

Aplicación de red social de vida saludable



**UNIVERSIDAD
DE GRANADA**

Trabajo de Fin de Máster

María Jesús López Salmerón

Índice

CAPÍTULO 1. Introducción	5
1.1. Motivación	5
1.2. Objetivos del proyecto	6
1.3. Características deseadas	7
1.5. Planificación temporal	8
CAPÍTULO 2. Fundamentos / Estado del arte	11
2.1 Accesibilidad	11
Guía del Ministerio de Asuntos Económicos y Transformación Digital [3]	12
Propiedades de los componentes de la interfaz	12
Navegación	13
Diseño de la interfaz	15
Ayuda de entrada	17
Adaptabilidad temporal	17
Alternativas de entrada	18
Alternativas de salida	19
Elementos molestos innecesarios	19
WCAG [4, 5, 6]	19
Perceptible	20
Operable	24
Comprensible	28
Robusto	31
2.2. Apps similares	32
2.3. Tecnologías útiles para el proyecto	37
Frameworks	37
Bases de datos	41
Alojamientos webs	44
Marcos frontend	50
Herramientas de HTML	52
Iconos	54
CAPÍTULO 3. Desarrollo	57
3.1. Descripción de la aplicación a crear	57
3.2. Arquitectura del sistema	60
3.3. Metodología	61
3.4. Glosario de términos	63
3.5. Product Backlog	64
3.6. Sprint Backlog	72
3.7. Primer sprint	75
Historias de Usuario	75
Bocetos	78
Base de datos	85
Implementación	86

Configuración del servidor y de la base de datos	86
Registro	87
Pin parental	91
Inicio de sesión	95
Recuperación de cuenta	97
Pruebas	105
Retrospectiva	107
3.8. Segundo sprint	107
Historias de Usuario	107
Bocetos	110
Base de datos	115
Implementación	115
Perfil del usuario	115
Cierre de sesión	117
Edición del perfil	117
Cancelación de la cuenta	119
Pruebas	120
Retrospectiva	121
3.9. Tercer sprint	121
Historias de Usuario	121
Bocetos	126
Base de datos	140
Implementación	141
Creación de retos	141
Listado de retos y filtraciones	146
Visualización de retos	153
Pruebas	159
Retrospectiva	160
3.10. Cuarto sprint	161
Historias de Usuario	161
Bocetos	163
Base de datos	166
Implementación	167
Pruebas	170
Retrospectiva	170
3.11. Quinto sprint	171
Historias de Usuario	171
Bocetos	174
Base de datos	178
Implementación	179
Buscador y petición de amistad de nuevos amigos	179
Listado de amigos	181
Perfil de amigo	182
Pruebas	184

Retrospectiva	184
3.12. Sexto Sprint	185
Historias de Usuario	185
Bocetos	189
Base de datos	192
Implementación	193
Eliminación, inserción y listado de animadores	193
Listado y envío de mensajes de ánimo	197
Pruebas	201
Retrospectiva	202
3.13. Séptimo Sprint	202
Historias de Usuario	202
Bocetos	208
Base de datos	215
Implementación	215
Inserción, eliminación y listado de participantes	215
Edición, eliminación e inicio de retos y cambio de coordinador	218
Pruebas	223
Retrospectiva	223
3.14. Octavo Sprint	224
Historias de Usuario	224
Bocetos	225
Base de datos	230
Implementación	230
Listado de notificaciones	230
Petición de amistad	233
Pruebas	238
Retrospectiva	238
CAPÍTULO 4. Conclusiones y Trabajos futuros	239
4.1. Conclusiones	239
4.2. Valoración personal	245
4.3. Trabajos futuros	246
ANEXO 1. Diagramas finales del proyecto	249
Diagrama de Entidad/Relación	249
Diagrama de clases	251
Estructura interna de ficheros del proyecto	253
ANEXO 2. Diseño de los bocetos según la guía de accesibilidad	256
Bibliografía	265

CAPÍTULO 1. Introducción

1.1. Motivación

Todo el mundo se propone distintos tipos de retos en su día a día, aunque mayormente se hace de forma inconsciente. Esos desafíos pueden ser desde ahorrar una cierta cantidad de dinero para invertirlo en una necesidad o capricho (como la compra de una casa o el viaje de sus sueños), cambiar ciertos hábitos (como dejar de fumar o hacer, de forma más continuada, ejercicio físico), realizar ciertas acciones para superar sus peores miedos (por ejemplo, hablar en público) o aumentar sus conocimientos (tales casos como aprobar unos exámenes o unas asignaturas o aprender un idioma nuevo).

Sin embargo, el principal problema que tenemos cada uno de nosotros es que, cuando vemos que no podemos, o que pensamos que no es útil o no tenemos los apoyos o los incentivos suficientes para terminarlo, nos rendimos fácilmente aun habiendo hecho grandes avances para lograr la meta.

Una idea para solucionar este problema es tener, de alguna forma, un sistema en el que nos recuerde nuestro objetivo, la razón por la que lo realizamos y nuestro avance hacia la meta. También ayuda compartir nuestra intención con otros y recibir sus apoyos.

Hay algunas aplicaciones o páginas webs que nos pueden ayudar a conseguir nuestros objetivos. Sin embargo, tienen limitaciones, por ejemplo, la mayoría de ellas son para unos retos específicos (tanto los objetivos como los pasos a realizar están ya prefijados por los propietarios de estas), no nos recuerdan la meta por la que decidimos plantear este desafío, no hay apenas apps en las que se puedan desarrollar retos que mejoren la salud mental, o no son totalmente accesibles para cualquier usuario, como las personas con discapacidad física e intelectual. Estos últimos requisitos se plantean porque es un centro de educación especial, Fundación Purísima Concepción de Granada, el que nos ha propuesto que desarrollemos una aplicación con estas características y funcionalidad para sus usuarios.

Las características ausentes de aplicaciones ya existentes son las causantes de la propuesta de este proyecto cuyo objetivo principal es crear una red social, para cualquier persona y para cualquier clase de dispositivo, en la que nos propongamos un reto personalizado (tanto el objetivo a conseguir como los pasos a realizar) y, cuando estemos en nuestros momentos más negativos durante el desarrollo de ese desafío, recordemos por qué

lo planteamos, el avance obtenido desde el momento que se comenzó y el apoyo de nuestros seres queridos para lograr esa meta tan esperada.

Por lo tanto, el código desarrollado a lo largo de este proyecto encargado de generar cada una de las funcionalidades deseables dentro de esta red social lo encontramos dentro un repositorio público de la plataforma de control de versiones [GitHub](#).

1.2. Objetivos del proyecto

Una vez comentados los motivos por los que decidimos comenzar con la aventura de crear una aplicación de una red social que nos ayude a tener una vida saludable apta para cualquier persona, es momento de centrarnos en los objetivos que debemos cumplir a lo largo de este proyecto.

Por lo tanto, los objetivos planteados que puedan lograr satisfacer las necesidades detalladas en el apartado anterior, son:

- En primer lugar, hay que realizar una planificación temporal del trabajo que necesitamos realizar para llevar a cabo este proyecto, como su investigación, el diseño y el desarrollo de la aplicación y el coste que tendría llevarlo a cabo.
- Planificado temporalmente el proyecto, debemos de investigar distintas guías de accesibilidad del momento que sean aptas para el tipo de proyecto que deseamos desarrollar para conocer qué tipo de características debe tener la interfaz de la aplicación.
- Cuando se termine de diseñar la aplicación, debemos repasar que la aplicación cumpla con las guías de accesibilidad que hemos encontrado y realizar los cambios necesarios sobre los bocetos, si estos no cumplen con las normas impuestas en las guías, para lograr generar una aplicación accesible.
- A continuación, tenemos que mirar qué aplicaciones hay en el mercado, que se trate sobre la realización de retos saludables, y analizar las características que tengan cada una de ellas.
- Realizada la investigación de guías de accesibilidad y de aplicaciones similares a la que deseamos crear, hay que pensar qué tipos de tecnologías necesitamos para desarrollar e implementar la aplicación y, dentro de cada tipo, ver qué herramientas nos encontramos que sean útiles para nuestro proyecto.

- Una vez identificadas las tecnologías que vamos a utilizar, debemos desarrollar el proyecto de la siguiente forma:
 - Diseñar las funcionalidades que deseamos plasmar en nuestra aplicación y detallar cómo debería realizarse.
 - Crear los bocetos en donde se podrán llevar a cabo las funcionalidades anteriores.
 - Diseñar, a través de un diagrama de Entidad/Relación, la base de datos que se necesita para almacenar y manejar los datos de la aplicación.
 - Plantear la estructura que va a tener la aplicación dentro del framework seleccionado.
 - Generar el diagrama de clases que nos ayude a ver la forma en la que debería implementar las funcionalidades de la aplicación.
 - Implementar las funcionalidades, según se haya detallado y según estén plasmadas en los correspondientes diagramas, estructuras y bocetos.
 - Realizar las pruebas necesarias para verificar que se cumple las restricciones y las respuestas detalladas en cada una de las funcionalidades.

1.3. Características deseadas

Para lograr esa aplicación ideal para todos los gustos y perfiles, es necesario que tenga implementadas las siguientes características:

- Poder utilizarla en cualquier dispositivo, sin importar si se utiliza en un ordenador, en una tablet o en un smartphone, ni el sistema operativo que use, ni si es de última generación o no.
- Generar una aplicación que no requiera de conocimientos en tecnología por parte del usuario, puesto que será intuitiva y sencilla de utilizar.
- Poder ser utilizada por cualquier persona, aunque esta tenga ciertos problemas físicos, al implementar ciertas técnicas de accesibilidad (por ejemplo, dictar textos, transcribir audios o navegación a través del teclado), aprobadas por organizaciones tanto nacionales como internacionales, que ayuden a manejarla.

- Crear cualquier reto indicando el objetivo, su categoría y su recompensa.
- Decidir los pasos a realizar para alcanzar el objetivo de un reto.
- Añadir amigos a mi desafío para que lo hagan conmigo.
- Incorporar amigos para que me apoyen, a través de mensajes de ánimo, durante el desarrollo del reto.
- Insertar evidencias para en el futuro poder ver el avance obtenido hasta el momento.
- Calificar cómo me he sentido haciendo uno de los pasos del reto.

1.5. Planificación temporal

En cuanto a la planificación realizada para poder llevar a cabo este proyecto (el desarrollo de una aplicación que ofrezca una la red social de vida saludable), es aquella que mostramos en el siguiente diagrama de Gantt.

En este diagrama podemos ver que crear la red social, junto a sus previas investigaciones, ha durado aproximadamente año y medio puesto que he estado compaginando la realización de este proyecto junto a mi trabajo de jornada completa (incluyendo horas de cursos y otros conocimientos necesarios para emprender mi trabajo dentro de mi empresa). Por lo tanto, las jornadas de trabajo dentro de este proyecto no han superado las cuatro horas diarias.

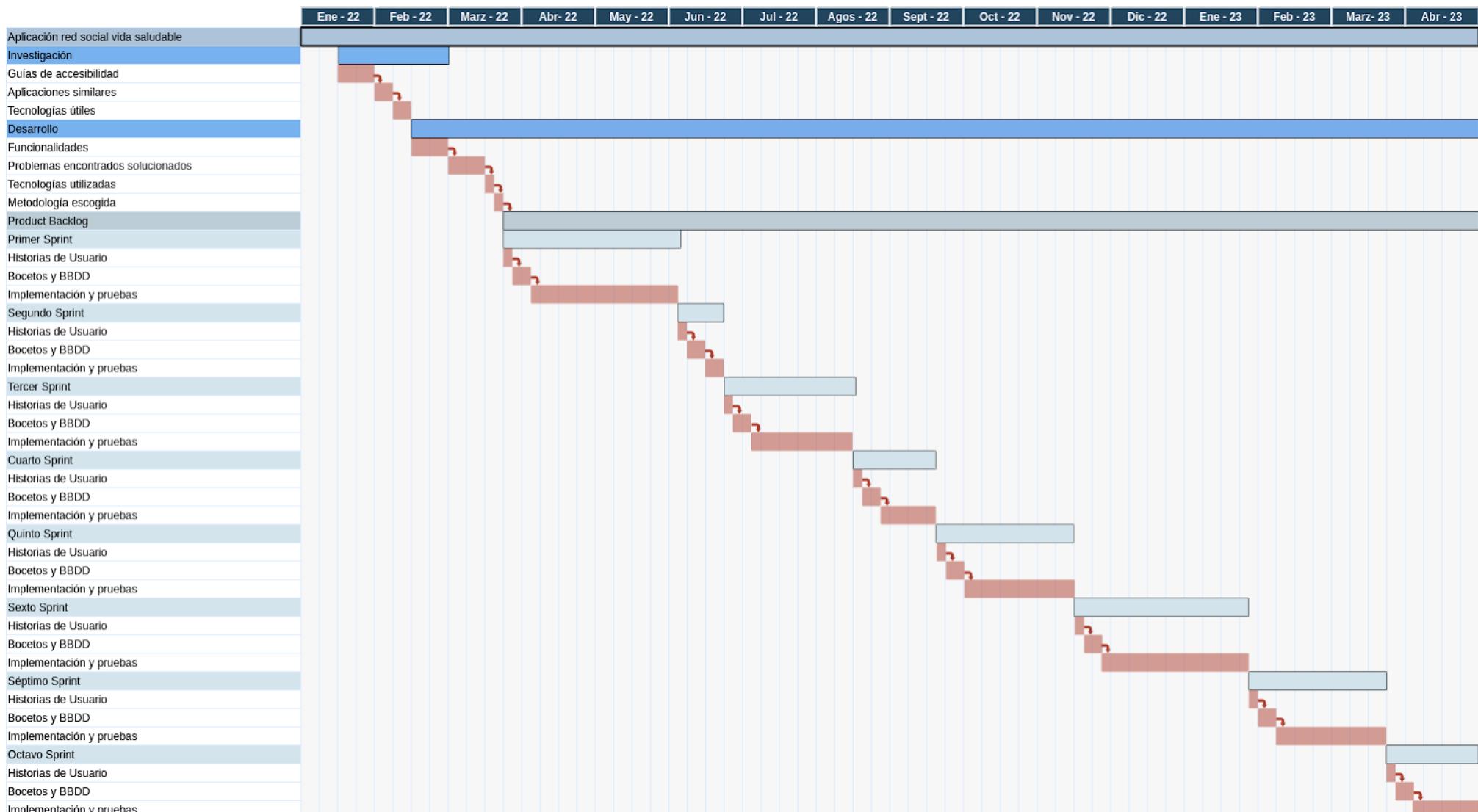


Figura 1: Diagrama de Gantt de la planificación de aplicación de la red social saludable.

Por último, el presupuesto que tendría si este proyecto lo hubiera vendido realmente a un cliente sería el que vemos desglosado a continuación.

Gastos elegibles	Importe solicitado
GASTOS DE PERSONAL	20.000 euros
Ingeniero informático (10 eur/h, 4h/día, 500 días)	20.000 euros
GASTOS DE EJECUCIÓN	
Costes de adquisición de material inventariable	240 euros
Monitor	70 euros
Tablet	100 euros
Móvil	70 euros
Costes de consultoría, prestación de servicios, suministros, etc	920 euros
Gestoría para contratos y balances	500 euros
Gastos de mantenimiento del servidor durante un año	420 euros
Costes de investigación contractual, conocimientos técnicos y patentes	350 euros
Solicitud de registro de propiedad intelectual	150 euros
Solicitud de marca	200 euros
GASTOS COMPLEMENTARIOS	
Otros gastos de funcionamiento derivados de la actividad de desarrollo	1.000 euros
Instalación de la aplicación en el servidor	500 euros
Implementación de base de datos y almacenamiento interno	500 euros
TOTAL	22.510 euros

Tabla 1: Presupuesto desglosado de los gastos de gestión y creación del proyecto.

CAPÍTULO 2. Fundamentos / Estado del arte

Como hemos comentado en la introducción de esta memoria, nuestro objetivo es poder crear una aplicación en la que cualquier persona, sin importar los conocimientos o las discapacidades que tenga para poder manejar un dispositivo o una aplicación, en la que pueda marcarse unos objetivos que desee cumplir y poder gestionar su evolución a lo largo de su realización.

A lo largo de este capítulo, vamos a comentar las distintas guías de accesibilidad que hay en la actualidad para considerarlas en nuestro diseño de la aplicación, revisaremos algunas de las aplicaciones que hemos hallado con características similares a las que deseamos que tenga nuestra aplicación y, por último, analizaremos las distintas tecnologías que necesitaremos para el desarrollo de la aplicación y las variedades que hay de cada una de ellas que nos puedan servir para este proyecto.

2.1 Accesibilidad

En esta sección, hablaremos con detalle de las distintas guías que hay para que una aplicación sea accesible. Antes de comentar las distintas directrices para considerarla, debemos explicar qué es una aplicación accesible.

Se considera cualquier aplicación accesible cuando cualquier usuario, sin importar sus discapacidades cognitivas o físicas, pueda utilizar todas sus funcionalidades de forma natural, fácil y cómoda al tener la aplicación un diseño de interfaz comprensible, intuitivo y utilizable [1, 2]. Por lo tanto, la aplicación debe contener las características necesarias para que cualquier persona pueda acceder a ella, como por ejemplo posibilidad de adaptación de tamaño o color de los elementos visuales de la página, convertir a audio los textos o convertir a texto los audios, dictar a la aplicación las palabras que se quiera insertar en ella o incluir pictogramas que reflejen el significado de un campo o valor de un formulario.

A continuación, comenzaremos a explicar brevemente los pasos que hay que seguir para que la aplicación sea accesible en dos guías distintas: según el *Ministerio de Asuntos Económicos y Transformación Digital* del Gobierno Español y según la organización World Wide Web Consortium en las guías *Web Content Accessibility Guidelines (WCAG)* para desarrollo de aplicaciones web tanto para escritorios, como tablets y móviles.

Guía del Ministerio de Asuntos Económicos y Transformación Digital [3]

El *Ministerio de Asuntos Económicos y Transformación Digital* encomienda las siguientes recomendaciones para poder generar cualquier aplicación lo más accesible posible:

Cuando se habla de accesibilidad TIC, se refiere a que cualquier tipo de usuario puede acceder a toda la información y funcionalidades ofrecidas por una aplicación. De no ser así, dejaría de ser accesible. Esto puede suceder porque no se haya planteado bien la aplicación o por causas ajenas como que las plataformas móviles para su creación no soporten totalmente los servicios de accesibilidad requeridos.

Las distintas aplicaciones que se realicen y quieran evitar los problemas de accesibilidad deben ser corregidas en las etapas más tempranas de su desarrollo, puesto que será más fácil y más barato hacerlo poco a poco que una vez puestas en el mercado.

A continuación, se comentan las diferentes directrices, comunes a todos los sistemas operativos móviles, de las cuales se puede proporcionar accesibilidad a cualquier aplicación:

Propiedades de los componentes de la interfaz

- *Nombre del componente.* Debe:
 - No incluir el tipo de elemento, repetición de palabras dentro de la herramienta.
 - No describir visualmente el componente, sino describir su acción.
 - No usar nombres repetidos.
 - Contener el texto visible de la etiqueta o que contenga el nombre accesible al texto visible.
- *Rol.* Hace referencia al tipo de componente gráfico que está representando.
- *Estado.* Indica en qué situación se encuentra el componente gráfico.
- *Valor.* Propiedad utilizada más comúnmente para componentes interactivos de edición.
- *Descripción.* Es una propiedad optativa que se debe utilizar cuando un componente es ambiguo mediante el uso del nombre.

- *Acceso mediante lenguaje de interfaz.* Acceder a las propiedades anteriores mediante el propio lenguaje que se utilice para la definición de la interfaz gráfica de la aplicación.
- *Acceso mediante programación.* Para aquellos componentes de la interfaz que se deban cargar de manera dinámica, las plataformas deben proporcionar el acceso programático a los atributos de esos componentes mediante la capa de accesibilidad.
- *Componentes estándar del sistema.* Son componentes que tienen en común los fabricantes de distintas plataformas para que puedan integrar la accesibilidad en la capa de componentes gráficos predefinidos dentro de las API de sus sistemas. Recomendables para que la interfaz sea compatible con los servicios de accesibilidad y con los productos de apoyo. Estos implican los siguientes aspectos:
 1. Añadir texto descriptivo a los controles de la interfaz.
 2. Comprobar que se puede acceder a todos los componentes que acepten una interacción con un controlador direccional.
 3. Comprobar que los mensajes de audio estén acompañados por una alternativa visual o táctil.
 4. Comprobar su correcto funcionamiento utilizando únicamente los servicios y características de navegación accesibles.
- *Componentes personalizados.* Son componentes gráficos creados por nosotros mismos, aprovechando las funcionalidades de las APIs de los sistemas, para alcanzar la funcionalidad que requiere nuestra aplicación. Además de la visualización del componente o de su correspondiente comportamiento, hay que transmitir la información que se desea a las herramientas de apoyo. Es necesario:
 - Garantizar la accesibilidad añadiendo la información apropiada a cada atributo de los componentes de la interfaz.
 - Comprobar que se proporciona la información de accesibilidad necesaria para que los servicios de accesibilidad y los productos de apoyo funcionen correctamente.

Navegación

Es una de las partes más importantes a tener en cuenta. Es esencial que el usuario sepa en qué pantalla se encuentra y cómo desplazarse a otra en todo momento. Por lo tanto,

es necesario que la disposición visual sea lo más consistente posible y se haga scroll lo menos posible. Las características que se utilizan para la navegación son los siguientes:

- *Foco del sistema.* Puntero hacia un componente gráfico que principalmente es más necesario para aquellas personas que utilizan una navegación secuencial puesto que van saltando de componente en componente hasta llegar a aquel que desea activar.
- *Visibilidad del foco.* Permitir que el foco sea visible siempre en su desplazamiento a lo largo de la interfaz. Si la capa de accesibilidad de la plataforma no la ofrece, el desarrollador debe proporcionarla.
- *Elementos receptores del foco.* Sólo deben recibir el foco aquellos elementos que permitan realizar una cierta funcionalidad o sean imprescindibles para comprender totalmente la información que se muestra por pantalla.
- *Orden de navegación.* El foco no se desplaza teniendo en cuenta la disposición visual, sino un orden en el que se encuentran los componentes siguiendo un esquema de árbol, lo que puede ser útil a la hora de eliminar información innecesaria puramente decorativa.
- *Contenido adicional.* Debe mostrarse cuando el puntero esté sobre él, con la posibilidad de moverse sin perder el contenido hasta que se retire el foco sobre él.
- *Cambio de contexto.* Aquel evento que desencadena un cambio en el lienzo de la aplicación que previamente ha de ser notificado al usuario si se produce de manera automática. En ningún caso se debería recargar el lienzo automáticamente sin una acción explícita del usuario.
- *Pasos de navegación.* Los pasos que un usuario debe dar para alcanzar la funcionalidad deseada deben ser los menos posibles (como regla general son 3 pasos).
- *Navegación por procesos.* Para aquellas tareas que requieren una navegación a lo largo de un proceso, éste debe ser secuencial y no terminar hasta que se alcanza la última pantalla. Además se debe incluir una pantalla con el progreso completado de esa tarea, e incluso dar la posibilidad de poder navegar hacia delante y retroceder.
- *Cancelación del puntero.* Para prevenir interacciones accidentales al pulsar sobre algún componente sin querer. La mejor forma de evitarlo es lanzando acciones evento *up* (cuando levantamos el dedo de la tecla física o de la pantalla) en vez de evento *down* (cuando presionamos sobre la pantalla o tecla física) para que, si se

pulsa sin querer, se centre el puntero en el foco y poder cancelar esa operación no deseada deslizando el puntero fuera de esa área.

- *Activación mediante movimiento.* Cualquier funcionalidad que se pueda operar con el movimiento del dispositivo se puede ejecutar a través de la interfaz y se debe poder desactivar dicha operación a través de un movimiento específico prefijado previamente o mediante un botón de activación dentro de la aplicación. Se pueden activar mediante sensores que responden a movimientos como la inclinación, el balanceo o las sacudidas del dispositivo.

Diseño de la interfaz

- *Tamaño.* Para una correcta visualización de la interfaz, se deben cumplir las siguientes condiciones:
 - Las distintas áreas interactivas de los componentes deben ser espaciosas para que no se produzcan errores o la imposibilidad de llegar a esa funcionalidad.
 - El área útil de interacción debe ser entre 7 y 10 mm, pudiendo ser mayor si se desea.
 - El área de relleno que rodea un componente también sirve como zona de interacción del mismo.
 - La separación entre las distintas áreas interactivas debería ser de al menos 1 mm para evitar la activación involuntaria de funciones.
 - El tamaño del texto debe ser suficiente para albergar cualquier información internacionalizada. Además, se debe dar la posibilidad de agrandarlo.
- *Contraste.* Dependiendo del tipo de componente de la interfaz, se realizarán distintas medidas:
 - Texto. Si es pequeño (menos de 18pt en normal o 14pt en negrita), el contraste debe ser una relación mínima de 4:5:1. Si es grande (mayor a 18pt en normal o 14pt en negrita), una relación mínima de 3:1.
 - Contenido no textual. Se debe asegurar un ratio mínimo de contraste de al menos 3:1 en los colores adyacentes de los componentes de la interfaz y de objetos gráficos.

- Indicador del foco. Debe tener un mínimo de contraste con el fondo para poder saber en qué componente está enfocando en ese mismo instante.
 - Se consideran excepciones a estos contrastes aquellos objetos gráficos cuya representación es esencial para transmitir la información, es decir, para no alterar su significado (logos, imágenes de la vida real, ...).
- *Destellos*. La duración de estos no debe superar las 3 veces por segundo para no ocasionar convulsiones o ataques a ciertos usuarios. También es aconsejable evitarlos en superficies grandes o centrales de la pantalla.
- *Lenguaje*. Es esencial para la aplicación, por lo tanto es necesario que se centren en:
 - Los distintos idiomas de la aplicación.
 - Los textos sean los más claros y concisos posible.
 - Los textos deben ser correctos ortográfica y sintácticamente.
 - Si el Sistema Operativo lo permite, se deberían marcar los cambios de idioma en los textos.
 - Un lenguaje sencillo y cercano al usuario, tratando con respeto a cualquier persona.
- *Iconos*. No siempre son reconocibles por todos los usuarios, por lo que se debe proporcionar una alternativa textual. Se puede dejar decidir al usuario si prefiere texto o iconos o mostrar un tooltips cuando se mantiene pulsado sobre un componente.
- *Consistencia*. Ayuda a que los usuarios naveguen de forma más cómoda y eficiente. Hay que considerar que los colores sean homogéneos y seguir las guías de estilo de cada plataforma.
- *Orientación*. Las maquetaciones de las aplicaciones se han de realizar de manera que no restrinjan su visualización a una determinada orientación, excepto en aquellos casos en que es esencial mostrar su contenido en una determinada orientación.
- *Reajuste*. El diseño se debe adaptar correctamente a los distintos tamaños de pantalla.
- *Espaciado del texto*. Se tiene que realizar de forma que no se pierda contenido o funcionalidad cuando el usuario modifique algunas de las características del texto. Para facilitar el acceso se exige:

- Ajustar el alto de la línea hasta al menos 1.5 veces el tamaño de la fuente.
- Espaciar los párrafos hasta el doble del tamaño de la fuente.
- Ajustar el espacio entre letras hasta 0.12 veces el tamaño de la fuente.
- Ajustar el espaciado entre las palabras hasta 0.16 veces el tamaño de la fuente.
- *Agrupaciones y secciones.* Para facilitar la lectura a través de agrupaciones con los distintos componentes de la interfaz, y mediante títulos y cabeceras para su separación.

Ayuda de entrada

- *Autocompletado.* Intentando predecir las palabras que quieren escribir.
- *Portapapeles.* Reduce el esfuerzo de llenar ciertos formularios o documentos.
- *Entrada de voz.* Para aquellas personas que les es imposible o difícil escribir a través de un teclado.
- *Restricciones correctas.* Para orientar al usuario sobre qué es lo que puede insertar y para detectar errores.
- *Mensajes de error concretos.* Deben ser precisos y concretar en qué campo de un formulario se ha producido el problema.

Adaptabilidad temporal

En ciertas ocasiones se pueden presentar contenidos o funciones temporalmente dependientes, pero son difíciles de acceder por ciertos perfiles. En esta clasificación se engloba cualquier elemento o funcionalidad que desaparezca pasado un tiempo. Por lo tanto, se puede solucionar de las siguientes maneras:

- Dejar un plazo determinado considerablemente para completar la tarea.
- Dar la opción de ampliar ese plazo manualmente hasta un límite de 24 horas.

Dichas acciones temporales no deben ser mensajes volátiles cuya información sea crítica, ni ocultar los controles principales.

Además, se debe dejar restaurar fácilmente aquellos controles secundarios que haya activado el usuario a través de alguna funcionalidad de la aplicación que se hayan ocultado pasado un determinado lapso.

Alternativas de entrada

- *Crear nuevos atajos.* Teniendo en cuenta en no interferir en los prefijados por el sistema. Además, no debemos reutilizar gestos que estén ya asignados por otras acciones. Hay distintos tipos de atajos: por movimientos, por voz o por teclado. Los atajos de teclado debe cumplir al menos una de las siguientes condiciones:
 - Existencia de un mecanismo que desactive el atajo de teclado o de gesto.
 - Existencia de un mecanismo que permite asignar el atajo de teclado para emplear otra tecla o imprimible (Ctrl, Alt, ...).
 - El atajo sólo está disponible cuando el foco del teclado esté en el componente adecuado.

El objetivo de realizar de esta manera los atajos de teclado es evitar que las personas que interactúen mediante voz activen alguna funcionalidad de forma accidental al pronunciar alguna letra.

- *Indicaciones no dependientes de la entrada.*
- *Utilizable por cualquier método de entrada soportado por el sistema.*
- *Para gestos complicados.* Introducir botones que simulen esos gestos.
- *Para dobles pulsaciones.* Para aquellas personas que no pueden realizar doble pulsación, se debe ofrecer una alternativa que realice la funcionalidad que tiene asignada la doble pulsación.
- *Para pulsaciones mantenidas.* Ofrecer una alternativa que haga su función sin tener que mantener pulsado un elemento.
- *Para combinaciones de teclas.* Los sistemas operativos cuentan con combinaciones de teclado de manera secuencial. Si no, se debe evitar su uso.
- *Asistentes de voz.* Herramientas basadas en la inteligencia artificial que consiste en el procesamiento del lenguaje natural o de comandos para ejecutar ciertas acciones dentro del sistema.

Alternativas de salida

- *Subtítulos*. Para aquellos contenidos que utilicen un canal sonoro, con el cual deben estar ambos sincronizados y ser lo más legibles posibles.
- *Audiodescripción*. Para describir aquellos componentes visuales asíncronamente.
- *Uso del color*. Para representar cualquier tipo de meta-information. Para ayudar a ciertos perfiles, se puede añadir una etiqueta, ícono o similar y no usar solo el color.
- *Alternativa visual*. A veces transmitimos información visual a través de una señal. Pero hay usuarios que son incapaces de detectar dichos cambios, por lo que se necesita un canal alternativo para hacerles llegar la información.
- *Alternativa auditiva*. Se pueden notificar de ciertos cambios a través del canal auditivo. Hay personas que por este medio no lo pueden percibir, por lo tanto es necesario utilizar un canal alternativo al audio que transmite la información.
- *Alternativa háptica*. Avisar al usuario de ciertos cambios a través del tacto, como las vibraciones. Hay dispositivos que ofrecen esta funcionalidad, pero hay muchos otros que no como ciertos tablets o convertibles. Estos terminales tienden a utilizar las mismas plataformas que sus homólogos de telefonía, de modo que se puede ofrecer las notificaciones pertinentes por canales alternativos.

Elementos molestos innecesarios

En las aplicaciones, además de introducir componentes meramente funcionales, hay componentes decorativos. Algunos de ellos pueden llegar a ser intrusivos y dificultar la interacción con la app. Por lo tanto, se debe reducir al mínimo este tipo de elementos y, si son necesarios, poder pausarlos, detenerlos u ocultarlos.

WCAG [4, 5, 6]

Según la guía de accesibilidad de la organización *Web Content Accessibility Guidelines*, una aplicación puede ser accesible a un cierto grado, siendo A el mínimo para declarar a la aplicación como accesible y AAA el máximo. Esta guía se ha ido ampliando y adaptando a la accesibilidad en aplicaciones móviles (desde la versión WCAG 2.0 hasta la última, WCAG 2.2 [7]), conforme esta clase de dispositivos han aumentado su uso y sus capacidades. Las propiedades que toda aplicación debe cumplir son las que se comentan a continuación:

- *Perceptible*. Se comenta las distintas formas en las que debemos crear un componente visualmente dentro de la aplicación.
- *Operable*. Describe cómo deberíamos activar cada una de las funcionalidades de la app.
- *Comprensible*. Explica las distintas maneras de redactar y de mostrar al usuario la información dentro de la aplicación.
- *Robusto*. Especifica las distintas directrices que debe utilizar el desarrollador a la hora de programar la aplicación

Perceptible

- Distinguible. Hace más fácil para el usuario ver y escuchar el contenido incluyendo la separación entre el foreground y el background:
 - Tamaño de la pantalla. Usualmente, utilizamos pantallas pequeñas por lo que hay un límite a la hora de la cantidad de información que se debe mostrar. Unas mejores prácticas a este tipo de pantallas incluyen:
 - Minimizar la cantidad de información en comparación con las pantallas de escritorio. Es decir, mostrar la cantidad de información dependiendo de la pantalla que se está visualizando.
 - Proporcionar un tamaño predeterminado razonable para el contenido y el control de la aplicación.
 - Adaptar la longitud del texto al ancho de la ventana gráfica.
 - Colocación de los campos del formulario debajo de sus etiquetas.
 - Zoom. Requiere que el texto sea redimensionable sin tecnología de asistencia hasta un 200% sin necesidad de desplazamiento horizontal. Para poder realizarlo se pueden utilizar los siguientes métodos:
 - Asegurarse de que el metaelemento de la ventana no bloquee el zoom mediante gesto, para que pueda ampliarse. Se recomienda utilizar técnicas que admitan el cambio de texto sin necesidad de una panorámica horizontal.
 - Compatibilidad con fuentes del sistema que siguen las preferencias del usuario para el tamaño del texto.

- Proporcionar controles para cambiar el tamaño del texto.
- Contraste. Ofrece dos tipos de criterios distintos:
 - Mínimo. Requiere un contraste de al menos 4,5:1 (o 3:1 para texto a gran escala).
 - Mejorado. Requiere un contraste de al menos 7:1 (o 4,5:1 para texto a gran escala).
 - En cuanto al texto o a las imágenes que forman parte de un componente inactivo (componentes que no tienen ninguna funcionalidad dentro de la aplicación) y a los logotipos, no tienen requisitos de contraste.
- Uso del color. No se utiliza únicamente como un medio visual para transmitir información, indicar una acción, provocar una respuesta o distinguir un elemento visual de otro.
- Control de audio. Si la aplicación incluye un audio que se ejecuta automáticamente y tiene más de 3 segundos de duración, que aparezcan los botones de control de audio (pause, stop y control del volumen independiente del sistema operativo).
- Imágenes de texto. Solo se debe usar para los siguientes casos:
 - Si se puede personalizar según los requisitos del usuario.
 - Si es una presentación particular del texto esencial para la información que transmite.
- Audio de fondo. Puede presentarse de las siguientes maneras:
 - Sin sonidos de fondo.
 - Con la posibilidad de apagarlos.
 - Con menos de 20 decibelios que el del primer plano, excepto los sonidos ocasionales que duren entre uno y dos segundos.
- Presentación visual. Para que sea lo más visible posible se deben aplicar los siguientes mecanismos:
 - El usuario puede seleccionar los colores del primer plano y del fondo.

- El ancho no supera los 80 caracteres.
 - El texto no está justificado (alineado a los márgenes izquierdo y derecho)
 - La altura de línea debe ser de al menos 1,5 veces el tamaño de la fuente.
 - El espacio entre párrafos debe ser de al menos 2 veces el tamaño de la fuente.
 - El espacio entre letras debe ser de al menos 0,12 veces el tamaño de la fuente.
 - El espaciado entre palabras debe ser de al menos 0,16 veces el tamaño de la fuente.
- Reorganización del texto en una página. El contenido se puede presentar sin pérdida de información o funcionalidad, y sin necesidad de desplazarse en dos dimensiones, excepto cuando las partes del contexto requieran capas bidimensionales para su uso o significado, para:
- Contenido de desplazamiento vertical con un ancho equivalente a 320px.
 - Contenido de desplazamiento horizontal a una altura equivalente a 256px.
- Contraste sin texto. La presentación visual que debe tener una relación de contraste de al menos 3:1 frente a los colores adyacentes son los siguientes elementos:
- Componentes de interfaz de usuario necesarios para diferenciarlos unos de otros y estados de la interfaz, excepto en el caso de los componentes inactivos o cuando la apariencia del componente la determina el agente de usuario (aplicación informática que funciona como cliente en un protocolo de red, como navegadores o herramientas encargadas de la creación de páginas como WordPress) y el autor no la modifica.

- Objetos gráficos necesarios para comprender el contenido, excepto cuando sea una presentación particular de gráficos esencial para la información que se transmite.
- Contenido en Hover (cuando el ratón está sobre un elemento) o Focus (cuando el cursor está en ese elemento). Dónde apuntar o desapuntar el puntero o el foco del teclado desencadena contenido adicional para que se vuelva visible y luego se oculte.
- Medios de comunicación en el tiempo. Recomienda las siguientes alternativas:
 - Vídeo o audio pregrabado cuando es una alternativa para presentar información equivalente para el contenido de texto.
 - Subtítulos para todo aquel contenido de audio pregrabado o en directo en medios sincronizados que no son una alternativa para el texto que pueda contener la aplicación.
 - Proporcionar una pequeña o extendida descripción del audio o del vídeo para aquellos medios sincronizados.
 - Incorporar interpretación en el lenguaje de signos en todos los audios.
- Adaptable. El contenido se puede presentar de diferentes maneras sin perder información o estructura.
 - La información, la estructura y las relaciones transmitidas a través de la presentación se pueden determinar mediante programación o están disponibles en el texto.
 - Cuando un párrafo en el que se presenta el contenido afecta a su significado, se puede determinar mediante programación una secuencia de lectura correcta.
 - Las instrucciones aportadas para la comprensión y para poder operar sobre la aplicación no se basan únicamente en las características sensoriales de los componentes como la forma, el color, el tamaño, la localización, la orientación o el sonido.
 - El contenido no debe restringirse a una única orientación de visualización, excepto que sea esencial una orientación específica para su funcionamiento.

Operable

- Control de teclado para dispositivos con pantalla táctil.
 - Todas las funcionalidades se deben operar a través de la interfaz del teclado sin requerir tiempos específicos para pulsar ciertas teclas individuales. También se puede utilizar cuando se requiere una entrada que depende de la ruta de movimiento del usuario y no solo de los puntos finales.
 - Si el foco del teclado se puede mover a un componente de la aplicación usando el teclado, entonces dicho foco se puede alejar de ese componente usando la interfaz de teclado y, si requiere más elementos además de las flechas y el tabulador, se informa al usuario de esos métodos de movimiento del foco.
 - Si se implementa un atajo de teclado utilizando solo letras (tanto en minúscula como en mayúscula), signos de puntuación, números o símbolos:
 - Debe haber un mecanismo para desactivar el atajo.
 - Debe haber un mecanismo para reasignar el atajo para usar uno o más caracteres no imprimibles.
 - Debe activarse solo cuando el foco esté en un componente específico.
- Tiempo suficiente para que el usuario pueda leer y utilizar el contenido.
 - El tiempo no es una parte esencial del evento o actividad presentada por el contenido, excepto de los medios sincronizados no interactivos y los eventos en tiempo real.
 - Si hay un límite de tiempo establecido:
 - Debe el usuario poder desactivarlo o ajustarlo.
 - Debe poder ampliarse el tiempo y avisar al usuario antes de que expire el tiempo. Excepto cuando el tiempo sea una parte requerida de un evento en tiempo real, que el límite sea imprescindible y si se permitiera su ampliación invalidaría la actividad o que el límite de tiempo supere a las 20 horas.
 - Para aquella información en movimiento, en parpadeo, en desplazamiento o que se actualice automáticamente que comience automáticamente, dure más

de 5 segundos y se presente en paralelo con otro contenido, debe haber un mecanismo con el que el usuario pueda pararlo, detenerlo u ocultarlo, excepto si es una parte de una actividad esencial.

- Las interrupciones pueden ser pospuestas o eliminadas por el usuario, excepto si implican una emergencia.
 - Cuando caduque la sesión de un usuario, este puede continuar con la actividad sin pérdida de datos después de volver a autenticarse.
 - Advertir al usuario sobre la duración de cualquier inactividad que pueda causar la pérdida de datos, a menos que estos se conserven durante más de 20 horas.
- Convulsiones y reacciones físicas:
 - La aplicación no contiene nada que parpadee más de tres veces en cualquier período de un segundo.
 - La animación de movimiento desencadenada por la interacción se puede desactivar, a menos que sea esencial para la funcionalidad o para la información que transmite.
 - Navegabilidad:
 - Mecanismo para eludir bloques de contenido que se repiten en varias páginas.
 - Títulos que describan el tema o el propósito de la página.
 - Si se navega secuencialmente por la aplicación y afectan el significado o la operación, los componentes enfocables reciben el enfoque en un orden que preserva el significado y la operatividad.
 - Hay un mecanismo disponible para permitir que el propósito de cada enlace se identifique solo a partir del texto, excepto cuando el propósito sea ambiguo para los usuarios.
 - Los encabezados y las etiquetas describen el tema o el propósito.
 - Cualquier interfaz operable por teclado tiene un modo de operación donde el indicador de enfoque del teclado es visible.
 - Los encabezados de las secciones se utilizan para organizar el contenido.

- Apariencia del enfoque:
 - Mínimo: Cuando reciben el foco del teclado los componentes:
 - Área de contraste: relación de contraste de al menos 3:1 entre los colores en los estados de enfocado y desenfocado.
 - Área mínima: el área de contraste es al menos tan grande como:
 - El área de un perímetro de 1 píxel CSS de espesor del componente desenfocado
 - El área de una línea de 4 píxeles CSS de grosor a lo largo del lado más corto de un cuadro delimitador mínimo del componente desenfocado, y no menos de 2 píxeles CSS.
 - El área de contraste tiene una relación de contraste de al menos 3:1 frente a los colores adyacentes en el componente enfocado, o el área de contraste tiene un grosor de al menos 2 píxeles CSS.
 - El elemento que tiene el foco no está completamente oculto por el contenido creado por el autor.
 - Mejorado: Cuando los componentes reciben el foco:
 - Área de contraste: relación de contraste de al menos 4,5:1 entre los colores en los estados desenfocado y enfocado.
 - Área mínima: el área de contraste es al menos el doble del área de un perímetro de 1 píxel CSS del componente desenfocado.
 - El indicador de enfoque no está completamente oculto por ningún contenido del autor.
 - Para el contenido web con localizadores de saltos de página, hay un mecanismo para navegar a cada localizador.
- Modalidades de entrada. Facilitar la operación de la funcionalidad a través de varias entradas más allá del teclado.

- Se puede operar usando un solo puntero si:
 - El evento descendente no se utiliza para ejecutar ninguna parte de la función.
 - Hay un mecanismo disponible para abortar la función antes de la finalización o deshacer la función después de la finalización.
 - Cuando se levanta el dedo del ratón o del botón del teclado invierte cualquier resultado del evento de cuando se pulsa el botón anterior.
 - Completar la función cuando se ha pulsado cualquier botón es esencial.
- El nombre de los componentes de interfaz con etiquetas contiene el texto que representa visualmente.
- La funcionalidad que puede ser operada por el movimiento del dispositivo o del usuario también puede ser operada por los elementos de la interfaz y ser desactivada, excepto cuando:
 - El movimiento se utiliza para operar la funcionalidad a través de una interfaz compatible con accesibilidad.
 - El movimiento es esencial para la función.
- Tamaño y espaciado del componente táctil. Los componentes de la interfaz deben ser lo suficientemente grandes y espaciosos entre ellos para que el usuario pueda clickear sobre el elemento deseado. Para ello hay que:
 - Asegurarse de que los objetos tengan al menos 9 mm de alto y 9 mm de ancho.
 - Asegurarse de que los objetos cercanos al tamaño mínimo estén rodeados por una pequeña cantidad de espacio interactivo.
 - Asegurar que el objeto está disponible a través de un enlace o control equivalente en la misma página.
 - Garantizar que el objeto está en una oración o bloque.
 - Rectificar que el tamaño del objeto lo determina el agente de usuario y no lo modifica el autor.

- El contenido web no restringe el uso de las modalidades de entrada, excepto cuando la restricción sea necesaria para garantizar la seguridad del contenido o para respetar la configuración del usuario.
- Gestos de pantalla táctil. Algunas mejores prácticas incluyen:
 - Los gestos de las aplicaciones deben ser fáciles de realizar.
 - Activación de elementos a través del evento mouseup o touchend.
 - Los gestos que se crean para el uso en la aplicación no deben coincidir con los ya creados para la utilización dentro del sistema operativo.
- Gestos de manipulación de dispositivos. Muchas tecnologías permiten activar ciertos controles manipulando físicamente el dispositivo. También se debe de poder realizar dichas acciones a través del teclado.
- Colocación de botones donde sean de fácil acceso. Situarlos donde se puedan alcanzar fácilmente cuando el dispositivo se sostiene en diferentes posiciones.
- Los targets tienen un área de al menos 24x24 píxeles CSS excepto donde:
 - El desplazamiento del target es de al menos 24 píxeles CSS para cada target adyacente.
 - El target está en un bloque de texto.
 - Una presentación particular del target es esencial para la información que transmite.

Comprendible

- Leíble. Hacer que el contenido textual sea leíble y comprensible.
 - El idioma predeterminado de la app se puede determinar mediante programación, excepto de los nombres propios, vocabulario específico, términos técnicos y palabras propias del idioma indeterminado.
 - Hay un mecanismo disponible para identificar definiciones específicas de palabras o frases utilizadas de manera inusual o restringida.

- Se encuentra disponible un mecanismo para identificar la forma expandida el significado de las abreviaturas.
 - Está disponible un mecanismo para identificar la pronunciación de las palabras para conocer su concepto dentro del diccionario, en contexto, es ambiguo sin conocer la pronunciación.
- Predecible. Crear aplicaciones que se muestren y funcionen de manera predecible.
 - Cuando cualquier componente de la interfaz de usuario recibe el foco, no inicia un cambio de contexto.
 - Cambiar la configuración de cualquier componente de la interfaz no provoca un cambio de contexto a menos que se haya informado previamente al usuario sobre el comportamiento.
 - Los componentes que tienen la misma funcionalidad dentro de un conjunto de páginas se identifican de forma coherente.
 - Los cambios de contexto se inician sólo por solicitud del usuario o a través de un mecanismo disponible.
 - Para cada página dentro de un conjunto de páginas que proporciona una o más formas de encontrar la ayuda sin necesidad de complicar la búsqueda de ayuda dentro de la navegación de la aplicación:
 - Detalles de contacto (número de teléfono, correo electrónico, horario de atención al cliente, ...).
 - Mecanismo de contacto humano (mediante chats, formularios de contacto o canal de redes sociales).
 - Opción de autoayuda (una página de preguntas frecuentes).
 - Mecanismo de contacto totalmente automatizado, algún programa que intente ayudar con el problema del usuario de forma automatizada.
 - Cuando el cursor del puntero o el enfoque del teclado hace visible un componente de la interfaz, mostrar la información necesaria para identificar que los componentes están disponibles, excepto cuando:
 - La información necesaria para identificarlos está disponible a través de un componente equivalente que está visible en la misma página o en

un paso diferente en un proceso de varios pasos sin necesidad de pasar el puntero o enfocar con el teclado.

- El componente se proporciona específicamente para mejorar la experiencia de navegación con el teclado.
 - Se dispone de un mecanismo para hacer visible la información de forma persistente.
 - Es esencial ocultar la información necesaria para identificar el componente sin demasiados elementos por pantalla.
- Asistencia de entrada. Ayudar a los usuarios a evitar y corregir errores.
 - Si se detecta automáticamente un error de entrada, se identifica el elemento que tiene el error y se describe el error al usuario en forma de texto. Si se conocen las sugerencias para su corrección, estas se aportan al usuario, a menos que pongan en peligro la seguridad o el propósito del contenido.
 - Se proporcionan etiquetas o instrucciones cuando el contenido requiere la intervención del usuario.
 - La ayuda sensible al contexto está disponible.
 - Para las aplicaciones que requieren que el usuario envíe información:
 - Los envíos son reversibles.
 - Los datos introducidos se comprueban en busca de errores de entrada y el usuario tiene la oportunidad de corregirlos.
 - Hay un mecanismo disponible para revisar, confirmar y corregir la información antes de finalizar la presentación.
 - Para cada paso en un proceso de autenticación que se basa en una prueba de función cognitiva, hay al menos otro método de autenticación disponible que no se basa en ese tipo de prueba o hay un mecanismo disponible para ayudar al usuario a completar la prueba.
 - La información previamente ingresada por o proporcionada al usuario que se requiere que se ingrese nuevamente en el mismo proceso y en la misma sesión de usuario es rellenada automáticamente o está disponible para que el usuario la seleccione, excepto cuando:

- La información que deba ingresar de nuevo es esencial.
 - La información es necesaria para garantizar la seguridad del contenido.
 - La información anterior no es válida.
- Diseño. Facilitar la comprensión a través de la correcta colocación de los distintos elementos.
 - Cambiar la orientación de la pantalla. Se deberían soportar ambas orientaciones. Si no es posible admitir ambas orientaciones, se debe asegurar de que sea fácil para todos los usuarios cambiar la orientación para volver a un punto en el que se admite la orientación de su dispositivo.
 - Diseño consistente.
 - Colocación de elementos importantes de la página antes del desplazamiento de la página. Para no hacer a los usuarios desplazarse para llegar a ellos.
 - Agrupación de elementos operables que realizan la misma acción.
 - Proporcionar una indicación clara de que los elementos son procesables. Se debe distinguir claramente entre los elementos accionables (como botones o enlaces) y los que no lo son (como el texto).
 - Proporcionar instrucciones para pantallas táctiles personalizadas y gestos de manipulación de dispositivos. Incorporar instrucciones para explicar qué gestos se pueden usar y para qué se utilizan.

Robusto

- Configure el teclado virtual para el tipo de entrada de datos requerido. Ayuda a evitar errores y garantiza que los formatos sean correctos.
- Proporcionar métodos fáciles para la entrada de datos. Reduciendo la cantidad de entrada de texto y proporcionando otros elementos interactivos más sencillos.
- Utilizar las propiedades características de cada dispositivo.
- En el contenido implementado mediante lenguaje de marcado para aquellas aplicaciones que realizan el intercambio de información con otras aplicaciones:
 - Los elementos tienen etiquetas de inicio y finalización completas.

- Los elementos se anidan de acuerdo con sus especificaciones.
- Los elementos no contienen atributos duplicados y cualquier identificador es único, excepto donde las especificaciones permitan estas características.
- Los mensajes de estado se pueden determinar mediante programación a través del rol o las propiedades, de modo que se puedan presentar al usuario mediante tecnologías de asistencia sin recibir atención.
- Para todos los componentes de la interfaz de usuario:
 - El nombre y la función se pueden determinar mediante programación.
 - Los estados, las propiedades y los valores que puede establecer el usuario se pueden realizar mediante programación.
 - La notificación de cambios en estos elementos está disponible para los agentes de usuario, incluidas las tecnologías de asistencia.

2.2. Apps similares

En este apartado, vamos a comentar un poco sobre algunas de las aplicaciones que hemos encontrado que tengan una funcionalidad similar a la aplicación que hemos planteado al principio de este proyecto. Dichas aplicaciones son:

- *Qapital* [7]: Aplicación de pago que únicamente está disponible para móviles y tablets, y cuyo objetivo es crear retos financieros. El usuario decide la cantidad de dinero que quiere ahorrar, invertir o reservar y para qué la quiere (para viajar, para futuros estudios,... , todos objetivos ya prefijados por la aplicación) y la app sugiere cómo debe realizarlo y le comenta su progreso diario. Para ese control, debe insertar su cuenta bancaria en él. Además de los retos individuales, permite realizar la administración de una cuenta de pareja.

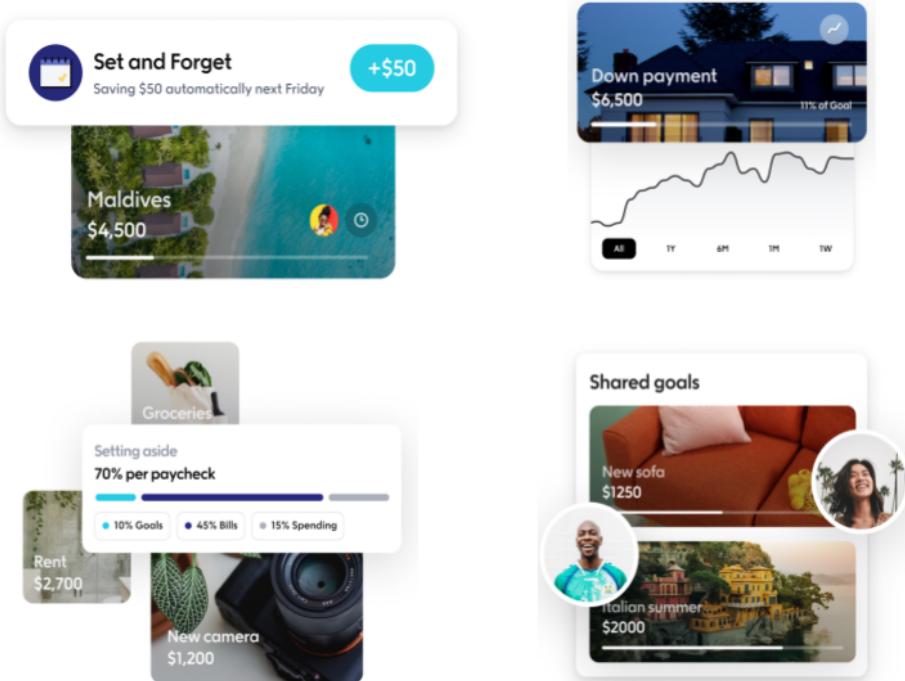


Figura 1: Capturas de pantalla de la aplicación Qapital [7].

Si prestamos atención a las capturas de pantalla de la Figura 1, podemos ver arriba a la izquierda cómo guarda (sin necesidad de realizar el propio usuario) una cantidad de dinero para una actividad; arriba a la derecha se muestra una gráfica donde se ve la cantidad de dinero que se ha guardado junto al porcentaje del objetivo conseguido; abajo a la izquierda, se muestra el presupuesto de lo que se va a gastar cada día (ahorrar para conseguir el objetivo, pagar facturas, ...), y abajo a la derecha, están los objetivos que se comparten con otros usuarios de la aplicación.

- *GoodReads* [8]: Es una página web que muestra los retos personales que hace un usuario sobre lecturas, como cuántos libros leer en una temporada o qué tipos de libros quiere comenzar a leer. En ella se puede ver el progreso de cada persona y animarles para que sigan con el reto, como podemos ver en la imagen derecha de la Figura 2. Los usuarios son los que deciden los retos que se proponen, además de los pasos que deben realizar dentro de estos retos, como proponerse cuántos libros va a leer y en cuanto tiempo lo va a realizar.

The screenshot shows the GoodReads homepage with the following sections:

- Groups**: A search bar for "Group name, description" and a "Search groups" button.
- Featured groups**:
 - Read With Jenna (Official)**: 22145 members - Active 2 days ago. Description: When anyone on the TODAY team is looking for a book recommendation, there is only one person to turn to: Jenna Bush Hager. Jenna will select a book and as you read along, we'll be posting updates ...
- Popular groups**:
 - Goodreads Librarians Group**: 200671 members - Active a minute ago. Description: A place where all Goodreads members can work together to improve the Goodreads book catalog. Non-librarians are welcome to join the group as well, to comment or request changes to book records. Fo...
 - What's the Name of That Book???**: 99650 members - Active 3 minutes ago. Description: Can't remember the title of a book you read? Come search our bookshelves. If you don't find it there, post a description on our UNSOLVED message board. 1. GENRE and PLOT DETAILS are mandatory in t...
 - Reese's Book Club x Hello Sunshine**: 126993 members - Active an hour ago. Description: Hey Y'all, We've been reading together for awhile and we don't know about you, but we're ready to hear
- COMMUNITY** (on the right):
 - Sally wants to read 28 books in 2020. 28 books. Like
 - Tris Plev wants to read 10 books in 2020. 10 books. Like
 - Helen Vaughan wants to read 80 books in 2020. 80 books. Like
 - Maria wants to read 25 books in 2020. 25 books

Figura 2: Capturas de pantalla de la página web GoodReads [8].

- **Challenges** [9]: Una aplicación que consiste en retarse a realizar ciertas actividades físicas (controlados por relojes inteligentes) y ciertos hábitos de bienestar (como comer saludable o tener pensamientos positivos), ya sea por parejas, con un grupo de gente, o individualmente. La persona puede proponer el reto y cada usuario irá alcanzando el objetivo según él quiera realizar, mostrando por pantalla el avance realizado en cada momento y los puntos conseguidos con esos avances, si es un reto colectivo. Esta aplicación es de pago (unos 5\$/mes), aunque, si eres el ganador de uno de los retos colectivos (el que más puntos consiga en el reto), el premio obtenido es una tarjeta de regalo con un cierto valor económico.

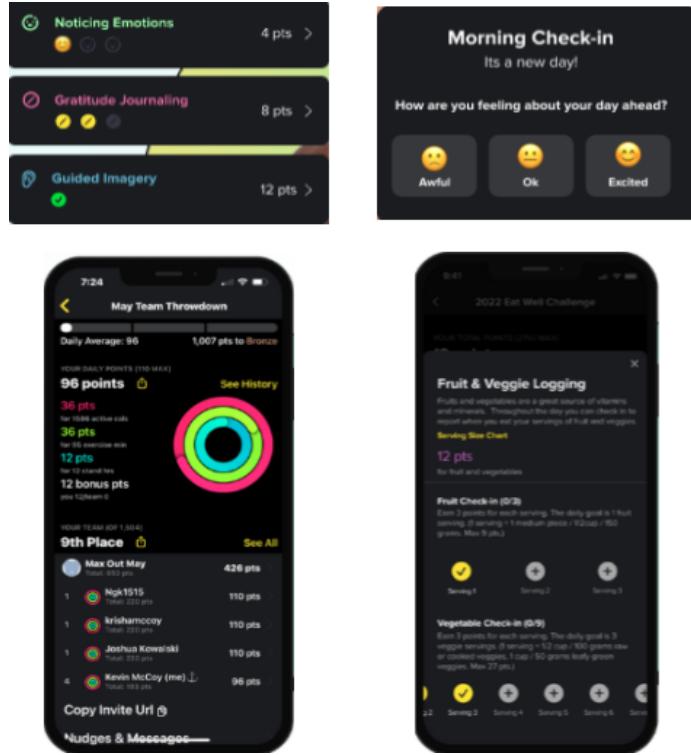


Figura 3: Capturas de pantalla de la app Challenges [9].

- *21 Days Challenge* [10]: Es otra aplicación móvil para crear retos sobre la alimentación, salud mental o ejercicio físico. En ella sólo hay retos individuales ya prefijados, que los amigos del usuario pueden apoyar mandando mensajes de ánimo, como la segunda imagen de la siguiente figura.



Figura 4: Capturas de pantalla de algunas funcionalidades de 21 Days Challenge [10].

- *Habitus* [11]: Es una aplicación, únicamente para móviles o tablets, que permite crear retos para cambiar cualquier hábito cotidiano, ya sea para verlo únicamente el usuario, o para que pueda verlo cualquier usuario de la app, que sea invitado del propietario del reto. En ella se puede ver y comparar el avance de cada uno de los participantes del reto. Además, en los retos se pueden especificar cómo van a ser las etapas que van a formar parte del reto y adjuntar pruebas de cada una de ellas para saber cómo las va superando.

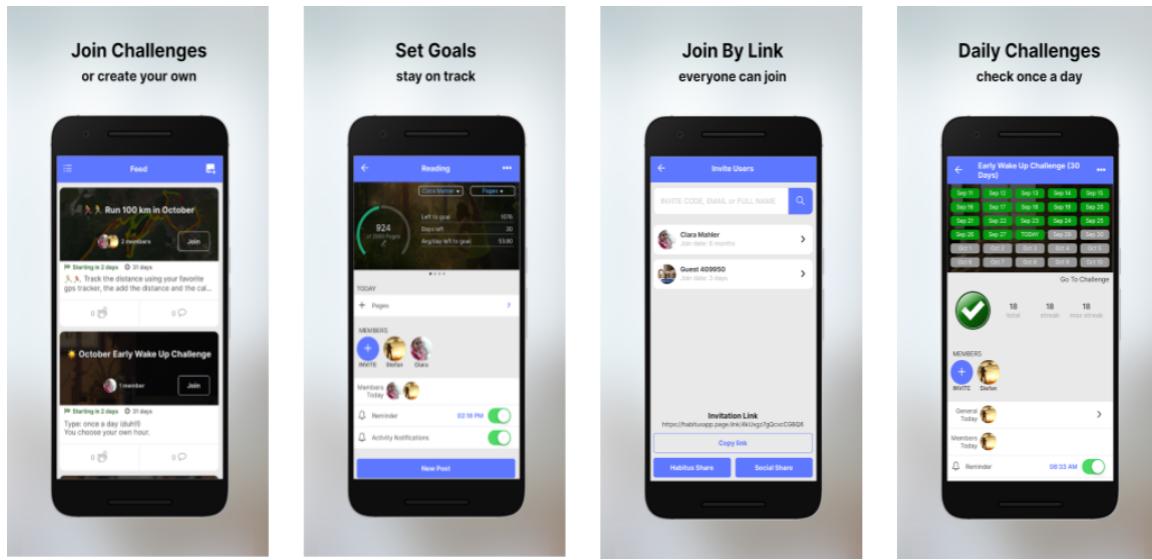


Figura 5: Capturas de pantalla de la app *Habitus* [11].

A continuación, en la siguiente tabla se muestra un pequeño resumen de las características que tiene cada una de las aplicaciones anotadas anteriormente:

Características	<i>Qapital</i> [7]	<i>GoodReads</i> [8]	<i>Challenges</i> [9]	<i>21 Days Challenge</i> [10]	<i>Habitus</i> [11]
Tipo de retos	Individuales y de parejas	Individuales	Individuales y colectivos	Individuales	Individuales y colectivos
Retos prefijados	No	No	No	Sí	No
Etapas prefijadas	Sí	No	No	Sí	No
Gamificación	No	No	Sí	No	No
Tipo de licencia	De pago	Gratis	De pago	Gratis y de pago	Gratis y de pago
Mensajes de	No	Sí	No	Sí	No

apoyo					
Progreso	Sí	Sí	Sí	No	Sí
Chat	No	No	No	No	Sí
Plataforma	Móvil y tablet	Todos	Móvil y tablet	Móvil y tablet	Móvil y tablet
Accesibilidad	Incompleta	Incompleta	Incompleta	Incompleta	Incompleta

Tabla 2: Resumen de las características de aplicaciones similares.

Investigadas las aplicaciones que tienen características similares al de nuestro proyecto, podemos decir que, de las que hemos visto, la que más se puede asemejar a la aplicación que nosotros tenemos en mente crear es la aplicación *Habitus* [11] puesto que puede generar un reto personificado, crear todas las etapas que necesite para conseguir el objetivo del reto, añadir personas al reto como participantes y añadir pruebas y ver el avance del reto tanto el del propio de usuario como del resto de participantes.

Sin embargo, no es apta para todos los dispositivos, puesto que únicamente puede ser instalado en un móvil o en una tablet, ni permite añadir usuarios al reto para que nos anime a conseguir nuestro objetivo, ni es totalmente accesible para cualquier usuario, según lo estudiado por las guías de accesibilidad anteriores (por ejemplo, no tiene un icono algunos de los botones de la aplicación que indiquen la acción que va realizar, sólo tiene una breve descripción, como podemos ver en la tercera imagen de la Figura 5).

2.3. Tecnologías útiles para el proyecto

En este apartado nos vamos a centrar en ver qué clases de tecnologías necesitaríamos para poder desarrollar el proyecto final y qué tecnologías podemos utilizar para llevarla a cabo.

Frameworks

En primer lugar, nos vamos a enfocar en ver cómo podríamos crear la base de la aplicación web. Para ello, lo que necesitamos es seleccionar el framework perfecto para generar dicha aplicación. Algunos de los frameworks encontrados en el mercado para usarlos en este tipo de aplicaciones son los siguientes:

- *Django* [12, 13, 14, 15]. Framework multiplataforma de alto nivel de código abierto y gratuito, escrito en Python (lenguaje de programación de alto nivel multiparadigma

utilizado para el desarrollo de aplicaciones de cualquier tipo y ámbito) [16], para desarrollar aplicaciones web con el patrón de diseño modelo-template-vista, a través de múltiples funciones por defecto.

Se encarga de ayudar a los desarrolladores a la hora de crear apps incluyendo diversos paquetes y middleware para su desarrollo (como su propio motor de plantillas, el manejo directamente con la base de datos a través de ORM [17] sin necesidad de escribir las consultas mediante el lenguaje del tipo de base de datos utilizada, un panel de administración o el manejo de usuarios), además de diversos tipos de documentaciones de la gran mayoría de sus funcionalidades.

Con respecto a crear aplicaciones seguras, Django cuenta con múltiples características de seguridad, como CSRF y XSS, y actualizaciones automáticas de sus herramientas.

En relación con el aprendizaje, si es la primera vez que se trabaja con este framework, cuesta bastante estar familiarizado con su forma de trabajar pero, una vez acostumbrado, se puede automatizar gran parte de sus procesos.

Otros problemas que podríamos comentar de Django es que no es tan flexible a la hora de incorporar herramientas de terceros y que, al tener tantas funcionalidades y paquetes, pesa demasiado.

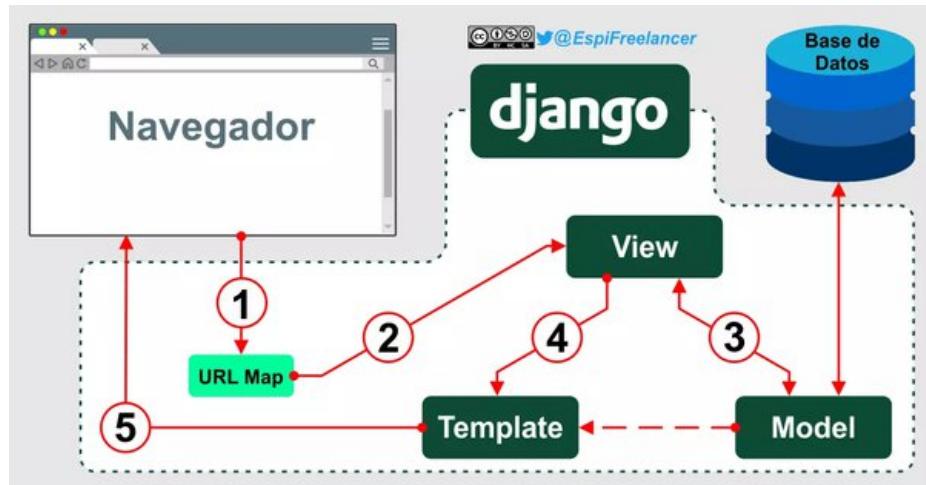


Figura 6: Estructura del proyecto de Django.

- *Flask* [13, 18, 19]. Framework minimalista, también escrito en Python [16], que desarrolla páginas webs con la cantidad de código mínima mediante el patrón de modelo-vista-controlador.

Flask depende del kit de herramientas Werkzeug WSGI (especificación para saber cómo se comunica el servidor web con la aplicación) [20], el motor de plantillas Jinja [21] (similar a la que ofrece Django) y el kit de herramientas Click CLI. Aunque tiene un conjunto de funcionalidades reducido.

Al utilizar mayormente aplicaciones de terceros, no se puede decir que Flask sea seguro ni garantizar que se puedan actualizar automáticamente sus características.

En principio, Flask es muy ligero en comparación con otros frameworks, pero, conforme se van añadiendo librerías y paquetes para añadir funcionalidades que no tiene, va aumentando considerablemente su tamaño.

Por último, en cuanto a la documentación sobre este framework, hay diversos canales y foros que ayudan a utilizar correctamente esta tecnología.

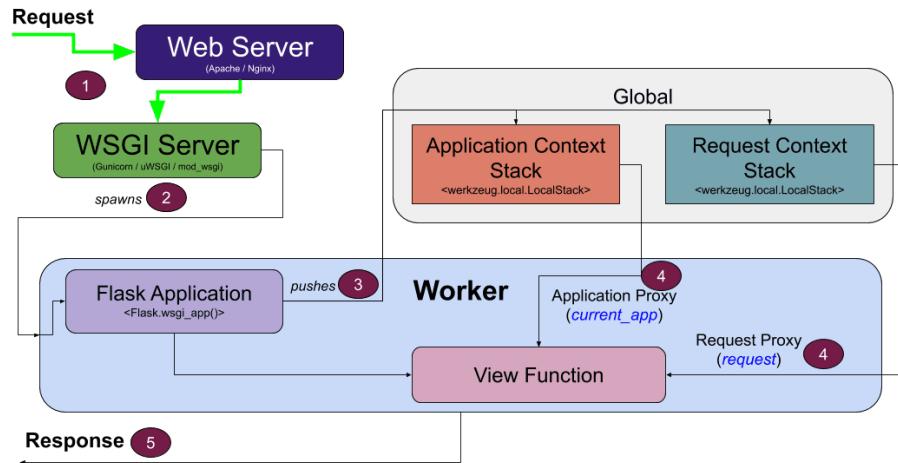


Figura 7: Estructura de un proyecto de Flask.

- ExpressJS [15, 22, 23, 24, 25]. Framework web escrito en JavaScript (es un lenguaje de programación ligero e interpretado con soporte para programación orientada a objetos, imperativa y declarativa) [26] y alojado dentro del entorno de ejecución de código abierto NodeJS [27] que sigue la estructura del modelo-vista-controlador.

Proporciona varias funciones que hacen que el desarrollo de aplicaciones web sea rápido y fácil. Además, permite definir rutas de su aplicación basadas en métodos HTTP y URL a través de middleware.

En cuanto a la curva de aprendizaje de este framework, es bastante sencillo, fácil y rápido de implementar y de configurarlo. Sin embargo, es difícil de comprender el código puesto que no hay una forma de organizar la estructura del proyecto.

Es fácil de servir archivos estáticos y recursos de su aplicación y conectar con bases de datos, como MongoDB o MySQL.

Únicamente es necesario un solo lenguaje de programación para crear el backend y el frontend de la aplicación al ser JavaScript también un lenguaje de programación creado para desarrollar scripts dentro de HTML.

Para mostrar los datos recibidos del servidor, necesita utilizar plantillas de terceros para generar la página correspondiente.

ExpressJS ofrece un mecanismo de enrutamiento progresivo que ayuda a mantener la condición de la página web con la ayuda de sus URL.

Por último, simplifica la depuración al proporcionar una herramienta de depuración que puede identificar la parte precisa de la página web que tiene errores o fallos.

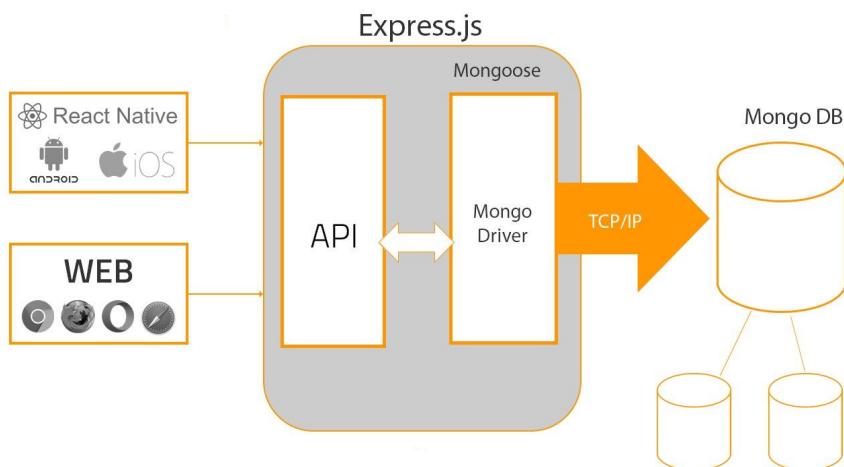


Figura 8: Ejemplo de la estructura de un proyecto realizado con ExpressJS.

Características	Django [12]	Flask [18]	ExpressJS [22]
Arquitectura	MTV	MVC	MVC
Lenguaje	Python	Python	JavaScript
Curva de aprendizaje	Alta	Baja	Baja
Seguridad	Alta	Baja	Baja
Funcionalidades de terceros	No	Sí	Sí
Flexibilidad	Baja	Alta	Alta

Peso	Grande	Ligero	Ligero
Motor de plantillas integrado	Sí	No	No
ORM	Sí	No	No

Tabla 3: Resumen de las características de los frameworks vistos.

Con respecto al framework seleccionado para desarrollar la red social, preferimos utilizar *Django* frente al resto por las siguientes razones:

Como ya contamos con experiencia utilizando este framework, la curva de aprendizaje es muy pequeña por lo que nos podemos centrar en especificar el aprendizaje a los problemas que queramos resolver en nuestro proyecto y generar una aplicación de la forma más compleja posible.

Además, no tenemos que pensar cómo integrar herramientas de terceros para poder crear una base de datos y conectarla al proyecto puesto que lo tiene integrado. También dispone de la herramienta ORM (que podemos ver reflejada en la *Tabla 3*) que, con únicamente decir dentro del proyecto cómo van a ser las tablas, las consultas y las modificaciones que queremos realizar sobre ella en Python, es más que suficiente.

Tampoco es necesario utilizar librerías de terceros para generar las plantillas de HTML con la que irán los datos recogidos del servidor.

Por último, podemos crear distintos formularios con una seguridad garantizada puesto que dispone de herramientas encargadas de que no ocurra situaciones de riesgo en nuestro sistema.

Bases de datos

Una vez comentados los frameworks que pueden ser candidatos para la creación de la aplicación que proponemos en este proyecto, nos centraremos en elegir qué tipo de base de datos vendría mejor para almacenar y manejar los datos con los que se trabajará.

Antes, vamos a explicar por qué elegimos, dentro de las clases de base de datos, las relacionales y son por las siguientes razones:

- Para poder realizar consultas más complejas (por ejemplo, de tipo JOIN).
- Para poder crear claves extranjeras (foreign keys) entre distintas tablas y, así, poder realizar el borrado de datos en cascada fácilmente.

A continuación, hemos investigado las características y las funcionalidades de los siguientes sistemas gestores de base de datos:

- *PostgreSQL* [28, 29, 30]. Sistema Gestor de Base de Datos en SQL robusto y potente que lo podemos encontrar en múltiples sistemas operativos. Además, su instalación es gratuita y sencilla de realizar.

Es capaz de ajustarse al número de CPU y a la cantidad de memoria disponible de forma óptima, así podemos realizar varias peticiones a la vez sin problema.

También se pueden realizar consultas de lectura sobre ella cuando la base de datos está en backup, por lo que el sistema no se paralizará totalmente.

Es relativamente lento en inserciones y actualizaciones en bases de datos pequeñas, PostgreSQL está diseñado para ambientes de alto volumen.

PostgreSQL es compatible con muchos servidores web, como Apache, Nginx o LiteSpeed.

Es capaz de trabajar con diversos lenguajes de programación para insertar nuevas funciones como PHP, Python, Perl, Ruby o Java, entre otros.

En cuanto al soporte técnico, no tiene teléfonos ni soportes en línea, sino foros en los que resuelven dudas o problemas.

- *MySQL* [31, 32, 33, 34, 35]. Sistema de Administración y Gestión de las Bases de Datos multiplataforma de open source y para desarrollos de aplicaciones de desarrollo web. Posee una versión comercial (que la gestiona la compañía Oracle) y otra parte que es un código abierto.

MySQL es veloz al realizar operaciones, y garantiza un buen rendimiento de las aplicaciones. Además, es fácil de instalar y configurar.

No maneja de manera tan eficiente una base de datos con un tamaño muy grande, está pensada en utilizar bases de datos de tamaño normal a pequeñas.

En cuanto al rendimiento y a la estabilidad, MySQL suele responder rápidamente a las consultas gracias a la estructura de datos.

Por último, MySQL se actualiza constantemente y mantiene cada una de sus características bien documentadas, además de ofrecer soporte de personas técnicas especializadas en esta clase de base de datos.

- *MariaDB* [35, 36, 37]. Sistema Gestor de Base de Datos relacionales en SQL con algunas de las estructuras y de las características de MySQL, y algunas de ellas incluso mejoradas.

En cuanto a los datos de las tablas, MariaDB soporta el enmascaramiento de los datos (protege la información sensible) y las columnas dinámicas.

No soporta el tipo de datos JSON nativo y la autenticación SHA-2, como otros gestores como MySQL sí lo implementan.

Por último, tiene un soporte limitado, por lo que, si ocurre algún problema, es difícil de contactar para recibir ayuda.

Características	PostgreSQL [28]	MySQL [30]	<i>MariaDB</i> [36]
Fácil de instalar	Sí	Sí	Sí
Gratis	Sí	Algunas características	Sí
Multiplataforma	Sí	Sí	Sí
Escalabilidad	Sí	Sí	Sí
Estabilidad	Sí	Sí	Sí
Potencia y robustez	Sí	Sí	Sí
Extensibilidad	Sí	No	No
Tipo de base de datos	Bases de datos enormes	Bases de datos normales	Bases de datos normales
Uso de servidores web	Sí	Sí	Sí
Lenguajes de programación soportados	PHP, Python, Perl, Ruby y Java	PHP, Perl y Python	PHP, Perl y Python
Soporte	Foros	Técnicos especialistas	Foros

Tabla 4: Resumen de las características de las bases de datos seleccionadas.

Por lo tanto, de las conclusiones obtenidas de la tabla resumen, el sistema gestor de bases de datos que hemos seleccionado es *MySQL* puesto que son sencillas de instalar, se utilizan para bases de datos de aplicaciones web como la nuestra y ofrece soportes de técnicos expertos. Además, es una de las bases de datos que soporta *Django*.

Alojamientos webs

En cuanto a los distintos alojamientos que podemos encontrar en el mercado y que sean aptos para la aplicación de red social que vamos a desarrollar, hemos hallado los que vamos a comentar a continuación:

- *PythonAnyWhere* [39, 40]. Host que está orientado para proyectos en Python y que funciona con Amazon Web Services [41].

Los planes se adaptan a todos los niveles de usuarios de Python, desde los principiantes absolutos hasta los gurús de las aplicaciones. Incluso ofrecen una licencia gratuita para saber cómo es el entorno.

Usar PythonAnywhere para lanzar su proyecto es simple y no muy diferente a lanzarlo en local.

Además, es sencillo instalar la aplicación puesto que únicamente es necesario saber dónde se va a alojar dentro del almacenamiento del host los ficheros del programa y utilizar las distintas herramientas que tiene automatizadas para realizar las distintas configuraciones para lanzar y crear el proyecto.

Por último, tener la aplicación alojada en este host cuesta únicamente 5 dólares al mes.

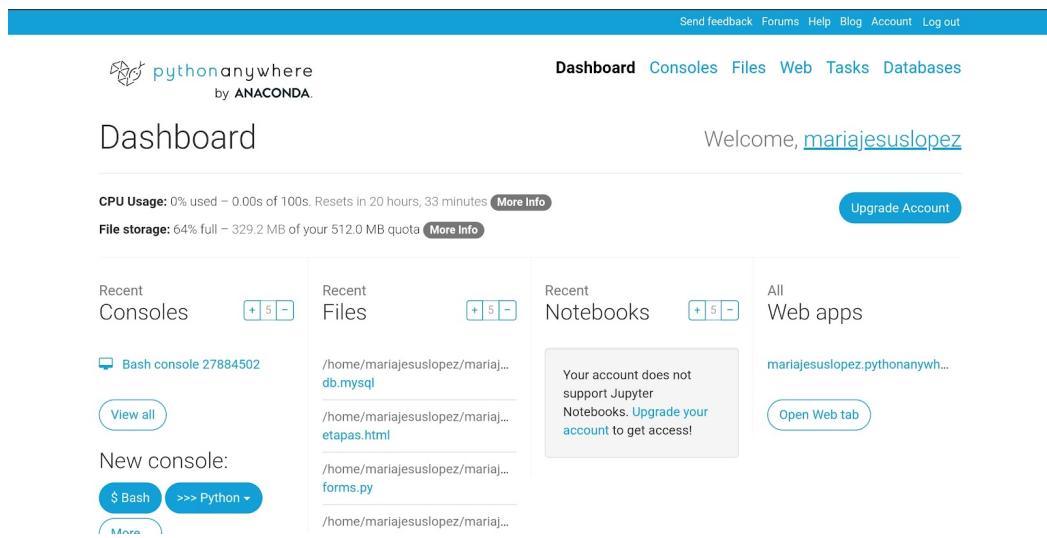


Figura 9: Captura de pantalla con las funcionalidades de una cuenta gratis de PythonAnyWhere.

- A2 Hosting [39, 42]. Servidor virtual privado no administrado, famoso por ser amigable para los desarrolladores, en el que se puede crear cualquier tipo de servicio web.

Únicamente es necesario previamente configurar las distintas herramientas que necesita la aplicación, como la instalación del lenguaje de programación a utilizar, y el entorno virtual.

Después de eso, solamente es necesario configurar el proyecto (el framework y las distintas herramientas que se utilizarán dentro de la aplicación) de la forma en la que se desee.

En cuanto al precio por tener nuestra aplicación almacenada en este servidor, también es de 5 dólares al mes.

