



# GroupFinder

11/06/2021

---

Antonio Manzano Muñoz  
IES Rafael Alberti  
Cádiz

## 1. Índice

### 1. Índice

1. Expectativas y objetivos
2. Tecnologías
3. Antecedentes

### 2. Introducción

### 3. Descripción

### 4. Instalación

### 5. Guía de estilos y prototipado

### 6. Diseño

### 7. Desarrollo

### 8. Pruebas

### 9. Despliegue

### 10. Manual

### 11. Conclusiones

### 12. Bibliografía

### 13. Anexo

## 2. Introducción

### Expectativas y objetivos

El objetivo de esta aplicación web es crear un punto de encuentro para los jugadores del juego World of Warcraft para que puedan crear y buscar grupos basados en un nivel calculado por una empresa externa al juego de la cual consumiremos dicha información a través de una API. Dicho nivel va vinculado a cada personaje del juego, el cual podrás vincular a tu cuenta de usuario para así poder buscar grupo. El usuario creador del grupo podrá elegir a qué usuarios acepta en dicho grupo teniendo en cuenta los parámetros que él crea convenientes.

### Tecnologías

Las tecnologías usadas en el desarrollo son las siguientes:

- React: Esta elección ha sido sencilla, es de los frameworks más potentes de frontend además de uno de los más usados, el soporte y el futuro que le da el hecho de que sea propiedad de Facebook la convierte en una elección más que segura. Otro punto importante es que es el único visto en clase y que además estoy usando en la FCT.
- Flask: A la hora de elegir framework de backend empecé sin tener preferencia por ninguno, pero cuanto más trabajaba con Flask en la FCT más me enamoraba de Python y de dicho framework, por lo cual acabé tomando la decisión de que sería el lenguaje que usaría para el backend de mi aplicación.
- PostgreSQL: Ya había trabajado en la FCT con PostgreSQL y su conexión con Flask en la FCT, además del uso de sus datos con el ORM SQLAlchemy, por lo cual una vez más decidí elegir la tecnología con la que estaba trabajando paralelamente en la empresa.

## Antecedentes

No conozco de aplicaciones que cumplan la misma función que la que ofrece mi producto, hay otras webs y aplicaciones que dan datos de los personajes pero ninguna se encarga de ayudarte a encontrar grupos para jugar. Tengo constancia de que en el pasado [warcraftlogs](#) intentó hacer algo parecido, pero nunca terminaron de implementarlo. También hay foros donde la gente se organiza para buscar compañeros, pero nuestra aplicación le da un toque mucho más moderno y actual.

### 3. Descripción

El resultado obtenido es el esperado en prácticamente todas las funcionalidades añadidas, a continuación vamos a ver un desglose de las funcionalidades y su comparación con las expectativas.

- Login y registro. En esta parte no solo cumplimos con el objetivo de tener un login y un registro, sino que además lo superamos al añadir una validación por email al registro, lo que le da una capa extra de seguridad.
- Vinculación del personaje a la cuenta del usuario. En esta funcionalidad encontramos el único objetivo al que no se ha llegado de manera completa. Podemos vincular un personaje existente en el juego con nuestra cuenta de usuario, pero no hay ningún tipo de validación de que ese personaje realmente te pertenece ya que para implementar dicha funcionalidad haría falta poder usar un login a través de Battlenet para verificar la cuenta al cual no es posible acceder. Por lo tanto, sí, podemos añadir personajes a nuestra cuenta aunque no sean nuestros.
- Creación de grupo. La creación de grupo funciona sin problemas, añadiendo los campos necesarios para que el usuario que vaya en busca de grupo tenga toda la información necesaria.
- Gestión de grupo. La gestión del grupo la realiza el usuario que lo crea, pudiendo añadir y/o eliminar gente de dicho grupo. Se realiza en la misma ventana que la creación de grupo, solo que esta cambia al detectar que el usuario ya pertenece a un grupo.
- Búsqueda de grupo. Funciona tan bien como esperábamos, muestra una lista de grupos con su información y un botón que nos permite unirnos a la lista de candidatos de dicho grupo.

## 4. Instalación

La instalación del proyecto es sencilla, el primer paso es acudir al [repositorio](#) y descargarlo en una carpeta de nuestro ordenador, entramos en la carpeta front y en nuestra cmd ejecutamos el comando `'npm install'`, una vez se instale todo procedemos a volver a la carpeta anterior en la cual se encuentra el archivo `docker-compose.yml` y una vez más en nuestra cmd situada en ese directorio ejecutamos el comando `'docker compose up'`. Este proceso tarda un tiempo en realizarse.

```
front_1 | @wdwds@: Project is running at http://172.21.0.5/
front_1 | @wdwds@: webpack output is served from
front_1 | @wdwds@: Content not from webpack is served from /app/public
front_1 | @wdwds@: 404s will fallback to /
front_1 | Starting the development server...
front_1 |
front_1 | Compiled with warnings.
front_1 |
front_1 |   src/pages/GroupCreation/GroupCreation.jsx
front_1 |     Line 138:22:  Expected '===' and instead saw '=='  eqeqeq
front_1 |
front_1 |   src/pages/Personajes/Personajes.jsx
front_1 |     Line 23:8:  React Hook useEffect has missing dependencies: 'cargarpersonajes' and 'props'. Either include
front_1 | them or remove the dependency array. However, 'props' will change when *any* prop changes, so the preferred fix is to d
front_1 | estructure the 'props' object outside of the useEffect call and refer to those specific props inside useEffect  react-ho
front_1 | oks/exhaustive-deps
front_1 |
front_1 | Search for the keywords to learn more about each warning.
front_1 | To ignore, add // eslint-disable-next-line to the line before.
front_1 |
```

Cuando en nuestra consola veamos lo mismo que en la imagen anterior nuestra aplicación ya estará instalada, podremos visitar los distintos puntos:

- Nuestra aplicación ya funcionando. [GroupFinder](#)
- [Adminer](#), que es un gestor de base de datos del que podremos ver la distinta información de nuestra aplicación. Para logearnos debemos de poner como motor de base de datos PostgreSQL, en servidor pondremos db, como usuario postgres, como contraseña root y como base de datos groupfinder.

## 5. Guía de estilos y prototipado

### Tipografía







Para nuestra web necesitamos una tipografía que no se haga pesada y sea liviana para las lecturas más largas. Por eso me he decantado por Open Sans ya que en su formato Regular 400 me parece perfecta para lecturas largas y que además da un toque moderno y desenfadado a la web ya que no es demasiado seria.

Almost before we knew it, we had left the ground.

## Color

El proyecto está enfocado en World of Warcraft un juego con muchos años de historia tanto con este título como con otros juegos que hacen de precuela de la historia de dicho universo. El color usado para resaltar enlaces, palabras importantes y pequeños detalles usaremos un amarillo característico de World of Warcraft. (Código de color: #ffd100) El fondo de la página web será de tonos negros y grises para facilitar la lectura ,además el negro nos transmite elegancia y satisfacción. Al elegir los tonos negros como color de fondo la letra será de color blanco. Además a los usuarios registrados se les identificará según el color de su clase (colores oficiales por parte de World of Warcraft).

