

Trabajo Fin de Grado

Murcy, una aplicación web para crear y responder preguntas de forma lúdica.

Murcy, a web application to create and answer questions in a playful way.

Autor

Jorge Rambla González

Director

Francisco Javier López Pellicer

Escuela de ingeniería y arquitectura

2020

Quisiera agradecer a varias personas la ayuda que me han prestado en la realización de este trabajo de fin de grado. Entre ellas, en primer lugar, a mi tutor por todo el apoyo y tiempo dedicado para hacer esto posible.

A todos mis amigos y amigas que han sido los pilares fundamentales durante el transcurso del grado. Gracias por las fuerzas transmitidas todos estos años y nunca haber dudado de que esto era posible.

Y a toda mi familia, que siempre ha demostrado un apoyo constante, en especial a mi abuelo José González, que siempre ha creyó en mí, y lo seguirá haciendo desde mi memoria.

Resumen ejecutivo

El objetivo de este proyecto es la evolución y expansión de Murcy, una prueba de concepto que permite crear y jugar quizzes. Se desea que Murcy se convierta en un producto mínimo viable para desarrollar pruebas con usuarios reales.

Para llevar a cabo esta evolución, se ha realizado un análisis de las alternativas existentes en el mercado para poder extraer nuevas funcionalidades para Murcy, los riesgos existentes y la mejor metodología de desarrollo para Murcy. Y gracias a este análisis se ha podido terminar el proyecto con resultados satisfactorios.

Gracias a todo el proceso realizado se ha podido completar la evolución de Murcy a un producto mínimo viable con que realizar pruebas con usuarios reales.

Tabla de Contenidos

1	Introducción	3
1.1	Motivación	3
1.2	La evolución de los <i>quizzes</i> en la Web	3
1.3	Objeto del proyecto.....	4
1.4	Alcance.....	4
1.5	Estructura de la memoria.....	5
2	Análisis	6
2.1	Estudio de alternativas	6
2.2	Estudio de la prueba de concepto.....	8
2.3	Requisitos de partida	9
2.4	Metodología de gestión	10
2.5	Herramientas y tecnologías	12
2.6	Riesgos	13
3	Solución desarrollada	15
3.1	Arquitectura	15
3.2	Subsistemas.....	15
3.2.1	Modelo de datos	16
3.2.2	Servicios y lógica de negocio.....	17
3.2.3	Navegación de la aplicación web.....	17
3.2.4	Infraestructura	18
3.3	Adaptar prueba de concepto.....	19
3.4	Evolutivo y nuevos cambios.....	19
3.4.1	Evolución de la UI.....	19
3.4.2	Registro e inicio de sesión.....	21
3.4.3	Solicitudes para ser editor	22
3.4.4	Creación de preguntas	24
3.4.5	Creación de <i>quizzes</i>	25
3.4.6	Funcionalidad principal	26
3.4.7	Nueva funcionalidad y mejoras.....	27
3.4.8	Optimizaciones	28
3.5	DevOps	29
4	Organización y gestión	31
4.1	Organización	31
4.2	Gestión	31
4.3	Valoración económica del proyecto	32

5	Conclusiones	34
5.1	Resultado final.....	34
5.2	Valoración	34
5.3	Proyecto a futuro	34
6	Bibliografía.....	36
A	Análisis	37
A.1	Historias de usuario priorizadas.....	37
A.2	Requisitos no funcionales	38
A.3	Herramientas utilizadas	39
B	Diseño	41
B.1	Diagrama de entidades.....	41
B.2	Diagrama de clases	43
B.3	Diagrama de Navegación	44
B.4	Diagramas de secuencia relevantes	45
C	Evolutivo – Interfaces antes y después.....	48
D	Pruebas	53
E	Documentación de API Web.....	56
E.1	Como leer esta documentación	56
E.2	API de Murcy	56
E.2.1	Usuario	56
E.2.2	Solicitud de editor	58
E.2.3	Workflow.....	60
E.2.4	Pregunta.....	61
E.2.5	Quiz.....	66
E.2.6	Respuesta	72
E.3	API de la aplicación de envío de correos electrónicos	73
F	Documentación de instalación del sistema	76

Tabla de ilustraciones

<i>Ilustración 2.1 SurveyMonkey</i>	6
<i>Ilustración 2.2 Socrative</i>	6
<i>Ilustración 2.3 Kahoot!</i>	6
<i>Ilustración 2.4 Encuesta de SurveyMonkey</i>	7
<i>Ilustración 2.5 Quizzes de Kahoot!</i>	7
<i>Ilustración 2.6 Quizzes de Socrative</i>	7
<i>Ilustración 2.7 Antiguo listado sin paginación</i>	8
<i>Ilustración 2.8 Antiguo login sin errores claros</i>	8
<i>Ilustración 2.9 Canvas físico</i>	13
<i>Ilustración 3.1 Arquitectura de alto nivel de Murcy</i>	15
<i>Ilustración 3.2 Diagrama de entidades sencillo</i>	16
<i>Ilustración 3.3 Diagrama de navegación simplificado</i>	18
<i>Ilustración 3.4 Diagrama de despliegue de Murcy</i>	18
<i>Ilustración 3.5 Pantalla para crear nuevas preguntas</i>	20
<i>Ilustración 3.6 Diseño del menú lateral</i>	20
<i>Ilustración 3.7 Lista de quizzes públicos en la antigua prueba de concepto</i>	20
<i>Ilustración 3.8 Lista de quizzes públicos en la prueba de concepto</i>	20
<i>Ilustración 3.9 Nuevo inicio de sesión</i>	21
<i>Ilustración 3.10 Nueva pantalla de solicitud</i>	22
<i>Ilustración 3.11 Nueva pantalla de listados</i>	23
<i>Ilustración 3.12 Nueva pantalla para crear preguntas</i>	24
<i>Ilustración 3.13 Nueva pantalla de lista de preguntas</i>	25
<i>Ilustración 3.14 Crear quiz nuevo</i>	25
<i>Ilustración 3.15 Seleccionar preguntas nuevo</i>	25
<i>Ilustración 3.16 Ordenar preguntas nuevo</i>	25
<i>Ilustración 3.17 Nueva pantalla de lista de quizzes</i>	26
<i>Ilustración 3.18 Nueva lista de quizzes para jugar</i>	26
<i>Ilustración 3.19 Nueva pantalla información</i>	26
<i>Ilustración 3.20 Nueva pantalla para jugar</i>	26
<i>Ilustración 3.21 Nueva visualización del resultado final</i>	27
<i>Ilustración 3.22 entidades en el sistema mal funcionando</i>	29
<i>Ilustración 3.23 entidades en el sistema funcionando correctamente</i>	29
<i>Ilustración 3.24 Pipeline de los distintos entornos</i>	29
<i>Ilustración 3.25 Pipeline de CI y CD del FrontEnd</i>	30
<i>Ilustración 6.1 Diagrama de entidades completo</i>	42
<i>Ilustración 6.2 Diagrama de clases</i>	43
<i>Ilustración 6.3 Diagrama de navegación completo</i>	44
<i>Ilustración 6.4 Diagrama de secuencia de los filterChain</i>	45
<i>Ilustración 6.5 Diagrama de secuencia del middleWare de autenticación</i>	46
<i>Ilustración 6.6 Diagrama de secuencia del proceso de creación de un usuario</i>	47
<i>Ilustración 6.7 Inicio de sesión antiguo</i>	48
<i>Ilustración 6.8 Inicio de sesión nuevo</i>	48
<i>Ilustración 6.9 Solicitud para ser editor antiguo</i>	48
<i>Ilustración 6.10 Solicitud para ser editor nuevo</i>	48
<i>Ilustración 6.11 Lista de solicitudes antiguo</i>	49
<i>Ilustración 6.12 Lista de solicitudes nuevo</i>	49
<i>Ilustración 6.13 Crear pregunta - antiguo</i>	49

<i>Ilustración 6.14 Crear pregunta - nuevo</i>	49
<i>Ilustración 6.15 Listado de preguntas antiguo</i>	50
<i>Ilustración 6.16 Listado de preguntas nuevo</i>	50
<i>Ilustración 6.17 Crear un quiz antiguo</i>	50
<i>Ilustración 6.18 Crear un quiz nuevo - información</i>	50
<i>Ilustración 6.19 Crear un quiz nuevo - preguntas</i>	51
<i>Ilustración 6.20 Crear un quiz nuevo - orden</i>	51
<i>Ilustración 6.21 Gestión de quizzes antiguo</i>	51
<i>Ilustración 6.22 Gestión de quizzes nuevo y menú contextual</i>	51
<i>Ilustración 6.23 Lista de quizzes para jugar antiguo</i>	52
<i>Ilustración 6.24 Lista de quizzes para jugar nuevo</i>	52
<i>Ilustración 6.25 Jugar un quiz antiguo</i>	52
<i>Ilustración 6.26 Jugar un quiz nuevo - información</i>	52
<i>Ilustración 6.27 Jugar un quiz nuevo - jugar</i>	52
<i>Ilustración 6.28 Postman y las colecciones</i>	53
<i>Ilustración 6.29 entidades en el sistema mal funcionando</i>	54
<i>Ilustración 6.30 entidades en el sistema funcionando correctamente</i>	54

1 Introducción

Hoy en día cada vez es mayor la cantidad de gente que usa las tecnologías digitales para ocio y sus tareas diarias, pero no encuentran una solución que satisfaga sus necesidades. A razón de estas necesidades surge una oportunidad de mercado que ofrezca una herramienta que satisfaga estas necesidades, en este caso los quizzes. Las aplicaciones de quizzes ya existentes como *Kahoot*, a simple vistazo lucen sencillas, pero lejos de la realidad los procesos necesarios para su funcionamiento son complejos y requieren de una gran inversión. [1].

En este TFG se va a desarrollar un producto mínimo viable denominado Murcy, a partir de una prueba de concepto realizada anteriormente en la asignatura de Gestión de Proyecto Software, para comprobar con usuarios reales si soy capaz de desarrollar un sistema de quizzes que potencialmente tenga una buena acogida. En caso afirmativo, sería el paso para, en un futuro, de convertir a Murcy en una solución para el uso lúdico de los *quizzes*.

1.1 Motivación

Un *quiz*¹ es una herramienta para compartir conocimiento o entretenimiento entre usuarios u obtener una retroalimentación de otros usuarios. Otra de las características implícitas de los *quizzes*, es que las respuestas son limitadas y pueden ser de opción única o múltiple. Por estos motivos, y dada la propia definición de *quiz*, no se contemplan términos como acertijo o encuesta ya que no hacen alusión al formato lúdico de Murcy. Otro de los puntos importantes del formato *quiz* es que ofrece una recompensa personal instantánea, al acabar de jugar obtienes una retroalimentación de cuál ha sido tu resultado en el *quiz* que acabas de completar. Además, dada su sencillez a la hora de responderlos y de crearlos, simplifica al máximo la carga de los usuarios.

Se pueden distinguir distintos tipos de usuarios. De un lado tenemos aquellos que van a usar la aplicación con un rol principalmente lúdico y, de otro lado tenemos, aquellos cuyos fines no van a ser completamente lúdicos. Los primeros buscan jugar, crear y compartir *quizzes*. Los segundos persiguen crear *quizzes* para satisfacer otro tipo de necesidades personales, como, por ejemplo, estudiar, o aprender sobre determinados temas que otros usuarios hayan creado.

1.2 La evolución de los *quizzes* en la Web

Hace 15 años para poder crear un *quiz* y hacerlo llegar a los usuarios existía una empresa que tenía que diseñarlo, fabricarlo y distribuirlo. En la actualidad, las tareas que anteriormente se debían hacer en formato físico se están adaptando a un formato

¹ Se hace uso del anglicismo *quiz* / *quizzes*, dado que representa el termino usado por los potenciales usuarios a los que va dirigida la aplicación.

digital como, por ejemplo, hacer la lista de la compra, tomar apuntes, comunicación, etc. Por estos motivos, los juegos también deben tomar este rumbo. En el caso de Murcy, este va a ser nuestro objetivo, ofrecer la posibilidad a los usuarios de crear y compartir sus *quizzes* rápidamente.

Si analizamos los estudios más recientes disponibles [2], se observa que la cuota de mercado de los dispositivos móviles es del 92%. Por lo tanto, tiene más sentido para poder alcanzar a la mayor cantidad de usuarios crear una aplicación para dispositivos móviles, pero sin cerrar la puerta a otras plataformas como, ordenadores personales o Tablet.

1.3 Objeto del proyecto

El objeto del proyecto es diseñar y desarrollar un producto mínimo viable de *quizzes*, para hacer pruebas con usuarios, tomando como base una prueba de concepto desarrollada previamente en la asignatura de Gestión de Proyecto Software. Esta tiene multitud de problemas con los que sería inviable poder realizar pruebas con usuarios. Para poder solventar estos problemas hay que evolucionar sistema mediante el uso de nuevas tecnologías, nuevos patrones e implementación de nuevas interfaces para poder ofrecer a los usuarios el mejor servicio posible, rápido y seguro. Además, de resolver los problemas de la prueba de concepto, el producto mínimo viable tendrá aquellas funcionalidades necesarias para conseguir unas pruebas reales y que aporten toda aquella información que necesito para convertir a Murcy en una aplicación completa.

1.4 Alcance

Al iniciar el TFG, la prueba de concepto Murcy era una aplicación de preguntas y respuestas online, desarrollada con React Native y Spring Boot, con problemas de rendimiento y falta de documentación, dado que se trataba de una prueba de concepto realizada para una asignatura.

Al finalizar este trabajo, el producto mínimo viable Murcy se habrá desarrollado en una tecnología más adecuada, se habrá mejorado la usabilidad de la propia aplicación y se habrán realizado las correspondientes mejoras para solucionar los problemas de rendimiento.

Queda incluido dentro del ámbito de este proyecto realizar un análisis de la situación de partida de Murcy para identificar las funcionalidades que se desean utilizar en el producto mínimo viable. Así mismo identificar todos los problemas que provocan los fallos de rendimiento para poder corregirlos mediante nuevos patrones de diseño. Además, se añadirán todas aquellas funcionalidades necesarias para convertir a Murcy en un producto mínimo viable con el que realizar pruebas con usuarios reales. Estas actividades se pueden realizar dentro del marco temporal del TFG.

En ningún caso, queda dentro del ámbito del este proyecto convertir a Murcy en una aplicación completa con la que competir y generar beneficios.

1.5 Estructura de la memoria

La memoria cuenta con un primer apartado de análisis, en el que se explica todo el proceso de investigación y estudio sobre las aplicaciones similares del mercado y el proceso que se debe seguir para acabar el proyecto con éxito. A continuación, se habla sobre el desarrollo e implementación del proyecto, entrando más en detalle de todos los pasos seguidos durante el proceso. Seguidamente se encuentran los resultados y conclusiones del proyecto, donde se puede leer como ha resultado el proyecto tras finalizar el tiempo de ejecución y su consecuente reflexión. Finalmente, en los anexos se puede encontrar la información al completo del proyecto como los diagramas completos, los requisitos definidos para Murcy, diagramas relevantes para conocer el funcionamiento de la aplicación, la documentación de la API, los manuales de instalación e imágenes del proceso evolutivo.

2 Análisis

Esta sección presenta el análisis realizado de todos los aspectos influyentes en el resultado final de este proyecto. En primer lugar, se ha realizado un estudio de las soluciones de *quizzes* que existen en el mercado, con el objetivo de analizar y extraer ideas para Murcy. A continuación, se ha realizado el análisis de la prueba de concepto para extraer aquellos requisitos que han sido validados en esta, así como todos los problemas que impedían realizar pruebas con usuarios reales. Tras este, se describe la metodología de trabajo seleccionada para este proyecto, así como las herramientas que se deben utilizar para poder completar con éxito el proyecto. Y finalmente se ha realizado un estudio de los riesgos existentes y las correspondientes acciones de mitigación planificadas.

2.1 Estudio de alternativas

A continuación, se comparan los distintos estilos de *quizzes* que se ofrecen en la Web para la captura de nuevas ideas para el producto mínimo viable. Se han seleccionado 3 empresas que son representativas de distintos tipos de *quizzes*. Tras este análisis se podrá extraer una conclusión de que debería incluir Murcy para competir en el mercado.



Ilustración 2.1 SurveyMonkey



Ilustración 2.2 Socrative



Ilustración 2.3 Kahoot!

SurveyMonkey [3]: dentro de las opciones analizadas, se trata de la solución más empresarial, cuyo principal nicho de negocio es la obtención de información mediante el uso de encuestas en formato *quiz*. Ofrece planes gratuitos muy limitados, planes individuales y planes empresariales. Su principal fuente de obtención de ingresos es la venta de su producto a particulares y empresas. Ver Ilustración 2.4.

Socrative [4]: se trata de una solución orientada a la docencia, donde es el profesor el encargado de crear los *quizzes*, y los alumnos de responderlos. Posee opciones para crear entornos donde los alumnos responden en tiempo real a las preguntas, y opciones para que el profesor pueda analizar estas respuestas en tiempo real. Ofrece un plan gratuito muy completo, pero ofrece planes más completos, en los cuales se establece la cantidad de alumnos por entorno, cantidad de entornos creados, herramientas extras para analizar los datos, etc. Ver Ilustración 2.5.

Kahoot! [5]: se trata de una solución orientada a la gamificación, donde se incentiva la conectividad entre los participantes. Ofrece *quizzes* en tiempo real. Al igual que los planes anteriores ofrece planes gratuitos que limitan la cantidad de participantes simultáneos y ofrece packs de pago donde amplían estos límites y nuevos tipos de preguntas. Ver Ilustración 2.6.

1. Estimado Docente elija la opción.

- Pre- escolar
- Primer ciclo básico
- Segundo Ciclo
- PIE

* 2. Estimado profesor

Al terminar el presente año, necesitamos evaluar cuanto hemos avanzado en las metas propuestas en el Proyecto Educativo Institucional (PEI). Es por ello que solicitamos que respondas esta encuesta que nos permitirá conocer como estas percibiendo las acciones realizadas en pos del avance y mejoramiento de la escuela de Hualpén.

Al responder la encuesta marque frente a cada indicador, considerando la siguiente escala:

	Muy En desacuerdo	Desacuerdo	De Acuerdo	Muy de acuerdo
2- Existe una organización de los padres y apoderados, asesorada por docentes.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3- Existe una organización de estudiantes en funcionamiento, asesorada por docentes.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Ilustración 2.4 Encuesta de SurveyMonkey

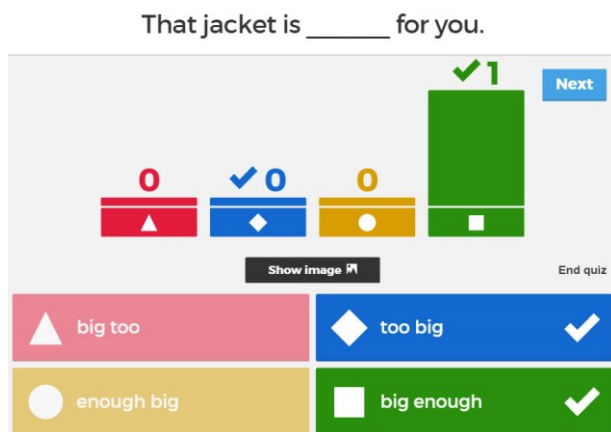


Ilustración 2.5 Quizzes de Kahoot!

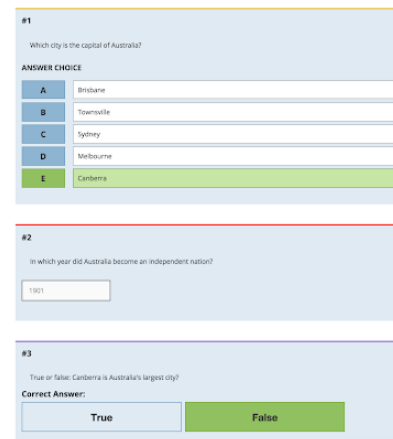


Ilustración 2.6 Quizzes de Socrative

Como conclusión tras haber analizado el mercado existente, se observa que las soluciones que ofrecen modelos basados en *quizzes* son muy **intuitivas** de utilizar, por lo que el Producto mínimo viable debe ser fácil de usar para los usuarios teniendo una interfaz sencilla y cuidada. Por otro lado, encontramos la falta de una aplicación de *quizzes* que permita **compartirlos** con intenciones lúdicas, por lo que voy a incluir esta característica en el producto mínimo viable.