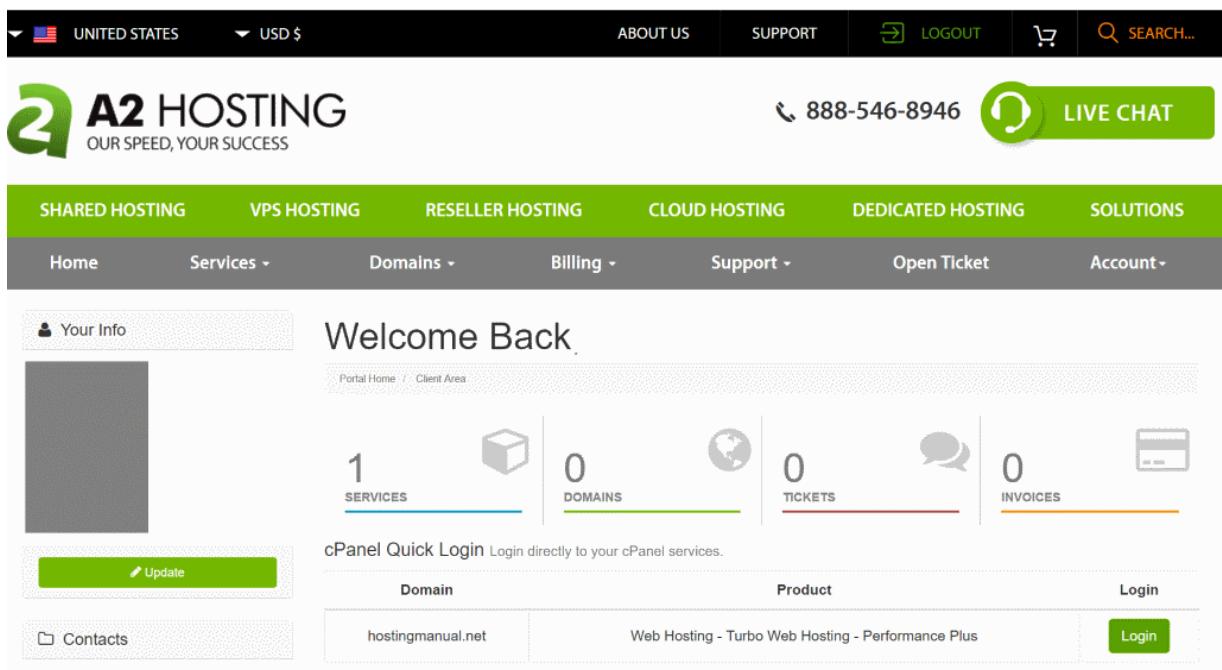


Figura 10: Captura de pantalla del alojamiento A2 Hosting.

- Digital Ocean [39, 43]. Servidor virtual creado para todo tipo de desarrolladores, sin importar el fin por el que se solicita dicho servidor.

En él, únicamente se paga el costo de entrada más las tecnologías en la nube que se necesitan para desplegar la aplicación, por lo que únicamente se paga por aquello que se utiliza.

Digital Ocean únicamente requiere que se configure el servidor tanto a nivel de hardware como de software, por lo que es altamente configurable. Por lo tanto, para

aquellas personas principiantes o que en un principio no sepan lo que va a consumir su aplicación, es necesario dedicar más tiempo y esfuerzo en crear el entorno.

The screenshot shows the DigitalOcean interface for managing teams. On the left, a sidebar lists various services: Projects, Manage, Apps, Droplets, Kubernetes, Volumes, Databases, Spaces, Container Registry, Images, Networking, and Monitoring. The 'Droplets' option is highlighted. The main area is titled 'example-team'. It includes a search bar at the top right. Below the title, there are sections for 'Details' (with email sammy@digitalocean.com and droplet limit 10), 'Address' (with placeholder 'Add your address'), and 'Phone' (with placeholder 'Add phone number'). A button 'Edit Team Profile' is located in the top right of this section. Below this is a table titled '3 Members' with columns for Name, Role, 2 FA, and Status. The table lists three members:

Name	Role	2 FA	Status	Action
sammy (you) sammy@digitalocean.c...	Owner	On	Joined	Actions
billier@example.com	Biller	On	Joined	Actions
member@example.com	Member	—	Pending	Actions

Figura 11: Captura de pantalla dentro de una cuenta en Digital Ocean.

- *Amazon Web Services (AWS)* [41, 44]. Proveedor de servidores virtuales creado por la empresa de *Amazon* en la que cualquier desarrollador puede implementar en ellos cualquier tipo de proyecto (aplicaciones web, análisis de datos, aprendizaje automático, ...) de cualquier sector (publicidad, servicios financieros, videojuegos, ...).

El desarrollador puede elegir los componentes y las tecnologías que desea que tenga su servidor. Por ello, dependiendo de lo que el usuario seleccione, costará más o menos tener disponible ese servidor.

Sin embargo, la desventaja de crear desde 0 el servidor es que el desarrollador debe ser el que instale en él todas las herramientas necesarias para el desarrollo de su proyecto.

Además, cuenta con una gran cantidad de servidores físicos creados en múltiples países (algunos de ellos incluso en España) que el usuario puede elegir dónde alojar su proyecto.

Por último, ofrece un servicio gratuito, con algunas restricciones con respecto a las tecnologías y a las herramientas que se desea instalar en el servidor, durante un plazo de 12 meses.

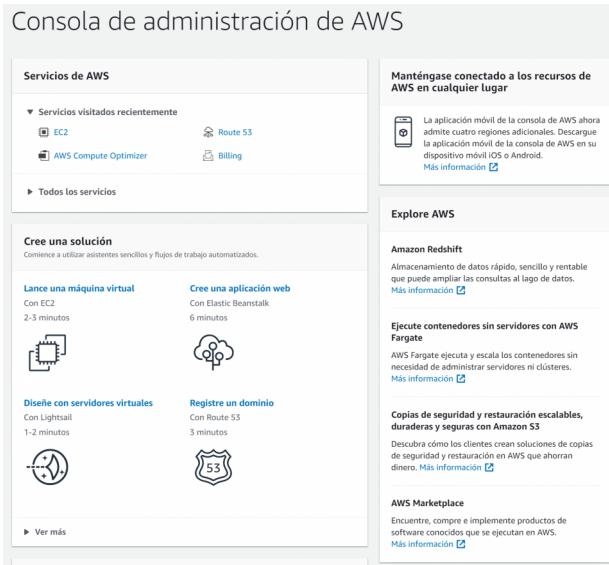


Figura 12: Captura de pantalla de ejemplo de servicios ofrecidos por AWS.

- *Google Cloud* [45, 46]. Proveedor de recursos de computación en la nube de la empresa de *Google* que sirve para desarrollar, implementar y operar distintos tipos de aplicaciones (páginas web, análisis de datos, almacenamiento de información, aprendizaje automático, ...) a través de la web.

Antes de crear el servidor, puede indicar las características que puede tener, como su sistema operativo, el tamaño del disco duro o el tipo de tarjeta gráfica. Y todo esto, pagando únicamente por lo que utilice.

Sólo debe instalar, una vez creado el servidor virtual, las herramientas que necesita para desplegar el proyecto en él.

A los nuevos clientes de Google Cloud, ofrece un pequeño crédito de 300 USD para que prueben las distintas tecnologías que ofrece.

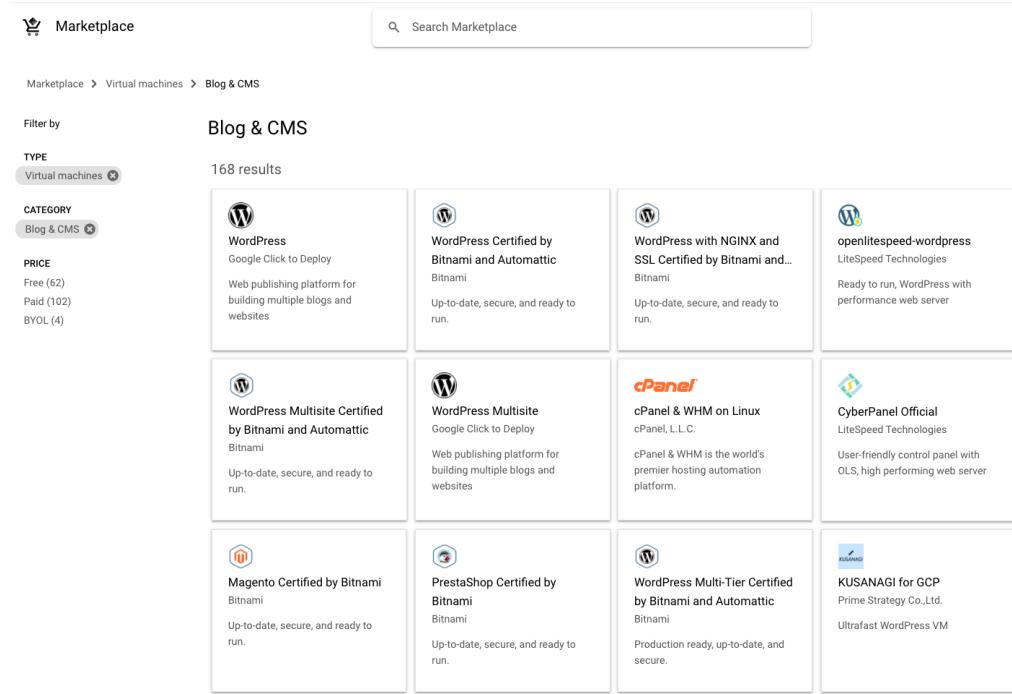


Figura 13: Ejemplo de blogs soportados dentro del servidor Google Cloud.

- *Microsoft Azure* [47, 48]. Proveedor de servicios en la nube creado por la empresa *Microsoft* en la que permite que cualquier clase de desarrollador pueda implementar su proyecto (ya sea tipo web, análisis de datos o de inteligencia artificial) en un servidor web totalmente personalizado.

El desarrollador puede indicar a este proveedor el tipo de servicio que necesita, además de las distintas herramientas y tecnologías que quiere que estén disponibles dentro del servidor.

Únicamente paga por aquellos servicios en donde supere la cantidad mensual gratuita o aquellos servicios que solamente se disponen si se pagan una cierta cantidad.

Además, *Microsoft Azure* ofrece un plan en el que te dan 200 USD para utilizarlo durante 30 días, cuando empiece el usuario a utilizar este proveedor, y múltiples servicios gratuitos para siempre.

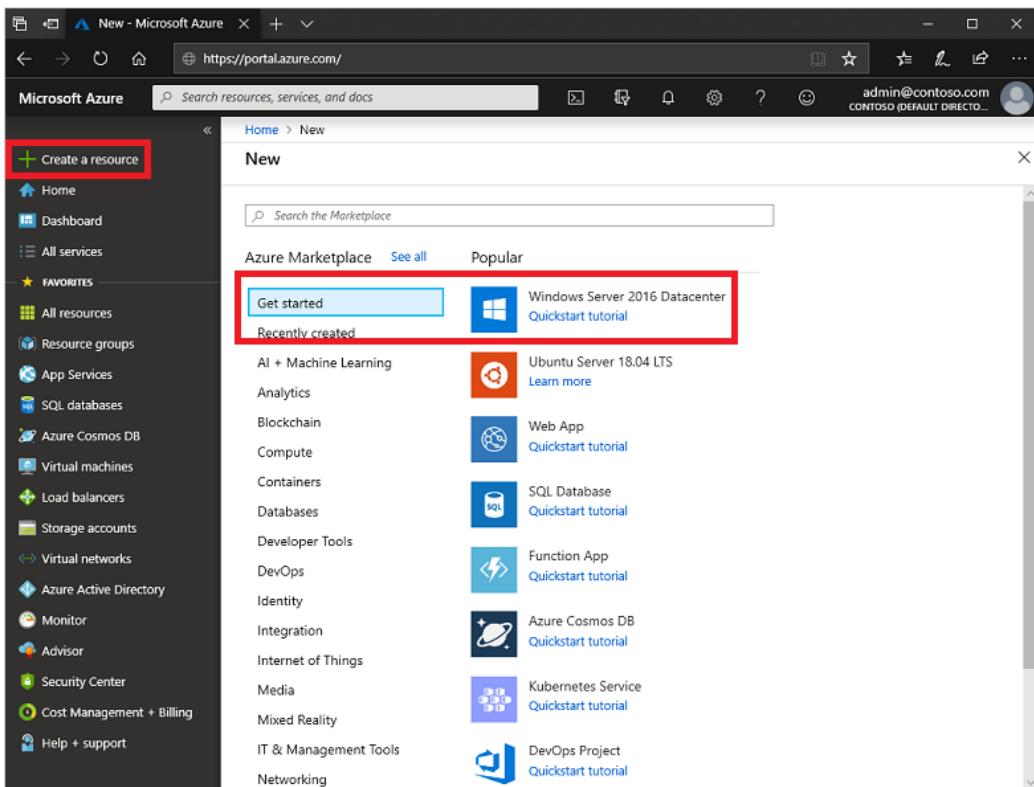


Figura 14: Ejemplo de creación de un servidor con el sistema operativo Windows en Microsoft Azure.

Características	PythonAnyWhere [40]	A2 Hosting [42]	Digital Ocean [43]	AWS [41]	Google Cloud [45]	Microsoft Azure [47]
Tipo de servidor	Python	Cualquiera	Cualquiera	Cualquiera	Cualquiera	Cualquiera
Parte gratuita	Sí	No	No	Sí (durante 12 meses)	Sí (300 USD)	Sí (además de 200 USD)
Fácil instalación	Sí	Sí	Sí	Sí	Sí	Sí
Fácil configuración	Sí	Sí	No	No	No	No
Precio	5\$/mes	5\$/mes	Depende configuración	Depende configuración	Depende configuración	Depende configuración

Tabla 5: Resumen de las características de los alojamientos buscados.

De los alojamientos vistos (y resumidas sus características en la Tabla 5), el elegido para desplegar nuestra aplicación en cualquier dispositivo es *PythonAnywhere* al ser un

alojamiento exclusivamente de *Python* (ya que nuestro proyecto de *Django* está escrito en ese lenguaje de programación) y no es necesario realizar configuraciones del entorno para poder crear el proyecto con el framework seleccionado, además de tener la posibilidad de poder crearlo gratuitamente, con ciertas limitaciones, y así poder tener la posibilidad de lanzar la demo de esta red social.

En ella, además de estar almacenado el código de la aplicación y de lanzar el proyecto a la web (en nuestro caso se encontrará en mariajesuslopez.pythonanywhere.com), podremos crear la base de datos MySQL, donde se almacenará y se consultará la información necesaria para la app.

Marcos frontend

En este apartado nos centraremos en las bibliotecas que nos ayudarán a diseñar la parte visual de nuestra aplicación. Para ello estuvimos investigando qué bibliotecas hay en el mercado que puedan cumplir con nuestras necesidades. Las encontradas fueron las siguientes:

- *Bootstrap* [49]. Marco frontend de código abierto para la creación de sitios web con Responsive Design.

Simplifica el desarrollo de páginas web debido a que tienen ya componentes prediseñados fáciles de usar dentro de cualquier aplicación y totalmente personalizables. Además, tiene incorporado un sistema de cuadrícula que alinea fácilmente los elementos.

Para utilizar esos componentes y crear ciertas animaciones sobre ellos, tiene ya añadidos dichas funcionalidades a través de complementos JQuery y de funciones en Javascript.

Por último, este marco es aplicable e integrable con otras tecnologías y es compatible con cualquier navegador.

Figura 12: Sistema de cuadrícula de Bootstrap.

- *Skeleton* [50]. Framework CSS simple y ligero de naturaleza receptiva y que adopta un enfoque de tipo móvil primero, pero que se puede utilizar con cualquier tamaño de pantalla.

Su sistema está compuesto de una cuadrícula fluida de 12 columnas que consta de filas y columnas para la distribución de elementos de una página web. Además, su sintaxis es fácil de implementar en un proyecto.

Es útil si se tiene pensado en realizar una aplicación pequeña, en la que no sean necesarios muchos elementos visuales, puesto que ofrece la creación y funcionalidad de los componentes más básicos.

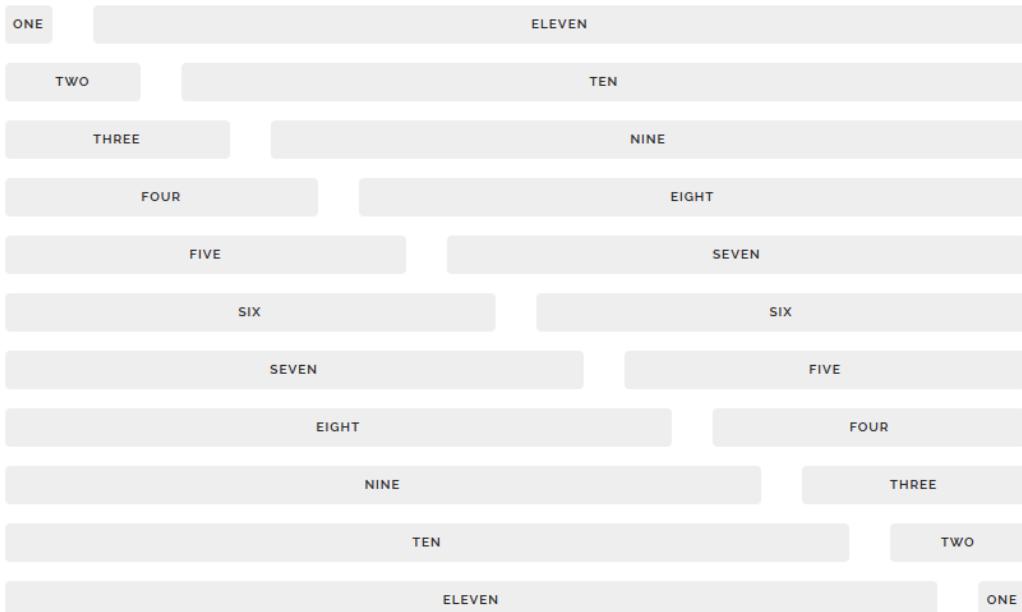


Figura 13: Sistema de cuadrícula de Skeleton.

- *Pure.css* [51]. Marco CSS que está agrupado en módulos CSS. Es ligero puesto que está pensado para ser utilizado en dispositivos móviles.

Es apropiado para ser utilizado dentro de proyectos relativamente pequeños. Además, está pensado para crear elementos de distintos tamaños cumpliendo con la normativa Responsive Design.

Por último, Pure.css tiene ya creados ciertos componentes para ser utilizados en cualquier aplicación, de los cuales se pueden customizar.

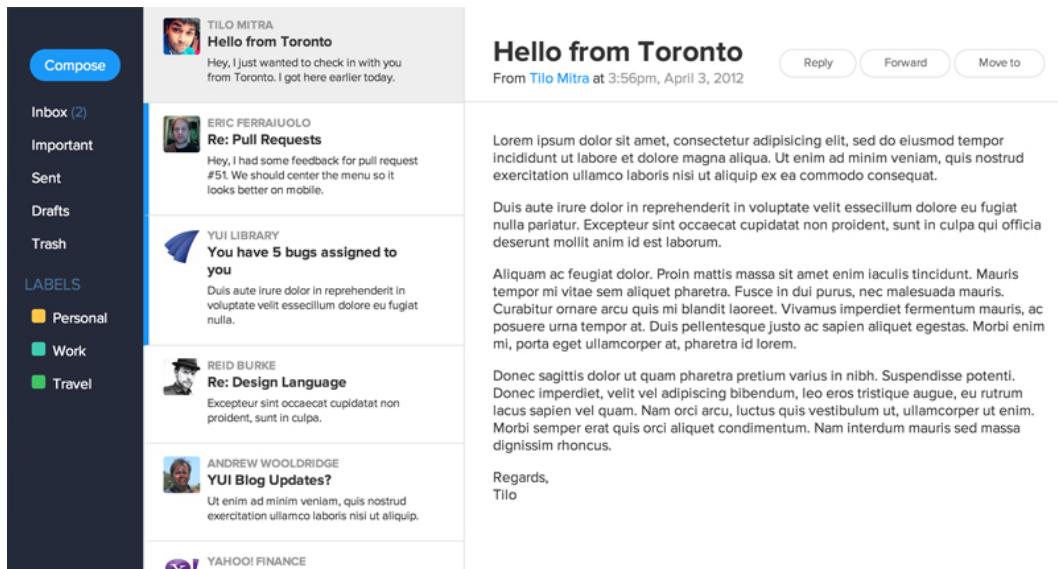


Figura 14: Ejemplo de estructura de una aplicación con Pure.css

Características	Bootstrap [49]	Skeleton [50]	Pure.css [51]
Responsive Design	Sí	Sí	Sí
Sistema de cuadrícula	Sí	Sí	No
Cantidad de componentes	Amplia	Pequeña	Pequeña
Animación	Sí	No	No

Tabla 6: Resumen de las características de los marcos de frontend hallados en el mercado.

Con respecto al marco de frontend, pensamos utilizar, dentro de los explicados en la tabla resumen anterior, la biblioteca *Bootstrap* puesto que, además de ser Responsive Design y de tener un sistema de cuadrícula para organizar y cambiar de tamaño los elementos de las páginas de esta aplicación, tiene una gran cantidad de componentes para añadir al proyecto que se pueden personalizar y añadir distintos elementos de animación sobre dichos componentes. También, este marco se puede integrar a la perfección dentro de un proyecto de *Django*.

Herramientas de HTML

Para poder realizar frontends dinámicos con la menor cantidad de código posible, es necesario tener en cuenta herramientas dentro de HTML que nos ayuden a implementar dichas funcionalidades. Esas herramientas son las dos que vamos a comentar a continuación:

- HTMX [52]. Biblioteca que permite acceder a funciones del navegador directamente desde HTML, en lugar de utilizar JavaScript.

Extiende y generaliza la idea central de HTML como hipertexto, abriendo más funcionalidades directamente dentro del lenguaje:

- Cualquier elemento puede emitir una solicitud HTTP.
- Cualquier evento puede desencadenar solicitudes.
- Se puede utilizar cualquier verbo HTTP, no solo GET y POST.
- Se puede actualizar una parte de la ventana mediante una solicitud, sin necesidad de cargar la ventana completa.

```
<script src="https://unpkg.com/htmx.org@1.9.2"></script>
<!-- have a button POST a click via AJAX --&gt;
&lt;button hx-post="/clicked" hx-swap="outerHTML"&gt;
    Click Me
&lt;/button&gt;</pre>

```

Figura 15: Ejemplo de una petición POST a través de HTMX.

- Alpine.js [53]. Herramienta mínima y robusta para crear ciertas acciones directamente en el marcado de HTML. Esta herramienta funciona como una alternativa a JQuery en la creación de páginas más modernas.

Dentro del HTML, gracias a esta herramienta, podemos realizar ciertas funcionalidades para hacer que una página sea lo más dinámica posible, como definir valores a un componente o iniciar ciertas acciones, a través del código insertado en JavaScript.

```
<html>
<head>
    <script defer src="https://cdn.jsdelivr.net/npm/alpinejs@3.x.x/dist/cdn.min.js">
</head>
<body>
    <h1 x-data="{ message: 'I ❤️ Alpine' }" x-text="message"></h1>
</body>
</html>
```

Figura 16: Ejemplo de un código utilizando la herramienta Alpine.js.

Características	HTMX [52]	Alpine.js [53]
Funciona a través de atributos del HTML	Sí	Sí
Funciona para cualquier componente HTML	Sí	Sí
Maneja eventos	Sí	No
Emite solicitudes HTTP	Sí	No

Tabla 7: Resumen de las características de las herramientas de HTML.

De las herramientas vistas en la tabla resumen *Tabla 7*, es HTMX por las siguientes razones: la primera es que podemos realizar peticiones HTTP para enviar o recibir información del servidor, la segunda, que podemos cambiar ciertos elementos de la página, sin necesidad de recargar la página entera, y la última es que podemos realizar las peticiones con cualquier evento, no únicamente a través de clicks o respuestas de formularios gracias a que se puede utilizar dentro de un atributo de cualquier componente HTML.

Iconos

Lo último que necesitamos para ponernos en marcha con el desarrollo de la aplicación de red social de retos para una vida saludable es encontrar una biblioteca o una página que nos ayude a insertar iconos dentro de la parte visual de la aplicación para describir a través de un símbolo la funcionalidad, el dato a insertar o la información mostrada y así lograr que la página visualizada sea lo más accesible posible para cualquier usuario. Las páginas con iconos que hemos encontrado son:

- *Font Awesome* [54]. Tiene un conjunto de iconos y de herramientas, divididos en diferentes categorías y siendo algunos de pago y otros no, que se insertan en nuestra página web a través de elementos HTML y links de CSS para mostrarlos por pantalla.

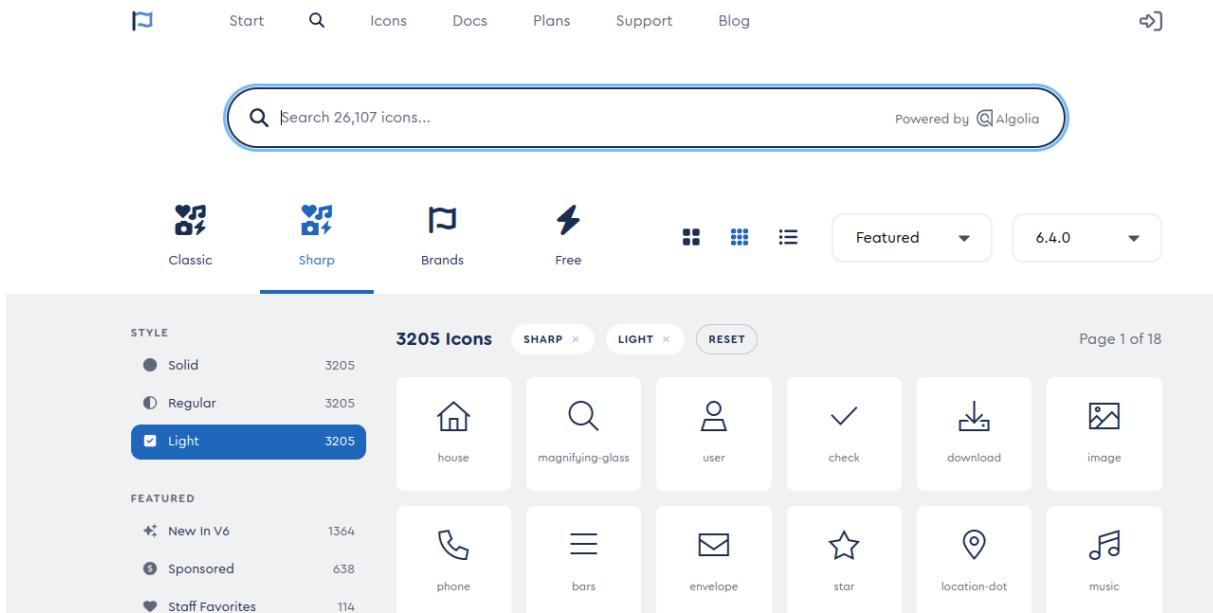


Figura 17: Captura de pantalla de la página oficial de Font Awesome.

- *Line Icons* [55]. Paquete de iconos diseñados por profesionales en todos los formatos vectoriales, que son fáciles de usar en la web ya que se realiza a través de la inserción de un componente de HTML. Ofrece una gran variedad de iconos, separados por categorías, y por si son gratis o de pago.

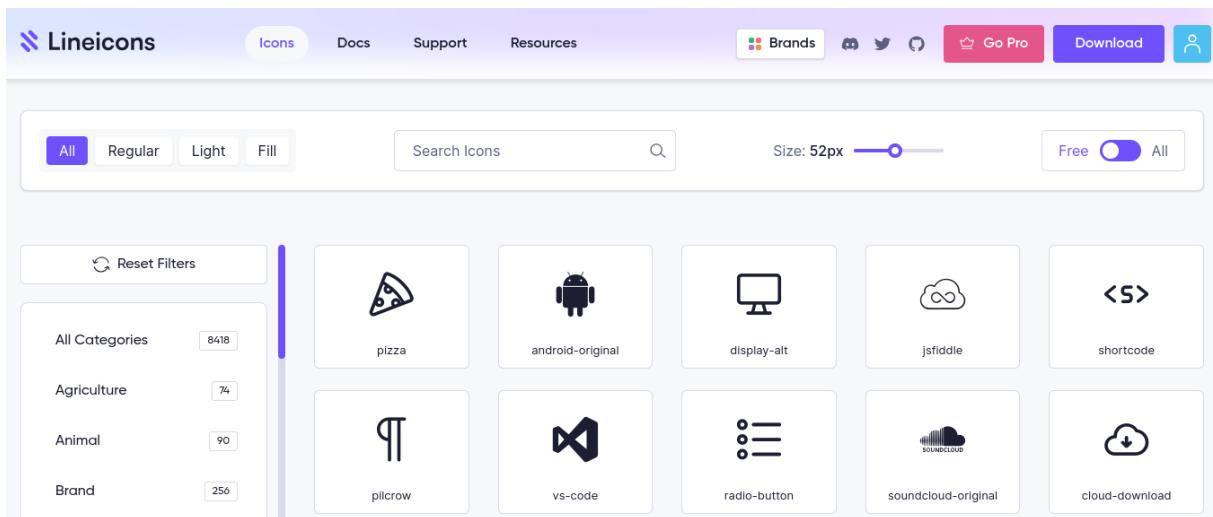


Figura 18: Buscador de iconos oficial de Line Icons.

- *Flat Icon* [56]. Página con un gran conjunto de iconos vectoriales para insertar en los proyectos webs gratuitos. Están disponibles en formatos SVG, PNG, PSD, EPS y BASE64. Además, se puede personalizar un ícono dentro de esta página.

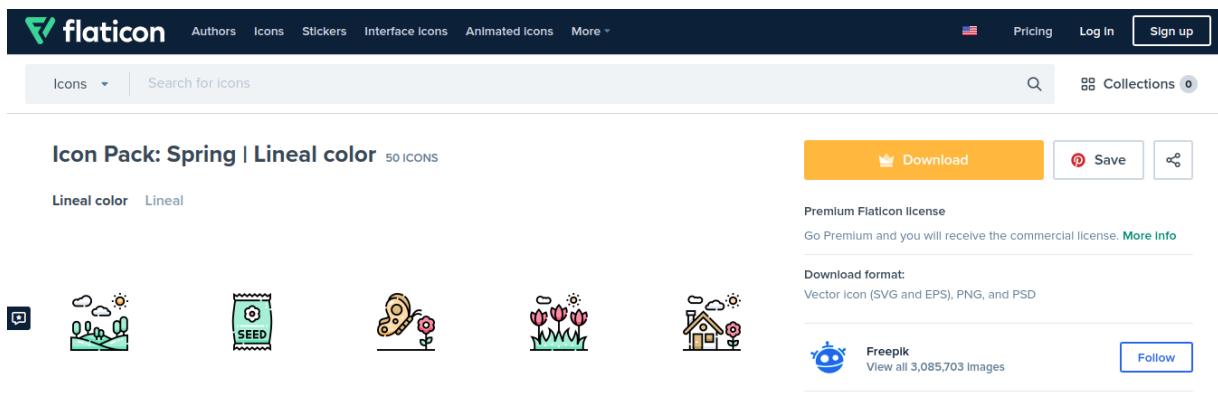


Figura 19: Ejemplo de búsqueda de un ícono en Flat Icon.

Características	Font Awesome [55]	Line Icons [56]	Flat Icon [57]
Modo de insertar en el proyecto	A través de HTML	A través de HTML	A través de una imagen
Divididos por categorías	Sí	Sí	Sí
De pago	Algunos	Algunos	Gratis

Tabla 8: Resumen de las características de las bibliotecas de íconos encontradas.

Con respecto a la biblioteca de íconos utilizada para guiar al usuario (para indicar la información visualizada, la acción que realiza un botón o el dato que debe escribir o dictar), hemos pensado que la que es más apropiada para este proyecto es *Font Awesome* puesto que ofrece los íconos que nosotros necesitamos en nuestra aplicación y porque es muy fácil de incorporarlo al frontend de la red social al realizarse a través de la creación de elementos HTML.

CAPÍTULO 3. Desarrollo

3.1. Descripción de la aplicación a crear

Una vez investigadas las guías de accesibilidad, las aplicaciones similares que podemos encontrar en el mercado y las tecnologías que podemos utilizar en nuestro proyecto, es momento de describir la aplicación que deseamos generar con aquellos requisitos que logren satisfacer los motivos por los que decidimos emprender este proyecto. Esos requisitos, que algunos de ellos coinciden con o se han tomado de las aplicaciones similares vistas, son los comentados a continuación:

- Poder utilizar la aplicación en cualquier dispositivo, sin importar tamaño de pantalla o sistema operativo.
- Sea accesible y usable para cualquier usuario (tenga o no discapacidad sensorial y cognitiva o intelectual): pueda utilizar las funcionalidades de la aplicación sin complicaciones y entienda lo que está realizando o viendo.
- Poder registrarse en la aplicación para tener unificados todos mis retos.
- Poder crear retos de cualquier tipo. Para diferenciar unas de otras, se crearán secciones según de lo que vaya el reto (salud, conocimiento, miedos, finanzas).
- Crear retos individuales (que sean exclusivamente para el usuario que lo ha creado) o colectivos (que sean, además del propietario del reto, para un grupo de amigos seleccionados por el creador).
- Poder especificar el objetivo del reto y las etapas que se deberán realizar (como la cantidad de etapas a realizar y el objetivo que se debe cumplir en cada una de ellas).
- Poder incorporar en mis retos amigos que me apoyen en el proceso de lograr cumplir con el objetivo del reto.
- Mandar mensajes de ánimo en los retos de un amigo, ya sea modo texto, audio, imagen o vídeo de apoyo.
- La posibilidad de añadir pruebas o evidencias de cualquier formato (texto, audio o vídeo) en una etapa.
- Incluir dentro del reto la recompensa que se obtendrá cuando se supere el reto.

- Poder consultar por qué etapa se va, los objetivos alcanzados, los mensajes de apoyo dados y recibidos, y el estado de consecución del reto de otros compañeros, si es colectivo y el creador del reto me da permiso para ello.
- Poder saber en qué estado de avance se encuentra los retos que forma parte un usuario (propuesto, en proceso, finalizado)

Por lo tanto, las funcionalidades que deseamos que nuestra aplicación tenga para cumplir con los objetivos de este trabajo son las siguientes:

1. *Funcionalidades relacionadas con el usuario:* Nos queremos centrar en todas las funcionalidades principales correspondientes con el usuario (para diferenciar aquella persona que está registrada en el sistema de la que no):
 - Registrar un usuario.* Insertar una nueva persona al sistema dando previamente sus datos personales, como su nombre y su correo electrónico.
 - Iniciar sesión.* A partir del correo dado para su registro, poder entrar a la aplicación siempre que lo desee el usuario.
 - Recuperar la cuenta.* En el momento que el cliente no recuerde cómo acceder a su cuenta, darle los pasos a seguir para poder acceder de nuevo a ella.
 - Cerrar sesión.* Eliminar la sesión que tenga activa en un dispositivo.
 - Visualizar el perfil de usuario.* Visualizar parte de los datos que el usuario haya dado para su registro en la aplicación.
 - Editar el perfil.* Modificar gran parte de sus datos personales, como la contraseña con la que accede al sistema.
 - Borrar la cuenta.* Eliminar definitivamente todos los datos que tenga almacenados en la aplicación.
 - Recibir notificaciones.* Permitir la creación de notificaciones para avisar al usuario de algún cambio de su cuenta.
 - Listado de notificaciones.* Listar las notificaciones que le han llegado a un cliente pero que aún no las ha leído.
2. *Funcionalidades relacionadas con las amistades:* Nos ubicamos en las funcionalidades correspondientes con las amistades que pueda tener un usuario:

- a. *Lista de amigos.* Listado de otros usuarios, previamente registrados en la aplicación, que se consideran amigos (que se siguen) entre ellos.
 - b. *Ver amigo.* Ver el perfil de una persona que se tenga como amiga.
 - c. *Solicitar amistad.* Preguntar a un usuario si quiere crear una unión de amistad entre ellos dos.
 - d. *Aceptar amistad.* Aceptar la petición de amistad y comenzar a seguir a esa persona también.
 - e. *Rechazar amistad.* Rechazar la petición de amistad.
 - f. *Borrar amigo.* Dejar de seguir a esa persona y eliminar los datos que tengan en común.
3. *Funcionalidades relacionadas con los retos:* Nos enfocaremos en las actividades principales de este proyecto, la generación de retos:
- a. *Crear reto.* Escribir el objetivo, las etapas junto a sus subobjetivos que tendrán, las personas que formarán parte animando o participando del nuevo reto.
 - b. *Editar reto.* Modificación de los datos de un reto ya creado.
 - c. *Lista de retos.* Listado de todos los retos en los que participe o anime esa persona.
 - d. *Ver reto.* Muestra los datos insertados anteriormente de un reto dado.
 - e. *Filtrar retos.* Mostrar aquellos retos que cumplan una condición. Se podrá filtrar por reto individual o colectivo, por categoría, por estado de avance o por si está animando en él.
 - f. *Añadir evidencias dentro de la etapa actual.* Insertar en la etapa en la que se está trabajando en ese momento una evidencia de su progreso.
 - g. *Enviar apoyo.* Mandar mensajes de ánimo a los participantes del reto.
 - h. *Borrar reto.* Eliminar todos los datos relacionados con el reto y borrarlo dentro de la cuenta de cada usuario que formaba parte de él (ya sea como uno de los amigos que realizaba el reto con él como el amigo que se encargaba de apoyarlo).

Funcionalidades
<p><i>Funcionalidades relacionadas con el usuario:</i></p> <ul style="list-style-type: none"> ● Registrar usuario. ● Iniciar sesión. ● Recuperar cuenta. ● Cerrar sesión. ● Ver perfil. ● Editar perfil. ● Borrar cuenta. ● Recibir notificaciones ● Listado de notificaciones
<p><i>Funcionalidades relacionadas con la amistad:</i></p> <ul style="list-style-type: none"> ● Lista de amigos. ● Ver amigo. ● Solicitar amistad. ● Aceptar amistad. ● Borrar amigo.
<p><i>Funcionalidades relacionadas con los retos:</i></p> <ul style="list-style-type: none"> ● Crear reto. ● Editar reto. ● Lista de retos. ● Ver reto. ● Filtrar retos. ● Añadir evidencias dentro de la etapa actual. ● Enviar apoyo. ● Borrar reto.

Tabla 9: Resumen de las funcionalidades que tendrá la aplicación a construir.

3.2. Arquitectura del sistema

En este apartado, mostraremos un esquema de cómo funciona el proyecto a través del diagrama de arquitectura del proyecto de la *Figura 20*.

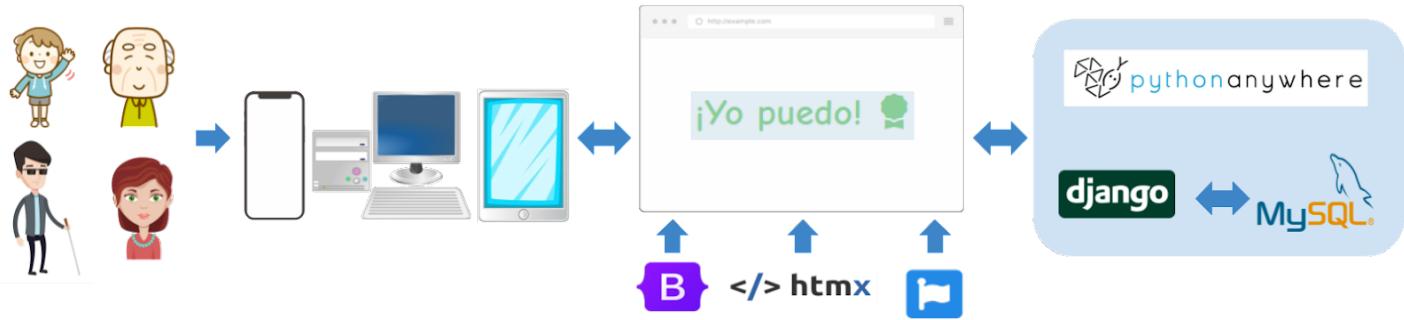


Figura 20: Diagrama de arquitectura de la aplicación de red social saludable.

En ella, podemos ver que los pasos que sigue el proyecto para realizar cada una de las funcionalidades comentadas en el [product backlog](#):

1. El usuario, a través de un ordenador, una tablet o un smartphone, se comunica con la aplicación a través del navegador.
2. Una vez que el usuario ha indicado qué es lo que quiere realizar a través de una petición HTTP, se mira a qué URL quiere ir.
3. Una vez identificada la URL, se llama a la función correspondiente de esa URL dentro del controlador de la aplicación de *Django*, dentro del alojamiento *PythonAnyWhere*.
4. Dentro de la función, se consulta o se modifica, a través de los modelos creados, los datos necesarios de esa funcionalidad en la base de datos de *MySQL*.
5. Ya obtenidos los datos, se genera la plantilla (*template*) con los datos resultantes, a través de *Bootstrap*, para el marco de frontend; de *HTMX*, para las acciones dinámicas de la página web; y de *Font Awesome*, para los iconos de la aplicación.
6. Por último, se manda al navegador la plantilla con la información resultante para mostrarla al usuario.

3.3. Metodología

La metodología que se ha decidido utilizar para el desarrollo de este proyecto es la *Metodología Agile* [57, 58, 59]. Básicamente su objetivo es desarrollar un producto o un servicio de calidad cubriendo las necesidades del cliente en pequeñas porciones, que pueden ser usadas cada una de ellas sin la necesidad de estar terminadas cada una de las funcionalidades del proyecto. Dicho de otra forma, podemos mostrar los avances del proyecto al cliente y este probarlos aunque no esté terminado.

Las razones por las que se ha preferido este tipo de metodología frente a otras, como puede ser la metodología por cascada, son por las que se van a explicar a continuación:

- Mejora el servicio o producto ya que con esta metodología se pueden apreciar mucho más pronto que con otras los errores o los problemas que puede tener una parte del proyecto gracias a la continua comunicación entre el equipo del proyecto y los clientes.
- Ofrece una mejor comunicación con el cliente puesto que este puede formar parte de las distintas etapas del proceso de desarrollo del producto y así obtener el feedback del cliente sobre aquellas tareas completadas.
- Genera mayor compromiso, motivación e implicación entre los desarrolladores de la aplicación al estar constantemente reunidos entre ellos comentando los avances y los problemas que acarrean cada uno de ellos.
- Causa mayor rapidez y eficiencia, e incluso, menor coste porque se trabajan todas las partes del proyecto, asignadas por el especialista correspondiente, y se pueden enviar al cliente pequeñas partes funcionales del servicio para que pueda comentar sobre ellas y corregir ciertos aspectos.
- Al “trocear” el proyecto en partes más pequeñas y al reunirse el equipo con frecuencia para comentar los avances y los bloqueos de cada uno, se consigue aumentar la productividad del equipo.
- Se puede acelerar el retorno de la inversión al poder pactar con el cliente ir recibiendo una pequeña cantidad de dinero como pago según las partes del proyecto completadas.

Ya aclarado el tipo de metodología seleccionado para desarrollar la aplicación de red social de retos para una vida saludable, nos pusimos mano a mano a comenzar con la creación de la aplicación.

En primer lugar, y tal y como comentamos en la introducción de esta memoria, la idea de crear una aplicación que nos permita crear retos para mantener una vida saludable surgió de la *Fundación Purísima Concepción* [60], centro especializado en tratar con personas con discapacidad en Granada.

Para poder lograr desarrollar este proyecto lo mejor posible, estuvimos reuniéndonos con la directora de este centro, *Inmaculada Garrido*, que en adelante y para esta memoria llamaremos cliente, para que nos explicase cómo ella había imaginado que fuera su

aplicación y para qué clase de personas había pensado que la utilizase. También nos reunimos con ella, según fuimos avanzando en las historias de usuario y en el desarrollo de los bocetos, para que nos diera su opinión sobre si eran las funcionalidades suficientes, si necesitaban ciertas restricciones o si hacían falta ciertos cambios en el diseño gráfico de la aplicación para que fuera más accesible y entendible para los usuarios finales.

En cuanto a los pasos realizados con esta metodología para el desarrollo de este proyecto, decidimos dividirlo en las siguientes etapas:

1. Describiremos los términos necesarios para entender los roles y las características que vamos a desarrollar dentro de la aplicación en un glosario.
2. Comentaremos las historias de usuario que describen cada una de las funcionalidades que indicamos que eran necesarias para nuestra aplicación.
3. De las historias de usuario creadas, las ordenamos por prioridad y las dividiremos en las iteraciones con la misma duración.
4. Para cada iteración, detallaremos las tareas a realizar junto a los diseños de bocetos y de base de datos necesarios para su desarrollo y a sus implementaciones.

3.4. Glosario de términos

Para que en los siguientes apartados de esta memoria se pueda comprender lo mejor posible a lo que nos queremos referirnos en cada aspecto del diseño de esta aplicación, a continuación ofrecemos los siguientes términos:

- **Persona.** Tipo de usuario que disfruta de los servicios que ofrece el sistema.
- **Sistema.** La aplicación que se va a crear cumpliendo los requisitos del proyecto.
- **Servicios.** Las funcionalidades (las HUs) que tiene el sistema.
- **Cuenta.** Perfil de una persona creada dentro del sistema.
- **Reto.** Objetivo que la persona desea cumplir en un periodo determinado.
- **Reto individual.** Reto en el que participa solamente una persona.
- **Reto colectivo.** Reto en el que participa más de una persona.
- **Objetivo.** La meta que quiera cumplir una persona.
- **Etapa.** Paso a realizar para completar un reto.

- **Categoría.** Clase que define un reto. Estas son: Deporte, conocimiento, finanzas y miedos.
- **Estado.** Situación en la que está un reto o una etapa de un reto. Solo hay tres estados: propuesto, en proceso y finalizado.
- **Coordinador.** Persona propietaria de un reto que tiene permisos especiales sobre ese reto, como su edición o su eliminación.
- **Evidencia / Prueba.** Evidencias en cualquier formato que un participante envía en una etapa para saber el progreso en su reto.
- **Amigo.** Persona que se tiene una relación de amistad con otra persona y que pueden formar parte de animarse entre ellos o de realizar un reto juntos.
- **Tutor.** Persona responsable del usuario.
- **Tutelado.** Usuario que está supervisado por un tutor.
- **Notificación.** Mensaje que recibe una persona sobre algún cambio en su cuenta. Estas notificaciones son de los siguientes tipos: nuevas solicitudes de amistad, ser añadido a un reto como participante o como animador y recibimiento de mensajes de ánimo de uno de sus retos.
- **Animador.** Usuario que es amigo de la persona que está realizando el reto que únicamente tiene permitido apoyar a esa persona a lograrlo.
- **Ánimo.** Mensaje de apoyo, dentro de un reto, que manda uno de los animadores del reto en cuestión.
- **Superanimador.** Animador que puede ver los comentarios de ánimo del resto de animadores de un reto.
- **Participante.** Usuario que realiza el reto.

3.5. Product Backlog

En cuanto a la forma en que vamos a implementar las funcionalidades anteriores, vamos a definir las historias de usuario ordenadas según su prioridad. Esas historias de usuario son las siguientes:

- **[HU1] Como persona, necesito registrarme en el sistema para poder acceder a los servicios del sistema.**

- Quién: Persona.
 - Cuándo: Cuando quiera acceder al sistema por primera vez.
 - Entonces: La persona puede acceder a los servicios que ofrece el sistema cuando desee.
- **[HU2] Como persona, necesito iniciar sesión en el sistema para poder acceder a los servicios del sistema.**
 - Quién: Persona.
 - Cuándo: Cuando quiera acceder al sistema.
 - Entonces: La persona puede acceder a los servicios que ofrece el sistema.
- **[HU3] Como persona, necesito recuperar mi cuenta cuando no me acuerde de mi contraseña.**
 - Quién: Persona.
 - Cuándo: Cuando quiera acceder al sistema pero no se acuerda de su contraseña para entrar en él.
 - Entonces: Se modifica la contraseña guardada dentro de la base de datos del sistema.
- **[HU4] Como persona, necesito cerrar mi sesión.**
 - Quién: Persona.
 - Cuándo: Cuando quiera dejar de acceder al sistema con un dispositivo sin necesidad de eliminar su cuenta.
 - Entonces: Se eliminan los datos de la sesión y debe volver a iniciar sesión para acceder a los servicios del sistema.
- **[HU5] Como persona, necesito ver mi perfil.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver sus datos personales.
 - Entonces: Se muestran los datos guardados en el sistema de ese usuario.
- **[HU6] Como persona, deseo cambiar mi perfil.**

- Quién: Persona.
 - Cuándo: Cuando quiera modificar sus datos personales.
 - Entonces: Se muestran los datos personales guardados y se guardan los cambios realizados.
- **[HU7] Como persona, deseo cancelar mi cuenta.**
 - Quién: Persona.
 - Cuándo: Cuando quiera restringir los servicios del sistema.
 - Entonces: La persona se eliminará del sistema y ya no podrá utilizar de nuevo los servicios de la aplicación con esa cuenta.
- **[HU8] Como persona, necesito crear un reto individual.**
 - Quién: Persona.
 - Cuándo: Cuando quiera retarse a llegar a un cierto reto.
 - Entonces: La persona tiene un reto que debe cumplir según en las etapas que lo haya dividido.
- **[HU9] Como persona, necesito ver mis retos según el tipo de reto (individual o colectivo).**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver los retos que tiene creados, los que le quedan por alcanzar, los ya realizados y los que está animando.
 - Entonces: Se muestra una lista de retos según el tipo señalado.
- **[HU10] Como persona, necesito filtrar mis retos según el estado en el que se encuentren.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver los retos de un estado dado previamente.
 - Entonces: Se muestra una lista de retos con el estado indicado.
- **[HU11] Como persona, necesito filtrar mis retos según la categoría que puedan tener.**

- Quién: Persona.
 - Cuándo: Cuando quiera ver los retos de una categoría concreta un estado dado previamente.
 - Entonces: Se muestra una lista de retos.
- **[HU12] Como persona, necesito ver con detalle un reto.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver un reto en concreto.
 - Entonces: Devuelve el reto, los animadores y los participantes que forman parte del reto, las pruebas de cada una de las etapas, las calificaciones de cada una de las etapas de esa persona y los mensajes de ánimo de cada etapa.
 - **[HU13] Como participante, quiero añadir una prueba de mi avance de un reto.**
 - Quién: Participante.
 - Cuándo: Cuando quiera indicar lo que se ha progresado durante la realización del reto.
 - Entonces: Se añadirá la prueba en la etapa correspondiente del reto.
 - **[HU14] Como participante, quiero ver las pruebas de una de las etapas de uno de mis retos.**
 - Quién: Participante.
 - Cuándo: Cuando quiera ver lo que ha progresado durante la realización del reto.
 - Entonces: Se muestran un conjunto de pruebas de la correspondiente etapa.
 - **[HU15] Como participante, quiero calificar cómo me he sentido realizando una etapa de uno de mis retos.**
 - Quién: Participante.
 - Cuándo: Antes de que se indique el final de una de las etapas del reto.

- Entonces: Se guarda la calificación de cómo le ha ido en esa etapa y se marca esa etapa como finalizada y la siguiente a esta en proceso, si todos los participantes la han calificado. Si era la última etapa, se marca el reto como finalizado.
- **[HU16] Como participante, quiero ver mi calificación de una etapa.**
 - Quién: Participante.
 - Cuándo: Cuando desee ver mi calificación de una etapa de un reto concreto.
 - Entonces: Se muestra la calificación de esa etapa.
- **[HU17] Como persona, necesito pedir una solicitud de amistad a un amigo.**
 - Quién: Persona.
 - Cuándo: Cuando quiera incluir amigos en su cuenta.
 - Entonces: Se notifica al usuario destinatario de la petición de amistad del usuario solicitante.
- **[HU18] Como persona, necesito aceptar o rechazar una solicitud de amistad.**
 - Quién: Persona
 - Cuándo: Cuando el usuario haya recibido una solicitud de amistad.
 - Entonces: Si se acepta, se crea una nueva amistad, con lo cual esta persona me empezará a seguir. Sino, se elimina la solicitud de amistad.
- **[HU19] Como persona, necesito buscar nuevos amigos.**
 - Quién: Persona.
 - Cuándo: Cuando quiera incluir amigos en su cuenta.
 - Entonces: Se muestra una lista de usuarios según los resultados de su consulta.
- **[HU20] Como persona, necesito consultar la lista de mis amigos.**
 - Quién: Persona.
 - Cuándo: Cuando quiera consultar su lista de amigos.

- Entonces: Se muestra una lista de usuarios que sean amigos de esa persona.
- **[HU21] Como persona, quiero dejar de seguir a un amigo.**
 - Quién: Persona.
 - Cuándo: Cuando no quiere seguir siendo amigo de un usuario.
 - Entonces: Se eliminan todas las relaciones dentro del sistema con esa persona.
- **[HU22] Como persona, me gustaría ver el perfil de uno de mis amigos.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver, con detalle, el perfil de uno de sus amigos.
 - Entonces: Se muestran los datos de ese amigo más los retos que tienen en común.
- **[HU23] Como coordinador, necesito añadir en mis retos amigos para que me apoyen.**
 - Quién: Coordinador.
 - Cuándo: Cuando necesite ayuda de sus amigos para poder alcanzar el objetivo.
 - Entonces: Se añade la lista de usuarios al reto como animadores.
- **[HU24] Como coordinador, me gustaría decidir qué animador puede ver los mensajes del resto, además de los suyos propios.**
 - Quién: Coordinador.
 - Cuándo: Cuando haya asignado a los animadores y quiera elegir aquellos usuarios que puedan ver los comentarios del resto.
 - Entonces: Se añaden como "superanimadores" aquellas personas seleccionadas.
- **[HU25] Como animador, necesito dejar un mensaje de apoyo a mi amigo.**
 - Quien: Animador.

- Cuándo: Cuando sea animador de un reto de uno de sus amigos y desee dar apoyo.
 - Entonces: Se añade, dentro de la etapa en la que se encuentra actualmente su amigo, un mensaje de ánimo.
- **[HU26] Como persona, me gustaría ver todos los mensajes de ánimo de una de las etapas de mi reto.**
 - Quién: Persona.
 - Cuándo: Cuando sea animador del reto dado y haya mensajes de ánimo dentro de la etapa seleccionada.
 - Entonces: Se muestra un conjunto de mensajes de apoyo.
- **[HU27] Como animador, quiero dejar de seguir apoyando el reto.**
 - Quién: Animador.
 - Cuándo: Cuando el animador ya no desee seguir apoyando a un amigo en su reto.
 - Entonces: Se elimina como animador de ese reto el usuario.
- **[HU28] Como coordinador, quiero eliminar a uno de mis animadores.**
 - Quién: Coordinador.
 - Cuándo: Cuando el coordinador no quiera que ese animador siga apoyándole en un reto dado.
 - Entonces: El animador seleccionado ya no forma parte del reto dado.
- **[HU29] Como persona, quiero crear un reto colectivo con mis amigos.**
 - Quién: Persona.
 - Cuándo: Cuando quiera retarse a llegar a un cierto objetivo con unos amigos, que pasarán a ser participantes del reto.
 - Entonces: La persona crea tiene un reto a realizar con uno o varios amigos que deben cumplir según en las etapas que lo hayan dividido.
- **[HU30] Como persona, quiero añadir a mi reto individual personal participantes.**

- Quién: Persona.
 - Cuándo: Cuando quiera retarse a llegar a un cierto objetivo, que inicialmente lo había planeado para hacerlo solo y ahora quiere añadir amigos a retarse con él.
 - Entonces: La persona tiene un reto con uno o varios amigos que deben cumplir según en las etapas que lo hayan dividido, con lo que un reto individual se convertiría en un reto colectivo.
- **[HU31] Como coordinador, deseo pasar la coordinación de un reto colectivo a uno de los participantes.**
 - Quién: Coordinador.
 - Cuándo: Cuando desee dejar de ser coordinador y pasarle esa responsabilidad a otro participante del reto.
 - Entonces: Se cambia de coordinador.
- **[HU32] Como coordinador, deseo modificar los datos de mi reto.**
 - Quién: Coordinador.
 - Cuándo: Cuando desee modificar alguno de los datos de uno de sus retos.
 - Entonces: Se modifica el reto dado.
- **[HU33] Como coordinador, deseo eliminar el reto.**
 - Quién: Coordinador.
 - Cuándo: Cuando desee eliminar definitivamente uno de sus retos.
 - Entonces: Se elimina el reto del sistema.
- **[HU34] Como coordinador, necesito comenzar el reto.**
 - Quién: Coordinador.
 - Cuándo: Cuando necesite o desee iniciar uno de sus retos.
 - Entonces: Se cambia el estado del reto.
- **[HU35] Como coordinador, quiero eliminar a uno de mis participantes.**

- Quién: Coordinador.
 - Cuándo: Cuando la persona no quiera que ese participante siga en el reto.
 - Entonces: El participante seleccionado ya no forma parte del reto.
- **[HU36] Como persona, quiero consultar mis notificaciones.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver sus notificaciones.
 - Entonces: Se muestra una lista de notificaciones de esa persona.
 - **[HU37] Como persona, quiero ver detalladamente una notificación recibida.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver detalladamente una notificación.
 - Entonces: Se muestra el detalle de la notificación.
 - **[HU38] Como persona, deseo conocer si tengo notificaciones nuevas.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ser informada de tener nuevas notificaciones.
 - Entonces: Se notifica al usuario, visualmente, si hay notificaciones nuevas.

3.6. Sprint Backlog

Dentro de esta sección, indicaremos en cuántos sprints vamos a hacer la aplicación y qué historias de usuario implementaremos en cada uno de dichos sprints.

En cuanto al tiempo estimado de realización de cada una de las iteraciones de este proyecto, hemos pensado en llevarlas a cabo en un máximo de seis semanas.

Por lo tanto, la división de carga de trabajo de este proyecto la haremos de la siguiente manera:

- *Primer sprint:*
 - [HU1] Como persona, necesito registrarme en el sistema para poder acceder a los servicios del sistema.

- [HU2] Como persona, necesito iniciar sesión en el sistema para poder acceder a los servicios del sistema.
 - [HU3] Como persona, necesito recuperar mi cuenta cuando no me acuerde de mi contraseña.
- *Segundo sprint:*
 - [HU4] Como persona, necesito cerrar mi sesión.
 - [HU5] Como persona, necesito ver mi perfil.
 - [HU6] Como persona, deseo cambiar mi perfil.
 - [HU7] Como persona, deseo cancelar mi cuenta.
 - *Tercer sprint:*
 - [HU8] Como persona, necesito crear un reto individual.
 - [HU9] Como persona, necesito ver mis retos según el tipo de reto (individual o colectivo).
 - [HU10] Como persona, necesito filtrar mis retos según el estado en el que se encuentren.
 - [HU11] Como persona, necesito filtrar mis retos según la categoría que puedan tener.
 - [HU12] Como persona, necesito ver con detalle un reto.
 - *Cuarto sprint:*
 - [HU13] Como participante, quiero añadir una prueba de mi avance de un reto.
 - [HU14] Como participante, quiero ver las pruebas de una de las etapas de uno de mis retos.
 - [HU15] Como participante, quiero calificar cómo me he sentido realizando una etapa de uno de mis retos.
 - [HU16] Como participante, quiero ver mi calificación de una etapa.
 - *Quinto sprint:*
 - [HU17] Como persona, necesito pedir una solicitud de amistad a un amigo.

- [HU18] Como persona, necesito aceptar o rechazar una solicitud de amistad.
 - [HU19] Como persona, necesito buscar nuevos amigos.
 - [HU20] Como persona, necesito consultar la lista de mis amigos.
 - [HU21] Como persona, quiero dejar de seguir a un amigo.
 - [HU22] Como persona, me gustaría ver el perfil de uno de mis amigos.
- *Sexto sprint:*
 - [HU23] Como coordinador, necesito añadir en mis retos amigos para que me apoyen.
 - [HU24] Como coordinador, me gustaría decidir qué animador puede ver los mensajes del resto, además de los suyos propios.
 - [HU25] Como animador, necesito dejar un mensaje de apoyo a mi amigo.
 - [HU26] Como persona, me gustaría ver todos los mensajes de ánimo de una de las etapas de mi reto.
 - [HU27] Como animador, quiero dejar de seguir apoyando el reto.
 - [HU28] Como coordinador, quiero eliminar a uno de mis animadores.
 - *Séptimo sprint:*
 - [HU29] Como persona, quiero crear un reto colectivo con mis amigos.
 - [HU30] Como persona, quiero añadir a mi reto individual personal participantes.
 - [HU31] Como coordinador, deseo pasar la coordinación de un reto colectivo a uno de los participantes.
 - [HU32] Como coordinador, deseo modificar los datos de mi reto.
 - [HU33] Como coordinador, deseo eliminar el reto.
 - [HU34] Como coordinador, necesito comenzar el reto.
 - [HU35] Como coordinador, quiero eliminar a uno de mis participantes.
 - *Octavo sprint:*

- [HU36] Como persona, quiero consultar mis notificaciones.
- [HU37] Como persona, quiero ver detalladamente una notificación recibida.
- [HU38] Como persona, deseo conocer si tengo notificaciones nuevas.

Una vez indicadas las historias a realizar en cada sprint, detallaremos qué conjunto de tareas hemos desarrollado en cada sprint, mostrando resultados como los bocetos que hemos diseñado, y, la base de datos generada hasta ese momento.

En cada iteración, y para documentar la parte del backend, vamos también completando un diagrama de clases y Entidad/Relación, e implementando código en ficheros. Para simplificar esta memoria, hemos optado por mostrar solo la versión final de esta documentación en el [ANEXO 1](#).

En cada una de las iteraciones se presentan bocetos de usuario que se van mejorando. Muchas de sus mejoras tienen que ver con la aplicación de guías de usabilidad revisadas en el segundo capítulo de esta memoria. En el [ANEXO 2](#) se hace una revisión de cómo hemos tenido en cuenta esas guías.

3.7. Primer sprint

En el primer sprint nos vamos a centrar en la creación de cuenta y acceso al sistema por parte de un usuario. Por lo tanto, las funcionalidades que hemos pensado que se deben realizar en el primer sprint de desarrollo del producto son las siguientes:

Historias de Usuario

- **[HU1] Como persona, necesito registrarme en el sistema para poder acceder a los servicios del sistema.**
 - Quién: Persona.
 - Cuándo: Cuando quiera acceder al sistema por primera vez.
 - Entonces: La persona puede acceder a los servicios que ofrece el sistema cuando desee.
 - CoS:
 - Los datos requeridos para registrar a una persona son: nombre, correo electrónico, contraseña y foto de perfil.

- El correo electrónico será la clave primaria.
- Si el correo no es de la estructura esperada (“ejemplo@ejemplo.com”) o ya existe dentro del sistema, se devuelve un error.
- Si la contraseña introducida no cumple con los requisitos (debe contener números, letras y símbolos, además de tener una longitud de entre 8 y 16 caracteres), se devuelve un error.
- Se puede escribir la contraseña o elegir aquellas imágenes asociadas a unas palabras según los requerimientos que debe tener la contraseña.
- Se debe introducir una foto en el perfil. Si no, se mostrará un mensaje de error indicando que es obligatorio.
- Para aquellos campos textuales (nombre, correo electrónico y contraseña), se permiten ser escritos o dictados por el usuario.
- Se envía correo electrónico con una clave aleatoria de 10 dígitos, generada por el sistema, y se guarda en la base de datos para confirmar el consenso por parte del tutor para el registro.
- Se envía al correo electrónico una clave aleatoria de 15 dígitos fija, generada por el sistema, y se guarda en la base de datos para los casos en los que no se puede acceder a la clave temporal mandada.
- Cada campo del formulario que el usuario debe llenar debe tener un título y un ícono asociado a ese campo.
- El usuario debe escribir la clave aleatoria temporal o la fija para confirmar el permiso de registro en el sistema.
- Si el usuario falla 3 veces en escribir la clave de confirmación, no se guarda la cuenta y se le obliga a empezar de 0.
- Si todos los datos mandados son los correctos y la clave mandada es la generada por el sistema, se guarda en la base de datos la siguiente información:
 - El nombre.
 - El correo electrónico.

- La contraseña (previamente cifrada).
 - La ubicación local de la foto de perfil proporcionada.
 - Una clave temporal.
 - Una clave fija.
- **[HU2] Como persona, necesito iniciar sesión en el sistema para poder acceder a los servicios del sistema.**
 - Quién: Persona.
 - Cuándo: Cuando quiera acceder al sistema.
 - Entonces: La persona puede acceder a los servicios que ofrece el sistema.
 - *CoS:*
 - Se le pedirá al usuario la inserción del correo electrónico y de la contraseña asociado a su cuenta para poder acceder.
 - Si el correo no es de la estructura esperada (“ejemplo@ejemplo.com”) o no existe dentro del sistema, se devuelve un error.
 - Si la contraseña enviada no es la misma que la que el sistema tiene almacenada, se devuelve un error.
 - Para aquellos campos textuales (nombre, correo electrónico y contraseña), se permiten ser escritos o dictados por el usuario.
 - Se envía correo electrónico con una clave aleatoria de 10 dígitos, generada por el sistema, para confirmar el consentimiento por parte del tutor para el registro.
 - Cada campo del formulario que el usuario debe llenar debe tener un título y un ícono asociado a ese campo.
 - El usuario debe escribir la clave aleatoria temporal o la fija para confirmar el permiso de acceso al sistema.
 - Si el usuario falla 3 veces en escribir la clave de confirmación, se le manda a la pantalla de registro para que empiece de 0.

- Si se cumplen con todos los requisitos para ingresar en el sistema, se envía a la página de listado de retos.
- [HU3] **Como persona, necesito recuperar mi cuenta cuando no me acuerde de mi contraseña.**
 - Quién: Persona.
 - Cuándo: Cuando quiera acceder al sistema pero no se acuerda de su contraseña para entrar en él.
 - Entonces: Se modifica la contraseña guardada dentro de la base de datos del sistema.
 - CoS:
 - Para recuperar la cuenta, el usuario debe indicar el correo electrónico de la cuenta que quiera recuperar.
 - El correo escrito por el usuario, debe estar registrado en el sistema.
 - Se enviará a la dirección de correo electrónico de esa cuenta un correo con un enlace para verificar que es esa persona y se avisará por pantalla del envío de dicho correo.
 - Si pincha sobre el enlace enviado, se manda directamente a la persona a que cambie su contraseña.
 - Si la contraseña introducida no cumple con los requisitos (debe contener números, letras y símbolos, además de tener una longitud de entre 8 y 16 caracteres), se devuelve un error.
 - Se puede escribir la contraseña o elegir aquellas imágenes asociadas a unas palabras según los requerimientos que debe tener la contraseña.
 - Si la contraseña introducida es la correcta, se manda al usuario para que inicie sesión con la nueva contraseña.

Bocetos

En primer lugar, en las *Figuras 21, 22 y 23* podemos ver los bocetos en ordenador, tablet y móvil (en el mismo orden que las figuras mencionadas) de las funcionalidades de

registrar e iniciar sesión de un usuario. En ellas podemos ver en ambas páginas un formulario sencillo donde el usuario puede escribir sus datos personales para darse de alta en el sistema o para ingresar de nuevo en él.

Los bocetos de escritorio y de tablet son iguales a la hora de registrar e iniciar sesión un usuario, mientras que el de móvil es ligeramente distinto. La diferencia que podemos encontrar es que cuando queramos iniciar sesión en el formato móvil aumenta el tamaño de la parte de la pantalla encargada de mostrar el inicio de sesión, para no sobrecargar de información la pequeña pantalla del cliente, mientras que en el resto de los formatos mostraremos una ventana pequeña encima de la principal, para no estar cargando de nuevo otra página para mostrar únicamente un pequeño formulario ni volver a traer al usuario la de registro si quiere volver atrás.

Para volver a la página del registro, si se está accediendo a la app a través del móvil bastaría con dar al botón de la esquina superior izquierda, mientras que en el ordenador o la tablet sería pulsar sobre una parte de la pantalla en la que no se encuentre la ventana.

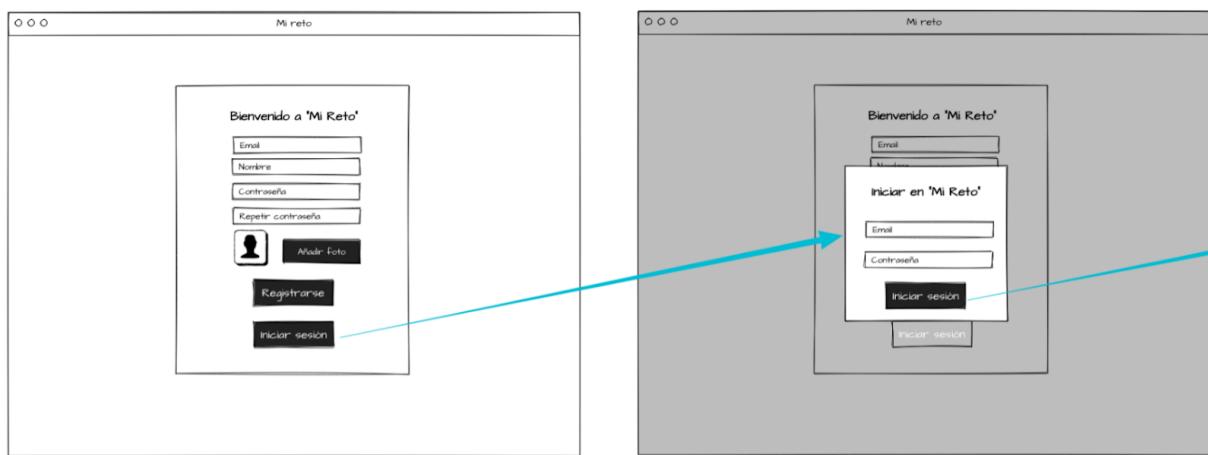


Figura 21: Registro e inicio de sesión en escritorio.

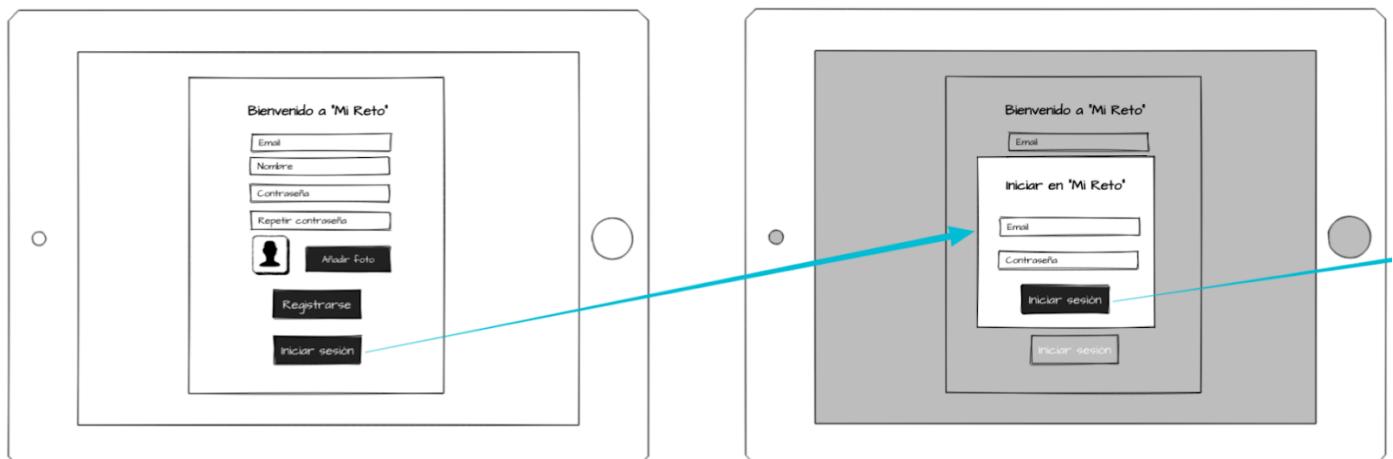


Figura 22: Funcionalidades de registro e inicio sesión de un usuario en formato tablet.

guías de WCAG

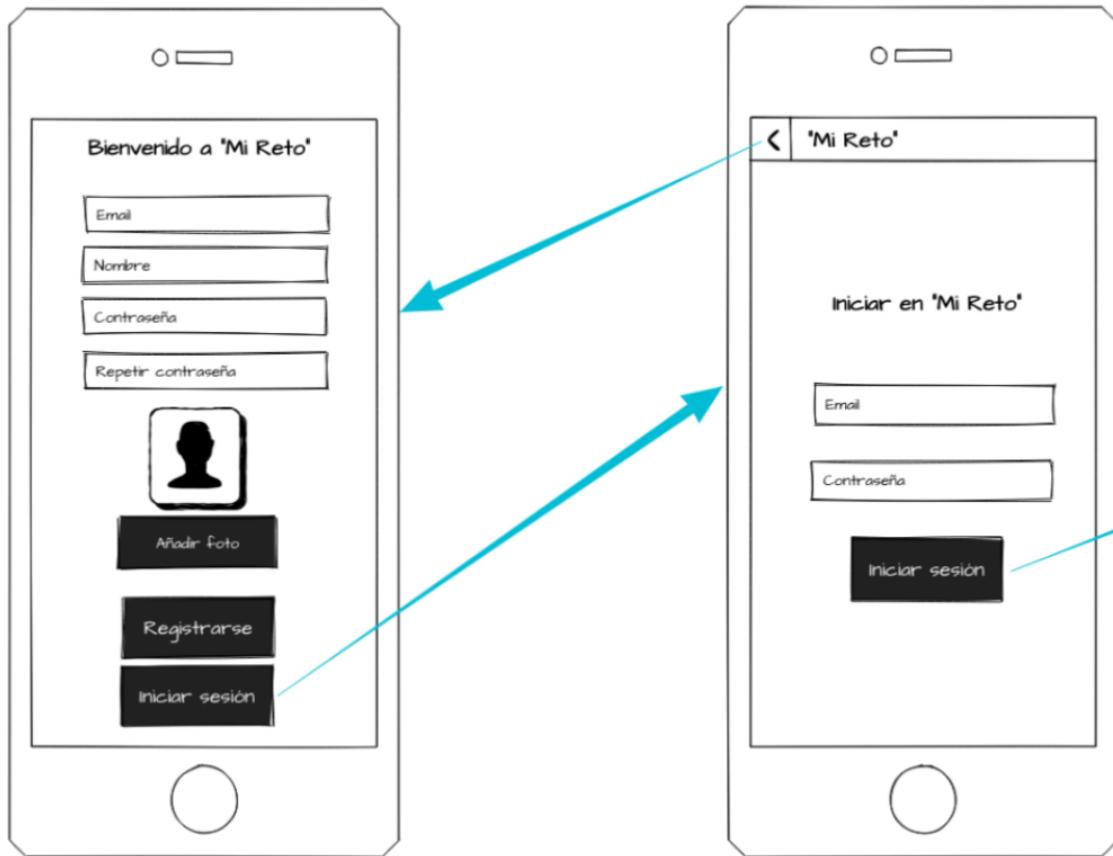


Figura 23: Versión móvil de inicio de sesión y registro.

Las desventajas que hallamos al observar detenidamente los anteriores bocetos son las siguientes, la mayoría de ellas relacionadas con accesibilidad y detectadas durante la reunión con el cliente:

- Para una persona con discapacidad cognitiva, dificultades de lectura/escritura o que no conozca el idioma, no sabría qué poner en cada campo del formulario ni las funcionalidades de cada botón de la pantalla. Por ello, en los siguientes diseños nos preocupamos de añadir pequeños símbolos junto al texto descriptivo de la funcionalidad o de los datos a introducir que expliquen de forma visual, cuál es el contenido que deben tener los apartados del formulario para que nuestra aplicación cumpla con la propiedad comprensible de la [guías de WCAG](#), como podemos ver detenidamente en el [ANEXO 2](#).
- Solamente se permite escribir los datos que nos pide el formulario a través de un teclado, sin dar otras posibilidades. Así que a partir de ese momento, en todos los formularios debe haber un botón que active el micrófono y traduzca lo que esté escuchando a texto.

- Solo se puede introducir una contraseña textual. Para ayudar a aquellas personas con discapacidad física o cognitiva, o con personas con pocos conocimientos sobre escritura, hemos pensado en añadir las contraseñas visuales puesto que las imágenes serían botones más grandes que las teclas de un teclado y fáciles de memorizar.
- No hay un control parental. Por lo tanto, es necesario diseñar de alguna forma dicho control para que los padres o los tutores del usuario estén al tanto del deseo de esa persona de entrar en el sistema.

Por lo tanto, generamos otra versión para cada pantalla, lo que soluciona los problemas que hemos comentado. Además, a partir de esta versión, decidimos crear los bocetos con colores para así planear y testear qué colores serían los más accesibles.

Esos bocetos los podemos observar en las *Figuras 23, 24 y 25* para ordenador, tablet y móvil. Los cambios que se han realizado en comparación con los primeros bocetos son los siguientes:

- Incorporación de pictogramas en las etiquetas de los elementos visuales de las pantallas.
- Adjuntar la posibilidad de dictar el texto a través de un mecanismo de traducción auditiva a textual.
- Inserción de la contraseña a través de imágenes, además de ser dictada o escrita.
- Implantación de un control parental en el inicio de sesión y registro de un cliente. Para aquellas pantallas de escritorio y de tablet, se mostrará como una ventana pequeña encima de la principal, donde el usuario deberá insertar el código aleatorio enviado, mientras que en los móviles será por medio de una pantalla completa.
- Diferenciar las acciones con distintos colores en los botones:
 - Los botones de activar el micrófono con el de seleccionar imágenes o con el de dictar el texto al usuario.
 - El botón y el título de la página de inicio de sesión con los de registrarse, para que así pueda conocer la acción del botón con el título de la página al tener el mismo color.

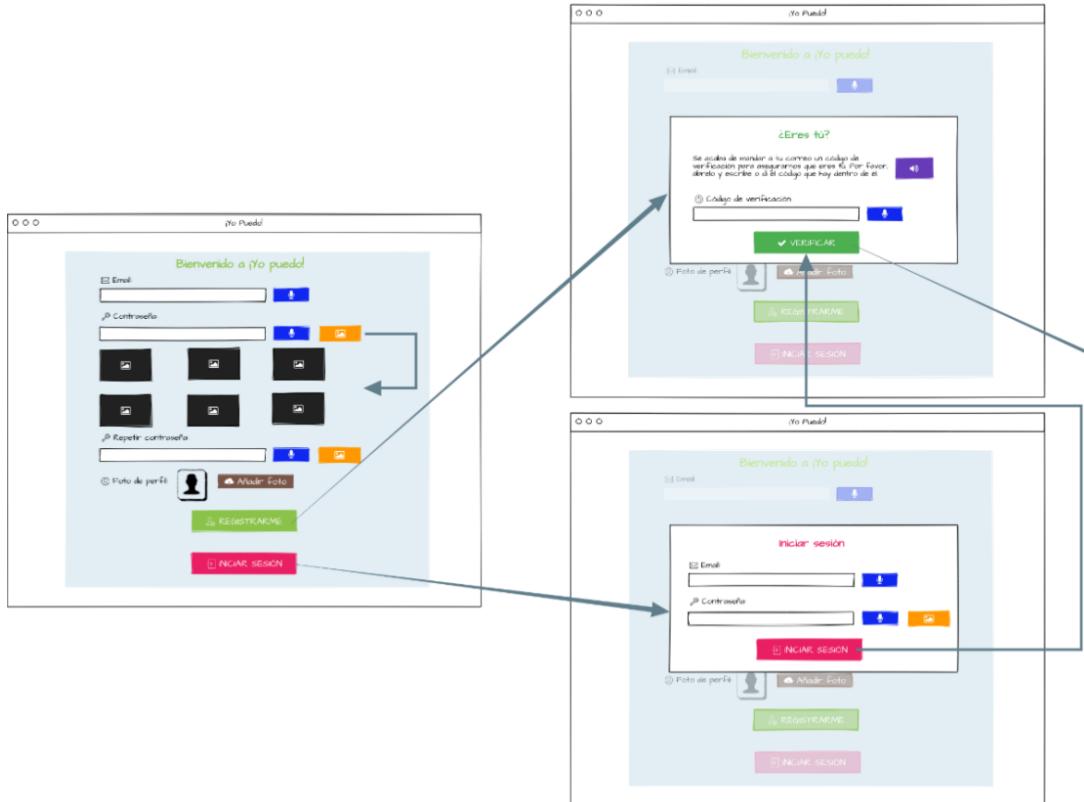


Figura 24: Segunda versión de inicio de sesión y registro en formato escritorio.

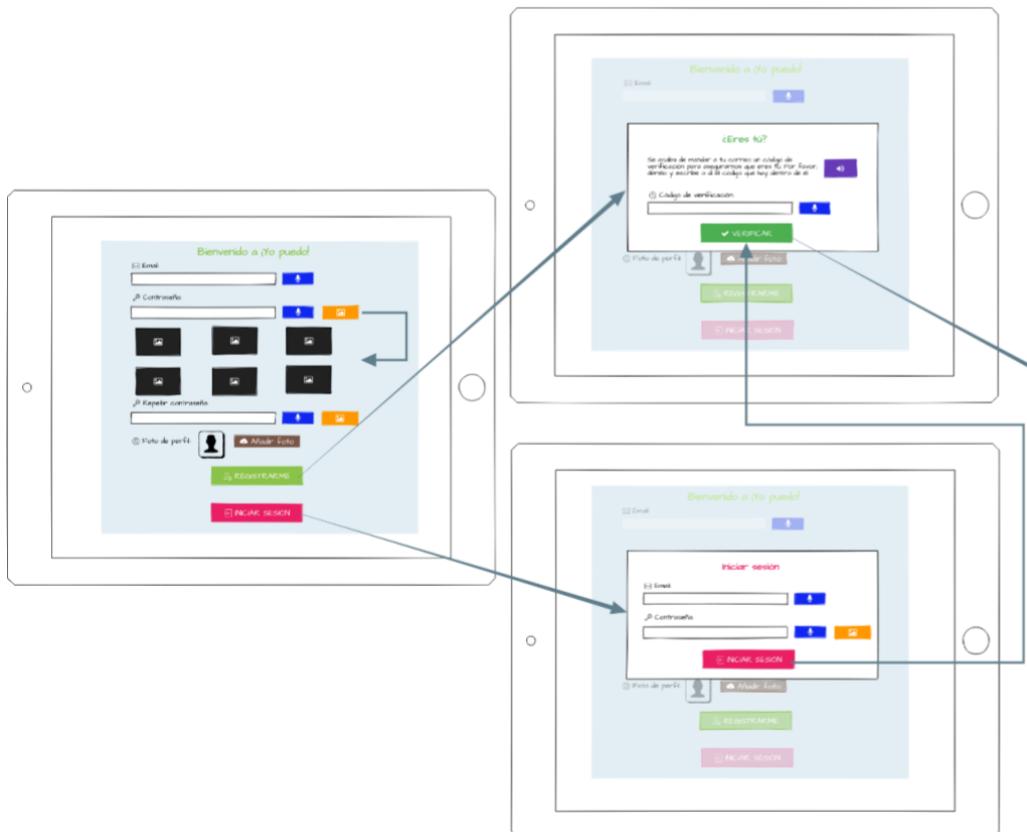


Figura 25: Segunda versión de la visualización de inicio de sesión y registro de un usuario en tablet.