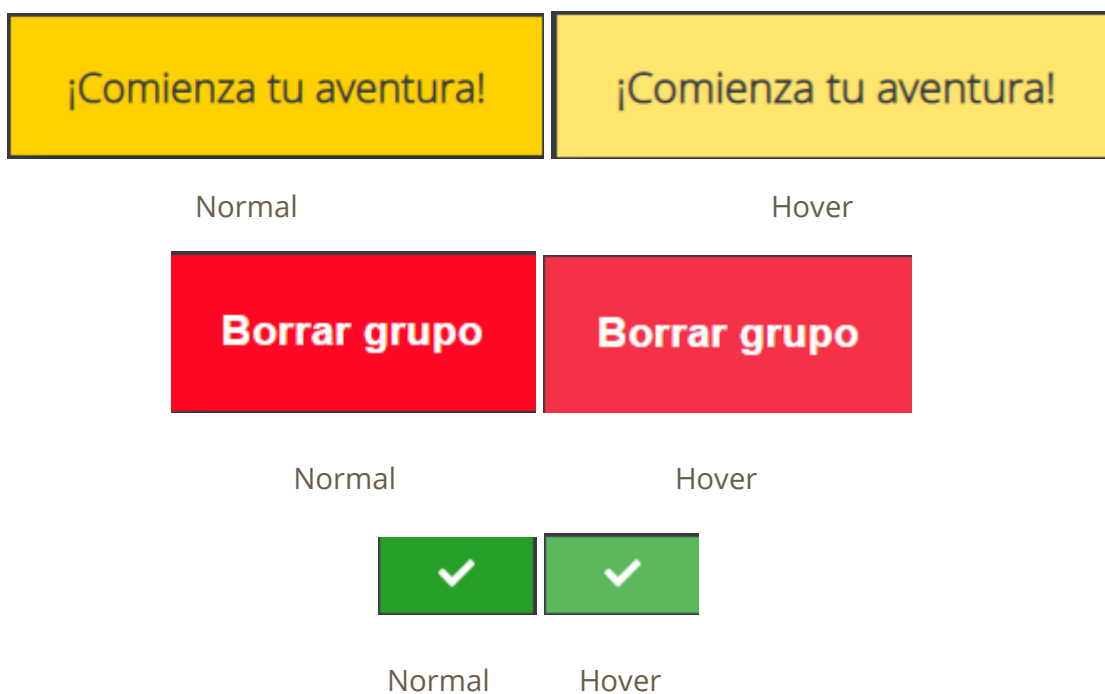
## Resumen de colores usados:

-  #ffd100: Como comenté anteriormente este color es el usado para resaltar detalles, también es el color principal de los botones de la página.
-  #fde9a2: Color que se usa en los botones al hacer hover para resaltar que estamos pasando por encima de uno.
-  #373a40: Este es el gris elegido para ser el color del cuerpo de nuestra página, del header y del footer.
-  #242424: Este es el segundo gris, escogido para ser el color de fondo de la web.
-  #ff0825: Este es el color rojo que llevarán los botones que realicen una acción que se asocie a botones para rechazar o incluso abandonar grupos.
-  #27a027: Este verde se usará única y exclusivamente en el botón que nos permite invitar a gente al grupo, ya que se asocia al verde a este tipo de cosas.



## Botones

A continuación vamos a ver los distintos tipos de botones que contiene nuestra página web:



## Iconografía

Estos son los iconos usados en nuestra aplicación:



Usado para aceptar a gente en el grupo.



Usado para rechazar gente que desea unirse al grupo.



Usado para recargar la tabla de candidatos o de grupo.



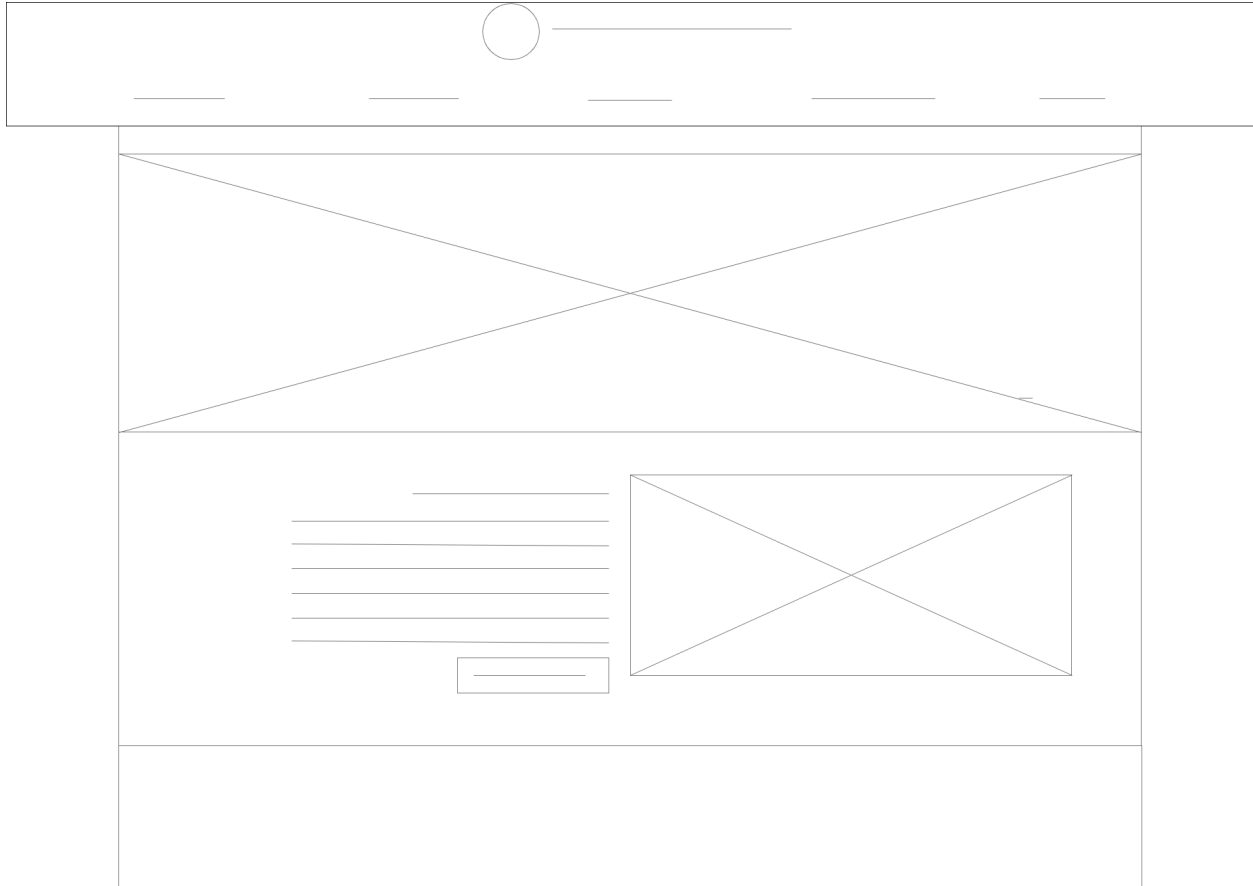
Usado para solicitar unirse a grupo.