2.2 Estudio de la prueba de concepto

Además del análisis de las aplicaciones ya existentes, se ha realizado un análisis de todas las pantallas existentes en la prueba de concepto y de su funcionamiento. Las principales conclusiones de este análisis son:

- **Excesiva carga cognitiva:** existen pantallas innecesarias que provocan al usuario tardar más en realizar acciones sencillas y repetitivas.
- **La navegación no es intuitiva:** la URI de la aplicación no cambia en ningún momento y las acciones de retroceso y avance dentro de las opciones de navegación del navegador o dispositivos móviles no funcionan correctamente, únicamente se puede avanzar o retroceder mediante la interacción con la aplicación.
- **Falta de información en los formularios:** los errores no son claros, la mayoría no aclaran que ha provocado el error.
- **Falta de paginación:** en caso de existir gran cantidad de elementos la carga de trabajo y su consecuente temporalidad son muy altas.
- **Falta de usabilidad:** No se puede utilizar correctamente la aplicación desde dispositivos móviles.

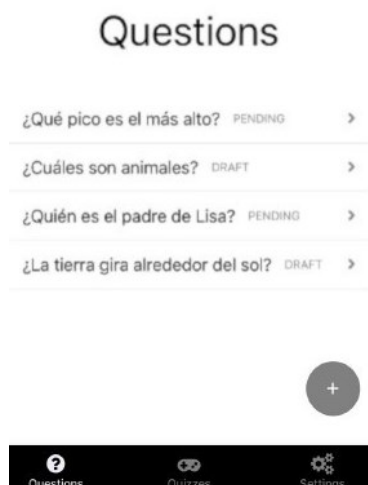


Ilustración 2.7 Antiguo listado sin paginación

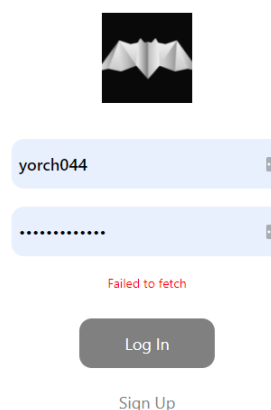


Ilustración 2.8 Antiguo login sin errores claros

En la Ilustración 2.7 se puede observar la falta de paginación y en la Ilustración 2.8 se puede observar la falta de información en los formularios.

Por lo anterior, es inviable utilizar la aplicación desarrollada durante la prueba de concepto para realizar pruebas con usuarios reales.

2.3 Requisitos de partida

Como señalamos en la introducción, el objetivo de este TFG es crear un producto mínimo viable para poder realizar pruebas con usuarios reales para ver su viabilidad como aplicación en el mercado manteniendo la funcionalidad validada en la prueba de concepto, pero reimplementadas en una tecnología más apropiada y añadir aquella funcionalidad extra necesaria. En consecuencia, gran parte de los requisitos funcionales proceden de la prueba de concepto realizada en la asignatura de Gestión de Proyecto Software, la cual permite a los usuarios crear *quizzes*, pero que no estaba pensada para usuarios reales. Uno de estos problemas era la no existencia de un rol de administración que pueda gestionar el completo de la aplicación como por ejemplo crear nuevos editores. Además, se introducen nuevos requisitos no funcionales para satisfacer los nuevos criterios de calidad.

Una aplicación de este estilo necesita cubrir los siguientes roles que son comunes a todas las alternativas existentes y a la oportunidad identificada (ver 2.1):

Tabla 1 Roles de la aplicación

Rol	Descripción
Usuario sin registrar	Aquella persona que aún no posee una cuenta en la aplicación de <i>quizzes</i> .
Usuario	Aquella persona que ya posee una cuenta en la aplicación y cuya acción principal va a ser interaccionar con los <i>quizzes</i> públicos.
Editor	Es un <i>usuario</i> que, además, tiene habilitadas las herramientas de creación de <i>quizzes</i> .
Revisor	Es un <i>editor</i> que tiene habilitadas las herramientas de moderación, mediante las cuales, poder decidir que <i>quizzes</i> son públicos y cuáles no.
Administrador	Aquella persona con acceso a todas las herramientas de la aplicación y al panel de gestión de esta, y que además puede actuar como revisor sin ser usuario.

Los roles anteriores esperan poder realizar una serie de tareas en la aplicación. Parte de estas tareas se han validado en la prueba de concepto y otras proceden de las alternativas analizadas. Estas las podemos describir usando la técnica denominada *Historias de Usuario*. Las actividades que realizan los roles se describen con la fórmula “Como <rol>, quiero <actividad> para <objetivo satisfecho>”. Además, han sido priorizadas según el método MoSCoW [6] (ver Anexo A.1).

La siguiente lista contiene las historias de usuario más relevantes identificadas que el prototipo debe ser capaz de dar soporte.

- Como usuario sin registrar, quiero poder crearme una cuenta para usar Murcy.
- Como usuario, quiero poder iniciar y cerrar sesión en la aplicación para poder jugar *quizzes*.
- Como usuario, quiero poder solicitar acceso a las herramientas de edición para poder crear mis propios *quizzes*.
- Como usuario, quiero poder jugar a los *quizzes* para pasar un buen rato.
- Como editor, quiero poder crear, editar y eliminar mis propias preguntas para poder añadirlas a un *quiz*.
- Como editor, quiero poder crear, editar y eliminar mis propios *quizzes* para poder compartir *quizzes* y entretener a los usuarios.
- Como editor, quiero poder tener *quizzes* privados y públicos para publicar únicamente aquellos *quizzes* que quiera.
- Como revisor, quiero poder aprobar o denegar una petición para publicar un *quiz* para mantener la calidad de la aplicación.
- Como revisor, quiero poder aprobar o denegar una solicitud de acceso a las herramientas de edición para poder mantener la calidad de la aplicación.

Además, se han identificado los siguientes requisitos no funcionales surgidos del análisis realizado a la prueba de concepto (ver 2.2). Para resolver la excesiva carga cognitiva se debe ofrecer una buena experiencia de usuario haciendo que el acceso a la aplicación sea instantáneo y la carga de datos lo más rápida posible. Además, para poder garantizar la seguridad de los datos de los usuarios, la comunicación entre la aplicación y el servidor debe estar realizada mediante protocolo HTTPS y el login debe ser seguro, no exponiendo ni almacenando los datos personales del usuario en entornos no seguros. Finalmente dado el bajo presupuesto para realizar este producto mínimo viable, se deben reducir al mínimo los costes de desarrollo y mantenimiento.

A continuación, vamos a revisar en detalle el alcance de los requisitos no funcionales relacionados con la gestión y la tecnología y cómo se han implementado.

2.4 Metodología de gestión

Para el desarrollo de este prototipo la técnica más adecuada es una metodología *Agile*, dada la existencia de iteraciones sobre el propio prototipo que va a tener un ciclo de vida variable y debe poder sufrir cambios a lo largo de su desarrollo. Además, esta metodología está muy bien documentada y como desarrollador del proyecto dadas experiencias profesionales se cómo trabajar bajo este tipo de metodologías.

El único inconveniente de este tipo de metodología es que está orientada al trabajo en equipo, pero en este caso al ser una única persona, han sido necesario cambiar procedimientos / reuniones que son habituales en esta metodología. La Tabla 2 recoge estos cambios.

Tabla 2 Procedimientos de la metodología Agile

Original	Adaptación	Descripción
Planificación de <i>sprint</i>	Planificación de <i>sprint</i>	Se mantiene la planificación del <i>sprint</i> de forma unipersonal.
Reunión de Grooming	Reflexión de Grooming	Se mantiene la sesión de grooming, pero en vez de ser en formato reunión, será a modo de reflexión personal.
Pila de producto	Pila de producto	Se mantiene la idea original.
Historias de usuario	Historias de usuario	Se mantiene la idea original, pero se encuentran priorizadas (ver 2.3).

Al inicio del proyecto se estimó un tiempo de desarrollo útil de un mínimo 3 meses organizado en *sprints* de 3 semanas si no había incidencias notables (para ver el resultado final de la planificación del proyecto ver 4). Un *sprint* es un tiempo de desarrollo durante el cual se realizan tareas, por definición son breves y están acotados temporalmente. Por otro lado, un lanzamiento es un compromiso por tener lista una parte del proyecto en una cantidad de *sprints* determinada. Dada la cantidad de *sprints* que se plantean, se ha decidido que lo mejor es establecer tres lanzamientos.

Lanzamiento 1 – análisis previo y documentación: en este lanzamiento se habrán generado los artefactos de documentación de instalación de la aplicación de partida y la aplicación actual. Se habrán creado los entornos de desarrollo y toda la configuración requerida para la integración continua, despliegue continuo y análisis estático de código. En lo relacionado a la migración, se habrá tenido que hacer un análisis de las pantallas existente y se habrá realizado el nuevo diseño de estas. Y finalmente en lo relacionado con desarrollo, los usuarios sin registrar se podrán registrar en la aplicación e iniciar sesión en la misma. [1 *sprint*]

Lanzamiento 2 – Evolutivo de la prueba de concepto: en este lanzamiento se entregará la primera versión de la aplicación en la cual se podrá jugar a los *quizzes*, finalizando así, la migración de la aplicación previa. [2 *sprints*]

Lanzamiento 3 – nueva funcionalidad, pruebas y depuración: en este lanzamiento se asegura que el código de negocio este seguro, mediante una correcta validación del código, con el fin de que, en futuros lanzamientos, no exista riesgo de poder estropear una funcionalidad antigua. Además, todos los requisitos referidos al usuario básico deben estar implementados y funcionando. [1 *sprint*]

2.5 Herramientas y tecnologías

Esta sección presenta un resumen del análisis de las tecnologías que se han seleccionado. Si se quiere ver este informe en detalle ver Anexo A.3 para ver el desglose completo de herramientas y tecnologías. Este análisis está organizado según el ámbito: Aplicación (Servidor y aplicación web), infraestructura y despliegue, desarrollo y gestión.

Diseño e implementación del servidor web: Las siguientes herramientas y tecnologías han sido utilizadas para realizar el desarrollo del BackEnd de la aplicación.

Servidor Web	
Spring Boot	Excelente framework para desarrollar aplicaciones Java. Se posee conocimientos de esta tecnología.
PostgreSQL	Base de datos relacional de código abierto muy potente y con gran versatilidad.

Diseño e implementación de la aplicación web: Las siguientes tecnologías han sido utilizadas para realizar el desarrollo del FrontEnd de la aplicación.

Aplicación Web	
Angular	Tecnología utilizada para el desarrollo de la aplicación web. Se ha elegido dada su gran versatilidad y potencia a la hora de crear aplicaciones web.
Angular material	Framework para el desarrollo de aplicaciones web en angular.

Construcción del software: Estas herramientas y plataformas han sido utilizadas para agilizar y asegurar el desarrollo y las pruebas de la aplicación.

Construcción del software	
Docker	Tecnología utilizada para acelerar y agilizar los despliegues y las pruebas en local.
Travis-CI	Plataforma utilizada para realizar integración y despliegue continuo.
SonarCloud	Plataforma utilizada para mantener la calidad y seguridad del código.

Despliegue e infraestructura de las aplicaciones: Las siguientes tecnologías han sido utilizadas para poder desplegar y usar las aplicaciones desarrolladas en entornos reales.

Despliegue / infraestructura	
Node.js	Tecnología utilizada para hacer que el FrontEnd funcione bien desplegado.
Java	Utilizada para los desarrollos. Se ha elegido dado a los conocimientos previos de la misma

Desarrollo del proyecto: Las siguientes herramientas han sido utilizadas para poder desarrollar el proyecto.

Desarrollo	
Canvas físico	Una superficie física donde poder gestionar el progreso de los distintos <i>sprints</i> (ver Ilustración 2.9).
Git	Herramienta de control de versiones utilizada para la gestión de cambios.

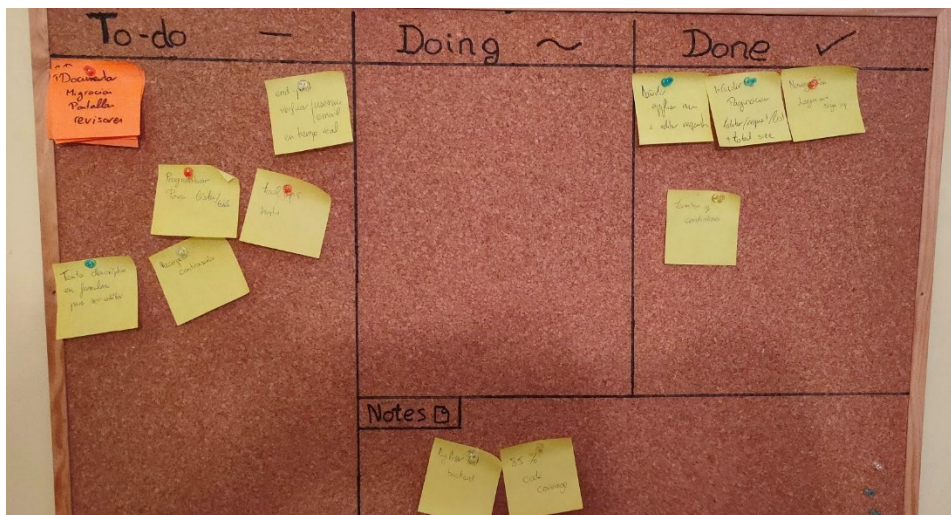


Ilustración 2.9 Canvas físico

2.6 Riesgos

La gestión del riesgo se refiere a reducir la probabilidad e impacto de los eventos adversos en el proyecto. Dado que el proyecto se va a gestionar con una metodología agile, su carácter iterativo, implícitamente hace que la gestión del riesgo forme parte del ciclo de vida del proyecto. Aun así, en esta fase de análisis se han identificado los siguientes riesgos y su mitigación, que ha sido implementada.

Tabla 3 Riesgos detectados y su correspondiente mitigación

Original	Descripción	Mitigación
Fallo del hardware de desarrollo	Falle el hardware personal utilizado para desarrollar el proyecto.	Invertir en nuevo hardware para el proyecto y su uso personal.
Caída de las plataformas de despliegue online	Caída de las plataformas de despliegue en remoto (ej.. Heroku).	Para las pruebas, tener preparada la aplicación para usar el entorno local aun cuando se despliegue en remoto. Para la aplicación realizar un rápido despliegue en un proveedor similar.
Caída de los servicios de terceros necesarios	Los servicios de terceros dejen de funcionar (ej.. Gmail).	Utilizar otro servicio similar que nos ofrezca el mismo tipo de solución.

Original	Descripción	Mitigación
Falta de tiempo por trabajo	Dado que se he comenzado a trabajar a tiempo completo no se sabe si esto repercutirá en la rapidez con la que podré realizar el TFG.	Alargar el tiempo de desarrollo del TFG según se requiera. La metodología utilizada nos lo permite.
Conocidos desconocidos	Dado que se va a aprender una nueva tecnología es posible que durante el comienzo del desarrollo se comentan fallos.	Crear una tarea para solucionar los problemas del desarrollo inicial con los conocimientos actuales
Desconocidos no conocidos	Son cosas que ni siquiera sabías que debías averiguar. Son cosas no identificadas.	Una vez identificadas, se les asignará una prioridad y se solucionará en su debido momento.

Un ejemplo de cómo se ha aplicado este análisis de riesgos fue un incidente originado por la caída puntual de la plataforma Heroku utilizada para realizar pruebas. El servicio de login de esta plataforma dejó de funcionar [7] y por lo tanto no se podían hacer pruebas online. En consecuencia, se aplicó la política mitigación: realizar pruebas de forma local, ya que había soporte para ello.

3 Solución desarrollada

En esta sección se recoge todo el proceso de convertir la prueba de concepto inicial en un producto mínimo viable con el cual poder realizar pruebas con usuarios reales (ver 2.3).

3.1 Arquitectura

Murcy se compone de una aplicación cliente muy ligera y de un BackEnd con controladores, servicios y repositorios. Los controladores son los encargados de captar las peticiones. Los servicios se encargan de la lógica de negocio. Los repositorios sirven para encapsular el acceso a la capa de persistencia de datos, en la que se utiliza una base de datos relacional para almacenar la información de la aplicación. Otro de los puntos que se pueden observar es la comunicación con el servicio de correo electrónico explicado en el punto 3.4.7. Para poder ofrecer una visual de cómo es Murcy, a continuación, se muestra el diagrama de arquitectura de la solución desarrollada. Se puede observar la comunicación entre los distintos componentes del sistema y su clara separación entre su funcionalidad.

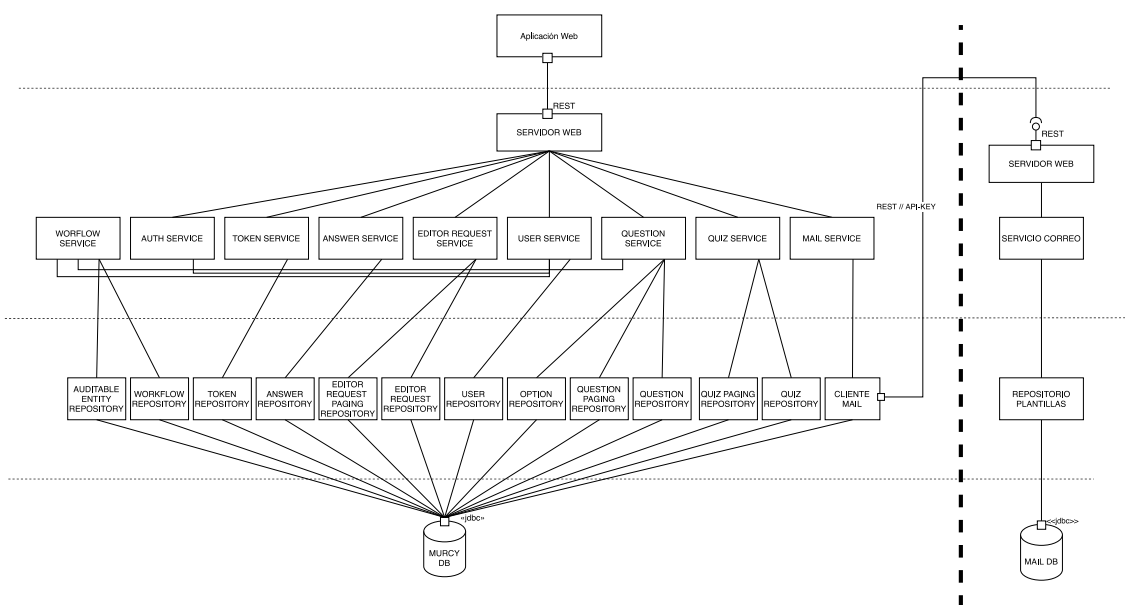


Ilustración 3.1 Arquitectura de alto nivel de Murcy

3.2 Subsistemas

A lo largo de esta subsección se ofrece una perspectiva más detallada del sistema, mostrando los diagramas necesarios para su comprensión. Estos diagramas han sido simplificados para ofrecer una visión sencilla del sistema, en caso de querer profundizar en el siguiente punto se recomienda leer el anexo B6B.

3.2.1 Modelo de datos

Dentro de Murcy, podemos diferenciar las entidades principales involucradas en el proceso de negocio, estas entidades son:

Entidad	Papel dentro del sistema
Usuario	Entidad encargada de guardar la información que permite identificar a una persona que utiliza la aplicación.
Pregunta	Entidad encargada de mantener la información sobre las preguntas creadas por los usuarios.
Opciones	Objeto valor de entidad, esta entidad representa las distintas respuestas posibles en una pregunta. Sin la pregunta, esta entidad carece de persistencia en el sistema.
Quiz	Entidad encargada de mantener la información de un <i>quiz</i> , al igual que todas las preguntas añadidas.
Respuesta	Entidad encargada de mantener la información de las respuestas de los <i>quizzes</i> .
Solicitud de editor	Entidad encargada de mantener la información de las peticiones realizadas por los usuarios para ser editores.

