



## **Trabajo de Final de Grado**

**GRADO DE INGENIERÍA INFORMÀTICA**

**Facultad de Matemáticas i Informàtica**

**Universidad de Barcelona**

---

**Creación de una red social de arte**

---

**Sergi Hurtado Abad**

Director: Josep Vañó Chic  
Realitzado en: Departament de  
Matemàtiques i Informàtica

Barcelona, 01 de febrer de 2018

## Resum

Aquest projecte consisteix en l'anàlisi, el disseny i l'implementació d'una pàgina web que actua com a xarxa social entre artistes i com una plataforma perquè artistes puguin donar-se a conèixer.

Per a la implementació d'aquest projecte, s'ha utilitzat el framework Angular 6 per la part del Frontend, el framework Laravel per la part del Backend, els llenguatges de programació web HTML, LESS, TypeScript i PHP i el gestor de base de dades MySQL.

Aquest lloc web permet als usuaris poder pujar publicacions a la web, xatejar entre usuaris o grups d'usuaris i visualitzar i comentar les publicacions dels usuaris que segueixen. A més, la web ofereix la possibilitat de poder publicar conjuntament creant equips entre diversos usuaris, on cada un té un paper definit dins de l'equip.

Aquesta web es pot dividir en les següents pantalles: La pantalla de les llistes, conté tota la base de dades de dibuixos, còmics i sèries d'animació pujades pels usuaris, també hi ha les llistes d'usuaris i les llistes d'equip. La pantalla de Xats, conté tots els xats públics de l'app, els xats privats de l'usuari i els xats de l'equip. La pantalla de Friends updates, conté les últimes 30 publicacions dels usuaris o equips als que segueix la persona loguejada. Finalment, la pantalla del perfil mostra tota la informació d'un usuari o un equip en concret i la galeria d'aquest amb les seves publicacions.

Pel que fa a les funcionalitats que proporciona la web, el programa permet pujar dibuixos, còmics, capítols, animacions i episodis i filtrar per puntuació, data, visites, filtres o per recerca; dins els detalls de cada publicació, els usuaris també poden votar-perquè escali en el rànquing o comentar-la. En l'apartat de xats, el programa permet crear xats, afegir gent a un grup de xat i xatejar en un xat. En els perfils d'equip o usuari, el programari permet seguir a aquests usuaris, crear un equip, modificar la informació del seu perfil i afegir o eliminar membres d'un equip o editar el seu paper en un equip.

## Resumen

Este proyecto consiste en el análisis, diseño e implementación de una página web que actúa como red social entre artistas y como una plataforma para que artistas puedan darse a conocer.

Para la implementación de este proyecto, se ha utilizado el framework Angular 6 para la parte del Frontend, el framework Laravel para la parte del Backend, los lenguajes de programación web HTML, LESS, TypeScript y PHP y el gestor de base de datos MySQL.

Este sitio web permite a los usuarios poder subir publicaciones a la web, chatear entre usuarios o grupos de usuarios y visualizar y comentar las publicaciones de los usuarios que siguen. Además, la web ofrece la posibilidad de poder publicar conjuntamente creando equipos entre varios usuarios, donde cada uno tiene un rol definido dentro del equipo.

Esta web se puede dividir en las siguientes pantallas: La pantalla de las listas, contiene toda la base de datos de dibujos, cómics y series de animación subidas por los usuarios, también están las listas de usuarios y las listas de equipo. La pantalla de Chats, contiene todos los chats públicos de la app, los chats privados del usuario y los chats del equipo. La pantalla de Friends updates, contiene las últimas 30 publicaciones de los usuarios o equipos a los que sigue la persona logueada. Finalmente, la pantalla del perfil muestra toda la información de un usuario o un equipo en concreto y la galería de este con sus publicaciones.

Con respecto a las funcionalidades que proporciona la web, el programa permite subir dibujos, cómics, capítulos, animaciones y episodios y filtrarlos por puntuación, fecha, visitas, filtros o por búsqueda; dentro de los detalles de cada publicación, los usuarios también pueden votarla para que escale en el ranking o comentarla. En el apartado de chats, el programa permite crear chats, añadir gente a un grupo de chat y chatear en un chat. En los perfiles de equipo o usuario, el software permite seguir a estos usuarios, crear un equipo, modificar la información de su perfil y añadir o eliminar miembros de un equipo o editar su rol en un equipo.

## Abstract

This project consists in the analysis, design and implementation of a web page that acts as a social network between artists and as a platform for artists to get to know each other.

In order to develop this project, the Angular 6 framework has been used for the Frontend part, the Laravel framework for the Backend part, the HTML, LESS, TypeScript and PHP web programming languages and the MySQL database manager.

This website allows users to upload posts to the web, chat between users or groups of users and view and comment on the publications of the users that they follow. In addition, the web offers the possibility of publishing jointly creating teams between several users, where each one has a defined role within the team.

This web can be divided into the following screens: The screen of the lists, contains all the database of drawings, cómics and animation series uploaded by users, there are also user lists and team lists. The Chats screen contains all the public chats of the app, the private chats of the user and the chats of the team. The Friends updates screen contains the last 30 publications of the users or teams that the person logged in follows. Finally, the profile screen shows all the information of a specific user or team and the gallery of this with its publications.

With regard to the functionalities provided by the web, the program allows you to upload drawings, cómics, chapters, animations and episodes and filter them by punctuation, date, visits, filters or by search; Within the details of each publication, users can also vote for it to climb the ranking or comment on it. In the chats section, the program allows you to create chats, add people to a chat group and chat in a chat. In team or user profiles, the software allows you to follow these users, create a team, modify your profile information and add or remove members of a team or edit their role in a team.

# Índice

RESUM.....	2
RESUMEN .....	3
ABSTRACT .....	4
ÍNDICE.....	5
ÍNDICE DE ILUSTRACIONES.....	7
<b>1 INTRODUCCIÓN.....</b>	<b>8</b>
1.1. MOTIVACIONES Y CONTEXTO DEL PROYECTO.....	8
1.2 DESCRIPCIÓN DEL PROYECTO .....	9
1.3 OBJETIVOS .....	9
1.3.1 <i>Objetivos generales</i> .....	9
1.3.2 <i>Objetivos específicos</i> .....	10
1.4 PLANIFICACIÓN TEMPORAL .....	10
1.4.1 <i>Diagrama de Gantt</i> .....	11
<b>2 ANÁLISIS .....</b>	<b>12</b>
2.1 COMPARACIÓN CON PÁGINAS WEBS SIMILARES .....	12
2.2 REQUISITOS INICIALES.....	13
2.2.1 <i>Requisitos funcionales</i> .....	13
2.2.2 <i>Requisitos no funcionales</i> .....	16
2.2.3 <i>Requisitos técnicos</i> .....	17
2.3 CASOS DE USO .....	18
2.3.1 <i>Diagrama de caso de uso</i> .....	18
2.3.2 <i>Descripción de los casos de uso textuales</i> .....	23
<b>3 DISEÑO .....</b>	<b>35</b>
3.1 ARQUITECTURA .....	35
3.1.1 <i>Controladores y Servicios</i> .....	35
3.1.2 <i>Modelo</i> .....	36
3.1.3 <i>Vista</i> .....	36
3.1.4 <i>Diagrama completo de la arquitectura</i> .....	36
3.2 DIAGRAMAS DE LA APLICACIÓN .....	37
3.2.1 <i>Diagrama de la base de datos</i> .....	37
3.2.2 <i>Diagrama de las clases del Modelo</i> .....	38
3.3 COMPONENTES DE LA VISTA .....	39
3.3 PANTALLAS.....	39
3.4 PROTOTIPADO GRÁFICO .....	44
<b>4 IMPLEMENTACIÓN .....</b>	<b>48</b>
4.1 CONFIGURACIÓN DEL PROYECTO Y LOS SERVIDORES.....	48
4.2 ESQUEMA ORGANIZATIVO DE FICHEROS.....	49
4.2.1 <i>Frontend</i> .....	49
4.2.1 <i>Backend</i> .....	50
4.3 EL MODELO .....	51
4.3.1 <i>Las clases</i> .....	51
4.3.2 <i>Las relaciones</i> .....	52
4.3.3 <i>Herencias</i> .....	52
4.3.4 <i>Migrations</i> .....	53
4.3.5 <i>Query</i> .....	53
4.4 STORAGE .....	54

4.5 MIDDLEWARE .....	55
4.6 LOG IN Y SIGN UP .....	55
4.6.1 Sign up.....	55
4.6.2 Login .....	56
4.7 ROUTING .....	57
4.8 BARRA DE NAVEGACIÓN LATERAL.....	58
4.8.1 Filtros y Tags.....	58
4.8.2 Ordenar .....	59
4.8.3 Search .....	60
4.9 CHATS .....	61
4.10 FRIEND'S ACTIVITY .....	62
4.11 EQUIPOS.....	62
4.11.1 Creación de un equipo.....	63
4.11.2 Gestionar equipos.....	63
4.11.3 Publicar como equipo y eliminar publicaciones .....	64
4.12 COMENTARIOS, VOTOS Y VISITAS .....	65
4.13 VISORES.....	66
4.13.1 Visor de cómics .....	66
4.13.2 Visor de episodios .....	66
<b>5 CONCLUSIONES .....</b>	<b>68</b>
5.1 FUTURAS AMPLIACIONES.....	69
<b>GLOSARIO .....</b>	<b>70</b>
<b>BIBLIOGRAFÍA .....</b>	<b>71</b>

# Índice de ilustraciones

ILUSTRACIÓN 1: DISTRIBUCIÓN DEL TIEMPO .....	10
ILUSTRACIÓN 2: PLANIFICACIÓN DEL PROYECTO .....	11
ILUSTRACIÓN 3: DIAGRAMA DE CASOS DE USO (TOPBAR) .....	19
ILUSTRACIÓN 4: DIAGRAMA DE CASOS DE USO (LIST VIEW) .....	20
ILUSTRACIÓN 5: DIAGRAMA DE CASOS DE USO (USER'S PROFILE/TEAM'S PROFILE) .....	21
ILUSTRACIÓN 6: DIAGRAMA DE CASOS DE USO (CHAT ROOMS VIEW) .....	22
ILUSTRACIÓN 7: DIAGRAMA DE CASOS DE USO (MANAGE TEAMS) .....	23
ILUSTRACIÓN 8: DIAGRAMA COMPLETO DE LA ARQUITECTURA .....	36
ILUSTRACIÓN 9: DIAGRAMA DE LA BASE DE DATOS .....	37
ILUSTRACIÓN 10: DIAGRAMA DE LAS CLASES DEL MODELO .....	38
ILUSTRACIÓN 11: PROTOTIPO DE LA PANTALLA PRINCIPAL DESLOGUEADO .....	45
ILUSTRACIÓN 12: PROTOTIPO DE LA PANTALLA PRINCIPAL LOGUEADO .....	46
ILUSTRACIÓN 13: PROTOTIPO DEL PERFIL DE UN USUARIO .....	46
ILUSTRACIÓN 14: PROTOTIPO DE LA PANTALLA DE FRIEND'S UPDATES .....	47
ILUSTRACIÓN 15: ESTRUCTURA DE FICHEROS DEL FRONTEND .....	48
ILUSTRACIÓN 16: ESTRUCTURA DE FICHEROS DE LOS COMPONENTES .....	49
ILUSTRACIÓN 17: ESTRUCTURA DE FICHEROS DEL BACKEND .....	50
ILUSTRACIÓN 18: ESTRUCTURA DE FICHEROS DE STORAGE .....	54
ILUSTRACIÓN 19: VISTA DEL SIGN UP .....	55
ILUSTRACIÓN 20: VISTA DEL LOG IN .....	57
ILUSTRACIÓN 21: BARRA DE NAVEGACIÓN LATERAL .....	58
ILUSTRACIÓN 22: TEMPLATE DE CREACIÓN DE DIBUJOS .....	59
ILUSTRACIÓN 23: OPCIONES DE ORDENAR LA LISTA .....	59
ILUSTRACIÓN 24: VISTA DE LISTS (DRAWS) .....	60
ILUSTRACIÓN 25: VISTA DE CHAT ROOMS (TEAM CHATS) .....	61
ILUSTRACIÓN 26: LISTA DE MIEMBROS, SEGUIDORES Y SEGUIDOS EN EL PERFIL .....	62
ILUSTRACIÓN 27: VISTA DE FRIEND'S UPDATES .....	62
ILUSTRACIÓN 28: LISTA DE MIEMBROS DE UN EQUIPO .....	63
ILUSTRACIÓN 29: VISTA DE MANAGE TEAMS .....	64
ILUSTRACIÓN 30: VISTA DEL PERFIL DE UN EQUIPO .....	65
ILUSTRACIÓN 31: VISTA DE COMENTARIOS DE UNA PUBLICACIÓN .....	65
ILUSTRACIÓN 32: VISTA DE LA SECCIÓN DE VOTOS Y VISITAS EN UNA PUBLICACIÓN .....	66
ILUSTRACIÓN 33: LISTA DE EPISODIOS CON EL VISOR DESPLEGADO .....	67

# 1 Introducción

## 1.1. Motivaciones y contexto del proyecto

Actualmente hay muchas aplicaciones o páginas web enfocadas en el mundo artístico que están dedicadas exclusivamente a subir fotos de tus dibujos o series de animación hechas fotograma a fotograma, pero ninguna es realmente práctica a la hora de establecer una conexión entre los artistas donde puedan hacer proyectos conjuntos o para dar a conocer a estos usuarios que realmente cuelgan buen contenido en la web y se esfuerzan por mejorar su arte, es decir, ninguna es muy consciente de la importancia de la parte social de estas páginas webs ni de que los usuarios se pongan en contacto entre ellos para realizar proyectos conjuntos, para participar en concursos o para aprender de otros artistas a través de chats públicos o privados.

Dejando de un lado la parte más social, también se ha observado que ninguna página web de arte logra tener cómics, dibujos y animaciones a la vez y casi ninguna las organiza en forma de ranking de votos o visitas, es decir, normalmente son webs especializadas únicamente en colgar un único tipo de arte. Con este proyecto se pretende agrupar estos tres tipos de arte en una sola página web para que el usuario pueda tener más variedad a la hora de publicar y a la hora de visualizar publicaciones, y de esta forma tener más variedad de artistas en la web.

Por este motivo, se realizará una página web para cubrir la necesidad de tener una red social dedicada únicamente al arte, que permita a sus usuarios comunicarse entre ellos, crear alianzas para determinados proyectos, recibir feedback de su trabajo gracias a otros usuarios de la web y sobre todo que puedan darse a conocer a través del ranking de la web.