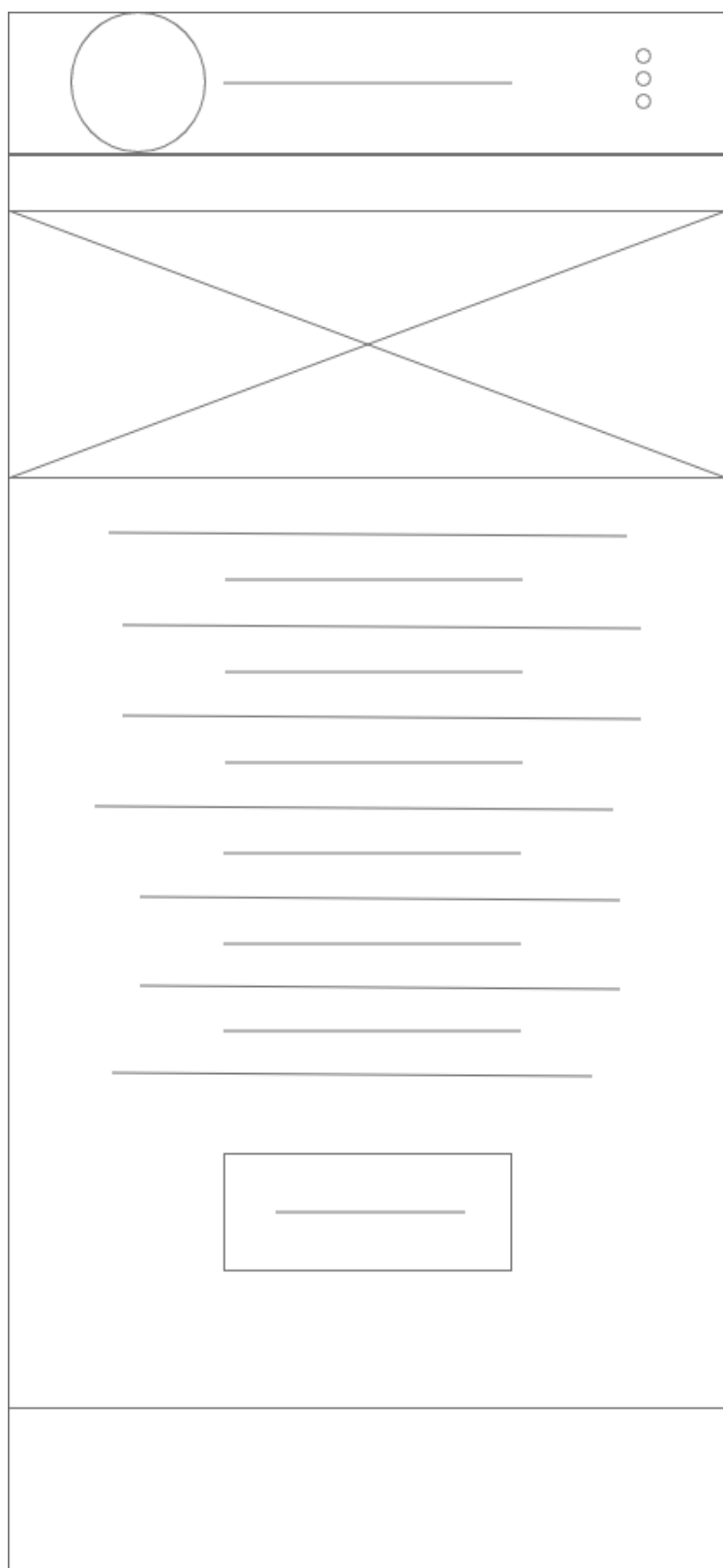
Pulsar para poder cambiar el Battletag en el perfil.