Y a continuación, se presenta un diagrama simplificado representado estas entidades (para consultar el modelo más detallado o el diagrama de clases para tener una visión más profunda del sistema, estos se encuentran en los anexos B.1 y B.2).

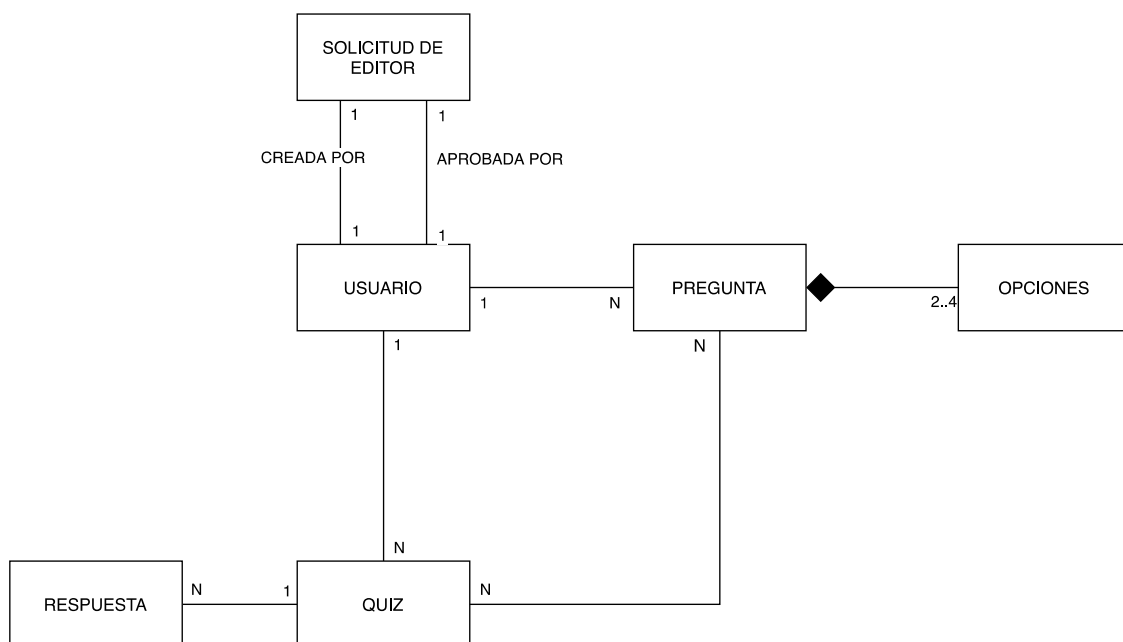


Ilustración 3.2 Diagrama de entidades sencillo

3.2.2 Servicios y lógica de negocio

Atendiendo al diagrama de la subsección anterior (ver Ilustración 3.2) y el diagrama de arquitectura de alto nivel (ver Ilustración 3.1) se va a explicar cuál es la correspondencia entre dichas las entidades y los servicios. Para simplificar este apartado en aquellos puntos que se indique **[entidad]**Componente (se refiere al componente asociado a la entidad indicada, por ejemplo, UsuarioService hace referencia al servicio encargado del proceso de los usuarios).

Servicio	Pertenencia y relevancia en el proceso de negocio
AuthService	Este servicio es el encargado de asegurar que un usuario únicamente accede a las entidades a las que tiene acceso y no realiza ninguna acción que sobre pase su autorización.
[entidad] Service	Cada uno de estos servicios se encarga del proceso y ciclo de vida de la entidad asociada.
Caída de los servicios de terceros necesarios	Los servicios de terceros dejen de funcionar.
MailService	Este servicio se encarga de ofrecer una interfaz sobre el "MailClient", encapsulando cada una de las opciones como operaciones independientes.
MailClient	Este componente del sistema se encarga de realizar la comunicación con la API encargada de gestionar las plantillas y enviar correos electrónicos.
[entidad] Repository	Son los componentes del sistema encargados de proporcionar una interfaz para realizar operaciones sobre la base de datos.
[entidad] Paging Repository	Son los componentes del sistema encargados de proporcionar una interfaz para realizar búsquedas paginadas de entidades sobre la base de datos.

Cada uno de estos servicios encapsula únicamente el ciclo de vida de la entidad relacionada, es por ello por lo que los servicios realizan una comunicación entre ellos, en caso de necesitar realizar una operación sobre otra entidad.

3.2.3 Navegación de la aplicación web

La navegación por la aplicación viene dada por el rol del usuario, de tal manera que un editor no puede acceder a las vistas de un revisor, pero los revisores sí que pueden acceder a las vistas de un editor. Con el fin de poder comprender la navegabilidad básica de Murcy se muestra el siguiente diagrama de navegación de la aplicación web en formato reducido. Ver Anexo B.3 para consultar el diagrama completo.

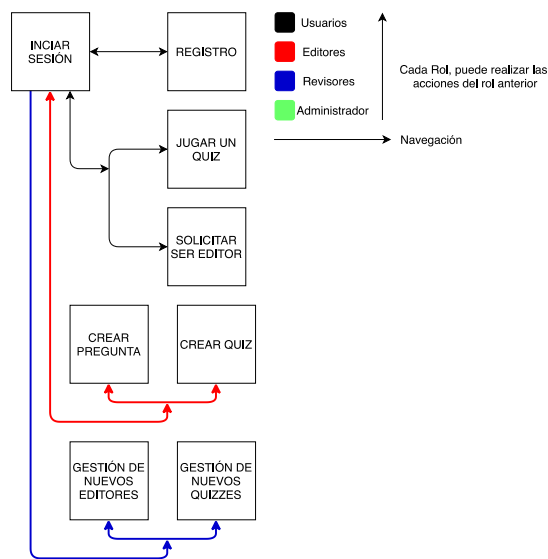


Ilustración 3.3 Diagrama de navegación simplificado

3.2.4 Infraestructura

A la hora de entender el proyecto actual como un producto mínimo viable con el cual se va a hacer pruebas con usuarios reales, hay que hablar de cómo está funcionando la aplicación en el mundo real, con los usuarios reales. Por ello, Murcy cuenta con dos entornos, uno de pruebas, en el que se desarrolla y se prueba la nueva funcionalidad, y el entorno de producción, en el que se encuentra la aplicación estable que usan los usuarios (para más información consultar el punto 3.5). Estos dos entornos se encuentran replicados, y poseen la misma infraestructura.

A continuación, se presenta un diagrama para entender la infraestructura de los entornos:

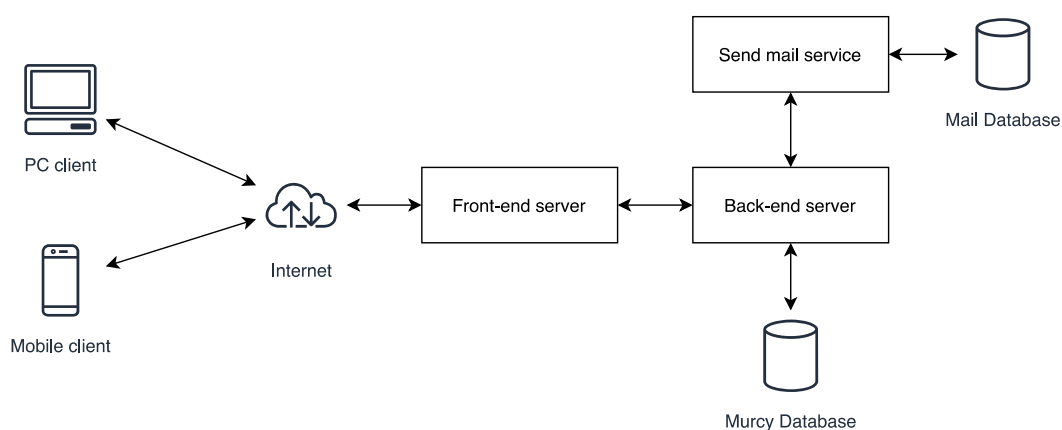


Ilustración 3.4 Diagrama de despliegue de Murcy

3.3 Adaptar prueba de concepto

A la hora de dar comienzo al desarrollo del producto mínimo viable, se ha preparado la prueba de concepto ya existente para funcionar de forma local, mediante el uso de Docker. Este desarrollo es necesario hacerlo inicialmente para poder ofrecer un entorno de prueba rápido e independiente de factores externos.

En este proceso se han creado los siguientes servicios (ver anexo F):

- **Base de datos Postgres:** se ha creado el servicio de base de datos, y se ha modificado la inicialización para que pueda servir tanto a la aplicación antigua, como al producto mínimo viable.
- **Servidor de correo electrónico:** la creación de este servicio es esencial para realizar pruebas y desarrollo en local, y cumple el objetivo de hacer que la aplicación pueda funcionar sin necesidad de tener acceso a servicios externos.
- **Aplicación prueba de concepto (API y aplicación web):** con la creación de este servicio se ha conseguido poder lanzar y usar la aplicación de forma rápida y sencilla, proceso que durante el proceso de migración se va a repetir múltiples veces.

3.4 Evolutivo y nuevos cambios

A lo largo de esta subsección se procede a dar entrada a la fase de desarrollo del evolutivo de convertir la prueba de concepto a un producto mínimo viable. Partiendo de un diseño de interfaces, continuando con las fases propias del evolutivo y terminando con la nueva funcionalidad y pruebas realizadas.

3.4.1 Evolución de la UI

Como primer paso del evolutivo y atendiendo a la planificación realizada en la fase de análisis (ver 2.4), se ha realizado un diseño de las nuevas interfaces de Murcy. Ver la Ilustración 3.5 y la Ilustración 3.6. Tras haber finalizado este primer esbozo a papel de todas las pantallas de Murcy, se ha procedido a ir adaptando la funcionalidad previamente existente a los nuevos requisitos de la aplicación. Ver la Ilustración 3.7 y la Ilustración 3.8

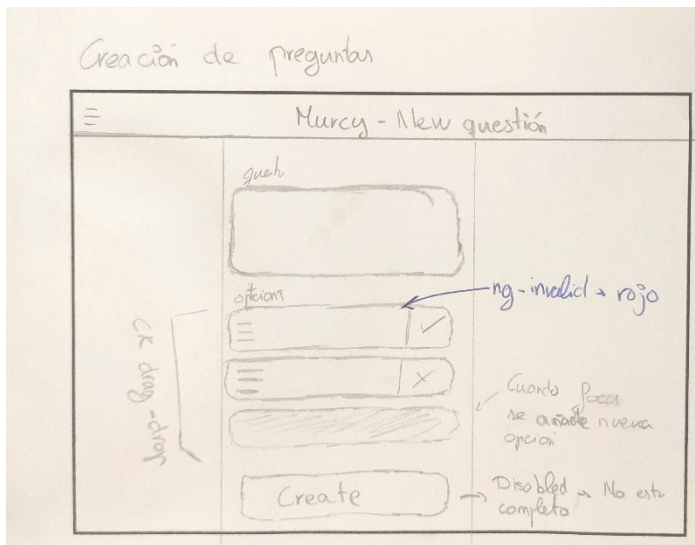


Ilustración 3.5 Pantalla para crear nuevas preguntas

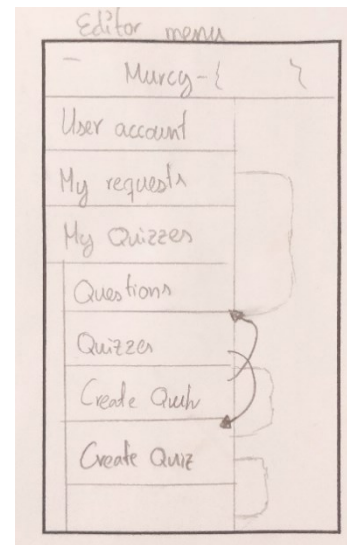


Ilustración 3.6 Diseño del menú lateral



Ilustración 3.7 Lista de quizzes públicos en la antigua prueba de concepto

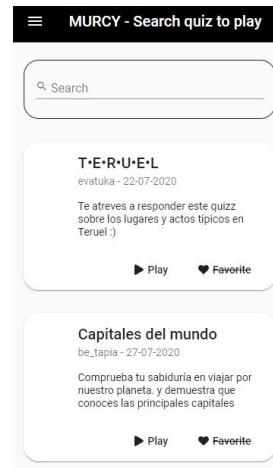


Ilustración 3.8 Lista de quizzes públicos en la prueba de concepto

3.4.2 Registro e inicio de sesión

Como primer paso del evolutivo se ha decidido realizar tanto el registro de nuevos usuarios como su posterior inicio en la aplicación. Esta decisión viene dada por diagrama de navegación presentado en el punto 3.2.3 ya que se trata del punto de entrada de todos los usuarios al sistema.

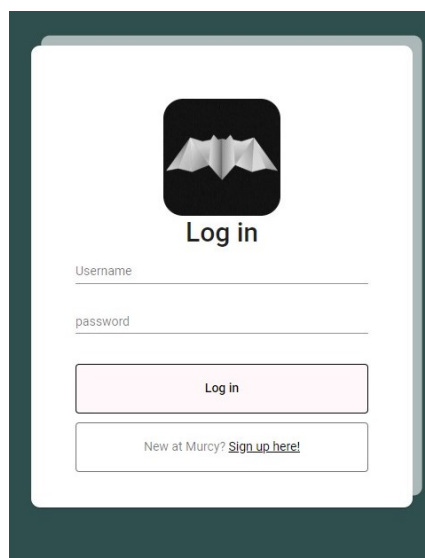


Ilustración 3.9 Nuevo inicio de sesión

Durante este proceso se han añadido controles de errores y aquellas notificaciones necesarias para mantener al usuario informado de cómo han resultado las operaciones realizadas. Algunos de estos cambios han sido:

- **Falta de retroalimentación en los formularios:** en la prueba de concepto únicamente se mostraba el código del error obtenido, sin embargo, en este evolutivo se muestran errores como: usuario/contraseña no correctos, correo electrónico no valida, este correo ya está en uso y este usuario ya existe.
- **Confirmación de registro:** en la prueba de concepto no se informaba al nuevo usuario de la confirmación electrónica, sin embargo, ahora, se muestra una notificación que te invita a comprobar tu bandeja de entrada para poder comenzar a usar la cuenta.

En este proceso la funcionalidad de la API ha sufrido muy pocos cambios, manteniendo en mayor parte la funcionalidad inicial, pero se han realizado los siguientes cambios:

- Mensajes de error traducidos al inglés, en el servicio original, la mayoría de las respuestas se devolvían en inglés, sin embargo, todas las de usuarios se devolvían en castellano.
- Se ha modificado el diagrama de entidades para asegurar la unicidad del nombre de usuario.

- En la aplicación origen existía una tarea programada que liberaba el nombre de usuario creado con dos días de margen, esta tarea no funcionaba como debía y se ha corregido su funcionamiento.

3.4.3 Solicitudes para ser editor

Tras realizar todas aquellas tareas requeridas para completar el primer lanzamiento de la aplicación según el análisis realizado previamente en el punto 2.4, se comienza el segundo, en el cual se requiere que toda la funcionalidad existente en la prueba de concepto se encuentre en el evolutivo desarrollado.

Siguiendo la navegación de la aplicación (ver 3.2.3), el siguiente punto de crítico para su correcto funcionamiento son los editores. Los usuarios no pueden jugar hasta que algún editor haya creado algún *quiz*. Por lo tanto, como primer punto en este segundo desarrollo se va a realizar el desarrollo de la solicitud de nuevos editores. Siguiendo por la gestión de estas solicitudes por parte de los revisores.

Como factor importante añadido a la nueva implementación de la aplicación web, ha sido tener accesible todo el historial de una solicitud, y la posibilidad de modificar el mensaje de esta. Toda esta funcionalidad ya existía en la prueba de concepto, pero no estaba siendo utilizada. De esta forma se refuerza el objetivo de mantener informado al usuario de la evolución de su petición.

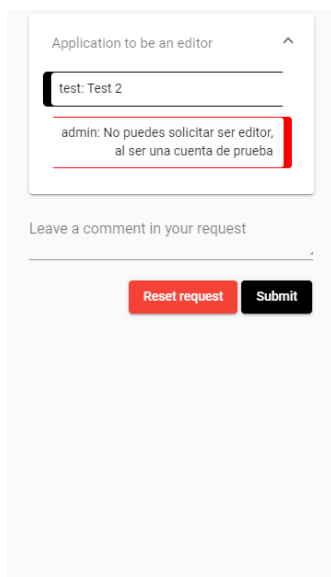


Ilustración 3.10 Nueva pantalla de solicitud

Durante el proceso para la crear una solicitud de editor, en la API únicamente se han modificado aquellas cadenas que estaban en castellano.

Una vez finalizada el evolutivo de la pantalla para solicitar ser un editor, se ha procedido a desarrollar la funcionalidad de aprobar o denegar estas solicitudes.

Al igual que en la pantalla de solicitud, con el objetivo de mantener siempre al usuario, en este caso los revisores, informados, se ha añadido al diseño el historial de solicitudes. Otro de los factores importantes ha sido el desglose de los filtros para aprovechar toda la potencia de estos. En la aplicación origen únicamente se podía filtrar por abiertas y cerradas.

ID	Title	Status
19	Application to be an editor	APPROVED
38	Application to be an editor	APPROVED
42	Application to be an editor	APPROVED
44	Application to be an editor	APPROVED
48	Application to be an editor	APPROVED

Ilustración 3.11 Nueva pantalla de listados

Durante el proceso de migración se ha desechado toda la lógica existente en la API, dado que no ofrecía ningún tipo de paginación, resultando en unas respuestas mucho más pesadas y que hacían que el sistema tuviera unos tiempos de respuestas mayores. Por lo que a las peticiones se han añadido nuevos filtros de búsqueda y opciones de paginación (ver anexo E.2 para leer la documentación al completo).

```
GET /api/request/editor/list?closed=...&approved=...
```

```
GET /api/request/editor/list?closed=...&approved=...&page=...&size=...
```

Todo este cambio ha supuesto que se cambien todas las capas involucradas en la búsqueda de solicitudes: controladores, servicios y repositorios. Este cambio no estaba estimado dentro de los cálculos iniciales de la migración (ver 2.4)

3.4.4 Creación de preguntas

Continuando con el ciclo para que los usuarios puedan jugar, el siguiente paso a realizar es la creación de preguntas.

Atendiendo al requisito de que la aplicación tiene que ser fácil de usar y ayudar al usuario a simplificar el proceso se ha decidido aprovechar al máximo todas las posibilidades que ofrece el framework utilizado, Angular Material, en concreto se ha decidido usar el SDK de arrastrar disponible. Esta herramienta permite reordenar elementos de una lista arrastrándolos a las posiciones deseadas, permitiendo a los usuarios reordenar los elementos sin necesidad de borrar y volver a escribir el contenido. Esta nueva funcionalidad puede verse en la Ilustración 3.12.

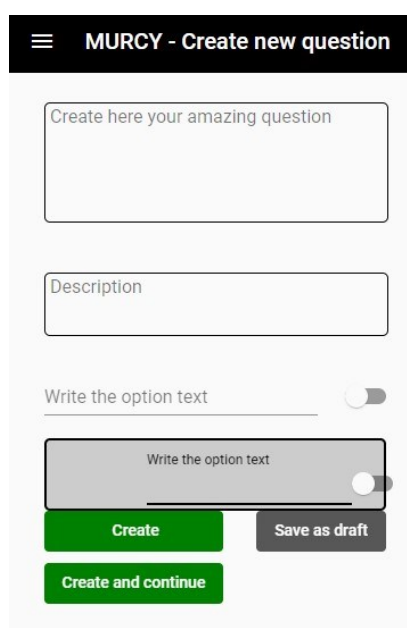


Ilustración 3.12 Nueva pantalla para crear preguntas

Con el fin de permitir a los editores tener control de todas las preguntas que han creado y poder editarlas y eliminarlas, la siguiente pantalla introducida en el proyecto ha sido el listado de preguntas de un usuario.