Este proyecto no solo ayudaría a los artistas, también sería una buena herramienta para personas que están buscando artistas para sus proyectos o quizás para alguna empresa que necesite algún artista para algún producto suyo, de esta forma se forjarían equipos de usuarios que no tiene por qué ser necesariamente artistas y que estarían trabajando para un proyecto en conjunto. Es decir, es una muy buena herramienta para personas que estén buscando artistas para contratarlos en su empresa o simplemente para realizar proyectos suyos.

Así pues, la idea es crear una red social de arte, con la que la gente pueda conseguir recompensas realizando arte, darse a conocer como artistas, ponerse en contacto con otros artistas para aprender y crecer y que puedan entrar en un equipo de gente para realizar un proyecto conjunto gracias a su perfil de usuario que actúa como currículum de artista, donde tienen toda su información personal, su galería de arte, sus seguidores y los equipos de los cuales él es miembro.



## 1.2 Descripción del proyecto

El proyecto consiste en la realización de una página web que funcione como red social donde los usuarios puedan subir sus dibujos, sus cómics o sus animaciones y puedan votar, comentar y visualizar las obras de otros artistas y que incluya salas de chats donde poder crear equipos dentro de la web o simplemente para hablar entre ellos.

Podemos dividir este proyecto en dos aspectos principales:

- **Chat rooms y Teams:** Es la funcionalidad más importante de la página web, ya que el principal objetivo de esta página web es poder ponerse en contacto con otros artistas y formar equipos de usuarios con un objetivo común.

Esta funcionalidad consiste en crear salas de chat para varios usuarios o unirse en estas, donde puedan hablar y organizarse entre ellos. También se ofrece la posibilidad de crear cuentas conjuntas de equipo.

Una vez forman equipo, pueden publicar dentro de la página web como equipo conjunto y todo el reconocimiento que ganen sus publicaciones se verá reflejado en el perfil de equipo donde figurarán todos sus participantes y todas las obras realizadas por estos.

- **Red Social con rankings:** Esta página web funciona como una red social, es decir los usuarios pueden subir a su perfil dibujos, cómics o animaciones, pueden seguir a otros usuarios y ser notificados cuando esta suba algo, y pueden comentar el trabajo de otros artistas.

En la página principal de la web se mostrarán todos los dibujos, cómics, animaciones o usuarios ordenados en ranking, popular y/o aplicando filtros. De esta forma, los artistas pueden darse a conocer escalando en el ranking de un determinado tipo de arte en una determinada sección de filtros.

Para realizar esta página web, se usará el framework Angular en la parte de Frontend, Laravel para el Backend y MySQL para la base de datos de la web.

## 1.3 Objetivos

### 1.3.1 Objetivos generales

El objetivo principal de este proyecto es poder crear una herramienta donde los usuarios puedan encontrar a otros usuarios para trabajar conjuntos y darse a conocer como artistas.

### 1.3.2 Objetivos específicos

Profundizar en diferentes conocimientos relacionados con el desarrollo de páginas web. Algunas de las tecnologías que se usarán son las siguientes:

- Angular (TypeScript, HTML y LESS/CSS)
- MySQL
- Laravel (PHP)

### 1.4 Planificación temporal



Nombre	Fecha de inicio	Fecha de fin
☐ ● Presentación	18/09/18	24/09/18
● Introducción y motivaciones	18/09/18	19/09/18
● Objetivos	20/09/18	21/09/18
● Planificación	24/09/18	24/09/18
☐ ● Análisis	25/09/18	4/10/18
● Requisitos	25/09/18	27/09/18
● Casos de uso	28/09/18	4/10/18
☐ ● Diseño	5/10/18	19/10/18
● Arquitectura	5/10/18	11/10/18
● Diagramas de la aplicación	12/10/18	17/10/18
● Prototipado gráfico	18/10/18	19/10/18
☐ ● Implementación	22/10/18	31/12/18
● Configuración y creación del proyecto	22/10/18	30/10/18
● Autenticación de usuarios	31/10/18	6/11/18
● Creación de la pantalla Lists	7/11/18	20/11/18
● Creación de la pantalla del perfil	21/11/18	30/11/18
● Creación de la pantalla Chat Rooms	3/12/18	7/12/18
● Creación de la pantalla Friend's Updates	10/12/18	12/12/18
● Implementación de filtros y rankings	13/12/18	18/12/18
● Limpieza de código y reparación de bugs	19/12/18	31/12/18
☐ ● Memoria y presentación	1/01/19	29/01/19
● Memoria	1/01/19	17/01/19
● Presentación	18/01/19	29/01/19

Ilustración 1: Distribución del tiempo

## 1.4.1 Diagrama de Gantt

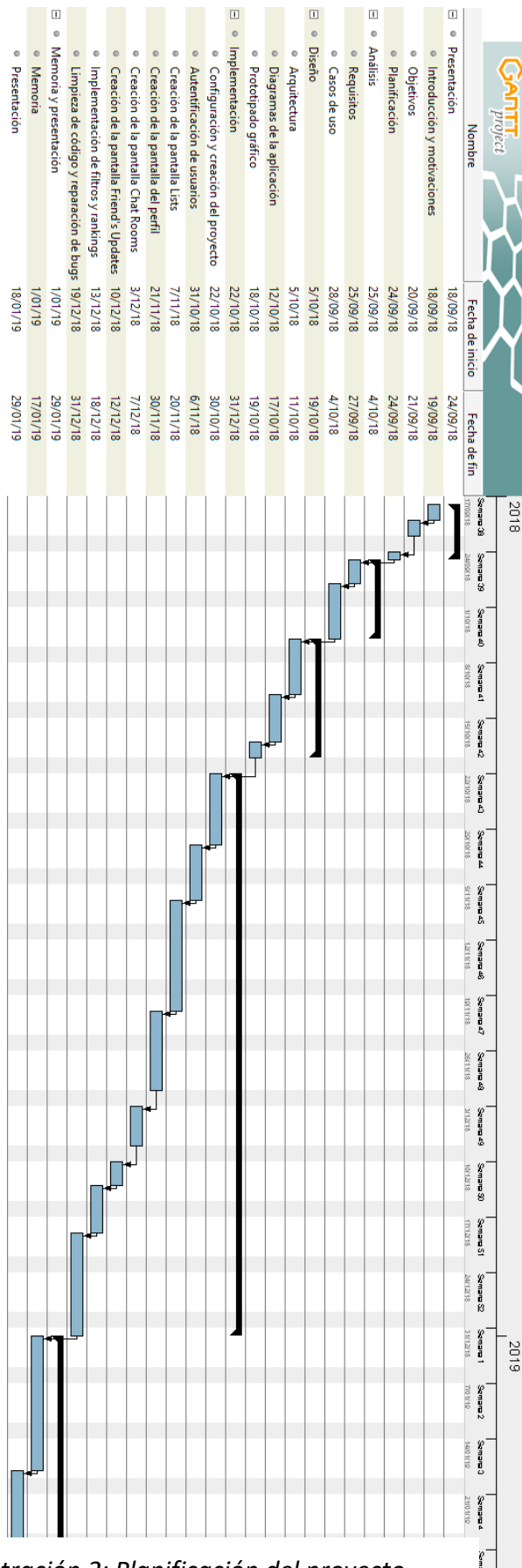


Ilustración 2: Planificación del proyecto

## 2 Análisis

### 2.1 Comparación con páginas webs similares

Una de las páginas webs más importantes y más popular hoy en día en el mundo del arte y que actúe a la vez como red social es **DeviantArt**, aún siendo tan popular esta web no ofrece mucha comunicación entre usuarios y no ofrece la posibilidad de realizar proyectos conjuntos. Por este motivo el objetivo principal de este proyecto es desarrollar un buen sistema que permita a los usuarios comunicarse entre ellos formando equipos, grupos o simplemente un chat entre dos personas. Dentro de este objetivo entra la posibilidad de crear equipos de usuarios donde tengan su propio perfil de equipo y este pueda ganar prestigio publicando obras en nombre del equipo en conjunto, esta posibilidad no se ofrece en ninguna otra web de arte, es decir siempre son cuentas de usuario separadas sin ningún tipo de alianzas entre ellas.

Otro aspecto que **DeviantArt** y casi todas las webs de arte no tienen en cuenta, es a la hora de publicar otro tipo de arte aparte de dibujos, por ejemplo, los cómics o animaciones, muy pocas webs ofrecen la posibilidad de subir tu cómic y las que lo hacen, no está muy bien gestionado, un ejemplo es **pixiv**, donde aparte de subir dibujos, también te permite subir cómics, pero colgando únicamente capítulos sueltos sin estar ordenados ni dentro de ningún cómic en concreto.

Por lo tanto, otro objetivo para este proyecto sería mejorar la subida de cómics ordenándolos mejor y organizándolos en cómics, capítulos y páginas y crear una base de datos de cómics indies con sus valoraciones y sus comentarios y también con animaciones y dibujos.

Siguiendo con **DeviantArt**, analizando los filtros se puede ver que no están muy bien separados, es decir mezclan categorías con filtros y con otros tipos de arte que no tienen nada que ver con el dibujo como la fotografía y animaciones, también una vez entramos en un filtro aparecen más filtros de este filtro, esto es muy poco intuitivo y puede llegar a ser difícil de entender. Otro aspecto por mejorar en este proyecto es hacer una lista de filtros bien pensada y ordenada que dependa de la categoría en la que estemos.

Dentro del mismo mundo del arte, tenemos la página **ArtStation**, esta página existe con el objetivo de que los artistas encuentren trabajo y entren en el mundo laboral, por lo tanto este proyecto no se ha enfocada hacia este objetivo de entrar en el mundo laboral, ya que **ArtStation** tiene muy bien organizado las ofertas de trabajo y ya están las empresas más importantes como Blizzard o Ubisoft, así que esta proyecto tendrá un enfoque más hacia una red social y a participar el trabajos entre pocas personas más que a entrar en el mundo laboral dentro de grandes empresas.

## 2.2 Requisitos iniciales

### 2.2.1 Requisitos funcionales

Los requisitos funcionales corresponden a las funcionalidades principales que ofrece la aplicación a los usuarios. Estos requisitos se han dividido según en qué pantalla de la aplicación se encuentra el usuario o del tipo de rol del usuario.

#### **Lists**

- Visualización de toda la base de datos de dibujos, cómics, animaciones, usuarios y equipos separados en diferentes vistas.
- Aplicar filtros o categorías a la lista en la que está el usuario y/o ordenar los elementos según más populares (con más visitas), con más votos o más nuevos.
- Cada tipo de publicación (dibujo, cómic y animación) debe tener diferentes filtros donde escoger.
- Se debe poder buscar un dibujo, cómic, animación, usuario o un equipo por su respectivo nombre.
- Debe haber un contador de visitas que registre cuantos usuarios han visto una publicación en concreto y mostrar el número en los detalles de esta publicación.
- Cada dibujo, animación y cómic tiene nombre, autor, descripción, puntuación, visualizaciones y comentarios.
- Un usuario puede visualizar un dibujo, cómic o animación y comentarlo y votarlo para modificar su posición en el ranking.
- Si un usuario ha subido una publicación, este no puede votar su propia publicación. Tampoco puede votar una publicación que haya subido un equipo del cual es miembro.
- Un usuario únicamente puede eliminar sus propios comentarios.
- Un usuario puede modificar la nota de un dibujo que había votado previamente
- Los cómics deben estar organizados por capítulos y estos por páginas, y el usuario puede visualizar cada página de cada capítulo a través del visor de la web.
- Las animaciones deben estar organizadas por episodios y dentro de cada episodio, abrirse un visor de videos para poder visualizar el episodio.

- Preguntar al usuario si está seguro de eliminar una publicación siempre que se clique a eliminar.

### **Teams/Team's Profile**

- Un usuario puede crear un equipo en cualquier momento sin ninguna restricción y puede añadir o eliminar a los miembros que él quiera y ascenderlos a administrador (de esta forma estos miembros pueden añadir y eliminar usuarios también)
- Dentro de cada equipo, los miembros tendrán un rol asignado por ellos que pueden editar.
- Dentro del perfil del equipo se deben mostrar sus miembros, con su rol, su nombre y su foto y diferenciándolos si son administradores o no.
- Dentro del perfil de equipo se debe mostrar también la foto del equipo, el nombre, la descripción, la fecha en la que se creó y todas las obras realizadas por el equipo.
- Un usuario puede dejar un equipo en cualquier momento, si el equipo se queda sin usuarios, el equipo queda eliminado de la base de datos.
- En el momento en que se cree un equipo se debe crear automáticamente un chat de equipo con todos los miembros en la pantalla de chat rooms
- Un usuario miembro de un equipo puede subir desde el perfil de su equipo cualquier cómic, animación, dibujo, episodio o capítulo que quiera en cualquier momento y esta publicación quedará en nombre del equipo.
- Un usuario miembro de un equipo puede eliminar desde el perfil de su equipo cualquier cómic, animación, dibujo, episodio o capítulo que quiera.

### **Login**

- El usuario tiene que poder loguearse o registrarse en cualquier pantalla de la app.
- El registro debe tener nombre, nombre de usuario, contraseña, fecha de nacimiento y email.
- El login se realiza introduciendo únicamente el nombre de usuario y la contraseña.
- El token de sesión debe caducar al mes, y el usuario tendrá que hacer login otra vez.

## **Friend's Activity**

- Cada vez que un usuario suba una publicación a su perfil, la aplicación deberá mostrar al resto de usuarios que siguen a este la publicación que ha subido dentro de esta vista. Así pues, esta vista consistirá en una lista de notificaciones de las personas que son seguidas por el actual usuario, donde se mostrarán las últimas subidas de ellos.
- Una notificación dentro de esta lista consistirá en un texto donde se especifica el tipo de publicación, la foto y el nombre del usuario autor de la notificación y el archivo que ha subido con el nombre.
- El usuario puede acceder directamente al link de una publicación a través de esta vista.
- Esta pantalla debe tener un límite de como máximo las 30 últimas publicaciones de los seguidores del usuario.

## **User's Profile**

- Un usuario puede cambiar la biografía de su perfil y su foto de usuario en cualquier momento.
- Dentro del perfil de usuario deben estar todas sus obras realizadas hasta la fecha separadas por animaciones, cómics o dibujos y ordenados según el ranking.
- Un usuario puede subir a su perfil cualquier cómic, animación, dibujo, episodio o capítulo que quiera en cualquier momento.
- Un usuario puede eliminar un dibujo, un cómic, un capítulo, una animación o un episodio siempre que lo haya subido él.
- Dentro del perfil de usuario también deben encontrarse su foto de perfil, su nombre, su descripción y la fecha en la que se unió a la aplicación.
- Dentro del perfil de usuario, también encontramos su lista de seguidores, su lista de seguidos y su lista de equipos en los que él es miembro.
- En caso de entrar en un perfil distinto al nuestro, tenemos la posibilidad de seguir o dejar de seguir a este usuario en caso de que ya lo sigamos.

