UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

TeacherQuiz: una herramienta de apoyo para dinamizar la enseñanza. Evaluación y desarrollo de la idea de negocio

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Sergio Vacas Bodoque

Tutor: Patricio Letelier Torres

Curso 2019 - 2020

Para mis padres por darme el apoyo y las alas que me han permitido volar

Agradecimientos

A los profesores que a lo largo de mi vida me han ayudado a adquirir los conocimientos que culminan este trabajo.

A mis amigos, por su apoyo incondicional durante todos estos años de carrera.

A todos los participantes de los experimentos y los usuarios que han ayudado a probar la aplicación, por su paciencia y dedicación.

Resum

L'ensenyament està enfrontada a grans desafiaments i transformacions associades als nous mètodes d'aprenentatge. Hi ha una clara tendència cap als mètodes actius, amb classes més dinàmiques, amb alta interacció entre els estudiants i el professor. Com a suport a aquests nous escenaris hi ha una gamma d'eines que com Kahoot, Google Forms i Socrative que de forma senzilla i entretinguda permeten el professor interactuar amb els alumnes dins i fora de l'aula. En aquest tipus d'eines el professor pot gestionar preguntes i qüestionaris, llançant la seva execució perquè els estudiants responguin i es puguin observar els resultats en línia a temps real. No obstant això, cadascuna d'aquestes eines tenen una orientació i funcionalitats específiques, sent interessant poder tenir-les en una sola eina. A més, en aquest context d'eines seria molt útil per al professor poder fer més gran explotació de les dades dels test, poder transformar-los fàcilment en avaluacions de l'estudiant, crear comunitat per compartir test entre professors, versionat de preguntes, etc.

Per l'anterior, en aquest TFG s'estableix com a objectiu el desenvolupament d'una idea de negoci associada a una eina software per a la gestió de qüestionaris, prenent com a referència les aplicacions esmentades i estenent les funcionalitats per aconseguir un producte atractiu. Aquest TFG s'emmarca en Start.inf, l'espai d'emprenedoria de l'ETSINF i es desenvoluparà seguint el mètode *Lean Startup*, la qual cosa inclou l'avaluació de la idea de negoci, la posada en marxa de el projecte i el seu desenvolupament basant-se a l'almenys dos MVP amb els seus corresponents experiments amb *early adopters*. El producte es desenvoluparà amb tecnologia web cuidant que la seva interfície s'adapti adequadament a diferents dispositius.

Paraules clau: qüestionaris y evaluació online, mètode d'aprenentatge, metodologies àgils, Lean Startup, Scrum, emprendiment

Resumen

La enseñanza está enfrentada a grandes desafíos y transformaciones asociadas a los nuevos métodos de aprendizaje. Hay una clara tendencia hacia los métodos activos, con clases más dinámicas, con alta interacción entre los estudiantes y con el profesor. Como apoyo a estos nuevos escenarios hay una gama de herramientas que como Kahoot, Google Forms y Socrative que de forma sencilla y entretenida permiten al profesor interactuar con los alumnos dentro y fuera del aula. En este tipo de herramientas el profesor puede gestionar preguntas y cuestionarios, lanzando su ejecución para que los estudiantes respondan y se puedan observar los resultados en línea a tiempo real. Sin embargo, cada una de estas herramientas tienen una orientación y funcionalidades específicas, siendo interesante poder tenerlas en una sola herramienta. Además, en este contexto de herramientas sería muy útil para el profesor poder hacer mayor explotación de los datos de los test, poder transformarlos fácilmente en evaluaciones del estudiante, crear comunidad para compartir test entre profesores, versionado de preguntas, etc.

Por lo anterior, en este TFG se establece como objetivo el desarrollo de una idea de negocio asociada a una herramienta software para la gestión de cuestionarios, tomando como referencia las aplicaciones mencionadas y extendiendo las funcionalidades para conseguir un producto atractivo. Este TFG se enmarca en Start.inf, el espacio de emprendimiento de la ETSInf y se desarrollará siguiendo el método *Lean Startup*, lo cual incluye la evaluación de la idea de negocio, la puesta en marcha del proyecto y su desarrollo en base a al menos dos MVP con sus correspondientes experimentos con *early adopters*. El producto se desarrollará con tecnología web cuidando que su interfaz se adapte adecuadamente a diferentes dispositivos.

Palabras clave: cuestionarios y evaluación online, método de aprendizaje, metodologías ágiles, Lean Startup, Scrum, emprendimiento

Abstract

Teaching is facing great challenges and transformations associated with new learning methods. There is a clear trend towards active methods that includes more dynamic classes with high interaction between students and the teacher. To support these new scenarios, there is a range of tools such as Kahoot, Google Forms and Socrative that allows the teacher to interact with students inside and outside the classroom in a simple and entertaining way. In this type of tool, the teacher can manage questions and quizzes, launching them so the students respond and its answers can be observed live online. However, each of these tools has a specific orientation and functionality, being interesting to have them in one tool. In addition, in this context of tools it would be very useful for the teacher to be able to make greater use of the test data, to be able to easily transform it into student evaluations, to create a community to share *quizzes* between teachers, versioning questions, etc.

Therefore, this thesis establishes the objective of developing a business idea associated with a software tool for managing questionnaires, taking the aforementioned applications as a reference and extending the functionalities to achieve an attractive product. This thesis is part of Start.inf, the entrepreneurship space of the ETSInf and will be developed following the *Lean Startup* method, which includes the evaluation of the business idea, the implementation of the project and its development based on at least two MVPs and its corresponding experiments with early adopters. The product will be developed with web technology taking care that its interface adapts appropriately to different devices.

Key words: quizzes and online evaluation, learning method, agile methodologies, Lean Startup, Scrum, entrepreneurship

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

1.1 Introducción

Hoy en día la enseñanza está evolucionando a medida que lo hacen las tecnologías de comunicación, además se puede observar una clara inclinación de este cambio hacia dinamizar las clases fomentando la participación de los alumnos, puesto que cada vez se integran más técnicas como el aprendizaje orientado a proyectos, la docencia inversa o el aprendizaje cooperativo. Junto con estas técnicas también se incorporan tecnologías en las que los profesores se pueden apoyar para facilitar esa interacción, preparar material o automatizar tareas.

En mi propia experiencia como alumno he observado como distintos profesores realizan pruebas durante las clases con el objetivo de obtener *feedback* sobre los conocimientos que estaban consiguiendo los alumnos y cómo de efectivas resultaban las clases. Estas pruebas, que en su mayoría se basan en algunas preguntas de respuesta múltiple, se realizaban en papel hasta que comenzaron a surgir aplicaciones para cumplir esta necesidad como Kahoot, Google Forms o Socrative. Estas herramientas no solo facilitan al profesor el proceso de crear, realizar y corregir estas pruebas, sino que también aportan otras funcionalidades como la visualización de resultados a tiempo real y además tienen sus propias funcionalidades específicas dependiendo del público al que estén orientadas.

Estas herramientas son de gran utilidad para el profesor, pero siguen haciendo un uso pobre de los datos obtenidos al realizar estas pruebas, esto motivó la idea de una aplicación que no solo mejorará los sistemas de gestión del material creado por el profesor, sino que a su vez usará las respuestas de los alumnos para realizar estadísticas de las preguntas realizadas aportando todo tipo de información que ayude a mejorarlas, seguimientos y evaluaciones personales de forma que se pueda convertir en una herramienta de evaluación completa, creando una comunidad docente que pueda compartir material, etc.

De esta forma nace la idea *Teacher's Quiz*, una aplicación que aporta esas funcionalidades manteniendo una interfaz simple y que pueda ser usada desde cualquier dispositivo. En su desarrollo se empleará la metodología *Lean Startup* para asegurar que el proyecto avanza en la dirección más apropiada en cada fase, y además técnicas de otras metodologías ágiles que faciliten su gestión.

1.2 Objetivos

La finalidad de este trabajo es llevar a cabo un proyecto de emprendimiento enfocado en obtener un producto que ayude a los profesores a dinamizar la enseñanza mediante una herramienta software que será accesible desde cualquier dispositivo a través de la web. Para alcanzar esta meta se han definido objetivos más concretos que debieran alcanzarse durante su desarrollo:

- Fijar una estrategia de negocio basada en el mercado actual.
- Comprobar la viabilidad financiera del proyecto.
- Obtener un producto software con una alta escalabilidad y tolerancia a cambios.
- Validar este producto en un entorno real.

Por otra parte también existen objetivos personales ligados a este proyecto como trabajo fin de grado:

- Profundizar en el uso real de técnicas, metodologías, herramientas y procedimientos estudiados académicamente.
- Adquirir experiencia en lo que implica llevar a cabo el desarrollo completo de un producto software en todas sus especializaciones.
- Ahondar en el proceso de crear una empresa basada en tecnología.

1.3 Estructura de la memoria

Este trabajo expone la generación de una idea de negocio junto con el posterior desarrollo de una aplicación web que se materializa esta idea en un producto software. La exposición de todo este proceso se ha distribuido de la siguiente forma:

- **Capítulo 2**: Muestra la documentación que ha ayudado a definir la idea de negocio incluyendo las técnicas empleadas para desarrollarla, análisis del mercado en el que se espera implantar y estudio económico que muestra su viabilidad.
- **Capítulo 3**: Explica las metodologías y procedimientos utilizados durante el desarrollo del software que se crea con el objetivo de probar esta idea de negocio.
- **Capítulo 4**: Especifica las tecnologías utilizadas para implementar la aplicación web, arquitecturas empleadas y desafíos técnicos más relevantes.
- **Capítulo 5**: Muestra la cronología y distribución de las horas invertidas.
- **Capítulo 6**: Exposición de las conclusiones y plantea los próximos pasos.

Posteriormente se enumeran las referencias bibliográficas que han sido consultadas para llevar a cabo este trabajo y finalmente el apartado de anexos.

CAPÍTULO 2

Evaluación de la idea de negocio

2.1 Resumen de Teacher's Quiz

Teacher's Quiz es una herramienta que brinda a profesores un entorno amigable desde el cual crear, gestionar, compartir y lanzar exámenes interactivos, los cuales puedan ser respondidos desde cualquier dispositivo y corregidos automáticamente por este mismo sistema. Tras el lanzamiento de estos exámenes proporcionará una vista detallada de las respuestas a cada pregunta a tiempo real, así como estadísticas más concretas sobre el conjunto de respuestas dadas una vez finalizado este proceso. Tras esto el alumno que haya participado obtendrá su nota y el profesor tendrá a su disposición tanto las estadísticas ya nombradas, como un seguimiento de estos estudiantes a través de los test en los que hayan participado.

De esta manera se dota al profesor de los datos necesarios para adaptar sus preguntas. Aquí es donde Teacher's Quiz gana una gran fuerza competitiva con respecto a otras aplicaciones similares en la actualidad, ya que los principales competidores en este sector (Kahoot[1], Google Forms[2] y Socrative[3]) proveen una manera rápida de crear y lanzar un examen, pero con limitaciones muy claras respecto del mantenimiento de la batería de preguntas. Por ello se va a ofrecer una interfaz desde la cual se puedan modificar preguntas o crear nuevas versiones de preguntas, las cuales se actualizarán automáticamente en todos los exámenes en los que se estén utilizando.

Para ello se ha decidido crear una aplicación web con un diseño basado en lo principios del *responsive design*, lo que permitirá acceder a ella desde cualquier dispositivo y por lo tanto evitar crear varios clientes para esta misma aplicación ahorrando tiempo y dinero para el despegue de esta idea, además de esta forma no habrá que comprometer a los distintos usuarios a descargar una aplicación.

2.2 Idea de negocio

La idea de negocio es el principio de todo emprendimiento, en ella se plasma la idea principal que se desarrolla en cuatro puntos clave: clientes, oferta, infraestructura y viabilidad económica. Para ello se ha empleado la técnica *Lean Canvas* que permite describir ideas de negocio en una sola página mejorando su visibilidad y por tanto facilitando su modificación para poder adaptar la idea de una forma ágil a medida que avanza.

[1]Kahoot: https://kahoot.com/
[2]Google Forms: https://www.google.es/intl/es/forms/about/
[3]Socrative: https://socrative.com/

Lean Canvas surge de la especialización del *Business Model Canvas* propuesto por *Alexander Osterwalder* [1] y representa los puntos clave de una idea de negocio en una tabla con 9 secciones: clientes, problema, proposición de valor, solución, canales, ingresos, costos, métricas y ventaja competitiva. Además, se ha seguido ese orden preciso para la redacción de estos puntos, ya que, es el más óptimo para conceptualizar la idea y se han emplazado de forma que se cree una estructura que deja a un lado el mercado y al otro la empresa, entorno, procesos y sus activos para ayudar en la visualización de los conceptos.

El resultado de este proceso se muestra en la figura 2.1 siendo esta la versión actual de este documento, aunque podrá sufrir más cambios en el futuro a medida que se desarrolle el proyecto. Entre sus apartados hay que hacer especial mención al número 9, pues la sección «Ventaja competitiva» hace referencia a las características propias de la idea de negocio que no posee ningún otro competidor o no lo puede copiar la competencia, lo cual es complicado en la industria del software, pues cualquier nueva funcionalidad puede ser implementada también por la competencia, por tanto lo que se ha añadido en este apartado se debe a que ningún otro competidor posee esta ventaja por ahora, cuando ya no sea así esto dejará de ser una ventaja.

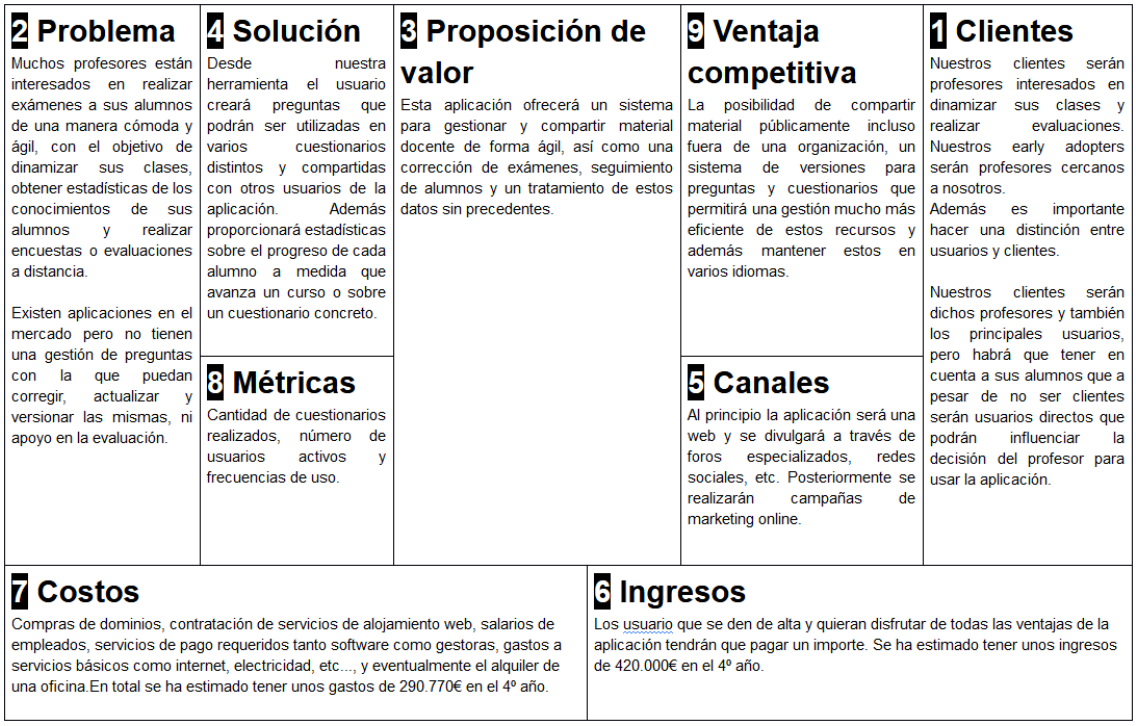

Figura 2.1: *Lean Canvas*

2.3 Estudio de mercado

Una parte esencial en la preparación de una idea de negocio es conocer otros productos similares que se encuentren en el mercado actualmente para detectar que ofrecen y que no, de esta manera se pueden definir los servicios básicos que se deberán implementar en nuestro producto, de forma que no se entre en el mercado con alguna carencia básica, pero también tener en cuenta las características de las que prescinde el resto y por tanto con las que se puede innovar.

Con este objetivo se ha realizado un análisis de los principales competidores existentes en la actualidad y se ha plasmado el resultado en una tabla en la que se compara sus características con las propuesta por Teacher's Quiz.

2.3.1. Google forms

Google forms es una opción muy genérica y pensada principalmente para encuestas, aunque su fácil accesibilidad, presentación de los resultados, integración con otros servicios de Google y simplicidad, la hacen muy atractiva para usuarios principiantes en esta clase de aplicaciones. Por tanto a pesar de no estar enfocada principalmente a realizar exámenes, si es un competidor a tener en cuenta.

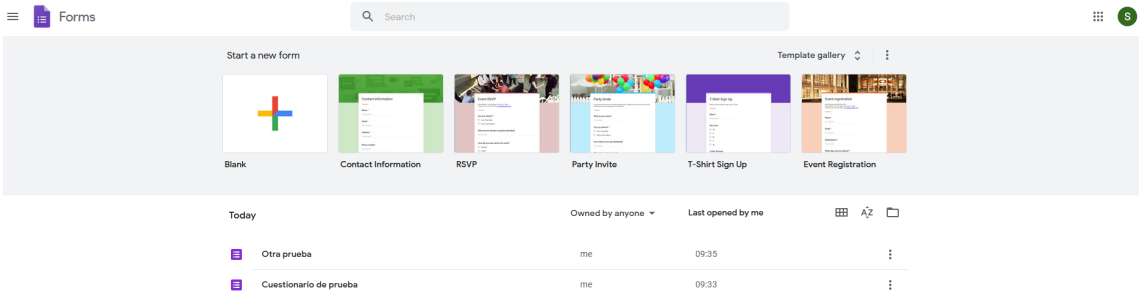

Figura 2.2: Google forms: Página principal

Como se observa en la figura 2.2, al entrar en esta aplicación se muestra directamente una lista con los formularios a los que tiene acceso el usuario, este listado tiene varias opciones de organización como ordenar alfabéticamente o crear directorios. Para crear un nuevo formulario se puede partir de una plantilla o no, en cualquier caso el usuario se traslada a una pantalla como la relejada en la figura 2.3.

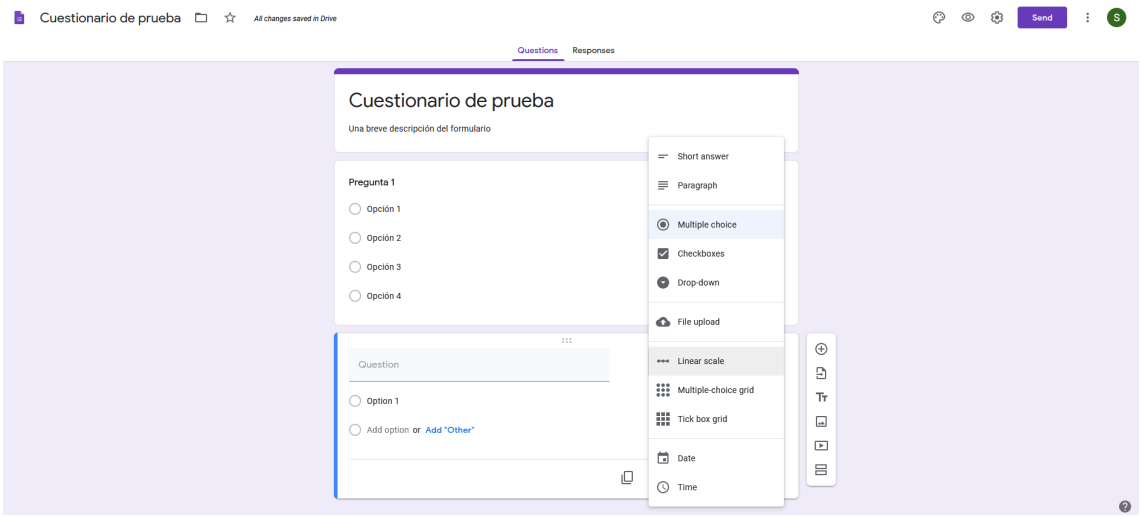

Figura 2.3: Google forms: Página de creación de formulario

En esta misma pantalla de creación o edición de un nuevo formulario se produce la gestión de las distintas preguntas que se vayan a utilizar en el formulario. Asimismo se observa una amplia variedad en los tipos de preguntas disponibles. Tras crear las preguntas a utilizar, para permitir que se pueda responder a este formulario hay que pulsar el botón *Send*, este muestra un modal como el que se observa en la figura 2.4.

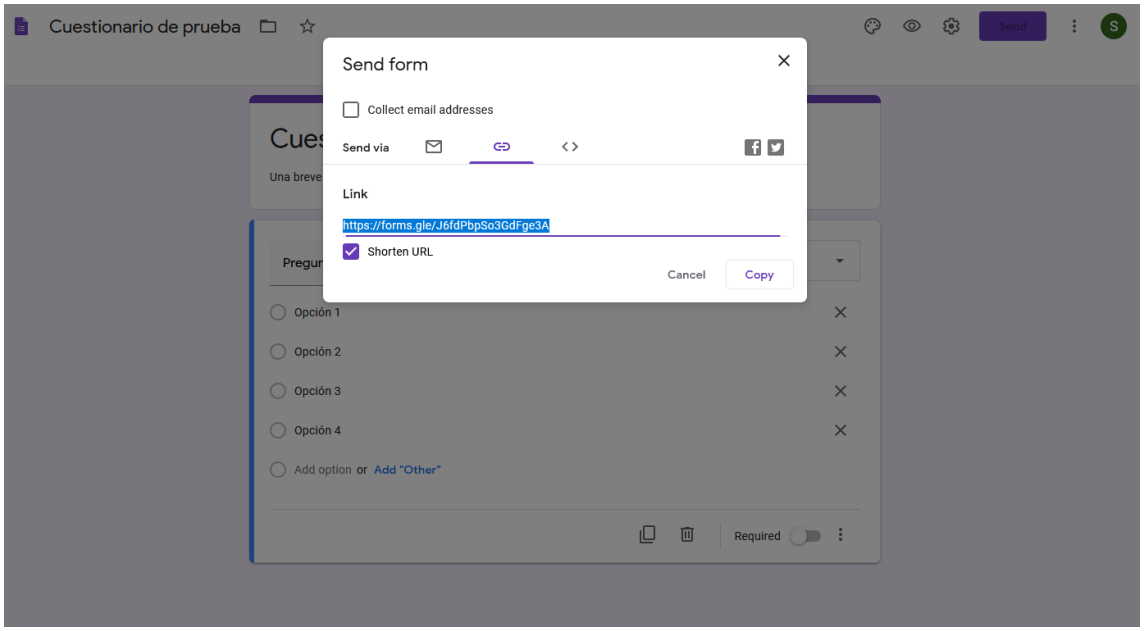

Figura 2.4: Google forms: Modal de lanzamiento

Desde esta ventana modal se configura el lanzamiento del cuestionario, pudiendo compartirlo a través de un email, de un enlace o se puede embeber en una web con un código HTML.

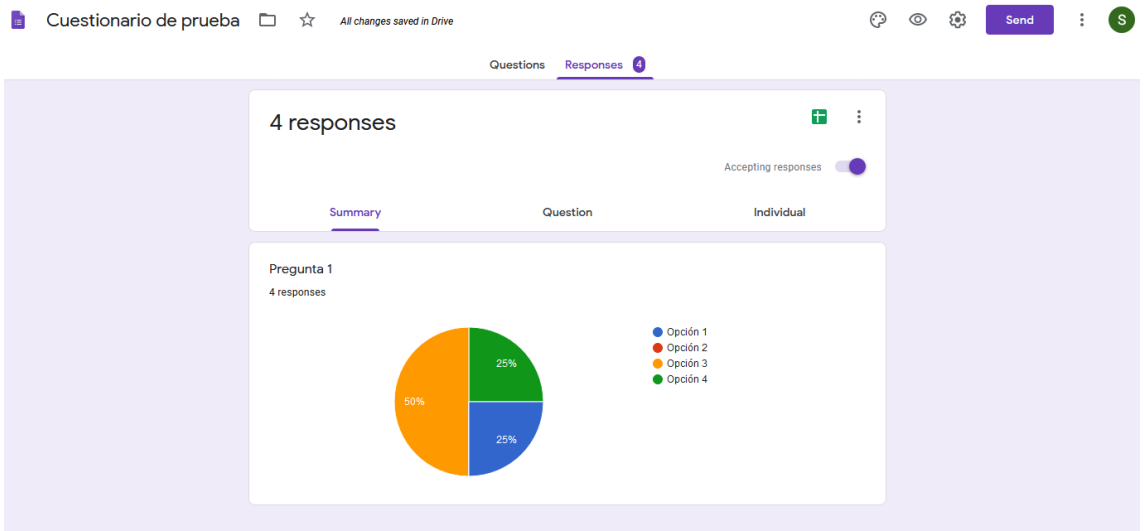

Figura 2.5: Google forms: Página de resultados

También desde esta página se puede cambiar de pestaña en el menú situado justo arriba del titulo para ver las respuestas dadas por los usuarios como se muestra en la figura 2.5 donde se pueden ver los porcentajes de respuesta a cada posible opción, la cantidad de respuestas a cada pregunta en la pestaña *Question*, y las opciones marcadas por cada usuario en la pestaña *Individual*.

2.3.2. Kahoot

Kahoot es una aplicación enfocada a un uso más lúdico, su cliente objetivo son profesores del sector de la educación secundaria o inferior. Con ella se puede realizar cuestionarios con preguntas de respuesta múltiple que se usan principalmente para amenizar las clases a modo de juego mientras se repasa la materia de la asignatura.

Al identificarse en la aplicación muestra un panel de control donde de prioriza mostrar las ultimas noticias en el centro de la pantalla dejando a los lados y en el menú superior las opciones necesarias para acceder al material creado como se observa en la figura 2.6.

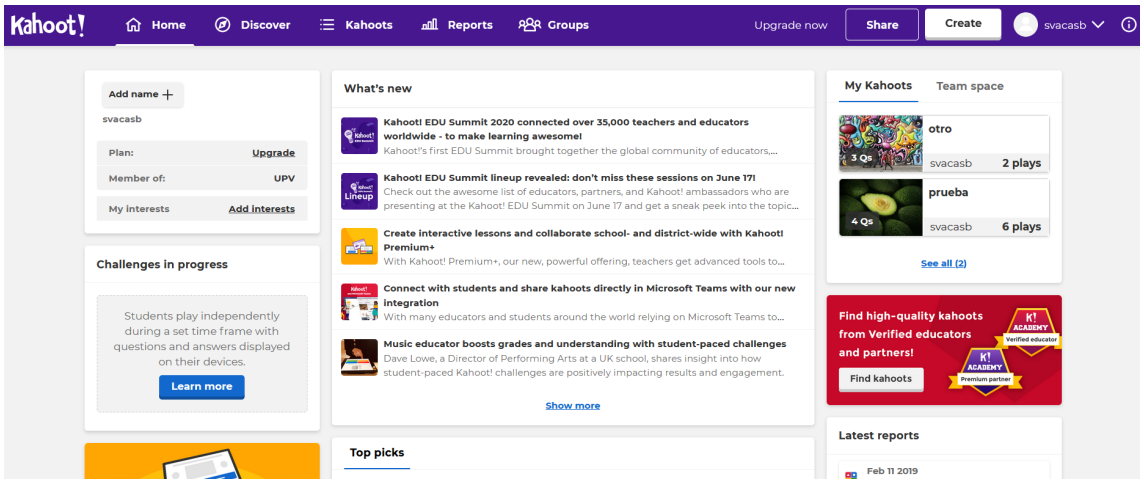

Figura 2.6: Kahoot: Página principal

Al acceder a *Kahoots* se encuentras los cuestionarios del usuario y se pueden lanzar pulsando el botón *Play* o crear nuevos desde las opciones situadas encima del listado de cuestionarios tal y como muestra en la figura 2.7.

