



Universidad de  
Oviedo



# **ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN**

## **GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN**

### **ÁREA DE LENGUAJES Y SISTEMAS INFORMÁTICOS**

#### **IA/Boratory: PLATAFORMA PARA LA CREACIÓN DE EXPERIMENTOS UTILIZANDO A/B TESTING**

**D. NONIDE MIRANDA, Sergio**

**TUTOR: D. BLANCO AGUIRRE, Raquel**

**COTUTOR: D. MENGUEZ ÁLVAREZ, Guillermo**

**FECHA: Julio, 2022**

# Índice

1	Introducción y motivación .....	9
1.1	Introducción.....	9
1.2	Motivación.....	10
2	Objetivos y alcance .....	11
2.1	Acceso a la aplicación .....	12
2.2	Experimentos .....	12
2.3	Activaciones .....	12
2.4	Algoritmo de asignación a los grupos del experimento .....	13
2.5	Consumir los experimentos creados .....	14
3	Estudios y análisis previos .....	15
3.1	Aplicaciones similares en el mercado .....	15
3.1.1	Firestore A/B Testing .....	15
3.1.2	Optimizely .....	15
3.1.3	Adobe Target .....	15
3.2	Tecnologías para el <i>backend</i> .....	16
3.2.1	C# y .Net .....	16
3.2.2	Java .....	16
3.2.3	Go.....	16
3.2.4	JavaScript.....	16
3.3	Tecnologías para el <i>frontend</i> .....	17
3.3.1	Angular .....	17
3.3.2	React .....	17
3.3.3	Vue.....	17
3.4	Motores de base de datos .....	17

3.4.1	PostgreSql .....	17
3.4.2	MySQL .....	17
3.4.3	MongoDB .....	18
3.5	Control de versiones en remoto .....	18
3.5.1	GitHub .....	18
3.5.2	GitLab .....	18
3.5.3	Bitbucket .....	18
3.6	Elección final .....	19
4	Planificación .....	20
4.1	Metodología utilizada .....	20
4.2	Tabla de las tareas a desarrollar .....	20
4.3	Diagrama de Gantt .....	23
5	Presupuesto .....	24
5.1	Presupuesto Hardware .....	24
5.2	Presupuesto Software .....	25
5.3	Amortización del inmovilizado material .....	25
5.4	Presupuesto de mano de obra .....	25
5.5	Presupuesto total .....	26
6	Análisis .....	27
6.1	Requisitos de usuario .....	27
6.1.1	Requisitos funcionales .....	27
6.1.2	Requisitos no funcionales .....	28
6.2	Mapa de historias de usuario .....	29
6.3	Historias de usuario .....	30
6.3.1	Diseñar una API básica para el <i>backend</i> .....	30
6.3.2	Explorar el uso de Swagger en la API .....	30
6.3.3	Aplicar los cambios hablados de la sesión anterior .....	30

6.3.4	Sistema de registro en la aplicación.....	30
6.3.5	Sistema de autenticación en la aplicación.....	31
6.3.6	Validación de experimentos.....	31
6.3.7	Eliminación de experimentos.....	31
6.3.8	Listado de experimentos .....	31
6.3.9	Creación de experimentos nuevos .....	32
6.3.10	Restricción en los nombres duplicados de experimentos.....	32
6.3.11	Eliminación dinámica de asignaciones .....	32
6.3.12	Descripción en los experimentos.....	32
6.3.13	Descripción en las asignaciones .....	32
6.3.14	Modificación de los experimentos .....	33
6.3.15	Sistema de balanceo .....	33
6.3.16	Activar una sola asignación.....	33
6.3.17	Asignaciones manuales .....	33
6.3.18	Deshabilitar experimentos.....	34
6.3.19	Pedir asignaciones para webs externas.....	34
6.3.20	Web de ejemplo.....	34
6.3.21	Listar las asignaciones manuales.....	34
6.3.22	Eliminar asignaciones manuales .....	35
6.3.23	Notificar del balanceo asíncrono.....	35
6.3.24	Hacer segura la concurrencia .....	35
6.3.25	Compartir experimentos .....	35
6.3.26	Agrupar experimentos .....	36
6.3.27	Explorar el uso de tecnologías Cloud.....	36
6.4	Modelo de dominio.....	37
6.5	Prototipos de pantalla .....	38
6.5.1	Pantalla inicial.....	38

6.5.2	Pantalla de registro de nuevos usuarios .....	38
6.5.3	Pantalla de inicio de sesión .....	39
6.5.4	Pantalla principal .....	39
6.5.5	Pantalla de creación de experimentos .....	40
6.5.6	Pantalla de visualización de experimentos .....	40
6.6	Mapa de navegación .....	41
7	Diseño del sistema .....	42
7.1	Arquitectura .....	42
7.2	Arquitectura del frontend.....	43
7.3	Arquitectura del backend.....	44
7.4	Diseño físico de los datos .....	45
7.5	Modelo público de la API.....	46
7.6	Patrones de diseño implementados.....	48
7.7	Estándares utilizados .....	49
7.7.1	JSON .....	49
7.7.2	REST.....	49
7.7.3	JWT.....	50
7.8	Entorno de pruebas .....	52
7.9	Diseño de las pruebas .....	52
7.10	Ejecución de las pruebas.....	54
7.10.1	Creación de usuarios nuevos .....	54
7.10.2	Autenticación .....	54
7.10.3	Validar experimentos nuevos .....	56
7.10.4	Comprobar si el experimento nuevo ya existe .....	59
7.10.5	Validar asignaciones .....	60
7.10.6	Validar el propietario del experimento.....	62
7.10.7	Generar asignación balanceada .....	64

8	Manual de usuario.....	66
8.1	Introducción.....	66
8.2	Puesta en marcha .....	66
8.3	Pantalla de inicio .....	66
8.4	Pantallas de acceso .....	68
8.5	Pantalla de registro para nuevos usuarios de la aplicación.....	69
8.6	Pantalla principal .....	69
9	Ampliaciones .....	77
9.1	Agrupamiento de experimentos por campañas.....	77
9.2	Sistema de propietarios múltiples.....	77
9.3	Diferentes algoritmos de balanceo seleccionables .....	77
10	Conclusiones .....	78
11	Referencias.....	79

# Índice de figuras

## Índice de tablas

Tabla 1. Tareas a desarrollar.....	22
Tabla 2. Presupuesto Hardware .....	24
Tabla 3. Presupuesto software .....	25
Tabla 4. Amortización del inmovilizado material .....	25
Tabla 5. Presupuesto mano de obra .....	26
Tabla 6. Presupuesto total.....	26

## Índice de imágenes

Imagen 1. Diagrama de Gantt.....	23
Imagen 2. Mapa de historias de usuario .....	29
Imagen 3. Modelo de dominio.....	37
Imagen 4. Pantalla inicial .....	38
Imagen 5. Pantalla de registro .....	38
Imagen 6. Pantalla de inicio de sesión.....	39
Imagen 7. Pantalla principal .....	39
Imagen 8. Pantalla de creación de experimentos.....	40
Imagen 9. Pantalla de visualización de experimentos .....	40
Imagen 10. Mapa de navegación .....	41
Imagen 11. Esquema de la arquitectura de la aplicación.....	42
Imagen 12. Esquema de la arquitectura del frontend .....	43
Imagen 13. Esquema de la arquitectura de la API.....	44
Imagen 14. Representación física de experimentos.....	45
Imagen 15. Representación física de asignaciones.....	45
Imagen 16. Representación física de usuarios.....	46
Imagen 17. Representación física de clientes .....	46
Imagen 18. Modelo de experimentos público .....	47
Imagen 19. Esquema de JWT .....	51

Imagen 20. Pantalla principal .....	67
Imagen 21. Barra de navegación pública.....	67
Imagen 22. Opciones de acceso.....	67
Imagen 23. Pantalla de inicio de sesión.....	68
Imagen 24. Pantalla de registro .....	69
Imagen 25. Pantalla principal .....	70
Imagen 26. Crear nuevo experimento.....	70
Imagen 27. Cierre de sesión .....	70
Imagen 28. Formulario de creación de experimentos sin asignaciones .....	71
Imagen 29. Formulario de creación de experimentos con asignaciones .....	71
Imagen 30. Lista de experimentos .....	72
Imagen 31. Pantalla con los detalles de un experimento .....	73
Imagen 32. Opciones para realizar sobre un experimento.....	73
Imagen 33. Clave de un experimento .....	74
Imagen 34. Asignación recibida por la API .....	74
Imagen 35. Opción para activar una asignación.....	75
Imagen 36. Asignaciones manuales .....	75
Imagen 37. Formulario de modificación de un experimento.....	76
Imagen 38. Mensaje de confirmación de la eliminación de un experimento .....	76



# 1 Introducción y motivación

## 1.1 Introducción

Con los años, ha habido un enorme aumento en el uso de tiendas online que ofrecen productos y servicios en su web. El maximizar las ventas de estos es una prioridad para estas organizaciones. Una de las formas para lograr este objetivo es la implementación de experimentos de tipo A/B testing.

Las pruebas A/B son un tipo de experimentos utilizados en el marketing digital y la analítica web para identificar cambios en las webs que puedan maximizar un determinado resultado. Por ejemplo, maximizar las ventas de un producto en una tienda online en donde dos usuarios distintos pueden ver contenidos diferentes, como anuncios en diferentes lugares o precios diferentes para el mismo producto. De esta manera se pueden obtener métricas vinculadas a estos experimentos que determinarán la optimalidad de cada opción y finalmente se podrá tomar una decisión en base a los resultados.

Un experimento A/B está formado por un conjunto de activaciones o asignaciones, donde cada una de ellas representa un cambio en la web. En este conjunto se encuentra la activación denominada grupo de control, que no implementa ningún cambio, y varias activaciones que representan los distintos cambios para la web. Cada activación es un par clave-valor, donde la clave es el nombre de la activación (grupo de control, activación 1, activación 2, ...) y el valor es el porcentaje de webs que aplicarán esa activación, al que se le denomina porcentaje de activación.

En este TFG se pretende ofrecer una plataforma para la creación de estos experimentos para pruebas A/B. Mediante la creación de un servicio web que administre los distintos experimentos A/B y permita a las distintas webs consultarlos. La lógica de los cambios a realizar en las webs es completamente abstracta al servicio. Será una activación lo que una web reciba cuando consulte el servicio y en función del valor recibido aplicará un cambio u otro.

El TFG se compone de un servicio web que se encarga de la administración de los experimentos y de almacenarlos en una base de datos y una web que sirve de interfaz para comunicarse con este servicio y así crear y modificar los distintos experimentos A/B. El servicio web es una API RESTful HTTP que se comunica con una base de datos de tipo NoSQL llamada MongoDB. Está programado en Go, por su alto rendimiento, su simplicidad en la sintaxis, su velocidad de programación y compilación y su muy buen soporte para la concurrencia. La web está hecha en angular, un *framework* de JavaScript que simplifica la creación de webs y ayuda a la depuración gracias al uso de TypeScript que ofrece seguridad de tipos sobre JavaScript. La aplicación se meterá en un contenedor Docker y se subirá al Cloud para su puesta en producción.

## 1.2 Motivación

Actualmente no existe una gran diversidad de herramientas que permitan la realización de experimentos de tipo A/B testing de forma simple y sencilla, mientras que la necesidad y demanda de este servicio está en constante aumento con la digitalización de los negocios. Todos los días se crean y ponen en marcha nuevas páginas web que implementan de una u otra forma un modelo de negocio, la optimización de estas páginas web es de gran importancia para sus creadores y es este el principal problema a resolver.

Adicionalmente, este TFG tiene como objetivo el uso de las tecnologías más modernas y punteras para el desarrollo de servicios y aplicaciones web, que sean escalables y no queden obsoletos en el corto plazo.

Finalmente, el proyecto se ha realizado en colaboración con la empresa HP SCDS, lo que implica la necesidad y el objetivo de un desarrollo serio y profesional que se adecue a la forma de trabajar de la empresa y cumpla con los objetivos y requisitos planteados por la misma.

## 2 Objetivos y alcance

El objetivo principal del proyecto consiste en el desarrollo de una aplicación web que permita la creación y utilización de experimentos de tipo A/B. Permitiendo a organismos externos implementar estos experimentos en sus propias webs o productos digitales y así facilitarles la extracción de métricas para optimizar su negocio.

Para ello, el usuario dispone de acceso a una web donde podrá registrarse y configurar sus propios experimentos. Estos experimentos quedarán expuestos públicamente para su consumo en el momento de su creación, el usuario dispone de la URL para consumirlo que será facilitada a las webs o servicios los cuales formarán parte de dicho experimento.

El sistema irá asignando a los usuarios de cada web o servicio participante un grupo del experimento y se mantendrá la consistencia de dicha asignación con el tiempo. Además, las activaciones podrán ser modificadas manualmente por el administrador del experimento en todo momento.

Con todos estos requisitos en mente, el problema a resolver es el de desarrollar un servicio web que se encargue de la administración y puesta en marcha de los experimentos este servicio es lo que se conoce comúnmente como *backend*. El *backend* es la parte de la aplicación que el usuario final no puede ver. Su función es acceder a la información que se solicita, a través de la aplicación, para luego combinarla y devolverla al usuario final. Usualmente es un programa que está en ejecución en un servidor remoto. Además, se desarrollará una página web que permita la creación y modificación de estos a los diferentes usuarios que quieran tener sus experimentos A/B. Esta parte es la que se conoce como *frontend*. El *frontend* es la parte del desarrollo web que se dedica a la interfaz de un sitio web, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos. Es la parte de la aplicación con la que interactúa el usuario final.

## 2.1 Acceso a la aplicación

La aplicación tiene que implementar un sistema de autenticación basado en tokens con caducidad temporal. En concreto la tecnología a utilizar será JWT (JSON Web Tokens), con la que no se requiere que el servidor que almacene ningún tipo de información de sesión de cada usuario. Cada token es firmado por el servidor digitalmente con una clave privada y este solo verificará dicha firma digital en cada petición procedente del cliente. Los usuarios se podrán registrar e iniciar sesión desde la página web, una vez autenticados recibirán del servidor un token de acceso con caducidad temporal y ya podrán realizar todas las operaciones que necesiten sobre sus experimentos A/B.

## 2.2 Experimentos

Los experimentos se conforman de varios atributos, estos son:

- Id: Identificador único de cada experimento.
- Name: Nombre del experimento, único entre los experimentos de cada usuario.
- Description: Una descripción del experimento para ayudar a los usuarios a diferenciarlos. Este atributo es opcional.
- Assignments: El grupo de activaciones que componen el experimento (grupo de control y las distintas variaciones que implementará el experimento).
- Owner: Indica quién el propietario o administrador del experimento.
- ExperimentKey: Clave única que permite el consumo del experimento desde webs o servicios externos. Se generará de forma dinámica.

Para la creación de un experimento es necesario indicar el nombre, una descripción opcional y las distintas activaciones.

## 2.3 Activaciones

Las activaciones se conforman de varios atributos, estos son;

- AssignmentName: Nombre de la activación. Sigue la siguiente secuencia obligatoria: C, A1, A2, A3, ....

- **AssignmentValue:** Porcentaje de activación para su asignación. Dependiendo del porcentaje dado a cada usuario que participe en el experimento, este pertenecerá a una activación u otra.
- **AssignmentDescription:** Una descripción para ayudar a los usuarios a diferenciar los distintos grupos. Este atributo es opcional.

La suma de todos los **AssignmentValue** de las activaciones que forman parte de un experimento ha de ser 100. Sin embargo, una activación puede encontrarse desactivada teniendo un valor de 0. Por el contrario, el experimento puede desactivarse dejando el grupo de control con valor de 100.

## 2.4 Algoritmo de asignación a los grupos del experimento

El algoritmo que decide que activación va a asignar a cada usuario que participe en el experimento es el componente más importante del servicio web. Es importante que, si en un futuro los porcentajes de cada asignación son modificados, las nuevas asignaciones se vayan balanceando hasta quedar en equilibrio con los nuevos porcentajes. Es por esto por lo que el algoritmo debe otorgar las asignaciones cumpliendo los valores de estas solo cuando los usuarios que participan en el experimento cumplen la misma distribución que los porcentajes teóricos de dicho experimento.

En caso de haber un desbalance entre los porcentajes teóricos del experimento y los que están actualmente repartidos entre los participantes, el algoritmo tiene que forzar poco a poco un rebalanceo de manera que durante este periodo no se respetan del todo los valores de las asignaciones. Estos desbalances ocurren cuando el administrador del experimento decide cambiar el porcentaje de activación de las asignaciones que componen el experimento cuando este ya contaba con un grupo de participantes.

