indica la situación favorable.	La encuesta se cierra, se refresca la vista, se muestra el componente de encuesta cerrada (si sigue siendo valida) y salta el diálogo.	Sí
5. Expulsión del alumno de un formulario abierto cuando el docente cierra la encuesta asociada.	El alumno debe ser expulsado del formulario sin previo aviso debido al cierre repentino de la encuesta. Una vez expulsado se debe notificar la situación extraordinaria mediante un diálogo emergente.	El alumno es inmediatamente expulsado del formulario. El formulario no se envía. Una vez redireccionado, se muestra el diálogo explicando la situación.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
6. La encuesta recién cerrada se elimina de la lista de encuestas del alumno al actualizar el índice principal del alumno.	La encuesta debería desaparecer como efecto de haber sido cerrada.	Una vez se cierra la encuesta, al refrescar el índice de alumnos, no se muestra más a no ser que se vuelva a abrir.	Sí
7. Intento de responder al formulario de una encuesta cerrada sin actualizar el índice.	No debe dejar entrar al formulario y tras pulsar el botón "Responder" de la encuesta la página se autorrefresca eliminando así el componente de la encuesta recién cerrada.	Debido a que se comprueba si la encuesta estña abierta antes de mostrar el formulario, se bloquea sua acceso y se refresca el índice de encuestas de alumno, eliminándola así de la lista encuestas del alumno.	Sí

6.8. Sprint 7: Generación y presentación de informes

El desarrollo de este sprint, tiene como objetivo la generación y presentación de informes para los docentes. Los informes son una herramienta esencial para los docentes, ya que les permite obtener valiosas perspectivas basadas en los resultados de las encuestas completadas por los alumnos en sus asignaturas. Estos informes se dividen en dos tipos: informes personales y comparativos.

Los informes personales proporcionan a los docentes la posibilidad de consultar la media obtenida pata cada pregunta de una asignatura en particular de cualquiera de los años académicos en los que la hayan impartido. Además, se les brinda la opción de hacer clic en una pregunta específica y visualizar una gráfica que muestra la evolución de la media obtenida de dicha pregunta a lo largo de los años en esa misma asignatura. Estos informes personalizados les permiten realizar un seguimiento detallado del progreso de sus alumnos y analizar los cambios en las respuestas a lo largo del tiempo.

Por otro lado, **los informes comparativos** son una poderosa herramienta que permite a los docentes comparar la nota media de cada pregunta de una de sus asignaturas respectiva a cierto año académico, en relación a la media del centro, grupo, titulación, etc. al que pertenece la situación docente del informe. Esto proporciona una visión más amplia y contextualizada de los resultados, lo que ayuda a los docentes a identificar tendencias, detectar áreas de mejora y evaluar el rendimiento de sus alumnos en comparación con un entorno académico más amplio.

Esta funcionalidad de generación y presentación de informes ofrece a los docentes una visión profunda y detallada de los resultados de las encuestas de sus alumnos. Tanto los informes personalizados y comparativos brindan la información necesaria para tomar decisiones informadas sobre su enseñanza, identificar oportunidades de mejora y seguir mejorando la experiencia educativa en sus asignaturas.

6.8.1. Análisis y diseño

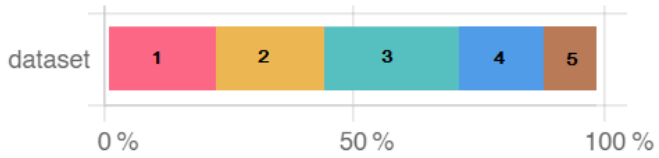
Una vez autenticado, como se ha explicado en el anterior sprint, el docente accede a su interfaz principal, donde se visualizan dos botones distintos, cada uno asociado a una funcionalidad específica: “Consultar encuestas” y “Consultar informes”. Al seleccionar el segundo botón, se redirige al docente a un nuevo índice de informes docentes, donde encontrarán dos botones adicionales: “Consultar informes personales” y “Comparativa de resultados”. Cada uno de estos botones proporciona acceso uno de los dos tipos de informes mencionados en la introducción al sprint. Queda claro, por tanto, que el proceso de generación y presentación de informes se inicia en la interfaz de usuario (front-end). Al hacer clic en alguno de los dos botones se abrirá el informe asociado a la espera de que el docente seleccione los parámetros necesarios para poder visualizar los resultados.

Al acceder al **informe personal**, se le presentará un menú desplegable que enumerará todas las asignaturas en las que ha participado como docente. Una vez seleccionada una asignatura en particular, se habilitará un segundo menú desplegable que permitirá

al docente elegir un año académico en el cual haya impartido la asignatura previamente seleccionada. Después de haber seleccionado ambas opciones en los menús desplegables, se activará un botón denominado "Mostrar informe". Como resultado de pulsar dicho botón se generará y mostrará en pantalla el informe seleccionado.

En la parte superior del informe, se mostrará un conjunto de datos de relevancia que proporcionan contexto al informe. Estos datos incluirán el nombre de la asignatura, el centro educativo, el grado académico, el grupo... Entre otros parámetros que describen la situación docente actual. Además, se incluirán datos adicionales de interés, como el número de plazas matriculadas en la asignatura o el número de cuestionarios totales rellenados. Estos datos adicionales ayudarán a contextualizar y brindar una visión más completa del informe presentado.

En la sección inferior del informe, se presenta el resultado del análisis realizado. Se enumeran las preguntas de la encuesta asociada a la situación docente seleccionada. Para cada pregunta, se muestra la respuesta media obtenida a partir de todos los formularios completados por los alumnos que forman parte de la situación docente en cuestión. Además de la media, se proporciona información adicional de forma visual mediante un diagrama de barras normalizado que muestra el recuento de puntuaciones obtenidas para cada pregunta. Es decir, se muestra cuántos alumnos han puntuado con un 1, 2, 3, 4 o 5 en cada pregunta, tanto en términos absolutos como en forma porcentual. Para ello, se quiere hacer una gráfica similar a la que se muestra en siguiente Figura 6.12.



6.12. Figura: Ejemplo de diagrama de barras normalizado.

Asimismo, se brinda la opción de hacer clic en una pregunta específica para visualizar la gráfica que muestra la evolución de la media obtenida para esa pregunta a lo largo de los años. Estos informes personalizados proporcionan al docente una visión detallada del progreso de las valoraciones en su asignatura a lo largo del tiempo, permitiéndole realizar un seguimiento minucioso y evaluar las tendencias en las respuestas a lo largo de los años.

Por otro lado, al acceder al **informe comparativo**, igual que en el otro informe, se presentará un menú desplegable que enumera las asignaturas en las que ha participado como docente. Una vez seleccionada una asignatura específica, se habilitará el menú desplegable para que el docente pueda elegir el año académico correspondiente.

Una vez seleccionadas tanto la asignatura como el año académico en los menús desplegables, se habilitará un tercer menú desplegable que permitirá al docente elegir con qué media desea realizar la comparación. El docente tendrá la opción de comparar la media de resultados de la asignatura y año académico seleccionados con la media del grupo de la asignatura seleccionada, la propia asignatura seleccionada, el centro educativo al que pertenece la asignatura, el departamento, u otras opciones relevantes.

Al seleccionar las opciones en los tres menús desplegables, se activará el botón "Mostrar informe". Al hacer clic en dicho botón, se generará y presentará en pantalla el informe comparativo correspondiente. Es importante resaltar que el informe comparativo permitirá al docente realizar una comparación precisa y específica de la media de resultados de la asignatura y año académico seleccionados en relación con la media elegida para la comparación. Esta herramienta brinda una visión analítica y detallada de cómo se sitúan los resultados de los alumnos en relación con otros parámetros relevantes, lo que facilita la toma de decisiones y la mejora continua de la enseñanza en la asignatura.

Además, se incluye una columna adicional que presenta un diagrama de barras para cada pregunta, permitiendo la comparación entre la media del docente y la media de referencia seleccionada. El diagrama de barras proporciona una representación visual clara y concisa de cómo se sitúa la media del docente en relación con la media de referencia. Esto permite al docente identificar visualmente y de manera mucho mas rápida las diferencias y similitudes en los resultados y evaluar su desempeño en comparación con los parámetros establecidos.

6.8.1.1. Sprint backlog

Los pasos necesarios para ejecutar el sprint incluyen:

Back-end:

- Establecimiento de las rutas necesarias para obtener el todos los datos necesarios para los informes.
- Definición de los métodos requeridos en "dbQuerys.js" para recuperar los datos respectivos a las respuestas de los formularios.

Front-end:

- Creación y desarrollo de las pantallas generales de los informes.
- Creación y desarrollo de los componentes de las 3 diferentes gráficas.
- Creación y desarrollo de la cabecera formulario del informe con la presentación de los datos que definen la situación docente actual obtenidos del back-end.
- Creación y desarrollo del cuerpo del formulario del informe con la presentación de los datos de las respuestas y calificaciones obtenidas del back-end.

6.8.2. Implementación

Back-end:

En el contexto del desarrollo del back-end, se procederá a explicar, en primer lugar, la implementación de todas las funcionalidades que son compartidas por ambos tipos de informes. Una vez finalizadas las implementaciones comunes, se procederá a explicar los detalles individuales de como se recupera la información de cada tipo de informe en el back-end. Este enfoque asegura una comprensión exhaustiva de las funcionalidades generales antes de profundizar en las especificidades de cada informe.

Por lo tanto, en el caso de ambos informes, el primer paso consiste en obtener todas las asignaturas en las cuales el docente ha participado como profesor. Además, dichas asignaturas deben tener los informes publicados por la administración, es decir, se ha determinado que sean visibles para el docente.

Para lograr esto, se implementa una nueva ruta en la API denominada "*/docente/-getAsignaturasPublicadas*". Esta URL de la API invoca la función "*getAsignaturasPublicadas(cod_doc)*" en la capa de datos, la cual recibe como parámetro el identificador del docente. En esta función, se ejecuta una consulta que selecciona el nombre de la asignatura, el año del curso y una concatenación de códigos de situaciones docentes correspondientes a las asignaturas en las cuales el docente participa como profesor o profesora. Asimismo, al igual que en el sprint anterior, se debe tener en cuenta el caso crítico en el cual existen situaciones docentes agrupadas y, en consecuencia, deben ser devueltas como una única entidad. Como resultado a la llamada, se devuelve una estructura semejante a la mostrada en el siguiente cuadro.

```
[
  {
    ‘‘nombre_asignatura’’: ‘‘Matemática aplicada’’,
    ‘‘21/22’’: { ‘‘situaciones_docentes’’: [‘‘sd010’’] },
    ‘‘22/23’’: { ‘‘situaciones_docentes’’: [‘‘sd000’’] }
  },
  {
    ‘‘nombre_asignatura’’: ‘‘Cálculo I’’,
    ‘‘22/23’’: { situaciones_docentes’’: [‘‘sd005’’,‘sd004’’,‘‘sd001’’]}
  }
]
```

Ejemplo de JSON obtenido tras la llamada a "*getAsignaturasPublicadas()*"

En este ejemplo se muestran las dos asignaturas que el docente autenticado ha impartido. Para cada una de ellas, se obtienen todos los años en los que el docente ha impartido dicha asignatura. Además, para cada año de la asignatura, se obtiene la situación docente a la cual pertenece. En el caso específico de este docente, se observa que

ha impartido la asignatura de "Matemática Aplicada" en los cursos académicos 21/22 y 22/23. Además, en el año académico 22/23, el docente ha impartido también la asignatura de "Cálculo I", la cual agrupa diferentes situaciones docentes. Esto puede deberse a la presencia de alumnos provenientes de diferentes titulaciones o grupos en dicha asignatura.

Con los datos proporcionados en el JSON, ahora es posible generar los desplegables necesarios para que el docente pueda seleccionar una asignatura que haya impartido y el año académico correspondiente de dicha asignatura para visualizar el informe. Por tanto el próximo paso es generar el informe una vez el docente pulse el botón "Consultar Informe". Como se ha adelantado, el informe consta de dos partes, la cabecera y el contenido. En la cabecera del informe se muestra un conjunto de datos relevantes que proporcionan contexto, como el nombre de la asignatura, el grado académico, el grupo, etc. Son los parámetros que describen la situación docente actual. Además, se presentan otros datos como el número de plazas matriculadas y cuestionarios completados por los alumnos.

Con el fin de obtener los datos relevantes de la situación docente actual, se ha implementado una nueva ruta en la API denominada "*/docente/getDatosSD*". Esta ruta permite obtener un JSON con todos los datos de interés para ser mostrados en la cabecera del informe. Al utilizar esta ruta, se pasa como parámetro una situación docente específica o, en el caso de situaciones docentes agrupadas, un conjunto de situaciones docentes. La API procesa esta información y recupera los datos correspondientes de la base de datos. El JSON devuelto contiene todos los datos necesarios para mostrar en la cabecera del informe.

Una vez cargada la cabecera del informe, se procede a cargar el cuerpo del informe mediante una nueva ruta en el servidor llamada "*/docente/getResultadosInformePersonal*". Esta ruta recibe la situación docente, ya sea agrupada o individual, y se encarga de obtener los resultados de las respuestas proporcionadas por los alumnos respectivas a dicha situación o situaciones docentes. La implementación de esta función es más compleja que las anteriores, ya que implica consultas extensas a la base de datos. Resumidamente, se obtiene la encuesta asociada a la situación docente. A continuación, se recorren todas las preguntas de dicha encuesta. Utilizando el código de cada pregunta y la situación docente proporcionada como parámetro, se recopilan las respuestas correspondientes a cada pregunta desde la base de datos. Para cada pregunta, además de recopilar datos como el id de la pregunta o el texto vinculado, se genera una estructura similar a la siguiente:

```
{ cod_respuesta: '1', cuantos: 0 }
{ cod_respuesta: '2', cuantos: 0 }
{ cod_respuesta: '3', cuantos: 0 }
{ cod_respuesta: '4', cuantos: 0 }
{ cod_respuesta: '5', cuantos: 0 }
```

Estructura para almacenar respuestas de una pregunta numérica.

La estructura anterior es un ejemplo de una pregunta numérica. Gracias a esta estructura, es posible almacenar el número de respuestas para cada opción de respuesta, lo que facilita la realización de cálculos adicionales como el total de respuestas para la pregunta o la media numérica de calificación obtenida.

Hasta este punto, la implementación de ambos informes ha sido idéntica, abarcando la obtención de: las asignaturas históricas personales del docente junto con los años en los que las ha impartido, los datos de la cabecera del informe que describen la situación docente y los datos del cuerpo del informe, que incluyen cada pregunta junto con una serie de datos identificativos y el recuento de respuestas para cada opción de respuesta. A partir de este punto, se explicará por separado la continuación de la implementación de cada informe en el back-end, diferenciando entre los informes personales y los informes agrupados.

Informes Personales:

Como se mencionó anteriormente, en los informes personales se desea visualizar el historial de la calificación media obtenida para una pregunta a lo largo de los años en los que el docente ha impartido la asignatura correspondiente. Para obtener esta información, es necesario definir un nuevo método en la API. La ruta asignada para esto es "*/docente/getHistoricoPregunta*", la cual recibe como parámetros el código de la asignatura y el código de la pregunta. La ruta mencionada tiene que devolver una estructura que refleje, para cada situación docente en la que participe el docente y se imparta la asignatura correspondiente, la media obtenida en esa pregunta. Esta estructura estará ordenada de manera ascendente por año.

En el back-end, concretamente en la capa de datos, se ha implementado un método llamado "*getSituacionesAsignatura()*" que recibe como parámetros el identificador del docente asociado (extraído del token), el código de la asignatura y el código de la pregunta. Este método obtiene el listado de situaciones docentes junto con el año del curso asociado en los que el docente ha impartido esa asignatura para la asignatura proporcionada.

Por cada entrada en esta estructura generada, se llama a otro método denominado "*getResultadosPregunta()*" que recibe como parámetros al docente actual y el código de la pregunta de la cual queremos la media de respuestas a lo largo de los años. Este método, de manera similar a lo visto anteriormente, obtiene una estructura que lista todas las respuestas para la pregunta actual y la cantidad de personas que han seleccionado cada opción de respuesta. Esto permite posteriormente calcular la media para esta pregunta y, por consiguiente, para todas las situaciones docentes asociadas dicha pregunta.

Comparativa de Resultados:

En los informes comparativos, se requiere implementar la funcionalidad de comparar la calificación media obtenida por un docente para una asignatura y año específico con la media del el grupo, del departamento, etc. Para lograr esto, se deben crear los métodos necesarios para obtener las calificaciones medias en cada uno de estos ámbitos.

Para cumplir con este objetivo, se han implementado 6 nuevas rutas en la API, cada una con su respectivo método en la capa de datos. Estas rutas se utilizan para obtener la media respectiva en: la asignatura, el grupo, el departamento, el curso, la titulación y el centro al que pertenece la situación docente de la asignatura personal que está revisando el docente. Los 6 métodos son muy similares y comparten una lógica común. Para esta explicación, tomaremos como ejemplo la obtención de la media del grupo.

En la llamada a "*/docente/getMediaGrupo*", se proporcionan como parámetros el código del grupo y el código de la encuesta asociada. Estos datos se utilizan para invocar el método "*getMediaGrupo()*" en la capa de datos del back-end. Este método se encarga de obtener el listado de situaciones docentes que están asociadas al grupo y a la encuesta mencionada.

Una vez completado el listado de situaciones docentes, se llama a otra función denominada "*getMediasGeneral()*". Esta función es utilizada por las 6 funciones de cálculo de medias y su objetivo es recorrer todas las preguntas y obtener el recuento de respuestas para cada pregunta de cada situación docente de la lista de situaciones docentes proporcionada. Luego, se agrupan todas las respuestas de las preguntas y se suman los recuentos de respuestas.

El último paso consiste en calcular la media utilizando el número de alumnos que han respondido a cada respuesta y la calificación asignada. El resultado final es un JSON que contiene todas las preguntas de la encuesta y, para cada pregunta, la media obtenida para cada respuesta. Esta media se calcula como la media ponderada de todas las situaciones docentes en las que participa el grupo proporcionado como parámetro.

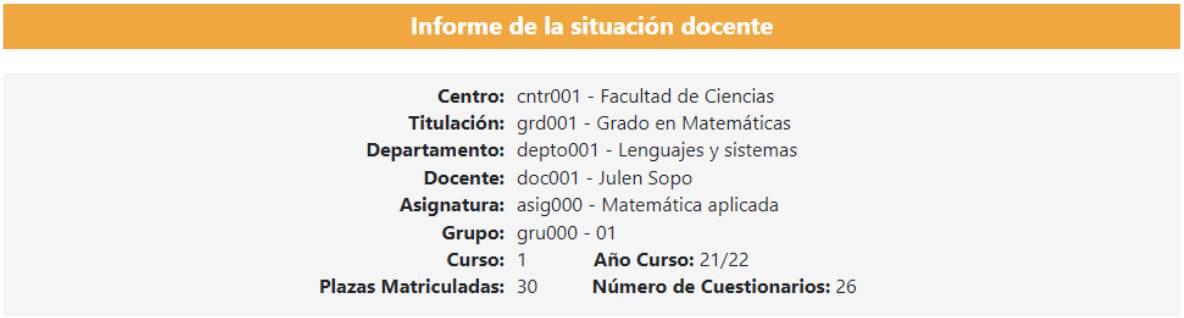
Front-end:

El docente nada mas autenticarse, accede al índice de docentes donde puede bien consultar las encuestas o consultar los informes. Si hace clic en el segundo botón se abre el menú de informes del docente donde aparecen dos nuevas opciones "Consultar informes personales" y "Comparativa de resultados". Tanto en el back-end como en el front-end, existen similitudes en la implementación de estos dos informes. Por lo tanto, aquí también, se abordará primero la implementación común a ambos y luego se describirá la implementación específica de cada uno, que difiere entre sí.

En primer lugar, se debe proceder a la creación de dos componentes de Angular, uno para cada tipo de informe. En cada componente, se requiere la habilitación de una serie de elementos desplegables. El primero corresponde a las asignaturas que imparte el docente, obtenidas mediante una solicitud a la ruta "*/docente/getAsignaturasPublicadas*" en el back-end. A continuación, se debe habilitar un segundo desplegable, el cual se mostrará una vez se seleccione una opción en el primero. Este segundo desplegable proporcionará una lista de los años en los que el docente actual ha impartido la asignatura seleccionada en el primer desplegable.

Los datos del segundo desplegable también se obtienen de la llamada anterior a la API. En el caso de la comparativa de resultados, se requerirá crear un tercer desplegable

para indicar el tipo de comparación deseada, ya sea por asignatura, centro, departamento, etc. Una vez que se haya seleccionado un valor en todos los desplegables, se habilitará un botón con el texto "Mostrar Informe". Al hacer clic en dicho botón, se realiza una solicitud a la ruta de la API llamada "*/docentes/getDatosSD*", la cual obtiene el conjunto de datos relevantes para la cabecera del informe. Estos datos se recopilan y se muestran de manera ordenada en la vista correspondiente. A continuación, se presenta la cabecera del informe generado, la cual puede observarse en la Figura 6.13.



6.13. Figura: Ejemplo de cabecera de un informe.

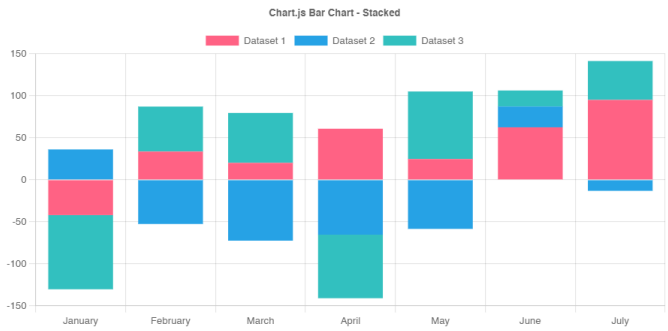
Una vez generada la cabecera del informe con éxito, se procede a la generación del cuerpo del informe. Para ello, se hace uso de la ruta "*/docente/getResultadosInforme-Personal*", y se realiza un procesamiento de la respuesta obtenida. En primer lugar, se eliminan las preguntas no numéricas, ya que no es posible calcular una media para ellas. Para las preguntas restantes, se calcula la media de cada respuesta.

El informe se estructura en varias columnas. Después de la llamada a la API y el procesamiento de la respuesta, se pueden completar dos de las columnas del informe. En la columna de la izquierda, se muestra el texto de la pregunta, mientras que en otra columna de la derecha se muestra la media obtenida para esa pregunta. Esto deja un espacio vacío en el informe. Vamos a ver ahora como se rellena dicho espacio en el informe en función del tipo del mismo.

Informes Personales:

En el espacio en blanco de cada fila se coloca un gráfico de barras ponderado similar al mostrado en la Figura 6.12. Cada bloque del gráfico representa el porcentaje de cada calificación de la pregunta actual. Este gráfico proporciona una visualización adicional que complementa la información del informe, facilitando una comprensión más rápida y eficiente. De esta manera, se logra generar el cuerpo del informe de forma clara y visualmente atractiva, presentando tanto el texto de las preguntas, la media obtenida como la representación gráfica de las calificaciones.

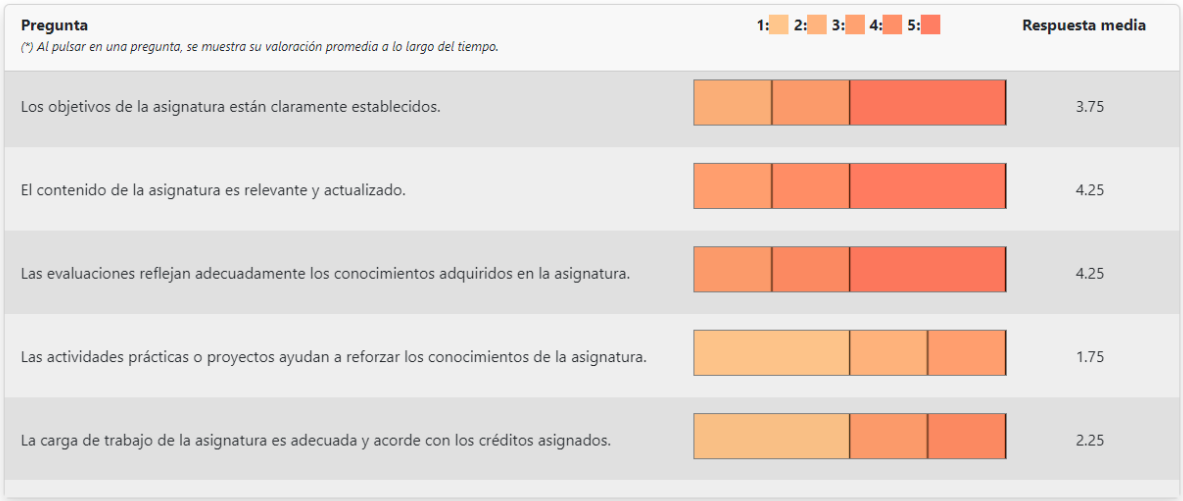
El gráfico se genera dinámicamente utilizando la biblioteca de gráficos de *JavaScript chartjs*. Se debe crear el componente correspondiente al gráfico en Angular, siguiendo la estructura de implementación proporcionada en los ejemplos de la documentación de *chartjs*. En este caso, se utiliza un gráfico de barras agrupado (Ver Figura 6.14).



6.14. Figura: Gráfico de barras agrupado de *chartjs*.

Para adaptar el gráfico a nuestras necesidades, se realizan algunas modificaciones. En primer lugar, se define que los datos serán proporcionados externamente al componente, a través de un *input*. En cuanto al aspecto visual, se ajusta la escala del gráfico para que siempre se muestre en un porcentaje del 100 %, normalizado. Además, se rota el gráfico para que el eje principal sea el eje y, lo que resulta en una visualización horizontal. Por último, se pueden realizar ajustes adicionales, como la eliminación de la cuadrícula, la personalización de la escala o la asignación de colores a cada segmento del gráfico.

El informe presenta una estructura de filas y columnas, y en cada fila se incrusta de manera dinámica el gráfico recién creado. Para generar el gráfico, se le pasan los datos necesarios, que en este caso se encuentran en una lista llamada "cuantos". Esta lista almacena el número de respuestas obtenidas para cada pregunta. El componente del gráfico procesa la lista y genera el gráfico correspondiente, ajustando los datos de manera adecuada. El gráfico resultante refleja el porcentaje de respuestas de cada respuesta sobre el total para cada pregunta. Una vez que el gráfico se ha incrustado en la columna correspondiente y todas las columnas del cuerpo del informe han sido completadas, el informe adquiere su aspecto final:



6.15. Figura: Ejemplo del cuerpo del informe personal.

Una funcionalidad destacada del informe personal es la capacidad de visualizar el

histórico de valoración media de una pregunta específica a lo largo de los años en los que el docente ha impartido la asignatura correspondiente. Esta funcionalidad se activa al hacer clic en el texto de la pregunta dentro del informe personal y se abre una nueva pestaña que muestra la información requerida.

Para implementar la funcionalidad de mostrar el histórico de valoración media de una pregunta en el informe personal, se debe crear una nueva función en el front-end llamada "*mostrarGraficoPregunta()*", que reciba como parámetro el código de la pregunta actual. Esta función, utilizando el servicio que se comunica con el back-end, realiza una llamada a la API en la ruta "*/docente/getHistoricoPregunta*".

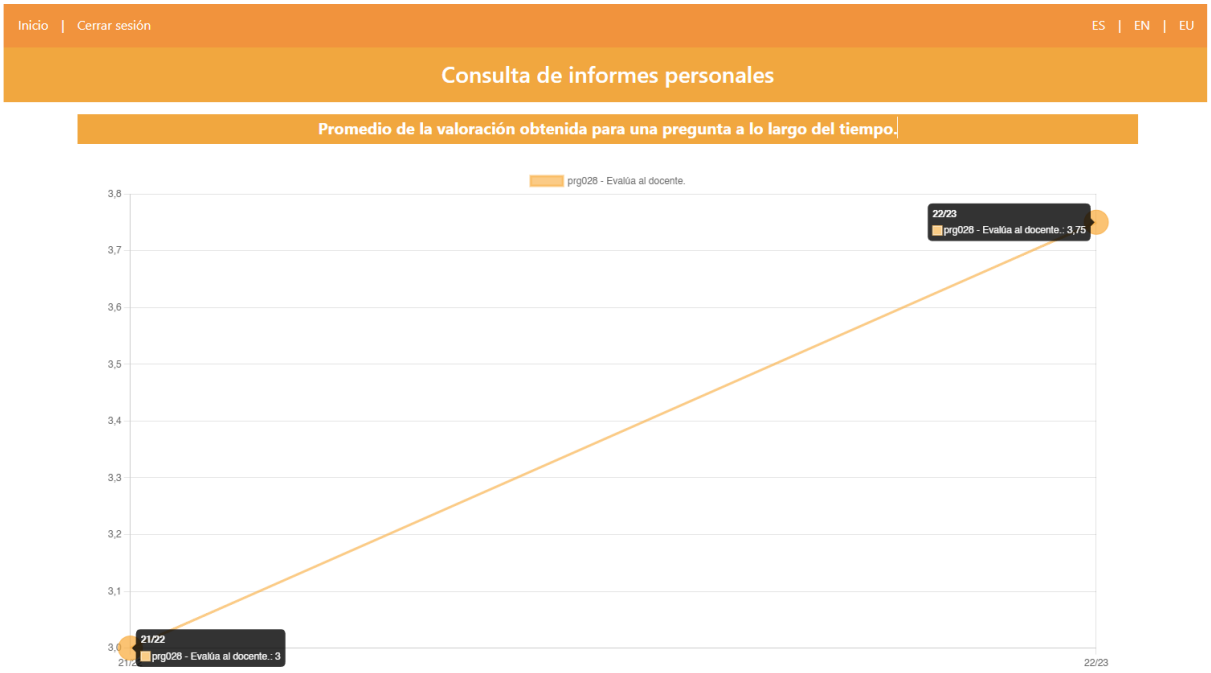
La ruta mencionada recibe el código de la asignatura y el código de la pregunta como parámetros. La llamada a esta ruta devuelve una lista de situaciones docentes, donde cada situación docente representa un año específico en el cual el docente ha impartido la asignatura actual. Para cada situación docente de la lista, se obtiene el recuento de respuestas para la pregunta en cuestión, de manera similar a lo mencionado anteriormente. A partir de este recuento de respuestas, se calcula la media obtenida para dicha pregunta.

Al igual que con el gráfico anterior, es necesario desarrollar un componente Angular que represente un gráfico lineal utilizando el estilo de diseño conocido como *Point Styling.* Para ello, utilizaremos la biblioteca *chartsjs* y configuraremos el gráfico de acuerdo a las recomendaciones proporcionadas en la API de *chartsjs*. Los datos que se mostrarán en el gráfico serán recibidos como entrada por el componente desde el cual se invoca.

Por lo tanto, al pulsar en el texto de una pregunta, se abre una nueva pestaña que muestra el gráfico histórico de las respuestas promedio de esa pregunta en ese contexto. Sin embargo, surge un problema al transferir los datos del primer componente al segundo si éste se desea abrir en una nueva pestaña. Recordemos que el objeto *Router* puede pasar datos como parámetros, pero estos son visibles en la URL y podrían ser modificados, lo que alteraría el gráfico. Por lo tanto, debemos utilizar el servicio de datos creado para transferir los datos desde un componente a otro (implementado en el sprint 3). El inconveniente es que el enrutador no puede redirigir al usuario a una nueva ruta abriendo una nueva pestaña, solo puede hacerlo en la pestaña actual.

Por lo tanto, debemos recurrir al objeto *window*, pero éste no es compatible con el servicio de datos implementado. Se necesita encontrar una nueva manera de transferir datos de una pestaña a otra. La forma más efectiva es utilizando el almacenamiento local del navegador. Antes de abrir la nueva pestaña, el primer componente coloca todos los datos necesarios en el almacenamiento local y, una vez que el segundo componente se abre, los lee y los elimina de inmediato. De esta manera, nos aseguramos de transferir los datos correctamente y al borrarlos, nos aseguramos de que no queden permanentemente visibles.

Después de haber implementado la funcionalidad al completo, al hacer clic en el texto de una pregunta, se transfieren los datos requeridos desde el primer componente al segundo. Este último componente se abre en una nueva pestaña y usa los datos del componente anterior como input en el gráfico. A continuación, se muestra un ejemplo de la vista del componente que exhibe el historial de promedios de una pregunta específica en una asignatura que el docente autenticado ha impartido durante dos años.