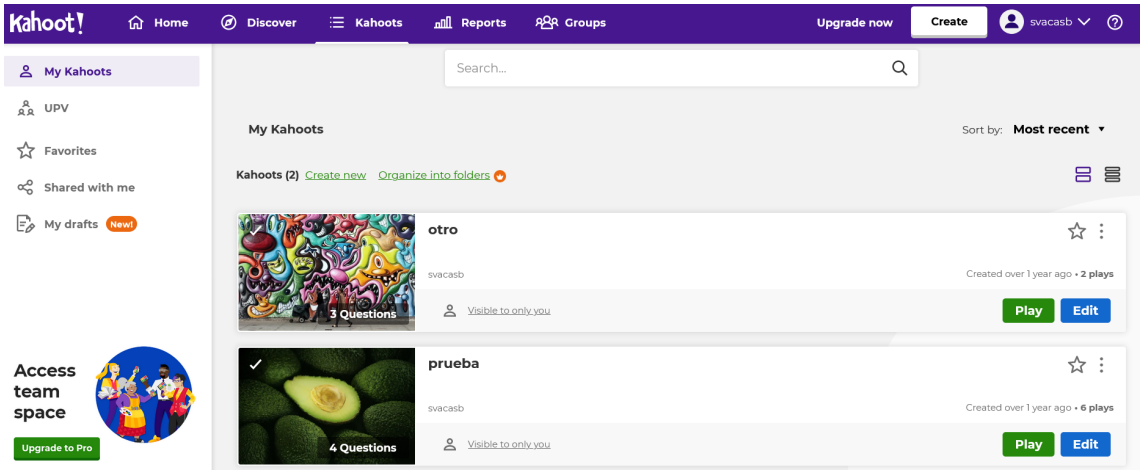

Figura 2.7: Kahoot: Página con mis cuestionarios

Si se opta por crear un nuevo *Kahoot*, se accede a la pantalla mostrada en la figura 2.8, aquí se pueden añadir únicamente preguntas de respuesta múltiple con un máximo de 4 respuestas pudiendo contener ente 1 y 3 verdaderas. También se define el tiempo que se dispondrá para responder la pregunta entre valores preestablecidos y el valor de cada pregunta dentro del cuestionario entre 3 valores también predefinidos.

Al final la creación y lanzar el cuestionario se accede a una pantalla donde se pueden seleccionar diversas opciones para el lanzamiento como dar un orden aleatorio a las preguntas y las respuestas.

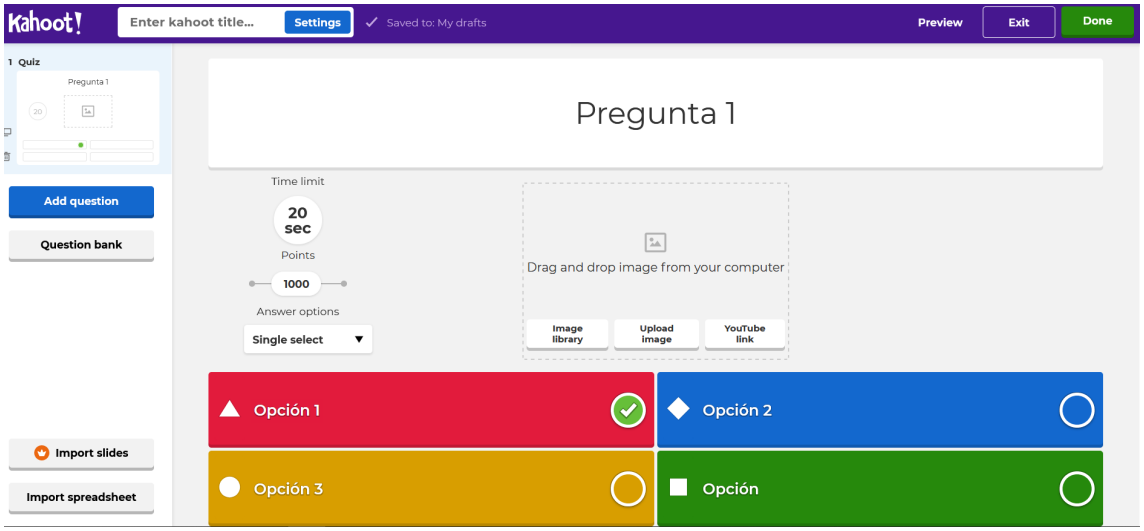

Figura 2.8: Kahoot: Página de creación de cuestionario

Posteriormente de deja al usuario en una pantalla de espera donde aguardará a que los alumnos se conecten teniendo la opción de iniciar el cuestionario en cualquier momento. Tras este inicio en la pantalla del usuario se mostrará la pregunta y sus posibles respuestas, igual que en la figura 2.9, con la idea de proyectar esta pantalla de forma que todos los alumnos la puedan ver, y mientras los alumnos verán en las pantallas de sus dispositivos cuatro botones con los colores representativos de cada respuesta.

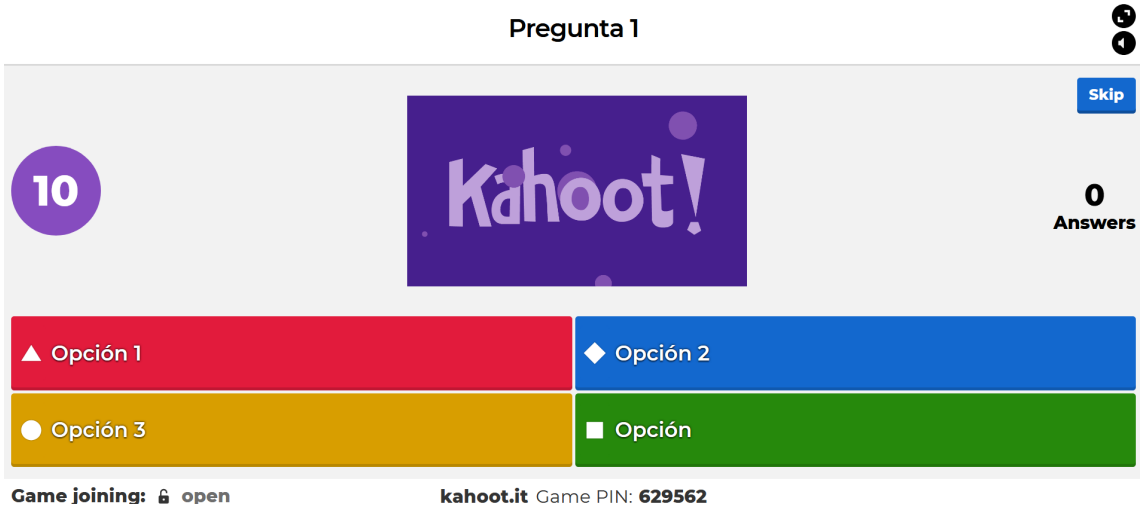

Figura 2.9: Kahoot: Página que muestra la pregunta

Cuando termina cada termina el tiempo definido para una pregunta se muestra la solución y la cantidad de veces que se ha seleccionado cada posible respuesta. Una vez terminado el examen aparece una pantalla con los alumnos que han obtenido las mejores puntuaciones. Finalmente se puede acceder a los resultados del cuestionario que como se observa en la figura 2.10, contiene un registro detallado de las respuestas dadas por cada alumno a cada pregunta.

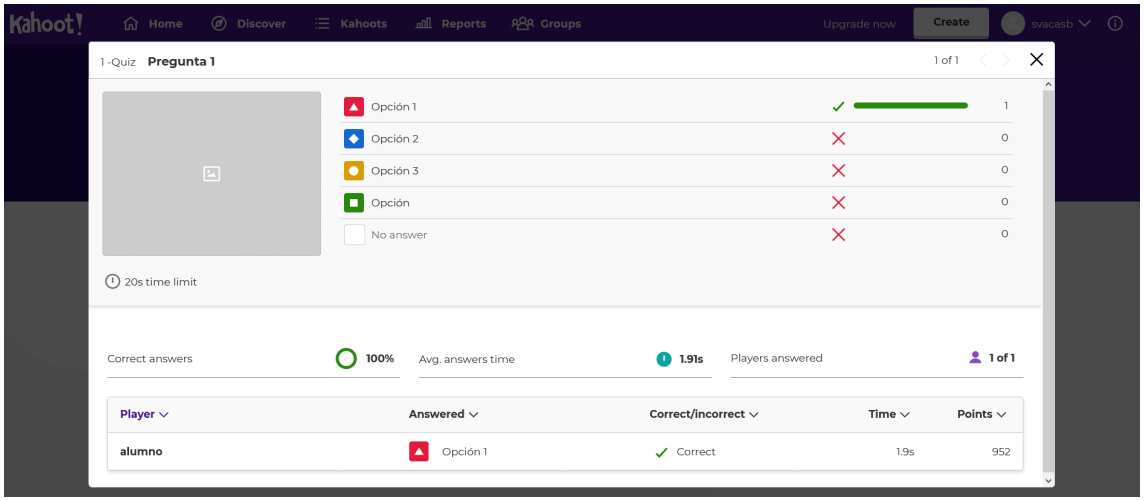

Figura 2.10: Kahoot: Página de resultados

2.3.3. Socrative

Esta herramienta que se centra en atraer como usuarios a profesores de cualquier nivel de la educación. Presenta una interfaz que muestra seriedad a la vez que da un nivel gestión de exámenes y generación de informes buenos, que le sitúan como una de las mejores opciones para el uso de este tipo de herramientas en enseñanzas superiores.

La primera pantalla mostrada tras autenticarse en la aplicación como profesor es la página de lanzamientos, esta ofrece varias opciones para lanzar un cuestionario definido o preguntas rápidas como se observa en la figura 2.11.

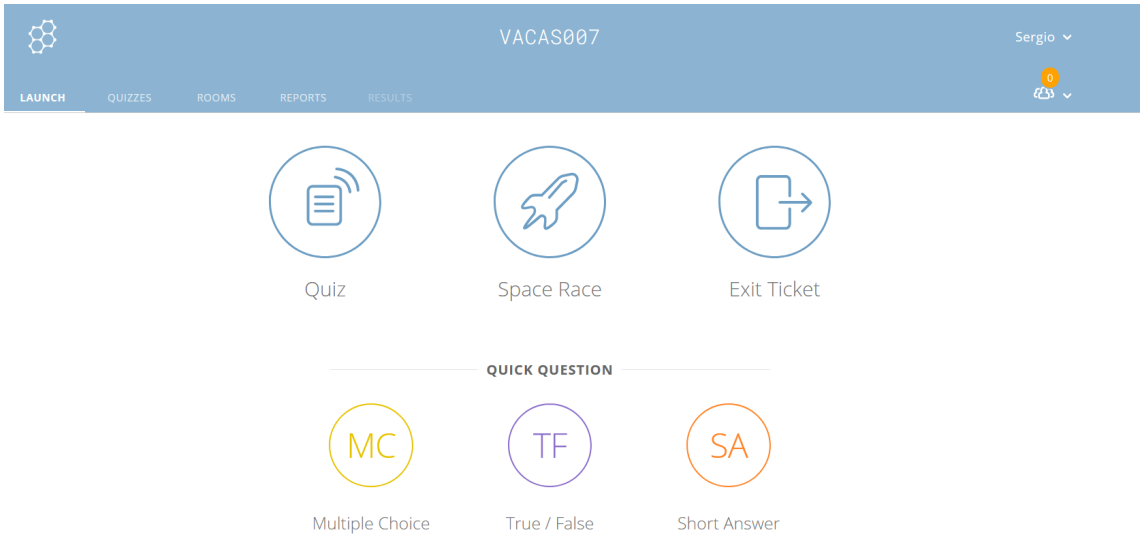

Figura 2.11: Socrative: Página de lanzamiento

Desde el menú de la cabecera se puede acceder a *QUIZZES* donde se encuentra la lista de cuestionarios del usuario como muestra la figura 2.12, desde aquí además de las funciones básicas de gestión (creación, modificación, eliminación y lectura) se pueden

realizar otras acciones como: clasificar cuestionarios por carpetas, unir dos distintos o duplicar uno existente.

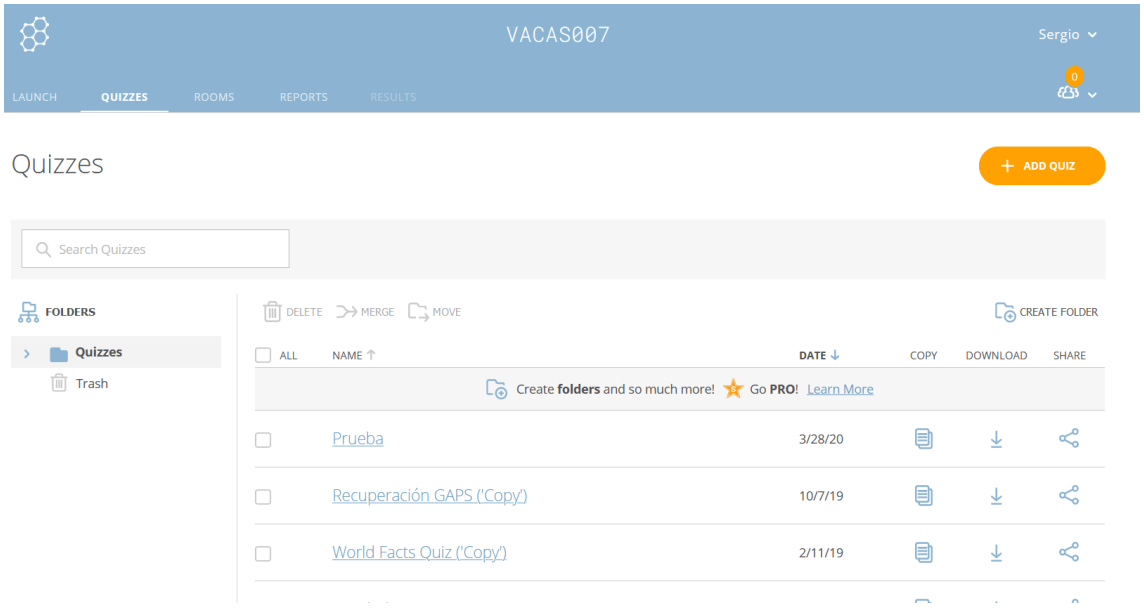

Figura 2.12: Socrative: Página con mis cuestionarios

Al entrar a crear un cuestionario se accede a la pantalla mostrada en la figura 2.13, aquí se observan los tres tipos de preguntas disponibles: respuesta múltiple, verdadero / falso y respuestas cortas.

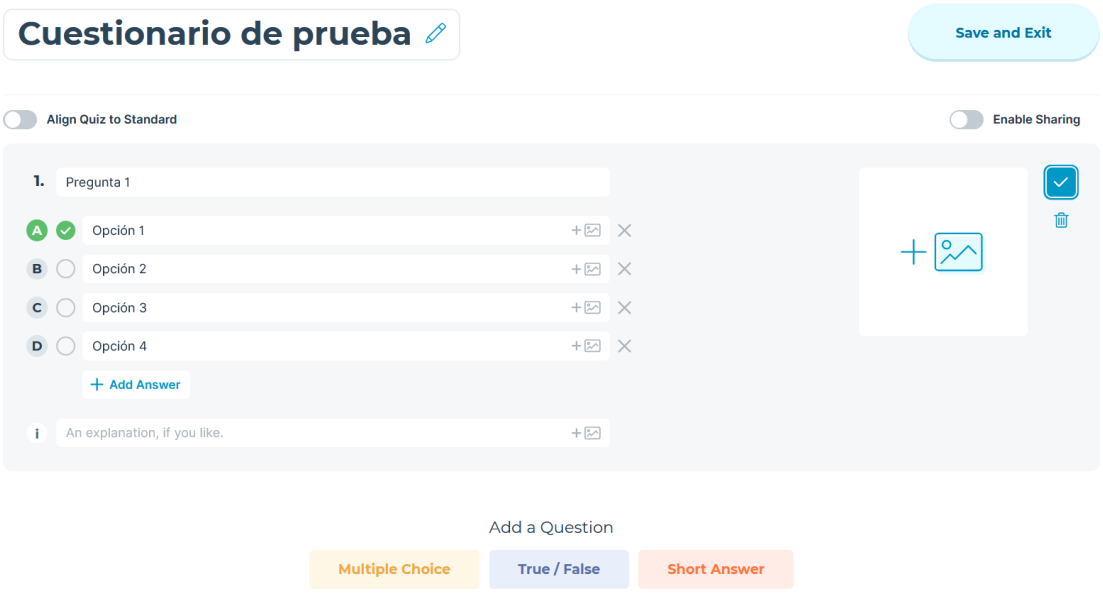

Figura 2.13: Socrative: Página de creación de cuestionario

Es necesario disponer de una sala donde poder lanzar los cuestionarios, estas salas se gestionan de manera similar a los cuestionarios desde la pantalla *ROOMS* que se muestra en la figura 2.14.

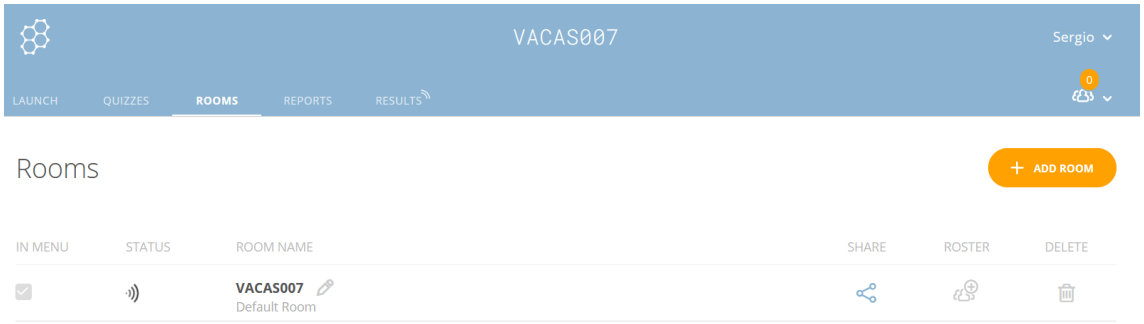

Figura 2.14: Socrative: Página con mis salas

Para lanzar un cuestionario hay que volver a la página de lanzamientos mostrada en la figura 2.11 y seleccionar al opción *QUIZ*, esto mostrará las opciones de configuración para este lanzamiento. Tras esto el usuario es redirigido a la pantalla *RESULTS* donde se muestras las respuestas a tiempo real como se observa en la figura 2.15.

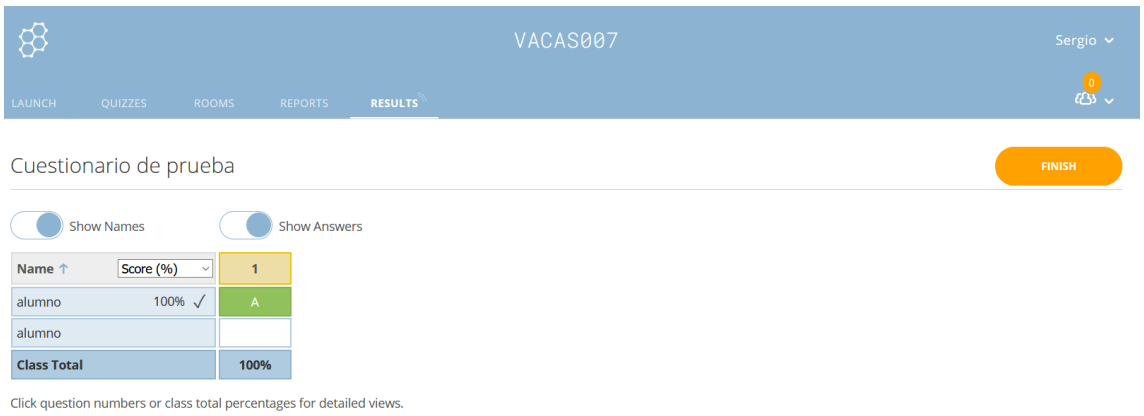

Figura 2.15: Socrative: Página de resultados

Finalmente al terminar el lanzamiento se pueden descargar estos resultados en formato CSV o PDF organizados por alumno o pregunta, un ejemplo del informe PDF por pregunta se muestra en la figura 2.16.

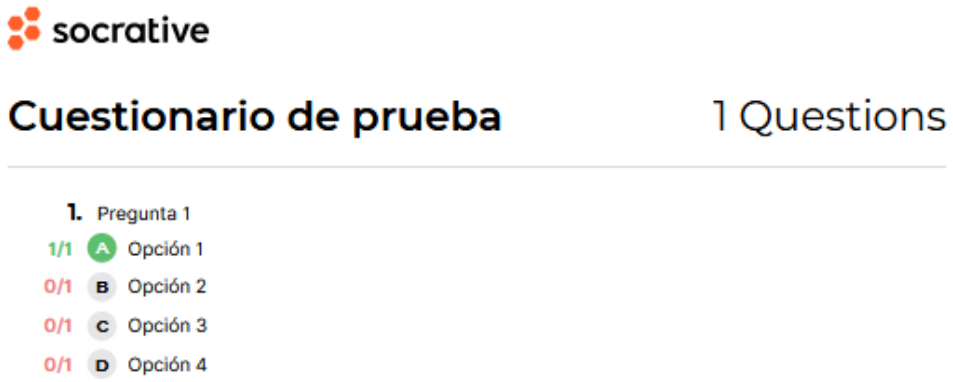

Figura 2.16: Socrative: Informe por pregunta

2.3.4. Comparación de carácterísticas

Tras analizar las herramientas de los principales competidores se ha elaborado una tabla reflejando las características que ofrecen en sus versiones menos limitadas. Esta comparación se muestra en la tabla 2.1 y ha ayudado a definir las características que es necesario incluir en el producto para no quedar por detrás del resto, y que características son las que se ha detectado que no tienen las demás herramientas e implementarlas supone una ventaja. Estas últimas destacarán este producto de la competencia y son: sistema de versiones de preguntas y exámenes, permitir internacionalizar las preguntas almacenando versiones en distintos idiomas, y compartir el material creado con otros usuarios que no tiene el porqué pertenecer a la misma organización que el usuario que lo creó.

Tabla 2.1: Comparación de los principales competidores

Propiedad	**Socrative**		**Kahoot**	**Google form**	**Teacher's Quiz**
	60$ /año	**100$/año**	**120$ /año**	**Free**	**30$ /año**
Número de salas (exámenes simultáneos)	20	20	Ilimitado	Ilimitado	Ilimitado
Estudiantes por sesión	50	150	Ilimitado	Ilimitado	Ilimitado
Estadísticas generales de respuestas	✓	✓	✓	✓	✓
Visualiza resultados en tiempo real	✓	✓	✗	✓	✓
Acceso desde cualquier dispositivo	✓	✓	✓	✓	✓
Revisión de exámenes individuales	✓	✓	✓	✓	✓
Compartir exámenes por código o enlace	✓	✓	✓	✓	✓
Cronometro	✗	✗	✓	✗	✓
Acceso restringido con identificación de estudiante	✓	✓	✗	✗	✓
Combinar exámenes	✓	✓	✗	✗	✓
Organizar exámenes en carpetas	✓	✓	✓	✓	✓
Alzar la mano en silencio	✓	✓	✗	✗	✓

Añadir logo corporativo	✗	✗	✓	✓	✓
Librería de imágenes ofrecidas por la aplicación	✗	✗	✓	✗	✓
Especificar Tiempo límite de respuesta a una pregunta	✗	✗	✓ 120 segundos o menos	✗	✓ Se puede especificar hora de inicio y fin
El profesor controla cuándo avanzar de pregunta	✓	✓	✗	✗	✓
Mostrar orden de posibles respuestas aleatoriamente	✓	✓	✓	✓	✓
Mostrar orden de preguntas aleatoriamente	✓	✓	✓	✗	✓
Formato de preguntas	Respuesta múltiple, Verdadero/ Falso y respuesta abierta	Respuesta múltiple, Verdadero/ Falso y respuesta abierta	Respuesta múltiple	Respuesta múltiple, respuesta corta, respuesta larga, desplegable, tabla de preguntas	Respuesta múltiple, Verdadero/ Falso
Máximo de posibles respuestas en preguntas de respuesta múltiple	Ilimitado	Ilimitado	4	Ilimitado	Ilimitado
Especificar idioma del examen	✗	✗	✓	✗	✓
Especificar autoría del examen	✗	✗	✓	✗	✓
Especificar autoría de la pregunta	✗	✗	✓	✗	✓
Probar el examen	✗	✗	✓	✓	✓
Barra de búsqueda de exámenes	✓	✓	✓	✓	✓
Duplicar examen	✓	✓	✓	✓	✓

Programar fecha y hora de inicio de examen	✗	✗	✗	✗	✓
Programar fecha y hora de fin de examen	✗	✗	✓	✗	✓
Crear y editar exámenes con otros usuarios	✗	✗	✓	✓	✓
Crear Grupo de profesores	✗	✗	✓	✗	✓
Compartir revisiones de exámenes	✗	✗	✓	✓	✓
Identificación que asegura que solo se conecta gente presente en el aula (ej. Combinación de números que va cambiando)	✗	✗	✓	✗	✓
Importar examen desde hoja de cálculo o CSV	✓	✓	✗	✗	✓
Importar pregunta desde hoja de cálculo o CSV	✗	✗	✓	✗	✓
Exportar resultados en PDF	✓	✓	✗	✗	✓
Exportar resultados en Hoja de cálculo	✓	✓	✓	✓	✓
Compartir exámenes públicamente con otros usuarios	✓	✓	✓	✓	✓
Compartir preguntas públicamente con otros usuarios	✗	✗	✗	✗	✓
Buscar exámenes compartidos públicamente	✗	✗	✓	✗	✓
Buscar preguntas compartidas públicamente	✗	✗	✗	✗	✓
Versionado de preguntas	✗	✗	✗	✗	✓

Versionado de exámenes	✗	✗	✗	✗	✓
Preguntas disponibles en distintos idiomas	✗	✗	✗	✗	✓

De esta comparativa se puede observar la ventaja competitiva de Teacher's Quiz, pues incluye características de las que disponen también el resto de productos a la vez que incorpora importantes funcionalidades exclusivas que permiten compartir preguntas y exámenes públicamente con otros usuarios y un sistema de versiones para preguntas y cuestionarios que permitirá una gestión mucho más eficiente de estos recursos y además mantener estos en varios idiomas.

2.4 Modelo de negocio

Actualmente existen varios modelos de negocio que podrían aplicarse a esta idea cada uno con sus puntos a favor y en contra, algunos de los modelos que se han descartado son:

- **Tradicional**: Es una opción más habitual en cualquier industria, consiste en el pago a cambio de un producto o servicio recibido. Pero este método aunque tradicionalmente ha funcionado fuera de Internet e incluso a través de tiendas *online*, para aplicaciones como las que se proponen en este trabajo no ha mostrado buenos resultados debido a la fácil aparición de productos similares con modelos de negocio más flexibles pudiendo hacer que esta idea fracase.
- **Publicitario**: Ofrece un servicio de forma gratuita, y obtiene rentabilidad a través de anuncios mostrados en la página web. Este modelo tiende a ser rentable si se posee una base de usuarios grande, además si un alumno que estuviera realizando un examen e hiciera *click* en un anuncio, perdería la concentración en este proceso y esto podría repercutir en generar malas opiniones de la aplicación por parte de los profesores. Debido a esto no se tendrían que mostrar anuncios en las pantallas donde se responde a las preguntas y por tanto no se podría aprovechar una parte importante del tráfico web.

Finalmente la opción seleccionada ha sido un modelo *freemium*. Este modelo capta usuarios ofreciendo ciertos servicios de una forma gratuita para que posteriormente se sientan atraídos por las ventajas que ofrece una versión *premium* de pago. Aquí existen también varias modalidades, en función de que características del producto que se limiten, las consideradas han sido:

- **Publicidad**: Similar al modelo publicitario descartado, con la diferencia de poder eliminar los anuncios de la página con la versión *premium*.
- **Funcionalidad**: En la versión gratuita se restringe el acceso a ciertas opciones a las que solo se puede acceder mediante la versión de pago.
- **Uso**: Se permite el uso ilimitado y gratuito un número concreto de veces.
- **Tiempo**: Es posible usar la aplicación tanto como se quiera de forma gratuita durante un tiempo limitado.

- **Capacidad**: Restringe la cantidad de almacenamiento que se puede utilizar en la herramienta.
- **Tipo de cliente**: Los usuarios pagan solo si pertenecen a un grupo que se ha designado como de pago, por ejemplo, haciendo que no se requiera pago para usuarios particulares pero si para empresas.

Para esta idea de negocio se han seleccionado varias de estas modalidades *freemium* con el objetivo de combinarlas entre sí y no restringir una de las características intensivamente, sino limitar un poco de varias, con lo que el usuario final pueda experimentar una buena parte de la propuesta de valor, pero que a su vez acabe siendo necesaria la versión *premium* si se hace un uso real de este producto. Las modalidades y sus limitaciones son:

- **Funcionalidad**

  - Solo se podrá realizar un lanzamiento al mismo tiempo.

  - No se podrán crear nuevas versiones de preguntas.

  - No se mostrarán estadísticas de las respuestas a cada pregunta.

- **Capacidad**

  - Se limitará la cantidad de salas, preguntas y *quizzes* que se pueden crear.

- **Tipo de cliente**

  - Los alumnos siempre podrán realizar las acciones que se esperen de estos usuarios de forma gratuita.

Por último, el precio de la versión *premium* se ha fijado en 30€/año.

2.5 Proyección económica a 5 años

Se ha realizado una estimación simple de ingresos y gastos en los primeros 5 años de vida del producto. Para ello se ha tenido en cuenta solo los usuarios que se estima adquirirán la licencia *premium* (que se ha fijado en 30€/año) y con el objetivo de prever la cantidad de estos, se tiene en cuenta los usuarios actuales de los principales competidores (*Kahoot* 6 millones[4] y *Socrative* 3 millones[5]) y se asume que al cabo de estos 5 años al menos se debería alcanzar el 1 % de la aplicación con menos usuarios por lo tanto 30.000.