Por todas estas razones, el algoritmo, en caso de desbalance, asignará a los nuevos participantes la asignación que cuente con el error absoluto mayor y positivo entre el porcentaje teórico del experimento y el real. De esta forma se irá corrigiendo poco a poco aquellas asignaciones más desbalanceadas después de una modificación de los valores del experimento, ya que, al ir asignando la asignación con mayor error absoluto, esta irá poco a poco disminuyendo el error hasta que sea igual a cero u otra asignación pase a tener el mayor error absoluto.

## 2.5 Consumir los experimentos creados

Una vez creado un experimento se puede empezar a participar en él. Para ello quedará expuesta una URL a la que poder realizar peticiones y será el servicio web el que devolverá una nueva asignación para cada usuario.

La URL está compuesta de dos partes, la primera es la dirección de la API con la clave del experimento a utilizar y la segunda es una cadena de texto a elección del participante que será el que lo identifique, es decir, futuras peticiones con la misma cadena de texto tendrán consistencia en el tiempo. El uso de una nueva cadena de texto que el experimento no tenga registrada se tratará como un nuevo participante de este. De esta forma, cada web externa podrá utilizar a conveniencia la cadena de texto que considere por cada usuario que tenga.

### 3 Estudios y análisis previos

En este apartado se abordará la descripción de distintas aplicaciones existentes en el mercado, así como la comparación entre las distintas alternativas barajadas para poder llevar a cabo el TFG.

#### 3.1 Aplicaciones similares en el mercado

##### 3.1.1 Firebase A/B Testing

Firebase A/B Testing [1] cuenta con la tecnología de Google Optimize, ayuda a optimizar tu experiencia con las aplicaciones y facilita la ejecución, el análisis y el escalado de los experimentos de productos y de marketing. Firebase A/B Testing funciona con Firebase Cloud Messaging para que puedas probar diferentes mensajes de marketing, y con Remote Config, para que puedas probar los cambios en la aplicación. La principal desventaja se encuentra en la complejidad del uso de estos servicios y que estás bloqueado al uso de la plataforma de Google Cloud.

##### 3.1.2 Optimizely

Optimizely [2] es un conjunto de herramientas de CRO que se dedica principalmente a grandes clientes a nivel empresarial. CRO, en inglés *Conversion Rate Optimization*, es una de las estrategias de marketing más usadas actualmente. Optimizely se centra en ofrecer servicios de personalización y experimentación Web. Sin embargo, sus servicios pueden ejecutar experimentos en aplicaciones móviles y plataformas de mensajería. Su servicio de A/B testing puede ejecutar simultáneamente múltiples pruebas en la misma página. No obstante, Optimizely solo es adecuado para empresas muy grandes que tienen equipos de experimentación dedicados. Además, su plataforma de A/B testing no es gratuita.

##### 3.1.3 Adobe Target

Adobe Target [2] es una herramienta empresarial que proporciona servicios de A/B testing, ofrece muy buena integración con Google Analytics y puede generar informes para pruebas de experiencia de usuario. Esta herramienta es más adecuada para clientes que utilizan todo el paquete de marketing de Adobe ya que ofrece una integración completa entre sus productos y servicios. Su principal desventaja es que esta herramienta

está disponible solamente con la suscripción a Adobe que te permite utilizar muchos servicios a parte de este que igual el usuario no necesita.

## 3.2 Tecnologías para el *backend*

### 3.2.1 C# y .Net

C# es uno de los lenguajes de programación más populares para el desarrollo *backend* en la actualidad. Una de sus mejores características es el manejo de la automatización basada en servidor a través del *framework* ASP.Net. Sin embargo, hay muchas funcionalidades que requieren que el código se ejecute sobre sistemas Windows, limitando algunas de sus capacidades.

### 3.2.2 Java

Java es otro de los lenguajes de programación más populares para el desarrollo de tecnologías *backend*. Cuenta con un enorme número de bibliotecas de código abierto muy extensas. No obstante, la programación en java requiere más tiempo que con otras alternativas y consume muchos más recursos hardware que estas.

### 3.2.3 Go

Go, también conocido como Golang, es un lenguaje de programación de código abierto creado por Google. A pesar de ser un lenguaje muy joven tiene un rendimiento similar a C, pero con la sintaxis amigable parecida a Python, actualmente es utilizado en aquellos programas que requieran alto rendimiento. Facilita el uso de buenas prácticas a la hora de programar, mantiene un excelente rendimiento con grandes volúmenes de información, tiene una curva de aprendizaje menor que alternativas como Java y está muy centrado en la programación paralela y concurrente a la vez que en la simplicidad para programar en él. Su principal uso se centra en el desarrollo de *backend*, para lo que está muy optimizado.

### 3.2.4 JavaScript

JavaScript se ha mantenido durante varios años consecutivos como el lenguaje de programación más popular entre los desarrolladores según diferentes encuestas de StackOverflow [5]. JavaScript ha evolucionado de un lenguaje de cliente *frontend* exclusivo a manejar también el desarrollo de *backend*. Node.js y otros *frameworks* permiten que el código de JavaScript se ejecute en el *backend*. No obstante JavaScript es



un lenguaje de programación interpretado, lo que le hace tener menor rendimiento que sus competidores.

### 3.3 Tecnologías para el *frontend*

#### 3.3.1 Angular

Lanzado en 2010 por Google, es un *framework* JavaScript para la creación de páginas web. Su principal ventaja es que incluye todas las herramientas necesarias para el desarrollo de páginas web con cierto grado de complejidad sin necesidad de depender de librerías externas.

#### 3.3.2 React

Lanzado en 2013 por Facebook, es un *framework* JavaScript que se centra en sitios web de alto tráfico de usuarios. Es en la actualidad la alternativa más utilizada en el desarrollo *frontend*. Además, está considerado como la alternativa para desarrollo *frontend* con menor curva de aprendizaje.

#### 3.3.3 Vue

Lanzado en 2014, desarrollado por Evan You, es un *framework* JavaScript progresivo. Está ganando mucha popularidad en los últimos años. Es una alternativa mucho más ligera que el resto, ocupando tan solo 80kb.

### 3.4 Motores de base de datos

#### 3.4.1 PostgreSQL

PostgreSQL es un sistema de base de datos relacional de código abierto con más de 30 años de desarrollo activo, cuenta con una sólida reputación por su fiabilidad, polivalencia de funciones, gran rendimiento y muy buena escalabilidad.

#### 3.4.2 MySQL

MySQL fue inicialmente desarrollado en 1995 por MySQL AB, empresa adquirida por Sun Microsystems y que luego paso a ser propiedad de Oracle Corporation. Es un motor de base de datos de tipo relacional. Sus principales ventajas son su velocidad y bajo consumo de recursos.

### **3.4.3 MongoDB**

MongoDB es un sistema de base de datos NoSQL orientado a documentos de código abierto, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico. Es mucho más flexible que motores de bases de datos relacionales y su rendimiento es muy superior a estos si el conjunto de datos no está altamente relacionado entre sí.

## **3.5 Control de versiones en remoto**

### **3.5.1 GitHub**

Es el sitio más popular en la actualidad para almacenar repositorios Git. El sistema está diseñado para permitir a los usuarios crear fácilmente sistemas de control de versiones basados en Git.

### **3.5.2 GitLab**

El servicio también está desarrollado en la base del control de versiones de Git. A pesar de que la funcionalidad de GitLab es similar a su principal competidor, GitHub, este tiene mejores herramientas CI/CD que son muy valoradas en el entorno empresarial.

### **3.5.3 Bitbucket**

Este servicio también es muy similar a los previos indicados e implementa la mayoría de sus características con ligeras diferencias. Bitbucket está mejor orientado a los equipos de desarrollo profesional, ya que proporciona grandes beneficios para ellos, como una integración con Jira, revisión de código avanzado. Al mismo tiempo, con el crecimiento del equipo, Bitbucket ofrece condiciones de precios más adecuadas comparadas con GitHub y GitLab.

### 3.6 Elección final

Para la elección de las tecnologías ha tenido un enorme peso las preferencias y forma de trabajar de la empresa colaboradora con este trabajo fin de grado, HP SCDS.

Para el *backend* la elección ha sido el lenguaje de programación Go, por las ventajas indicadas anteriormente, además de haber sido la principal propuesta de HP SCDS para el desarrollo del servicio web. En conjunto, se utilizará una librería llamada Gin para simplificar el desarrollo del servicio web.

Para el *frontend* se utilizará Angular, ya que incluye todo el conjunto de herramientas necesario para el desarrollo de una interfaz de usuario profesional y escalable sin tener que depender de librerías externas que puedan suponer problemas de compatibilidad.

La base de datos a utilizar es MongoDB, ya que el conjunto de datos no tiene un alto grado de interrelación, pero es necesario un alto nivel de flexibilidad. El usar una base de datos NoSQL simplifica mucho el diseño de la base de datos para este proyecto.

Finalmente, el servicio de control de versiones en remoto a utilizar es GitLab. Es el servicio utilizado para trabajar en la empresa colaboradora HP SCDS, ya que cuenta con un muy buen sistema CI/CD. Además, HP SCDS cuenta con servidores para compilación y ejecución de pruebas ligado a este servicio que se van a utilizar en el desarrollo del proyecto.

## 4 Planificación

### 4.1 Metodología utilizada

Para este proyecto se ha optado por utilizar una metodología ágil para su desarrollo, concretamente se ha utilizado SCRUM. La metodología Scrum es un proceso para llevar a cabo un conjunto de tareas de forma regular con el objetivo principal de trabajar de manera colaborativa, es decir, para fomentar el trabajo en equipo. Con este método de trabajo lo que se pretende es alcanzar el mejor resultado de un proyecto determinado.

Por ello, la planificación del proyecto se divide en 10 sprints, de 2 semanas de duración. Al final de cada sprint se dispone de un entregable completo y autosuficiente que se somete a una retrospectiva.

### 4.2 Tabla de las tareas a desarrollar

En la tabla 1 se muestran las historias de usuario a realizar, junto con la estimación de su duración y fecha de su realización. Así como las tareas dependientes de otras.

ID	SPRINT	TAREA	DURACIÓN	COMIENZO	FIN	PREDECE_ SORAS
1	1	Diseñar una API básica para el backend	14 días	20/1/22	3/2/22	-
2	2	Explorar el uso de Swagger en la API	5 días	3/2/22	8/2/22	-
3	3	Sistema de registro en la aplicación	4 días	3/3/22	7/3/22	4
4	4	Sistema de autenticación en la aplicación	5 días	7/3/22	12/3/22	4

5	4	Validación de experimentos	5 días	12/3/22	17/3/22	4
6	5	Eliminación de experimentos	4 días	17/3/22	21/3/22	7
7	5	Listado de experimentos	5 días	21/3/22	26/3/22	7
8	5	Creación de experimentos nuevos	5 días	26/3/22	31/3/22	7
9	6	Restricción en los nombres duplicados de experimentos	3 días	31/3/22	3/4/22	9, 10
10	6	Eliminación dinámica de asignaciones	4 días	3/4/22	7/4/22	10
11	6	Descripción en los experimentos	4 días	7/4/22	11/4/22	-
12	6	Descripción en las asignaciones	3 días	11/4/22	14/4/22	-
13	7	Modificación de los experimentos	2 días	14/4/22	16/4/22	7
14	7	Sistema de balanceo	12 días	16/4/22	28/4/22	10, 15
15	8	Activar una sola asignación	3 días	28/4/22	1/5/22	15
16	8	Asignaciones manuales	3 días	1/5/22	4/5/22	15
17	8	Deshabilitar experimentos	4 días	4/5/22	8/5/22	15

<b>18</b>	8	Pedir asignaciones para webs externas	4 días	8/5/22	12/5/22	10
<b>19</b>	9	Web de ejemplo	10 días	12/5/22	22/5/22	20
<b>20</b>	9	Listar las asignaciones manuales	4 días	22/5/22	26/5/22	18
<b>21</b>	10	Eliminar asignaciones manuales	6 días	26/5/22	1/6/22	22
<b>22</b>	10	Notificar del balanceo asíncrono	3 días	1/6/22	4/6/22	15
<b>23</b>	10	Hacer segura la concurrencia	5 días	4/6/22	-	15

Tabla 1. Tareas a desarrollar

4.3 Diagrama de Gantt

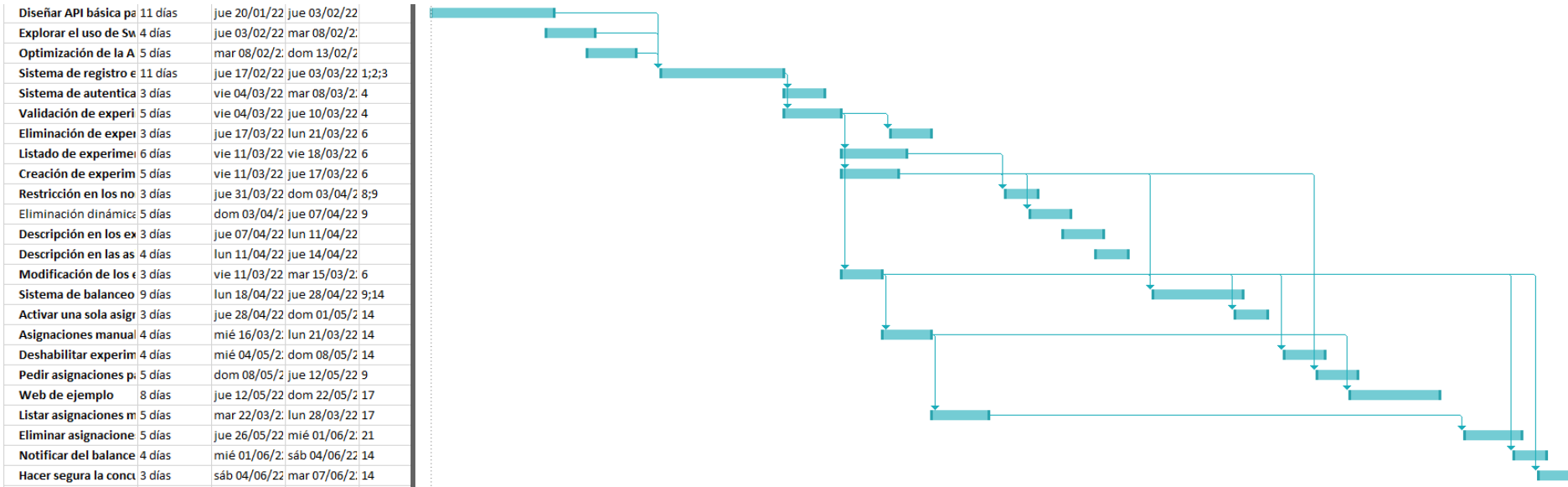


Imagen 1. Diagrama de Gantt

# 5 Presupuesto

En este apartado se detalla el presupuesto correspondiente al desarrollo de este proyecto. El presupuesto se desglosa en 5 apartados: Hardware, Software, amortización del inmovilizado material, mano de obra y finalmente el presupuesto total.

## 5.1 Presupuesto Hardware

En este apartado quedan reflejados todos los elementos hardware necesarios para la realización del proyecto. Todo el equipamiento necesario se encuentra en la siguiente tabla.

ID	Descripción	Unidad	Cantidad	Precio unitario (sin IVA)	Total (sin IVA)
6.1.1	PC clónico (Ryzen 7 2700X, 16GB RAM, 1TB SSD)	U	1	1500,00€	1.500,00€
6.1.2	Monitor Samsung 24”	U	2	120,00€	240,00€
6.1.3	Ratón USB Logitech	U	1	30,00€	30,00€
6.1.4	Teclado USB KROM Kernel	U	1	40,00€	40,00€
Total					1.810,00€

Tabla 2. Presupuesto Hardware

El presupuesto correspondiente al Hardware requerido para el proyecto es de MIL OCHOCIENTOS DIEZ EUROS.



## 5.2 Presupuesto Software

En este apartado se desglosa el coste de todas las licencias y herramientas software necesarias para el desarrollo del proyecto. Todas las herramientas necesarias se encuentran en la siguiente tabla.

ID	Descripción	Unidad	Cantidad	Precio unitario (sin IVA)	Total (sin IVA)
6.2.1	Licencia de Microsoft Windows 10 Pro. 64 bits	U	1	259,00€	259,00€
6.2.2	Licencia Microsoft Office 365	U/año	1	69,00€	69,00€
6.2.3	Licencia Microsoft Project 2019	U	1	15,00€	15,00€
<b>Total</b>					343,00€

Tabla 3. Presupuesto software

El presupuesto correspondiente a las licencias de Software requeridas para el proyecto asciende a TRESCIENTOS CUARENTA Y TRES EUROS.

## 5.3 Amortización del inmovilizado material

Material	Coste de adquisición	Tiempo de amortización	Tiempo de uso	Amortización
Hardware	1810,00€	7 años	6 meses	129,29€
Software	343,00€	4 años	6 meses	42,88€
<b>Total</b>				172,17€

Tabla 4. Amortización del inmovilizado material

El total correspondiente a la amortización del inmovilizado material asciende a CIENTO SETENTA Y DOS EUROS Y DIECISIETE CENTIMOS.