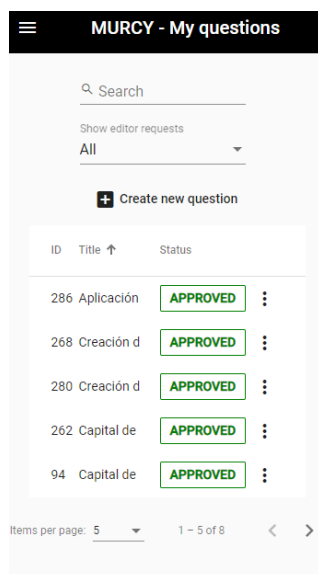


Ilustración 3.13 Nueva pantalla de lista de preguntas

Al igual que en la lista de solicitudes, se ha tenido que cambiar el servidor para poder dar solución al requisito no funcional de peticiones pequeñas.

3.4.5 Creación de *quizzes*

Para finalizar la rama de creación de contenido, faltan los *quizzes* y al igual que en el punto anterior se ha hecho uso del SDK de arrastrar para facilitar el uso de la aplicación a los usuarios.

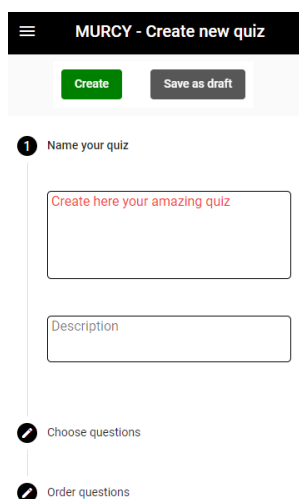


Ilustración 3.14 Crear quiz nuevo

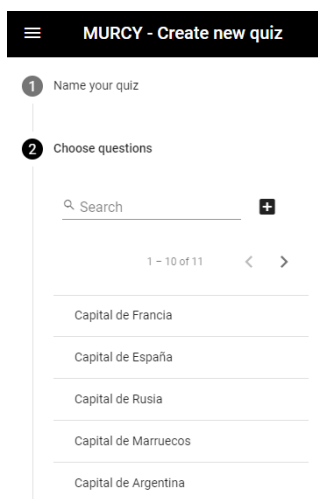


Ilustración 3.15 Seleccionar preguntas nuevo

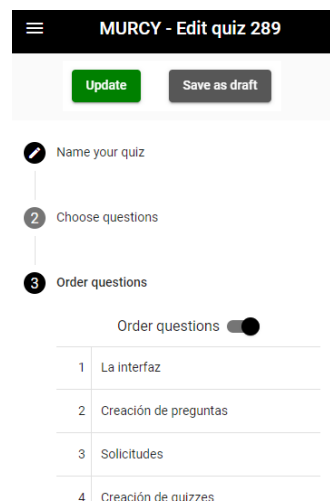


Ilustración 3.16 Ordenar preguntas nuevo

Y con el objetivo, al igual que en puntos anteriores, de permitir a los editores gestionar sus *quizzes* se ha reimplementado manteniendo un diseño común, la lista de *quizzes*.

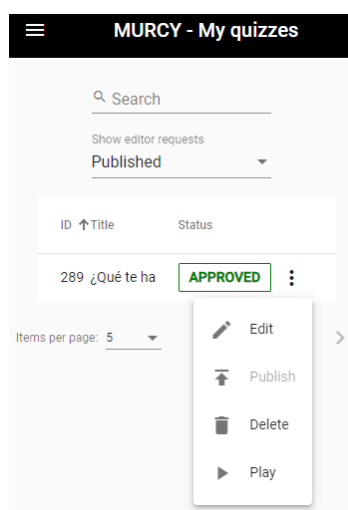


Ilustración 3.17 Nueva pantalla de lista de quizzes

3.4.6 Funcionalidad principal

Una vez introducida toda la funcionalidad para crear contenido, los usuarios quieren poder buscar y jugar a los *quizzes* que los editores han creado. En este punto, se ha realizado un cambio completo respecto a la prueba de concepto, dado que el estilo de juego rompía con todo el estándar estético de la aplicación.



Ilustración 3.18 Nueva lista de quizzes para jugar



Ilustración 3.19 Nueva pantalla información

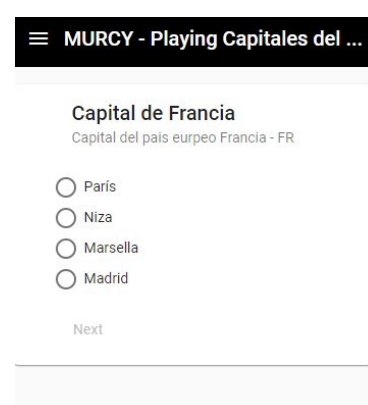


Ilustración 3.20 Nueva pantalla para jugar

Otro de los puntos importantes en el evolutivo de esta funcionalidad era la inexistencia de una lógica de negocio para responder los *quizzes*. Estos se respondían y evaluaban desde la propia aplicación web, impidiendo mantener una trazabilidad de los *quizzes* que un usuario ha respondido y cuál ha sido su resultado en el mismo. Al tratarse de una prueba de concepto esta funcionalidad estaba maquetada pero no implementada.

En el evolutivo actual se crea la respuesta en el servidor y se devuelven el valor de la puntuación obtenida, previamente calculada según las políticas de Murcy.

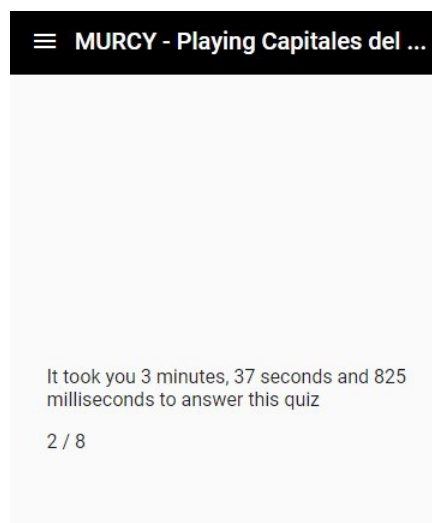


Ilustración 3.21 Nueva visualización del resultado final

3.4.7 Nueva funcionalidad y mejoras

Hasta este punto queda finalizado el segundo lanzamiento, con la completa adaptación de los requisitos validados en la prueba de concepto y comienza la preparación para el tercer y último lanzamiento, que se va a centrar en mejorar la aplicación, rendimiento y depurar la experiencia de usuario.

Con el fin de mejorar la estabilidad de la aplicación se ha decidido separar toda la lógica encargada de enviar correos a un servicio separado de la funcionalidad principal de Murcy, pudiendo de esta manera mantener los dos desarrollos independientes. Aprovechando esta mejora se ha introducido la funcionalidad de gestionar las plantillas desde este mismo servicio. Con este cambio se consigue estabilizar aquellas operaciones que requieran del envío de correos electrónicos.

Para poder hacer esta separación se ha tenido que desarrollar la nueva API, y adoptar un proceso de seguridad para asegurar una comunicación exclusiva entre los dos servicios. Para ello se ha asegurado una comunicación mediante un canal https y con una cabecera de API-KEY que únicamente conocen los servicios implicados. De esta forma se asegura que únicamente se realizan estas operaciones desde Murcy.

Con el fin de mejorar la experiencia de crear *quizzes*, ha sido añadir atajos. En este caso se ha habilitado la opción de crear preguntas desde la creación de los *quizzes*. Habilitando esta opción se permite a los usuarios crear un *quiz* sin abandonar la pantalla de creación en ningún instante.

Otro de las mejoras ofrecidas a la usabilidad de la aplicación por parte de los usuarios ha sido el uso de iconos en toda la aplicación para ayudar a los usuarios a orientarse más fácilmente y ayudar a crear un modelo mental de las acciones a seguir.

3.4.8 Optimizaciones

Y la última mejora introducida a la aplicación, ha permitido reducir la cantidad de llamadas a la base de datos a más de la mitad, mejorando los tiempos de respuesta de 1300 ms a 700ms. El problema venía dado al construir lista de entidades a partir de listas de identificadores, tal que:

```
for id in ids:
    list.add(BD.search(id))
end
```

Se realizaban tantas llamadas como elementos tiene la lista. Esto repercute en el tiempo de ejecución. Para solucionar este problema, se ha implementado la siguiente solución:

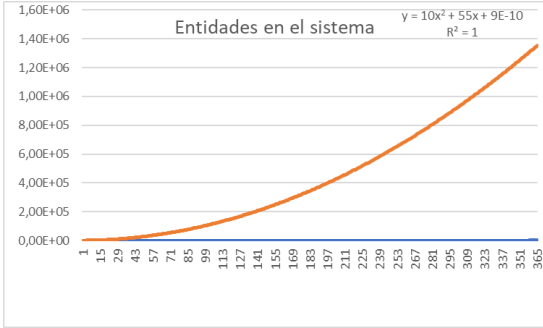
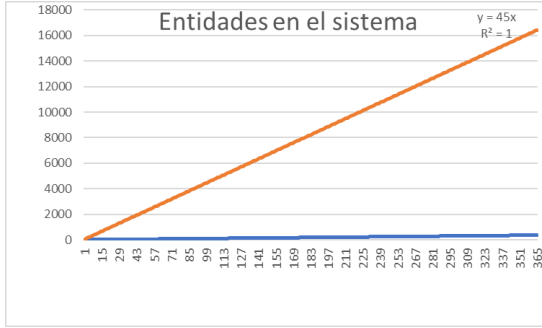
```
lista = BD.search(ids)
lista.sort(ids)
```

Que obtiene una lista buscando las entidades cuyo id este en la lista de ids y posteriormente se ordenan si es necesario.

Durante una prueba de carga, se identificó un gran problema en la lógica de negocio, que, de no haber sido identificada, hubiera supuesto a corto plazo la inviabilidad económica del proyecto, o una pérdida masiva de información. Este problema surge de una prueba realizadas con clientes http que lanzan peticiones REST a la aplicación y estas peticiones se van ampliando hasta un valor estipulado. Durante estas pruebas se observaron crecimientos anómalos en la base de datos, siguiendo un crecimiento potencial. Un crecimiento que debía ser lineal en base a los usuarios se había convertido en un punto crítico del sistema. Para encontrar el error se volvió a realizar este tipo de prueba con un menor crecimiento para observar el punto de fallo. Tras una semana de investigación y pruebas, se consiguió arreglar el problema ocasionado por la librería de persistencia utilizada. A continuación, se procede a explicar el caso de pruebas con el cual se identificó el problema.

Un escenario donde los usuarios van creciendo a un ritmo de 5 nuevos editores cada día, cada día cada editor crea 1 pregunta y modifica otra con 4 posibles respuestas. Al modificar una pregunta, las opciones anteriores son eliminadas y las nuevas añadidas. Para consultar las gráficas ver Tabla 4.

Tabla 4 Funcionamiento real y esperado

Funcionamiento previa corrección	Funcionamiento correcto
 <p><i>Ilustración 3.22 entidades en el sistema mal funcionando</i></p>	 <p><i>Ilustración 3.23 entidades en el sistema funcionando correctamente</i></p>
<p>Formula que describe las estimaciones de entidades en el sistema:</p> $f(x) = 10x^2 + 55x$	<p>Formula que describe las estimaciones de entidades en el sistema:</p> $f(x) = 45x$

3.5 DevOps

A la hora de desarrollar este evolutivo e ir creando los distintos lanzamientos previstos, se han creado dos entornos de trabajo: pruebas y producción. En el primero entorno se irán realizando los nuevos desarrollos mientras que en el de producción únicamente se mostrar los cambios al final de cada uno de los lanzamientos. Atendiendo al requisito de reducir los costes de desarrollo al mínimo posible (ver 4.3) se ha elegido Heroku como herramienta de despliegue. Al comienzo del desarrollo y al finalizar un lanzamiento los entornos se encontrarán en el mismo estado, pero el único entorno que va a sufrir cambios continuos va a ser pruebas.

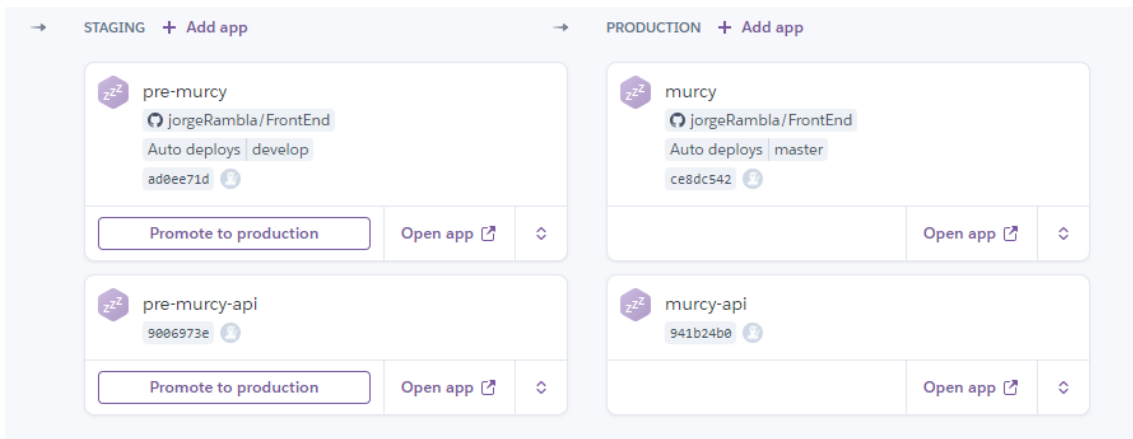


Ilustración 3.24 Pipeline de los distintos entornos

El proceso de integración continua se ha realizado con la herramienta Travis-CI, en la cual se ha realizado un enlace entre la herramienta y cada uno de los repositorios de la aplicación, *FrontEnd* y *BackEnd*.

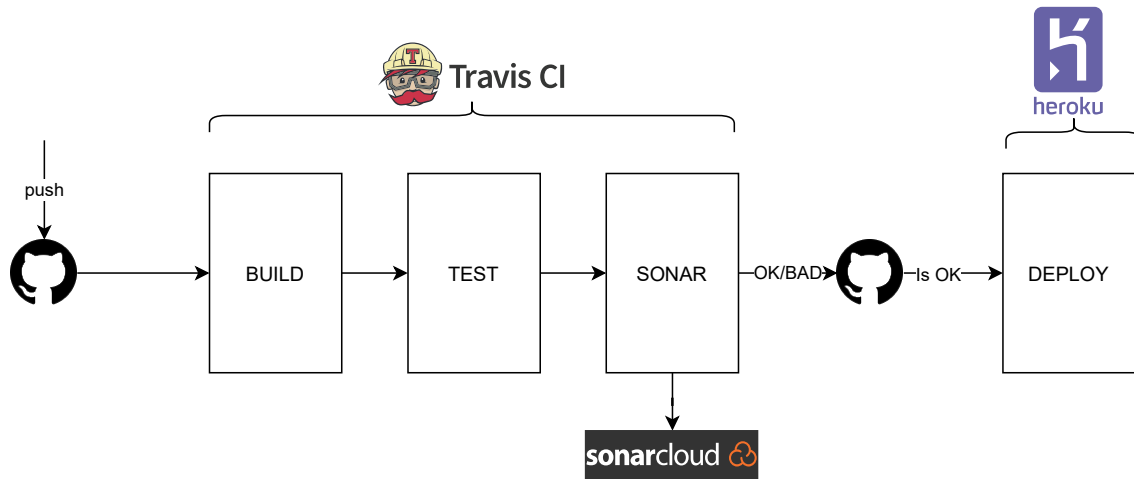


Ilustración 3.25 Pipeline de CI y CD del FrontEnd

Esta *Pipeline* ha sido diseñada e implementada con la aplicación de Travis-CI que permite detectar nuevos cambios en cada uno de los proyectos asociados, analizar y testear el código y en caso de estar correcto se procede a desplegar la aplicación en Heroku o en el caso del FrontEnd avisar a GitHub que las pruebas han ido correctamente para que Heroku pueda desplegar los nuevos cambios.

4 Organización y gestión

A la hora de hablar de organización y gestión hay que volver a pensar en el análisis inicial realizado a la hora de comenzar el proyecto (ver 2.4) para poder ver las diferencias con el resultado de este. A lo largo de esta sección se analizará como ha resultado este prototipo mínimo viable respecto al análisis inicial.

4.1 Organización

Atendiendo a la tabla (ver Tabla 2) de los procedimientos agile y su correspondiente adaptación a un rol unipersonal, se ha conseguido mantener la ejecución de estos provenientes y técnicas a lo largo de todo el desarrollo.

Sin embargo, a lo largo del proyecto se ha encontrado un riesgo desconocido no conocido (ver 2.6), la COVID-19, que ha afectado al desarrollo del proyecto alargando el tiempo de ejecución 3 meses. Gracias a haber utilizado una metodología de gestión agile, se ha podido adaptar los plazos existentes al nuevo alcance temporal.

4.2 Gestión

Al comienzo del proyecto la fase de análisis realizada ha procedido correctamente, siguiendo los plazos inicialmente estipulados, pero tal y como se ha comentado en la sección anterior, a partir de ese instante temporal se ha tenido que adaptar el desarrollo a la nueva situación imprevista.

Este proceso se ha podido realizar fácilmente dada la versatilidad de la metodología utilizada. El segundo lanzamiento pasó de durar un único *sprint* a durar 3, asumiendo un retraso de 1 mes y medio respecto a la planificación inicial. Y el tercer lanzamiento pasó de durar un único *sprint* a durar 2 *sprints*, volviendo a asumir otro retraso de 3 semanas.

A la hora de realizar el análisis de riesgos se determinó que el hardware de desarrollo podía fallar o no cumpliera los requisitos mínimos. Y así sucedió, la RAM del dispositivo de desarrollo era lenta e insuficiente. Por ello entendiendo a la mitigación propuesta para este riesgo, se adquirió más RAM para poder desarrollar este proyecto.

Como punto fundamental de la gestión del proyecto ha estado el desglose de las historias de usuario en tareas fácilmente manejables. Este proceso de *Grooming* comenzó siendo difícil de realizar individualmente, dado que siempre lo había realizado en grupo. Pero tras las primeras sesiones el resultado fue similar a las reuniones realizadas en otras experiencias *Agile*.

4.3 Valoración económica del proyecto

A la hora de hablar de una valoración económica, dejar claro que los análisis realizados a continuación corresponden a crear un producto mínimo viable, y en ningún momento se va a hablar del coste real de la puesta en funcionamiento de Murcy como aplicación completa, dado el desconocimiento de la carga de usuarios finales que llegaría a usar la aplicación.

Dentro de este análisis, se analiza el costo de los dos entornos utilizados por la aplicación, el entorno de pruebas y desarrollo, en el que se intentan reducir los costes a lo mínimo posible, y el entorno de producción, en cual se busca un funcionamiento rápido para los usuarios.