Los mayores gastos a tener en cuenta durante el primer año son referentes al *hosting*, Marketing, el salario del CEO *(Chief Executive Officer)* los servicios de *Google Firebase*, una pasarela de pago [3], gestoría, compras de nombres de dominio y gastos derivados del uso de las instalaciones en la que se realice la actividad de la empresa. Durante el primer año no se considera necesaria la adquisición o arrendamiento de un local ya que solo existirán dos empleados, la adquisición de nombres de dominio comenzará con 24€ como inversión inicial e irá aumentando durante los años. En cuanto a los servicios de *Google Firebase* se pagan de forma variable dependiendo del tráfico que se genere hacia estos, por lo tanto se tiene en cuenta con una inversión más alta de la que se asume por el número de usuarios/año, de forma que da margen por si se le da un uso más exhaustivo del esperado por usuario. Este año se espera además de la incorporación de un CMO *(Chief Marketing Officer)* el cual podrá dedicarse a desarrollar una estrategia de marketing.

[4] Números clave de kahoot. Consultado en https://kahoot.com/company/#key-numbers
[5] Cantidad de usuarios de Socrative.Consultado en 2020 https://socrative.com/about-us/

Durante el segundo año se entiende que todos los gastos aumentarán juntos con los usuarios *premium*.

Al llegar al tercer año se contratará a un desarrollador senior para que se encargue del mantenimiento de la aplicación, desarrollos requeridos y soporte.

En el cuarto se buscará incorporar a un CTO *(Chief Technology Officer)* que tome el liderazgo del departamento de desarrollo y un CFO *(Chief Financial Officer)* debido a el volumen monetario que se espera que maneje la compañía para entonces. Además teniendo en cuenta que en estos momentos existirán al menos 5 empleados se valorará alquilar un local desde el que podamos trabajar de una forma más cercana y para eso se ha reservado una parte del presupuesto tanto para el local en sí como para los gastos derivados.

Por último el quinto año se incorporarán dos desarrolladores junior para apoyar al mantenimiento, soporte técnico y la implementación de nuevas funcionalidades. También se incorporará un trabajador con perfil administrativo.

Todo este proceso se encuentra detallado económicamente en la figura 2.18, donde se han estimado los ingresos y gastos por cada año durante los primeros cinco años de la compañía.

El resumen de esta proyección económica se plasma la figura 2.17 donde se muestra un gráfico relatando de una forma más visual los resultado de lo expuesto en este apartado. Como se observa en esta figura los dos primeros años no se obtendrán suficientes beneficiosos para amortizar los gastos, por tanto se cierra con pérdidas hasta el segundo año, pero a partir de este momento se empiezan a producir más ingresos que gastos hasta que el tercer año se consigue recuperar la inversión realizada para los años anteriores. A partir de este año los ingresos comienzan a ser cada vez mayores.

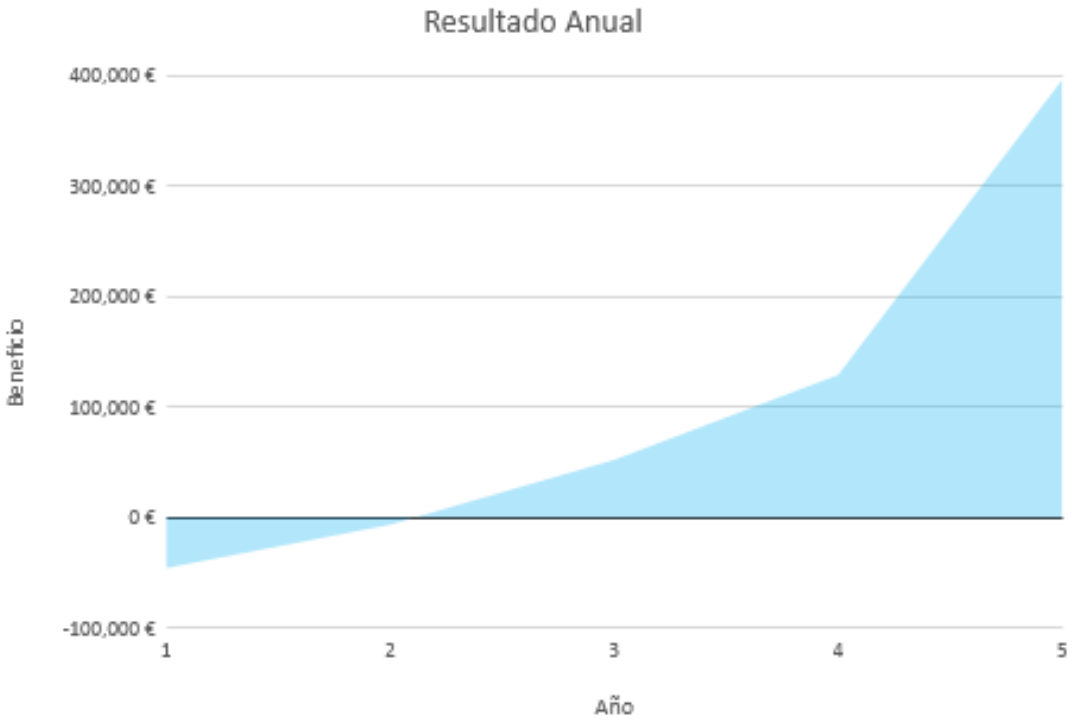

Figura 2.17: Gráfico de la proyección económica

	Años				
Número de licencias vendidas	**1**	**2**	**3**	**4**	**5**
Licencias	400	2000	6000	14000	30000
Ingresos Anuales					
30€ x Licencia	12,000 €	60,000 €	180,000 €	420,000 €	900,000 €
Total Ingresos	12,000 €	60,000 €	180,000 €	420,000 €	900,000 €
Gastos Anuales					
Nombres de dominio	24 €	50 €	100 €	150 €	200 €
Hosting	300 €	400 €	500 €	3,000 €	8,000 €
Firebase RealTimeDB	500 €	2,000 €	4,000 €	6,500 €	8,000 €
Marketing	6,000 €	12,000 €	19,000 €	60,000 €	150,000 €
TPV Virtual: pasarela de pago	120 €	660 €	1,980 €	4,620 €	9,900 €
Gestoría	600 €	600 €	1,000 €	2,000 €	2,000 €
Alquiler oficina, muebles oficina e instalaciones	0 €	0 €	0 €	20,000 €	30,000 €
Internet, electricidad, agua, teléfono, etc.	300 €	600 €	900 €	4,500 €	5,500 €
CTO	0 €	0 €	0 €	40,000 €	45,000 €
CMO	25,000 €	25,000 €	35,000 €	40,000 €	45,000 €
CFO	0 €	0 €	0 €	35,000 €	40,000 €
CEO	25,000 €	25,000 €	35,000 €	40,000 €	45,000 €
Desarrollador Senior	0 €	0 €	30,000 €	35,000 €	40,000 €
Desarrollador Junior 1	0 €	0 €	0 €	0 €	25,000 €
Desarrollador Junior 2	0 €	0 €	0 €	0 €	25,000 €
Personal administrativo	0 €	0 €	0 €	0 €	25,000 €
Total gastos	***57,844 €***	***66,310 €***	***127,480 €***	***290,770 €***	***503,600 €***
Resultado Anual	-45,844 €	-6,310 €	52,520 €	129,230 €	396,400 €
Resultado Anual Acumulado	-45,844 €	-52,154 €	366 €	129,596 €	525,996 €

Figura 2.18: Proyección económica

2.6 Análisis DAFO

Con el objetivo de clarificar cuales son los mayores puntos a favor y en contra de esta idea, y de esa forma saber hacia donde orientar futuras estrategias se ha realizado un análisis DAFO, definido como: «una herramienta que permite al empresario analizar la realidad de su empresa, marca o producto para poder tomar decisiones de futuro» [4].

Tabla 2.2: Matriz DAFO

Debilidades ▪ Poca experiencia en gestión empresarial. ▪ Al principio seremos una marca desconocida.	**Fortalezas** ▪ Innovaciones en el mantenimiento de cuestionarios que nos sitúan por encima de la competencia. ▪ No se requiere un coste muy elevado para la puesta en marcha.
Amenazas ▪ Otras compañías que trabajan en el campo de la educación pero que no ofrecen estos servicios podrían empezar a hacerlo y convertirse en competencia. ▪ Productos ya establecidos en el mercado podrían observar las carencias en sus servicios que hemos detectado y reaccionar para no perder usuarios.	**Oportunidades** ▪ Creciente demanda en aplicaciones que dinamicen la enseñanza. ▪ Sinergias con otras compañías del campo de la educación que todavía no ofrecen un servicio similar.

Como se observa en la tabla 2.2, las mayores debilidades observadas tienen que ver con la falta de experiencia del equipo en gestión empresarial y el hecho de ser un nuevo competidor desconocido. Por otro lado también se dispone de fortalezas como las innovaciones aportadas en el mantenimiento de cuestionarios y el bajo coste de la puesta en marcha del proyecto.

También se han detectado amenazas sobre todo debido a la vulnerabilidad de los productos software ante la copia de sus funcionalidades para integrarlas en herramientas de la competencia o de nuevo competidores. Aun así existen oportunidades como la creciente demanda de este tipo de herramientas lo que hace que cada vez crezca más nuestro mercado y la posible colaboración con otras herramientas del sector de la educación que no ofrezcan todavía el servicio que nuestra aplicación aporta.

2.7 Conclusiones de la evaluación

La idea de negocio propuesta no solo ofrece innovaciones en el mercado creciente de las aplicaciones destinadas a dinamizar y automatizar ciertos aspectos de la enseñanza, además gracias a estos estudios se han podido definir las características concretas que se van a implementar y que marcan una propuesta de valor muy clara con respecto a la competencia.

Por ejemplo, con el *Lean Canvas* se ha plasmado la directrices generales sobre el problema detectado en este mercado el cual no ofrece sistemas de mantenimiento del material didáctico suficientemente eficiente, también la solución que aporta desde Teacher's Quiz, los clientes objetivo a los que se dirigirán las campañas de *marketing* que en este caso serán profesores de cualquier fase de la enseñanza puesto que los alumnos, pese a poder influenciar al profesor en la decisión de que herramienta elegir, no se considera que esta influencia pueda ser suficiente como para considerarlos un objetivo en el *marketing*, al menos por ahora. Este estudio también ha ayudado a concretar que métricas se tendrán en cuenta tras la puesta en marcha del producto, canales por los que se distribuirá la aplicación y costos e ingresos a esperar.

Por otra parte, se ha realizado un estudio de la competencia en el que se han examinado las aplicaciones más usadas en este sector, lo que ha permitido observar las características que tienen el resto de herramientas y por tanto tendrán que ser incorporadas en la aplicación resultante de este proyecto, pero también las carencias de estas aplicaciones sobre todo en cuanto al mantenimiento del material creado, al uso de los datos que se pueden recuperar de las respuestas y las restricciones en cuanto a compartir este material fuera de una organización predefinida.

Además se ha concluido que el modelo de negocio *freemium* es el que mejor se adapta a este producto, permitiendo crear una versión gratuita pero limitada en funcionalidades con la que atraer posibles usuarios que posteriormente paguen la licencia *preemium* fijada en 30€/año para poder disfrutar de todas las funcionalidades de la aplicación.

Gracias a la proyección económica llevada a cabo, se considera que esta idea de negocio requeriría una inversión inicial aproximada de 52.000€, los cuales se destinarían a cubrir gastos de los dos primeros años, ya que a partir del tercero se obtendrían beneficios. A cambio de este dinero el inversor obtendría un porcentaje de la compañía. Con el objetivo de conocer que porcentaje se tendría que ceder a este propósito, se ha calculado el EBITDA [2] *(Earnings Before Interest Taxes Depreciation and Amortization)* del quinto año y se ha multiplicado por 10, ya que este es un valor aproximado al usado como multiplicador Europa para empresas del sector software ofrecido a través de Internet. El producto de esta multiplicación da un valor aproximado de la empresa en esos momentos, el resultado obtenido es 3.964.000€, por lo tanto, ofreciendo un 15 % de la compañía un inversor recuperaría 11,4 veces su inversión en 5 años.

Finalmente, se ha realizado un análisis DAFO que servirá de partida para un futuro plan de *marketing* basado en las fortalezas y oportunidades que observadas para esta idea de negocio que parte con innovaciones que la hacen única en un mercado cada vez mayor y que no descarta aprovechar las sinergias con otras compañías del sector.

CAPÍTULO 3
Desarrollo de la idea de negocio

Para llevar acabo la idea propuesta se ha optado por seguir la metodología *Lean Startup* [5], esta se basa en realizar hipótesis sobre la idea de negocio e ir verificándolas a medida que se avanza, marcando objetivos en el desarrollo que dan lugar a versiones tempranas del producto con las funcionalidades necesarias para poder probar estas hipótesis, cada una de estas versiones forman un mínimo producto viable o MVP (por sus siglas en inglés *Minimum Viable Product*). Estos MVP se ponen a prueba con un conjunto de usuarios *early adopters* y de esta forma recibir *feedback* de estos potenciales clientes. Es aquí donde radica el potencial de esta metodología, ya que, con esta información se pueden validar las hipótesis propuestas y gracias a esto dirigir el desarrollo del producto en una dirección u otra, incluso cambiar estrategias o la misma idea de negocio si es preciso.

Todo este proceso es considerado un ciclo y la idea es repetir estos ciclos a cada evolución propuesta del producto. Es por esto que todo lo nombrado en el capítulo 2 ha sido modificado repetidas veces adaptándose a las necesidades y observaciones que se han obtenido durante su desarrollo, por tanto, no debe tomarse lo expuesto en estos análisis como algo inamovible sino más bien como una imagen del estado actual esta idea de negocio, la cual seguirá evolucionando a cada iteración.

Con el objetivo de poder hacer frente al nivel de adaptabilidad que requiere *Lean Startup* se han incorporado distintas técnicas de otras metodologías ágiles, así como siguiendo los principios del manifiesto ágil [7].

Una vez definida la metodología se plasmó la idea de negocio en el *Lean canvas* donde comenzó coger forma, para posteriormente realizar un estudio de mercado que mostraría las características concretas de los productos competidores, con esto se pudo definir exactamente que se quería incorporar en Teacher's Quiz, tanto características que ya poseyeran estas otras aplicaciones, como lo que se pretendía ofrecer que nadie más tuviera.

Tras esto se actualizó el *Lean canvas* con una perspectiva más consciente de las ventajas con las que se contaba y comenzó el desarrollo de una proyección económica para confirmar la viabilidad del producto y tener una idea de que inversión inicial sería necesaria para ponerlo en marcha.

Al mismo tiempo, ahora que se tenia una visión clara de las funcionalidades a incorporar se definió un mapa de características ilustrado en la figura 3.1.

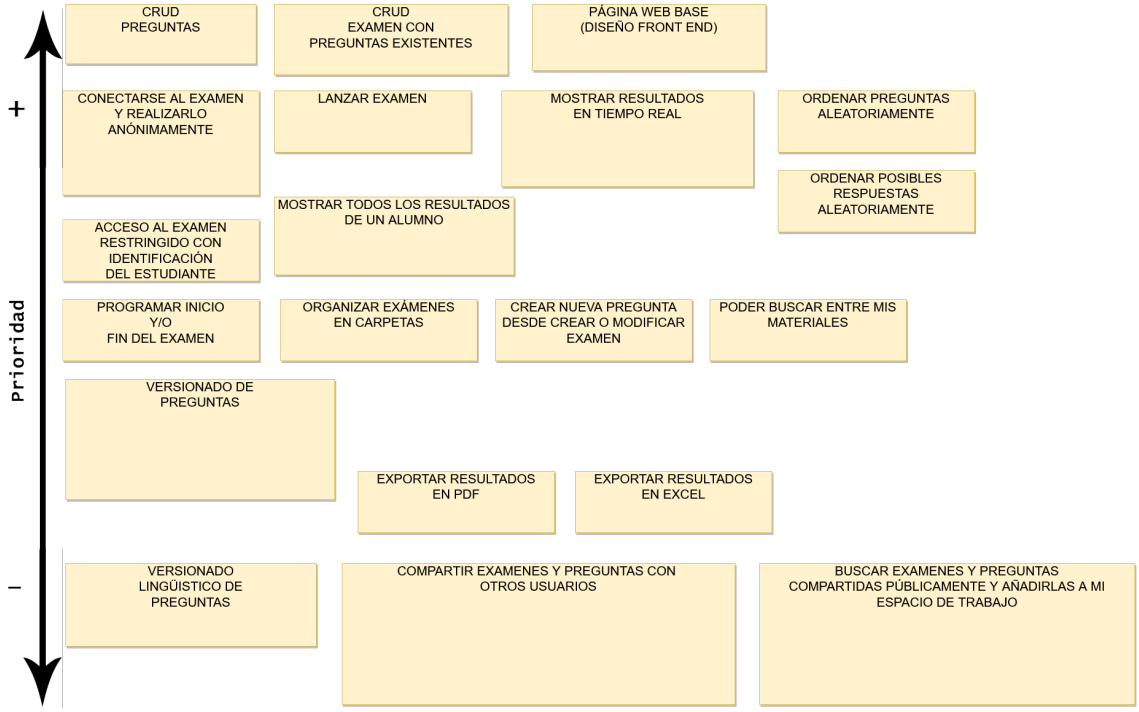

Figura 3.1: Mapa de características inicial

En este mapa se definen las características que se van a desarrollar, también están ordenadas por prioridad de arriba a bajo siendo las de la parte superior las más prioritarias y con su tamaño se simboliza una estimación del esfuerzo que requerirán, siendo las más grandes las que se piensa que serán más costosas. Este mapa tiene una gran importancia debido a que ilustra de una manera clara la prioridad y complejidad de las características, esto permite realizar una estrategia para el desarrollo del producto final, se pueden resolver cuestiones como: ¿Qué se podría incluir en el primer MVP?, o ¿De qué funcionalidades se podría llegar a prescindir en caso de añadir más características en el futuro y que no se pudiera abarcar todo?.

Finalmente estas características se fraccionan en tareas más simples y se convierten en unidades de trabajo (UT) que se incorporan al *backlog*, el cual es una pila que contiene las tareas a realizar en el desarrollo. Para estas UT se definen pruebas de aceptación (PA), que son los requisitos que debe cumplir la UT para que sea aceptada como terminada. El *backlog* se ha decidido gestionar a través de un tablero *kanban* en el que se definen 5 columnas, como se muestra en la figura 3.2, estas representan el estado actual de la UT:

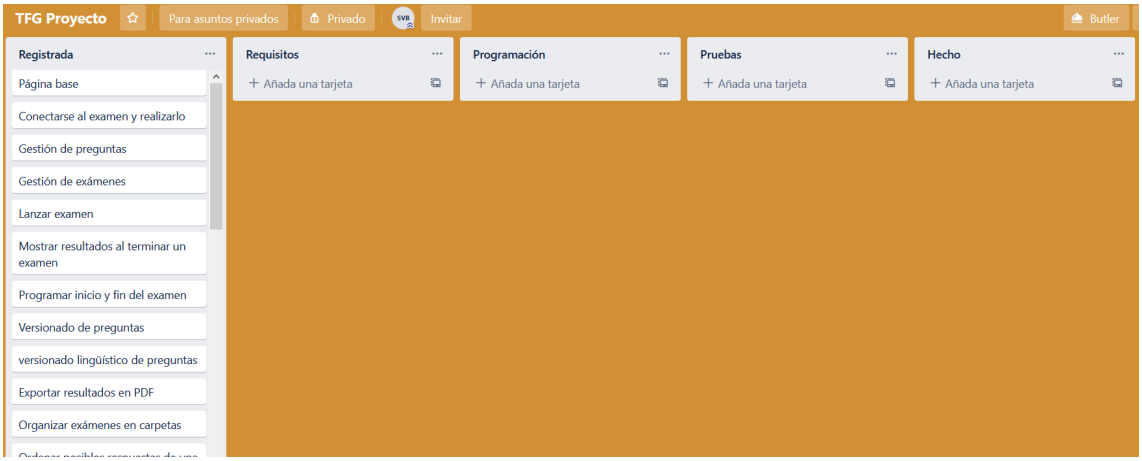

Figura 3.2: Tablero *kanban*

- **Registrada**: Representa el *backlog*, en esta columna se almacenan las UT que se extraen del mapa de características.
- **Requisitos**: Una vez se decide que unidades de trabajo e van a abordar en una iteración, estas se especifican para que expresen de una forma clara lo que se pretende conseguir, se definen las PA que se tendrán que superar para dar esta unidad de trabajo por finalizada, se estiman y colocan en un orden dependiendo de su prioridad. El resultado se puede observar en la figura 3.3.

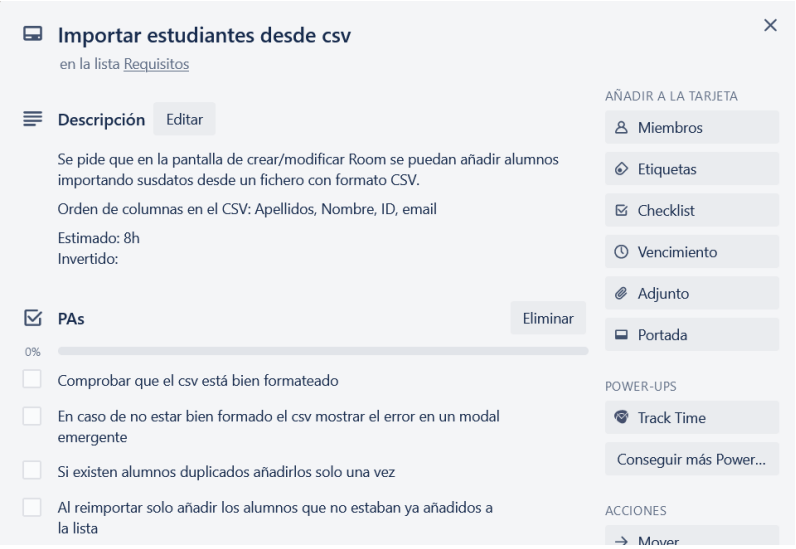

Figura 3.3: Ejemplo de una UT

- **Programación**: Cuando se comienza el desarrollo de una de estas tareas se mueve a esta columna.
- **Pruebas**: Contiene las unidades de trabajo que se han realizado y están en proceso de pruebas para confirmar si están terminadas o tienen que volver a programación.
- **Hecho**: Por último si pasan las pruebas establecidas se mueven a esta sección donde se encuentran las UT terminadas y terminan como se se ve en la figura 3.4.

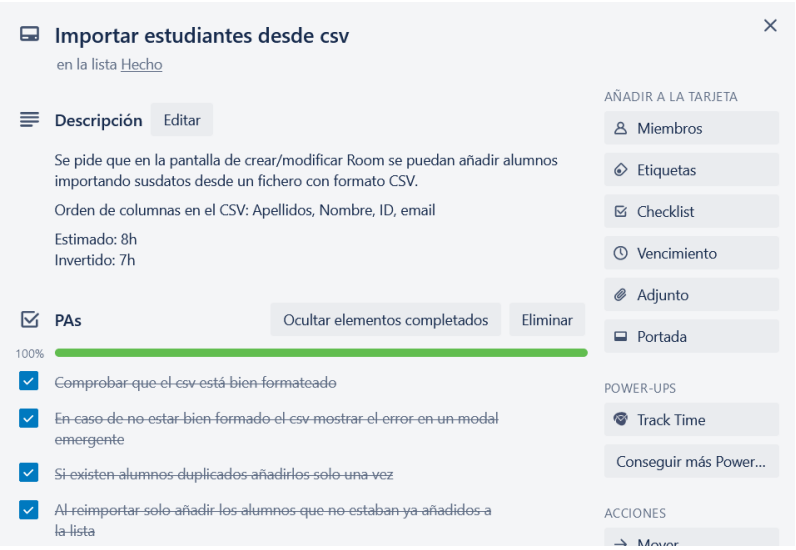

Figura 3.4: Ejemplo de una UT completada

Cuando todas las UT que forman parte de un ciclo de desarrollo pasan por todo este proceso se obtiene un resultado que es puesto a prueba por *early adopters* en experimentos. Los resultados de estas pruebas son *feedback* con el que será validado si se avanza en la dirección correcta dando pie a valorar si son necesarios cambios y aportando datos que ayuden a definir el próximo ciclo de *Lean Startup*.

3.1 Desarrollo del primer MVP

Al comienzo del proyecto hay que decidir cuales van a ser los objetivos para el primer ciclo. A través de una reunión se consensuó que para primer MVP se debería cubrir las funciones más básicas de la idea de negocio y que pueda soportar las conexiones de un grupo de entre 30 y 40 alumnos.

Con el mapa de características definido se puede tomar la decisión de qué funcionalidades se van a incluir en el primer ciclo, la idea es cubrir lo mínimo para crear y lanzar un cuestionario que los alumnos puedan responder. Las características seleccionadas fueron las que se marcan en rojo en la figura 3.5.

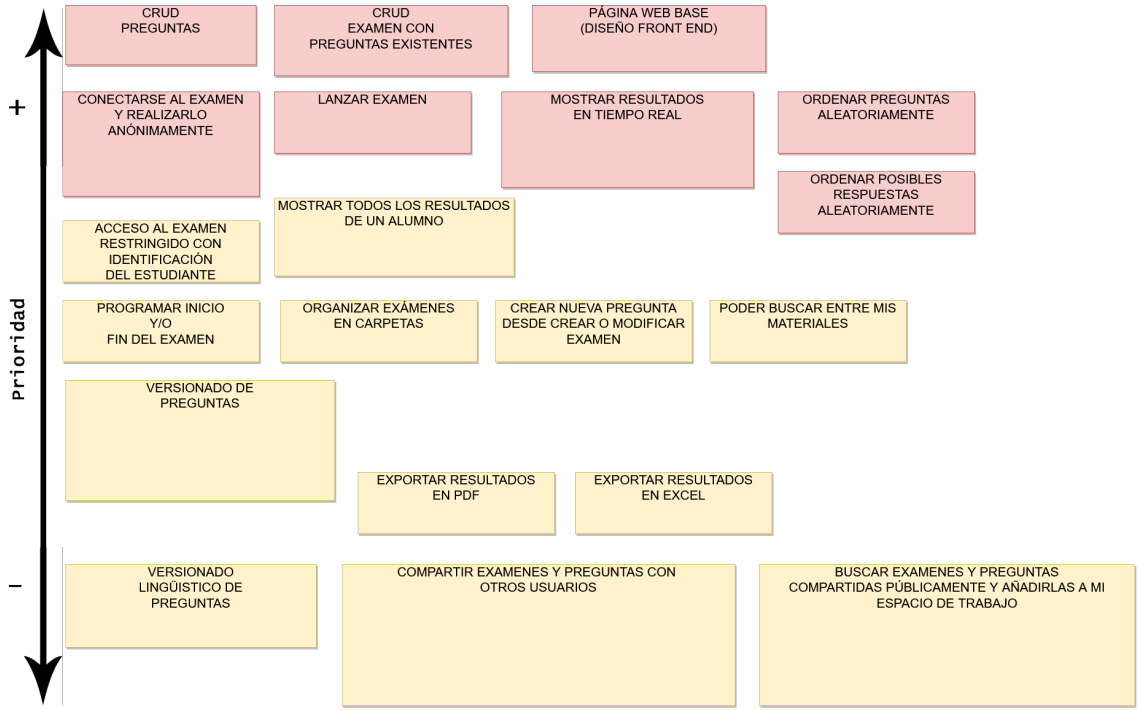

Figura 3.5: Mapa de características del primer MVP

Con esta selección se creará una página web base desde la que empezar a trabajar, la cual contendrá un sistema de autenticación muy sencillo, también se podrá gestionar de una forma básica preguntas y exámenes, y lanzar los cuestionarios para que los alumnos respondan, además se mostrarán los resultados obtenidos por estos alumnos.

Tras la elección se movieron las UT implicadas a la columna de requisitos del tablero *kanban* como se ve en la figura 3.6 y se completaron con la especificación necesaria.