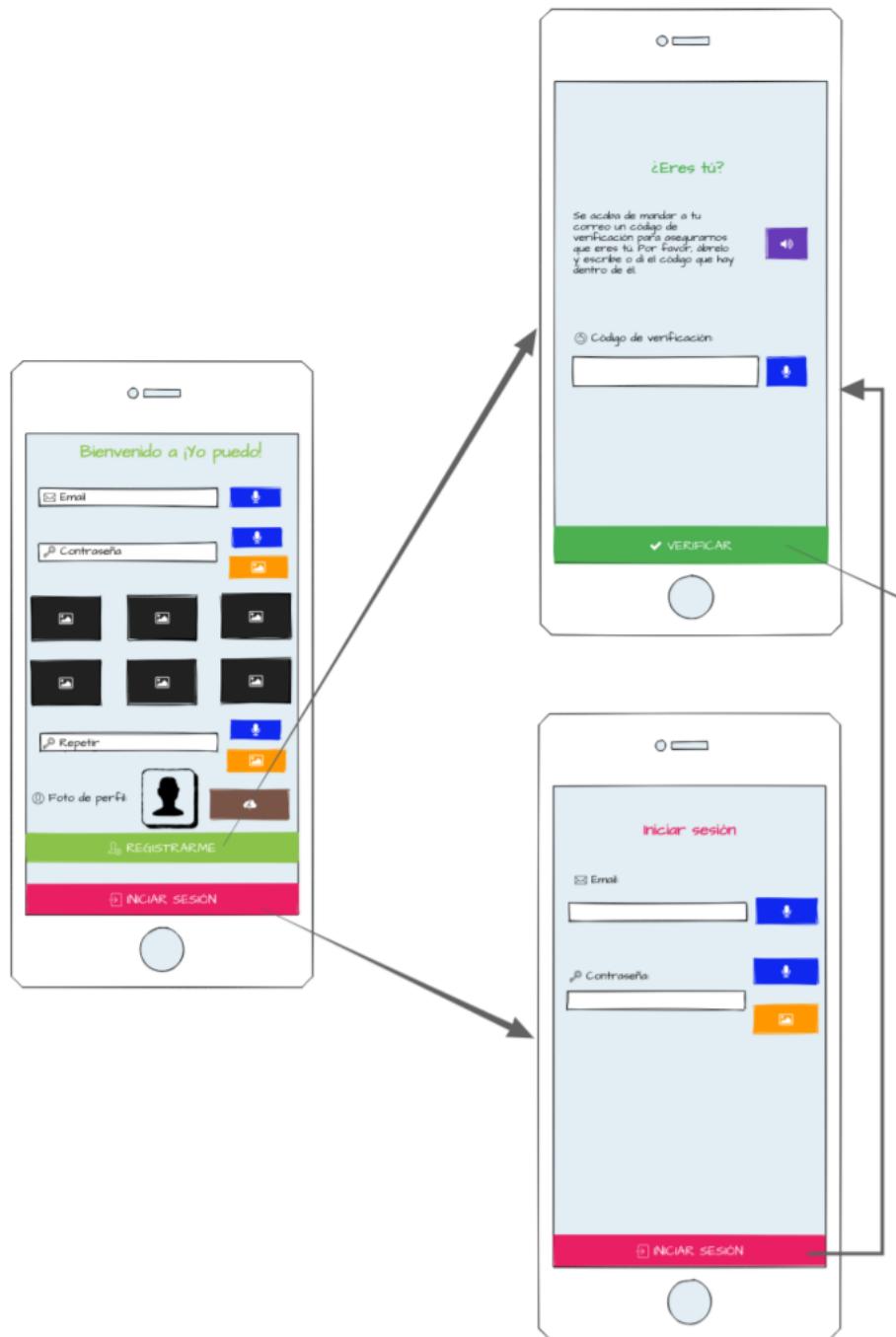


Figura 26: Segunda versión de los bocetos en móvil de inicio de sesión y registro.

Aún habiendo mejorado bastante el diseño de la interfaz de usuario de estas funcionalidades, hemos encontrado pegas para no considerarlas perfectas aún durante otra reunión mantenida con el cliente. En concreto, hemos detectado que si un usuario no recuerda su contraseña, no se halla ninguna forma de recuperar su cuenta.

Ese defecto fue arreglado en los siguientes bocetos de escritorio (*Figura 27*), de tablet (*Figura 28*) y de móvil (*Figura 29*):

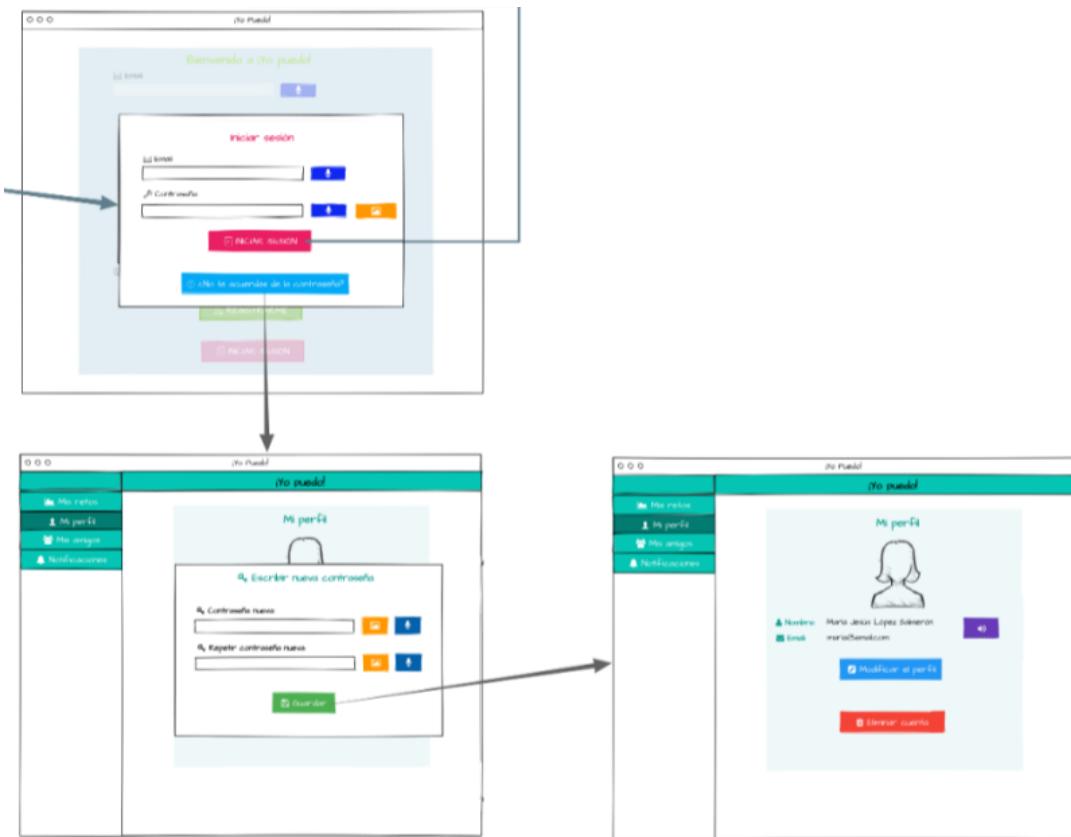


Figura 27: Tercera versión con la recuperación de cuenta para ordenador.

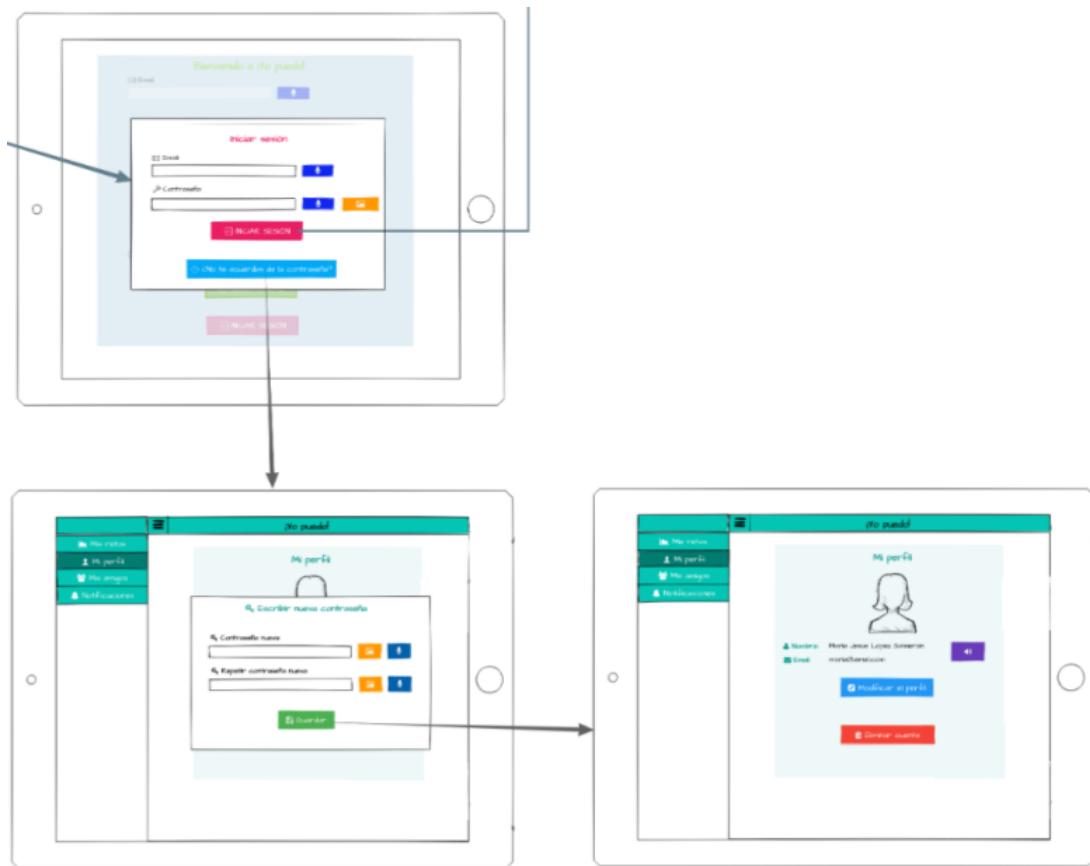


Figura 28: Tercera versión con la recuperación de una cuenta para tablets.

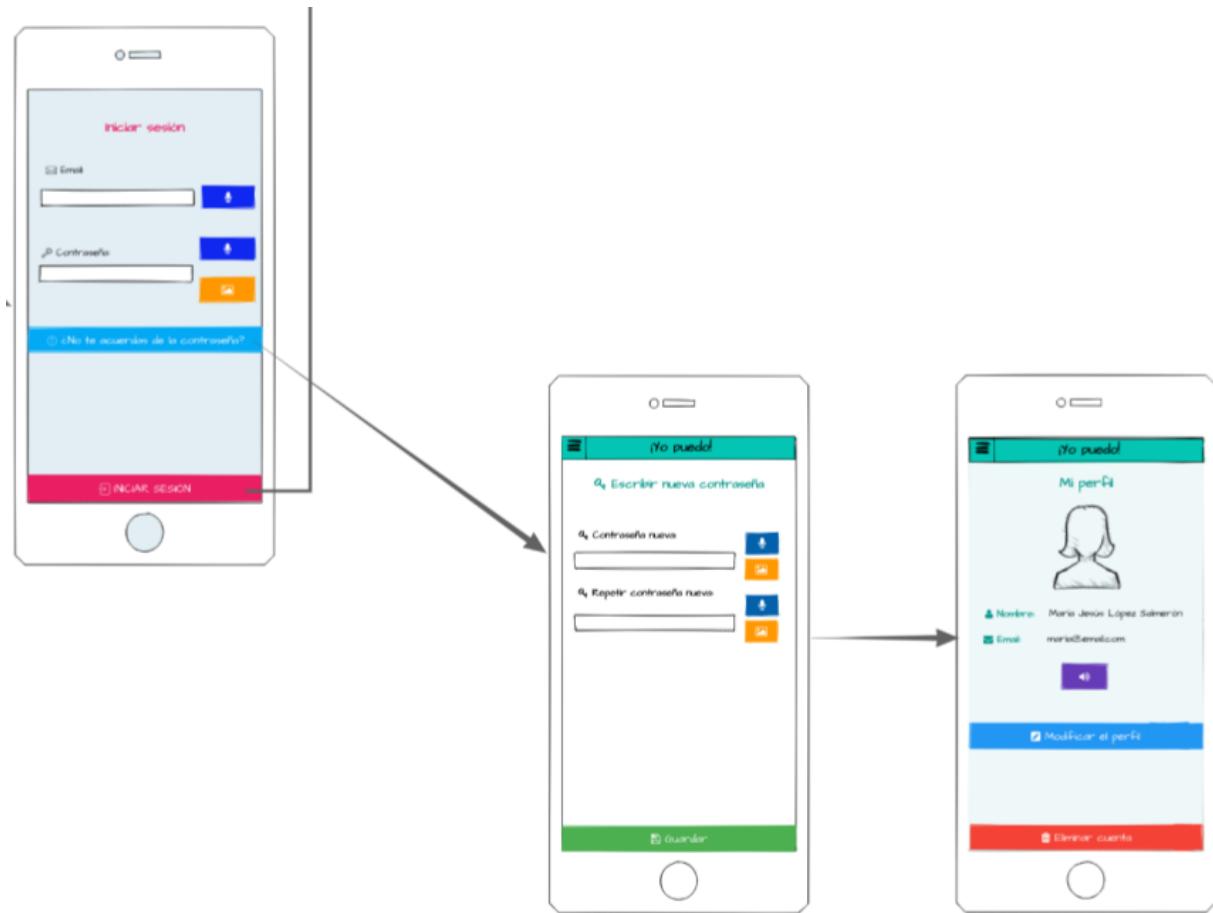


Figura 29: Tercera versión móvil junto a recuperación de cuenta para móvil.

No ha sido necesario modificar los diseños de la interfaz gráfica para algunas funcionalidades: la parte del registro de una persona y la pantalla del pin parental.

En cuanto al inicio de sesión, podemos ver un nuevo botón azul claro “¿No te acuerdas de la contraseña?” que tiene como función mandar un correo al usuario con un enlace que lo lleva directamente a la página donde debe cambiar la contraseña y, una vez guardados los cambios, acceder a su perfil.

Únicamente, para salir de las pantallas de iniciar sesión o recuperar cuenta, sería necesario añadir una cruz en la esquina de la ventana pequeña para poder salir de ella y volver a la pantalla de registro.

Base de datos

En este apartado vamos a presentar la base de datos que generamos en ese momento para poder implementar las historias de usuario comentadas anteriormente. En dichas historias únicamente generamos una entidad, *Usuario*, que va a ser la encargada de

almacenar y manejar los datos que necesita el sistema para el inicio de sesión de una persona.

La información que va a utilizar, según las historias que hemos creado anteriormente son: email (que será el dato que nos ayude a identificar a los usuarios dentro del sistema), nombre y foto de perfil (para que el resto de personas puedan conocer a la otra persona que no sea por su email), contraseña, clave fija y clave aleatoria (para cerciorarnos de que la persona que realiza una acción sobre el sistema sea la dueña de la cuenta).

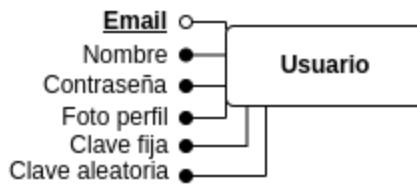


Figura 30: Diagrama de Entidad/Relación del primer sprint.

Implementación

En cuanto a la implementación de las tareas de este sprint, vamos a dividirlas en las siguientes secciones: en la configuración del servidor y de la base de datos, en el registro de un usuario en la aplicación, en el pin parental, en el inicio de sesión y en la recuperación de la cuenta de un usuario.

Configuración del servidor y de la base de datos

Antes de comenzar a implementar las funcionalidades descritas en el apartado de [Historias de Usuario](#), tenemos que configurar el alojamiento web para que permita utilizar nuestra aplicación [61].

Por lo tanto, creamos una cuenta de usuario dentro de *PythonAnyWhere* y, una vez iniciada sesión, abrimos una consola, clonamos el proyecto (que se encuentra dentro de la plataforma *GitHub*) y, por último, preparamos el despliegue de la aplicación [62].

Una vez preparado el despliegue de la aplicación dentro del alojamiento seleccionado, indicamos dentro del servidor (como en la pantalla de la *Figura 31*) y de la configuración del proyecto dónde se van a encontrar los archivos estáticos y de media de nuestro proyecto dentro del almacenamiento interno del servidor [61].

URL	Directory	Delete
/media/	/home/mariajesuslopez/mariajesuslopez.pythonanywhere.com/TFM/media/	
/static/	/home/mariajesuslopez/mariajesuslopez.pythonanywhere.com/TFM/static/	
Enter URL	Enter path	

Figura 31: Configuración del servidor para indicar dónde se encuentran los archivos estáticos y de media.

A continuación, es momento de preparar la base de datos, en este caso MySQL. En primer lugar, creamos el usuario, que vamos a utilizar dentro de la aplicación para que se comunique con la base de datos, y generamos la base de datos del proyecto (nombrada como *mariajesus.lopez\$YOPUEDO*), toda esa información la podemos ver mostrada en la Figura 32 [61]. Seguidamente, definimos dentro de la configuración del proyecto (dentro del fichero *settings.py*) qué tipo de base de datos se va a utilizar junto a la información necesaria para que la aplicación se pueda conectar a la base de datos del servidor [63].

The screenshot shows the MySQL settings page on PythonAnywhere. At the top, there are navigation links: Send feedback, Forums, Help, Blog, Account, and Log out. Below that is the PythonAnywhere logo and a link to ANACONDA. The main section is titled "MySQL settings". It shows "Connecting:" with fields for Database host address (mariajesuslopez.mysql.pythonanywhere-services.com) and Username (mariajesuslopez). A yellow box highlights the "mariajesuslopez\$YOPUEDO" entry under "Your databases". Other entries include "mariajesuslopez\$default". There are also "Dashboard", "Consoles", "Files", "Web", "Tasks", and "Databases" links at the top right.

Figura 32: Captura de pantalla de la configuración de la base de datos dentro de PythonAnyWhere.

Registro

Ya configurado el servidor y conectada la base de datos del proyecto a la aplicación, vamos a empezar a implementar la funcionalidad de registrarse al sistema.

En primer lugar, vamos a crear la tabla que contenga los datos de un usuario dentro de la base de datos (la entidad *Usuario* de la *Figura 30*). *Django* ofrece ya un modelo que almacena los datos de un usuario [64], pero vimos que no tenía los datos suficientes para que nosotros pudiéramos utilizarla en nuestra aplicación, por lo que decidimos crear un modelo de usuario personalizado [65]. Después, migramos dicho modelo para que se convierta en tabla dentro de la base de datos [63].

Posteriormente de la creación del almacenamiento de usuarios dentro de la base de datos, nos centramos en la creación del formulario para recoger esos datos. Para ello únicamente indicamos los campos necesarios (como podemos ver en el resultado de la implementación en la *Figura 33*) junto a las características que deben tener (tanto restricciones como los atributos que deseamos tener [66] para, por ejemplo, poder implementar una visualización a través del marco *Bootstrap* añadiendo una clase al elemento de HTML).

Una vez creado el formulario, es momento de crear la parte backend de esta tarea. Dicha tarea únicamente pedimos que haga lo siguiente:

1. Crear un objeto con el formulario anterior.
2. Realizar la acción correspondiente dependiendo de la petición que haga el usuario:
 - a. Si la petición es una petición GET, enviamos el formulario vacío.
 - b. Si es una petición POST, recogemos los datos del formulario y lo validamos [67]. Si los datos son los correctos, encriptamos la contraseña [68], guardamos los datos del usuario en la base de datos, guardamos la foto de perfil añadida [69] en el almacenamiento físico del servidor y procedemos a lanzar el pin parental. Sino, enviamos el formulario anterior con sus correspondientes mensajes de error.

Creado el backend, nos centramos ahora en el frontend. Únicamente cogemos el formulario dado, junto a los botones necesarios y los títulos de cada campo (además de los iconos de *Font Awesome* describiéndolos), como de la pantalla de la izquierda de los bocetos de la *Figura 24*, *25* y *26*, y lo mostramos de forma ordenada, gracias a *Bootstrap*, como podemos ver en la *Figura 33*.

Bienvenido a ¡Yo puedo! 🌱

Email: ejemplo@ejemplo.com 

Nombre: María Jesús López 

Contraseña:  

Repetir contraseña:  

Foto de perfil: Seleccionar archivo Ninguno archivo selec.

 REGISTRARSE

 INICIAR SESIÓN

Bienvenido a ¡Yo puedo! 🌱

Email: ejemplo@ejemplo.com 

Nombre: María Jesús López 

Contraseña:  

Repetir contraseña:  

Foto de perfil: Seleccionar archivo Ningun...o selec.

 REGISTRARSE

 INICIAR SESIÓN

Bienvenido a ¡Yo puedo!

Email: ejemplo@ejemplo.com

Nombre: María Jesús López

Contraseña:

Repetir contraseña:

Foto de perfil: Seleccionar archivo Ninguno archivo selec.

REGISTRARSE

INICIAR SESIÓN

Figura 33: Implementación de registro en ordenador, móvil y tablet.

Como podemos observar en las distintas capturas de pantalla de la Figura 33, el usuario puede insertar el texto tanto escribiendo o dictándolo a través del botón del micrófono (como el que señalamos en la Figura 34). Dicha transcripción de audio a texto [70, 71] la realizamos en el frontend a través de Javascript activando el micrófono del dispositivo y escuchando la conversación que posteriormente se transcribirá (en el idioma que hayamos configurado previamente).

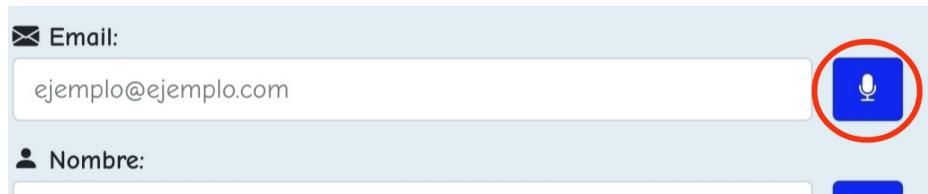


Figura 34: Implementación del botón de transcripción.

Con respecto a la inserción de la contraseña a través de imágenes, decidimos realizarlo mediante un botón (como podemos ver en la imagen superior de la Figura 35) en el que, si se pulsa sobre él se despliega la lista de imágenes posibles (recogidas de la página oficial de Pictogramas de ARASAAC [72]) para crear una contraseña cuando se cliquea sobre ellas (la segunda imagen de la Figura 35).

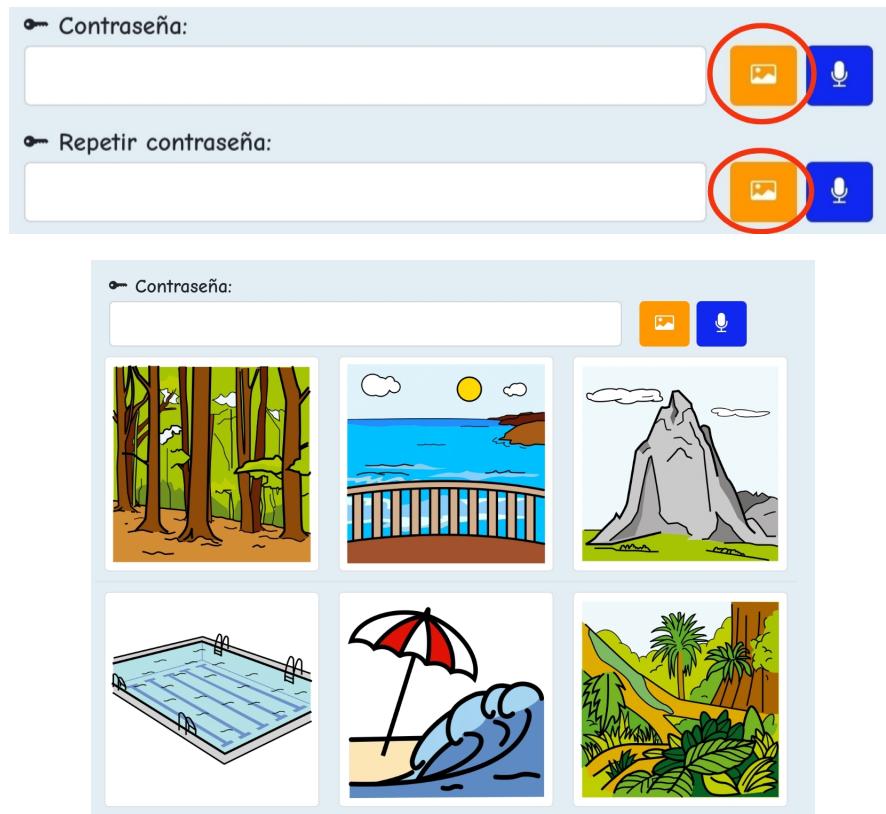


Figura 35: Inserción de la contraseña a través de imágenes.

Pin parental

Cuando estuvimos diseñando las funcionalidades que va a tener nuestro proyecto , nos dimos cuenta que hay algunas de ellas (como esta tarea o aceptar nuevos amigos dentro de una cuenta) que necesitan un cierto control, puesto que la aplicación está pensada para ser utilizada por todo tipo de usuarios, como por ejemplo niños y ciertas personas con una discapacidad que necesitan ser supervisadas por un tutor.

En primer lugar, pensamos realizar ese control parental obligando al nuevo usuario a su registro en el sistema para que escribiera, además de una contraseña y de un correo electrónico, un pin que solo conozca un familiar o tutor, y así poder asegurarnos de que éste ha dado su consentimiento.

Sin embargo, esta solución tiene una garantía de seguridad muy fina, es decir, el problema que vimos a esta solución es que cualquier persona podría aprender dicha clave, ya sean niños o personas tuteladas, por lo que se pasaría por alto ese control.

Otra solución que consideramos que podría ser válida sería utilizar una verificación en dos pasos [73, 74, 75, 76]. El usuario cuando quiera, por ejemplo, aceptar una invitación

de seguimiento de un amigo debe escribir el código aleatorio que se le haya enviado (ya sea por correo o al teléfono móvil) para confirmar que realmente tiene permisos para hacerlo.

A esta última solución se le vieron los siguientes contratiempos:

- A. Si decidimos elegir el método de envío de la verificación a través del correo electrónico, el tutelado podría utilizar su propio correo electrónico para que le envíen a él la clave en vez de a sus tutores o podría tener acceso al correo de sus tutores y así se volvería a perder de nuevo el control parental sobre ese usuario.
- B. Si decidimos mandarlo a través de un SMS, seguimos teniendo el mismo problema que con la anterior opción. Si el tutelado dispone de un teléfono (ya sea propio o que tenga facilidades para coger el de sus tutores sin su permiso), escribiría el pin aleatorio sin supervisión. Además, se requiere de la contratación de un servicio de terceros para enviarlo.
- C. Si realizamos el control a través de la biometría (huellas dactilares, reconocimiento facial, reconocimiento de voz) en vez de utilizar una clave, no todos los dispositivos son capaces de soportar esa tecnología, porque puede que se acceda a la aplicación mediante un ordenador o porque puede que el dispositivo que se esté usando sea de una versión anterior a la creación de la biometría. También puede darse el caso de que el dispositivo que esté utilizando ese usuario tenga configurada la biometría para que verifique que él es un usuario autorizado cuando realice alguna acción en la que se requiera permisos del propietario.
- D. Si determinamos que el pin parental puede estar dentro de un dispositivo como un USB, NFC o BTLE (Bluetooth Low Energy), aunque le ahorraremos al usuario la necesidad de estar escribiendo un código y facilitamos aquellos momentos en los que no dispone de una conexión de internet, siempre tiene que llevar encima ese dispositivo para activar la funcionalidad controlada y también puede que el tutelado tenga acceso a ese dispositivo sin la supervisión de su tutor.
- E. Si el usuario pierde o no puede acceder al método seleccionado para crear el control parental, nunca podrá aceptar aquellas partes que requieran un control especial. Una opción para solventar dicha situación es que se puede ofrecer un código de respaldo de una longitud considerable y permanente, dado cuando se registre el usuario.

A continuación, en la siguiente tabla se muestra un pequeño resumen de los métodos que se pueden utilizar junto a sus ventajas y desventajas:

Método	Ventajas	Desventajas
<i>Clave escrita por el tutor</i>	<ul style="list-style-type: none"> • Sencilla implementación • Sencilla para el usuario • Inserción inmediata 	<ul style="list-style-type: none"> • Fácil de recordar por otras personas ajenas al control
<i>Envío de clave aleatoria al correo</i>	<ul style="list-style-type: none"> • Sencilla implementación • Sencilla para el usuario • Rápida 	<ul style="list-style-type: none"> • El tutelado puede registrar su correo personal • El tutelado puede utilizar el correo del tutor sin supervisión
<i>Envío de clave aleatoria al móvil</i>	<ul style="list-style-type: none"> • Sencilla implementación • Sencilla para el usuario • Rápida 	<ul style="list-style-type: none"> • El tutelado puede registrar su propio teléfono • El tutelado puede utilizar el teléfono del tutor sin supervisión • Contratación de un servicio de terceros
<i>Biometría</i>	<ul style="list-style-type: none"> • Sencilla para el usuario • Rápida • Segura • Sin necesidad de insertar clave 	<ul style="list-style-type: none"> • No todos los dispositivos la disponen • Puede estar configurado para que acepte también la del tutelado
<i>Clave en un dispositivo externo</i>	<ul style="list-style-type: none"> • Sencilla para el usuario • Rápida • Segura • Sin necesidad de insertar clave 	<ul style="list-style-type: none"> • Necesidad de llevar el dispositivo encima cuando requiera la clave • El tutelado puede tener acceso al dispositivo

Tabla 10: Resumen de las opciones del pin parental.

Analizando todos los métodos, y aunque se hayan encontrado algunas dificultades con la verificación en dos pasos, pensamos que es la mejor manera de realizar el control parental.

Como recogemos el correo electrónico para identificar al nuevo usuario que se registra en el sistema o al usuario que va a realizar una acción comprometida (como aceptar una solicitud de amistad), se va a optar por enviar la clave aleatoria [77] de 10 dígitos

(disponible un cierto periodo de tiempo para confirmar la acción) a esa misma dirección de correo [78, 79, 81], como podemos ver en la *Figura 36*. Por ello, el tutor legal de ese usuario debe hacerse cargo del uso correcto de la dirección de email aportada para abrirse la cuenta.

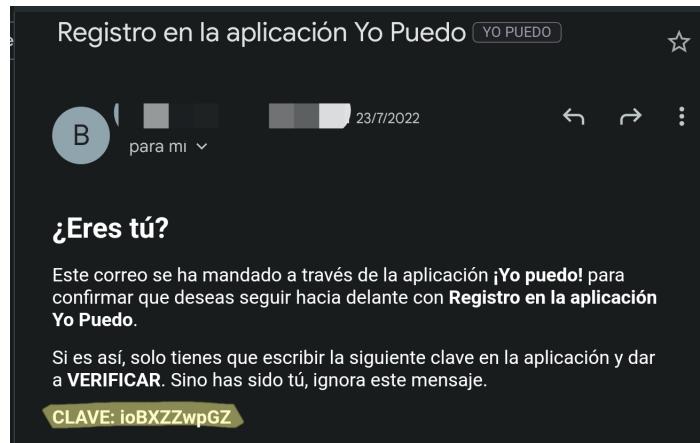


Figura 36: Ejemplo de correo electrónico con la clave aleatoria para registrarse en la aplicación.

Además, como se ha comentado anteriormente, le mandaremos un correo electrónico con una clave de 15 dígitos numéricos fija cuando se registre por primera vez en la aplicación, tal y como lo mostramos en la *Figura 37*. Dicha clave deberá guardarla a conciencia el usuario puesto que esta solamente se ofrece una única vez y esa es cuando se ha registrado en el sistema.

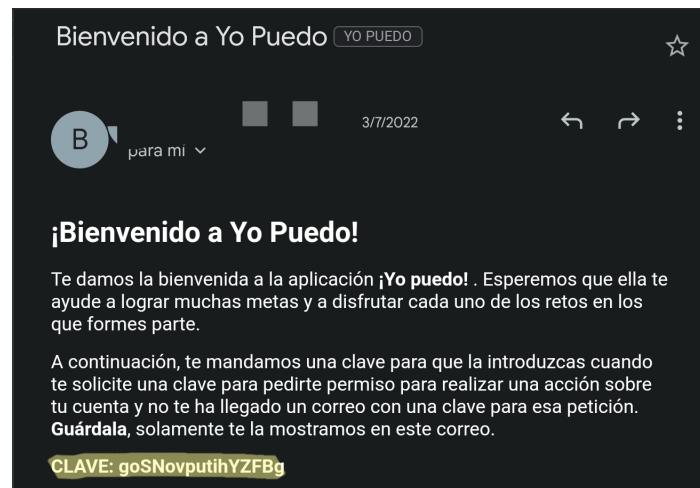


Figura 37: Ejemplo de correo electrónico con la clave fija tras el registro del usuario.

En el momento en el que se le manda la clave aleatoria, el usuario debe escribir en el formulario correspondiente (como el que se muestra en la *Figura 38*, que corresponde con la pantalla superior derecha de los bocetos de la *Figuras 25*) la clave de 10 dígitos o la fija de

15 dígitos para que el sistema tenga la confirmación del tutor para poder realizar la acción correspondiente.

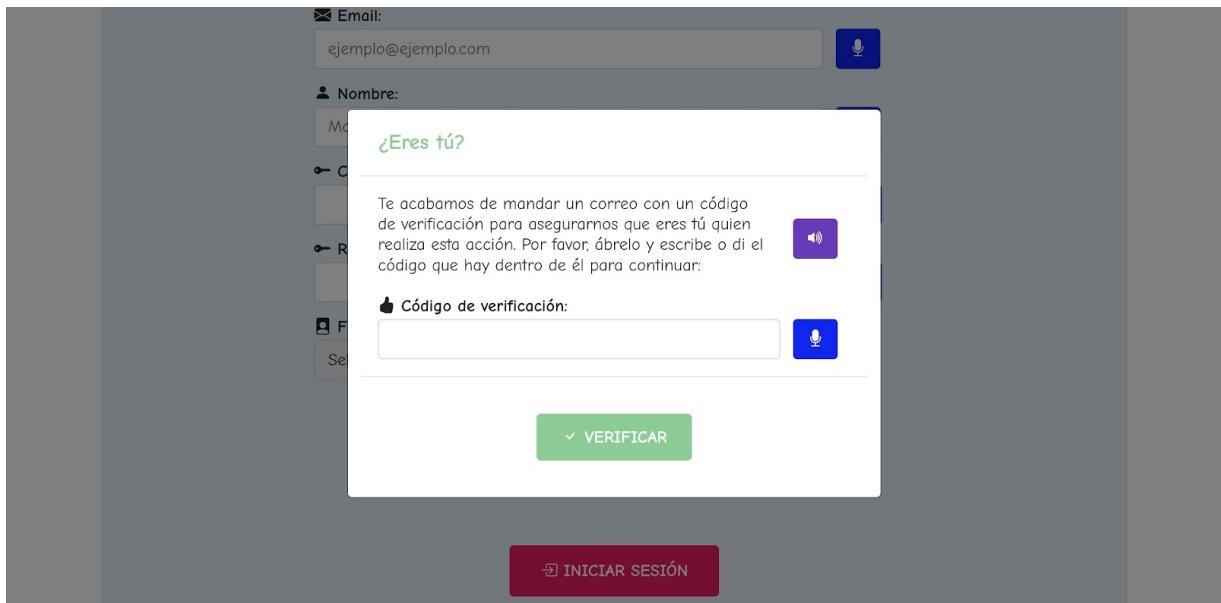


Figura 38: Ejemplo de pantalla de pin parental para el registro de un usuario en tablet.

El formulario que se ha creado para poder escribir la clave aleatoria mandada aparece en pantalla a través de un modal (creado a través de *Bootstrap* y de *HTMX*) que no se puede salir [81, 82, 83] (hasta que no escriba la clave o se equivoque 3 veces seguidas), después de registrarse el usuario y de mandar los correos electrónicos correspondientes.

Si nos fijamos en la *Figura 38*, además de pedir el código de verificación enviado al correo electrónico, hay un mensaje introductorio para que sepa lo que hay que realizar. Para aquellas personas que no pueden ver correctamente el mensaje, implementamos un altavoz (como el que señalamos en la *Figura 39*) para que haga la síntesis de voz [84] a través del frontend utilizando JavaScript.

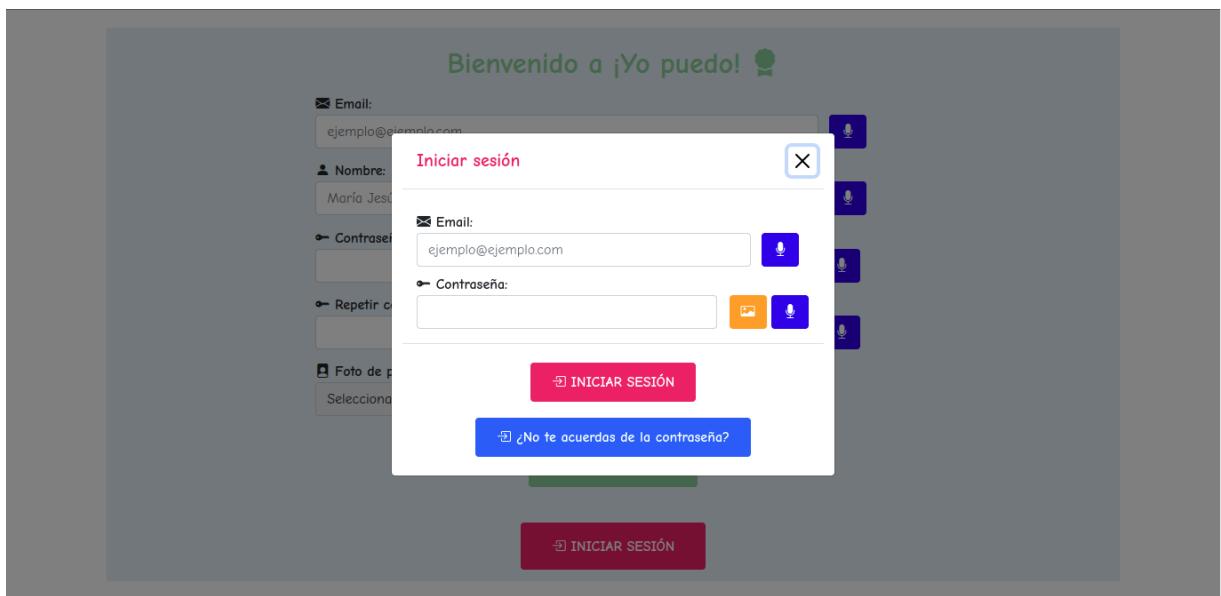
Te acabamos de mandar un correo con un código de verificación para asegurarnos que eres tú quien realiza esta acción. Por favor, ábrelo y escribe o di el código que hay dentro de él para continuar:



Figura 39: Ejemplo de síntesis de voz en la aplicación.

Inicio de sesión

Con respecto al inicio de sesión del usuario (que ya previamente se ha registrado en la aplicación), la parte frontend de la tarea la realizamos a través de un modal [81, 82, 83] parecido al del pin parental (excepto que se puede cerrar si se pulsa sobre la X que hay en la esquina superior derecha del modal) con un formulario en donde el usuario debe escribir el email y la contraseña para iniciar sesión (como los mostrados en la *Figura 40*, creados a partir de la pantalla inferior de las *Figuras 24, 25 y 26*). Este modal se abre cuando el usuario ha pinchado sobre el botón rosa de la página de registro en la *Figura 33*.



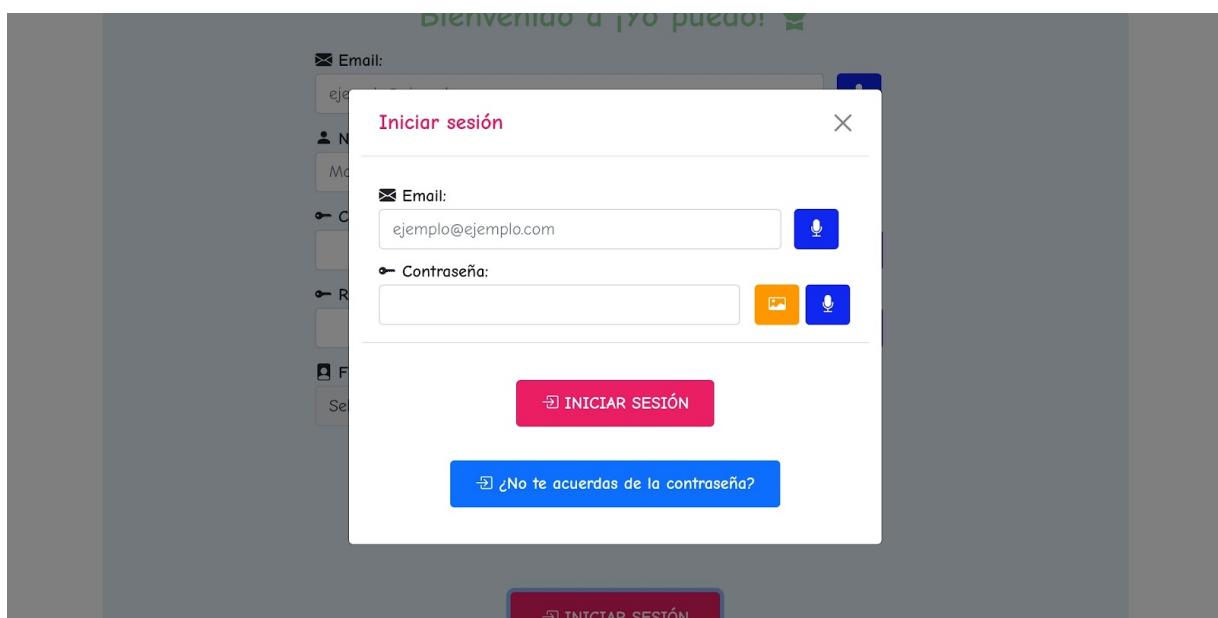
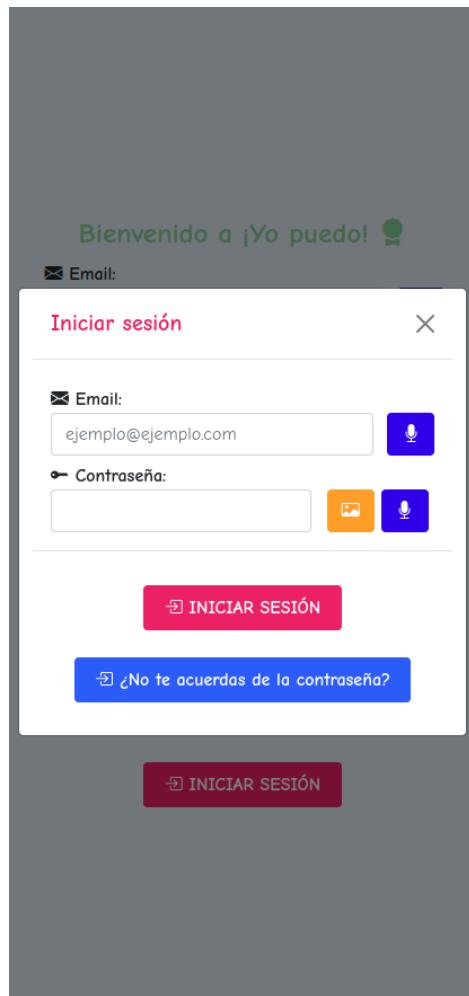


Figura 40: Implementación de inicio sesión en ordenador, móvil y tablet.

Tanto la transcripción de los campos de texto como la inserción de la contraseña a través de imágenes la realizamos de la misma manera que cuando la hicimos en la pantalla

de registro, además de utilizar las mismas herramientas para visualizar los elementos necesarios en el frontend.

En cuanto al backend de esta tarea, básicamente hace estos pasos:

1. Crea el formulario con el que se va a iniciar sesión.
2. Dependiendo del tipo de petición que desea el usuario, se hace las siguientes acciones:
 - a. Si la petición es una petición GET, únicamente mandamos el formulario vacío.
 - b. Si es una petición POST, recogemos la información del formulario y la validamos [67]. Si está todo correcto, lanzamos el pin parental. Sino, mandamos de nuevo a la misma pantalla con los mensajes de error correspondientes.

Recuperación de cuenta

Durante el diseño de esta aplicación, pudimos ver que, una vez que el usuario se haya registrado en el sistema, si desea iniciar sesión de nuevo en él pero no se acuerda de la contraseña de su cuenta, nosotros debemos de aportar un método que se encargue de ayudar a ese usuario a recuperar su cuenta.

Por lo tanto, las distintas opciones que hemos pensado para recuperar la contraseña de una cuenta en nuestro proyecto son las siguientes:

1. *Creación de una contraseña “falsa”* [85, 86]: Cuando el usuario solicite la recuperación de su contraseña, escribe el correo electrónico de la cuenta que quiere recuperar, el sistema genera una contraseña aleatoria y se la envía directamente al correo electrónico de su cuenta para que el usuario pueda entrar en su cuenta y directamente escribir la nueva contraseña.

Si pensamos en las ventajas que tiene este método, es que este cambio es instantáneo y se puede generar una contraseña muy segura para que no estén hackeando la cuenta mientras el usuario está intentando recuperarla. Los problemas que se podrían observar a esta solución son que se manda directamente la contraseña sin cifrar al usuario, que no todas las personas podrían introducir con soltura la contraseña aleatoria generada, que se perdería la contraseña original y que se podría corromper la base de datos de la aplicación al manipularse el archivo donde se almacenan estos datos.

2. *Email con los pasos para recuperar la cuenta* [87, 88]: Cuando el usuario indique que no se acuerda de su contraseña, debe escribir o dictar su dirección de correo electrónico y el sistema verificará que ese correo está dentro de su base de datos. Si está dentro del sistema, envía un correo a esa dirección con los pasos que debe seguir el usuario para restablecer la contraseña (llevarlo a una nueva página donde escriba la nueva contraseña e iniciar sesión después de verificar que la contraseña cumple con los requisitos). Además, este correo solo tendrá un tiempo limitado de validez para efectuar el restablecimiento de la cuenta.

En este caso, podemos ver que las ventajas que puede ofrecer esta opción es que la recuperación de la cuenta es rápida y sencilla, y que no se envía información sensible a través del correo electrónico. Sin embargo, también podemos considerar las desventajas que tiene, que son que el usuario no tenga a mano el correo electrónico y que puede que éste escriba otra dirección de correo electrónico distinta a la de la cuenta por lo que si ese correo no existe dentro de la base de datos de la aplicación deberíamos informar de esta situación al usuario. Pero si decidíramos avisar al usuario de la inexistencia de esa dirección, podríamos estar exponiendo información confidencial a un extraño que quisiera acceder a nuestra cuenta.

3. *Iniciar sesión con alguna cuenta de otras redes sociales* [87]: Permitir que un usuario se registre en la aplicación utilizando una cuenta que tenga con otras redes sociales, como Gmail, Facebook o Twitter.

Para esta solución podemos destacar como ventajas frente a las anteriores opciones que es muy sencilla y que no hace falta que introduzca ni se guarde ningún tipo de contraseña, puesto la red social elegida será la que se encargue de certificar la cuenta . Por lo tanto, no haría falta por nuestra parte generar ningún protocolo de recuperación de cuentas. Aunque como desventajas se podrían señalar que si borra la cuenta de esa red social, ya no podría acceder a la nuestra, puesto que no podemos confirmar su existencia en el sistema de la otra red social. También podría ocurrir que el usuario no tenga cuenta previa en estas redes sociales, con lo que este sistema no serviría. Además, cualquier persona tutelada si tiene acceso a dichas redes sociales se saltaría el pin parental por lo que no tendrían un control adecuado sobre el uso de nuestra aplicación.

4. *Preguntas secretas* [88]: Antes de registrarse el usuario en la aplicación, le hacemos elegir una de una serie de preguntas que él debe responder con algo que él únicamente conozca, como por ejemplo, el lugar de nacimiento de su madre o el

nombre de su mascota. Una vez registrado, el sistema cifra la respuesta y la guarda en la base de datos.

Como ventajas de esta solución podríamos comentar que es sencilla de implementar por parte del desarrollador, es fácil de recordar la respuesta por parte del usuario y que es instantánea la recuperación de la cuenta. Aunque como desventajas podríamos decir que el usuario puede olvidar las respuestas que dió en su momento y que una persona que conozca lo suficiente al usuario, sabría sus respuestas, y por tanto podría acceder al sistema sin ninguna complicación (por lo que esta solución es una de las más inseguras de implementar), ya que normalmente no hay una gran variedad de respuestas diferentes.

Si decidimos complicar un poco más las preguntas para que sean más difíciles de averiguar sus respuestas, puede que el usuario le cueste más todavía recordar o encontrar la respuesta correcta.

5. *Vía SMS* [88]: El sistema envía el código SMS al teléfono móvil que tiene guardado la cuenta que se desea recuperar.

Las ventajas de esta solución son que es una de las opciones más seguras y que el cambio se haría instantáneamente, aunque como inconvenientes podríamos comentar que es difícil implementarla en el sistema, que hay que registrarse en un servicio de pago de un tercero para el envío de SMSs y que hay que pedirle al usuario su número de teléfono para realizar el envío de mensajes.

En la siguiente tabla, mostramos un pequeño resumen de las opciones comentadas en los párrafos anteriores sobre cómo podríamos implementar la recuperación de contraseñas:

Método	Ventajas	Desventajas
<i>Creación de una contraseña “falsa”</i> [85, 86]	<ul style="list-style-type: none">• Cambio instantáneo• Contraseña temporal segura	<ul style="list-style-type: none">• Exposición de la contraseña• Difícil de insertarla para ciertos usuarios• Perder la contraseña original• Corromper la B.D.
<i>Email con los pasos para recuperar la cuenta</i> [87, 88]	<ul style="list-style-type: none">• Rápida• Sencilla• No se envía	<ul style="list-style-type: none">• No tener el correo a mano• Escriba el correo incorrecto• Avisar al usuario de si se ha

	información sensible	enviado o no el email
<i>Iniciar sesión con alguna cuenta de otras redes sociales [87]</i>	<ul style="list-style-type: none"> • Rápida • Sencilla • No nos encargamos de la contraseña 	<ul style="list-style-type: none"> • Dependencia de otra red social • Difícil implementación del pin parental
<i>Preguntas secretas [88]</i>	<ul style="list-style-type: none"> • Sencilla • Fácil de recordar • Cambio instantáneo 	<ul style="list-style-type: none"> • Fácil de averiguar la respuesta • No hay gran variedad de respuestas distintas • Si complicamos las preguntas, las respuestas son más difíciles de recordar o encontrar
<i>Vía SMS [88]</i>	<ul style="list-style-type: none"> • Segura • Instantánea 	<ul style="list-style-type: none"> • Difícil de implementar • Compra de servicio de terceros para el envío de SMSs • Pedir el número de teléfono en el registro

Tabla 11: Resumen de los métodos pensados para la recuperación de contraseña.

Al ver los posibles métodos que podríamos implementar dentro de nuestra aplicación para recuperar la contraseña, hemos decidido que la opción más apropiada para este proyecto es el envío de los pasos que debe seguir el usuario para restablecer la cuenta por las siguientes razones:

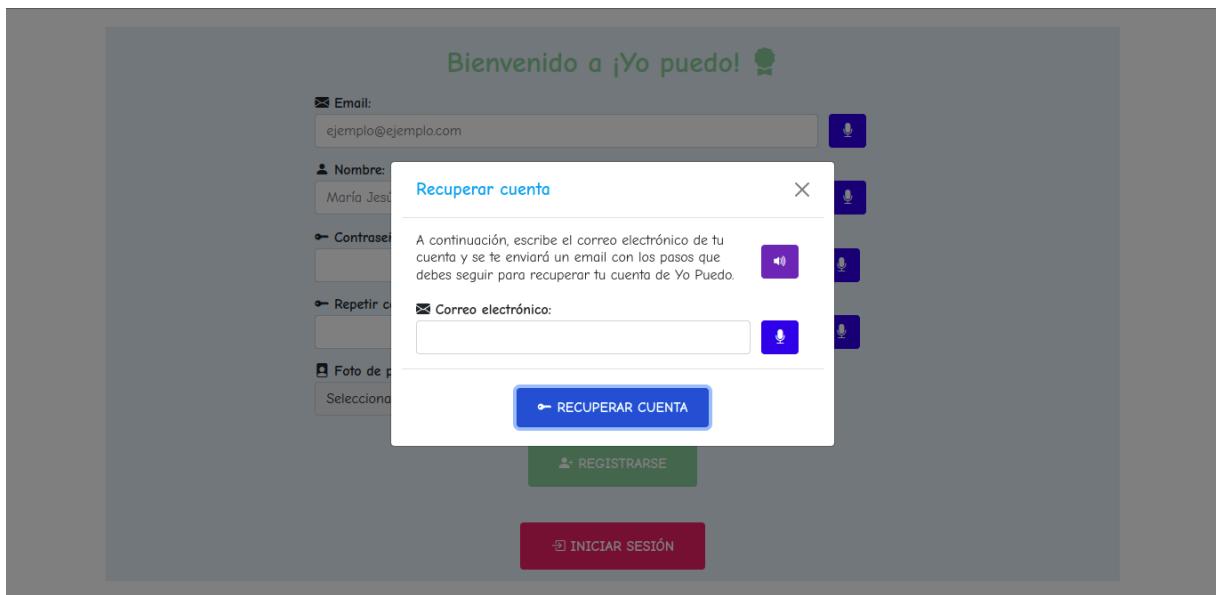
- Como ya disponemos del correo electrónico del usuario para su identificación dentro del sistema, lo podemos utilizar también para la recuperación de su cuenta en el momento en el que este no sepa su contraseña.
- Evitamos que los usuarios tutelados cambien la contraseña sin la supervisión de sus tutores, por lo que seguimos manteniendo en cierta manera el pin parental.
- Es una de las opciones más seguras puesto que no se manda a través del correo ninguna información sensible y, además, hacemos que el enlace a dicha página sea accesible durante un cierto tiempo.
- A través del email a esa dirección, comprobamos que el usuario que ha solicitado la recuperación de la cuenta es el propietario de dicha cuenta.

- Es sencillo para el usuario poder restablecer la contraseña puesto que únicamente es necesario pinchar en el enlace enviado y luego escribir y confirmar la nueva contraseña.
- El framework seleccionado tiene implementado este tipo de tecnología para la recuperación de una cuenta [78].

Únicamente creamos las plantillas para el envío de correos electrónicos (por ejemplo, el de la *Figura 41*) y para la página en donde debe escribir, en primer lugar, el correo de la cuenta a recuperar (los que mostramos en la *Figura 42*) y, por último, donde escribe la nueva contraseña (como las capturas de pantalla de la *Figura 43*).



Figura 41: Ejemplo de correo electrónico para recuperar la cuenta.



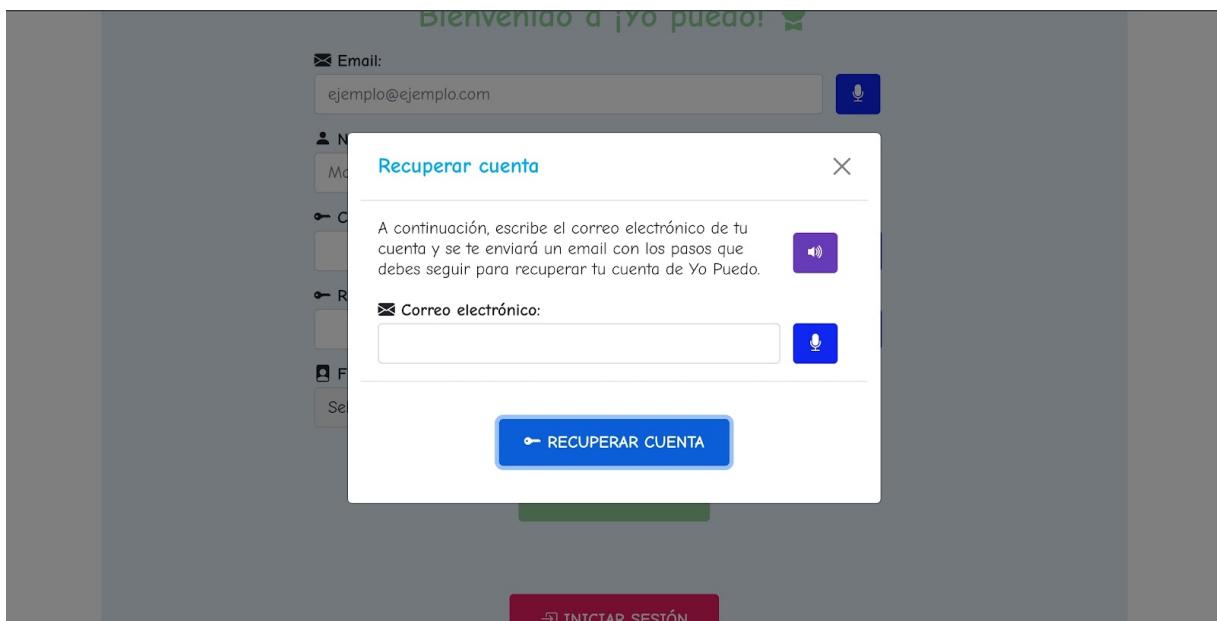
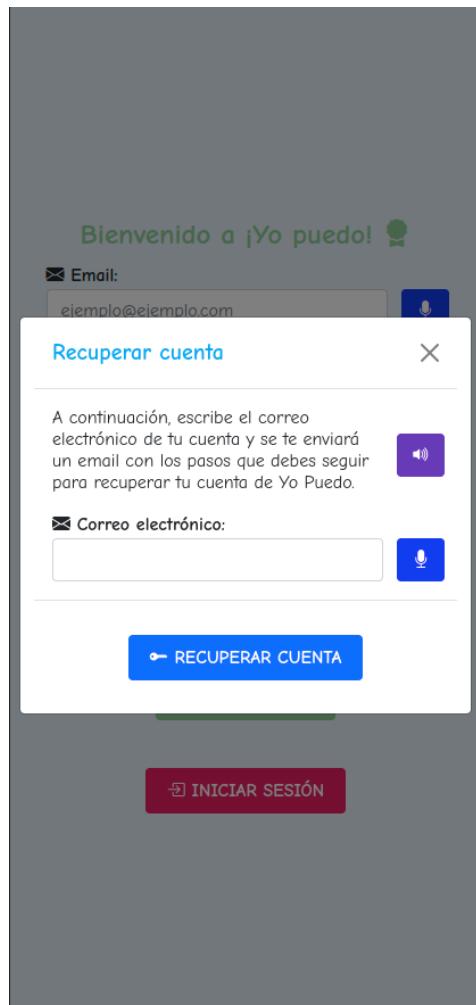


Figura 42: Petición de recuperación de cuenta para ordenador, móvil y tablet.

Una vez recibido el correo de la Figura 41 y pinchado sobre el correspondiente enlace (el que pone *Recuperar cuenta*), nos dirige a la página que vemos en la Figura 43 donde el

usuario escribirá la nueva contraseña y su repetición (correspondiente al segundo paso de las *Figuras 27, 28 y 29*).

The screenshot shows a light blue rectangular form titled 'Recuperar cuenta de ¡Yo puedo!'. It contains two input fields: 'Contraseña nueva:' and 'Contraseña nueva (confirmación:)'. Each field has a placeholder text 'Contraseña nueva:' and two small orange and blue icons. Below the fields is a green button labeled 'CAMBIAR CONTRASEÑA'.

Figura 43: Formulario de recuperación de contraseña.

Cuando el usuario escribe la nueva contraseña y esta tiene las características necesarias para aceptarlas, muestra al usuario un mensaje por pantalla indicando que todo ha ido correctamente, como el de la *Figura 44*.

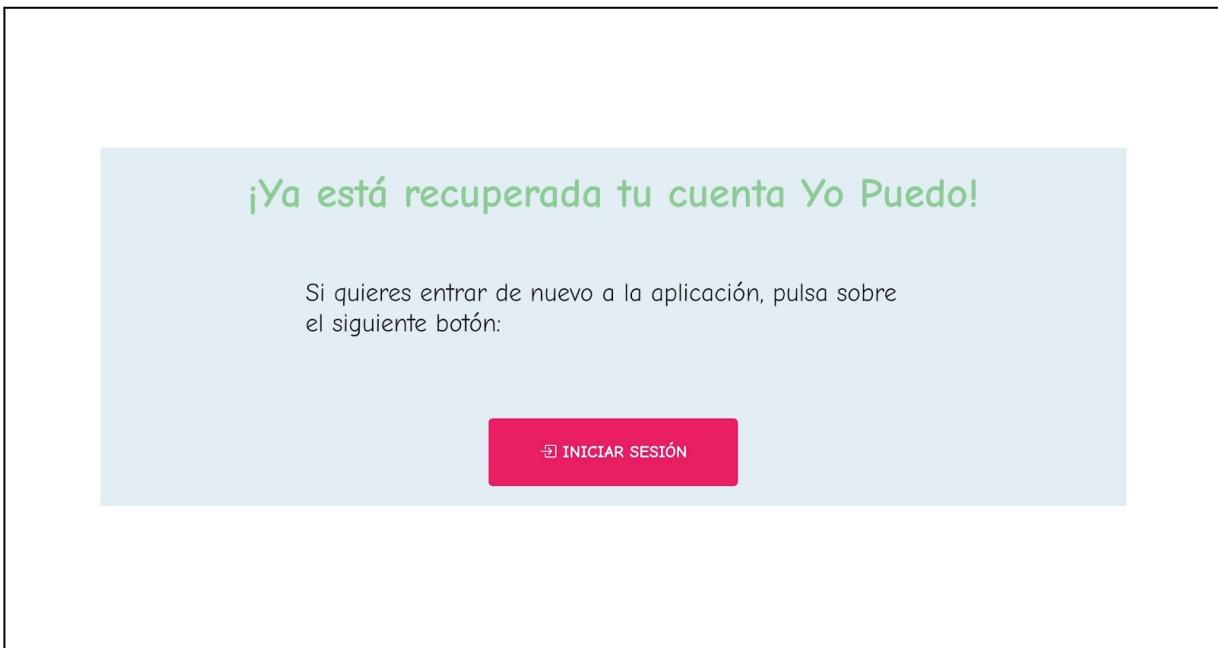


Figura 44: Mensaje de recuperación satisfactoria.

Después de que el usuario escribe el email de la cuenta que quiere recuperar (en el formulario mostrado en las tres capturas de la *Figura 42*), una vez enviado el mensaje a su correo electrónico, avisamos al usuario mediante un mensaje por pantalla, como el mostrado en la *Figura 45*, en el que, además de poder leerlo, el usuario puede activar la síntesis de voz para que lo lea por él en voz alta [84].

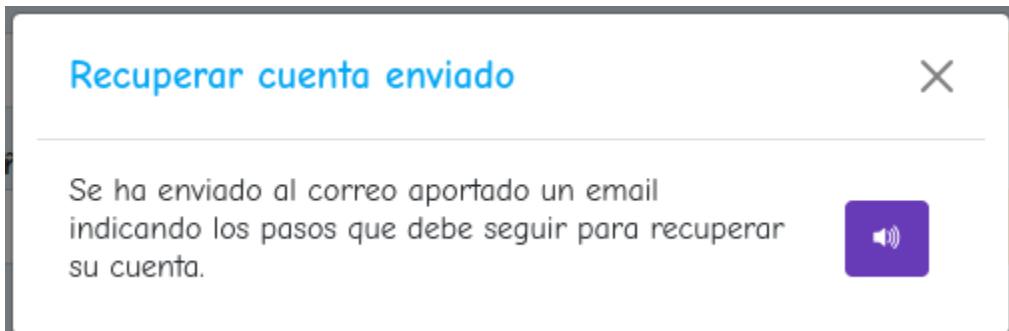


Figura 45: Confirmación de envío de email con las instrucciones para recuperar la cuenta.

Pruebas

En cuanto a las pruebas realizadas para este sprint, las vamos a dividir en las efectuadas sobre los bocetos y en las efectuadas sobre la implementación.

Con respecto a las realizadas sobre los bocetos, hemos generalizado distintas pruebas de accesibilidad sobre los distintos bocetos para verificar que cumplen con las normas implementadas en las distintas [guías de accesibilidad](#) comentadas en el primer capítulo de este documento. Es por eso la razón por la que en el apartado de [bocetos](#) de este primer sprint hemos presentado distintas versiones de las tareas realizadas.

En cuanto a las pruebas ejecutadas sobre la implementación de estas tareas, hemos hecho las siguientes comprobaciones:

- Si creamos un objeto con los datos de un posible usuario a través del framework, se guarda correctamente en la base de datos y son accesibles.
- Cuando registramos un usuario, miramos que:
 - El formulario creado y, si se produce un error, que devuelva el mensaje esperado de las siguientes validaciones:
 - El email dado debe tener la estructura de un correo electrónico.
 - El email dado no exista previamente en la base de datos de la aplicación.

- La contraseña debe tener una longitud entre 8 y 16 caracteres.
 - La contraseña debe tener un número, una minúscula, una mayúscula y un símbolo como mínimo.
 - La contraseña repetida y la original sean iguales.
 - La foto de perfil dada sea una imagen.
- La respuesta del backend cuando:
 - No se introducen los datos correctos.
 - Se guarda el usuario correctamente.
- Cuando iniciamos sesión, observamos que:
 - El formulario creado realiza correctamente las validaciones:
 - El email dado debe existir.
 - La contraseña dada sea la guardada con el email correspondiente.
 - La respuesta del backend cuando:
 - Se introduce un email que no existe.
 - Se escribe una contraseña incorrecta.
 - Los datos introducidos son los correctos.
- Cuando se activa el pin parental, verificamos que:
 - Recibe un correo electrónico con la clave aleatoria (otro con la clave fija si se ha solicitado el registro en la aplicación)
 - Da permiso para acceder con la clave aleatoria.
 - Realiza la acción pedida con la clave fija.
 - Manda mensaje de error cuando la clave introducida no es la misma que la clave aleatoria ni que la fija.
 - Cuando falla 3 veces y la petición solicitada es registrarse, elimina la cuenta de usuario de la base de datos.
- Cuando solicitamos la recuperación de cuenta, comprobamos que:

- El email introducido existe.
- Recibe el correo electrónico con los pasos de recuperación.
- La contraseña debe tener una longitud entre 8 y 16 caracteres.
- La contraseña debe tener un número, una minúscula, una mayúscula y un símbolo como mínimo.
- La contraseña repetida y la original sean iguales.

Retrospectiva

En conclusión, a lo largo de esta iteración hemos podido llevar a cabo todas las tareas planificadas, aunque hemos tenido que alargar un poco la duración de este sprint puesto que algunas de ellas hemos tardado más de lo pensado, como el caso de registrarse en la aplicación (por problemas surgidos a la hora de crear la tabla personalizada para almacenar la información del usuario dentro de *Django*, las validaciones de las contraseñas y la subida de la foto de perfil al servidor) y el pin parental (al tener problemas a la hora de la aparición del modal para poder escribir la clave recibida).

3.8. Segundo sprint

En cuanto al segundo conjunto de tareas a realizar, nos vamos a centrar en las funcionalidades relacionadas con la cuenta del usuario, como el cierre de sesión o la modificación de los datos. Estas tareas son:

Historias de Usuario

- **[HU4] Como persona, necesito cerrar mi sesión.**
 - Quién: Persona.
 - Cuándo: Cuando quiera dejar de acceder al sistema con un dispositivo sin necesidad de eliminar su cuenta.
 - Entonces: Se eliminan los datos de la sesión y debe volver a iniciar sesión para acceder a los servicios del sistema.
 - CoS:
 - Se eliminarán los datos necesarios que hacen que mantengan la sesión abierta.

- Se retornará a la página de registro del sistema, por si desea entrar de nuevo, debe escribir los datos necesarios para volver a iniciar sesión.
- [HU5] **Como persona, necesito ver mi perfil.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver sus datos personales.
 - Entonces: Se muestran los datos guardados en el sistema de ese usuario.
 - CoS:
 - Se mostrarán los datos de la persona del identificador dado: nombre, email, foto de perfil.
 - Únicamente se accederá a estos datos cuando la persona haya iniciado una sesión en el sistema.
- [HU6] **Como persona, deseo cambiar mi perfil.**
 - Quién: Persona.
 - Cuándo: Cuando quiera modificar sus datos personales.
 - Entonces: Se muestran los datos personales guardados y se guardan los cambios realizados.
 - CoS:
 - Solo podrán hacerlo aquellos usuarios que estén registrados y hayan iniciado sesión en el sistema.
 - Únicamente podrá modificar los siguientes datos de la cuenta: nombre, contraseña y foto de perfil.
 - Si la contraseña introducida no cumple con los requisitos (debe contener números, letras y símbolos, además de tener una longitud de entre 8 y 16 caracteres), se devuelve un error.
 - Se puede escribir la contraseña o elegir aquellas imágenes asociadas a unas palabras según los requerimientos que debe tener la contraseña.

- Se debe introducir una foto en el perfil. Si no, se mostrará un mensaje de error indicando que es obligatorio.
 - Para aquellos campos textuales (nombre y contraseña), se permiten ser escritos o dictados por el usuario.
 - Cada campo del formulario que el usuario debe llenar debe tener un título y un ícono asociado a ese campo.
 - Si todos los datos mandados son los correctos y la clave mandada es la generada por el sistema, se guarda en la base de datos la siguiente información:
 - El nombre.
 - La contraseña (previamente cifrada).
 - La ubicación local de la foto de perfil proporcionada.
 - Antes de guardar la nueva foto de perfil, eliminamos la foto de perfil anterior del servidor.
- **[HU7] Como persona, deseo cancelar mi cuenta.**
 - Quién: Persona.
 - Cuándo: Cuando quiera restringir los servicios del sistema.
 - Entonces: La persona se eliminará del sistema y ya no podrá utilizar de nuevo los servicios de la aplicación con esa cuenta.
 - CoS:
 - Deberá estar la persona que desea realizar esta acción registrada e iniciar sesión en el sistema.
 - Se envía correo electrónico con una clave aleatoria de 10 dígitos, generada por el sistema, y se guarda en la base de datos para confirmar el consentimiento por parte del usuario de querer eliminar la cuenta.
 - Cada campo del formulario que el usuario debe llenar debe tener un título y un ícono asociado a ese campo.

- El usuario debe escribir la clave aleatoria temporal o la fija para confirmar el permiso de eliminación de cuenta en el sistema.
- Si el usuario falla 3 veces en escribir la clave de confirmación, no se elimina la cuenta y se le obliga a empezar de 0.
- Si la clave mandada es la generada por el sistema, se realiza lo siguiente (cuando las funcionalidades comentadas estén implementadas, que se harán en los siguientes sprints de desarrollo):
 - Se eliminan todos los retos individuales de esa persona y sus archivos generados.
 - Se elimina el usuario de los retos en los que esté como animador o participante.
 - Se eliminan los mensajes de apoyo dados por esa persona y sus archivos generados.
 - Se eliminan las pruebas añadidas en los retos y sus archivos generados.
 - Se eliminan las calificaciones de los retos en los que participa el usuario.
 - Se eliminan los retos colectivos en los que sea coordinador y los archivos generados.
 - Se eliminan las amistades que tendría ese usuario con otros.
 - Se elimina la foto de perfil almacenada en el servidor.

Bocetos

Cuando estuvimos diseñando la creación y el inicio de sesión del perfil de un cliente, pensamos que deberíamos añadir una pantalla donde pudiera ver sus datos personales (aquellos que se pueden mostrar públicamente), borrar definitivamente su cuenta cuando pulse el botón correspondiente y, además, escriba la clave que confirme que desea realizarlo, y otra donde pueda editar algunos de esos datos (como la foto de perfil o la contraseña) cuando crea necesario. Esas pantallas podemos verlas en distintos formatos de dispositivos en las siguientes figuras:

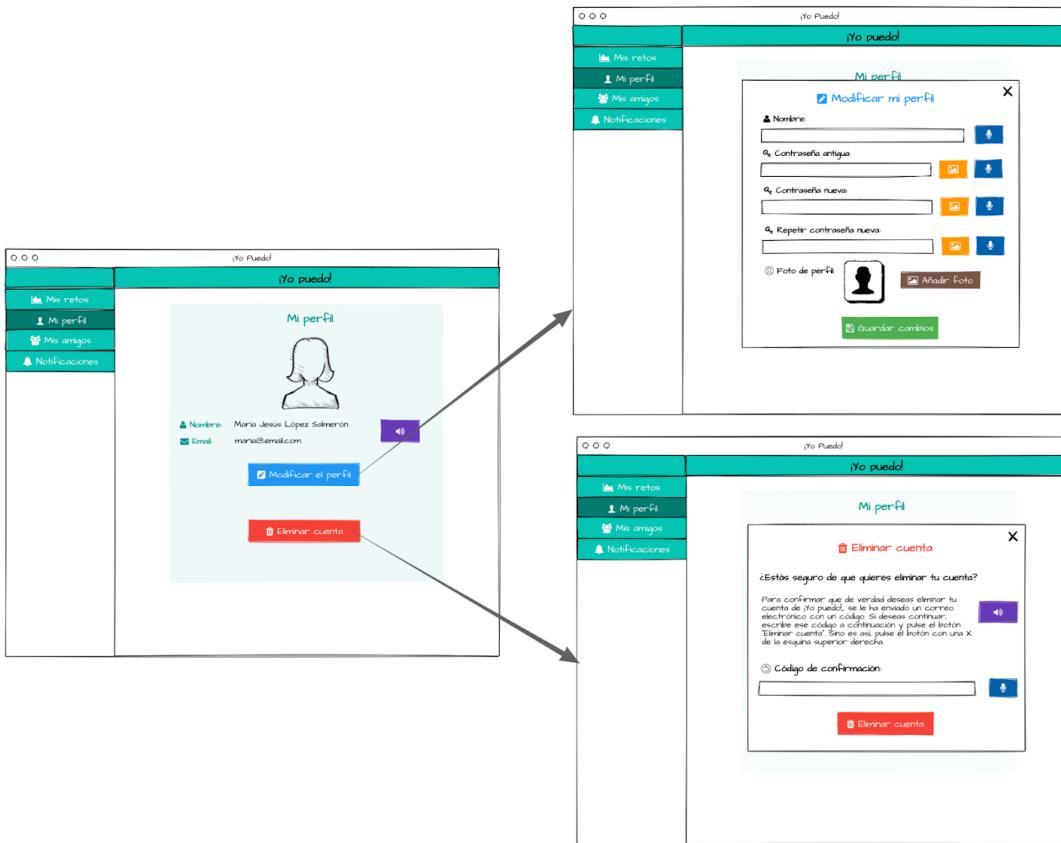


Figura 46: Primera versión del perfil de un usuario en formato escritorio.

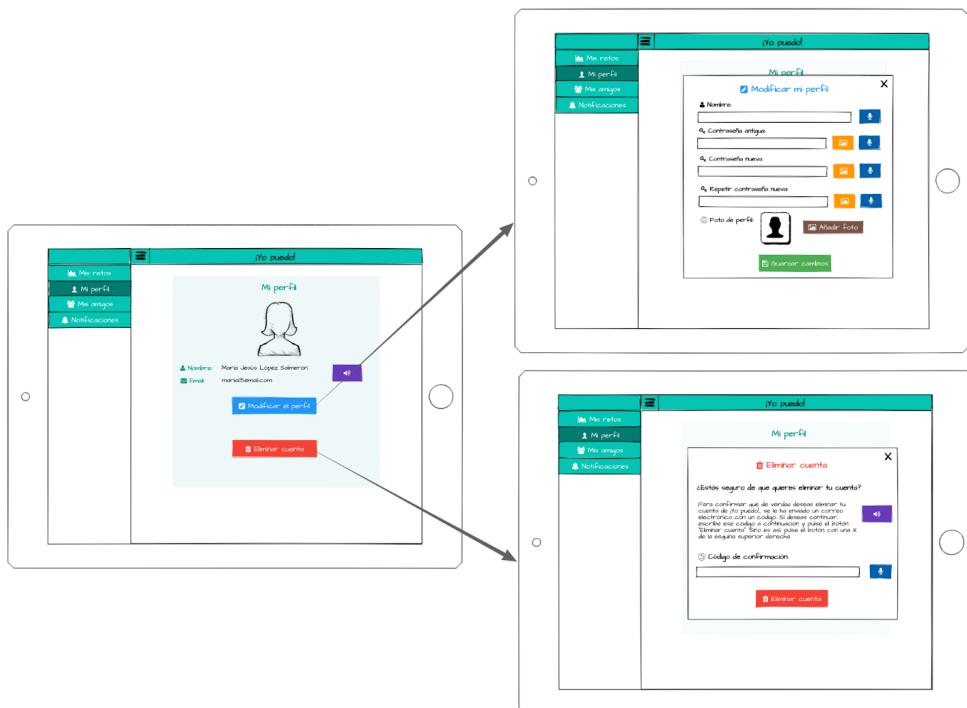


Figura 47: Primera versión del perfil de usuario en tablet.

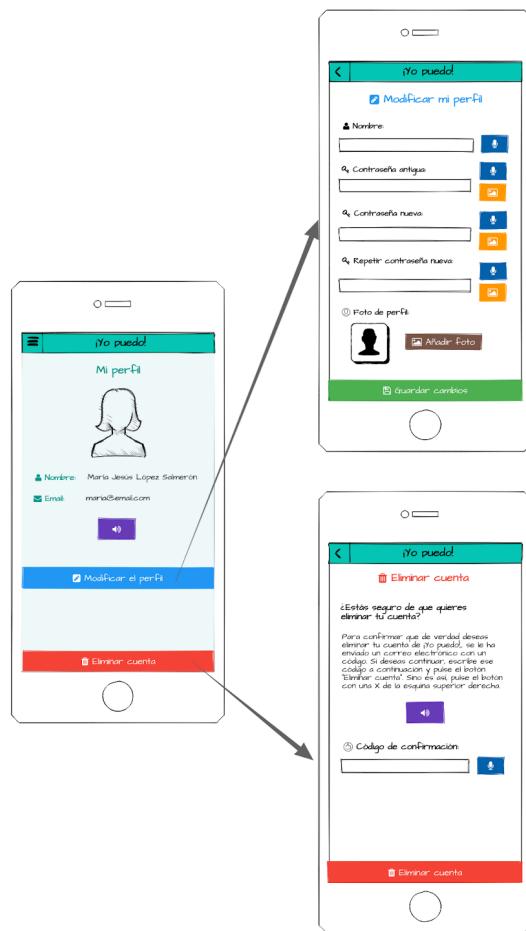


Figura 48: Primera versión del perfil del usuario para móviles.

En la primera pantalla que mostramos al usuario de las Figuras 46, 47 y 48, cuando pincha en “Mi perfil” dentro del menú de la aplicación, visualizamos el nombre y el correo con el que está registrado, además de su foto de perfil.

Como cuando damos a iniciar sesión o a añadir participante o animador, aparece una ventana pequeña encima de la principal con la pantalla encargada de esas funcionalidades para ordenadores y tablets, en móvil se mostraba dicha ventana en pantalla completa para que se vean todos los elementos y se puedan pulsarlos sin ningún problema. Pues en el momento en el que queramos modificar un perfil o eliminar la cuenta sucede de igual manera.

Con respecto a la pantalla de edición del perfil, únicamente permitiremos al usuario modificar su nombre, la contraseña y su foto de perfil. Cuando se pulse sobre “Guardar cambios”, reflejaremos los nuevos cambios archivados.

En el caso de eliminar el perfil, se muestra una ventana donde el usuario debe escribir el código aleatorio enviado al correo electrónico para confirmar que es el verdadero propietario de la cuenta el que desea eliminarlo es el verdadero propietario de la cuenta.

El único inconveniente que vimos durante una de las reuniones tenidas con el cliente es que el usuario no tiene la posibilidad de cerrar sesión de su cuenta en ninguna parte de la aplicación, por lo que hemos imaginado que el mejor lugar donde podríamos colocar dicha funcionalidad es en la visualización del perfil del usuario. Si esta opción estuviera siempre disponible, el usuario podría pulsarla por error, de esta forma está más oculta pero accesible.

Por lo tanto, en la última versión de la visualización del perfil de un usuario (encontrado en las Figuras 49, 50 y 51 para ordenador, tablet y móvil correspondientemente), lo único que hemos incorporado nuevo a los bocetos es la posibilidad de salir de la aplicación sin necesidad de borrar la cuenta por completo del sistema, es decir, de poder cerrar la sesión dentro del dispositivo en el que se esté utilizando mediante el botón naranja dentro de los datos principales del usuario.