Dados los requisitos relacionados con la velocidad de la aplicación y de minimizar los costes se ha decidido utilizar un servicio de PaaS (*Plataforma As A Service*), en concreto, se ha decidido usar Heroku [8], dado que ofrece un tier gratuito donde desplegar nuestros servicios. Este tier gratuito, tiene limitaciones respecto a un tier superior de pago, como que, tras 30 minutos sin actividad, el servicio hiberna, hasta que vuelve a detectar actividad. Y en el caso del entorno (ver 3.5) de producción, donde se van a realizar pruebas con usuarios reales, se ajusta a un despliegue de bajo coste sin retardos por inactividad. A continuación, se adjunta una tabla con el desglose de precios de los distintos entornos:

Entorno de desarrollo		
API	Tier gratuito	0 \$
PostgreSQL DB	Tier "Dev"	0 \$
Aplicación Web	Tier gratuito	0 \$

Entorno de producción		
API	Tier "Hobbie"	7 \$
PostgreSQL DB	Tier "Hobbie basic"	9 \$
Aplicación Web	Tier gratuito	7 \$

Máquinas compartidas		
API	Tier gratuito	0 \$
PostgreSQL DB	Tier "Dev"	0 \$

Otro de los puntos importantes a la hora de realizar este producto mínimo viable es su coste de desarrollo. De esta forma se puede conocer cuánto costaría desarrollar Murcy como solución, incluyendo el mantenimiento. A continuación, se realiza una estimación de horas en base a las distintas fases del desarrollo del producto mínimo viable:

La siguiente estimación se realiza en base a la experiencia laboral previa con un coste de 14.21 € la hora ($[(17500 \text{ (salario)} / 1600 \text{ (horas)}) * 1.3 \text{ (S.S.)}]$). También se ha realizado la estimación del coste si se hubiera encargado a un desarrollador externo, con un coste de 35 € la hora. Para visualizar esta comparación se presenta la siguiente tabla:

Tabla 5 Valoración de costes

Desarrollo	Horas	Coste 14,21 €/H	Coste 14,21 €/H
Análisis previo	27	383,67 €	945,00 €
Trabajo previo	32	454,72 €	1.120,00 €
Evolutivo	148	2.103,08 €	5.180,00 €
Nueva funcionalidad	63	895,23 €	2.205,00 €
Operaciones de DevOps	15	213,15 €	525,00 €
Documentación	50	710,50 €	1.750,00 €
Total	335	4.760,35 €	11.725,00 €

Como cierre al proyecto, analizar cuál ha sido el coste de producir este producto mínimo viable y su consecuente mantenimiento para realizar pruebas de usuario. El desarrollo de este proyecto supondría un total de 335 horas, con un coste de 14,21 € la hora, dando un total de 4.760,35 €, comparado con los 11.725 € que costaría encargarse de este producto mínimo viable a una empresa. Además del coste del desarrollo hay que añadir los costes de mantener a Murcy disponible a los usuarios, con el fin de poder si es viable en el mercado. El coste mensual de esta operación hace un total de 23 \$ que, al cambio a la moneda local, el euro, hace un total de 19,36 € mensuales.

5 Conclusiones

5.1 Resultado final

Como cierre a este proyecto, se ha conseguido realizar un producto mínimo viable preparado para realizar pruebas con usuarios reales. Este producto mínimo viable se ha desarrollado utilizando un *stack* tecnológico muy utilizado en la industria del desarrollo de aplicaciones web, al igual que técnicas que actualmente se sitúan como estándar de calidad y garantía de aquellos proyectos que tienen detrás una gran inversión económica.

Por otro lado, el proyecto no ha consistido en un nuevo desarrollo, sino que ha sido todo lo contrario, ha consistido en una evolución. Este tipo de proyectos tienen sus ventajas como que los requisitos de la aplicación están implícitos en la misma, pero también tienen sus desventajas, en este caso el mayor problema encontrado durante la migración ha sido, no haber previsto cuantos cambios iban a tener que ser realizadas en la capa de negocio.

5.2 Valoración

Gracias a este proyecto he aprendido nuevos patrones de diseño, y he conseguido realizar una aplicación web bonita, y funcional, siendo este mi punto débil. Por otro lado, he aprendido a realizar operaciones de mantenimiento e infraestructura, realización de pruebas. También este proyecto ha servido para forzarme a pensar soluciones reales para problemas reales.

Teniendo en cuenta el tiempo dedicado a aprender, desarrollar y depurar la aplicación, y por lo tanto el coste personal que ha supuesto, veo viable continuar con el desarrollo de Murcy a largo plazo.

Y finalmente este proyecto me ha servido para saber que este es en el mercado que quiero trabajar. Ya me gustaba el desarrollo de aplicaciones, pero el proceso de realizar una ingeniería de verdad me ha ayudado a comprender las posibilidades de esta.

5.3 Proyecto a futuro

Tras haber finalizado el desarrollo de un producto mínimo viable, como primer paso a seguir, va a ser lanzar la aplicación a los usuarios para ver si tiene aceptación. Y en caso de tener aceptación por parte de los usuarios se procederá a añadir funcionalidad y convertir a Murcy en una aplicación real añadiendo funcionalidad poco a poco, como:

- **Completar lista de requisitos funcionales:** todas aquellas funcionalidades que no han tenido cabida en el producto mínimo viable(ver Anexo A.1).
- **Mejorar la seguridad de los usuarios:** implementar medidas de seguridad para validar que una cuenta no ha sido robada.

- **Más variedad de interacciones:** implementar nuevos tipos de preguntas, *quizzes* en grupo en tiempo real, rankings.
- **Profundizar en la administración:** crear herramientas de administración para poder gestionar los flujos de las distintas peticiones, como por ejemplo auto aprobar las solicitudes de editor.
- **Mejora de la infraestructura:** conseguir que el entorno de producción se resiliente, modificando el despliegue para utilizar tecnologías que lo permitan.
- **Negocio:** implementar un modelo mediante el cual poder obtener beneficio económico, y poder auto sostener los costes de esta.
- **Usabilidad:** permitir la agrupación de preguntas en carpetas, para que los editores puedan organizar sus repositorios de preguntas.

Esta continuación del proyecto se realizaría como hobby en los momentos libres y en aquellos periodos vacacionales que permitan mantener un periodo de desarrollo considerable (4 o 5 días).

6 Bibliografía

- [1] «Crunchbase,» [En línea]. Available: <https://www.crunchbase.com/organization/kahoot->. [Último acceso: 19 septiembre 2020].
- [2] Deloitte, «Deloitte - Estudio de Consumo Móvil en España,» [En línea]. Available: <https://www2.deloitte.com/es/es/pages/technology-media-and-telecommunications/articles/consumo-movil-espana.html>. [Último acceso: 14 agosto 2020].
- [3] «SurveyMonkey,» [En línea]. Available: <https://es.surveymonkey.com/>. [Último acceso: 13 septiembre 2020].
- [4] «socrative,» [En línea]. Available: <https://www.socrative.com/>. [Último acceso: 13 septiembre 2020].
- [5] «Kahoot,» [En línea]. Available: <https://kahoot.com/>. [Último acceso: 13 septiembre 2020].
- [6] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/MoSCoW_method. [Último acceso: 18 septiembre 2020].
- [7] «Heroku status,» 13 8 2020. [En línea]. Available: <https://status.heroku.com/incidents/2097>. [Último acceso: 21 septiembre 21].
- [8] Heroku, «Heroku pricing,» [En línea]. Available: <https://www.heroku.com/pricing>. [Último acceso: 10 septiembre 2020].

A Análisis

A.1 Historias de usuario priorizadas

Las historias de usuario han sido priorizadas según el método MoSCoW [6], que consiste en priorizar los requisitos para ofrecer los mayores y más inmediatos beneficios comerciales

Las siguientes historias de usuario se han añadido al proyecto:

- **Como** usuario registrado y potencial usuario **quiero** tener una página de inicio donde poder elegir si registrarme o hacer login en la aplicación **para** usar la aplicación.
- **Como** potencial usuario **quiero** que se envíe un correo de confirmación tras registrarme en la aplicación **para** que nadie use mi correo sin autorización.
- **Como** usuario registrado **quiero** poder hacer login **para** usar la aplicación.
- **Como** usuario **quiero** poder cerrar sesión **para** dejar de usar la aplicación.
- **Como** usuario **quiero** solicitar ser editor **para** poder acceder a las herramientas de creación de contenido.
- **Como** usuario **quiero** poder buscar quizzes **para** elegir a cuál jugar.
- **Como** usuario **quiero** poder responder un quiz previamente seleccionado **para** pasar el rato.
- **Como** editor **quiero** poder crear preguntas **para** crear quizzes.
- **Como** editor **quiero** poder consultar mi lista de preguntas **para** poder crear quizzes.
- **Como** editor **quiero** poder modificar y eliminar mis preguntas **para** crear quizzes.
- **Como** editor **quiero** poder marcar una pregunta que aún no ha sido terminada como borrador **para** poder terminarla en un futuro.
- **Como** editor **quiero** poder crear quizzes **para** que otros usuarios jueguen.
- **Como** editor **quiero** poder marcar un quiz que aún no ha sido terminado de crear **para** poder terminarlo en un futuro.
- **Como** editor **quiero** poder consultar, modificar y eliminar un quiz **para** poder modificar la información de mis quizzes.
- **Como** editor **quiero** poder publicar un quiz que previamente he creado **para** que los usuarios puedan jugarlo.
- **Como** revisor **quiero** poder consultar las solicitudes de nuevos editores **para** poder mantener la calidad de la aplicación.

Las siguientes historias de usuario se ha soportan parcialmente en el proyecto:

- **Como** usuario **quiero** que se me notifique por correo cuando se resuelva una petición **para** mantenerme informado.
- **Como** usuario **quiero** saber a qué quizzes he jugado **para** no repetir.
- **Como** usuario, editor, revisor y administrador **quiero** poder consultar y modificar mi perfil **para** actualizar mi información.
- **Como** editor **quiero** poder retirar la publicación de un quiz que previamente he creado **para** que los usuarios dejen de poder jugarlo.
- **Como** revisor **quiero** poder consultar, aprobar y rechazar peticiones de publicación de quizzes **para** mantener la calidad de la aplicación.
- **Como** revisor **quiero** poder consultar la lista de todos los editores **para** poder eliminar editores que no cumplan las normas.
- **Como** revisor **quiero** poder modificar las plantillas de notificación por correo **para** poder modificar los datos que se envían por correo.
- **Como** administrador **quiero** poder eliminar revisores **para** mantener la calidad de la aplicación.
- **Como** administrador **quiero** poder crear usuarios **para** añadirlos en el sistema sin que tengan que registrarse ni confirmar su cuenta.

Las siguientes historias de usuario estarán en el proyecto:

- **Como** usuario **quiero** poder marcar quizzes como favoritos **para** así poder recomendarlos a mis amigos.
- **Como** editor **quiero** poder categorizar mis preguntas **para** poder filtrarlas a la hora de crear quizzes.
- **Como** editor **quiero** poder categorizar mis quizzes **para** que los usuarios lo encuentren mejor.
- **Como** editor **quiero** poder obtener estadísticas de mis quizzes **para** saber cómo se comportan los usuarios en mis quizzes.
- **Como** editor **quiero** tener ventajas en la creación de preguntas respecto a editores menos activos **para** destacar entre otros editores que no aportan tanto como yo a la aplicación.
- **Como** editor **quiero** poder obtener las estadísticas de las preguntas que estén en los quizzes **para** cómo se comportan los usuarios al responder una pregunta.
- **Como** revisor **quiero** poder consultar la información pública de los editores **para** poder ofrecer una mejor experiencia de usuario.
- **Como** administrador **quiero** poder cambiar los flujos de los quizzes **para** aprobarlos automáticamente si se cumplen algunas condiciones.

A.2 Requisitos no funcionales

A continuación, se listan los requisitos no funcionales de Murcy agrupados por alcance dentro del proyecto:

Accesibilidad:

- La aplicación debe ofrecer una experiencia sencilla a los usuarios.
- Los colores de la aplicación deben tener una ratio de contraste de 12 puntos mínimo.

Auditabilidad:

- Las entidades del sistema deben ser auditables. Saber quién ha creado cada entidad y que día y hora han sido modificadas.

Disponibilidad:

- La aplicación debe estar siempre en estado activo.
- La aplicación debe mostrar la información de forma inmediata.
- La aplicación debe poder realizar funciones de juego con una conexión mala o inexistente.

Costo:

- El mantenimiento de la aplicación debe suponer el menor desembolso posible.

Seguridad:

- Las comunicaciones deben realizarse mediante canales que ofrezcan garantías de seguridad.
- No se deben almacenar datos del usuario en la aplicación web.
- Los usuarios no deben perder el acceso a su cuenta por culpa de terceros.

Rendimiento:

- El tiempo de respuesta a la hora de obtener datos debe ser inferior a 750 ms.

Compatibilidad:

- La aplicación debe ser compatible con todos los dispositivos móviles inteligente con acceso a internet.
- La aplicación debe ser compatible con la mayor cantidad de dispositivos de sobremesa con acceso a internet.

A.3 Herramientas utilizadas

Diseño e implementación del servidor web: Las siguientes herramientas y tecnologías han sido utilizadas para realizar el desarrollo del BackEnd de la aplicación.

Servidor Web	
Spring Boot	Excelente framework para desarrollar aplicaciones Java. Se posee conocimientos de esta tecnología.
PostgreSQL	Base de datos relacional de código abierto muy potente y con gran versatilidad.
Gradle	Gestor de dependencias y script de compilación del proyecto.

Diseño e implementación de la aplicación web: Las siguientes tecnologías han sido utilizadas para realizar el desarrollo del FrontEnd de la aplicación.

Aplicación Web	
Angular 8	Tecnología utilizada para el desarrollo de la aplicación web. Se ha usado la versión , dado que era la versión más estable en el momento de comienzo del proyecto.
Angular material	Framework para el desarrollo de aplicaciones web en angular.

Construcción del software: Estas herramientas y plataformas han sido utilizadas para agilizar y asegurar el desarrollo y las pruebas de la aplicación.

Construcción del software	
Docker	Tecnología utilizada para acelerar y agilizar los despliegues y las pruebas en local.
Docker-Compose	Herramienta utilizada para encapsular todos los proyectos de Docker bajo un mismo sistema.
Travis-CI	Plataforma utilizada para realizar integración y despliegue continuo.
SonarCloud	Plataforma utilizada para mantener la calidad y seguridad del código.

Despliegue e infraestructura de las aplicaciones: Las siguientes tecnologías han sido utilizadas para poder desplegar y usar las aplicaciones desarrolladas en entornos reales.

Despliegue / infraestructura	
Node.js	Tecnología utilizada para hacer que el FrontEnd funcione bien desplegado.
Ngix	Tecnología utilizada para realizar el despliegue de las aplicaciones y usar protocolos HTTPS
Java 8	Utilizada como plataforma para el continuar el desarrollo de Murcy, dado que es la versión utilizada en la prueba de concepto.
Java 11	Utilizada para los nuevos desarrollos. Se debe usar en vez de la versión anterior, dado que las versiones anteriores han dejado de estar mantenidas.
Heroku	Plataforma utilizada para el despliegue de las aplicaciones.

Desarrollo: Las siguientes herramientas han sido utilizadas para poder desarrollar el proyecto.

Desarrollo	
Canvas físico	Una superficie física donde poder gestionar el progreso de los distintos <i>sprints</i> (ver Ilustración 2.9).
Git	Herramienta de control de versiones utilizada para la gestión de cambios.
JetBrains suite	Suite de programas de desarrollo muy completa, gratuita para estudiantes, pero con opciones gratuitas para todo el mundo.
Postman	Herramienta utilizada para probar peticiones HTTP fácilmente.

B Diseño

B.1 Diagrama de entidades

Dentro de Murcy, podemos diferenciar las entidades principales involucradas en el proceso de negocio, estas entidades son:

Entidad	Papel dentro del sistema
Usuario	Entidad encargada de guardar la información que permite identificar a una persona que utiliza la aplicación.
Pregunta	Entidad encargada de mantener la información sobre las preguntas creadas por los usuarios.
Opciones	Objeto valor de entidad, esta entidad representa las distintas respuestas posibles en una pregunta. Sin la pregunta, esta entidad carece de persistencia en el sistema.
Quiz	Entidad encargada de mantener la información de un quiz, al igual que todas las preguntas añadidas.
Respuesta	Entidad encargada de mantener la información de las respuestas de los quizzes.
Solicitud de editor	Entidad encargada de mantener la información de las peticiones realizadas por los usuarios para ser editores.
Workflow	Entidad encargada de mantener la trazabilidad y la evolución de una entidad
TokenConfirmation	Entidad encargada de gestionar las confirmaciones de los usuarios. Utilizada para confirmar la cuenta en el registro.
Rol	Entidad encargada de gestionar los permisos de los usuarios
Respuesta Individual	Esta Entidad encargada de mantener la información de una respuesta sobre una pregunta.

Y a continuación, se presenta un diagrama simplificado representado estas entidades.

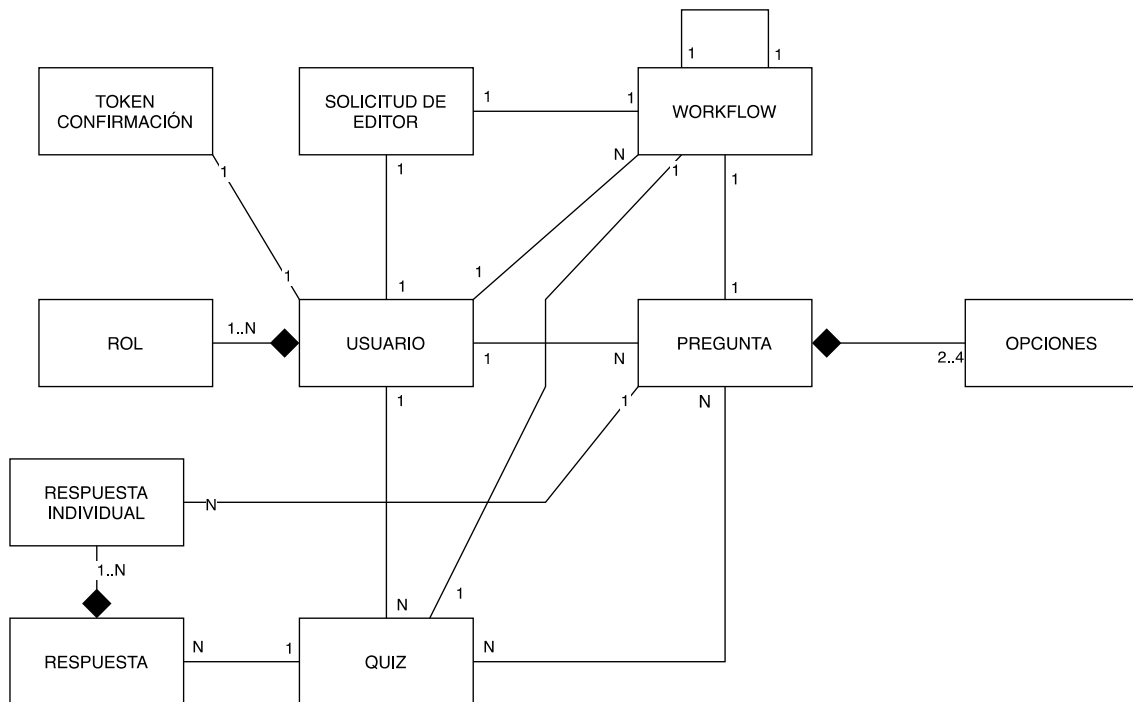


Ilustración 6.1 Diagrama de entidades completo

B.2 Diagrama de clases

El siguiente diagrama de clases permite identificar a todas las entidades de Murcy, su conexión entre ellas y los atributos que permiten a estas entidades dar forma y sentido a Murcy como aplicación de Quizzes.