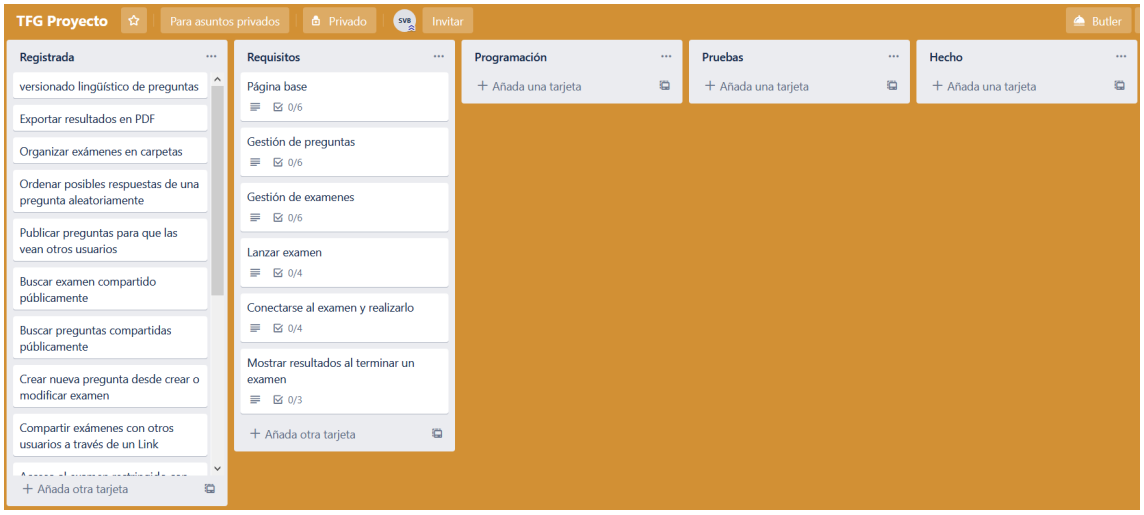

Figura 3.6: Tablero *kanban* al inicio del primer MVP

Esta especificación consta de una descripción del trabajo a realizar y las pruebas de aceptación para cada tarea:

- **Página base**: Crear la estructura básica de la aplicación, integrando las tecnologías elegidas para poder realizar una versión sencilla con la que poder hacer un *login* simple con la arquitectura propuesta y utilizando la plantilla HTML seleccionada. Además se podrá modificar los datos del usuario actual para tener un ejemplo simple de modificación de entidades.

  Pruebas de aceptación:

  - La aplicación puede realizar una versión muy simple de un *login*.
  - Se utiliza la plantilla HTML seleccionada.
  - La contraseña de los usuarios almacenada cifrada en MD5.
  - El usuario actual puede actualizar sus datos.
  - Al modificar el email del usuario solo se acepta una cadena con formato de email.

- **Gestión de preguntas**: El profesor podrá crear, editar y eliminar preguntas. Toda las preguntas del profesor conectado se mostrarán en una tabla paginada.

  Pruebas de aceptación:

  - Se puede crear preguntas.
  - Se pueden modificar preguntas creadas.
  - Se pueden borrar preguntas creadas.
  - Se pueden borrar preguntas creadas.
  - Se puede especificar el autor de cada pregunta.
  - Las preguntas tienen que tener un mínimo de 2 posibles respuestas.
  - Validar los campos antes de enviar el formulario.

- **Gestión de exámenes**: En el apartado exámenes el profesor podrá crear modificar y borrar exámenes, estos estarán mostrados en una tabla paginada.

  Pruebas de aceptación:

- Se puede crear Exámenes.
- Se pueden modificar exámenes creados.
- Se pueden borrar exámenes creados.
- Se puede añadir preguntas ya existentes a los exámenes desde una opción en la tabla de MyQuestions.
- Se pueden eliminar preguntas del examen.
- Se almacena la fecha de última modificación

- **Lanzar examen**: El profesor podrá lanzar un examen para que lo realicen aquellas personas que tengan el código del examen.

  Pruebas de aceptación:

  - Se podrán lanzar exámenes ya creados.
  - Se generará un código para acceder al examen.
  - Se podrá conectar cualquiera que tenga el código.
  - Se podrá conectar cualquiera que tenga el código.

- **Conectarse al examen y realizarlo**: Cualquiera con el código de un examen podrá conectarse y realizarlo

  Pruebas de aceptación:

  - Con el código dado al lanzar el examen se podrá entrar al mismo.
  - El usuario podrá avanzar de pregunta cuando quiera.
  - Al llegar a la última pregunta en lugar del botón avanzar se mostrará el botón terminar
  - No se podrá retroceder a la pregunta anterior.

- **Mostrar resultados de un examen a tiempo real**: El profesor podrá ver resultados individuales de las preguntas que responden los alumnos a tiempo real.

  Pruebas de aceptación:

  - Se mostrará si cada respuesta es correcta o no de los usuarios que estén realizando el examen.
  - Los ids de *questions* de la tabla de resultados serán un enlace a una pantalla que muestre el detalle de la pregunta.

Se ha realizando estimaciones del tiempo que costará su desarrollo utilizando para ello números de la sucesión de Fibonacci, esto se debe a que al no poder elegir cualquier número de horas en unas tareas el equipo se verá forzado a elegir un número algo mayor del que pondrían y en otras uno un poco menos de forma que esas diferencias puedan hacer que infraestimaciones o sobreestimaciones se compensen, además el salto entre números es mayor conforme va avanzando la sucesión con lo cual aumenta el margen de incertidumbre conforme se trata de tareas mayores. La idea de utilizar estos números está extraída de metodologías de estimación que utilizan esta serie de números con ese mismo propósito, por ejemplo *Planning poker* [6]. La estimación junto con el tiempo real invertido en la primera entrega se muestra en la tabla 3.1.

Tabla 3.1: Distribución de horas por tarea - primera entrega

UT	Estimado	Invertido
Página base	13h	15h
Gestión de preguntas	21h	36h
Gestión de exámenes	21h	15h
Lanzar examen	13h	20h
Conectarse al examen y realizarlo	21h	11h
Mostrar resultados en linea de un examen en ejecución	5h	6h
Total	**94h**	**103h**

Tras el desarrollo de estas tareas se validó el resultado con el *product owner*[1]. De esta validación se extrajo que era necesario incluir algunas funcionalidades adicionales antes de realizar el primer experimento. Estos nuevos requerimientos no habían sido tenidas en consideración previamente, de forma que se añadieron al *backlog* y se repitió el proceso de estimación, desarrollo y pruebas. Las estimaciones y horas dedicadas se muestran en la tabla 3.2 y las especificaciones de estas tareas son:

- **Los titulos serán enlaces**: tanto en preguntas como en examenes los títulos mostrados en la tabla de gestión serán en laces que llevarán a la página de modificación

  Pruebas de aceptación:

  - Los títulos de MyQuestions enlazaran con updateQuestion.
  - Los títulos de MyQuizzes enlazaran con updateQuiz.

- **Barra de búsqueda en las tablas de gestión**: Mostrar una barra de búsqueda en todas las tablas de gestión desde donde se puedan filtrar elementos.

  Pruebas de aceptación:

  - Campo de búsqueda en la parte superior de la tabla.
  - Filtra por los datos visibles en la tabla.

- **Cambios en lo campos del formulario de pregunta**: Eliminar del formulario o modificar los campos acordados

  Pruebas de aceptación:

  - No aparece el campo autor.
  - No aparece el campo autor.
  - Reemplazar el literal del campo titulo por pregunta.

- **Detener examen**: Crear un botón junto a los resultados de cada examen que lo detenga.

  Pruebas de aceptación:

  - Mostrar botón en resultados que termine la ejecución de ese examen.
  - Tras terminar la ejecución los estudiantes no podrán seguir respondiendo

[1]Product owner: Es un rol dentro del contexto del desarrollo ágil. Su tarea principal es representar las necesidades del cliente entre otras. En este trabajo el tutor ha asumido este papel en cuanto a validar el trabajo realizado, definir prioridades y los MVP

Tabla 3.2: Distribución de horas por tarea - segunda entrega

UT	Estimado	Invertido
Los títulos serán enlaces	0h	0.5h
Barra de búsqueda en tablas de gestión	1h	0.5h
Cambios en lo campos del formulario de pregunta	0h	0h
Detener examen	5h	4h
Total	**6h**	**5h**

3.1.1. Experimento 1

El objetivo del primer experimento es probar que la aplicación podía soportar conexiones simultáneas de unos 35 alumnos que estuvieran realizando un cuestionario, obtener *feedback* de esta primera versión y la toma de ideas sobre cuales serían las funcionalidades más necesarias en la próxima entrega o si fuera necesario cambiar alguna parte del flujo básico de la aplicación para hacerla más intuitiva.

Este experimento fue realizado el 17 de diciembre de 2019 con el profesor y alumnos de la asignatura *Proyecto de Ingeniería de Software*, y se realizaría un cuestionario de 20 preguntas que respondería el grupo de mañanas formado por 38 alumnos y posteriormente el grupo de tardes que cuenta con 40.

Durante el experimento no se produjo ningún error en la aplicación de forma que se pudo llevar a cabo la construcción del cuestionario el día anterior desde un ordenador y los alumnos pudieron resolverlo a través de sus teléfonos móviles sin incidentes, exceptuando el de una alumna la cual cerró por accidente el navegador y tuvo que volver a empezar el cuestionario desde cero.

Tras el experimento se hizo una reunión de retrospectiva en la que se analizó el experimento, en ella se alcanzaron varias conclusiones:

- El proceso por el cual los alumnos responden era claro e intuitivo, ya que ninguno tuvo dudas sobre su funcionamiento a pesar de que no se les había explicado nada de la aplicación anteriormente.
- Es sería útil añadir una funcionalidad que permitiera proseguir los exámenes en caso de una interrupción inesperada del proceso.
- El proceso por el cual se añaden preguntas a un cuestionario es ineficiente y no muy intuitivo, hace falta mejorarlo.
- La aplicación es capaz de soportar conexiones simultáneas en el mismo *quiz* de al menos 40 alumnos.
- Se pudo observar las respuestas a tiempo real de los alumnos lo que prueba la efectividad del sistema implementado.
- No se debe poder modificar o eliminar un cuestionario ni una pregunta si estos han sido utilizados, con el objetivo de no alterar la información de las respuestas almacenadas. Por tanto es necesario implementar un sistema que permita realizar cambios en preguntas sin preservando la integridad de los resultados almacenados.
- Es necesario que las distintas respuestas de una pregunta conserven el orden dado por el profesor para poder incluir respuestas como: «Todas las anteriores».
- Poder dar de alta alumnos con el fin de restringir el acceso a un *quiz* y el profesor lo requiere.

- Mostrar los resultados de todos los cuestionarios realizados en una misma clase.
- Desde la pantalla de resultados, se requiere poder visualizar una pregunta para realizar revisiones de los cuestionarios en clase tras ser respondidos por los alumnos.
- Es necesario conocer la fecha de última modificación de un *quiz*.

Estas conclusiones se estudiaron y se realizaron nuevas historias de usuario basadas en ellas para volver a comenzar el ciclo de *Lean Startup* que llevará al segundo MVP.

3.2 Desarrollo del segundo MVP

La retrospectiva del primer ciclo proporcionó información importante con la que se redefinió el mapa de características añadiendo nuevas funcionalidades y decidiendo de nuevo cuales son las prioridades. El resultado de estas decisiones de plasma en el mapa que se muestra en la figura 3.7 donde se muestran las características ya implementadas en rojo y las que formarán parte del segundo MVP de color azul y verde. Esta separación en dos colores es debida a que el desarrollo de este MVP está dividido en dos entregas distintas.

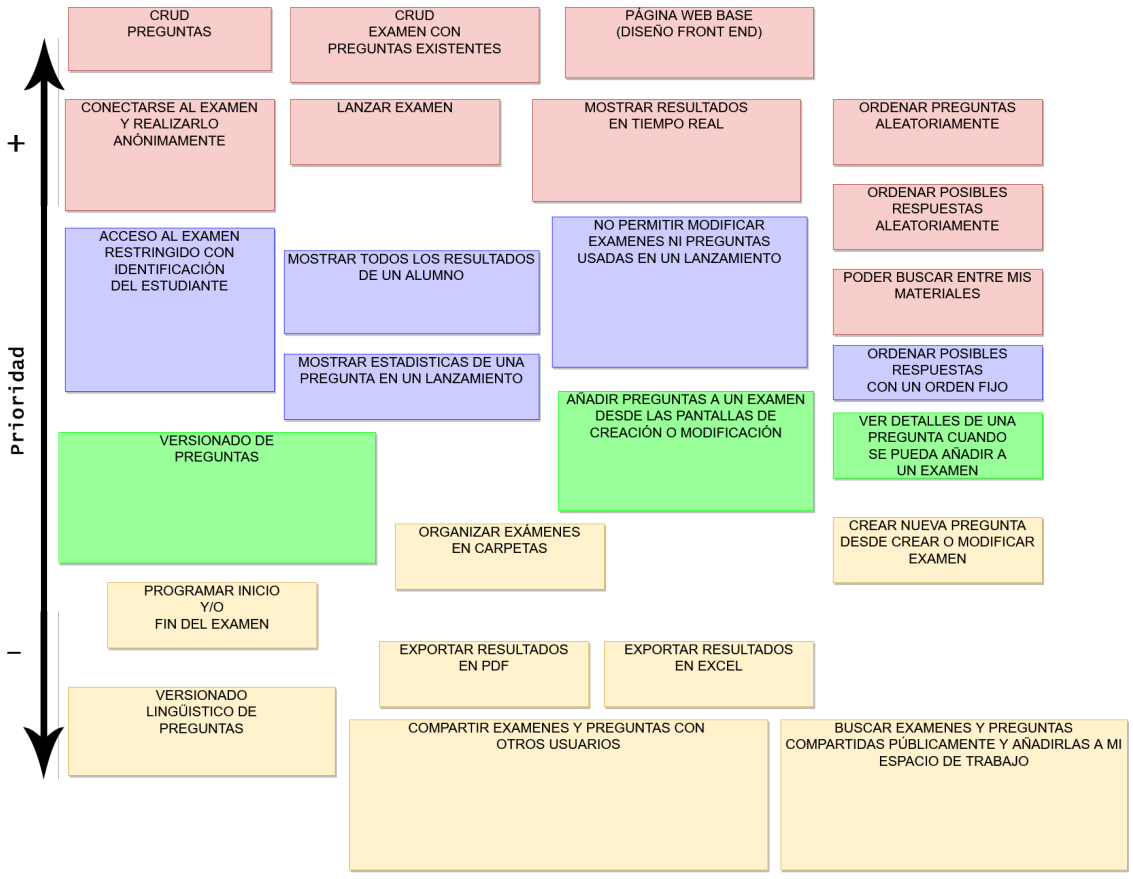

Figura 3.7: Mapa de características del segundo MVP

Para la primera de estas entregas se ha acordado realizar las tareas marcadas en azul, de forma que se obtenga una versión desde la cual se evite que la base de datos que almacena los resultados pueda quedar inconsistente, para ello se impedirá que quede restringida la modificación y eliminación de exámenes o preguntas que hayan sido utilizadas, se permitirá crear duplicados que si sean modificables de estos elementos y además se mostrarán en nuevas pantallas estadísticas relacionadas con los lanzamientos realizados.

Esto sumado con otras tareas menores permitirá obtener una versión en la que no solo se mejorarán las características del primer MVP gracias al *feedback* recibido, sino que además se empezará a hacer un uso más amplio de los resultados en cada lanzamiento.

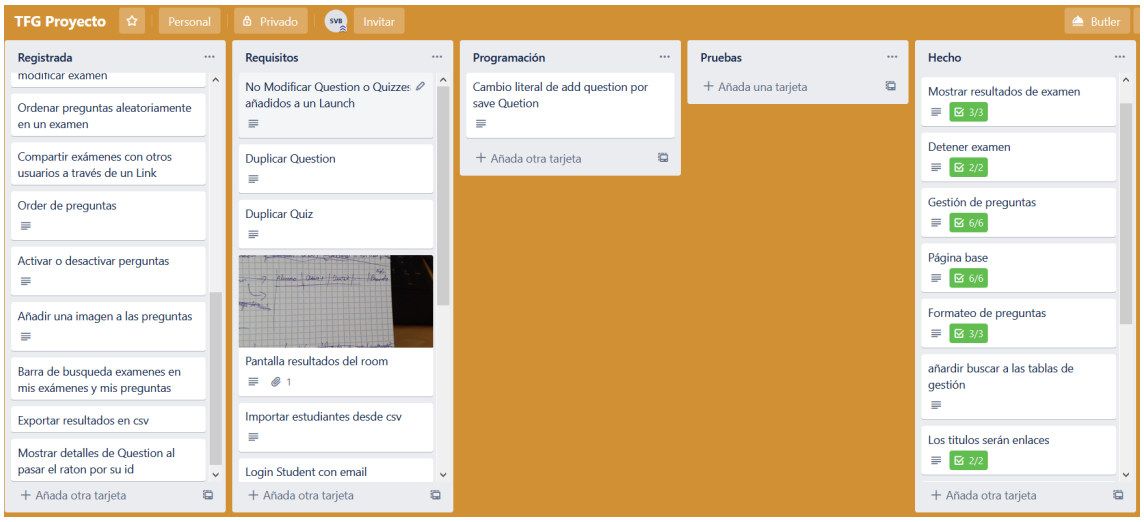

Figura 3.8: Tablero *kanban* al inicio del segundo MVP

Tras la reunión en la que se decide este nuevo alcance, de nuevo las característica se dividen en UT para añadirse al *backlog* si no lo estaban ya como se muestra en la figura 3.8, se cumplimentas sus requisitos junto con sus pruebas de aceptación como se describe a 3.3 y tras esto se inicia su programación.

- **No Modificar *question* o *quizzes* añadidos a un Launch**: No se permite eliminar o modificar *questions* ni *quizzes* si están en un launch.

  Pruebas de aceptación:

  - Tras lanzar un *quiz* este no podrá ser modificado ni eliminado.
  - Tras lanzar un *quiz* sus preguntas no podrán ser modificadas ni eliminadas.

- **Duplicar *Question***: Botón en el listado de *questions* que permita duplicar una pregunta.

  Pruebas de aceptación:

  - Al duplicar una pregunta se crea una nueva con los mismos atributos y respuestas excepto por el id.
  - El botón e mostrará en el listado de *MyQuestions*.

- **Duplicar *Quiz***: Botón en el listado de *quizzes* que permita duplicar un *quiz*.

  Pruebas de aceptación:

  - Al duplicar un *quiz* se crea uno nuevo con los mismos atributos y *questions* excepto por el id.
  - El botón e mostrará en el listado de *MyQuizzes*.

- **Pantalla resultados del *room*** : Crear una página para mostrar estadísticas de resultados obtenidos en los *quizzes* realizados en un *room* a partir de los datos de Firebase.

  Pruebas de aceptación:

  - La tabla que muestre los resultados tendrá en una columna los nombres de los alumnos y una columna por cada *quiz* realizado en ese *room* .

- **Importar estudiantes desde csv**: Se pide que en la pantalla de crear/modificar *room* se puedan añadir alumnos importando sus datos desde un fichero con formato CSV.

  Pruebas de aceptación:

  - Comprobar que el csv está bien formateado.
  - En caso de no estar bien formado el csv mostrar el error en un modal emergente.
  - Si existen alumnos duplicados añadirlos solo una vez.
  - Al reimportar solo añadir los alumnos que no estaban ya añadidos a la lista.

- **Los estudiantes entran con Código + email o nombre**: En la pantalla de iniciar *quiz* se pide un código y un email o nombre.

  Pruebas de aceptación:

  - Comprobar que el código introducido existe.
  - Si el *quiz* es privado el email es obligatorio y el nombre opcional.
  - Si el *quiz* es público el nombre es obligatorio y el email opcional.

- **Gestión de *students* desde *room***: Permitir eliminar y añadir *students* desde la pantalla de creación/modificación de un *room* .

  Pruebas de aceptación:

  - Se pedirán los datos: ID, Nombre, Apellidos, email.
  - Se podrá eliminar *students* ya añadidos.
  - Si el *student* ya está en el *room* no se añadirá.

- **Orden de las *Answers***: Se pide que las posibles respuestas a una pregunta se muestren siempre en el mismo orden para así poder definir respuestas como «Todas las anteriores».

  Pruebas de aceptación:

  - Las posibles respuestas se muestran siempre en el mismo orden en la pantalla de crear/codificar preguntas.
  - Las posibles respuestas se muestran siempre en el mismo orden al responder un *quiz*, y este orden es el mismo que se define en la pregunta.

- **Mostrar resultados de todos los *quiz* realizados en un *room* por *student***: Modificar la página «resultados de *room*» para mostrar resultados de todos los *quizzes* ejecutados en un *room* por cada alumno desde los resultados de la base de datos MySQL.

  Pruebas de aceptación:

  - La tabla que muestre los resultados tendrá en una columna los nombres de los alumnos y una columna con la nota de cada *quiz* realizado en ese room.
  - La última columna de la tabla mostrará la media aritmética de las notas obtenidas en los *quizzes* realizados en ese *room*.

- **Ordenar *questions* por id Inverso**: Se pide que se ordene por defecto las *questions* de la pantalla *MyQuestions* por id.

  Pruebas de aceptación:

- Las *questions* de la pantalla *MyQuestions* están ordenadas por defecto por id.

- **Añadir al *room* fecha de creación**: Al crear un nuevo *room* almacenar su fecha de creación.

  Pruebas de aceptación:

  - Mostrar la fecha de creación en el listado de *rooms* del usuario activo.
  - Mostrar la fecha de creación en la pantalla de resultados.

- **Cambio literal de *add question* por *save Quetion***: Cambiar texto del botón add *question* a save Quetion.

  Pruebas de aceptación:

  - El botón de guardado de la pantalla Crear/modificar pregunta, tiene el texto: *save Quetion*.

- **Poner *Quiz* y *Question* alterable false al lanzar un *quiz***: Añadir una variable para indicar si un *quiz* o *question* se puede modificar o eliminar.

  Pruebas de aceptación:

  - Al lanzar un *quiz*, este y sus *questions* cambiarán su estado alterable a false.
  - Los elementos con estado alterable a false no podrán ser modificados o eliminados.

- **Ocultar/Mostrar resultado en viewQuestion**: Botón para ocultar y mostrar los resultados estadísticos de una pregunta y así poder hacer la revisión más participativa.

  Pruebas de aceptación:

  - En la pantalla viewQuestion se mostrará la solución de la pregunta y estadísticas de sus respuestas en el lanzamiento seleccionado.
  - Un botón mostrará u ocultará la solución de la pregunta y los datos estadísticos.

Tabla 3.3: Distribución de horas por tarea - tercera entrega

UT	Estimado	Invertido
Cambio literal de *add question* por *save Quetion*	0h	0h
No Modificar *questions* o *quizzes* añadidos a un Launch	3h	4h
Duplicar *question*	3h	2h
Duplicar *Quiz*	2h	0.5h
Orden de las *Answers*	2h	3h
Gestión de *students* desde *room*	5h	10h
Importar estudiantes desde csv	8h	7h
Mostrar resultados de todos los *quiz* realizados en un *room* por *student*	2h	5h
Pantalla resultados del *room*	5h	6h
Ordenar *questions* por id inverso	0h	0h
Añadir al *room* fecha de creación	1h	1h
Los estudiantes entran con Código + email o nombre	3h	3.5h
Poner *Quiz* y *Question* alterable false al lanzar un *quiz*	1h	2h
Ocultar/Mostrar resultado en *view Question*	2h	2h
Total	**37h**	**46h**

Después del desarrollo de estas UT, la versión resultante se validó con el *product owner*. En la revisión de esta tercera entrega se concluyó que eran necesarias algunas funcionalidades más para obtener un segundo MVP, por un lado era necesario agilizar el proceso por el cual se incluía una pregunta en un cuestionario y por otro lado también se debía implementar una de las características más distintivas de esta aplicación frente a la competencia, el sistema de versiones para preguntas. Por ello se seleccionaron las características en verde de la figura 3.7 y se procedió a su estimación indicada en la tabla 3.4, su especificación, y posteriormente su desarrollo.

- **Versionado de *Questions***: Realizar un versionado suave, en el cual se apunte a la versión anterior de la pregunta, a la primera y a la siguiente.

  Pruebas de aceptación:

  - Al modificar una pregunta que no sea alterable se muestra un modal que indica que no es posible modificar esa pregunta.
  - El modal que no permite modificar la pregunta ofrece cancelar la modificación, duplicar la pregunta o crear una nueva versión.
  - Al lanzar un examen se lanzará la última versión se sus preguntas.
  - Los resultados de un lanzamiento apuntarán a la versión respondida de cada pregunta.

- **Añadir *quiz* desde *addQuestion***: Se pide poder añadir una pregunta a un *quiz* desde la pantalla *addQuestion*.

  Pruebas de aceptación:

  - En *addQuestion* se muestran los *quizzes* pertenecientes al usuario activo.
  - Se pueden marcar y desmarcar *quizzes*.
  - Al guardar la pregunta esta se añadirá a los *quizzes* que estén seleccionados.
  - Cuando un *quiz* no es alterable se indica y no se le pueden añadir o quitar preguntas.

- **Añadir/quitar *question* en *quiz***: Es necesario poder añadir y quitar *questions* de un *quiz* desde la pantalla de crear/modificar *quiz*.

  Pruebas de aceptación:

  - Mostrar un modal que muestre todas las preguntas del usuario activo.
  - Se pueden añadir preguntas a un *quiz* desde este modal.
  - Se pueden eliminar preguntas de un *quiz* desde la pantalla addQuiz.

- **Mostrar detalle *questions* desde *addQuizz***: Desde la pantalla *addQuiz* se podrá abrir un modal que muestre los detalles de una pregunta seleccionada.

  Pruebas de aceptación:

  - Los títulos de las preguntas abren un modal que muestran su titulo y posibles respuestas.

Tabla 3.4: Distribución de horas por tarea - cuarta entrega

UT	Estimado	Invertido
Versionado de *questions*	21h	10h
Añadir *Quiz* desde *addQuestion*	5h	4h
Añadir/quitar *question* en *quiz*	5h	2h
Mostrar detalle *questions* desde *addQuiz*	2h	4h
Total	**33h**	**18h**

Tras esta entrega se propusieron algunos pequeños cambios y se detectaron varios errores en el comportamiento de la aplicación, es por eso que la figura 3.9, que representa el estado del tablero *kanban* tras esta entrega, se puede apreciar que algunas UT están marcadas con una etiqueta roja, esto indica que son pertenecientes a un error detectado en la versión actual desplegada en Preproducción, al ser este un entorno al que algunos usuarios tienen acceso para realizar pruebas de la aplicación, estas UT tienen una prioridad superior a las demás, de forma que se resuelvan primero y tras esto se produzca un despliegue con la solución a estos errores para evitar en la medida de lo posible que los usuarios experimenten errores en este entorno. Por lo tanto esta etiqueta roja se usa para simular que esas UT se encuentran en un carril de emergencia.

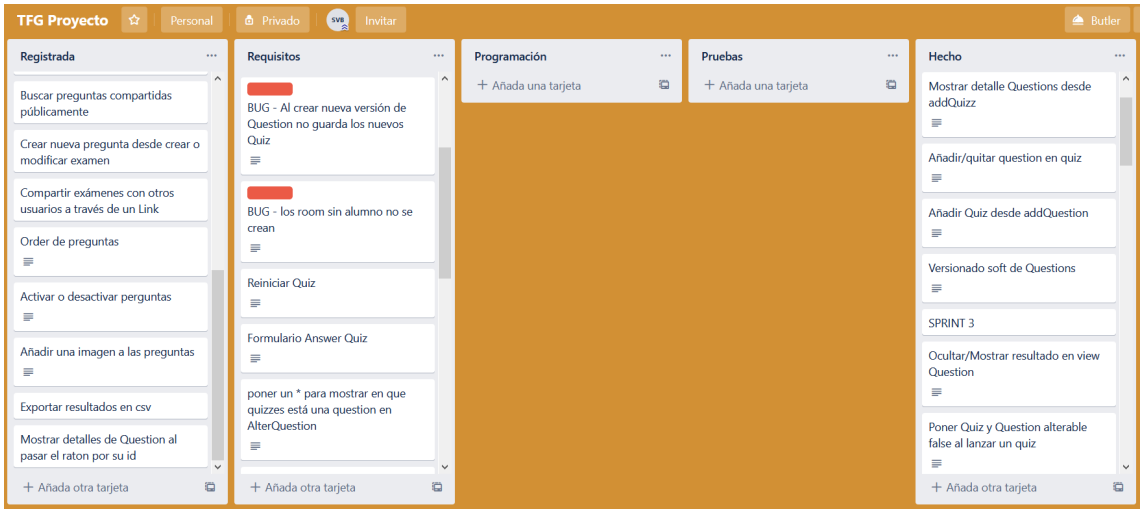

Figura 3.9: Tablero *kanban* al inicio del primer MVP

Como en el resto de entregas se puede encontrar su estimación y tiempo invertido en la tabla 3.5, y sus especificaciones son las siguientes:

- **BUG no permite al entrar a un *quiz* público sin email**: Se ha detectado un error que impide realizar un *quiz* público sin introducir un email.

  Pruebas de aceptación:

  - Se puede acceder a realizar un *quiz* público con el campo email como opcional.

- **Los *room* sin alumno no se crean**: Detectado *bug* que no permite crear un *room* sin alumnos.

  Pruebas de aceptación:

  - Se puede crear un *room* sin alumnos.

- **Al crear nueva versión de *question* no guarda los nuevos *Quiz***: Un *bug* hace que no se inserte una nueva versión de una pregunta en los *quiz* seleccionados en los que no estuviera su versión previa.