## 5.4 Presupuesto de mano de obra

En este apartado se ve reflejado el coste relativo a la mano de obra requerida para el desarrollo del proyecto.

ID	Descripción	Unidad	Cantidad	Precio unitario (sin IVA)	Total (sin IVA)
6.4.1	Horas de arquitecto de software	Horas	40	25,00€	1.000,00€
6.4.2	Horas de desarrollador backend en Go	Horas	150	20,00€	3.000,00€
6.4.3	Horas de desarrollador web en Angular	Horas	100	20,00€	2.000,00€
	Horas de scrum máster HP	Horas	40	30,00€	1.200,00€
	Horas Tutora Universidad de Oviedo	Horas	40	30,00€	1.200,00€
<b>Total</b>					8.400,00€

Tabla 5. Presupuesto mano de obra

El precio total de la mano de obra del proyecto asciende a OCHO MIL CUATROCIENTOS EUROS.

## 5.5 Presupuesto total

Finalmente se presenta el coste total para el desarrollo y realización del proyecto.

Descripción	Coste
<b>Total inmovilizado material</b>	172,17€
<b>Mano de obra</b>	8.400,00€
<b>Gastos generales</b>	1.114,38€
<b>Beneficio industrial</b>	514,33€
<b>Precio total (sin IVA)</b>	10.200,88€
<b>Coste total (con IVA)</b>	12.343,06€

Tabla 6. Presupuesto total

El presupuesto total del proyecto es de DOCE MIL TRESCIENTOS CUARENTA Y TRES EUROS Y SEIS CENTIMOS.

## 6 Análisis

### 6.1 Requisitos de usuario

#### 6.1.1 Requisitos funcionales

1. El usuario podrá crear un experimento a través del interfaz web
  - 1.1 El experimento tendrá un nombre único en el sistema
  - 1.2 El experimento tendrá un conjunto de asignaciones
    - 1.2.1 Cada asignación tendrá un nombre
    - 1.2.2 Cada asignación tendrá un porcentaje de activación
    - 1.2.3 La suma de los porcentajes de activación tiene que ser 100
    - 1.2.4 Cada asignación tendrá una descripción
  - 1.3 El experimento tendrá una descripción
2. El usuario podrá ver una lista con sus experimentos en la interfaz web
  - 2.1 Se mostrará la lista completa de todos los experimentos pertenecientes al usuario
    - 2.1.1 La lista mostrará solo el nombre de los experimentos
3. El usuario podrá eliminar sus experimentos en la interfaz web
  - 3.1 Se mostrará una opción correspondiente a cada experimento para su eliminación
    - 3.1.1 Se pedirá una confirmación de eliminación
4. El usuario podrá modificar sus experimentos en la interfaz web
  - 4.1 El usuario podrá modificar todos los campos de los experimentos
  - 4.2 Cuando se cambie la asignación de las activaciones se intentará mantener la asignación en la medida de lo posible para las parejas clave/tratamiento ya generadas, respetando los nuevos porcentajes.
  - 4.3 El usuario podrá añadir nuevas asignaciones al experimento
  - 4.4 El usuario podrá eliminar asignaciones del experimento
5. El usuario podrá deshabilitar el experimento en la interfaz web
  - 5.1 Un experimento deshabilitado tendrá el grupo de control con el porcentaje de activación a 100 y el resto de las asignaciones a 0
6. EL usuario podrá activar una asignación en exclusiva para un experimento en la interfaz web

- 6.1 Habrá una opción que permita la selección de una de las asignaciones del experimento.
- 6.2 La asignación seleccionada pasará a tener un porcentaje de activación de 100 y el resto pasarán a 0
- 7. Un usuario externo que utilice el servicio de experimentación siempre deberá obtener el mismo tratamiento
  - 7.1 El proyecto incluirá un servicio web que expondrá una API que permitirá obtener el valor del experimento para una determinada clave única.
- 8. El usuario podrá crear una excepción para una clave desde la interfaz web.
  - 8.1 El usuario podrá crear una excepción para una determinada clave, asignando un tratamiento específico a su elección.

#### **6.1.2 Requisitos no funcionales**

- 1. Seguridad
  - 1.1 La aplicación será accesible mediante credenciales de usuario.
  - 1.2 Los usuarios podrán estar autenticados como máximo 24 horas.
- 2. Validación de formularios
  - 2.1 Todos los formularios validarán sus campos en tiempo real.
- 3. Interfaces de usuario
  - 3.1 Las interfaces de usuario han de ser intuitivas y amigables al usuario final.
- 4. Pruebas
  - 4.1 La aplicación contará con un sistema de integración continua que ejecutará las pruebas en cada *commit* al repositorio.

## 6.2 Mapa de historias de usuario

En la siguiente imagen se puede observar la distribución del mapa de las historias, divididas en las categorías, diseño de la API, diseño de la interfaz de usuario y gestión de usuarios.

	Diseño de la API	Diseño de la interfaz de usuario			Gestión de usuarios
Sprint 1	Diseñar una API básica para el <i>backend</i>				
Sprint 2	Explorar el uso de Swagger en la API				
Sprint 3	Optimización de la API				
Sprint 4	Validación de los experimentos	Sistema de registro en la aplicación	Sistema de autenticación en la aplicación		Sistema de gestión de usuarios
Sprint 5		Eliminación de experimentos	Listado de experimentos	Creación de experimentos	
Sprint 6	Restricción en los nombres duplicados de experimentos	Eliminación dinámica de asignaciones	Descripción en los experimentos	Descripción en las asignaciones	
Sprint 7	Sistema de balanceo de asignaciones	Modificación de los experimentos			
Sprint 8	Pedir asignaciones para webs externas	Activar una sola asignación	Asignaciones manuales	Deshabilitar experimentos	
Sprint 9		Web de ejemplo	Listar asignaciones manuales		
Sprint 10		Eliminar asignaciones manuales	Notificar del balanceo asíncrono	Hacer segura la concurrencia	

Imagen 2. Mapa de historias de usuario

## 6.3 Historias de usuario

### 6.3.1 Diseñar una API básica para el *backend*

Como desarrollador quiero tener un diseño de la API básico en Go con las operaciones básicas sobre experimentos.

Criterios de aceptación:

- Crear la estructura del nuevo proyecto en Go.
- Crear las operaciones básicas de creación, modificación, eliminación y listado de los experimentos.
- Conectar la API con la base de datos.

### 6.3.2 Explorar el uso de Swagger en la API

Como desarrollador quiero explorar la idea de incluir Swagger para la documentación del *backend*.

Criterios de aceptación:

- Leer la documentación oficial de Swagger y OpenAPI.
- Investigar las librerías que permiten su inclusión en proyectos de Go.

### 6.3.3 Aplicar los cambios hablados de la sesión anterior

Como desarrollador quiero implementar los distintos cambios y arreglos discutidos en la reunión de retrospectiva del sprint anterior.

Criterios de aceptación:

- Los tokens JWT deben tener expiración.
- Implementación del patrón repositorio dentro de la base de datos.
- Uso de entidades diferentes para la API y el modelo.
- Cargar la configuración desde un archivo, terminar la ejecución si falla.
- Proporcionar un archivo de ejemplo para la configuración.
- Utilizar la inyección de dependencia.

### 6.3.4 Sistema de registro en la aplicación

Como usuario quiero poder registrarme en la aplicación.

Criterios de aceptación:

- Implementar un sistema para crear y registrar usuarios nuevos.
- Implementar el modelo de usuario dentro de la aplicación y la base de datos.
- Crear una pantalla para el registro de nuevos usuarios en la aplicación.

### 6.3.5 Sistema de autenticación en la aplicación

Como usuario quiero autenticarme en la aplicación.

Criterios de aceptación:

- Implementar un sistema de autenticación en la aplicación utilizando JWT.
- Crear una pantalla de log in.

### 6.3.6 Validación de experimentos

Como desarrollador quiero validar los experimentos creados por los usuarios.

Criterios de aceptación:

- Implementar un sistema que valide que los experimentos en la API.
- Implementar un sistema que valide los experimentos en el *frontend*.

### 6.3.7 Eliminación de experimentos

Como usuario quiero eliminar un experimento.

Criterios de aceptación:

- Implementar la función completa de la eliminación de experimentos.
- Crear una opción en la interfaz de usuario para la eliminación de experimentos.

### 6.3.8 Listado de experimentos

Como usuario quiero poder ver todos mis experimentos creados.

Criterios de aceptación:

- Implementar la función completa de listado de los experimentos pertenecientes al usuario.
- Crear una sección en la interfaz de usuario para listar los experimentos.

### 6.3.9 Creación de experimentos nuevos

Como usuario quiero crear experimentos nuevos.

Criterios de aceptación:

- Implementar la funcionalidad completa de creación de experimentos nuevos.
- Crear una pantalla para la creación de nuevos experimentos.

### 6.3.10 Restricción en los nombres duplicados de experimentos

Como usuario no quiero que haya nombres duplicados entre mis experimentos.

Criterios de aceptación:

- Desarrollar un sistema que impida la creación de experimentos con el mismo nombre de uno ya existente entre los pertenecientes al usuario.

### 6.3.11 Eliminación dinámica de asignaciones

Como usuario quiero tener la posibilidad de eliminar asignaciones de forma dinámica mientras se crea un experimento.

Criterios de aceptación:

- Implementar un formulario dinámico que permita eliminar asignaciones del experimento a crear.

### 6.3.12 Descripción en los experimentos

Como usuario quiero tener descripciones en los experimentos.

Criterios de aceptación:

- Añadir un apartado opcional en el proceso de creación de experimentos para la descripción.

Mostrar las descripciones de los experimentos en su listado.

### 6.3.13 Descripción en las asignaciones

Como usuario quiero tener descripciones en las asignaciones de los experimentos.

Criterios de aceptación:



- Añadir un apartado opcional en el proceso de creación de experimentos para añadir una descripción a las asignaciones.
- Mostrar las descripciones de las asignaciones junto a los detalles del experimento.
- Permitir modificar las descripciones de las asignaciones.

#### **6.3.14 Modificación de los experimentos**

Como usuario quiero modificar mis experimentos.

Criterios de aceptación:

- Implementar la funcionalidad completa de modificación de experimento.
- Crear una pantalla para la modificación de experimentos.

#### **6.3.15 Sistema de balanceo**

Como cliente quiero un sistema de balanceo que mantenga las asignaciones otorgadas cuando el experimento en el que participo sea modificado.

Criterios de aceptación:

- Implementar un algoritmo que balancee las asignaciones cuando se modifican los porcentajes de un experimento.
- Minimizar las reasignaciones de los clientes.

#### **6.3.16 Activar una sola asignación**

Como usuario quiero activar una única asignación del experimento.

Criterios de aceptación:

- Implementar la funcionalidad de activar únicamente una asignación.
- Incluir una opción en la interfaz de usuario que permita activar únicamente una asignación.

#### **6.3.17 Asignaciones manuales**

Como usuario quiero poder cambiar la asignación de forma manual para un ID específico.

Criterios de aceptación:

- Implementar un sistema que sobrescriba las asignaciones para un ID específico.
- Impedir que el algoritmo de balanceo modifique las asignaciones manuales.
- Incluir esta opción en la interfaz de usuario.

#### **6.3.18 Deshabilitar experimentos**

Como usuario quiero deshabilitar un experimento.

Criterios de aceptación:

- Implementar una funcionalidad que desactive todas las asignaciones dejando el grupo de control con el 100%.
- Incluir esta opción en la interfaz de usuario.

#### **6.3.19 Pedir asignaciones para webs externas**

Como cliente quiero pedir asignaciones para participar en un experimento para mi web.

Criterios de aceptación:

- Exponer una ruta de la API para la creación de asignaciones a clientes.
- Mostrar la misma asignación para un mismo cliente.
- Permitir a los clientes escoger el ID a utilizar.

#### **6.3.20 Web de ejemplo**

Como defensor de TFG quiero tener una web externa de pruebas que implemente un experimento de esta aplicación.

Criterios de aceptación:

- Crear una página web que implemente un experimento A/B testing creado en la aplicación.
- Hacer que la web muestre la utilidad de estos experimentos.

#### **6.3.21 Listar las asignaciones manuales**

Como usuario quiero poder ver las asignaciones manuales de un experimento.

Criterios de aceptación:

- Implementar la funcionalidad de ver las asignaciones manuales de los experimentos.
- Incluir en la interfaz de usuario la opción para ver las asignaciones manuales.

#### **6.3.22 Eliminar asignaciones manuales**

Como usuario quiero poder eliminar las asignaciones manuales.

Criterios de aceptación:

- Implementar la funcionalidad para eliminar las asignaciones manuales de un experimento.
- Incluir la nueva opción en la interfaz de usuario.
- Al eliminar una asignación manual el ID al que pertenece deberá quedar libre para una nueva asignación.

#### **6.3.23 Notificar del balanceo asíncrono**

Como usuario quiero ser notificado del balanceo en segundo plano a la hora de modificar experimentos.

Criterios de aceptación:

- Incluir un aviso que notifique a los usuarios de que el balanceo ocurre en segundo plano de forma asíncrona.

#### **6.3.24 Hacer segura la concurrencia**

Como desarrollador quiero tener seguridad en el proceso concurrente de reasignación.

Criterios de aceptación:

- Implementar el sistema de reasignación de manera concurrente.
- El sistema debe ser seguro y no modificar asignaciones ya modificadas.
- El sistema debe asegurar que las modificaciones son atómicas.

#### **6.3.25 Compartir experimentos**

Como usuario quiero autorizar a otros usuarios para ver y modificar mis experimentos.

Criterios de aceptación:

- Implementar un sistema de propiedad múltiple para los experimentos.
- Los usuarios poseedores de un experimento deben poder ver y modificar dichos experimentos.
- Un experimento debe poder tener un número ilimitado de poseedores.
- Cualquier cambio realizado sobre un experimento ha de ser visible por el resto de los usuarios poseedores de dicho experimento.

### **6.3.26 Agrupar experimentos**

Como usuario quiero tener mis experimentos organizados en distintas campañas u objetivos.

Criterios de aceptación:

- Todos los experimentos deben pertenecer a una campaña.
- Los experimentos deben mostrarse en la interfaz de usuario organizados por las campañas a las que pertenecen.
- Las campañas deben permitir tener un número ilimitado de experimentos.
- Un experimento solo puede formar parte de una única campaña.

### **6.3.27 Explorar el uso de tecnologías Cloud**

Como desarrollador quiero explorar el uso de tecnologías Cloud para fomentar la escalabilidad y redundancia del proyecto.

Criterios de aceptación:

- Investigar sobre las distintas tecnologías disponibles para incluir en el proyecto.
- Incluir el uso de dichas tecnologías de manera justificada en el proyecto.

## 6.4 Modelo de dominio

A continuación, se muestra el modelo de dominio de la aplicación.

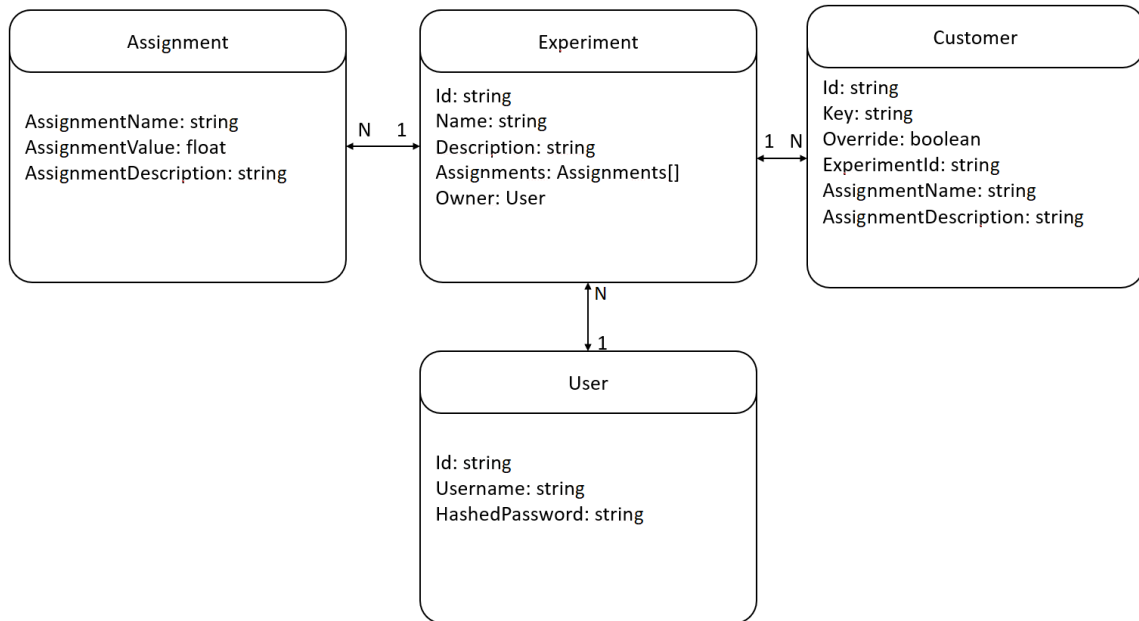


Imagen 3. Modelo de dominio

El modelo se compone por cuatro entidades, los experimentos (Experiment), las asignaciones (Assignment), los clientes (Customer) y los usuarios (User). Los experimentos contienen varias asignaciones lo que provoca una relación de 1 a N siendo N el número de asignaciones máximo que en este caso es infinito. Los experimentos son de un único usuario lo que genera una relación de 1 a N siendo N el número de experimentos que puede tener un usuario. Finalmente, los clientes participan en un experimento pudiendo un experimento tener varios participantes, de esta forma la relación es de 1 a N siendo N el número de clientes que participan en el experimento.