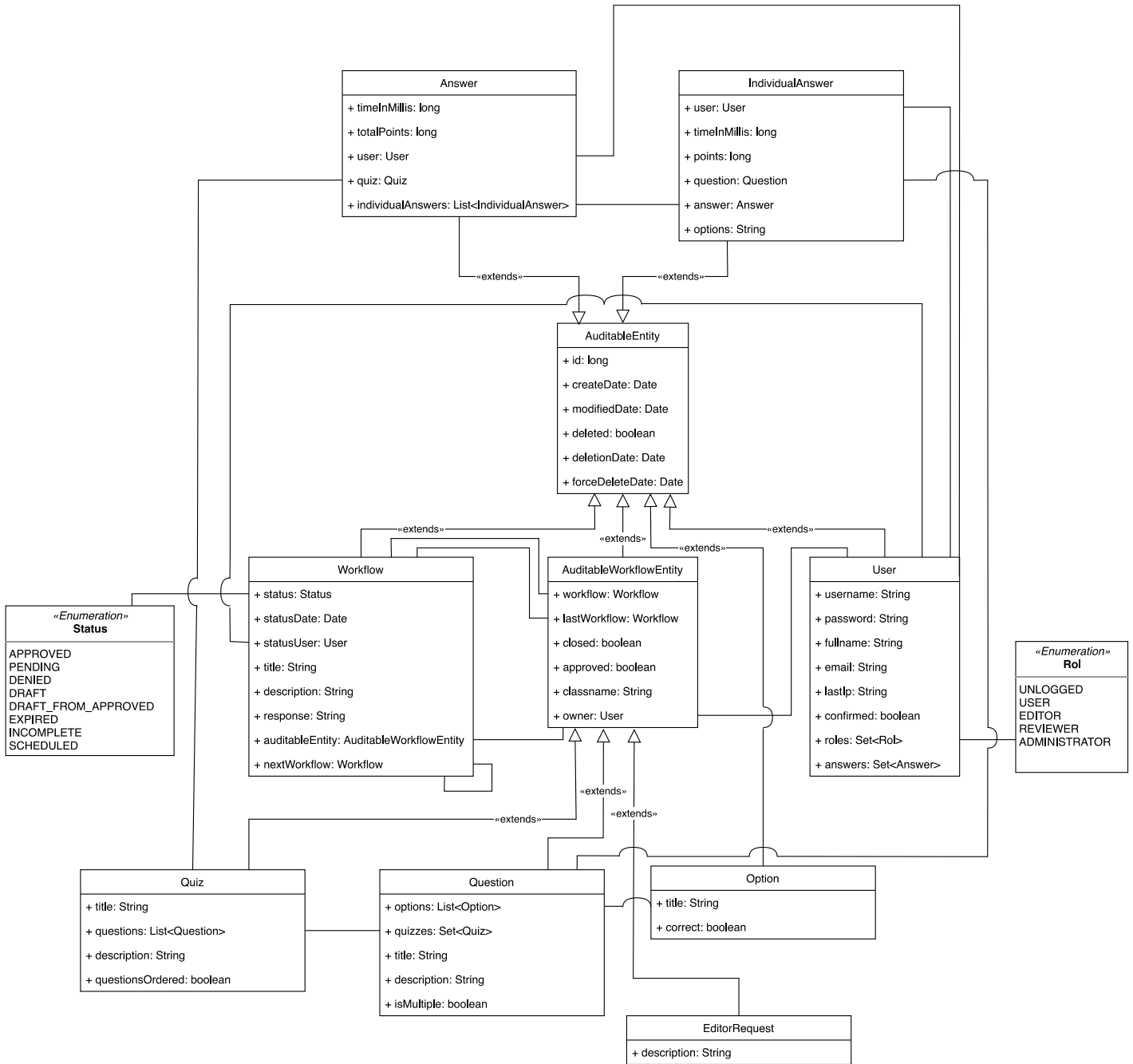


Ilustración 6.2 Diagrama de clases

B.3 Diagrama de Navegación

La navegación por la aplicación viene dada por el rol del usuario, de tal manera que un editor no puede acceder a las vistas de un revisor, pero los revisores sí que pueden acceder a las vistas de un editor. Con el fin de poder comprender la navegabilidad de Murcy se muestra el siguiente diagrama de navegación de la aplicación web:

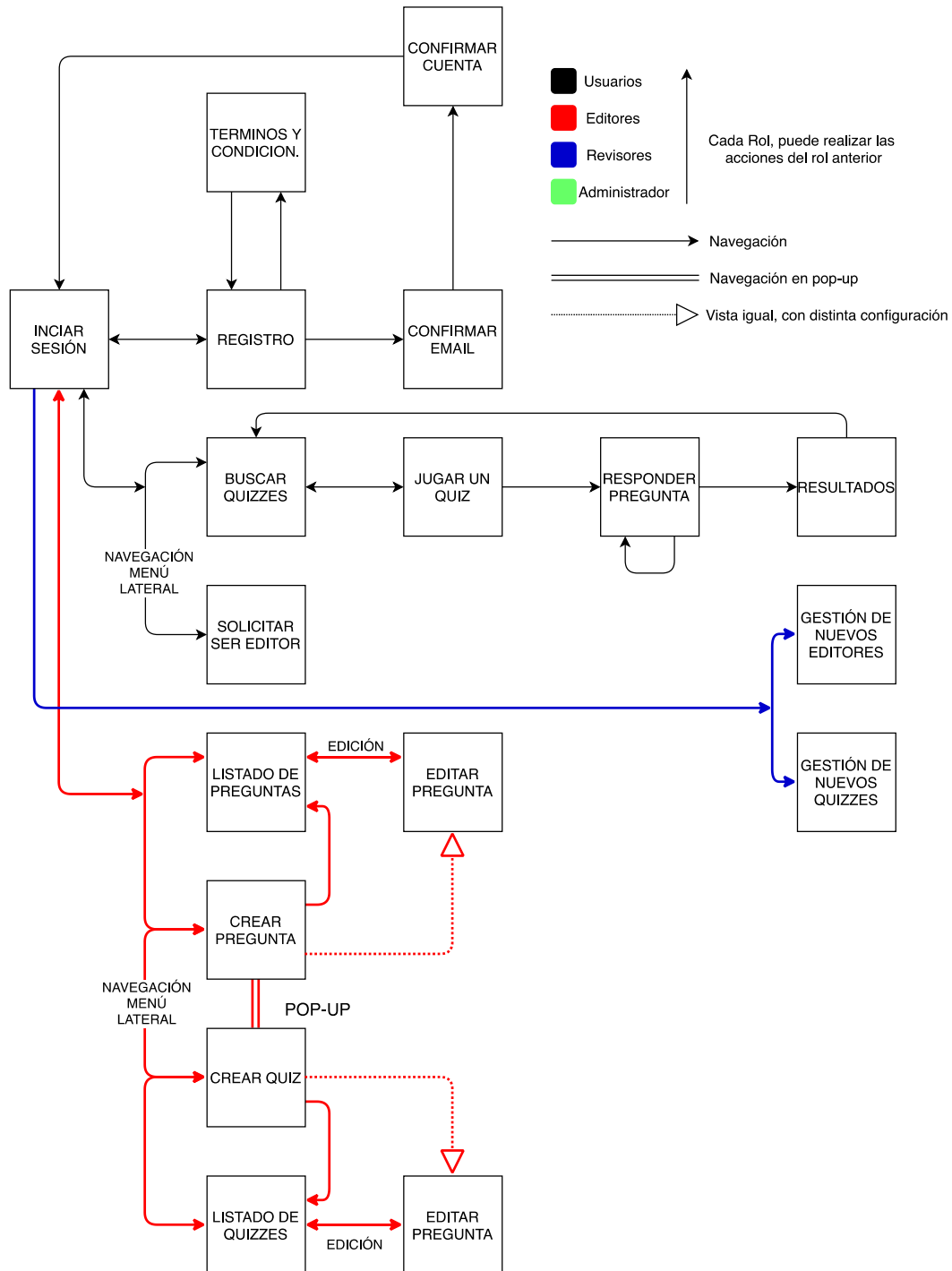


Ilustración 6.3 Diagrama de navegación completo

B.4 Diagramas de secuencia relevantes

A continuación, se presentan distintas partes de la lógica de negocio que se comprenden mejor con un pequeño diagrama de secuencia.

En primer lugar, entender cómo funcionan los filtros de autenticación. Al entrar cualquier petición en el sistema, esta pasa por unos filtros de *whitelisting*, en los que se comprueba si se trata de una petición que no necesita autenticación. En caso afirmativo la libera hacia el controlador correspondiente. En caso contrario, se verifica que exista y que sea válida, en caso afirmativo la manda al controlador correspondiente, en caso contrario devuelve un HTTP error de no autorizado (403)

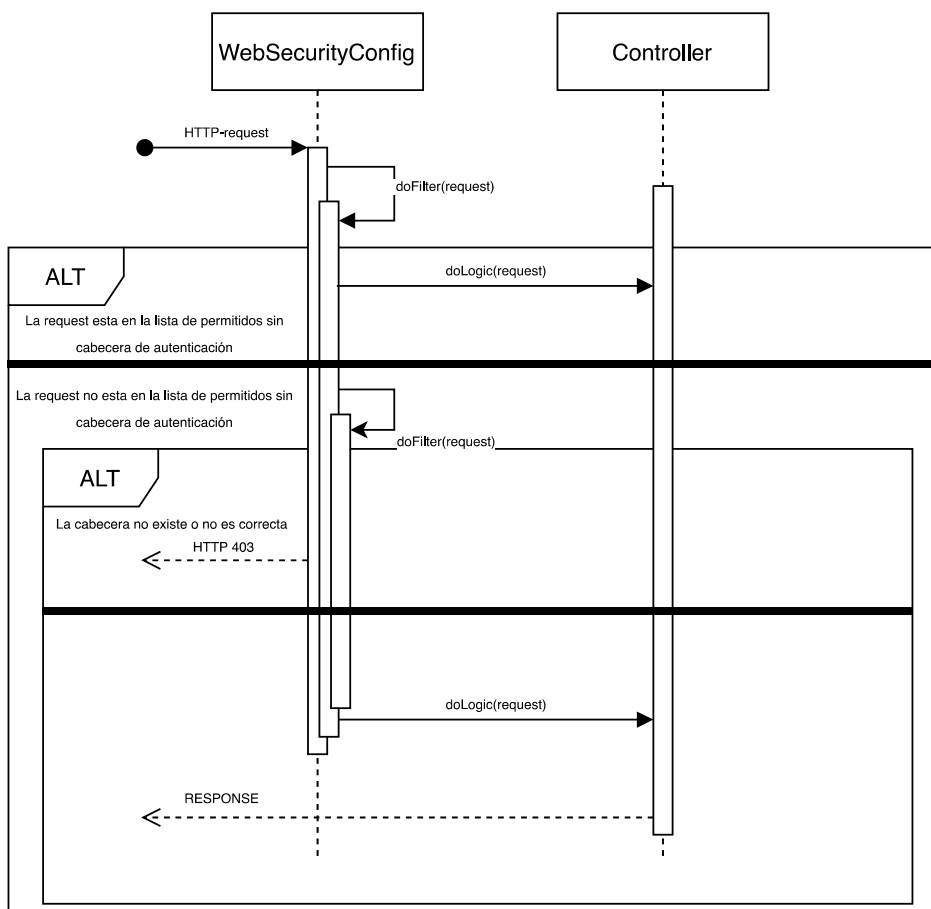


Ilustración 6.4 Diagrama de secuencia de los filterChain

El segundo diagrama propuesto hace referencia a la función de middleware que permite filtrar los endpoints en base a una política de mínimo rol. En caso de que el usuario no posea el mínimo rol necesario se le devolverá un HTTP error de 403, no autorizado. Para ello cada endpoint de los controladores que requieran de esta funcionalidad, llaman a esta función, con la petición entrante y el rol mínimo. A continuación, se comprueba que el usuario que ha realizado esta operación exista, en caso afirmativo se comprueba si puede realizar las acciones del rol, en caso afirmativo se devuelve el usuario como solicitante, y en caso contrario se devuelve un HTTP error de 403, no autorizado.

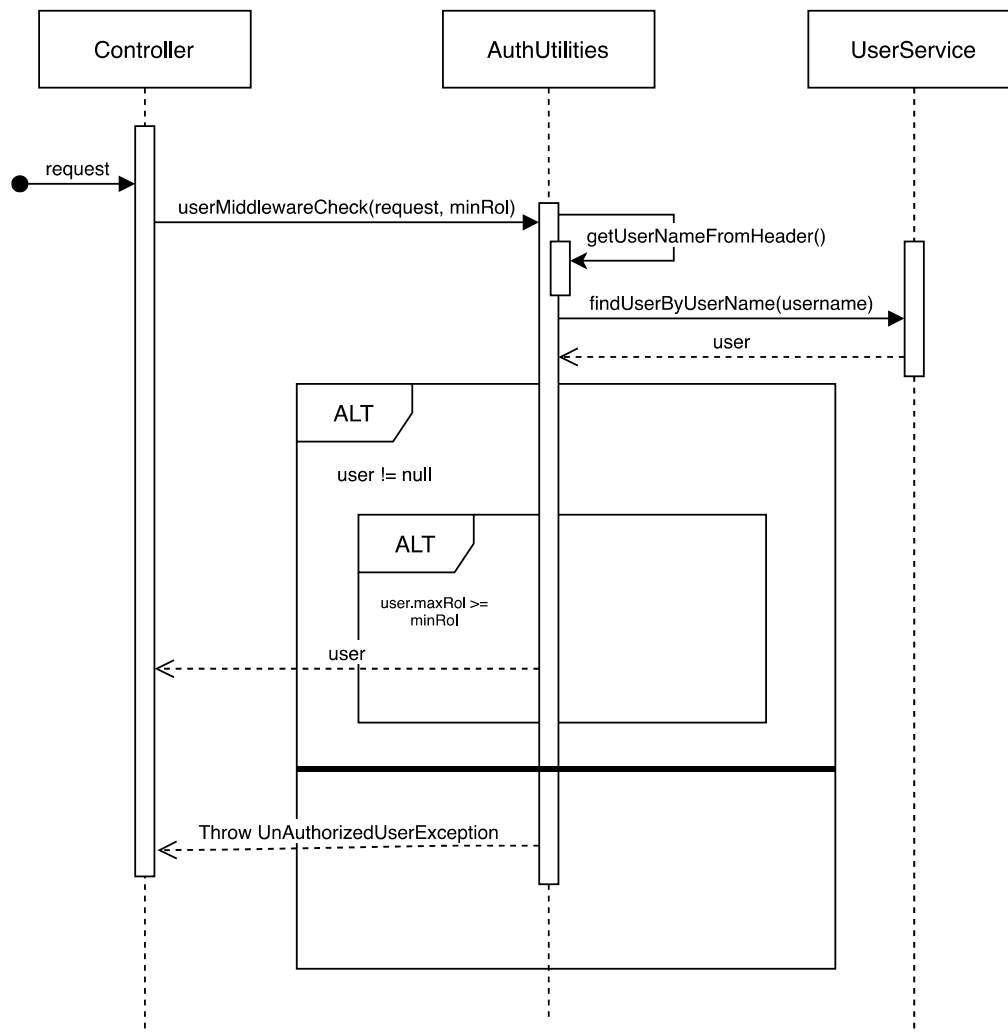


Ilustración 6.5 Diagrama de secuencia del middleWare de autenticación

Y finalmente como tercer y último diagrama de secuencia relevante para entender se propone el proceso de crear un usuario. Tras entrar la solicitud de nuevo usuario, se comprueban que los datos introducidos sean válidos, en caso contrario se devuelve un HTTP erro 400 de Bad Request. En caso de ser válido, en primer lugar, se encripta la contraseña del usuario y se almacena en la persistencia de datos. Tras esto se envía una solicitud para enviar un correo de confirmación a la API de correo a través de la interfaz de conexión MailClient.

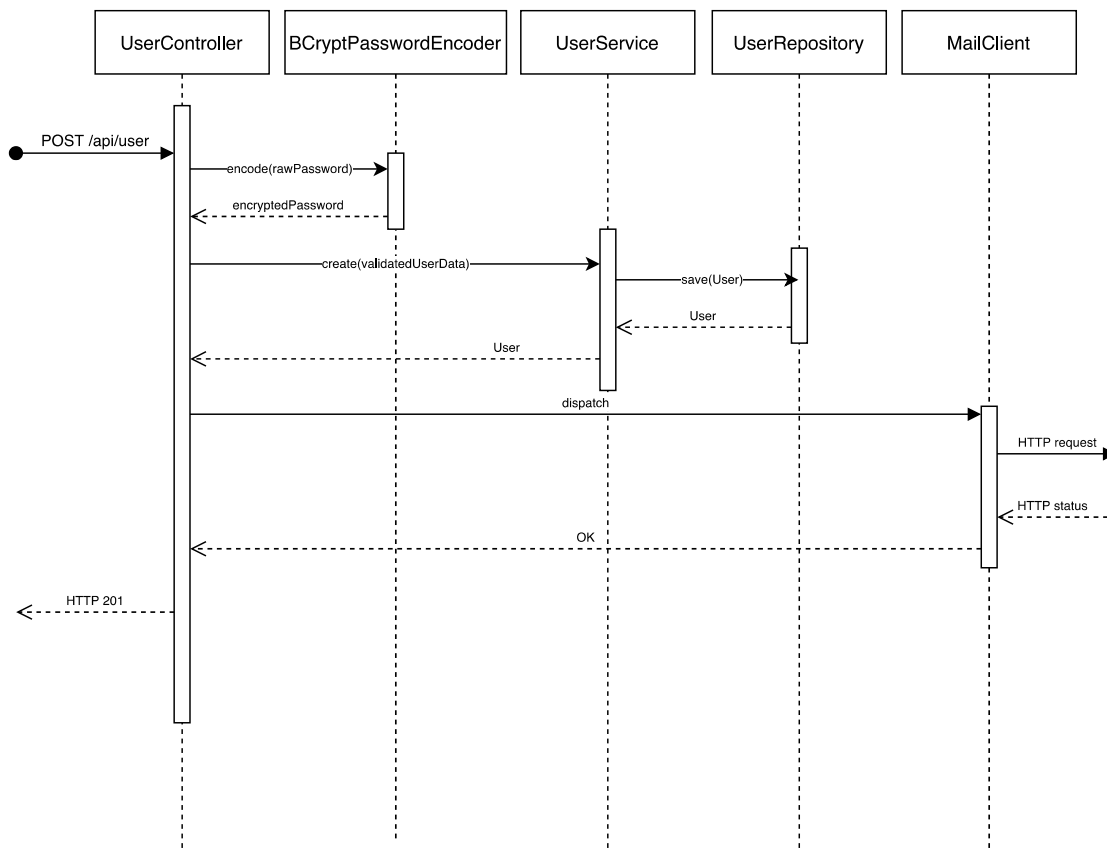


Ilustración 6.6 Diagrama de secuencia del proceso de creación de un usuario

C Evolutivo – Interfaces antes y después

Vistas antes y después del inicio de sesión.

Old login interface (Ilustración 6.7) showing a logo, input fields for Username and Password, a Log In button, and a Sign Up link.

Ilustración 6.7 Inicio de sesión antiguo

New login interface (Ilustración 6.8) showing a logo, a Log in heading, input fields for Username and password, a Log in button, and a link for new users: New at Murcy? Sign up here!

Ilustración 6.8 Inicio de sesión nuevo

Vistas antes y después de las solicitudes para ser editor.

Old request form (Ilustración 6.9) showing a Request title, a Description input field, a Send button, and a Cancel link.

Ilustración 6.9 Solicitud para ser editor antiguo

New application form (Ilustración 6.10) showing a title, a text input field with test data, an error message from admin: No puedes solicitar ser editor, al ser una cuenta de prueba, a comment field, and buttons for Reset request and Submit.

Ilustración 6.10 Solicitud para ser editor nuevo

Vistas antes y después de las pantallas de gestión de solicitudes.

Editors Requests

Request



Ilustración 6.11 Lista de solicitudes antiguo

ID	Title	Status
19	Application to be an editor	APPROVED
38	Application to be an editor	APPROVED
42	Application to be an editor	APPROVED
44	Application to be an editor	APPROVED
48	Application to be an editor	APPROVED

Items per page: 5 1 - 5 of 9

Ilustración 6.12 Lista de solicitudes nuevo

Vistas antes y después de la creación de preguntas. En la nueva vista se puede observar el uso del SDK de arrastrar.

New Question

Question and description

Title

Description

Answers
Fill in at least 2 answers

Answer 1

Answer 2

Draft Public

[Return](#)

Ilustración 6.13 Crear pregunta - antiguo

MURCY - Create new question

Create here your amazing question

Description

Write the option text

Write the option text

Ilustración 6.14 Crear pregunta - nuevo

Vistas antes y después de las listas de preguntas.



Ilustración 6.15 Listado de preguntas antiguo

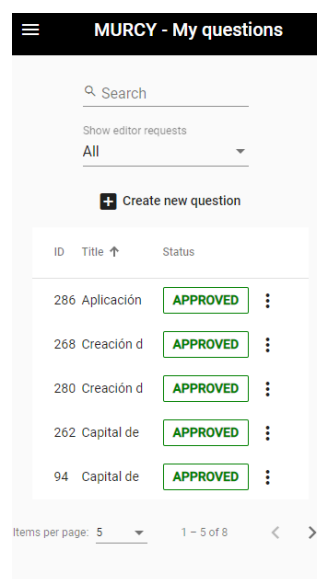


Ilustración 6.16 Listado de preguntas nuevo

Antes y después de las vistas de creación de quizzes.