## **Chat Rooms**

- Separación de la vista en la sección de chats públicos, chats privados y chats de equipo.

- Un usuario puede enviar mensajes en cualquier chat público, en sus chats de equipo y en los chats privados en los cual él sea miembro.
- Un usuario puede crear un chat público y todos los usuarios de la app pueden escribir mensajes en este chat.
- Un usuario puede crear un chat privado y solo los usuarios que él añade al chat pueden escribir mensajes.
- Solo los chats públicos y los privados pueden ser eliminados por él o los administradores de dichos chats.
- Cuando un equipo es eliminado de la base de datos, el chat de este equipo también queda eliminado.
- Establecer límite de mensajes por chat, es decir que solo se guarden los últimos 50 mensajes de cada chat para no sobrecargar la base de datos.

## **Usuarios/Roles**

### Usuario no logueado

Persona que entra en el sistema sin estar logueada, únicamente tiene acceso a la vista de listas, y dentro de esta puede realizar todos los requisitos de esta a excepción de votar y comentar publicaciones.

### Usuario logueado

Persona que se ha registrado previamente en el sistema y se loguea en la web. Tiene acceso a todas las vistas de la aplicación y las funcionalidades que ofrece cada vista.

### Usuario administrador de equipo

Persona que es líder de un equipo de gente. Tiene las mismas funcionalidades que un usuario logueado, pero con la posibilidad de eliminar o añadir miembros a su equipo y de ascenderlos a administradores.

## **2.2.2 Requisitos no funcionales**

Los requisitos no funcionales corresponden a las propiedades y características que nuestro sistema tiene que ofrecer sin entrar dentro de las funcionalidades del programa. Estos requisitos se han dividido en requisitos de interfaz y de seguridad.



## Interfaz

- La aplicación se tendrá que adaptar al tipo de pantalla que se esté usando, sin recortar palabras ni suprimir botones o texto.
- La barra de navegación deberá estar presente en todo momento, en cualquier vista o funcionalidad.
- Los filtros, la búsqueda y la forma de ordenar las listas deberá estar escondida en un side nav para no sobrecargar mucho la pantalla.

## Seguridad

- La contraseña de los usuarios deberá cumplir unos requisitos mínimos para garantizar la seguridad de la cuenta del usuario.
- Las contraseñas de los usuarios se deberán guardar de forma encriptada para que nadie pueda tener acceso a esos datos.
- Cada vez que se hagan peticiones a la API con permisos de usuario se debe confirmar tanto en Frontend como en Backend que estamos logueados y que somos el usuario adecuado para esta petición.
- Se hará validación de todos los datos en los formularios antes de editar la base de datos.

### 2.2.3 Requisitos técnicos

Los requisitos técnicos corresponden a las herramientas que se usaran para desarrollar e implementar la aplicación web. Se han dividido en la parte de Frontend y la parte de Backend.

#### Frontend

- Se usará el framework Angular 6 para poder implementar una Single Page Application.
- El lenguaje de programación Typescript y Javascript para implementar las funcionalidades básicas de la vista como por ejemplo abrir un menú y los servicios como por ejemplo llamadas a la API.

- El lenguaje HTML para implementar las diferentes vistas de la aplicación.
- El lenguaje LESS para organizar y editar los estilos de los componentes de la página. Este lenguaje se convierte en CSS al ser compilado.
- La librería Angular Material para mejorar el estilo de los componentes de la aplicación.

## Backend

- Se usará el framework Laravel.
- El lenguaje PHP para implementar el controlador de la app, todas las clases de modelo de la app y las migraciones a la base de datos.
- Se usará XAMPP para ejecutar el servidor de la base de datos.
- El gestor de base de datos será MySQL ya que viene instalado con XAMPP y la lectura de datos con este gestor es muy rápida.
- Se usará JWT token para guardar la sesión del usuario.

## Chats

- Para los chats se usará un socket.io por cada chat activo.

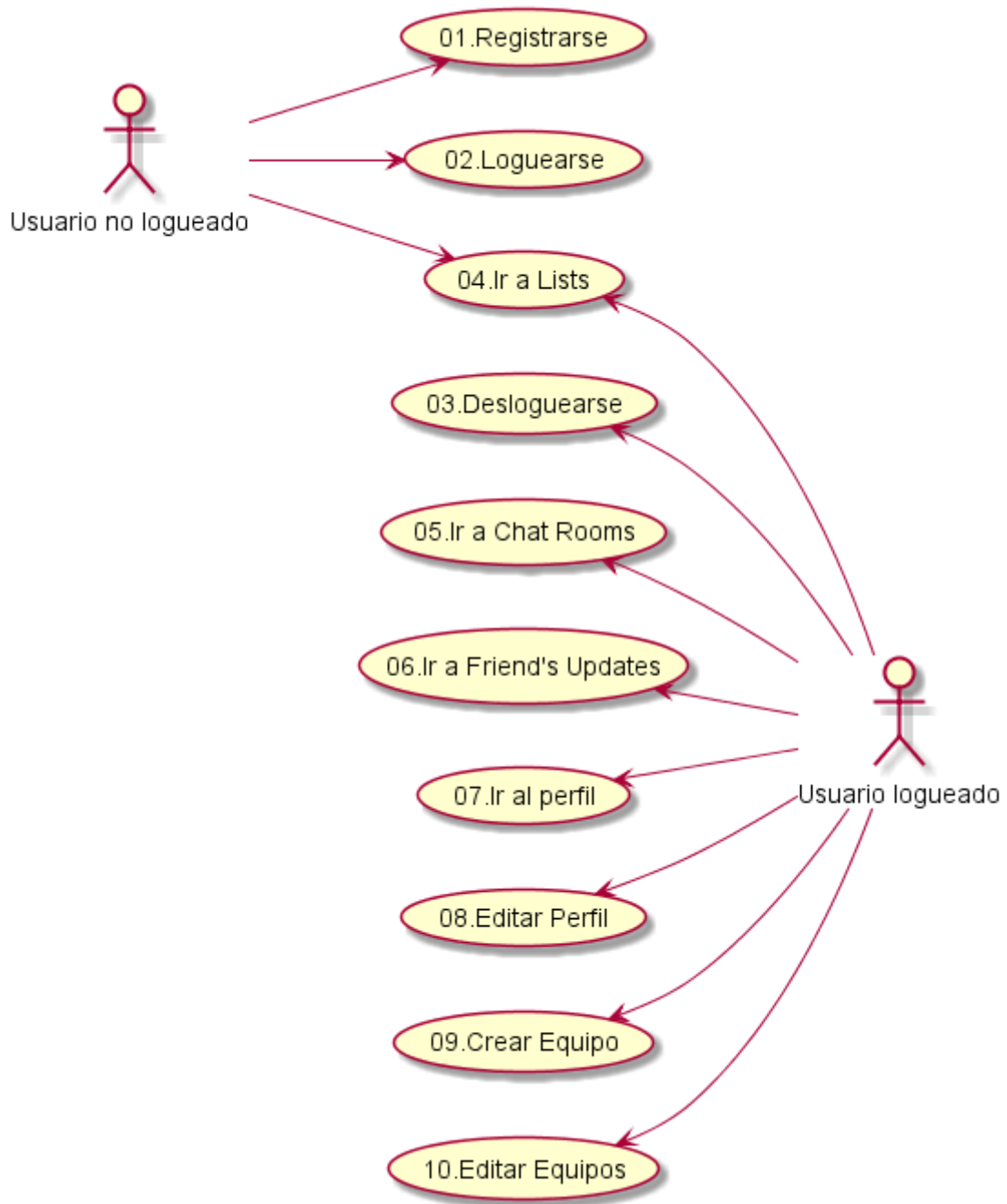
## 2.3 Casos de Uso

### 2.3.1 Diagrama de caso de uso

Se ha dividido en 5 diagramas de caso de uso diferentes, los cuatro últimos corresponden a diferentes vistas de la aplicación y el primer diagrama corresponde al TopBar Navigation, que es la barra principal de la página web, que estará siempre presente en todas las vistas de la aplicación.

### 2.3.1.1 TopBar Navigation

Los casos de uso de TopBar Navigation estarán siempre disponibles en cualquier vista de la aplicación.



*Ilustración 3: Diagrama de casos de uso (TopBar)*

### 2.3.1.2 List View

List View es la vista principal de la aplicación, están todos los dibujos, cómics, animaciones y usuarios ordenados según determinados filtros.

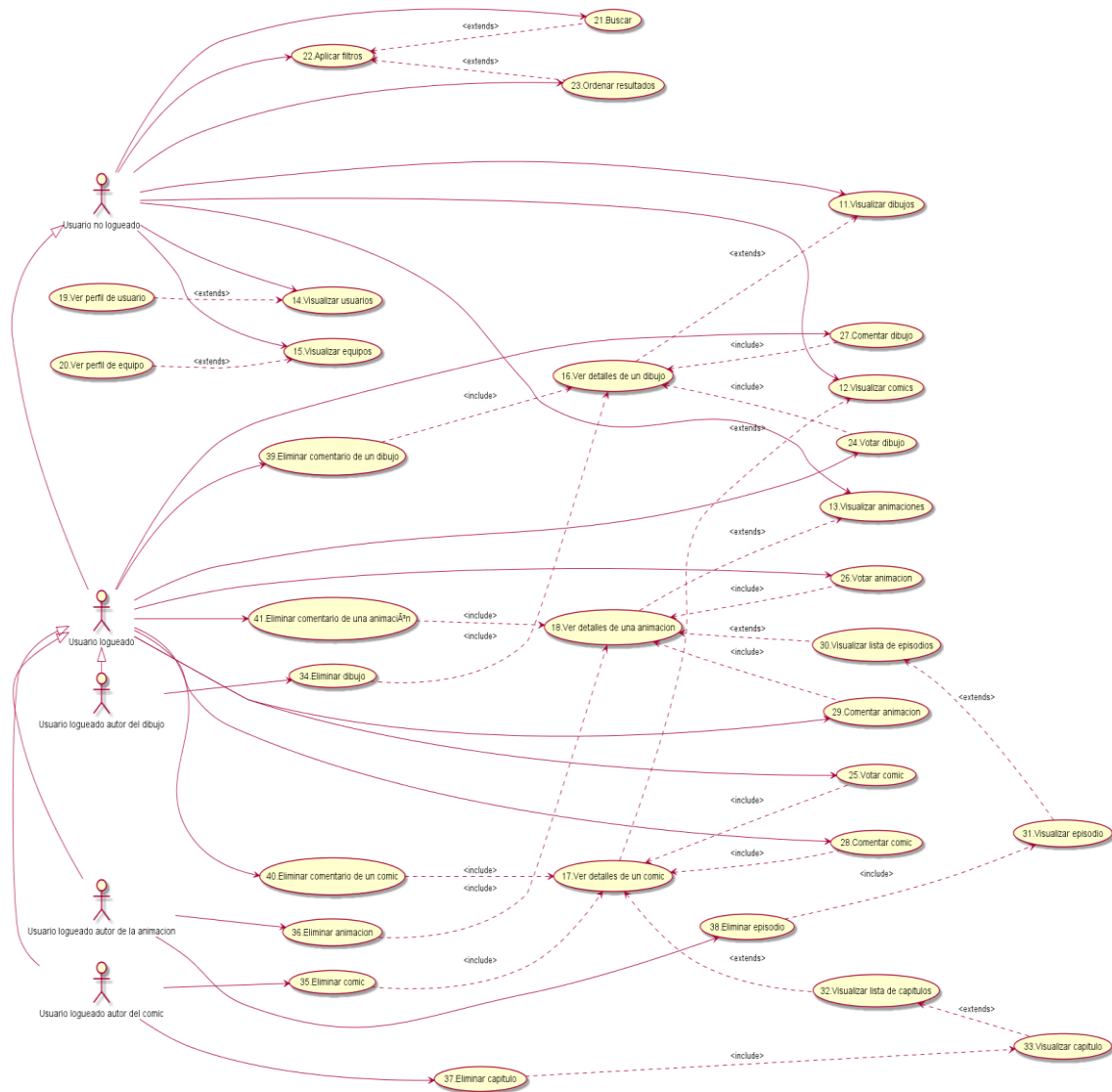


Ilustración 4: Diagrama de casos de uso (List View)

### 2.3.1.3 User's Profile/Team's Profile

Esta vista corresponde al propio perfil del usuario, al perfil de otro usuario diferente o al perfil de un equipo del cual somos miembros o no.

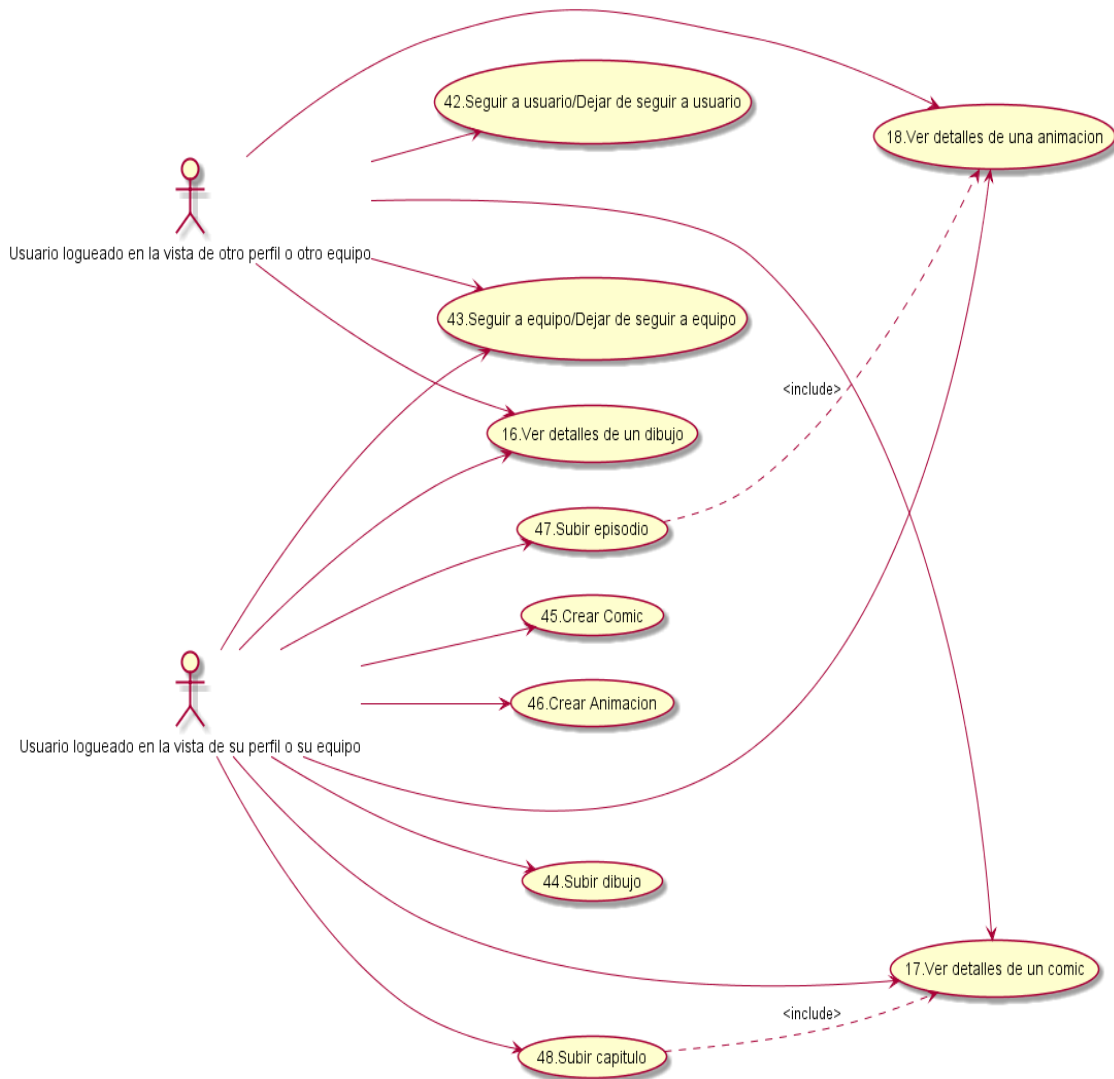


Ilustración 5: Diagrama de casos de uso (User's Profile/Team's Profile)

### 2.3.1.4 Chat Rooms View

Esta vista es el apartado de equipos y grupos de chat, donde los usuarios crean grupos o equipos y trabajan conjuntamente para un proyecto conjunto.