## Wireframes

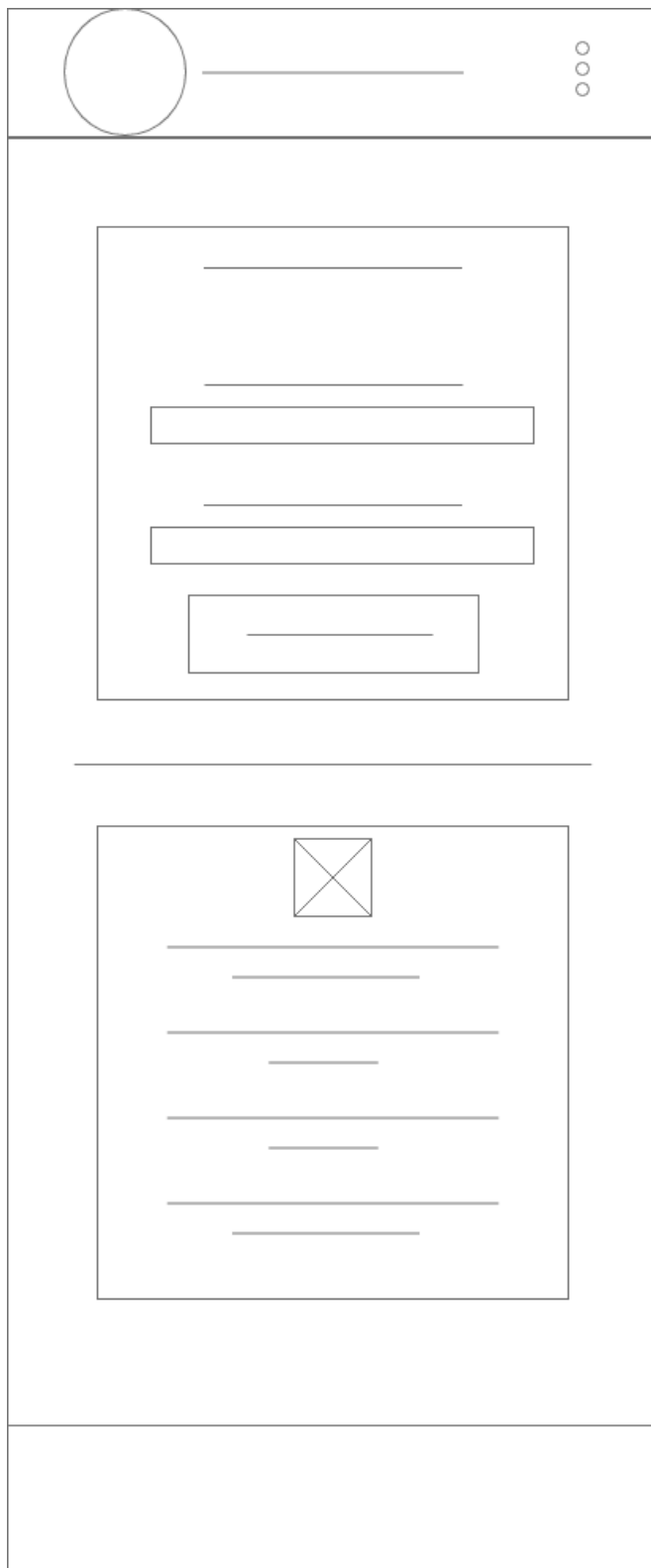
### Home



Web







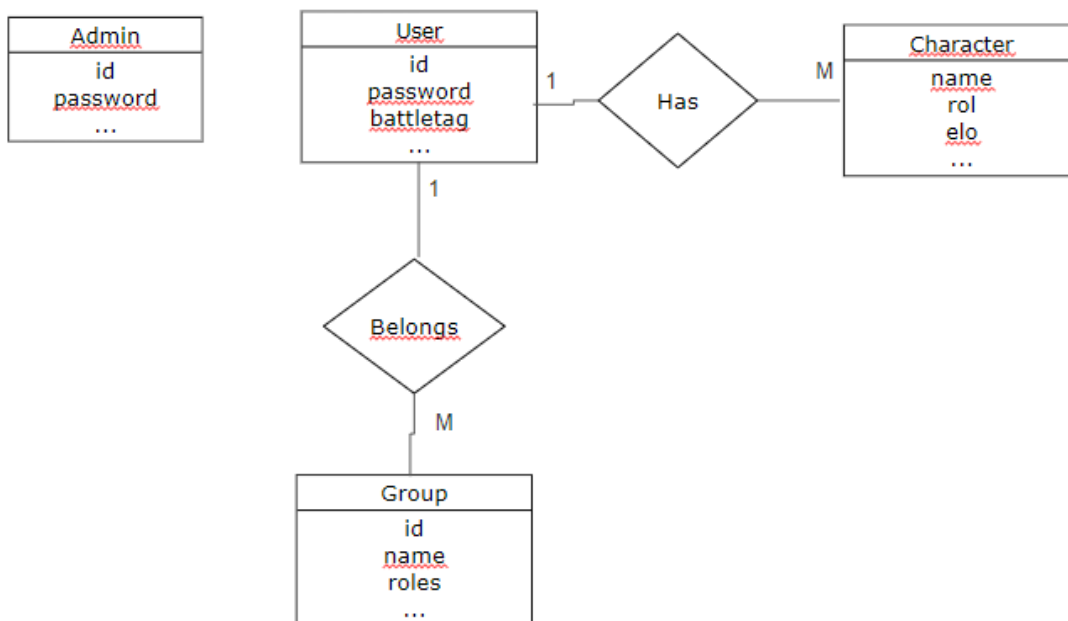
Móvil

El resto de los wireframes se encuentran en el anexo.

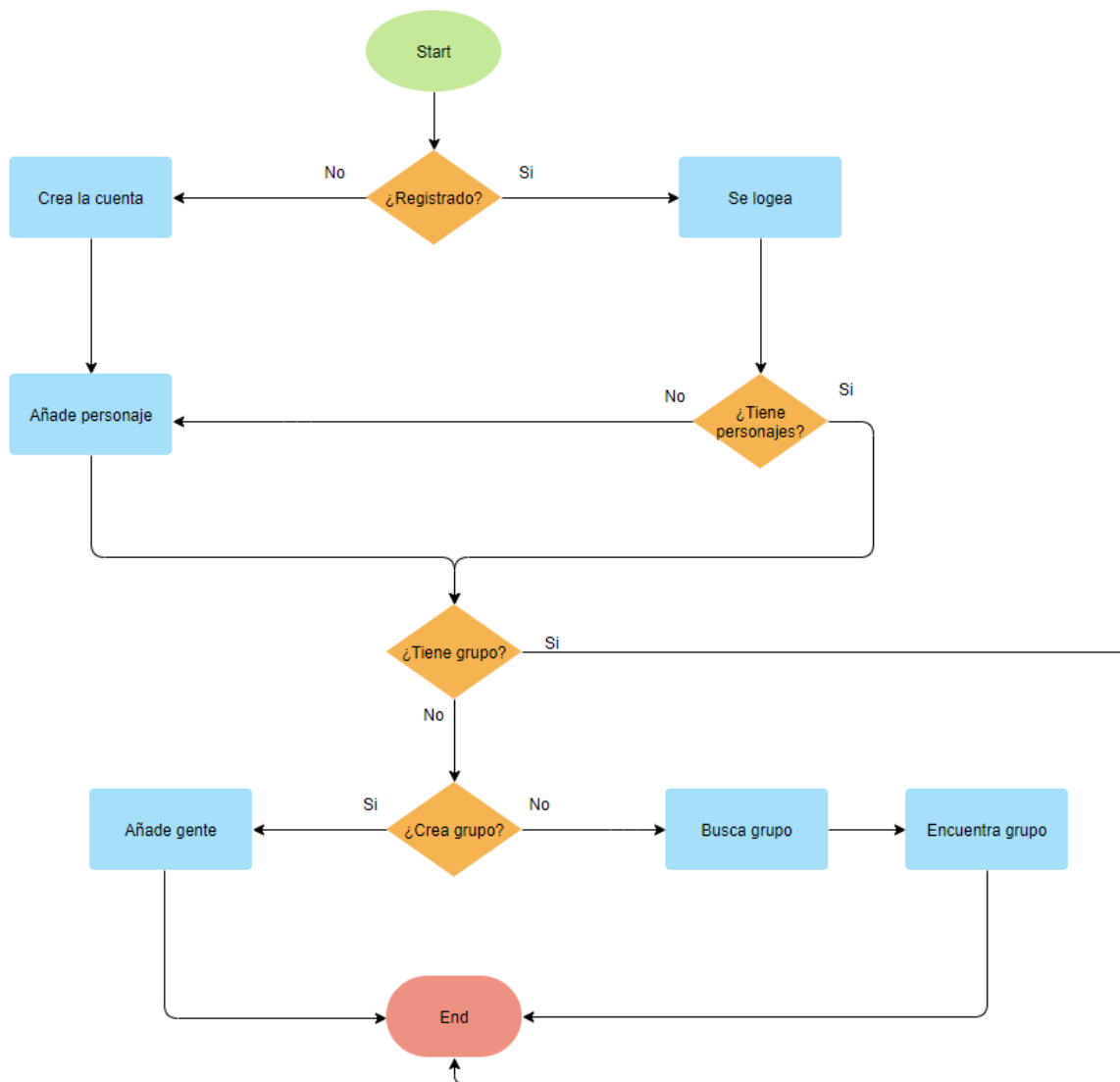
## 6. Diseño

Nuestra base de datos es de tipo relacional, a continuación veremos los distintos esquemas:

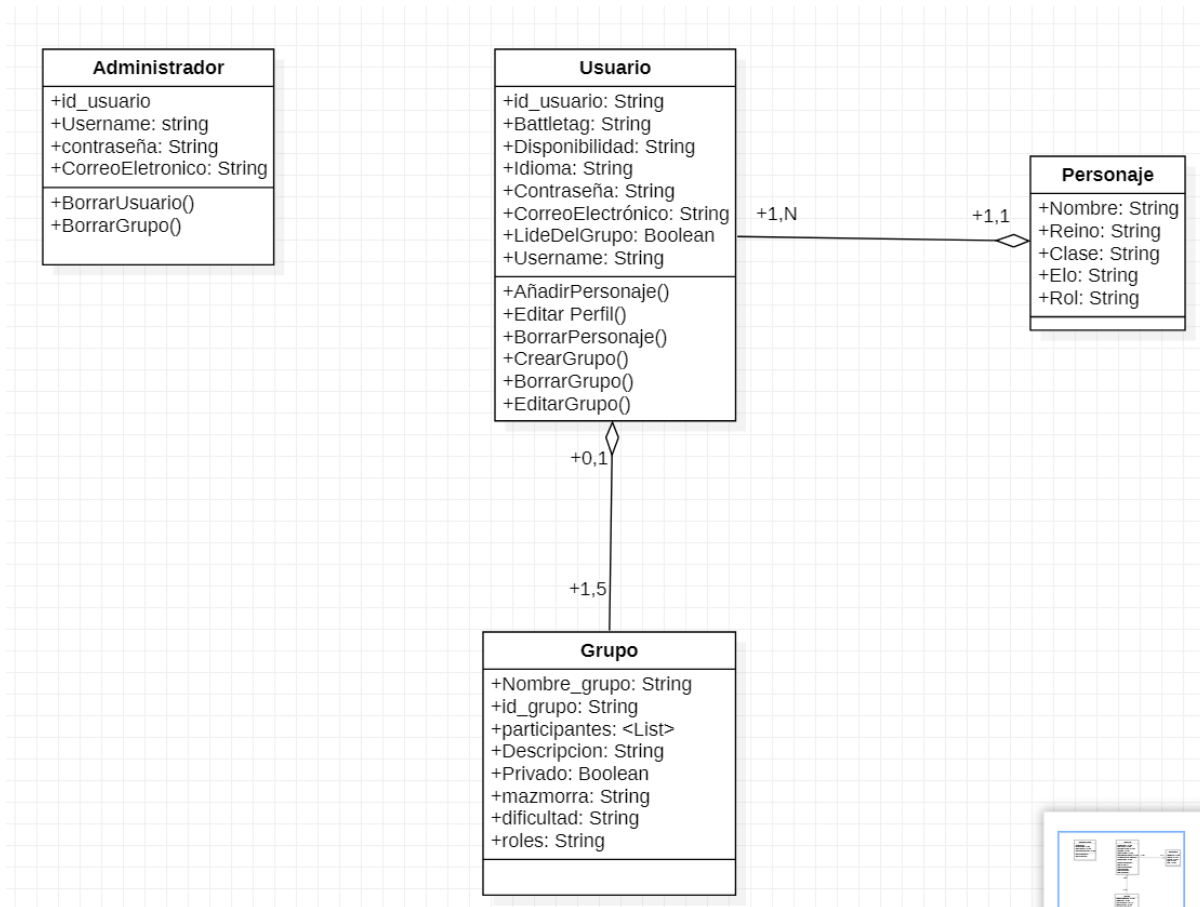
- Entidad-relación:



- Diagrama de flujo:



- Diagrama de clase:





Los endpoints de la aplicación son los siguientes:

- **/user/register** Este endpoint nos permite registrar un usuario.
- **/user/activate/<string:codigo>** Este endpoint nos permite verificar la cuenta del usuario.
- **/user/login** Este endpoint nos permite loguearnos y nos devuelve los datos necesarios del usuario.
- **/character/register/<int:user\_id>** Este endpoint nos permite añadir un personaje a un usuario.
- **/characters/<int:user\_id>** Este endpoint nos devuelve la información de un personaje en específico.
- **/group/creation/<int:user\_id>** Este endpoint crea un grupo.
- **/characters/group/<int:group\_id>** Devuelve los personajes de un grupo.
- **/group/<int:group\_id>** Devuelve los datos de un grupo específico.
- **/group/<int:group\_id>/request/<int:user\_id>** Realiza la petición para poder unirse a un grupo.
- **/characters/queue/<int:group\_id>** Muestra los usuarios que están en la lista de candidatos de un grupo.
- **/group/user/<int:user\_id>** Acepta al usuario candidato en el grupo.
- **/group/user/<int:user\_id>** Rechaza al usuario candidato del grupo.
- **/groups** Devuelve la información de todos los grupos.
- **/user/<int:user\_id>** Devuelve la información de un usuario concreto.
- **/user/battletag/<int:user\_id>** Nos permite cambiar el battletag de un usuario.
- **/user/logout** Nos permite cerrar sesión metiendo en la blacklist al token correspondiente.
- **[DELETE]/group/<int:user\_id>** Nos permite borrar el grupo.

Hay un sistema de protección de endpoints a través de tokens el cual no está implementado en todos ya que hacía la labor de desarrollo muy pesada, puede comprobar que funciona en el endpoint `/user/<int:user_id>`.

## 7. Desarrollo

El desarrollo de mi aplicación empezó a principios de curso con el diseño de la web en la asignatura de DIW, en dicha asignatura hice wireframes y mockups de la aplicación, además de el esqueleto en html y el estilo en css, los cuales usaría posteriormente en react para darle forma a la web.

Luego, empecé con el back del cual tuve una primera versión en Flask que era funcional y contaba tanto de modelos, como registro, login y varios endpoints necesarios para mi aplicación pero la organización y estructura se estaba volviendo demasiada enrevesada por cual decidí rehacerlo de cero, usando la librería Flask-Restful de Flask lo cual me llevo un tiempo hasta que conseguí pulirlo y adaptarme a la nueva manera de hacer las cosas.

Siguiendo con el back y una vez teniendo la estructura mucho más limpia y eficaz empecé por hacer un registro el cual mandaba un correo electrónico con la clave para verificar la cuenta y un login, con los cuales no tuve ningún problema. Otro paso importante para dejar terminada la estructura del proyecto fue relacionar los modelos mencionados anteriormente para a la hora de hacer los endpoints poder hacer las queries.

El siguiente paso fue hacer los endpoints necesarios para la pantalla de personajes, entre los cuales están:

- Un endpoint que devuelve todos los personajes de un usuario en caso de que este los tuviera.
- Otro el cual a través de dos datos (nombre y reino) nos permite añadir el personaje a dicho usuario.

No tuve ningún problema en esta parte ya que había hecho cosas parecidas en la FCT, la única decisión importante que tuve que tomar fue no añadir la conexión de la cuenta de usuario de mi aplicación con la cuenta de Battle.net del usuario, ya que Blizzard no lo permite en aplicaciones de terceros que no hayan pasado un control previo, por lo cual en esta versión puedes añadir un personaje a tu cuenta que realmente en el juego no sea tuyo, lo cual no es demasiado problemático porque en nuestra aplicación solo se muestran datos ya públicos.

Una vez terminados estos endpoints y probarlos en Postman pasé a hacer estas ventanas en React, a nivel de diseño probé varias formas de añadir y mostrar los distintos personajes lo cual me llevó un tiempo. Una vez realizada la decisión lo siguiente fue hacer la conexión entre el front y el back con las llamadas a los endpoints descritos anteriormente lo cuál no me supuso demasiado problema.

El siguiente paso fue desarrollar los endpoints de la página de tu grupo, en los cuales desarrollé los siguientes endpoints:

- Usé de nuevo el endpoint que devuelve todos los personajes de un usuario para poder.
- Un endpoint que nos permite crear un grupo con los datos que nosotros queramos.
- Uno que nos devuelve todos los usuarios que pertenecen a nuestro grupo.
- Otro que devuelve todos los usuarios que son candidatos a unirse a nuestro grupo.
- Un endpoint que nos permita añadir a cada candidato a nuestro grupo.
- Y otro que nos permite rechazarlos.

El front de esta parte es el más complejo de la aplicación ya que hay muchas comprobaciones, la primera es que el usuario está logueado y verificado, si es así te deja entrar en la ventana, luego comprueba si ese usuario ya pertenece a un grupo o no, si no pertenece a un grupo nos muestra un formulario con el cual podrá crear el grupo a través del endpoint correspondiente y la pantalla cambia dinámicamente para mostrarnos la información del grupo actual, los usuarios que forman parte de nuestro grupo y los candidatos a dicho grupo, si el usuario ya pertenece a un grupo sin necesidad de crearlo se le muestra automáticamente esta ventana.

Por último hice el back de la página de buscar grupo, la cual contiene los siguientes endpoints:

- Un endpoint que nos devuelve todos los grupos.
- Un endpoint que nos permite solicitar acceso a dicho grupo, lo que hará que para los usuarios del grupo aparezcamos en la parte de candidatos.

El front de esta parte fue sencillo, un select que nos indica que elijamos el personaje con el cual queremos solicitar la entrada al grupo y una tabla la cual nos muestra los distintos grupos disponibles con un botón el cual hace uso del endpoint que nos permite solicitar acceso al grupo.

La herramienta usada para el control de versiones es GitHub junto a SourceTree que nos permite realizar las tareas básicas de git como hacer push y commits a través de una interfaz gráfica haciendo mucho más sencillo el uso de dicha tecnología.

## 8. Pruebas

Las pruebas a entregar es la documentación generada en postman de los distintos endpoints usados en nuestra aplicación. El cual podemos encontrar en el siguiente [enlace](#).