## 6.5 Prototipos de pantalla

### 6.5.1 Pantalla inicial

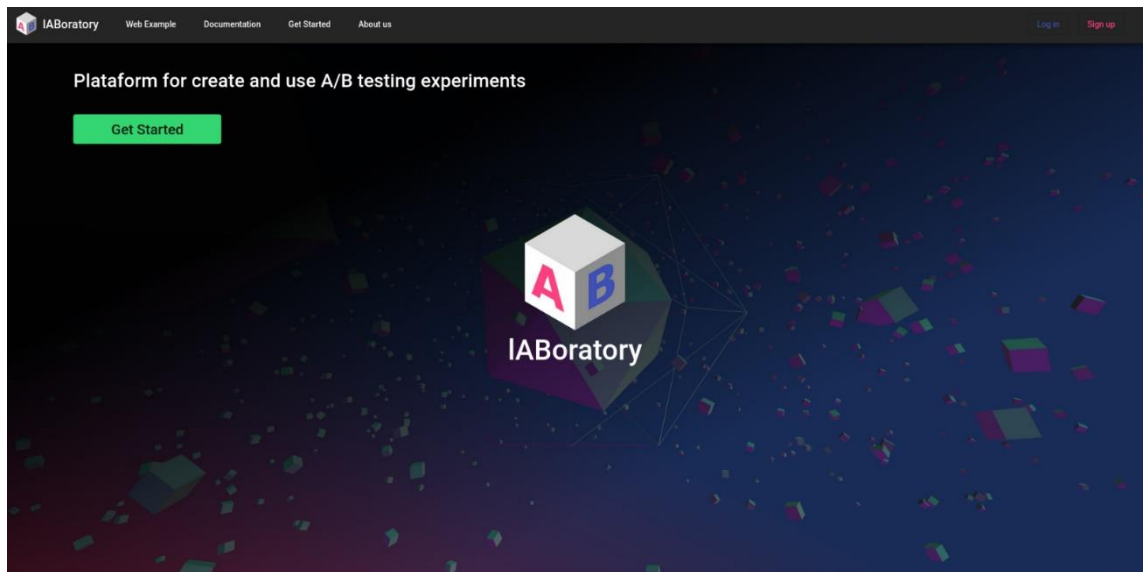


Imagen 4. Pantalla inicial

La pantalla inicial (Imagen 4) contiene las opciones de registro y log in y un enlace a una web de ejemplo que implementa un experimento de la plataforma.

### 6.5.2 Pantalla de registro de nuevos usuarios

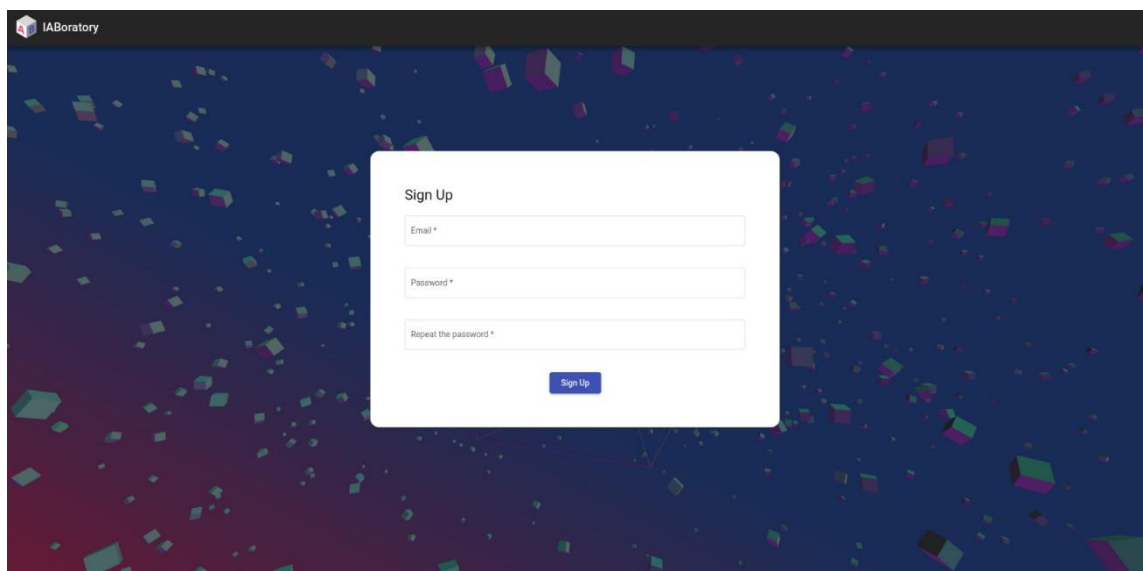


Imagen 5. Pantalla de registro

La pantalla de registro de nuevos usuarios (Imagen 5) consiste en un formulario de registro donde se pide un email, una contraseña y la confirmación de dicha contraseña.

### 6.5.3 Pantalla de inicio de sesión

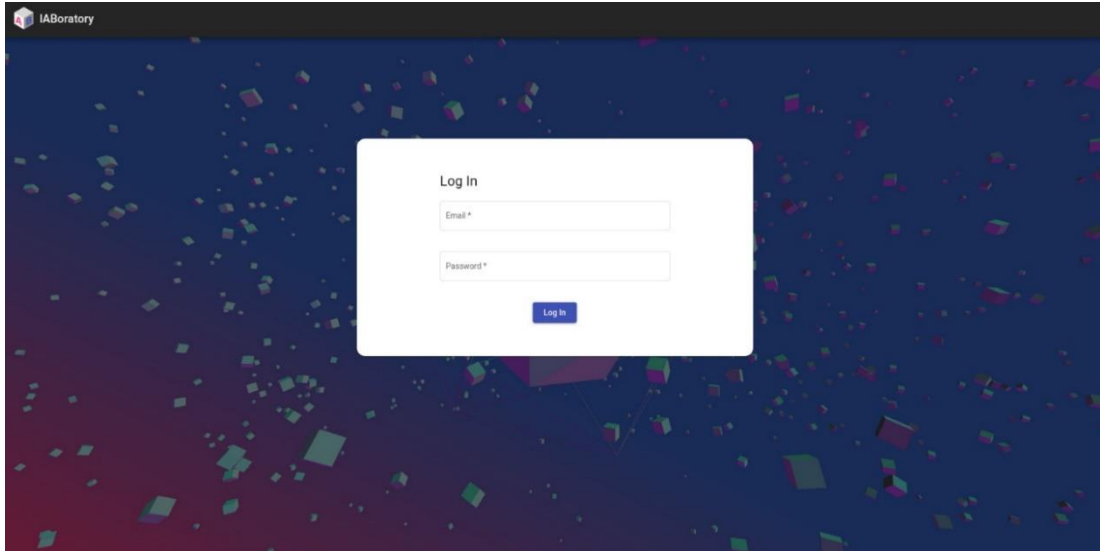


Imagen 6. Pantalla de inicio de sesión

La pantalla de inicio de sesión (Imagen 6) consiste en un formulario donde se pide el email del usuario y su contraseña.

### 6.5.4 Pantalla principal

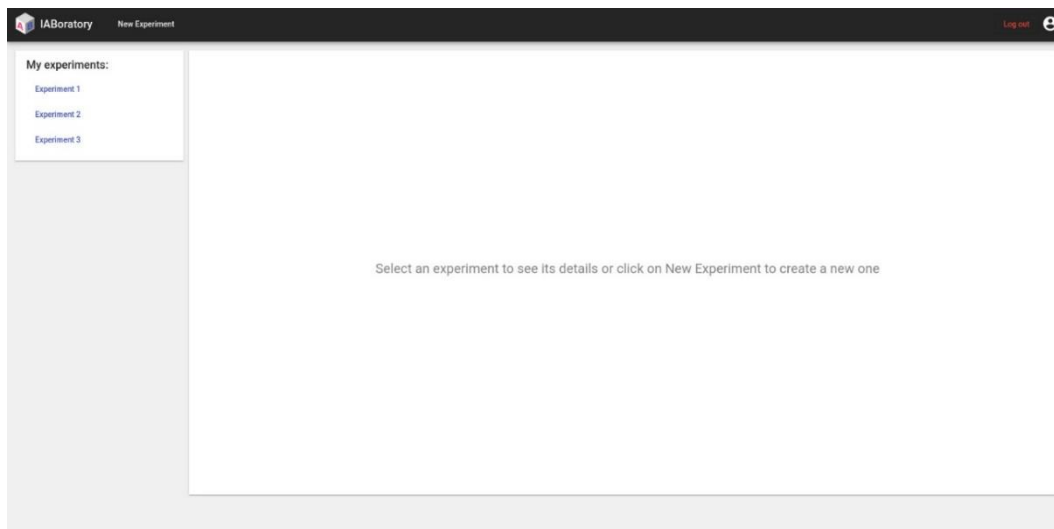


Imagen 7. Pantalla principal

La pantalla principal de la aplicación (Imagen 7) se conforma de tres partes. La barra de navegación superior contiene las opciones de crear un nuevo experimento y la opción de cerrar la sesión actual. La segunda parte es la lista de experimentos del usuario que se encuentra en el lateral izquierdo de la pantalla. Finalmente, en la zona central tenemos un espacio que mostrará diferente información dependiendo de la opción seleccionada.

### 6.5.5 Pantalla de creación de experimentos

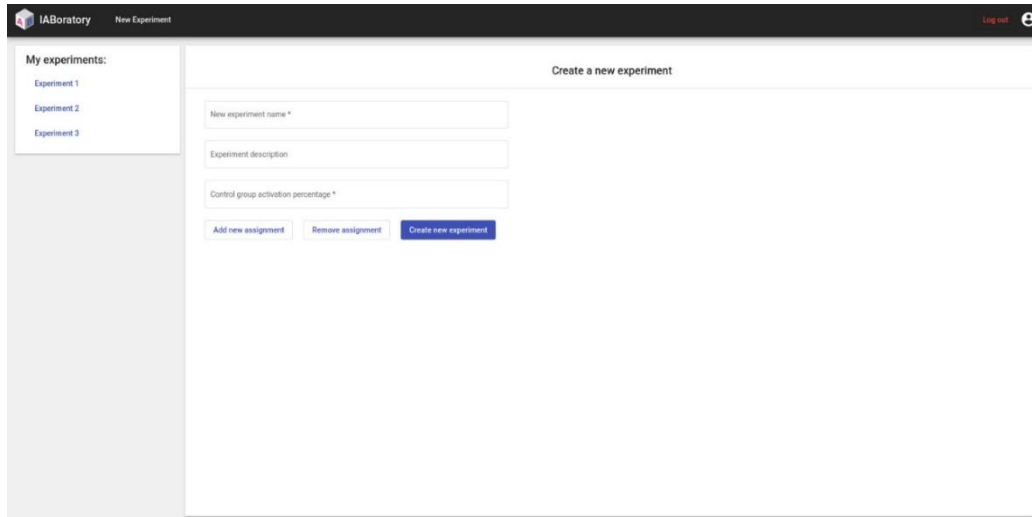


Imagen 8. Pantalla de creación de experimentos

La pantalla de creación (Imagen 8) contiene un formulario para la creación de experimentos nuevos con campos dinámicos para generar las asignaciones deseadas.

### 6.5.6 Pantalla de visualización de experimentos

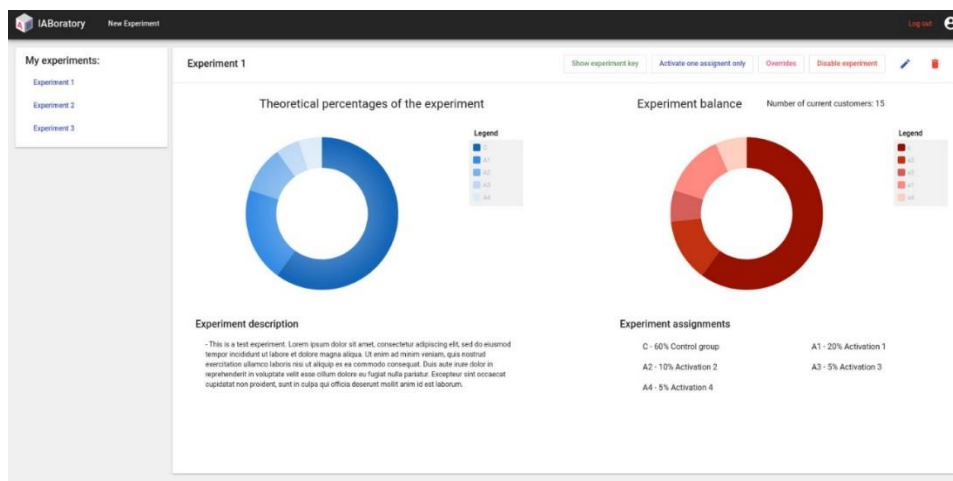


Imagen 9. Pantalla de visualización de experimentos



La pantalla de visualización de experimentos (Imagen 9) muestra toda la información sobre el experimento seleccionado. Se muestra el nombre y descripción del experimento, así como los porcentajes de activación y descripciones de las asignaciones que forman parte del experimento. Además, se muestran dos gráficas de sectores, la primera muestra la distribución de los porcentajes teóricos del experimento y la segunda mostrará el porcentaje de distribución de asignaciones que hay en el momento de su visualización, indicando el número de asignaciones otorgadas a cada grupo. Finalmente, en la parte superior derecha tenemos la lista de las opciones que se puede efectuar sobre los experimentos, mostrar la clave del experimento, activar una asignación, hacer asignaciones manuales, desactivar el experimento, modificar el experimento y eliminarlo.

## 6.6 Mapa de navegación

La aplicación cuenta de tres principales pantallas, la pantalla de inicio, una de acceso y la pantalla principal. Todas ellas se encuentran interrelacionadas entre sí de la siguiente forma.

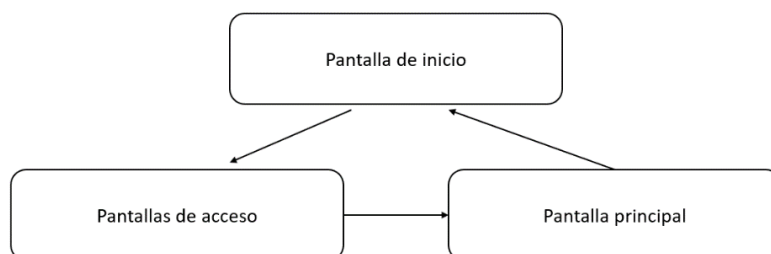


Imagen 10. Mapa de navegación

## 7 Diseño del sistema

### 7.1 Arquitectura

La aplicación estará formada por tres componentes principales, cada componente será un programa autosuficiente que se puede ejecutar en un servidor autónomo. La comunicación entre los distintos componentes será realizada a través de la red de internet mediante el protocolo IP.

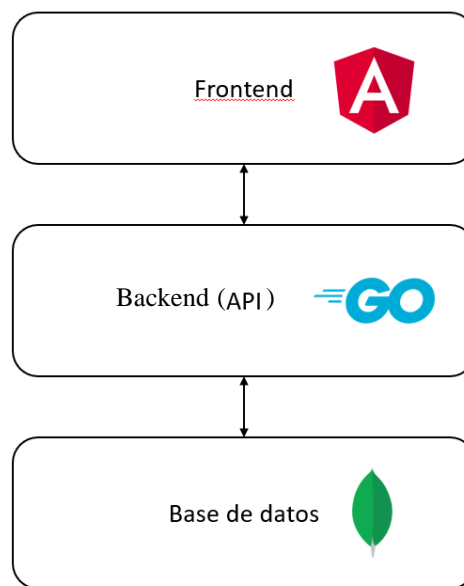


Imagen 11. Esquema de la arquitectura de la aplicación

El primer componente es el *frontend*, el cliente web de la aplicación. Este componente se implementará en Angular y su objetivo es ofrecer a los usuarios de la aplicación una manera amigable e intuitiva de interactuar con la aplicación. Este componente se comunica directamente con la API, pero no tiene acceso directo a la base de datos por tema de seguridad. La comunicación entre el *frontend* y la API se realiza a través del protocolo HTTP y los datos que se intercambian están en formato JSON.