6.16. Figura: Ejemplo de un informe histórico de pregunta.

En la Figura 6.16 se puede observar un informe histórico para la última pregunta del informe: "Evalúa al docente". Este docente ha impartido dicha asignatura durante dos años académicos, el 21/22 y el 22/23. Se observa además que ha mejorado la puntuación en dicha pregunta, fácilmente observable por la tendencia ascendente del gráfico. En el curso 21/22 su nota media era de 3 y en 22/23 ha subido 0.75 puntos dejándola en 3.75. Esta funcionalidad, permite obtener una visión más completa y detallada del rendimiento y la evolución de la valoración de los alumnos en relación con una pregunta en particular. Esto proporciona al docente información valiosa para analizar tendencias, identificar áreas de mejora y tomar decisiones informadas para el desarrollo y la adaptación de su práctica docente.

Comparativa de Resultados:

En los informes comparativos, se utiliza una estructura de 4 columnas. Dos de ellas son las mismas que se encuentran en el informe personal, una para mostrar el texto de la pregunta actual y otra para mostrar la respuesta media. La tercera columna se reserva para mostrar los datos de la comparativa seleccionada en el desplegable y la cuarta se utiliza para mostrar un gráfico comparativo entre la media personal y la comparada.

Para obtener estos datos de comparativa, se consume el recurso correspondiente de la API, dependiendo de qué tipo de media se desee comparar. Una vez obtenido el JSON con los datos de la comparativa, se realiza el procesamiento necesario para obtener la media ponderada a partir del recuento de respuestas. Finalmente, esta media ponderada se coloca en la vista, en la columna correspondiente de cada fila del informe.

Para la columna restante, se desea incrustar una gráfica que visualice la comparación entre la media personal y la media comparada. Siguiendo el mismo enfoque que antes, se crea un componente de Angular específico para el gráfico y se configura de acuerdo a las preferencias de diseño, como márgenes, colores y ejes.

Una vez que el gráfico está configurado, se le pasan los datos necesarios, que en este caso son la media personal y la media comparada. Estos datos se utilizan para generar dos barras horizontales en el gráfico, que representan visualmente la media de cada valor. El objetivo es que con un simple vistazo se pueda determinar si la media personal está por encima o por debajo de la media comparada. Esta visualización brinda una poderosa herramienta para comprender rápidamente la situación.

Una vez que el gráfico se ha incrustado en la columna correspondiente y todas las columnas del cuerpo del informe han sido completadas, el informe adquiere su aspecto final (Ver Figura 6.17).

6.17. Figura: Ejemplo del cuerpo del informe comparativo.

6.8.3. Revisión del sprint

Después de la reunión con el cliente, se validó que la implementación de los informes cumplía con los requisitos acordados previamente. Sin embargo, durante la reunión surgieron algunos cambios y ampliaciones adicionales que se debían realizar en la implementación. Estos cambios y ampliaciones fueron discutidos y acordados con el cliente para asegurar su satisfacción con el producto final.

En primer lugar, se acordó añadir el número de personas que habían respondido a cada pregunta en el informe personal. La implementación de esta funcionalidad no presentó mayores complicaciones, ya que solo fue necesario añadir una columna adicional a la

izquierda de la gráfica para mostrar el recuento total de respuestas para cada respuesta. Con esta ampliación realizada, el informe personal adquirió su aspecto final, como se muestra en la Figura 6.18.

6.18. Figura: Ejemplo del cuerpo del informe personal actualizado.

En la Figura 6.18 se observa como se ha añadido la columna adicional mencionada, para este caso concreto todas las preguntas han obtenido 4 respuestas excepto la primera dado que alguno de esos alumnos no la habrá respondido.

El segundo cambio fue en el informe comparativo. En lugar de comparar con el histórico del grupo, departamento, centro, etc., se decidió obtener la media de comparación **del mismo año** que el informe personal que se estaba visualizando. Para lograr esto, se modificó la consulta a la base de datos para obtener únicamente las respuestas del año al que hacían referencia las respuestas del informe personal actual.

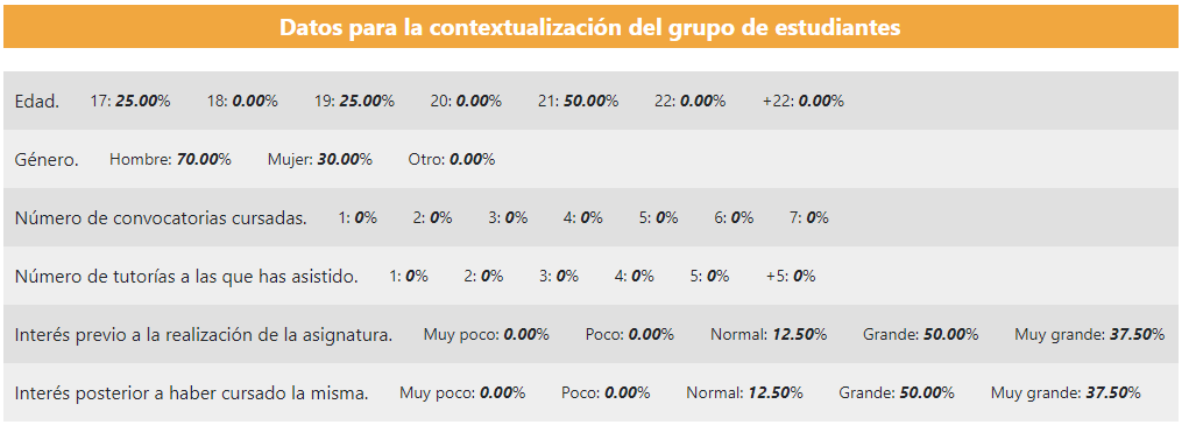
Además, se agregó una nueva comparativa llamada "histórico asignatura". En este caso, la media se calcula tomando en cuenta todos los años en los que se ha impartido dicha asignatura. Es importante destacar que, además de la nueva comparativa "histórico asignatura", se mantiene la comparativa con la "asignatura", y son diferentes. La comparativa con "asignatura" corresponde a la nota media de la asignatura obtenida en el **curso actual** por todos los docentes que la imparten, mientras que "histórico asignatura", tiene en cuenta todos los años registrados, no solo el actual.

Después de las ampliaciones realizadas, se planteó una tercera y última modificación en el informe personal. Esta vez se requería agregar un nuevo apartado que se mostraría antes del cuerpo del informe y hará referencia a los resultados de las preguntas no numéricas del grupo de alumnos con el objetivo de proporcionar contexto al informe. Dado que solo se pueden obtener medias para las preguntas numéricas en el informe, las preguntas no numéricas habían sido excluidas. Sin embargo, ahora se utilizarán para contextualizar el informe. Para cada pregunta no numérica, se desea construir un apartado que muestre el porcentaje de respuesta para cada opción.

Antes de la implementación, se realizaron cambios en la base de datos debido a que algunas preguntas, como "Número de convocatorias cursadas" o "Número de tutorías asistidas", se consideraban preguntas numéricas debido a que las respuestas eran números. Sin embargo, para la nueva sección de contexto del grupo de estudiantes en el informe, se debieron considerar estas y otras preguntas como textuales en lugar de numéricas. Esto se hizo para distinguir claramente entre las preguntas que evalúan la docencia y aquellas que proporcionan información contextual sobre el grupo de estudiantes.

También fue necesario realizar modificaciones significativas en la función encargada de obtener los datos de las respuestas del formulario de una situación docente. Para las respuestas numéricas, se obtenían todas las respuestas con un contador de respuestas igual a cero, y se iba rellenando en función al numero de respuestas en la base de datos para cada pregunta. Las que no tuvieran ninguna respuesta se devolvían con el contador a cero. Implementar esta solución para las preguntas numéricas era posible debido a que todas las respuestas numéricas tenían el mismo código. Sin embargo, para las respuestas textuales, no se podía aplicar una solución semejante con otra estructura ya que cada respuesta textual es única, incluso si comparten el mismo texto. Por ejemplo, las respuestas "poco" en las preguntas "Interés previo a la realización de la asignatura" y "Interés posterior a haber cursado la misma" se consideran respuestas distintas, aun teniendo el mismo texto, debido a que pertenecen a preguntas diferentes.

Una vez realizadas las modificaciones en la base de datos, el último paso es presentar los datos obtenidos desde el back-end. Para ello, se crea una nueva sección en el informe llamada "Datos para la contextualización del grupo de estudiantes". En esta sección, se mostrará cada pregunta textual de la misma manera que en el cuerpo del informe con las numéricas. En este caso, en lugar de mostrar una media (que no es posible para preguntas textuales), se listarán todas las posibles respuestas junto con el porcentaje obtenido por cada respuesta. La implementación de la vista asociada de la sección de contextualización del informe se pueden observar en la Figura 6.19.



6.19. Figura: Ejemplo del informe de contextualización del grupo de estudiantes.

6.8.4. Pruebas

El objetivo de estas pruebas es garantizar la adecuada apertura, programación y posterior cierre de las encuestas, abordando situaciones críticas para comprobar su correcto funcionamiento.

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.1. El docente no tiene asignaturas.	El desplegable asociado al listado de asignaturas del docente no debería mostrar ninguna opción y por tanto no se debería activar el botón de "Mostrar informe".	El desplegable no tiene elementos y por tanto no puede visualizar ningún informe.	Sí
1.2. El docente tiene asignaturas de situaciones docentes sin agrupar (**simple**).	Se listan las asignaturas impartidas por el docente en toda su carrera. Al seleccionar una asignatura se debería activar el segundo desplegable para escoger el año de dicha asignatura para visualizar el informe.	Se han registrado correctamente las asignaturas de situaciones docentes sin agrupar en el desplegable. Al pulsar en una se habilita el segundo desplegable.	Sí
1.3. El docente tiene asignaturas de situaciones docentes **agrupadas**.	El docente tiene varias situaciones docentes agrupadas que hacen referencia a la misma asignatura del mismo año. Al estar agrupadas se debería mostrar como una sola asignatura en el desplegable, no como una asignatura diferente por cada situación docente.	Se han registrado correctamente las asignaturas de situaciones docentes agrupadas en el desplegable y no se observan duplicidades debido a las situaciones docentes agrupadas.	Sí
1.4. El docente tiene asignaturas cuyos informes no se han publicado por administración.	Las situaciones con informes no publicados no deben salir en el des plegable de asignaturas.	Las situaciones docentes no publicadas no se muestran en el desplegable.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.5. El docente tiene una asignatura **simple** que ha impartido un solo año.	Seleccionar una asignatura simple que ha sido impartida solo un año debería habilitar el segundo desplegable que muestra el único año en el que se ha impartido la asignatura.	Se ha seleccionado una asignatura del primer desplegable. Posteriormente en el segundo, se muestra el único año de impartición.	Sí
1.6. El docente tiene una asignatura **agrupada** que ha impartido un solo año.	No se debe mostrar la asignatura en el desplegable tantas veces como agrupada esté, sólo debe aparecer una vez. Seleccionar una asignatura agrupada que ha sido impartida solo un año debería habilitar el segundo desplegable que muestra el único año en el que se ha impartido la asignatura.	Se ha seleccionado una asignatura del primer desplegable. Posteriormente en el segundo, se muestra el único año de impartición.	Sí
1.7. El docente tiene al menos una asignatura **simple** que ha impartido varios años.	El desplegable asociado al listado de asignaturas del docente debería dicha asignatura solo una vez independientemente de las veces que se haya impartido. Al escogerla, se habilita el segundo desplegable para escoger entre los años impartidos.	El primer desplegable solo muestra la asignatura una vez y al escoger la asignatura se habilita el segundo desplegable que muestra el listado de todos los años impartidos.	Sí
1.8. El docente tiene al menos una asignatura **agrupada** que ha impartido varios años.	El desplegable de asignaturas del docente debe mostrar cada asignatura una sola vez, sin importar cuántas veces esté agrupada. Al seleccionar una asignatura, se activará otro desplegable para elegir entre los años impartidos.	El primer desplegable solo muestra la asignatura una vez y al escoger la asignatura se habilita el segundo desplegable que muestra el listado de todos los años impartidos.	S

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
2.1. Visualización de la información de la situación docente **simple** del informe.	En la cabecera del informe de la situación docente simple se deben mostrar datos relevantes como el grupo, departamento, curso, así como el número de matrículas y respuestas registradas, con presentación estética y centrada.	Se muestra la información de la situación docente simple de manera agradable a la vista.	Sí
2.2. Visualización de la información de la situación docente **agrupada** del informe.	En la cabecera del informe de la situación docente agrupada se mostrarán datos relevantes como el grupo, departamento, curso, matrículas y respuestas registradas. Esto incluirá la recopilación y suma de datos de todas las situaciones docentes agrupadas. Finalmente, se presentarán estos datos de forma estética y centrada.	Se muestra la información de la situación docente agrupada de manera correcta y teniendo en cuenta todas las situaciones de la lista. Se realiza el contraste y suma de los datos correctamente. Finalmente, se presenta de manera agradable a la vista.	Sí
3.1. Las preguntas no numéricas del informe no han recibido respuestas.	Si una pregunta no numérica no ha sido respondida el porcentaje de cada respuesta asociada a la pregunta se muestra con un 0 %.	Las preguntas sin respuestas registradas, se visualizan con el porcentaje de respuesta a 0.	Sí
3.2. Las preguntas no numéricas del informe tienen respuestas (situación docente **simple**).	Si una pregunta no numérica ha sido respondida en una situación docente simple, el informe debe mostrar el porcentaje de gente que ha respondido a cada opción de la pregunta.	Las preguntas con respuestas registradas se visualizan con el porcentaje de respuesta pertinente respecto al total de alumnos que han respondido esa pregunta.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
3.3. Las preguntas no numéricas del informe tienen respuestas (situación docente **agrupada**).	Si una pregunta no numérica ha sido respondida en una o varias situaciones docentes agrupadas, el informe debe mostrar el porcentaje de gente que ha respondido a cada opción de la pregunta considerando todas las situaciones agrupadas.	Las preguntas con respuestas registradas se visualizan con el porcentaje de respuesta pertinente respecto al total de alumnos que han respondido esa pregunta considerando la suma de todas las situaciones docentes agrupadas.	Sí
4.1. El formulario asociado a la situación docente no tiene respuestas registradas.	Si una o varias preguntas numéricas no han sido respondidas, el informe debe mostrar el campo de la media con un guión "-" y en lugar de la gráfica se muestra un mensaje indicando la situación.	La media se muestra con un guión para indicar que no ha habido respuesta y en lugar del gráfico se muestra el mensaje "No hay respuestas".	Sí
4.2. El formulario asociado a la situación docente **simple** tiene respuestas registradas.	Por cada fila del informe se deben recopilar el total de respuestas para cada opción de la pregunta. Con dichos datos se debe calcular la media de respuesta para cada pregunta y se debe insertar en la columna correspondiente. De la misma manera, se debe crear el gráfico que refleje el porcentaje de respuestas para cada opción.	Por cada fila del informe se obtiene el recuento de respuestas para cada opción. Con esos datos se calcula la media y se inserta en la última columna del informe. Además, se crea e inserta el gráfico con el porcentaje de respuestas para cada opción. A la izquierda del gráfico se inserta el numero total de respuestas a dicha pregunta.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
4.3. El formulario asociado a la situación docente **agrupada** tiene respuestas registradas.	Se recopilan las respuestas para cada opción de la pregunta en el informe, teniendo en cuenta todas las respuestas de todas las situaciones docentes. Se debe calcular la media de respuesta y presentarla en la columna correspondiente. También se debe generar un gráfico que refleje el porcentaje de respuestas para cada opción.	En el informe, para cada opción se obtiene el recuento de respuestas, sumando todas las respuestas de las situaciones agrupadas, y se calcula la media, insertándola en la última columna. También se incluye un gráfico con el porcentaje de respuestas por opción, mostrando el número total de respuestas a la izquierda del gráfico.	Sí
5.1. Se hace clic en una pregunta que no tiene respuestas registradas.	Si se hace clic en una pregunta sin respuestas, se debería abrir el gráfico de respuesta media a lo largo de los años en dicha asignatura con dicho profesor pero estar vacío, al menos en el año actual del informe.	Al hacer clic en una pregunta, se abre una nueva pestaña con el gráfico asociado a la media histórica de respuestas de esa pregunta en la asignatura y con el profesor correspondiente. Si no hay respuestas, el gráfico aparecerá vacío en la columna del año actual del informe.	Sí
5.2. Se hace clic en una pregunta con respuestas registradas de una situación docente **simple**.	Si se hace clic en una pregunta con respuestas registradas de una situación docente simple, se debería abrir el gráfico de respuesta media a lo largo de los años mostrando la respuesta media a dicha pregunta en cada año registrado.	Al pulsar en una pregunta se abre una nueva pestaña con el gráfico asociado a la media de respuestas históricas de dicha pregunta en dicha asignatura con dicho profesor.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
5.3. Se hace clic en una pregunta con respuestas registradas de una situación docente **agrupada**.	Si se hace clic en una pregunta con respuestas registradas de una situación docente agrupada, se debería abrir el gráfico de respuesta media a lo largo de los años mostrando la respuesta media, respectiva al conjunto de respuesta de todas las situaciones agrupadas de pregunta actual en cada año registrado.	Al pulsar en una pregunta se abre una nueva pestaña con el gráfico asociado a la media ponderada de todas las situaciones docentes de dicha pregunta en dicha asignatura con dicho profesor a lo largo de los años.	Sí
6.1. **Comparativa de informes**. Se compara el informe actual de una situación docente con la media histórica de la asignatura.	En una columna se debe insertar la respuesta media de la pregunta para la situación docente actual, en otra columna se debe colocar la media histórica de esa asignatura, independientemente del profesor o situación docente. Finalmente, en la última columna se debe agregar un gráfico que compare ambas medias.	La primera media es correcta y es igual que en el informe personal, la segunda media calcula se inserta correctamente y el gráfico es creado correctamente a partir de los datos de las anteriores dos columnas.	Sí
6.2. **Comparativa de informes**. Se compara el informe actual de una situación docente con la media de la asignatura.	La columna que varía es la de la comparativa. Se debe mostrar la media de resultados de dicha asignatura en el año escogido independientemente de qué profesor la imparta en qué situación docente.	La media se calcula correctamente teniendo en cuenta todas las situaciones que incluyan la asignatura y año actual.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
6.3. **Comparativa de informes**. Se compara el informe actual de una situación docente con la media del grupo.	La columna que varía es la de la comparativa. Se debe mostrar la media de resultados de dicho grupo en el año escogido independientemente de qué asignatura sea la impartida.	La media se calcula correctamente teniendo en cuenta todas las situaciones que incluyan al grupo y año actual.	Sí
6.4. **Comparativa de informes**. Se compara el informe actual de una situación docente con la media del departamento.	La columna que varía es la de la comparativa. Se debe mostrar la media de resultados de dicho departamento en el año escogido, haciendo la media de todos los informes obtenidos de dicho departamento y año.	La media se calcula correctamente teniendo en cuenta todas las situaciones que incluyan al departamento y año actual.	Sí
6.5. **Comparativa de informes**. Se compara el informe actual de una situación docente con la media del curso.	La columna que varía es la de la comparativa. Se debe mostrar la media de resultados de dicho curso en el año escogido, haciendo la media de todos los informes realizados por los alumnos de dicho curso.	La media se calcula correctamente teniendo en cuenta todas las situaciones que incluyan al curso actual del informe en el año actual.	Sí
6.6. **Comparativa de informes**. Se compara el informe actual de una situación docente con la media de la titulación.	La columna que varía es la de la comparativa. Se debe mostrar la media de resultados de dicha titulación en el año escogido, haciendo la media de todos los informes obtenidos esa titulación independientemente del curso, grupo, asignatura, etc. La media de informes de toda la carrera en el curso escogido.	La media se calcula correctamente teniendo en cuenta todas las situaciones que incluyan la titulación y año actual.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
6.3. **Comparativa de informes**. Se compara el informe actual de una situación docente con la media del centro.	La columna que varía es la de la comparativa. Se debe mostrar la media de resultados del centro subyacente de la situación del informe actual en el año escogido.	La media se calcula correctamente teniendo en cuenta todas las situaciones que incluyan al centro y año actual, es decir todos los informes del centro en el año actual independientemente del curso, grado, titulación, etc.	Sí

6.9. Sprint 8: Gestión de encuestas para admins.

El octavo sprint no fue originalmente contemplado en el plan de implementación, ya que se había acordado centrarse en el desarrollo completo de las funcionalidades para docentes y alumnos, dejando la de administración fuera debido a la carga de trabajo considerable que implicaba. No obstante, he tomado la decisión de ampliar este proyecto e incorporar a SiREnO la funcionalidad que considero de mayor relevancia para los administradores del sistema: la gestión de las encuestas.

Esta nueva funcionalidad implementada en SiREnO dedicado a la administración universitaria, proporciona una lista completa de todas las encuestas, tanto las vigentes como las expiradas, permitiendo su clasificación mediante dos opciones de filtrado. La primera opción permite filtrar las encuestas según el porcentaje de respuestas obtenidas en cada una, mostrando solo aquellas con un porcentaje de respuesta igual o menor al seleccionado. La segunda opción permite seleccionar el año del curso correspondiente para visualizar las encuestas específicas de dicho año. En cualquier caso, las encuestas se ordenan por la fecha de expiración, es decir, primero se muestran las encuestas expiradas y luego aquellas que tienen menos tiempo para expirar. En caso de empate en el primer criterio, se ordenan según las que menos veces han sido abiertas.

La importancia de esta herramienta radica en la posibilidad de obtener una visión integral de la participación y el interés de los alumnos en las encuestas. Gracias al análisis del porcentaje de respuestas y la cantidad de veces que se han abierto las encuestas, se pueden identificar posibles anomalías o tendencias inesperadas. En caso de detectar situaciones peculiares, la herramienta permitirá la reapertura de las encuestas, incluso si ha expirado el plazo original. El procedimiento de reactivación deberá incorporar la automatización del envío de un correo electrónico a todos los alumnos involucrados en la situación docente asociada a la encuesta que se desea abrir. El propósito de esta acción es notificar formalmente la apertura extraordinaria de dicha encuesta.

Otra funcionalidad clave que se pide incorporar es la capacidad de seleccionar varias encuestas simultáneamente y abrir todas ellas de forma conjunta, evitando la necesidad de abrir cada encuesta de manera individual. Esta mejora permitirá un manejo más eficiente y ágil para el administrador al poder realizar acciones en lote. Esta funcionalidad proporcionará una experiencia más fluida y aumentará la eficiencia en la gestión de encuestas, al reducir considerablemente el tiempo y esfuerzo requerido para llevar a cabo tareas administrativas relacionadas con las mismas.

6.9.1. Análisis y diseño

Cuando un administrador acceda al sistema SiREnO, se encontrará con su menú principal que constará de cuatro funcionalidades, una de las cuales es la "Administración de encuestas". Al seleccionar esta opción, se abrirá un submenú dedicado a la administración de las encuestas, que incluye dos funcionalidades nuevas: "Administración de encuestas en curso o finalizadas" y "Administración de preguntas de encuesta". La funcionalidad a implementar es la primera de las dos.

Al acceder a la funcionalidad "Administración de encuestas en curso o finalizadas", se presentarán los dos desplegables mencionados previamente en la introducción. El primer desplegable permitirá al administrador seleccionar un porcentaje de respuestas recibidas para las encuestas. Con la selección de este desplegable, se filtrarán las encuestas según el porcentaje de respuestas obtenidas en cada una, mostrando únicamente aquellas que tengan un porcentaje de respuesta igual o menor al valor seleccionado.

El segundo desplegable permitirá al administrador seleccionar el año del curso correspondiente, con el propósito de visualizar las encuestas específicas de dicho año. Con esta opción, se facilitará al administrador la posibilidad de enfocarse en las encuestas asociadas a un período académico determinado.

Una vez que se han seleccionado los dos desplegables, se activará un botón denominado "Mostrar Encuestas", cuyo propósito es listar todas las encuestas que cumplen con los filtros seleccionados. Al hacer clic en dicho botón, se desplegará en la pantalla la lista completa de encuestas correspondientes. Es importante mencionar que, en determinadas ocasiones, si el filtrado es demasiado permisivo, como por ejemplo, mostrar todas las encuestas de todos los años con un porcentaje de respuesta igual o menor al 90 %, el listado resultante podría ser muy extenso, lo que a su vez podría ralentizar el proceso de carga tanto del sistema como del navegador. Además, la visualización de una lista tan extensa podría dificultar la gestión de las encuestas al presentarse todas en una única página.

Con el fin de abordar esta situación, se va a implementar la funcionalidad de paginación de encuestas. Esta característica establece un límite de encuestas por página y las divide en varias páginas según corresponda. De esta manera, se optimiza la eficiencia del proceso tanto para el sistema como para la presentación al administrador, ya que se evita la carga masiva de datos y se proporciona una experiencia de manejo más ágil y ordenada al presentar las encuestas de manera segmentada en diferentes páginas.

Como se ha mencionado anteriormente, se tiene la necesidad de habilitar la funcionalidad que permita abrir múltiples encuestas de manera simultánea. Para lograrlo, se procederá a agregar un elemento de selección a cada una de las encuestas. Al seleccionar el *checkbox* de una encuesta en particular, ésta quedará marcada como seleccionada. A su vez, al seleccionar una o varias encuestas, se habilitará un botón adicional denominado "Abrir Seleccionadas", ubicado a la izquierda del botón previamente mencionado. Como resultado de pulsar dicho botón, se iniciará el proceso para abrir todas las encuestas seleccionadas.

Indiferentemente de si se pretende abrir una única encuesta de manera directa o una selección de encuestas, el procedimiento de apertura será idéntico. En primera instancia, se mostrará un cuadro de diálogo emergente para seleccionar la fecha de finalización de la apertura extraordinaria. Una vez que se verifique que dicha fecha es posterior a la fecha actual, se procederá a abrir un segundo diálogo emergente. En este último, se presentará un texto predeterminado que será utilizado como cuerpo del correo electrónico que se enviará a los alumnos asociados a la encuesta o encuestas que se deseen abrir.

El contenido del texto puede mantenerse intacto o ser modificado para personalizar el mensaje de apertura según sea necesario. Al pulsar el botón de aceptar en este segundo cuadro de diálogo, se procederá a la apertura de la o las encuestas hasta el plazo indicado. Junto con la apertura, se efectuará automáticamente el envío del correo electrónico a todos los alumnos asociados, conteniendo como cuerpo del correo el texto previamente definido en el diálogo emergente.

Finalmente, es relevante resaltar que cuando un administrador abre una encuesta, el docente correspondiente podrá visualizar que dicha encuesta ha sido abierta a través de su menú de encuestas actuales. Por otra parte, si un docente decide abrir una encuesta, dicha acción se reflejará en la página de administración, donde la encuesta se mostrará como abierta. No obstante, la diferencia radica en que un administrador posee la capacidad de cerrar una encuesta que fue abierta por un docente, mientras que la inversa no es aplicable; es decir, un docente no puede cerrar una encuesta abierta por un administrador.

6.9.1.1. Sprint backlog

Las tareas que se deben llevar a cabo para la implementación del sprint son las siguientes:

Back-end:

- Implementación de un nuevo módulo en la capa de coordinación destinado a gestionar las rutas relacionadas con la administración.
- Definición de las rutas pertinentes para obtener el conjunto de encuestas que cumplan los filtros de selección.
- Definición de los métodos necesarios en “dbQuerys.js’.
- Definición de mecanismos de seguridad para evitar abrir una encuesta abierta, evitar que el docente cierre una encuesta abierta por administración, etc.

Front-end:

- Creación y desarrollo de los componentes de las encuestas para administración.
- Configuración de un servicio que consuma la API del back-end para obtener el conjunto de encuestas válidas del docente.
- Creación dinámica de componentes de encuestas basada en datos del back-end.

- Generación de la lógica de la paginación y división del contenido en paginas.
- Creación e implementación de servicios para que los poder llevar a cabo selecciones de encuestas múltiples y poder aplicar los filtros deseados.

6.9.2. Implementación

Back-end:

El primer paso implica la creación de un nuevo módulo en la capa de coordinación dedicado a las solicitudes para la administración. Este módulo se denominará "admins.js", y será el encargado de manejar todas las rutas que comiencen con la cabecera "/admin/".

Dentro de este módulo, se definirán las rutas que estarán habilitadas para la API, y solo podrán ser utilizadas por usuarios autenticados con el rol de administrador. Para asegurar el acceso exclusivo de administradores a estas rutas, se implementará un verificador de token, similar al utilizado para alumnos y docentes. El *middleware* verificador del token, además de validar el token, realizará una comprobación adicional para verificar que el rol asociado sea efectivamente el de un administrador. De esta manera, se garantizará que únicamente los usuarios con rol de administrador puedan acceder a las funcionalidades habilitadas en estas rutas.

El primer paso consiste en implementar una ruta que permita obtener los **distintos** años académicos ("21/23", "22/23"...) asociados a las campañas existentes. Es fundamental conocer que para cada año de un curso, en principio, existen cuatro campañas: dos de grado y dos de máster para por cada cuatrimestre. Por tanto, el objetivo es evitar obtener múltiples veces un mismo año y en su lugar obtener una única instancia. El propósito es obtener el listado de todos los años académicos **distintos** asociados a todas las campañas registradas en el sistema. Este listado se usará en el front-end para rellenar uno de los desplegables.

La segunda ruta habilitada se llama "*/admin/getEncuestas*" y es similar a la ruta "*/docente/getEncuestas*" implementada en el sprint 5, utilizada para listar las encuestas asociadas a campañas válidas en el índice de encuestas los docentes. No obstante, la nueva ruta "*/admin/getEncuestas*" recibe dos parámetros adicionales procedentes de la selección realizada por el personal de administración en los desplegables de filtrado. Estos dos parámetros son: el año de la campaña para la cual se desean mostrar las encuestas y el porcentaje máximo de respuestas que pueden tener las encuestas listadas. Cabe mencionar que esta llamada también obtiene las encuestas asociadas a campañas que han expirado.

Con estos datos, la capa de comunicación llama a la capa de datos, concretamente al método *getEncuestasValidasAdmin()*, al cual se le pasan los dos parámetros recibidos desde el front-end. El resultado obtenido es una estructura similar a la obtenida al realizar la llamada al método *getEncuestasValidasDocente()* del sprint 5.

En relación con las encuestas listadas, se ha implementado una lógica propia para su funcionamiento. Tanto la lógica como el diseño de estas encuestas son, también, similares a las encuestas de los docentes del sprint 5. Es importante recordar que existen dos tipos posibles de encuestas: las **cerradas** y las **abiertas**. En el caso de las encuestas cerradas, la administración tiene la capacidad de abrirlas, tal y como lo podían hacer los docentes.

Cuando una **encuesta cerrada** es abierta por la administración, se realiza una solicitud al back-end para reflejar esta situación. La apertura de la encuesta se efectúa mediante una petición a la ruta "*/admin/abrirEncuestaAdmin*". A su vez, esta petición invoca el método correspondiente en la capa de datos para establecer una nueva fecha de apertura. Es relevante destacar que, a diferencia del caso de los docentes, no se verifica si la encuesta que se pretende abrir está expirada, dado que la administración debe poder abrir encuestas incluso si la campaña asociada está vencida. En el proceso de apertura, la encuesta se marca como abierta por la administración en lugar de abierta por el docente.

Cuando la administración lleva a cabo la apertura de una encuesta, se desea informar a todos los alumnos que están involucrados en la situación docente asociada, con el propósito de notificarles que pueden completar el formulario correspondiente. Para lograr esto, después de realizar exitosamente la apertura de la encuesta, se activa otro método en el back-end denominado "*mandarMensajeApertura()*", el cual recibe tanto el asunto como el contenido del correo electrónico junto con el listado de situaciones docentes. Este método es responsable de obtener las direcciones de correo electrónico de todos los alumnos asociados a la situación o situaciones, en caso de estar agrupadas, de la encuesta recién abierta. Una vez obtenida la lista de correos electrónicos, se procede a enviar masivamente correos, uno a cada alumno.

Para habilitar este envío, se utiliza el módulo *Nodemailer* en conjunto con la configuración para el envío vía Gmail. *Nodemailer* es una biblioteca para aplicaciones de *Node* que facilita el envío de correos electrónicos. Para poder usarlo, se debe definir el remitente, destinatario, asunto y contenido para cada correo electrónico a ser enviado. Además, por motivos de seguridad, se recomienda evitar utilizar directamente la contraseña actual de la cuenta de Gmail. En su lugar, se ha generado una contraseña específica para esta aplicación con permisos más restrictivos, de tal manera que solo se permite el envío de correos a un destinatario específico.