## 9. Despliegue

La tecnología usada para el despliegue es Docker ya que es la que hemos visto en clase, para esto he creado un archivo docker-compose.yml en el directorio raíz de mi proyecto y dos archivos Dockerfile, uno en la carpeta del front y otra en la del back.

## 10. Manual


Empezar a usar la aplicación web es sencillo, primero, recomiendo borrar el almacenamiento local del navegador ya que puede llegar a dar problemas.

Ahora sí, una vez estamos en el inicio de nuestra página web sin haber iniciado sesión solo podremos acceder a la ventana de búsqueda de grupo, en la cual por ahora solo veremos los grupos creados que hay en la aplicación. Para todo lo demás hace falta estar registrado y logueado por lo cual nuestro siguiente paso es ir a la ventana de login/register y proceder a registrarnos.

Para realizar el registro recomiendo poner un email válido para poder recibir el código de confirmación, en caso de no querer ingresar una dirección de correo real el código se puede obtener de la cmd ya que he dejado habilitado un print por si a la hora de probar el proyecto el profesorado no desea usar en correo electrónico personal.

Una vez registrados verificamos la cuenta, ya que sin esto no se nos dejará iniciar sesión, listo todo esto, iniciamos sesión y se nos redireccionará a la ventana de personajes, donde podremos añadir un personaje ya existente en el juego World Of Warcraft a nuestra cuenta (datos de personajes existentes para poder realizar las pruebas en el archivo personajes.txt) . Una vez tengamos uno o más personajes podremos disfrutar de las demás funcionalidades de nuestra página. Podemos ir a la página de Buscar grupo y ahora además de ver los grupos creados podremos elegir el personaje con el cuál queremos optar a entrar en esos grupos y un botón que ahora sí nos dejará ser candidatos a entrar al grupo deseado. Si ya tenemos grupo o queremos crear uno el siguiente paso es ir a la pantalla Tu grupo, si no tenemos grupo veremos un formulario que al rellenar se crea nuestro grupo, del cual serías el líder que es quién elige quién puede acceder al grupo o no.

Si ya tenemos grupo pero no somos el líder (a través de la ventana de Buscar grupo hemos optado a uno y nos han aceptado) veremos los miembros y los candidatos pero no



podremos realizar ninguna acción con ellos, si podremos abandonar el grupo(en caso de ser líder ese botón sirve para borrar el grupo, no solo abandonarlo). En caso de ser líder si que podemos expulsar a los miembros del grupo y aceptar o rechazar a los candidatos.

Por último tenemos la ventana de Perfil en la cual podemos ver los datos de nuestro usuario, editar nuestro battletag y cerrar sesión.

## 11. Conclusiones

El resultado comparado con la idea inicial, como ya se ha ido comentando durante los distintos puntos tratados en este documento, es bastante bueno, teniendo en cuenta el no poder haber accedido al logueo de Blizzard. En diseño considero que la aplicación ha mejorado bastante, aun con un margen de mejora amplio que es una de las mejoras futuras pendientes.

Una mejora futura sería mejorar el sistema de protección de rutas del frontend, poner en funcionamiento el refresh token en el front,añadir un chat y un apartado de guías. Otra mejora sería añadir un sistema de almacenamiento local más funcional, porque el localStorage puede llegar a dar problemas, desgraciadamente esto lo descubrí tarde y no tuve tiempo de gestionarlo de distinta manera.

Como conclusión personal decir que estoy contento con el resultado de la web ya que cumple con la función con la que fue ideada a principios de curso, aunque tenga margen de mejora.

## 12. Bibliografía

Documentación oficial de Flask. (s. f.). Documentación oficial de Flask. Recuperado 11 de junio de 2021, de <https://flask.palletsprojects.com/en/2.0.x/>

Documentación oficial de React. (s. f.). Documentación oficial de React. Recuperado 11 de junio de 2021, de <https://es.reactjs.org/docs/getting-started.html>

Flask RestFul. (s. f.). Flask Restful. Recuperado 11 de junio de 2021, de <https://flask-restful.readthedocs.io/en/latest/>

RealPython. (s. f.). Realpython. Recuperado 11 de junio de 2021, de <https://realpython.com/>

Stackoverflow. (s. f.). Stackoverflow. Recuperado 13 de mayo de 2021, de <https://es.stackoverflow.com/>

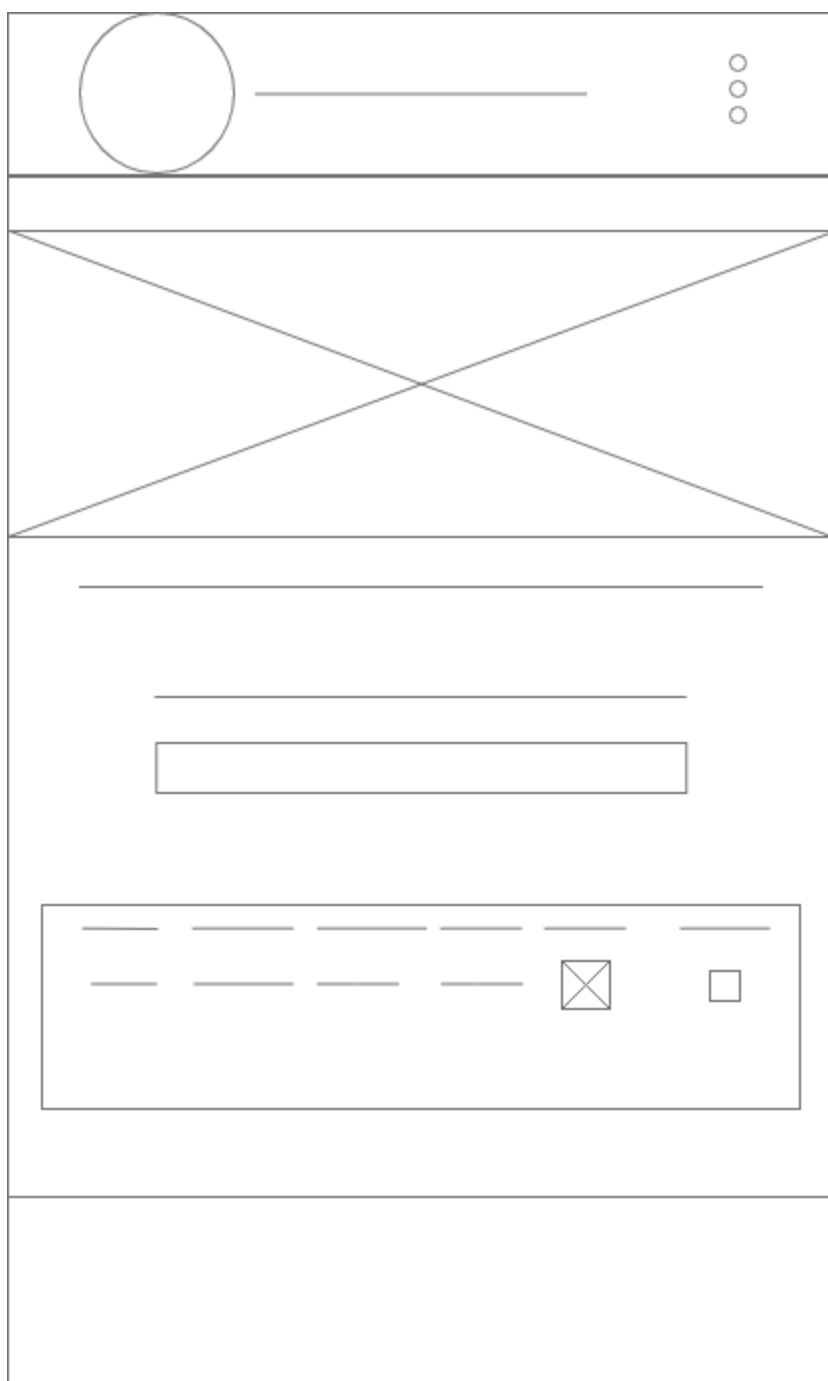
Colores de clase. (s. f.). Colores de clase. Recuperado 11 de junio de 2021, de [https://wowpedia.fandom.com/wiki/Class\\_colors](https://wowpedia.fandom.com/wiki/Class_colors)