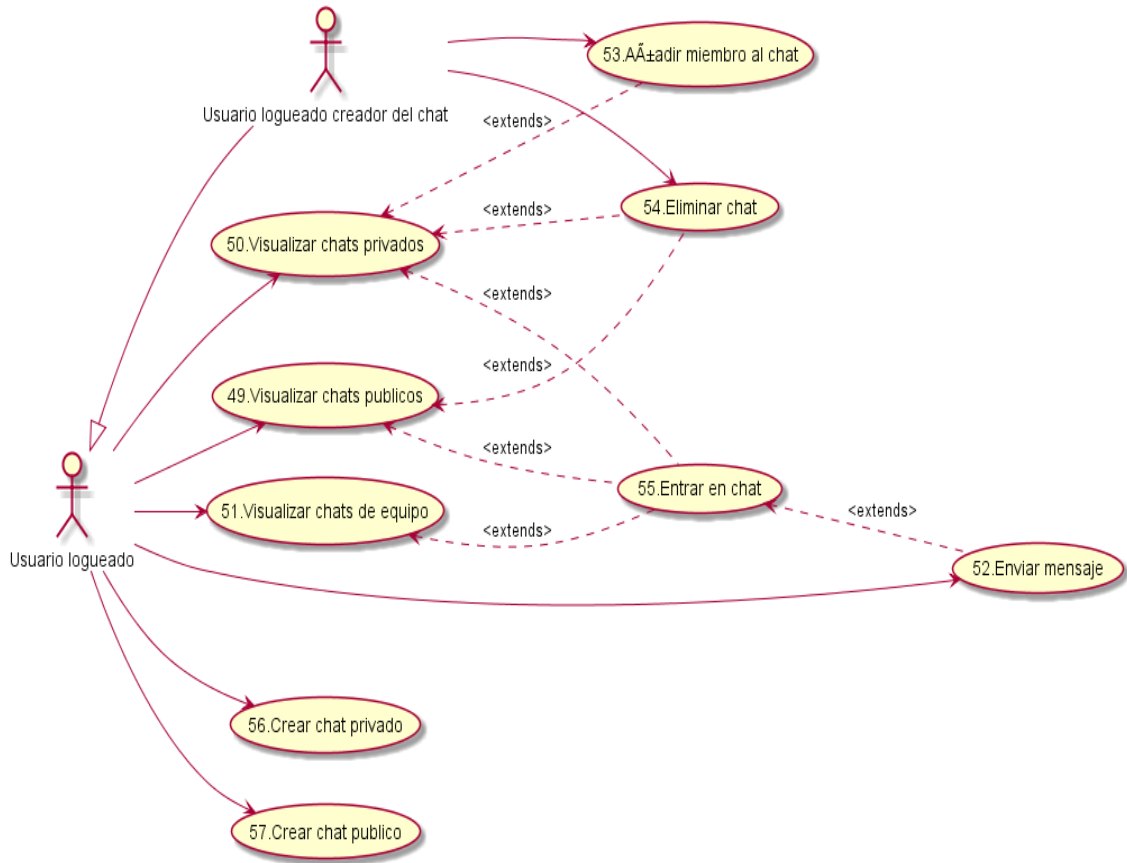


Ilustración 6: Diagrama de casos de uso (Chat Rooms View)

### 2.3.1.5 Manage Teams

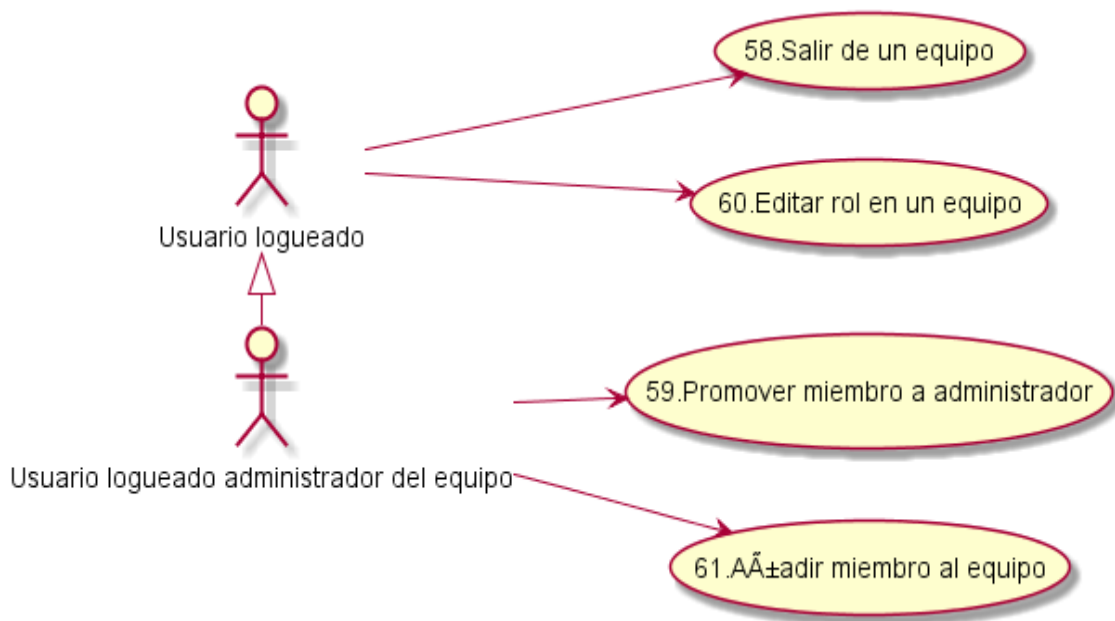


Ilustración 7: Diagrama de casos de uso (Manage Teams)

### 2.3.2 Descripción de los casos de uso textuales

Se han recogido en este apartado los casos de uso que se consideran más importantes.

La totalidad de los casos de uso se encuentra en el siguiente enlace:

<https://www.dropbox.com/s/16hdz9su170xbt5/Casos%20de%20uso%20textuales.docx?dl=0>

Descripción	UC01 – Registrarse
Actores	Usuario no logueado
Precondiciones	No estar logueado en el sistema.
Flujo básico	<ol style="list-style-type: none"> <li>1. Usuario: Selecciona Sign up en el TopBar.</li> <li>2. Sistema: Muestra el formulario de registro.</li> <li>3. Usuario: Introduce sus datos en un formulario y aprieta el botón aceptar.</li> <li>4. Sistema: Valida los datos necesarios y registra al usuario en</li> </ol>

	el sistema.
<b>Flujo alternativo</b>	4a. Sistema: Si los datos no son válidos, no se crea ninguna cuenta en el sistema.
<b>Postcondiciones</b>	El usuario queda registrado en el sistema.

<b>Descripción</b>	<b>UC02 – Loguearse</b>
<b>Actores</b>	Usuario no logueado
<b>Precondiciones</b>	No estar logueado en el sistema.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Usuario: Selecciona Log in en el TopBar.</li> <li>2. Sistema: Muestra el formulario de log in.</li> <li>3. Usuario: Introduce sus datos en un formulario y aprieta el botón aceptar.</li> <li>4. Sistema: Valida los datos necesarios crea la sesión de usuario.</li> </ol>
<b>Flujo alternativo</b>	4a. Sistema: Si los datos no son válidos, el usuario no puede entrar al sistema.
<b>Postcondiciones</b>	El usuario queda logado en el sistema.

<b>Descripción</b>	<b>UC08 – Editar Perfil</b>
<b>Actores</b>	Usuarios logueados
<b>Precondiciones</b>	-
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Usuario: Selecciona su nombre en el TopBar</li> <li>2. Sistema: muestra un conjunto de opciones.</li> <li>3. Usuario: Selecciona Edit Profile.</li> <li>4. Sistema: Muestra al usuario un formulario donde puede cambiar su foro de perfil y su descripción.</li> <li>5. Usuario: Cambia su foto de perfil y acepta.</li> <li>6. Sistema: Devuelve al usuario a la página de su perfil con la</li> </ol>



	foto cambiada.
<b>Flujo alternativo</b>	<p>5a. Usuario: Cambia su descripción y acepta.</p> <p>6a. Sistema: Devuelve al usuario a la página de su perfil con la descripción cambiada.</p> <p>5b. Usuario: Cambia su descripción y su foto de perfil y acepta.</p> <p>6b. Sistema: Devuelve al usuario a la página de su perfil con la descripción cambiada y la foto de perfil cambiada.</p>
<b>Postcondiciones</b>	El perfil queda editado con la nueva información que ha añadido el usuario.

<b>Descripción</b>	<b>UC09 – Crear Equipo</b>
<b>Actores</b>	Usuarios logueados
<b>Precondiciones</b>	-
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Usuario: Selecciona su nombre en el TopBar.</li> <li>2. Sistema: muestra un conjunto de opciones.</li> <li>3. Usuario: Selecciona Create Team.</li> <li>4. Sistema: Muestra al usuario un formulario con los campos necesarios para la creación de un equipo.</li> <li>5. Usuario: Rellena todos los datos y acepta.</li> <li>6. Sistema: Valida los datos y devuelve al usuario a la página principal con el equipo creado.</li> </ol>
<b>Flujo alternativo</b>	6a. Sistema: Si los datos no son válidos, no se crea ningún equipo en el sistema.
<b>Postcondiciones</b>	Se crea un nuevo equipo donde el usuario es el único miembro y administrador de este.

<b>Descripción</b>	<b>UC21 – Buscar</b>
<b>Actores</b>	Todos
<b>Precondiciones</b>	El usuario está en la pantalla de Lists.

<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Usuario: Abre el NavBar.</li> <li>2. Sistema: Muestra un NavBar con un conjunto de filtros, formas de ordenar la lista y una barra de búsqueda.</li> <li>3. Usuario: Introduce una letra o más en la barra de búsqueda.</li> <li>4. Sistema: Filtra los resultados de forma que contengan esa letra o letras en el nombre del dibujo, del cómic, de la animación, del usuario o del equipo (dependiendo en que pestaña esté buscando el usuario).</li> </ol>
<b>Flujo alternativo</b>	
<b>Postcondiciones</b>	La lista queda filtrada por el texto que ha introducido el usuario.

<b>Descripción</b>	<b>UC22 – Aplicar filtros</b>
<b>Actores</b>	Todos
<b>Precondiciones</b>	El usuario está en la pantalla de Lists.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Usuario: Abre el NavBar.</li> <li>2. Sistema: Muestra un NavBar con un conjunto de filtros, formas de ordenar la lista y una barra de búsqueda.</li> <li>3. Usuario: Selecciona los filtros que el desee.</li> <li>4. Sistema: Filtra los resultados de forma que el dibujo, el cómic o la animación (dependiendo en que pestaña esté buscando el usuario) tenga un tag que sea igual a uno de los filtros seleccionados.</li> </ol>
<b>Flujo alternativo</b>	
<b>Postcondiciones</b>	La lista queda filtrada por los filtros que ha seleccionado el usuario.

<b>Descripción</b>	<b>UC23 – Ordenar resultados</b>
<b>Actores</b>	Todos
<b>Precondiciones</b>	El usuario está en la pantalla de Lists.

<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Usuario: Abre el NavBar.</li> <li>2. Sistema: Muestra un NavBar con un conjunto de filtros, formas de ordenar la lista y una barra de búsqueda.</li> <li>3. Usuario: Selecciona una forma de ordenar la lista (más nuevos, más populares o mejor votados).</li> <li>4. Sistema: Ordena los resultados de la forma que ha elegido el usuario.</li> </ol>
<b>Flujo alternativo</b>	<ol style="list-style-type: none"> <li>3a. Usuario: Selecciona una forma de ordenar la lista y aplica filtros.</li> <li>4a. Sistema: Ordena los resultados de la forma que ha elegido el usuario y aplica los filtros seleccionados por el usuario.</li> </ol>
<b>Postcondiciones</b>	La lista queda ordenada de la forma que ha seleccionado el usuario.

<b>Descripción</b>	<b>UC24 – Votar dibujo</b>
<b>Actores</b>	Usuario logueado
<b>Precondiciones</b>	El usuario está en la pantalla de detalles de un dibujo.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: Muestra un combo box con la nota actual del usuario en ese dibujo.</li> <li>2. Usuario: Selecciona una nota en el combo box.</li> <li>3. Sistema: Actualiza la nota media del dibujo y guarda la nota del usuario.</li> </ol>
<b>Flujo alternativo</b>	1a. Sistema: El usuario no había votado previamente y el sistema no muestra ninguna nota en el combo box.
<b>Postcondiciones</b>	Se modifica la nota del dibujo.

<b>Descripción</b>	<b>UC27 – Comentar dibujo</b>
<b>Actores</b>	Usuario logueado
<b>Precondiciones</b>	El usuario está en la pantalla de detalles de un dibujo.