La implementación para llevar a cabo el cierre de una **encuesta abierta**, independientemente de que haya sido abierta por un docente o administración, es análoga a la utilizada por los docentes en el sprint 5. Para lograrlo, se utiliza la ruta "*/admin/cerrarEncuesta*", la cual se encarga de invocar el método correspondiente en la capa de datos para obtener la apertura actual vigente de dicha encuesta y finalizarla. Cerrar la encuesta implica asignar como fecha de finalización la fecha actual, es decir, la que corresponde al instante exacto en el cual se ha pulsado el botón "Cerrar Encuesta". De esta forma, se cierra la encuesta y se registra la fecha de cierre, evitando que los usuarios puedan continuar respondiendo al formulario asociado después de este punto.

Front-end:

El primer paso consiste en desarrollar dos servicios que faciliten la comunicación entre el componente encargado de listar las encuestas y las propias encuestas listadas. Específicamente, se implementarán dos servicios, "*ListadoEncuestasService*" y "*SelectedEncuestasService*".

El primer servicio llamado "*ListadoEncuestasService*" se utiliza para permitir la comunicación entre componentes y facilitar la actualización del listado de encuestas en el componente principal donde se listan. Cuando se necesita actualizar el listado, dicho componente principal puede llamar al método "*mostrarEncuestasFiltros()*" del servicio, lo que desencadena una recarga de los datos del listado en el componente principal acorde a los filtros seleccionados.

El segundo servicio, el "*SelectedEncuestasService*", se utiliza para mantener un registro de las encuestas seleccionadas en el componente principal. Permite agregar, eliminar, obtener y vaciar las encuestas seleccionadas, lo que es útil para realizar acciones sobre un conjunto de encuestas específicas, como en este caso abrirlas o realizar otras operaciones relacionadas con ellas.

Se ha creado un componente adicional para implementar la lógica y diseño del listado de encuestas para la administración. En primer lugar, se han añadido dos desplegables a dicho componente, como se ha mencionado, estos desplegables se usan para filtrar el listado de encuestas que se desea mostrar. El primer desplegable permitirá seleccionar un porcentaje de respuestas recibidas para las encuestas. El segundo desplegable permitirá seleccionar el año del curso correspondiente, con el propósito de visualizar las encuestas específicas de dicho año. Al seleccionar un valor para ambos desplegables, se habilitará un botón "Mostrar Encuestas", que tras pulsarlo, se procederá a obtener desde el back-end el listado de encuestas que concuerdan con dichos filtros.

Por cada encuesta se generará de manera dinámica un componente que haga referencia a la misma. En lo que respecta a los componentes de las encuestas, son muy similares a los componentes utilizados por los docentes. Estos componentes también se dividen en dos categorías: encuestas abiertas y encuestas cerradas. Tanto la lógica como el diseño de estos dos componentes no presentan puntos destacables o diferencias significativas.

Para abordar el problema de un listado extenso de encuestas, que puede afectar el rendimiento del proceso de generación y manejo de las mismas, se implementará una funcionalidad de paginación para dividir las encuestas en diferentes páginas en función del número máximo de encuestas que se desean visualizar por página. Para lograr esto, tras realizar la llamada para obtener las encuestas después de aplicar los filtros, se calculará el número total de encuestas recuperadas desde el back-end. Una vez que se conoce el número de encuestas y se ha establecido el límite de encuestas por página, se realizará una división para determinar la cantidad de páginas necesarias.

Si el número de encuestas supera el límite por página, se habilitará la paginación en la vista, mostrando un conjunto de botones que representarán cada número de página disponible. Al seleccionar un botón que corresponde a una página específica, se activará

la lógica asociada para cambiar la página y se mostrarán las encuestas asociadas a dicha página. Esta implementación permitirá una mejor organización y manejo de las encuestas, mejorando así la experiencia de usuario y evitando el impacto negativo en el rendimiento debido a un listado extenso.

Es posible que si el límite máximo de encuestas por página es reducido y, al obtener un listado extenso de encuestas desde el back-end, se generen tantas páginas que la paginación resulte saturada. Para mitigar este problema, se ha establecido un número máximo de páginas a mostrar en la paginación. A partir de dicho número, se presentará la paginación en forma de un desplegable (ver Figura 6.20). De esta manera, se evitará la saturación visual de la paginación y se ofrecerá una experiencia más amigable al usuario.

6.20. Figura: Paginación con límite de paginas establecido a 5.

Finalmente, es necesario implementar la lógica asociada al cambio de página cuando se presiona un botón que corresponde a un número de página en particular. El método "cambiarPagina()" es una función del componente que se utiliza para cambiar de página.

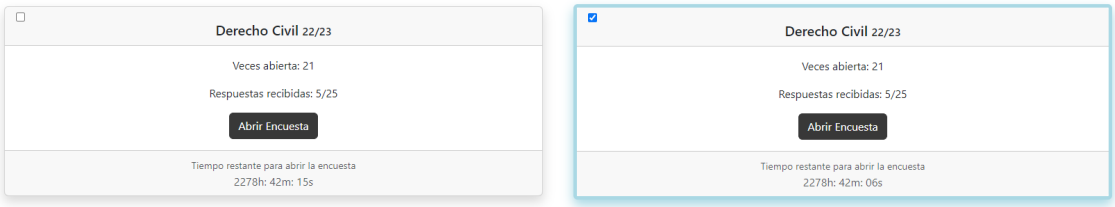
```
cambiarPagina(pagina: number) {
  this.paginaActual = pagina;
  this.mostrarEncuestasFiltros();
  this.selectedEncuestasService.vaciarEncuestas();
}
```

método "*cambiarPagina()*"

Dentro de este método, se actualiza la variable "paginaActual" con el índice de la página que debe mostrarse en la interfaz. Posteriormente, se invoca a la función "mostrarEncuestasFiltros()" para cargar las encuestas correspondientes a la nueva página seleccionada, manteniendo los filtros aplicados previamente. Por último se vacía el listado de encuestas seleccionadas si las hubiera.

En relación a la última funcionalidad, se pretende habilitar la capacidad de seleccionar varias encuestas de manera simultánea, permitiendo así abrirlas todas de manera conjunta. Para lograr esto, se incorporará un elemento *checkbox* al componente de la encuesta.

Cuando el *checkbox* esté marcado, se activará la propiedad "seleccionado" del componente, lo que implicará, en primer lugar, un cambio en el estilo visual para indicar que ha sido seleccionado. En la Figura 6.21 se puede observar la diferencia entre una encuesta estando seleccionada o no.



6.21. Figura: Diferencia entre encuesta no seleccionada y seleccionada.

En el componente "*ListadoEncuestasAdminComponent*", a fin de gestionar las encuestas seleccionadas, se implementará una lista denominada "encuestasSeleccionadas" que contendrá los objetos correspondientes. Si se selecciona una o varias encuestas, el botón "Activar Seleccionadas" será habilitado. Al presionar este botón, todas las encuestas seleccionadas serán abiertas, desencadenando los dos diálogos previos tal como se había realizado previamente. Tras validar los diálogos, se procederá a recorrer la lista "encuestasSeleccionadas' y se invocará el método "*activar()*" de cada encuesta recorrida en el listado de encuestas seleccionadas.

6.9.3. Revisión del sprint

Tras la reunión con el cliente, se confirmó que la implementación de los informes cumplía con los requisitos previamente acordados. Durante esta reunión, también se identificaron cambios y mejoras adicionales, los cuales fueron discutidos y aceptados para garantizar la completa satisfacción del cliente con el resultado final del proyecto.

El primer cambio fue que en el desplegable respectivo al año de la campaña, el lugar de usar el año del curso ("21/22","22/23"...) se desea poder filtrar por campaña. Recordemos que un año consta de cuatro campañas, dos de máster y dos de grado, una de cada por cuatrimestre. Por tanto lo que antes era una opción ahora son cuatro. Hubo que cambiar alguna consulta a la base de datos y algo de la lógica del desplegable. En la siguiente Figura 6.22 se muestra un ejemplo del listado de un conjunto de campañas registradas.

6.22. Figura: Listado del segundo desplegable del componente.

El segundo cambio fue añadir un botón que seleccionase todas las encuestas de la página actual. Dicho botón debía obtener cada encuesta de la página actual y guardarla en la lista de encuestasSeleccionadas, una vez añadidas se debía activar la propiendas *seleccionada* de cada componente. Esta acción consigue el mismo efecto que seleccionar manualmente todas las encuestas de la página.

6.9.4. Pruebas

El propósito de llevar a cabo las pruebas, como siempre, radica en asegurar el correcto funcionamiento de la funcionalidad administrativa recién implementada, abordando de manera minuciosa situaciones críticas para verificar su óptimo funcionamiento.

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
1.1. No hay campañas registradas.	El desplegable asociado al listado de campañas no debería mostrar ninguna opción y por tanto no se debería activar el botón de "Mostrar encuestas".	El desplegable no tiene elementos y por tanto no puede visualizar ninguna campaña (no hay).	Sí
1.2. Existen campañas registradas.	Se listan todas las campañas registradas en el sistema identificadas por el nombre que se le dio al crearlas, acompañado del año al que pertenecen para mayor exactitud.	Se han registrado todas las campañas existentes junto con el año que pertenecen en el desplegable. Al pulsar en una se habilita el segundo desplegable.	Sí
2.1. Aplicación de **filtros** permisivos para listar todas las encuestas del sistema.	Al escoger las opciones de "<=100 %" para el número de respuestas y "Todos los años" para el desplegable de las campañas, se deberían listar todas las encuestas del sistema.	Al escoger filtros tan permisivos, que realmente no filtran nada, se muestran todas las encuestas del sistema.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
2.2. Aplicación de **filtros** tan restrictivos que no se liste ninguna encuesta.	Al seleccionar "<=0 %" para el número de respuestas en una campaña con todas las encuestas completadas, no deberían aparecer encuestas listadas, mostrando en su lugar un mensaje indicativo.	No se lista ninguna encuesta y en su lugar aparece en pantalla el mensaje: "Vaya... Parece que no hay encuestas que cumplan estos filtros."	Sí
3.1. **Paginación**. Obtención de encuestas por debajo del límite por página.	Se deberían listar todas las encuestas que cumplan los filtros establecidos. Al haber menos encuestas que el límite establecido por página solo se deberían habilitar una página y desactivar la paginación.	Se muestran todas las encuestas que coinciden con el filtro establecido y al haber menos del límite de la página se desactiva la paginación.	Sí
3.2. **Paginación**. Obtención de encuestas por encima del límite por página.	Se deberían listar todas las encuestas que cumplan los filtros establecidos. Al haber más encuestas que el límite establecido por página se deberían habilitar varias páginas y activar la paginación en la parte inferior de la pantalla.	Se muestran todas las encuestas que coinciden con el filtro establecido y al haber más del límite de la página se activa la paginación en la parte inferior de la pantalla.	Sí
3.3. **Paginación**. Obtención de encuestas por encima del límite por página y del límite de páginas totales.	Se deberían listar todas las encuestas que cumplan los filtros establecidos. Al haber más encuestas que el límite establecido por página se deberían habilitar varias páginas y activar la paginación en la parte inferior de la pantalla. Cuando se supera el límite de páginas establecido, las páginas adicionales se presentan en un desplegable. (Ver Figura 6.22).	Se muestran todas las encuestas que coinciden con el filtro establecido y al haber más del límite de la página se activa la paginación en la parte inferior de la pantalla. Además, en las páginas posteriores al límite de paginación, se visualizan en un menú desplegable.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
4.1. **Seleccionar** una o varias encuestas.	Al clicar el *cehckbox* de una campaña ésta debería pasar a estar seleccionada (Ver Figura 6.21). Además, se debería introducir en la lista de encuestas seleccionadas.	Al seleccionar una encuesta, su aspecto visual cambia, y al imprimir la lista de encuestas seleccionadas se comprueba que el objeto se ha introducido correctamente.	Sí
4.2. Cambiar de página con encuestas **seleccionadas**.	Al cambiar de página, se debería perder las selecciones realizadas. La lista de encuestas seleccionadas se debería vaciar y al volver a la misma página deben mostrarse como deseleccionadas.	Al cambiar de pagina e imprimir el listado de encuestas seleccionadas, éste se muestra vacío. Al regresar a la página previa, las encuestas previamente seleccionadas se encuentran deseleccionadas.	Sí
4.3. Seleccionar todas las encuestas de la página.	Al hacer clic en el botón que selecciona todas las encuestas, todas deberían ser marcadas, deberían cambiar visualmente, y todos los elementos de las encuestas presentes en la página actual deberían ser agregados a la lista de encuestas seleccionadas.	Al pulsar el botón, todas se muestran como seleccionadas y se comprueba que todos los objetos se añaden a la lista de de encuestas seleccionadas.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
5.1. **Abrir** individualmente una encuesta.	Al pulsar en el botón "Abrir Encuesta" de un componente de encuesta cerrada, deberían saltar el diálogo de la fecha de cierre y el del mensaje de apertura, tras validar ambos, se debería establecer la nueva apertura para la encuesta y refrescar la vista para incluir el componente de encuesta abierta recién actualizado.	Al pulsar el botón de "Abrir Encuesta' en el componente de una encuesta cerrada, tras validar los dos diálogos, se abre la encuesta en el plazo establecido y la vista se refresca para añadir el componente de encuesta abierta.	Sí
5.2. **Cerrar** una encuesta abierta.	Al pulsar en el botón "Cerrar Encuesta" de un componente de encuesta abierta, debería cerrarse de manera correcta.	Igual que con las encuestas de docentesm al pulsar el botón de cierre de una encuesta abierta se pone fin a la apertura actual y se actualiza la vista para que dicho componente pase a ser una encuesta cerrada.	Sí
5.3. **Cerrar** una encuesta abierta por un docente.	Al pulsar en el botón "Cerrar Encuesta" de un componente de encuesta abierta, debería cerrarse de manera correcta, independientemente de quién la haya abierto.	Sin importar quien la haya abierto, al presionar el botón de cierre de una encuesta abierta, se concluye la apertura actual y se actualiza la vista, convirtiendo dicho componente en una encuesta cerrada.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
5.4. Docente intenta **cerrar** una encuesta abierta por administración.	En caso de que un docente intente cerrar una encuesta abierta por administración, no se le concederá la autorización para llevar a cabo dicha acción, y se mostrará un diálogo emergente que indique la situación.	Si el docente pulsa el botón de "Cerrar Encuesta' en el componente de una encuesta abierta por administración, no se le permite llevar a cabo la acción y salta un diálogo emergente indicando que dicha encuesta ha sido abierta por administración.	Sí
5.5. **Abrir** un conjunto de encuestas seleccionadas.	Si se seleccionan varias encuestas de una pagina y se pulsa el botón "Abrir Seleccionadas" se deberían abrir de manera simultanea todas las encuestas seleccionadas.	Tras pulsar el botón "Abrir Seleccionadas" habiendo seleccionado un conjunto de encuestas y tras validar los dos diálogos emergentes asociados a la apertura, todas las encuestas se abren. Tras la apertura se refresca la vista.	Sí
5.6. **Abrir** individualmente una encuesta habiendo varias seleccionadas.	Si se seleccionan varias encuestas de una pagina y se pulsa el botón "Abrir Encuesta" del componente individual de una de las encuestas cerradas seleccionadas, se debería abrir solamente la encuesta a la cual se ha pulsado el botón, independientemente de las que hayan sido seleccionadas.	Con varias encuestas seleccionadas y tras pulsar el botón "Abrir Encuesta" del componente individual de una de las encuestas cerradas seleccionadas, únicamente se abre la encuesta a la cual se ha pulsado el botón, independientemente de las que hayan sido seleccionadas.	Sí

Prueba	Rdo. Esperado	Rdo. Obtenido	¿Exitosa?
5.7. **Envío automático del email** de notificación de apertura a los alumnos implicados.	Tras abrir satisfactoriamente una encuesta, se debería notificar vía email a todos los alumnos asociados a la situación docente de la encuesta recién abierta. Les debería llegar un correo cuyo cuerpo del mensaje sea el introducido por la administración en el diálogo emergente asociado.	El correo electrónico se envía correctamente y es recibido exitosamente por los alumnos pertinentes. El cuerpo del mensaje es el texto introducido por la administración en el diálogo destinado para ello.	Sí
5.8. Modificación del cuerpo del mensaje del **email**.	Al abrir una encuesta se muestra el diálogo asociado al cuerpo del email automático que se va a enviar, si no se modifica se enviará dicha plantilla pero si el administrador escribe algo o modifica dicha plantilla es ese texto el que debe ser adjuntado como cuerpo del mensaje en lugar de la propia plantilla.	Tras modificar el texto del diálogo emergente asociado al cuerpo del mensaje del email, se comprueba que efectivamente es dicho texto modificado el que se envía en el cuerpo del correo a los alumnos pertinentes de la situación docente asociada.	Sí

7. Conclusiones y trabajo futuro

Después de haber llevado a cabo un proyecto de gran envergadura por primera vez, utilizando tecnologías previamente no empleadas, se hace necesario realizar una autocrítica y reflexión para evaluar exhaustivamente el trabajo realizado y examinar los pasos seguidos durante el desarrollo del proyecto, contrastándolos con la planificación inicial.

Una vez finalizado el proyecto, es fundamental realizar un análisis objetivo y distanciado, permitiendo una valoración imparcial de los resultados obtenidos y de la eficiencia en el uso de las tecnologías empleadas. Esta autocrítica permitirá identificar áreas de mejora, oportunidades de optimización y lecciones aprendidas.

Además, durante este proceso de reflexión, es importante revisar la planificación inicial y compararla con la ejecución efectiva del proyecto. Esto permitirá identificar posibles desviaciones, factores que influyeron en el desarrollo y ajustes que puedan ser implementados en futuros proyectos similares.

La capacidad de aprender de los desafíos enfrentados y de las experiencias adquiridas no solo desarrollará mis habilidades, sino también el enfoque y la eficiencia para proyectos futuros. La autocrítica constructiva y la reflexión objetiva son prácticas fundamentales en el proceso de crecimiento y mejora continua en el ámbito profesional y tecnológico.

7.1. Evaluación de los objetivos

En el apartado 2.2 del Documento de Planificación del Proyecto (DOP), se establecieron objetivos específicos que debían ser cumplidos durante la ejecución del proyecto. Cada objetivo será evaluado individualmente con el propósito de analizar el grado de cumplimiento alcanzado para cada uno de ellos.

Estos objetivos fueron los establecidos inicialmente para el proyecto de SiREnO:

- *Crear un sistema online capaz de sustituir al sistema actual de encuestas físicas de la universidad.*

Tras la implementación de las funcionalidades acordadas de SiREnO, se ha logrado cumplir el objetivo establecido inicialmente. Dicho objetivo consistía en crear un sistema que, aunque no fuera el producto final con todas las funcionalidades desarrolladas, tuviera sentido por sí mismo y pudiera reemplazar el sistema actual de encuestas físicas que se lleva a cabo en la universidad al final de cada cuatrimestre.

Se han desarrollado todas las funcionalidades acordadas en la captura de requisitos del sistema. Estas funcionalidades incluyen la capacidad de definir quién debe responder a cada encuesta de opinión, sobre qué docente y asignatura, así como establecer un plazo programado para responder a las encuestas y analizar los resultados para presentarlos de manera gráfica e informativa, entre otras. Todo esto ha sido realizado con un diseño intuitivo, limpio y orientado a la facilidad de uso para los usuarios.

Adicionalmente, se ha creado una aplicación en el proyecto para implementar la funcionalidad más relevante de la parte de administración. Como resultado, se ha logrado llevar a cabo con éxito todas las funcionalidades esenciales que establecen una base sólida para el proyecto, permitiendo futuras expansiones y completar el sistema en su totalidad.

- *Simplificar el proceso de gestión de las encuestas en la universidad.*

SiREnO ha simplificado la gestión de encuestas en la universidad, haciendo que el proceso sea más accesible, rápido, preciso y confidencial. Al migrar las encuestas al entorno online, los estudiantes, profesores y personal de administración de la universidad ahora pueden acceder a las encuestas desde cualquier lugar y en cualquier momento. Ya no están limitados por horarios específicos o ubicaciones físicas para participar en las encuestas, lo que aumenta la tasa de respuesta y la representatividad de las muestras.

Este nuevo enfoque también ha permitido un ahorro significativo de tiempo y recursos. Con las encuestas en línea, ya no es necesario imprimir grandes cantidades de formularios en papel, distribuirlos manualmente y luego recopilarlos uno por uno. Este proceso solía ser lento y tedioso.

El sistema online también ha mejorado la precisión y reducido la posibilidad de errores en la lectura de las respuestas por la máquina de digitalización. Además, las encuestas en línea ofrecen una mayor velocidad de análisis. Una vez que se completa la encuesta, los resultados se almacenan automáticamente en una base de datos, lo que facilita su procesamiento y análisis.

- *Crear informes de autoevaluación para docentes.*

Una de las características más notables del nuevo sistema es su capacidad para generar informes de autoevaluación destinados a los docentes. Mediante la recopilación de los datos proporcionados por los alumnos, se ha logrado crear un histórico que permite analizar la evolución de las calificaciones y respuestas a lo largo de los años. Esto brinda a los docentes y administradores una visión más completa y detallada de su progreso y áreas de mejora.

Gracias a SiREnO, los docentes pueden contextualizar sus resultados y compararlos con la media de otras asignaturas, el conjunto del centro educativo, los distintos grupos de alumnos y diversas medias de referencia. Esta perspectiva comparativa resulta de gran ayuda para establecer objetivos realistas y fomentar la mejora continua en el rendimiento docente.

Si lo quisieran, los docentes también podrían evaluar su desempeño a lo largo de los años en una asignatura, observando la media obtenida para cada pregunta de las

encuestas repartidas a lo largo de los años. Estos informes personalizados proporcionan al docente una visión detallada del progreso de las valoraciones en su asignatura a lo largo del tiempo, permitiéndole realizar un seguimiento minucioso y evaluar las tendencias en las respuestas a lo largo de los años.

- *Implementar medidas de seguridad y privacidad robustas.*

Con el propósito de asegurar la privacidad, accesibilidad y seguridad en el sistema, se han implementado diversas medidas de protección tanto en el nivel del back-end como en el front-end. Se ha llevado a cabo un minucioso control para determinar qué usuarios tienen acceso a rutas específicas, según sus roles designados.

Esta medida de seguridad ha sido aplicada en el lado del cliente mediante barreras de acceso, y también en el nivel del back-end, donde se asegura el acceso a los recursos mediante un middleware que verifica la validez e identidad del token utilizado. Estas precauciones garantizan un ambiente protegido y confiable para los usuarios del sistema.

Además de las medidas de seguridad previamente explicadas e implementadas durante el desarrollo, se ha asegurado ejecutar una solución que no deje ningún rastro en la base de datos respecto a quién ha respondido y quién no ha respondido a la encuesta, así como el total anonimato de las respuestas individuales. Una vez que el plazo de la encuesta finaliza, todos los registros relacionados con qué alumnos deberían haber respondido a qué encuesta son eliminados de forma irreversible, lo que imposibilita conocer esta información posteriormente.

- *Optimizar el proceso de apertura y cierre de encuestas.*

Es evidente que se ha optimizado de manera significativa el proceso de apertura y cierre de las encuestas. Anteriormente, este proceso implicaba la recopilación en papel de los formularios en la secretaría, los cuales luego eran distribuidos a los alumnos para que los respondieran. Cuando todos los alumnos habían respondido, el delegado debía recoger todos los formularios. Al finalizar la clase, tanto el profesor como el delegado tenían que entregar en secretaría el sobre con la documentación necesaria.

No obstante, gracias a la implementación de SiREnO, se ha eliminado por completo el papeleo y la burocracia asociada a las encuestas. Ahora, el proceso de pasar una encuesta es tan simple como presionar el botón de .ªbrir Encuestaz programar el plazo de apertura. Los alumnos, simplemente, acceden a la encuesta recién abierta, rellenan el formulario correspondiente y envían sus respuestas. El envío y persistencia de las respuestas se realiza de manera automática, sin necesidad de realizar acciones adicionales.

El proceso de cierre de la encuesta también ha sido agilizado. La encuesta finaliza automáticamente al terminar el plazo de apertura establecido, o en caso extraordinario, el docente puede cerrarla manualmente con un solo clic. Estas mejoras han simplificado significativamente la realización de encuestas, ahorrando tiempo valioso tanto para los docentes como para los estudiantes, y eliminando la carga administrativa previamente necesaria.

- *Evaluar y demostrar mis aptitudes técnicas.*

Después de completar el proyecto, he demostrado la habilidad de aplicar los conocimientos adquiridos durante mi formación como ingeniero informático. He exhibido mi capacidad para llevar a cabo un pensamiento crítico al capturar los requisitos del cliente y analizar y diseñar las funcionalidades del sistema en consonancia con sus necesidades.

Además, he logrado crear un sistema integral desde cero, a pesar de no tener experiencia previa con las tecnologías que iba a utilizar. Este sistema ha sido desarrollado de manera coherente, con alta modularidad y preparado para futuras expansiones, facilitando su ampliación en el futuro. Asimismo, he demostrado mis aptitudes para generar interfaces de usuario que presenten la información necesaria de forma atractiva, garantizando la usabilidad e intuición en su manejo.

Por último, y desde una perspectiva personal, considero especialmente relevante destacar mi capacidad para adaptarme de manera eficiente a un entorno de trabajo notablemente distinto al que estaba acostumbrado. En mi opinión, este logro refleja una de las facetas de la inteligencia, donde la habilidad de adaptación y flexibilidad son fundamentales para enfrentar nuevos desafíos con éxito.

- *Demostrar habilidades de investigación.*

El desarrollo de este sistema ha servido como una valiosa oportunidad para poner a prueba y fortalecer mis habilidades de investigación. Al inicio del proyecto, me enfrenté al desafío de trabajar con tecnologías con las cuales no tenía experiencia previa. Por ende, llevé a cabo una exhaustiva labor de investigación con el objetivo de familiarizarme con el funcionamiento y uso de dichas tecnologías, así como de comprender la creación y estructuración de un proyecto de esta envergadura y alcance.

Durante esta etapa, he dedicado tiempo a estudiar cursos y consultar diversas guías o manuales que me proporcionaron una base sólida previa a la implementación del sistema. Sin embargo, el verdadero aprendizaje se ha producido a lo largo del proceso de desarrollo. Durante la implementación de SiREnO, he adquirido valiosos conocimientos y habilidades prácticas al enfrentar los desafíos y obstáculos que han surgido, lo cual ha sido enriquecedor para mi crecimiento profesional.

- *Satisfacción personal y autorrealización.*

En lo personal, me siento altamente satisfecho con el desempeño y logro del objetivo alcanzado tras toda la dedicación y esfuerzo realizado. Durante el desarrollo del proyecto, enfrenté diversos desafíos y momentos difíciles que me generaron desmotivación, incluso llegando a contemplar la posibilidad de abandonar el proyecto. Sin embargo, mantuve mi perseverancia y continué avanzando, enfocándome en satisfacer los requisitos implícitos del sistema y cumplir con las expectativas de las partes interesadas.

Gracias a esta perseverancia y enfoque, logré finalizar exitosamente el proyecto, obteniendo resultados satisfactorios en cada etapa y generando un alto nivel de satisfacción personal. El haber llevado a cabo todas las funcionalidades acordadas de manera exitosa y además haber sido capaz de realizar una ampliación extra del trabajo, me genera un sentimiento de autorrealización muy grande.

7.2. Planificación estimada vs real

Estimación vs. Realidad de las tareas.		
Tarea	**Estimada**	**Real**
Preparación pre proyecto.	**71 horas**	**39.5 horas**
Reuniones preliminares con el tutor.	6 horas	5.5 horas
Consultar documentación y proyectos similares.	5 horas	3 horas
Definición de tareas y objetivos.	6 horas	3 horas
Refrescar conocimientos sobre creación de diagramas.	10 horas	5 horas
Seleccionar las herramientas de trabajo.	4 horas	3 horas
Aprendizaje de las nuevas tecnologías.	40 horas	20 horas
Documentación.	**82 horas**	**147.5 horas**
Elaborar el Documento de Objetivos del Proyecto.	40 horas	39.5 horas
Redacción de la memoria del trabajo.	55 horas	108 horas
Captura de requisitos.	**61.5 horas**	**50 horas**
Diagrama de casos de uso y jerarquía de actores.	6.5 horas	5.5 horas
Modelo de dominio.	10 horas	7 horas
Modelo de dominio a BBDD.	10 horas	8 horas
Diagrama de casos de uso extendidos.	15 horas	12 horas
Diagrama de secuencia.	20 horas	17.5 horas
Tiempo total empleado.	**214.5 horas**	**237 horas**

7.1. Tabla: Duración estimada vs real de las tareas planificadas.

En la tabla previa, se han desglosado los tiempos estimados y reales de las tareas pertenecientes a los tres primeros bloques del Diagrama de Estructura de Trabajo (EDT). Es importante destacar que estas tareas no se encuentran dispuestas en orden cronológico dentro del proyecto. Por ejemplo, la redacción de la memoria del trabajo ha sido llevada a cabo de manera simultánea mientras se implementaban las funcionalidades de los sprints. A continuación, se presenta una comparativa entre los tiempos estimados y los tiempos reales de cada sprint.

En la planificación inicial, se consideró una duración promedio de 50 horas para cada sprint. Luego de realizar los cálculos iniciales de la planificación temporal, se estableció trabajar 3 horas diarias durante los días laborables, desde el inicio del proyecto hasta la fecha de entrega. Esto resultó en un total de 414 horas dedicadas al proyecto, de las cuales 274 se asignaron al desarrollo de todos los sprints y la documentación asociada. Conocida la carga horaria estimada para todos los sprints y la duración media de cada uno, se determinó que sería factible completar 5 sprints en su totalidad. Cada uno de estos sprints representa una funcionalidad del sistema a desarrollar.

A continuación se presenta una tabla que enumera todos los sprints realizados y describe la comparación entre el tiempo estimado y el tiempo real empleado para su ejecución.

Estimación vs. Realidad de las tareas.

Tarea	Estimada	Real
Desarrollo del sistema	**250 horas**	**247.5 horas**
Inicialización del back-end y front-end.	**5 horas**	**7 horas**
Desarrollo del back-end	2.5 horas	2.5 horas
Desarrollo del front-end	2.5 horas	4.5 horas
Sprint 1: Login.	**50 horas**	**32 horas**
Análisis y diseño	7.5 horas	2 horas
Implementación software	27.5 horas	22 horas
Pruebas	15 horas	8 horas

7.2. Tabla: Duración estimada vs real de las tareas planificadas.

Estimación vs. Realidad de las tareas.

Tarea	Estimada	Real
Sprint 2: Listar encuestas a alumnos.	**50 horas**	**20.5 horas**
Análisis y diseño	7.5 horas	2 horas
Implementación software	27.5 horas	12.5 horas
Pruebas	15 horas	6 horas
Sprint 3: Mostrar formulario de encuesta.	**50 horas**	**22 horas**
Análisis y diseño	7.5 horas	1.5 horas
Implementación software	27.5 horas	13.5 horas
Pruebas	15 horas	7 horas
Sprint 4: Responder a un formulario.	**50 horas**	**17 horas**
Análisis y diseño	7.5 horas	2 horas
Implementación software	27.5 horas	7 horas
Pruebas	15 horas	8 horas
Sprint 5: Consultar encuestas docente.	**50 horas**	**22 horas**
Análisis y diseño	7.5 horas	2 horas
Implementación software	27.5 horas	13 horas
Pruebas	15 horas	7 horas
Sprint 6: Abrir, programar y cerrar encuestas.	**50 horas**	**31 horas**
Análisis y diseño	7.5 horas	3 horas
Implementación software	27.5 horas	19 horas
Pruebas	15 horas	9 horas

7.3. Tabla: Duración estimada vs real de las tareas planificadas.

Estimación vs. Realidad de las tareas.		
Tarea	**Duración**	**Duración**
Sprint 7: Generación y presentación de informes.	**50 horas**	**55 horas**
Análisis y diseño	7.5 horas	4 horas
Implementación software	27.5 horas	42 horas
Pruebas	15 horas	9 horas
Sprint 8: Gestión de encuestas para admins.	**50 horas**	**48 horas**
Análisis y diseño	7.5 horas	4 horas
Implementación software	27.5 horas	35 horas
Pruebas	15 horas	9 horas

7.4. Tabla: Duración estimada vs real de las tareas planificadas.

En cuanto a la planificación y ejecución del tiempo de los sprints, hay varias observaciones interesantes. En primer lugar, las horas estimadas para el desarrollo coinciden casi exactamente con las horas reales. Se había previsto emplear alrededor de 250 horas en la implementación de los sprints, y finalmente se utilizaron 247.5 horas para el desarrollo de los sprints + 7 horas previas para inicializar el back-end y front-end.