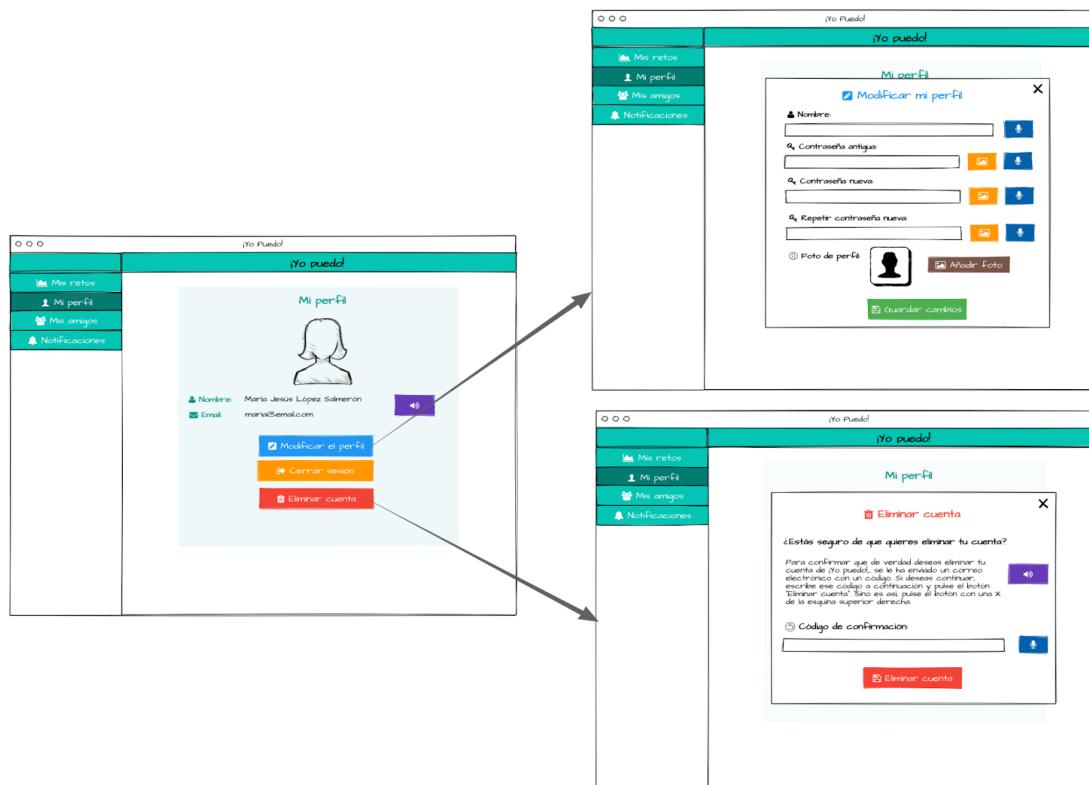


Figura 49: Última versión de “Mi perfil” para escritorio.

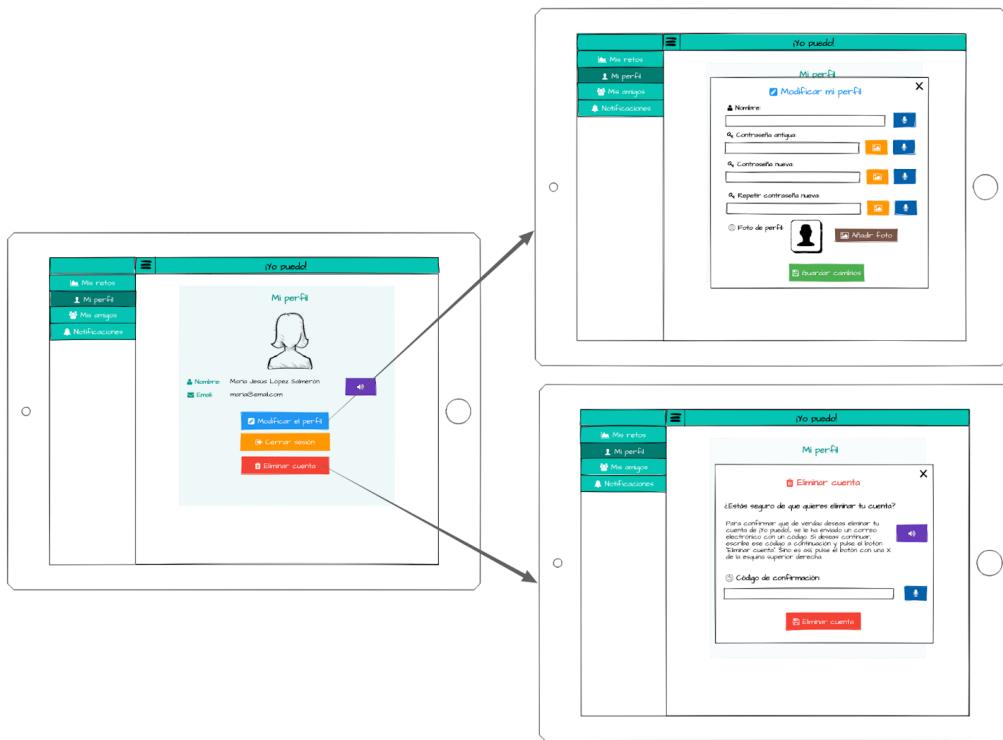


Figura 50: Última versión de “Mi perfil” para tablets

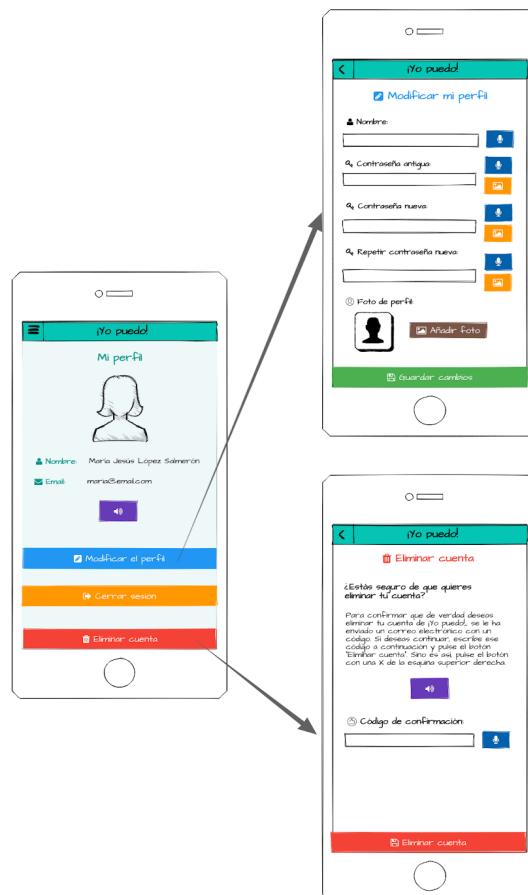


Figura 51: Última versión de “Mi perfil” en móviles.

Base de datos

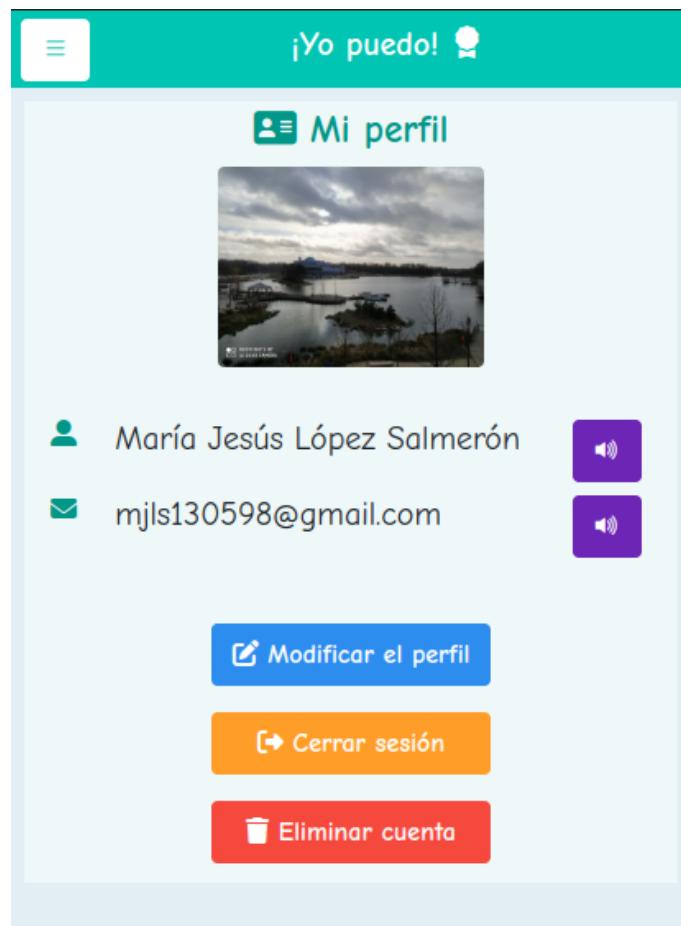
El diagrama de Entidad/Relación que debemos generar para poder implementar las historias de usuario de la 4 a la 7 sigue siendo el mismo que el primer sprint, mostrada en la *Figura 30*, puesto que esas historias únicamente se centran en las funcionalidades sobre las cuentas de usuarios.

Implementación

La implementación realizada durante el segundo sprint de este proyecto la vamos a comentar en cuatro secciones distintas, una por cada historia de usuario que decidimos llevar a cabo en esta iteración.

Perfil del usuario

Con respecto a la visualización del perfil de usuario (mostradas en distintos dispositivos en la *Figura 52*), únicamente lo que realizamos dentro del backend es recuperar los datos del usuario que ha iniciado sesión en la aplicación previamente [89] y añadirlo al template correspondiente.



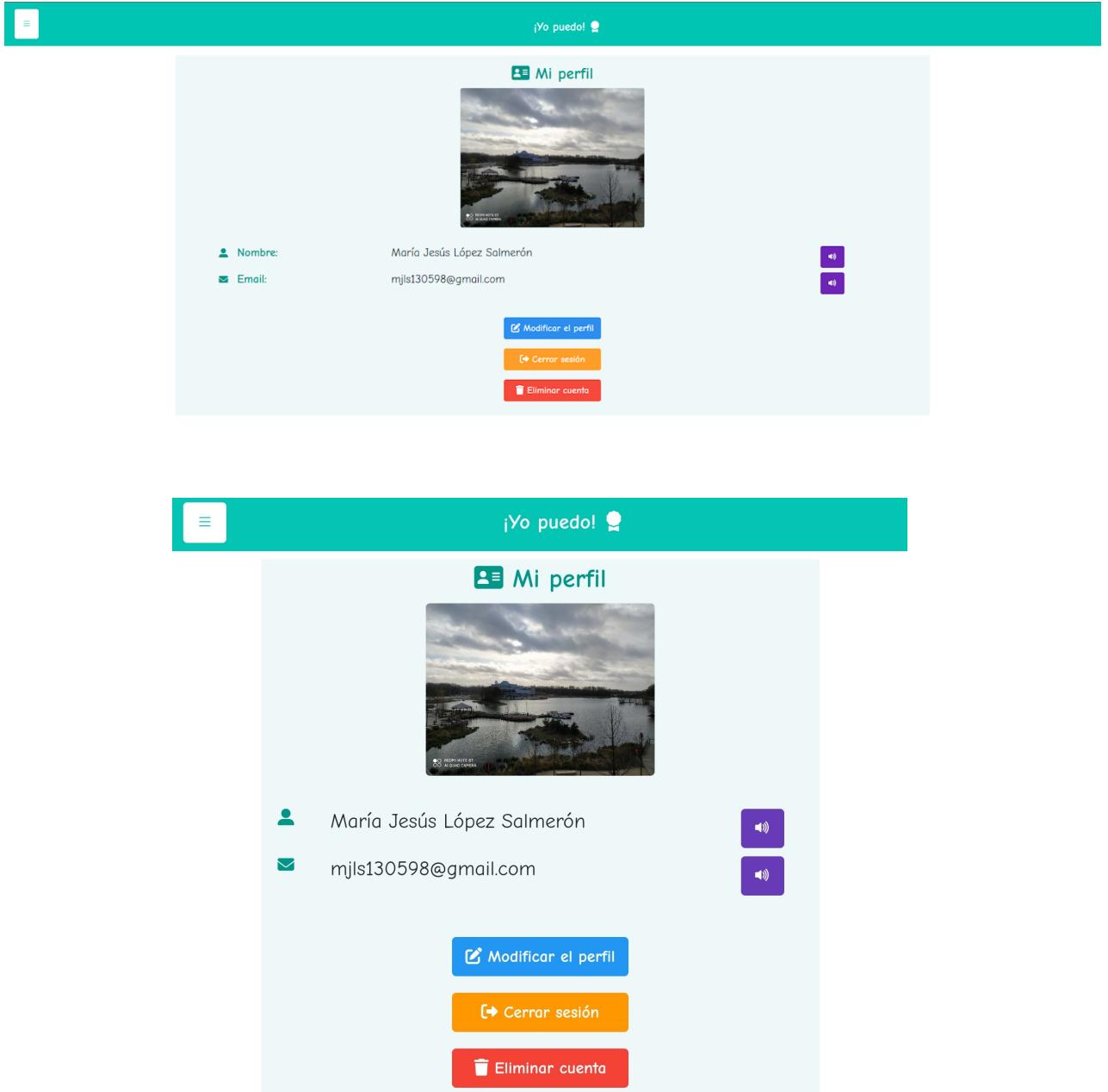


Figura 52: Capturas de pantalla para móvil, ordenador y tablet de la visualización de perfil de un usuario.

En cuanto al frontend, lo único que hemos hecho es realizar ciertos retoques sobre los distintos elementos de la página, dependiendo del tamaño de la pantalla, ya planteados en las *Figuras 49, 50 y 51*:

- Disminuir el tamaño de la imagen con respecto a la anchura del dispositivo [90, 91].
- Para aquellas pantallas con una anchura pequeña, únicamente mostrar los iconos de los campos de texto mostrados debajo de la foto de perfil [92].

Cierre de sesión

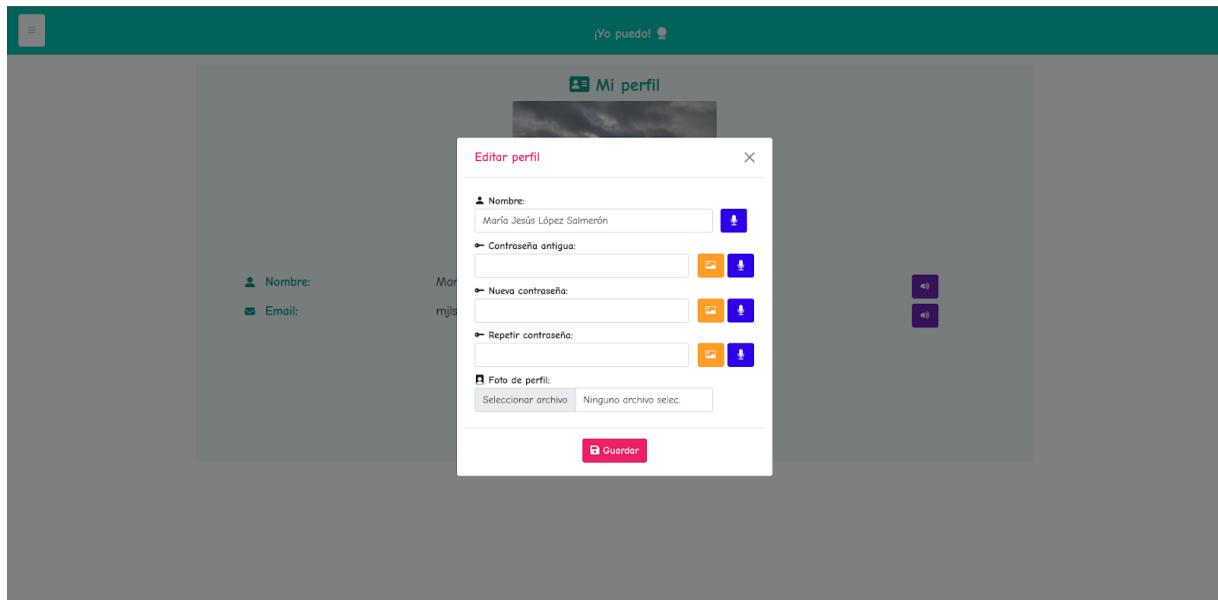
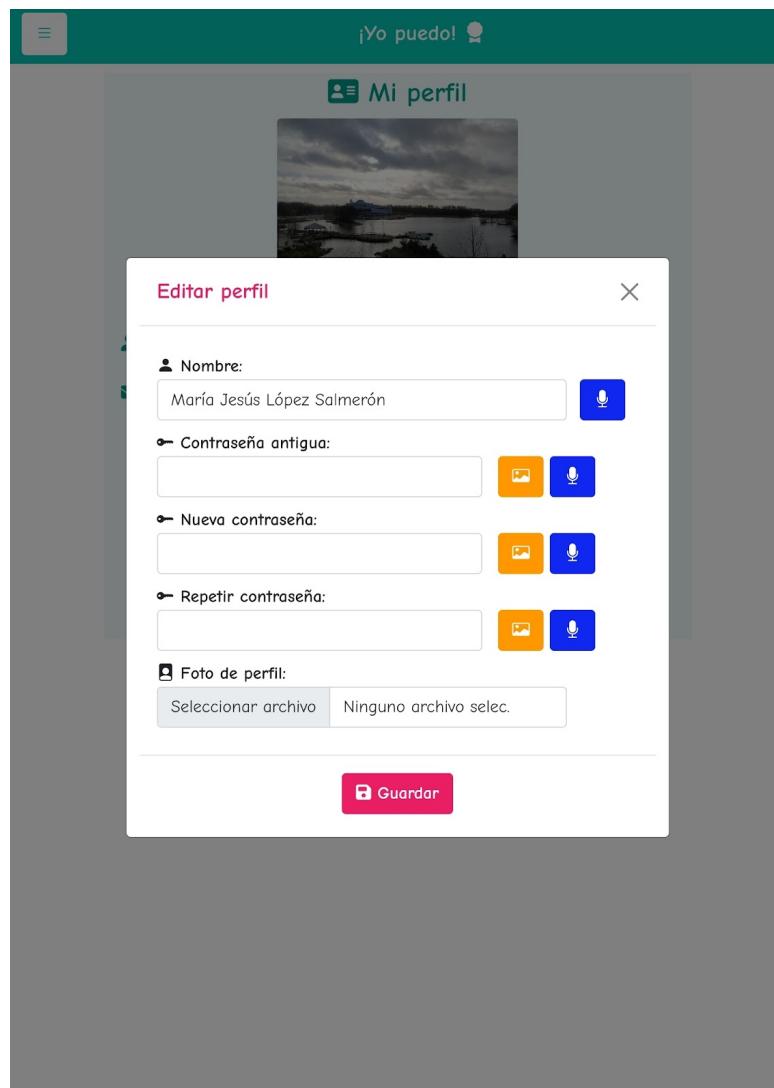
La implementación del cierre de sesión fue sencilla y fácil de realizar. En la parte de frontend solamente incluimos un botón en la visualización del perfil del usuario (los botones naranjas de la *Figura 52*) que únicamente llama al backend a través del atributo de *HTMX* para que elimine la sesión, de un usuario que previamente ha iniciado sesión, en ese dispositivo y redirija a la página donde cualquier persona puede registrarse o iniciar sesión en la aplicación.

Edición del perfil

En cuanto a la implementación de la edición del perfil de un usuario, la hicimos de la siguiente manera:

Primero, creamos un botón dentro del perfil del usuario (los botones azules de la *Figura 52*) para que se lance esta funcionalidad.

Una vez lanzada, dentro del backend se recoge la información del usuario y aparece un modal mediante *HTMX* [81, 82, 83], que únicamente se puede cerrar a través del botón de la esquina superior derecha X, con el formulario relleno con los datos permitidos parecido al de registrarse (excepto que no puede modificar su dirección de correo electrónico y, antes de escribir la contraseña nueva, debe escribir la contraseña antigua). Este formulario lo podemos ver en las pantallas de los diferentes dispositivos en la *Figura 53*, que se basaron en la pantalla superior derecha de las *Figuras 49, 50 y 51*.



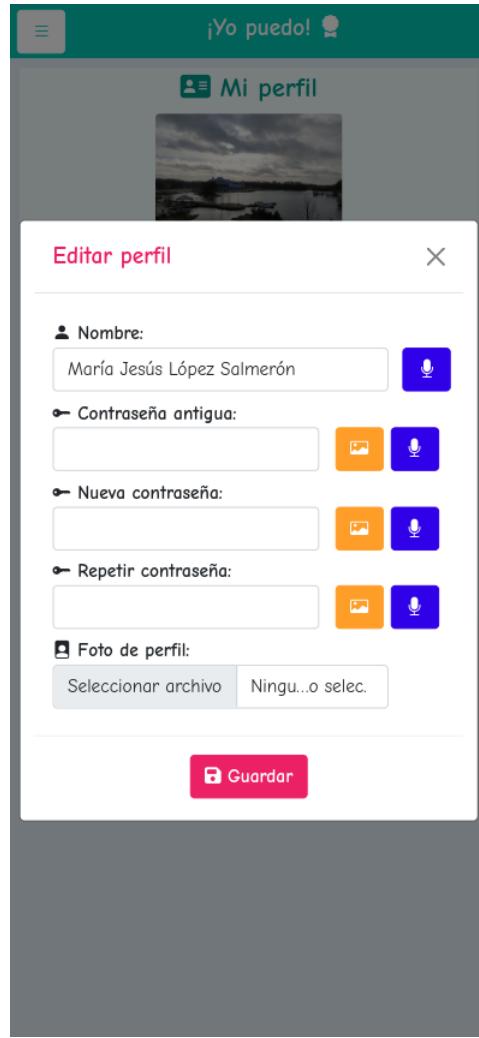


Figura 53: Capturas de pantalla de la edición del perfil de usuario para tablet, ordenador y móvil.

Cuando el usuario desea guardar los cambios realizados, el backend comprueba que los datos enviados son los correctos y modifica el registro con la información de ese usuario dentro de la base de datos.

Cancelación de la cuenta

Como sucedió con el cierre de sesión, la implementación de cancelación de la cuenta también fue rápida de llevar a cabo puesto que únicamente añadimos el botón rojo dentro del perfil de usuario (que podemos ver en la Figura 52) que llama al backend para que envíe un correo electrónico con la clave aleatoria (por ejemplo, el de la Figura 54) y para que active el pin parental (como el de la iteración anterior anterior).

Cuando el usuario haya introducido correctamente la clave (ya sea la aleatoria o la fija), el sistema elimina el usuario dentro de la base de datos.

Eliminar la cuenta de Buscador TFG de YoPuedo

Recibidos x



para mí ▾

15 mar 2023, 8:27



¿Eres tú?

Este correo se ha mandado a través de la aplicación ¡Yo puedo! para confirmar que deseas seguir hacia delante con **Eliminar la cuenta de Buscador TFG de YoPuedo**.

Si es así, solo tienes que escribir la siguiente clave en la aplicación y dar a **VERIFICAR**. Si no has sido tú, ignora este mensaje.

CLAVE: zuhVhrtYPt

Figura 54: Ejemplo de email de confirmación de cancelación de cuenta.

Pruebas

Siguiendo con los pasos realizados para este sprint, avanzaremos con las pruebas realizadas sobre las tareas de esta iteración. Estas pruebas las podemos dividir de la siguiente forma:

- Relacionado con la implementación de las tareas:
 - Cuando llamamos a visualización del perfil de usuario, realizamos dos acciones:
 - Miramos que nos redirige a la página de registro si el usuario no ha iniciado sesión.
 - Observamos que muestra la información correcta del usuario cuando el usuario haya iniciado sesión previamente.
 - Cuando lanzamos el cierre de sesión, llamar a una funcionalidad que requiera que el usuario haya iniciado sesión previamente y ver que nos redirige a la página de registro.
 - Cuando pedimos la edición de datos personales, observamos que:
 - Si un usuario no ha iniciado sesión, redirija a la página de registro de la aplicación.
 - Si el usuario ha iniciado sesión y ha modificado sus datos, validamos que:

- La contraseña antigua dada sea la guardada en la base de datos.
 - La contraseña nueva tenga entre 8 y 16 caracteres.
 - La contraseña nueva tenga como mínimo una minúscula, una mayúscula, un número y símbolos.
 - La contraseña nueva y la repetida sean iguales.
 - La foto de perfil sea de formato imagen.
- En relación con los bocetos, realizamos las pruebas necesarias para ver que cumplían con las normas de las guías de accesibilidad. Es por eso que creamos dos versiones distintas para este sprint.

Retrospectiva

En relación con las retrospectivas de este sprint, llevamos a cabo algunas de las tareas del sprint anterior y las planificadas para esta iteración en el tiempo de duración marcado para cada sprint de este proyecto.

3.9. Tercer sprint

En este sprint nos vamos a encargar de algunas funcionalidades relacionadas con los retos: la creación de uno individual, la diferenciación en la consulta entre individuales y colectivos, la visualización de un reto una vez creado y la obtención del listado de retos según los filtros aplicados, y según categoría de reto.

Historias de Usuario

- **[HU8] Como persona, necesito crear un reto individual.**
 - Quién: Persona.
 - Cuándo: Cuando quiera retarse a llegar a un cierto reto.
 - Entonces: La persona tiene un reto que debe cumplir según en las etapas que lo haya dividido.
 - CoS:
 - El usuario debe haber iniciado sesión en el sistema previamente.

- El identificador (que se generará de manera automática) es la clave primaria.
- El reto debe tener los siguientes elementos: título, objetivo, categoría, recompensa, etapas.
- Tiene que haber más de una etapa y no más de cinco.
- Cada etapa debe tener los siguientes elementos: identificador, objetivo y estado.
- El objetivo y la recompensa pueden ser de los siguientes tipos de formatos: vídeo, texto, audio o imagen.
- Para aquellos objetivos y/o recompensa que no sean textuales, sino que se haya adjuntado un fichero (de imagen, vídeo o audio), se guardará la ubicación del fichero dado por el usuario y almacenado dentro del sistema.
- El estado inicial de todas las etapas pasan a “*Propuesto*”.
- El estado inicial del reto pasa a “*Propuesto*”.
- Para facilitar al usuario a la hora de crear un reto, lo dividiremos en pestañas de color distinto: la primera será la que contendrá la información general del reto (título, objetivo, categoría y recompensa) y la segunda será en la que se creen las distintas etapas de los retos. En la segunda pestaña se subdivide a su vez en pestañas según la cantidad de etapas que se cree.
- El usuario que ha creado ese reto pasa a ser el “coordinador” del reto, para indicar que es el usuario que tiene permisos para modificar el reto.
- Dicho usuario se añade como participante del reto por si en un futuro deja de ser el coordinador del reto creado.
- Si todo está correcto, se guardan los datos generales del reto juntos y cada una de las etapas por separado y se va a la página donde muestra el reto creado.

- [HU9] Como persona, necesito ver mis retos según el tipo de reto (individual o colectivo).
 - Quién: Persona.
 - Cuándo: Cuando quiera ver los retos que tiene creados, los que le quedan por alcanzar, los ya realizados y los que está animando.
 - Entonces: Se muestra una lista de retos según el tipo señalado.
 - CoS:
 - Se realiza una consulta sobre los retos en los que participa o anima esa persona según el tipo indicado.
 - Si el tipo indicado es individual, se mostrarán aquellos retos en los que hay sólamente un participante. Si es colectivo, se enviarán aquellos que tengan más de un participante.
 - Se mostrará, de cada uno de los retos resultantes, la categoría, el título y el objetivo.
 - Se paginará la lista de retos cuando su cantidad supere a 5 retos.
- [HU10] Como persona, necesito filtrar mis retos según el estado en el que se encuentren.
 - Quién: Persona.
 - Cuándo: Cuando quiera ver los retos de un estado dado previamente.
 - Entonces: Se muestra una lista de retos con el estado indicado.
 - CoS:
 - Se realiza una consulta sobre los retos en los que participa o anima .
 - Se mostrará, de cada uno de los retos resultantes, la categoría, el título y el objetivo.
 - Se paginará la lista de retos cuando su cantidad supere a 5 retos.
- [HU11] Como persona, necesito filtrar mis retos según la categoría que puedan tener.

- Quién: Persona.
- Cuándo: Cuando quiera ver los retos de una categoría concreta un estado dado previamente.
- Entonces: Se muestra una lista de retos.
- CoS:
 - Se consultan sobre los retos guardados dentro de ese usuario que se encuentren en el estado dado.
 - Devuelve una lista de retos.
 - Se paginará la lista de retos cuando su cantidad supere a 5 retos.
- [HU12] **Como persona, necesito ver con detalle un reto.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver un reto en concreto.
 - Entonces: Devuelve el reto, los animadores y los participantes que forman parte del reto, las pruebas de cada una de las etapas, las calificaciones de cada una de las etapas de esa persona y los mensajes de ánimo de cada etapa.
 - CoS:
 - Se devuelve la siguiente información:
 - Título.
 - Objetivo.
 - Categoría.
 - Recompensa.
 - Etapas. Que cada una de ellas a su vez mostrarán:
 - Objetivo.
 - Pruebas, si el usuario es uno de los participantes.

- Calificaciones de ese usuario, si el usuario es uno de los participantes.
 - Mensajes de ánimo. Verá todos los mensajes si el usuario es un participante del reto o un superanimador y verá los suyos si es uno de los animadores del reto.
- Participantes del reto, excluyendo al usuario. Que estos, a su vez, mostrarán:
 - Su foto de perfil.
 - Su nombre.
- Animadores del reto, excluyendo al usuario. Que estos mostrarán:
 - Su foto de perfil.
 - Su nombre.
 - Si es un superanimador o no.
- Se dividirá la información en etapas:
 - General (título, objetivo, categoría y recompensa),
 - Etapas (que a su vez se dividirá en pestañas cada una de las etapas),
 - Participantes,
 - Animadores.
- En aquellos objetivos o recompensas en las que sean audios, se transcriben los audios a textos.
- En aquellos elementos textuales, se crean audios con el contenido de dicho elemento.
- Cada elemento del reto, como los títulos de cada una de las pestañas, tienen que tener un título y un ícono asociado a la información que muestra.

- Cada pestaña creada debe ser de un color distinto para ayudar al usuario a saber qué información se está mostrando.

Bocetos

En cuanto a los bocetos que exhiben las funcionalidades de mostrar el listado de retos de un cliente, mostrar los datos de un reto en concreto y la creación de un reto (mostradas en las *Figuras 55, 56 y 57*), podemos divisar que más o menos son parecidos entre los distintos formatos de pantallas, lo único que los diferencia es el espaciado de los distintos elementos de las páginas.

Además de la información que se muestra en una parte de la ventana, creamos un menú en el lateral izquierdo (visualizado en cada una de las pantallas de la *Figuras 55 y 57*) de la pantalla en donde el usuario puede ir a una funcionalidad o a otra dentro del sistema, como ir a ver a los amigos o cerrar sesión. En el formato móvil, se oculta dicho menú pero se puede mostrar u ocultar de nuevo mediante un botón que se encuentra en la esquina superior izquierda.

Con respecto a las características que le faltan al menú para considerarlo apto para todos los usuarios, están el que es necesario incorporar elementos visuales que les hagan diferenciar unos elementos de otros (para así cumplir con la propiedad [Comprendible](#) de la guía WCAG), como se ha comentado en el apartado anterior en la distinción de elementos de la página de inicio.

En los bocetos que describen la funcionalidad del listado de retos (la primera pantalla de las *Figuras 55, 56 y 57*) se distinguen los siguientes elementos:

- Un botón donde lleva a añadir un nuevo reto en la colección de la cuenta del cliente.
- Un desplegable para filtrar los retos según el estado de avance en el que se encuentren.
- La lista de retos en la que, si se pincha sobre uno de ellos, lleva a ese reto presentando información con detalle. En la lista, aparece:
 - Nombre de los retos.
 - El estado de avance de los retos a través de un ícono.

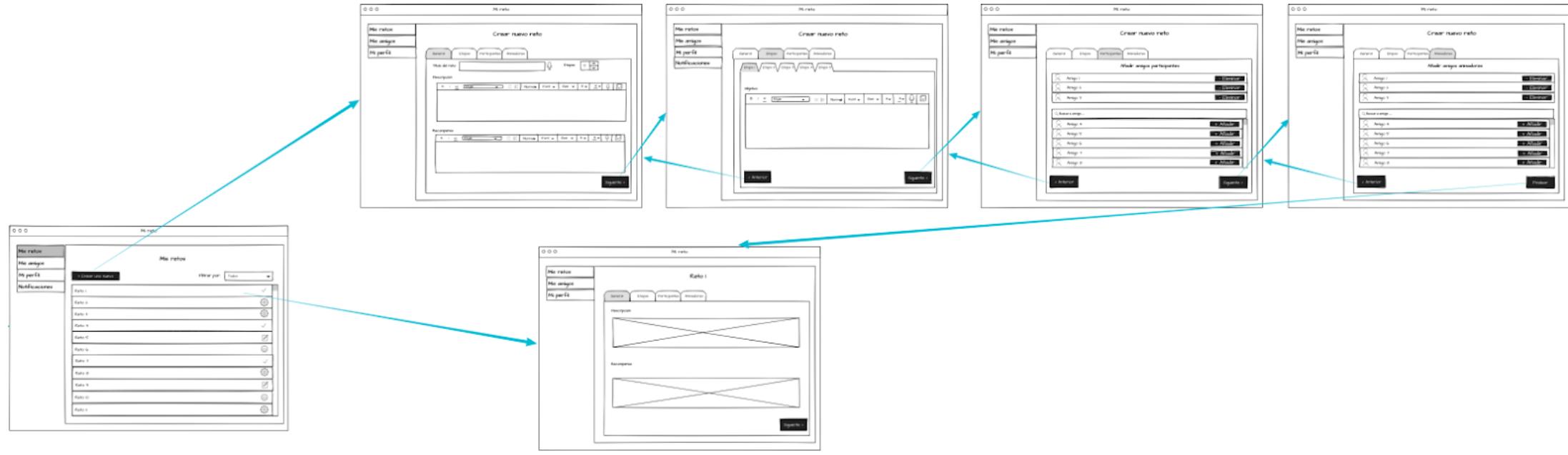


Figura 55: Primera versión de listado de retos, visualización de un reto y creación en pantalla de ordenador.

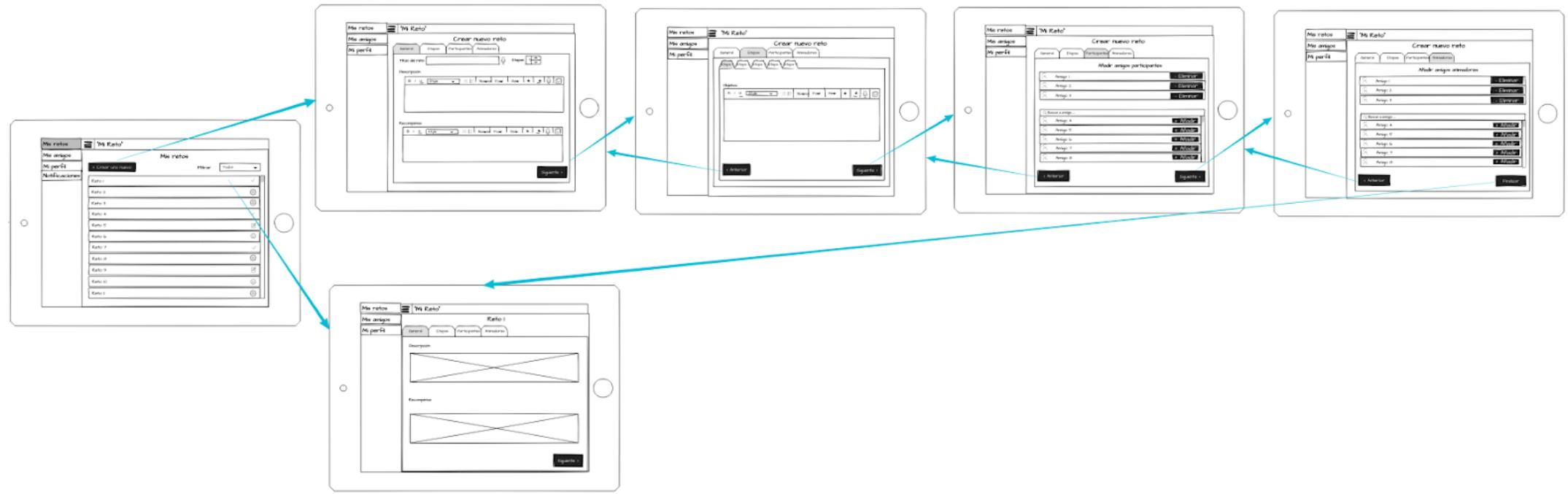


Figura 56: Primera versión en tableta de la visualización del listado de retos de un usuario y de un reto en concreto y la creación de un nuevo reto.

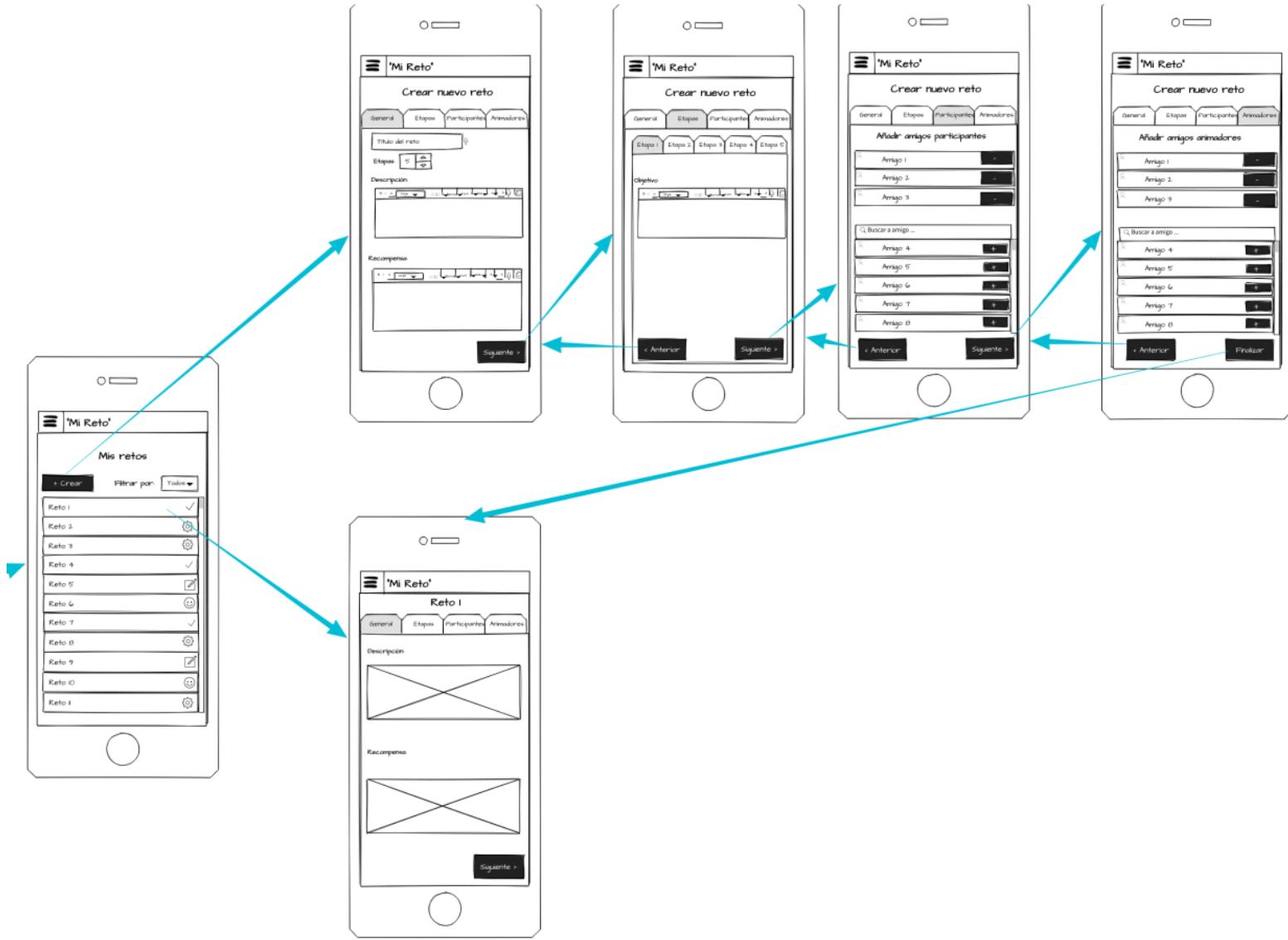


Figura 57: Primera versión de lista y detalle de un reto junto a su creación en formato móvil

En consenso con el cliente, los inconvenientes (muchos de ellos relacionados con accesibilidad o con mejoras de usabilidad) que fuimos capaces de percibir al mostrar de esta manera los retos de una persona son:

- Debemos diferenciar los retos, además de por el estado de avance en el que se encuentran cada uno de ellos, según la categoría que tengan. Por lo tanto, tanto en el listado como a la hora de crear un nuevo reto vamos a incorporar un conjunto de categorías y ser capaces de filtrar por categoría también. Además, en cada reto se mostrará la categoría a la que pertenece éste a través de un símbolo.
- Los objetivos de los retos no deben ser únicamente texto, sino que también puedan ser imágenes, vídeos o audios, garantizando una alternativa al texto, tal y como indican las guías de accesibilidad. Por lo tanto, además de la inserción del objetivo del reto, debe mostrarse el título que defina el reto.
- Separación de los retos según el estado y si son retos no propios en los que se está animando. Para facilitar al usuario la ubicación del tipo de reto visualizado, les mostramos distintos colores para su diferenciación, tanto las pestañas como la parte del fondo de pantalla que se encarga de visualizar el listado. Además de marcar en negrita la pestaña que tiene señalada. En el caso del formato móvil, para poder ahorrar espacio, mostraremos los pictogramas que describen el estado de los retos mostrados de las pestañas que no están activas, mientras que para la que esté visualizando retos al usuario también se mostrará su título.
- Incorporación de botón en cada fila para acceder a un reto específico.
- En vez de mostrar la lista al completo en la ventana, se divide en distintas páginas el conjunto de retos. Por lo tanto, al final de la página nos encontramos con una herramienta para facilitar la navegación por la paginación de los retos.

En relación con la visualización de los datos que se han guardado de un reto, únicamente hemos mostrado los datos generales del reto (su descripción y la recompensa que se lleva cuando lo supere) junto a unas pestañas donde el usuario pueda ver el resto de características de ese reto, puesto que, en un primer momento, decidimos que el resto de pestañas fueran iguales que cuando se crea un nuevo reto.

Los impedimentos que podríamos ver en el diseño del boceto de esta funcionalidad son los siguientes: el primero es que no se encuentra ningún botón donde el usuario pueda editar o eliminar ese reto, el segundo es que deberíamos cambiar “Descripción” por “Objetivo” para no confundir al usuario sobre el dato pedido, el tercero es que deberíamos

diferenciar la visualización de un participante de un reto y la de un animador añadiendo en la pestaña de etapas las calificaciones, las evidencias y los mensajes de ánimo según el rol que tenga ese usuario dentro del reto seleccionado y, por último, es que faltan símbolos gráficos para identificar cada elemento de la página.

En cuanto a la visualización de los retos desde distintas perspectivas y al poder realizar distintas acciones dependiendo del rol del usuario dentro del reto, hemos decidido desarrollarlo en distintos sprints. En el caso de las acciones de un participante, vamos a llevarla a cabo en el siguiente sprint, mientras que las del animador en el sexto.

Con respecto a la pantalla que muestra la información de un reto ya creado (*Figura 58* para formato móvil, *Figura 59* para la versión de ordenador y *Figura 60* para tablets), mostramos toda la información que podría tener un reto (información general, sus etapas, los animadores y los participantes, excepto él mismo, que estén incluidos dentro del reto), aunque este no sea colectivo o no tenga animadores en él. En cada subpestaña (o mejor dicho, en cada etapa) que encontramos en la pestaña de “Etapas”, mostraremos el objetivo de una etapa.

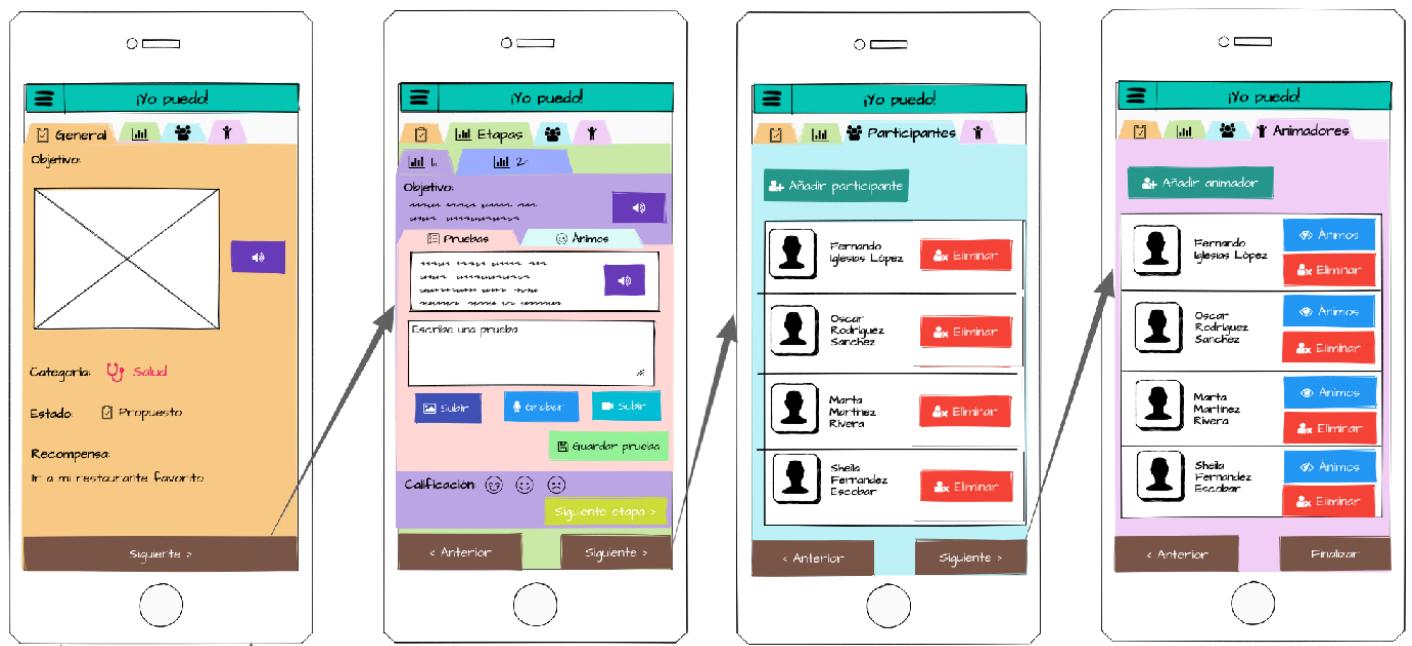


Figura 58: Segunda versión de la visualización de un reto desde la vista de un participante en formato móvil.

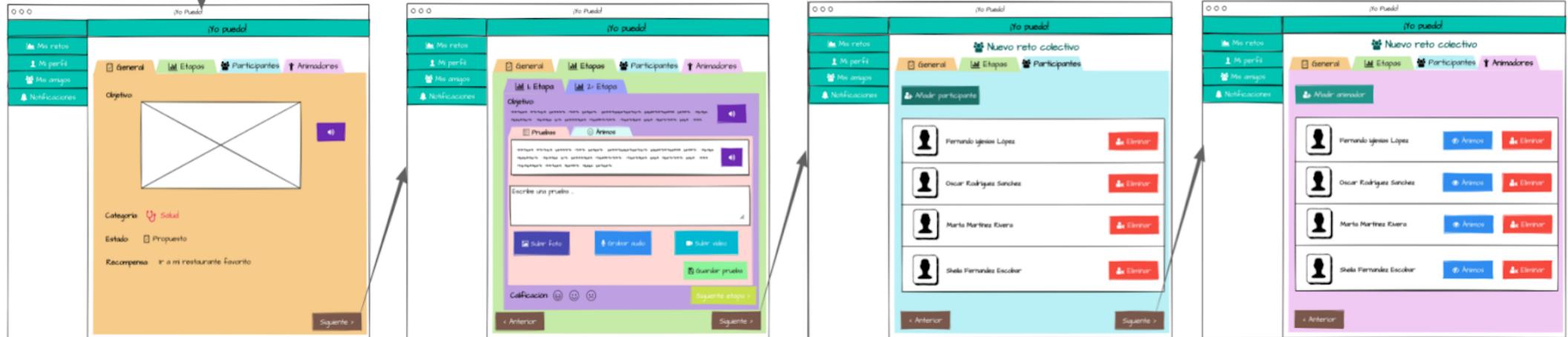


Figura 59: Segunda versión de la visualización de un reto desde la vista de un participante para ordenadores.

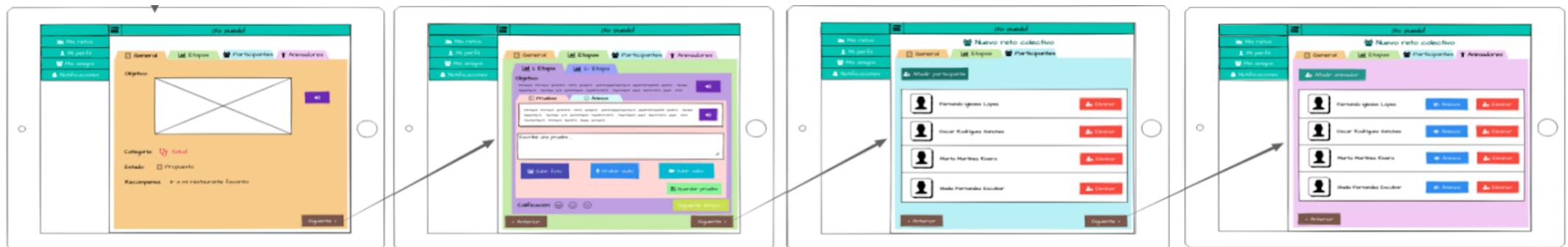


Figura 60: Segunda versión de la visualización de un reto desde la vista de un participante en tablets.

En cuanto a los bocetos relacionados con la creación de un reto, podemos apreciar los distintos pasos que tiene que realizar un usuario para añadir un nuevo reto a su colección.

Si nos centramos en los pasos que exhiben cada uno de los diseños, en el de la esquina izquierda se encarga de recoger los datos más generales del reto: el título del reto, una descripción de él, el número de etapas en las que se va a dividir y la recompensa que va a obtener una vez superadas todas las etapas. Una vez escritos los datos principales del reto, detallamos el objetivo a conseguir en cada una de las etapas. Cuando hayamos finalizado de introducir los objetivos de cada etapa creada, nos pasamos a añadir aquellos amigos que queramos que nos apoyen en este reto (presentado en la tercera pantalla), a esas personas les llamaremos “animadores”, y, seguidamente, aquellos que queramos que hagan el reto con nosotros (visualizado en el último boceto), dicho de otra manera, los “participantes” del reto. Estos dos últimos pasos no son obligatorios de realizar, sino que dejamos al usuario decidir si quiere añadir amigos con un rol específico a su reto o no.

Los cambios que pensamos realizar para los futuros bocetos para mejorar el diseño de creación de retos, al no ser realmente intuible o no totalmente accesible su uso por la forma planteada y por la falta de información en sus formularios tras una reunión con el cliente, son:

- No mostrar todas las pestañas con los pasos que tiene que realizar el usuario, sino que cuando el usuario vaya pasando a las siguientes tareas de creación se vayan añadiendo las pestañas para así ayudar al usuario a centrarse en el paso correspondiente.
- No añadir en la primera ventana las etapas que va a tener el reto, sino que en el paso de llenar los datos de las etapas sea donde se crean según vaya necesitando. Así no tiene que volver a la primera pestaña para cambiar el número de etapas, sino que lo hará en su correspondiente paso.
- En vez de pedir el título del reto y la descripción, pediremos el título, el objetivo, la recompensa y la categoría de este.
- En los campos que se puedan escribir o insertar fotos, vídeos o audios, como el objetivo y la recompensa, se pondrá un campo de texto y varios botones por si desea insertar esa información en otro formato no textual, en vez de ese tipo de editor porque así facilita a cualquier persona el hacerlo a añadir el objetivo o la recompensa de la manera que él desee.

- En cuanto a la forma de incorporar animadores o participantes, en sus correspondientes pantallas agregamos un botón que lleve al usuario a una ventana aparte donde pueda buscar y seleccionar los amigos que desea incluir y una lista (vacía o con algún usuario) de los usuarios que se asocian adjuntan al reto. De esta forma, hacemos que la incorporación de amigos al reto sea de la manera más limpia y sencilla posible. Esta idea la implementaremos en el sexto sprint en cuanto a la inserción de animadores (al ser la etapa del desarrollo del prototipo en la que nos vamos a centrar en todas las funcionalidades del animador de un reto) y en el séptimo cuando se añadan participantes (será la etapa en la que nos centremos en el desarrollo de retos colectivos y las distintas acciones que se pueden hacer sobre él).
- Separar la creación de un reto individual de un reto colectivo preguntando, antes de comenzar a llenar la información de este, el tipo de reto que desea iniciar. Si es uno individual, desaparecería la pestaña de participantes durante la creación. Y sino, estaría primero la selección de participantes antes que la de animadores. En el séptimo sprint, veremos el diseño de los bocetos cuando se crean retos colectivos.
- Con respecto a los animadores, pensamos que debería haber “superanimadores”, es decir, usuarios que están animando y que puedan ver todos los mensajes de apoyos del reto. Esta idea la pensamos para aquellos casos en los que se utiliza con menores y/o tutelados y sea necesario la supervisión de comentarios sobre los retos. Por lo tanto, cuando se añadan los amigos que vayan a ser animadores, se puede dar la opción de marcar a algunos de esos usuarios elegidos como superanimadores. Al ser otra posible funcionalidad, pensamos que lo mejor sería centrarnos en ella en otro sprint, en este caso en el sexto sprint, que nos centraremos en las historias de usuario que puede hacer un usuario, con el rol de animador, dentro de un reto.
- Se propone incorporar, junto a cualquier elemento textual de la página, elementos gráficos que identifiquen el significado de la frase.

En cuanto a los cambios que vamos a realizar basados en la primera versión de los bocetos de este sprint, primero vamos a hablar sobre el listado de retos de un usuario. Si nos centramos en las *Figuras 61, 62 y 63*, podemos apreciar que se ha dividido cada estado en el que puede estar en pestañas de distintos colores y, además, hemos insertado como otra pestaña aquellos retos en los que anima dicho usuario. También podemos darnos cuenta que cada reto tiene los siguientes elementos: la categoría a través de su nombre y de un ícono que las identifica, el título del reto y el objetivo (mediante un texto, foto, audio o

vídeo junto a la transcripción de aquellos elementos no textuales) y un ícono que nos redirija a ver con detalle ese reto para lograr que la aplicación sea comprensible, según las [guías de accesibilidad](#) (tal y como se explica en el [ANEXO 2](#)).

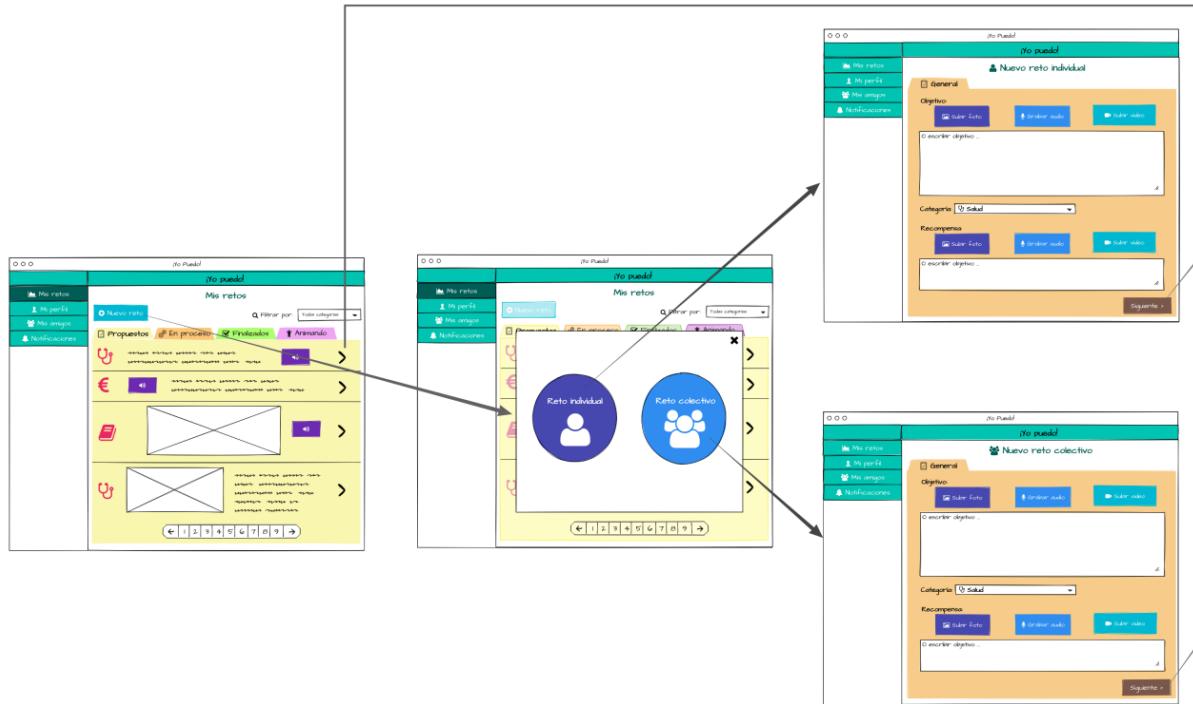


Figura 61: Listado de retos y selección del tipo de reto para su creación para pantallas de escritorio.

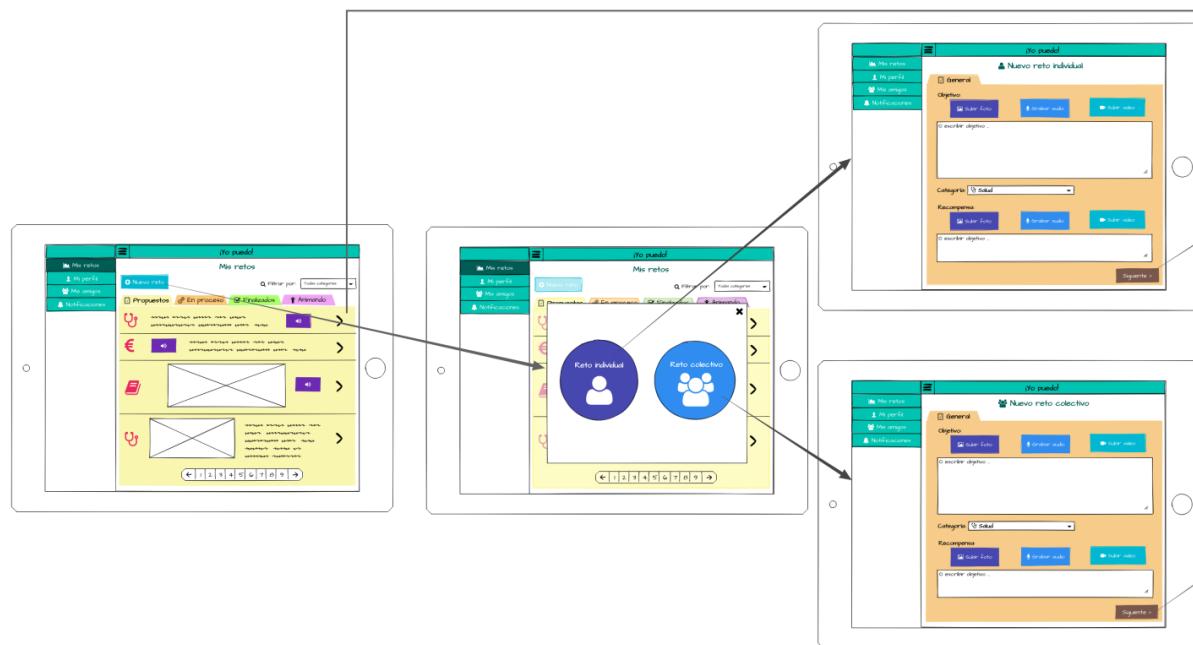


Figura 62: Listado de retos y selección de tipo de reto antes de su creación en tablet.

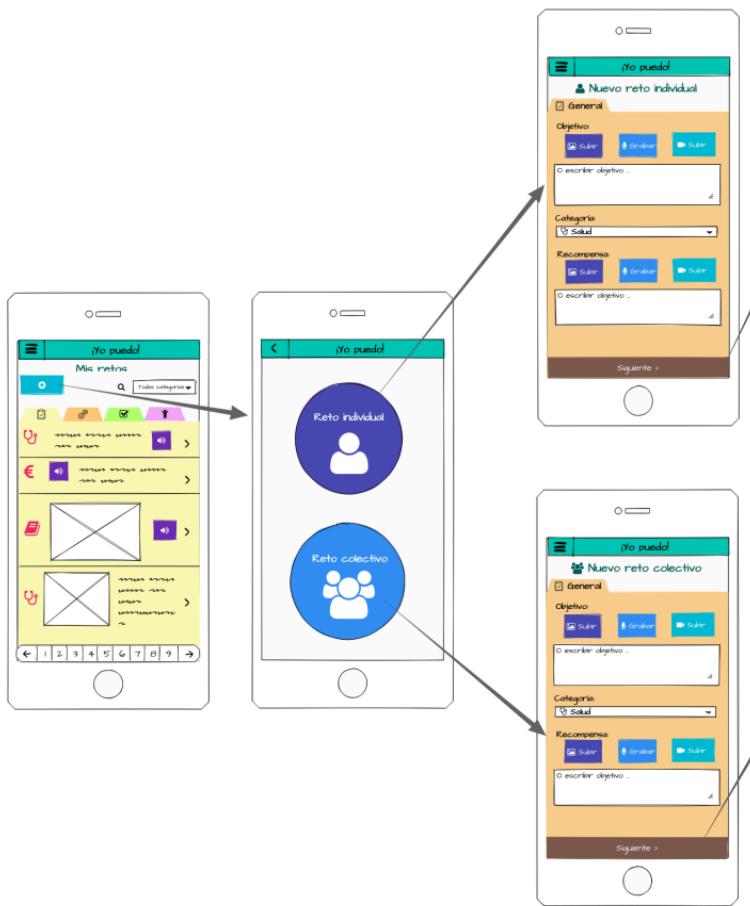


Figura 63: Listado de retos y selección del tipo de reto antes de su creación para móviles.

De esta forma no sabemos si los retos mostrados son colectivos o individuales, por lo que pensamos que primero debemos preguntar al usuario si quiere ver los retos individuales o colectivos previamente como lo hacemos antes de crear un nuevo reto. Esta petición la vemos reflejada en las *Figuras 64, 65 y 66*.

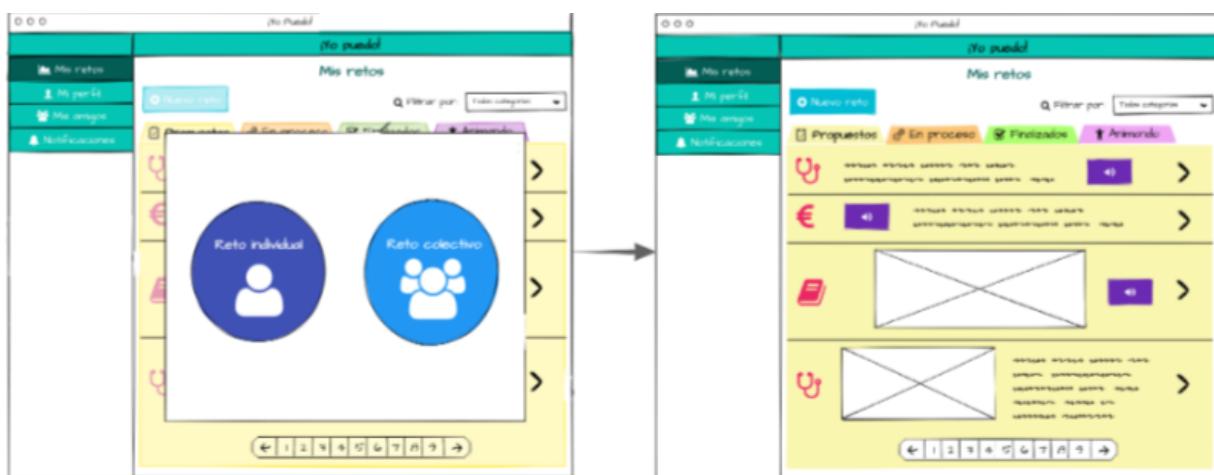


Figura 64: Filtrado por tipo de reto (individual o colectivo) en formato ordenador.

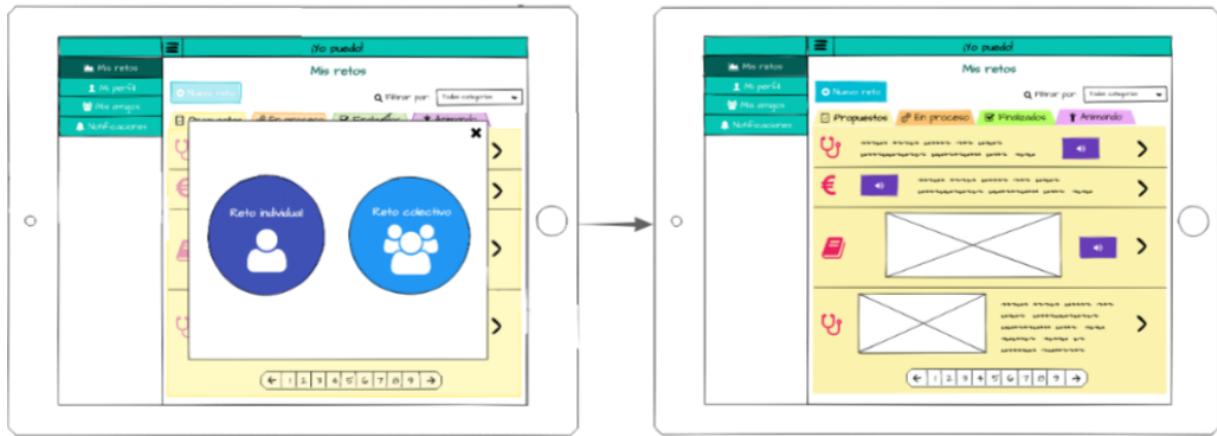


Figura 65: Filtrado por tipo de reto (individual o colectivo) en formato tablet.

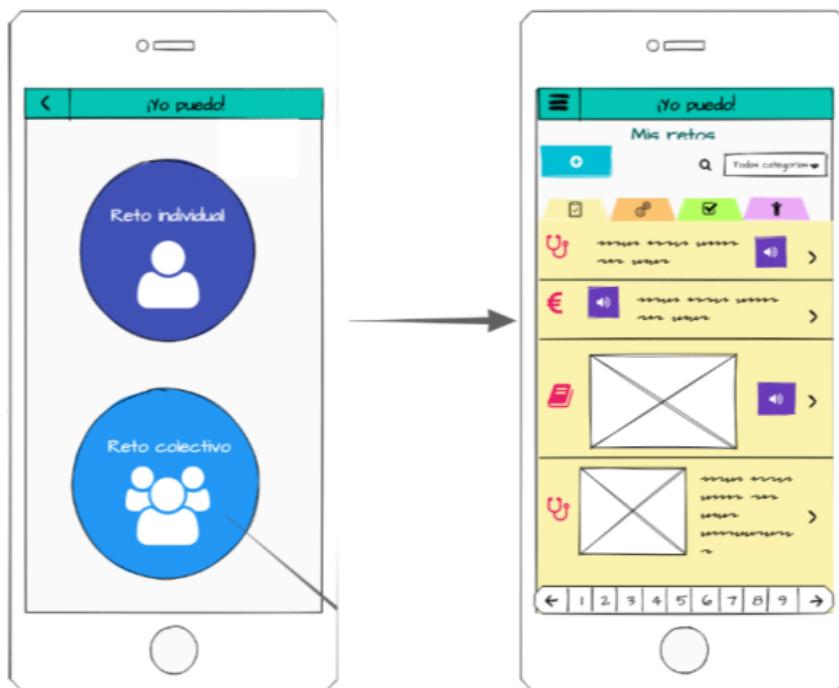


Figura 66: Filtrado por tipo de reto (individual o colectivo) en formato móvil.

En segundo lugar, vamos a explicar cómo llevamos a cabo los cambios comentados sobre la creación de nuevos retos. Señalamos que, antes de comenzar a escribir los datos del reto, el usuario debería indicar si es un reto individual o colectivo, como podemos ver en los tres bocetos mostrados de las Figuras 61, 62 y 63. Por lo tanto, cuando pulse sobre el botón “Nuevo reto” se abrirá una pequeña ventana dentro de la aplicación con dos botones grandes: uno para iniciar un reto individual (con el símbolo de una persona) y el otro uno colectivo (con el símbolo de un conjunto de personas).

Una vez seleccionado el tipo de reto que deseamos añadir a nuestra propia colección, comenzaremos a escribir los datos necesarios para poder guardarlos en el

sistema. Para saber qué selección hemos hecho, durante la creación del reto veremos un título indicando el tipo de reto que se está creando o editando.

En las *Figuras 67, 68 y 69* podemos observar que nos muestra los pasos para crear un reto individual apareciendo, paso a paso, las nuevas pestañas según vaya el usuario llenando sobre el reto: primero la parte general, a continuación el objetivo de cada etapa y por último la inserción de animadores al reto. En cuanto a la creación de retos colectivos, decidimos centrarnos, tanto su diseño como su implementación, para más adelante, en concreto para el séptimo sprint.

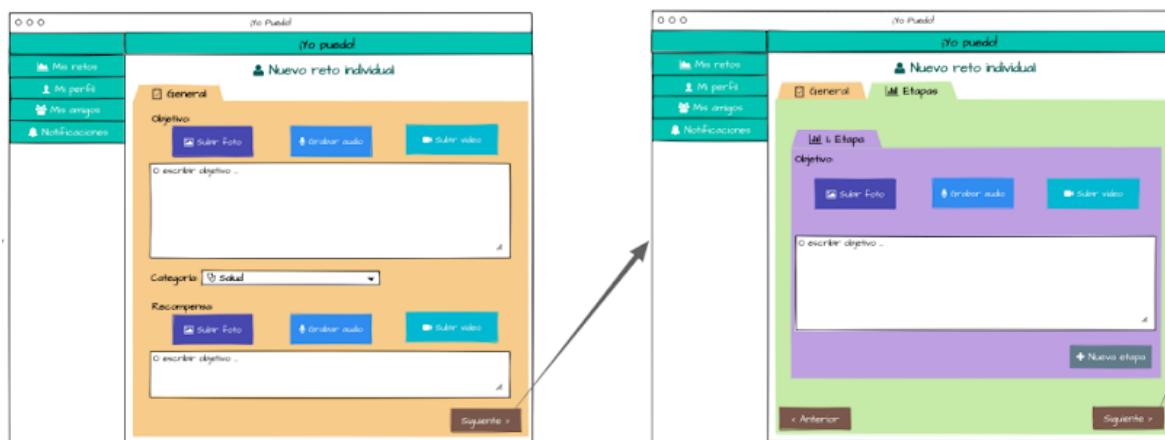


Figura 67: Segunda versión en formato escritorio de la creación de un reto individual.

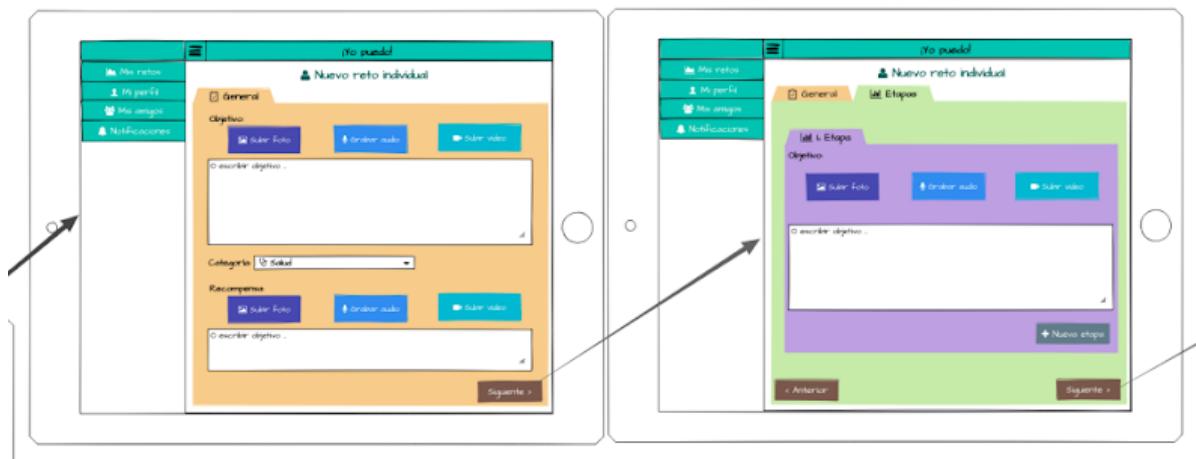


Figura 68: Segunda versión de la interfaz de usuario de la creación de un reto individual en formato tablet.

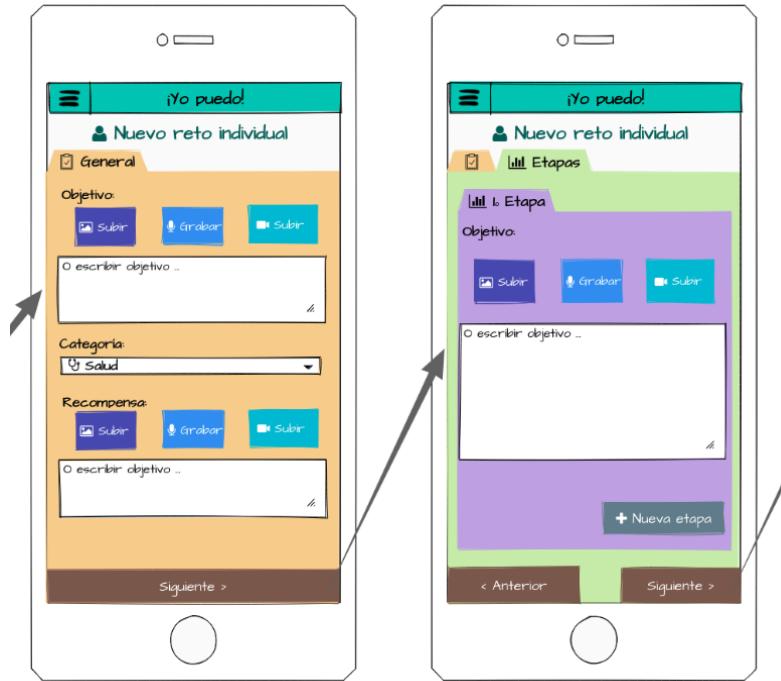


Figura 69: Segunda versión del diseño móvil de la creación de un reto individual.

De los bocetos de la segunda versión de la creación de un reto individual, hemos conseguido, de cada uno de los problemas hallados en la primera versión, encontrar la solución más idónea:

- Facilitar al usuario que inserte información en el formato que él quiera para los objetivos y recompensas del reto. Para ello, hemos introducido tres botones (uno para añadir una imagen, otro para insertar un audio y otro para subir un vídeo), además del campo de texto, para que el cliente sea el que decida cómo adjuntar la información.
- Indicar la categoría del reto, que en un primer momento no permitíamos al usuario indicar qué clase de reto quiere realizar. Su solución fue insertar en la pestaña general del reto un campo donde el cliente selecciona la categoría del nuevo reto, ya prefijadas por el sistema.
- No preguntar la cantidad de etapas que va a tener al inicio de su creación. Sino cuando el usuario va indicando el objetivo de cada etapa, conforme vaya viendo que necesita partir aún más los pasos para cumplir el objetivo general del reto, añadir más etapas sobre la marcha.

- Mostrar poco a poco los pasos de la generación de un reto. Se llevará a cabo mediante pestañas que van apareciendo y abriéndose poco a poco según el usuario vaya llenando los datos correspondientes en cada paso.

Base de datos

En cuanto a la base de datos diseñada para poder implementar todas las entidades, comentadas en cada una de las historias anteriores, junto a sus respectivas relaciones es la que vemos en la siguiente figura.

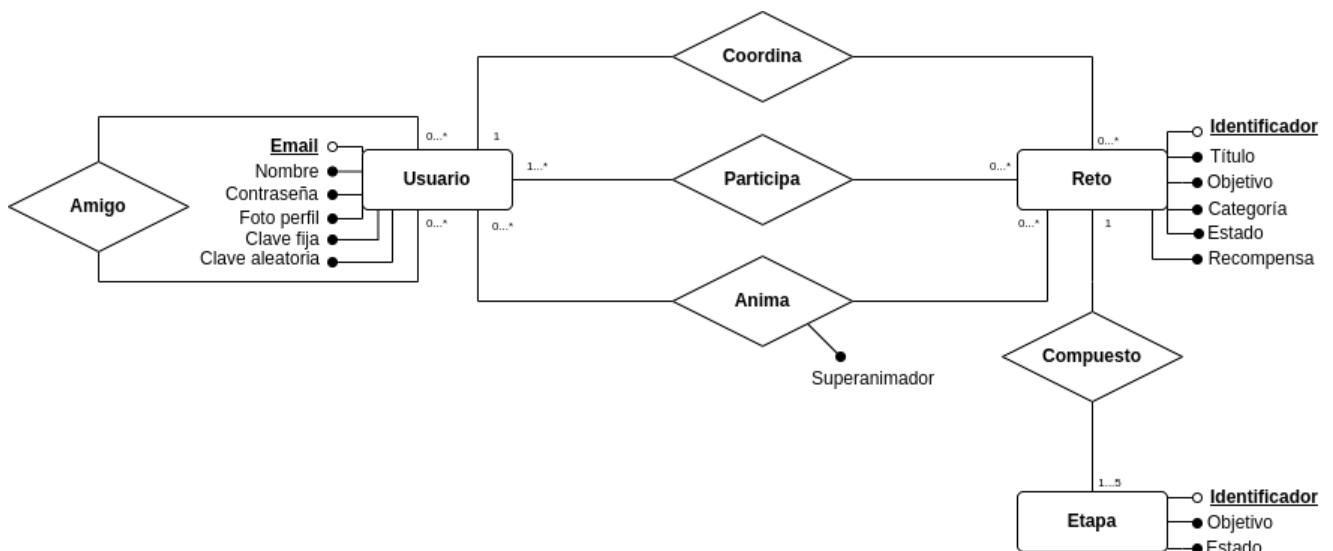


Figura 70: Base de datos para la implementación del tercer sprint.

Para este sprint se han añadido las siguientes entidades: *Reto* y *Etapa*, de las cuales se han unido con las siguientes relaciones:

- *Coordina*, que se une a la entidad *Usuario* para poder identificar al propietario de un objeto de la entidad *Reto*.
- *Participa*, que se une también *Reto* con *Usuario* para poder conocer a aquellas personas que participan en un reto y para saber cuándo un reto es individual (cuando solo tiene un participante) y cuando es colectivo (cuando tiene más de un participante).
- *Animar*, otra de las uniones *Reto-Usuario* para ver qué personas son las encargadas de animar en un reto específico. En esta relación se tendrá una característica más, que será la que nos ayude a identificar qué animadores tienen más privilegios o menos a la hora de visualizar los mensajes de ánimos de ese reto.

- *Compuesto*, una relación entre la entidad *Reto* y la entidad *Etapa* que nos ayudará a tener más información de un reto específico.
- *Amigo*, una relación entre la entidad *Usuario* para referenciar qué dos usuarios son amigos entre sí.

Implementación

Con respecto a la implementación tanto de las funcionalidades dentro del backend como de los bocetos en la parte frontend, hablaremos de cómo hemos realizado cada una de las [historias de usuario](#) de esta iteración.

Creación de retos

En cuanto a la creación de nuevos retos individuales, primero preguntamos al usuario si el tipo de reto que quiere generar (si de tipo individual o de tipo colectivo), tal y como lo planteamos en la segunda pantalla de los bocetos plasmados en las *Figuras 61, 62 y 63* y que fueron implementados de la forma que mostramos en la *Figura 71*.



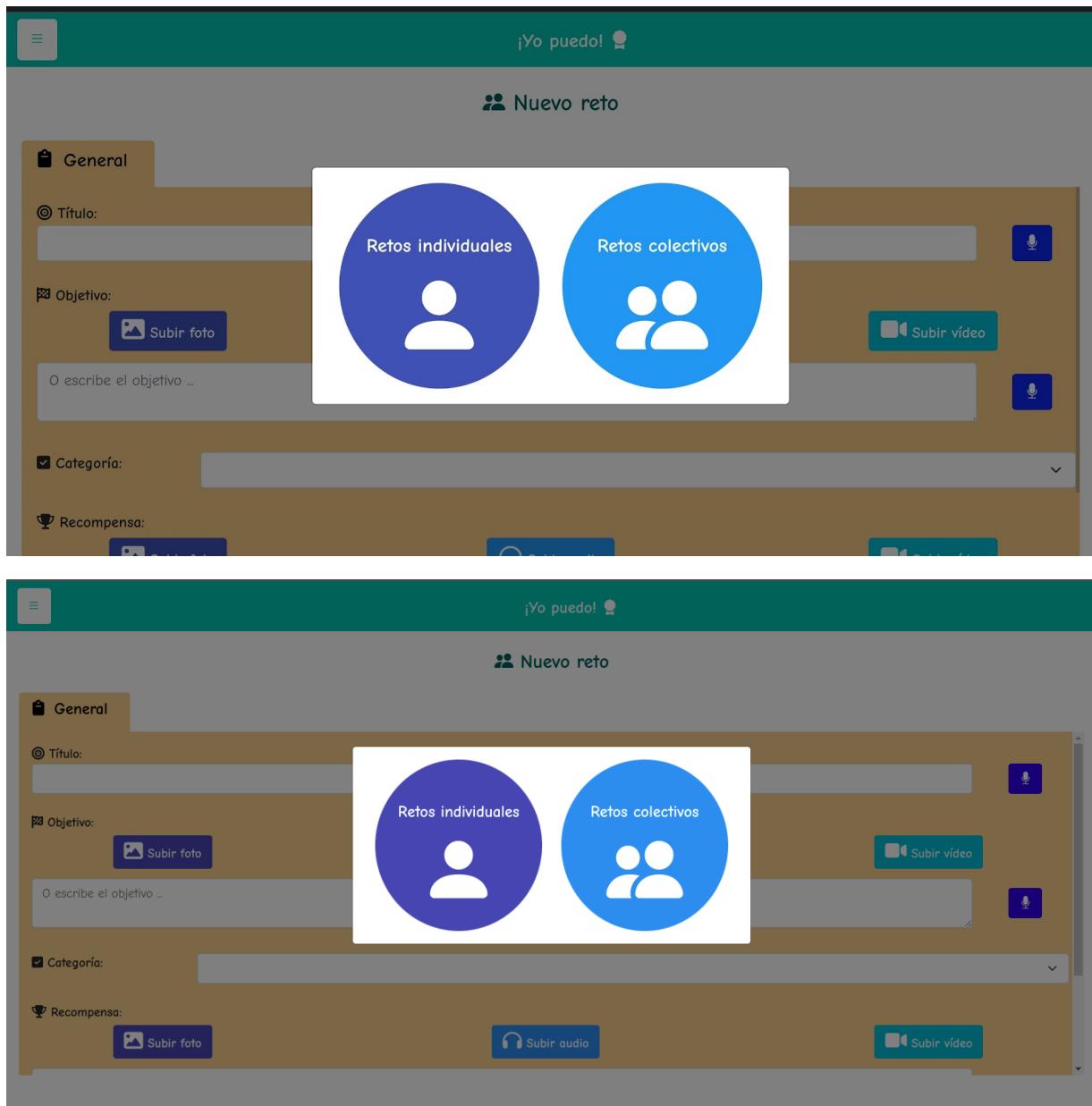


Figura 71: Primer paso antes de comenzar a crear un reto para pantallas móviles, tablets o escritorios.