<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: Muestra los comentarios del dibujo y una región donde el usuario puede escribir su comentario.</li> <li>2. Usuario: Escribe un comentario y hace clic en aceptar.</li> <li>3. Sistema: Actualiza los comentarios del dibujo.</li> </ol>
<b>Flujo alternativo</b>	
<b>Postcondiciones</b>	El comentario del usuario se guarda en la base de datos.

<b>Descripción</b>	<b>UC31 – Visualizar episodio</b>
<b>Actores</b>	Todos
<b>Precondiciones</b>	El usuario está en la pantalla de lista de episodios.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Usuario: Hace clic en un episodio de la lista.</li> <li>2. Sistema: Abre un reproductor de video dentro de la misma pantalla.</li> <li>3. Usuario: Reproduce el video.</li> </ol>
<b>Flujo alternativo</b>	<ol style="list-style-type: none"> <li>3a. Usuario: Reproduce el video y clic a otro episodio.</li> <li>4a. Sistema: cierra el reproductor de video actual y abre uno nuevo para este episodio.</li> </ol>
<b>Postcondiciones</b>	-

<b>Descripción</b>	<b>UC33 – Visualizar capítulo</b>
<b>Actores</b>	Todos
<b>Precondiciones</b>	El usuario está en la pantalla de lista de capítulos.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Usuario: Hace clic en un capítulo de la lista.</li> <li>2. Sistema: Abre un visor en forma de modal dentro de la misma pantalla.</li> <li>3. Usuario: pasa las páginas del capítulo.</li> <li>4. Usuario: clic al botón de cerrar el capítulo.</li> </ol>

	5. Sistema: cierra el modal y muestra la lista de capítulos.
Flujo alternativo	
Postcondiciones	-

Descripción	UC34 – Eliminar dibujo
Actores	Usuario logueado autor del dibujo
Precondiciones	El usuario está en los detalles de su propio dibujo.
Flujo básico	<ol style="list-style-type: none"> <li>1. Sistema: Muestra el botón para eliminar este dibujo.</li> <li>2. Usuario: Hace clic en el botón.</li> <li>3. Sistema: Muestra un confirm box preguntando si está seguro de eliminarlo.</li> <li>4. Usuario: Acepta el confirm box.</li> <li>5. Sistema: Elimina el dibujo y devuelve al usuario a la página principal.</li> </ol>
Flujo alternativo	<ol style="list-style-type: none"> <li>4a. Usuario: Cancela el confirm box.</li> <li>5a. Sistema: Cancela la petición de eliminar el dibujo y cierra el confirm box.</li> </ol>
Postcondiciones	El dibujo queda eliminado de la base de datos, así como todos sus comentarios.

Descripción	UC39 – Eliminar comentario de un dibujo
Actores	Usuario logueado autor del comentario
Precondiciones	El usuario está en los detalles de un dibujo.
Flujo básico	<ol style="list-style-type: none"> <li>1. Sistema: Muestra la lista de comentarios del dibujo.</li> <li>2. Usuario: Hace clic en el botón de eliminar de un comentario suyo.</li> <li>3. Sistema: Elimina el comentario y actualiza la lista de</li> </ol>

	comentarios.
<b>Flujo alternativo</b>	
<b>Postcondiciones</b>	El comentario queda eliminado del dibujo.

<b>Descripción</b>	<b>UC42 – Seguir a usuario/Dejar de seguir a usuario</b>
<b>Actores</b>	Usuario logueado
<b>Precondiciones</b>	El usuario está en el perfil de otro usuario diferente.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: Valida que el usuario no siga ya al usuario del perfil actual y muestra un botón para seguir al usuario.</li> <li>2. Usuario: Hace clic al botón de seguir al usuario.</li> <li>3. Sistema: actualiza los followers y los following de ambos usuarios y cambia el botón de seguir por dejar de seguir.</li> </ol>
<b>Flujo alternativo</b>	1a. Sistema: el usuario ya sigue a este usuario del perfil, así que, en vez de mostrar el botón de seguir, muestra el botón de dejar de seguir.
<b>Postcondiciones</b>	El usuario podrá ver las últimas publicaciones del usuario al que sigue en friend's updates.

<b>Descripción</b>	<b>UC43 – Seguir a equipo/Dejar de seguir a equipo</b>
<b>Actores</b>	Usuario logueado
<b>Precondiciones</b>	El usuario está en el perfil de un equipo.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: Valida que el usuario no siga ya al equipo del perfil actual y muestra un botón para seguir al equipo.</li> <li>2. Usuario: Hace clic al botón de seguir al equipo.</li> <li>3. Sistema: actualiza los followers y los following del usuario y el equipo y cambia el botón de seguir por dejar de seguir.</li> </ol>
<b>Flujo alternativo</b>	1a. Sistema: el usuario ya sigue a este equipo del perfil, así que, en vez de mostrar el botón de seguir, muestra el botón de dejar de seguir.

<b>Postcondiciones</b>	El usuario podrá ver las ultimas publicaciones del equipo al que sigue en friend's updates.
------------------------	---

<b>Descripción</b>	<b>UC44 – Subir dibujo</b>
<b>Actores</b>	Usuario logueado
<b>Precondiciones</b>	El usuario está en su propio perfil.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: Muestra un botón en la galería que permite añadir dibujos al usuario.</li> <li>2. Usuario: Hace clic en este botón.</li> <li>3. Sistema: Muestra un formulario al usuario.</li> <li>4. Usuario: Rellena el formulario introduciendo el nombre, la descripción, el archivo png o jpg y selecciona los tags del dibujo.</li> <li>5. Sistema: Valida los datos introducidos y crea el dibujo.</li> </ol>
<b>Flujo alternativo</b>	5a. Sistema: Los datos introducidos por el usuario no son correctos y el sistema no crea el dibujo.
<b>Postcondiciones</b>	Se crea el dibujo.

<b>Descripción</b>	<b>UC52 – Enviar mensaje</b>
<b>Actores</b>	Usuario logueado
<b>Precondiciones</b>	El usuario está en la pantalla de Chat Rooms.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Usuario: Selecciona un chat.</li> <li>2. Sistema: muestra los últimos 50 mensajes de ese chat y un espacio donde el usuario puede redactar su mensaje.</li> <li>3. Usuario: Escribe un mensaje y clic en enviar.</li> <li>4. Sistema: Envía el mensaje al chat.</li> </ol>
<b>Flujo alternativo</b>	4a. Sistema: Si ya hay 50 mensajes en el chat, se envía el mensaje y

	se elimina el mensaje más antiguo.
<b>Postcondiciones</b>	El mensaje llega a todos los miembros del chat.

<b>Descripción</b>	<b>UC53 – Añadir miembro al chat</b>
<b>Actores</b>	Usuario logueado y con chats privados
<b>Precondiciones</b>	El usuario está en la pantalla de Chat Rooms, en la pestaña de Private Chats.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: muestra en cada chat un input text y un botón para añadir miembros.</li> <li>2. Usuario: Escribe el nombre de usuario de la persona que quiera añadir y clic al botón.</li> <li>3. Sistema: Añade al usuario al chat.</li> </ol>
<b>Flujo alternativo</b>	3a. Sistema: No encuentra al usuario porque no existe y no añade a nadie.
<b>Postcondiciones</b>	Un nuevo usuario queda añadido al chat.

<b>Descripción</b>	<b>UC54 – Eliminar chat</b>
<b>Actores</b>	Usuario logueado y con chats privados o públicos creados por él
<b>Precondiciones</b>	El usuario está en la pantalla de Chat Rooms, en la pestaña de Private Chats o Public Chats.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: muestra un botón de eliminar chat en cada chat privado o público donde el usuario sea el creador.</li> <li>2. Usuario: Hace clic al botón de eliminar chat.</li> <li>3. Sistema: Elimina el chat.</li> </ol>
<b>Flujo alternativo</b>	
<b>Postcondiciones</b>	El chat y todos sus mensajes queda eliminado y nadie tiene acceso a él.



Descripción	UC56 – Crear chat privado
Actores	Usuario logueado
Precondiciones	El usuario está en la pantalla de Chat Rooms, en la pestaña de Private Chats.
Flujo básico	<ol style="list-style-type: none"> <li>1. Sistema: Muestra un botón al final de la lista de chats, para añadir un nuevo chat privado.</li> <li>2. Usuario: Hace clic en el botón.</li> <li>3. Sistema: Muestra un formulario de creación del chat.</li> <li>4. Usuario: Rellena el formulario introduciendo el nombre y la descripción del chat.</li> </ol> <ol style="list-style-type: none"> <li>1. Sistema: Valida los datos y crea el chat privado.</li> </ol>
Flujo alternativo	
Postcondiciones	Se crea un nuevo chat privado con un único participante.

Descripción	UC58 – Salir de un equipo
Actores	Usuario logueado que es miembro de al menos un equipo
Precondiciones	El usuario está en la pantalla de Manage Teams.
Flujo básico	<ol style="list-style-type: none"> <li>1. Sistema: Muestra todos los equipos del usuario con todos sus miembros.</li> <li>2. Usuario: Hace clic en salir del equipo.</li> <li>3. Sistema: Elimina al usuario de equipo y actualiza la lista de equipos.</li> </ol>
Flujo alternativo	3a. Sistema: Elimina al usuario del equipo y si el equipo se queda sin miembros, elimina el equipo de la base de datos.
Postcondiciones	El usuario ya no tendrá acceso al equipo en manage teams ni a publicar en nombre del equipo.

Descripción	UC59 – Promover miembro a administrador
-------------	---

<b>Actores</b>	Usuario logueado administrador de un equipo
<b>Precondiciones</b>	El usuario está en la pantalla de Manage Teams.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: Muestra todos los equipos del usuario con todos sus miembros.</li> <li>2. Usuario: Hace clic en promover a administrador a un miembro de un equipo donde él sea administrador.</li> <li>3. Sistema: Pasa este miembro a administrador y actualiza la información de la página.</li> </ol>
<b>Flujo alternativo</b>	
<b>Postcondiciones</b>	El usuario promovido ahora podrá promover a otros miembros y añadir a otros miembros.

<b>Descripción</b>	<b>UC60 – Editar rol en un equipo</b>
<b>Actores</b>	Usuario logueado que es miembro de al menos un equipo
<b>Precondiciones</b>	El usuario está en la pantalla de Manage Teams.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: Muestra todos los equipos del usuario con todos sus miembros.</li> <li>2. Usuario: Escribe su rol y hace clic en editar rol en un equipo.</li> <li>3. Sistema: Cambia el rol del usuario en ese equipo y actualiza los datos.</li> </ol>
<b>Flujo alternativo</b>	
<b>Postcondiciones</b>	-

<b>Descripción</b>	<b>UC61 – Añadir miembro al equipo</b>
<b>Actores</b>	Usuario logueado administrador de un equipo
<b>Precondiciones</b>	El usuario está en la pantalla de Manage Teams.
<b>Flujo básico</b>	<ol style="list-style-type: none"> <li>1. Sistema: Muestra todos los equipos del usuario con todos</li> </ol>

	<p>sus miembros.</p> <ol style="list-style-type: none"> <li>2. Usuario: Escribe un nombre de usuario y hace clic en añadir miembro a un equipo.</li> <li>3. Sistema: Añade este usuario al equipo como miembro.</li> </ol>
Flujo alternativo	<ol style="list-style-type: none"> <li>3a. Sistema: No encuentra al usuario porque no existe y no añade a nadie.</li> </ol>
Postcondiciones	Un nuevo usuario queda añadido al equipo.

## 3 Diseño

### 3.1 Arquitectura

No se ha podido seguir una arquitectura puramente MVC, ya que Angular sigue una arquitectura basada en componentes (CBSE). A pesar de esto, se ha diseñado una arquitectura que permita tener las mismas propiedades que una arquitectura MVC de la siguiente manera:

#### 3.1.1 Controladores y Servicios

Los controladores serán clases con un conjunto de métodos que servirán para recoger los datos del modelo, editarlos y guardarlos y devolver el resultado a los componentes de la vista que han lanzado la request.

Se usará PHP con Laravel para programar estos controladores y estarán localizados dentro de App/Http/Controllers. Estos controladores estarán separados según el modelo de datos que queremos editar.

Para ejecutar los métodos del controlador se programará una API routing que contenga todas las rutas de la API para poder enrutar todas las peticiones HTTP que envíen los Servicios y ejecutar el método del controlador indicado a esa petición.

Los Servicios estarán programados con TypeScript en Angular en la parte de Frontend y se usarán desde los componentes de la vista, donde se realizarán las llamadas a la API gracias al API routing definido previamente y de estas llamadas, recibirán un json y un código con un mensaje de error concreto en caso de que la petición haya salido mal.

### 3.1.2 Modelo

El Modelo de la web consistirá en el conjunto de clases que se mapearan desde la base de datos. Estará definido en PHP con Laravel en la zona de Backend. Cada tabla de la base de datos corresponde a un modelo que se guardan como clases en PHP a excepción de las tablas pivot que solo guardan relaciones muchos a muchos.

Dentro del modelo se usará **Eloquent ORM** para mapear los datos de la base de datos al modelo ya que Laravel incluye este sistema de mapeo automáticamente y permite unificar los datos de programación de objetos con una base de datos relacional.

### 3.1.3 Vista

La vista se programará con Angular y TypeScript en la parte de Frontend y estará formada por componentes que controlan una zona de espacio de la pantalla como un toolBar, una lista, un formulario, etc. Cada componente estará compuesto por un fichero Typescript que se encargará de la lógica de dicho componente, un fichero HTML que mostrará el componente y un fichero LESS para situar y decorar el componente. Estos Componentes usaran los Servicios explicados anteriormente para ejecutar los métodos de los controladores definidos en Laravel.