Esta estimación se basó en la planificación de realizar 5 sprints, cada uno con una duración estimada de 50 horas. No obstante, en la práctica, se han alcanzado **8 sprints completos**, con un promedio de duración de 31.8 horas por cada uno. Esto resulta en un ratio de duración de sprints por número de sprints notablemente mejorado en comparación con la estimación inicial.

Otro aspecto relevante es la distribución de horas entre las distintas fases del desarrollo. En la planificación inicial, se estimó que el análisis y diseño ocuparían el 15 %, el desarrollo el 55 %, y las pruebas el 35 %. Ahora, es necesario contrastar esos porcentajes con las horas reales dedicadas a cada fase para evaluar su correspondencia. Estos cálculos se pueden observar en la tabla 7.5.

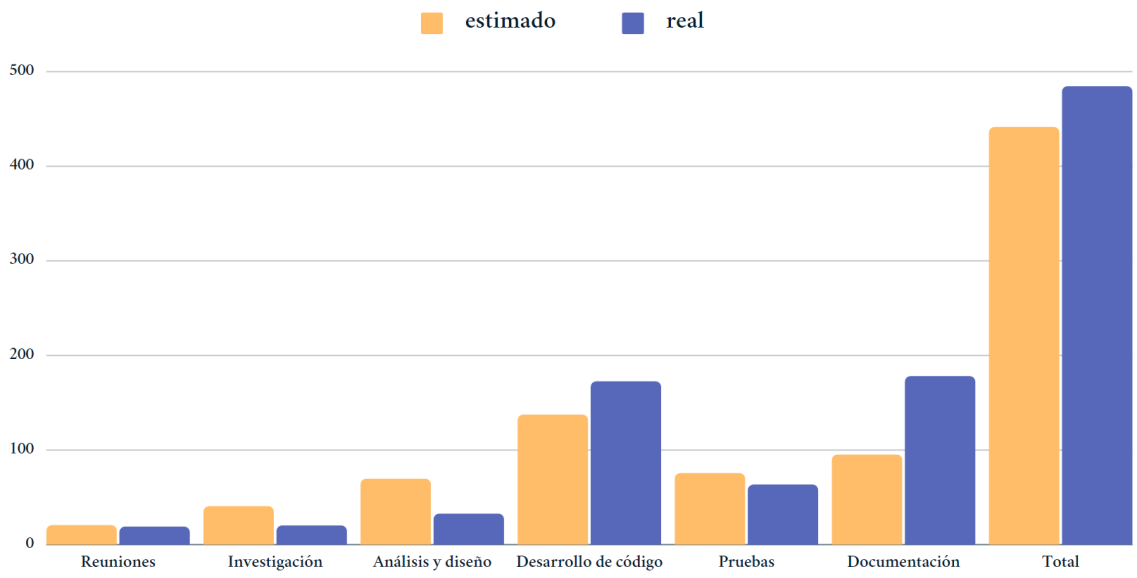
Estimación vs. Realidad de las tareas.		
Tarea	**Estimada**	**Real**
Fases de desarrollo: Desglose.	**250 horas**	**247.5 horas**
Análisis y diseño	37.5 horas **(15 %)**	20.5 horas **(8 %)**
Implementación software	137.5 horas **(55 %)**	164 horas **(67 %)**
Pruebas	75 horas **(30 %)**	63 horas **(25 %)**

7.5. Tabla: Carga estimada vs real de las cada fase del desarrollo.

Luego de realizar la comparativa entre las horas planificadas a cada fase del desarrollo y las horas dedicadas, se evidencia un aumento significativo en el porcentaje de tiempo dedicado a la implementación del software, superando con creces a las demás fases. Asimismo, se observa que el tiempo empleado en el análisis y diseño de cada funcionalidad es aproximadamente la mitad de lo planeado originalmente. Por otro lado, el tiempo destinado a las pruebas es muy similar entre lo planificado y lo realizado en la práctica.

Es relevante resaltar que todos estos datos han sido recopilados de manera progresiva a medida que se avanzaba en el proyecto. La duración de cada tarea, tanto en la fase de implementación como en la etapa previa o en la documentación, ha sido registrada diariamente después de llevar a cabo el trabajo correspondiente en el día.

Para efectuar la última comparación entre las horas estimadas y reales en el desarrollo del proyecto, se muestra la Figura 7.1.



7.1. Figura: Gráfica comparativa de estimación vs. realidad de horas por tareas.

En la Figura 7.1 se muestra la relación entre las horas estimadas y las reales del proyecto. En general, la duración de cada bloque de trabajo ha sido estimada correctamente. Se ha invertido ligeramente menos tiempo en las fases previas al desarrollo, pero sin descuidarlas, cumpliendo con el tiempo necesario; la estimación tenía un exceso de horas. El desarrollo ha tomado más tiempo del esperado debido a la decisión de realizar una ampliación en el proyecto. En lugar de los 5 sprints estimados inicialmente, se completaron 8 sprints en total.

La mayor discrepancia entre la planificación y la realidad se ha observado en el desarrollo de la documentación. Se ha dedicado casi el doble del tiempo previsto. Esta dedicación es comprensible, ya que la documentación es la única representación de mi trabajo antes de la defensa y su cuidado es crucial. En términos generales, el proyecto ha tomado aproximadamente 50 horas más de lo planificado. Sin embargo, el desarrollo ha sido bastante irregular en el tiempo, ya que no se mantuvo una constancia diaria de 3 horas debido a asuntos personales y sobretodo, por la carga del último cuatrimestre de la carrera. No obstante, se recuperó el tiempo perdido trabajando horas extras después de finalizar el curso, ya que había más disponibilidad y tiempo para dedicar al proyecto.

7.3. Identificación y mitigación de riesgos

Durante el transcurso del proyecto, se han manifestado diversos riesgos identificados en el apartado 2.7. A continuación, se detallarán las medidas preventivas implementadas para evitar aquellos que no llegaron a materializarse. Posteriormente, se procederá a listar y explicar en detalle todos los riesgos que finalmente se manifestaron, así como las estrategias adoptadas en el plan de contingencia para mitigarlos al máximo posible.

- *Conocimiento nulo o escaso de cierta tecnología.*

La aparición de este riego era evidente, lo que activó la implementación del plan de prevención establecido inicialmente. Este plan se centró en la documentación exhaustiva y la investigación profunda de las nuevas tecnologías a ser empleadas, ya sea mediante la revisión de documentación técnica o la participación en cursos pertinentes.

- *Elección incorrecta de las tecnologías.*

A pesar de haber implementado el plan de prevención del riesgo previo, me encontré con dificultades en una tecnología específica, no debido a falta de conocimiento, sino a su adaptabilidad al proyecto. Aunque *MongoDB* resultó ser más adecuado en algunos aspectos, gracias a su lenguaje *JavaScript* y su mejor integración con el *stack MEAN*, la complejidad de la base de datos y la gran cantidad de relaciones entre tablas (ver Figura 5.1) hicieron que una base de dato relacional, como *MySQL*, fuera más apropiada. Este tipo de bases de datos son reconocidas por su eficiente gestión de las relaciones entre tablas y la garantía de integridad referencial.

Además, debido a la naturaleza del sistema de encuestas, se han requerido consultas complejas que implican la interacción de múltiples tablas y operaciones de agregación. *MySQL*, como base de datos relacional, ofrece una potente capacidad para manejar este tipo de consultas mediante el lenguaje *SQL*, lo que facilita la extracción y análisis de datos.

- *Requisitos del sistema no tenidos en cuenta.*

Para este riesgo se llevo a cabo el plan de prevención de manera exitosa. ya que, en primer lugar, se realizaron numerosas reuniones con el cliente para capturar cuidadosamente los requisitos del sistema. Además, con el objetivo de evitar incumplir los requisitos y arrastrar fallos durante el proyecto, se establecieron reuniones periódicas tras la implementación de cada sprint. Estas reuniones periódicas han sido fundamentales para identificar y corregir cualquier discrepancia, cambio o ampliación necesaria en el desarrollo del proyecto. Gracias a esta comunicación efectiva con el cliente, se ha logrado mantener un enfoque más preciso y se ha asegurado la alineación del producto final con las expectativas del cliente.

- *Cálculo erróneo en la planificación temporal.*

El cálculo inicial de la planificación temporal se llevó a cabo de manera precisa; sin embargo, el cumplimiento de dicha planificación ha presentado dificultades. La presencia de un parón en el proyecto obligó a una posterior reevaluación de la planificación temporal. Como consecuencia de este parón, se redujo significativamente el tiempo disponible para completar el trabajo, lo que resultó en una distribución de la carga de trabajo a lo largo de los días muy distante de las 3 horas diarias previstas en los días laborables. Durante los dos últimos meses, fue necesario trabajar entre 6 y 8 horas al día para cumplir con los objetivos establecidos.

- *Riesgos de ámbito personal.*

En cuanto a los riesgos de ámbito personal, se presentaron dos situaciones. La primera, la indisposición del desarrollador debido a una carga intensa de exámenes y trabajos. A pesar de llevar a cabo el plan de prevención, que implicaba planificar con antelación los exámenes y trabajos del último cuatrimestre de la carrera para evitar interferencias con el TFG, surgieron semanas con alta carga académica que no se pudieron compaginar. Para abordar esta situación, se optó por aplicar el plan de contingencia que trataba de posponer temporalmente el TFG para priorizar las actividades académicas urgentes y luego reajustar la planificación temporal para retomar el desarrollo del TFG de forma adecuada.

La segunda situación se refiere a la desmotivación experimentada con el proyecto. Después de terminar con la carga adicional de trabajo mencionada en el párrafo anterior, me encontré con un proyecto aplazado de gran envergadura y prácticamente sin haber comenzado la implementación, lo que generó un bloqueo mental completo. Incluso consideré la posibilidad de reducir las funcionalidades a implementar o hasta abandonar el proyecto por completo. Para abordar esta situación, me comuniqué con mi tutor para informarle sobre mi estado y él me ayudó dándome apoyo y consejos. Finalmente, logré superar el bloqueo mental y tecnológico, lo que me permitió desarrollar exitosamente todo el proyecto.

7.4. Lineas futuras

En relación al futuro de SiREnO, la principal tarea sería completar el sistema mediante la incorporación de las restantes funcionalidades detalladas en la captura de requisitos del proyecto, tal como se menciona en el apartado 4. Cabe destacar que todas las funcionalidades relacionadas con alumnos y docentes están implementadas en su totalidad tanto en el back-end como en el front-end, pero si se quisiera, se ha dejado preparado para añadir alguna funcionalidad extra a alguno de estos dos roles.

Por otra parte, aún se encuentran pendientes de desarrollo las funcionalidades de administración, que generalmente involucran operaciones de gestión destinadas a insertar, modificar o eliminar elementos en la base de datos. Estas operaciones, conocidas como altas, bajas y modificaciones, son comunes en aplicaciones de este tipo. Por ejemplo, será necesario implementar la capacidad de definir nuevas encuestas, campañas, así como la adición de usuarios, asignaturas, centros, grados, entre otros elementos en el sistema. Estas funcionalidades suelen ser relativamente sencillas y comparten muchas similitudes en su implementación.

Adicionalmente, se requerirá trabajar en la integración con *Moodle* para facilitar el acceso al sistema desde esta plataforma. Esta integración permitirá a los usuarios interactuar con SiREnO de manera más cómoda y eficiente, aprovechando las ventajas y herramientas que *Moodle* ofrece.

7.5. Reflexión personal

Una vez concluido todo el trabajo y al observar el proyecto en su estado final, operativo y funcional, tengo una profunda sensación de autorealización. El haber realizado un proyecto que podría potencialmente ser de utilidad para la universidad en el futuro, en lugar de quedar en el olvido como un trabajo más, me brinda una gran satisfacción personal. Como mencioné al explicar las razones para seleccionar este proyecto para mi Trabajo de Fin de Grado, uno de mis motivos fundamentales fue crear algo con aplicaciones prácticas.

Este proyecto me ha permitido adquirir una amplia gama de conocimientos técnicos sobre tecnologías y metodologías de desarrollo de software que previamente desconocía. Ha enriquecido mi comprensión y enfoque sobre el proceso de desarrollo, brindándome una perspectiva distinta. Además, ha fomentado mi capacidad de investigación en lo que respecta al descubrimiento y aprendizaje de nuevos conocimientos que antes me eran ajenos.

Aunque no será el más extenso en mi experiencia profesional, si que ha sido ha sido el más extenso que he abordado hasta la fecha y además de forma individual. Este trabajo me ha proporcionado una valiosa oportunidad para adquirir un mayor entendimiento de cómo abordar las diversas etapas de un proyecto de esta escala. Me ha permitido comprender la importancia de llevar a cabo una planificación realista y la relevancia de realizar un análisis y diseño exhaustivos antes de emprender la ejecución, ya que ha demostrado ser crucial para evitar cambios posteriores que retrasen la entrega del proyecto.

A pesar de los objetivos logrados, es importante destacar que el desarrollo del proyecto no fue precisamente un camino de rosas. Como mencioné en el apartado previo, durante el transcurso me encontré con un bloqueo significativo, tanto a nivel tecnológico como mental. Esta situación se originó debido a la necesidad de posponer el desarrollo para atender otras fuentes de trabajo previas. Al concluir esas responsabilidades adicionales, estaba agotado mentalmente y me encontré con el desafío de abordar este proyecto en un período considerablemente más reducido del inicialmente planificado. Además, debía familiarizarme con tecnologías y herramientas nuevas para mí, lo cual generó una sensación de abrumo que, lamentablemente, me llevó a dudar de mi capacidad para llevar a cabo el proyecto; una percepción que, me demostré a mi mismo que era errónea.

Para superar este bloqueo mental que me llevaba a pensar de forma continuada que no era capaz de completar el proyecto, decidí sentarme en mi silla y desarrollar la primera funcionalidad del proyecto en lugar de sobreanalizarla. Esta decisión resultó beneficiosa, ya que el bloqueo mental estaba siendo causado por una nube excesiva de pensamientos. Lo que realmente necesitaba era enfrentar el problema de manera práctica y demostrarme a mí mismo que era capaz de superarlo. Gracias a esta estrategia, superé mi bloqueo mental y recuperé la confianza en mis habilidades para llevar a cabo el proyecto.

Para finalizar, quiero dejar una nota que mencionaba un profesor de economía que tuve, que se refiere a la "Décima y última ley de liderazgo de Shackleton: Nunca abandone, siempre hay otro movimiento." Esta cita resuena en mi experiencia, ya que muestra la importancia de la perseverancia y la determinación en la consecución de los objetivos, y es precisamente esa actitud la que me ha llevado a sentirme plenamente realizado con este proyecto.

8. Bibliografía

1. Universidad del País Vasco / Euskal Herriko Unibertsitatea. (*2022*). *DATOS GENERALES UPV/EHU*. Recuperado de `https://www.ehu.eus/es/web/gardentasun-ataria/datu-orokorrak`

2. Ofiterra. (*2023*). *Coste mayorista de papel a4*. Recuperado de `https://ofiterra.es/mayorista-de-papel-a4-Madrid`

3. Reprotel Copistería. (*2023*). *Coste mayorista de fotocopias*. Recuperado de `https://www.reprotel.com/categoria/copisteria-lowcost/`

4. Universidad del País Vasco / Euskal Herriko Unibertsitatea. (*2022*). *Datos generales de transparencia UPV/EHU*. Recuperado de `https://www.ehu.eus/es/web/gardentasun-ataria/datu-orokorrak`

5. talent. (*2023*). *Salarios y ofertas de trabajo*. Recuperado de `https://es.talent.com/`

6. glassdoor. (*2023*). *Futuro profesional*. Recuperado de `https://www.glassdoor.es/`

7. jobted. (*2023*). *Ofertas de empleo*. Recuperado de `https://www.jobted.es/`

8. Chartjs. (*2023*). *Chart Samples*. Recuperado de `https://www.chartjs.org/docs/latest/samples/information.html`

9. Kevin Davila. (*2021*). *Canal especializado en desarrollo web*. Recuperado de `https://www.youtube.com/@KevinDavilaDev/featured`

10. Bootstrap. (*2023*). *Bootstrap frontend toolkit*. Recuperado de `https://getbootstrap.com/`

11. Proyectosagiles. (*2021*). *Qué es Scrum*. Recuperado de `https://proyectosagiles.`

org/que-es-scrum/

12. Stack Overflow. (*2023*). *Stack Overflow*. Recuperado de https://stackoverflow.com/

13. npmjs. (*2023*). *jsonwebtoken*. Recuperado de https://www.npmjs.com/package/jsonwebtoken

14. mysql. (*2023*). *MySQL Documentation*. Recuperado de https://dev.mysql.com/doc/

Anexo I: Casos de uso extendidos

El primer anexo se enfocará en llevar a cabo un análisis exhaustivo de los casos de uso mencionados en la documentación. Estos casos de uso extenderán la descripción de las interacciones entre los actores y el sistema en diversas circunstancias, proporcionando una visión más completa de su funcionamiento.

En los casos de uso extendidos, se prestará especial atención a aspectos como las precondiciones y postcondiciones, el flujo de eventos y las interfaces gráficas asociadas a cada caso de uso.

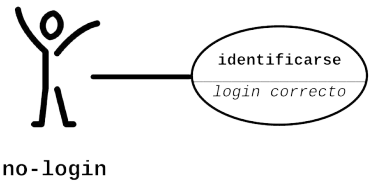
El análisis exhaustivo de los casos de uso extendidos ayudará a garantizar que todos los aspectos relevantes sean considerados y abordados adecuadamente en el diseño y desarrollo del sistema, brindando una base sólida para satisfacer las necesidades de los usuarios y cumplir con los requisitos del negocio.

Estos casos de uso extendidos, al igual que los casos de uso de la documentación, se agruparán en función del actor que los ejecuta.

1. Actor *no-login*

A continuación se muestran en orden los casos de uso extendidos del actor *no-login*.

1.1. Identificarse



Nombre: identificarse

Descripción: el caso de uso de "identificarse" describe el proceso mediante el cual un usuario accede al sistema SiREnO proporcionando sus credenciales de autenticación para obtener acceso autorizado

Actores: no-login

Precondiciones:

– El usuario debe estar registrado en el sistema y tener una cuenta válida.

– El sistema debe estar disponible para procesar las solicitudes.

Requisitos no funcionales:

– La interfaz de inicio de sesión debe ser intuitiva y fácil de usar para los usuarios.
– Las credenciales de inicio de sesión (nombre de usuario y contraseña) deben transmitirse de forma segura a través de una conexión cifrada.

Flujo de Eventos:

1. El usuario entra a SiREnO e inicia el proceso de inicio de sesión (Figura 8.1) ingresando su *username* y contraseña en los campos correspondientes.

[Si las credenciales son válidas]:

[Si el usuario tiene más de un rol]:

2AA. Se le da la opción de elegir un rol (Figura 8.2).

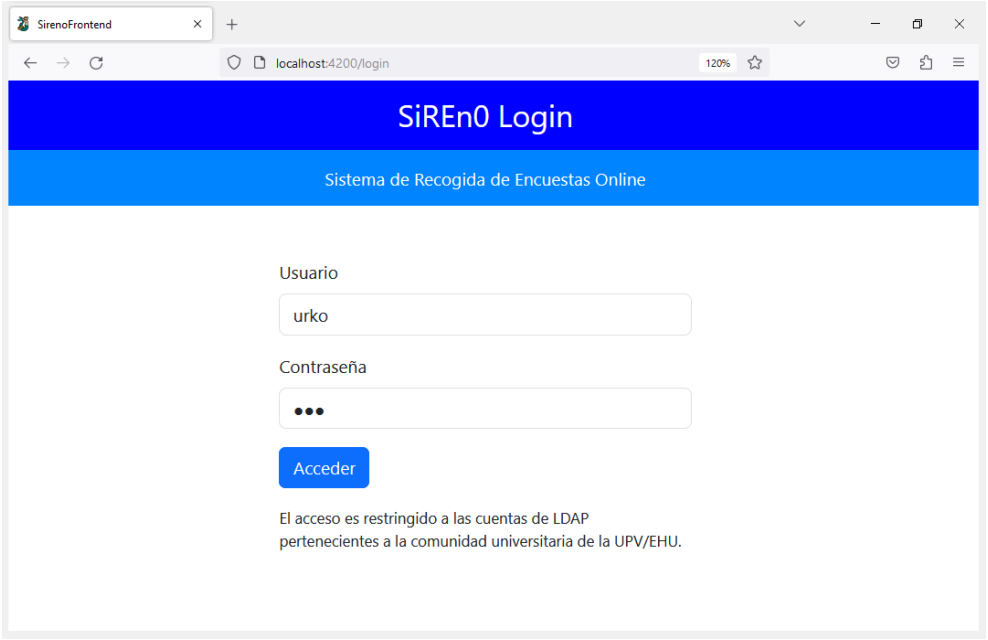
2A. El sistema redirige al usuario a la página correspondiente según su rol.

[Si las credenciales no son válidas]:

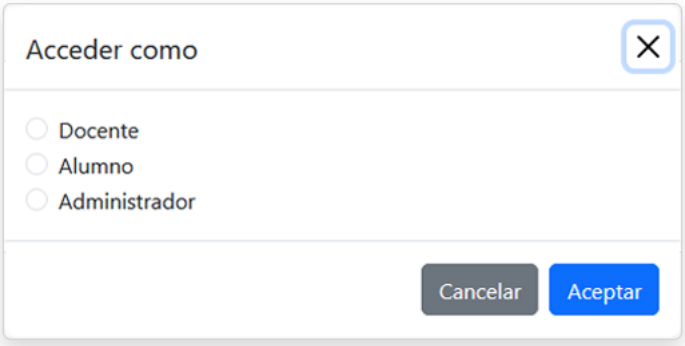
2B. El acceso no es concedido y se muestra un error por pantalla (Figura 8.3).

Postcondiciones:
– El usuario ha iniciado sesión correctamente y tiene acceso autorizado a las funcionalidades y recursos asociados a su cuenta de usuario.

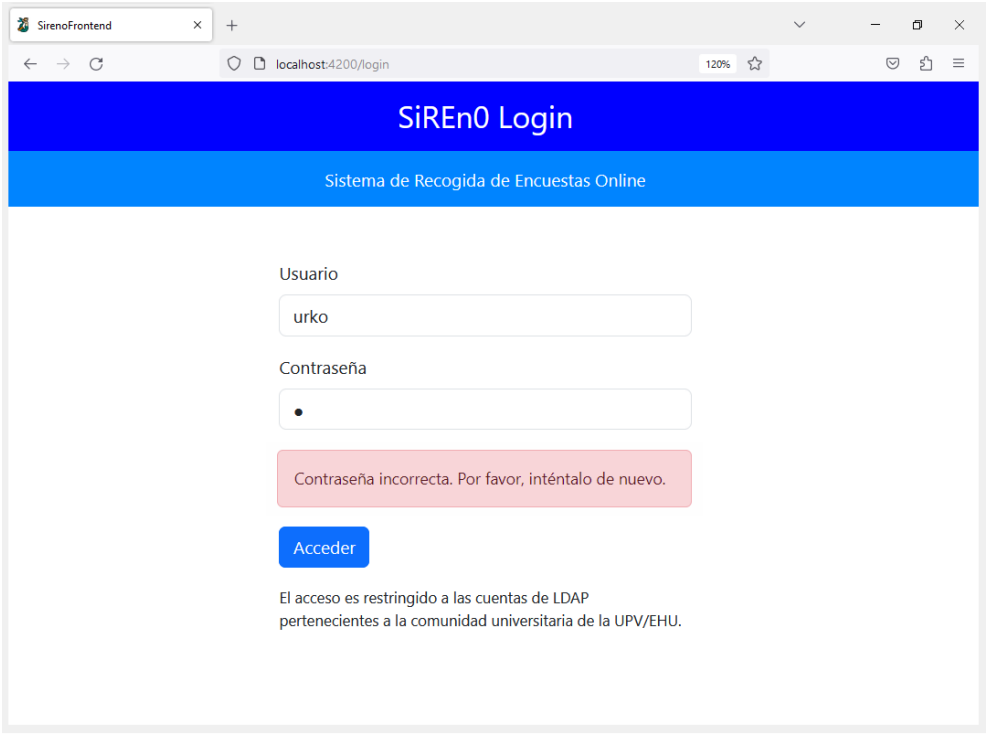
Interfaz gráfica:



8.1. Figura: Formulario de inicio de sesión del front-end.



8.2. Figura: Ventana emergente de elección de roles.

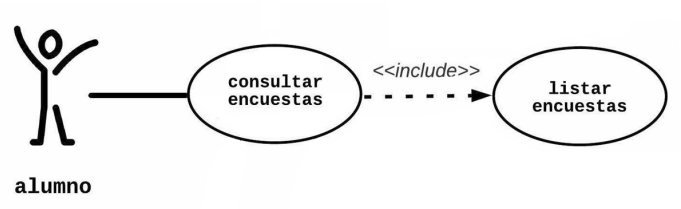


8.3. Figura: Inicio de sesión incorrecto.

2. Actor *alumno*

A continuación se muestran en orden los casos de uso extendidos del actor *alumno*.

2.1. Listar encuestas



Nombre: listar encuestas

Descripción: el caso de uso de "listar encuestas" es el proceso mediante el cual un alumno obtiene el conjunto de encuestas pendientes que tiene por responder en dicho momento.

Actores: alumno

Precondiciones:

– El usuario debe tener una cuenta registrada en el sistema.

– El usuario debe acceder con rol de alumno.

Requisitos no funcionales:

– La lista de encuestas debe mostrarse de manera ordenada y legible para el usuario.

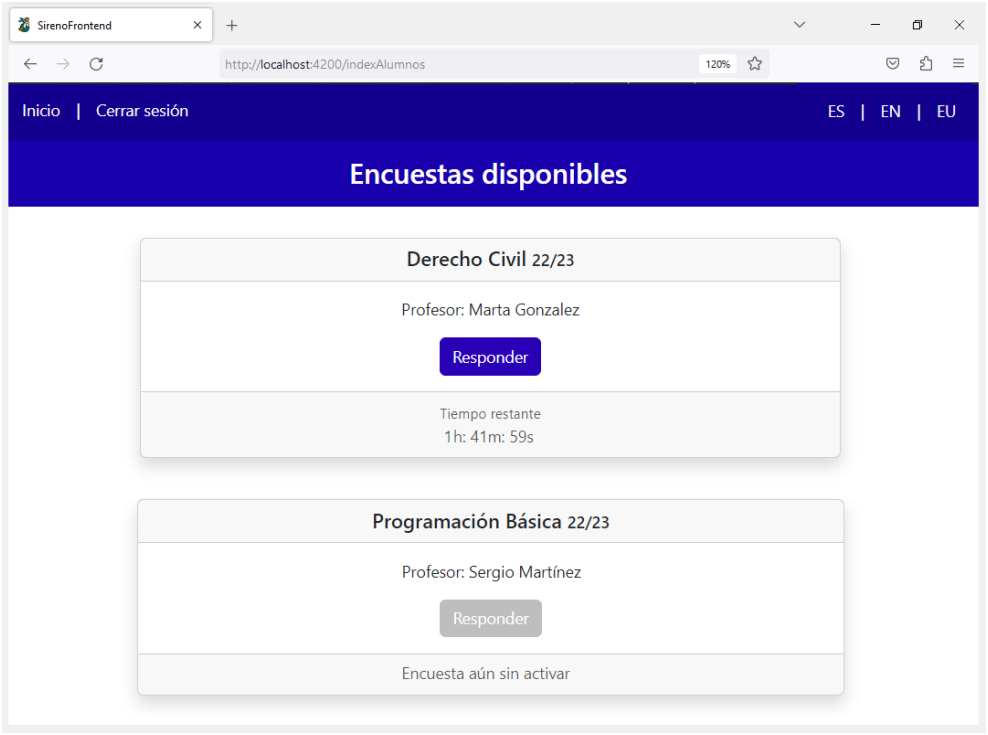
– El acceso a la funcionalidad de listar encuestas debe estar restringido a usuarios autenticados.

Flujo de Eventos:

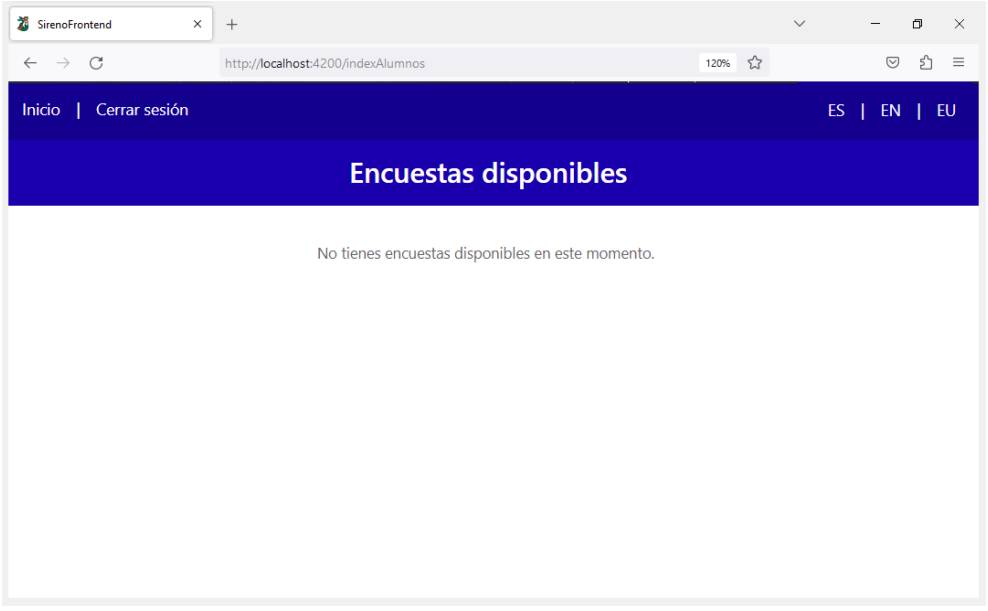
1. El usuario inicia sesión de manera exitosa en el sistema con rol de alumno.
   [Si tiene encuestas disponibles]:
   2A. Se le muestra al alumno el conjunto de encuestas ordenadas según su estado de apertura y la fecha de finalización de la apertura (Figura 8.4).
   [Si no tiene encuestas disponibles]:
   2B. Se le muestra un mensaje indicando la situación (Figura 8.5).

Postcondiciones:
– El usuario puede ver la lista de encuestas disponibles.

Interfaz gráfica:

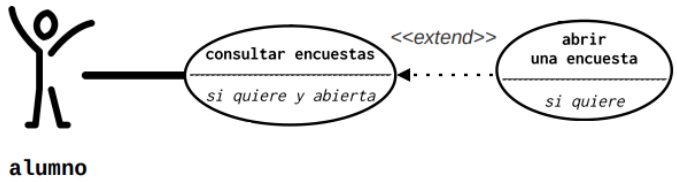


8.4. Figura: Índice de alumnos con encuestas de campañas válidas.



8.5. Figura: Índice de alumnos con sin encuestas.

2.2. Abrir una encuesta



Nombre: abrir una encuesta

Descripción: este caso de uso describe el proceso en el que un alumno puede acceder y visualizar una encuesta asociada a una campaña activa.

Actores: alumno

Precondiciones:

– El alumno debe tener, al menos, una encuesta asociada a una campaña activa.