Una vez que el usuario ha decidido el tipo de reto que desea crear, comenzamos a insertar los datos necesarios para crear un reto (los datos generales y las etapas que van a formar parte). Para ello lo que hacemos es dividir esas dos clases de datos en pestañas [93], junto al texto indicando el tipo de información que se va a insertar (siempre cuando las pantallas son grandes y cuando se esté mostrando la información de esa pestaña para pantallas pequeñas) y su correspondiente icono de *Font Awesome*, que ambas pestañas formarán parte del mismo formulario [94].

En la primera pestaña mostraremos los datos generales que debe tener un reto (como podemos ver su implementación en la Figura 72, que nos basamos en la primera

pantalla de los bocetos de las *Figuras 67, 68 y 69*), es decir, el título, el objetivo, la categoría y la recompensa.

¡Yo puedo!

Nuevo reto individual

General

④ Título:

☒ Objetivo:
O escribe el objetivo ...

☒ Categoría:

🏆 Recompensa:
O escribe la recompensa ...

¡Yo puedo!

Nuevo reto individual

General

④ Título:

☒ Objetivo:
O escribe el objetivo ...

☒ Categoría:

🏆 Recompensa:
O escribe la recompensa ...

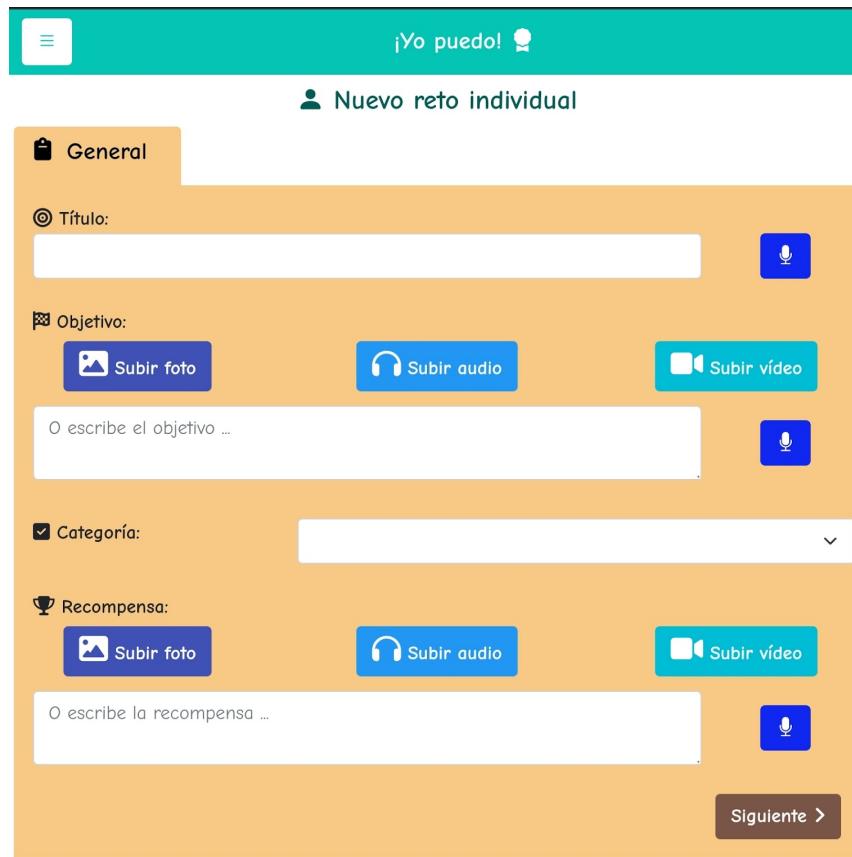


Figura 72: Pantalla general de la creación de un reto para ordenador, móvil y tablet.

Como ya hicimos cuando desarrollamos la funcionalidad de registrar a un usuario o editar sus datos, cada campo del dato que el usuario debe introducir tendrá su título y su correspondiente ícono, además de permitir que el usuario dicte la información que desea, si este campo es uno de texto.

Sin embargo, hay algunos de ellos, como el objetivo y la recompensa, que se pueden insertar, además de texto, otra clase de datos: una imagen, un audio o un vídeo. Esta clase de información decidimos que el usuario la suba a la aplicación a través de tres botones personalizados [95], como los de la *Figura 73*, para que así no ocupará mucho espacio en pantalla con los botones originales de HTML para la inserción de ficheros.

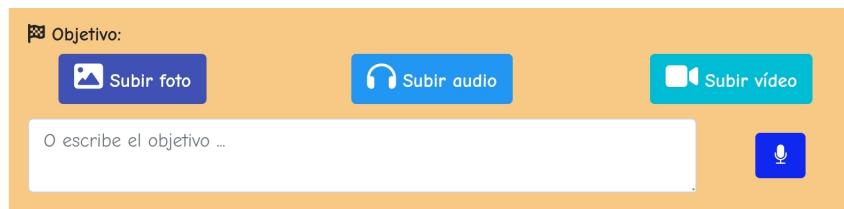
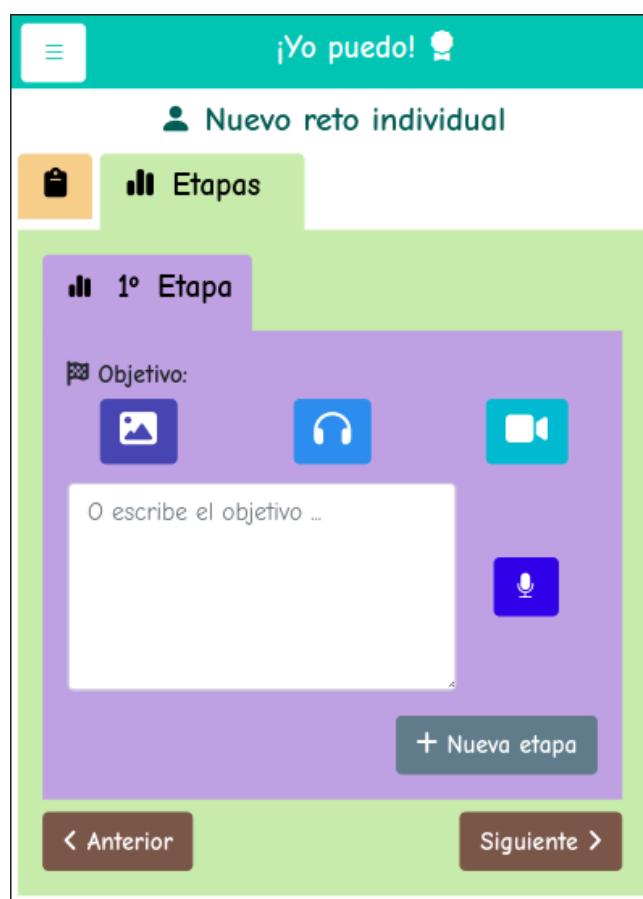


Figura 73: Ejemplo de botón personalizado para inserción de una imagen.

Por último, dentro de la pestaña general de un reto nos queda por indicar la categoría a la que pertenece el reto. Para que fuera fácil tanto para el administrador de la aplicación como para el usuario final, decidimos que esas categorías ya estuvieran definidas. Por lo tanto, creamos una lista de opciones [96] (ahorro, miedo, conocimientos y salud) para que el usuario pueda elegir una de ellas.

Cuando el usuario ya ha terminado de incluir la información general del reto, pasamos a que introduzca las etapas que desea realizar cuando el usuario pulse sobre el botón de navegación (*Siguiente >*) que se encuentra en la esquina derecha de la pantalla. En primer lugar, creamos un conjunto de formularios con la misma estructura [97, 98], divididos en pestañas también, en la que únicamente debe introducir el objetivo, de la misma forma que permitimos indicar el objetivo en la pestaña general. Para no mostrarle al usuario todos los formularios que puede llenar en pantalla, decidimos que fuera el usuario el que solicitara a la aplicación una nueva etapa (a través del botón de la esquina inferior derecha del formulario *Nueva etapa*) y se creará el formulario correspondiente dinámicamente [99]. Para la creación de esta pestaña nos basamos en la segunda pantalla de los bocetos de las *Figuras 67, 68 y 69*, que podemos ver los resultados de dicha implementación en la *Figura 74*.



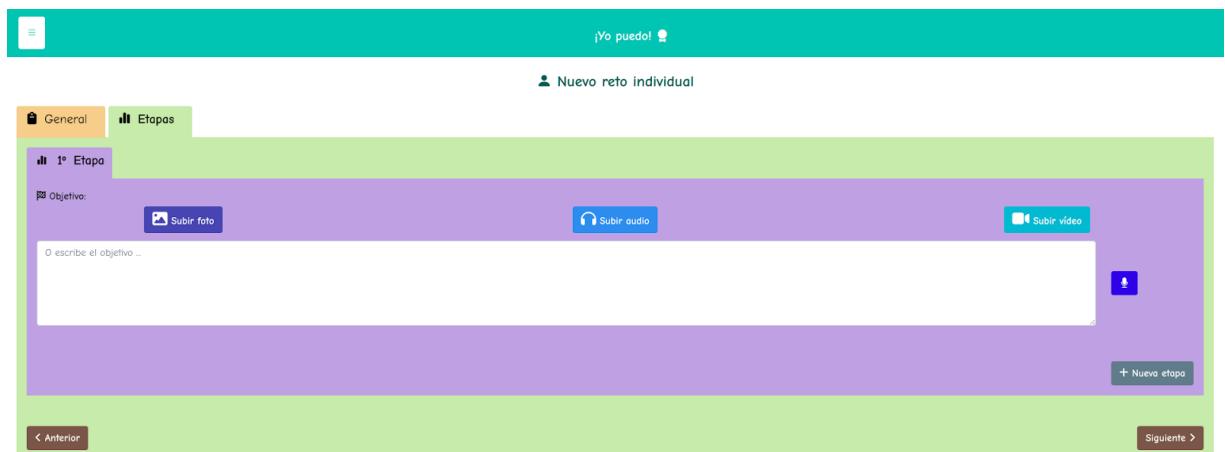
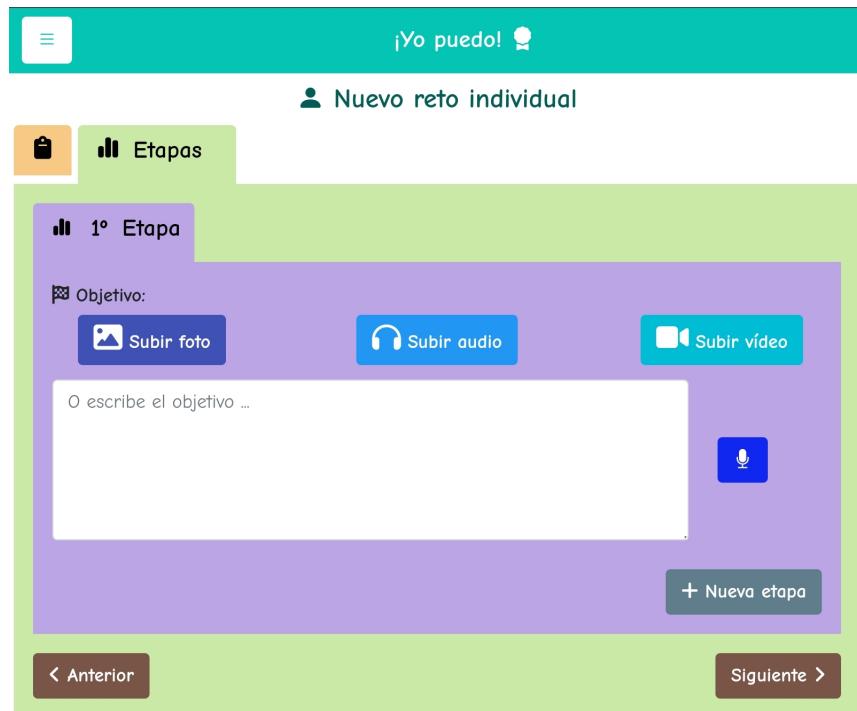


Figura 74: Pantalla de creación de etapas dentro de un reto para móvil, tablet y ordenador.

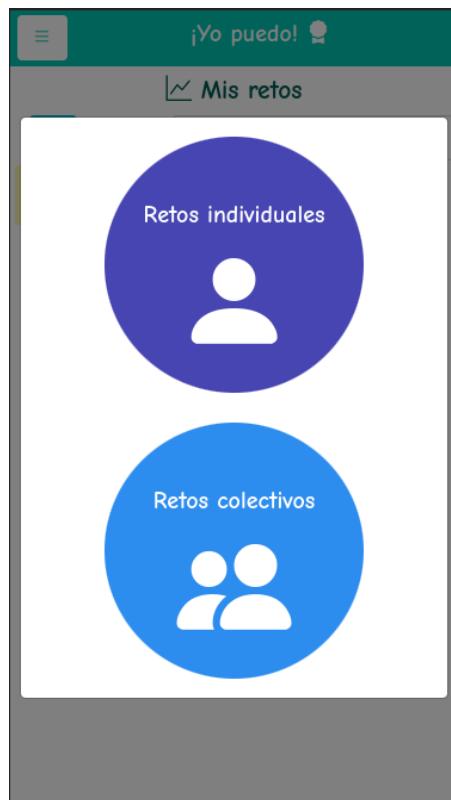
Terminado de indicar los datos generales del reto y de crear las etapas en las que se va a dividir, se manda dicha información al backend y comprueba que los datos mandados son correctos. Si las verificaciones salen correctamente, guardamos en la entidad *Reto* la información general del reto y creamos tantos registros en la entidad *Etapa* como etapas creadas.

Listado de retos y filtraciones

Después de crear un reto, es momento de implementar la funcionalidad de visualizar la lista de retos y las filtraciones que se pueden hacer sobre dicha lista (por tipo, estado de avance y categoría). Por lo tanto, comenzaremos a hablar sobre los pasos tomados para

listar los retos de un usuario según la consulta que desea realizar. Todas estas consultas únicamente lo podrán hacer todos los usuarios que previamente hayan iniciado sesión dentro de la aplicación.

La primera filtración que se haría, justo antes de enviar el listado de retos de los que forma parte el usuario al frontend, es la del tipo de retos que el usuario desearía visualizar (individuales o colectivos) a través de un modal con dos botones (similar al realizado para la creación de retos) como podemos ver en las *Figura 75* (que se basan en la primera pantalla de cada boceto de las Figuras 64, 65 y 66).



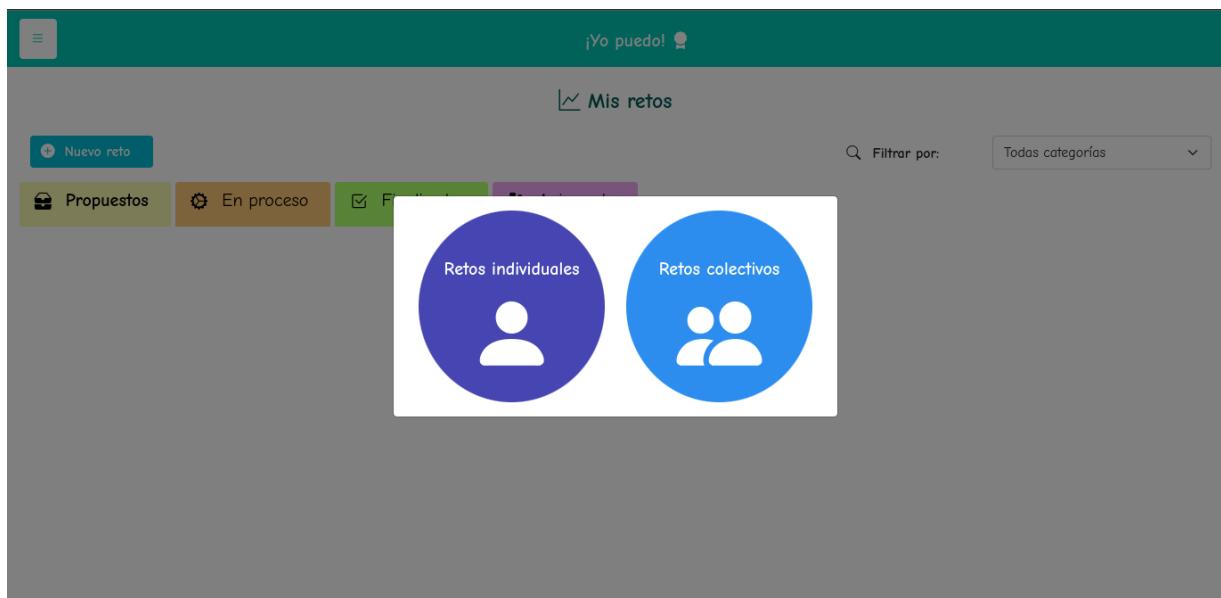
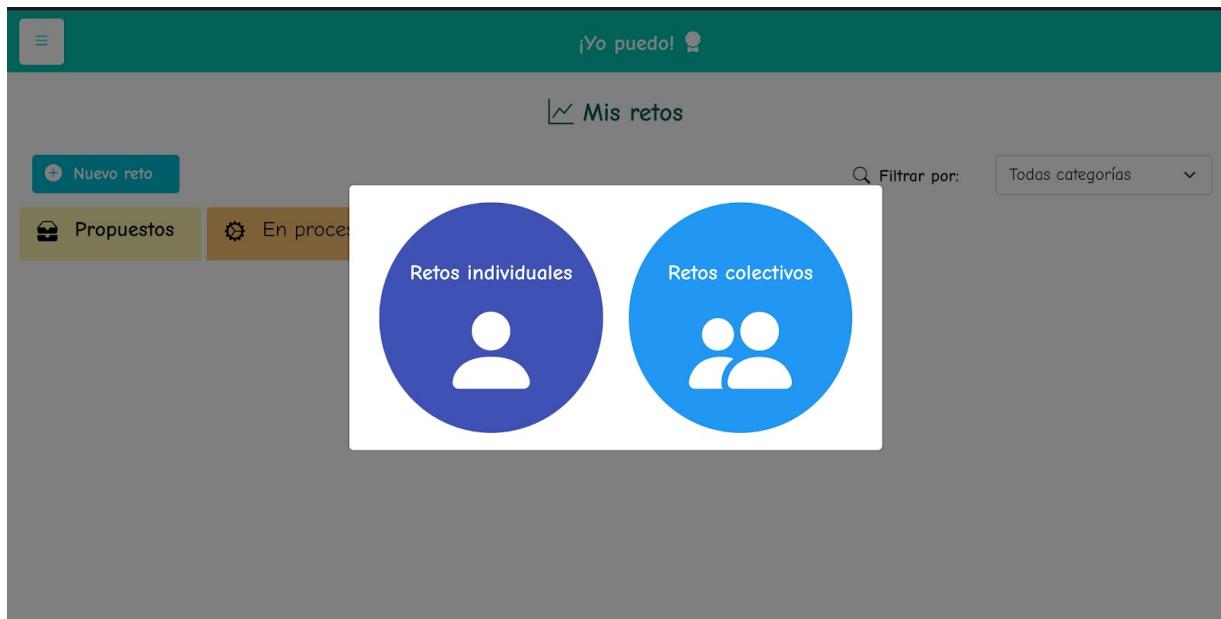


Figura 75: Filtración de retos por tipo de reto (individual o colectivo) en móvil, tablet y ordenador.

Una vez indicado el tipo deseado en la filtración de retos por tipos, dentro del backend hacemos una consulta INNER JOIN [99] con las tablas *Retos* y *Participante* (resultado de la [base de datos](#) creada a partir del diagrama Entidad/Relación de la Figura 70) para obtener los retos individuales (si únicamente tiene un solo participante) o los colectivos (si tiene más de un participante dentro del reto) mediante el atributo de SQL COUNT [100, 101].

Como necesitamos separar los retos por el estado de avance para que el usuario sepa, antes de entrar a ver el detalle del reto, si se ha iniciado el reto o no o si ya ha finalizado, en el frontend sepáramos mediante pestañas creadas a través de Bootstrap [93]

(como las mostradas dentro de la *Figura 76*) el conjunto de retos según su estado (*Propuesto*, *En proceso* y *Finalizado*), con el texto indicando el estado (siempre cuando las pantallas son grandes y cuando se esté mostrando el listado de retos con ese estado para pantallas pequeñas) y su correspondiente ícono de *Font Awesome*.



Figura 76: Filtración de retos por estado a través de las pestañas.

Para lograr visualizar los retos por estado correspondiente de la pestaña seleccionada, además de por el tipo elegido previamente, en la parte frontend llamamos al backend a través de HTMX (cuando se pulsa sobre una de las pestañas) indicando el estado que se desea visualizar para realizar la consulta anterior un poco más compleja. En esa consulta realizamos dos filtraciones [102]: una que sea recoger los retos de un estado dado, y la otra que la cantidad de participantes que forme el reto sea la indicada según el tipo de reto elegido (individual solo un participante y colectivo más de un participante).

Si nos fijamos en el conjunto de pestañas de la *Figura 76*, hay otra pestaña más que no entra dentro de los posibles valores del estado de un reto, llamada *Animando*. Esta última pestaña es donde se le permite al usuario visualizar el listado de retos en los que forma parte con el rol de animador.

Para saber qué retos anima este usuario, se realiza de la misma manera que cuando filtramos los retos por estado, excepto que en la consulta no se filtra por el estado sino que se hace una consulta INNER JOIN un poco más compleja para ver si ese usuario es uno de los animadores del reto.

Ya comentadas dos de las principales filtraciones que se pueden hacer sobre una consulta de retos, es momento de hablar de la visualización de los retos en la parte frontend. Como podemos ver en la *Figura 77*, cada uno de los retos que forma parte del listado resultante muestra la siguiente información:

- En la esquina izquierda, mostramos la categoría a la que pertenece el reto a través del texto (si son pantallas grandes) y de un ícono de *Font Awesome*.
- En la parte centro del reto mostramos la siguiente información:
 - En la parte superior, el título del reto (que siempre será texto).

- En la parte inferior, el objetivo del reto (que puede ser un texto, una imagen, un audio o un vídeo).
- En la esquina derecha, damos la posibilidad de ver en detalle el reto mediante un botón con la flecha >.



Figura 77: Ejemplo de un reto del listado.

Con respecto a los objetivos que fueran de tipo audio, decidimos que era mejor transcribir ese audio a texto [103] cuando el usuario pulsara al play para aquellas personas con problemas auditivos, tal y como lo mostramos en la *Figura 78*.



Figura 78: Ejemplo de transcripción de un audio.

Por último, nos queda filtrar los retos mostrados en pantalla por la categoría que deseé el usuario. Por lo tanto, en la parte frontend de la aplicación creamos en la esquina superior derecha de la pantalla, donde se visualiza el listado de retos, una lista de opciones (como la que mostramos en la *Figura 79*) en la que, una vez seleccionada la categoría deseada, llama al backend para realizar las consultas que hemos comentado en las otras filtraciones, pero añadiendo otra más.



Figura 79: Filtración por categoría.

El resultado de la visualización del listado de retos, junto a todas las filtraciones que se puede hacer sobre ellos, podemos ver en la *Figura 80* (que se basó en la primera pantalla de los bocetos de las *Figuras 61, 62 y 63*). En dicha página, permitimos al usuario iniciar la creación de un reto desde 0 a través de la llamada a esa funcionalidad pulsando al

botón *Nuevo reto* (que se encuentra en la esquina izquierda, por encima de las pestañas de los estados de avance de un reto).

Además, al final de dicha página podemos ver que hay un paginador [104] para cada pestaña para así facilitar al usuario ver todos los retos con las filtraciones deseadas sin necesidad de realizar scroll hacia abajo.



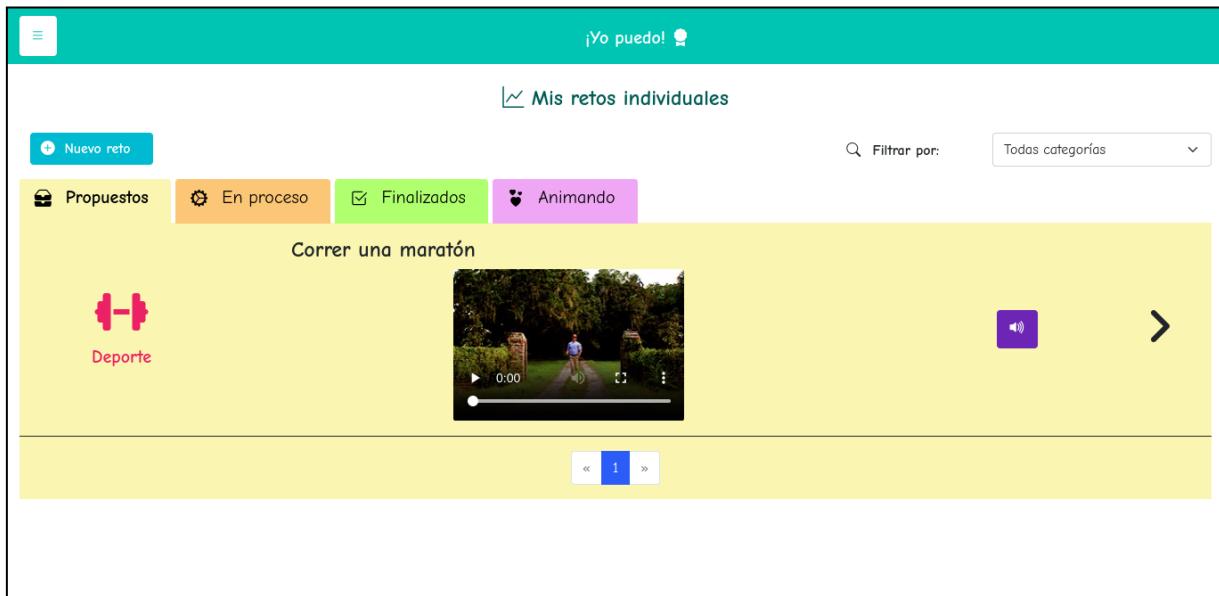
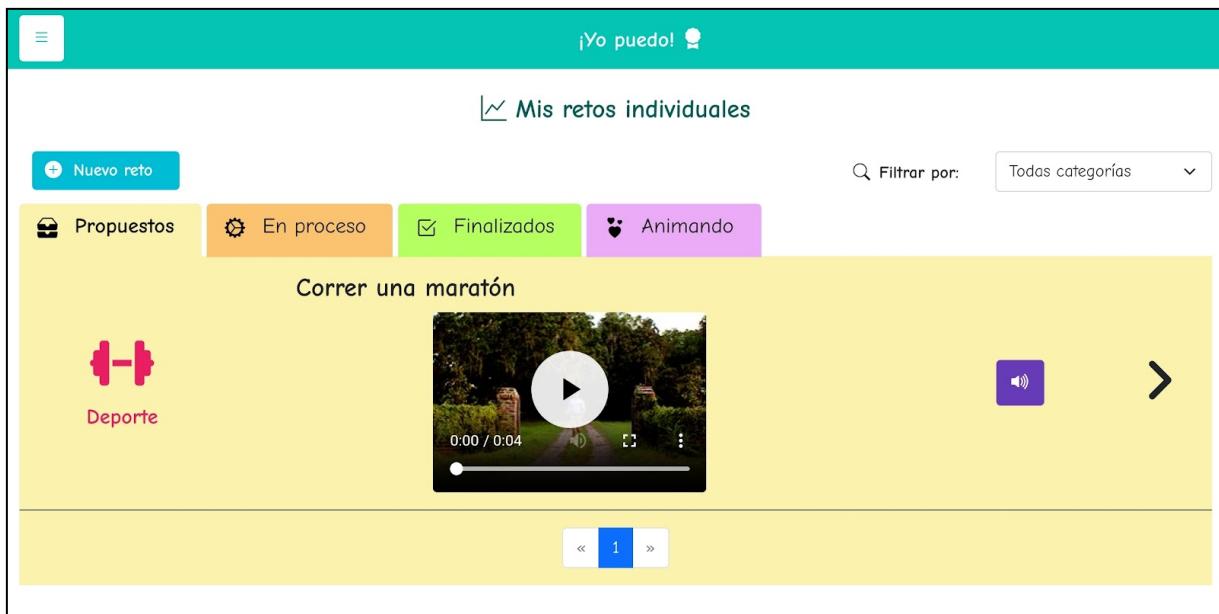


Figura 80: Listado de retos en móvil, tablet y ordenador.

Por último, para poder acceder a esta pantalla y las que se hagan en las siguientes iteraciones, decidimos realizar un menú [105] con las principales funcionalidades de la aplicación junto a iconos que describan qué es lo que puede hacer en ellas (mis retos, mis amigos, mi perfil y notificaciones) como el de la *Figura 81* (creado a partir de la primera pantalla de los bocetos de las *Figuras 61 y 62*).

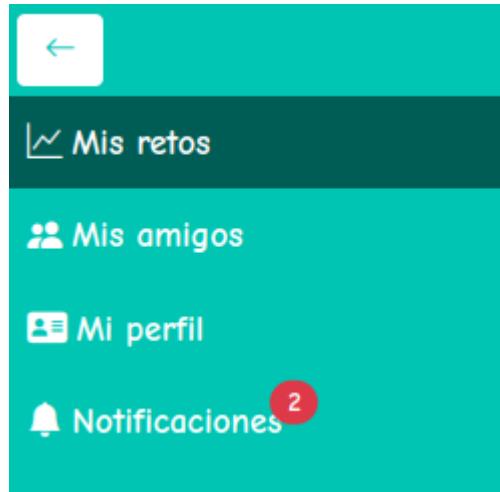


Figura 81: Menú de la aplicación.

Visualización de retos

Una vez creadas las funcionalidades de crear un nuevo reto individual y de crear el acceso al listado de retos, junto a sus filtraciones, que puede tener un usuario, es momento de ver la información detalladamente de uno de esos retos. Como ya dijimos en el diseño de los bocetos, cada rol debe tener una vista distinta. Por lo tanto ahora mostraremos la parte que todos los usuarios deberían ver.

Ya recogida la distinta información del reto realizando las consultas INNER JOIN [99] necesarias dentro del backend, mostramos dichos datos en el frontend de la aplicación. Para el caso de las consultas de los animadores y participantes que forman parte del reto, devolvemos aquellos que no sean el propio usuario que está visualizando el reto []. Por lo tanto, la estructura de la visualización de los datos obtenidos de las diversas consultas es similar a la de creación de retos:

Primero empezamos mostrando los detalles generales del reto (objetivo, categoría y recompensa) con su título y su ícono descriptivo, tal y como la mostramos en la *Figura 82* y las diseñamos en la primera ventana de los bocetos de las *Figuras 58, 59 y 60*. Para aquellos objetivos y recompensas que sean de tipo audio, se implementa la misma transcripción [103] que hicimos en la *Figura 78*, y para los que sean de tipo texto, creamos el botón para que se lo dicte al usuario.

≡

¡Yo puedo! 🏃

↗ Correr una maratón

General

Objetivo:

Categoría: Deporte

Recompensa: Viajar por el mundo en busca de maratones

Editor Eliminar Siguiente >

≡

¡Yo puedo! 🏃

↗ Correr una maratón

General

Objetivo:

Categoría: Deporte

Recompensa: Viajar por el mundo en busca de maratones

Editor Eliminar Siguiente >

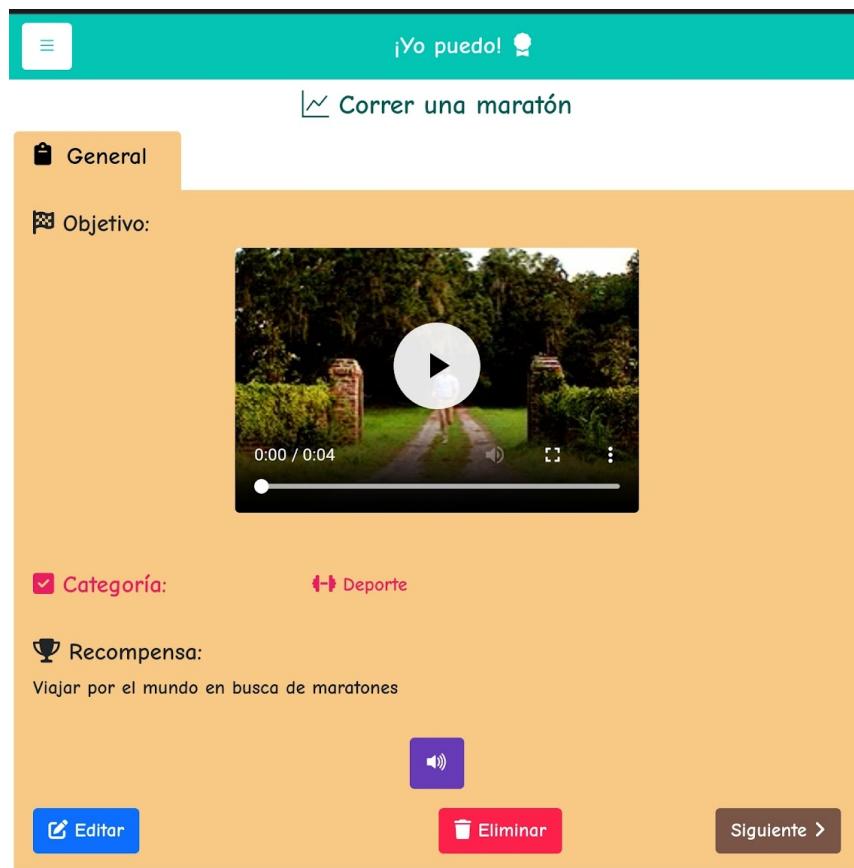


Figura 82: Ejemplo de visualización de la pestaña general en ordenador, móvil y tablet.

En cuanto a las etapas, como ya hicimos en la creación del reto y como las planteamos en la segunda pantalla de los bocetos de las Figuras 58, 59 y 60, fueron divididas en pestañas donde cada una de ellas se visualiza su objetivo y un botón para navegar entre ellas.





Figura 83: Ejemplo de visualización de la pestaña de etapas en ordenador, móvil y tablet.

Después de mostrar la información del reto, mostramos al usuario qué otras personas forman parte del reto con el rol de participante (*Figura 84*) a través de una foto de perfil y su nombre, diseñado a partir de la tercera pantalla de los bocetos de las *Figuras 58, 59 y 60*.

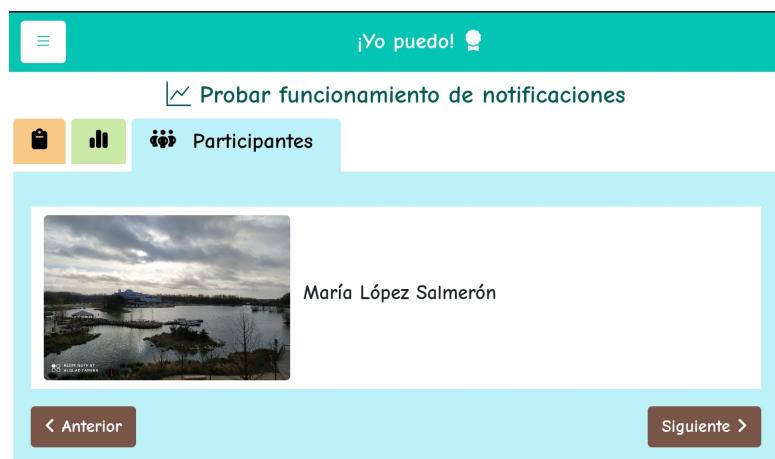
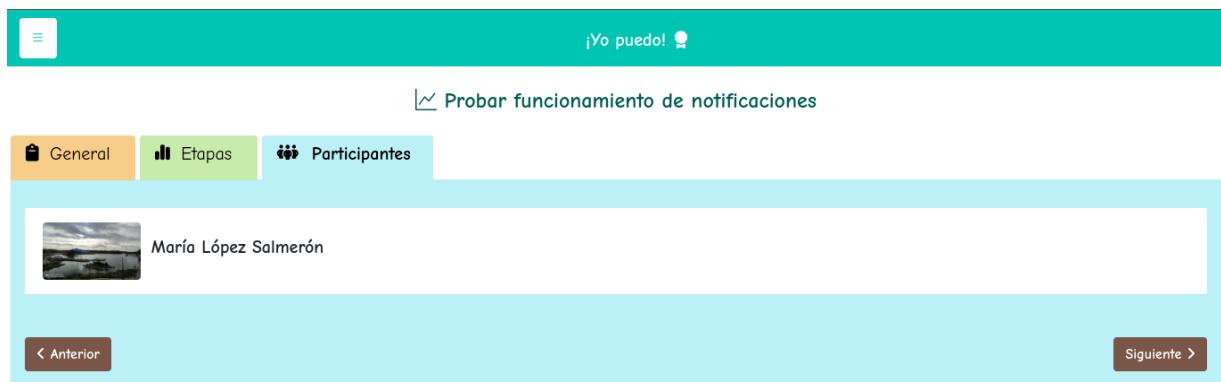


Figura 84: Ejemplo de visualización de participantes de un reto en ordenador, móvil y tablet.

Por último, enviamos a la parte del frontend de la aplicación la lista de animadores que forman parte del reto dado, junto a su foto de perfil, su nombre y si tienen o no permiso para ver todos los mensajes de ánimo guardados (si es un superanimador o no) dentro del reto (a través del botón azul de la parte derecha de cada fila de animadores indicando

mediante un ojo tachado si no es un superanimador y mediante un ojo solo si es un superanimador). Esta pantalla (las que mostramos de distintos dispositivos en la *Figura 85*) fue implementada a partir del diseño de la última ventana de los bocetos de las *Figuras 58, 59 y 60*.

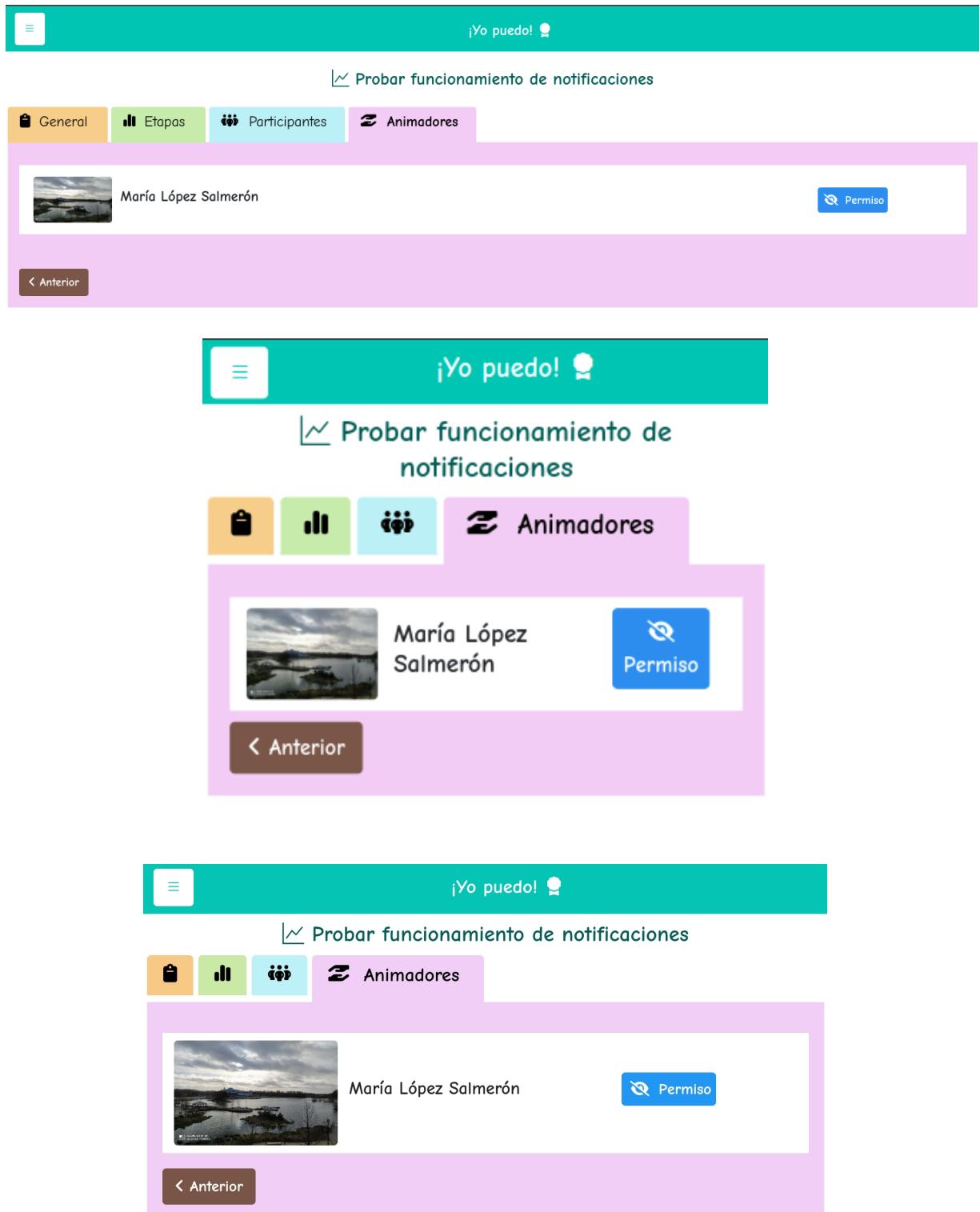


Figura 85: Ejemplo de visualización de animadores de un reto en ordenador, móvil y tablet.

Pruebas

Con respecto a las pruebas que implementamos en esta iteración, las dividiremos en dos grandes bloques:

- La parte frontend de la aplicación, en la que, durante el diseño de los bocetos, comprobamos que se cumplían con las distintas propiedades de las guías de accesibilidad (como se explicará detalladamente en el [ANEXO 2](#) en la forma en la que se cumplió).
- La parte backend de las historias de usuario, revisando que realiza lo esperado el controlador y se almacena en la base de datos lo deseado:
 - Comprobamos que únicamente pueden acceder a estas funcionalidades aquellos usuarios que hayan iniciado sesión. Los que no, la aplicación los redirige a la página de registro.
 - Antes de crear un reto, revisamos que se cumplen las siguientes validaciones:
 - Debe tener un título, que se puede introducir escribiendo o dictando su contenido.
 - Debe tener un objetivo, que se puede insertar a través de texto (escrito o hablado), imagen, audio o vídeo.
 - Debe tener una recompensa, que se puede insertar a través de texto (escrito o hablado), imagen, audio o vídeo.
 - Debe tener una categoría, de las indicadas por la aplicación (ahorro, conocimientos, miedo y deporte).
 - Debe tener al menos una etapa y máximo 5.
 - Cada etapa debe tener un objetivo, que se puede introducir a través de texto (escrito o hablado), imagen, audio o vídeo.
 - Debe tener un coordinador y un participante, que son la misma persona.
 - En cuanto a la base de datos, verificamos que cuando creamos un reto (junto a sus etapas, participantes y animadores) se guarda en la base de datos y,

cuando deseamos visualizar, se puede acceder a la información correctamente.

- Con respecto al listado de retos, examinamos que se muestran los retos esperados según las categorías, estados y tipos dados.
- La visualización de un reto únicamente las puede ver un participante, el coordinador o un animador del reto, si no es uno de esos roles o no existe el reto, devuelve mensaje de error.

Retrospectiva

En cuanto a las funcionalidades completadas durante este sprint, se han realizado todas las planificadas, aunque se tardó un poco más de lo esperado en la parte de crear etapas dentro de un reto puesto que hubo ciertos problemas para que comprobara que cada una de las etapa creadas cumplían las validaciones impuestas sobre ellas.

Sin embargo, durante la realización de algunas de las funcionalidades de este sprint nos dimos cuenta que la visualización de un reto debe depender del rol del usuario que lo está viendo, por lo que en futuras iteraciones añadiremos dentro del reto las siguientes características:

- Si es un participante, puede ver y añadir evidencias y calificaciones en las etapas. Esta funcionalidad la llevaremos a cabo en el siguiente sprint.
- Si es un animador, puede escribir y ver los mensajes de apoyo (los suyos sí es un animador normal y todos si es un superanimador) dentro de una etapa. Además, podrá dejar de formar parte del reto si lo desea. Estas funcionalidades las efectuaremos en el sexto sprint.
- Si es un participante, verá todos los mensajes de ánimo de una etapa dada. Las haremos en el sexto sprint del proyecto.
- Si es un coordinador, puede editar, eliminar e iniciar el reto y añadir participantes, animadores y superanimadores una vez creado el reto. Estas funcionalidades las realizaremos en el séptimo sprint.

3.10. Cuarto sprint

En cuanto a los casos realizados durante el cuarto sprint, son aquellos que están relacionados con el avance de los retos del usuario, como la inserción de evidencias, la consulta de evidencias, la calificación de etapas, y la consulta de la calificación de etapas.

Historias de Usuario

- **[HU13] Como participante, quiero añadir una prueba de mi avance de un reto.**
 - Quién: Participante.
 - Cuándo: Cuando quiera indicar lo que se ha progresado durante la realización del reto.
 - Entonces: Se añadirá la prueba en la etapa correspondiente del reto.
 - CoS:
 - El usuario debe haber iniciado sesión en el sistema.
 - El usuario debe ser uno de los participantes del reto.
 - El identificador (que se generará de manera automática) es la clave primaria.
 - La prueba puede ser de los siguientes tipos de formatos: vídeo, texto, audio o imagen.
 - Se guardará la prueba indicada, el usuario que la añadió, en la etapa en la que se realizó y su identificador.
 - En aquellas pruebas que no sean textuales, sino que se haya adjuntado un fichero (de imagen, vídeo o audio), se guardará la ubicación del fichero dado y almacenado dentro del sistema.
 - Únicamente se podrán insertar las pruebas en los retos que estén en estado “*En proceso*”.
 - Solamente se añadirán las pruebas en las etapas que estén en estado “*En proceso*”.
 - Cada elemento del formulario tendrá su correspondiente título e ícono que indique la información que debe añadir el usuario.

- La prueba se insertará cuando se esté visualizando la información de la etapa dada.
- **[HU14] Como participante, quiero ver las pruebas de una de las etapas de uno de mis retos.**
 - Quién: Participante.
 - Cuándo: Cuando quiera ver lo que ha progresado durante la realización del reto.
 - Entonces: Se muestran un conjunto de pruebas de la correspondiente etapa.
 - CoS:
 - El usuario debe haber iniciado sesión en el sistema.
 - El usuario debe ser uno de los participantes del reto.
 - Se devuelve información de todas las pruebas almacenadas en esa etapa.
 - Se muestran las pruebas dentro de la etapa seleccionada del reto cuando se visualice la información de dicho reto.
- **[HU15] Como participante, quiero calificar cómo me he sentido realizando una etapa de uno de mis retos.**
 - Quién: Participante.
 - Cuándo: Antes de que se indique el final de una de las etapas del reto.
 - Entonces: Se guarda la calificación de cómo le ha ido en esa etapa y se marca esa etapa como finalizada y la siguiente a esta en proceso, si todos los participantes la han calificado. Si era la última etapa, se marca el reto como finalizado.
 - CoS:
 - El usuario debe haber iniciado sesión previamente.
 - Solo pueden calificar los participantes del reto.
 - Solo pueden ser calificadas aquellas etapas que no estén en estado “Propuesto”.

- La calificación se realiza a través de la selección del emoticono que mejor defina cómo le haya ido al participante en la etapa.
- La calificación se realizará cuando se esté mostrando la información de la etapa dada.
- Hay 3 tipos de calificaciones: mal, normal y muy bien.
- Si se selecciona uno de los emoticonos, se guarda en la base de datos el participante que lo realizó, la calificación y la etapa.
- Cuando se guarde la calificación se realiza lo siguiente:
 - Primero, se comprueba que todos los participantes del reto han calificado. Si es así, el estado de la etapa calificada pasa a “Finalizado”.
 - Segundo, se mira si hay más etapas. Si queda aún alguna etapa por realizar, pasamos la siguiente etapa a estado “En proceso”. Si era la última etapa que quedaba por hacer, pasamos el estado del reto a “Finalizado”.
- [HU16] **Como participante, quiero ver mi calificación de una etapa.**
 - Quién: Participante.
 - Cuándo: Cuando desee ver mi calificación de una etapa de un reto concreto.
 - Entonces: Se muestra la calificación de esa etapa.
 - CoS:
 - El usuario debe haber iniciado sesión previamente.
 - Tiene que ser un usuario que participe en ese reto.
 - Se muestran los tres emoticonos que forman parte de la calificación, marcando el guardado de esa persona en esa etapa.

Bocetos

Como hemos indicado en los inconvenientes de los bocetos de la visualización de un reto en el tercer sprint, vimos que es importante separar la visualización que tendría un

participante de la de un animador añadiendo distintas acciones y elementos en ella. Por ello, vamos a mostrar qué es lo que vería si dicho usuario fuera un participante del reto.

Con respecto a la pantalla que muestra la información de un reto ya creado para aquellos usuarios que sean participantes del reto (*Figura 86* para la versión de ordenador, *Figura 87* para tablets y *Figura 88* para formato móvil), mostramos toda la información que podría tener un reto (información general, sus etapas, los animadores y los participantes, excepto él mismo, que estén incluidos dentro del reto), aunque este no sea colectivo o no tenga animadores en él.

A diferencia del boceto creado en el tercer sprint, en cada subpestaña (o mejor dicho, en cada etapa) que encontramos en la pestaña de “Etapas”, además de mostrar el objetivo de una etapa, podemos insertar (si esa etapa está en estado “En proceso”) y ver las pruebas que se quieran evidenciar en ella y los mensajes de apoyo que hayan dejado los animadores a través de otras dos pestañas (para así cumplir con la propiedad de navegación, como explicamos en el [ANEXO 2](#)) y, al final de la etapa, la calificación de cómo se ha sentido esa persona realizando esa etapa mediante la elección de uno de los tres iconos, tal y como mostramos en las *Figuras 86, 87 y 88*.

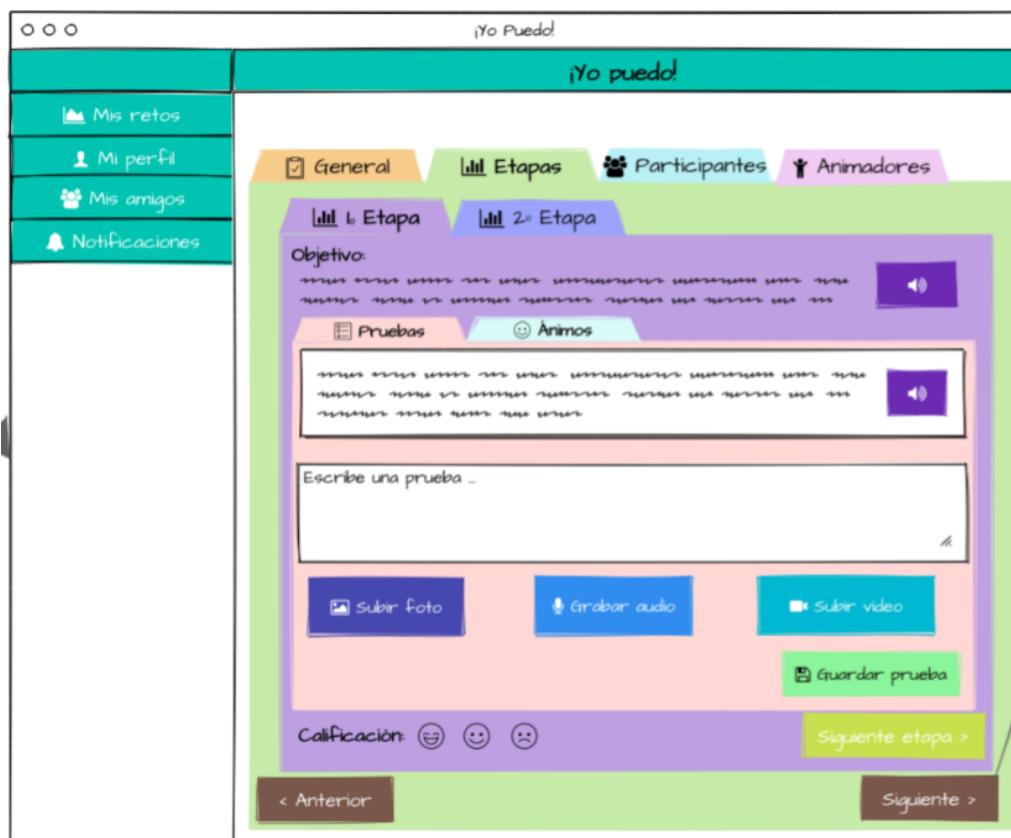


Figura 86: Segunda versión de la visualización de un reto desde la vista de un participante para ordenadores.

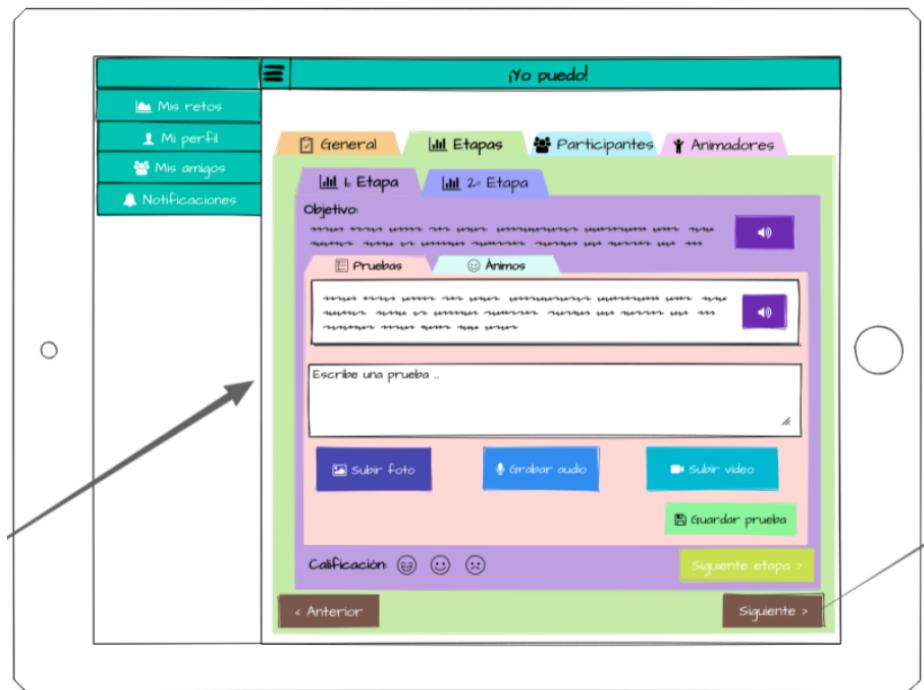


Figura 87: Segunda versión de la visualización de un reto desde la vista de un participante en tablets.

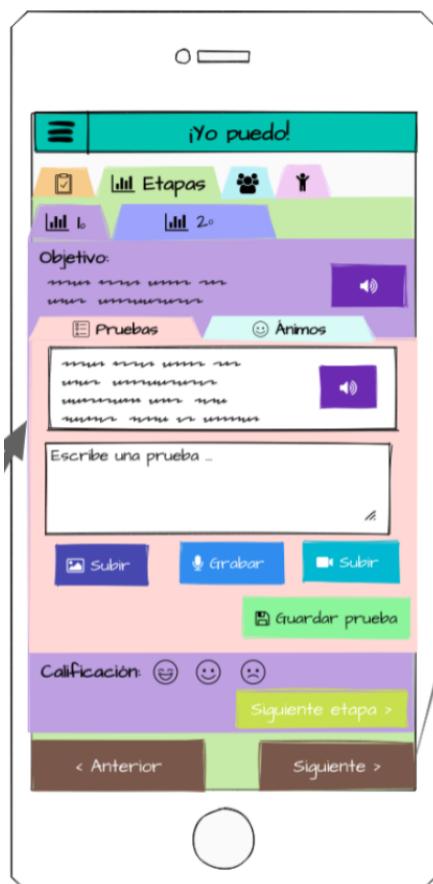


Figura 88: Segunda versión de la visualización de un reto desde la vista de un participante en formato móvil.

Aún estando bastante más completa que la primera versión, seguimos encontrando cosas a mejorar tras una reunión con el cliente y una revisión de las guías de accesibilidad en los diseños:

- Falta un mecanismo que permita al usuario poder eliminar el reto.
- Echamos en falta una funcionalidad que permita al usuario editar la información del reto.
- Necesitamos diferenciar entre participante y coordinador del reto para que sea el coordinador la persona que tenga los permisos de borrar y editar el reto colectivo. Por defecto, el coordinador será la persona que cree el reto.
- Si añadimos a los diseños de los retos colectivos un coordinador, debemos permitir al coordinador dejar de serlo y pasarle los permisos a otro participante del reto.

Por lo que, además de centrarnos en la inserción de participantes y en la creación de retos colectivos en el séptimo sprint, llevaremos a cabo todas las actividades que acabamos de comentar que echamos en falta al diseñar estos bocetos.

Base de datos

Para poder implementar los datos necesarios y las relaciones entre ellos en este sprint, hemos diseñado la Entidad/Relación mostrada en el diagrama de la *Figura 89*. En él, no prestamos atención a las relaciones *Coordina* y *Anima* puesto que únicamente estas historias hablan de acciones que realiza un participante sobre uno de sus retos.

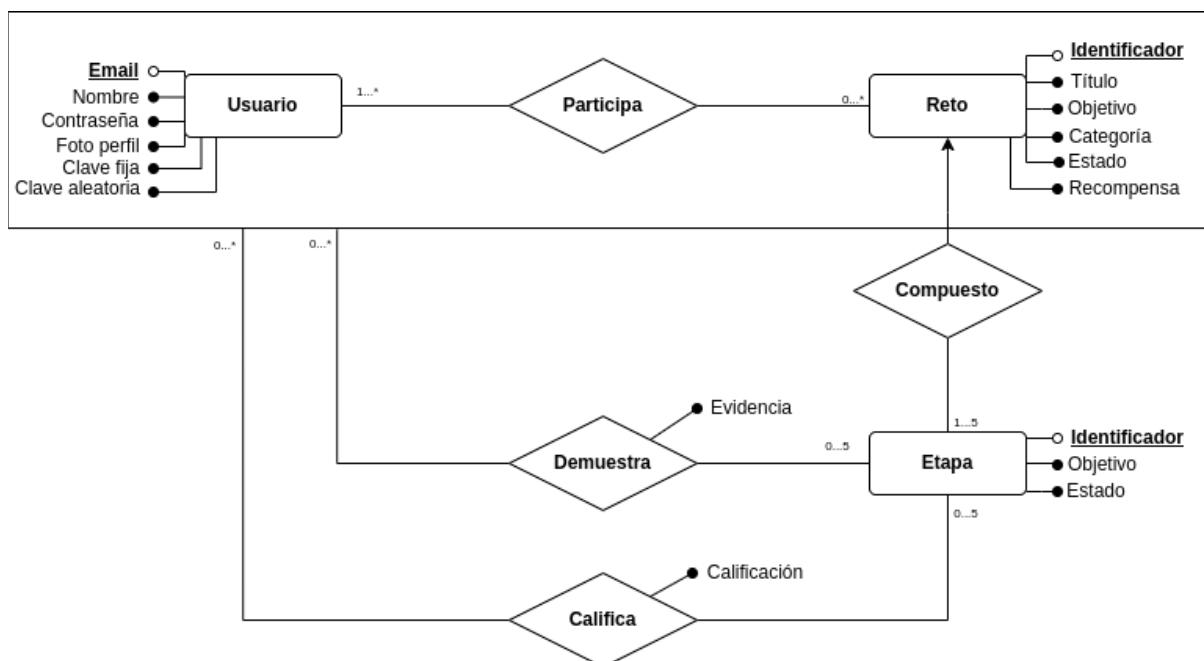


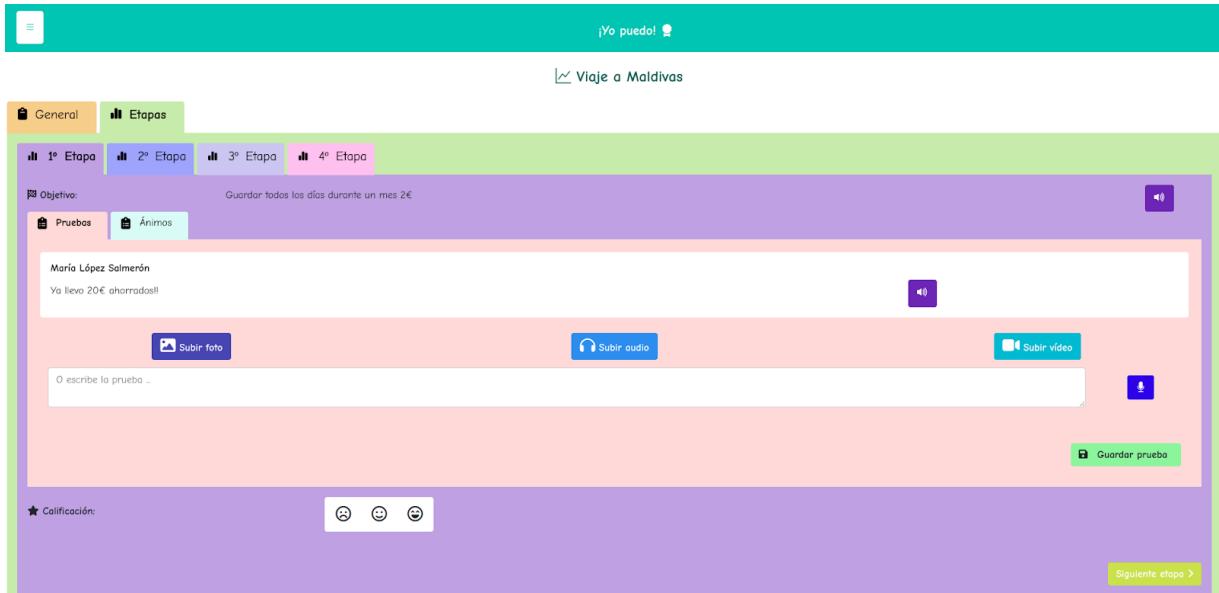
Figura 89: Diagrama de Entidad/Relación para la implementación del cuarto sprint.

Como dichas historias de usuario son acciones que un participante puede realizar sobre una de las etapas de un reto, necesitamos tener una agregación con la relación *Participa*. Generamos dos tipos de relaciones con la agregación anterior:

- *Demuestra*, que se encarga de guardar las evidencias de un participante a su paso por una etapa de un reto dado.
- *Califica*, que se encarga de guardar el grado de satisfacción de un participante al realizar una etapa de un reto dado.

Implementación

En cuanto a la implementación de este sprint, vamos a explicar las dos acciones que únicamente puede hacer un participante: evidenciar la realización de una de las etapas del reto y calificar la etapa una vez finalizada. Como ya digimon en los [bocetos](#) de esa iteración, las evidencias y las calificaciones se van a realizar dentro de una etapa (exactamente en las Figuras 86, 87 y 88), por lo que ambos se mostrarán en la misma página, como se muestra en la Figura 90.



☰

¡Yo puedo! 🧑

↗ Viaje a Maldivas

กระเป๋า Etapas

☒ Objetivo:

Guardar todos los días durante un mes 2€

🔊

秬 Pruebas 🗂️

María López Salmerón
Ya llevo 20€ ahorrados!!

🔊

📷 🎵 🎥

O escribe la prueba ...

🎙

🔗

★ Calificación:

😢 😊 😃

Siguiente etapa >

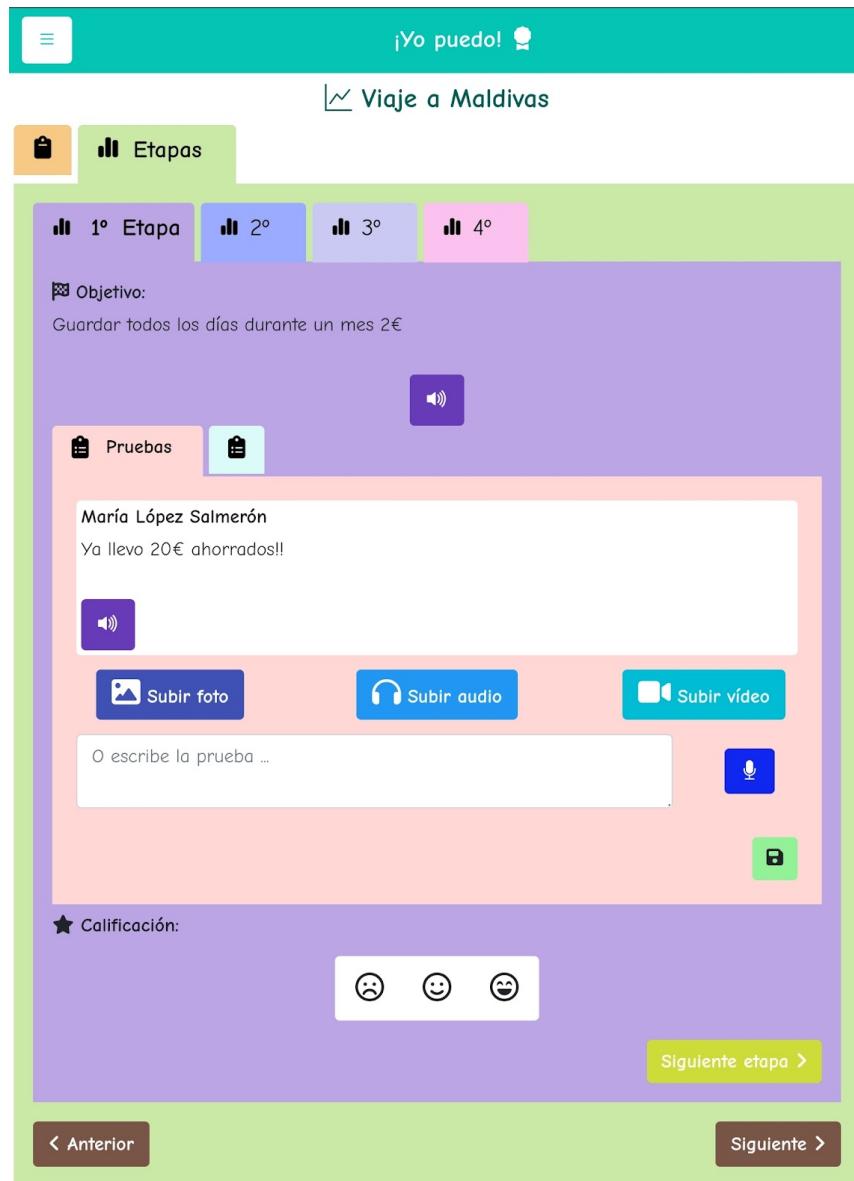


Figura 90: Visualización de una etapa si el usuario es un participante del reto para móvil, ordenador y tablet.

En primer lugar, creamos una pestaña, dentro de la pestaña que contiene el objetivo de una de las etapas, en la que el usuario podrá ver y/u oír las evidencias suyas y las de otros participantes, además de insertar nuevas evidencias a través de un formulario (ya sea a través de escribir o dictar el texto o de subir una imagen, audio o vídeo, como ya hicimos, por ejemplo, a la hora de introducir el objetivo de una etapa en la creación del reto). Cuando un usuario desea subir una evidencia a la etapa, se guarda en la relación *Demuestra* (del diagrama de la Figura 89) la evidencia, junto al participante que lo realizó y en la etapa que lo hizo.

Por último, para indicar que se ha terminado con esa etapa y que estamos listos de pasar a la siguiente (o de indicar a la aplicación que ya hemos terminado si es la última) es necesario calificarla previamente. Como ya comentamos de hacerla en el paso de diseño de frontend, el usuario le puede resultar más fácil calificar la etapa a través de la elección de uno de los iconos de *Font Awesome*, que lo que haremos a través de Javascript es [107] traducir el ícono por texto (cara triste - mal, cara alegre - normal, cara súper alegre - muy bien) y llamar al backend con ese dato para que lo guarde en la base de datos, dentro de la relación *Califica*. Una vez guardada, marcamos con negrita la opción seleccionada.

Pruebas

En relación a las pruebas realizadas en esta iteración, para comprobar que cada una de las funcionalidades funciona como lo esperado, las dividimos en dos bloques:

- La parte del frontend de la aplicación, mediante los bocetos realizados verificamos que se cumplían con las cuatro propiedades principales de las [guías de accesibilidad](#), tal y como las comentaremos detalladamente en el [ANEXO 2](#).
- La parte del backend de la aplicación, las cuales comprobamos que hiciera correctamente estas acciones:
 - No permitir ver ni enviar, tanto las evidencias como las calificaciones, a personas que no participan en el reto.
 - Cuando se guarda una evidencia o una calificación, miramos que esos datos se han guardado correctamente y que son totalmente accesibles.

Retrospectiva

Con respecto a la retrospectiva de este sprint, podemos indicar que realizamos todas las funcionalidades planificadas para esta iteración en menos tiempo de lo esperado. Por lo que comenzamos el siguiente sprint antes de lo esperado.

Además, diseñando estas historias de usuario, nos dimos cuenta de que eran necesario implementar las siguientes acciones dentro de un reto: editar o eliminar un reto, añadir el rol de coordinador (junto a su propia visualización y funcionalidades que únicamente puede realizar él, como editar o eliminar el reto) y tener la posibilidad de cambiar de coordinador con uno de los participantes del reto.

3.11. Quinto sprint

A lo largo de este sprint realizamos aquellas funcionalidades que estén relacionadas con los amigos: pedir, aceptar o rechazar solicitudes de amistad, buscar nuevos amigos, ver listado de amigos, dejar de seguir y ver perfil del amigo.

Historias de Usuario

- **[HU17] Como persona, necesito pedir una solicitud de amistad a un amigo.**
 - Quién: Persona.
 - Cuándo: Cuando quiera incluir amigos en su cuenta.
 - Entonces: Se notifica al usuario destinatario de la petición de amistad del usuario solicitante.
 - CoS:
 - La persona debe haber iniciado sesión en el sistema.
 - Se enviará una notificación al usuario seleccionado preguntando si quiere añadir esa persona en su lista de amigos.
- **[HU18] Como persona, necesito aceptar o rechazar una solicitud de amistad.**
 - Quién: Persona
 - Cuándo: Cuando el usuario haya recibido una solicitud de amistad.
 - Entonces: Si se acepta, se crea una nueva amistad, con lo cual esta persona me empezará a seguir. Sino, se elimina la solicitud de amistad.
 - CoS:
 - El usuario debe haber iniciado sesión.
 - Debe tener una solicitud de amistad previamente el usuario.
 - Si se rechaza la solicitud de amistad, se elimina dicha solicitud.
 - Si se acepta la solicitud, se realiza lo siguiente:
 - Se envía a ese usuario una clave aleatoria a su correo electrónico para confirmar la aceptación de dicha solicitud.

- Se introduce la clave mandada en la aplicación.
 - Si la clave introducida es la misma que la generada por el sistema, se crea una nueva amistad entre el usuario que ha aceptado la solicitud y el usuario que la envió.
- **[HU19] Como persona, necesito buscar nuevos amigos.**
 - Quién: Persona.
 - Cuándo: Cuando quiera incluir amigos en su cuenta.
 - Entonces: Se muestra una lista de usuarios según los resultados de su consulta.
 - CoS:
 - El usuario debe, previamente, haber iniciado sesión en el sistema.
 - La consulta se realizará sobre el listado de usuarios que tiene el sistema, excluyendo la lista de amigos que tenga el usuario, buscando por el nombre o por el correo electrónico.
 - La consulta se podrá hacer escribiendo o dictando el texto deseado.
 - Se devuelve una lista de usuarios en la que se muestra la foto de perfil y el nombre.
 - Se realiza una paginación con 3 amigos en cada página como máximo.
- **[HU20] Como persona, necesito consultar la lista de mis amigos.**
 - Quién: Persona.
 - Cuándo: Cuando quiera consultar su lista de amigos.
 - Entonces: Se muestra una lista de usuarios que sean amigos de esa persona.
 - CoS:
 - El usuario debe haber iniciado sesión en el sistema antes de consultar la lista.

- Los usuarios devueltos deben ser los amigos de la persona que realiza la consulta.
 - Se devuelve una lista de usuarios, mostrando su foto de perfil y su nombre.
 - Se realiza una paginación con 5 amigos en cada página como máximo.
- **[HU21] Como persona, quiero dejar de seguir a un amigo.**
 - Quién: Persona.
 - Cuándo: Cuando no quiere seguir siendo amigo de un usuario.
 - Entonces: Se eliminan todas las relaciones dentro del sistema con esa persona.
 - CoS:
 - El usuario debe haber iniciado sesión previamente en el sistema.
 - Se elimina la amistad que hay entre los dos usuarios.
 - **[HU22] Como persona, me gustaría ver el perfil de uno de mis amigos.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ver, con detalle, el perfil de uno de sus amigos.
 - Entonces: Se muestran los datos de ese amigo más los retos que tienen en común.
 - CoS:
 - El usuario debe haber iniciado sesión.
 - El amigo que se quiera consultar y el usuario deben tener una amistad previamente a la consulta.
 - Se devuelve los datos del amigo que se quiera consultar: nombre, correo electrónico y foto de perfil.
 - Se devuelve los retos en los que estén el usuario y el amigo consultado como participante y como animador.

- Los retos que hay en común se paginará en 3 retos como máximo por página.

Bocetos

Como comentamos al principio de este documento, durante la motivación que nos llevó a llevar a cabo este proyecto, necesitamos realizar la inserción de participantes y de animadores a los retos. Para ello es necesario que esas personas formen parte del grupo de amigos del creador o del coordinador del reto. Por lo tanto, el usuario debe poder ver los amigos que tiene en su cuenta, además de añadir nuevos amigos y eliminar aquellos con los que no quiera seguir teniendo contacto. Todas esas funcionalidades están dentro de las Figuras 91, 92 y 93 para distintos tipos de dispositivos.

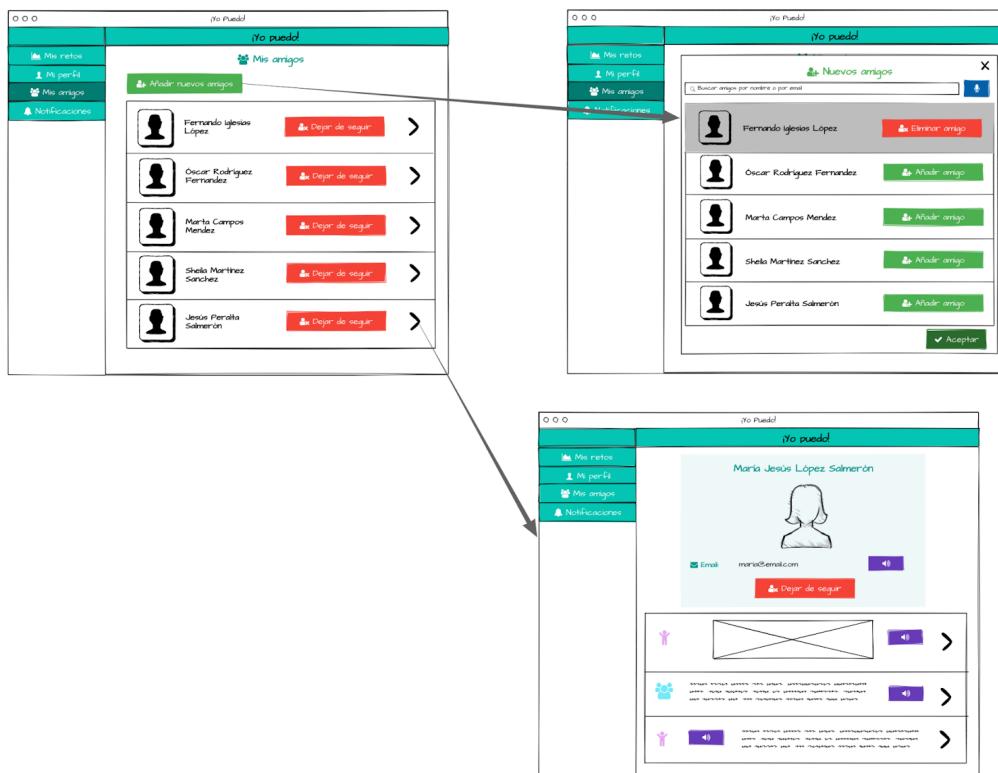


Figura 91: Listado de amigos de un usuario y visualización del perfil desde el escritorio.

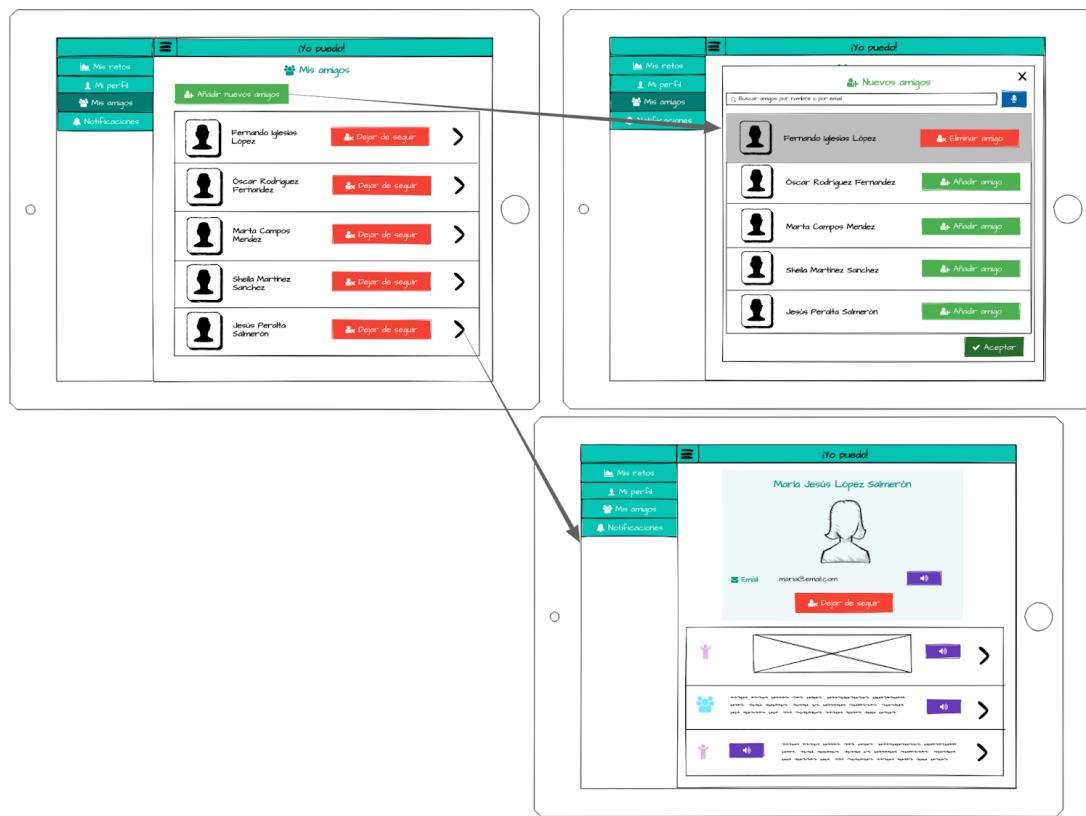


Figura 92: Listado de amigos y visualización del perfil para tablets.

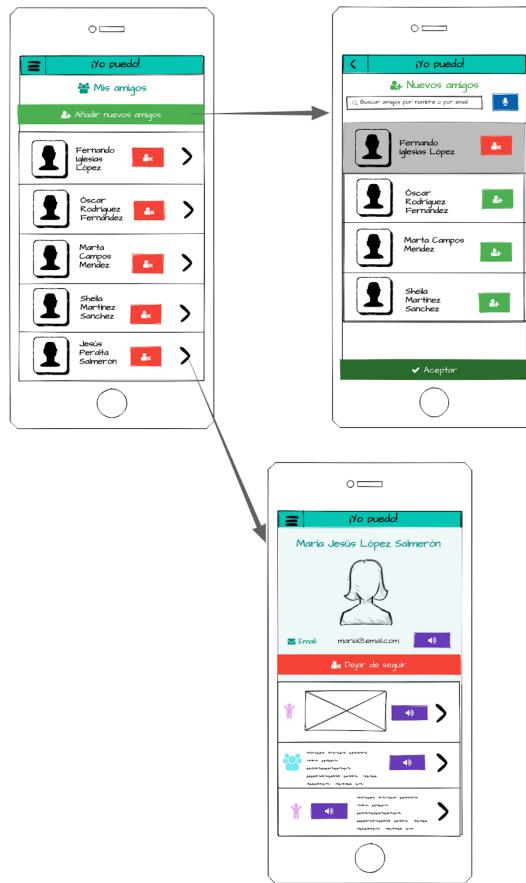


Figura 93: Listado de amigos y visualización del perfil de uno de ellos en formato móvil.