### 3.1.4 Diagrama completo de la arquitectura

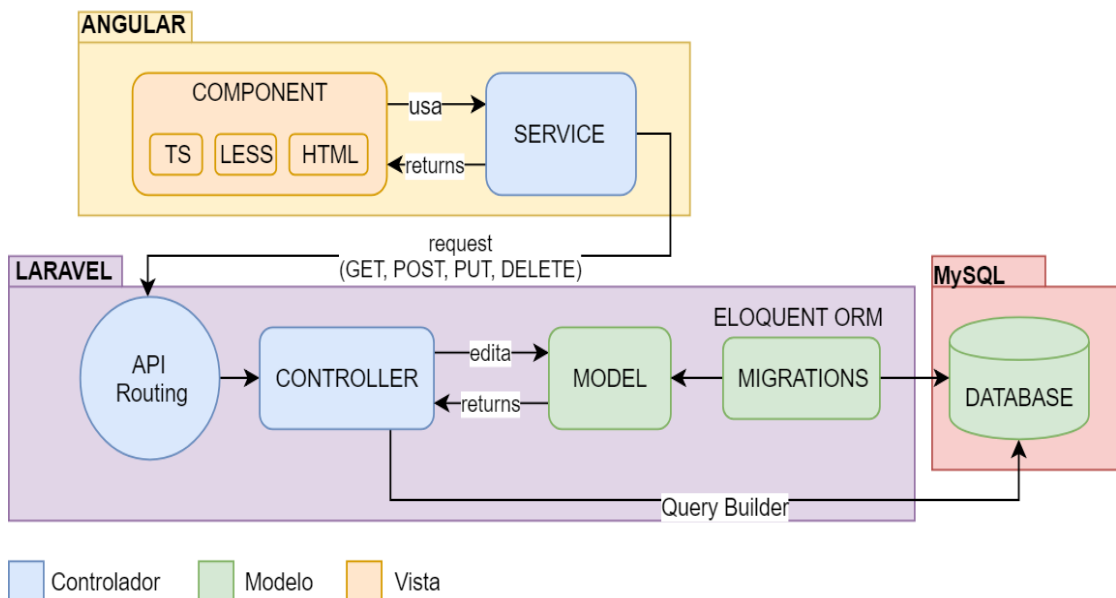


Ilustración 8: Diagrama completo de la arquitectura

## 3.2 Diagramas de la aplicación

### 3.2.1 Diagrama de la base de datos

Se usará una base de datos relacional con MySQL en la cual las tablas de esta base de datos estarán mapeadas en las clases del modelo a excepción de las tablas pivot, que su función es gestionar las relaciones Many to Many. Con tal de gestionar las herencias, se ha puesto la misma primary key al padre y a los hijos para luego recoger toda la información del objeto con un inner join en el controlador.

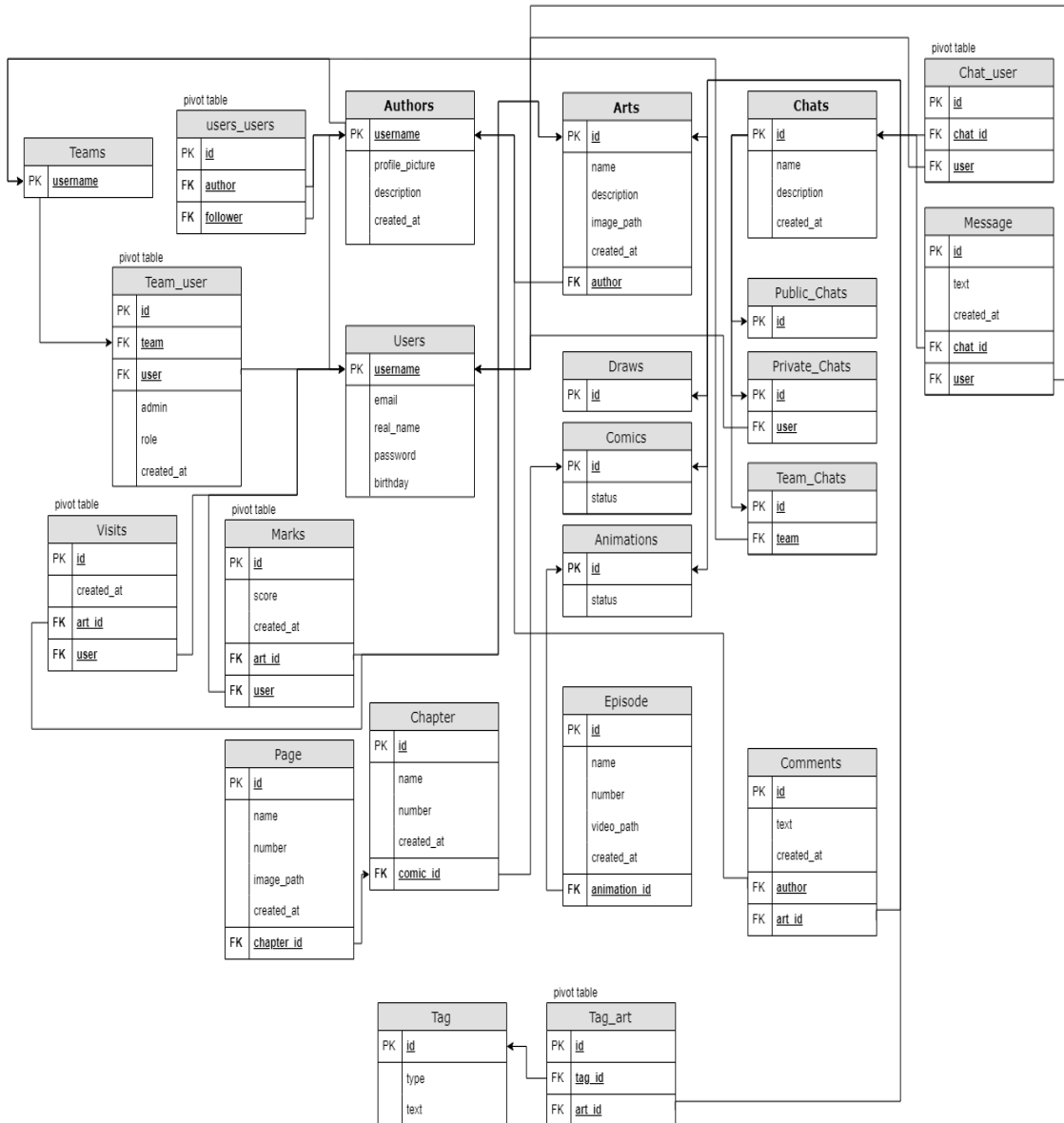


Ilustración 9: Diagrama de la base de datos

### 3.2.2 Diagrama de las clases del Modelo

Se crearán las clases del modelo necesarias para mapear todas las tablas de la base de datos y las relaciones muchos a muchos y muchos a uno representadas con el símbolo de agregación en el diagrama.

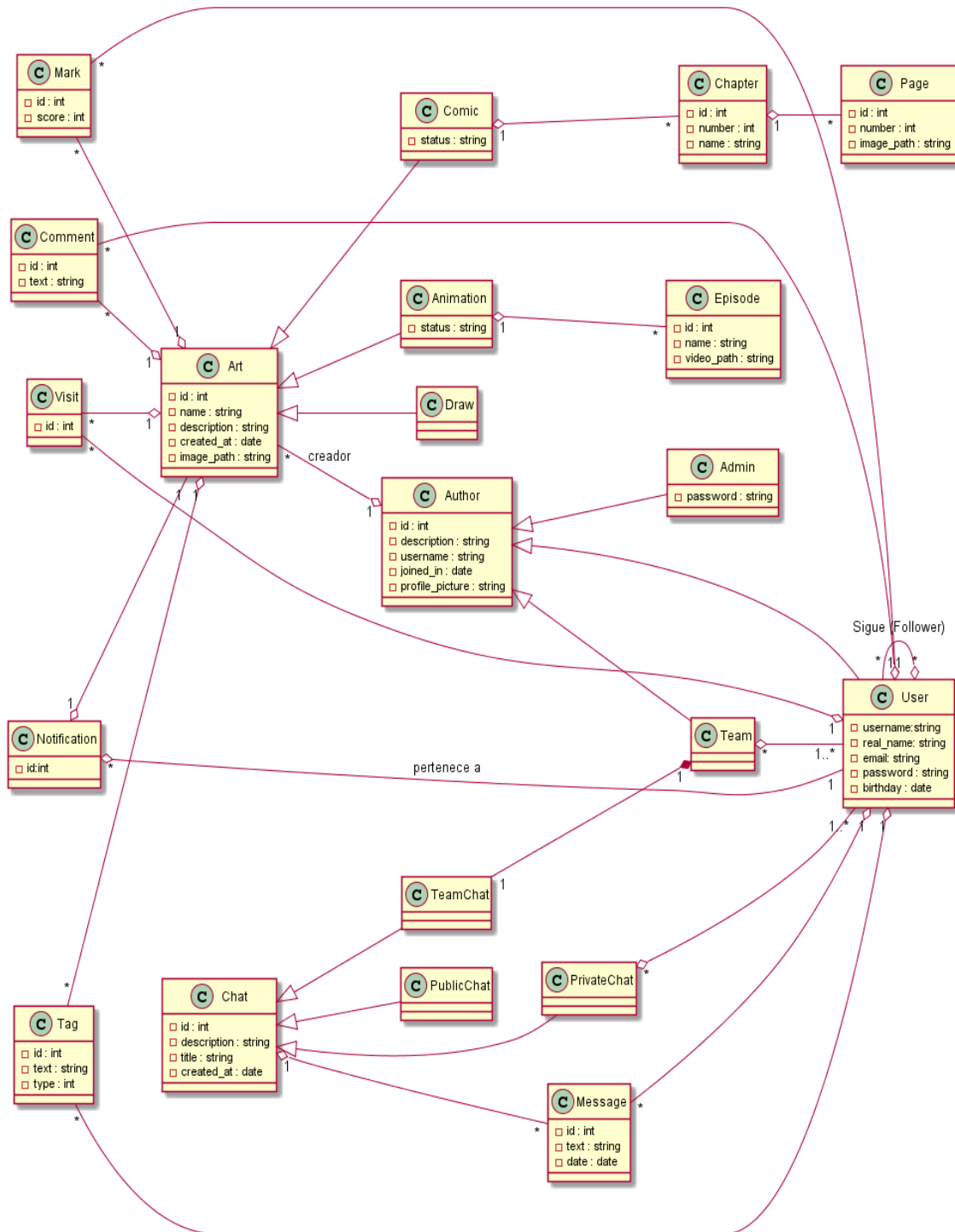


Ilustración 10: Diagrama de las clases del modelo

### 3.3 Componentes de la Vista

En la parte de la vista habrá una serie de componentes con propia arquitectura. Estos componentes se relacionarán entre ellos a través de herencias entre componentes, o servicios entre dos o más componentes.

Por ejemplo, el componente de filtros de búsqueda necesita saber en todo momento que tipo de lista se está visualizando, este problema se puede solucionar haciendo que filtros sea hijo de lista o añadiendo un servicio que gestione este problema entre estos dos componentes.

El otro tipo de relación entre componentes se usará para un componente que tenga dentro más componentes. La mayoría de las componentes con relación padre-hijo de la web seguirán la siguiente arquitectura: Un componente contenedor del componente padre, el componente padre y dentro de este los componentes hijos.

### 3.3 Pantallas

Dentro de la web, se pueden diferenciar diferentes tipos de pantalla, cada una de estas pantallas tiene una ruta concreta diferente al resto de pantallas. Estas pantallas estarán formadas por diferentes componentes necesarios para poder llevar a cabo las funcionalidades de esa pantalla.

Las funcionalidades de registro e inicio de sesión estarán activas en todas las pantallas de la web, sin necesidad de tener que moverse a otra pantalla diferente.

Nombre	Descripción y funcionalidades	Usuarios que pueden acceder
Login	Corresponde al Login de la web, que se puede acceder desde el ToolBar en cualquier pantalla de la web. El formulario del login consistirá en dos campos: nombre de usuario y contraseña.	Todos
Register	También accesible desde cualquier pantalla. Los campos de este formulario serán: nombre de usuario, nombre real, fecha de nacimiento, contraseña y email.	Todos

A continuación, se muestra una tabla de las diferentes pantallas de la web, con su nombre, la ruta para acceder a la pantalla, la descripción de esta, con todas sus funcionalidades y los usuarios que pueden acceder a esta pantalla.

Nombre	Ruta	Descripción y funcionalidades	Usuarios que pueden acceder
Lists	/	<p>Pantalla principal de la web, donde se mostrará la lista de dibujos, cómics, animaciones, usuarios y equipos.</p> <p>Dentro de esta pantalla, habrá un botón que desplegará otro componente de navegación a la derecha de la pantalla, este nos permitirá buscar un determinado elemento de la lista, aplicar filtros a la búsqueda activa (estos filtros dependerán del tipo de lista que estemos visualizando) y/u ordenar la lista de una forma determinada.</p> <p>A través de esta lista, se podrá entrar en los detalles de un elemento de esta.</p> <p>Otra funcionalidad de esta pantalla son los bloques de texto con parte de información de un elemento en concreto de la lista que aparece únicamente al pasar el ratón por encima de este.</p>	Todos
Chat Rooms	/chat-rooms	<p>Pantalla que recogerá toda la lista de chats. Podemos movernos entre chats públicos, chats privados y chats de equipo.</p> <p>También se tendrá la funcionalidad de crear chats públicos, de crear chats privados y de añadir miembros a chats privados.</p>	Usuarios logueados
Chat	/chat-rooms/{id}	<p>Esta pantalla mostrará un chat en concreto del chat rooms.</p> <p>Dentro del chat, se visualizarán los mensajes de este y se podrán enviar mensajes.</p> <p>A través de esta lista de chats, se podrá entrar directamente en un chat en concreto.</p>	Usuarios logueados



Friend's Activity	/friends-activity	<p>Esta pantalla únicamente mostrará en forma de notificaciones los últimos dibujos, episodios o capítulos que han colgado los usuarios a los que sigue el usuario logueado en este momento.</p> <p>Estas notificaciones están compuestas por: nombre de usuario, foto de usuario, nombre de la publicación, descripción de la notificación, fecha de creación y la imagen en caso de que sea un dibujo.</p> <p>A través de estas notificaciones se podrá acceder a la pantalla de los detalles de dicha publicación o al perfil del usuario.</p>	Usuarios logueados que siguen a otros usuarios
Draw details	/draw/{id}	<p>En esta pantalla se mostrarán todos los detalles de un dibujo en concreto, la media de las puntuaciones de este, el total de visualizaciones que ha recibido y la lista de comentarios del dibujo.</p> <p>Las funcionalidades de esta pantalla dependen del tipo de usuario:</p> <p>Si el usuario está logueado, puede votar dicha publicación y comentar en esta o eliminar sus propios comentarios.</p> <p>También existirá la posibilidad de eliminar el dibujo dentro de esta pantalla, pero únicamente si el usuario logueado es el autor de el dibujo o si pertenece al equipo que ha realizado el dibujo.</p>	Todos
Cómic details	/cómic/{id}	<p>Igual que la pantalla de draw details pero con los detalles de un cómic en concreto y con la funcionalidad extra de poder entrar en la lista de capítulos</p>	Todos
Animation details	/animation /{id}	<p>Igual que la pantalla de draw details pero con los detalles de un cómic en concreto y con la funcionalidad extra de poder entrar en la lista de episodios</p>	Todos
Chapters	/cómic/{id}	<p>Esta pantalla mostrará la lista de capítulos que tiene un cómic en</p>	

	/chapters	<p>concreto.</p> <p>Dentro de esta pantalla, se podrán visualizar cada página de un capítulo determinado a través de un visor.</p> <p>En caso de que el usuario sea autor de este cómic, se tendrá la funcionalidad de crear capítulos dentro de este, añadiendo paginas al nuevo capítulo y dándole un nombre y un número.</p>	
Episodes	/animation/{id}/episodes	<p>Igual que la pantalla de Chapters pero con animaciones y en vez de tener un visor de capítulos, se tendrá un visor de videos por cada episodio.</p> <p>Este visor se abrirá dentro de la misma pantalla al visualizar un episodio y se cerrará si se abre otro episodio.</p>	Todos
Team Creation	/create-team	<p>Esta pantalla mostrará un formulario con todos los campos necesarios para la creación de un equipo: Nombre, foto, descripción y rol en el equipo.</p> <p>Solamente dejará crear el equipo si todos los datos son correctos y el nombre de equipo no existe en la base de datos.</p>	Usuarios logueados
Manage Teams	/manage-teams	<p>En esta pantalla se mostrará la lista de equipos en los que el usuario logueado es miembro o administrador de estos.</p> <p>Las funcionalidades de esta pantalla en un equipo concreto dependerán de si el usuario es administrador de este equipo o no.</p> <p>El usuario podrá editar su propio rol o dejar un equipo siempre que sea miembro de este equipo, pero únicamente podrá eliminar miembros, añadir miembros y promover usuarios a administrador si él es un administrador de este equipo.</p>	Usuarios logueados que son miembros de algún equipo.