El segundo componente es la API, que se desarrollará en el lenguaje de programación Go, su objetivo es el de implementar toda la lógica de la aplicación hacer las consultas y escrituras necesarias en la base de datos y proporcionar tanto al cliente Web como a las

webs externas los datos requeridos. También implementa el sistema de creación y autenticación de usuarios en la aplicación. La API sigue los estándares REST bajo el protocolo HTTP.

El tercer componente es la base de datos, se ha decidido utilizar MongoDB. Esta base de datos en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

## 7.2 Arquitectura del frontend

Este componente de la aplicación se desarrollará en TypeScript utilizando el *framework* Angular. Este componente se divide en dos principales partes, la primera consiste en la parte visible de la aplicación con la que interactúa directamente el usuario. La segunda parte es la que se encarga de comunicarse con la API de la aplicación.

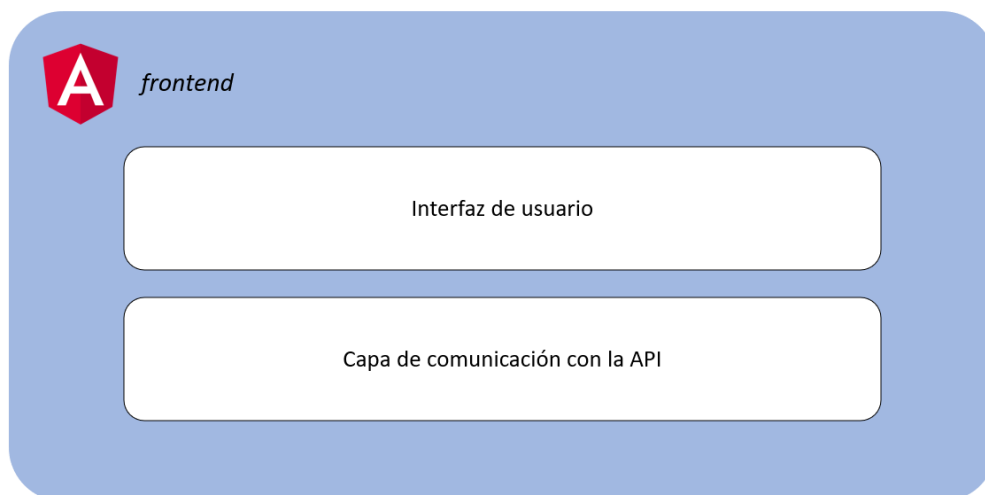


Imagen 12. Esquema de la arquitectura del frontend

La parte que se encarga de mostrar al usuario una interfaz con la que interactuar con la aplicación se basará en componentes, cada componente estará compuesto de una vista, una clase que funciona como controlador de la vista y una hoja de estilos CSS.

La segunda parte es la encargada de comunicarse con la API, esta parte está formada por servicios que se comunican vía HTTP con la API.

### 7.3 Arquitectura del backend

El *backend* se desarrollará en el lenguaje de programación Go, utilizando el *framework* Gin. Su arquitectura se compondrá de distintas capas de abstracción, cada una tiene un único objetivo determinado.

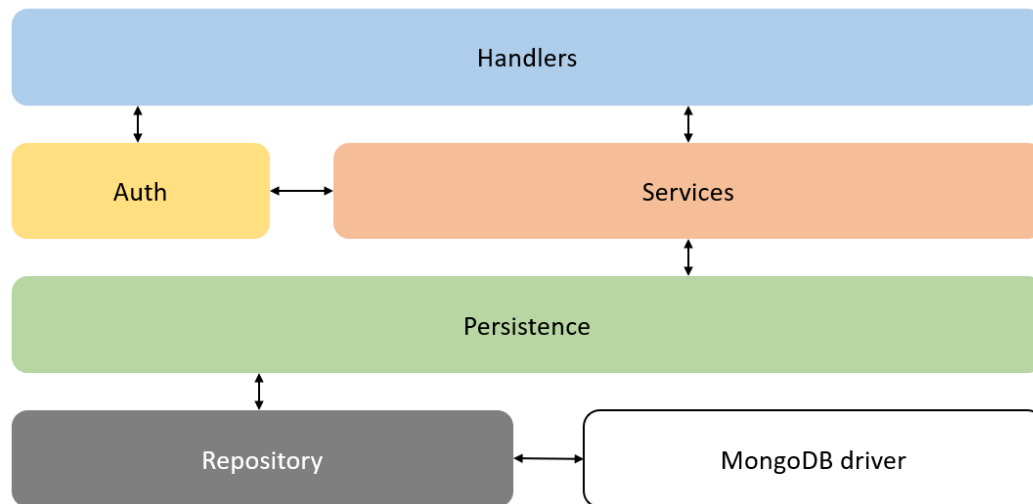


Imagen 13. Esquema de la arquitectura de la API

En la figura anterior se pueden observar las distintas capas del *backend* y su relación entre ellas.

La capa de los *handlers* o controladores se encarga de atender a todas las peticiones HTTP que puede recibir la API. Estas peticiones son procesadas y se extraen los datos si fuera necesario. A continuación, dependiendo de la petición recibida, se le pasa la información a la siguiente capa.

La capa de servicio o *services* se encarga de implementar la lógica principal de la aplicación. Crea, actualiza o elimina los distintos modelos con los que cuenta la aplicación (experimentos, usuarios, asignaciones, ...). También se encarga de realizar el balance de las asignaciones y comunicarse con la siguiente capa para la persistencia.

La capa de persistencia o *persistence*, que implementa el patrón repositorio, se encarga de la comunicación directa con la base de datos. Esta capa recibe las ordenes de la capa de servicio para realizar las lecturas y escrituras pertinentes en la base de datos. Al implementar el patrón repositorio esta capa está completamente abstraída del tipo de

base de datos utilizada por lo que, si en un futuro se desea cambiar la base de datos, sería una tarea extremadamente sencilla.

Adicionalmente se cuenta con una capa que implementa la lógica de la autenticación. Las peticiones que requieran estar autenticadas pasarán por esta capa antes de ir a la capa de servicio. En esta capa se realiza una sencilla validación del token de autorización.

## 7.4 Diseño físico de los datos

Con la utilización de MongoDB como base de datos, los datos se almacenan en disco en formato BSON. Como se indicó anteriormente, BSON es un formato binario de documentos JSON. A continuación, se describe la representación física de estos documentos.

### Experiment (Experimentos)

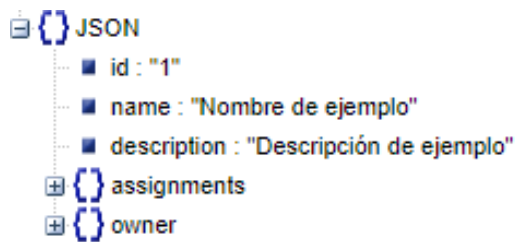


Imagen 14. Representación física de experimentos

### Assignment (Asignaciones)

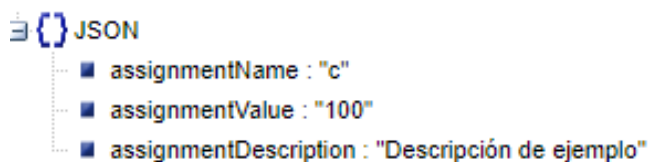


Imagen 15. Representación física de asignaciones

### User (Usuarios)

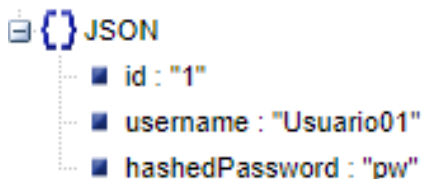


Imagen 16. Representación física de usuarios

### Customer (Clientes)

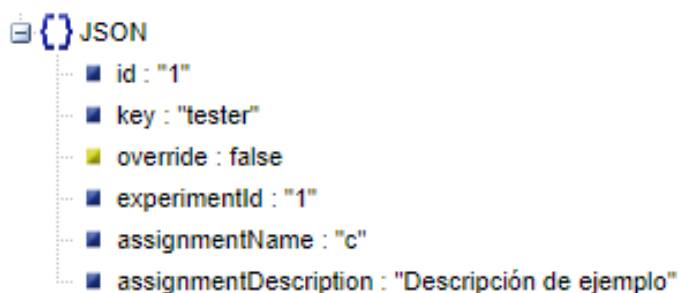


Imagen 17. Representación física de clientes

## 7.5 Modelo público de la API

Casi todas las aplicaciones móviles y web están respaldadas por al menos una base de datos. Estas aplicaciones suelen requerir una API web para exponer dichas bases de datos y poder así almacenar, consultar y leer los datos necesarios de la aplicación. Sin embargo, exponer el modelo de datos directamente a otros usuarios y desarrolladores no es el mejor enfoque para el diseño de una API, ya que, puede suponer riesgos para la seguridad de esta.

Los modelos de datos son una constante en el desarrollo web. Un modelo de datos define cómo se almacena y lee la información para respaldar una solución. Revela exactamente lo que la aplicación puede y no puede ofrecer al usuario final. Y por esto mismo, pensando en el usuario final, es mejor opción dividir el modelado de datos en dos. Se tendrá un modelo de datos para uso interno de la API completo, que se corresponde con modelo de dominio implementado en MongoDB que se ha descrito en el apartado del

diseño físico de los datos, y otro que se utilizará exclusivamente para mostrar los experimentos al usuario final. Este segundo modelo de datos se generará de forma dinámica y se almacenará de manera temporal en memoria, con el fin de mostrar información al usuario final. Este modelo de datos se corresponde con el del siguiente diagrama.

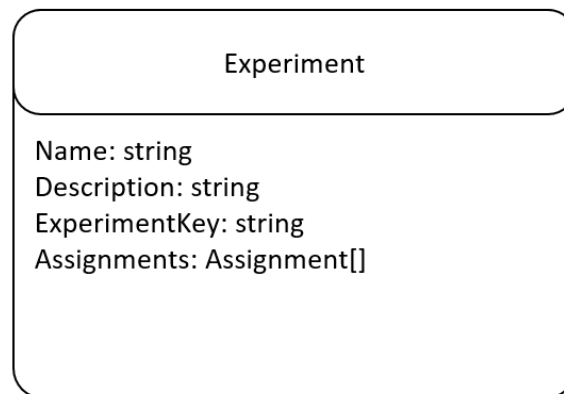


Imagen 18. Modelo de experimentos público

El propietario y el identificador quedan ocultos al usuario final por razones de seguridad y se generará de forma dinámica un campo llamado `ExperimentKey`, que sirve para que las webs externas puedan implementar dicho experimento.

Las ventajas de este enfoque son diversas. Los usuarios finales de la API quieren que sea familiar y sencilla para ellos y no para el equipo que la desarrolla, conceptos como el ID de la base de datos deberían permanecer ocultos para el usuario final. Los usuarios finales también requieren de la API que sea flexible y que ofrezca capacidades para hacer cosas sin crear muchas peticiones HTTP ni unir datos para lograr los resultados deseados, esto se consigue ofreciendo a los usuarios finales modelos que sean completos y no desacoplados como se guardan internamente. Finalmente, el exponer el modelo de datos tal y como se almacena puede suponer un riesgo para la seguridad de la API ya que, un atacante conocerá exactamente como se compone la base de datos otorgándole de ventajas en caso de haber vulnerabilidades explotables. La decisión para este proyecto ha sido por todo lo mencionado con anterioridad el separar el modelo de datos interno del que se expone al usuario final.

## 7.6 Patrones de diseño implementados

En la implementación de este TFG se utilizarán dos patrones de diseño en concreto: Singleton y Repositorio.

El patrón **Singleton** resuelve dos problemas al mismo tiempo. Garantizar que una clase tenga una única instancia y proporcionar un punto de acceso global a dicha instancia. Este patrón se lleva a cabo haciendo privado el constructor por defecto para evitar que otros objetos utilicen esta operación y creando un método de creación estático que actúe como constructor.

Este patrón se va a utilizar para administrar la configuración de la API del proyecto. Habrá un objeto que se encargará de almacenar todos los parámetros de configuración que la aplicación necesita para funcionar. Este objeto solo se debe crear una única vez y debe ser esta instancia del objeto la que se retorne siempre cuando sea consultado. El objeto que contiene los parámetros de configuración se crea al iniciarse la ejecución de la API y dispone de un método público y estático que retornará siempre la misma instancia.

El patrón de diseño de **Repositorio** es uno de los más populares para crear una aplicación de nivel empresarial. Tiene como restricción que se trabaja directamente con los datos de la aplicación y crea nuevas capas para las operaciones de la base de datos, la lógica de negocio, y la interfaz de usuario de la aplicación. Este patrón permite que el código de acceso a los datos puede ser reutilizado. Sea mucho más fácil implementar la lógica de dominio. También desacopla la lógica de la aplicación. La lógica de negocio puede ser probada fácilmente sin acceso a los datos. Y, finalmente, es una buena manera de implementar la inyección de dependencia que hace que el código sea más fácil de probar.

Este patrón se ha utilizado a la hora de desarrollar la capa de persistencia. De esta forma la aplicación se abstrae del tipo de base de datos a utilizar lo que facilita el reemplazo de esta o incluso permite que se puedan añadir opciones adicionales para la persistencia de los datos. Esto ha facilitado enormemente la implementación de la lógica de dominio y de la aplicación, además de facilitar las pruebas realizadas sobre la capa de servicio de la API.



## 7.7 Estándares utilizados

Con el fin de que la aplicación sea sencilla de utilizar, flexible y alargar su vida útil, se ha decidido ajustarse a los estándares actuales de la industria en el desarrollo web. Estos estándares definen la representación de los datos que viajan por la red, el tipo de protocolo de comunicación entre los distintos componentes y el protocolo de autenticación y autorización de los usuarios que utilizan la aplicación. A continuación, se desarrollarán los distintos estándares que se incorporarán en la aplicación.

### 7.7.1 JSON

Para el intercambio de datos por la red es muy utilizado el formato **JSON**. JSON es un formato que almacena información estructurada y se utiliza principalmente para transferir datos entre un servidor y un cliente. El archivo es básicamente una alternativa más simple y ligera a XML que cuenta con funciones similares. El formato JSON está compuesto de dos principales elementos que son la clave y el valor. Las claves deben ser cadenas de caracteres. Como su nombre indica, estas contienen una secuencia de caracteres rodeados de comillas. Los valores son un tipo de datos JSON válido. Puede tener la forma de un vector, objeto, cadena de caracteres, booleano, número o nulo. Un objeto JSON comienza y termina con llaves. Puede tener dos o más pares de claves/valor dentro, con una coma para separarlos. Así mismo, cada clave es seguida por dos puntos para distinguirla del valor.

Este formato se utilizará en el proyecto para transmitir la información entre el *frontend* y la API. También se utilizará para el almacenamiento ya que, la base de datos almacena los propios datos en este formato. Todos los modelos del dominio de la aplicación tendrán una representación en este formato para poder trabajar con ello.

### 7.7.2 REST

El estándar **REST** es una interfaz para conectar varios sistemas basados en el protocolo HTTP y sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON. REST se apoya en HTTP, los verbos que utiliza son exactamente los mismos, con ellos se puede hacer GET, POST, PUT y DELETE. El funcionamiento básico de REST se basa en crear una petición HTTP que contiene toda la información necesaria, es decir, una *request* que se manda a un servidor y esta tiene toda la información necesaria y solo espera una *response*, ósea una

respuesta en concreto. Se apoya sobre el protocolo HTTP que es el que se utiliza para las páginas web. HTTP es un protocolo que existe desde hace muchos años y que ya está consolidado, no se tiene que inventar ni realizar cosas nuevas. Además, todos los objetos se manipulan mediante URL. Los verbos HTTP que utiliza REST son:

- Post: Para crear recursos nuevos.
- Get: Para obtener un listado o un recurso en concreto.
- Put: Para modificar.
- Patch: Para modificar un recurso que no es un recurso de un dato, por ejemplo.
- Delete: Para borrar un recurso, un dato por ejemplo de nuestra base de datos.

Este protocolo permite separar el cliente del servidor. También proporciona escalabilidad, al tener la separación de estos conceptos, por tanto, se puede dedicar mayor esfuerzo a la parte del servidor.

En este proyecto el protocolo REST está integrado directamente en la API y es la forma en la que se comunica esta con el cliente web. También dispone de URLs públicas que pueden consultar las webs externas para pedir asignaciones sin tener que pasar por el cliente web lo que haría el proceso mucho más complicado y menos eficiente.