Requisitos no funcionales:

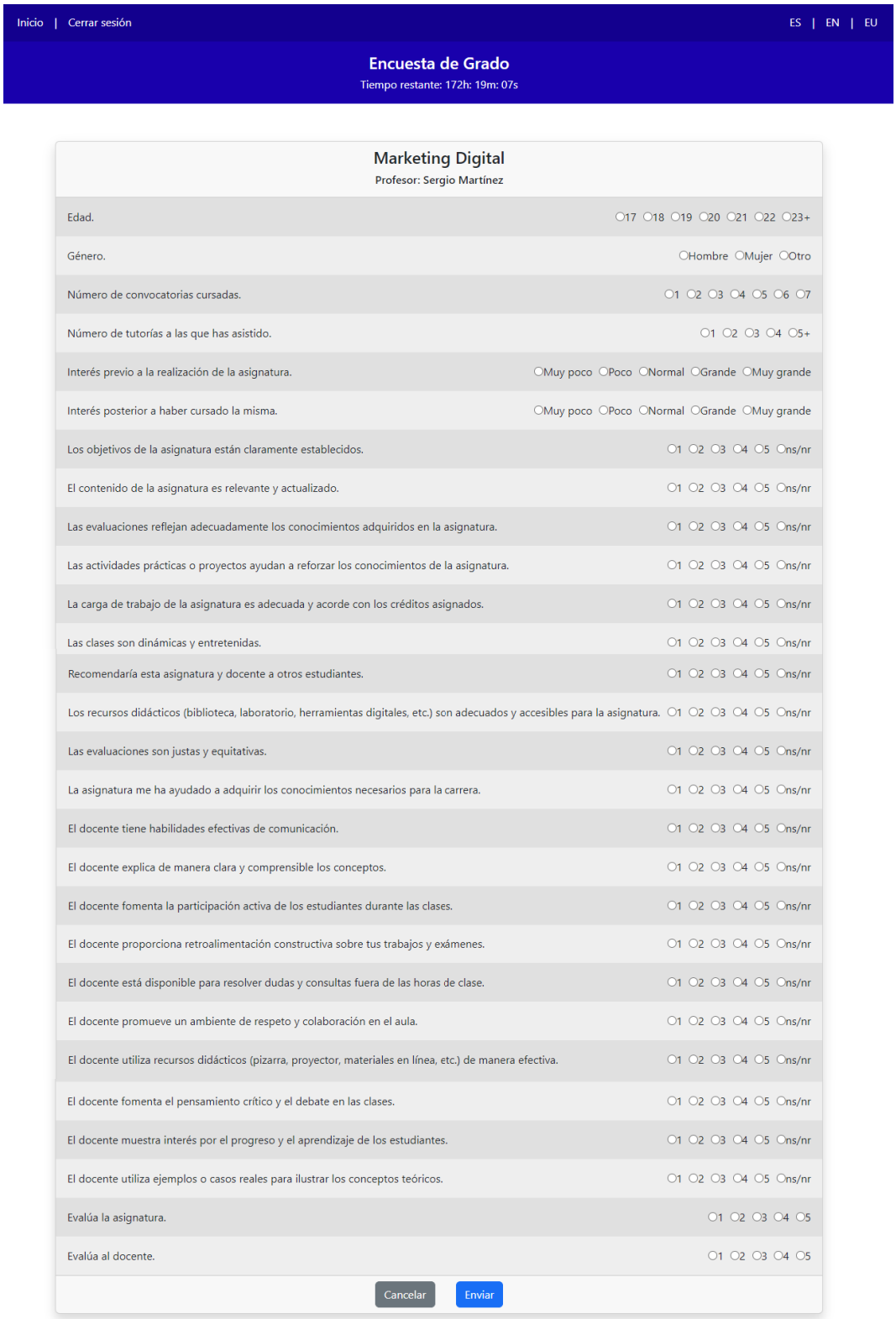
– El acceso a las encuestas están restringidos solo a los usuarios autorizados.
– Se deben manejar las situaciones críticas durante el proceso de apertura y respuesta del formulario (envío con respuestas vacías, cancelación de al encuesta...)

Flujo de Eventos:

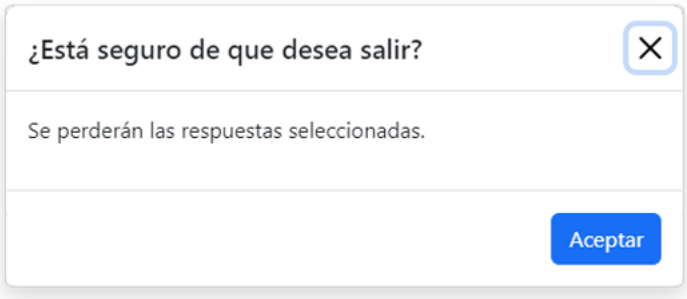
1. El alumno hace clic en el botón "Responder" de una encuesta abierta de su menú principal.
2. Se carga el formulario asociado a la encuesta (Figura 8.6).
   [Si el alumno pulsa el botón "Cancelar"]:
   3A. Se muestra un diálogo de cancelación la encuesta actual (Figura 8.7).
   [Si pulsa "Aceptar"]:
   3AA. El alumno vuelve a su índice principal.
   [Si cancela el diálogo]:
   3AB. El alumno se mantiene en la vista del formulario.
   [Si el alumno pulsa el botón "Enviar"]:
   [Si hay preguntas sin responder]:
   4AA. Se muestra un diálogo indicando la situación (Figura 8.8).
   [Si pulsa el botón "Aceptar"]:
   4AAA. Se valida el formulario.
   [Si no]:
   4AAB. Se mantiene la vista del formulario.
   4AB. Se valida el formulario.

Postcondiciones:
– El alumno ha abierto la encuesta asociada a la campaña seleccionada.

Interfaz gráfica:



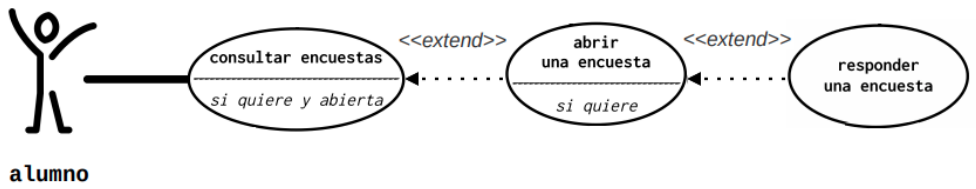
8.6. Figura: Formulario asociado a una encuesta de grado.



8.7. Figura: Diálogo de cancelación del proceso de respuesta.

8.8. Figura: Diálogo de preguntas sin responder.

2.3. Responder una encuesta



Nombre: responder una encuesta

Descripción: este caso de uso describe el proceso en el que un alumno puede enviar las respuestas de un formulario que acaba de responder.

Actores: alumno

Precondiciones:

– El alumno debe tener, al menos, una encuesta abierta.

– La encuesta abierta debe tener un formulario asociado.

Requisitos no funcionales:

– El acceso a las encuestas están restringidos solo a los usuarios autorizados.

– Se deben manejar las situaciones críticas durante el proceso de envío de las respuestas (validación de la situación docente, validación de la campaña...)

Flujo de Eventos:

1. El alumno hace clic en el botón “Enviar” del formulario.

[Si hay preguntas sin responder]:

2A. Se muestra un diálogo indicando la situación (Figura 8.9).

[Si pulsa el botón “Aceptar”]:

2AA. Se envía el formulario.

3AA. Se muestra el diálogo de envío satisfactorio (Figura 8.10).

[Si no]:

2AB. Se mantiene la vista del formulario.

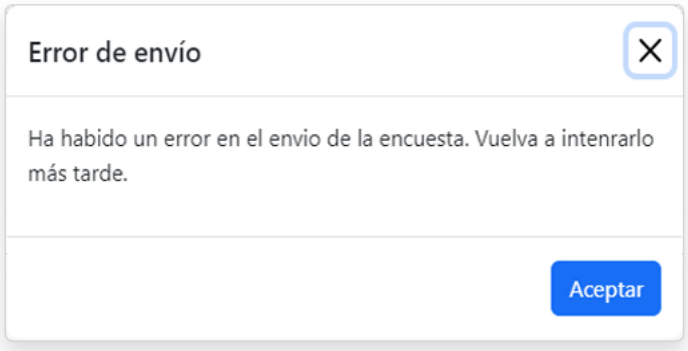
2B. Se envía el formulario.

3B. Se muestra el diálogo de envío satisfactorio (Figura 8.10).

Postcondiciones:
– Las respuestas proporcionadas por el usuario se registran en el sistema y se almacenan para su posterior análisis y procesamiento.
– El usuario recibe una confirmación por haber completado el formulario asociado a la encuesta.

Interfaz gráfica:

8.9. Figura: Diálogo de preguntas sin responder.

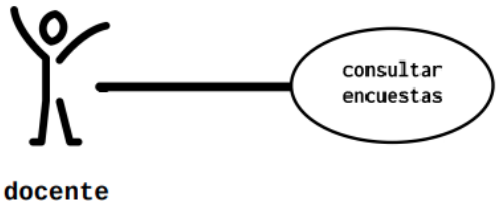


8.10. Figura: Diálogo de envio exitoso del formulario.

3. Actor *docente*

A continuación se muestran en orden los casos de uso extendidos del actor *docente*.

3.1. Consultar encuestas



Nombre: consultar encuestas

Descripción: el caso de uso de "consultar encuestas" es el proceso mediante el cual un docente obtiene el conjunto de encuestas asociadas a campañas válidas en las que figura como docente.

Actores: docente

Precondiciones:

– El usuario debe tener una cuenta registrada en el sistema.
– El usuario debe acceder con rol de docente.
– El docente debe acceder a su página de listado de encuestas.

Requisitos no funcionales:

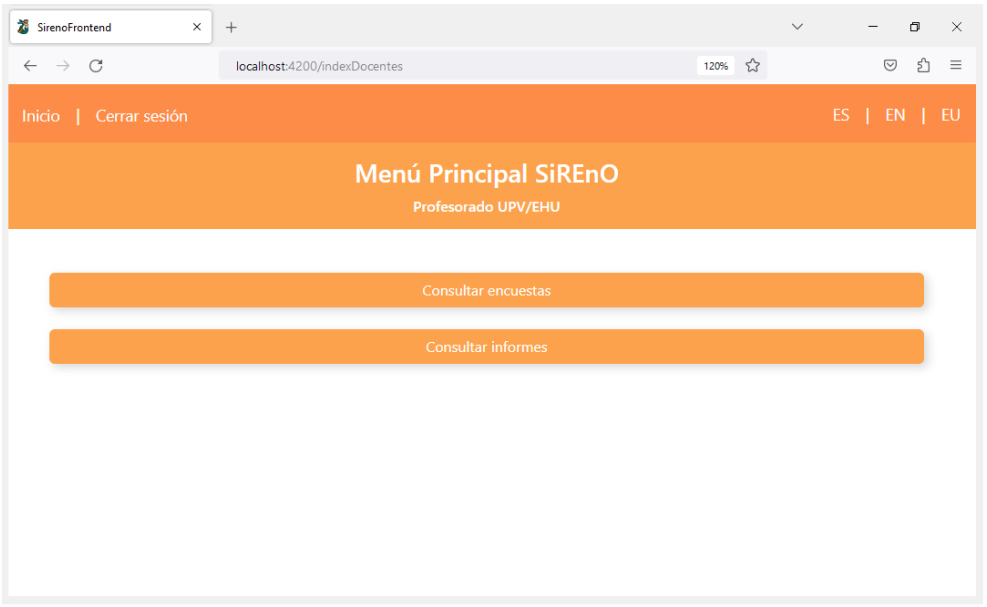
– La lista de encuestas debe estar ordenada y legible para el usuario.
– Solo usuarios autenticados como docente pueden acceder a esta funcionalidad.

Flujo de Eventos:

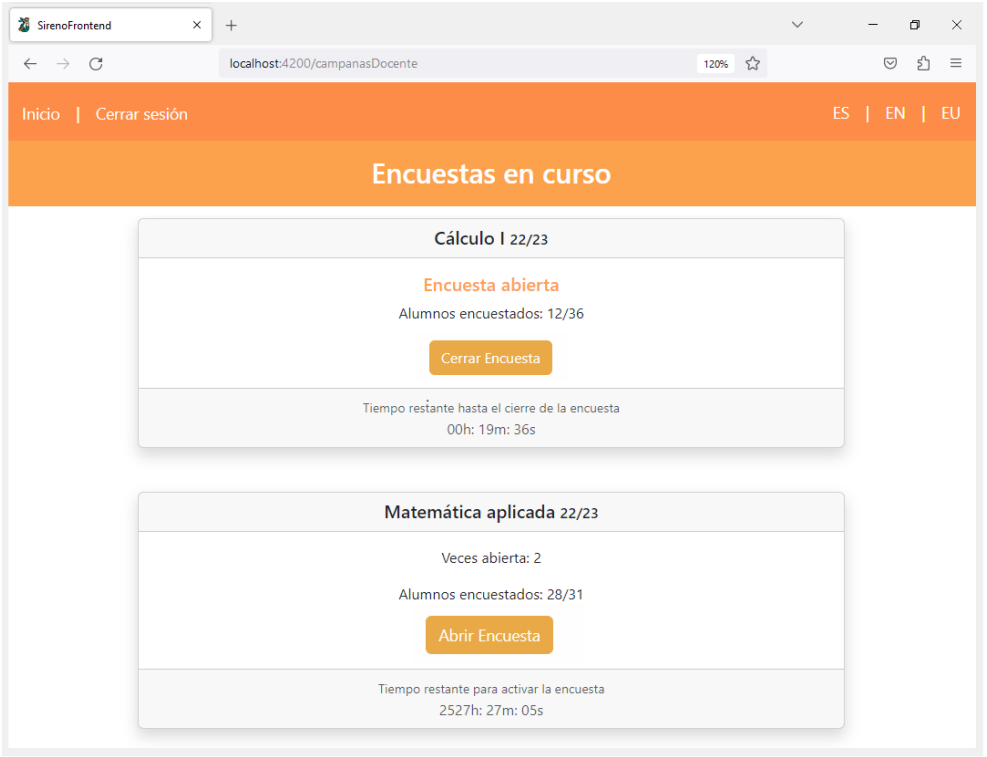
1. El usuario inicia sesión de manera exitosa en el sistema con rol de docente.
2. El docente es redirigido a su menú principal (Figura 8.11).
   [Si hace clic en "Consultar encuestas"]:
   [Si el docente tiene encuestas asociadas a campañas válidas]:
   3AA. Se le muestra al docente el conjunto de encuestas válidas ordenadas por veces abierta y la fecha de finalización de la campaña (Figura 8.13).
   [Si no tiene encuestas disponibles]:
   3AB. Se le muestra un mensaje indicando la situación (Figura 8.5).

Postcondiciones:
– El docente puede ver su lista de encuestas.

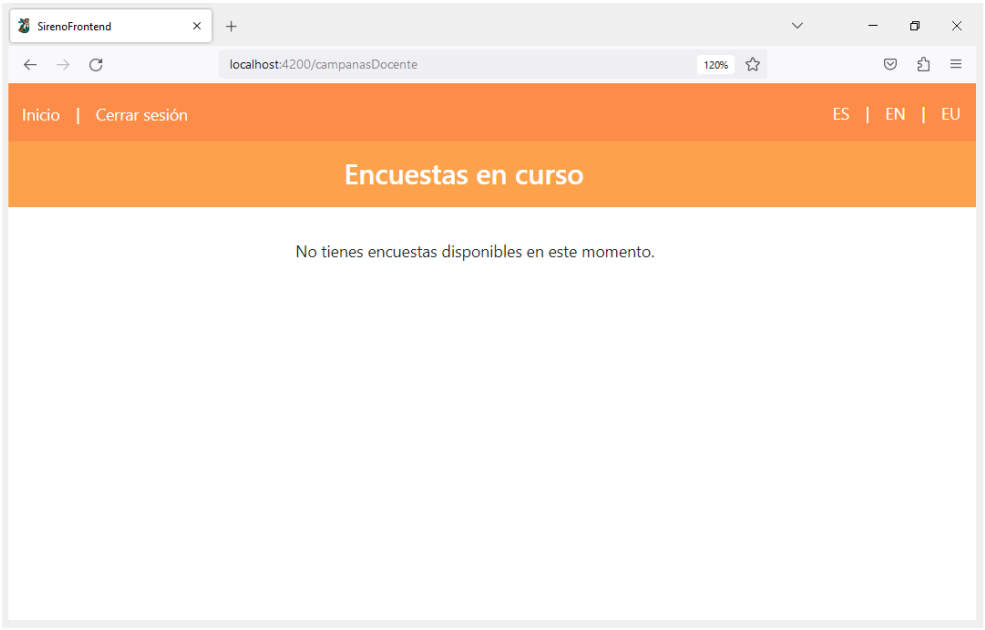
Interfaz gráfica:



8.11. Figura: Index de docentes.

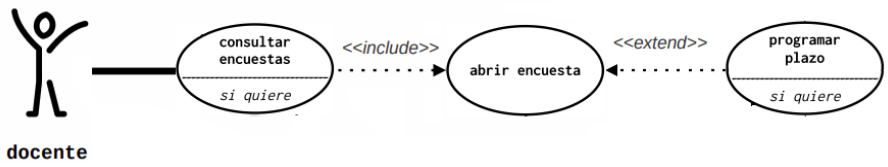


8.12. Figura: Índice de encuestas de docentes con encuestas válidas.



8.13. Figura: Índice de encuestas de docentes sin encuestas.

3.2. Abrir encuesta y programar plazo



Nombre: abrir encuesta y programar plazo

Descripción: este caso describe cómo el docente puede abrir una encuesta de una campaña válida y hacer visible el formulario asociado por un tiempo limitado. Los alumnos podrán responder al formulario de la encuesta durante este periodo y finalizará cuando expire el plazo de apertura.

Actores: docente

Precondiciones:

– El usuario debe acceder con rol de docente.

– El docente debe tener al menos una campaña válida.

Requisitos no funcionales:

– El sistema debe garantizar la integridad y seguridad de los datos.

– El sistema debe manejar varios participantes y respuestas al mismo tiempo.

– Solo usuarios autenticados como docente pueden acceder a esta funcionalidad.

Flujo de Eventos:

1. El usuario inicia sesión de manera exitosa en el sistema con rol de docente.
2. El docente es redirigido a su menú principal (Figura 8.11).

[Si hace clic en "Consultar encuestas"]:

[El docente tiene al menos una campaña válida]:

3AA. Clica en "Activar Encuesta" de una encuesta (Figura 8.14).

4AA. Selecciona una fecha de fin en el *popup* emergente (Figura 8.15).

[La fecha introducida es válida]:

5AAA. Se confirma con otro *popup* emergente que la encuesta ha sido abierta hasta la fecha indicada por el docente (Figura 8.22).

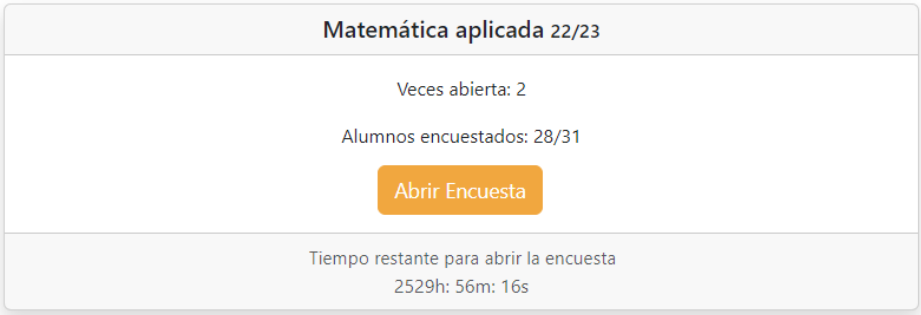
6AAA. La encuesta se muestra como abierta (Figura 8.23).

[Si no]:

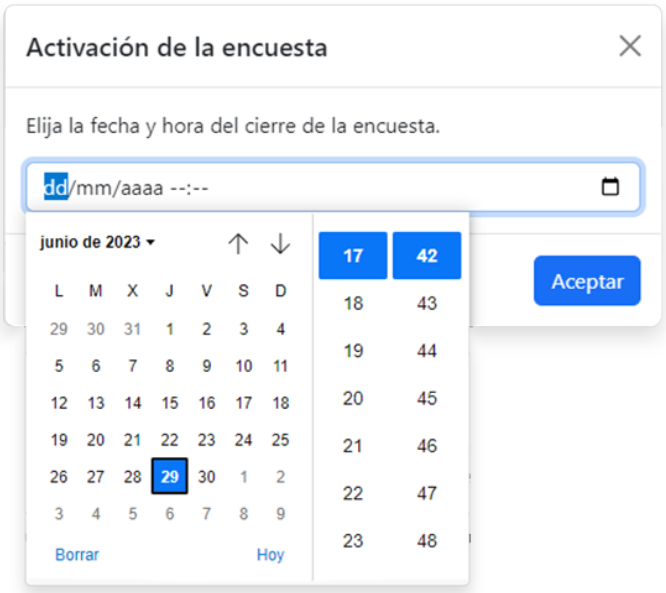
5AAB. Se indica el error mediante un *popup* emergente (Figura 8.24)

Postcondiciones:
– La encuesta se ha abierto hasta la fecha de fin seleccionada.
– Los alumnos ya pueden acceder al formulario asociado y rellenarlo.

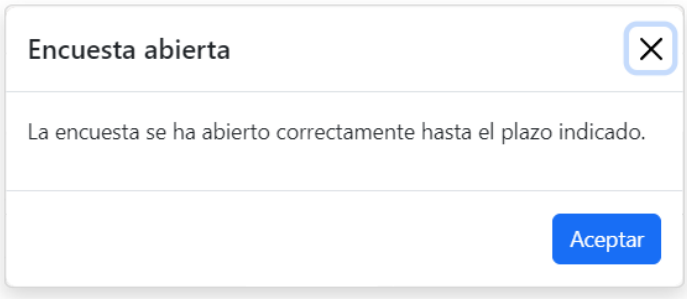
Interfaz gráfica:



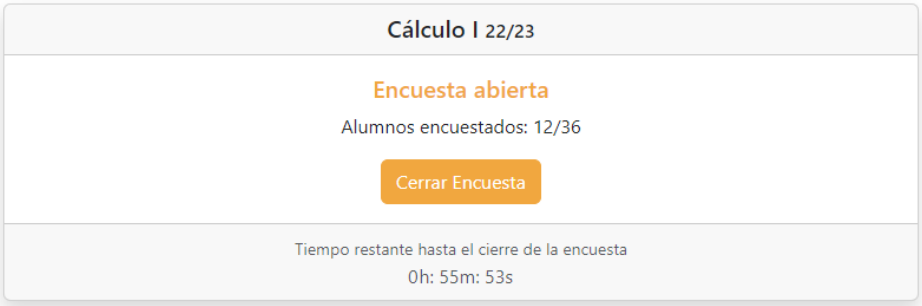
8.14. Figura: Encuesta cerrada asociada a una campaña válida.



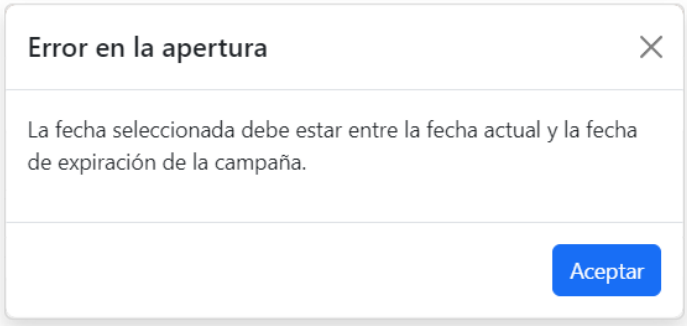
8.15. Figura: *Popup* para seleccionar el fin de la apertura de la encuesta.



8.16. Figura: *Popup* de confirmación de la apertura.

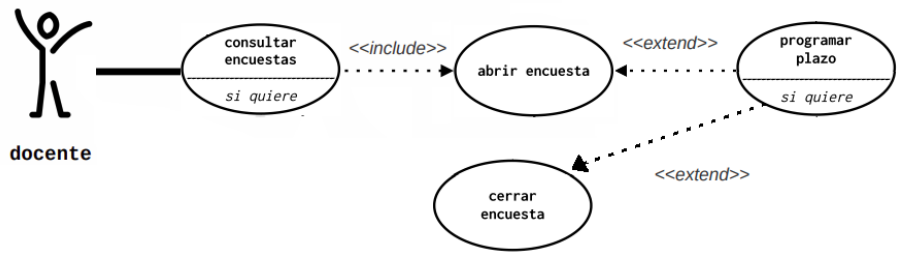


8.17. Figura: Encuesta abierta.



8.18. Figura: *Popup* de aviso de error de la apertura.

3.3. Cerrar encuesta



Nombre: cerrar encuesta

Descripción: este caso describe cómo el docente puede cerrar una encuesta abierta de manera manual antes de que se expire el plazo de apertura. Una vez cierre la encuesta, no se podrá responder al formulario asociado hasta que vuelva a ser abierta.

Actores: docente

Precondiciones:

– El usuario debe acceder con rol de docente.

– El docente debe tener al menos una encuesta abierta.

Requisitos no funcionales:

– El sistema debe verificar que la encuesta seleccionada esté abierta y aún no haya sido cerrada antes de permitir su cierre.

– El sistema debe mostrar mensajes de confirmación tanto antes como después del cierre de la encuesta.

Flujo de Eventos:

1. El usuario inicia sesión de manera exitosa en el sistema con rol de docente.
2. El docente es redirigido a su menú principal (Figura 8.11).

[Si hace clic en "Consultar encuestas"]:

[el docente tiene al menos una encuesta abiert]:

3AA. Clica en "Cerrar Encuesta" de una encuesta abierta (Figura 8.22).

4AA. El docente confirma su decisión mediante un *popup* (Figura 8.19).

[Si el cierre ha sido exitoso]:

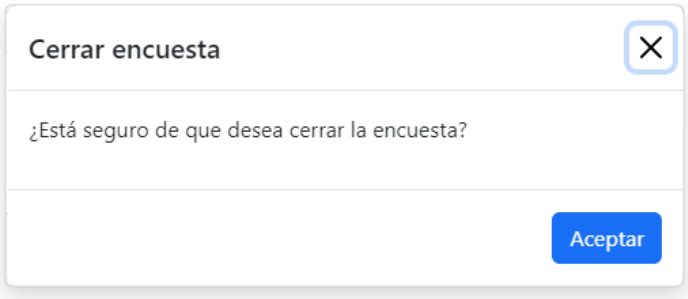
5AAA. Aparece un nuevo *popup* confirmando el cierre (Figura 8.20.

[Si no]:

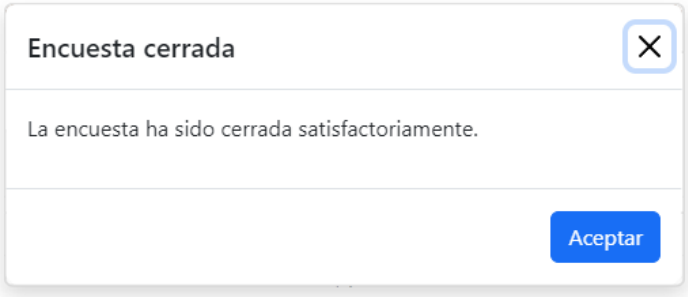
5AAB. Se indica el error mediante un *popup* emergente (Figura 8.21)

Postcondiciones:
– La encuesta seleccionada se marca como cerrada en el sistema.
– Los alumnos ya no pueden acceder al formulario asociado ni enviar respuestas.

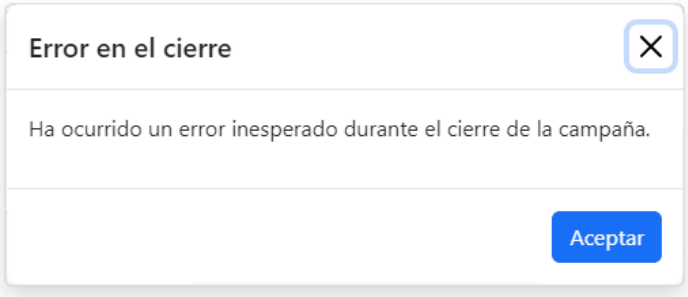
Interfaz gráfica:



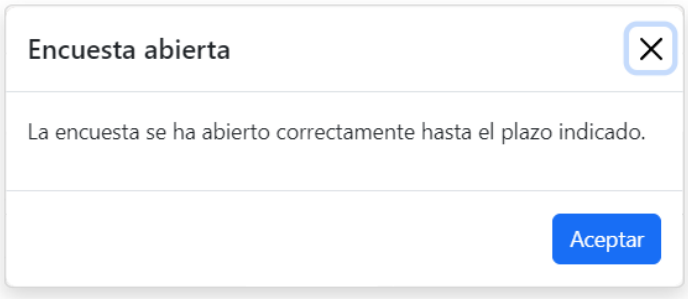
8.19. Figura: *Popup* de confirmación para el cierre de la encuesta.



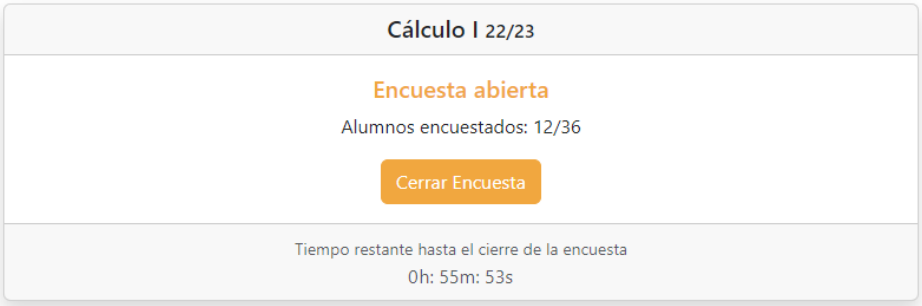
8.20. Figura: *Popup* de cierre satisfactorio de la encuesta.



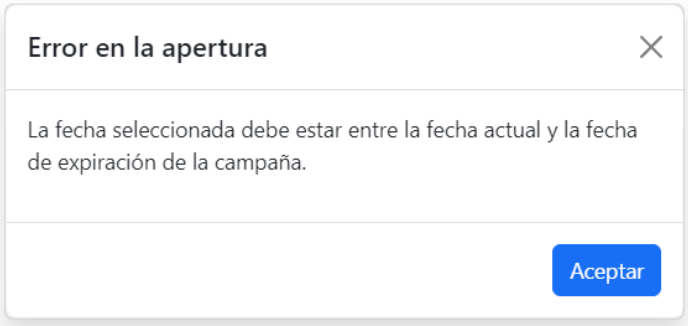
8.21. Figura: *Popup* de error en el cierre de la encuesta.



8.22. Figura: *Popup* de confirmación de la apertura.

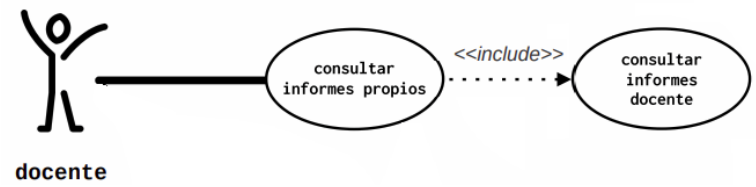


8.23. Figura: Encuesta abieta.



8.24. Figura: *Popup* de aviso de error en la apertura.

3.4. Consultar informe personal



Nombre: cerrar encuesta

Descripción: Este caso describe cómo el docente puede visualizar una serie de informes personales. Los informes personales proporcionan a los docentes la posibilidad de consultar la media de cada pregunta de una asignatura en particular de cualquiera de los años académicos en los que la hayan impartido.

Actores: docente

Precondiciones:

– El usuario debe acceder con rol de docente.
– El docente debe tener al menos un asignatura impartida la cual tiene los resultados de los informes publicados por administración.

Requisitos no funcionales:

– Los datos de los informes deben estar protegidos y solo accesibles por el docente correspondiente.
– La interfaz de usuario del sistema debe ser intuitiva y fácil de usar para que los docentes.

Flujo de Eventos:

1. El usuario inicia sesión de manera exitosa en el sistema con rol de docente.
2. El docente navega hasta la pantalla de informes personales.
3. Escoge una asignatura y año en la que la ha impartido y pulsa en “Consultar Informe”. (Figura 8.25).
4. Se cargan y muestran los datos descriptivos de la situación docente (Figura 8.28).

[Si hay respuestas registradas]:

5A. Se muestran los resultados del informe. De manera porcentual para las preguntas textuales y de manera absoluta junto con la gráfica para las preguntas numéricas. (Ver Figura 6.18 y Figura 6.19).

Flujo de Eventos:

[Si no hay respuestas registradas]:

5B. Se muestra el informe vacío. Se listan las preguntas de la encuesta pero no se muestran resultados. El porcentaje de respuesta es 0 % para las preguntas textuales y la media de cada respuesta numérica se muestra con un guión "-". La gráfica no se genera. (Ver Figura 8.30 y Figura 8.29).