User Profile	/user/  {username}	<p>Esta pantalla mostrará toda la información del usuario, así como su lista de seguidores, su lista de seguidos y su lista de equipos en los cuales él es miembro. También se mostrará en la misma pantalla, una galería con los dibujos, cómics y animaciones subidos por este.</p> <p>En caso de que el usuario logueado corresponda con el perfil de esta pantalla, existirá la funcionalidad de poder subir un dibujo, un cómic o una animación dentro de esta misma pantalla, a través de un pop-up que muestra el formulario de creación de dicha publicación, donde indicamos el nombre, el archivo, la descripción y los tags (los tags sirven para poder buscar por filtros).</p> <p>Si no corresponde a nuestro perfil, pero el usuario está logueado en el sistema, aparecerá la opción de seguir a este usuario o dejar de seguir en caso de que ya sigamos a este.</p>	Todos
Team Profile	/team/  {username}	<p>Esta pantalla mostrará toda la información del equipo, así como su lista de seguidores y su lista de miembros, indicando el rol de cada miembro, el nombre, su foto y si son administradores o no. También se mostrará en la misma pantalla, una galería con los dibujos, cómics y animaciones subidos por el equipo.</p> <p>En caso de que el usuario logueado sea miembro del equipo de esta pantalla, existirá la funcionalidad de poder subir un dibujo, un cómic o una animación dentro de esta misma pantalla, a través de un pop-up que muestra el formulario de creación de dicha publicación, donde indicamos el nombre, el archivo, la descripción y los tags (los tags sirven</p>	Todos

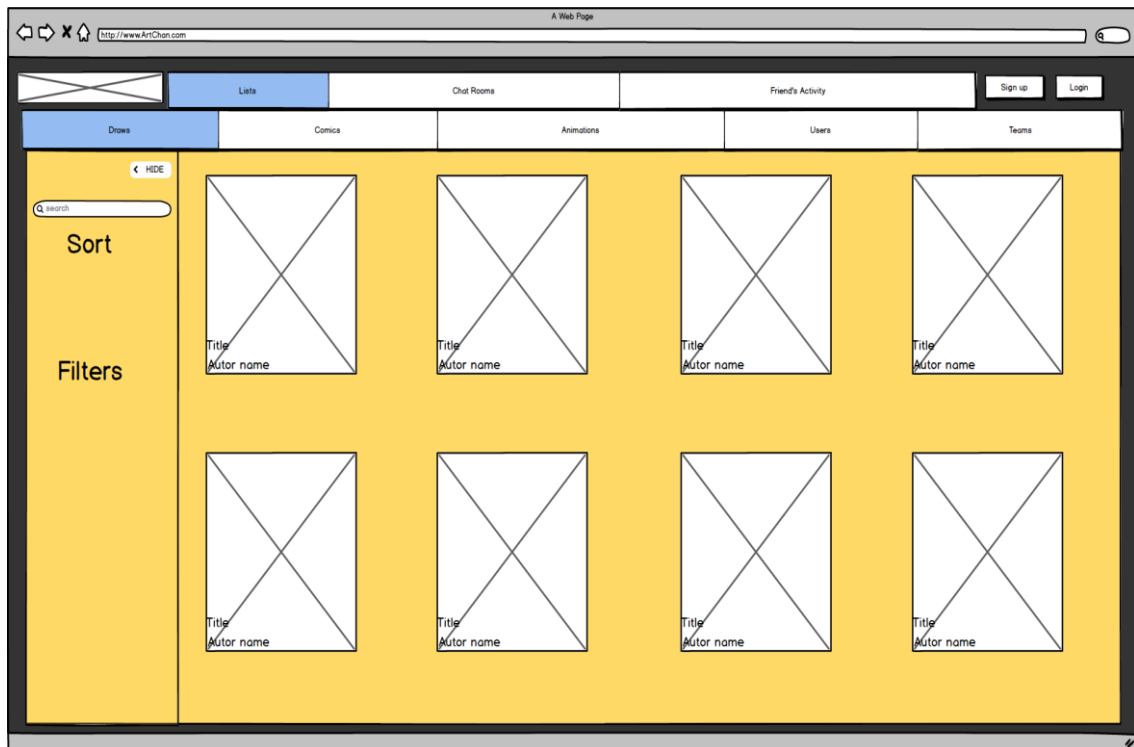
		para poder buscar por filtros).  También existirá la opción de seguir a este equipo o dejar de seguir en caso de que ya sigamos a este, aunque seamos miembros del equipo.	
Edit Profile	/edit	Esta pantalla consistirá en un formulario que permitirá al usuario poder editar su descripción y su foto de perfil.	Usuarios logueados

### 3.4 Prototipado gráfico

En este apartado se muestran los prototipos de las pantallas principales de la web, realizados con la herramienta **Balsamiq Mockups 3**. Estos prototipos no son los definitivos de la app y el diseño de la web final podría cambiar en algunos aspectos o detalles sobre todo gráficos. Lo importante en estos prototipos es representar en un único dibujo las funcionalidades que se ofrecen en una pantalla concreta de la web.

Así pues, a través de estos prototipos, podremos obtener un feedback mucho más preciso respecto a nuestra web y poder tomar las decisiones adecuadas si se tienen que cambiar ciertos aspectos.

A continuación, se mostrarán los prototipos realizados y por qué se han tomado estas decisiones de diseño en cada pantalla.

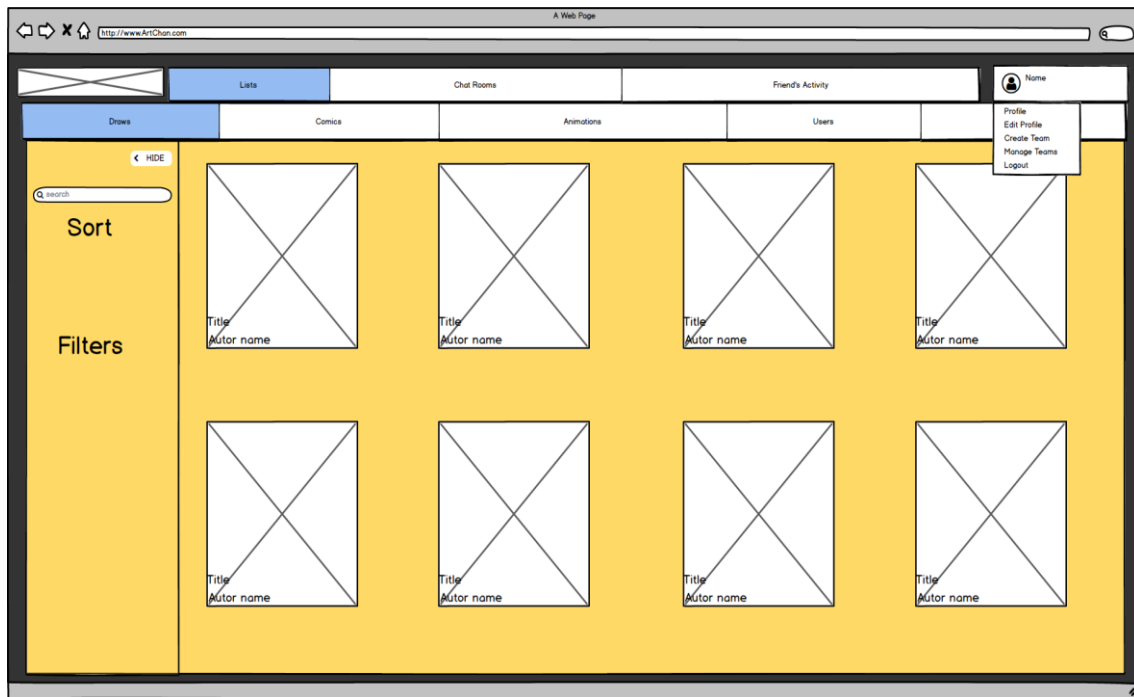


*Ilustración 11: Prototipo de la pantalla principal deslogueado*

En la figura 11 se muestra la pantalla principal de la aplicación. Se implementará un Top Bar que esté siempre presente en todas las pantallas de la web para poder acceder rápidamente a las principales funcionalidades de la web. Dentro de la pestaña de Lists habrá otro top bar más que permita cambiar el tipo de lista al usuario, este top bar estará siempre presente mientras no cambiemos de pantalla.

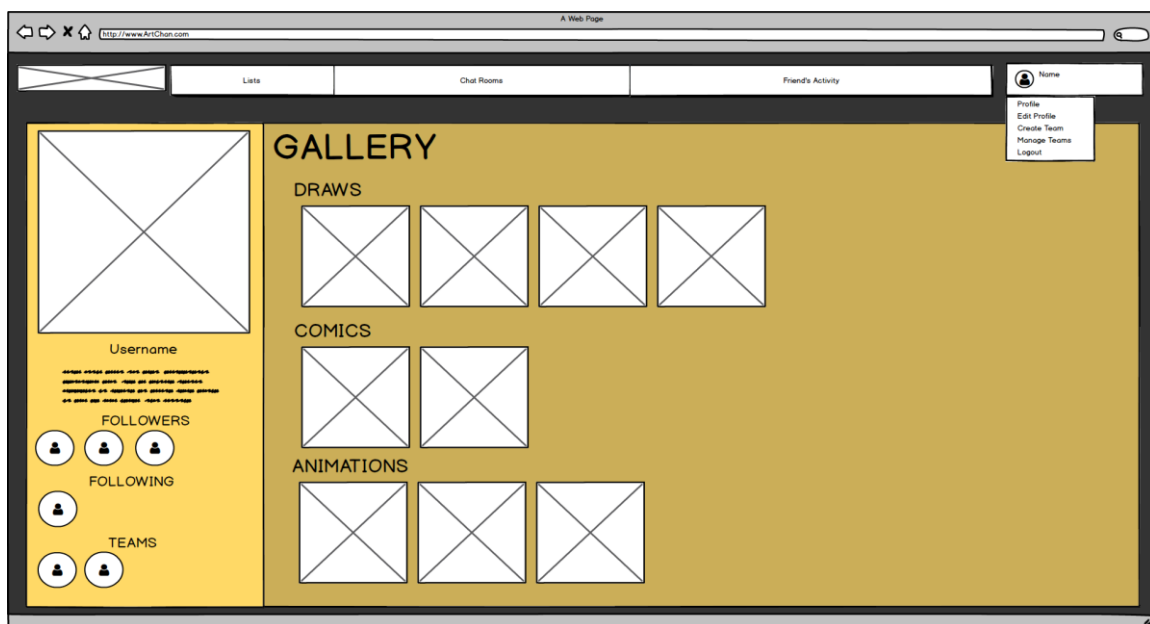
Los elementos de la lista aparecerán en forma de imágenes con un extracto de información sobre este elemento en las esquinas de este, para no sobrecargar la imagen de texto.

Los filtros, las formas de ordenar y la barra de búsqueda aparecerán ocultas por defecto y se podrán ocultar en cualquier momento una vez abiertas, de esta forma se permite que el usuario pueda ver más elementos sin tener que hacer tanto scroll y no se sobrecargará tanto la vista.



*Ilustración 12: Prototipo de la pantalla principal logueado*

Es la misma pantalla que la figura anterior, pero en este caso estamos logueados dentro de la web. La única diferencia en relación al aspecto es el Top Bar, donde en vez de aparecer los botones para loguearse o registrarse, aparecerá un combo box con diferentes opciones relacionadas con nuestra cuenta o nuestro perfil. Este combo box estará formado por la imagen de perfil del usuario y el nombre de este.



*Ilustración 13: Prototipo del perfil de un usuario*

En la figura 13, se muestra cómo quedaría el perfil de un usuario. Podemos separar esta pantalla en dos zonas claramente identificables:

La zona de la izquierda representará toda la información referente al usuario, así como su foto, su nombre, su email, su descripción y su nombre real. Dentro de esta zona también encontramos tres pequeñas listas, una de seguidores, una de seguidos y otra de equipos. Se han representado en forma de imágenes pequeñas cada elemento de la lista para así ahorrar más espacio y hacerlo más visual y más fácil de reconocer a cada elemento.

La zona de la derecha representará la galería del usuario, en esta zona estarán todas las publicaciones realizadas por el usuario, separadas por dibujos, cómics y animaciones. Los elementos de estas listas se representarán igual que en la página principal.

En caso de que estemos en nuestro propio perfil, al final de cada lista de la galería tendremos un botón que abrirá un pop up con un formulario para añadir un dibujo, un cómic o una animación, se ha decidido así ya que es más intuitivo y más rápido, ya que están los tres botones en la misma pantalla y el pop up se abre y se cierra en la misma pantalla.

En caso de que no estemos en nuestro propio perfil, tendremos un botón extra, que nos permitirá seguir al usuario o dejar de seguirlo (en caso de que ya lo sigamos).

En cuanto a los perfiles de equipo, serán muy parecidos a los perfiles de usuario, pero con diferente información, con una lista de miembros en la zona de la izquierda de la pantalla y sin lista de seguidos.