Modelos en Flask. (s. f.). Modelos en Flask. Recuperado 11 de junio de 2021, de <https://github.com/alxngr/flask-api-memorandum/blob/main/documentation/flask-restful.md#creating-a-model>

## 13. Anexo

## Tu grupo

Web

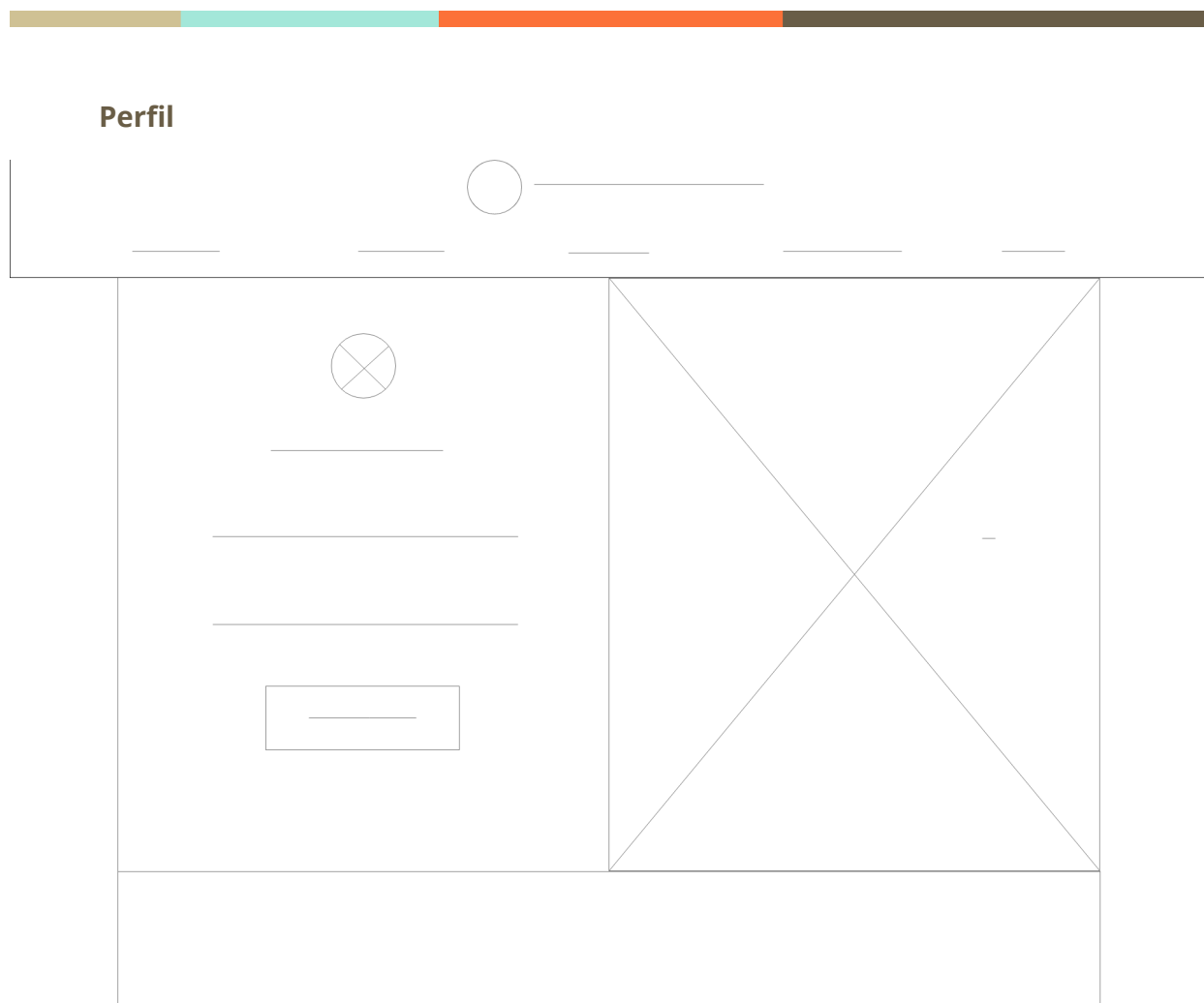


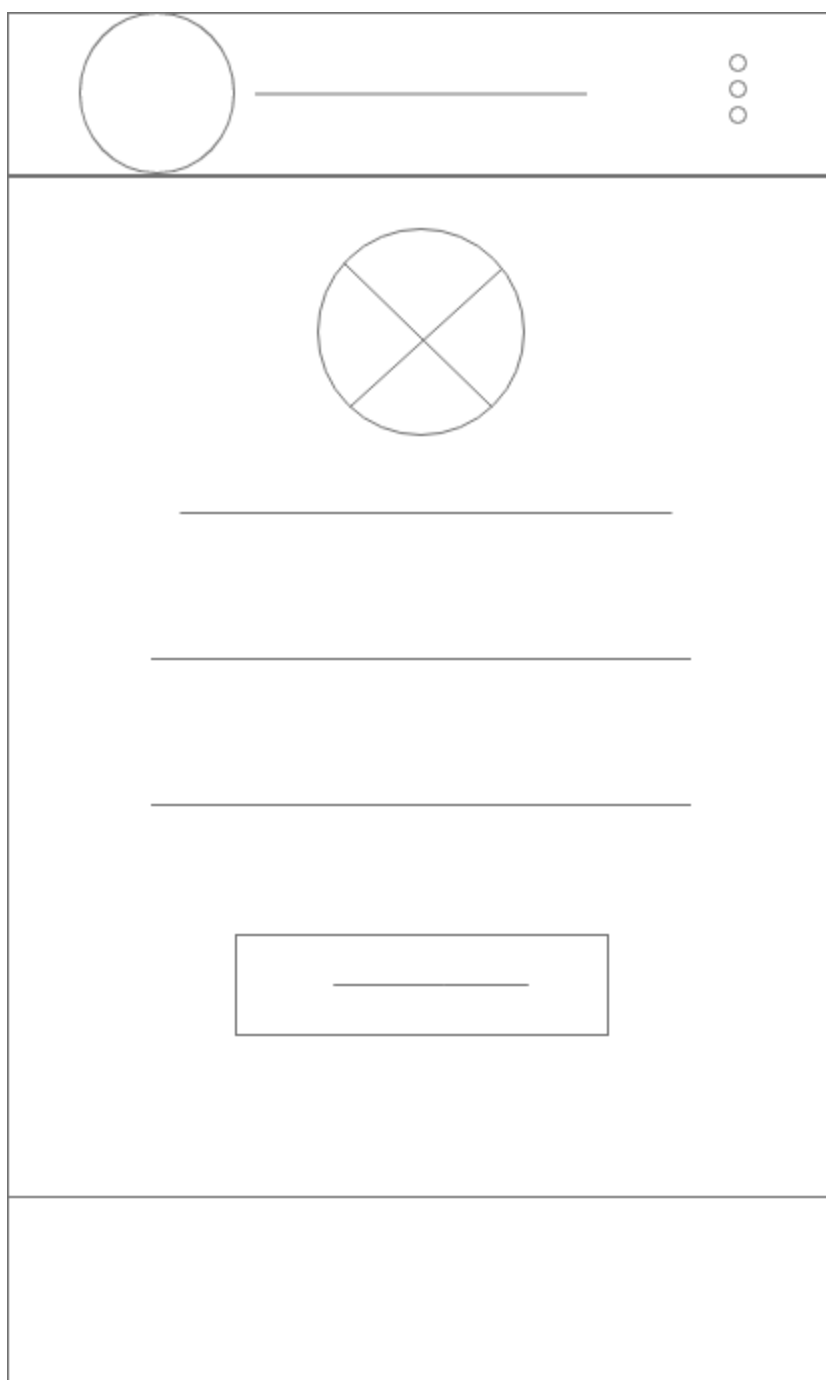
Móvil



The wireframe illustrates a mobile application interface. At the top, there is a header bar containing a circular profile picture placeholder, a horizontal line for a search bar, and a menu icon (three vertical dots). Below the header, the main content area is divided into three sections. The first section is a large container