Ilustración 6.17 Crear un quiz antiguo

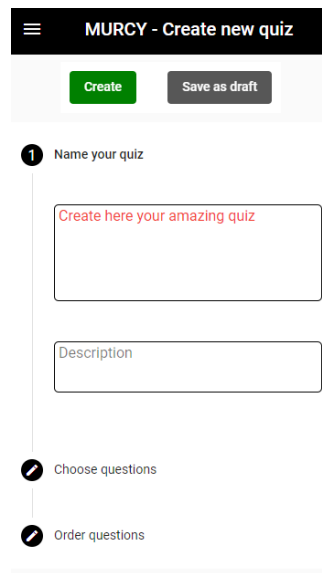


Ilustración 6.18 Crear un quiz nuevo - información

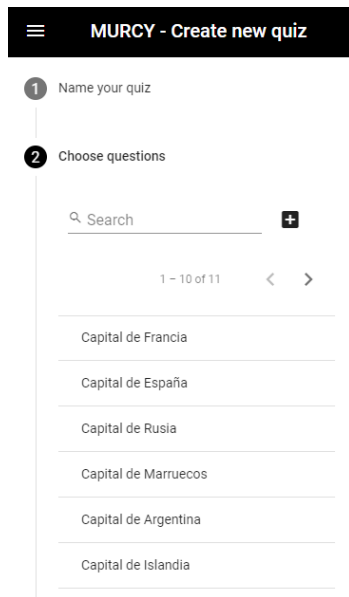


Ilustración 6.19 Crear un quiz nuevo - preguntas

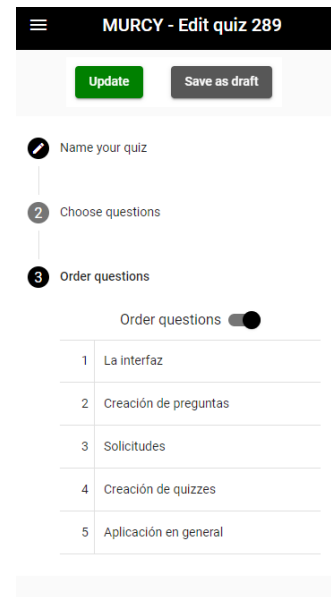


Ilustración 6.20 Crear un quiz nuevo - orden

Vistas antes y después de la gestión de quizzes.



Ilustración 6.21 Gestión de quizzes antiguo

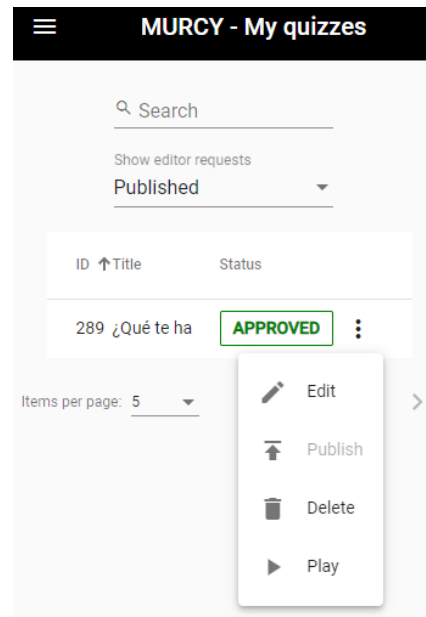


Ilustración 6.22 Gestión de quizzes nuevo y menú contextual

Vistas antes y después de la búsqueda y jugabilidad de los quizzes.



Ilustración 6.23 Lista de quizzes para jugar antiguo

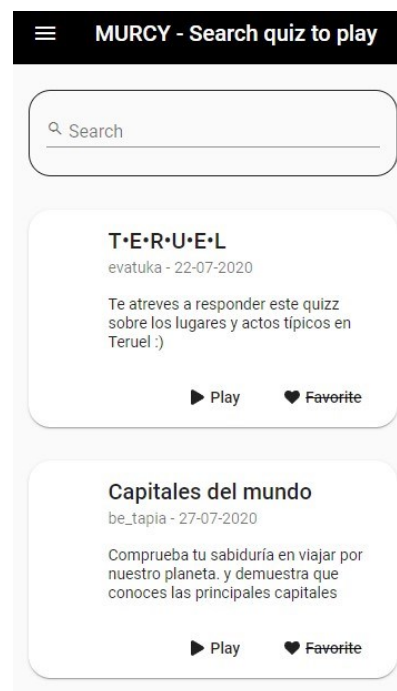


Ilustración 6.24 Lista de quizzes para jugar nuevo



Ilustración 6.25 Jugar un quiz antiguo



Ilustración 6.26 Jugar un quiz nuevo - información

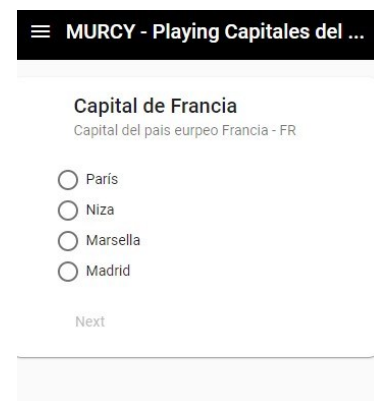


Ilustración 6.27 Jugar un quiz nuevo - jugar

D Pruebas

A continuación, se procede a analizar las pruebas diseñadas para Murcy. En primer lugar, aclarar los términos “*usuarios*”: en el siguiente contexto usuario se va a entender como un programa que lanza 5 peticiones cada segundo.

Mediante el uso de la herramienta de Postman se ha realizado todo el flujo de llamadas necesarias para hacer a Murcy funcionar, separadas según la lógica de negocio que involucran (ver 56) y se han agrupado en colecciones y carpetas.

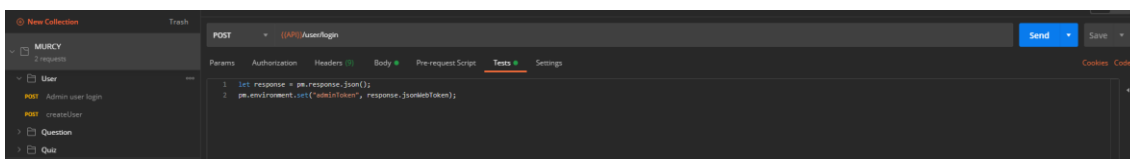
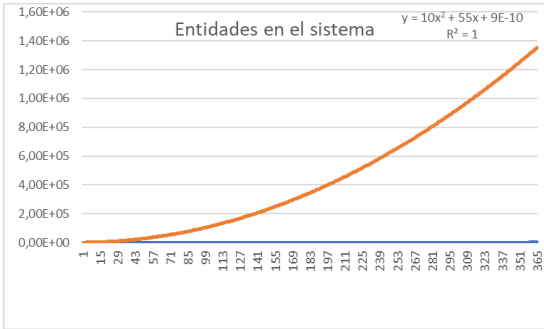
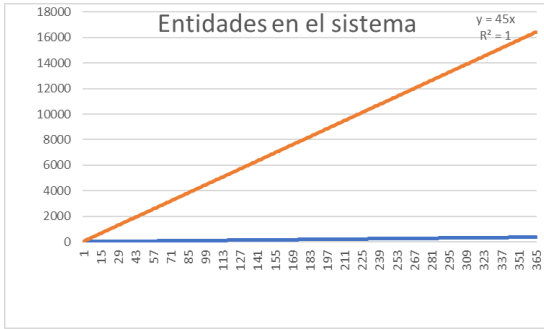


Ilustración 6.28 Postman y las colecciones

A continuación, se diferencian dos tipos de pruebas: pruebas de estrés y pruebas de carga. Gracias a este tipo de pruebas se ha conseguido identificar problemas en Murcy que afectarían en gran medida al resultado final.

Las pruebas de estrés consisten en iniciar la prueba e ir aumentando los usuarios de postman hasta un valor comprendido entre 50 y 100, de esta forma se puede ver cuando un endpoint de Murcy comienza a sufrir problemas y conseguir identificarlos gracias a los logs de la propia aplicación. Estas pruebas se han realizado con funciones de búsqueda.

Por otro lado, las pruebas de carga sirven para simular como va a evolucionar el sistema a lo largo del tiempo, y para ello es ir lanzando usuarios, pero en vez de ejecutarse en formato paralelo, se ejecutan en formato secuencial. Con este tipo de pruebas nos interesa probar las llamadas que crean, modifican y eliminan entidades. Gracias a este problema se ha descubierto un error que podría haber costado mucha cantidad de información o mucho dinero. A continuación, se muestra un ejemplo de la premisa de este tipo de pruebas y su correspondiente evaluación.

Actualización de preguntas	
<p>Un escenario donde los usuarios van creciendo a un ritmo de 5 nuevos editores cada día, cada día cada editor crea 1 pregunta y modifica otra con 4 posibles respuestas. Al modificar una pregunta, las opciones anteriores son eliminadas y las nuevas añadidas.</p>	
Funcionamiento previa corrección	Funcionamiento correcto
 <p><i>Ilustración 6.29 entidades en el sistema mal funcionando</i></p>	 <p><i>Ilustración 6.30 entidades en el sistema funcionando correctamente</i></p>
<p>Formula que describe las estimaciones de entidades en el sistema:</p> $f(x) = 10x^2 + 55x$	<p>Formula que describe las estimaciones de entidades en el sistema:</p> $f(x) = 45x$

Tal y como se puede observar en caso no haber encontrado este problema el coste de mantener información no relevante hubiera sido destructivo para este proyecto. A continuación, se muestra una tabla comparando valores normales, respecto a los valores erróneos.

Usuarios	Valor normal	Valor anómalo
1	45	65
2	90	150
3	135	255
4	180	380
5	225	525
6	270	690
7	315	875
8	360	1080
9	405	1305
10	450	1550
20	900	5100
30	1350	10650
40	1800	18200

Usuarios	Valor normal	Valor anómalo
50	2250	27750
100	5500	105500
150	7750	233250
200	11000	411000

Como se observa en la gráfica y en los datos, se espera un crecimiento lineal, pero se encuentra un crecimiento potencial, resultando que en 200 días según el problema propuesto (optimista respecto a las previsiones) si no se solucionase este error el sistema, la cantidad de entidades existentes sería un 3736% mayor.

E Documentación de API Web

E.1 Como leer esta documentación

Para poder leer esta documentación con rapidez, se ha evitado la duplicidad de información, por lo que aquellos formatos de cuerpo y respuesta que se repitan inicialmente son marcadas con un nombre y en sus próximas referencias únicamente se anota a estas entidades como: nombre «ref.». Existe navegación desde la referencia hasta la definición de la entidad.

E.2 API de Murcy

E.2.1 Usuario

POST	/api/user	
Descripción		
Petición para dar de alta a un nuevo usuario en la aplicación.		
Cabeceras		
	bearerAuth	No requerido
Cuerpo		application/json
<pre> { "username": string, "password": string, "email": string, "fullName": string, "sendMail": boolean } </pre>		
Respuestas		
	201	Usuario creado con éxito
	400	El usuario ya existe

POST	/api/user/confirm/{token}	
Descripción		
Petición para dar confirmar el registro de un usuario mediante el token enviado al correo.		
Parámetros		
	PATH	token Token por confirmar
Respuestas		
	201	Usuario confirmado
	404	El token no existe

POST	/api/login
Descripción	
Petición para iniciar sesión en la aplicación	
Cuerpo	application/json
<pre>{ "username": string, "password": string }</pre>	
Respuestas	
201	Usuario creado con éxito
Respuesta	application/json
<pre>{ "jsonWebToken": string }</pre>	
401	El login es incorrecto

GET	/api/user/info/{id}
Descripción	
Obtiene la información del usuario identificado por el identificador de la ruta, y en caso de ser omitido, devuelve la información del usuario identificado por la cabecera de autenticación.	
Parámetros	
PATH	id OPCIONAL – usuario del que se desea obtener la información
Cabeceras	
bearerAuth	Requerido
Respuestas	
200	Devuelve la información del usuario
Respuesta	application/json
<pre><u>usuario</u> = { "id": number, "username": string, "email": string, "fullName": string, "role": [string] }</pre>	
403	Permisos insuficientes para acceder a esta información.
404	El identificador solicitado no existe.

PUT	/api/user/info/{id}		
Descripción			
Actualiza la información del usuario identificado por el identificador de la ruta, y en caso de ser omitido, actualiza la información del usuario identificado por la cabecera de autenticación.			
Parámetros			
	PATH	id	OPCIONAL – usuario del que se desea cambiar la información
Cabeceras			
	bearerAuth		Requerido
Cuerpo			application/json
<u>usuario</u> «ref.»			
Respuestas			
201	Devuelve la información del usuario actualizada		
	Respuesta		application/json
<u>usuario</u> «ref.»			
403	Permisos insuficientes para acceder a esta información.		
404	El identificador solicitado no existe.		

E.2.2 Solicitud de editor

GET	/api/request/editor		
Descripción			
Obtiene las solicitudes de editor del usuario identificado por la cabecera			
Cabeceras			
	bearerAuth		Requerido
Respuestas			
200	Devuelve la solicitud		
	Respuesta		application/json
<pre> <u>solicitud_editor</u> = { "id": number, "description": string, "closed": boolean, "approved": boolean, "workflow": <u>workflow</u> = { "id": number, "title": string, "description": string, "status": string, "statusDate": DateTime, "statusBy": string, "response": string }, "lastWorkflow": <u>workflow</u> «ref.» } </pre>			
403	Permisos insuficientes para acceder a esta información		

POST	/api/request/editor	
Descripción		
Crea una solicitud de editor para el usuario identificado por la cabecera.		
Cabeceras		
	bearerAuth	Requerido
Cuerpo		application/json
<pre>{ "description": string }</pre>		
Respuestas		
	201	Solicitud creada
	403	Permisos insuficientes para realizar esta acción
	409	No se puede crear una solicitud, el usuario aún tiene una pendiente.

PUT	/api/request/editor	
Descripción		
Actualiza la solicitud de editor del usuario identificado por la cabecera.		
Cabeceras		
	bearerAuth	Requerido
Cuerpo		application/json
<pre>{ "description": string }</pre>		
Respuestas		
	202	Solicitud actualizada
	403	Permisos insuficientes para realizar esta acción

GET	/api/request/editor/list		
Descripción			
Obtiene todas las solicitudes del sistema en base a los filtros establecidos. Se requiere una autorización de revisor.			
Parámetros			
QUERY	all	Filtro booleano para ignorar el filtro "closed" y "approved"	
QUERY	closed	Filtro booleano para filtrar las solicitudes que ya han sido cerradas.	
QUERY	approved	Filtro booleano para filtrar las solicitudes que han sido aprobadas o denegadas.	
QUERY	page	Filtro numérico para indicar la página solicitada, en caso de que el valor sea -1, devolverá todos los resultados en una sola página.	
QUERY	size	Filtro numérico para indicar el tamaño de las páginas.	
QUERY	sortColumn	Filtro de texto para indicar la columna para ordenar.	
QUERY	sortType	Filtro de texto para indicar el orden del orden.	
Cabeceras			
	bearerAuth	No requerido	
Respuestas			
	200		
	Respuesta	application/json	
	[<u>solicitud_editor</u> «ref.»	
]		
	403	Permisos insuficientes para acceder a esta información.	

E.2.3 Workflow

PUT	/api/workflow/{id}/approve		
Descripción			
Aprueba un workflow y su correspondiente entidad asociada. Se requieren permisos de revisor para poder hacer esta acción.			
Parámetros			
PATH	id	Identificador del workflow que se desea aprobar	
Cabeceras			
	bearerAuth	Requerido	
Cuerpo			
		application/json	
	{	"response": string	
	}		
Respuestas			
	201	Entidad aprobada con éxito	
	403	Permisos insuficientes para realizar esta operación	
	404	Workflow con el id del PATH no se ha encontrado	

PUT	/api/workflow/{id}/deny		
Descripción			
Deniega un workflow y su correspondiente entidad asociada. Se requieren permisos de revisor para poder hacer esta acción.			
Parámetros			
	PATH	id	Identificador del workflow que se desea denegar
Cabeceras			
	bearerAuth	Requerido	
Cuerpo			application/json
<pre>{ "response": string }</pre>			
Respuestas			
	201	Entidad denegada con éxito	
	403	Permisos insuficientes para realizar esta operación	
	404	Workflow con el id del PATH no se ha encontrado	

E.2.4 Pregunta

POST	/api/question		
Descripción			
Crea una nueva pregunta en el sistema.			
Cabeceras			
	bearerAuth	Requerido	
Cuerpo			application/json
<pre>{ "title": string, "description": string, "publish": boolean, "options": [{ "title": string, "correct": boolean }] }</pre>			
Respuestas			
	201	Pregunta creada correctamente	
	403	Permisos insuficientes para realizar esta acción	

GET	/api/question/{id}
Descripción	
Obtiene la información de la pregunta identificada por el id de la ruta.	
Parámetros	
PATH	id Identificador de la pregunta que se quiere obtener.
Cabeceras	
bearerAuth	Requerido
Respuestas	
200	Devuelve la pregunta encontrada
Respuesta	application/json
<pre> pregunta = { "id": number, "title": string, "description": string, "isMultiple": boolean, "isPublic": boolean, "published": boolean, "workflow": workflow «ref.», "lastWorkflow": workflow «ref.», "ownerId": number, "ownerUsername": string, "options": [{ "id": number, "title": string, "correct": boolean }] } </pre>	
403	Permisos insuficientes para realizar esta acción
404	No se ha encontrado ninguna pregunta con ese identificador

GET	/api/question/list/{id}		
Descripción			
Obtiene las preguntas del usuario identificado por el identificador de la ruta, y en caso de ser omitido, devuelve las preguntas del usuario identificado por la cabecera de autenticación.			
Parámetros			
PATH	id	OPCIONAL – id del usuario del que se quiere obtener la lista de preguntas.	
QUERY	published	Filtro booleano para filtrar preguntas publicadas.	
QUERY	all	Filtro booleano para ignorar el filtro “published”	
QUERY	page	Filtro numérico para indicar la página solicitada, en caso de que el valor sea - 1, devolverá todos los resultados en una sola página.	
QUERY	size	Filtro numérico para indicar el tamaño de las páginas.	
QUERY	sortColumn	Filtro de texto para indicar la columna para ordenar.	
QUERY	sortType	Filtro de texto para indicar el orden del orden.	
QUERY	query	Filtro de texto para realizar búsquedas en base al texto introducido.	
Cabeceras			
	bearerAuth	Requerido	
Respuestas			
200	Lista de preguntas		
Respuesta	application/json		
[<u>pregunta</u> «ref.»		
]			
403	Permisos insuficientes para realizar esta operación		
404	El identificador solicitado no existe.		

PUT	/api/question/{id}		
Descripción			
Modifica la información de la pregunta identificada por el id de la ruta.			
Parámetros			
	PATH	id	Identificador de la pregunta que se quiere obtener.
Cabeceras			
	bearerAuth		Requerido
Cuerpo			application/json
<pre> { "title": string, "description": string, "publish": boolean, "options": [{ "title": string, "correct": boolean }] } </pre>			
Respuestas			
	201	Devuelve la pregunta actualizada	
	Respuesta		application/json
		<u>pregunta</u> «ref.»	
	403	Permisos insuficientes para realizar esta acción	
	404	No se ha encontrado ninguna pregunta con ese identificador	

DELETE	/api/question/{id}		
Descripción			
Elimina la información de la pregunta identificada por el id de la ruta.			
Parámetros			
	PATH	id	Identificador de la pregunta que se quiere eliminar.
Cabeceras			
	bearerAuth		Requerido
Respuestas			
	201	Pregunta eliminada	
	403	Permisos insuficientes para realizar esta acción	
	404	No se ha encontrado ninguna pregunta con ese identificador	

GET	/api/request/list		
Descripción			
Lista las preguntas con el objetivo de aprobarlas o denegarlas. Este endpoint forma parte de la aplicación original y dado que no se usa por el momento no ha sido actualizado a los nuevos requisitos.			
Parámetros			
	QUERY	closed	Filtro booleano para filtrar si las preguntas ya han sido revisadas.
	QUERY	approved	Filtro booleano para filtrar si las preguntas se han aprobado o denegado.
Cabeceras			
	bearerAuth		Requerido
Respuestas			
	200		
	Respuesta	application/json	
	[<u>pregunta</u> «ref.»]		
	403	Permisos insuficientes para realizar esta operación.	