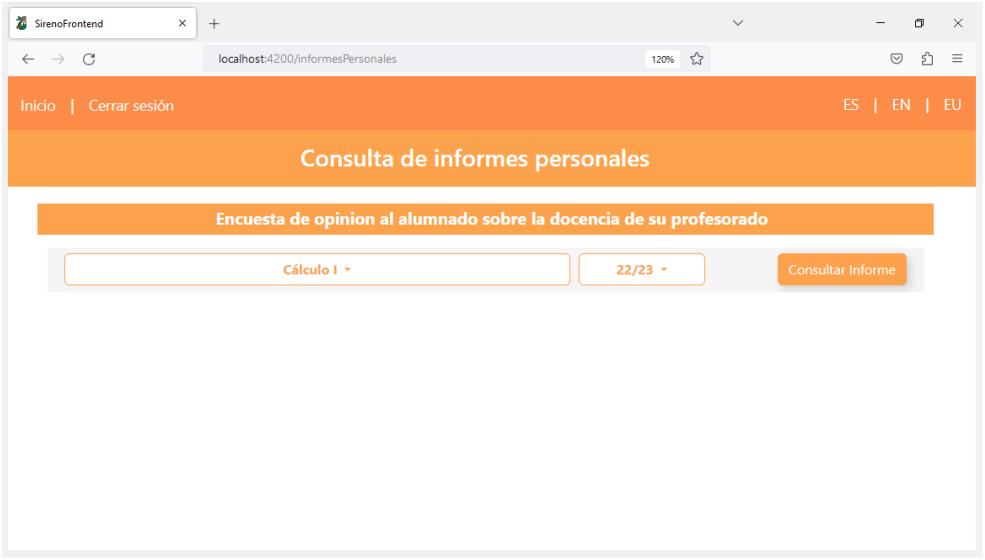
[Si se hace clic en una pregunta]:

6.A. Se abre el gráfico de la respuesta media histórica. (Ver Figura 8.31)

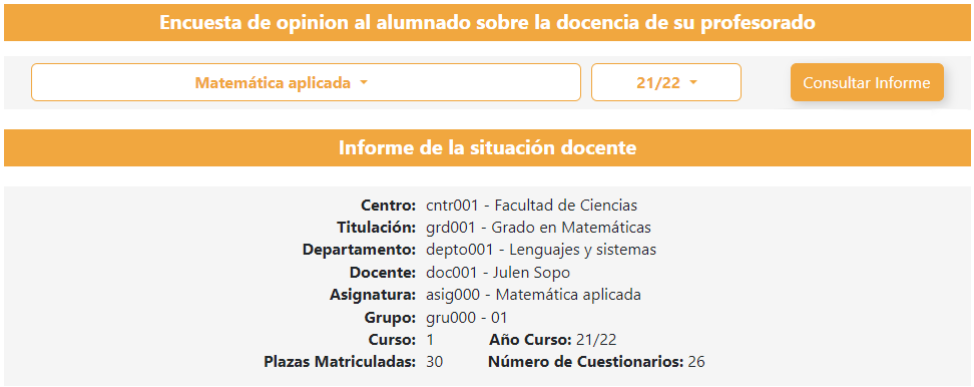
Postcondiciones:

– El docente obtiene informes personalizados basados en los resultados de las encuestas de sus alumnos.

Interfaz gráfica:

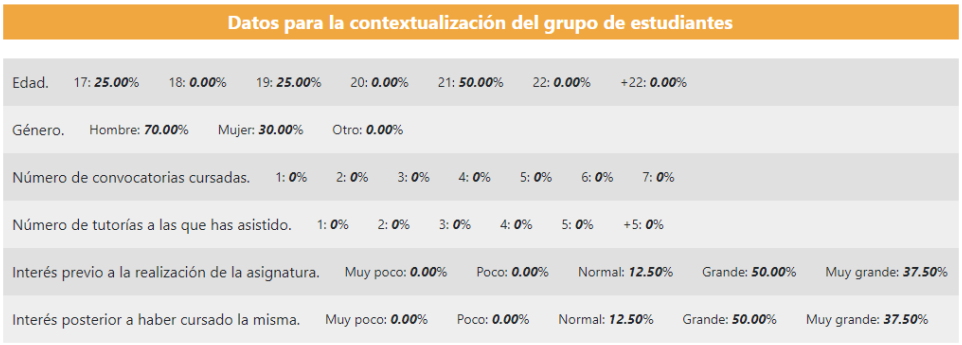


8.25. Figura: Desplegables para seleccionar el informe a mostrar.



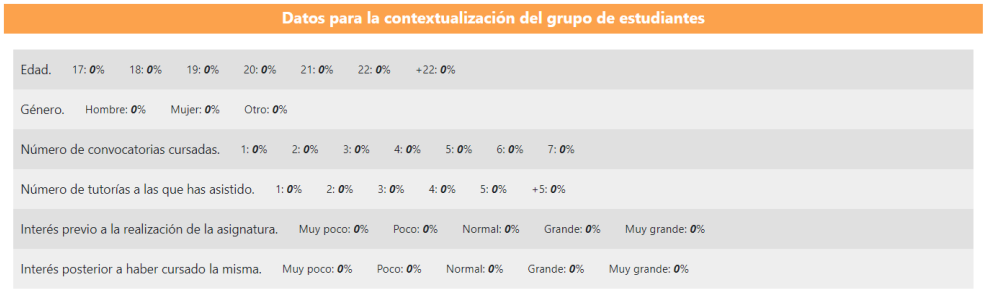
8.26. Figura: Cabecera del informe.

Interfaz gráfica:



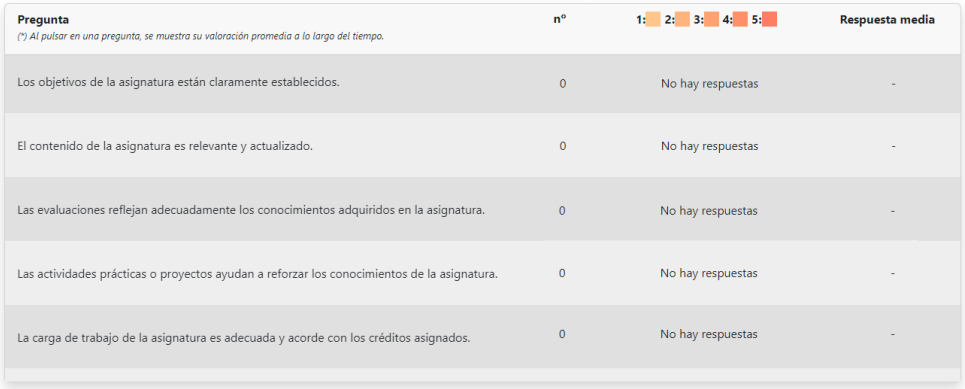
8.27. Figura: Respuestas textuales del informe.

8.28. Figura: Cuerpo del informe.



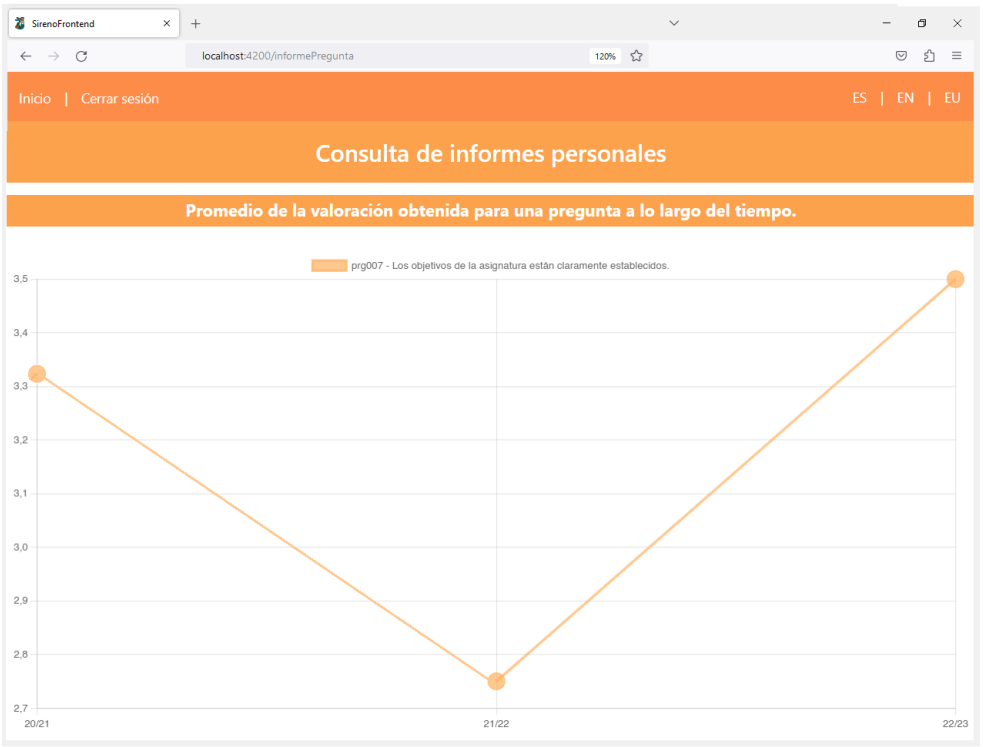
8.29. Figura: Preguntas textuales de un informe sin respuestas.

Interfaz gráfica:



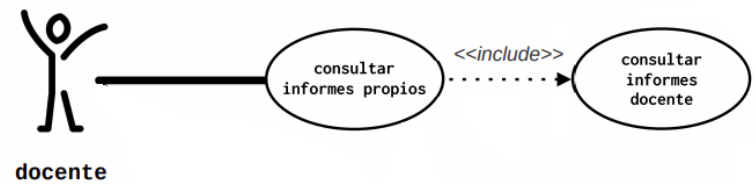
Pregunta *(*) Al pulsar en una pregunta, se muestra su valoración promedia a lo largo del tiempo.*	nº	1: 2: 3: 4: 5:	Respuesta media
Los objetivos de la asignatura están claramente establecidos.	0	No hay respuestas	-
El contenido de la asignatura es relevante y actualizado.	0	No hay respuestas	-
Las evaluaciones reflejan adecuadamente los conocimientos adquiridos en la asignatura.	0	No hay respuestas	-
Las actividades prácticas o proyectos ayudan a reforzar los conocimientos de la asignatura.	0	No hay respuestas	-
La carga de trabajo de la asignatura es adecuada y acorde con los créditos asignados.	0	No hay respuestas	-

8.30. Figura: Cuerpo de un informe sin respuestas.



8.31. Figura: Informe de la nota media histórica para una pregunta.

3.4. Consultar informe comparativo



Nombre: consultar informes docente

Descripción: Este caso describe cómo el docente puede visualizar una serie de informes comparativos. Los informes comparativos proporcionan a los docentes la posibilidad de consultar la media de cada pregunta de una asignatura en particular de cualquiera de los años académicos en los que la hayan impartido.

Actores: docente

Precondiciones:

– El usuario debe acceder con rol de docente.
– El docente debe tener al menos un asignatura impartida la cual tiene los resultados de los informes publicados por administración.

Requisitos no funcionales:

– Los datos de los informes deben estar protegidos y solo accesibles por el docente correspondiente.
– La interfaz de usuario del sistema debe ser intuitiva y fácil de usar para que los docentes.

Flujo de Eventos:

1. El usuario inicia sesión de manera exitosa en el sistema con rol de docente.
2. El docente navega hasta la pantalla de informes comparativos.
3. Escoge una asignatura, año en la que la ha impartido, una medida de comparación y pulsa en “Consultar Informe”. (Figura 8.32).
4. Se cargan y muestran los datos descriptivos de la situación docente (Figura 8.28).

[Si hay respuestas registradas]:

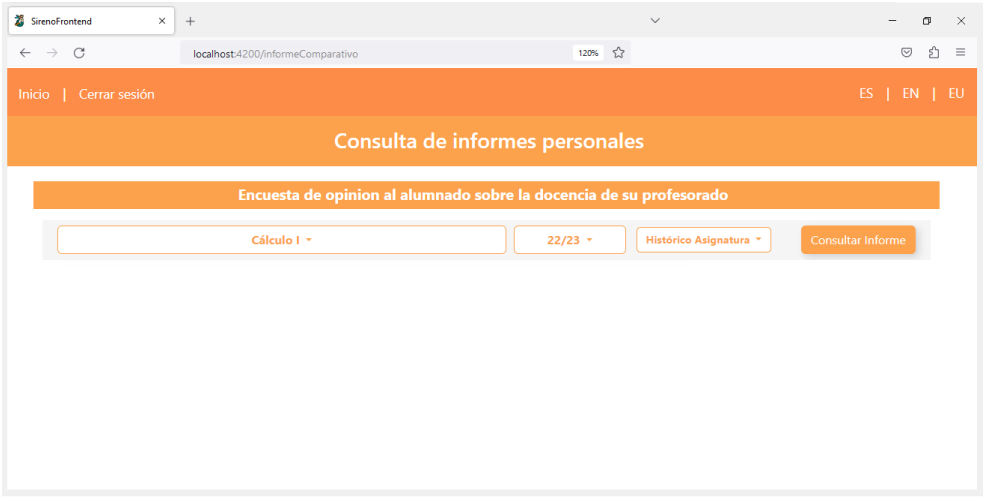
5A. Se muestran los resultados del informe junto con las comparativas. (Ver Figura 6.17).

[Si no hay respuestas registradas]:

5B. Se muestran el informe vacío. Se listan las preguntas de la encuesta

Postcondiciones:
– El docente obtiene informes comparativos basados en los resultados de las encuestas de sus alumnos.

Interfaz gráfica:



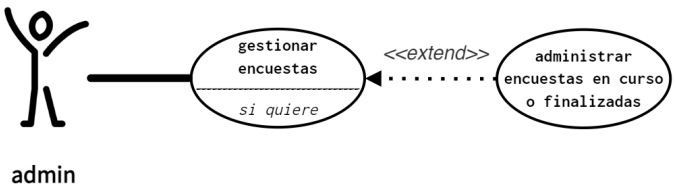
8.32. Figura: Desplegables para seleccionar el informe a mostrar.

8.33. Figura: Cuerpo del informe comparativo.

3. Actor *admin*

A continuación se muestran en orden los casos de uso extendidos del actor *admin*.

4.1. Gestionar encuestas: abrir, programar y cerrar



Nombre: gestionar encuestas

Descripción: Este caso de uso detalla las acciones de gestión de encuestas en curso o finalizadas que el administrador del sistema puede realizar en el subsistema de SiREnO.

Actores: admin

Precondiciones:

– El usuario debe tener una cuenta registrada en el sistema.
– El usuario debe acceder con rol de admin.
– El administrador accede al listado y filtra el listado de encuestas.

Requisitos no funcionales:

– La interfaz del subsistema de administración de SiREnO debe ser intuitiva y de fácil navegación para facilitar el uso eficiente por parte del Administrador.
– El tiempo de respuesta del sistema al realizar filtrados y reactivaciones debe ser óptimo para garantizar una experiencia de usuario ágil y sin demoras significativas.

Flujo de Eventos:

1. El usuario inicia sesión de manera exitosa en el sistema con rol de admin. (Figura 8.34).
2. El admin navega al menú de encuestas (Figura 8.35) y tras pulsar en la primer opción, se le redirige la vista asociada al listado de encuestas en curso o finalizadas (Figura 8.38).
3. El admin establece una serie de filtros para la obtención del listado de encuestas y pulsa “Mostrar encuestas”.

[Si hay más de una página]:

3A. Las encuestas se distribuyen en las páginas disponibles (Figura 8.38).

Flujo de Eventos:

[Si se cambia de página]:

3AA. La vista se refresca, se pasa de página y se obtiene el listado de encuestas de la página actual.

3B. Todas las encuestas se muestran en la página actual.

[Si hace clic en "Abrir encuesta" de un componente cerrado]:

4A. Se abre el diálogo para la fecha de cierre de la encuesta (Figura 8.36).

[Si introduce una fecha válida]:

4AA. Se abre el diálogo del texto del email (Figura 8.37).

[Si acepta el diálogo]:

4AAA. Se abre la encuesta y se manda el email 8.39.

[Si no acepta el diálogo]:

4AAB. No se abre la encuesta ni se manda el email.

[Si no introduce una fecha válida]:

4AB. Se cancela la operación.

[Si hace clic en "Cerrar encuesta" de una encuesta abierta]:

5A. Se cierra la encuesta y se refresca la vista.

[Si selecciona una o varias encuestas]:

6A. Las encuestas se seleccionan y su aspecto cambia (Figura 8.40).

[Si pulsa el botón "Abrir Seleccionadas"]:

6AA. Se abre el diálogo asociado a la fecha de cierre de las encuestas.

[Si introduce una fecha válida]:

6AAA. Se abre el diálogo del texto del email.

[Si acepta el diálogo]:

6AAAA. Se abre la encuesta y se manda el email.

[Si no acepta el diálogo]:

6AAAB. No se abre la encuesta ni se manda el email.

[Si no introduce una fecha válida]:

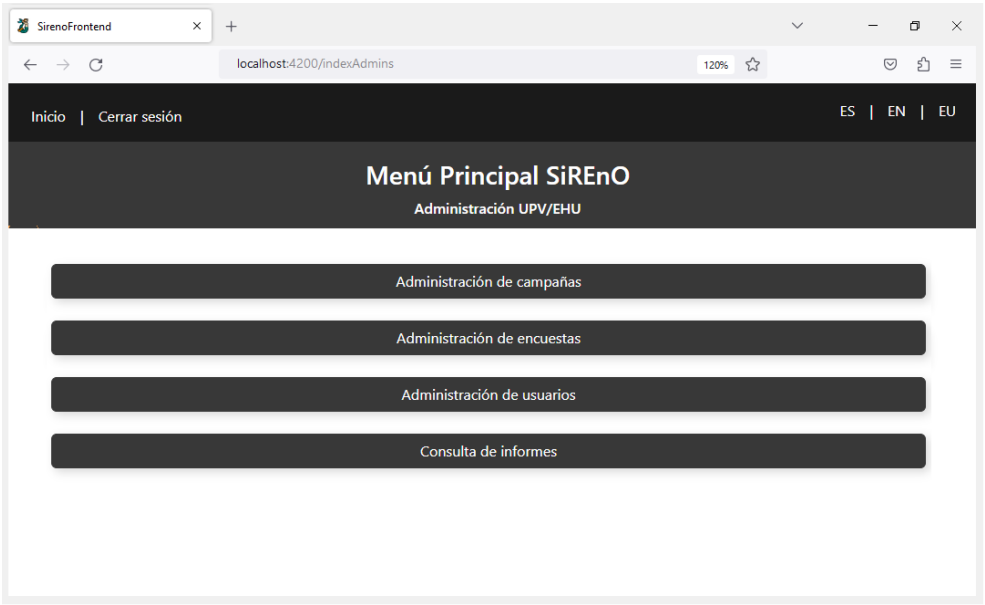
6AAB. Se cancela la operación.

Postcondiciones:

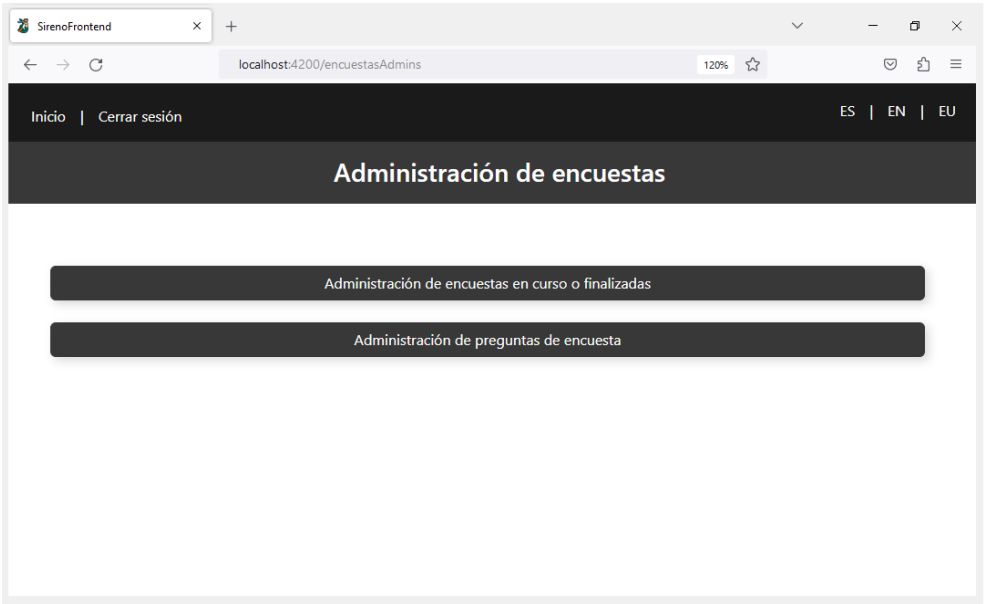
– La campaña seleccionada se marca como cerrada en el sistema.

– Los alumnos ya no pueden acceder a la encuesta ni enviar respuestas.

Interfaz gráfica:

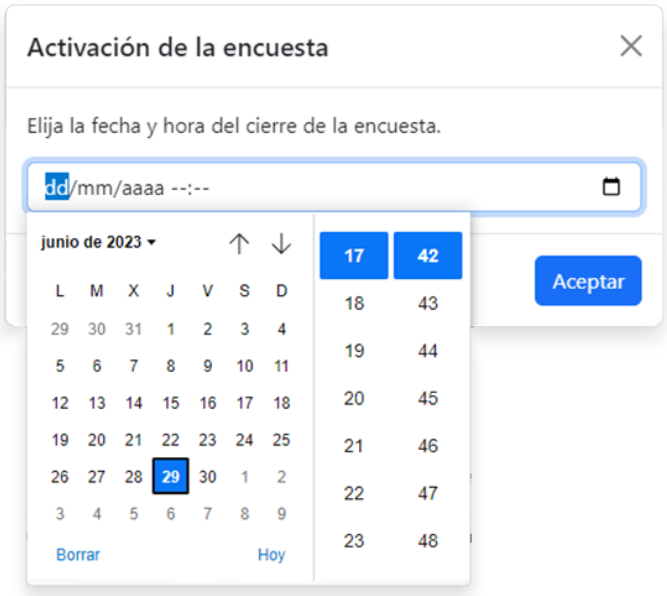


8.34. Figura: Índice principal del administrador.

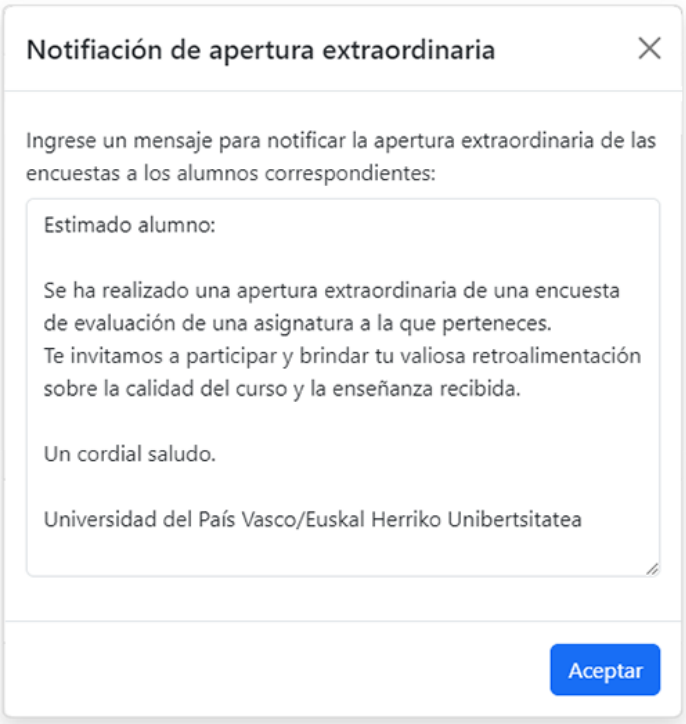


8.35. Figura: Índice de encuestas del administrador.

Interfaz gráfica:

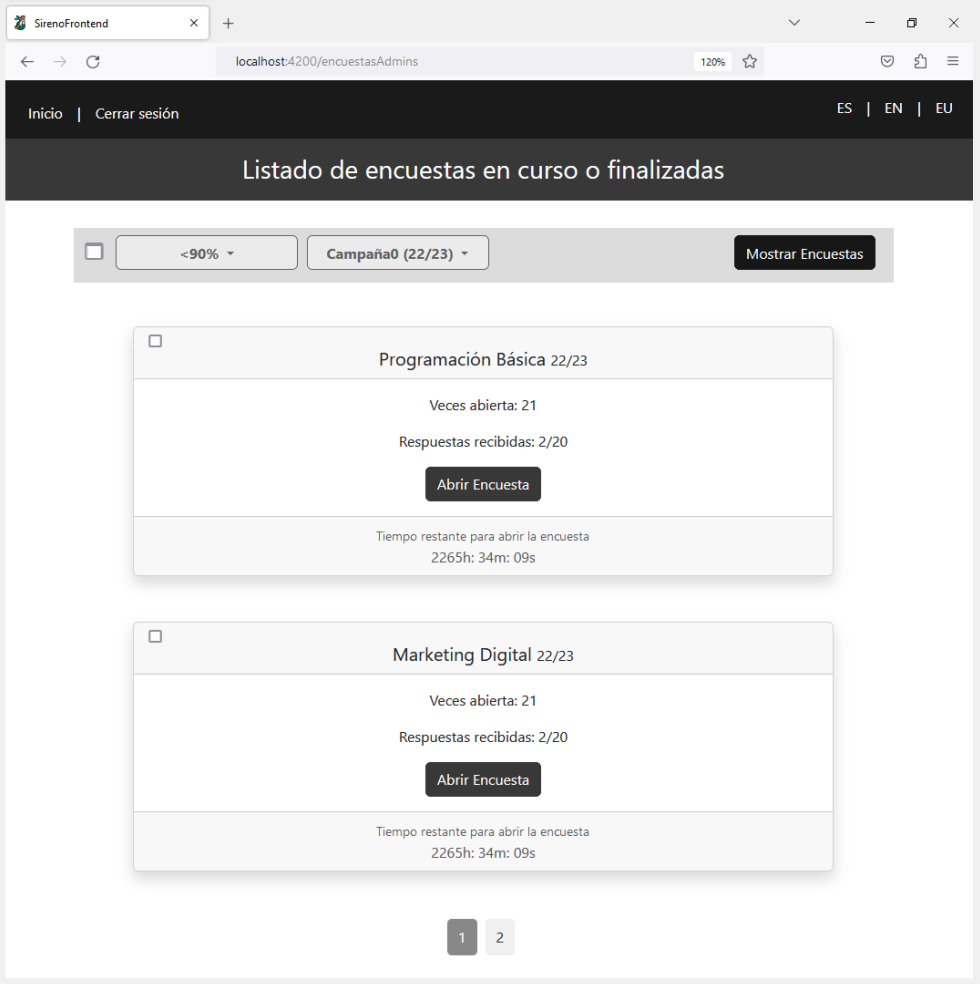


8.36. Figura: *Popup* para seleccionar el fin de la apertura.

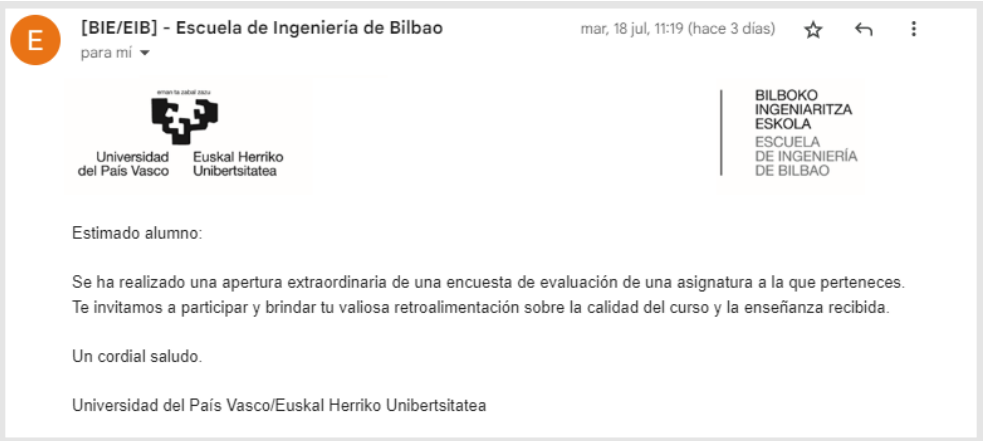


8.37. Figura: *Popup* para establecer el cuerpo del mensaje del mail.

Interfaz gráfica:

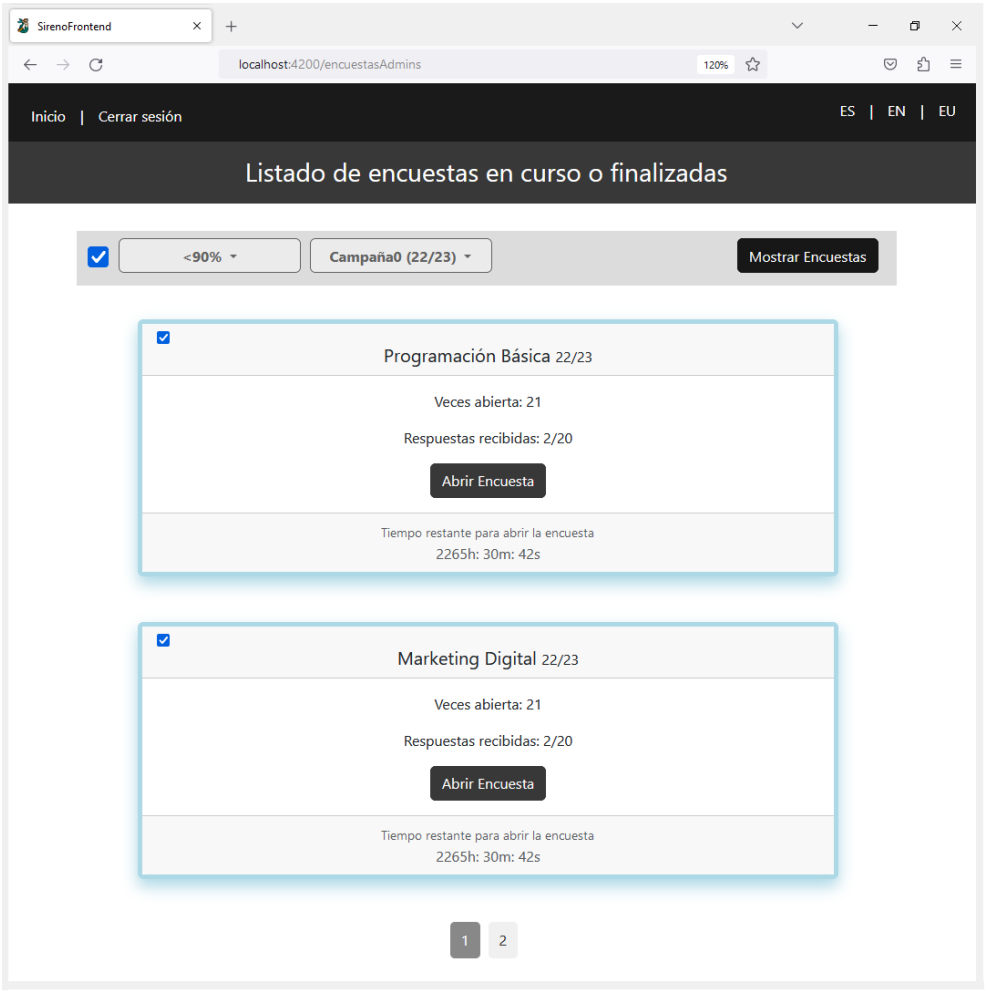


8.38. Figura: Listado de encuestas según filtros aplicados.



8.39. Figura: Ejemplo del email recibido notificando la apertura extraordinaria.

Interfaz gráfica:



8.40. Figura: Selección de todas las encuestas de la página.

Anexo II: Diagramas de secuencia

En el segundo y último anexo, se presentarán los diagramas de secuencia correspondientes a las principales funcionalidades de SiREnO. Estos diagramas capturan la secuencia de mensajes intercambiados entre los objetos durante la ejecución de una funcionalidad específica. Su finalidad es comprender cómo se comportan los objetos en relación con otros objetos y cómo se coordinan para llevar a cabo una determinada tarea.