  Pruebas de aceptación:

  - Al crear una nueva versión de una pregunta, esta se inserta en los *quizzes* seleccionados.

- **Reiniciar *Quiz***: Añadir un botón para reiniciar un *quiz* parado en «mis lanzamientos».

  Pruebas de aceptación:

  - Cuando se pare un lanzamiento se cambia el botón «parar» por el botón «reiniciar».

- **Formulario *Answer Quiz* Mostrar nombre o email**: Al entrar en la pantalla para iniciar un *quiz* introducir primero el código y dependiendo de si es un lanzamiento público o no pedir solo el email o el nombre.

  Pruebas de aceptación:

  - En la pantalla para iniciar *quiz* en un principio solo se pide el código del lanzamiento.
  - Tras introducir el código aparece el campo email si es un lanzamiento privado.
  - Tras introducir el código aparece el campo nombre si es un lanzamiento público.

- **Mostrar parpadeo de botón *live* solo cuando haya *quizzes* en marcha**: Cuando un lanzamiento está en marcha el punto rojo de resultados debe parpadear.

  Pruebas de aceptación:

  - Si hay lanzamientos en marcha el punto rojo de resultados parpadea.
  - Si no hay lanzamientos en marcha el punto rojo de resultados permanece gris.

- **Aclarar que el CSV de alumnos debe tener cabecera**: Añadir texto indicando que el csv de añadir alumnos debe llevar una cabecera.

  Pruebas de aceptación:

  - Un texto indica que el csv de añadir alumnos debe tener la cabecera: *Surname, Name, ID, email*.

- ***BUG* no permite entrar a realizar un *quiz***: Por alguna razón no se puede entrar a ningún *quiz*.

  Pruebas de aceptación:

  - Se puede entrar a responder los *quizzes* lanzados.

- ***MyLaunches* añadir columna público/privado**: En la pantalla *MyLaunches* se muestra una columna indicando si el lanzamiento es público o privado.

  Pruebas de aceptación:

  - En la pantalla *MyLaunches* se muestra una columna indicando si el lanzamiento es público o privado.

- **Poner un * para mostrar en que *quizzes* está una *question* en *AlterQuestion***: En la pantalla *alterQuestion* los *quizzes* no alterables deben estar indicados con un par de asteriscos.

  Pruebas de aceptación:

  - Los *quizzes* no alterables deben estar indicados con un par de asteriscos.

- **Quitar botón *schedule Quiz***: Eliminar botón *Schedule Quiz* de la pantalla *dashboard*.

  Pruebas de aceptación:

  - El botón *Schedule Quiz* no aparece en la pantalla *dashboard*.

- **Menú usuario - dejar solo perfil y *log out***: Eliminar las opciones no implementadas de este menú.

  Pruebas de aceptación:

  - El menú usuario solo muestra las opciones: perfil y *log out*.

- **Texto del botón *Start quiz* a *Answer Quiz***: Cambio del literal del botón *Start quiz* a *Answer Quiz*.

  Pruebas de aceptación:

  - El botón *Start quiz* ahora muestra el literal *Answer Quiz*.

- **Quitar literales *Action* y *close date* de results**: No se seguirán mostrando estos textos en la pantalla *results*.

  Pruebas de aceptación:

  - No se muestran los textos *Action* y *close date* en la pantalla resultados.

Tabla 3.5: Distribución de horas por tarea - quinta entrega

UT	Estimado	Invertido
BUG no permite al entrar a un *quiz* público sin email	1h	1h
Los *room* sin alumno no se crean	1h	0.5h
Al crear nueva versión de *question* no guarda los nuevos *quizzes*	1h	1h
Reiniciar *quiz*	1h	1h
Formulario *Answer Quiz*, Mostrar nombre o email	1h	3h
Mostrar parpadeo de botón *live* solo cuando haya *quizzes* en marcha	1h	2h
Aclarar que el CSV de alumnos debe tener cabecera	1h	1h
Conjunto de tareas marcadas con coste bajo ▪ BUG no permite entrar a realizar un *quiz* ▪ *MyLaunches* añadir columna público/privado ▪ Poner un * para mostrar en que *quizzes* está una *question* en *AlterQuestion* ▪ Quitar botón *schedule Quiz* ▪ Menú usuario - dejar solo perfil y *log out* ▪ Texto del botón *Start quiz* a *Answer Quiz* ▪ Quitar literales *Action* y *close date* de *results*	2h	1h
Total	**9h**	**10.5h**

Finalmente, una vez completadas estas tareas se volvió a producir otra entrega en la que se detectaron nuevos errores y se aceptaron dos pequeños cambios solicitados por el *product owner*. En la tabla 3.6 se detalla la estimación e inversión de tiempo en estos cambios y a continuación sus especificaciones:

- **Error al añadir *students* en un *room***: Un usuario ha reportado: «No he conseguido guardar los alumnos de un *room*, ni introduciéndolos uno a uno ni cargándolos desde fichero. Esto es clave porque lo usaré con una lista cerrada de alumnos».

  Pruebas de aceptación:

  - Se puede añadir alumnos a un *room* sin problemas.

- **Error al añadir *questions* a *quiz***: El *product owner* ha reportado: «Al crear un *quiz* muestra repetidas las preguntas que se añaden desde el formulario para añadir pero luego al actualizar y volver a entrar al *quiz* ya sale bien. También al cambiar el número de preguntas mostradas en el formulario para añadir repite las preguntas que ya estaban añadidas, al salir y volver a entrar al *quiz* ya salen bien.».

  Pruebas de aceptación:

  - No se muestran repetidas las preguntas recién añadidas a un *quiz*.

- **Terminar un *quiz* sin haber respondido a la última pregunta**: El *product owner* ha reportado: «No debería dejar terminar el *quiz* dejando la última pregunta vacía, ahora el botón *End quiz* lo permite.».

Pruebas de aceptación:

- No se puede terminar un *quiz* sin haber respondido a la última pregunta.

- **Error en el mensaje mostrado al tratar de eliminar un *Quiz***: Un usuario reporta: «Al intentar eliminar un *quiz* que tenía lanzamiento sale mal el mensaje de advertencia diciendo que no se puede modificar y que puede duplicarse.».

  Pruebas de aceptación:

  - Al intentar eliminar un *quiz* no modificable se muestra un mensaje que advierte que no se puede eliminar.

- **Orden de *questions* en la pantalla resultados**: Un usuario señala: «Cada vez que se regresa a la pantalla de resultados el orden de las preguntas en los *quizzes* realizados es diferente, debería conservarse el orden».

  Pruebas de aceptación:

  - En la pantalla de resultados el orden de las preguntas se mantiene igual siembre.

- **Reducir los dígitos del código de Lanzamiento**: El *product owner* pide: «Lo que me ha resultado incómodo para probar hacer los *quizzes* y en particular en el teléfono es introducir el código del *quiz*, es muy largo. ¿Podría ser un código solo de 4 letras mayúsculas y/o dígitos?».

  Pruebas de aceptación:

  - El código de un lanzamiento tiene una longitud de 4 dígitos.

Tabla 3.6: Distribución de horas por tarea - sexta entrega

UT	Estimado	Invertido
Error al añadir *Students* en un *room*	2h	0.5h
Error al añadir *questions* a *quiz*	1h	0h
Terminar un *quiz* sin haber respondido a la última pregunta	1h	1h
Error en el mensaje mostrado al tratar de eliminar un *quiz*	1h	0h
Orden de *questions* en la pantalla resultados	1h	0h
Reducir los dígitos del código de Lanzamiento	0h	0h
Total	**9h**	**11h**

3.2.1. Experimento 2

Para este segundo experimento se buscó poner a prueba esta herramienta con un uso más masivo, en lugar de realizar un solo cuestionario, esta vez la propuesta fue que se pudiera utilizar para realizar un seguimiento compuesto de varias pruebas a un grupo de alumnos.

Esto se pudo llevar a cabo en un grupo de 54 alumnos a través de 8 cuestionarios realizados entre el 2 de Julio y el 5 de Agosto de 2020. Este experimento se centraba en comprobar la satisfacción del profesor que lo lleva a cabo en cuanto a la funcionalidad añadida desde el primer experimento y puntos a favor sobre otras herramientas disponibles en el mercado. También se esperaba en este experimento poner recoger información sobre el uso de datos en el servicio de Firebase RealTimeDB para poder realizar una previsión del uso que se podría dar con las cantidades de usuarios previstas.

Los datos recogidos durante este experimento se muestran en las figuras 3.10, 3.11, 3.12, 3.13, 3.14 y 3.15. De estos datos se puede extraer que con alrededor de 50 alumnos y 8 cuestionarios con entre 5 y 15 preguntas cada uno, se han descargado 143MB con picos de 19MB a 30MB, se han producido picos de 10 a 50 conexiones simultáneas y se requirió hasta 421KB del almacenamiento en la base de datos no relacional. Por otra parte, la base de datos relacional empleada ha visto aumentado su tamaño de 528KB a 928KB durante este experimento, con lo cual se puede esperar un uso de alrededor de 400KB de almacenamiento para este tipo de cursos. Esta información resulta relevante en la estimación de requisitos necesarios en función de los usuario estimados.

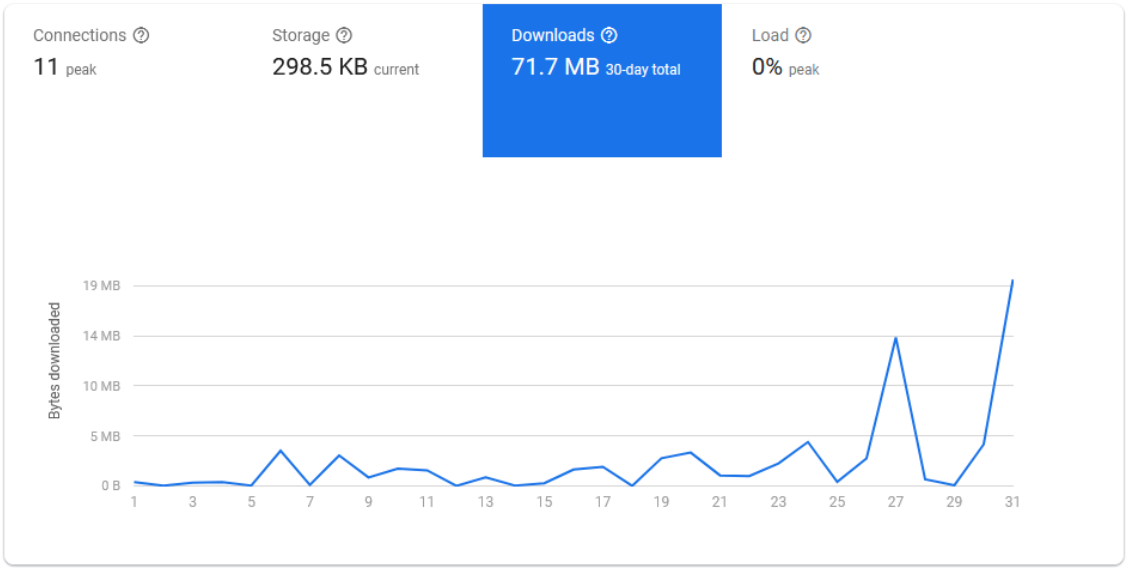

Figura 3.10: Datos descargados durante el segundo experimento en Julio

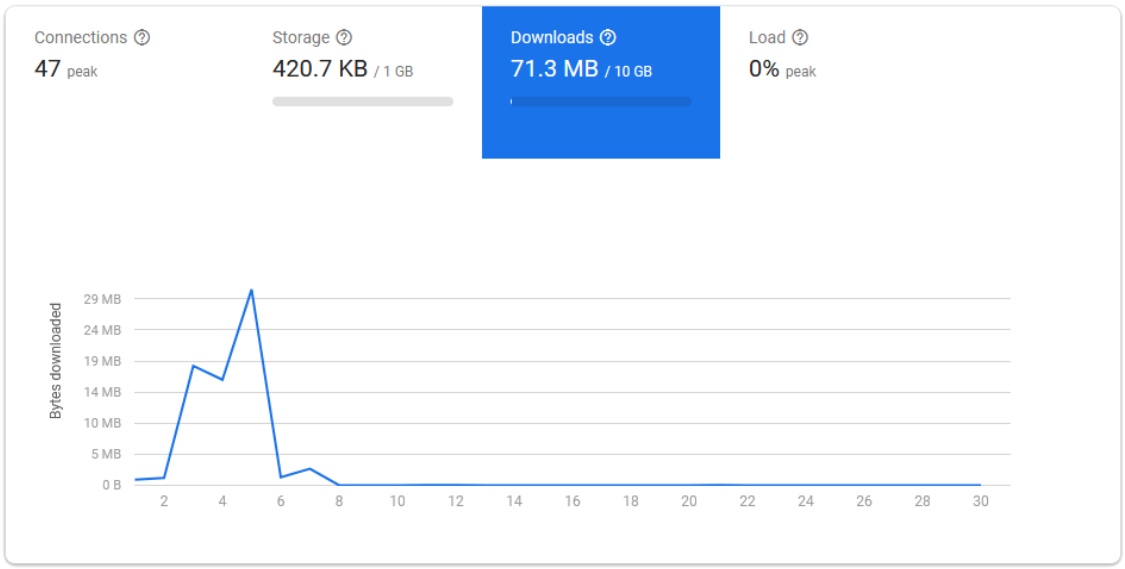

Figura 3.11: Datos descargados durante el segundo experimento en Agosto

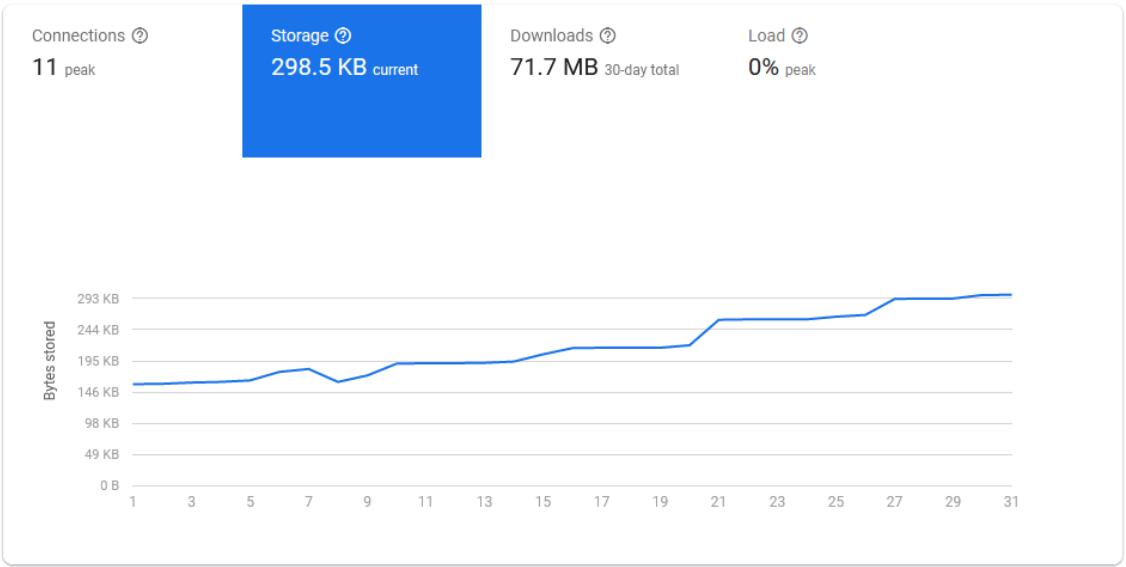

Figura 3.12: Almacenamiento durante el segundo experimento en Julio

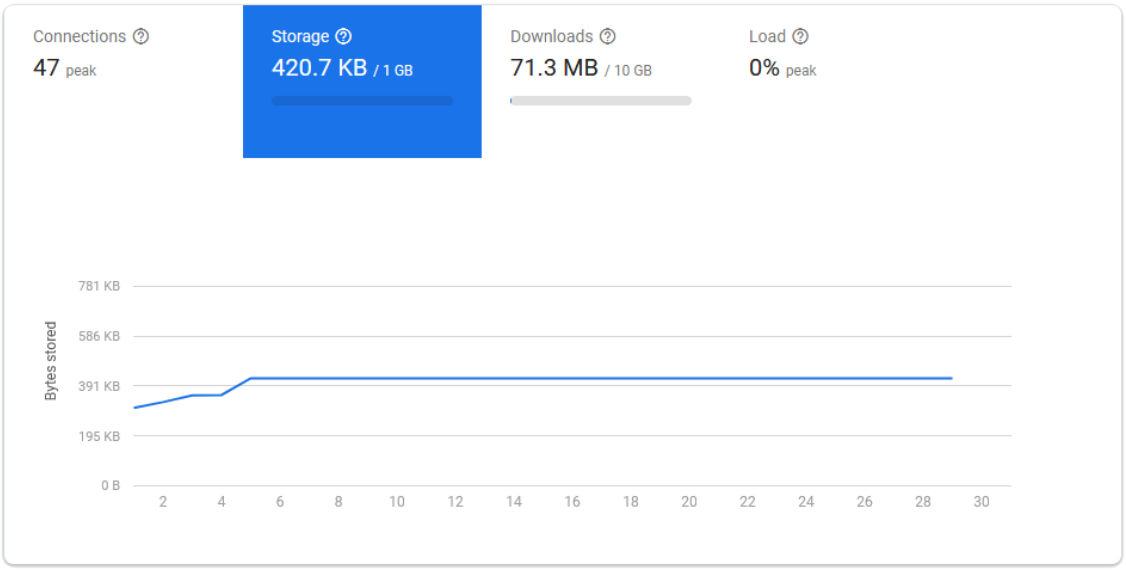

Figura 3.13: Almacenamiento durante el segundo experimento en Agosto

Figura 3.14: Conexiones durante el segundo experimento en Julio

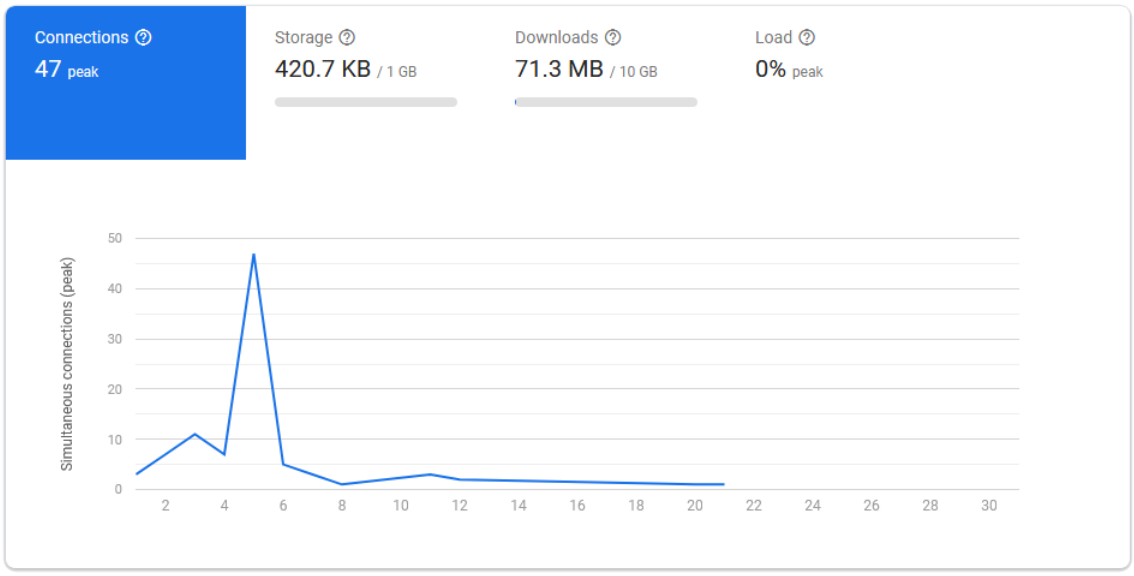

Figura 3.15: Conexiones durante el segundo experimento en Agosto

En la retrospectiva de este experimento, el profesor que formó parte de él señaló que valora muy positivamente frente a otras aplicaciones los siguientes puntos:

- Las preguntas se gestionen de forma separada a los exámenes.
- En concreto frente a Socrative: es muy útil poder tener *quizzes* preparados en un *room*.
- Está abierta a otros usuarios ajenos a una organización.

Este experimento ha probado la validez de las funcionalidades implementadas tras la retrospectiva del primer MVP ya que estas han sido utilizadas durante este experimento sin problemas y agilizando tareas antes más complejas. Además observando las opiniones de este profesor se puede extraer que a día de hoy este proyecto aporta unas claras ventajas en sus procesos más básicos frente a otras herramientas existentes que son apreciables ya en este segundo MVP, con lo que se ha seguido la dirección adecuada hasta ahora. Por otra parte también se han obtenido propuestas de mejora que se tendrán en cuenta para posteriores desarrollos:

- Poder reconectar con un *quiz* que se ha dejado a medias tras una pérdida de conexión.
- Poder crear equipos entre los alumnos de un *room*.
- Poder borrar lanzamientos.
- Poder añadir etiquetas a preguntas y cuestionarios para posteriormente buscarlos filtrando por estas etiquetas.

CAPÍTULO 4
Aspectos técnicos

4.1 Herramientas utilizadas

Las principales herramientas utilizadas para el desarrollo de este proyecto han sido seleccionadas en base al ahorro monetario, eligiendo para ello software con licencia que permitan su uso de forma gratuita, y tras esto el criterio ha sido la preferencia personal del equipo de desarrollo.

Empezando por el sistema de gestión de bases de datos, se ha utilizado como sistema principal MySQL[1], actualmente se encuentra bajo licencia dual (pública general y comercial), pero debido al uso que se le ha dado, este trabajo solo se vería afectado por la licencia pública general que en este caso es concretamente *GNU GPL v2* [8]. Para conectar con MySQL y gestionar bases de datos con un entorno gráfico se ha elegido HeidiSQL[2], que opera también bajo licencia *GNU GPL*. A parte de estas herramientas para el manejo de datos también se hace uso de dos servicios de DBaaS (*Database as a Service*) que proporcionarán bases de datos en la nube, los proveedores de estos servicios son JawsDB[3] y Firebase[4], este último no es un proveedor de DBaaS específicamente ya que su oferta es más amplia pero para este proyecto se va a usar como tal.

Los lenguajes de desarrollo se ha elegido Java 8 para la parte *back-end*[5] y JSP con JavaScript para el *front-end*[6]. Para esta elección se ha tenido en cuenta los conocimientos del equipo de desarrollo en estos lenguajes para minimizar la curva de aprendizaje inicial.

En lo referente al despliegue de la aplicación, de forma local se usa Apache Tomcat[7] que se adquiere bajo licencia *Apache 2.0* [9]. Por otra parte para los despliegues en la nube se estudió utilizar AWS (*Amazon Web Services*), pero a pesar de ser una buena opción desde un punto de vista tanto técnico como económico requería una inversión de tiempo en configuraciones del entorno a tener en cuenta, por lo tanto se buscaron opciones que pudieran permitir agilizar este proceso, para eso se eligió Heroku[8] que es un proveedor PaaS (*Platform as a service*) el cual reduce toda esta configuración a simplemente elegir algunas opciones básicas y añadir o quitar servicios de tu espacio de una forma sencilla.

[1] MySQL: https://www.mysql.com/
[2] HeidiSQL: https://www.heidisql.com/
[3] JawsDB: https://www.jawsdb.com/
[4] Firebase: https://firebase.google.com/
[5] *back-end* se refiere a la parte oculta para los usuarios de la aplicación y que lleva a cabo el procesado de los datos que introducen los usuarios desde el *front-end*.
[6] *front-end* hace referencia a la parte de la aplicación con la que lo usuarios van a interactuar directamente.
[7] Apache Tomcat: http://tomcat.apache.org/
[8] Heroku: https://www.heroku.com/

Como repositorio de versiones se ha elegido Git[9] que se obtiene con licencia *GNU GPL v2* y se ha elegido Bitbucket[10] como servicio de alojamiento frente a Github debido a oferta de repositorios privados y su integración con Trello[11] y otros productos de Altissan que puedan ser incorporados a este proyecto en el futuro.

Finalmente en cuanto al IDE (*Integrated Development Environment*), no se ha definido uno concreto para el proyecto dejando esto a la elección personal de cada desarrollador para que use el de su preferencia. Por el momento solo hay un programador en el proyecto y este ha elegido usar NetBeans[12] que dispone de licencia *Apache 2.0*.

4.2 Entornos de desarrollo

Se han definido dos entornos claramente diferenciados para esta aplicación:

- **Desarrollo**: está orientado al trabajo en local, el cual permitirá acceder a base de datos y realizar despliegues en la propia máquina del desarrollador. Para esto se ha instalado en el ordenador destinado al desarrollo un servidor MySQL y un Tomcat.
- **Preproducción**: este simula mejor el entorno en que se encontraría la aplicación cuando se publique y pueda ser usada por cualquier usuario. Para generar este entorno de ha usado Heroku, en esta página se ha configurado un servidor para aplicaciones java web, para ello se ha seleccionado un *Buildpack* ya definido en Heroku llamado *heroku/jvm* destinado a la compilación de aplicaciones java, además se ha añadido uno de los add-on gratuitos concretamente el de JawsDB MySQL para disponer de una base de datos en la nube.

El despliegue en cada entorno se realiza de forma distinta. El primer paso para el despliegue es compilar el proyecto, esto da como resultado un archivo con la extensión *war*, en el caso del entorno *Desarrollo* se puede integrar el servidor *Tomcat* instalado en el IDE que se esté usando y automatizar el despliegue de esta forma simplemente seleccionando *Run* se ejecutará la aplicación.

Por otra parte el entorno *Preproducción* necesita subir este archivo *war* a Heroku para que se pueda desplegar en la nube. Con este objetivo se hace uso de *Heroku CLI* [15], una herramienta creada por Heroku y distribuida con licencia *ISC* [10]. Una vez instalado *Heroku CLI* con motivo de automatizar los despliegues de ha desarrollado un *script batch* que lleve a cabo esta tarea:

```
@echo off
CD .\teachersquiz\dist
heroku war:deploy teachersquiz.war --app teachersquiz
pause
exit
```

Figura 4.1: Script para el despliegue de la aplicación en Heroku

[9] Git: https://git-scm.com/
[10] Bitbucket: https://bitbucket.org
[11] Trello: https://trello.com
[12] NetBeans: https://netbeans.org/

La figura 4.2 ilustra salida del *script* que muestra el proceso de despliegue por consola:

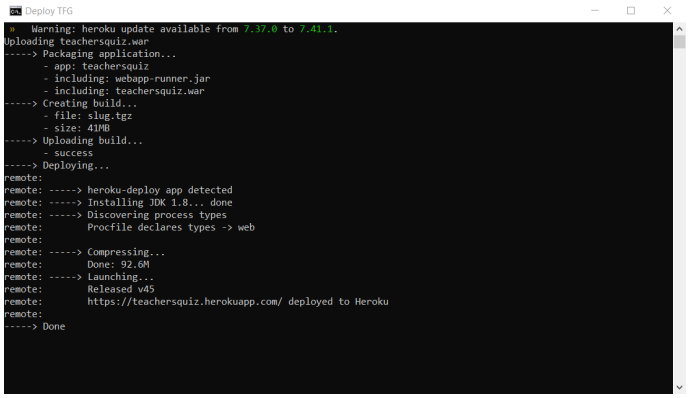

Figura 4.2: Salida del *script* de despliegue

4.3 Modelo de datos

En este proyecto se ha seguido los principios propuestos por el manifiesto ágil con lo cual se ha priorizado el desarrollo de software funcionando correctamente frente a la documentación extensiva, pero esto no quiere decir que esto segundo se haya dejado de lado, sino que se ha documentado solo lo que se ha considerado necesario o que aporta una ventaja grande a corto plazo. Los ejemplos más claros de esto son los modelos de datos, los cuales se han utilizado para tener una visión clara de la estructura de almacenamiento y el flujo de datos, esto ha ayudado a plantear ampliaciones de estos modelos cuando han sido necesarias u optimizaciones.