### 7.7.3 JWT

Finalmente, en el proyecto se incorporará **JWT [4]** (JSON Web Tokens) como método de autenticación. JWT es un estándar que está dentro del documento RFC 7519. En el mismo se define un mecanismo para poder propagar entre dos partes, y de forma segura, la identidad de un determinado usuario, además con una serie de *claims* o privilegios. Estos privilegios están codificados en objetos de tipo JSON, que se incrustan dentro de del *payload* o cuerpo de un mensaje que va firmado digitalmente. En la práctica, se trata de una cadena de texto que tiene tres partes codificadas en Base64, cada una de ellas separadas por un punto. Las tres partes que componen el token son las siguientes:

- Header: encabezado dónde se indica, al menos, el algoritmo y el tipo de token
  - Payload: donde aparecen los datos de usuario y privilegios, así como toda la información que queramos añadir, todos los datos que creamos convenientes.
- En este proyecto se incluye el nombre del usuario y una fecha de caducidad del token para brindarlo con una mayor seguridad.

- **Signature:** una firma que permite verificar si el token es válido, y aquí es donde se encuentra la característica principal de JWT, ya que, si se está tratando de hacer una comunicación segura entre partes y se puede coger cualquier token y ver su contenido con una herramienta sencilla que descodifique base64, esto dará una sensación de inseguridad.

La firma se construye de tal forma que vamos a poder verificar que el remitente es quien dice ser, y que el mensaje no se ha modificado por el camino. De esta forma, si alguien modifica el token por el camino, por ejemplo, inyectando alguna credencial o algún dato malicioso, entonces podríamos verificar que la comprobación de la firma no es correcta, por lo que no podemos confiar en el token recibido y la petición será completamente rechazada. Este método permite al servidor no almacenar ningún dato de sesión y delega la confianza del correcto almacenamiento del token en los usuarios finales.

**Encoded** PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

**Decoded** EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE
<pre>{   "alg": "HS256",   "typ": "JWT" }</pre>
PAYLOAD: DATA
<pre>{   "sub": "1234567890",   "name": "John Doe",   "iat": 1516239022 }</pre>
VERIFY SIGNATURE
<pre>HMACSHA256(   base64UrlEncode(header) + "." +   base64UrlEncode(payload),   your-256-bit-secret ) <input type="checkbox"/> secret base64 encoded</pre>

Imagen 19. Esquema de JWT

## 7.8 Entorno de pruebas

Para la realización de las pruebas, se utiliza el propio entorno de pruebas que proporciona el compilador Go y el *framework* Angular.

El compilador de Go proporciona, tanto los paquetes necesarios para la realización de pruebas como el binario para la ejecución de estas. El paquete para la realización de las pruebas se llama “testing” y proporciona las funcionalidades necesarias para la realización de pruebas unitarias y automatizarlas. Además, el propio compilador de Go se puede ejecutar para que, en vez de compilar la aplicación, ejecute todas las pruebas unitarias creadas dentro del proyecto y muestre estadísticas de éxito. También puede mostrar el porcentaje de métodos que están cubiertos por pruebas con intención de aumentar este porcentaje al máximo posible.

Angular realiza las pruebas con una librería llamada Karma que se puede lanzar directamente desde el propio angular ejecutando el comando “ng test”.

Todas estas pruebas se encuentran automatizadas con un proceso de integración continúa configurado en el repositorio de GitLab. Este proceso está dividido en dos fases y necesita del uso de contenedores Docker para su funcionamiento. La primera fase se encarga de compilar la aplicación. Si el proceso falla, se envía un email a los colaboradores del repositorio indicando el log de error del propio compilador. En caso de éxito, se lanza la segunda fase, que consiste en la ejecución de todas las pruebas unitarias de forma automatizada. Una vez superadas con éxito todas las pruebas, el repositorio queda en estado de “Pipeline passed”. Todo este proceso se ejecuta en cada nuevo *commit* que incorpora el repositorio, haciendo que cada cambio del código fuente se someta a un proceso de prueba e integración completo.

## 7.9 Diseño de las pruebas

### 1. Pruebas en la autenticación

#### 1.1 Creación de usuarios nuevos

##### 1.1.1 Proceso de creación de usuarios nuevos

#### 1.2 Autenticación

##### 1.2.1 Autenticación de usuarios

### 1.2.2 Autenticación de experimentos

## 2. Pruebas sobre los experimentos

### 2.1 Validar experimentos nuevos

#### 2.1.1 Validar experimento correcto

#### 2.1.2 Validar experimento con asignaciones erróneas

#### 2.1.3 Validar experimento con la suma de los porcentajes de activación desigual de 100

#### 2.1.4 Validar experimento que ya existe en la base de datos

### 2.2 Comprobar si el experimento nuevo ya existe

#### 2.2.1 Validar experimento que no existe en base de datos

#### 2.2.2 Validar experimento que si existe en base de datos

### 2.3 Validar asignaciones

#### 2.3.1 Validar asignaciones correctas

#### 2.3.2 Validar asignaciones con formato incorrecto

#### 2.3.3 Validar asignaciones duplicadas

#### 2.3.4 Validar asignaciones con porcentajes de activación incorrectos

### 2.4 Validar el propietario del experimento

#### 2.4.1 Validar experimento cuyo propietario coincide con el creador

#### 2.4.2 Validar experimento cuyo propietario no coincide con el creador

## 3. Pruebas en el algoritmo de balanceo

### 3.1 Generar asignaciones balanceadas

#### 3.1.1 Generar asignación balanceada

## 7.10 Ejecución de las pruebas

### 7.10.1 Creación de usuarios nuevos

Estado previo de la base de datos:

USER		
Id	Username	HashedPassword
-	-	-

Ejecución de las pruebas:

Prueba: 1.1.1	Creación de nuevos usuarios
Objetivo	Comprobar que un usuario se guarda correctamente en la base de datos con la contraseña nueva convertida en un hash.
Entradas	Nombre de usuario: Usuario Contraseña: hello world
Salida esperada	Username: "Usuario" HashedPassword: "b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9"
Resultado obtenido	Username: "Usuario" HashedPassword: "b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9"

### 7.10.2 Autenticación

Estado previo de la base de datos:

USER		
Id	Username	HashedPassword
1	"subject"	"b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9"

EXPERIMENT				
Id	Name	Description	Assignments	Owner
1	“subject”	“”	{ AssignmentName: “c”, AssignmentValue: 100, AssignmentDescription: “” }	{ Id: 1, Username: “subject”, HashedPassword: “b94d27...” }

Ejecución de las pruebas:

Prueba: 1.2.1	Autenticación de usuarios
Objetivo	Comprobar que para cada usuario que se autentica en la aplicación se genera un token con expiración correctamente.
Entradas	Usuario: subject
Salida esperada	JWT: “eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJzdWJqZWNoIiwiaWF0IjoxNTE2MjM5MDIyfQ.ZFLw7HLYEC8KCJ3dJN5US5cuU4gtKcOlGKkJJeQV9qxU”
Resultado obtenido	JWT: “eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJzdWJqZWNoIiwiaWF0IjoxNTE2MjM5MDIyfQ.ZFLw7HLYEC8KCJ3dJN5US5cuU4gtKcOlGKkJJeQV9qxU”

Prueba: 1.2.2	Autenticación de experimentos
Objetivo	Comprobar que para cada experimento creado se genera un token sin expiración correctamente.
Entradas	Experimento: subject

Salida esperada	JWT: “eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJzdWJqZWN0In0.Defqg0PElmfYmn5jVNyvbUP-IM-P8fhbby3JykQgtw8”
Resultado obtenido	JWT: “eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJzdWJqZWN0In0.Defqg0PElmfYmn5jVNyvbUP-IM-P8fhbby3JykQgtw8”

### 7.10.3 Validar experimentos nuevos

Estado previo de la base de datos:

EXPERIMENT				
Id	Name	Description	Assignments	Owner
1	“Name 1”	“”	{ }	{ }
2	“Name 2”	“”	{ }	{ }
3	“Name 3”	“”	{ }	{ }

Ejecución de las pruebas:

Prueba: 2.1.1	Validar experimentos nuevos
Objetivo	Comprobar la validez de un nuevo experimento.
Entradas	Id: "", Name: "Name", Description: "Description", Assignments: { { AssignmentName: "a1", AssignmentValue: 25.0, AssignmentDescription: "" }, { AssignmentName: "a2", AssignmentValue: 25.0, AssignmentDescription: "" }, { AssignmentName: "a3", AssignmentValue: 25.0, AssignmentDescription: "" }, { AssignmentName: "a4", AssignmentValue: 25.0, AssignmentDescription: "" }, },



	Owner: {Id: "", Username: "username 1", HashedPassword: "pw",}
Salida esperada	El experimento es válido.
Resultado obtenido	El experimento es válido.

Prueba: 2.1.2	Validar experimentos nuevos
Objetivo	Comprobar la validez de un nuevo experimento.
Entradas	Id:"", Name: "Name", Description: "Description", Assignments: { {AssignmentName: "Assignment 1", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "A2", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "a3", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "Assignment 4", AssignmentValue: 25.0, AssignmentDescription: ""}, }, Owner: {Id: "", Username: "username 1", HashedPassword: "pw",}
Salida esperada	Experimento inválido. (El formato del nombre de las asignaciones es incorrecto)
Resultado obtenido	Experimento inválido.

Prueba: 2.1.3	Validar experimentos nuevos
Objetivo	Comprobar la validez de un nuevo experimento.
Entradas	Id:"",

	Name: "Name", Description: "Description", Assignments: { {AssignmentName: "a1", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "a2", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "a3", AssignmentValue: 0.0, AssignmentDescription: ""}, {AssignmentName: "a4", AssignmentValue: 25.0, AssignmentDescription: ""}, }, Owner: {Id: "", Username: "username 1", HashedPassword: "pw",}
Salida esperada	Experimento inválido. (La suma de los porcentajes de activación no es 100)
Resultado obtenido	Experimento inválido

Prueba: 2.1.4	Validar experimentos nuevos
Objetivo	Comprobar la validez de un nuevo experimento.
Entradas	Id:"", Name: "Name 1", Description: "Description", Assignments: { {AssignmentName: "a1", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "a2", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "a3", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "a4", AssignmentValue: 25.0,

	AssignmentDescription: ""}, }, Owner: {Id: "", Username: "username 1", HashedPassword: "pw",}
Salida esperada	Experimento inválido. (El nombre del experimento ya se encuentra registrado en la base de datos)
Resultado obtenido	Experimento inválido.

#### 7.10.4 Comprobar si el experimento nuevo ya existe

Estado previo de la base de datos:

EXPERIMENT				
Id	Name	Description	Assignments	Owner
1	"Name 1"	""	{}	{}
2	"Name 2"	""	{}	{}
3	"Name 3"	""	{}	{}

Ejecución de las pruebas:

Prueba: 2.2.1	Comprobar si el experimento nuevo ya existe
Objetivo	Comprobar que el experimento nuevo introducido en el sistema no tenga el mismo nombre de otro ya registrado.
Entradas	Id:"", Name: "Name 4", Description: "", Assignments: {}, Owner: {}
Salida esperada	El experimento no existe.
Resultado obtenido	El experimento no existe.

Prueba: 2.2.2	Comprobar si el experimento nuevo ya existe
Objetivo	Comprobar que el experimento nuevo introducido en el sistema no tenga el mismo nombre de otro ya registrado.

Entradas	Id: "", Name: "Name 1", Description: "", Assignments: {}, Owner: {}
Salida esperada	El experimento ya existe.
Resultado obtenido	El experimento ya existe.

### 7.10.5 Validar asignaciones

Estado previo de la base de datos:

EXPERIMENT				
Id	Name	Description	Assignments	Owner
-	-	-	-	-

Ejecución de las pruebas:

Prueba: 2.3.1	Validar asignaciones
Objetivo	Comprobar si las asignaciones que componen el experimento tienen el formato correcto, no están duplicadas y sus porcentajes de activación suman 100.
Entradas	{ AssignmentName: "a1", AssignmentValue: 25.0, AssignmentDescription: "" }, { AssignmentName: "a2", AssignmentValue: 25.0, AssignmentDescription: "" }, { AssignmentName: "a3", AssignmentValue: 25.0, AssignmentDescription: "" }, { AssignmentName: "a4", AssignmentValue: 25.0, AssignmentDescription: "" },
Salida esperada	Las asignaciones son válidas.
Resultado obtenido	Las asignaciones son válidas.

Prueba: 2.3.2	Validar asignaciones
Objetivo	Comprobar si las asignaciones que componen el experimento tienen el formato correcto, no están duplicadas y sus porcentajes de activación suman 100.
Entradas	{AssignmentName: "Assignment 1", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "Assignment 2", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "Assignment 3", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "Assignment 4", AssignmentValue: 25.0, AssignmentDescription: ""},
Salida esperada	Las asignaciones son inválidas. (Formato incorrecto en el nombre de las asignaciones)
Resultado obtenido	Las asignaciones son inválidas.

Prueba: 2.3.3	Validar asignaciones
Objetivo	Comprobar si las asignaciones que componen el experimento tienen el formato correcto, no están duplicadas y sus porcentajes de activación suman 100.
Entradas	{AssignmentName: "a1", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "a2", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "a1", AssignmentValue: 25.0, AssignmentDescription: ""}, {AssignmentName: "a4", AssignmentValue: 25.0, AssignmentDescription: ""},

Salida esperada	Las asignaciones son inválidas. (Hay asignaciones con nombre repetido)
Resultado obtenido	Las asignaciones son inválidas.

Prueba: 2.3.4	Validar asignaciones
Objetivo	Comprobar si las asignaciones que componen el experimento tienen el formato correcto, no están duplicadas y sus porcentajes de activación suman 100.
Entradas	{ AssignmentName: "a1", AssignmentValue: 25.0, AssignmentDescription: "" }, { AssignmentName: "a2", AssignmentValue: 25.0, AssignmentDescription: "" }, { AssignmentName: "a3", AssignmentValue: 25.0, AssignmentDescription: "" }
Salida esperada	Las asignaciones son inválidas. (La suma de los porcentajes de activación no es igual a 100)
Resultado obtenido	Las asignaciones son inválidas.

#### 7.10.6 Validar el propietario del experimento

Estado previo de la base de datos:

USER		
Id	Username	HashedPassword
1	"username 1"	"pw"
2	"username 2"	"pw"

EXPERIMENT				
Id	Name	Description	Assignments	Owner
-	-	-	-	-

Ejecución de las pruebas:

Prueba: 2.4.1	Validar el propietario del experimento
Objetivo	Comprobar que el experimento nuevo introducido en el sistema realmente pertenece al usuario que lo está introduciendo.
Entradas	Experimento nuevo: {Id: 1, Name: "Name 1", Description: "", Assignments: {}}, Owner: { Id: 1, Username: "username 1", HashedPassword: "pw"}  Usuario creador del experimento: {Id: 1, Username: "username 1", HashedPassword: "pw"}
Salida esperada	El experimento pertenece al usuario que lo ha creado.
Resultado obtenido	El experimento pertenece al usuario que lo ha creado.

Prueba: 2.4.2	Validar el propietario del experimento
Objetivo	Comprobar que el experimento nuevo introducido en el sistema realmente pertenece al usuario que lo está introduciendo.
Entradas	Experimento nuevo: {Id: 1, Name: "Name 1", Description: "", Assignments: {}}, Owner: { Id: 1, Username: "username 1", HashedPassword: "pw"}  Usuario creador del experimento: {Id: 1, Username: "username 2", HashedPassword: "pw"}
Salida esperada	El experimento no pertenece al usuario que lo ha creado.
Resultado obtenido	El experimento no pertenece al usuario que lo ha creado.

### 7.10.7 Generar asignación balanceada

Estado previo de la base de datos:

EXPERIMENT				
Id	Name	Description	Assignments	Owner
1	Experimento de pruebas	“”	<pre>{   AssignmentName: “a1”,   AssignmentValue: 25,   AssignmentDescription: “” }, {   AssignmentName: “a2”,   AssignmentValue: 25,   AssignmentDescription: “” }, {   AssignmentName: “a3”,   AssignmentValue: 25,   AssignmentDescription: “” }, {   AssignmentName: “a4”,   AssignmentValue: 25,   AssignmentDescription: “” } </pre>	{ }

CUSTOMER				
Id	ExperimentId	Override	AssignmentName	AssignmentDescription
1	1	false	a1	Descripción de ejemplo
2	1	false	a2	Descripción de ejemplo
3	1	false	a4	Descripción de ejemplo



Ejecución de las pruebas:

Prueba: 3.1.1	Generar asignación balanceada
Objetivo	Comprobar que el algoritmo asigna correctamente nuevas asignaciones en situaciones de desbalanceo.
Entradas	-
Salida esperada	Customer: { "id": "4", "experimentId": "1", "override": false, "assignmentName": "a3", "assignmentDescription": "Descripción de ejemplo" }
Resultado obtenido	Customer: { "id": "4", "experimentId": "1", "override": false, "assignmentName": "a3", "assignmentDescription": "Descripción de ejemplo" }

## **8 Manual de usuario**

### **8.1 Introducción**

IA/Boratory es una aplicación web que permite a los usuarios crear, modificar y utilizar experimentos de A/B testing en sus webs o servicios para la extracción de métricas que optimicen un objetivo concreto. Estos experimentos quedan registrados dentro de la propia aplicación y son expuestos de manera pública para que distintos usuarios externos puedan pedir asignaciones nuevas y consultarlas posteriormente.