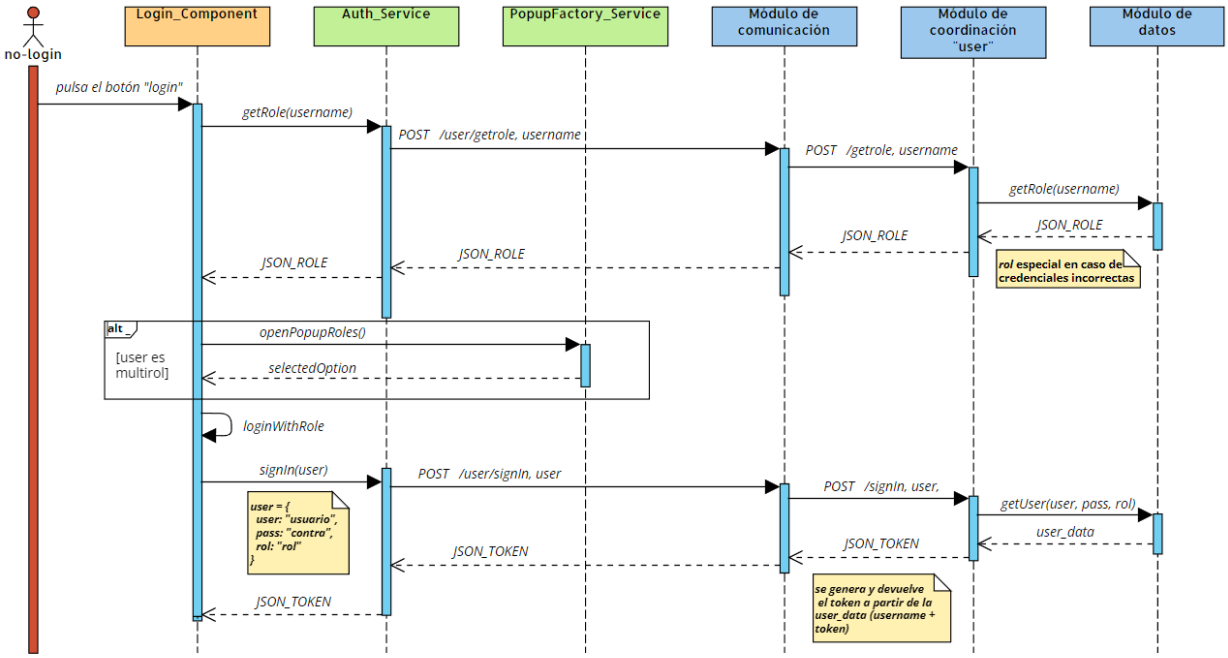
Cada diagrama representa una funcionalidad particular dentro de SiREnO, el cual consta de una parte de back-end y otra de front-end. Con el propósito de distinguir los tipos de entidades, se utilizarán diferentes colores en los diagramas de secuencia.

Los diagramas de secuencia se agruparán según el actor que los ejecuta.

1. Actor *no-login*

A continuación se muestran en orden los diagramas de secuencia correspondientes a los casos de uso extendidos del actor *no-login*.

1.1. Identificarse



El "módulo de coordinacion", realiza una llamada a la base de datos para obtener el rol o roles asociados al usuario. Esto se realiza para que en caso de tener varios roles, a la hora de loguearse se le pueda preguntar con cuál de ellos quiere iniciar sesión.

Cuando se define el rol con el que quiere iniciar sesión, se comprueban si las credenciales son correctas haciendo una llamada a la base de datos. Si son correctas, se genera un token válido y se devuelve. En caso contrario, se genera un token de error y se deniega el acceso. Las llamadas realizadas son las siguientes:

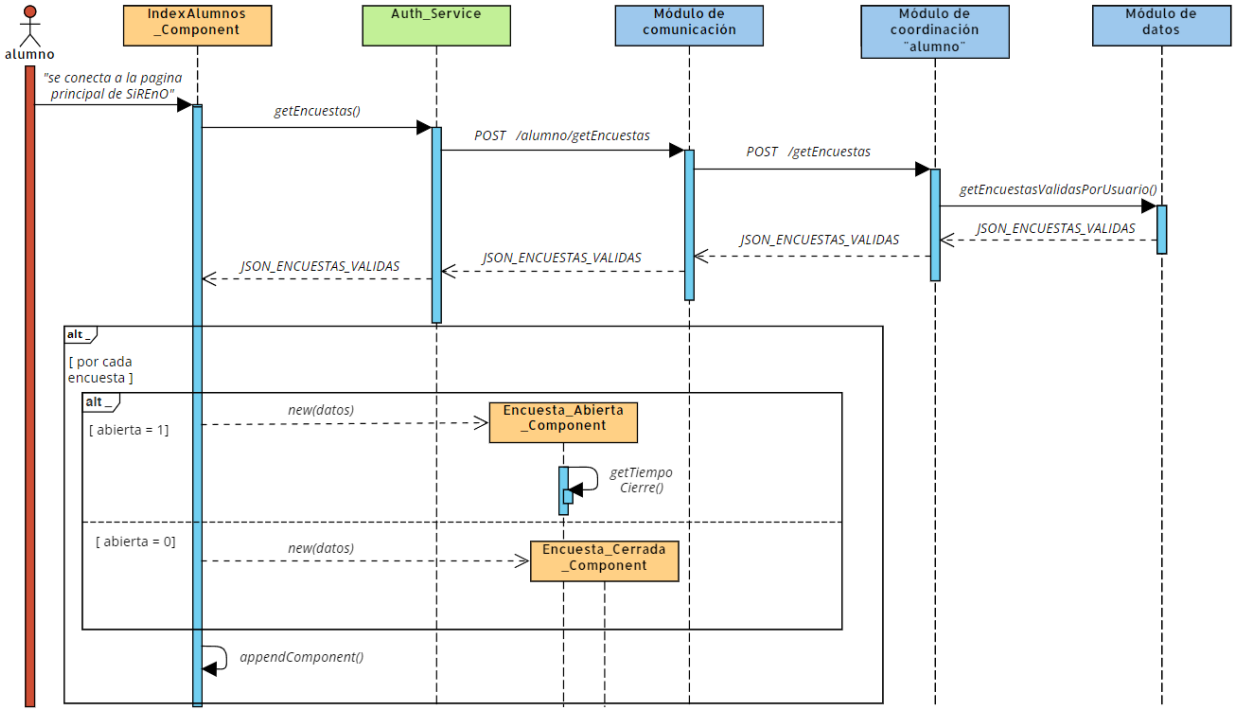
```
SELECT rol FROM usuario WHERE cod_usuario = ?

SELECT cod_usuario FROM usuario WHERE cod_usuario = ? AND contrasena = ?
```

2. Actor *alumno*

A continuación se muestran en orden los diagramas de secuencia correspondientes a los casos de uso extendidos del actor *alumno*.

2.1. Listar encuestas



Desde el front-end, haciendo uso de un servicio, se llama al "Módulo de comunicación", el cual va pasando el control hasta el "Módulo de datos" que realiza una llamada a la base de datos para obtener el conjunto ordenado de encuestas válidas.

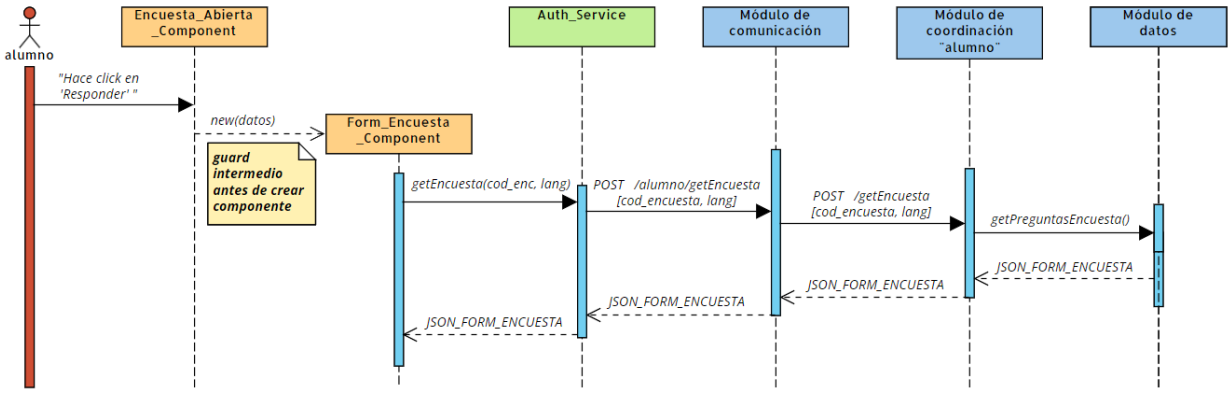
La finalidad de la consulta a la base de datos es recuperar todos los datos relevantes relacionados con cada encuesta. Para ello, se utiliza el identificador del alumno para obtener el listado de situaciones docentes a las que pertenece. A partir de este listado, se obtiene la encuesta asociada a cada situación docente. Además, es necesario conocer la información sobre la apertura de cada encuesta, así como el nombre de la asignatura o docente al que hacen referencia.

Esta lógica se implementa mediante la combinación de las tablas necesarias utilizando operaciones de JOIN. A continuación, se muestra la consulta SQL completa:

```
SELECT c.cod_campana, c.nombre, c.fecha_fin, c.abierta_antes,
      c.cod_encuesta, sd.cod_situacion_docente, sd.cod_asignatura,
      a.nombre_asignatura, sd.cod_docente, d.nombre_docente, sd.num_curso,
      sd.año_curso, ac.fecha_hora_cierre
FROM campana AS c
JOIN situacion_docente AS sd ON c.cod_campana = sd.cod_campana
JOIN alumno_situacion_doc AS asd ON sd.cod_situacion_docente =
      asd.cod_situacion_docente
JOIN asignatura AS a ON sd.cod_asignatura = a.cod_asignatura
JOIN docente AS d ON sd.cod_docente = d.cod_docente
LEFT JOIN activacion_campana AS ac ON sd.cod_situacion_docente =
      ac.cod_situacion_docente WHERE asd.cod_alumno = ?
AND c.fecha_ini <= NOW()
AND (c.abierta_antes = 0 AND c.fecha_fin >= NOW() OR c.abierta_antes = 1 AND
ac.fecha_hora_cierre >= NOW())
```

Las encuestas se devuelven en formato JSON. En el front-end, por cada encuesta del JSON, se crea un nuevo componente asociado y se inserta en la vista del índice de los alumnos.

2.2. Abrir una encuesta



Se invoca al "Módulo de comunicación" desde el front-end utilizando un servicio, y este módulo va transmitiendo el control al "Módulo de datos". A su vez, el "Módulo de datos" realiza una consulta a la base de datos con el fin de obtener un conjunto preguntas y respuestas que componen el formulario asociado a la encuesta.

En primer lugar se obtienen todas las preguntas (los códigos de cada pregunta) junto con el texto asociado en el idioma correspondiente. Además, para cada pregunta se obtiene un booleano que indica si el conjunto de respuestas son numéricas o textuales. Esta es al consulta asociada:

```
SELECT pe.cod_pregunta, tp.texto, p.numerica
FROM pregunta_en_encuesta pe
JOIN texto_pregunta tp ON pe.cod_pregunta = tp.cod_pregunta
JOIN pregunta p ON pe.cod_pregunta = p.cod_pregunta
WHERE pe.cod_encuesta = ? AND tp.cod_idioma = ?
```

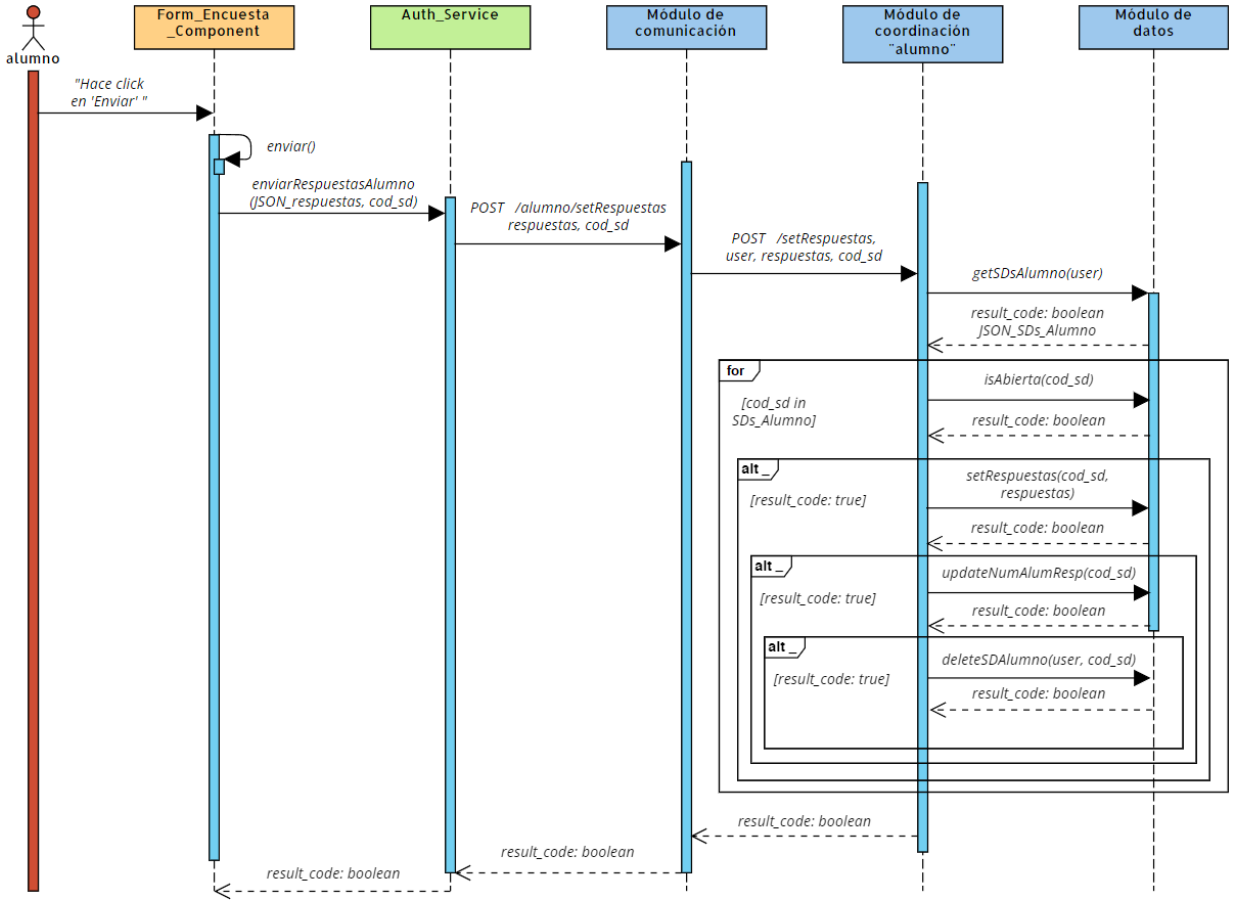
Para cada pregunta, se debe obtener las posibles respuestas según el tipo de pregunta. Si la respuesta a la pregunta es numérica, obtener las respuestas de "respuesta_numerica_de_pregunta":

```
SELECT rnp.cod_respuesta_numerica
FROM respuesta_numerica_de_pregunta rnp
WHERE rnp.cod_pregunta = ?
```

Si por lo contrario, la respuesta es textual, se debe obtener las respuestas de la tabla "respuesta_textual" y el texto asociado a cada respuesta de la tabla "texto_respuesta" en el idioma especificado:

```
SELECT tr.cod_respuesta_textual, tr.texto
FROM respuesta_textual rv
JOIN texto_respuesta tr ON rv.cod_respuesta_textual =
tr.cod_respuesta_textual
WHERE rv.cod_pregunta = ? AND tr.cod_idioma = ?
```

2.3. Responder una encuesta



El componente front-end invoca al back-end mediante el servicio de comunicación correspondiente, el cual facilita la interacción a través de la API. La petición se propaga al "Módulo de comunicación", que, al recibir una solicitud relacionada con la información de los alumnos, transfiere el control al "Módulo de coordinación" correspondiente a dicho ámbito. Este último realiza una serie de llamadas al "Módulo de datos" para llevar a cabo modificaciones en la base de datos.

En la primera llamada, se obtiene el conjunto de situaciones docentes asociadas al alumno identificado por el token incluido en el cuerpo de la petición.

```
SELECT cod_situacion_docente FROM alumno_situacion_doc WHERE cod_alumno = ?
```

Se comprueba que la situación docente actual relacionada con el formulario que se acaba de responder figura en la lista de situaciones docentes pendientes de alumno. En caso de así serlo, se comprueba si la encuesta asociada a dicha situación docente está abierta.

```
SELECT * FROM activacion_campana WHERE cod_situacion_docente = ? AND
fecha_hora_ini < ? AND fecha_hora_cierre > ?
```

Una vez se cumple la condición requerida, se procede a la inserción del conjunto de respuestas en la base de datos. Durante este paso, se realizan dos consultas diferentes dependiendo de si la respuesta es numérica o no. Para los dos tipos de respuestas, si la respuesta es nueva en la base de datos, se crea una nueva entrada con un contador inicializado en 1. Por el contrario, si la respuesta ya existe, se actualiza el contador incrementándolo en 1.

```
INSERT INTO respuesta_numerica_alumnos (cod_situacion_docente,
cod_respuesta_numerica, cod_pregunta, cuantos)
VALUES (?, ?, ?, 1)
ON DUPLICATE KEY UPDATE cuantos = cuantos + 1
```

```
INSERT INTO respuesta_textual_alumnos (cod_situacion_docente,
cod_respuesta_textual, cuantos)
VALUES (?, ?, ?, 1)
ON DUPLICATE KEY UPDATE cuantos = cuantos + 1
```

El siguiente paso es actualizar el número de alumnos que han respondido al formulario asociado a la encuesta de dicha situación docente:

```
UPDATE situacion_docente SET n_alum_respondido = n_alum_respondido + 1
WHERE cod_situacion_docente = ?
```

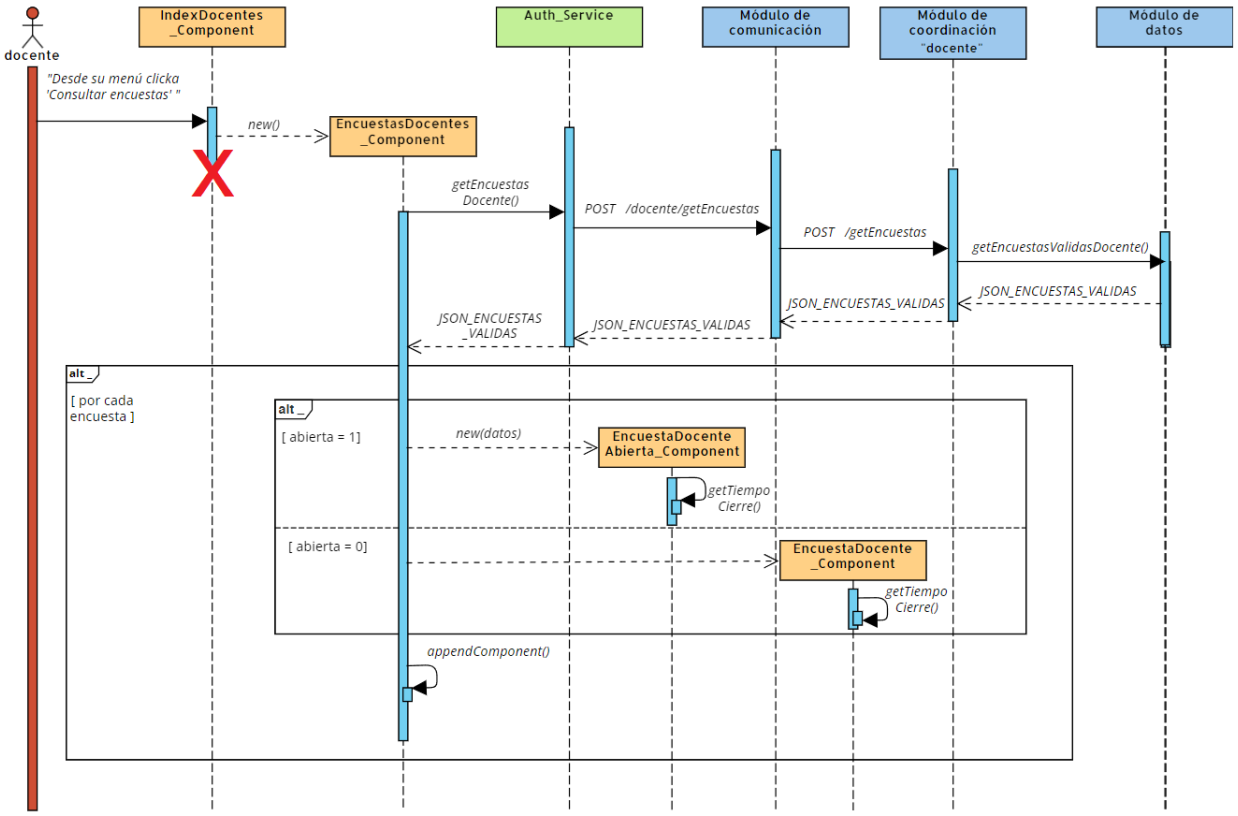
Para finalizar, el último paso es borrar la situación docente recién respondida del conjunto de situaciones docentes pendientes de responder del alumno actual:

```
DELETE FROM alumno_situacion_doc
WHERE cod_alumno = ? AND cod_situacion_docente = ?
```

3. Actor *docente*

A continuación se muestran en orden los diagramas de secuencia correspondientes a los casos de uso extendidos del actor *docente*.

3.1. Consultar encuestas



Desde el front-end, a través del servicio, se llama al "Módulo de comunicación", el cual pasa el control al "Módulo de datos" para obtener las encuestas asociadas a campañas válidas de manera ordenada desde la base de datos.

El propósito de esta consulta es recuperar todos los datos relevantes relacionados con cada encuesta. Se utiliza el identificador del docente para obtener el listado de situaciones docentes en las que el docente Figura como profesor. Esta lógica se implementa mediante JOINs combinando las tablas necesarias.

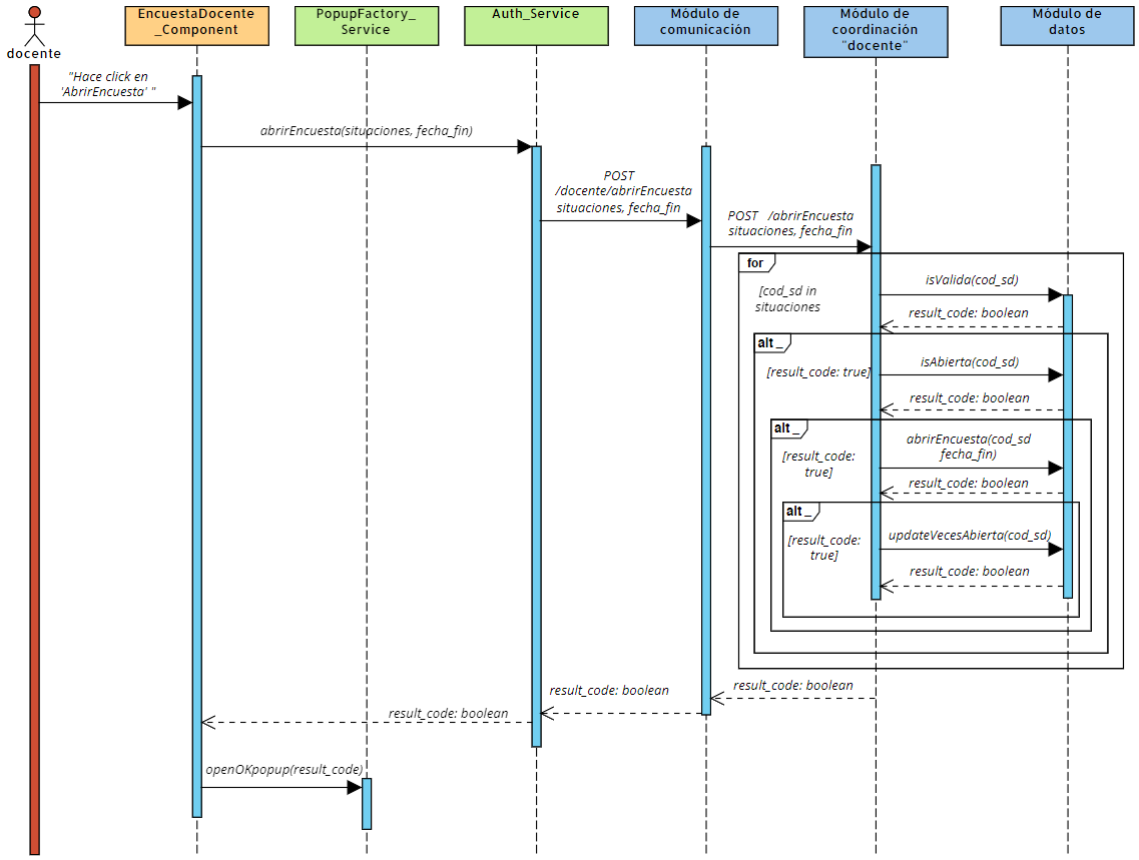
Esta sentencia de *SQL* realiza una consulta a la base de datos y selecciona varias columnas de diferentes tablas. La consulta une las tablas "situacion_docente", "campaña" y "asignatura", y utiliza condiciones para filtrar los resultados. Luego, ordena los resultados de acuerdo con varios criterios, como la existencia de una fecha de cierre, la fecha de apertura de la encuesta (si tiene) y la fecha de finalización de la campaña asociada. En resumen, la consulta busca información sobre la situación docente, los alumnos, las

asignaturas y las campañas, y organiza los resultados de manera específica.

En la siguiente página se muestra la sentencia completa de la llamada a la base de datos.

```
SELECT sd.cod_situacion_docente, sd.n_alum_total, sd.n_alum_respondido,
a.nombre_Asignatura, c.fecha_fin, sd.num_curso, c.año_curso, sd.activada,
sd.agrupado_con, ac.fecha_hora_cierre
FROM situacion_docente sd
INNER JOIN campana c ON sd.cod_campana = c.cod_campana
JOIN asignatura AS a ON sd.cod_asignatura = a.cod_asignatura
LEFT JOIN (
SELECT cod_situacion_docente, MAX(fecha_hora_ini) AS max_fecha_hora_ini
FROM activacion_campana
WHERE 'currentDate' <= fecha_hora_cierre
GROUP BY cod_situacion_docente
) ac_latest ON sd.cod_situacion_docente = ac_latest.cod_situacion_docente
LEFT JOIN activacion_campana ac ON ac_latest.cod_situacion_docente =
ac.cod_situacion_docente AND ac_latest.max_fecha_hora_ini = ac.fecha_hora_ini
WHERE sd.cod_docente = ? AND
'currentDate' BETWEEN c.fecha_ini AND c.fecha_fin
ORDER BY
CASE WHEN ac.fecha_hora_cierre IS NOT NULL THEN 0 ELSE 1 END,
CASE WHEN ac.fecha_hora_cierre IS NULL THEN 1 ELSE 0 END,
ac.fecha_hora_cierre ASC,
sd.activada ASC,
CASE WHEN ac.fecha_hora_cierre IS NULL THEN sd.activada END ASC,
c.fecha_fin ASC
```

3.2. Abrir encuesta y programar plazo



Desde el front-end, se invoca el servicio de "Módulo de comunicación" que, a su vez, transfiere el control al "Módulo de datos". El propósito de esta llamada es activar el conjunto de situaciones docentes proporcionadas como parámetro, estableciendo la fecha de finalización de la apertura según el valor también pasado como parámetro.

Se procede a recorrer el grupo de situaciones docentes contenidas en el conjunto, y se realizan una serie de verificaciones de seguridad para cada una de ellas. Antes de abrir una encuesta, se verifica si es válida y si no está actualmente abierta. Si cumple con estas condiciones, la encuesta se abre utilizando la fecha de finalización establecida y validada en el front-end. Una vez abierta, se actualiza el contador de veces abierta para la situación docente. Estas son las sentencias *SQL* asociadas a cada llamada al "Módulo de datos":

Primero se comprueba si la la situación docente pertenece a una campaña válida.

```
SELECT * FROM campana WHERE cod_campana = (SELECT cod_campana FROM
situacion_docente WHERE cod_situacion_docente = ?) AND ? >= fecha_ini AND
? <= fecha_fin
```

La segunda verificación consiste en comprobar si la encuesta no está abierta.

```
SELECT * FROM apertura_encuesta WHERE cod_situacion_docente = ? AND
fecha_hora_ini < ? AND fecha_hora_cierre > ?
```

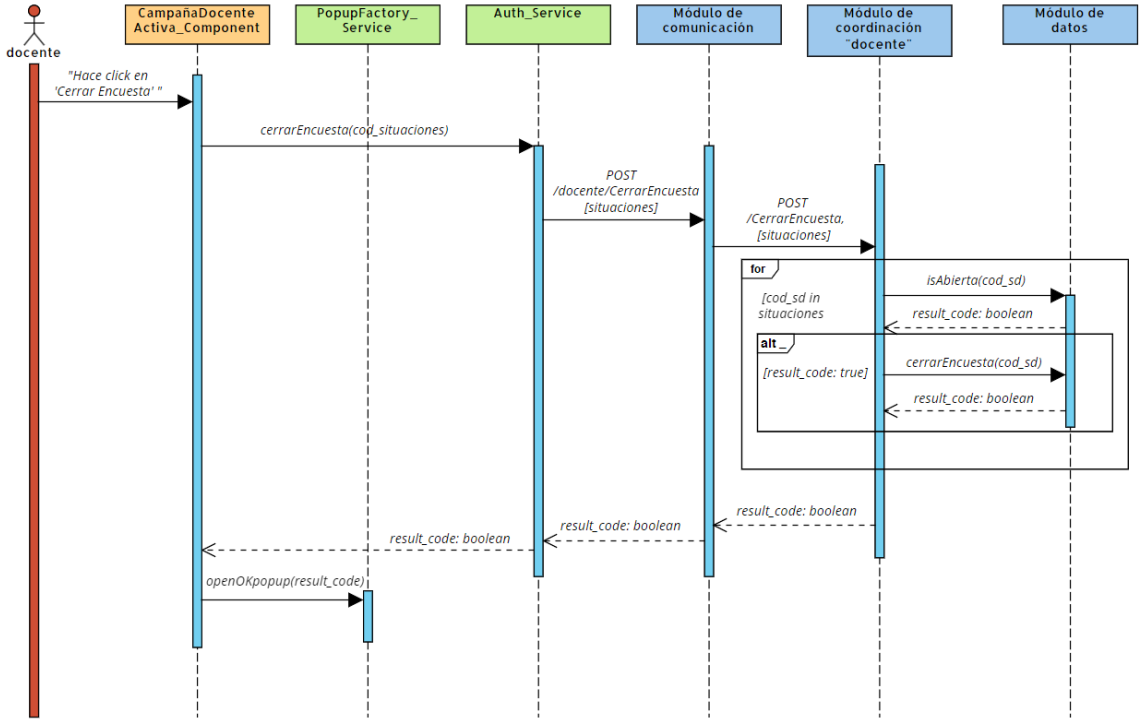
Cuando se cumplen ambas validaciones de manera satisfactoria, se procede a activar la campaña asociada a la situación docente actual.

```
SELECT c.fecha_ini, c.fecha_fin FROM campana c INNER JOIN situacion_docente
sd ON c.cod_campana = sd.cod_campana WHERE sd.cod_situacion_docente = ?
```

Por último, se actualiza el contador de veces abierta.

```
UPDATE situacion_docente SET activada = activada + 1 WHERE
cod_situacion_docente = ?
```

3.3. Cerrar encuesta



En el front-end, se realiza una llamada al servicio denominado "Módulo de comunicación", el cual se encarga de transferir el control al "Módulo de datos". El objetivo de esta llamada es cerrar las encuestas del conjunto de situaciones docentes que se proporcionan como parámetro.

Se procede a iterar a través del grupo de situaciones docentes presentes en el conjunto. Antes de cerrar una encuesta, se realiza una verificación para determinar si está abierta, ya que no tiene sentido cerrar una encuesta que no esté abierta. No se lleva a cabo una verificación adicional para comprobar su validez, dado que, por definición, toda encuesta abierta debe ser válida. Además, un requisito previo para abrir una encuesta que ésta sea válida. Si la encuesta cumple con la condición de estar abierta, se procede a su cierre, estableciendo como fecha de finalización el momento actual, incluyendo fecha y hora.