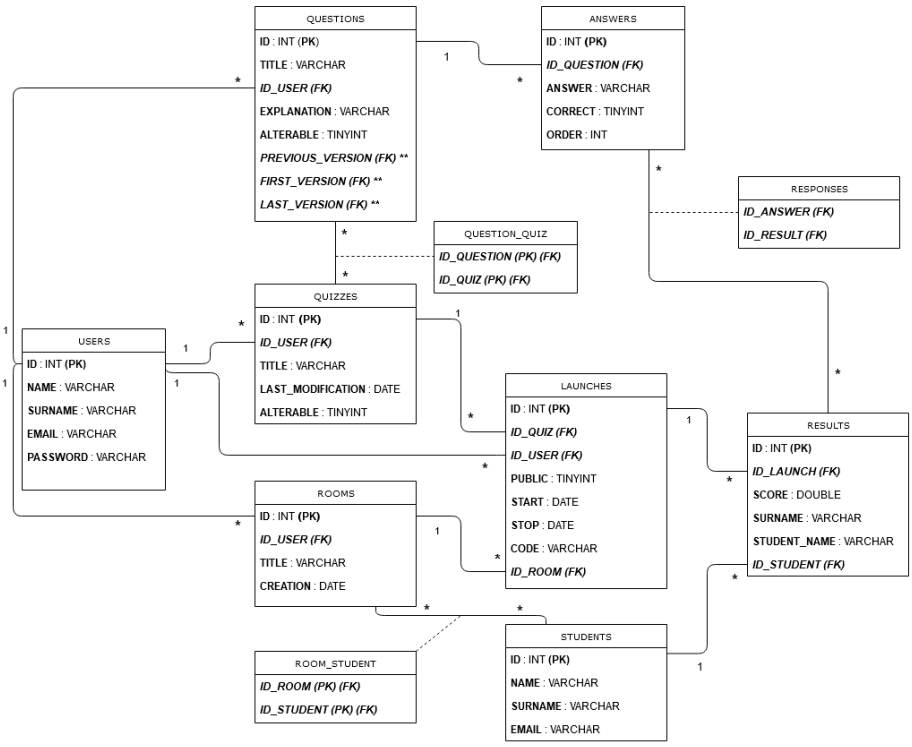

Figura 4.3: Modelo de datos en la base de datos *MySQL*

En la figura 4.3 se observa el modelo relacional ideado para trabajar como almacén de datos. Concretamente en la tabla *QUESTIONS* existen tres campos referentes a versiones de una misma pregunta que están marcados con **, a través de esa marca se ha querido simbolizar que esas claves ajenas apuntan a la clave primaria de esa misma tabla, se ha optado por esa notación para no enturbiar la visibilidad.

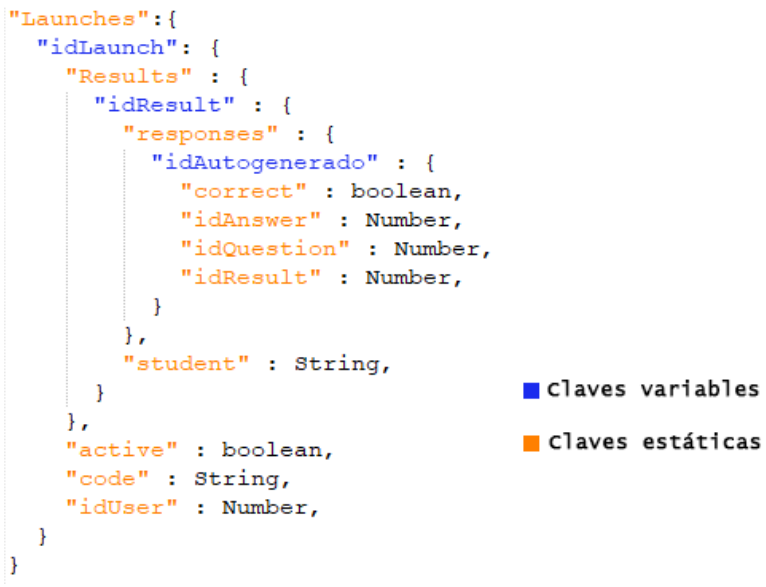

Figura 4.4: Modelo de datos en la base de datos *Firebase Realtime Database*

Por último en la figura 4.4 muestra como se han estructurado los datos en un almacén basado en objetos JSON y que será utilizado como memoria temporal para los datos que requieran un acceso rápido durante el lanzamiento de un *quiz*.

4.4 Arquitectura

En el desarrollo de la aplicación se ha seguido como patrón principal de arquitectura: Modelo vista controlador (MVC). Este patrón se basa en separar por completo la lógica de la aplicación, de la interfaz de usuario y conectar estos conceptos a través de un canal de comunicación predefinido, de forma que si una de estas partes cambia, el resto no necesitan ser alteradas.

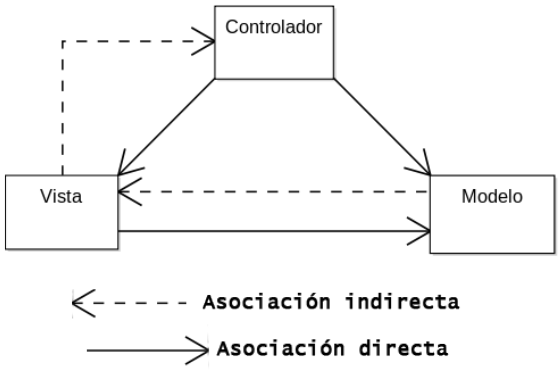

Figura 4.5: Modelo MVC [19]

Esto se realiza de la manera que se muestra en la figura 4.5. La parte llamada *Vista* hace referencia a la interfaz de usuario, la cual se comunica directamente con el *Controlador* enviándole o solicitando datos, este intermediario le pasa la solicitud o los nuevos datos al *Modelo*, el cual contiene la lógica de negocio de la aplicación, en caso de que se hayan solicitado datos los recupera de la base de datos, y si por el contrario se han enviado nuevos datos los envía para su almacenamiento. Mientras la estructura con la que estos datos se mueven entre los distintos componentes no cambie, se puede cambiar por completo una de estas siendo este proceso transparente para el resto.

Por ello, este patrón es ideal para un desarrollo ágil con alta tolerancia a cambios como el que propone *Lean Startup*, pero no solo por eso, concretamente trabajando con aplicaciones web hay que tener una especial predisposición al cambio, ya que, como se comenta en *Head First Servlets and JSP* refiriéndose al patrón MVC:

"With web apps, it's really important, because you should never assume that your business logic will be accessed only from the web! We're sure you've worked in this business long enough to know the only guarantee in software development: ***the spec always changes****"*[13].

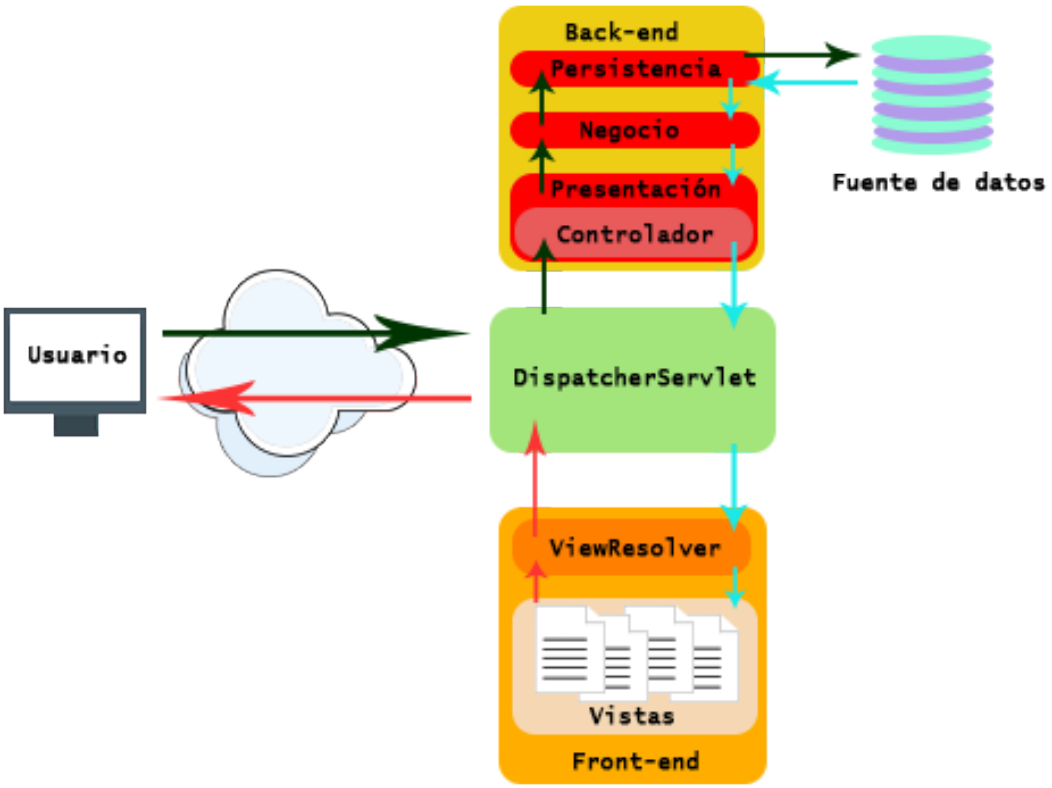

Figura 4.6: Arquitectura lógica

Para facilitar la implementación de la arquitectura MVC y otras funcionalidades se ha decidido utilizar *Spring Framework*[14] que está bajo licencia *Apache 2.0*. Este *framework* es uno de los más utilizados en java y ofrece una gran cantidad de funcionalidades, en este proyecto se han usado principalmente para realizar inyección de dependencias y su *framework* para desarrollo web con arquitectura MVC llamado *Spring MVC* [17]. *Spring* necesita la librería *Apache Commons Loggin*[15] que es distribuida bajo la misma licencia y también se añadirá al proyecto.

[13] Bryan Basham [16], extraído de la página 54
[14] Spring Framework: https://spring.io/projects/spring-framework
[15] Apache Commons Loggin: http://commons.apache.org/proper/commons-logging/

Con *Spring* implementando el modelo MVC en el proyecto, las peticiones desde el cliente se gestionarán como ilustra la figura 4.6.

Al realizar una petición el primero en actuar será el *DispatcherServlet* que la redirigirá al componente que tenga que tratar dicha petición. Por lo general esta petición será enviada al controlador el cual realizará el procesamiento oportuno en el *back-end*, y tras esto devolverá al *DispatcherServlet* el modelo y el nombre de la vista que se ha de retornar al usuario. A partir de aquí se pide al *ViewResolver* que genere la vista indicada, esta es devuelta al *DispatcherServlet*. Ya con la vista y el modelo adecuados se da la respuesta al usuario.

Además de MVC, dentro del *back-end*, se ha seguido una modelo de desarrollo en tres capas. Según esta arquitectura hay que separar el código en tres grupos claramente diferenciados o *capas*: Presentación, lógica de negocio y persistencia. Esto permite segmentar el código de forma que lo hace más mantenible añadiendo una separación adicional en el *Modelo* entre la lógica de negocio y el acceso a datos.

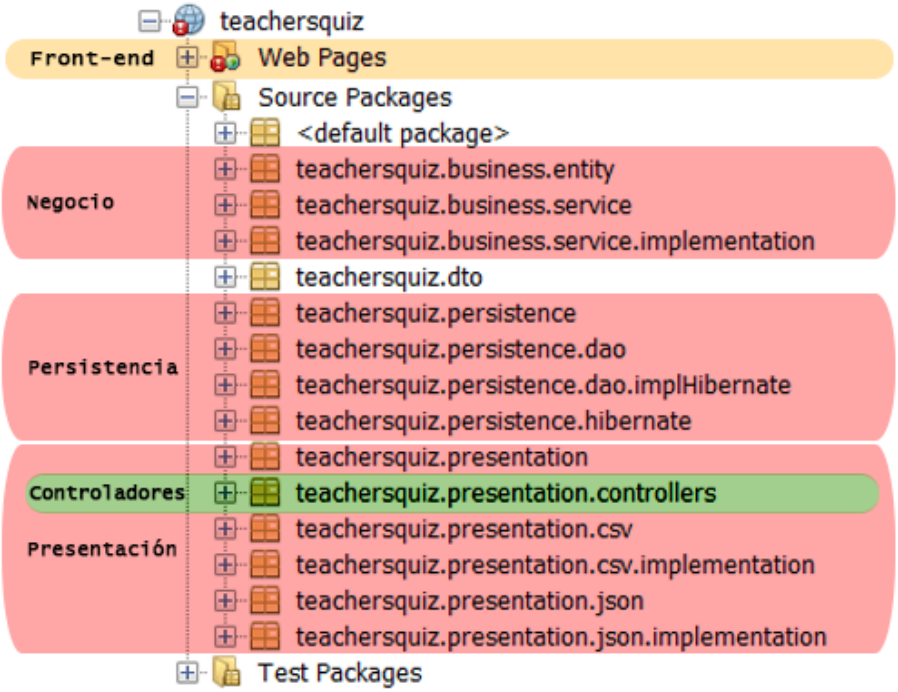

Figura 4.7: Arquitectura física

Se puede apreciar como funciona lógicamente en la parte *back-end* de la figura 4.6, y como se distribuyen los componentes físicamente en la figura 4.7.

4.5 Back-end

Una de las técnicas de programación usadas en *back-end* es ORM (*Object-Relational mapping*), esta facilita la interacción con base de datos, el objetivo es mapear las entidades de la aplicación para poder convertir estos objetos a tablas de una base de datos relacional, de esta forma se evita tener que escribir las consultas a base de datos en lenguaje SQL, esto se automatiza a través de herramientas que realizan la traducción (de entidades a tablas) y auto-generan las instrucciones necesarias en el lenguaje SQL que se esté empleando. Para este propósito se ha incorporado el *framework Hibernate*[16] distribuido con

[16]Hibernate: https://hibernate.org/

licencia *GNU LGPL* [11]. También se ha hecho uso de una librería complementaria para *Hibernate*, esta está destinada a validar que todos los atributos de las entidades cumplan una serie de requisitos antes de realizar una operación con la base de datos que pueda ser rechazada por estos motivos, la librería se llama *Hibernate Validator*[17] y se encuentra bajo licencia *Apache 2.0*. Junto con este *framework* también se necesita una librería de conexión con la base de datos, MySQL en este caso, esta librería es *MySQL Connector/J*[18], se usa con licencia *GNU GPL* [18].

Continuando con las herramientas de acceso a datos, con el objetivo de conseguir una completa independencia entre la lógica de negocio y el acceso a datos como propone la arquitectura en 3 capas, se ha implementado el patrón DAO *(Data Access Object)*. Para ello se ha realizado una interfaz genérica que proporciona los métodos principales que se usarán por todas las entidades, tras esto todas las clases DAO extenderán esta clase heredando así las dependencias necesarias para acceder a la fuente de datos y los métodos básicos para realizar un CRUD[19]. Esta estructura permite que cuando se necesite acceso a datos para una entidad simplemente se cree una interfaz DAO que herede de la genérica y defina los métodos que necesite esa entidad específicamente en caso que sea necesario alguno como se ilustra en la figura 4.8.

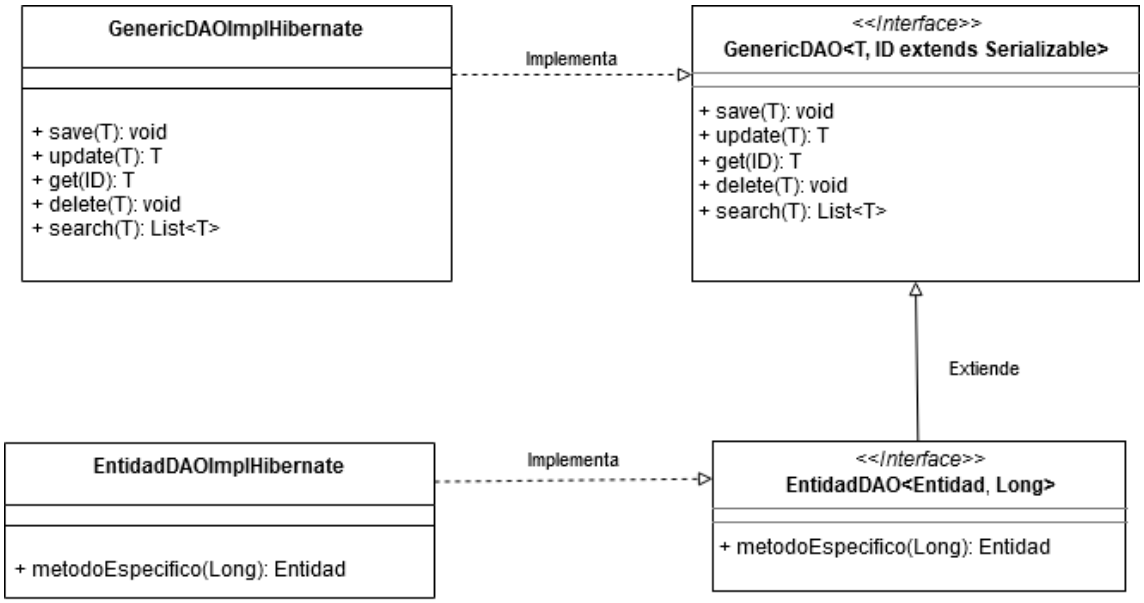

Figura 4.8: Estructura de los DAO

De esta forma en caso de querer cambiar la fuente de datos, simplemente habría que sustituir las implementaciones de estas clases, haciendo el proceso transparente para el resto de la aplicación.

Finalmente también se han usado otras librerías para realizar otras tareas más concretas, como *Jasypt*[20] para encriptar contraseñas en MD5, usada bajo licencia *Apache 2.0* y *Jackson*[21] que permite automatizar la transformación de objetos Java a JSON o CSV y también posee licencia *Apache 2.0*.

[17] Hibernate Validator: https://hibernate.org/validator/
[18] MySQL Connector/J: https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-overview.html
[19] CRUD (Create, Read, Update and Delete): hace referencia a las operaciones básicas de la capa de persistencia.
[20] Jasypt: http://www.jasypt.org/
[21] Jackson: https://github.com/FasterXML/jackson-docs

4.6 Front-end

Para construir la interfaz de usuario en el *front-end* se han empleado *Apache Tiles*[22] con la licencia *Apache 2.0*. Este *framework* es uno de los más sencillos de utilizar junto con el modelo MVC, por lo tanto es ideal para este proyecto. Su función es permitir realizar plantillas de las páginas a crear con los elementos más genéricos de estas, por ejemplo, la estructura principal de la página se va a mantener igual en casi toda la web, mientras que el menú principal, el pié de página y la cabecera son dos elementos que se podrán ver en casi todas las páginas de la aplicación pero que se podrían diseñar en archivos separados para que sean más fáciles de mantener, por último el contenido es lo que realmente va a cambiar dependiendo de a que página se esté accediendo. Por tanto podríamos crear una plantilla que contenga la estructura principal donde estos elementos irán posicionados, en otros archivos el menú principal, el pié de página y los diferentes contenidos que se puedan mostrar, y *Tiles* se encargará de montar todo en caliente como indica la figura 4.9.

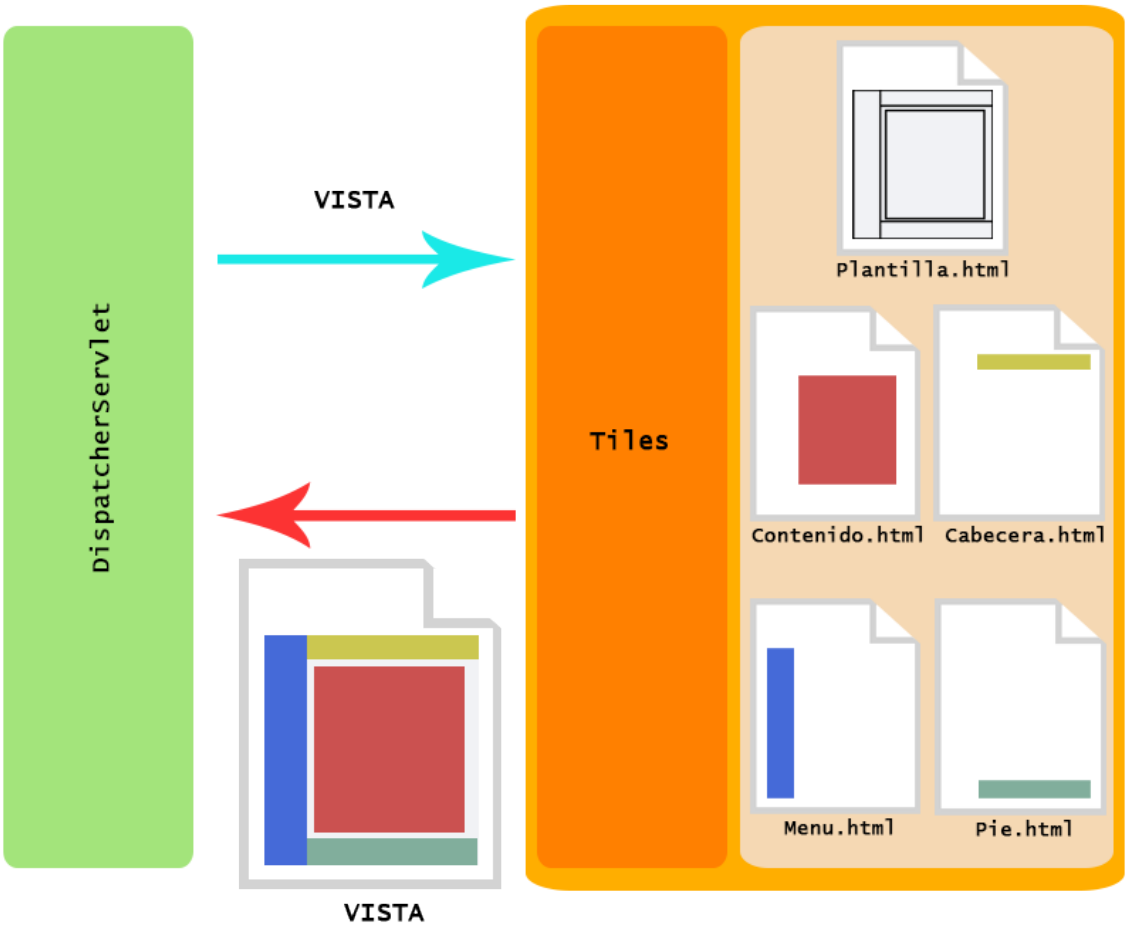

Figura 4.9: Construcción de la vista con *Tiles*

Por lo tanto *Tiles* realizará la función del *ViewResolver* vista en la figura 4.6 seleccionando partes de la vista pedida y devolviéndola montada.

En cuanto a la programación de las vistas, se ha utilizado HTML y CSS como lenguajes de diseño, el tratamiento de los datos del modelo realizado con JSP y se usa *JavaScript* para programar funcionalidades.

[22] Apache Tiles: http://tiles.apache.org/

Como diseño base desde el que partir, se ha utilizado una plantilla llamada *SB Admin 2* que ofrece *Start Bootstrap*[23] con licencia [12].

Para el uso más eficiente y ágil de *JavaScript* se utiliza el *framework JQuery*[24] obtenido bajo licencia *MIT*. Como complemento a este *framework* también se utiliza las librerías *JQuery Validation*[25] para facilitar las validaciones de los datos introducidos por los usuarios, *jsonpath*[26] con el objetivo de realizar consultas a un objeto JSON con un lenguaje similar a xpath, *sweetalert2*[27] usado para agilizar la creación de ventanas y mensajes emergentes, *DataTables*[28] que proporciona una manera fácil y rápida de crear tablas HTML con funcionalidades como paginado, ordenado o búsqueda ya implementadas de base, y por último *Chart.js*[29] para generación de gráficas de datos. Todas las librerías nombradas también tienen licencia *MIT*.

En cuanto al diseño, se ha hecho uso del *framework Bootstrap*[30] con licencia para facilitar la aplicación de un diseño *responsive* en la web para todos los componentes, de esta forma la aplicación se adaptará al tamaño de pantalla del dispositivo que se esté utilizando. También se usa una librería de iconos llamada *Font Awesome*[31] la cual tiene una versión gratuita que opera con distintas licencias, dependiendo de si se refiere a los iconos, fuentes o al código fuente, las cuales son por ese mismo orden: *CC BY 4.0* [13], *SIL OFL* [14], *MIT*.

4.7 Pruebas realizadas

Las pruebas que se han realizado en la aplicación se pueden dividir claramente en tres tipos: pruebas de caja blanca, de caja negra y pruebas de usabilidad.

Las de caja negra son las más sencillas y superficiales. En estas pruebas no se tiene en cuenta como trabaja internamente el método o flujo a probar, simplemente se introducen datos en él y se comprueba que la respuesta devuelta sea la esperada. Con esta técnica se prueban métodos que tengan poca probabilidad de fallar por su simplicidad y sobre todo pruebas de integración donde se prueba todo el flujo de una funcionalidad. Un ejemplo de estas pruebas es la actualización de un usuario de prueba en la base de datos, se ha definido un método que llama al servicio de usuarios, actualiza este usuario y comprueba que se ha guardado correctamente en la base de datos, tras esto lo vuelve a dejar como estaba.

Por otro lado también se han realizado algunas pruebas de caja blanca. Al contrario que las de caja negra en este tipo de pruebas se tiene en cuenta los posibles caminos que puede tener el flujo de ejecución dentro del método que se está probando. Este tipo de pruebas unitarias son más costosas de implementar y se han realizado solo en los métodos más complejos que incluyen validaciones de distintos campos y son los más críticos de la aplicación pues hay más posibilidades de que algo falle si en un futuro sufren cambios o *refactoring*, para probar estos métodos se ha usado el algoritmo del camino mínimo, implementado de forma que se realizaba un test para cada caso, uno para cuando se cumplía la restricción y otro cuando no. Para estas pruebas se ha definido también un ejemplo

[23] Start Bootstrap: https://startbootstrap.com/
[24] JQuery: https://jquery.com/
[25] JQuery Validation: https://jqueryvalidation.org/
[26] jsonpath: https://code.google.com/archive/p/jsonpath/
[27] sweetalert2: https://sweetalert2.github.io/
[28] DataTables: https://datatables.net/
[29] Chart.js: https://www.chartjs.org/
[30] Bootstrap: https://getbootstrap.com/
[31] Font Awesome: https://fontawesome.com/

en el que se llama al servicio de usuarios y se actualiza un usuario dado, la diferencia con el ejemplo explicado anteriormente es que en este caso se prueba únicamente el método que comprueba que los datos introducidos cumplen las condiciones definidas para ser actualizado en la base de datos, y cuando no, se devuelve el mensaje de error esperado, la diferencia con las pruebas de caja negra es que se aísla este método para ser probado de forma que no actualiza nunca la base de datos ni hace uso de otros servicios que sean usados en este método pues estos servicios externos son *mockeados*.

Para facilitar la implementación de estas pruebas se ha usado el *framework JUnit*[32] con licencia *EPL v 2.0* y el *framework Mockito* bajo licencia *MIT*. Este último se utiliza para realizar *mocking* de servicios que se usan en el método o flujo que se está probando pero que no son el objetivo de estas pruebas, haciendo que estos servicios devuelvan el resultado que se le defina ante ciertas casticistas de forma que el mal funcionamiento de procedimientos ajenos no afecte a las pruebas, como es el caso del ejemplo comentado en las pruebas de caja blanca.

Finalmente también se han realizado pruebas de usabilidad, estas son realizadas por usuarios de la aplicación pidiéndoles que realicen tareas y tras esto se toma nota de dificultades que se puedan dar en estas tareas y sobre todo de errores que se descubran. Estas pruebas consistían en recorrer flujos de la aplicación, como por ejemplo crear una nueva pregunta y añadirla a un cuestionario existente.

4.8 Desafíos de programación

A lo largo del desarrollo han surgido complicaciones e inconvenientes que han supuesto un desafío y han forzado al equipo a buscar soluciones para estas situaciones no esperadas, en esta sección se muestran las más destacables.

4.8.1. Genericidad en los DAO

Hibernate posee un lenguaje para definir consultas SQL llamado HQL (*Hibernate Query Languaje*) con el que se pueden definir acciones que se compilarán en el lenguaje de consultas que se use con la fuente de datos a la que esté conectado el *framework*, este lenguaje era usado en cada implementación de una clase DAO para crear unos métodos básicos de un CRUD además de las distintas consultas que se realizarán a una entidad, por ejemplo en la pantalla de autenticación se consulta si el email introducido pertenece a algún usuario existente en la base de datos. Al realizar varias implementaciones de estos DAO para distintas entidades era evidente que los métodos básicos del CRUD se repetían variando solo en la entidad que realizaba esta acción, por esto se decidió crear una clase DAO genérica de la que se pudieran heredar estos métodos básicos: `GenericDAO<T,ID>`, pero el mayor reto no se encontraba ahí. Todavía quedaban muchos métodos de consultas, como el mencionado de consultar usuario por su email, y a medida que crecía la aplicación eran más abundantes y complejos, esto estaba provocando que la capa de persistencia fuera cada vez más costosa de mantener.