for a list of items, each represented by a horizontal line. The second section is a smaller container for a list of items, each represented by a horizontal line. The third section is a container for a list of items, each represented by a horizontal line. Below the list sections, there are two filter sections. Each filter section contains a horizontal line, a checkbox, and a button. The bottom of the screen features a wide, empty rectangular area, likely a footer or a space for additional content.

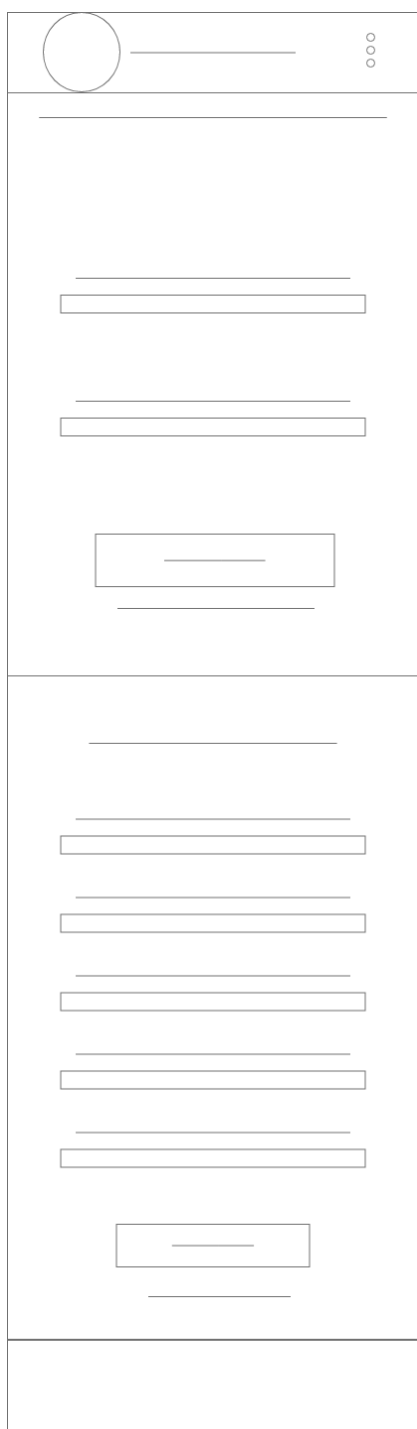
Móvil





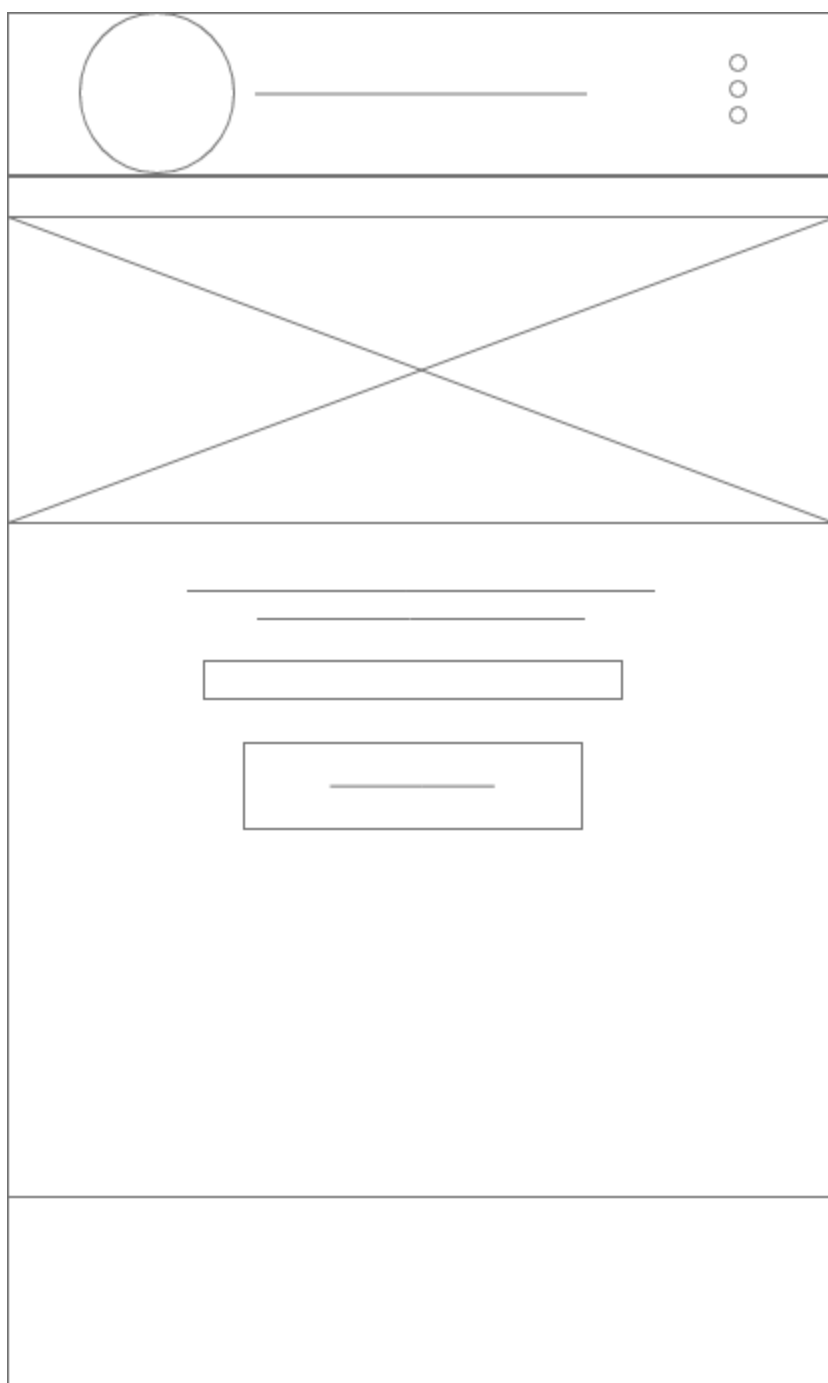
**Perfil**

## Web



Móvil

Web



Móvil