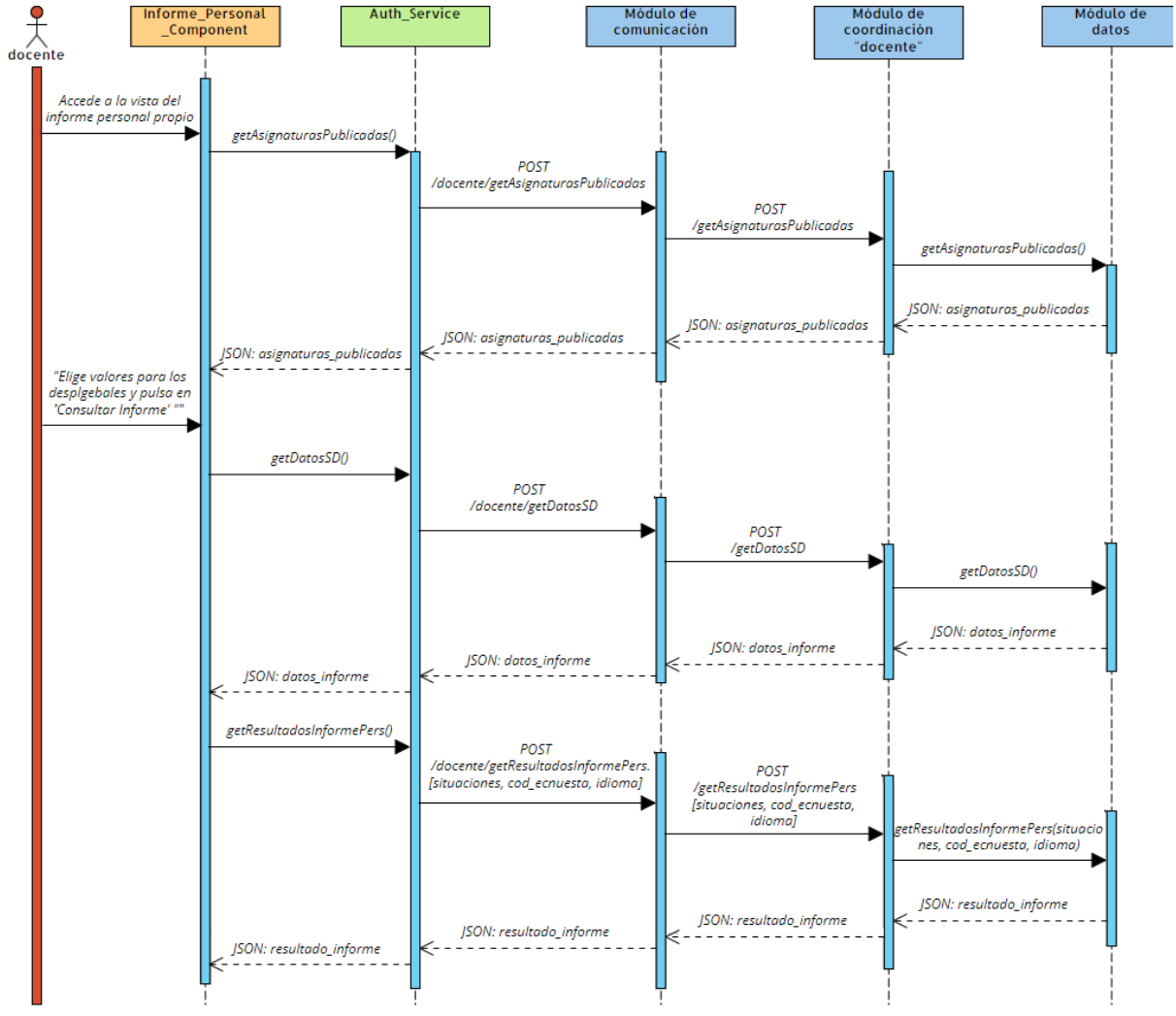
Primero se comprueba si la encuesta asociada a la situación docente está abierta.

```
SELECT * FROM activacion_campana WHERE cod_situacion_docente = ? AND
fecha_hora_ini < ? AND fecha_hora_cierre > ?SELECT * FROM activacion_campana
WHERE cod_situacion_docente = ? AND fecha_hora_ini < ? AND fecha_hora_cierre
> ?
```

Por último, se procede a cerrar la encuesta asociada a la situación docente actual, estableciendo como fecha de finalización la fecha y hora actual.

```
UPDATE activacion_campana SET fecha_hora_cierre = ? WHERE
cod_situacion_docente = ?
```

3.4. Informe personal



Cuando el docente accede al informe personal, el primer paso consiste en obtener, desde el back-end, el listado de asignaturas y los respectivos años en los que ha participado. Una vez obtenidos, estos datos se cargan en los desplegables disponibles en la interfaz. Cuando el docente selecciona las opciones de los desplegables, se habilita el botón "Consultar Informe". Al pulsar dicho botón, se realiza una primera llamada a la API para obtener los datos de la cabecera del informe, que describen la situación docente. Posteriormente, se efectúa una segunda llamada para obtener los resultados de los datos del informe. Finalmente, en el front-end, se realiza la presentación de los informes mediante gráficos y tablas.

El primer paso es recuperar el listado de asignaturas junto con los años respectivos en los que el docente ha participado.

```
SELECT asignatura.nombre_asignatura, campana.año_curso,
GROUP_CONCAT(situacion_docente.cod_situacion_docente) AS
agrupado_con FROM asignatura JOIN situacion_docente ON
asignatura.cod_asignatura = situacion_docente.cod_asignatura JOIN
campana ON situacion_docente.cod_campana = campana.cod_campana WHERE
situacion_docente.cod_docente = ? AND campana.informe_publicado = 1 GROUP BY
asignatura.cod_asignatura, asignatura.nombre_asignatura, campana.año_curso
```

Consulta *SQL* para recuperar los datos que describen la situación docente.

```
SELECT cpa.cod_encuesta,
c.cod_centro, c.nombre_centro,
t.cod_grado, t.nombre_grado,
d.cod_depto, d.nombre_depto,
p.cod_docente, p.nombre_docente,
a.cod_asignatura, a.nombre_asignatura,
g.cod_grupo, g.nombre_grupo,
s.num_curso, cpa.año_curso,
s.cod_campana, s.cod_situacion_docente,
s.n_alum_total, s.n_alum_respondido
FROM situacion_docente s
INNER JOIN centro c ON s.cod_centro = c.cod_centro
INNER JOIN grado t ON s.cod_grado = t.cod_grado
INNER JOIN docente p ON s.cod_docente = p.cod_docente
INNER JOIN asignatura a ON s.cod_asignatura = a.cod_asignatura
INNER JOIN grupo g ON s.cod_grupo = g.cod_grupo
INNER JOIN departamento d ON d.cod_centro = c.cod_centro
INNER JOIN campana cpa ON s.cod_campana = cpa.cod_campana
WHERE s.cod_situacion_docente = ?
```

Las consultas siguientes hacen referencia a aquellas utilizadas para recuperar los datos asociados a los formularios de la situación docente seleccionada, con el objetivo de llevar a cabo la generación del informe.

En primer lugar se obtiene el conjunto de preguntas de la encuesta asociada a la situación docente del informe.

```
SELECT pe.cod_pregunta, tp.texto, p.numerica FROM pregunta_en_encuesta pe
JOIN texto_pregunta tp ON pe.cod_pregunta = tp.cod_pregunta JOIN pregunta
p ON pe.cod_pregunta = p.cod_pregunta WHERE pe.cod_encuesta = ? AND
tp.cod_idioma = ?
```

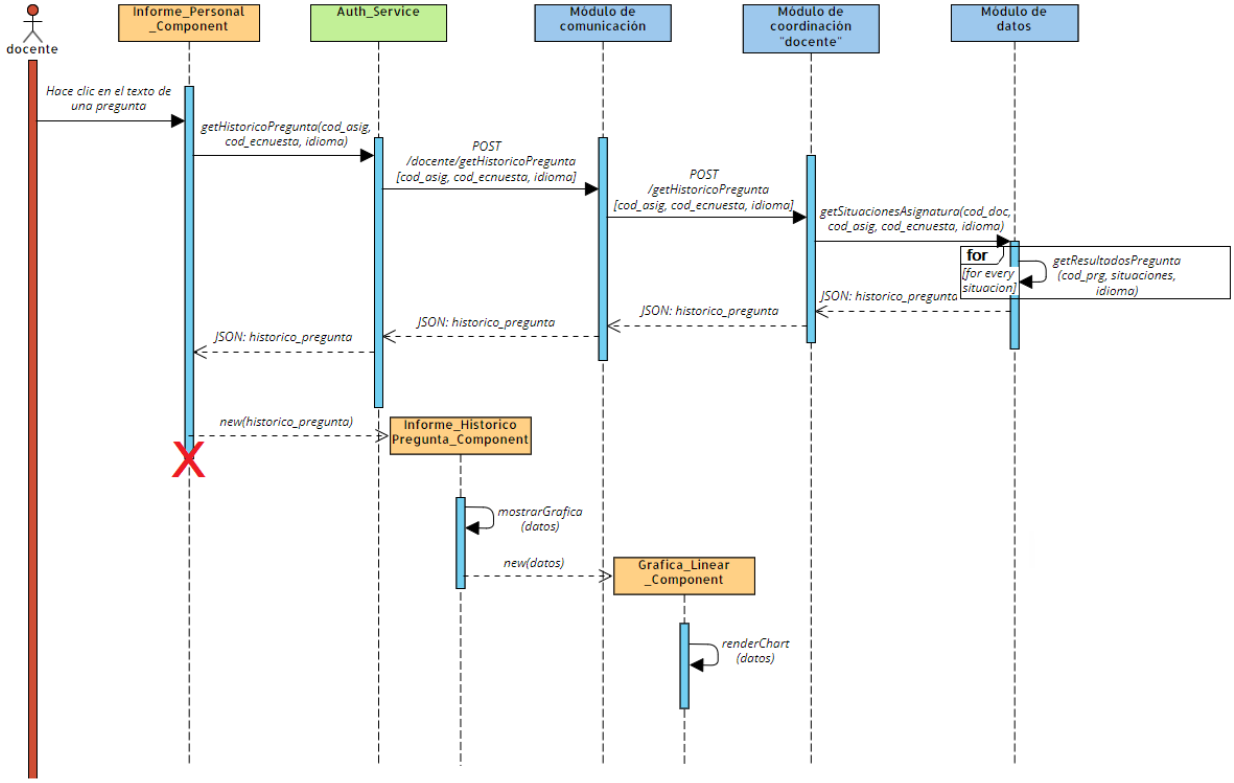
Posteriormente, por cada pregunta, se obtienen los resultados recibidos por los alumnos, el tratamiento de la obtención de resultados varía en función de si la pregunta es numérica o no. Esta es la consulta para la obtención de los resultados de una pregunta numérica.

```
SELECT rnp.cod_respuesta_numerica, IFNULL(rna.cuantos, 0) AS cuantos FROM
respuesta_numerica_de_pregunta rnp LEFT JOIN respuesta_numerica_alumnos
rna ON rna.cod_respuesta_numerica = rnp.cod_respuesta_numerica AND
rna.cod_pregunta = rnp.cod_pregunta AND rna.cod_situacion_docente = ? WHERE
rnp.cod_pregunta = ? AND rna.cod_respuesta_numerica IN ('1', '2', '3', '4',
'5') AND rna.cod_situacion_docente = ? GROUP BY rnp.cod_respuesta_numerica
```

Esta es la consulta para la obtención de los resultados de una pregunta numérica.

```
SELECT rv.cod_respuesta_verbal, tr.texto, IFNULL(rva.cuantos, 0) AS cuantos
FROM respuesta_verbal rv JOIN texto_respuesta tr ON rv.cod_respuesta_verbal
= tr.cod_respuesta_verbal LEFT JOIN respuesta_verbal_alumnos
rva ON rva.cod_respuesta_verbal = tr.cod_respuesta_verbal AND
rva.cod_situacion_docente = ? WHERE rv.cod_pregunta = ? AND tr.cod_idioma
= ? AND rva.cod_situacion_docente = ? GROUP BY tr.cod_respuesta_verbal‘
```

3.5. Informe histórico de pregunta



Cuando el docente hace clic en el texto de una pregunta, accede al informe histórico de esa pregunta. Para ello, el primer paso consiste en obtener, desde el back-end, las calificaciones que los alumnos han proporcionado para esa pregunta a lo largo de los años en las situaciones donde dicho docente ha estado presente. Con esta información, es posible calcular posteriormente la media para cada año.

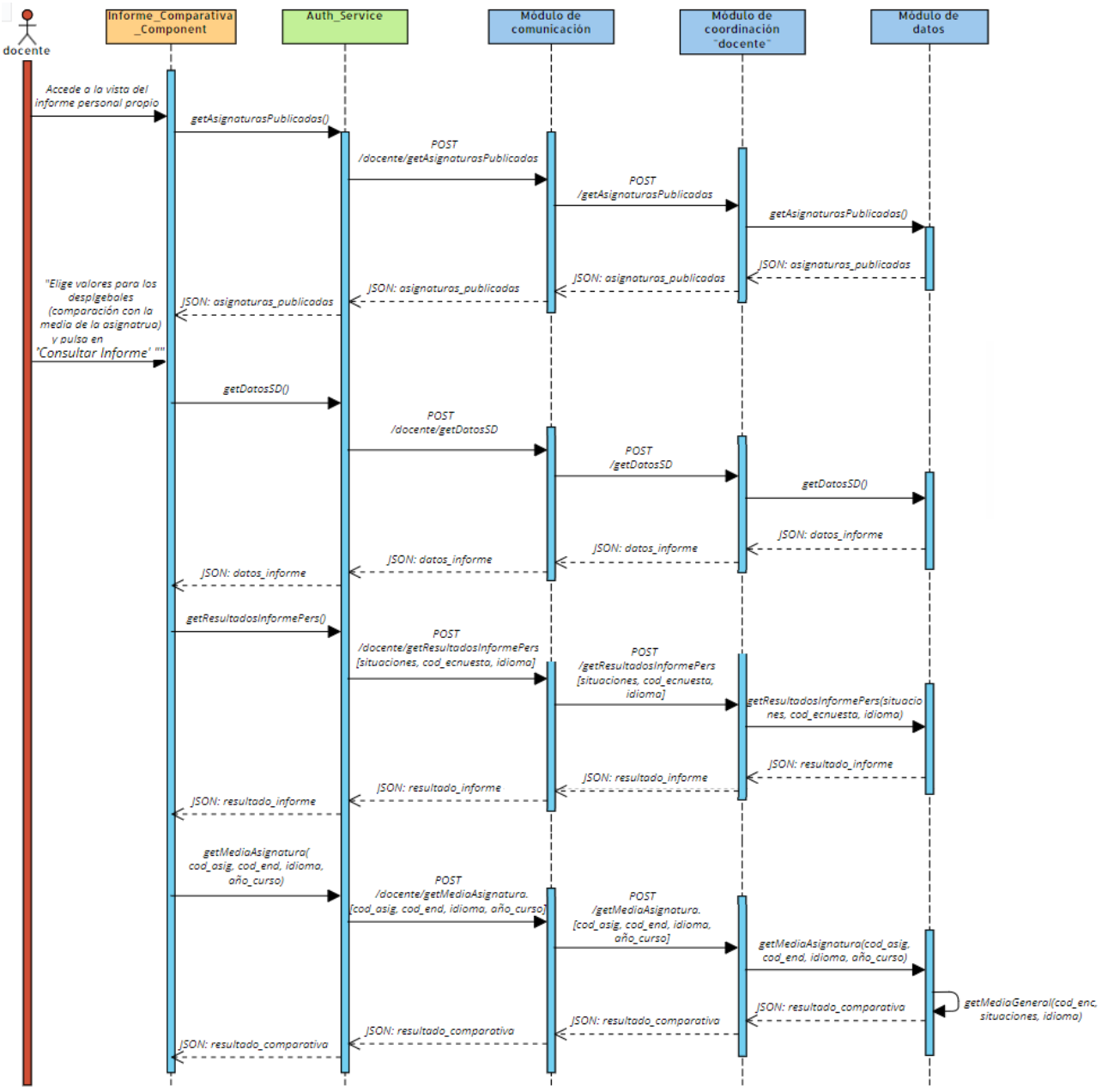
El primer paso consiste, por tanto, en obtener el listado de situaciones docentes y el curso asociado para todas las ocasiones en las que dicho docente ha impartido esa asignatura.

```
SELECT sd.cod_situacion_docente, c.año_curso FROM situacion_docente sd INNER
JOIN campana c ON sd.cod_campana = c.cod_campana WHERE sd.cod_docente = ?
AND sd.cod_asignatura = ?
```

Se recorre la lista de situaciones docentes recién obtenida. Por cada situación docente, se obtiene el número de respuestas de cada opción que ha obtenido dicha pregunta en el año concreto.

```
SELECT rnp.cod_respuesta_numerica, IFNULL(SUM(rna.cuantos), 0) AS cuantos
FROM respuesta_numerica_de_pregunta rnp
LEFT JOIN respuesta_numerica_alumnos rna ON rna.cod_respuesta_numerica
= rnp.cod_respuesta_numerica AND rna.cod_pregunta = rnp.cod_pregunta AND
rna.cod_situacion_docente IN (placeholders)
WHERE rnp.cod_pregunta = ? AND rna.cod_respuesta_numerica IN ('1', '2', '3',
'4', '5')
AND rna.cod_situacion_docente IN (placeholders)
GROUP BY rnp.cod_respuesta_numerica
```

3.6. Informe comparativo



El proceso de obtención y presentación de los datos es análogo al de los informes personales. La diferencia radica en que, en este caso, también se deben obtener los datos para la comparativa. Se tomará como ejemplo la media de la asignatura, es decir, todas las respuestas registradas para la asignatura seleccionada en la campaña seleccionada, independientemente del docente.

Para ello, lo primero es obtener el listado de situaciones docentes correspondientes a dicha asignatura y año:

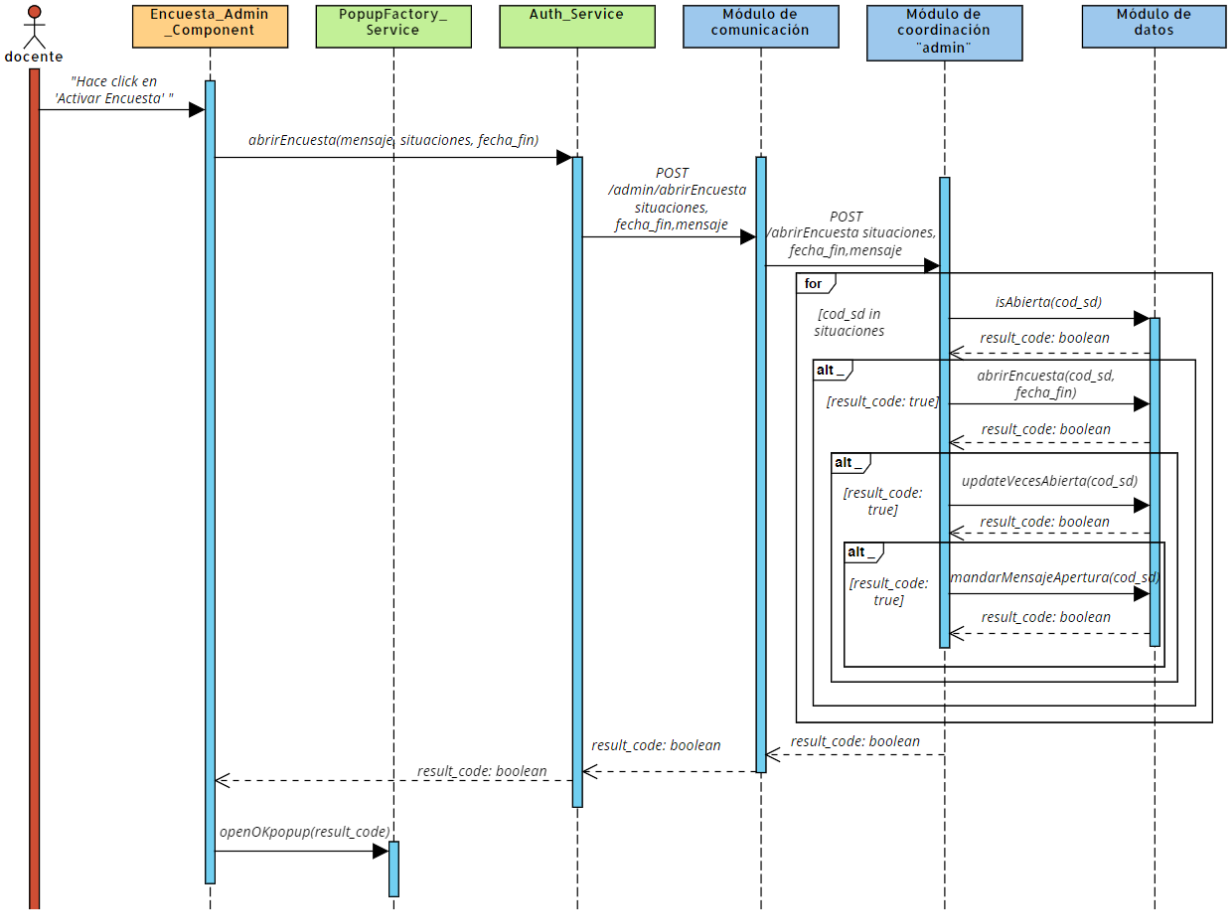
```
SELECT cod_situacion_docente FROM situacion_docente WHERE cod_asignatura = ?
AND cod_campana IN ( SELECT cod_campana FROM campana WHERE año_curso = ?
```

Una vez obtenido el listado de situaciones docentes, se llama al método *"getMediasGeral()"* utilizado por todos los métodos que deben calcular una media a partir de una situación docente. A este método se le pasa el listado de situaciones docentes junto con el código de la encuesta asociada. Luego, el método recorre el listado de situaciones, y por cada situación, llama al método *"getResultadosInformePersonal()"*, que fue explicado previamente y que obtiene el recuento de los resultados de todas las preguntas del informe. Después de recorrer todas las situaciones y obtener los resultados, se realiza una fusión del recuento de respuestas de todas las situaciones docentes.

4. Actor *admin*

A continuación se muestran en orden los diagramas de secuencia correspondientes a los casos de uso extendidos del actor *docente*.

4.1. Abrir y programar una encuesta



En el front-end se invoca el servicio para llamar al "Módulo de comunicación", que a su vez llama al "Módulo de datos" para abrir las encuestas del conjunto de situaciones docentes proporcionadas. Se verifica que la encuesta no está abierta previamente. Al validar la condición se procede a abrir la encuesta con la fecha de finalización establecida. Luego, se actualiza el contador de veces activadas para cada situación docente. Finalmente se mandan los correos pertinentes para notificar a los usuarios de la situación Las sentencias *SQL* asociadas se incluyen a continuación:

Primero se comprueba si la encuesta asociada a la situación docente no está abierta.

```
SELECT * FROM activacion_campana WHERE cod_situacion_docente = ? AND
fecha_hora_ini < ? AND fecha_hora_cierre > ?SELECT * FROM activacion_campana
WHERE cod_situacion_docente = ? AND fecha_hora_ini < ? AND fecha_hora_cierre
> ?
```

Si se cumple la anterior condición, se procede a abrir la encuesta asociada a la situación docente actual.

```
SELECT c.fecha_ini, c.fecha_fin FROM campana c INNER JOIN situacion_docente
sd ON c.cod_campana = sd.cod_campana WHERE sd.cod_situacion_docente = ?
```

Posteriormente, se actualiza el contador de veces abierta.

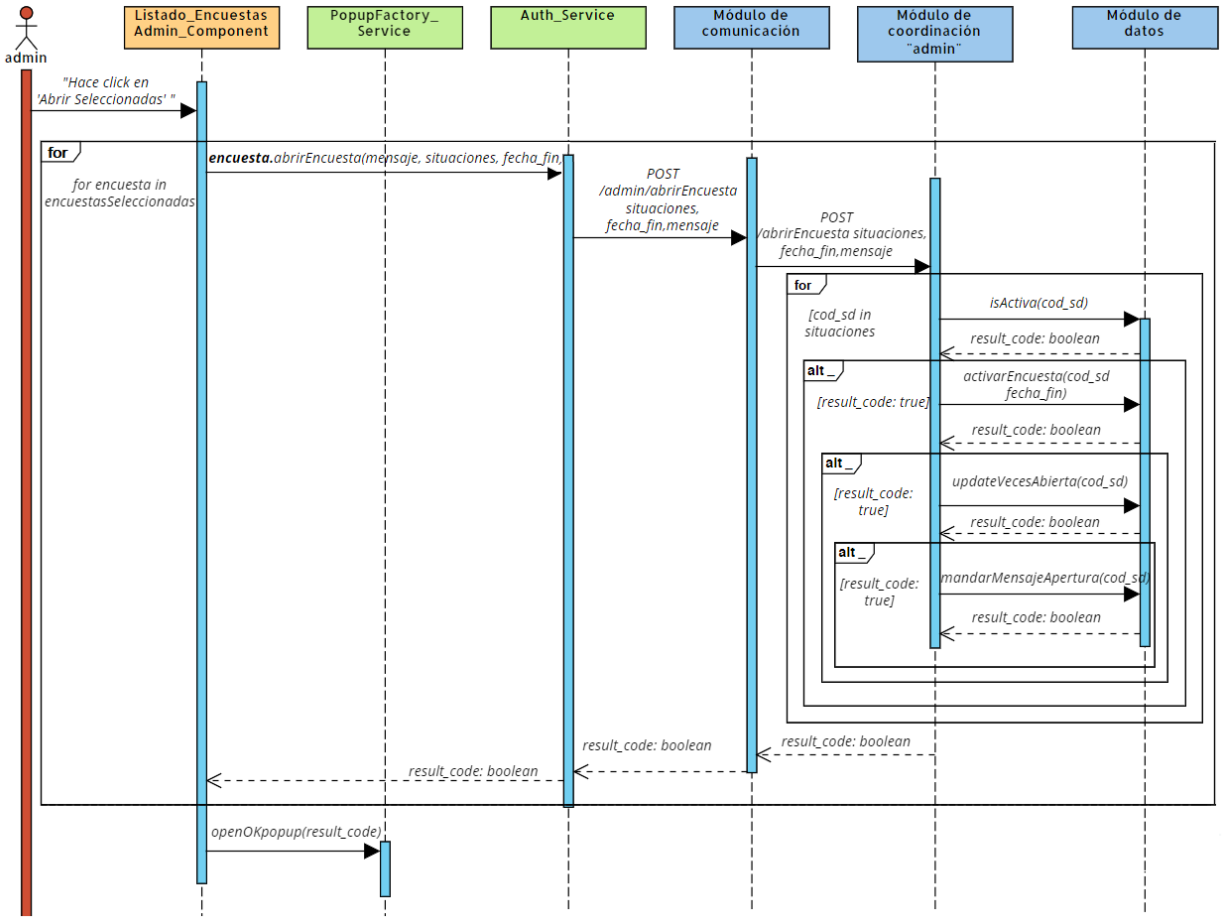
```
UPDATE situacion_docente SET activada = activada + 1 WHERE
cod_situacion_docente = ?
```

Por último, obtienen las direcciones de correo para mandar los emails a los alumnos de la situación o situaciones docentes cuyas encuesta se acaban de abrir.

```
SELECT cod_alumno FROM alumno_situacion_doc WHERE cod_situacion_docente = ?
```

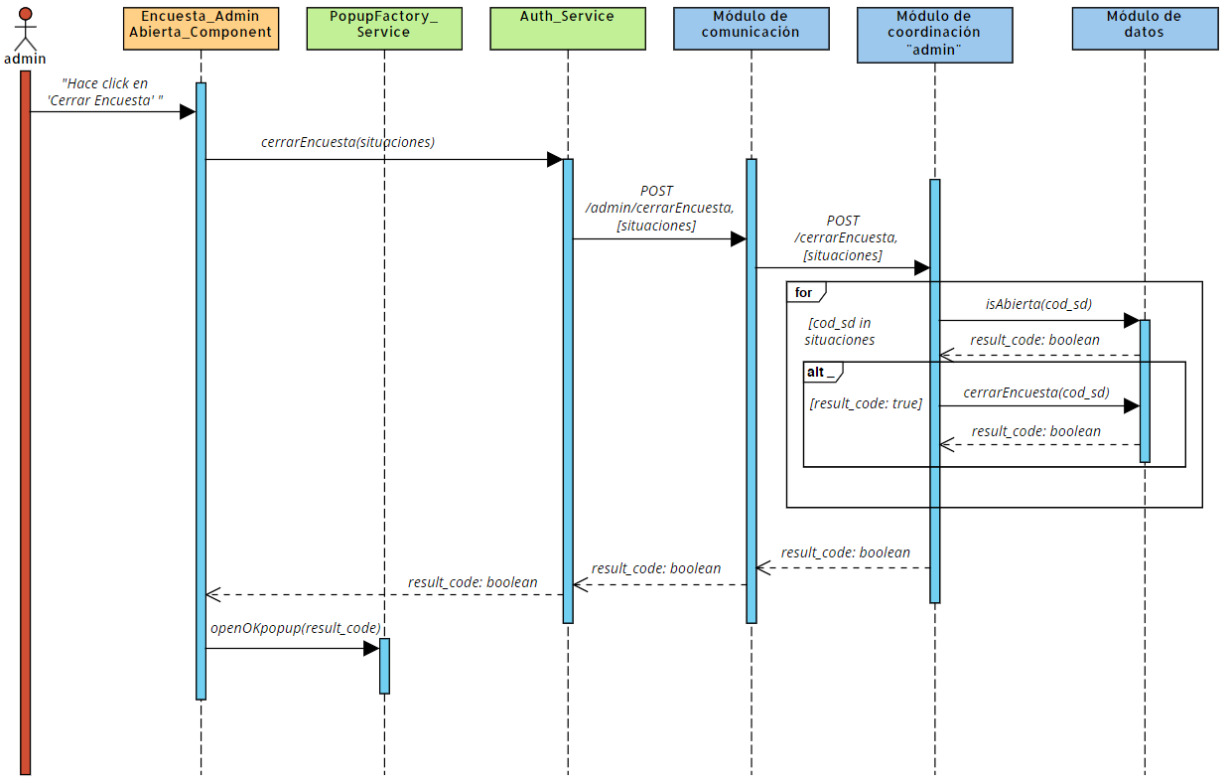
```
SELECT email FROM usuario WHERE cod_usuario IN cod_alumnos
```

4.2. Abrir y programar varias encuestas simultáneamente



El proceso es idéntico al de abrir una encuesta. En este caso, se obtiene la lista de encuestas seleccionada en la vista del administrador, y por cada encuesta, se llama al método "*abrirEncuesta()*" con los parámetros necesarios, tal como se muestra en el diagrama anterior.

4.2. Cerrar encuesta



El front-end, hace una petición al "Módulo de comunicación" para la ruta *"/admin/-cerrarEncuesta* con el fin de para cerrar las encuestas asociadas a las situaciones docentes proporcionadas como parámetro. Antes de cerrar una encuesta, se verifica si está abierta para evitar cerrar encuestas no abiertas. No es necesario comprobar su validez, ya que, como ocurría en el caso de esta funcionalidad pero para los docentes, todas las encuestas abiertas son válidas por definición. Por tanto, si la encuesta está abierta, se cierra estableciendo la fecha y hora actual como la fecha de finalización.

Consulta *SQL* para obtener la última fecha de apertura, con el propósito de verificar posteriormente si la encuesta se encuentra actualmente abierta.

```
SELECT * FROM activacion_campana WHERE cod_situacion_docente = ? AND
fecha_hora_ini < ? AND fecha_hora_cierre > ?SELECT * FROM activacion_campana
WHERE cod_situacion_docente = ? AND fecha_hora_ini < ? AND fecha_hora_cierre
> ?
```

Finalmente, se realiza el cierre de la encuesta asociada a la situación docente actual, estableciendo la fecha y hora actuales como la fecha de cierre.

```
UPDATE apertura_encuesta SET fecha_hora_cierre = ? WHERE
cod_situacion_docente = ?
```