Las funcionalidades principales de la aplicación web consisten en, la creación de experimentos de tipo A/B testing, la modificación de estos, su eliminación, la asignación manual de usuarios en los distintos grupos que conforman los experimentos, la activación de una asignación del experimento y finalmente deshabilitar el experimento.

### **8.2 Puesta en marcha**

EL proyecto consiste en una aplicación web en la que están disponibles todas las funcionalidades, por lo que no es necesario realizar ninguna instalación, es accesible desde cualquier navegador web con acceso a internet, ya sea desde un ordenador o un teléfono móvil.

### **8.3 Pantalla de inicio**

La pantalla principal sigue un esquema minimalista donde se muestran las principales opciones de las que dispone un usuario no registrado en la aplicación. La siguiente imagen muestra la pantalla principal.

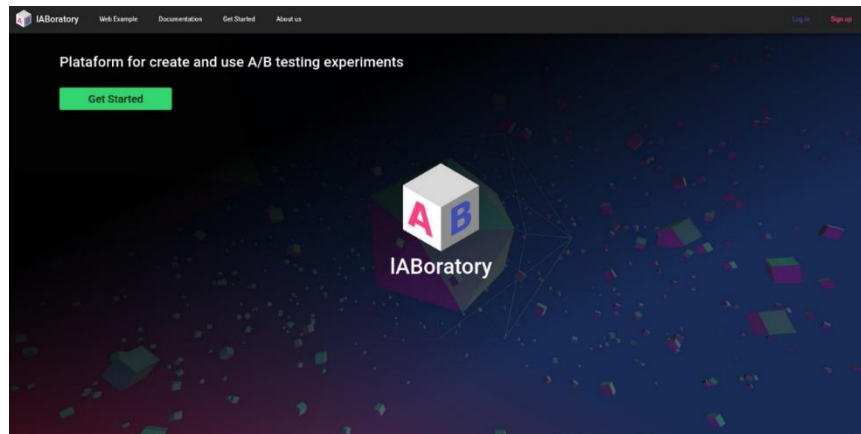


Imagen 20. Pantalla principal

Las opciones públicas de la aplicación se muestran en la barra de navegación (Imagen 21). Entre estas opciones contamos con un ejemplo de una web que implementa un experimento A/B a forma de demostración de su utilidad y funcionamiento. Adicionalmente tenemos enlaces a la documentación, una guía rápida y un apartado de “Sobre nosotros”.

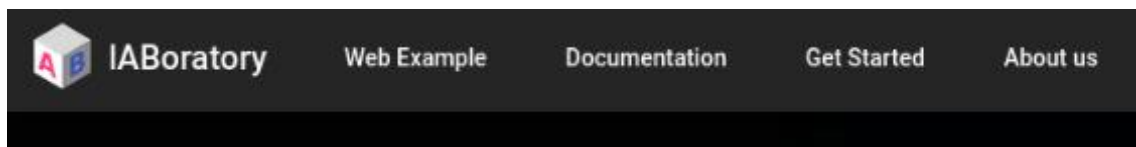


Imagen 21. Barra de navegación pública

En la parte derecha de la pantalla principal contamos con las dos principales acciones disponibles para un usuario no autenticado dentro de la aplicación (Imagen 22). La primera opción, “Log in”, permite a los usuarios ya registrados iniciar sesión en sus perfiles. La segunda, “Sign up”, permite a los usuarios que no disponen de una cuenta registrada dentro de la aplicación la creación y utilización de una nueva.

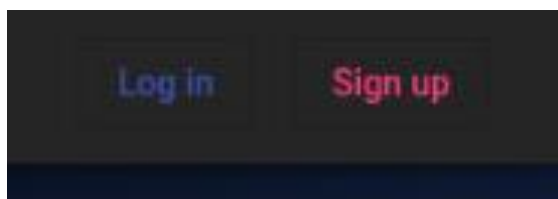


Imagen 22. Opciones de acceso

## 8.4 Pantallas de acceso

Pantalla de inicio de sesión a usuarios que ya cuentan con una cuenta registrada en la aplicación (Imagen 23).

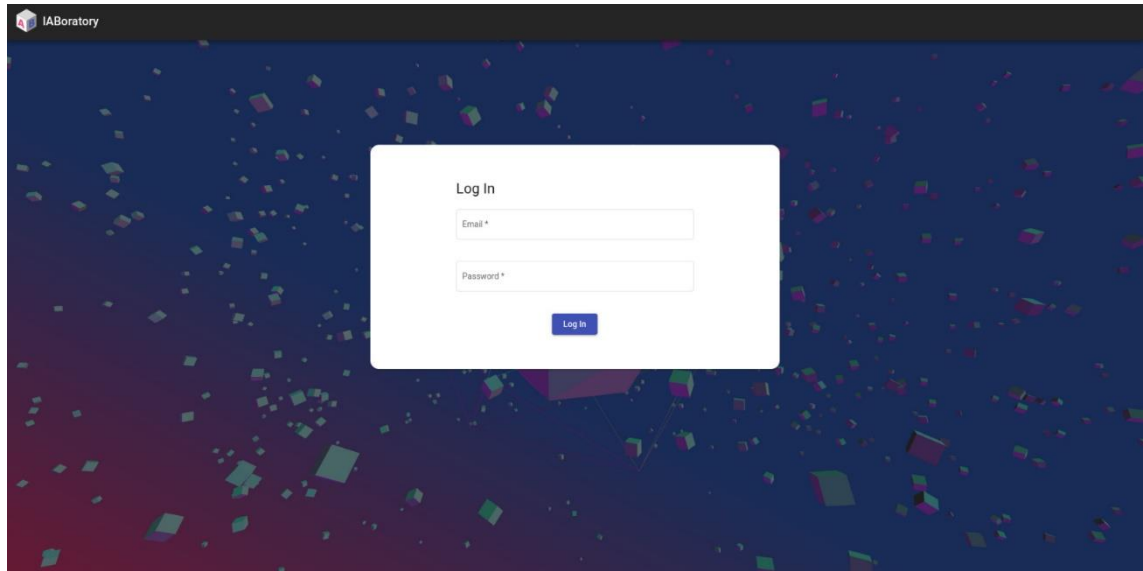


Imagen 23. Pantalla de inicio de sesión

En esta pantalla disponemos de un único formulario que cuenta con dos campos. En el primero se debe introducir el email del usuario que quiere autenticarse y en el segundo la contraseña del este. El botón que se encuentra debajo del formulario dará acceso a la pantalla principal de la aplicación en caso de que las credenciales sean correctas.

## 8.5 Pantalla de registro para nuevos usuarios de la aplicación

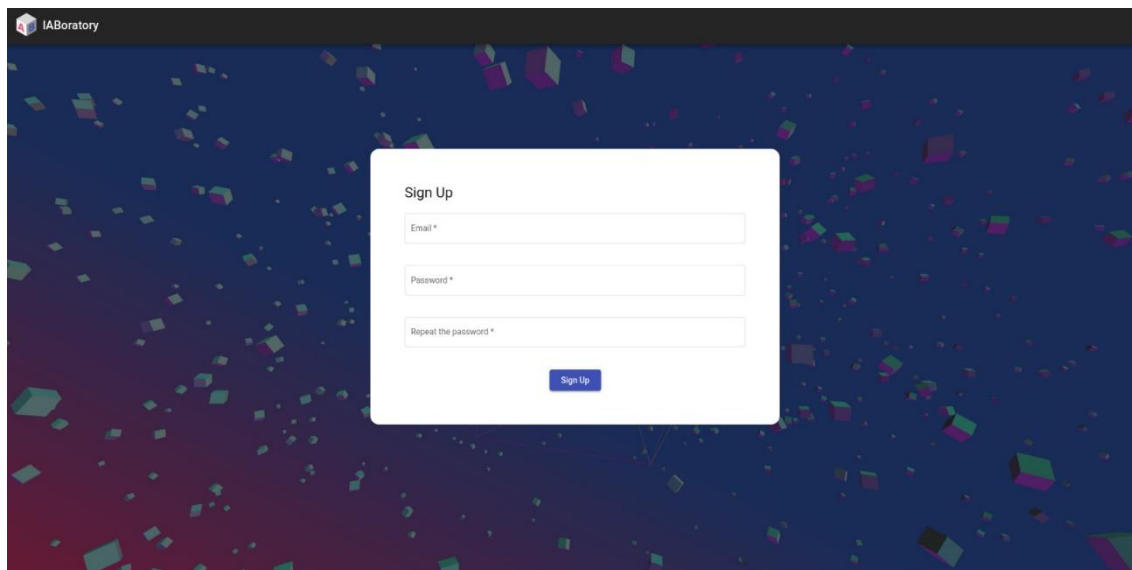


Imagen 24. Pantalla de registro

En esta pantalla (Imagen 24) disponemos de un único formulario que cuenta con tres campos. En el primero se debe introducir el email del usuario que quiere autenticarse, el nuevo email introducido no debe coincidir con el de ninguna cuenta registrada anteriormente en la aplicación. En el segundo campo se pide configurar una contraseña y finalmente en el tercero se pide que se vuelva a introducir la contraseña para prevenir errores de escritura por parte del usuario nuevo que le impida iniciar sesión en el futuro. El botón que se encuentra debajo del formulario dará acceso a la pantalla principal de la aplicación en caso de que los requisitos de registro se hayan cumplido (El email, ha de ser un email válido y no debe estar registrado en la aplicación y las contraseñas han de coincidir).

## 8.6 Pantalla principal

La pantalla principal cuenta con tres secciones: la barra de navegación, la lista de los experimentos creados por el usuario y el panel principal. La siguiente imagen muestra dicha pantalla.

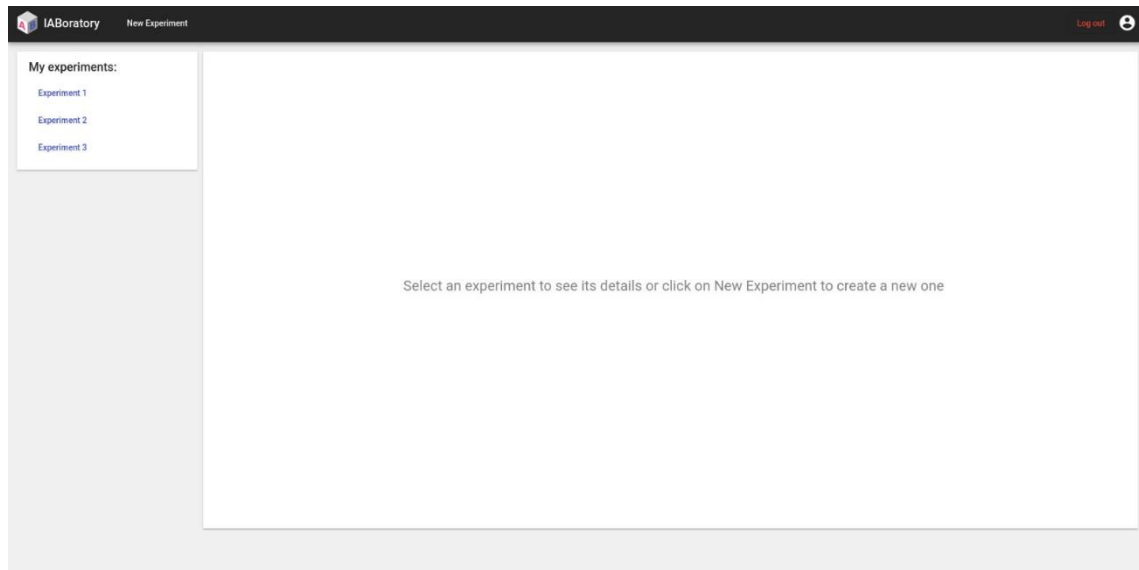


Imagen 25. Pantalla principal

En la barra de navegación contamos con dos funcionalidades. La primera funcionalidad consiste en la creación de nuevos experimentos (Imagen 26). La segunda funcionalidad sirve para cerrar sesión e ir a la pantalla inicial (Imagen 27).

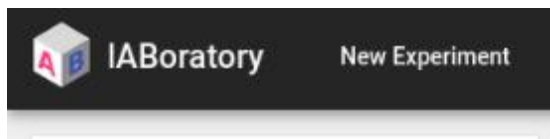


Imagen 26. Crear nuevo experimento

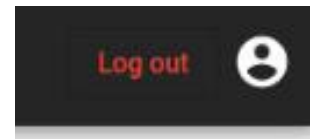
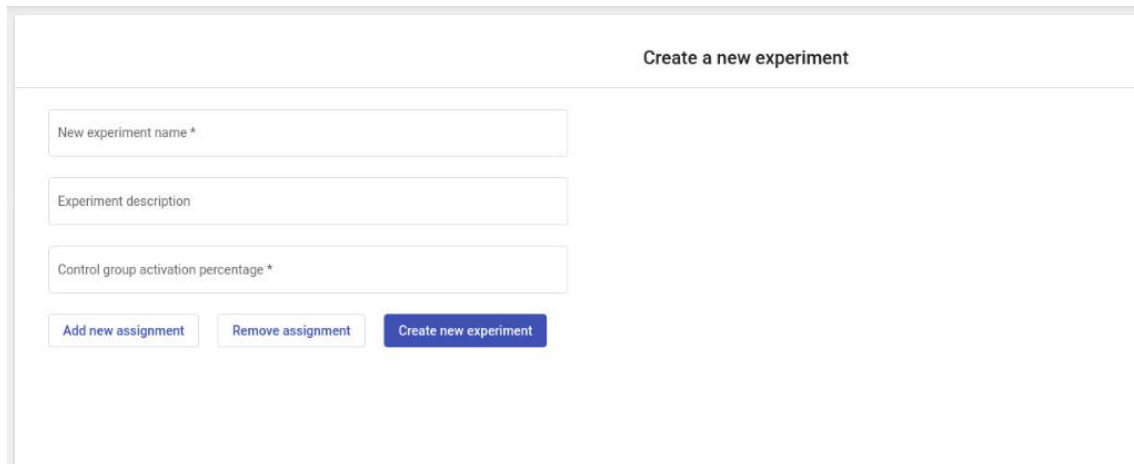


Imagen 27. Cierre de sesión

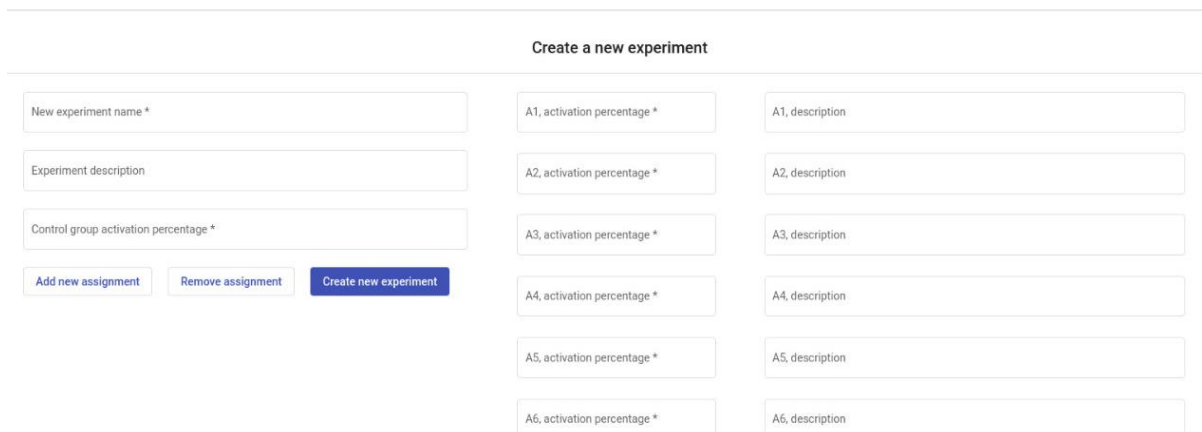
Al hacer clic en la opción de crear un nuevo experimento, en el panel de la derecha se generará un formulario para la configuración del nuevo experimento. En este formulario se pide el nombre del experimento una descripción opcional y el conjunto de asignaciones que lo conformarán. Para añadir o quitar asignaciones, contamos con las opciones de “Add new assignment” o “Remove assignment”. Inicialmente solo contamos con la asignación obligatoria que representa el grupo de control. La siguiente imagen muestra dicho formulario.



Formulario de creación de experimentos sin asignaciones. El formulario tiene un título "Create a new experiment". Contiene tres campos de texto: "New experiment name \*", "Experiment description" y "Control group activation percentage \*". Debajo de los campos hay tres botones: "Add new assignment", "Remove assignment" y "Create new experiment".