En la ventana principal de la opción “Mis amigos” del menú principal de la aplicación (osease, la primera pantalla de las *Figuras 91, 92 y 93*) se muestra el listado de amigos que tiene el usuario. Para cada usuario amigo se ofrecen dos funcionalidades, eliminarlo de la lista o ver su perfil.

En el perfil de un amigo (pantalla derecha inferior de las *Figuras 91, 92 y 93*), donde mostramos al usuario todos los datos necesarios para identificar a otro usuario, informamos al usuario del nombre y el correo de su amigo y mostramos por pantalla una lista de retos que tienen en común (ya sea porque el amigo o él están animando en un mismo reto, o porque formen parte de un reto colectivo como participantes). También, el usuario puede decirle al sistema su deseo de dejar de seguir a esa persona pulsando a su botón correspondiente dentro del perfil del amigo.

Para poder ver a mis amigos, primero debo buscarlos y añadirlos a mi cuenta, por lo que en la pantalla principal de las *Figuras 91, 92 y 93* encontramos un botón, por la esquina izquierda y encima de la lista de amigos, que nos permite añadir a nuevos amigos. Cuando se pulsa ese botón te lleva a una ventana (a una pequeña encima de la lista de amigos para pantallas grandes y a una a parte para móviles) en la que se puede buscar a un usuario por su nombre o por su email y seleccionar uno o varios de los usuarios resultado de la búsqueda, correspondientes a la pantalla derecha superior de las figuras mencionadas anteriormente en este párrafo.

La única pega que le ponemos a estos diseños (identificadas en la reunión tenida con el cliente) es el símbolo cogido para indicar al usuario para dejar de seguir a una persona, en vez de utilizar a una persona con una cruz elegiremos la papelera, como se muestran en las siguientes figuras:

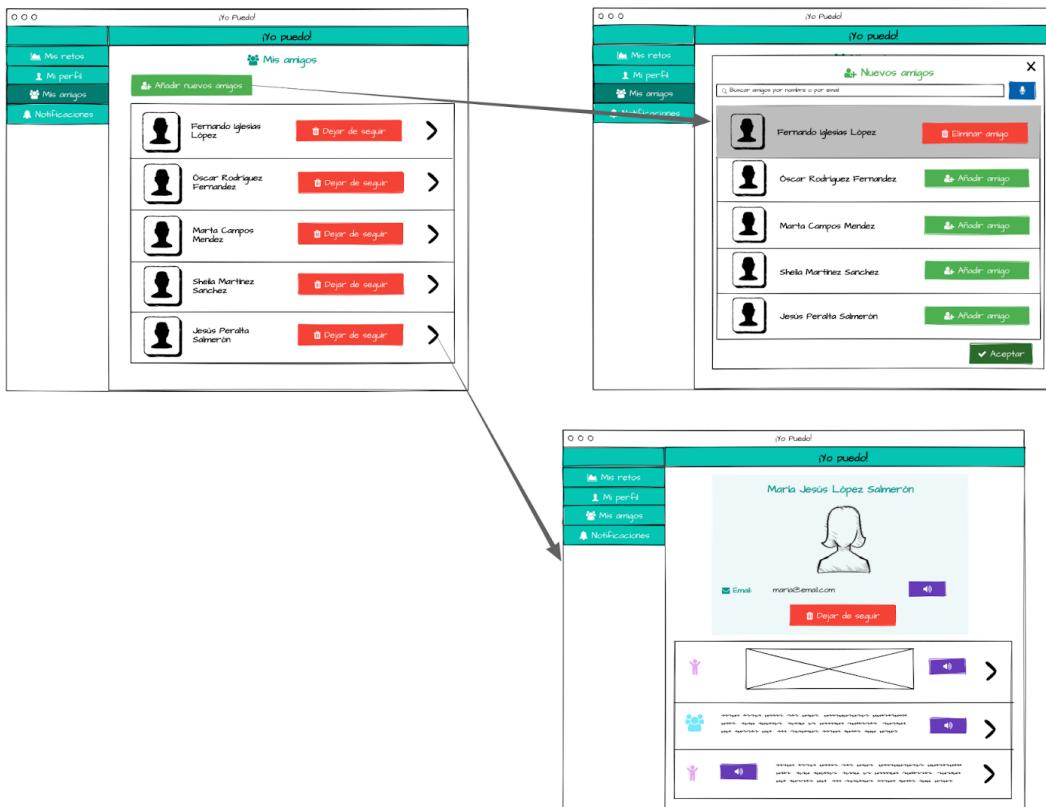


Figura 94: Segunda versión “Mis amigos” en formato escritorio.

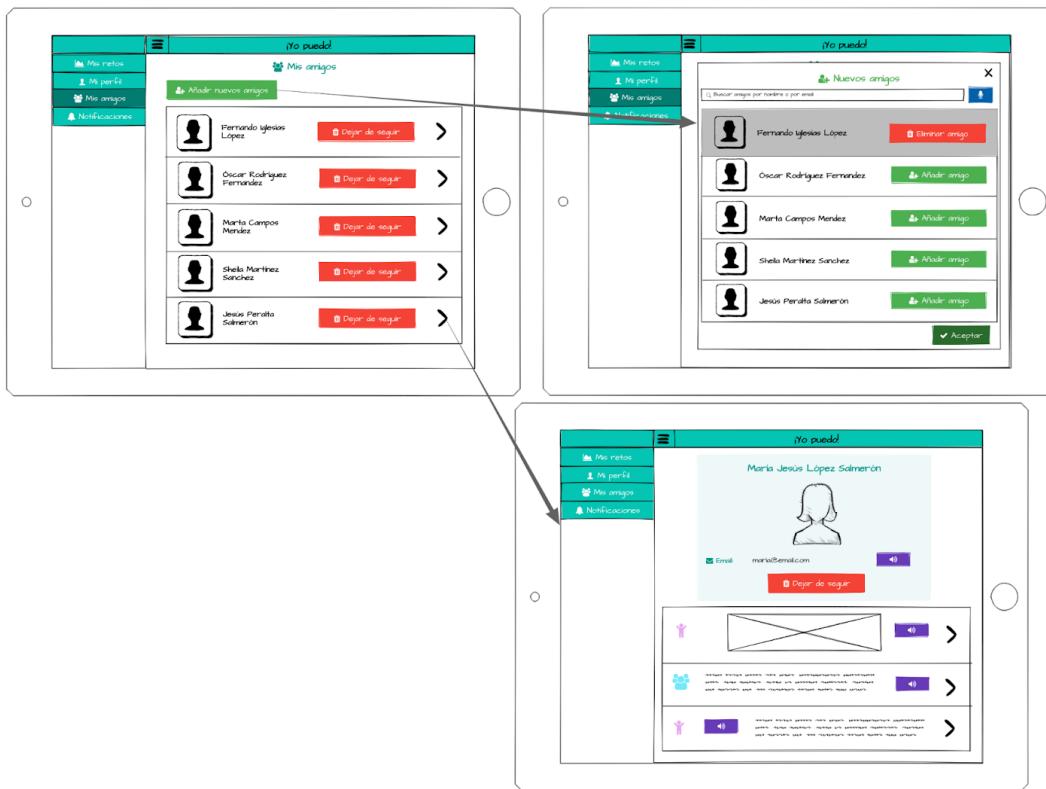


Figura 95: Segunda versión de “Mis amigos” para tablets.

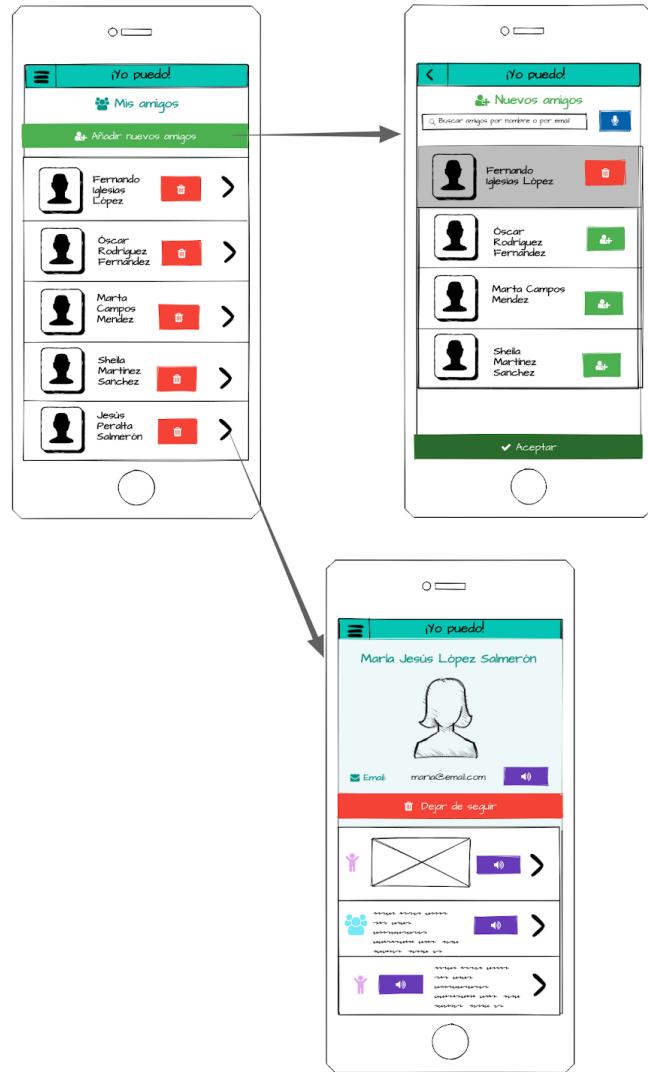


Figura 96: Segunda versión de “Mis amigos” en móvil.

Base de datos

En este sprint únicamente nos centraremos en la relación que necesita la entidad *Usuario* para la realización de las historias de usuario anteriores al estar relacionadas con la amistades que puedan tener los usuarios del sistema. Para ello, únicamente necesitamos una relación recursiva con la entidad *Usuario*, a la que llamaremos *Amigo*.

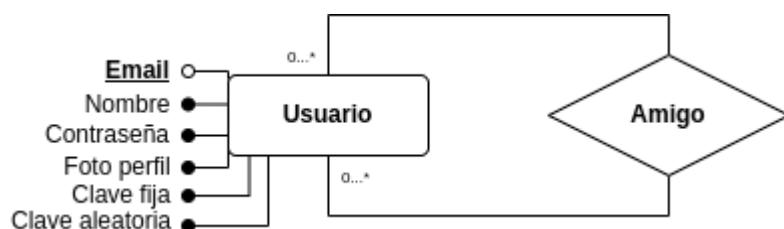


Figura 97: Diagrama Entidad/Relación para las historias de usuario del quinto sprint.

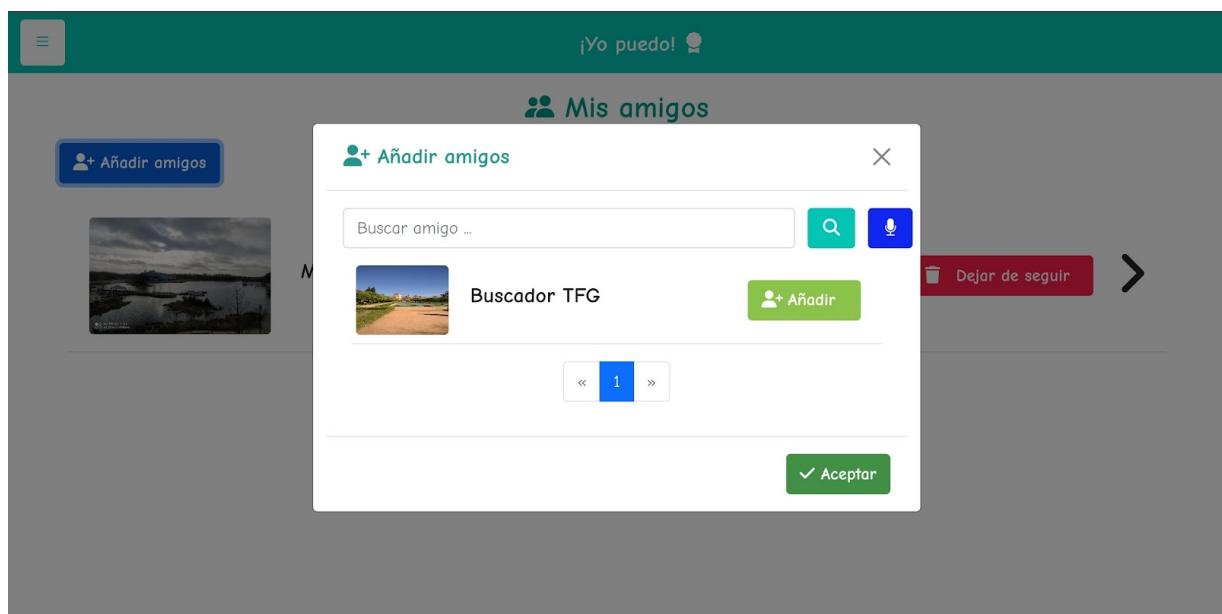
Implementación

Acerca de la implementación llevada a cabo en este sprint, decidimos dividir las explicaciones en buscador de nuevos amigos y la petición de amistad, visualizar el listado de amigos del usuario y el perfil de uno de esos amigos.

Buscador y petición de amistad de nuevos amigos

Con respecto al buscador de nuevos amigos, implementamos un modal (reflejado en la *Figura 98*) que contuviera un formulario (que es el propio buscador de usuarios dentro de la aplicación) y el resultado de la búsqueda (aquellas personas que no fueran amigos del usuario en cuestión y cumplan con la consulta dada). Este frontend corresponde a la pantalla derecha superior de los bocetos plasmados en las *Figuras 94, 95 y 96*.

Para añadir el usuario deseado a su lista de amigos, únicamente el usuario que realiza la búsqueda debe pulsar sobre el botón *Añadir* para indicar a la aplicación que desea ser amigo de esa persona y, cuando ya haya marcado los nuevos amigos a los que desea mandar una petición de amistad, debe pulsar sobre el botón *Aceptar*. Si todo ha salido como lo esperado, la aplicación envía una alerta [108, 109], como la de la *Figura 99*, para indicar al usuario que se han enviado las peticiones de amistad.



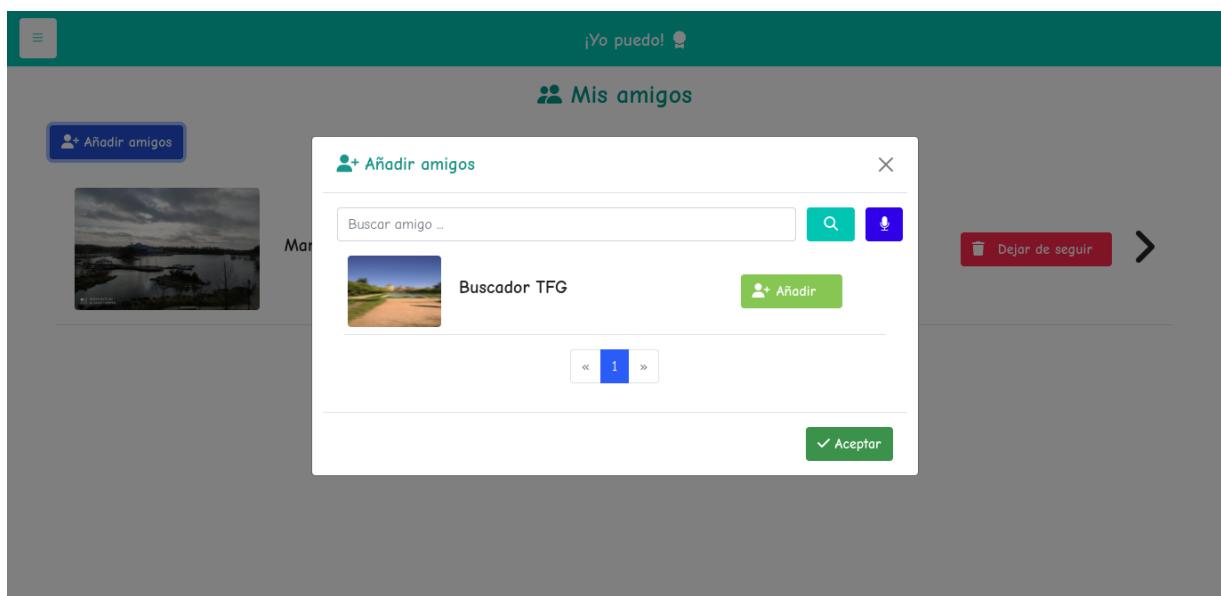
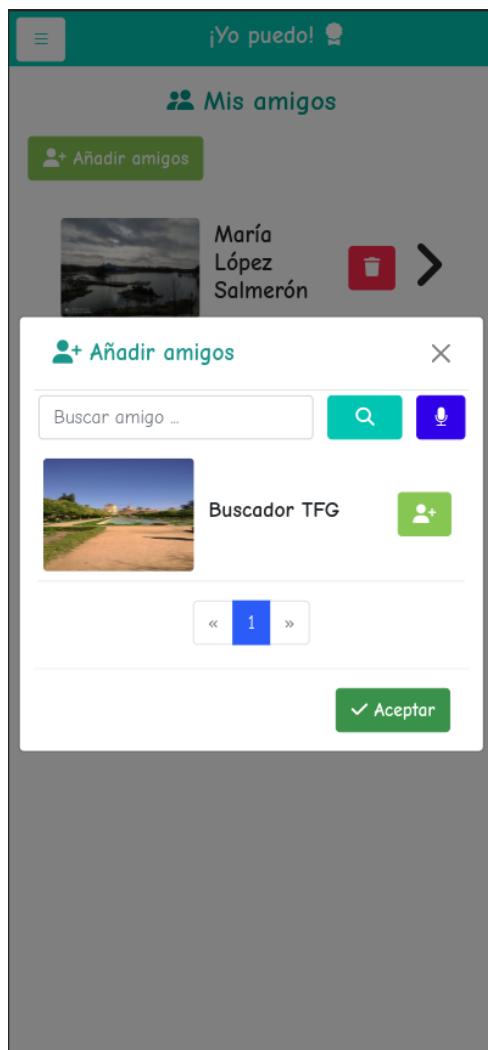


Figura 98: Buscador de nuevos amigos en tablet, móvil y ordenador.

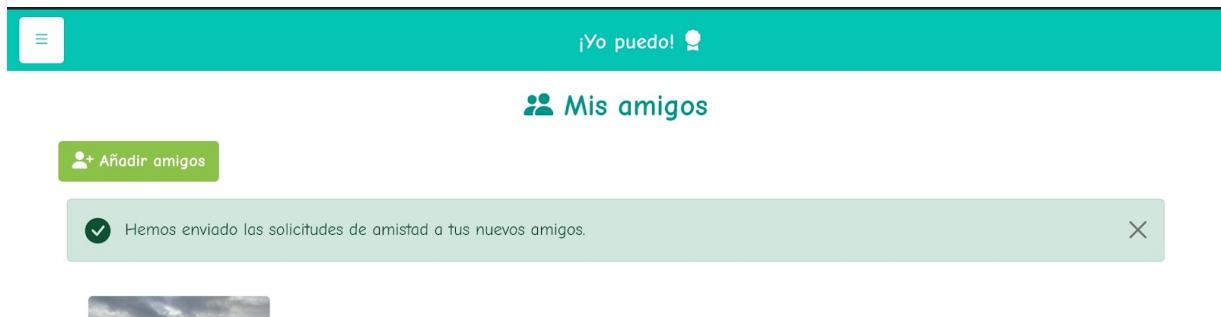


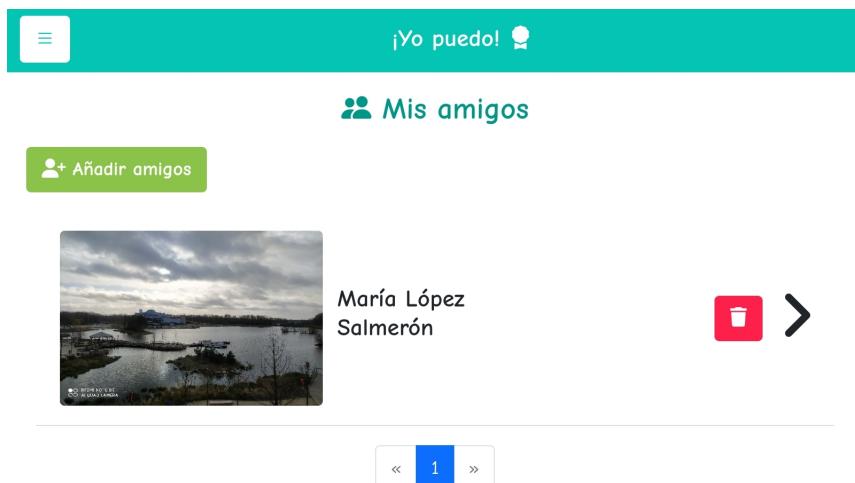
Figura 99: Mensaje de alerta después de enviar petición de amistad.

En cuanto a los siguientes pasos que deben seguir los usuarios a los que se les han enviado la petición de amistad, hemos pensado que la mejor forma de alertar al usuario de una nueva petición de amistad es a través de notificaciones dentro de la aplicación. Por lo que en el octavo sprint llevaremos a cabo las notificaciones de peticiones de amistad y la funcionalidad de aceptar o rechazar amistad.

Listado de amigos

Referente a la lista de usuarios que ya forman parte del grupo de amigos que un usuario, únicamente mostramos para cada usuario, tal y como podemos apreciar en la Figura 100 (que están basadas en la primera pantalla de las Figuras 94, 95 y 96), su foto de perfil, su nombre, un botón para que pueda eliminar esa amistad dentro de la aplicación (el botón con el icono de la basura y, para pantallas grandes, con el título de *Dejar de seguir*) y con otro botón para ver el perfil de ese amigo (el botón con la flecha >).

Para poder buscar y añadir nuevos amigos a la lista, creamos un botón (*Añadir amigos*) en la esquina superior izquierda, justo encima de la lista, para que activara el modal explicado en el apartado anterior.



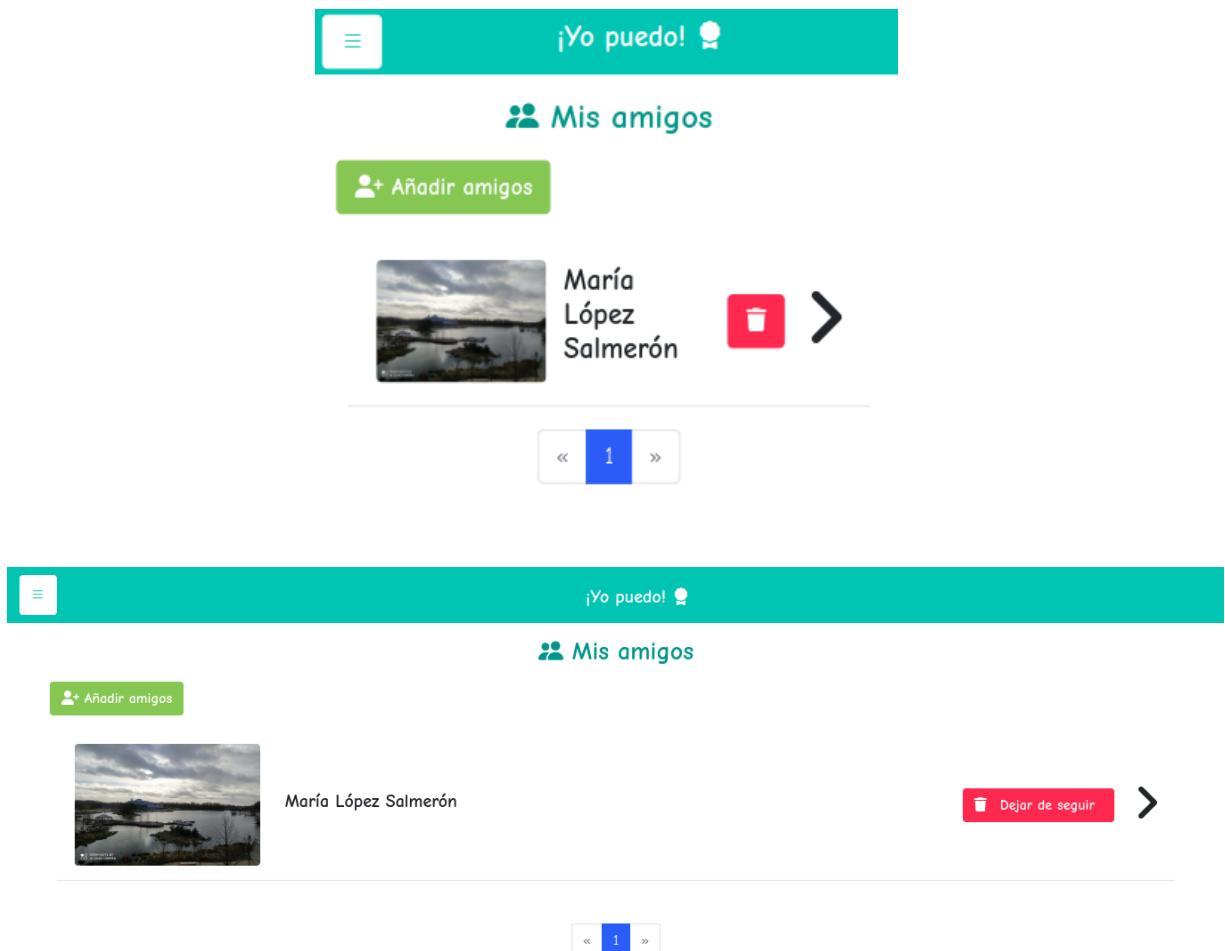


Figura 100: Visualización de listados de amigos para tablet, móvil y ordenador.

Perfil de amigo

Por último, lo último que nos queda para ver los amigos que forman parte de una cuenta de usuario es ver el perfil de cualquier amigo. Tal y como lo planteamos en los bocetos de las *Figuras 94, 95 y 96* (en la segunda pantalla inferior), mostramos como título de la pantalla el nombre del usuario que se está visualizando, su foto de perfil y el correo electrónico de ese usuario. Si recordamos a la implementación del perfil del usuario de la cuenta, tiene una estructura similar a la implementada para ver el perfil del amigo.

Además de la información personal del usuario, exponemos la lista de retos en los que ambos usuarios forman parte, ya sean como participantes o animadores de los retos en común. Como hicimos cuando mostramos el listado de usuario de un usuario, únicamente indicamos la categoría, el título y el objetivo de cada reto, además de la posibilidad de ver el detalle de ese reto.

Estas implementaciones las podemos ver reflejadas en cada imagen de la *Figura 101*.

☰

¡Yo puedo! 🎉

María López Salmerón



Email: mjls130598@hotmail.com

 Dejar de seguir

 **Conocimientos**

Comprobar pruebas
Mirar que la parte de pruebas funcione correctamente

 >

☰

¡Yo puedo! 🎉

María López Salmerón



mjls130598@hotmail.com

 Dejar de seguir

 **Comprobar pruebas**
Mirar que la parte de pruebas funcione correctamente

 >

 **Probar funcionamiento de notificaciones**
Mirar que realmente hace lo que debe hacer

 >

<< 1 >>

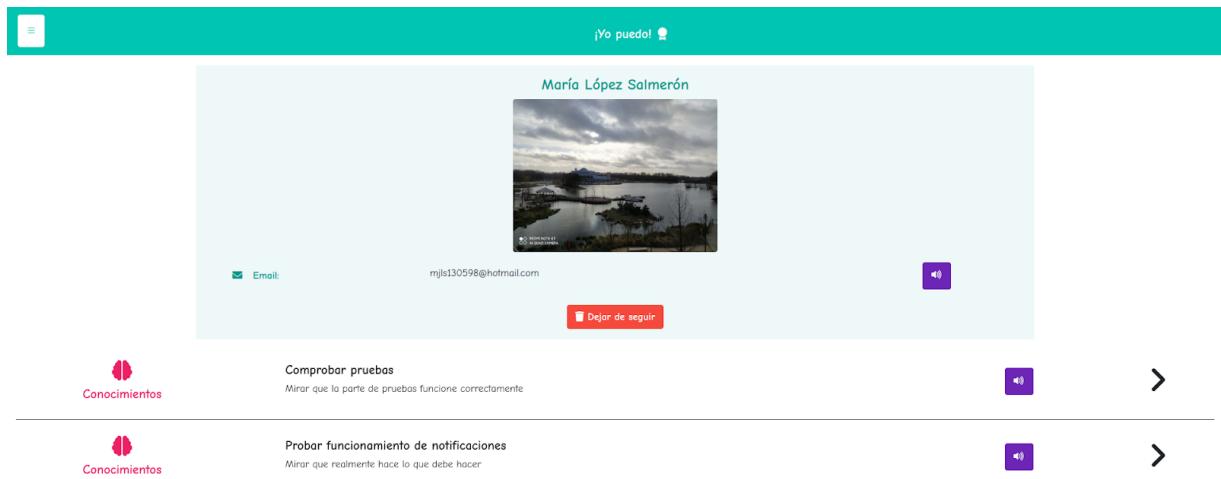


Figura 101: Ejemplo de perfil de un amigo en tablet, móvil y ordenador.

Pruebas

Con respecto a las pruebas realizadas sobre cada una de las funcionalidades de este sprint, vamos a explicarlas a continuación:

- Comprobamos, a través de los bocetos, que el frontend de las historias de usuario llevadas a cabo cumplen con las distintas normas de las guías de accesibilidad (vistas detalladamente en el [ANEXO 2](#)).
- Verificamos que, dentro del backend, cada una de las funcionalidades implementadas realizan las acciones esperadas:
 - Únicamente se puede acceder cuando el usuario haya iniciado sesión dentro de la aplicación.
 - Cuando pedimos el listado de amigos, realizamos la consulta dentro de la base de datos y recibimos la información esperada.
 - Cuando queremos consultar el perfil de un usuario, verificar que existe y que se muestran los datos esperados.
 - Cuando eliminamos a un amigo, se borra la unión entre ambos.

Retrospectiva

En consecuencia, hemos podido realizar algunas de las funcionalidades planteadas a lo largo de este sprint, puesto que la de aceptar o rechazar la solicitud de amistad, se necesitaba crear, de alguna forma, el acceso a una petición de amistad previamente

enviada. Por lo tanto, pensamos que la mejor manera de hacerlo es a través de notificaciones, donde el usuario puede ser avisado de la nueva petición de amistad y donde podrá ver detalladamente la información del posible amigo para aceptar o rechazar su petición de amistad. Además, pensamos utilizar esta nueva funcionalidad para avisar al usuario de nuevos mensajes de apoyo que reciba de sus retos y de la inserción a nuevos retos (como participante o como animador). Esta nueva funcionalidad la implementaremos en el octavo sprint del desarrollo de este proyecto.

3.12. Sexto Sprint

En el sexto sprint del desarrollo de este producto nos vamos a centrar en el manejo de usuarios como animadores dentro de un reto, como la inserción de animadores, el envío y la visualización de mensajes de ánimo, eliminar animador o dejar de seguir siendo un animador dentro de un reto.

Historias de Usuario

- **[HU23] Como coordinador, necesito añadir en mis retos amigos para que me apoyen.**
 - Quién: Coordinador.
 - Cuándo: Cuando necesite ayuda de sus amigos para poder alcanzar el objetivo.
 - Entonces: Se añade la lista de usuarios al reto como animadores.
 - CoS:
 - El usuario debe iniciar sesión previamente.
 - El usuario debe ser el coordinador del reto.
 - Las personas que se añadan como animadores deben ser amigos de ese usuario, que se buscarán por nombre o por email.
 - Se realizará cuando el reto no se ha terminado de crear o cuando el reto está en estado “Propuesto”.
 - Se añadirá una pestaña cuando se cree un reto o se edite donde podrá insertar los animadores.

- Cuando se hayan seleccionado y cuando se muestre el listado de posibles amigos a insertar, se mostrará el nombre y la foto de perfil de las personas seleccionadas.
 - Cuando se hayan guardado las inserciones, se notifica a los animadores del nuevo reto añadido.
- **[HU24] Como coordinador, me gustaría decidir qué animador puede ver los mensajes del resto, además de los suyos propios.**
 - Quién: Coordinador.
 - Cuándo: Cuando haya asignado a los animadores y quiera elegir aquellos usuarios que puedan ver los comentarios del resto.
 - Entonces: Se añaden como “superanimadores” aquellas personas seleccionadas.
 - CoS:
 - El usuario debe haber iniciado sesión.
 - El usuario debe ser el coordinador del reto.
 - El reto debe no haber terminado de crearse aún o estar en estado “Propuesto”.
 - Cuando se hayan elegido los usuarios que serán los animadores del reto, se podrá indicar si va a ser superanimadores o no cada uno de ellos a través de un botón que activará o desactivará dicha función.
 - Dentro de la lista de animadores del reto, se guarda, además de su información, si ese usuario es superanimador o no.
- **[HU25] Como animador, necesito dejar un mensaje de apoyo a mi amigo.**
 - Quien: Animador.
 - Cuándo: Cuando sea animador de un reto de uno de sus amigos y desee dar apoyo.
 - Entonces: Se añade, dentro de la etapa en la que se encuentra actualmente su amigo, un mensaje de ánimo.

- CoS:
 - El usuario debe haber iniciado sesión.
 - El usuario debe ser uno de los animadores del reto dado.
 - La etapa en la que se añade el mensaje de ánimo debe estar en estado “*En proceso*”.
 - El mensaje de ánimo puede ser de los siguientes tipos de formatos: vídeo, texto, audio o imagen.
 - Se guardará el mensaje indicado, el usuario que la añadió, en la etapa en la que envió el mensaje y su identificador.
 - En aquellos mensajes que no sean textuales, sino que se haya adjuntado un fichero (de imagen, vídeo o audio), se guardará la ubicación del fichero dado y almacenado dentro del sistema.
 - Se notifica a los participantes del reto que le han escrito un mensaje de ánimo en uno de sus retos.
 - Se insertará el mensaje a través de un formulario dentro de una pestaña que estará dentro de la etapa seleccionada.
- **[HU26] Como persona, me gustaría ver todos los mensajes de ánimo de una de las etapas de mi reto.**
 - Quién: Persona.
 - Cuándo: Cuando sea animador del reto dado y haya mensajes de ánimo dentro de la etapa seleccionada.
 - Entonces: Se muestra un conjunto de mensajes de apoyo.
 - CoS:
 - El usuario debe haber iniciado sesión previamente.
 - El usuario debe ser un participante o un animador del reto.
 - Si es un animador, se devuelve los que él haya escrito. Si es un superanimador o un participante, se manda todos los guardados.

- Se devuelve una lista de mensajes de apoyo, mostrando la siguiente información: el nombre del usuario que lo realizó y el mensaje de apoyo.
 - Esa lista estará en la misma pestaña en la que se crean los mensajes dentro de la etapa seleccionada. Aunque se mostrarán antes del formulario, que será visible para aquellos usuarios que sean animadores del reto.
- **[HU27] Como animador, quiero dejar de seguir apoyando el reto.**
 - Quién: Animador.
 - Cuándo: Cuando el animador ya no desee seguir apoyando a un amigo en su reto.
 - Entonces: Se elimina como animador de ese reto el usuario.
 - CoS:
 - El usuario debe haber iniciado sesión previamente.
 - El usuario debe ser un animador del reto dado.
 - Dejará de formar parte del reto como animador, por lo que no podrá volver de nuevo el reto ni estará dentro de su lista de retos.
- **[HU28] Como coordinador, quiero eliminar a uno de mis animadores.**
 - Quién: Coordinador.
 - Cuándo: Cuando el coordinador no quiera que ese animador siga apoyándole en un reto dado.
 - Entonces: El animador seleccionado ya no forma parte del reto dado.
 - CoS:
 - El usuario debe haber iniciado sesión.
 - El usuario debe ser el coordinador del reto dado.
 - El animador seleccionado debe formar parte del reto dado.

- El animador deja de formar parte de ese reto, por lo que no podrá visualizarlo ni estará en su lista de retos.

Bocetos

Con relación a la inserción de animadores, tanto en la parte de edición como en la de creación de un reto, hemos pensado en que se haga en la forma que muestran los bocetos de las *Figuras 102, 103 y 104*.

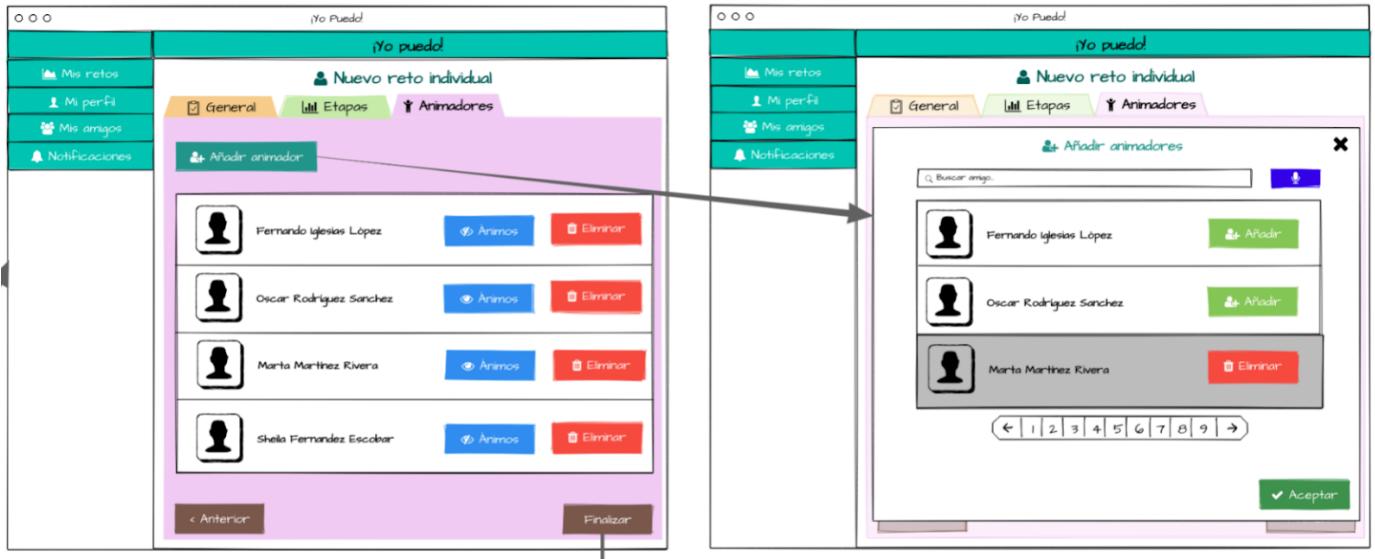


Figura 102: Visualización en formato escritorio de la inserción de animadores.

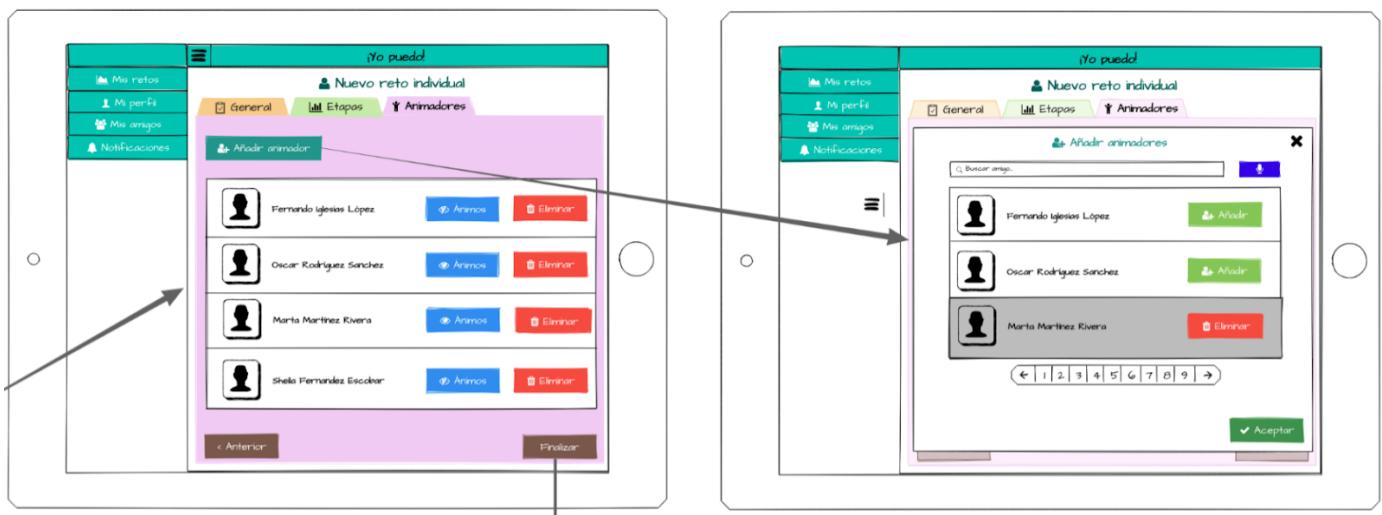


Figura 103: Interfaz de usuario de la inserción de animadores en formato tablet.

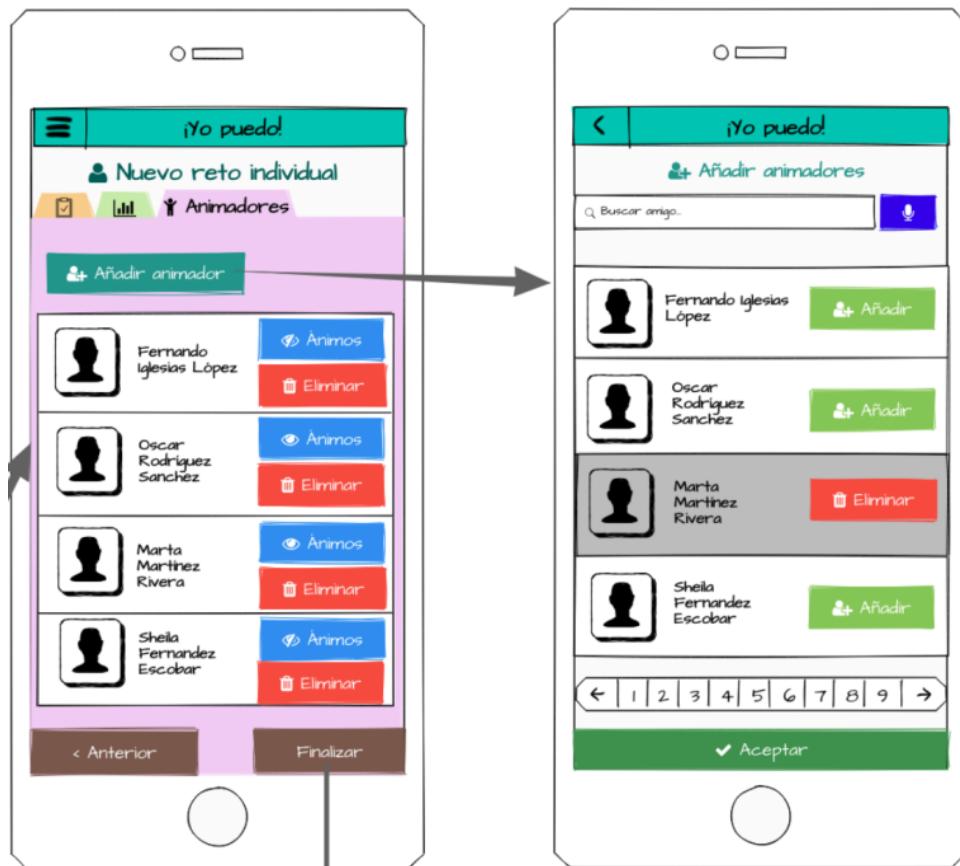


Figura 104: Diseño móvil de la inserción de animadores al reto.

Para eso, en la página principal de ese paso, mostramos la lista de amigos que forman parte del reto como animadores y la posibilidad de eliminarlos después de su inserción (mediante un botón individual a cada amigo), e incluimos un botón, en esa misma pantalla, que lleva a otra pantalla para adjuntar los amigos que el usuario desee.

De esta forma permitimos que el usuario decida los amigos que quiera añadir como animadores y facilitar su búsqueda en la lista de sus amigos a través de una pantalla separada a la de la edición o creación del reto.

Además, dejamos que el usuario sea el que decida qué animadores pueden ver todos los mensajes de apoyo, dicho de otra forma, qué animadores pasan a ser superanimadores. Hemos implementado esta funcionalidad a través de un botón en el que se indica si el animador puede o no ver los mensajes de ánimo de otros animadores.

El único inconveniente que pudimos apreciar en estos bocetos es que en la parte de "añadir/eliminar" superanimadores, el título que le damos al botón ("Ánimos") no es el más adecuado para su comprensión. Por lo tanto, decidimos llamarlo a partir de este momento como "Permiso", porque realmente con este botón lo que estamos haciendo es informar al

usuario de que si ese botón está activo para un amigo en concreto, esa persona tiene su permiso para ver los mensajes de ánimo del resto de animadores.

Si nos acordamos de los cambios que deberíamos hacer en cuanto a la visualización de los retos, en este sprint nos centramos en la visualización del reto por parte de un animador. Por lo tanto, en las *Figuras 105, 106* y *107* observaremos cómo lo hará un animador del reto.

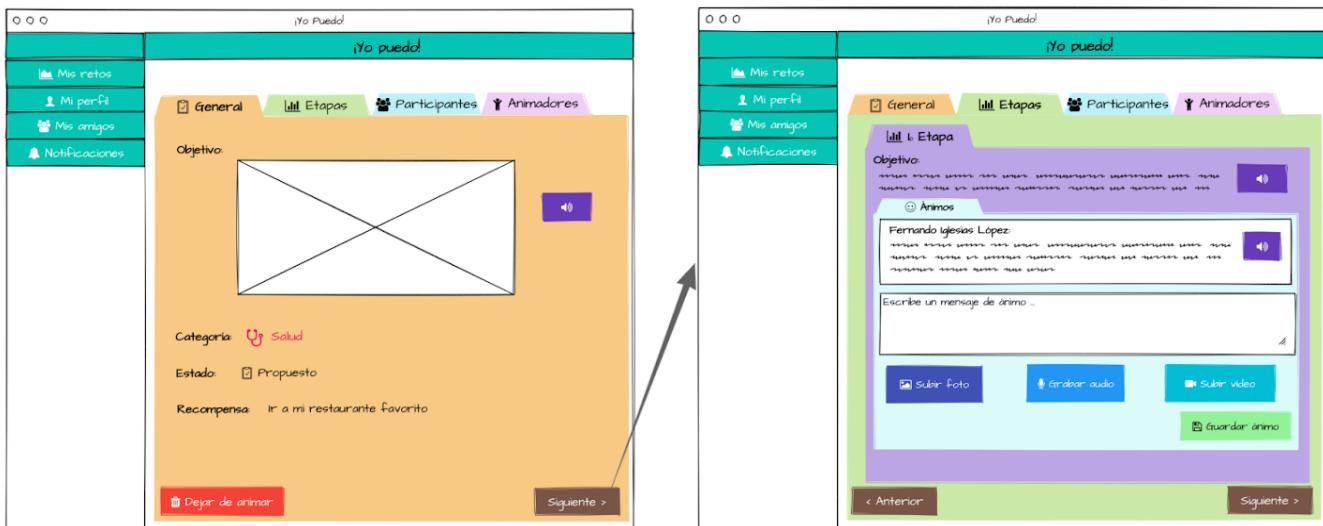


Figura 105: Visualización de un reto por parte del animador en un ordenador.

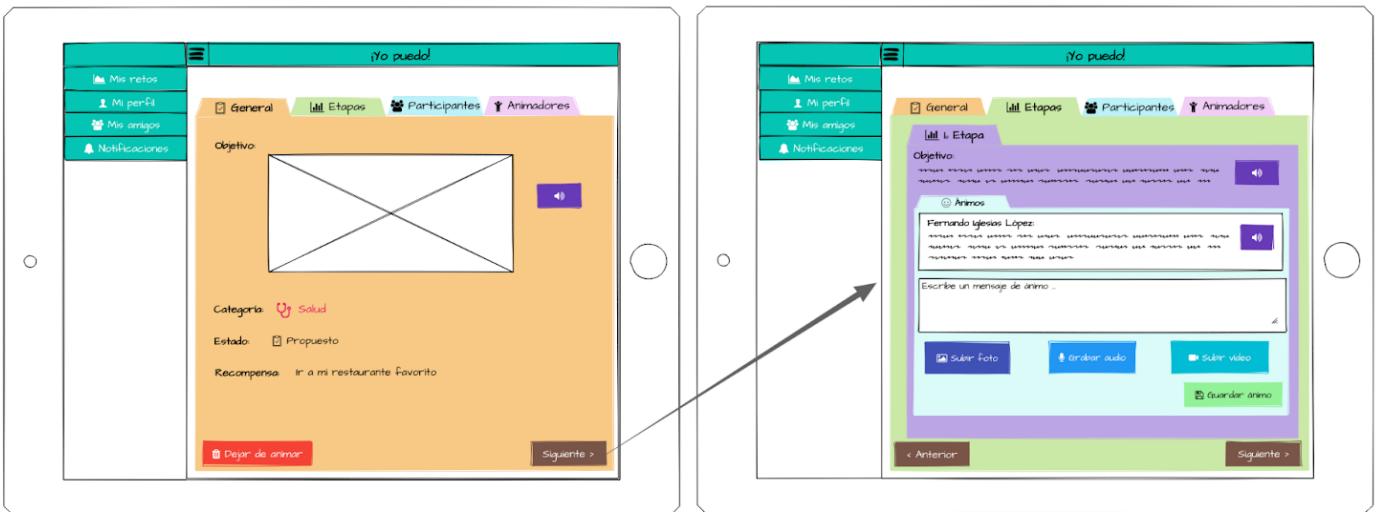


Figura 106: Visualización del reto desde la perspectiva de un animador para tablets.

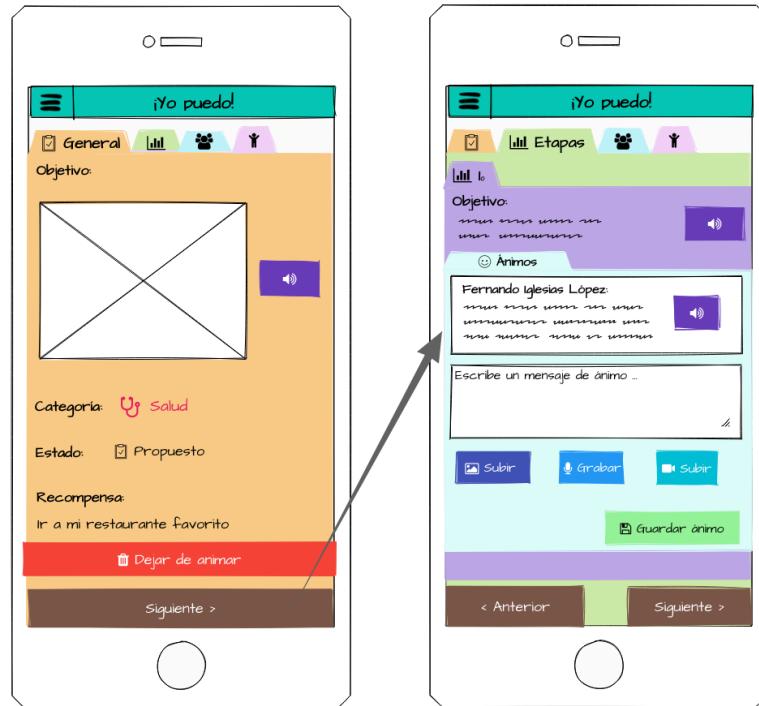


Figura 107: Segunda versión de la visualización del reto por parte de un animador en formato móvil.

El diseño de estas ventanas es similar a las del participante (Figuras 86, 87 y 88), excepto que el animador no tiene acceso a la pestaña evidencias dentro de una etapa, que puede dejar de formar parte de los animadores del reto pulsando al botón “Dejar de animar” y que puede ver sus mensajes de ánimo (todos si es superanimador) y publicar uno nuevo (si la etapa que se está visualizando está en estado “En proceso”).

Todos los diseños mostrados en este apartado cumplen con las normas dictadas en las [guías de accesibilidad](#), que podemos ver en detalle la forma en la que se implementaron en el [ANEXO 2](#).

Base de datos

Como ocurrió cuando explicamos la base de datos del cuarto sprint, únicamente nos centraremos en las acciones que puede o que se hacen sobre un animador de un reto. Por lo tanto, solamente prestaremos importancia a la relación *Anima* que hay entre la entidad *Usuario* y la entidad *Reto*.

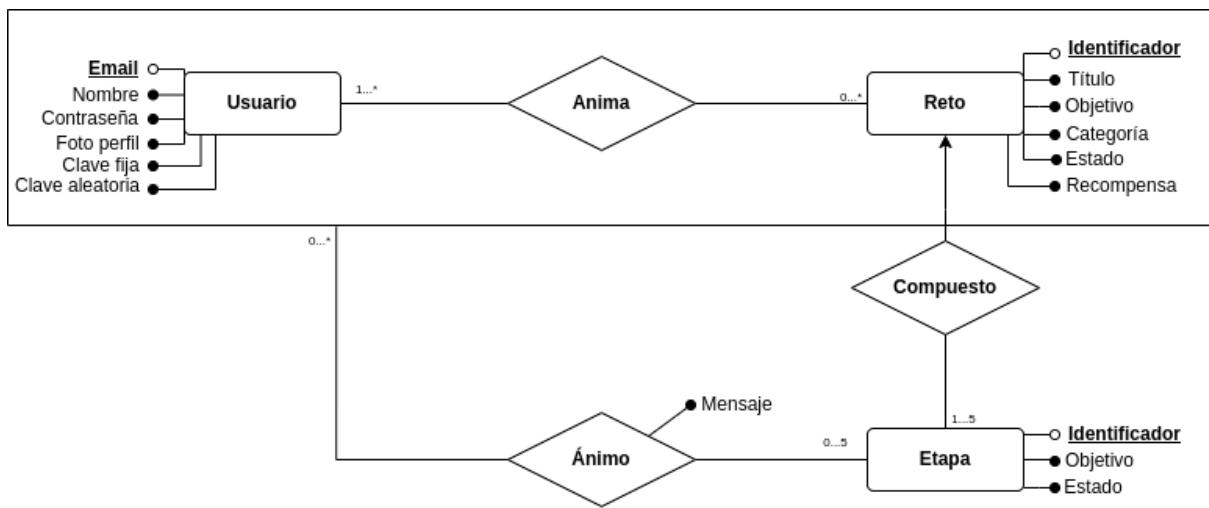


Figura 108: Diagrama Entidad/Relación del sexto sprint.

Para este sprint, creamos una agregación con la relación *Anima* con una relación con la entidad *Etapa* que tendrá como único atributo *Mensaje*, la cual la nombraremos como *Ánimo*, para poder cubrir con la funcionalidad de enviar y obtener mensajes de ánimo en la etapa de un reto.

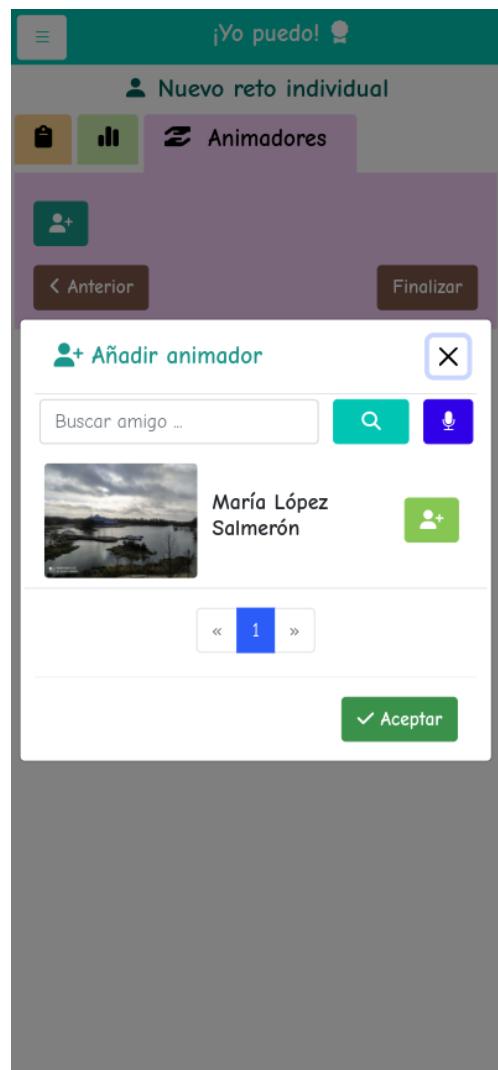
Implementación

Acerca de la implementación de las distintas funcionalidades con respecto a los animadores de un reto, dividiremos su explicación en distintas secciones: eliminación, inserción y listado de animadores, listado y envío de mensajes de ánimo.

Eliminación, inserción y listado de animadores

Con respecto a la inserción de animadores, lo hacemos, cuando editamos o creamos cualquier tipo de reto, creando un modal similar al generado para la búsqueda de nuevos amigos (el mostrado en la Figura 98, con su formulario de consulta y su listado de usuarios que cumplan esa consulta) excepto que, en vez de mostrar aquellos usuarios que no son amigos nuestros, visualizamos la lista de amigos.

Esta implementación fue basada en la segunda pantalla de los bocetos de las Figuras 102, 103 y 104 y se llevaron a cabo como enseñamos en la Figura 109.



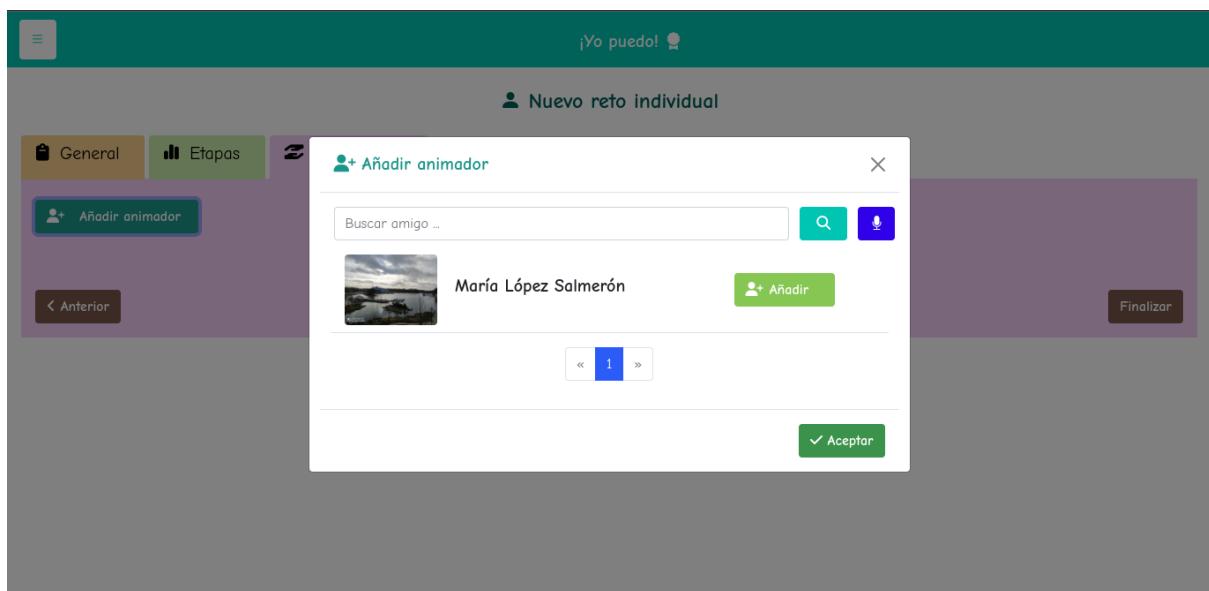
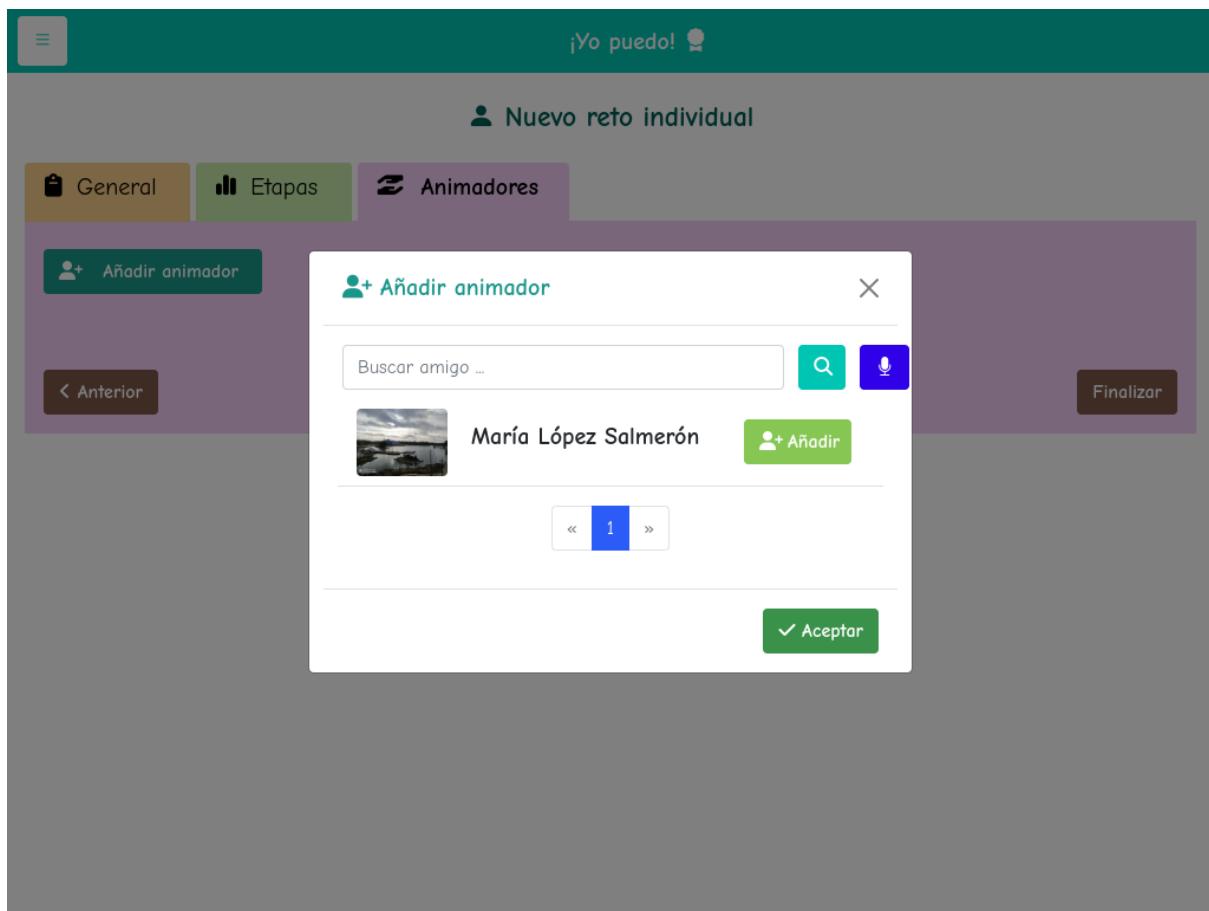


Figura 109: Inserción de animadores en móvil, tablet y ordenador.

En cuanto a la forma en la que se insertan los animadores al reto, lo primero que hacemos es permitir al usuario que señalice los usuarios que desee que formen parte del reto a través del botón *Añadir*. Una vez elegidos, debe pulsar al botón *Aceptar* donde se visualizarán y se guardarán temporalmente el listado dentro del reto, como podemos

apreciar en la *Figura 110*, que nos basamos en los diseños de la primera pantalla de las *Figuras 102, 103 y 104*. Antes de terminar de editar o crear el reto, el usuario puede eliminar los animadores que no deseé pulsando, para cada uno de ellos, el botón rojo *Eliminar* y, además, puede indicar de los animadores seleccionados cuáles de ellos desea que sea superanimador (aquel con el ícono del ojo) o no (aquel con el ícono del ojo tachado), por defecto todos ellos son animadores normales y corrientes.

Para activar el modal de la *Figura 109*, vemos en las pantallas de la *Figura 110* que hay un botón encima del listado de animadores llamado *Añadir animador* y/o el que tiene el símbolo de una persona más un +.

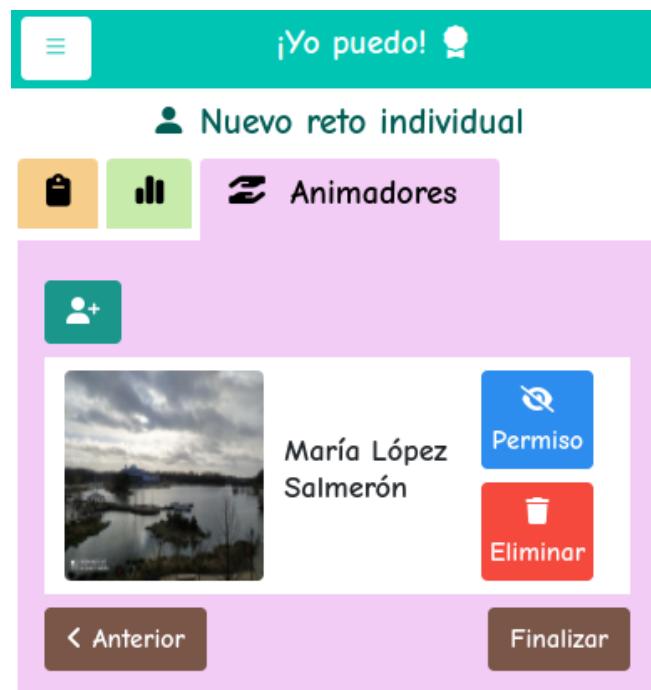




Figura 110: Listado de posibles animadores en móvil, tablet y ordenador.

Cuando termine de crear o de editar el reto, dentro del backend, la aplicación se encarga de guardar cada usuario en la base de datos dentro de la relación *Anima*, junto al resto de información del reto.

Si el animador desea dejar formar parte del reto, en la pantalla general del reto encuentra un botón en la esquina inferior izquierda (llamado *Dejar de apoyar* junto al ícono de la papelera) que puede pulsar para salir del reto. Dicho botón lo podemos ver reflejado en la Figura 111.

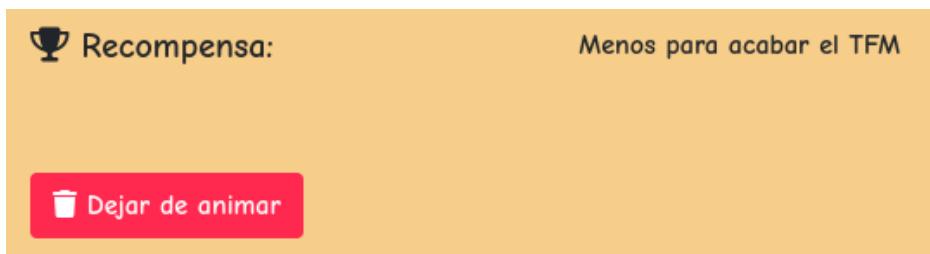


Figura 111: Botón para dejar de apoyar el reto.

Listado y envío de mensajes de ánimo

En cuanto a la visualización y envío de mensajes de apoyo, se realizará dentro de cada una de las etapas del reto cuando no estén en estado *Propuesto*. En dentro de la etapa, se mostrará una pestaña (junto a la pestaña de pruebas si es un participante) en la que aparecerán los mensajes y el formulario para insertar los mensajes (este último si es un animador del reto) tal y como podemos apreciar en la Figura 112, que se basaron en los bocetos de las Figuras 105, 106 y 107.

¡Yo puedo! 🎉

Comprobar pruebas

General Etapas

1º Etapa 2º Etapa

Objetivo: Comprobar que se visualiza correctamente pruebas

Ánimos

Maria López Salmerón
Puedes con esto y mucho más!! Eres una crack!

Subir foto Subir audio Subir video

O escribe el mensaje de ánimo ...

Guardar ánimo

Siguiente etapa >

< Anterior Siguiente >

¡Yo puedo! 🎉

Comprobar pruebas

Etapas

1º Etapa 2º

Objetivo: Comprobar que se visualiza correctamente pruebas

Ánimos

Maria López Salmerón
Puedes con esto y mucho más!! Eres una crack!

Subir foto Subir audio Subir video

O escribe el mensaje de ánimo ...

Guardar ánimo

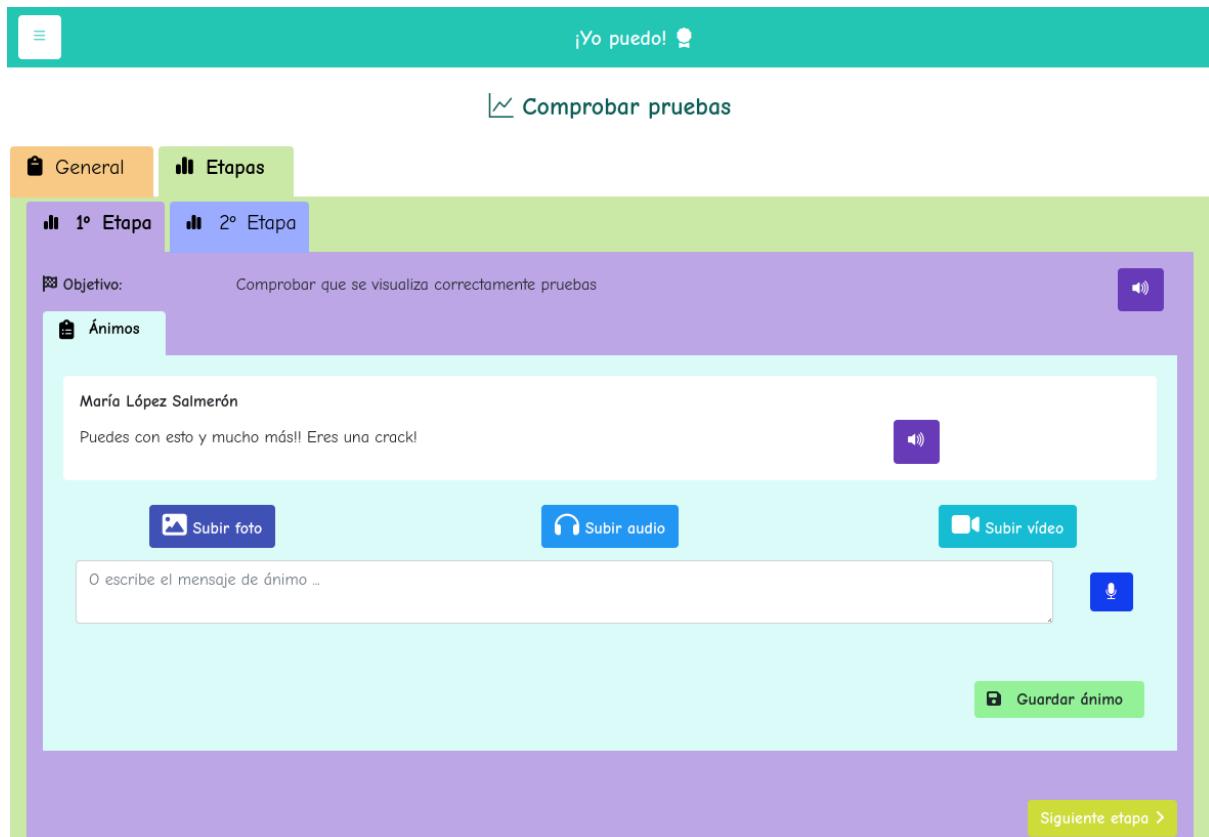


Figura 112: Listado y envío de mensajes de ánimo por parte de un animador en móvil, ordenador y tablet.

Acerca del envío de mensajes de ánimo, decidimos que se va a poder utilizar en aquellas etapas a través de un formulario en el que permitimos la inserción de mensajes a través de texto, imagen, audio o vídeo, como hicimos durante la creación de retos para añadir objetivos o recompensa en el [tercer sprint](#).

Una vez enviado el mensaje de ánimo desde el frontend, llama a la función del backend que se encarga de guardar en la base de datos los datos del animador y de la etapa en la que se realiza dentro de la relación *Ánimo* (señalizada dentro del diagrama de Entidad/Relación de la Figura 108. A continuación, actualizamos la lista de mensajes a través de HTMX.

Con respecto a la visualización de dichos mensajes ya guardados, lo que primero hacemos es preguntar qué tipo de rol tiene el usuario dentro del reto: si es un participante o un superanimador, verá todos los mensajes de ánimo almacenados en la etapa que se esté visualizando; y si es un animador normal, sólo verá los suyos. Ya conocidas las restricciones, realizamos las consultas necesarias para obtener y las devolvemos al frontend el mensaje, en el formato que sea con las herramientas necesarias para que sean accesible,

junto a la persona que lo realizó, como las de las *Figuras 112* (para los animadores) y *113* (para participantes).

¡Yo puedo! 🎉

↗ Comprobar pruebas

📁 Etapas

📊 1º Etapa 📊 2º

☒ Objetivo:
Comprobar que se visualiza correctamente pruebas

🔊

📝 Ánimos

Maria López Salmerón
Puedes con esto y mucho más!! Eres una crack!

🔊

★ Calificación:

😢 😊 😃

Siguiente etapa >

< Anterior Siguiente >

This screenshot shows a mobile application interface. At the top, it says '¡Yo puedo! 🎉'. Below that is a navigation bar with 'Etapas' selected. It shows two steps: '1º Etapa' and '2º'. Under '1º Etapa', there is an objective: 'Comprobar que se visualiza correctamente pruebas'. A speaker icon indicates audio content. Below this is a message from 'Maria López Salmerón': 'Puedes con esto y mucho más!! Eres una crack!', also with a speaker icon. There is a 'Calificación:' section with three smiley face icons. At the bottom are navigation buttons: '< Anterior' and 'Siguiente >'.

☰

¡Yo puedo! 🎉

↗ Comprobar pruebas

📁 General 📊 Etapas

📊 1º Etapa 📊 2º Etapa

☒ Objetivo:
Comprobar que se visualiza correctamente pruebas

📝 Pruebas 📄 Ánimos

Maria López Salmerón
Puedes con esto y mucho más!! Eres una crack!

🔊

★ Calificación:

😢 😊 😃

Siguiente etapa >

< Anterior Siguiente >

This screenshot shows a desktop application interface. At the top, it says '¡Yo puedo! 🎉'. Below that is a navigation bar with 'General' and 'Etapas' tabs, where 'Etapas' is active. It shows two steps: '1º Etapa' and '2º Etapa'. Under '1º Etapa', there is an objective: 'Comprobar que se visualiza correctamente pruebas'. A speaker icon indicates audio content. Below this is a message from 'Maria López Salmerón': 'Puedes con esto y mucho más!! Eres una crack!', also with a speaker icon. There is a 'Calificación:' section with three smiley face icons. At the bottom are navigation buttons: '< Anterior' and 'Siguiente >'.

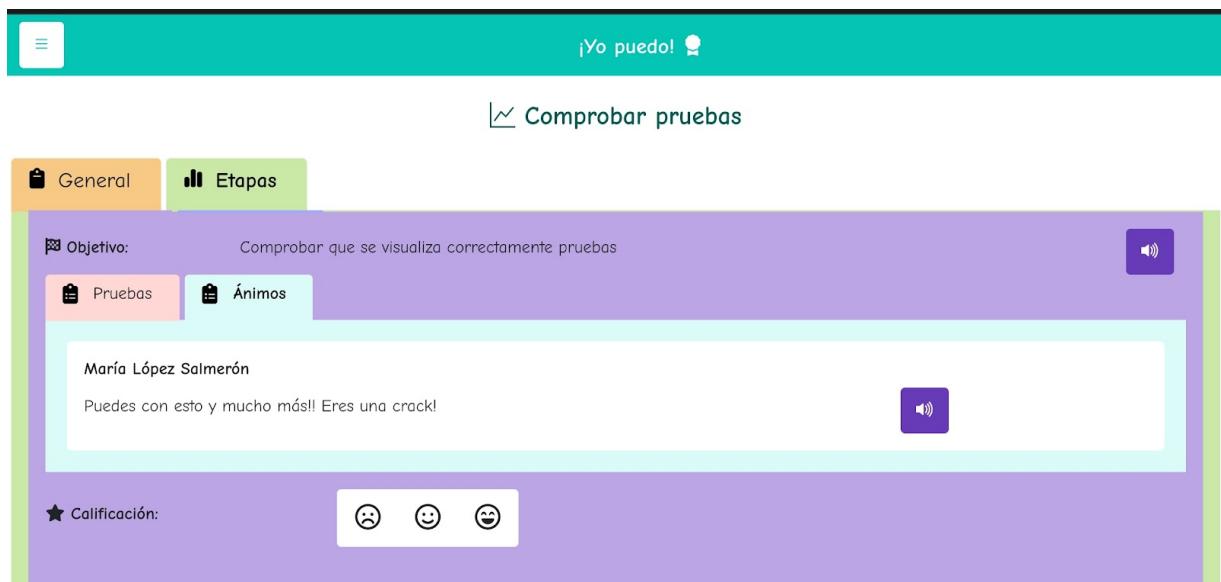


Figura 113: Listado de mensajes de ánimo por parte de un participante en móvil, ordenador y tablet.

Pruebas

Las pruebas que decidimos realizar para verificar el cumplimiento de las validaciones de cada historia de usuario hecha en este sprint son las comentadas a continuación:

- Comprobar que la aplicación es accesible mirando, cuando se estaban realizando los bocetos, que se cumplían con las normas de las [guías de accesibilidad](#). Es por eso que hablamos de varias versiones efectuadas dentro del apartado de [bocetos](#).
- En cuanto al backend, examinamos que:
 - Solo pueden acceder los usuarios que forman parte del reto y que hayan iniciado sesión.
 - El animador únicamente puede ver sus mensajes de ánimo y el superanimador y participante todos.
 - Solo se pueden añadir nuevos mensajes cuando la etapa dada está en estado distinto al de *Propuesto*.
 - Cuando el animador se elimina del reto (ya sea porque el coordinador lo ha eliminado o porque el mismo animador ha deseado dejar de serlo), no puede volver a entrar a ver el reto, ni incluir nuevos mensajes de apoyo dentro de ese reto.

Retrospectiva

Con respecto a la retrospectiva de esta iteración, hemos llegado a realizar todas las funcionalidades planteadas dentro de este sprint en el tiempo de duración que se estimó para cada sprint de este proyecto, sin ningún contratiempo en cuanto al desarrollo de las historias de usuario planificadas. Sin embargo, cuando el animador es incorporado al reto por el coordinador o el animador envía un mensaje de apoyo, dejamos la creación de notificación al usuario correspondiente para el octavo sprint.

3.13. Séptimo Sprint

A lo largo de este sprint nos vamos a encargar de realizar las funciones relacionadas con los retos colectivos, además de otras acciones que se realizan en cualquier reto, como su eliminación o su edición.

Historias de Usuario

- **[HU29] Como persona, quiero crear un reto colectivo con mis amigos.**
 - Quién: Persona.
 - Cuándo: Cuando quiera retarse a llegar a un cierto objetivo con unos amigos, que pasarán a ser participantes del reto.
 - Entonces: La persona crea tiene un reto a realizar con uno o varios amigos que deben cumplir según en las etapas que lo hayan dividido.
 - CoS:
 - El usuario debe haber iniciado sesión en el sistema previamente.
 - El identificador (que se generará de manera automática) es la clave primaria.
 - El reto debe tener los siguientes elementos: título, objetivo, categoría, recompensa, etapas y participantes.
 - Tiene que haber más de una etapa y no más de cinco.
 - Cada etapa debe tener los siguientes elementos: identificador, objetivo y estado.

- El objetivo y la recompensa pueden ser de los siguientes tipos de formatos: vídeo, texto, audio o imagen.
- Para aquellos objetivos y/o recompensa que no sean textuales, sino que se haya adjuntado un fichero (de imagen, vídeo o audio), se guardará la ubicación del fichero dado por el usuario y almacenado dentro del sistema.
- El estado inicial de todas las etapas pasan a “*Propuesto*”.
- El estado inicial del reto pasa a “*Propuesto*”.
- Para considerarse un reto colectivo, el usuario debe añadir al menos un participante al reto.
- Los participantes del reto deben ser, previamente, amigos del usuario que está creando el reto.
- Cuando se busquen y se añadan los amigos, se mostrará la foto de perfil y el nombre.
- Para facilitar al usuario a la hora de crear un reto, lo dividiremos en pestañas de color distinto: la primera será la que contendrá la información general del reto (título, objetivo, categoría y recompensa), la segunda será en la que se creen las distintas etapas de los retos y la tercera será en la que se inserten los participantes del reto. En la segunda pestaña se subdivide a su vez en pestañas según la cantidad de etapas que se cree.
- El usuario que ha creado ese reto pasa a ser el “coordinador” del reto, para indicar que es el usuario que tiene permisos para modificar el reto.
- Dicho usuario se añade como participante del reto por si en un futuro deja de ser el coordinador del reto creado.
- Si todo está correcto, se guardan los datos generales del reto juntos, cada una de las etapas por separado y cada participante, con el email como identificador, y se va a la página donde muestra el reto creado.
- Se notificará a los participantes de la creación del nuevo reto, una vez guardado.

- [HU30] Como persona, quiero añadir a mi reto individual personal participantes.
 - Quién: Persona.
 - Cuándo: Cuando quiera retarse a llegar a un cierto objetivo, que inicialmente lo había planeado para hacerlo solo y ahora quiere añadir amigos a retarse con él.
 - Entonces: La persona tiene un reto con uno o varios amigos que deben cumplir según en las etapas que lo hayan dividido, con lo que un reto individual se convertiría en un reto colectivo.
 - CoS:
 - El usuario debe haber iniciado sesión previamente.
 - El usuario debe ser el coordinador del reto.
 - Tiene que añadir como mínimo a un participante.
 - Los participantes añadidos deben ser amigos del creador del reto.
 - El reto debe estar en estado “Propuesto”.
 - Si todo está correcto, se notifica a los participantes del nuevo reto.
- [HU31] Como coordinador, deseo pasar la coordinación de un reto colectivo a uno de los participantes.
 - Quién: Coordinador.
 - Cuándo: Cuando desee dejar de ser coordinador y pasarle esa responsabilidad a otro participante del reto.
 - Entonces: Se cambia de coordinador.
 - CoS:
 - El usuario debe haber iniciado sesión.
 - El usuario debe ser el coordinador del reto.
 - El reto dado debe ser un reto colectivo.