La solución a este problema se inspiró en la misma que con los métodos básicos del CRUD, la propuesta fue estandarizar las consultas de forma que se pudiera implementar un método genérico en la clase `GenericDAO` y de alguna forma indicar por qué parámetros se iba a filtrar y que el propio método transportará estos a la consulta. Para ello se decidió que estos parámetros se mandaran en un objeto `Map<String, Object>` siendo la clave del mapa el nombre del atributo por el que se desea filtrar y el valor que lo cribará

[32] JUnit: https://junit.org

enviado como un `englishObject` para mayor abstracción. Con esta filosofía se desarrolla el método de la figura 4.10. Este código recorre el mapa de las propiedades que le llega como parámetro y contempla casos:

- Valor `null`: obtiene los registros con esa propiedad a `null`.
- Valor `String`: si el valor es de este tipo se traen los registros que contengan la cadena enviada.
- Valor `Collection`: cuando se introduce como parámetro es una colección se evalúa que el valor de la propiedad se encuentre dentro de la lista dada.
- Otro valor: se devuelven registros en los que la propiedad evaluada tenga una valor igual a el comparado.

```
@Override
public List<T> search(Map<String, Object> filter) throws PersistenceException {

    try {
        Session session = sessionFactory.getCurrentSession();

        Criteria criteria = session.createCriteria(entityType);
        if (filter != null) {
            for (String propertyName : filter.keySet()) {
                Object value = filter.get(propertyName);

                if (value != null) {
                    if (value instanceof String) {
                        criteria.add(Restrictions.like(propertyName, "%" + value + "%"));
                    } else if (value instanceof Collection) {
                        criteria.add(Restrictions.in(propertyName, (Collection) value));
                    } else {
                        criteria.add(Restrictions.eq(propertyName, value));
                    }
                } else {
                    criteria.add(Restrictions.isNull(propertyName));
                }
            }
        }

        List<T> entities = criteria.list();

        return entities;
    } catch (javax.validation.ConstraintViolationException cve) {
        throw new PersistenceException(cve);
    } catch (org.hibernate.exception.ConstraintViolationException cve) {
        throw new PersistenceException(cve);
    }
}
```

Figura 4.10: Método genérico *Search*

4.8.2. Respuestas mostradas a tiempo real

Uno de los requisitos de la aplicación consiste en mostrar una pantalla de resultados donde se pueda comprobar a tiempo real las respuestas de los alumnos indicando si estas han sido correctas o incorrectas como se muestra en la figura 4.11.

Student	Correct answers	success rate	164	165	166	167	168
**********	2/5	40%	✓	✓	✗	✗	✗
**********	5/5	100%	✓	✓	✓	✓	✓
**********	4/5	80%	✓	✓	✓	✓	✗
**********	4/5	80%	✓	✓	✓	✓	✗
**********	4/5	80%	✓	✓	✓	✗	✓
**********	4/5	80%	✓	✓	✓	✗	✓
**********	5/5	100%	✓	✓	✓	✓	✓

Figura 4.11: Tabla de resultados

Para resolver se estudiaron varias soluciones. La primera que se planteó fue realizar consultas asíncronas al *back-end* para comprobar los resultados cada pocos segundos, pero esta opción no era atractiva debido a la sobrecarga de peticiones al servidor.

Por esto se planteó usar *websockets* un protocolo de comunicación *full-duplex*[33]. Pero esta tecnología requiere un coste alto de tiempo en aprendizaje, implementación, pruebas y mantenimiento aunque *spring* dispone de un paquete para realizar esto llamado *Spring WebFlux*[34] que se podría integrar perfectamente simplificando este proceso. Una vez establecida la solución podría ser más rápida en comparación con otras tecnologías debido a que al no enviar y recibir mensajes por el protocolo HTTP no necesita seguir sus protocolos ni usar grandes cabeceras lo cual simplifica la comunicación pero mantener muchas conexiones abiertas requeriría cargar enormemente al servidor con lo cual limitaría la escalabilidad, por tanto no es una buena opción para la idea de negocio que se plantea.

Otra tecnología que se planteaba como solución fue SSE (*server-sent events*). Esta opción comenzó a parecer la más viable debido a que resolvía algunos problemas por los que se descartaron los *websockets*, por ejemplo, la comunicación no es *full-duplex*, en este caso solo se produciría un envío del servidor al cliente con lo cual no es tan costosa, es más fácil de configurar e implantar y además también se puede integrar desde *Spring WebFlux*. El problema de esta solución es de nuevo sobrecargar el servidor, ya que, aunque menos que los *websockets* también se reduce un poco la escalabilidad, pero otra característica es que si usa el protocolo HTTP lo que hace los envíos de paquetes más pesados pero esto haría esta comunicación más homogénea con el resto de comunicaciones de la aplicación. Sopesadas estas diferencias, se optó por este método pero al implementarlo se observó que estas actualizaciones no eran todo lo reactivas que se esperaban y en muchas ocasiones no se podían contemplar los cambios en la base de datos tan rápido como sería necesario. Esto último es debido a que este desarrollo se ha realizado con el plan gratuito de Heroku, por tanto los recursos y uso de la base de datos de JawsDB tienen

[33] full-duplex: es un sistema de comunicación que permite el envío y recepción de información simultáneamente, el ejemplo más claro e una llamada telefónica

[34] Spring WebFlux: https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.html

ciertos límites, por tanto esta solución se validó en el entorno de Desarrollo debido a que se usa un servidor de datos en local, pero al desplegar la solución en Preproducción se observaron las carencias ya nombradas y se decidió buscar otras soluciones.

Por último se probó una idea distinta. Lo que se propuso esta vez era utilizar *Google Firebase*, se trata de una plataforma de Google desde la que se dispone distintos servicios que ofrecen un entorno donde desplegar aplicaciones. Uno de estos servicios es *Firebase Realtime Database* [20] el cual consiste en una base de datos no relacional que almacena objetos JSON, ideado para maximizar la reactividad. Esto ofrece lo que se buscaba, resuelve desventajas de las otras tecnologías propuestas, y aporta más ventajas. La idea consiste en crear un entorno en *Firebase* del que solo se use el servicio *Realtime Database* con lo cual se puede decir que en este proyecto funcionará como un servicio DBaaS pero no se descarta usar en un futuro otros de los servicios de este entorno, con esto funcionando se realiza una conexión con el *front-end* y otra con el *back-end* mediante las librerías ofrecidas por Google para JavaScript y Java respectivamente. Esta conexión directa al cliente servirá para conectar un escuchador que se mantenga activo en el momento que se entra en la pantalla donde se muestran los resultados y se desconecta cuando se salga de ella, entretanto cada vez que se produzca alguna actualización en alguno de los lanzamientos mostrados este escuchador los actualizará. Por otro lado, cuando un alumno responde una pregunta la respuesta de esta se manda al *back-end*, el cual la almacena en ambas bases de datos. Así se descarga de todos estos procedimientos la base de datos relacional, que será relegada solo a la persistencia de datos y esta base de datos no relacional guardará información solo de forma temporal, ya que solo contiene datos de los lanzamientos en curso, convirtiéndola así en una especie de memoria caché destinada solo a esta función. Comparado con las otras alternativas, carga mucho menos la base de datos y el servidor principal de la aplicación ya que *Firebase* se sitúa en un servidor distinto, es mucho más fácil de implantar en el proyecto y las comunicaciones con esta nueva base de datos las gestiona por completo las librerías dispuestas para ello permitiendo al desarrollador realizar ese escuchador que mantenga actualizada la página de resultados simplemente con el código que se muestra en la figura 4.12.

```
firebase.database().ref('Launches').on('value', function (snapshot) {

    // Código a ejecutar cuando se añada una respuesta

});
```

Figura 4.12: Tabla de resultados

4.8.3. Versionado de preguntas

En el segundo MVP se llegó a la conclusión de que tras haber lanzado un *quiz* no se podía permitir al usuario modificar ni borrar las preguntas que lo componían, ya que, de ser así los resultados almacenados podrían no coincidir con lo que respondieron los alumnos en su momento, en caso de que se hubiera cambiado las respuestas o el enunciado de manera significativa. Por ello se decidió incluir un sistema de versiones que permitiera modificar preguntas ya usadas creando una nueva versión y se mantuvieran almacenadas las versiones anteriores para respetar la integridad de los datos.

En un principio se planteó la idea de crear una tabla en la base de datos en la que se fueran almacenado versiones anteriores de las preguntas, pero debido al cambio que podría suponer esto en la lógica de la aplicación se trató de buscar una solución más simplificada. Otra propuesta era aprovechar la misma tabla que almacena las preguntas

pero en la cual se creará un nuevo campo que referencie al identificador de otra pregunta, de esta forma se podría obtener una lista enlazada con todas las versiones de una misma pregunta. Pero este proceso seguía sin ser práctico cuando se trataba de consultar la última versión de la pregunta o la primera, así que para simplificar el proceso de forma que con consultas simples se pueda recuperar todo lo necesario sobre las versiones de cada pregunta se ideó un sistema de tres referencias como se muestra en la figura 4.13. Una referencia irá dirigida a la primera versión de la pregunta (`FIRST_VERSION`), otra a la siguiente (`NEXT_VERSION`) y por último a la anterior (`PREVIOUS_VERSION`).

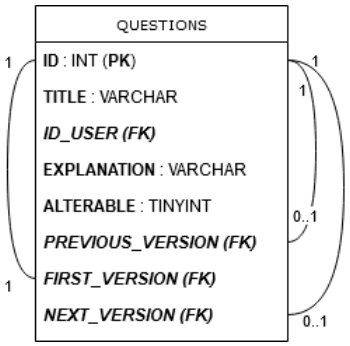

Figura 4.13: Patrón de versiones

Esto simplificó mucho las consultas que se requieren en la aplicación para integrar el sistema de versiones, ya que se esta forma no había que recorrer listas enlazadas o mantener otras entidades, se podía obtener nos resultados necesarios implementando estas consultas:

- Lista de preguntas en su versión actual: obtener las que su campo `NEXT_VERSION` sea nulo.
- Última versión de una pregunta concreta: filtrar por las preguntas que tengan la misma `FIRST_VERSION` y además `NEXT_VERSION` con valor nulo.
- Identificador a mostrar al usuario de una pregunta: el campo `FIRST_VERSION`, de esta forma da la sensación que al crear una nueva versión se ha actualizado y sigue siendo la misma pregunta.
- Retroceder a la versión anterior: de convierte la pregunta cuyo identificador sea igual a `PREVIOUS_VERSION` en la ultima versión igualando su valor `NEXT_VERSION` a nulo, y se borra la versión que antes era la última.

Con estas tres consultas se puede gestionar lo que se necesita por ahora del sistema de versiones, el cual al crear un *quiz* lo enlaza con la primera versión de las preguntas asignadas, pero al lanzarlo se obtiene las últimas versiones de estas preguntas y al responder se almacena el identificador de la versión que se ha respondido junto con el resultado, por lo tanto la respuesta de un alumno está relacionada con la versión respondida y si se ven los resultados en un futuro se podrá ver la pregunta tal y como la visualizo el alumno al responderla.

4.8.4. Carga de entidades muy pesadas

Como se puede observar en la figura 4.3, el modelo de datos relacional está interconectado de forma que se generan bucles como por ejemplo el que forman las conexiones entre: *QUESTIONS*, *QUIZZES*, *lAUNCHES*, *RESULTS* y *ANSWERS*. Esto puede llevar a

que cuando el sistema ORM recupera una entidad traiga demasiada información consigo, tomando como ejemplo el bucle nombrado, al solicitar una pregunta se obtendría desde la base de datos todos los cuestionarios relacionados con esa pregunta, todos los lanzamientos que impliquen esos cuestionarios y todos los resultados de estos lanzamientos junto con las respuestas dadas en estos, por supuesto con esas respuestas se alcanzaría a sus respectivas preguntas y de esta forma se obtiene una cantidad de datos innecesaria en la mayoría de ocasiones. Para minimizar la cantidad de datos que se cargan desde la base de datos y los que se envían al cliente se ha hecho uso de dos técnicas.

Una de estas técnicas es el uso de DTO (*data transfer object*), esto son objetos que contienen los atributos necesarios para la comunicación entre cliente y servidor de forma que no se envían en esa comunicación entidades enteras.

La otra técnica es el patrón de diseño *Lazy loading*. El objetivo con este tipo de carga es traer de la base de datos solo lo necesario, pongamos que tenemos dos entidades relacionadas entre sí y cargamos una de las dos, aplicando este patrón únicamente se carga la que se ha pedido y la otra no se carga hasta el momento que se quiera acceder a ella, si es que se utiliza. Este patrón se puede usar en *Hibernate* simplemente configurándolo al definir relaciones entre entidades, por ejemplo, en la entidad *Question* existe una colección de la entidad *Answer*, al definir esa relación en *Hibernate* se puede indicar el tipo de inicialización.

```
@OneToMany(mappedBy="question", fetch = FetchType.LAZY)
private Set<Answer> answers;
```

Figura 4.14: *Lazy loading* con *Hibernate*

Pero *Lazy loading* tiene un problema, el cual se da cuando se posee una relación como la que se muestra en la figura 4.14 y es que al no cargar la lista de objetos hasta que no se accede a ellos, si se pretende recorrer esta lista se producen tantas consultas a la base de datos como elementos sean consultados de la colección pues estos son cargados uno a uno. Esto es conocido como el problema n+1 y hace que este patrón pensado para optimizar las cargas termine por generar pérdida de rendimiento. La solución a este problema es crear la consulta para recuperar esta entidad a mano y añadirle un `LEFT JOIN FETCH question.answers`, pero de esta forma se tendría que hacer una consulta personalizada con cada entidad que se vea afectada, esto supondría perder parte de la ventaja que da la clase `GenericDAO`, ya que se tendrían que crear muchas consultas personalizadas. Finalmente se encontró otra forma de resolver este problema a través del objeto `org.hibernate.Hibernate` [21] el cual tiene un método que permite la inicializar colecciones, de manera que ejecutándolo antes de recorrer la colección por primera vez se produciría esta carga con solo una consulta, el código para este ejemplo se muestra en la figura 4.15.

```
Hibernate.initialize(question.getAnswers());
```

Figura 4.15: Inicializar colección con *Hibernate*

CAPÍTULO 5
Cronología del TFG

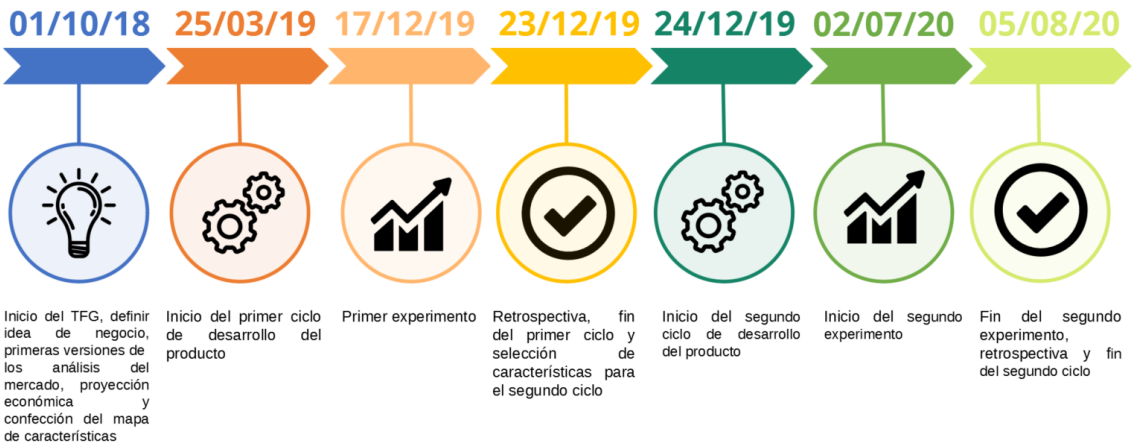

Figura 5.1: Línea temporal del trabajo

Como se observa en la figura 5.1, existen siete eventos importantes en el desarrollo del trabajo que se podrían resumir en tres etapas principales que son las siguientes:

- **01/10/18 - 12/02/19**: Inicio del TFG, definir idea de negocio, primeras versiones de los análisis del mercado, proyección económica y confección del mapa de características.
- **25/03/19 - 23/12/19**: Primer ciclo de desarrollo del producto. Este periodo incluye las reuniones de planificación del primer MVP y su desarrollo; las distintas entregas con sus demos y retrospectivas; el primer experimento y la retrospectiva final.
- **24/12/19 - 20/06/20**: Segundo ciclo de desarrollo del producto, el cual se compone de las mismas fases que el primer ciclo.

Pero durante estos periodos no se ha dispuesto de una dedicación completa a este trabajo, por lo que para mostrar una visión más aproximada del esfuerzo invertido también se han contabilizado las horas efectivas invertidas para cada actividad, y esta información se plasma en la tabla 5.1.

Tabla 5.1: Distribución de horas invertidas en el TFG

Actividad	Tiempo invertido
Formación e investigación	42h
Estudios de mercado y definición de la idea de negocio	37h
Desarrollo del producto	193.5h
Redacción de la memoria	74h
Total	**356,5h**

CAPÍTULO 6
Conclusiones y Trabajo Futuro

6.1 Conclusiones

En este trabajo se ha expuesto el procedimiento que se ha llevado a cabo para desarrollar una idea de negocio y analizar su potencial en el mercado actual, con ello se ha podido definir la estrategia de negocio más óptima para el mercado al que se introducirá.

También se ha expuesto una proyección económica la cual concluye que una inversión de 52.000€ darían viabilidad a este proyecto, asimismo esta inversión multiplicaría por 11,4 su valor en 5 años.

Por otro lado se ha desarrollado un producto software con una arquitectura que permite realizar cualquier cambio en tecnologías o lógica de negocio rápidamente y con la mínima repercusión en el resto de la aplicación, además desplegada en un entorno basado en proveedores de servicios con tarifas de pago por uso, las cuales ofrecen una escalabilidad sin límite.

Durante el desarrollo de este producto se ha documentado, en forma de unidades de trabajo y sus especificaciones, las tareas realizadas y el tiempo invertido en ellas. Gracias a esto se puede reflexionar sobre cómo mejorar el proceso de desarrollo, por ejemplo, el tiempo total invertido en desarrollo ha sido 193.5 horas y el estimado total es 188 horas, por lo que se puede concluir que las estrategias de estimación han resultado muy eficientes, ya que solo hay una diferencia de un 3 %. Sin embargo, en las estimaciones del principio de cada fase de desarrollo no se tuvo en cuenta las correcciones surgidas de las revisiones del *product owner* u otros usuarios. En ese caso la cifra cambia pues el tiempo estimado total de las entregas 1, 3 y 4, que son las dedicadas al desarrollo de las características acordadas desde un principio para ambos ciclos, suma 164 horas y la diferencia con las horas totales de desarrollo, contando todas las entregas, es del 18 %. Esta ya es una diferencia a considerar pues pudo haber provocado la cancelación de experimentos ya programados debido a este retraso, por ello para estimar fechas de finalización para ciclos de desarrollo se tendrá en cuenta un 20 % del tiempo estimado como incertidumbre.

Finalmente este producto ha sido puesto a prueba en un entorno real en dos experimentos donde se han validado y se han adquirido conocimientos vitales en su evolución.

De acuerdo con todo lo indicado, se consideran cumplidos los objetivos marcados para el desarrollo de este trabajo. Pero además, al principio de este documento también se exponían unos objetivos personales, los cuales han sido ampliamente logrados. Este TFG ha supuesto un auténtico reto para este equipo de desarrollo compuesto, hasta el momento, por un solo desarrollador *full stack* que ha aprendido, luchado, perdido y ganado con cada una de las tecnologías, técnicas y metodologías que se emplean en este trabajo. Esto

ha permitido que se adquiera experiencia en todas las disciplinas de desarrollo de una aplicación web, a la vez que se han puesto en práctica conocimientos teóricos estudiados en distintas asignaturas cursadas a lo largo del grado, se podría decir que casi todas las asignaturas han influido de alguna forma en la formación necesaria para realizar este trabajo, pero hay que hacer especial mención a las asignaturas de: Proceso de Software, Proyecto de Ingeniería de Software, Diseño de Software, Mantenimiento y Evolución de Software, Ingeniería del software y Deontología y profesionalismo; pues estas asignaturas han contribuido especialmente a la adquisición de los conocimientos específicos que han sido necesarios en este proyecto.

6.2 Trabajo Futuro

Actualmente se dispone de un producto software que estaría por comenzar su tercer MVP y ya es capaz de cubrir las funcionalidades más básicas esperadas del mismo. Ya se pueden crear y mantener preguntas y cuestionarios, es posible también gestionar salas donde se pueden añadir alumnos a lo que realizar un seguimiento, con todo esto lanzar un examen en estas salas el cual puede ser respondido por cualquiera que tenga el código que lo referencia o solo los alumnos de la sala seleccionada. Por último, se obtienen resultados de las respuestas a tiempo real junto con un seguimiento de las notas obtenidas por estos alumnos en todos los cuestionarios lanzados en una misma sala.

En lo referente a la parte de negocio, será necesario buscar socios que puedan encargarse de la parte relativa al *marketing* de forma que se comience a trabajar este aspecto que durante el lanzamiento será muy importante, además de comenzar a buscar posibles formas de financiación para cubrir la puesta en marcha del producto.

Referencias

[1] Alexander Osterwalder e Yves Pigneur. *Business Model Generation*. Wiley, 2010.

[2] Definición del indice EBITDA. Consultado en noviembre de 2018 https://www.bancosantander.es/es/diccionario-financiero/ebitda.

[3] Precio de TPV. Consultado en octubre de 2018 https://www.bbva.es/empresas/productos/tpv/tpv-virtual.html.

[4] Definición del análisis DAFO por el Ministerio de industria, comercio y turismo. Consultado en octubre de 2018 https://dafo.ipyme.org/Home.

[5] Eric Ries. *El método Lean Startup*. Crown Business Publishing, S.A., 2011.

[6] Mike Cohn. *Agile Estimating and Planning*. Prentice Hall, 2005.

[7] Manifiesto ágil. Consultado en enero de 2019 https://agilemanifesto.org/iso/es/manifesto.html.

[8] GNU General Public License, version 2. Consultado en marzo de 2019 https://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

[9] Apache License. Consultado en marzo de 2019 http://www.apache.org/licenses/LICENSE-2.0.

[10] ISC License. Consultado en marzo de 2019 https://www.isc.org/licenses/.

[11] GNU Lesser General Public License v3. Consultado en marzo de 2019 https://www.gnu.org/licenses/lgpl-3.0.html.

[12] MIT License. Consultado en marzo de 2019 https://opensource.org/licenses/MIT.

[13] CC BY 4.0 License. Consultado en marzo de 2019 https://creativecommons.org/licenses/by/4.0/.

[14] SIL OFL License. Consultado en marzo de 2019 https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL.

[15] Heroku CLI. Consultado en abril de 2019 https://devcenter.heroku.com/articles/heroku-cli.

[16] Bryan Basham, Kathy Sierra y Bert Bates *Head First Servlets and JSP*. O'Reilly Media, Inc., 2008.

[17] Spring MVC. Consultado en marzo de 2019 https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html.

[18] MySQL Connector/J: Manual de licencia. Consultado en marzo de 2019 https://downloads.mysql.com/docs/licenses/connector-j-5.1-gpl-en.pdf.

[19] Figura Modelo MVC extraído en julio de 2020 de: https://commons.wikimedia.org/wiki/File:ModelViewControllerDiagram_es.svg.

[20] Documentación de Firebase Realtime Database. Consultado en abril de 2019 https://firebase.google.com/docs/database?hl=es-419.

[21] Documentación del objeto Hibernate. Consultado en junio de 2019 https://docs.jboss.org/hibernate/orm/5.3/javadocs/org/hibernate/Hibernate.html.

APÉNDICE A
Anexos

A.1 Manual de usuario

Al entrar a la aplicación aparece una pantalla como la mostrada en la figura A.1 desde donde se puede acceder a la autenticación como profesor o a responder un cuestionario.

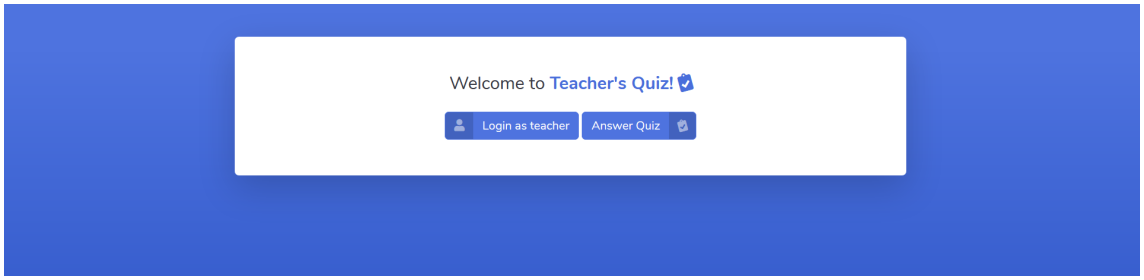

Figura A.1: Teacher's Quiz: autenticación

Seleccionando la opción de autenticación como profesor, se puede introducir un email y contraseña para acceder a la aplicación, donde la primera pantalla mostrada es el *Dashboard* como se ve en la figura A.2 y donde se encuentran dos enlaces principales, uno para lanzar un cuestionario y otro para ir a la pantalla de resultados.

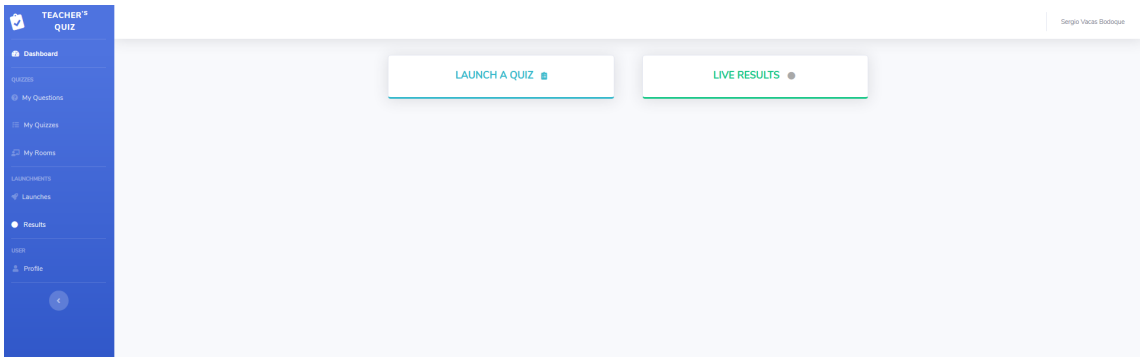

Figura A.2: Teacher's Quiz: Dashboard

A.1.1. Preguntas

Desde el menú principal mostrado a la izquierda se puede acceder a la opción *My Questions* donde aparecerá una pantalla con la lista de preguntas creadas por el usuario al igual que la expuesta en la figura A.3.

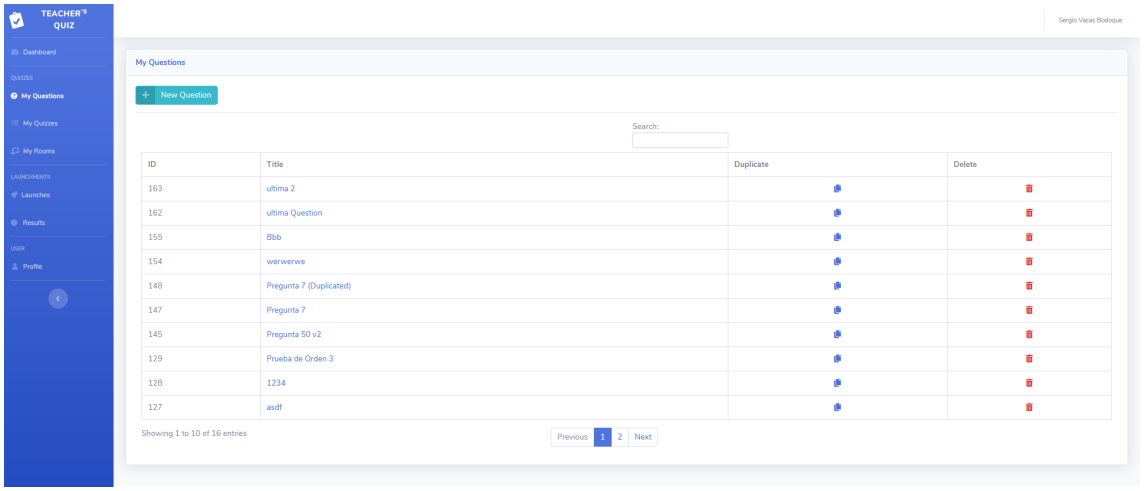

Figura A.3: Teacher's Quiz: Mis preguntas

Desde esta tabla se puede duplicar o eliminar preguntas, pero también se pueden crear nuevas pulsando el botón *New Question* el cual enlaza con la página de la figura A.4. En esta pantalla se introducen los datos necesarios de una pregunta, pudiendo añadirle tantas posibles respuestas como se quiera.