Imagen 28. Formulario de creación de experimentos sin asignaciones

Sin embargo, a medida que añadimos asignaciones nuevas se irán generando nuevos campos del formulario para configurar cada una. Por cada asignación nueva se ha de indicar el porcentaje de activación de esta y adicionalmente se puede indicar una descripción opcional. La siguiente imagen muestra dicho formulario.



Formulario de creación de experimentos con asignaciones. El formulario tiene un título "Create a new experiment". Contiene tres campos de texto: "New experiment name \*", "Experiment description" y "Control group activation percentage \*". Debajo de los campos hay tres botones: "Add new assignment", "Remove assignment" y "Create new experiment". A la derecha de los botones hay una lista de asignaciones con los campos "A1, activation percentage \*", "A1, description", "A2, activation percentage \*", "A2, description", "A3, activation percentage \*", "A3, description", "A4, activation percentage \*", "A4, description", "A5, activation percentage \*", "A5, description", "A6, activation percentage \*", "A6, description".

Imagen 29. Formulario de creación de experimentos con asignaciones

Finalmente, una vez configurado el experimento tenemos la opción de "Create new experiment". Cuando el usuario hace clic en esta opción se validará el experimento, este debe tener un nombre distinto a los experimentos ya creados por el usuario, debe tener todos los campos obligatorios cubiertos y además la suma de los porcentajes tiene que ser 100. Si el experimento es válido esta sección pasará a mostrar los detalles del nuevo experimento y este quedará registrado en la base de datos.

En la segunda sección de la pantalla principal contamos con la lista de todos los experimentos pertenecientes al usuario que ha iniciado sesión. Los experimentos deben tener siempre distintito nombre entre los pertenecientes a un mismo usuario. La funcionalidad de esta sección consiste en mostrar los detalles de cada experimento en el panel de la derecha cuando el usuario hace clic sobre uno de ellos. La siguiente imagen muestra una lista de experimentos de ejemplo.



Imagen 30. Lista de experimentos

La tercera sección como vimos anteriormente sirve para mostrar el formulario de creación de los experimentos, sin embargo, su funcionalidad más importante es la de mostrar los detalles de los experimentos creados. Para mostrar estos detalles se debe crear un experimento nuevo o hacer clic en el en la sección que muestra la lista de los experimentos ya creados por el usuario. Se muestra el nombre y descripción del experimento además de sus asignaciones. También se muestran dos gráficas, la primera muestra el valor teórico de los porcentajes que implementa el experimento, este coincide con los porcentajes indicados en sus asignaciones. La segunda gráfica muestra el número de usuarios que están formando parte del experimento y con que asignación cuentan. A la larga la gráfica de la derecha tenderá a igualarse a la de la izquierda suponiendo que no haya usuarios con asignaciones configuradas de forma manual. Esta opción se muestra en la siguiente imagen con un experimento de ejemplo.



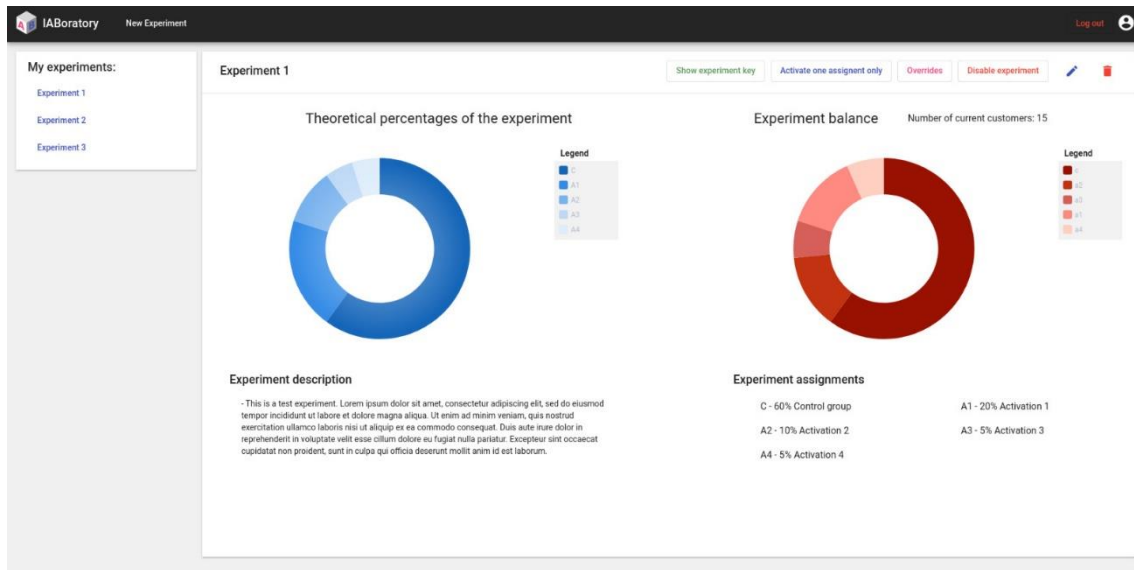


Imagen 31. Pantalla con los detalles de un experimento

Al mostrar los detalles de un experimento contamos también con las funciones que podemos realizar sobre este. Estas funcionalidades están disponibles en la parte superior y se muestran en la siguiente imagen.



Imagen 32. Opciones para realizar sobre un experimento

“Show experiment key” mostrará la clave del experimento para poder consumirlo, todas las webs que quieran implementar este experimento necesitan contar con esta clave. Tenemos dos opciones, la primera copia al portapapeles la clave para poder distribuirla y la segunda ejemplifica como conseguir una asignación para un usuario cualquiera usando la clave del experimento. La siguiente imagen muestra la clave de un experimento de ejemplo.



Imagen 33. Clave de un experimento

Como se puede apreciar en la imagen de abajo, al hacer clic sobre la opción “Get Assignment” abre una nueva pestaña en el navegador. Al consultar la url con la clave del experimento se otorga una asignación. El usuario que recibe la asignación también se indica en la url, en este caso el usuario es “test”. De esta forma las webs externas que desean utilizar los experimentos solo necesitan conocer la clave y el nombre del usuario que participará en el experimento y automáticamente se le otorgará una nueva asignación. La siguiente imagen muestra una asignación recibida desde la API como ejemplo.

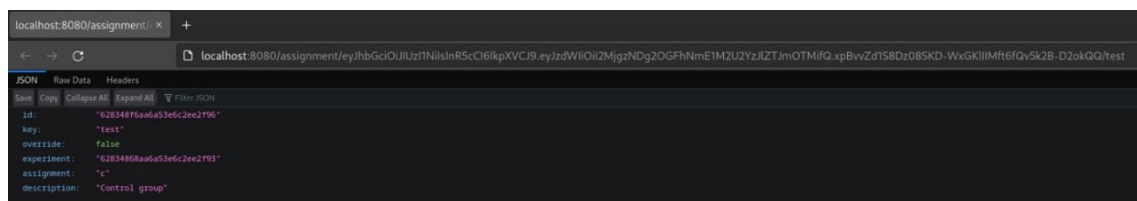


Imagen 34. Asignación recibida por la API

“Activate one assignment only” mostrará un diálogo para seleccionar una de las asignaciones con las que cuenta el experimento. Al hacer clic en una de ellas los porcentajes de activación cambiarán pasando a estar todos en 0% salvo la opción elegida que pasará al 100%. No solo cambiarán los porcentajes teóricos, sino que también se inicia una tarea en segundo plano que cambiará todas las asignaciones otorgadas con anterioridad a la nueva opción indicada. La finalidad de esta funcionalidad es la de hacer que el experimento se quede bloqueado en una única asignación. “Disable experiment” realiza una función similar, pero sobre el grupo de control, dejando desactivadas todas las otras asignaciones. La siguiente imagen muestra dicha opción.

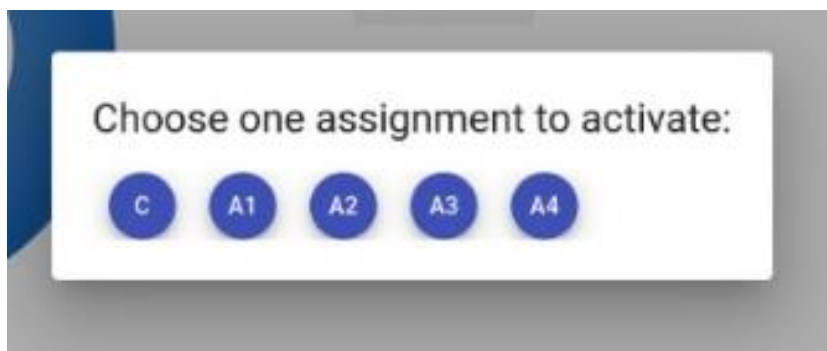


Imagen 35. Opción para activar una asignación

“Overrides” permite al usuario configurar manualmente asignaciones a usuarios concretos, así como eliminar estas asignaciones manuales. En la parte superior se mostrará una lista con las asignaciones manuales con las que cuenta el experimento. En la parte inferior contamos con un campo donde se introducirá el nombre del usuario a configurar y después la lista de las asignaciones disponibles para hacer la asignación manual. La siguiente imagen muestra dicha opción.

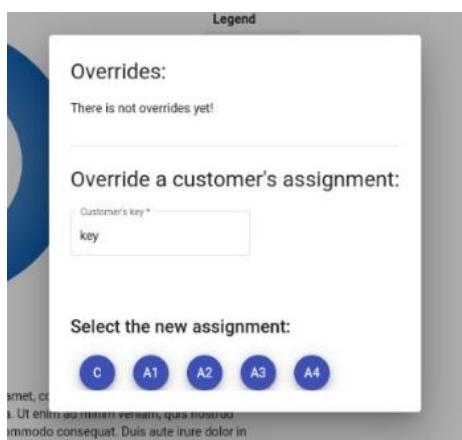
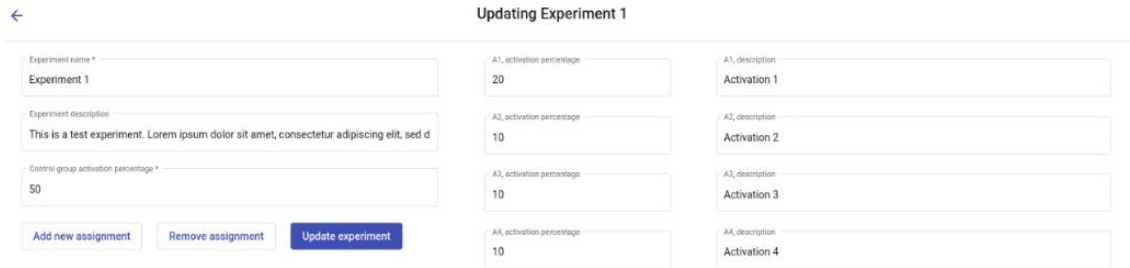


Imagen 36. Asignaciones manuales

Las dos últimas opciones sirven para editar los valores del experimento o para eliminarlo permanentemente. En caso de querer editar el experimento se mostrará el siguiente formulario. Este cuenta con los mismos requisitos que el formulario de creación de un experimento nuevo. La siguiente imagen muestra dicho formulario.



\*The update of the values will be done in a background task

Imagen 37. Formulario de modificación de un experimento

Al igual que en la creación de un nuevo experimento se pueden modificar todos los campos y añadir o eliminar asignaciones. En caso de eliminar alguna asignación la cual tienen usuarios que participan en el experimento, sus asignaciones cambiarán a la más popular del experimento.

Finalmente, al hacer clic sobre el botón para eliminar de forma permanente el experimento, saldrá un cuadro de confirmación. En caso de eliminar el experimento, se eliminará de la base de datos todas las asignaciones a los usuarios realizadas y no podrá ser posible volver a consultar asignaciones. La siguiente imagen muestra dicho cuadro de diálogo.

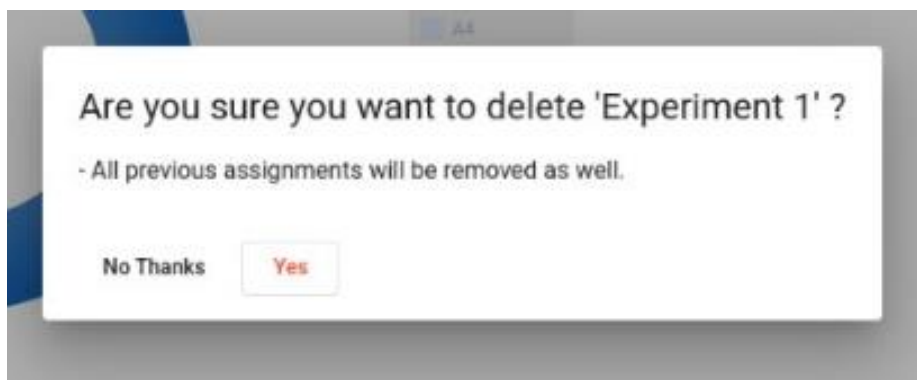


Imagen 38. Mensaje de confirmación de la eliminación de un experimento

## **9 Ampliaciones**

Para este proyecto me gustaría plantear las siguientes tres ampliaciones. Dos de ellas no pudieron incorporarse en el desarrollo del TFG por falta de tiempo en la planificación.

### **9.1 Agrupamiento de experimentos por campañas**

La primera de todas es el agrupamiento de distintos experimentos por campañas, por ejemplo, campañas de marketing específicas que cuenten con más de un experimento. Esto funcionaría implementando un nivel jerárquico que se encuentre encima de los experimentos. Todos los experimentos pertenecerían a una campaña, incluso si esa campaña solo tuviera un experimento. Las campañas pueden tener una descripción más extensa sobre los objetivos que se pretenden alcanzar con los experimentos que esta contiene.

### **9.2 Sistema de propietarios múltiples**

La segunda ampliación consiste en un sistema por el cual varios usuarios de la aplicación puedan tener acceso a un mismo experimento. Todos los usuarios que pueden acceder a dicho experimento podrán modificarlo y estos cambios podrán ser vistos por el resto de los propietarios. La finalidad de esta ampliación consiste permitir el trabajo en equipo en la plataforma.

### **9.3 Diferentes algoritmos de balanceo seleccionables**

Finalmente, la última ampliación permitiría al administrador de un experimento seleccionar distintos algoritmos de balanceo de asignaciones cuando estas son modificadas. Actualmente la aplicación solo cuenta con un algoritmo de balanceo, pero dentro del contexto del marketing digital y para la recolección de datos puede llegar a ser muy útil disponer de varios algoritmos de balanceo que permitan un rebalanceo más rápido o que no importe si se cambian asignaciones preexistentes.

## 10 Conclusiones

Este ha sido el mayor proyecto de desarrollo de software en el que me he visto envuelto y uno de los pocos en los que he trabajado en todo el proceso de desarrollo, desde el diseño hasta el desarrollo de pruebas y puesta en funcionamiento. Sin embargo, el desarrollo de este proyecto ha sido muy gratificante y me ha permitido aprender y conocer un gran número de tecnologías y lenguajes de programación nuevos que desconocía.

Escogí este TFG porque estoy muy interesado en el desarrollo de aplicaciones web siguiendo arquitectura de microservicios. También me gustó mucho la propuesta de crear una plataforma de experimentación, me pareció una propuesta extremadamente útil hoy en día. Otra de las principales razones por la que escogí realizar este proyecto fue que al ser un proyecto en colaboración con la empresa HP SCDS me permitiría ver de cerca como se trabaja en el mundo profesional y aprender de buenos profesionales dentro del sector. Finalmente estaba muy interesado en realizar un proyecto utilizando el lenguaje de programación Go que desde que lo descubrí me pareció un lenguaje muy interesante a la hora de hacer desarrollo web y quería aprenderlo más en profundidad.

La experiencia completa del diseño y desarrollo de un software me ha gustado mucho y ha reforzado mi vocación por la ingeniería informática que desde el colegio tanto me ha llamado la atención. Me gustaría seguir participando en proyectos similares en un futuro.

La ayuda recibida por los tutores ha sido impecable, me han sabido guiar y ayudar en todo momento y he podido ampliar enormemente mis conocimientos adquiridos durante la carrera.

En conclusión, valoro mucho la experiencia adquirida en el desarrollo de este proyecto y me gustaría mucho poder seguir participando en distintos proyectos de desarrollo a lo largo de mi carrera profesional.

## 11 Referencias

- [1] Firebase AB testing: <https://firebase.google.com/docs/ab-testing> (15/06/22)
- [2] Optimizely: <https://www.optimizely.com/> (15/06/22)
- [3] Adobe Target: <https://business.adobe.com/es/products/target/adobe-target.html> (15/06/22)
- [4] JWT (JSON Web Token): <https://jwt.io/> (15/06/22)
- [5] Encuesta StackOverflow:  
<https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-programming-scripting-and-markup-languages> (15/06/22)