- La persona que va a pasar a ser coordinador debe ser un participante del reto.
 - Para seleccionar el participante que va a ser el nuevo coordinador, se mostrará de cada uno de ellos su nombre y su foto de perfil y se podrá buscar por nombre o por email.
 - El antiguo coordinador deja de tener permisos para editar el reto, solo es un participante del reto.
 - El nuevo coordinador obtiene los permisos necesarios para editar ese reto.
 - Se envía una notificación al nuevo coordinador informando del nuevo cambio de poderes.
- [HU32] **Como coordinador, deseo modificar los datos de mi reto.**
 - Quién: Coordinador.
 - Cuándo: Cuando desee modificar alguno de los datos de uno de sus retos.
 - Entonces: Se modifica el reto dado.
 - CoS:
 - El usuario debe haber iniciado sesión previamente.
 - El reto debe existir y estar en estado “Propuesto”, lo mismo que cada una de sus etapas.
 - El usuario debe ser el coordinador del reto.
 - El reto debe tener los siguientes elementos: título, objetivo, categoría, recompensa, etapas, participantes y animadores.
 - Tiene que haber más de una etapa y no más de cinco.
 - Cada etapa debe tener los siguientes elementos: identificador, objetivo y estado.
 - El objetivo y la recompensa pueden ser de los siguientes tipos de formatos: vídeo, texto, audio o imagen.

- Para aquellos objetivos y/o recompensa que no sean textuales, sino que se haya adjuntado un fichero (de imagen, vídeo o audio), se guardará la ubicación del fichero dado por el usuario y almacenado dentro del sistema.
 - El estado inicial de todas las etapas nuevas pasan a “*Propuesto*”.
 - Los participantes y los animadores del reto deben ser, previamente, amigos del coordinador.
 - Cuando se busquen y se añadan los amigos, se mostrará la foto de perfil y el nombre.
 - Para facilitar al usuario a la hora de editar un reto, lo dividiremos en pestañas de color distinto: la primera será la que contendrá la información general del reto (título, objetivo, categoría y recompensa), la segunda será en la que se creen las distintas etapas de los retos, la tercera será en la que se inserten los participantes del reto y la cuarta será la que se indiquen los animadores del reto. En la segunda pestaña se subdivide a su vez en pestañas según la cantidad de etapas que se cree.
 - Si todo está correcto, se guardan los datos generales del reto juntos, cada una de las etapas por separado, cada participante y cada animador, con el email como identificador, y se va a la página donde muestra el reto editado.
 - Se notificará a los nuevos participantes y a los nuevos animadores de la adjudicación a un nuevo reto, una vez guardado.
- [HU33] **Como coordinador, deseo eliminar el reto.**
 - Quién: Coordinador.
 - Cuándo: Cuando desee eliminar definitivamente uno de sus retos.
 - Entonces: Se elimina el reto del sistema.
 - CoS:
 - El usuario debe haber iniciado sesión.
 - El reto dado debe existir.

- El usuario debe ser coordinador del reto dado a eliminar.
 - Se elimina el reto dentro del sistema.
 - Se elimina para cada uno de los animadores y de los participantes ese reto de su lista de retos.
- **[HU34] Como coordinador, necesito comenzar el reto.**
 - Quién: Coordinador.
 - Cuándo: Cuando necesite o desee iniciar uno de sus retos.
 - Entonces: Se cambia el estado del reto.
 - CoS:
 - El usuario debe haber iniciado sesión.
 - El reto debe existir y estar en estado “Propuesto”.
 - Debe ser el coordinador del reto dado.
 - Se cambia el estado del reto a “En proceso”.
 - Se cambia el estado de la primera etapa a “En proceso”.
 - **[HU35] Como coordinador, quiero eliminar a uno de mis participantes.**
 - Quién: Coordinador.
 - Cuándo: Cuando la persona no quiera que ese participante siga en el reto.
 - Entonces: El participante seleccionado ya no forma parte del reto.
 - CoS:
 - El usuario debe iniciar sesión.
 - El reto debe existir y estar en estado “Propuesto”.
 - El participante seleccionado debe formar parte del reto.
 - El usuario debe ser el coordinador del reto dado.

- Se elimina ese usuario como participante en ese reto, por lo que no tiene acceso a la visualización del reto ni a la inserción de evidencias y de calificaciones dentro del reto.

Bocetos

Con respecto a la creación de retos colectivos, es similar a la creación de retos individuales junto a la inserción de animadores, excepto que añadimos otro paso en dicha creación: la inserción de participantes. La forma en la que se realizará dicha creación lo podemos ver en los bocetos de los distintos tipos de dispositivos de las *Figuras 114, 115 y 116*.

Añadir participantes a un reto (ya sea en la creación de uno colectivo o en la edición de un reto previamente creado) se realiza de la misma forma que un animador. Aunque, cuando ya están seleccionados los amigos que formarán parte del reto como participantes, únicamente podrá borrar cada selección individualmente, no como en el paso de los animadores, que permitimos al usuario, además de borrar a un animador, indicar si ese animador va a formar parte de los “superanimadores” del reto.

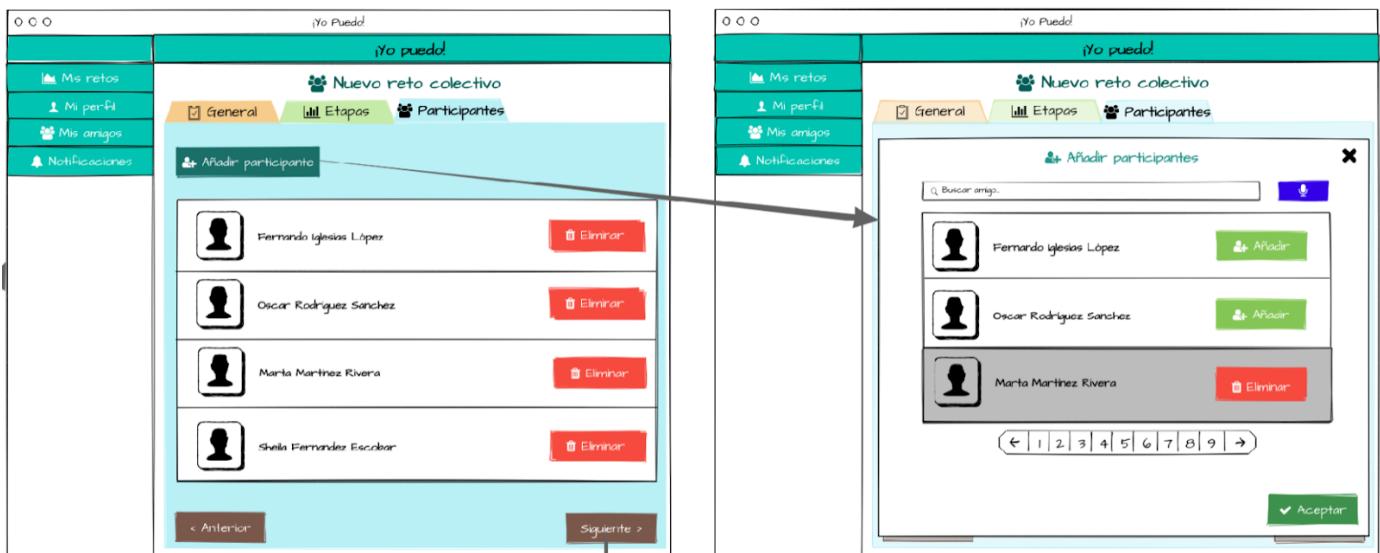


Figura 114: Interfaz en formato escritorio para la inserción de participantes.

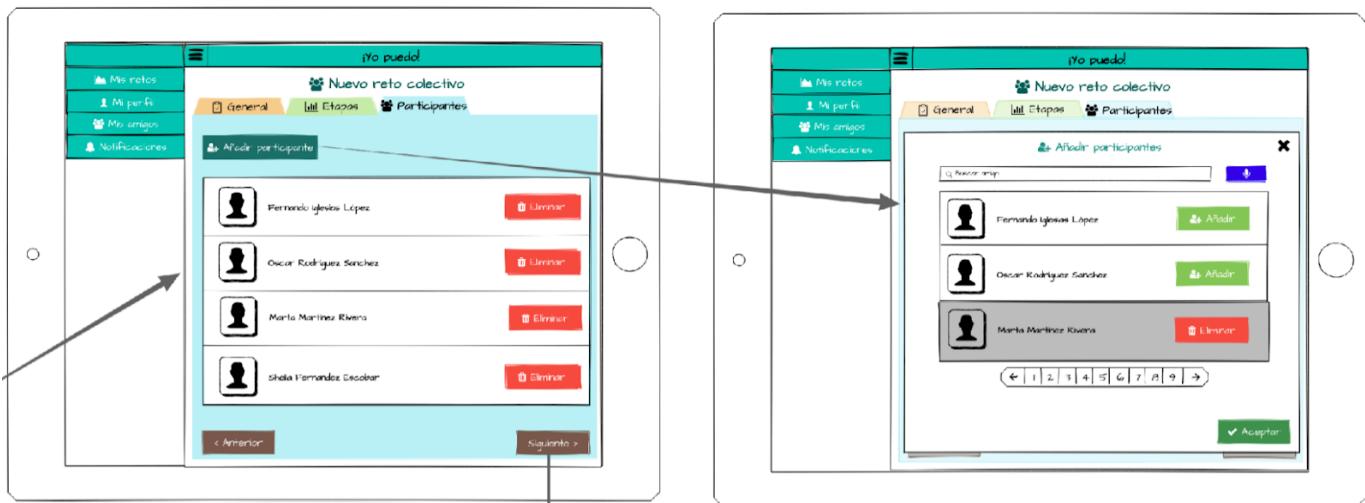


Figura 115: Diseño para tablets sobre la inserción de participantes.

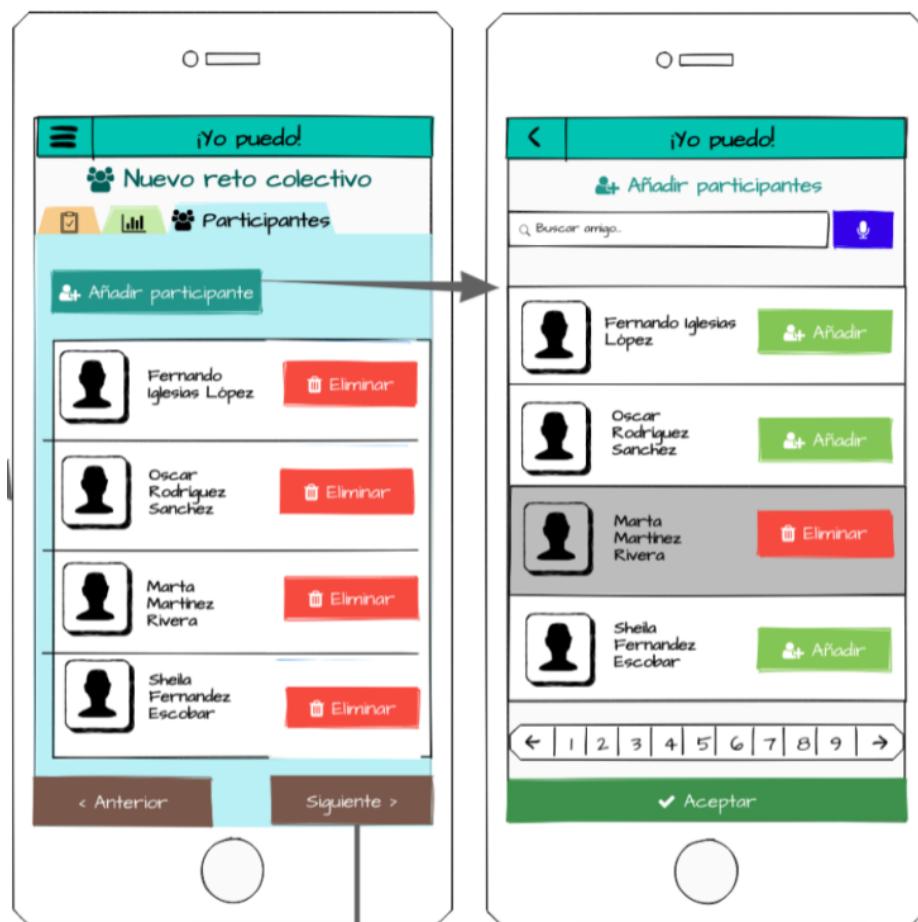


Figura 116: Boceto móvil de la inserción de participantes.

Recordando los bocetos realizados en el cuarto sprint del desarrollo de este proyecto, indicamos que faltaban ciertas funcionalidades que se podrían implementar en la parte de visualización de un reto cuando el usuario que visualizara un reto no fuera ni el participante ni el animador del reto, sino el coordinador. Por lo tanto, en la pantalla general

del reto, incluimos los botones a las funcionalidades faltantes (editar, eliminar y cambiar de coordinador), como mostramos en las *Figuras 117, 118* y *119*, cumpliendo con las normas de las guías de accesibilidad, tal y como lo detallamos en el [ANEXO 2](#).

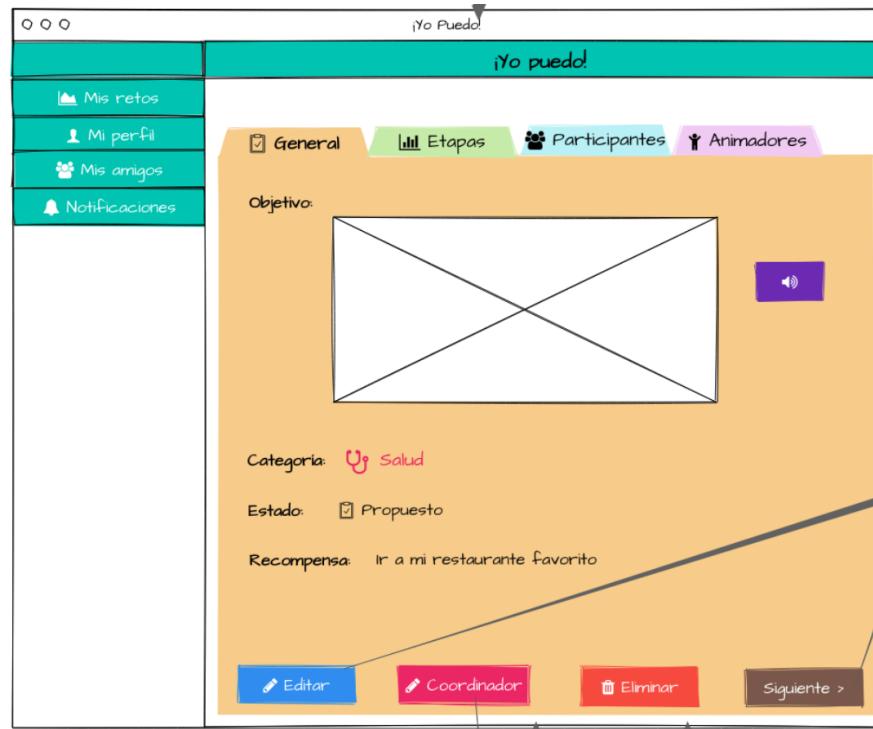


Figura 117: Visualización de un reto para ordenadores desde la perspectiva de un coordinador.

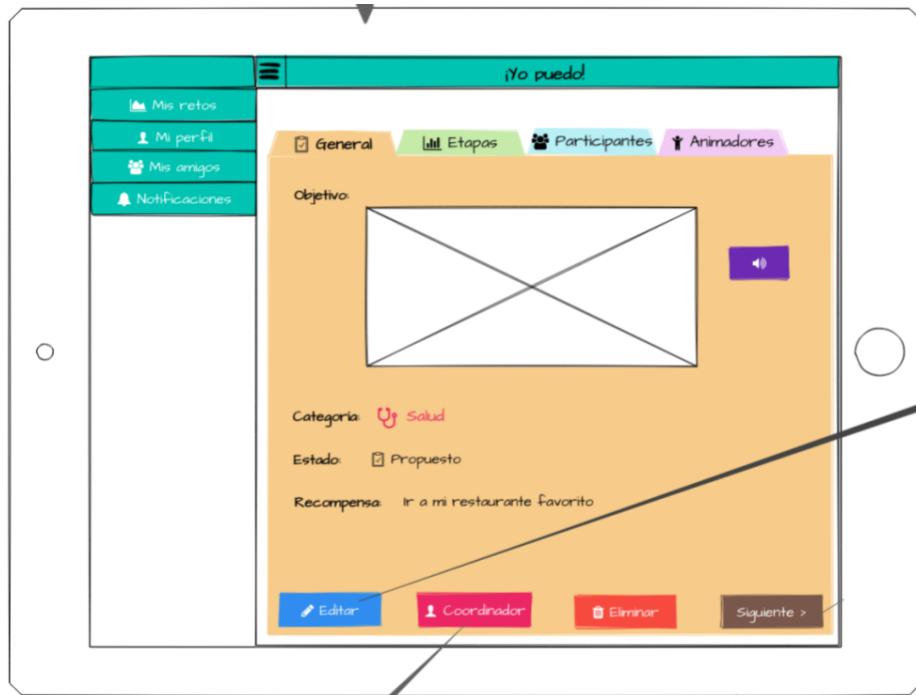


Figura 118: Visualización de un reto en tablets para coordinadores.



Figura 119: Visualización de un reto en formato móvil por parte de un coordinador.

Por lo tanto, en cuanto a la visualización de un reto por parte de un coordinador (*Figuras 117, 118 y 119*), lo que hemos realizado para insertar aquellas funcionalidades que faltaban son:

- Para poder editar un reto colectivo cuando el coordinador desee modificar alguno de sus datos, hemos añadido un botón azul en la pantalla general del reto para acceder a esa funcionalidad. Dicho botón estará disponible mientras que el reto esté en estado “Propuesto”. La edición de un reto (*Figuras 120, 121 y 122*) básicamente es exactamente igual que la creación de un reto colectivo excepto que ya vienen con valores en cada uno de los campos. Además, dentro de esa funcionalidad, podemos indicar si deseamos iniciar el reto si este está en estado “Propuesto”. Una vez pulsado ese botón, el reto pasa a estado “En proceso”.
- En el caso de que el coordinador decida eliminar definitivamente el reto para todos, hemos incorporado un botón rojo que haga dicha acción.

- Cuando el coordinador se ha cansado de ser el encargado de ese reto o no puede continuar siéndolo, podemos darle la posibilidad de dejar la coordinación a otro participante a través del botón rosa. Como podemos ver en los bocetos de edición, únicamente mostramos el listado de los participantes que pueden ser posibles candidatos a coordinador, pinchar sobre uno de ellos para marcarlo como el elegido (que veremos con el fondo gris y con un botón a su derecha para, si nos arrepentimos de la elección antes de aceptarla, eliminarla) y luego pulsar el botón que confirma el cambio.

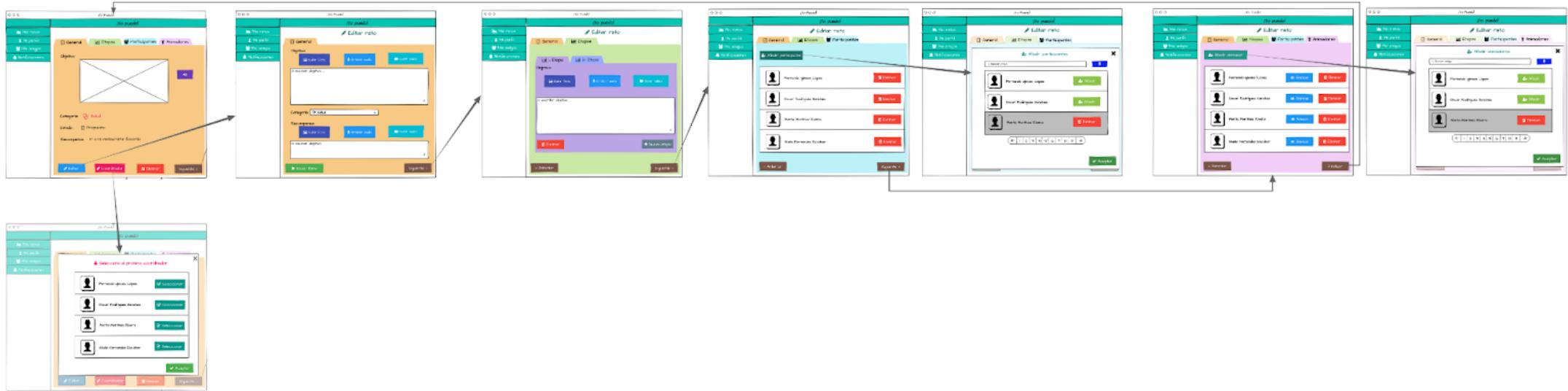


Figura 120: Edición y cambio de coordinador de un reto en formato escritorio.

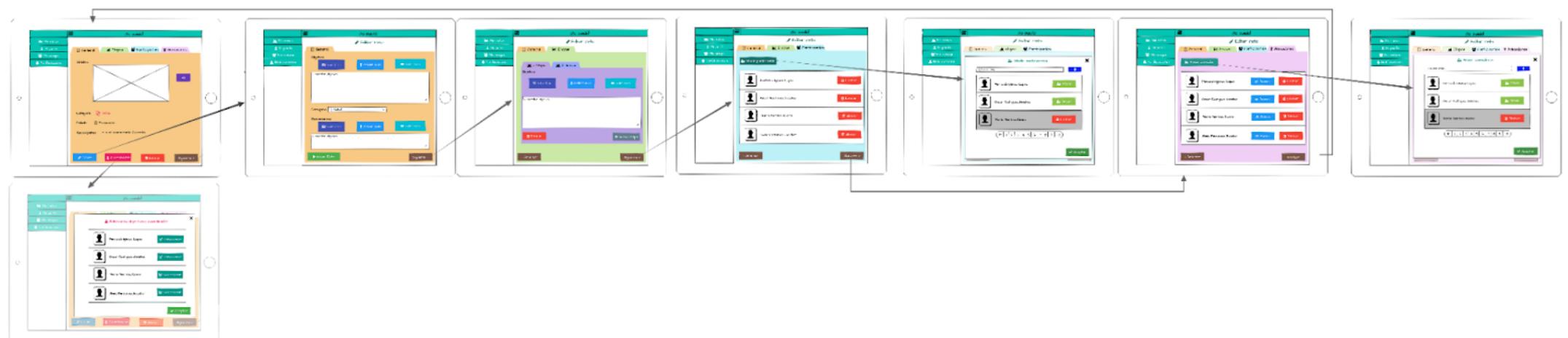


Figura 121: Edición y cambio de coordinador de un reto en tablets.



Figura 122: Edición y cambio de coordinador de un reto para móviles.

Además de las nuevas funcionalidades adjuntadas, hemos cambiado los símbolos de eliminar participante y animador por una papelera para que sea más fácil para el cliente identificar su función.

Base de datos

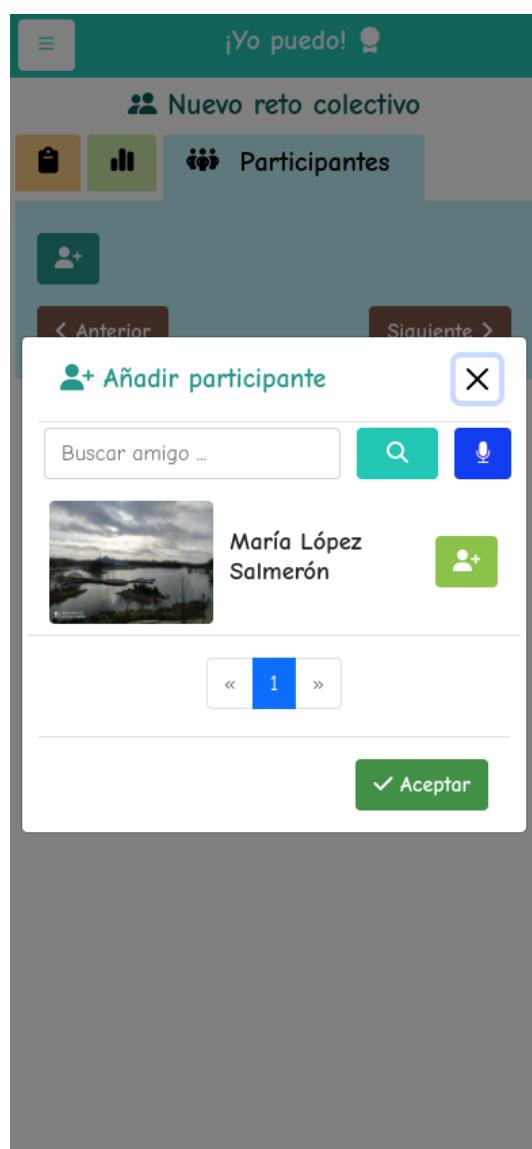
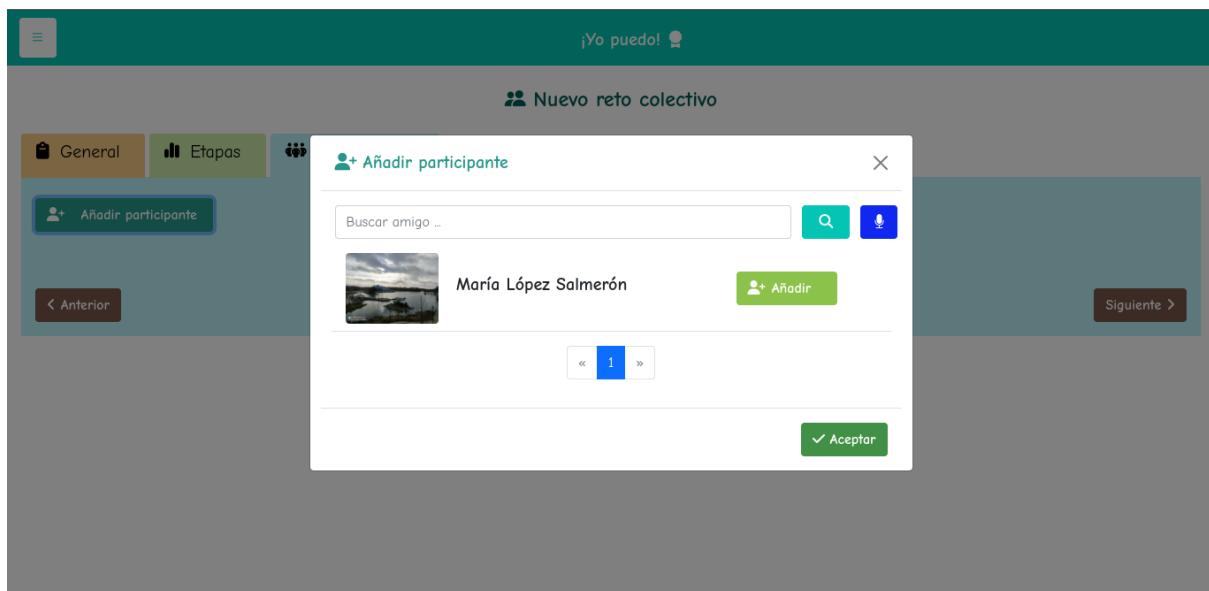
En cuanto a la base de datos que debemos crear para poder implementar las historias de usuario acordadas en este sprint, las relaciones que necesitamos para poder efectuar las funcionalidades acordadas ya se encuentran generadas en el cuarto sprint, mostrado en la *Figura 89*. Por lo tanto, no es necesario tener que crear desde cero la entidad con sus relaciones en esta etapa de desarrollo.

Implementación

Durante esta iteración, nos encargamos de realizar las funcionalidades de crear un reto colectivo (realmente únicamente implementar la inserción y la eliminación de participantes al reto), editar, iniciar y eliminar un reto y, por último, cambiar de coordinador.

Inserción, eliminación y listado de participantes

Como ya hemos mencionado anteriormente, cuando hablábamos sobre los bocetos de la inserción de participantes en un reto, se realiza de la misma forma que cuando se hizo la funcionalidad de [insertar, eliminar y listar los animadores](#) que van a formar parte del reto cuando se esté creando un reto colectivo o se esté editando un reto (se desarrolló dentro de la pestaña *Participantes*, a través de un modal como el de la *Figura 123*, que se activará dentro del botón *Añadir participante* dentro de la visualización de participantes de la *Figura 124*, donde el usuario puede consultar y señalar los amigos mediante el botón *Añadir* que él desee para formar parte del reto, cuando se pulse sobre *Aceptar*, que se mostrarán y se guardarán temporalmente dentro del reto, como en la *Figura 124*, y podrán ser eliminados gracias al botón *Eliminar* que se encuentra en cada uno de los supuestos participantes). Estas implementaciones se basaron en los bocetos diseñados de las *Figuras 114, 115 y 116*.



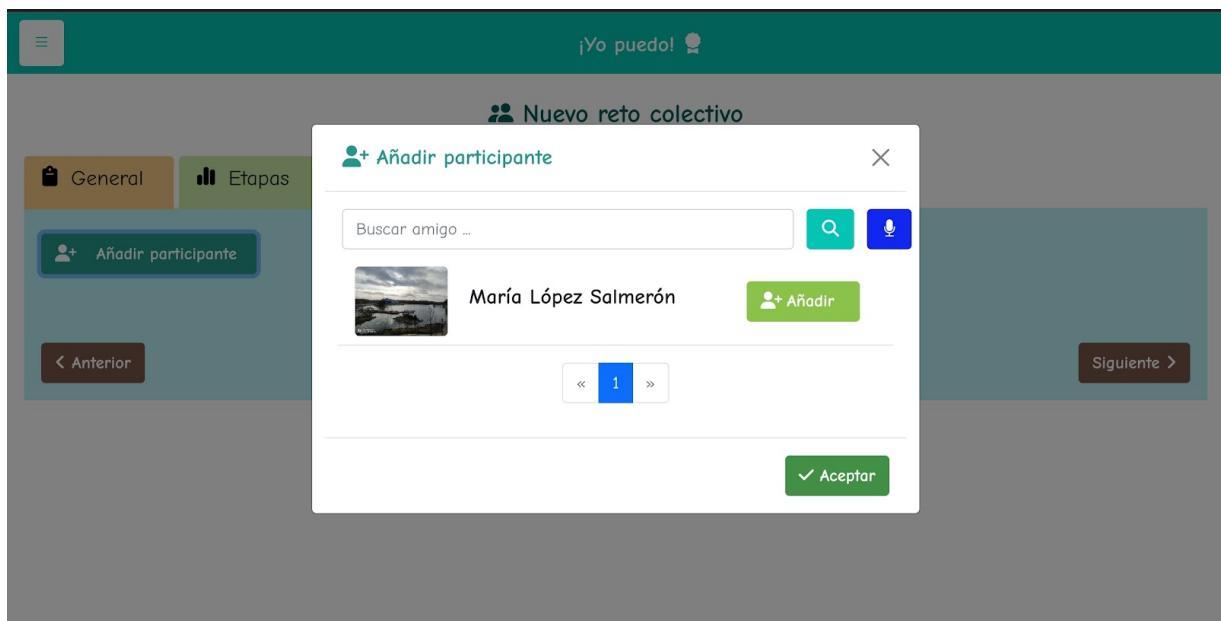


Figura 123: Modal para añadir participantes al reto en formato ordenador, móvil y tablet.





Figura 124: Listado de participantes en formato móvil, ordenador y tablet.

Edición, eliminación e inicio de retos y cambio de coordinador

Con respecto a la implementación de las funcionalidades que puede hacer un coordinador sobre un reto, tales como editar, cambiar de coordinador (si es un reto colectivo), iniciar el reto o eliminarlo, podemos acceder a través de los botones que mostramos en la *Figura 125*, que se encuentran al final de la página general de un reto dado. Se distribuyeron dichos botones en una misma gracias a las propiedades que tiene implementadas el marco de frontend *Bootstrap*.



Figura 125: Botones con las distintas funcionalidades que se pueden hacer sobre el reto (cuando el reto es colectivo).

Para eliminar el reto dentro de la aplicación es sencillo, solamente el usuario, si es el coordinador del reto dado, debe pulsar sobre el botón *Eliminar* junto al ícono de la papelera, que se encarga de llamar al backend para que elimine el reto y sus correspondientes informaciones (mensajes de ánimo, evidencias, calificaciones, ...) del sistema.

Sin embargo, para iniciar el reto el usuario debe primero pulsar al botón de *Editar* y, cuando la aplicación haya cargado la pantalla correspondiente de editar un reto (como el de la *Figura 126*), pulsar sobre el botón de la esquina inferior izquierda llamado *Iniciar*, junto al ícono del play. Cuando el usuario pulsa este botón, el backend únicamente lo que hace es cambiar el estado de ese reto y de la primera etapa al estado *En proceso* y redirigir a la visualización del reto.

¡Yo puedo!

Editar reto

General

Título:
Exponer en público

Objetivo:

O escribe el objetivo ...

Categoría:
Miedos

Recompensa:

Estar al mando de proyectos interesantes

¡Yo puedo!

Editar reto

General

Título:
Correr una maratón

Objetivo:

O escribe el objetivo ...

Categoría:
Deporte

Recompensa:

Viajar por el mundo en busca de maratones

The screenshot shows a user interface for editing a challenge. At the top, there's a green header bar with the text '¡Yo puedo!' and a profile icon. Below it, a blue bar says 'Editar reto'. The main area is titled 'General' and contains the following fields:

- Título:** Exponer en público
- Objetivo:** Subir foto (button), OBJETIVO.mp3 (button), Subir vídeo (button)
- Categoría:** Miedos
- Recompensa:** Subir foto (button), Subir audio (button), Subir vídeo (button)

At the bottom are two buttons: 'Iniciar reto' (Start challenge) and 'Siguiente >' (Next >).

Figura 126: Pantalla “General” en la edición de un reto en ordenador, móvil y tablet.

Como hemos dicho antes, el botón *Editar* nos redirige al formulario para editar los datos que forman parte del reto (como su información general, las etapas, los animadores y los participantes del reto). Este botón siempre estará accesible mientras el reto esté en estado *Propuesto*.

En cuanto al formulario que se implementa para editar un reto (mostrado a través de la Figura 126, que se basó en las Figuras 120, 121 y 122), es exactamente el mismo que explicamos en su momento para [crear un reto individual](#), añadir o eliminar [animadores](#) y [participantes](#), excepto que el formulario dado ya tiene rellenos los datos.

Cuando se termina de editar un reto, el backend se encarga de validar que los datos aportados corresponden a las condiciones impuestas y se procede a actualizar los registros correspondientes dentro de la base de datos.

Acerca del cambio de coordinador, el coordinador de ese reto en ese instante debe clickear sobre *Coordinador* (el segundo botón de la Figura 125), que se encarga de lanzar un modal con *HTMX* y *Bootstrap*, como el modal encomendado de añadir amigos como [animadores](#) o [participantes](#) al reto, donde puede consultar y elegir el futuro coordinador de los participantes del reto.

¡Yo puedo! 🎉

Exponer en público

General

Objetivo:

Categoría: Miedos

Recompensa: Estar al m...

Editor

Cambiar coordinador

Buscar amigo ...

María López Salmerón

Seleccionar

« 1 »

Aceptar

Siguiente >

The image shows a mobile application interface with a dark teal header containing the text "¡Yo puedo! 🎉". Below the header, there's a title "Exponer en público" with a checkmark icon. On the left side of the screen, there's a sidebar with a "General" tab selected. Inside the sidebar, there are sections for "Objetivo:", "Categoría: Miedos", and "Recompensa: Estar al m...". A blue "Editor" button is also visible. Overlaid on the main content is a white modal dialog titled "Cambiar coordinador". It contains a search bar with the placeholder "Buscar amigo ...", a magnifying glass icon, and a microphone icon. Below the search bar is a list item showing a small profile picture of a person and the name "María López Salmerón". To the right of the name is a green button with a checkmark icon labeled "Seleccionar". At the bottom of the modal are navigation arrows ("«", "1", "»") and a large green button with a checkmark icon labeled "Aceptar". In the top right corner of the main screen, there's a purple button with a speaker icon. In the bottom right corner of the main screen, there's a brown button labeled "Siguiente >".

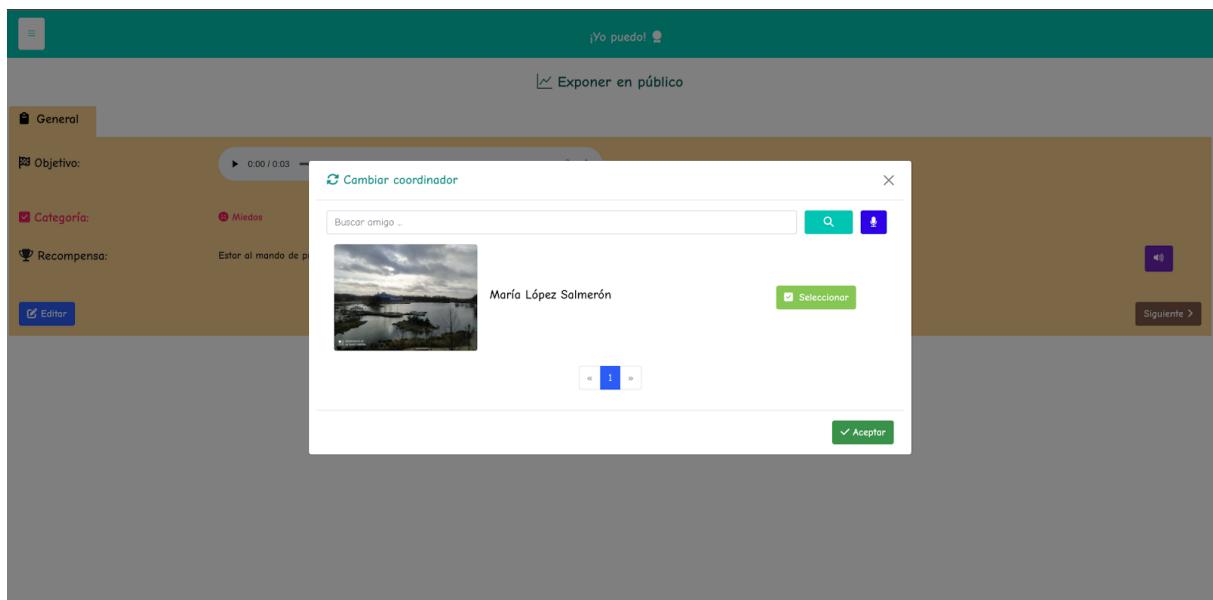
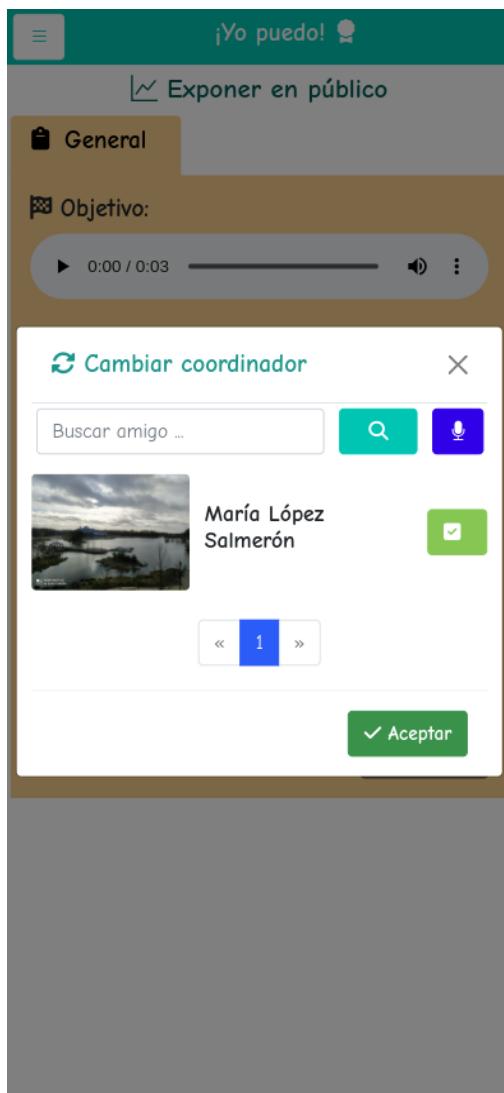


Figura 127: Modal de cambio de coordinador en tablet, móvil y ordenador.

Pruebas

Con respecto a las pruebas realizadas en esta iteración, los tests que hemos creado para comprobar que cada funcionalidad implementada hace lo esperado son:

- La accesibilidad de cada uno de los elementos de las páginas que engloban las historias de usuario marcadas, tal y como explicamos en el [ANEXO 2](#).
- Los usuarios que se muestran en los modales de añadir participantes o animadores son amigos del coordinador del reto.
- Los usuarios que se muestran en el modal de cambio de coordinador son participantes del reto dado.
- Devuelven los usuarios según la consulta realizada.
- Solo puede editar, eliminar, cambiar de coordinador e iniciar el reto el coordinador, si ha iniciado sesión previamente.
- Cuando se inicia un reto, el estado del reto y de la primera etapa están *En proceso*.
- Cuando se añade un participante, se encuentra dentro de la base de datos y es accesible.
- Cuando se elimina un participante o un animador, no puede volver a entrar al reto.
- Cuando se cambia de coordinador, un participante pasa a ser el nuevo coordinador y el antiguo solamente un participante.
- Cuando se edita un reto, antes de guardarla debe cumplir con las validaciones que cuando se crea un reto desde cero, las cuales fueron realizadas en el [tercer sprint](#).

Retrospectiva

Acerca de la retrospectiva de esta iteración, hemos realizado todas las historias de usuario dentro del plazo acordado en el [sprint backlog](#) al ser alguna de ellas similares a las realizadas anteriormente (como añadir participantes, editar parte del reto o cambiar de coordinador). Aunque hemos dejado la parte de notificar a los participantes que han sido añadidos para el último sprint del proyecto para realizar la parte de notificaciones, tanto la creación como la visualización en un único sprint.

3.14. Octavo Sprint

En cuanto a la última etapa de desarrollo del proyecto, nos encontramos con las tareas que nos faltan para poder ver las notificaciones que se crearon al realizar alguna de las actividades de otros sprints. Recordemos que las notificaciones vienen por parte de la aplicación cuando se hacen cambios en cuentas, amigos y participantes.

Historias de Usuario

- [HU36] Como persona, quiero consultar mis notificaciones.
 - Quién: Persona.
 - Cuándo: Cuando quiera ver sus notificaciones.
 - Entonces: Se muestra una lista de notificaciones de esa persona.
 - CoS:
 - La persona debe haber iniciado sesión previamente.
 - Se devuelve una lista de notificaciones.
 - Estarán ordenados según se han ido recibiendo.
 - Primero se mostrarán las notificaciones sin leer y después las ya leídas.
 - Las notificaciones sin leer estarán en negrita y las ya leídas en tamaño normal.
 - Cada una de las notificaciones mostrará la siguiente información:
 - El tipo de notificación: nueva solicitud de amistad, insertado en un nuevo reto como participante o como animador y nuevo mensaje de ánimo recibido en uno de sus retos.
 - El mensaje de la notificación.
 - El mensaje de la notificación debe ser dictado al usuario.
 - Si se pulsa sobre la notificación, te redirige a la página esperada según el tipo de notificación que sea la elegida.
 - [HU37] Como persona, quiero ver detalladamente una notificación recibida.

- Quién: Persona.
- Cuándo: Cuando quiera ver detalladamente una notificación.
- Entonces: Se muestra el detalle de la notificación.
- CoS:
 - El usuario debe haber iniciado sesión.
 - Debe existir la notificación seleccionada.
 - Se cambia el estado de la notificación a “Leída”.
 - Se devuelve el detalle de la notificación según el tipo que sea:
 - Si es una solicitud de amistad, se devolverá el perfil del posible amigo para poder aceptar o rechazar su solicitud.
 - Si es que el usuario se le ha añadido a un reto o que ha recibido un mensaje de ánimo de uno de sus retos, se devolverá la información de dicho reto.
- **[HU38] Como persona, deseo conocer si tengo notificaciones nuevas.**
 - Quién: Persona.
 - Cuándo: Cuando quiera ser informada de tener nuevas notificaciones.
 - Entonces: Se notifica al usuario, visualmente, si hay notificaciones nuevas.
 - CoS:
 - El usuario debe haber iniciado sesión.
 - Debe tener alguna notificación en estado “Recibida”.
 - Se indicará a través de un símbolo llamativo al lado del apartado que lleve a visualizar el listado de notificaciones.

Bocetos

Al realizar retos colectivos o añadir participantes, al añadir animadores a los retos, al recibir mensajes de ánimo en uno de sus retos como participante y al querer añadir nuevos amigos a la cuenta de un usuario, es necesario que avisamos a los usuarios

correspondientes de las novedades de sus perfiles. Es por eso que se crea una pantalla en donde estén todas las notificaciones del usuario.

Cada tipo de notificación tendrá un símbolo distinto para que el usuario sea capaz de diferenciar unas de otras (un reto nuevo como participante tendrá el símbolo de una gráfica, el de una nueva solicitud de amistad se representará con una persona más el símbolo + y el de un mensaje de apoyo será una carta).

Al pulsar sobre el botón con el símbolo “>” de una notificación (y así cumplir con una de las normas de la propiedad comprensible y perceptible explicadas en el [ANEXO 2](#)), nos lleva a la página correspondiente de dicha notificación. Si es de un mensaje de ánimo o de un nuevo reto como animador o participante, nos llevará directamente a la visualización del reto. Mientras que si es una notificación de amistad, visualizará el perfil de un usuario como los bocetos mostrados en las *Figuras 128, 129 y 130*.

Si accedemos a una notificación de una solicitud de amistad, tenemos la posibilidad de aceptarla o negarla. Si pulsamos sobre el botón “Aceptar”, para asegurarnos que tienen el permiso del tutor o que sea el propietario de la cuenta el que realiza dicha acción, pedimos que escriba el código aleatorio que se ha enviado a su email. Una vez aceptada, podemos ver su perfil.

Una vez aceptada la solicitud, nos dirige al perfil de ese usuario, como el que ya habíamos hablado en el quinto sprint (mostradas en las *Figuras 94, 95 y 96*).

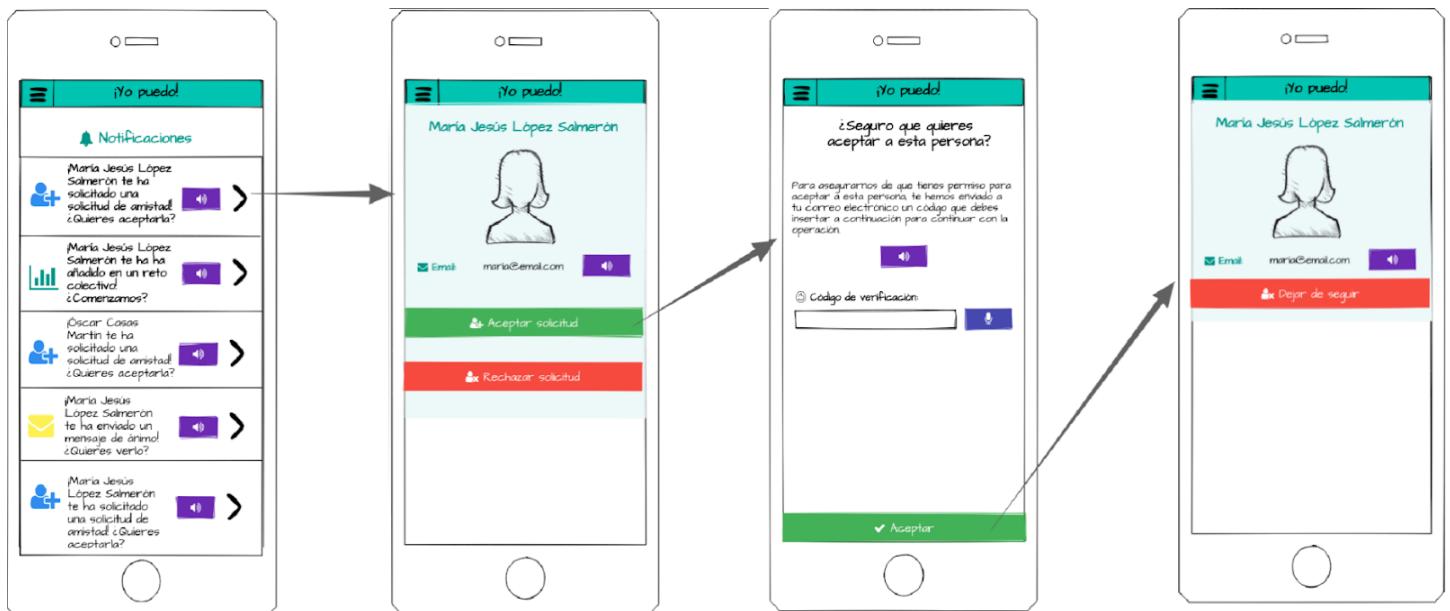


Figura 128: Primera versión de notificaciones y aceptación de solicitud de amistad en formato móvil.

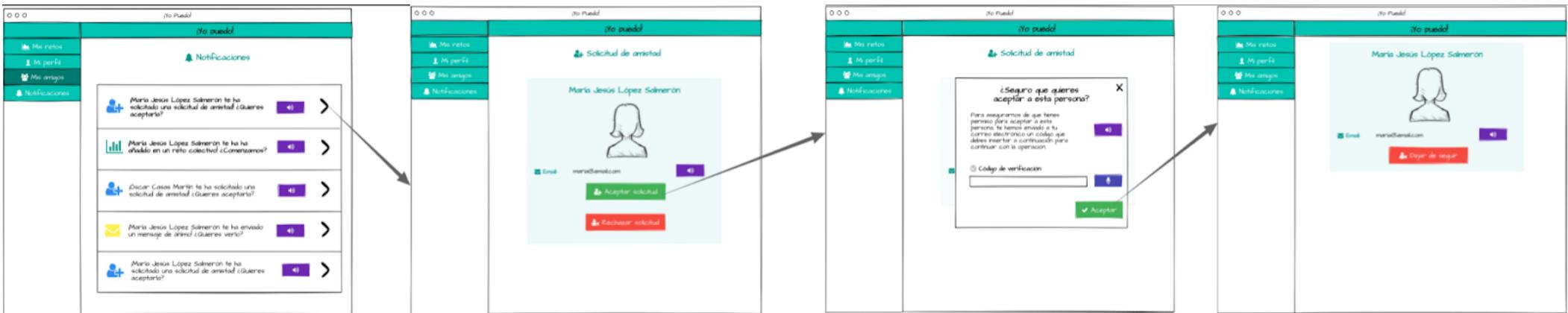


Figura 129: Primera versión de notificaciones y solicitud de amistad para ordenadores.

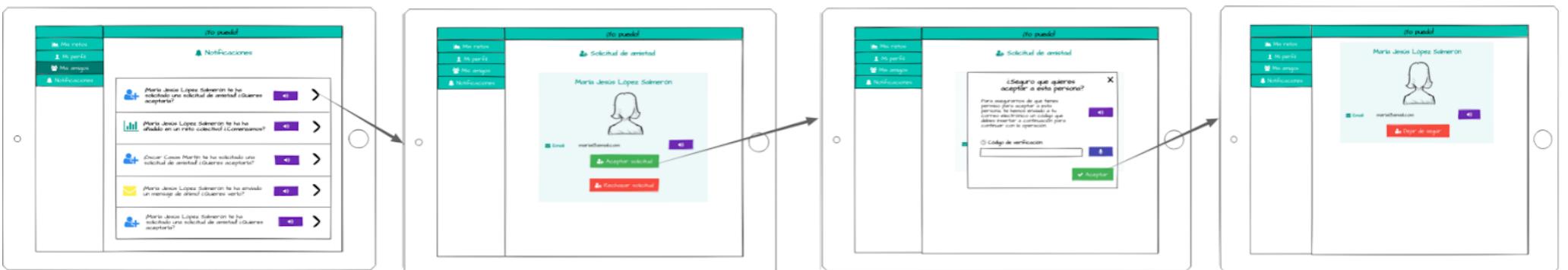


Figura 130: Primera versión de notificaciones y aceptación de solicitud de amistad en tablets.

Como en el listado de amigos y en la visualización de su perfil, durante una reunión con el cliente acordamos que debemos cambiar el símbolo elegido para negar y dejar de seguir por el de la papelera (como mostramos en los bocetos de las *Figuras 131, 132 y 133*).

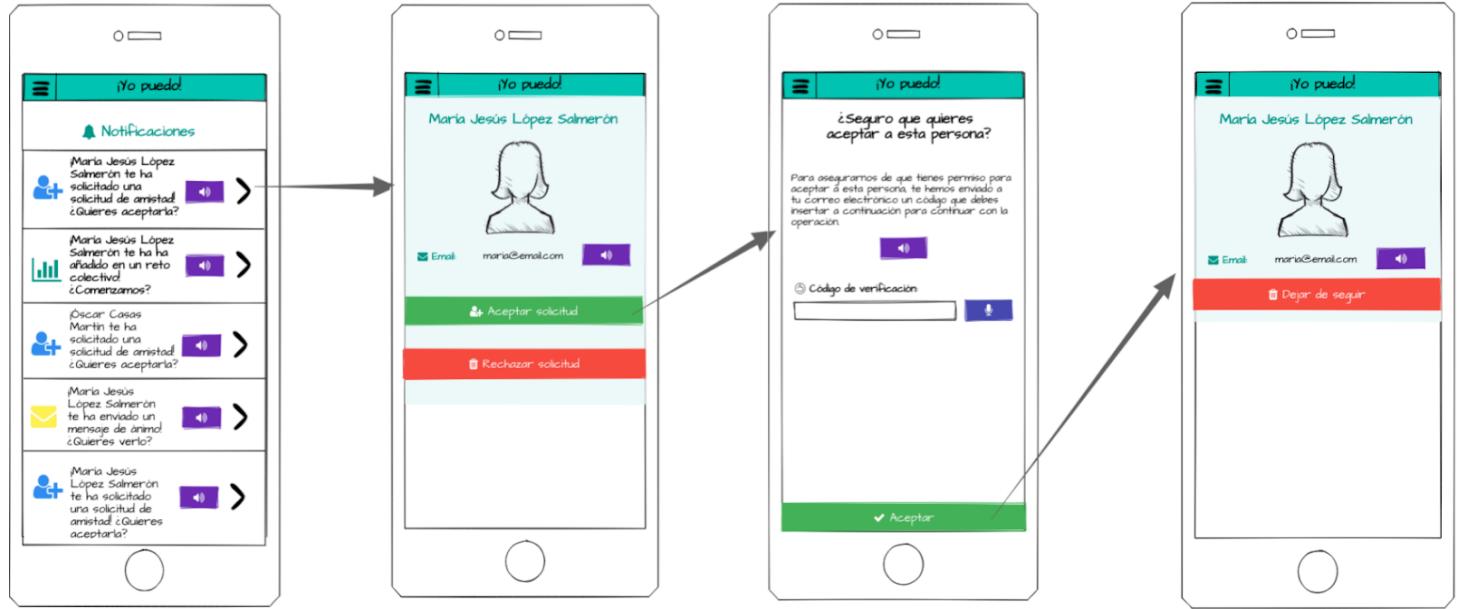


Figura 131: Segunda versión de notificaciones y solicitudes de amistad en formato móvil.

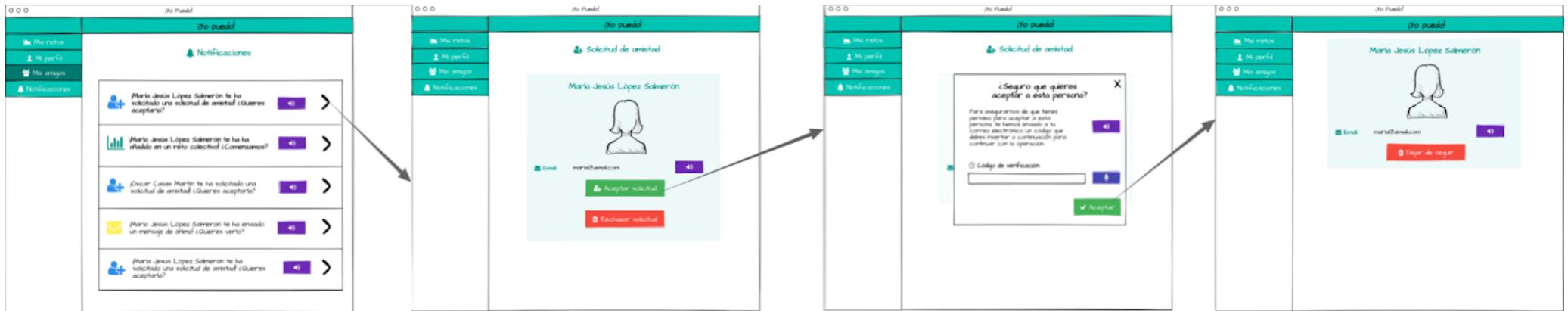


Figura 132: Segunda versión de notificaciones y solicitudes de amistad en escritorio.

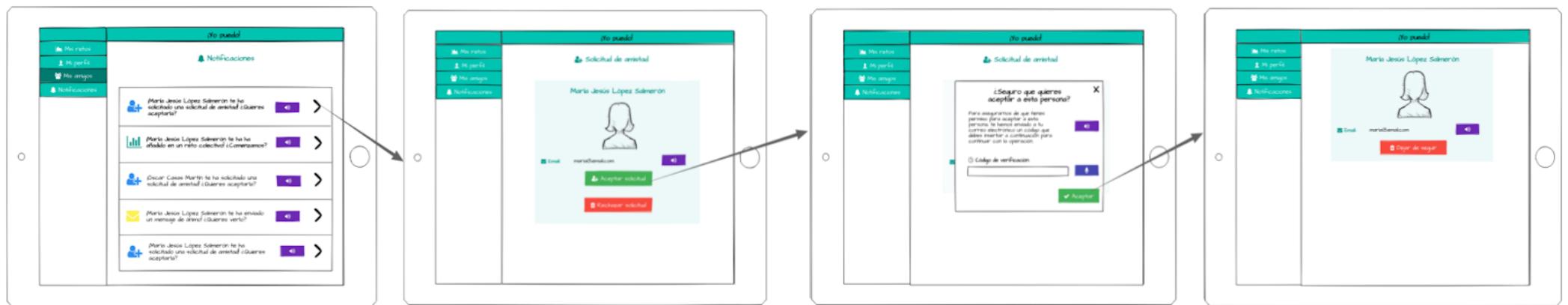


Figura 133: Segunda versión de notificaciones y solicitudes de amistad para tablets.

Base de datos

Con respecto a la creación de la base de datos necesaria para poder guardar y manejar los datos que tiene una notificación, la siguiente figura que vamos a mostrar es la parte del diagrama de Entidad/Relación que describe los datos utilizados para las historias de usuario anteriores.

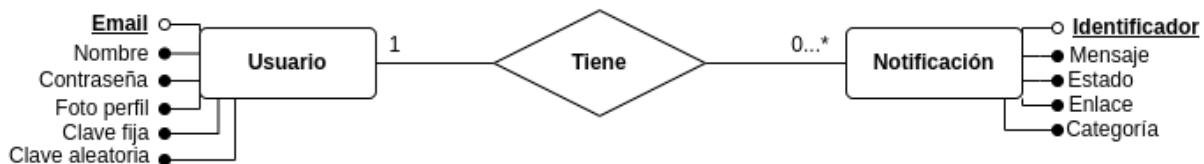


Figura 134: Diagrama de Entidad/Relación del último sprint del proyecto.

En este diagrama encontramos con dos entidades, *Usuario*, que ya estaba y se encarga de manejar la información de una persona dentro del sistema, y *Notificación*, que es la nueva entidad añadida para de almacenar los datos de una notificación que genera el sistema sobre una acción realizada por otro usuario sobre su cuenta. Ambas entidades están unidas por una relación llamada *Tiene*.

Implementación

En este último sprint del proyecto, realizamos las distintas funcionalidades que tiene una notificación: su creación (cuando añadimos un participante o un animador del reto, un animador envía un mensaje de apoyo a uno de los retos del usuario o cuando un usuario envía una petición de amistad a otro), el listado de las notificaciones y su visualización.

Listado de notificaciones

Con respecto al listado de notificaciones, mostramos una fila por notificación en la que se dividen en tres partes: la primera es la categoría de la notificación (invitación a un reto como participante o como animador, un nuevo mensaje de ánimo dentro de uno de los retos como participante o una nueva petición de amistad), la segunda un mensaje explicativo de la notificación con la posibilidad de escucharlo (a través de la síntesis de voz) y, por último, un enlace en forma de botón con el icono >, que redirige a la página especificada según la notificación (al reto si es la incorporación al reto o un nuevo mensaje de ánimo, o al perfil del usuario que ha solicitado la amistad). Este listado está basado en la primera pantalla de los bocetos realizados en las *Figuras 131, 132 y 133*, que fueron implementados gráficamente como las capturas de pantalla de la *Figura 135*.

Dentro del listado mostramos primero las notificaciones que se han recibido (las que tienen el mensaje en negrita, y después las leídas. Ambas están ordenadas según el orden que hayan sido creadas desde las más nuevas a las más antiguas.

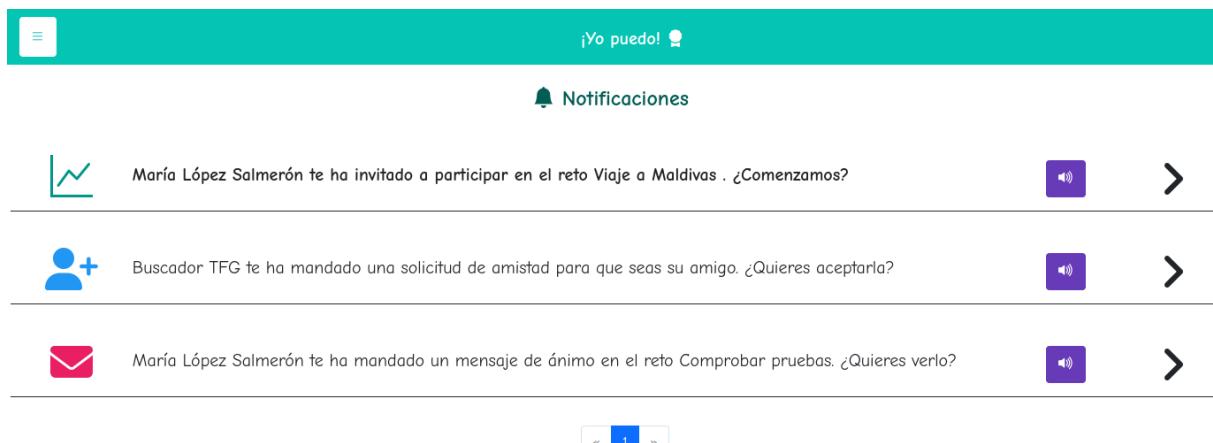




Figura 135: Listado de notificaciones para móvil, ordenador y tablet.

Para que se muestren las notificaciones, primero hay que crearlas y guardarlas en la base de datos para ser consultadas posteriormente. Como ya dijimos en las retrospectivas de los tres últimos sprints (cuando enviamos una petición de amistad, cuando añadimos un participante o un animador al reto y cuando enviamos, desde el rol de animador, un mensaje de amistad), implementamos la creación de las notificaciones en esta iteración.

Para ello, únicamente creamos la notificación en su correspondiente funcionalidad eligiendo la categoría, el mensaje y el enlace de la página a la que debe redirigir y guardándola en la base de datos dentro de la entidad *Notificación* (según creamos en el diagrama de Entidad/Relación de la Figura 134):

- En el caso en el que se han insertado nuevos participantes o animadores, se crea una notificación con la categoría del ícono de la gráfica, junto al mensaje de que se le ha invitado a formar parte de un reto y al enlace para visualizar el reto correspondiente cuando se guardan los usuarios a la tabla correspondiente en la base de datos.
- Con respecto al envío de nuevo mensaje de apoyo, es similar al anterior excepto que la categoría es el ícono de un sobre y el mensaje de la notificación informa de un nuevo mensaje de apoyo de un animador.
- En cuanto a la petición de amistad, una vez que el usuario haya confirmado a la aplicación que desea ser amigo de los usuarios elegidos, en el backend se envía a

cada usuario seleccionado la notificación con la categoría del ícono de una persona con un +, el mensaje informando de que un usuario quiere formar parte de tu grupo de amigos y un enlace con el perfil del usuario, que lo explicaremos en la siguiente sección.

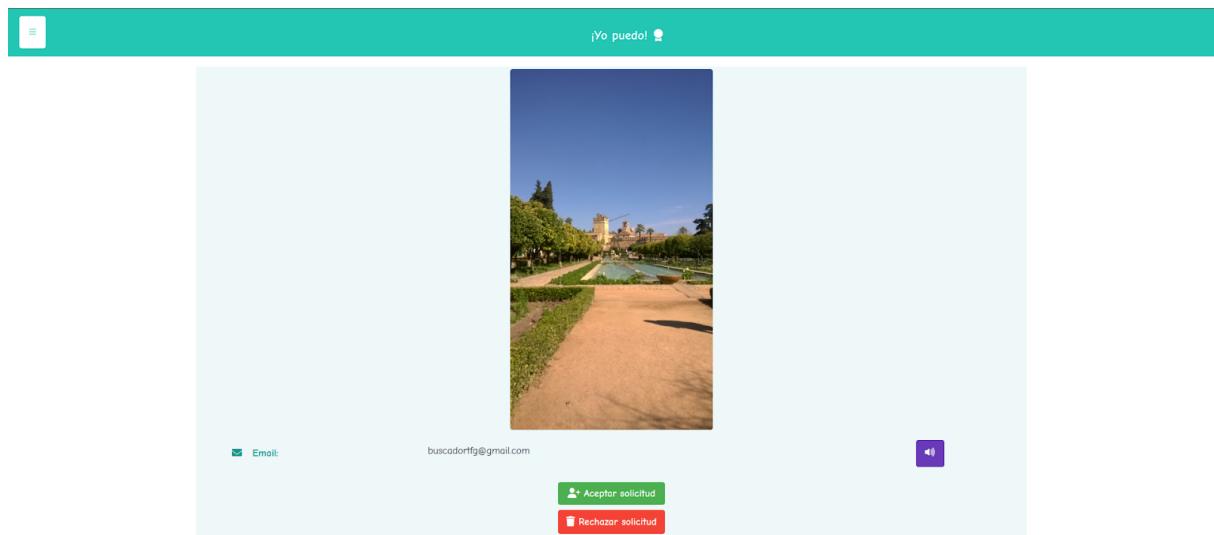
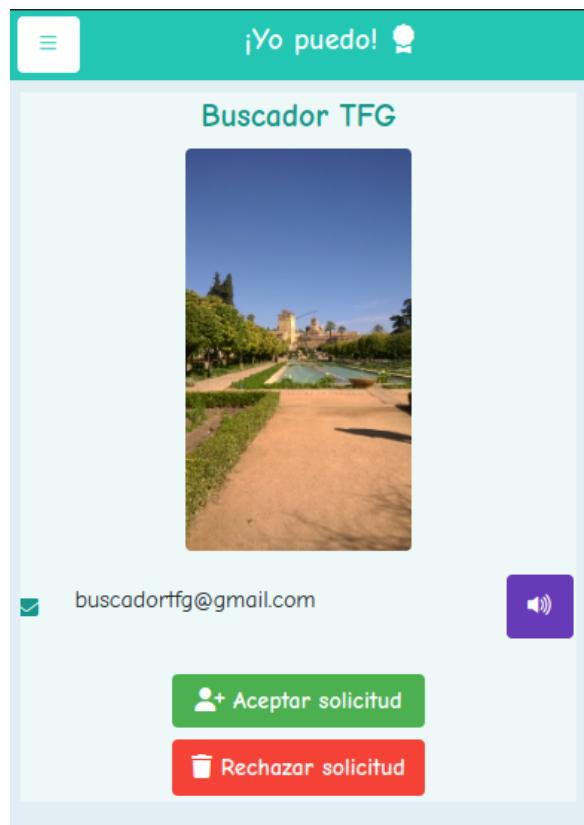
Para informar al usuario, dentro de la aplicación, de que tiene notificaciones sin leer, dentro del menú desplegable de la aplicación (como el que mostramos en su día en la *Figura 81*, al final de la implementación del [listado de retos y sus filtraciones](#)), añadimos un punto rojo con la cantidad de mensajes sin leer al lado de *Notificaciones*, como podemos ver en la *Figura 136*. Para saber la cantidad de mensajes que hay sin leer, realizamos una consulta en la base de datos para que nos devuelva, a través de COUNT, la cantidad de registros de ese usuario que tiene como estado *Recibido*.



Figura 136: Ejemplo de alerta de notificaciones sin leer.

Petición de amistad

Acerca de la petición de amistad, cuando el usuario pulsa el enlace correspondiente a una notificación de petición de amistad, el backend redirige a la visualización del perfil de un usuario (como el de la *Figura 137*), como ya implementamos el frontend en el [segundo](#) y [quinto](#) sprint, excepto que, en vez de tener los botones para dejar de seguir y una lista de retos o unos botones para realizar algún cambio a la cuenta o cerrar sesión, únicamente tiene dos botones con dos opciones distintas: aceptar la solicitud o rechazarla.



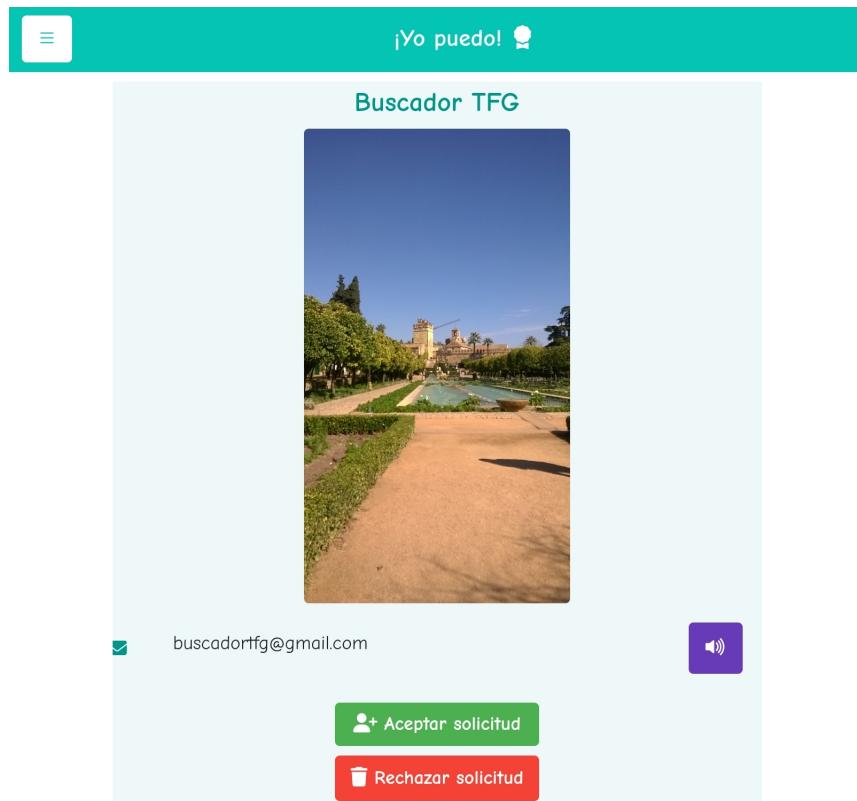
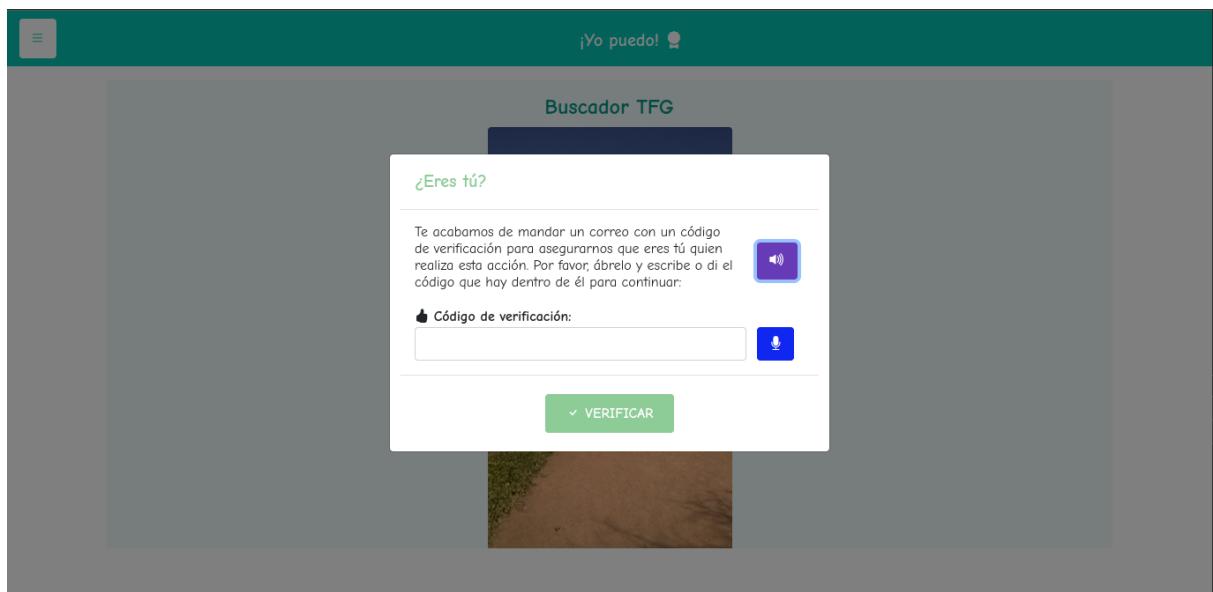
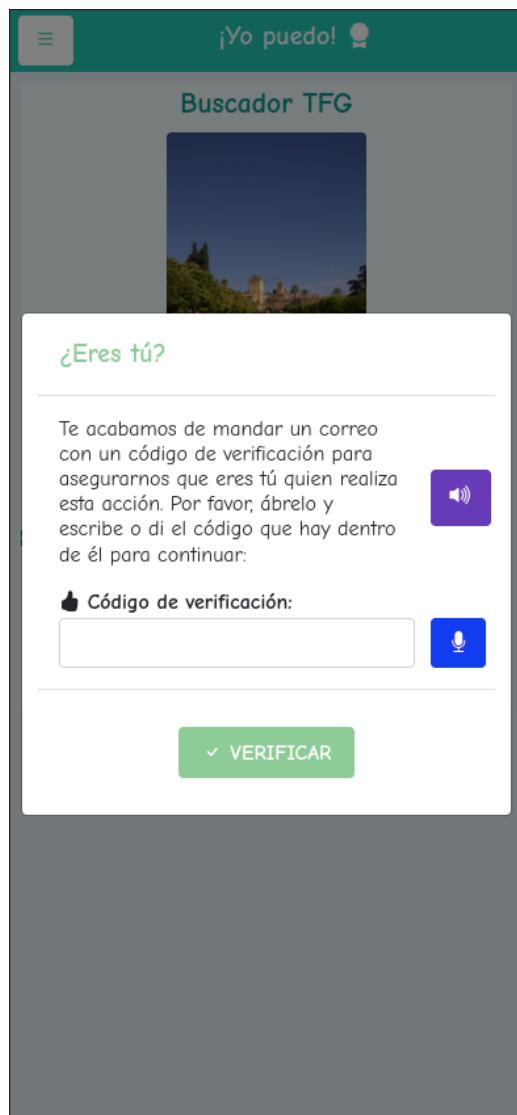


Figura 137: Ejemplo de petición de amistad en móvil, ordenador y tablet.

Si el usuario pulsa sobre el botón *Rechazar solicitud* con el ícono de una papelera, llama al backend para que rechace esa solicitud, o dicho de otra forma, elimina las notificaciones de amistad que tenga con ese usuario dentro de su cuenta.

En cambio, si el usuario decide aceptar la solicitud de amistad, debe pulsar sobre el botón *Aceptar solicitud* con el ícono de una persona más un + y el backend se encarga de confirmar su solicitud abriendo el modal del [pin parental](#), de la Figura 138, en el que tiene que escribir la clave mandada al correo electrónico (como el de la Figura 139), que previamente se creó y se guardó en la base de datos. Si la clave escrita es igual a la almacenada, se crea la unión entre los dos usuarios insertando un registro en la relación *Amigo*, que podemos ver en el diagrama de Entidad/Relación del [quinto sprint](#), en la Figura 97.

El diseño frontend de estas páginas fue generado, para cada dispositivo, en los bocetos (excepto la primera pantalla) que mostramos en las Figuras 131, 132 y 133.



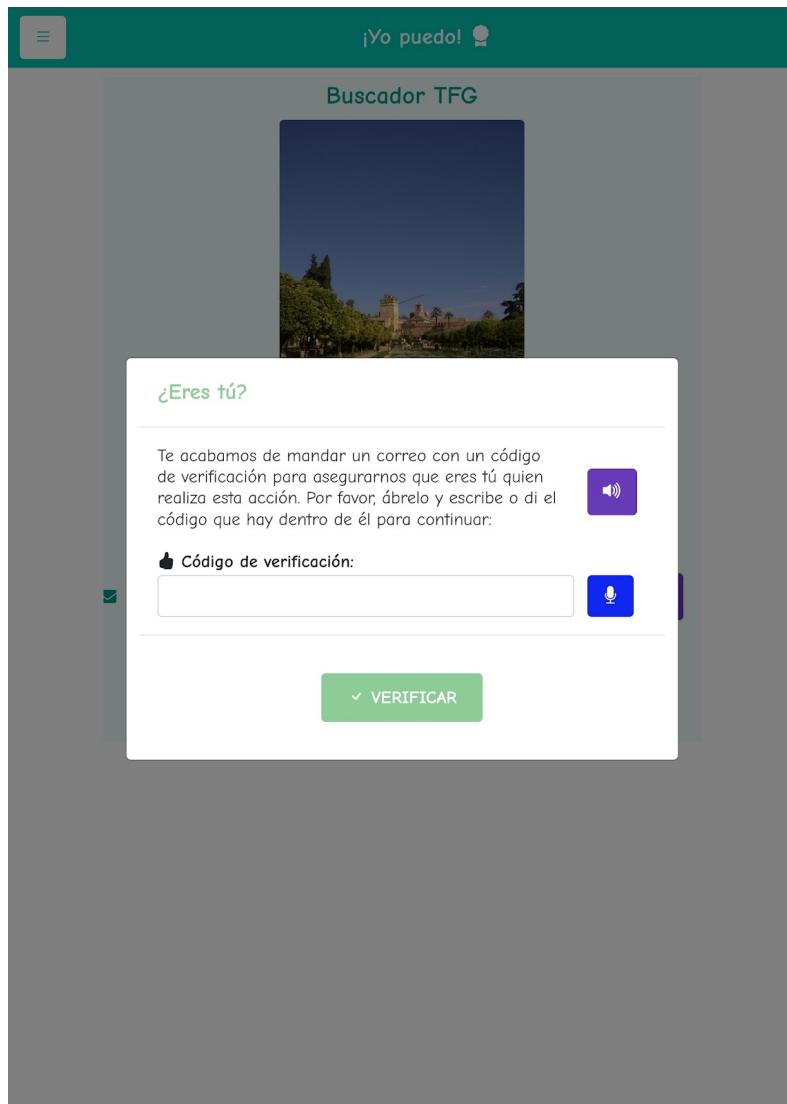


Figura 138: Confirmación de aceptación de amistad en móvil, ordenador y tablet.

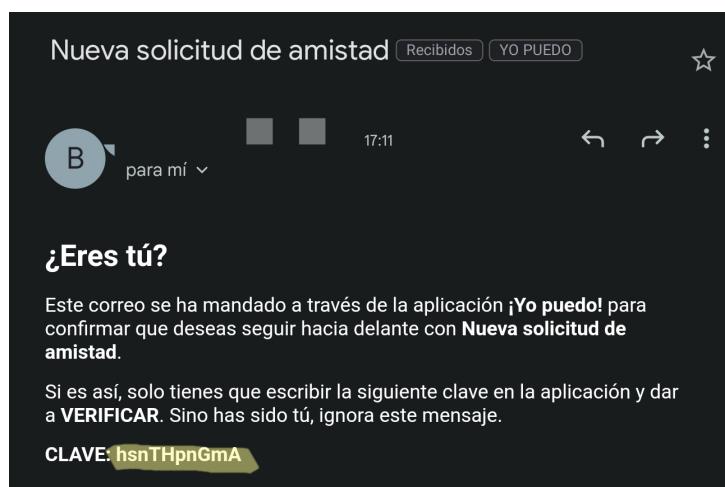


Figura 139: Ejemplo de correo electrónico para confirmar aceptación de amistad.

Pruebas

En cuanto a las pruebas realizadas en el último sprint del proyecto, las creadas son las que comentaremos a continuación:

- Con respecto al frontend que reflejan los datos resultantes y las funcionalidades de las historias de usuario relacionadas en este sprint, comprobamos que cumplen las normas de las [guías de accesibilidad](#) (tal y como lo justificamos en el [ANEXO 2](#)). Por eso generamos dos versiones distintas para que la aplicación fuera más comprensible (al cambiar el icono de una cruz por el de una papelera).
- Acerca de las pruebas realizadas al backend, decidimos llevar a cabo las siguientes:
 - Cuando se crea una notificación, verificamos que se ha guardado correctamente y que es accesible dentro de la aplicación.
 - Cuando creamos un reto colectivo o añadimos participantes al editar un reto, comprobamos que hay una notificación de invitación al reto para cada uno de los nuevos participantes.
 - Cuando añadimos a un reto (ya sea en su creación o en su edición) animadores, examinamos que hay una notificación de invitación al reto para cada uno de los nuevos animadores.
 - Cuando un animador manda un mensaje de ánimo dentro de un reto, verificamos que hay una notificación de ese mensaje para cada uno de los participantes del reto.
 - Comprobamos que únicamente pueden acceder al listado de notificaciones aquellos usuarios que hayan iniciado sesión previamente.

Retrospectiva

En conclusión, en este sprint hemos terminado de crear todas las historias de usuario planificadas, además de las que dejamos a lo largo de otras iteraciones (la creación de notificaciones cuando se realiza una acción sobre un reto o la cuenta de un usuario). Aún teniendo más tareas que las planificadas, se consiguió acabar antes del plazo acordado para este sprint, logrando acabar un poco antes la implementación de este proyecto.