GET	/api/question/{id}/answers		
Descripción			
Obtiene todas las respuestas a una pregunta identificada por el parámetro id del path			
Parámetros			
	PATH	id	Identificador de la pregunta de la que se quieren obtener las respuestas
Cabeceras			
	bearerAuth		Requerido
Respuestas			
	200	Lista de las respuestas	
	Respuesta	application/json	
	[<u>respuesta_individual</u> = { "id": number, "userId": number, "questionId": number, "timeInMillis": number, "submittedOptions": [number] }]		
	403	Permisos insuficientes para realizar esta operación	
	404	No se ha encontrado ninguna pregunta con ese identificador	

E.2.5 Quiz

POST	/api/quiz
Descripción	
Petición para crear un quiz en el sistema, se creará para el usuario identificado con bearerAuth	
Cabeceras	
bearerAuth	Requerido
Cuerpo	
	application/json
<pre>{ "title": string, "description": string, "questionIds": [number], "ordered": boolean, "publish": boolean }</pre>	
Respuestas	
201	Quiz creado correctamente
400	El cuerpo enviado no es correcto.
403	Permisos insuficientes para realizar esta acción

GET	/api/quiz/{id}
Descripción	
Obtiene la información del quiz identificado por el id del PATH	
Parámetros	
PATH	id Identificador del quiz a obtener
Cabeceras	
bearerAuth	Requerido
Respuestas	
200	Quiz obtenido
Respuesta	application/json
<u>quiz</u> «ref.»	
403	Insuficientes permisos para realizar esta operación
404	No se ha encontrado ningún quiz con el identificador introducido

GET	/api/quiz/list/{id}		
Descripción			
Devuelve la lista de los quizzes del usuario identificado por el id introducido en la ruta, y en caso de ser omitido, devolverá los quizzes del usuario identificado por la cabecera de autenticación.			
Parámetros			
PATH	id	OPCIONAL – id del usuario del que se quiere obtener la lista de quizzes.	
QUERY	published	Filtro booleano para filtrar quizzes publicados.	
QUERY	all	Filtro booleano para ignorar el filtro “published”	
QUERY	page	Filtro numérico para indicar la página solicitada, en caso de que el valor sea - 1, devolverá todos los resultados en una sola página.	
QUERY	size	Filtro numérico para indicar el tamaño de las páginas.	
QUERY	sortColumn	Filtro de texto para indicar la columna para ordenar.	
QUERY	sortType	Filtro de texto para indicar el orden del orden.	
QUERY	query	Filtro de texto para realizar búsquedas en base al texto introducido.	
Cabeceras			
	bearerAuth	Requerido	
Respuestas			
200	Lista de quizzes según los parámetros introducidos		
Respuesta	application/json		
<pre>[{ "quiz": { "id": number, "title": string, "description": string, "isPublic": boolean, "closed": boolean, "approved": boolean, "ownerId": number, "ownerUsername": string, "workflow": <i>workflow</i> «ref.», "lastWorkflow": <i>workflow</i> «ref.», "questions": [{ "pregunta": «ref.» }] } }]</pre>			
403	Insuficientes permisos para realizar esta operación		
404	Usuario con el identificador proporcionado no existe		

PUT	/api/quiz/{id}
Descripción	
Actualiza la información del quiz identificado por el id del PATH	
Parámetros	
PATH	id Identificador del quiz a actualizar
Cabeceras	
bearerAuth	Requerido
Cuerpo	application/json
<pre>{ "title": string, "description": string, "questionIds": [number], "ordered": boolean, "publish": boolean }</pre>	
Respuestas	
201	Quiz actualizado
Respuesta	application/json
<i>quiz</i> «ref.»	
403	Insuficientes permisos para realizar esta operación
404	No se ha encontrado ningún quiz con el identificador introducido

DELETE	/api/quiz/{id}
Descripción	
Elimina la información del quiz identificado por el id del PATH	
Parámetros	
PATH	id Identificador del quiz a eliminar
Cabeceras	
bearerAuth	Requerido
Respuestas	
201	Quiz eliminado
403	Insuficientes permisos para realizar esta operación
404	No se ha encontrado ningún quiz con el identificador introducido

GET	/api/quiz/{id}/public
Descripción	
Obtiene la información pública del quiz identificado por el id del PATH	
Parámetros	
PATH	id Identificador del quiz a obtener
Cabeceras	
bearerAuth	Requerido
Respuestas	
200	Quiz obtenido
Respuesta	application/json
<pre> quiz_publico = { "id": number, "title": string, "description": string, "questions": [{ "id": number, "title": string, "description": string, "isMultiple": boolean, "options": [{ "id": number, "title": string, "correct": boolean }] }] } </pre>	
403	Insuficientes permisos para realizar esta operación
404	No se ha encontrado ningún quiz con el identificador introducido

GET	/api/quiz/{id}/answers
Descripción	
Obtiene las respuestas de un quiz	
Parámetros	
PATH	id Identificador del quiz del que obtener las respuestas
Cabeceras	
bearerAuth	Requerido
Respuestas	
200	Respuestas
Respuesta	application/json
<pre> [respuesta «ref.»] </pre>	
403	Insuficientes permisos para realizar esta operación
404	Quiz identificado por el id introducido no encontrado

GET	/api/quiz/search		
Descripción			
Busca quizzes publicados			
Parámetros			
QUERY	page		Filtro numérico para indicar la página solicitada, en caso de que el valor sea -1, devolverá todos los resultados en una sola página.
QUERY	size		Filtro numérico para indicar el tamaño de las páginas.
QUERY	sortColumn		Filtro de texto para indicar la columna para ordenar.
QUERY	sortType		Filtro de texto para indicar el orden del orden.
QUERY	query		Filtro de texto para realizar búsquedas en base al texto introducido.
Cabeceras			
	bearerAuth		Requerido
Respuestas			
200	Quizzes de la búsqueda		
	Respuesta		application/json
	[<u>quiz_publico</u> «ref.»]
403	Insuficientes permisos para realizar esta operación		

POST	/api/quiz/{id}/answer
Descripción	
Crea una solución a un quiz	
Parámetros	
PATH	id Identificador del quiz donde se crea la respuesta
Cabeceras	
bearerAuth	Requerido
Cuerpo	
	application/json
<pre> { "individualAnswers": [{ "questionId": number, "timeInMillis": number, "optionsIds": [number] }] } </pre>	
Respuestas	
201	Respuesta creada correctamente
Respuesta	application/json
<pre> <u>respuesta</u> = { "id": number, "userId": number, "quizId": number, "timeInMillis": number, "totalPoints": number, "individualAnswers": [<u>respuesta individual</u> «ref.»] } </pre>	
403	Insuficientes permisos para realizar esta operación
404	Quiz identificado por el id introducido no encontrado

GET	/api/quiz/request/list		
Descripción			
Obtiene los quizzes para gestionar las peticiones. Accesible únicamente por revisores.			
Parámetros			
QUERY	all	Filtro booleano para ignorar el filtro "closed" y "approved"	
QUERY	closed	Filtro booleano para filtrar las solicitudes que ya han sido cerradas.	
QUERY	approved	Filtro booleano para filtrar las solicitudes que han sido aprobadas o denegadas.	
QUERY	page	Filtro numérico para indicar la página solicitada, en caso de que el valor sea -1, devolverá todos los resultados en una sola página.	
QUERY	size	Filtro numérico para indicar el tamaño de las páginas.	
QUERY	sortColumn	Filtro de texto para indicar la columna para ordenar.	
QUERY	sortType	Filtro de texto para indicar el orden del orden.	
Cabeceras			
	bearerAuth	Requerido	
Respuestas			
200	Lista de quizzes		
	Respuesta	application/json	
	[<u>quiz</u> «ref.»	
]		
403	Insuficientes permisos para realizar esta operación		

E.2.6 Respuesta

GET	/api/answer/{id}		
Descripción			
Obtiene una respuesta en base al identificador proporcionado			
Parámetros			
PATH	id	Identificador de la respuesta	
Cabeceras			
	bearerAuth	Requerido	
Respuestas			
200	Respuesta encontrada		
	Respuesta	application/json	
	<u>respuesta</u>	«ref.»	
403	Insuficientes permisos para realizar esta operación		
404	Respuesta no encontrada		

DELETE	/api/answer/{id}
Descripción	
Elimina una respuesta en base al identificador proporcionado	
Parámetros	
PATH	id Identificador de la respuesta
Cabeceras	
bearerAuth	Requerido
Respuestas	
201	Respuesta eliminada
403	Insuficientes permisos para realizar esta operación
404	Respuesta no encontrada

E.3 API de la aplicación de envío de correos electrónicos

POST	/mail/send
Descripción	
Envía la plantilla introducida, utilizando las variables al destinatario indicado.	
Cabeceras	
murcy-api-key	API-KEY requerido, única para consumidores.
Cuerpo	
application/json	
<pre>{ "toEmail": string, "templateName": string, "arguments": ["argument": string] }</pre>	
Respuestas	
200	Email enviado
400	Estos argumentos son obligatorios: [lista]
403	Falta la cabecera de murcy-api-key

GET	/template/list
Descripción	
Obtiene los identificadores de todas las plantillas del sistema	
Cabeceras	
murcy-api-key	API-KEY requerido, única para consumidores.
Respuestas	
200	Lista de plantillas simplificadas
Respuesta	
application/json	
<pre>[{ "id": number, "templateName": string }]</pre>	
403	Falta la cabecera de murcy-api-key

GET	/template/{id}
Descripción	
Obtiene los datos de la plantilla según el identificador proporcionado	
Parámetros	
PATH	id Id de la plantilla a obtener
Cabeceras	
murcy-api-key	API-KEY requerido, única para consumidores.
Respuestas	
200	Plantilla obtenida
Respuesta	application/json
<pre>{ "id": long, "templateName": string, "htmlTemplate": string, "css": string, "subject": string, "requiredArguments": [string] }</pre>	
403	Falta la cabecera de murcy-api-key
404	No existe ninguna plantilla con ese identificador

PUT	/template/{id}
Descripción	
Actualiza los datos de la plantilla según el identificador proporcionado	
Parámetros	
PATH	id Id de la plantilla a actualizar
Cabeceras	
murcy-api-key	API-KEY requerido, única para consumidores.
Cuerpo	
application/json	
<pre>{ "htmlBody": string, "css": string, "subject": string, "templateName": string }</pre>	
Respuestas	
201	Plantilla actualizada
Respuesta	application/json
<pre>{ "id": long, "templateName": string, "htmlTemplate": string, "css": string, "subject": string, "requiredArguments": [string] }</pre>	
403	Falta la cabecera de murcy-api-key
404	No existe ninguna plantilla con ese identificador

POST	/template
Descripción	
Crea una nueva plantilla en el sistema	
Cabeceras	
murcy-api-key	API-KEY requerido, única para consumidores.
Cuerpo	application/json
<pre>{ "htmlBody": string, "css": string, "subject": string, "templateName": string }</pre>	
Respuestas	
201	Plantilla actualizada
Respuesta	application/json
<pre>{ "id": long, "templateName": string, "htmlTemplate": string, "css": string, "subject": string, "requiredArguments": [string] }</pre>	
403	Falta la cabecera de murcy-api-key
404	No existe ninguna plantilla con ese identificador

F Documentación de instalación del sistema

La puesta en marcha de la aplicación se ha dispuesto un fichero Docker-file con el cual poder realizar una instalación limpia del sistema completo. Para poder realizar esta instalación es necesaria la herramienta docker y docker-Compose.

Para instalarla la herramienta se puede consultar el enlace oficial: <https://docs.docker.com/get-docker/>

AVISO: Para usuarios Windows que ejecuten la aplicación hay pasos que se deben realizar manualmente. Es muy recomendado que para seguir la instalación se haga en sistemas basados en Linux.

En un primer lugar, es obligatorio bajarse el repositorio: <https://github.com/jorgeRambla/TFG>

Una vez descargado el repositorio, hay que levantar los servicios comunes entre ambas aplicaciones: El servidor de correo electrónico y la base de datos, para ello hay que completar las siguientes configuraciones:

- shared_postgres.env.changeme
- shared_mail.env.changeme

Y finalmente eliminar la extensión .changeme.

Configuración propuesta de postgres:

```
# Postgres default user name
POSTGRES_USER=postgres
# Postgres default user password
POSTGRES_PASSWORD=admin
# Init db var, concat all postgres databases separated by:
#(xxxx:yyyy:zzzzz.....)
POSTGRES_MULTIPLE_DATABASES=murcy_old:murcy:murcy_mail
```

Configuración propuesta del servidor de correo:

```
# Set different options for mynetworks option (can be overwrite in
postfix-main.cf)
# empty => localhost only
# host => Add docker host (ipv4 only)
# network => Add all docker containers (ipv4 only)
PERMIT_DOCKER=host
# Please read [the SSL page in the
wiki](https://github.com/tomav/docker-mailserver/wiki/Configure-SSL)
for more information.
#
# empty => SSL disabled
# letsencrypt => Enables Let's Encrypt certificates
# custom => Enables custom certificates
# manual => Let's you manually specify locations of your SSL
certificates for non-standard cases
```

```

# self-signed => Enables self-signed certificates
SSL_TYPE=letsencrypt
# 0 => mail state in default directories
# 1 => consolidate all states into a single directory (`/var/mail-
state`) to allow persistence using docker volumes
ONE_DIR=1
# 0 => Debug disabled
# 1 => Enables debug on startup
DMS_DEBUG=1
# Configures the handling of creating mails with forged sender
addresses.
#
# empty => (not recommended, but default for backwards compatibility
reasons)
#     Mail address spoofing allowed. Any logged in user may
create email messages with a forged sender address.
#     See also https://en.wikipedia.org/wiki/Email\_spoofing
# 1 => (recommended) Mail spoofing denied. Each user may only send
with his own or his alias addresses.
#     Addresses with extension
delimiters(http://www.postfix.org/postconf.5.html#recipient\_delimiter)
are not able to send messages.
SPOOF_PROTECTION=0
# Enables a report being sent (created by pflogsumm) on a regular
basis.
# **0** => Report emails are disabled
# 1 => Using POSTMASTER_ADDRESS as the recipient
# => Specify the recipient address
REPORT_RECIPIENT=1
ENABLE_SPAMASSASSIN=0
ENABLE_CLAMAV=0
# If you enable Fail2Ban, don't forget to add the following lines to
your `docker-compose.yml`:
#     cap_add:
#         - NET_ADMIN
# Otherwise, `iptables` won't be able to ban IPs.
ENABLE_FAIL2BAN=1
ENABLE_POSTGREY=0

```

A continuación, vamos a levantar los servicios comunes, mediante los siguientes comandos:

- docker-compse up -d --build postgres
- docker-compose up -d --build mail

¡AVISO! Durante esta secuencia se crean los esquemas de la base de datos necesarios, en los sistemas Windows se deben crear manualmente, dado que la virtualización de Linux no es completa en los sistemas Windows. Se deberán crear los siguientes esquemas (si no se cambia la configuración del resto de aplicaciones): murcy y murcy_old.

Si deseas añadir nuevas cuentas de correo electrónico, puedes hacerlo mediante el siguiente comando:

- ./setup_mail.sh email add <tunombre>@murcy.es passwd

Una vez levantados los servicios básicos podemos levantar tanto la aplicación original, como la aplicación actual, pero antes de nada se deben rellenar los ficheros de configuración de entorno:

- enviroment_old.env.changeme
- enviroment_new.env.changeme

Y finalmente eliminar la extensión .changeme.

A continuación, se muestra la configuración por defecto para la aplicación antigua:

```
#
# DATABASE MUST BE INCLUYED IN shared_postgres.env >
POSTGRES_MULTIPLE_DATABASES env var
#
POSTGRES_DB_MURCY_OLD=murcy_old
#
# INITAL USER USERNAME
#
ADMIN_USERNAME=yorch044
#
# INITAL USER CREATED PASSWORD
#
ADMIN_PASS=yorch044
#
# BACK-END URL, MUST BE ACCESIBLE LOCALLY OR REMOTELLY
#
BACK_END_URL=http://localhost:9090
#
# FRONT-END URL, MUST BE ACCESIBLE LOCALLY OR REMOTELLY
#
FRONT_END_URL=http://localhos:9000
#
# SECRET WORD USED FOR JWT AUTH, THIS WORD MUST BE SECRET, DON'T SHARE
IT WITH ANYBODY
#
JWT_SECRET=secretToken
#
# FROM MAIL, MUST BE REAL MATCH BETWEEN MAIL ALL LISTED ENV VARIABLES
#
MAIL=admin@murcy.es
MAIL_PASSWORD=passwd123
MAIL_SERVER=mail
MAIL_SERVER_PORT=25
MAIL_PROTOCOL=sntp
```

A continuación, se muestra la configuración por defecto para la aplicación actual:

```
#
# ACTIVE PROFILES. docker PROFILE IS MANDATORY. LIST OTHER PROFILES
# - debug: show startup messages, ERROR log level and INFO level for
app code, no dependencies.
# - reduce-logging: show startup messages, ERROR log level and INFO
level for http handling in controllers.
#
ACTIVE_PROFILES=docker
```

```
#
# DATABASE MUST BE INCLUDED IN shared_postgres.env >
POSTGRES_MULTIPLE_DATABASES env var
#
POSTGRES_DB_MURCY=murcy
#
# INITIAL USER EMAIL
# DEFAULT VALUE IF NOT SET: admin@murcy.es
#
ADMIN_EMAIL=admin@murcy.es
#
# INITIAL USER USERNAME
# DEFAULT VALUE IF NOT SET: admin
#
ADMIN_USERNAME=admin
#
# INITIAL USER PASSWORD
# DEFAULT VALUE IF NOT SET: supersecretpassword
#
ADMIN_PASS=supersecretpassword
#
# REFERED TO PERSITENCE AFTER ENTITY IS DELETED, IF true ENTITIES WILL
DELETED ON DELETE OP,
# ELSE IF false ENTITIES WILL ONLY MARKED AS DELETED.
# DEFAULT VALUE IF NOT SET: FALSE
#
HARD_DELETE=true
#
# BACK-END URL, MUST BE ACCESIBLE LOCALLY OR REMOTELLY
#
BACK_END_URL=http://localhost:8090
#
# FRONT-END URL, MUST BE ACCESIBLE LOCALLY OR REMOTELLY
#
FRONT_END_URL=http://localhost:8099
#
# SECRET WORD USED FOR JWT AUTH, THIS WORD MUST BE SECRET, DON'T SHARE
IT WITH ANYBODY
#
JWT_SECRET=secretToken
#
# JWT EXPIRATION TIME
#
# DEFAULT VALUE IF NOT SET: 0
JWT_DAYS=1
# DEFAULT VALUE IF NOT SET: 5
JWT_HOURS=0
# DEFAULT VALUE IF NOT SET: 0
JWT_MINUTES=0
#
# MAIL API SECRET API_KEY, MUST BE EQUALS TO environment_mail-api.env
# IF EMPTY secretapikey VALUE WILL BE USED
#
API_KEY=secretapikey
```

Y finalmente se requiere la configuración del servicio de correo:

```
#
# DATABASE MUST BE INCLUDED IN shared_postgres.env >
POSTGRES_MULTIPLE_DATABASES env var
#
POSTGRES_DB_MURCY_MAIL=murcy_mail
#
# FROM MAIL, MUST BE REAL MATCH BETWEEN MAIL ALL LISTED ENV VARIABLES
#
MAIL=admin@murcy.es
MAIL_PASSWORD=passwd123
MAIL_SERVER=mail
MAIL_SERVER_PORT=25
MAIL_PROTOCOL=smtp
MAIL_AUTH=true
MAIL_TTLS=true
#
# MAIL API SECRET API_KEY, MUST BE EQUALS TO environment_new.env
# IF EMPTY secretapikey VALUE WILL BE USED
#
API_KEY=secretapikey
```

Una vez definidas las variables necesarias se pueden arrancar simultáneamente ambas aplicaciones al mismo tiempo, con los siguientes comandos:

- docker-compose up -d --build murcy_app_old
- docker-compose up -d --build murcy_app