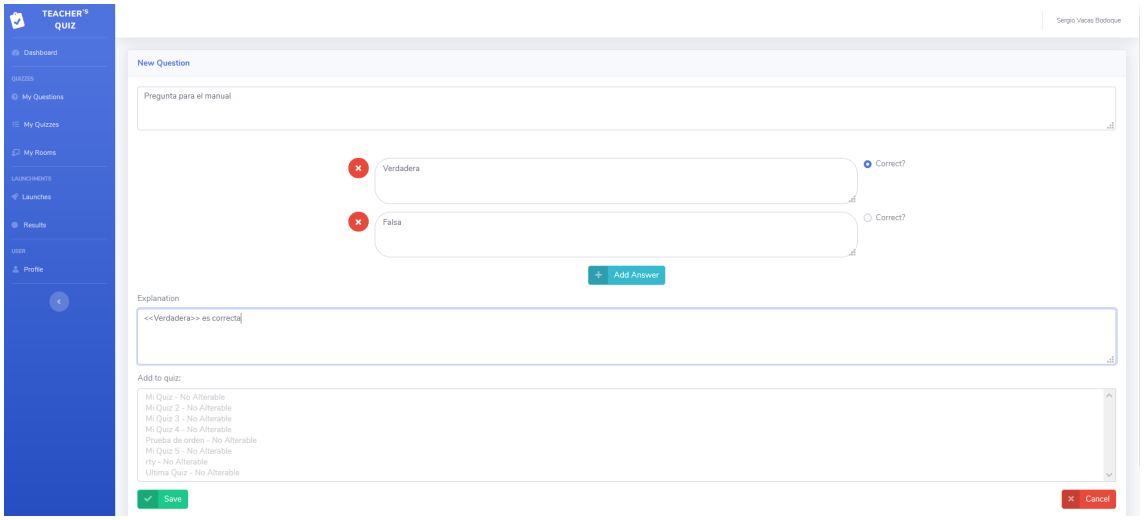

Figura A.4: Teacher's Quiz: Crear pregunta

En la pantalla *My Questions* también se puede modificar preguntas pulsando sobre su titulo, esto traslada al usuario a una pantalla similar a la de creación pero con los datos de la pregunta seleccionada, pero si se trata de modificar una pregunta que ha sido usada en un lanzamiento no se permitirá su modificación, sin embargo, aparecerá un dialogo como el mostrado en la figura A.5 ofreciendo al usuario las opciones de realizar esas modificaciones en un duplicado de la pregunta, crear una nueva versión de esta pregunta con esas modificaciones o cancelar el proceso.

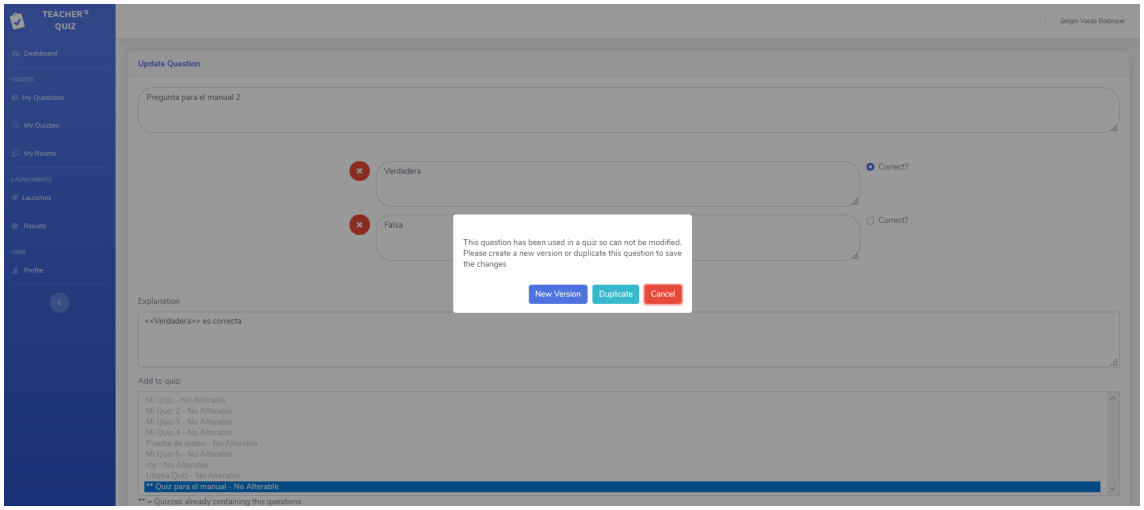

Figura A.5: Teacher's Quiz: Crear pregunta

A.1.2. Cuestionarios

Otra de las opciones en el menú principal es *My Quizzes*, como se ve en la figura A.6 esta es similar a la pantalla de gestión de preguntas y también permite realizar las mismas acciones.

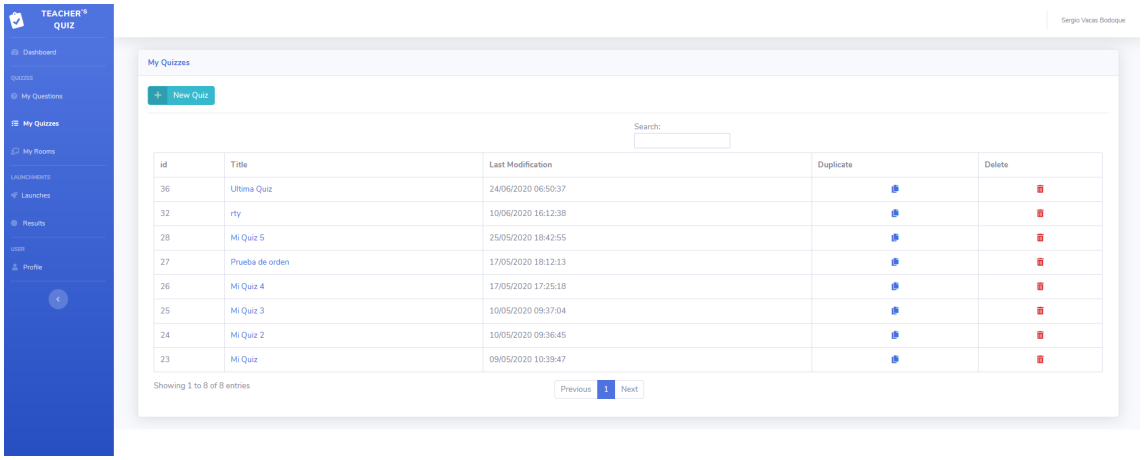

Figura A.6: Teacher's Quiz: Mis cuestionarios

Es posible crear un cuestionario con el botón *New Quiz*, el cual muestra la pantalla de la figura A.7.

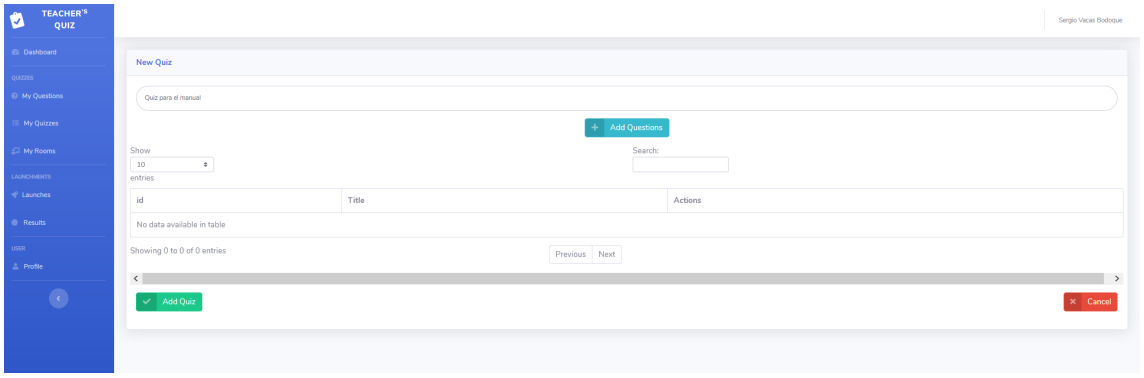

Figura A.7: Teacher's Quiz: Crear cuestionario

El botón *Add Question* muestra una ventana modal que permite añadir preguntas ya creadas a este nuevo cuestionario mostrando el listado de preguntas del usuario como ilustra la figura A.8.

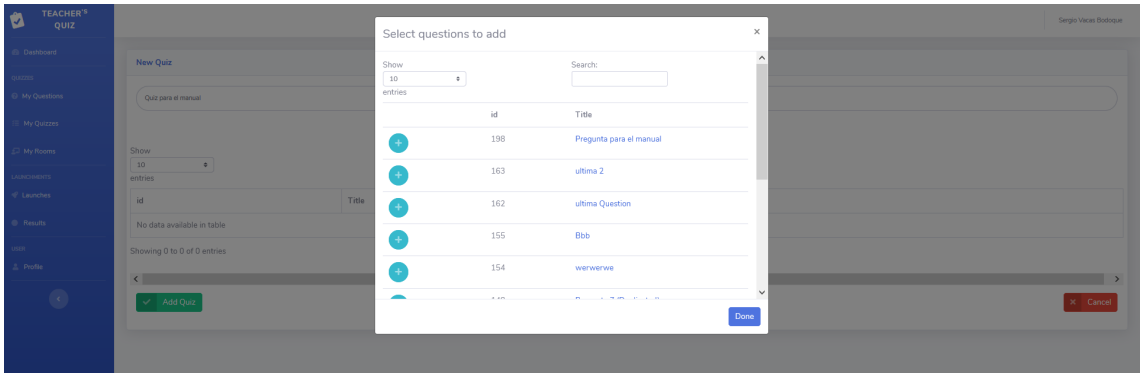

Figura A.8: Teacher's Quiz: Añadir preguntas a un cuestionario

A.1.3. Salas

Volviendo al menú principal, la opción *My Rooms* enlaza con una página de gestión igual a las vistas para preguntas y cuestionarios pero con opciones distintas, pues como se observa en la figura A.9 las salas no pueden duplicarse pero tienen una opción para mostrar puntuaciones referentes al seguimiento de los alumnos en esta sala.

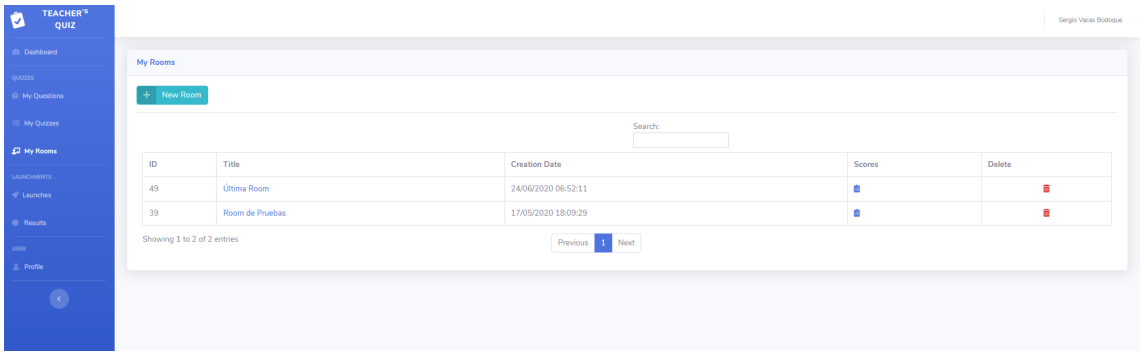

Figura A.9: Teacher's Quiz: Mis salas

Para crear una nueva sala hay que pulsar el botón *New Room*, este traslada al usuario a la página que se muestra en la figura A.10.

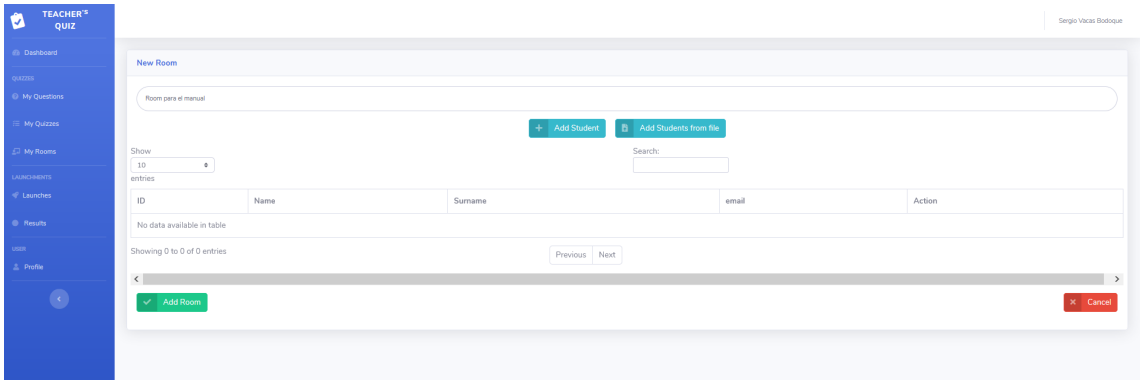

Figura A.10: Teacher's Quiz: Crear sala

Desde esta pantalla de añade un nombre para la sala y se pueden añadir usuarios de forma masiva desde un archivo con formato CSV con el botón *Add Students from file* o añadirlos uno a uno con el botón *Add Students* el cual hará que aparezca una ventana modal que guiará el proceso como en la figura A.11.

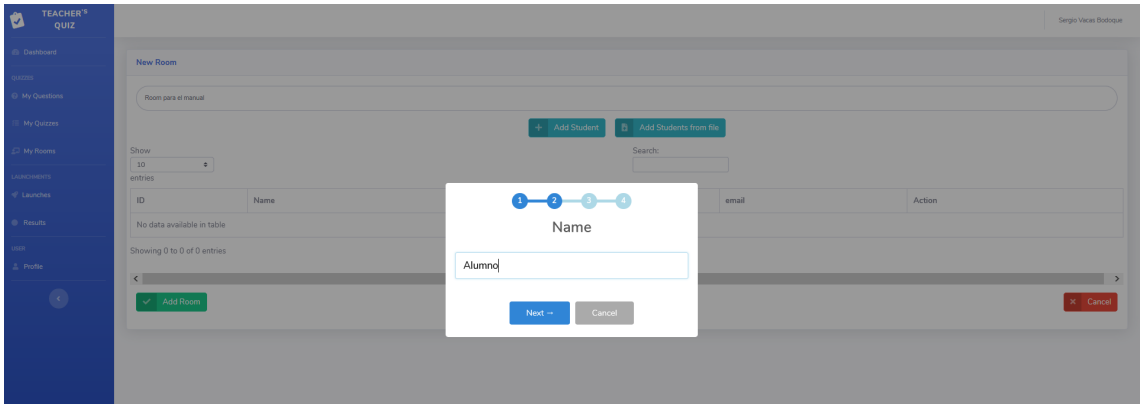

Figura A.11: Teacher's Quiz: Añadir alumno

Otra de las cosas que ofrece la pantalla *My Rooms* es acceder a las puntuaciones obtenidas por los alumnos de esa sala en los cuestionarios privados, se puede acceder a esta pantalla desde la opción *Scores*. Como se observa en la figura A.12, en esta pantalla se muestran una tabla con los cuestionarios lanzados en esta sala y una última columna con la media aritmética de las notas obtenidas por los alumnos en esto cuestionarios.

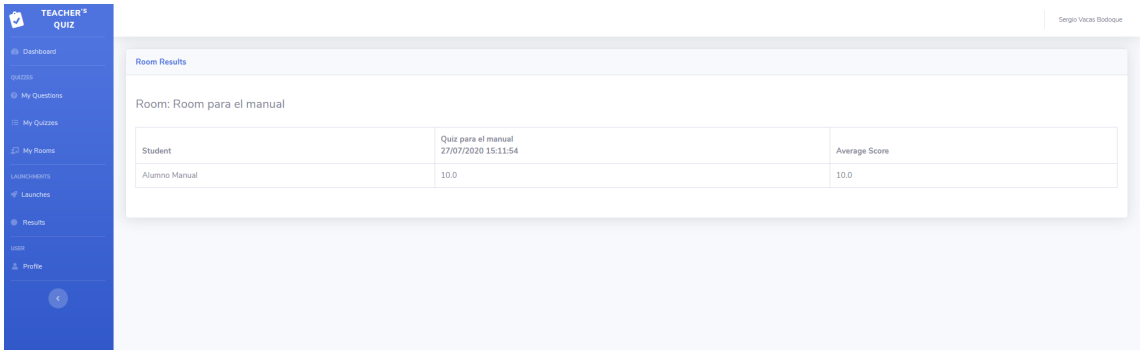

Figura A.12: Teacher's Quiz: Resultados de la sala

A.1.4. Lanzamientos

En la opción *Lanunches* del menú principal se accede a una pantalla donde se pueden ver los lanzamientos creados y sus atributos como de muestra en la figura A.13.

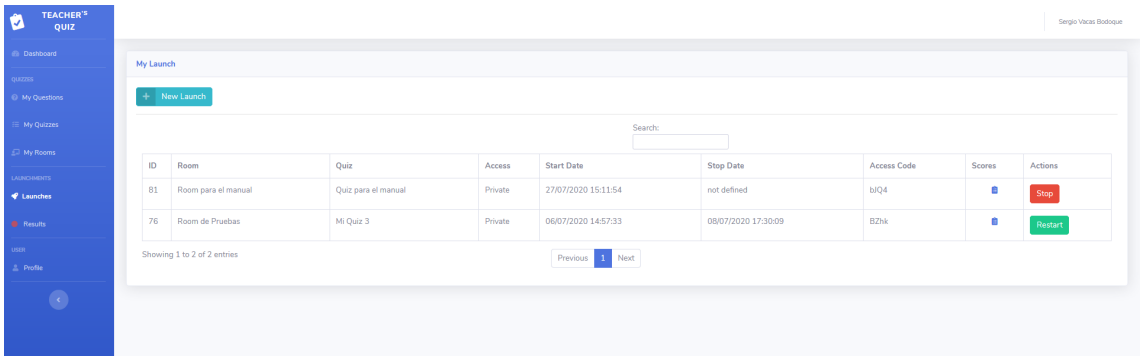

Figura A.13: Teacher's Quiz: Lanzamientos

Esta pantalla ofrece las opciones de parar un lanzamiento o reiniciar uno ya detenido, también tiene un enlace a los resultados del lanzamiento desde *Scores*.

A.1.5. Resultados

Desde la opción del menú principal *Results* se puede acceder a la página mostrada en la figura A.14 donde se pueden observar las respuestas en tiempo real dadas por los alumnos a un cuestionario que esté lanzado actualmente.

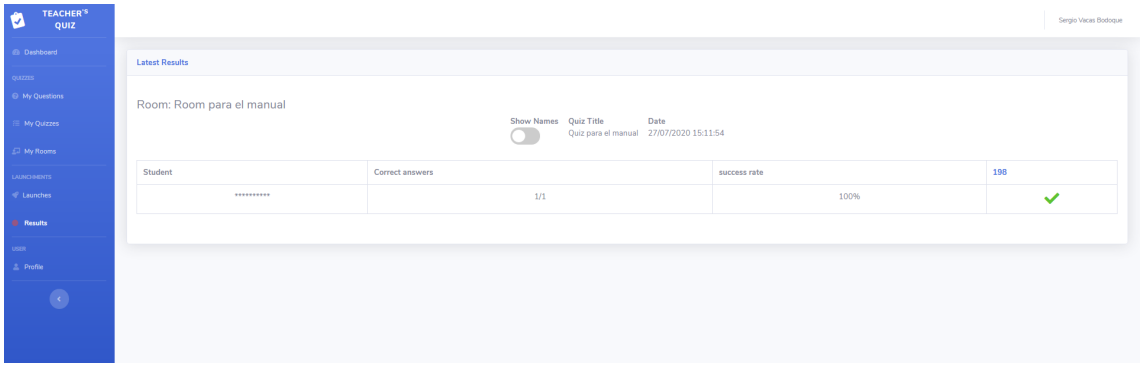

Figura A.14: Teacher's Quiz: Resultados

En esta tabla muestran columnas con las distintas preguntas que componen en cuestionario lanzado, la cabecera de cada columna que pertenece a una pregunta muestra su identificador, pulsando sobre este número se mostrará la pantalla que aparece en la figura A.15.

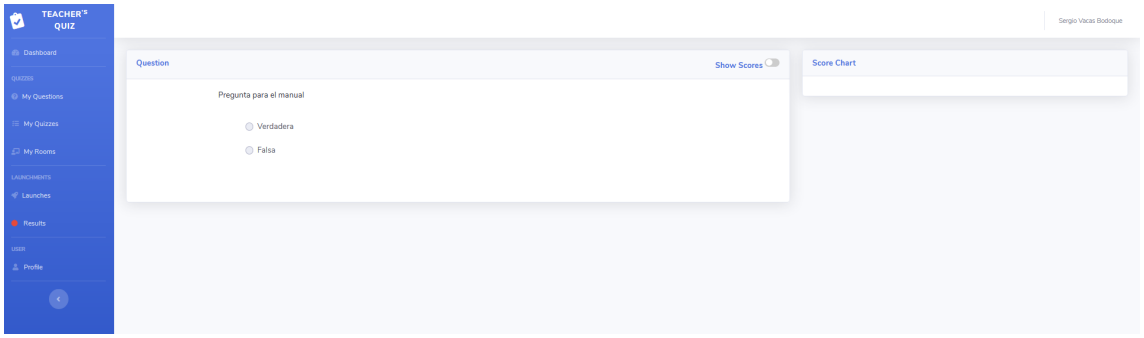

Figura A.15: Teacher's Quiz: Pregunta en un lanzamiento

Aquí se puede ver la pregunta seleccionada, pero pulsando el *switch* con el texto *Show Scores* se mostrarán estadísticas del porcentaje de elección de cada respuesta en ese lanzamiento y un gráfico que ilustra la cantidad de aciertos y fallos, además de la explicación definida para esa pregunta si la tuviera igual que se ve en la figura A.16.

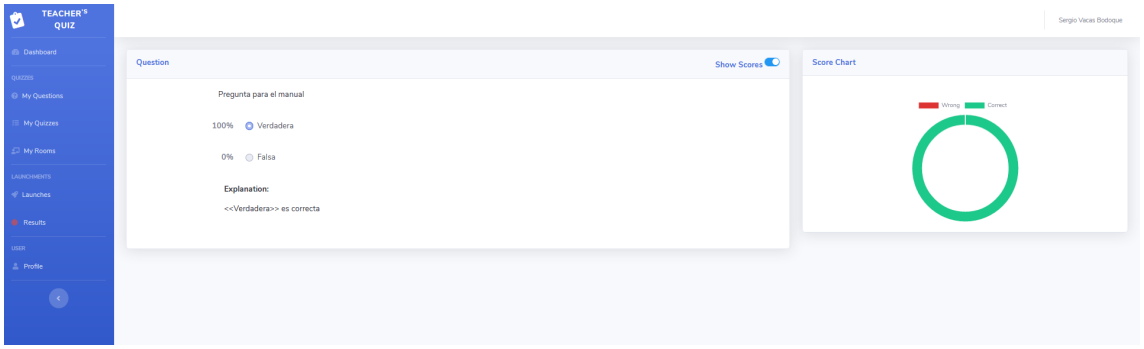

Figura A.16: Teacher's Quiz: Resultados de una pregunta en un lanzamiento

A.1.6. Perfil

También se cuenta con una página *Profile*, desde donde se pueden actualizar los datos del usuario actual como se observa en la figura A.17

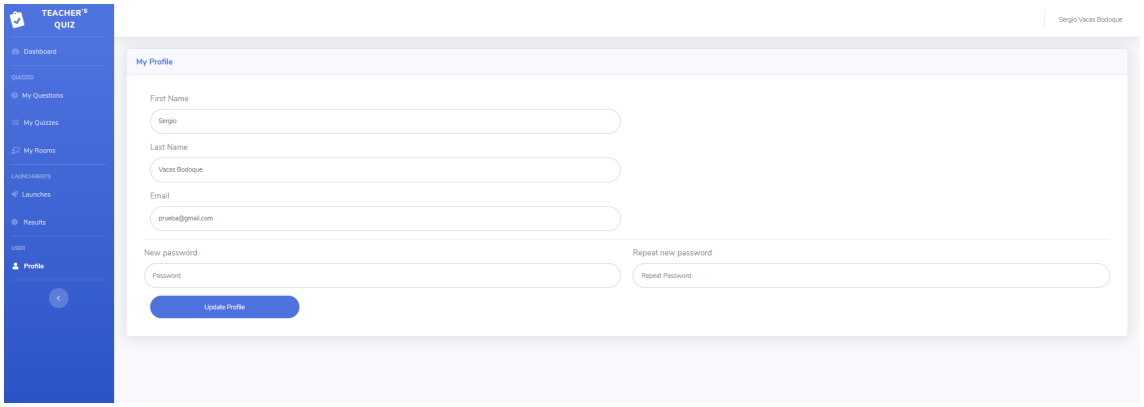

Figura A.17: Teacher's Quiz: Perfil del usuario

A.1.7. Responder un cuestionario

Para responder un cuestionario se parte de la página inicial ya mostrada en la figura A.1, pero en este caso hay que entrar en la opción *Answer Quiz* que nos llevará a la pantalla de la figura A.18.

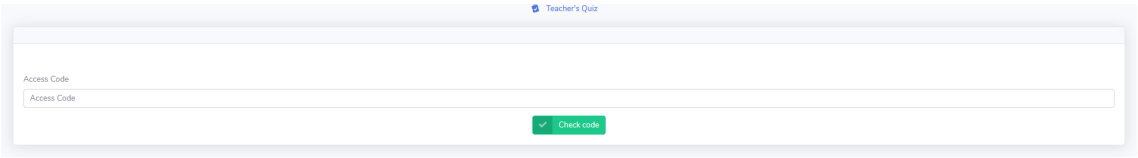

Figura A.18: Teacher's Quiz: Unirse a un lanzamiento

Aquí se introduce el código del lanzamiento al que se pretende acceder y se pulsa el botón *Check code* que si comprueba que es un código existente en un lanzamiento activo mostrará un campo para introducir un nombre o un email dependiendo de si se trata de un lanzamiento público o privado como es el caso de la figura A.19.

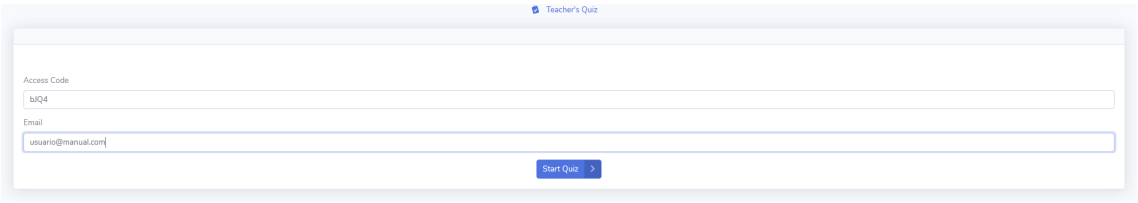

Figura A.19: Teacher's Quiz: Unirse a un lanzamiento privado

Tras esto ya se tiene acceso al cuestionario, este se completa respondiendo las preguntas de una en una, pulsando el botón *Next* o *End quiz* si al igual que en la figura A.20 se trata de la última pregunta.

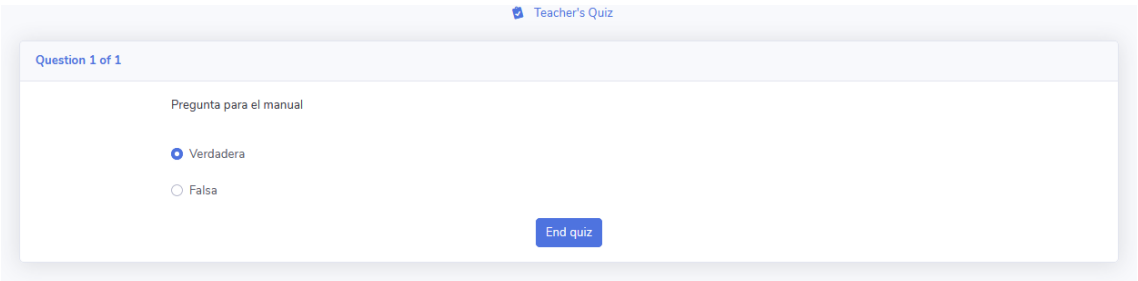

Figura A.20: Teacher's Quiz: Unirse a un lanzamiento privado

Por último, cuando se termina un cuestionario se muestran las preguntas acertadas por el alumno como se aprecia en la figura A.21.

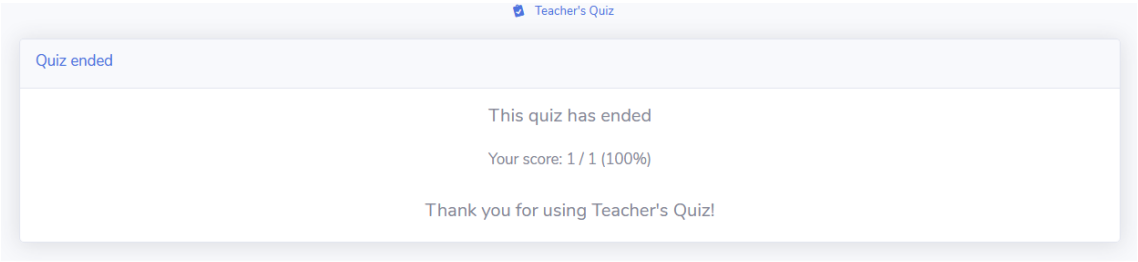

Figura A.21: Teacher's Quiz: Unirse a un lanzamiento privado