CAPÍTULO 4. Conclusiones y Trabajos futuros

4.1. Conclusiones

Como conclusión de este proyecto, una vez investigados ciertos competidores y las tecnologías a utilizar, diseñado e implementado cada una de las funcionalidades necesarias para poder lanzar este producto al mercado, es momento de indicar si los objetivos y los requisitos planteados en la [Introducción](#) de esta memoria se han llevado a cabo.

Sobre los [objetivos](#) cumplidos a lo largo de este proyecto, los llevamos a cabo de la siguiente forma:

- En cuanto a la [planificación temporal](#) del trabajo (creada junto al diagrama de Gantt dentro del apartado de introducción), consideramos dos grandes apartados, la investigación y el desarrollo, los cuales los dividimos a su vez en más partes, tal y como comentamos a continuación.
 - En la investigación:
 - Buscar información acerca de las guías de accesibilidad para aplicaciones móviles y estudiar cada una de ellas.
 - Posteriormente, investigar sobre las aplicaciones que tuvieran requisitos similares a los que nosotros planteamos.
 - Y por último, pensar qué clase de tecnologías podríamos usar para desarrollar la aplicación y qué herramientas hay para cada tecnología.
 - En el desarrollo:
 - Identificar las funcionalidades que debe tener la aplicación implementada.
 - Seleccionar las tecnologías y herramientas de desarrollo más apropiadas.
 - Buscar qué metodología se adaptaría mejor a nuestra aplicación.
 - Definir las historias de usuario que cubran las funcionalidades anteriores.

- Ordenar las historias anteriores según su prioridad y agruparlas en sprints.
 - Para cada sprint, diseñar los bocetos y la base de datos y realizar su implementación y sus pruebas.
- Acerca de las guías de accesibilidad investigadas relacionadas con el desarrollo de aplicaciones móviles, revisamos las de dos organizaciones: la del [Ministerio Español de Asuntos Económicos y Transformación Digital](#) y de [WCAG](#). Para ambas estudiamos las distintas normas y propiedades que debemos aplicar a nuestra aplicación con respecto al frontend, que básicamente podemos agruparlas ambas en cuatro categorías: perceptible, robusta, entendible y operable.
- A continuación, buscamos distintas [aplicaciones con requisitos similares](#) a los que deseábamos desarrollar en nuestro proyecto y comparamos, revisando las similitudes y diferencias que tendrían con el nuestro: *Qapital*, *GoodReads*, *21 Challenge Days*, *Challenges* y *Habitus*.
- Realizada la investigación de guías de accesibilidad y de aplicaciones similares a la que deseamos crear, investigamos las tecnologías necesarias para desarrollar e implementar la aplicación que son:
 - [Frameworks](#): encontramos *Django*, *Flask* y *ExpressJS*, estudiamos sus características y, según las características halladas (como la utilización de ORM, plantillas implementadas y funcionalidades desarrolladas dentro del framework), elegimos *Django*.
 - [Bases de datos](#): decidimos que, por el tipo de problema que íbamos a resolver, la base de datos sea una relacional, por lo que vimos que podíamos utilizar *PostgreSQL*, *MariaDB* y *MySQL*. Elegimos utilizar *MySQL* por las características que este sistema gestor de base de datos ofrece y al ser uno de los que utiliza *Django*, además de su soporte a posibles problemas dentro de esa empresa.
 - [Alojamientos Web](#): buscamos aquellos alojamientos donde pudiera ser desarrollada la aplicación, junto a su base de datos y almacenamiento interno, y seleccionamos, de las comentadas en su correspondiente sección, *PythonAnywhere* al ser fácil desarrollar un proyecto *Django*, sin necesidad de centrarse en configuraciones complicadas, y al poder crear la demo en él de forma gratuita.

- [Marcos frontend](#): de los hallados por la red y que fueran aplicables a un proyecto en *Django*, encontramos *Bootstrap* que es gratuito y fácil de utilizar, además de incluir animaciones y ciertas funcionalidades en JavaScript.
 - [Herramientas de HTML](#): solamente encontramos dos que se ajustaban más a lo que necesitábamos (*HTMX* y *Alpine*), de las cuales decidimos utilizar *HTMX* al poder incorporarse en HTML y poder realizar peticiones HTTP a través de él.
 - [Iconos](#): hay varias herramientas y páginas webs donde podemos obtener los iconos para desarrollar la parte frontend del proyecto, pero elegimos *FontAwesome* al poder incluirse como un elemento HTML y al ofrecer los iconos que precisamos.
- Una vez identificadas las tecnologías que vamos a utilizar, desarrollamos el proyecto de la siguiente forma:
 - Diseñamos las [funcionalidades](#) que deseábamos plasmar en nuestra aplicación, que se dividieron en tres bloques: las funcionalidades relacionadas con los usuarios, con las amistades y con los retos.
 - Planificamos la [arquitectura](#) del sistema que tiene nuestro proyecto (con las tecnologías anteriores), indicando los usuarios que van a formar parte, los dispositivos con los que pueden acceder a la aplicación, las herramientas que crearán la parte frontend de la aplicación dentro del navegador y las que se encargan de crear la parte backend y su comunicación con la base de datos.
 - Elegimos la [metodología](#) para llevar a cabo el proyecto, que en este caso vimos que era preferible la metodología Agile para mostrar al cliente partes de la aplicación, ya usables, cada cierto tiempo, y así poder ir negociando cambios y mejoras con él.
 - Generamos las [historias de usuario](#) que implementan las funcionalidades creadas, las ordenamos según su prioridad y las [agrupamos](#) en ocho sprints.
 - Para cada sprint, desarrollamos la siguiente información:
 - Para cada tarea de la iteración, indicar las condiciones de satisfacción necesarias para desarrollarla en la aplicación.

- Crear los bocetos en donde se podrán llevar a cabo las funcionalidades anteriores, cumpliendo con las guías de accesibilidad investigadas (tal y como lo hemos explicado en el [ANEXO 2](#)).
- Diseñar, a través de un diagrama de Entidad/Relación, la base de datos que se necesita para almacenar y manejar los datos de la aplicación. El resultado de todas las versiones de ese diagrama, según vamos terminando iteraciones, lo encontramos en el [ANEXO 1](#).
- Generar el diagrama de clases que nos ayude a ver la forma en la que debería implementar las funcionalidades de la aplicación. El resultado final de unir todas las clases creadas para el proyecto la encontramos en el [diagrama de clases final](#) del ANEXO 1.
- Plantear los ficheros y las funciones que va a implementar cada uno de ellos, y que se van a utilizar en la aplicación dentro del framework seleccionado. Como resultado de implementar todas las historias de usuario, en el ANEXO 1 encontramos la [estructura final de ficheros](#) que encontramos dentro del proyecto de *Django*.
- Implementar las tareas, según se hayan detallado y según estén plasmadas en los correspondientes diagramas, estructuras y bocetos. En cada implementación, comentamos la forma en la que las realizamos, junto a capturas de pantalla del resultado de la implementación en los tres dispositivos.
- Diseñar y lanzar las pruebas necesarias para verificar que se cumple las condiciones de satisfacción planificadas de cada tarea.

Hemos implementado todos los [requisitos](#) planteados, los cuales pasamos a recordar, y explicaremos cómo los hemos abordado.

- *Utilizar la red social en cualquier dispositivo, sin importar tamaño, versión o sistema operativo.* Gracias a la implementación del framework de *Django* [19] junto a la herramienta de CSS de *Bootstrap* [22], pudimos crear una aplicación web que se puede utilizar en cualquier navegador y en cualquier dispositivo. Además, en cada uno de los sprints del [Product Backlog](#) del proyecto creamos los bocetos para los tres dispositivos utilizados con normalidad para este tipo de actividades (ordenador, tablet y móvil).

- *Generar una aplicación intuitiva y sencilla de utilizar.* Dentro del apartado de *Bocetos* de los sprints del [*Product Backlog*](#), diseñamos los distintos bocetos con las siguientes características:
 - Indicamos los nombres descriptivos de los campos de los distintos formularios que el usuario debe llenar junto a su ícono representativo
 - Insertamos un título y/o un ícono que detalle brevemente la acción que va a realizar o la información que va a mostrar cada uno de los botones que hay en la aplicación
 - Mostramos el título, con un ícono explicativo, de lo que se está mostrando por pantalla o de la acción que está haciendo un usuario en una ventana dada.
 - Mostramos la información (tanto la que se quiere visualizar como la que el usuario debe completar) poco a poco. Por ejemplo, en los [*bocetos*](#) de la creación de un reto individual.
- *Poder ser utilizada por cualquier persona, aunque esta tenga ciertos problemas físicos.* Para ello investigamos [guías de accesibilidad del momento](#) para ver cómo debería ser la aplicación gráficamente antes de diseñar los distintos bocetos del [*Product Backlog*](#) y cómo deberíamos implementarla, una vez desarrollados éstos. Podemos ver en más detalle cómo hemos cumplido con cada una de las normas que debe tener una aplicación, como la de este proyecto en [ANEXO 2](#).
- *Crear cualquier reto indicando el objetivo, su categoría y su recompensa.* Este objetivo lo podemos ver implementado en las funcionalidades de la HU8 (del [tercer sprint](#) - creación del reto individual) y de la HU29 (del [séptimo sprint](#) - creación del reto colectivo) dentro de la pestaña “General” de ambas historias, encargadas de introducir estos datos antes de detallar en más profundidad sobre el reto. Además, hemos visualizado dicha información dentro de la HU12, también dentro de una pestaña con ese mismo nombre.
- *Decidir los pasos a realizar para alcanzar la meta de un desafío.* También en las historias HU8 (del [tercer sprint](#) - creación del reto individual), HU29 (del [séptimo sprint](#) - creación del reto colectivo) y HU32 (también del séptimo sprint - edición de un reto) hemos añadido otra pestaña (denominada “Etapas”), en la que se desglosa a su vez en subpestañas, en las que el usuario indica el objetivo que debe superar en cada paso creado para lograr el objetivo principal del reto. Además, en la historia H12 (encargada de visualizar la información de un reto) también mostramos al

usuario esa misma pestaña junto a sus subpestañas para que así sepa los pasos que va a realizar para alcanzar la meta, una vez creado el reto.

- *Añadir amigos a mi desafío para que lo hagan conmigo.* Implementamos la funcionalidades de las historias HU29 (creación del reto colectivo) y HU30 (inserción de participantes un un reto individual) del [séptimo sprint](#) en las que, además de las pestañas con datos que se necesitan para crear un nuevo reto o para guardar uno previamente creado, insertamos otra nueva pestaña, llamada "Participantes". En ella mostramos los amigos seleccionados como participantes, y también dónde podemos añadir a más amigos pulsando el botón correspondiente y seleccionando a otros usuarios de la lista de amigos del propietario del reto. Una vez guardados los participantes dentro de un reto, se muestran también dentro de la funcionalidad de H12 (visualización de un reto).
- *Incorporar amigos para que me apoyen, a través de mensajes de ánimo, durante el desarrollo del reto.* En primer lugar, el usuario debe indicar cuáles son los animadores del reto. Esta funcionalidad la implementamos en el [sexto sprint](#), concretamente en la HU23. Dicha selección se realiza cuando el reto se está editando o creando desde cero en la pestaña "Animadores", de la misma manera en la que se insertaron los participantes. En esta pestaña, también el usuario puede indicar qué amigo quiere que vea todos los mensajes de ánimo (dicho de otra forma, ser el superanimador) o no. En cuanto al envío de mensajes, detallado en la HU25 (también en el [sexta sprint](#)), lo implementamos cuando se está visualizando la información del reto, en concreto cuando se visualiza una etapa en estado "En proceso". Además, los podemos ver también dentro de la visualización de los datos de un reto (detallado en la HU26). A esta visualización pueden acceder tanto los animadores (si es un superanimador, los verá todos, sino, solamente los suyos) como los participantes de ese reto.
- *Insertar evidencias para ver el avance obtenido hasta el momento.* Para que el usuario sepa el avance realizado durante el proceso, hemos creado la funcionalidad de HU13 dentro del [cuarto sprint](#), que se implantó en la visualización de un reto para aquellos usuarios que fueran participantes del reto dado. En esa pantalla, dentro de una de las subpestañas de la pestaña "Etapas" (en aquella etapa que esté en estado "En proceso") se habilitó un formulario para añadir a la etapa la prueba en el formato imagen, vídeo, audio o texto. En esa misma ventana, también habilitamos la funcionalidad de ver las pruebas añadidas por los participantes (HU14, también del [cuarto sprint](#)), mostrando la prueba junto al nombre del participante.

- *Calificar la realización de una etapa del reto.* Este objetivo se encuentra detallado en la HU15 del [cuarto sprint](#) a través de la selección de uno de los iconos que describa cómo se ha sentido durante el desarrollo de una etapa. Esta selección el usuario la podrá hacer cuando la etapa seleccionada esté en estado “En proceso” durante la visualización de la información de un reto.

Por último, el código desarrollado para llevar a cabo esta red social de vida saludable está accesible para todo el mundo en mi repositorio personal dentro de la plataforma de control de versiones de [GitHub](#).

4.2. Valoración personal

En cuanto a la valoración del trabajo, ha sido un trabajo ameno y muy enriquecedor, en términos de aprendizaje. Me ha hecho ver mi capacidad para poder dirigir y formar parte de un proyecto a largo plazo y conocer de antemano sus entresijos poniéndome en el papel de una gestora de proyectos y habiéndome encargado de todas las tareas. Por lo que creo que este trabajo me ha ayudado a visualizar cómo repartir tareas en el caso de que tuviera varios trabajadores a mi cargo, viendo claramente las funcionalidades que tendría cada trabajador, tanto la parte de aquellos trabajadores encargados de estar con el cliente y obtener las ideas lo más claras posibles, como de la parte de aquellos encargados de diseñar gráfica y funcionalmente la aplicación, y de aquellos encargados de implementar las ideas recabadas y plasmadas en historias de usuario y bocetos. También he aprendido a saber buscar ideas para implementarlas en el proyecto, evaluando qué herramientas debía utilizar y cómo podía usarlas en la aplicación a través de distintos documentos y foros.

Además, me ha hecho estar más atenta a cumplir con las distintas normativas de accesibilidad e inculcarme estas guías para su uso en futuros proyectos, e incluso, pensar qué acciones o diseños podríamos implementar en la aplicación para que sea aún más fácil de utilizar y más inclusiva, para tenerlo en cuenta justo al principio de su planificación.

Acerca de la valoración de los resultados obtenidos después de su desarrollo, puedo decir que he logrado crear una aplicación en la que me siento satisfecha conmigo misma y en la que veo que he cumplido con los requisitos principales, planteados al principio de este documento. También, cabe recalcar que se ha desarrollado una aplicación fácil e intuitiva de utilizar y que se ha diseñado para que sea usada por una amplia lista de tipos de usuarios finales.

Con respecto a la relación con el cliente (en este caso de Inmaculada Garrido, la directora de la Fundación Purísima de la Concepción de Granada), ha sido un gran cliente

porque desde el primer momento sabía que era lo que quería hacer, ha ayudado mucho a lo largo del diseño de la aplicación aportando ideas y detectando ciertos problemas de accesibilidad que, personas como yo, no estamos acostumbradas a pensar o a actuar como una persona con discapacidad. Además, es una gran persona puesto que ha sido uno de mis "animadores" dentro de mi gran reto, lograr crear esta aplicación.

Para acabar con las conclusiones personales, gracias a este proyecto me he dado cuenta de la importancia que tiene centrarnos en lograr nuestros objetivos que nos planteamos a lo largo de nuestra vida, ya sean retos pequeños (como estudiar para un futuro examen) o grandes (como aprender a gestionar nuestras emociones en momentos difíciles), puesto que un simple reto puede cambiar, a gran escala, ciertos aspectos de nuestra vida (por ejemplo, nuestra salud o nuestra vida profesional), hacernos sentir mejor y valorarnos más a nosotros mismos recordando, en nuestros momentos más derrotistas, por qué decidimos llevarlo a cabo, el avance obtenido desde el minuto 0 y el cariño de nuestros seres más queridos.

Además, gracias a que uno de los objetivos de este proyecto es que fuera apto para todas las personas, me ha hecho darme cuenta que aún queda un largo camino para que todas las tecnologías sean accesibles, puesto que aún la gran mayoría no son aptas para todas las personas, y de la importancia que tiene empezar a plantear, desde el principio, cómo desarrollar cualquier invento (ya sea una aplicación, como este proyecto, o un aparato físico, como un ordenador) que pueda utilizar cualquier persona independiente de si tiene alguna clase de discapacidad o no.

4.3. Trabajos futuros

Ya comentados los objetivos y los requisitos de nuestro proyecto, planteados al inicio de este, y cómo hicimos para que estuvieran implementados en el producto final, es momento de detallar qué otras futuras funcionalidades podríamos añadir a nuestro proyecto. Las mejoras que tenemos en mente para el futuro son las que se van a explicar a continuación:

1. Queremos que la aplicación sea más accesible aún y no sea necesario crear una guía de uso en formato documento. Por ello, hemos pensado que para que sea más llamativa y así lograr que el usuario recuerde mejor los pasos de cada una de las funcionalidades lo mejor es crear una *guía interactiva* directamente sobre la aplicación para que así vaya aprendiendo a manejar la aplicación mientras la va utilizando.

2. Poder *compartir por otras redes sociales*, como Instagram, Facebook y Twitter, los avances de uno de mis retos para llamar la atención a otras personas a utilizar nuestra aplicación y así conseguir también el apoyo de mis amigos de esas redes sociales.
3. Dentro de la aplicación, únicamente podemos comunicarnos con nuestros amigos a través de los retos que tenemos en común, mediante mensajes de ánimo o evidencias, sobre los retos, pero no hay otra forma que las personas puedan hablar sobre sus asuntos personales (como acordar cómo debería ser su siguiente reto colectivo o preguntar antes a su amigo si desea hacer un reto con él). Por tanto, hemos pensado que otra alternativa para poder comunicarse entre ellos dentro de la app sea a través de la creación de un apartado donde se muestran los distintos *chats con amigos* (tanto grupales como por parejas).
4. Hemos visto que, cuando queremos insertar una evidencia, un objetivo o una recompensa en un formato distinto al de texto, directamente se elige el elemento dentro de la memoria del dispositivo. Otra manera de insertar dichos elementos podría ser a través de *generar* las fotos, vídeos o audios en ese *instante*, como lo hacemos a través de Whatsapp e Instagram.
5. Hasta el momento, únicamente sabemos si tenemos notificaciones pendientes de leer si entramos en la aplicación y abrimos el menú de navegación dentro de ella. Por lo tanto, podemos generar unas *alertas que nos avisen de las notificaciones nuevas*, incluso cuando la aplicación no está en primer plano.
6. Creada la visualización de un reto en particular y de la lista de retos, nos dimos cuenta que con el título del reto, aunque este se pudiera pasar a un audio, podría llegar a ser muy complejo entender su significado y algunas personas con dificultades de lectura no llegar a comprender exactamente de qué trata ese reto. Por ello, hemos pensado que, además del texto, podríamos indicar el título del reto a través de una imagen que se añadiría para que resuma lo que se quiere realizar en él.
7. Durante el proceso de desarrollo, nos dimos cuenta de que podríamos *dejar guardados los datos de sesión de una cuenta* para facilitar el acceso de personas con discapacidad . Esto no se ha implementado y pensamos que la mejor forma de realizarlo sería la siguiente: la primera vez que inicie sesión a la aplicación lo haría de la forma tradicional (si es necesario con ayuda de otra persona) y, posteriormente, se guardarían los datos esenciales para mantener una sesión dentro de un

dispositivo. A continuación, si se desea iniciar sesión con ella después de haber cerrado sesión, bastaría con pulsar, por ejemplo, sobre la foto de perfil de ese usuario, sin necesidad de introducir ni el correo electrónico, ni la contraseña, ni la clave confirmando el acceso de nuevo.

8. Para facilitar aún más al usuario a la hora de registrarse en la aplicación y ahorrarle introducir información para poder crear una cuenta, pensamos en aplicar la posibilidad de que *se registre e inicie sesión* en la red social a través de otras redes sociales, como Gmail, Facebook o Instagram.
9. De la forma en la que permitimos que los participantes de un reto añadan sus pruebas de cada una de las etapas, no hay un supervisor que mire si realmente se está cumpliendo con el objetivo de las etapas. Por lo tanto, queremos implementar, de alguna manera, el *feedback de un supervisor* que corrija o anime a los participantes que realmente no están cumpliendo con los objetivos, y dando la enhorabuena a aquellos que sí lo han logrado.
10. La aplicación puede necesitar ciertas actualizaciones, como nuevas categorías o cambiar la cantidad de etapas máximas a generar, o incluso podría requerir la creación de nuevas características. Estas modificaciones se podrían realizar a través de un *apartado de uso exclusivo para los administradores del sistema*.
11. La aplicación está implementada actualmente para el idioma en castellano. Para que podamos aumentar el rango de personas que quieran utilizar nuestra app es necesario *añadir más idiomas a ella*, además de configurar las distintas herramientas de accesibilidad (como el dictado y la transcripción) para que detecten el idioma y trabajen correctamente con él.

ANEXO 1. Diagramas finales del proyecto

Diagrama de Entidad/Relación

Con respecto a cómo estaría formada la base de datos de nuestra aplicación, veríamos como resultado final el diagrama Entidad/Relación de la *Figura 140*:

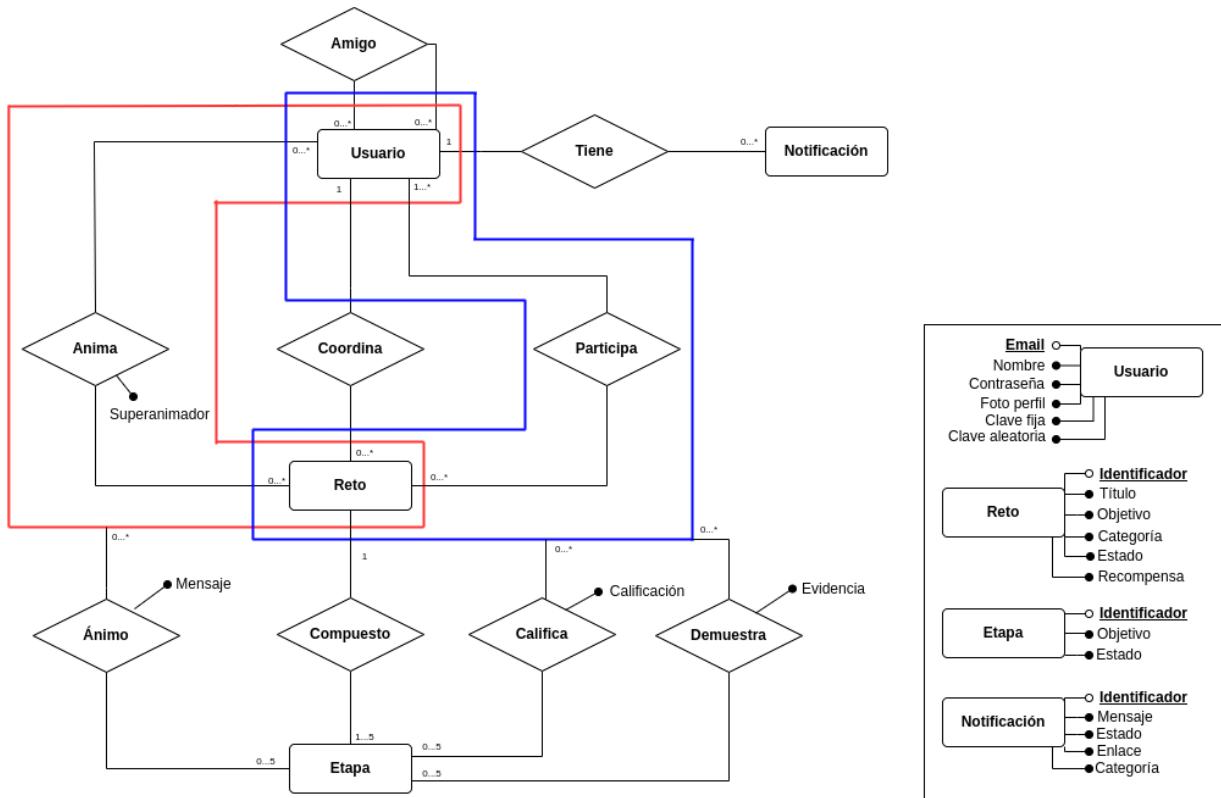


Figura 140: Diagrama de Entidad/Relación final.

En ella podemos ver que nos encontramos con las siguientes entidades principales de nuestra red social dentro del recuadro que se encuentra en la esquina inferior derecha:

- *Usuario*. Almacena los datos esenciales para identificar a una persona dentro de la aplicación.
- *Reto*. Guarda los datos generales que describen un reto.
- *Etapa*. Est醤 la informaci髇 que describe lo que se va a realizar en una etapa.
- *Notificación*. Se halla la informaci髇 que contiene cada una de las notificaciones.

Adem醘s de estas entidades, nos encontramos con unas agragaciones (agrupaciones entre entidades relacionadas):

- *Anima*. Relación con la entidades *Usuario-Reto* que nos da información de un usuario que forma parte del reto como animador. En ella nos encontramos con un atributo (*Superanimador*) que nos indica si esa persona forma como un animador normal y corriente o como un superanimador.
- *Participa*. Relación con las entidades *Usuario-Reto* nos indica el usuario que forma parte del reto como uno de sus participantes.

Por último, las relaciones que hallamos dentro de este diagrama son las que vamos a explicar a continuación:

- *Amigo*. Relación con la entidad *Usuario* para indicar que dos usuarios son amigos.
- *Tiene*. Relación con las entidades *Usuario-Notificación* para almacenar las notificaciones que tiene un usuario.
- *Coordina*. Relación con las entidades *Usuario-Reto* para indicar qué usuario es el coordinador de un reto.
- *Compuesto*. Relación con las entidades *Reto-Etapa* para saber las etapas que forman parte de un reto.
- *Ánimo*. Relación de la entidad *Etapa* con la agregación *Anima* en la que se guardan los mensajes de ánimo (almacenado en el atributo *Mensaje*) que envía un animador en una etapa.
- *Califica*. Relación de la entidad *Etapa* con la agregación *Participa* en la que se guarda la calificación (en el atributo *Calificación*) de un participante en una etapa.
- *Demuestra*. Relación de la entidad *Etapa* con la agregación *Participa* en la que se guarda, a través del atributo *Entidad*, la evidencia de un participante en una etapa.

Diagrama de clases

En relación a las clases generadas para el manejo de las diversas funcionalidades que ofrece la aplicación, podemos identificarlas en el siguiente diagrama de clases.

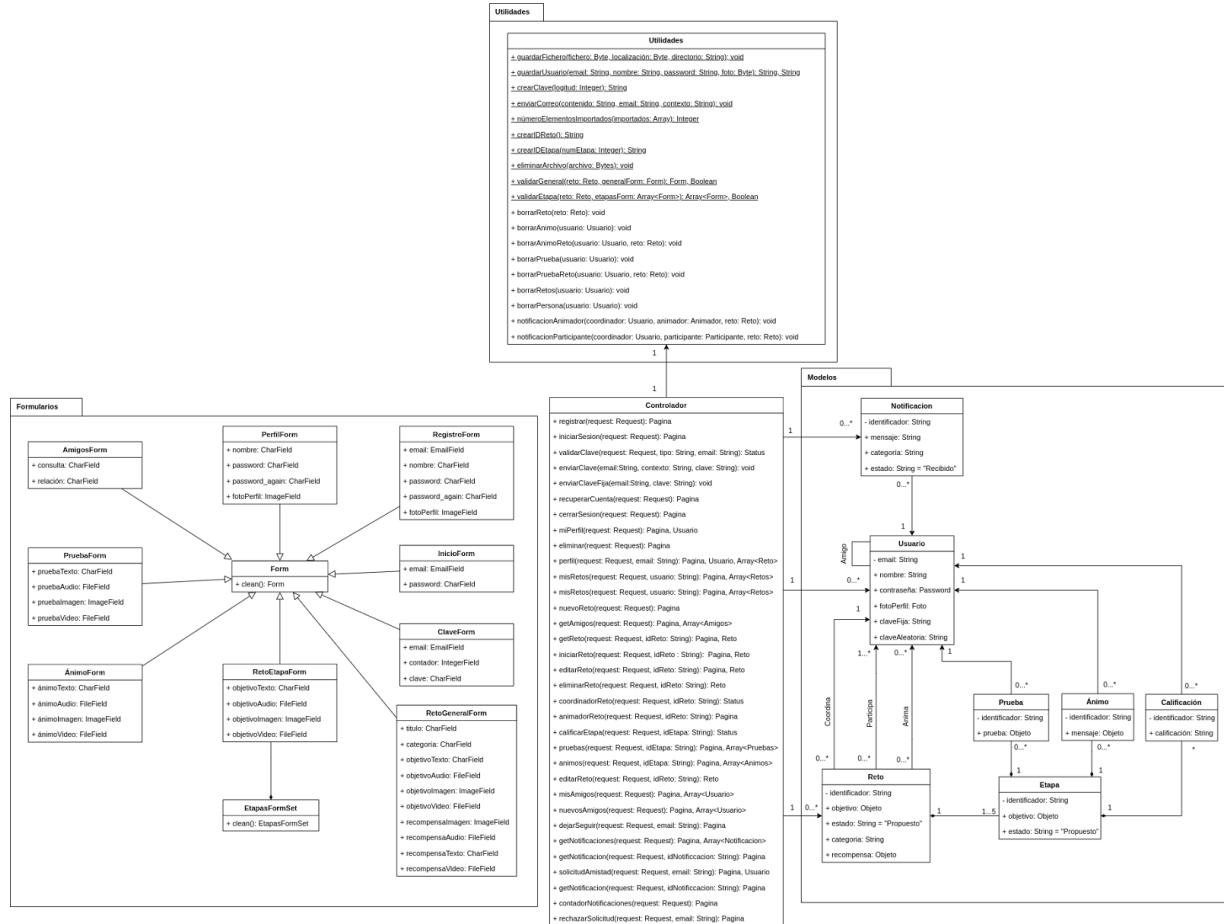


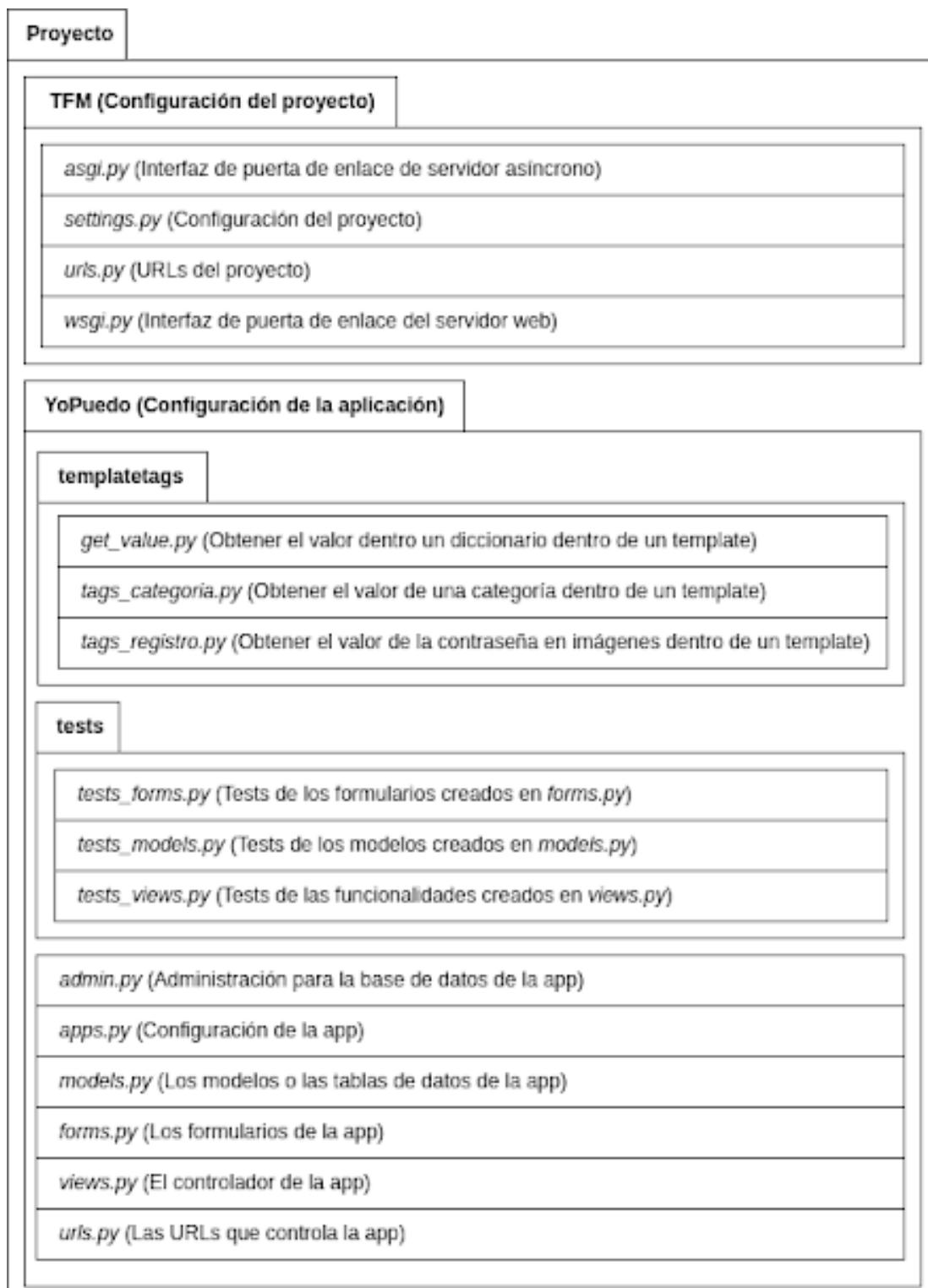
Figura 141: Diagrama de clases final del proyecto.

En el diagrama de clases podemos observar que están divididos en 3 paquetes distintos: el primero (el que se encuentra en el centro superior de la imagen) es el paquete que únicamente contiene la clase que se encarga de realizar algunas de las tareas más comunes de la aplicación (como guardar o borrar un archivo, generar claves aleatorias o crear identificadores); el segundo (el que se encuentra a la izquierda de la figura) es el paquete encargado de crear los campos de información de un formulario (junto a sus características y sus restricciones) y comprobar que los datos introducidos sobre cada uno de esos campos son los correctos que hereda de la clase *Forms* (creada ya por *Django*); y el tercero (el que nos encontramos a la derecha del diagrama) es el paquete encargado de manejar los datos de los modelos de la base de datos de la aplicación (*Usuario*, *Reto*, *Etapa*, *Notificación*, *Pruebas*, *Ánimo* y *Calificación*).

El caso de la clase que nos queda por mencionar (la que se encuentra justo en medio del diagrama) es el controlador de nuestra red social de retos, es decir, es donde están cada una de las funcionalidades detalladas en las historias de usuario del [producto backlog](#).

Estructura interna de ficheros del proyecto

En cuanto a la estructura de ficheros que genera este proyecto para poder manejar la aplicación de red social de vida saludable, es el que vemos en la siguiente figura, donde también se explica qué es lo que contiene el fichero o la carpeta correspondiente:



conf (Archivos de configuración)

.secret_key (Clave secreta de el proyecto)

db.mysql (Configuración para la conexión con la base de datos)

.email (Configuración para la conexión con la cuenta de correo electrónico de la app)

static (Archivos estáticos)**YoPuedo (Archivos estáticos de la app)****css (Archivos de Hojas de Estilo de la app)**

Todas las hojas de estilo de cada página que genera la aplicación

js (Scripts de JavaScript de la app)

AudioTranscript.js (Encargado de convertir audio a texto)

Password.js (Encargado de mostrar u ocultar las contraseñas a través de imágenes)

SpeechRecognition.js (Encargado de dictar un texto)

SpeechSynthesis.js (Encargado de transformar un texto a audio)

VisualizacionRetos.js (Encargado de cambiar la visualización por pantalla de un reto)

media (Archivos multimedia)**YoPuedo (Archivos multimedia de la app)****foto_perfil (Fotos de perfil de los usuarios)****password (Imágenes utilizadas como contraseñas)****<reto> (Imágenes relacionadas con un reto)****<etapa> (Imágenes relacionadas con una etapa)****Pruebas (Imágenes relacionadas con las pruebas de un reto)**

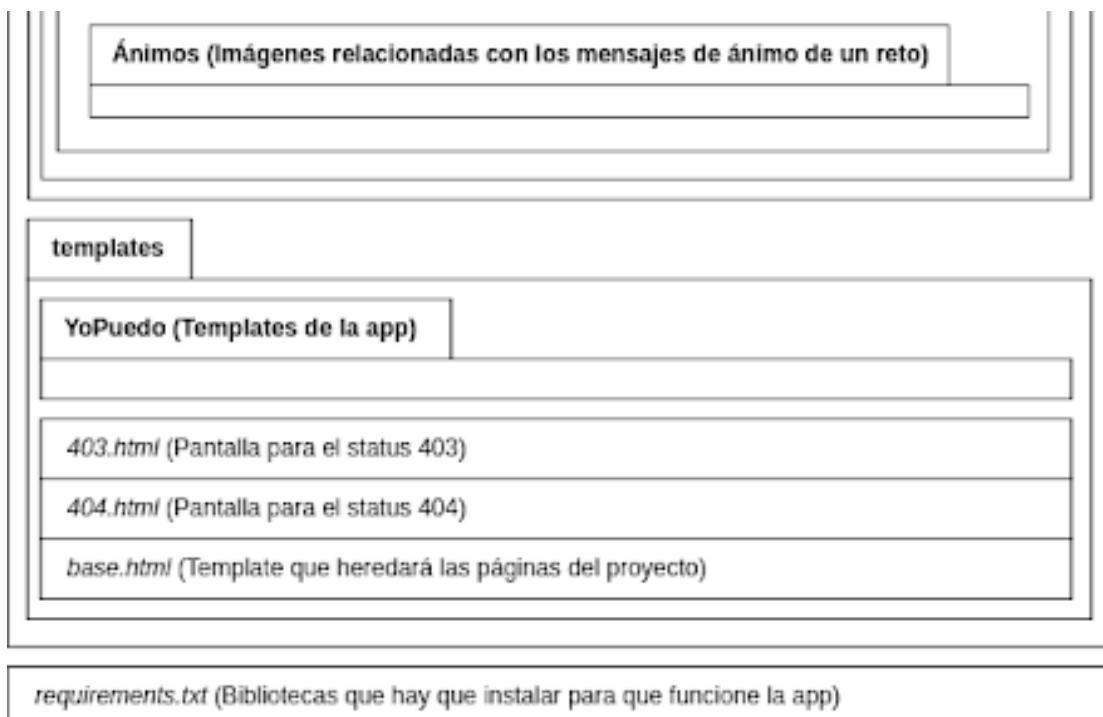


Figura 142: Estructura final de ficheros del proyecto.

ANEXO 2. Diseño de los bocetos según la guía de accesibilidad

En este anexo nos centraremos en comentar qué reglas dentro de la guía de accesibilidad, comentada al principio de este documento en el apartado de [Accesibilidad](#), se cumplen en los bocetos que hemos desarrollado en cada sprint de este proyecto.

Antes de comenzar a ver qué normas se han implementado en los diseños gráficos de la aplicación, debemos indicar qué guía de accesibilidad, de las dos encontradas ([según Ministerio de Asuntos Económicos y Transformación Digital \[1\]](#) y [según WCAG \[2, 3, 4\]](#)), es la que seleccionamos para su creación.

De las dos explicadas, preferimos utilizar la guía de accesibilidad de la organización WCAG al ser una organización internacional centrada únicamente en buscar nuevas reglas y corregir las ya creadas que mejoren, con el paso del tiempo y junto a la creación de nuevas tecnologías, la accesibilidad de una aplicación. Además, la primera guía, la del Ministerio, está basada en ella.

Por lo tanto, vamos a elegir algunos de los bocetos de la aplicación e iremos desglosando, para cada uno de los apartados que tiene esta guía, qué normas se pueden ver reflejadas en ellos.

En primer lugar, empezaremos a comentar de los bocetos de la *Figura 143* las características que tiene para que nosotros podamos decir que es [Perceptible](#):

- Hemos creado el diseño para distintos tamaños de pantalla, la más grande de un ordenador, una mediana de tablet y la pequeña de un móvil, organizando sus componentes y sus tamaños para que sea sencillo de ver y de utilizar, como podemos observar en los bocetos de visualización de retos de la *Figura 143*.

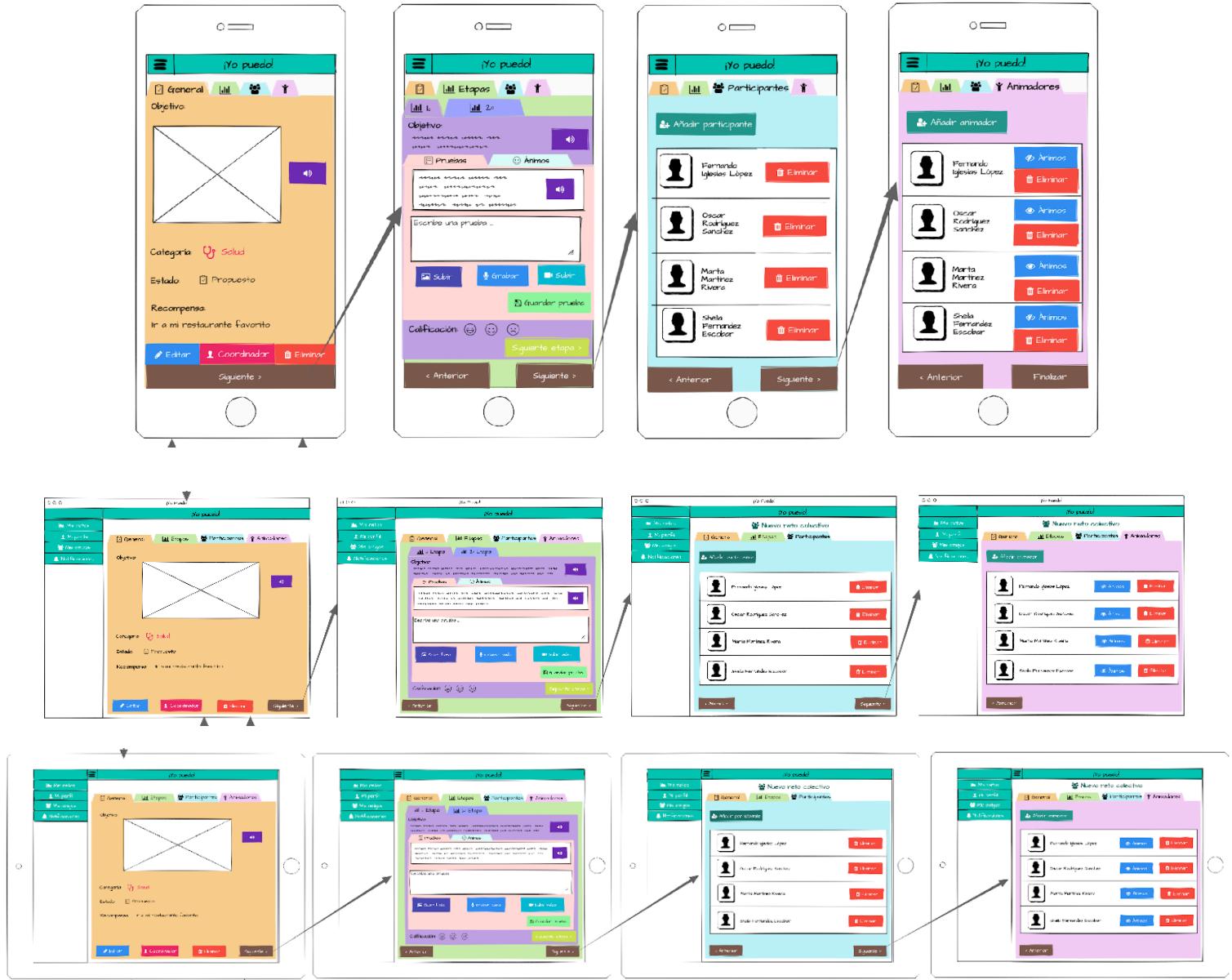


Figura 143: Visualización de un reto por parte de un coordinador.

- Se puede aumentar el tamaño de estas pantallas utilizando el zoom que tiene integrado el navegador en el que se ejecuta la aplicación.
- Podemos ver que ofrecemos el contraste necesario para que sea lo más visible posible cada uno de los textos, además del espaciado y tamaño necesario.
- Utilizamos los distintos colores para guiar al usuario lo que está en ese momento visualizando y diferenciar los distintos botones y apartados que tenga esa pantalla.
- No ofrecemos ningún audio de fondo, solamente los audios que una persona ponga en un reto con los botones necesarios para controlar dicho audio.

- Las imágenes únicamente se utilizan para poder describir una acción del reto o identificar a un usuario, que será el propietario del reto o el propietario de una cuenta el que decida qué imagen visualizar.
- Al estar organizadas las distintas páginas a través de pestañas o a través de paginación (como el de la *Figura 144*), no necesita el usuario hacer scroll vertical ni horizontalmente para ver toda la información de la pantalla de un solo vistazo.

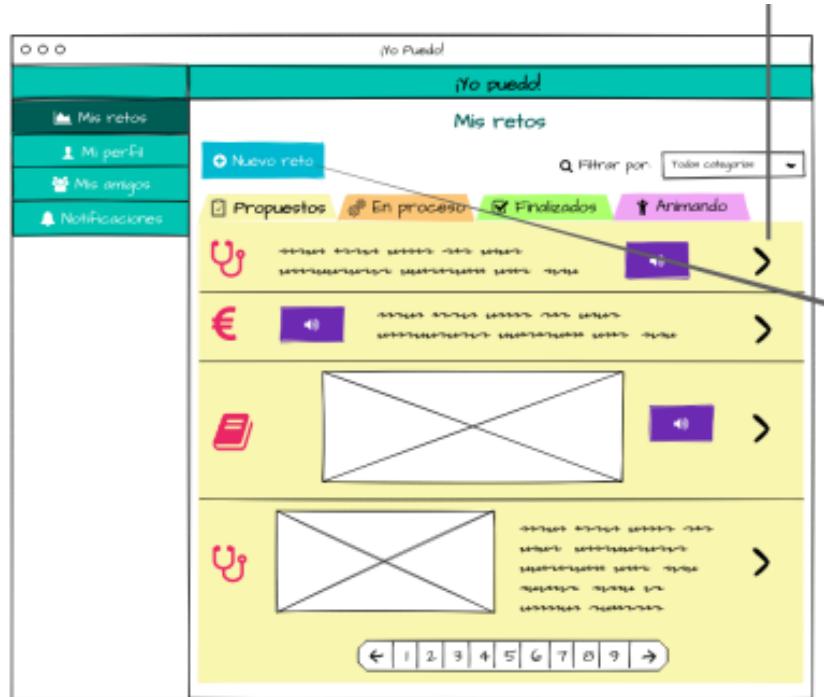


Figura 144: Boceto del listado de retos en formato ordenador.

- Para identificar distintos tipos de información (texto, imagen, audio o vídeo), o qué acción realiza un botón , se añaden iconos que ayuden a describirlos. Por ejemplo, en la *Figura 145* podemos ver los siguientes:

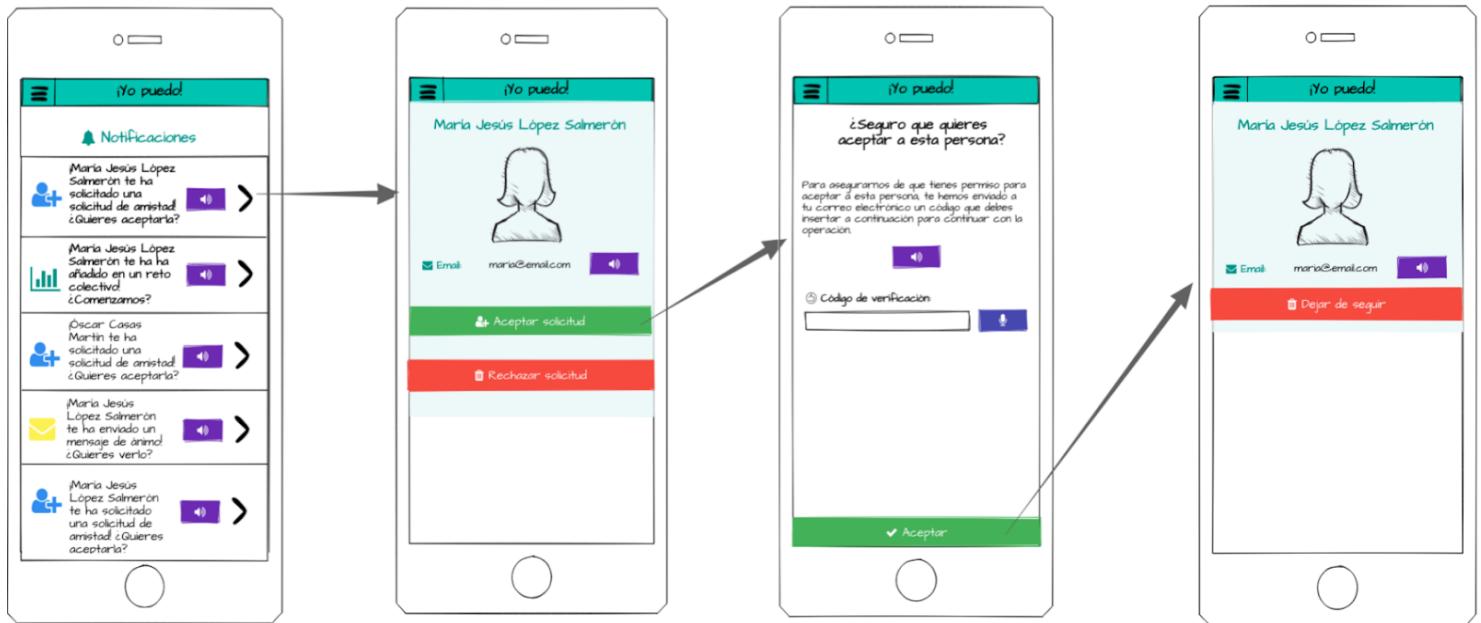


Figura 145: Bocetos del listado de notificaciones junto a la visualización de una solicitud de amistad.

- Las categorías de una notificación se muestran a través de símbolos que se colocan a la izquierda de ésta, y varían dependiendo de si es una solicitud, mensaje de ánimo o incorporación a un reto.
- La redirección a ver en detalle una notificación se realiza a través de un botón con “>” como símbolo.
- La identificación de que el dato mostrado en el perfil del usuario es el email a través de un ícono de un sobre junto al título de ese dato.
- La transcripción de un texto a través del botón, se representa con el símbolo del micrófono, y el de dictado a través de otro ícono con un altavoz, como el mostrado en la Figura 146.

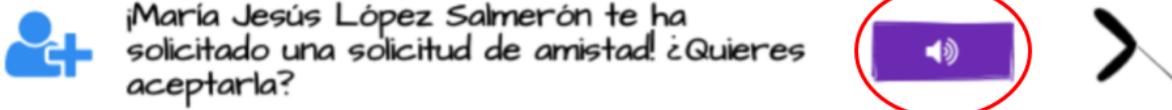


Figura 146: Ejemplo del botón de transcripción de texto.

- Los botones de dejar de seguir a un amigo y el de rechazar solicitud con el ícono de una papelera (Figura 147), y el de aceptar solicitud con el símbolo de una persona junto a un +, además de su acción en texto.



Figura 147: Botones Dejar de seguir y rechazar solicitud.

- Para aquellos elementos que sean audios, se transcribe dicho audio a texto (como diseñamos en la Figura 148). Y para aquellos textuales, se dictan al usuario (como la Figura 146).

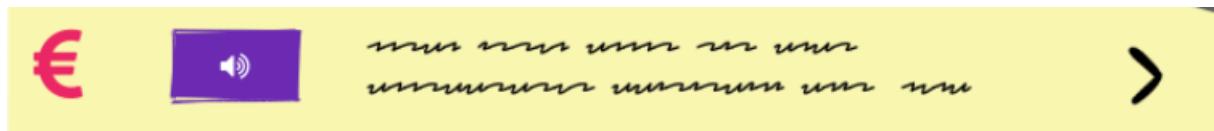


Figura 148: Ejemplo de transcripción de audio a texto.

Ya comentadas las normas reflejadas en esos bocetos con respecto a la característica *Perceptible*, es momento de centrarnos en aquellas reglas dentro de Operable:

- Todas las funcionalidades se pueden realizar a través de los distintos botones y menús que ofrece la aplicación en la interfaz.
- Se puede navegar a la perfección a cualquier componente desde el teclado, con las teclas que tiene por defecto el navegador.
- No es necesario utilizar combinaciones de teclas extras para que el usuario tenga que navegar o realizar otra acción sobre la app.
- El usuario no pierde los datos o la página donde se encuentra por inactividad, puesto que no se ha creado un temporizador que cierre la sesión de un usuario por inactividad. Únicamente los perdería cuando él decidiera cerrar el navegador o la sesión.
- Se mostrarán alertas al usuario de algún cambio realizado sobre su cuenta que se quitarán de la pantalla cuando el usuario pulse sobre un botón dentro de la alerta.

- Cuando se muestra la transcripción de un audio, permanece en pantalla hasta que el usuario decida ir a otra o recargue dicha página. Lo podemos ver en detalle en la *Figura 144*, que muestra el listado de los retos y, si prestamos atención al segundo reto (el que tiene el símbolo del euro) tiene un audio junto a su transcripción.
- Mostramos distintos mecanismos para navegar sobre una página. Si vemos en la *Figura 143*, tenemos las pestañas que hay dentro de un reto para entrar en las etapas del reto, además de los botones que redirigen hacia delante o hacia atrás en la parte inferior de la pantalla.
- Cada pestaña y botón de navegación presentan el título de la página que visualizarán si pulsan sobre él, o la acción que va a realizar, junto a un icono que lo describa.
- Ningún elemento de la aplicación parpadea ni produce una animación cuando se pulsa o se pasa sobre ella.
- Aquellas acciones como buscar y añadir animadores o participantes, se pueden cancelar pulsando sobre la X que se encuentra en la esquina superior de esa ventana (como podemos observar en la ventana del boceto de la *Figura 149*).

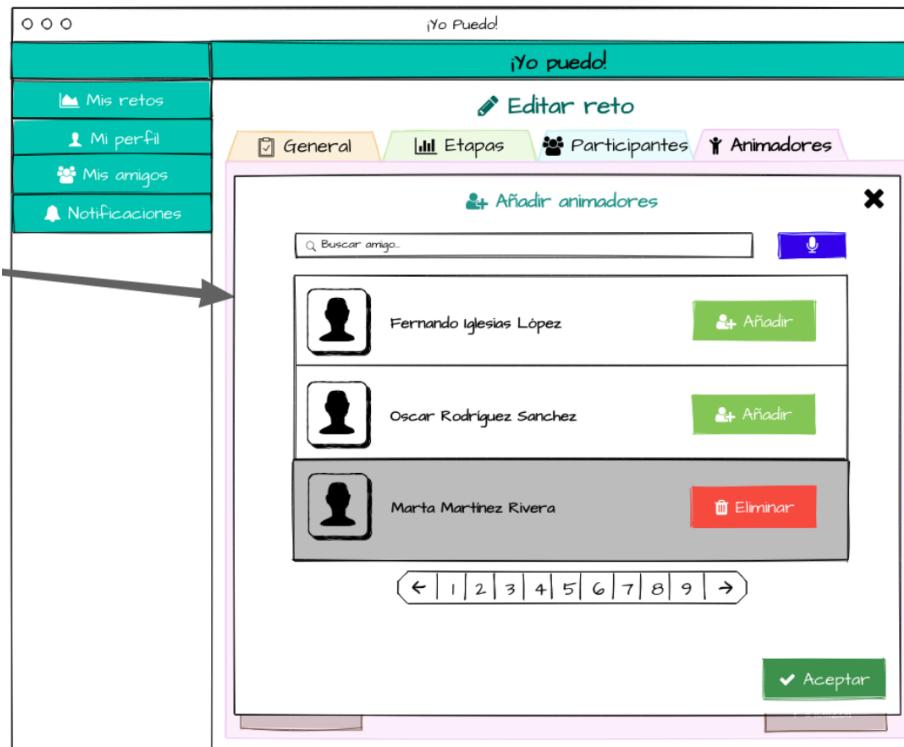


Figura 149: Boceto de selección de amigos como animadores.

- Los botones principales del reto, como los de su navegación, están en las esquinas de la pantalla mostrada para que sea fácil para el usuario pulsar sobre ellas. El resto de los botones, como la inserción de audios, imágenes o vídeos y la eliminación de animadores o participantes, son lo suficientemente grandes y puestos en los laterales de la pantalla para que sea sencillo utilizarlos y no se pulse sin querer.

En cuanto a las normas que dictan la asociación WCAG para que una aplicación sea [Comprendible](#), aplicamos de la siguiente manera:

- Cuando se dicta un texto de la aplicación, como podemos ver al pulsar el botón del altavoz en los bocetos anteriores, pronuncia cada una de las palabras con el acento predeterminado de la aplicación, es decir, en castellano.
- Los distintos componentes de la aplicación, como los botones o los títulos de las pestañas o de los campos de un formulario, tienen la descripción de su acción visible en la pantalla, por lo que no es necesaria una explicación sobre su uso.
- Si en el formulario se encuentran errores, se vuelve a mostrar el mismo formulario con los campos rellenados por el usuario, para que no tenga que volver a realizarlo y pueda enviarlo de nuevo con los cambios hechos. Para que sepa el usuario qué es lo que tiene que corregir se visualiza un mensaje debajo del campo correspondiente (con el título del campo correspondiente y del mensaje en rojo) sobre qué es lo que debería escribir o insertar.
- En algunos casos, la entrada de un campo no es únicamente un texto, sino que el usuario puede dictar a la aplicación lo que quiere insertar en dicho campo o introducir un audio, vídeo o imagen que describa lo pedido en ese campo (véase en la segunda pantalla de los bocetos de visualización de retos de la *Figura 143* cuando el coordinador quiera insertar una prueba en una etapa).
- Aunque se cambie la orientación de la pantalla, el diseño se adapta perfectamente para el ancho y el alto de la pantalla en esa orientación, sin perder su funcionalidad ni la forma en la que se realiza, como podemos justificar a través de la implementación de la visualización de un reto en la *Figura 150*.

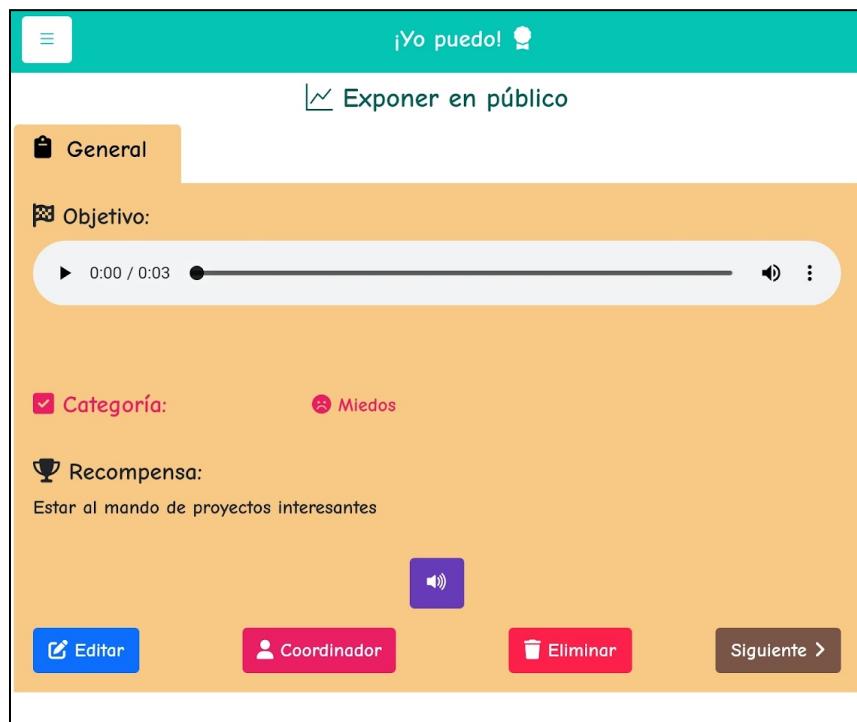


Figura 150: Visualización en horizontal y en vertical del detalle de un reto en una tablet.

Por último, la característica [Robusta](#), definida por WCAG, que es aplicable a nuestro problema, es que en la entrada de datos se puede insertar de varias formas: escribiendo, dictando, insertando imágenes, audios o videos (como podemos ver en los bocetos de la Figura 143 cuando se quiere insertar una prueba) y a través de la selección de imágenes que se convierte en texto (como la contraseña para iniciar sesión o registrarse de la Figura 151).

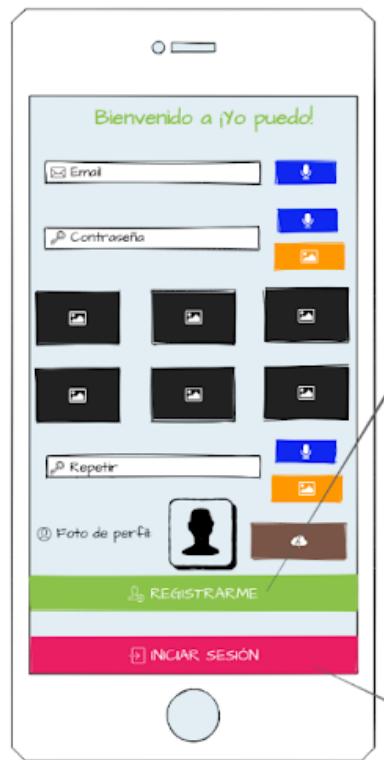


Figura 151: Boceto de inserción de datos para registrarse en el sistema en dispositivos móviles.

Bibliografía

1. Write, G. C. (2018, 28 agosto). *¿Qué es la Accesibilidad para Aplicaciones Móviles?* | abalit.org. Blog Abalit Technologies, Barcelona.
<https://www.abalit.org/blog/post/accesibilidad-para-apps/es>
2. Baena, M. R. (2021, 27 agosto). *¿Cómo crear una app accesible? ¿Cuál es su importancia?* - App&Web. App&Web.
<https://www.appandweb.es/blog/como-crear-app-accesible-importancia/#:~:text=Una%20app%20accesible%20es%20aquella,la%20aplicaci%C3%B3n%20sin%20ninguna%20dificultad>
3. *Guía de accesibilidad de aplicaciones móviles (apps)*. (s. f.).
https://www.google.com/url?client=internal-element-cse&cx=015503211893170276735:mgpiyhwz7g4&q=https://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Accesibilidad/pae_documentacion/pae_elnclusion_Accesibilidad_de_apps.html&sa=U&ved=2ahUKEwilwdflz6f_AhVHWaQEHQYTARYQFnoECAQQAQ&usg=AOvVaw2JW9eQQbfOp59FRjliwVSX
4. *Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile*. (2015, 26 febrero).
<https://www.w3.org/TR/mobile-accessibility-mapping/#wcag-2.0-and-mobile-content-applications>
5. *Web Content Accessibility Guidelines (WCAG) 2.1*. (2018, 5 junio).
<https://www.w3.org/TR/WCAG21/>
6. *Web Content Accessibility Guidelines (WCAG) 2.2*. (2023, 17 mayo).
<https://www.w3.org/TR/WCAG22/>
7. Initiative, W. W. A. (s. f.). *Accesibilidad Móvil en el W3C*. Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/standards-guidelines/mobile/es>
8. Qapital, Inc. (2023, 26 mayo). *Qapital - Set-and-forget your financial goals*. Qapital.
<https://www.qapital.com/>

9. *2020 Reading Challenge*. (s. f.). Goodreads.
<https://www.goodreads.com/challenges/11621-2020-reading-challenge>
10. *Challenges | Engage Employees Through Total Wellness and Skill Development* .
(s. f.). <https://challenges.app/>
11. *21 Days Challenge - Apps on Google Play*. (s. f.).
<https://play.google.com/store/apps/details?id=com.limatech.dayschallenge.dayschallenge>
12. ✓ *Habitus: Daily Habit Challen* - Apps on Google Play. (s. f.).
<https://play.google.com/store/apps/details?id=app.habitus.challenges>
13. *Django*. (s. f.). Django Project. <https://www.djangoproject.com/>
14. *Flask vs. Django: una comparativa de los frameworks de Python*. (2022, 11 febrero). IONOS Digital Guide.
<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/flask-vs-django/>
15. Zepeda, E. (2023, 1 junio). *¿Por qué deberías usar Django Framework?* Coffee bytes. <https://coffeebytes.dev/por-que-deberias-usar-django-framework/>
16. Patel, J., & Patel, J. (2023). *Django vs Express: 2 Best Web Frameworks Compared in 2023*. Monocubed. <https://www.monocubed.com/blog/django-vs-express/>
17. *Welcome to Python.org*. (2023, 31 mayo). Python.org. <https://www.python.org/>
18. Rubabe, S. (2021, 16 diciembre). *ORM for Python* - Pragmatech - Medium. *Medium*.
<https://medium.com/pragmatech/orm-for-python-b63cfbc39e7f>
19. *Welcome to Flask — Flask Documentation (2.3.x)*. (s. f.).
<https://flask.palletsprojects.com/en/2.3.x/>
20. Muñoz, J. D. (2023, 14 abril). *Qué es Flask*. OpenWebinars.net.
<https://openwebinars.net/blog/que-es-flask/>
21. Alonso, N. (2021, 12 diciembre). *¿Qué es un WSGI?* - Nacho Alonso - Medium.
Medium.
[https://medium.com/@nachoad/que-es-wsgi-be7359c6e001#:~:text=WSGI%20son%20las%20siglas%20de,%2Fpetici%C3%B3n%20\(o%20request\).](https://medium.com/@nachoad/que-es-wsgi-be7359c6e001#:~:text=WSGI%20son%20las%20siglas%20de,%2Fpetici%C3%B3n%20(o%20request).)

22. ¿Qué son las estructuras de control de Ninja? (2023, 17 marzo). KeepCoding.

<https://keepcoding.io/blog/que-son-las-estructuras-de-control-de-jinja/>

23. Express - Infraestructura de aplicaciones web Node.js. (s. f.).

<https://expressjs.com/es/>

24. Express Web Framework (Node.js/JavaScript) - Aprende desarrollo web | MDN.

(2022, 30 noviembre).

https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs

25. Express.js. (s. f.).

<https://www.tutorialsteacher.com/nodejs/expressjs#:~:text=Easy%20to%20configure%20and%20customize,Jade%2C%20Vash%2C%20EJS%20etc.>

26. Sharma, A. (2023). What Is Express JS In Node JS? Simplilearn.com.

https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js#what_is_express_js

27. JavaScript | MDN. (2023, 9 febrero).

<https://developer.mozilla.org/es/docs/Web/JavaScript>

28. Node.js. (s. f.). Node.js. <https://nodejs.org/>

29. PostgreSQL. (2023, 9 junio). PostgreSQL. <https://www.postgresql.org/>

30. Segovia, J. (2021). Ventajas y Desventajas de PostgreSQL. TodoPostgreSQL.

<https://www.todopostgresql.com/ventajas-y-desventajas-de-postgresql/>

31. HostingPlus. (2021, 31 diciembre). PostgreSQL: ventajas y desventajas | Blog |

Hosting Plus España. Hosting Plus.

<https://www.hostingplus.com.es/blog/postgresql-ventajas-y-desventajas/>

32. MySQL. (s. f.-b). <https://www.mysql.com/>

33. ▷ Ventajas y Desventajas de MySQL. (s. f.).

<https://codigosql.top/ventajas-y-desventajas-de-mysql/>

34. Aldeahost, S., & Aldeahost, S. (2021). Qué es MySQL Ventajas y desventajas |

Hosting OFERTA: Web Hosting México ▷ Aldeahost. Hosting OFERTA: Web Hosting

México ▷ Aldeahost. <https://aldeahost.com.mx/que-es-mysql-ventajas-y-desventajas/>

35. Tecnoadmin. (2019, 12 noviembre). ¿Qué es MySQL? Características, Ventajas, Desventajas e Instalación. *TecnoMagazine*. <https://tecnomagazine.net/mysql/>
36. Infante, D. C. H., & Infante, D. C. H. (2023). MariaDB vs MySQL: Diferencias clave, pros y contras, y más. *Tutoriales Hostinger*.
[https://www.hostinger.es/tutoriales/mariadb-vs-mysql#Pros_y_contrас_de_MariaDB](https://www.hostinger.es/tutoriales/mariadb-vs-mysql#Pros_y_contrاس_de_MariaDB)
37. García, F. (2023). MariaDB Ventajas y Desventajas. *Código SQL*.
<https://codigosql.top/mariadb/ventajas-desventaja/>
38. MariaDB.org. (2019, 13 noviembre). *MariaDB Foundation - MariaDB.org*.
<https://mariadb.org/>
39. Shim, T. (2023). Best Django Hosting: Where to Run Your Next Django Project? *Web Hosting Secret Revealed (WHSR)*.
<https://www.webhostingsecretrevealed.net/blog/web-hosting-guides/best-django-hosting-providers/>
40. *Host, run, and code Python in the cloud: PythonAnywhere*.
<https://www.pythonanywhere.com/>
41. AWS | Cloud Computing - Servicios de informática en la nube. (s. f.). Amazon Web Services, Inc. <https://aws.amazon.com/es/>
42. A2 Hosting. (2022, 4 marzo). *The Best Web Hosting Services at 20x Speeds | A2 Hosting*. <https://www.a2hosting.com/>
43. DigitalOcean | The Cloud for Builders. (s. f.). <https://www.digitalocean.com/>
44. ¿Qué es AWS? (s. f.). Amazon Web Services, Inc.
<https://aws.amazon.com/es/what-is-aws/#:~:text=AWS%20cuenta%20con%20una%20cantidad,autom%C3%A1tico%20e%20inteligencia%20artificial%2C%20lagos>
45. Google Cloud. (s. f.). *Servicios de cloud computing | Google Cloud*.
<https://cloud.google.com/?hl=es>
46. Trafaniuc, V. (2022, 28 junio). *Descubre qué es Google Cloud Platform y sus ventajas*. Maplink Blog.

<https://maplink.global/blog/es/que-es-google-cloud/#%C2%BFQu%C3%A9-es-Google-Cloud?>

47. *Servicios de informática en la nube | Microsoft Azure.* (s. f.).

<https://azure.microsoft.com/es-es>

48. *Servicio de aplicaciones web | Microsoft Azure.* (s. f.).

<https://azure.microsoft.com/es-es/products/app-service/web>

49. *Build fast, responsive sites with Bootstrap.* (s. f.). *Bootstrap.* <https://getbootstrap.com/>

50. *Skeleton: Responsive CSS Boilerplate.* (s. f.). <http://getskeleton.com/>

51. *Pure.* (s. f.). <https://purecss.io/>

52. *htmx - high power tools for html.* (s. f.). <https://htmx.org/>

53. *Alpine.js.* (s. f.). <https://alpinejs.dev/>

54. *Find Icons with the Perfect Look & Feel | Font Awesome.* (s. f.). *Find Icons with the Perfect Look & Feel | Font Awesome.* <https://fontawesome.com/icons>

55. *Free Line Icons for Designers and Developers - Lineicons.* (2021, 23 mayo).
Lineicons. <https://lineicons.com/>

56. Flaticon. (2023, 10 junio). *Hill Icon - 10127170.*

<https://www.flaticon.com/packs/spring-652>

57. Pursell, S. (2023, 20 enero). *Metodología Agile: qué es y cómo aplicarla a tu proyecto.* *HubSpot.* <https://blog.hubspot.es/marketing/metodologia-agile>

58. *Las etapas del ciclo de vida de desarrollo de software según la metodología ágil.* (2018, 10 agosto).

<https://www.lucidchart.com/blog/es/ciclo-de-vida-del-desarrollo-de-software-segun-la-metodologia-agil>

59. Apd, R. (2023). *¿Qué es la metodología Agile y cuáles son sus principales ventajas?* *APD España.* <https://www.apd.es/que-es-la-metodologia-agile-principales-ventajas/>

60. Fundación Purísima Concepción de Granada. (2022, 18 febrero). *Atención integral a personas con Diversidad Funcional - Fundación Purísima Concepción.* Fundación Purísima Concepción. <https://fpurisimaconcepcion.org/>

61. Python, R. (2023, 17 febrero). *Desplegar un proyecto de Django en PythonAnywhere*
- *Recursos Python*. Recursos Python.
[https://recursospython.com/guias-y-manuales/desplegar-un-proyecto-de-django-en-p
ythonanywhere/](https://recursospython.com/guias-y-manuales/desplegar-un-proyecto-de-django-en-pythonanywhere/)
62. Manzanero. (s. f.). *curso-python-auto/test-lab/09 - django.ipynb at main* .
manzanero/curso-python-auto. GitHub.
[https://github.com/manzanero/curso-python-auto/blob/main/test-lab/09%20-%20djang
o.ipynb](https://github.com/manzanero/curso-python-auto/blob/main/test-lab/09%20-%20djang
o.ipynb)
63. Moreno, D. M. F. (2018, 2 noviembre). Django — Conecta tu proyecto con la base de
datos MySQL. *Medium*.
[https://medium.com/@a01207543/django-conecta-tu-proyecto-con-la-base-de-datos-mysq
l-2d329c73192a](https://medium.com/@a01207543/django-conecta-tu-proyecto-con-la-base-de-datos-mysq
l-2d329c73192a)
64. *Django*. (s. f.-b). Django Project.
<https://docs.djangoproject.com/en/4.2/topics/auth/default/#user-objects>
65. Herman, M. (2023). Creating a Custom User Model in Django. *TestDriven.io*.
<https://testdriven.io/blog/django-custom-user-model/>
66. *How to add attributes to form field in template*. (s. f.). Stack Overflow.
[https://stackoverflow.com/questions/39399452/how-to-add-attributes-to-form-field-in-t
emplate](https://stackoverflow.com/questions/39399452/how-to-add-attributes-to-form-field-in-t
emplate)
67. *How to Create Custom Password Validators in Django*. (s. f.). Six Feet Up.
<https://sixfeetup.com/blog/custom-passwordValidators-in-django>
68. Kumar, B. (2023, 11 enero). How To Encrypt And Decrypt Password In Django -
Python Guides. *Python Guides*.
<https://pythonguides.com/encrypt-and-decrypt-password-in-django/#:~:text=Decrypt%>
69. GeeksforGeeks. (2021). Upload files in Python. *GeeksforGeeks*.
[https://www.geeksforgeeks.org/upload-files-in-python/20Password%3A%20Django%
20doesn't,the%20hash%20password%20or%20not.](https://www.geeksforgeeks.org/upload-files-in-python/20Password%3A%20Django%
20doesn't,the%20hash%20password%20or%20not.)

70. Delgado, C. (2017, 22 enero). Getting started with the Speech Recognition API in Javascript. *Our Code World*.
<https://ourcodeworld.com/articles/read/362/getting-started-with-the-speech-recognition-api-in-javascript>
71. Uso de la Web Speech API - Referencia de la API Web | MDN. (s. f.).
https://developer.mozilla.org/es/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API
72. ARASAAC. (s. f.). <https://arasaac.org/pictograms/search>
73. Rosencrance, L., Loshin, P., & Cobb, M. (2021). two-factor authentication (2FA).
Security.
<https://www.techtarget.com/searchsecurity/definition/two-factor-authentication>
74. Ramírez, I. (2019). Identificación en dos pasos: todos los métodos disponibles y sus ventajas e inconvenientes. *Xataka*.
<https://www.xataka.com/basics/identificacion-dos-pasos-todos-metodos-disponibles-sus-ventajas-e-inconvenientes>
75. Security, S. (2021, 5 agosto). *What Is Two Step Verification & How Does It Work?*
Savvy Security.
<https://cheapsslsecurity.com/blog/what-is-two-step-verification-how-does-2sv-work/>
76. Electronic Frontier Foundation. (2017, 25 septiembre). A guide to common types of two-factor authentication. *VentureBeat*.
<https://venturebeat.com/2017/09/24/a-guide-to-common-types-of-two-factor-authentication/>
77. Khan, M. W. (2023, 30 enero). Generar cadenas aleatorias en Python. *Delft Stack*.
<https://www.delftstack.com/es/howto/python/random-string-python/>
78. codigofacilito. (2016, 13 julio). 21.- *Curso Django - Recuperar contraseña por correo (facilito)* [Vídeo]. YouTube. https://www.youtube.com/watch?v=B3-AbVnls_8
79. codigofacilito. (2020, 10 marzo). *Envío de correos con Django* [Vídeo]. YouTube.
https://www.youtube.com/watch?v=XPa_duOg2Ko

80. Diaz, D. (2021, 4 agosto). How to Send Email with Django — SitePoint. *SitePoint*.
<https://www.sitepoint.com/django-send-email/>
81. Modal forms with Django+HTMX. (2022, 18 febrero). *Good Code Smell*.
<https://blog.benoitblanchon.fr/django-htmx-modal-form/>
82. Victorelec. (2020). Bootstrap don't close modal on click outside or ESC key.
Vicolinker .
<https://www.vicolinker.net/bootstrap-dont-close-modal-on-click-outside-or-esc-key/>
83. Contributors, M. O. J. T. A. B. (s. f.-b). *Modal*.
<https://getbootstrap.com/docs/5.0/components/modal/#via-javascript>
84. Fu, J. (2021, 29 diciembre). Perform Speech Synthesis in Your JavaScript Applications. *Medium*.
<https://betterprogramming.pub/perform-speech-synthesis-in-your-javascript-applications-ac3efa1eb6fa>
85. *Password Recovery Methods*. (s. f.).
<http://lastbit.com/password-recovery-methods.asp>
86. Greg, & Greg. (2019). Password Recovery Techniques: How does Access Password Recovery Tool Work? | Password Recovery for Access. *Password Recovery for Access*. <https://www.passwordaccess.com/es/access-password-recovery/>
87. Perez, A. (2018, 5 abril). Mejores prácticas de usabilidad para restablecer contraseñas. *Medium*.
<https://medium.com/@AlicePerez/mejores-practicas-de-usabilidad-para-restablecer-contrase%C3%B1as-d1b1502d2259>
88. TechRepublic. (2023, 2 junio). Five password reset options for online apps. *Security Archives*.
<https://www.techrepublic.com/article/five-password-reset-options-for-online-apps/>
89. *Homework: Adding security to your website - Django Girls Tutorial: Extensions*. (s. f.).
https://tutorial-extensions.djangogirls.org/en/authentication_authorization

90. Kseso. (2016, 9 agosto). *Recorte y centrado automático de imagen con object-fit: cover* 8.9.16. EsCss.

<https://escss.blogspot.com/2015/01/recorte-centrado-automatico-imagen.html>

91. Torres, P. (s. f.). *Recortar imágenes solo con CSS*. Codepen.

<https://codepen.io/tpedro/pen/xxbwdRm>

92. Agencia de Comunicación SMiLE. (s. f.). ▷ *Guía Bootstrap 5* [2023] SMiLE Comunicación. <https://smilecomunicacion.com/diseno-web/guia-bootstrap/>

93. *Bootstrap Tabs - examples & tutorial*. (s. f.). MDB - Material Design for Bootstrap.

<https://mdbootstrap.com/docs/standard/navigation/tabs/>

94. *Django* 🔗 *Handling multiple forms in single view*. (s. f.).

<https://www.tutorialspoint.com/django-handling-multiple-forms-in-single-view>

95. Paco. (s. f.). *Cómo personalizar un input file*. CodePen.

<https://codepen.io/borrowwebdesign/pen/oNXzXqK>

96. GeeksforGeeks. (2020). *ChoiceField* Django Forms. GeeksforGeeks.

<https://www.geeksforgeeks.org/choicefield-django-forms/>

97. *Django*. (s. f.-d). Django Project.

<https://docs.djangoproject.com/en/4.0/topics/forms/formsets/>

98. Singh, T. (2021, 25 diciembre). *Adding forms dynamically to a Django formset - All About Django* - Medium. *Medium*.

<https://medium.com/all-about-django/adding-forms-dynamically-to-a-django-formset-375f1090c2b0>

99. *How to annotate Count with a condition in a Django queryset*. (s. f.). Stack Overflow.

<https://stackoverflow.com/questions/33775011/how-to-annotate-count-with-a-condition-in-a-django-queryset>

100. #29196 (*Chaining multiple filters duplicates `INNER JOIN` for the final query*) – *Django*. (s. f.). <https://code.djangoproject.com/ticket/29196>

101. *Django - Annotate with count across ManyToMany relationships*. (s. f.). Stack Overflow.

<https://stackoverflow.com/questions/52733085/django-annotate-with-count-across-manytomany-relationships>

102. *Django*. (s. f.-c). Django Project.

<https://docs.djangoproject.com/en/4.1/topics/db/queries/#complex-lookups-with-q>

103. Rosanitsch, S. (2022). How to Build a JavaScript Audio Transcript Application. *News, Tutorials, AI Research*.

<https://www.assemblyai.com/blog/javascript-audio-transcript/>

104. Jaysha. (2020, 5 junio). *Using Django Pagination*. Ordinary Coders.

<https://ordinarycoders.com/blog/article/django-pagination>

105. *Bootstrap Sidebar Tutorial - Step-by-step tutorial with 5 sidebar templates [updated in 2021]*. (2022, 17 enero). Bootstrapious - Free Bootstrap Themes & Templates. <https://bootstrapious.com/p/bootstrap-sidebar>

106. *Django queries: how to filter objects to exclude id which is in a list?* (s. f.). Stack Overflow.

<https://stackoverflow.com/questions/2354284/django-queries-how-to-filter-objects-to-exclude-id-which-is-in-a-list>

107. *¿Cómo crear peticiones asíncronas con Javascript?* (s. f.). CódigoFacilito.

<https://codigofacilito.com/articulos/como-crear-peticiones-js>

108. Amorin, D. (2022). Cómo usar Django Messages Framework. *Diego Amorin*.

<https://diegoamorin.com/django-messages-framework/>

109. *Django*. (s. f.-e). Django Project.

<https://docs.djangoproject.com/en/4.2/ref/contrib/messages/>