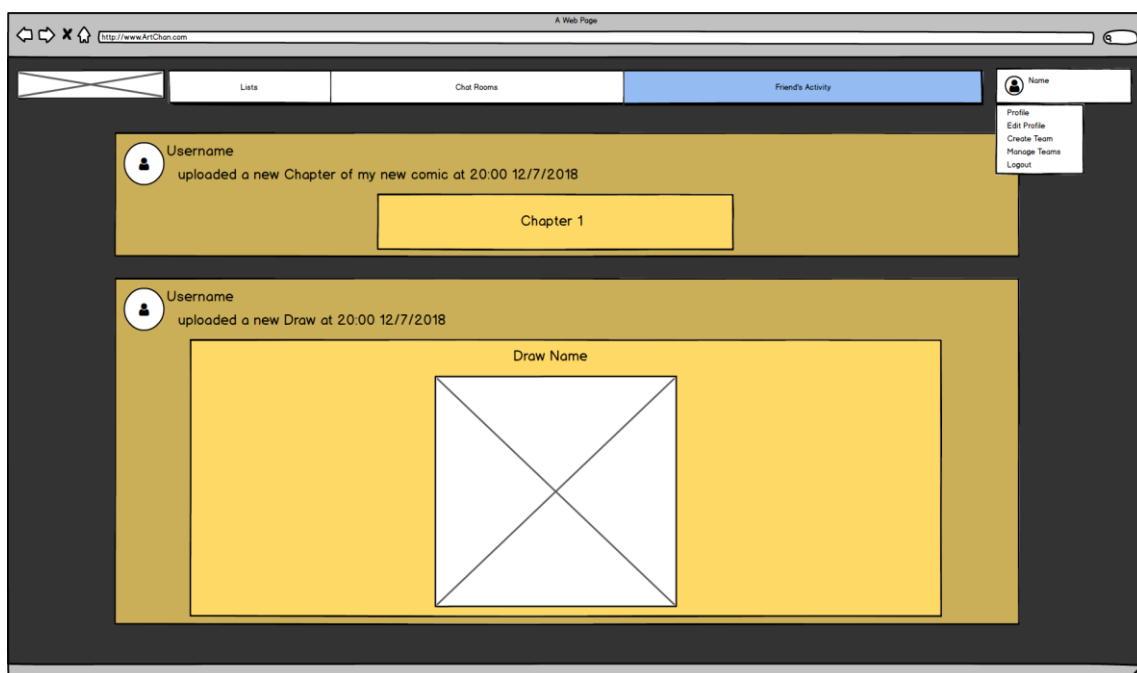


Ilustración 14: Prototipo de la pantalla de Friend's Updates

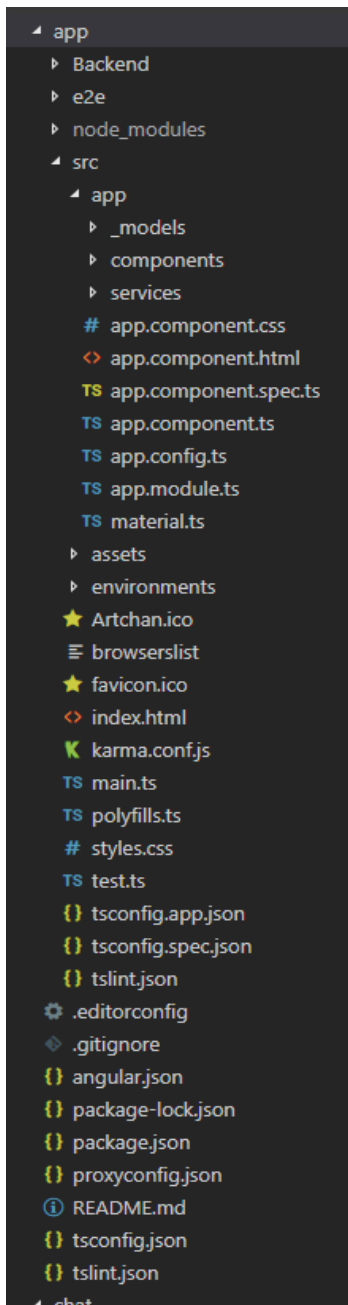
En este prototipo se muestra cómo quedaría la pantalla de Friend's Updates en la web. Estará compuesto de una lista vertical de notificaciones ordenadas por fecha, de más reciente a más antigua.

Hay tres tipos de notificaciones:

-Notificaciones de dibujo: Se mostrará el título del dibujo, el autor, la fecha en la que se publicó y una imagen en pequeño del dibujo

-Notificaciones de capítulo: Se mostrará el título y el número del capítulo, el autor, la fecha en la que se publicó y el cómic donde se ha publicado.

-Notificaciones de episodio: Se mostrará el título y el número del episodio, el autor, la fecha en la que se publicó y la animación donde se ha publicado.



## 4 Implementación

### 4.1 Configuración del proyecto y los servidores

Con tal de tener un control de versiones del proyecto, y un backup de seguridad para estas versiones, se ha creado un repositorio para este proyecto en github: <https://github.com/Madserker/IndieArt/>.

También se ha usado Trello para mantener un orden de las tareas realizadas en la app y para mantener toda la implementación documentada mientras se realizaba el proyecto.

Para este proyecto se ha usado el editor de código Visual Studio Code, a partir de este editor se ha programado la totalidad del código de la web.

En cuanto al Frontend, se ha instalado Angular 6 con Angular Cli para tener acceso a ciertos comandos que facilitan la creación de servicios y componentes para Angular, se ha descargado LESS para poder guardar variables en el CSS y poder tener un mayor control del estilo de la web y se importado Angular Material para modificar el estilo de muchos elementos de la web.

Con respecto al Backend, se ha realizado la instalación de

*Ilustración 15: Estructura de ficheros del Frontend*



PHP, Composer y Laravel y XAMPP para tener acceso a la base de datos y poder crear el servidor para esta.

Se han configurado los diferentes servidores que usa la web:

-El servidor Node del Frontend se ejecutará en el **puerto 4200**, a través del comando: “ng serve”.

-El servidor PHP de Laravel, corresponde a la API de la web y estará localizado en el **puerto 8000**, para ejecutar el servidor se usará el comando: “php artisan serve”.

-El servidor Node que se encarga de los chats de la aplicación estará en el **puerto 3000** y se ejecuta con el siguiente comando: “node index.js”.

-El servidor MySQL para la base de datos, se ejecutará a través de XAMPP y estará en el **puerto 3306**.

-El servidor Apache para gestionar la base de datos a través de la interfaz phpmyadmin, estará en el **puerto 80** y se ejecutará a través de XAMPP.

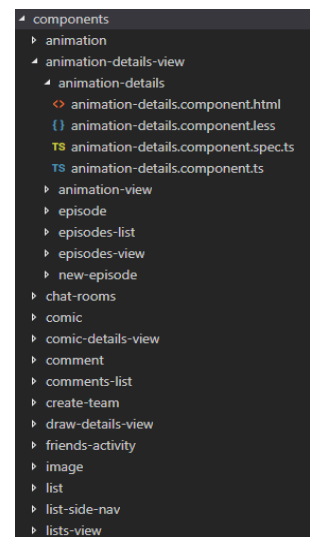
Todos los puertos ya están definidos por código, al usar los comandos descritos se ejecutarán directamente en esos puertos.

## 4.2 Esquema organizativo de ficheros

### 4.2.1 Frontend

Se han definido varios directorios en la parte de Frontend para mantener el código separado y organizado:

- `_models`: Contiene las interfaces de todos los modelos de la web que se usan en la vista de la web. La única función de estas interfaces es la de poder mapear el json que obtenemos de la API a objetos que podemos manipular con Angular.
- `Components`: Contiene todos los componentes de la vista, separados también por carpetas. Cada componente contiene un fichero html para mostrar el componente, un fichero css para definir los estilos del componente y dos ficheros typescript, uno para la lógica de este y asignar el html y el css que se usarán en este componente y otro para hacer los tests de este componente, que se ejecutan con el comando “ng test”. Se han creado un total de 53 componentes en toda la web, todos ellos relacionados entre sí a través de servicios, relaciones de padre-hijo o por routing entre pantalla y pantalla.

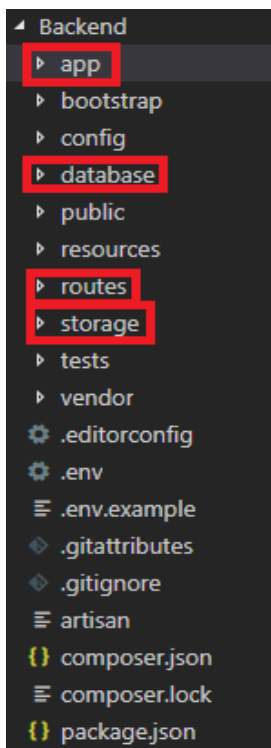


*Ilustración 16:*  
*Estructura de ficheros de los componentes*

- Services: Contiene ficheros typescript que se encargan de relacionar componentes entre sí y de llamar a la API de la web enviando requests de GET, PUT, POST o DELETE. Estos servicios contienen métodos y variables públicas que serán accesibles por cualquiera de los 53 componentes siempre que importen dicho servicio.
- Chat: Contiene el socket.io que se encarga de gestionar cada chat abierto de la web.

También podemos destacar ficheros importantes dentro de la carpeta src del Frontend:

- App.component: Este componente, únicamente se encargará de mostrar el resto de los componentes que usa una pantalla en concreto, dentro de este componente. Este grupo de componentes que muestra varía dependiendo de la ruta en la que estemos, para esto se ha usado: `<router-outlet></router-outlet>`
- App.module.ts: Se encarga de importar todos componentes, servicios y librerías de angular al proyecto y de gestionar las dependencias del proyecto. Dentro de este archivo, también se definirán todas las rutas de la web a las que el usuario puede acceder.
- Material.ts: Dentro de este fichero se importarán los elementos de Angular Material que usaremos en el proyecto.
- Styles.css: En este fichero css, se definirá el estilo global de la página. En este proyecto, este fichero únicamente se ha usado para dar color al background del fondo de la web.



- Index.html: Dentro de este fichero se importará la librería bootstrap y jquery, ya que no se usará Angular Material para todos los elementos de la web. Dentro de este html, también llamaremos al componente principal de la web: `<body><app-root></app-root></body>`

#### 4.2.1 Backend

Dentro de la parte de Backend, se pueden identificar los cuatro directorios más importantes de esta parte:

- App: Este directorio contiene todos los controladores de la web, todos los middlewares necesarios para el correcto funcionamiento de la api, todas las clases del modelo que se mapean desde la base de datos y una

clase para gestionar algunas excepciones como la expiración del token de sesión. Los controladores están situados en el directorio: `app/Http/Controllers`, los middlewares en: `app/Http/Middleware` y las clases del modelo están dentro del directorio de `app`.

- **Database:** Dentro de este directorio están creadas todas las clases necesarias para hacer las migraciones a la base de datos `mysql`.
- **Routes:** En este directorio estarán definidas todas las rutas a la api de laravel, en el fichero `api.php`. Por cada ruta se especificará la ruta, el método ejecutado del controller y los middlewares utilizados.
- **Storage:** En este directorio se guardarán todos los archivos que suban los usuarios a la web, ya que en la base de datos solo guardaremos las direcciones a estos archivos. Se ha realizado de esta forma ya que la api tarda demasiado en obtener archivos desde la base de datos.

Otro directorio que también tienen algo de peso en la app es `config`. Dentro de este fichero se encuentran los archivos necesarios para configurar el token de la sesión del usuario y para conectar la base de datos de `mysql` al proyecto Laravel.

## 4.3 El modelo

### 4.3.1 Las clases

Laravel facilita la creación de las clases del modelo gracias al comando: `php artisan make:model name`. Por defecto, Laravel vinculará automáticamente esta clase del modelo a la tabla de la base de datos con el mismo nombre a no ser que se especifique una tabla en concreto.

Dentro de las clases de los modelos de Laravel, se han definido propiedades de este modelo como que `primary key` se utilizará en la base de datos, si incrementamos el ID automáticamente al añadir una entrada en la lista y si guardamos la fecha en que se editó o se creó un objeto.

También se especifican que atributos son necesarios que se rellenen para crear la entrada en la lista y las relaciones `One to One`, `Many to Many` o `Many to One` entre otras clases del modelo.

## 4.3.2 Las relaciones

### 4.3.2.1 One to Many

Esta relación entre dos clases del modelo especifica que una de las clases tiene como atributo más de un objeto de otra clase diferente o al revés.

Para definir este tipo de relación, se ha especificado dentro de la clase del modelo de esta tabla un método que se encarga de decirle a Laravel que esta clase pertenece a una clase en concreto (`belongsTo`) y en la otra clase otro método donde se especifica que esta contiene varios objetos de la otra (`hasMany`). Aparte de especificarse en la clase del modelo, también se especifica en la migración de la tabla, la foreign key de la clase a la cual se pertenece. Un ejemplo de esta relación es Autores y Arte, un autor puede ser el creador de muchos artes, pero un arte solamente pertenece a un autor.

### 4.3.2.2 Many to Many

Esta relación entre dos clases del modelo especifica que ambas clases tienen como atributo más de un objeto de la otra clase.

Para definir este tipo de relación, es necesaria la creación de una tabla pivot, la función de la cual consiste en guardar que objeto pertenece a que otro objeto de la otra clase. Para realizar esta tabla pivot, se ha definido esta en las migraciones, pero no se ha creado ninguna clase del modelo que haga referencia a esta tabla, ya que Laravel ya nos ofrece métodos dentro de las propias clases implicadas en la relación para saber si un objeto determinado pertenece a otro objeto determinado de la otra clase de la relación. Para decir a Laravel que tabla pivot se está usando y que clases están implicadas en la relación se usará el método `belongsToMany` en ambas clases del modelo.

Dentro del proyecto, se ha tenido que aplicar esta relación en las notas de los usuarios, en las visitas de los usuarios y en los seguidores y seguidos de un usuario, esta última es una relación many to many de la misma clase, es decir varios usuarios pertenecían a varios usuarios.

## 4.3.3 Herencias

Se ha implementado una herencia en las clases del modelo y para simular esta herencia en las tablas de la base de datos y poder usarla en los controladores se ha realizado el siguiente procedimiento:

Se han puesto los mismos IDs en el padre y el hijo de la relación. Es decir, cuando se cree una animación, se creará también el padre (en este caso Art es el padre de Animation) con el mismo ID que se estableció al hijo. De la misma forma, cuando se

elimine uno, se buscará al otro con el mismo ID para eliminarlo también, como si fueran un único objeto.

Dentro del controller, al recuperar los datos de un hijo, se realiza una Query con un inner join del ID entre la tabla del padre y la tabla del hijo, de esta forma sacamos la información completa del objeto (los atributos del padre más los atributos del hijo).

Esta herencia era necesaria en muchos aspectos de la app, ya que por ejemplo se necesitaba una forma de decir a la clase comentarios a que otra clase (animación, cómic o dibujo) hace referencia su foreign key; a través de la herencia, se ha establecido una foreign key hacia la clase Arte donde los hijos son animación, cómic y dibujo y se puede saber en todo momento que tipo de arte es gracias a que no se solapan los IDs.

#### 4.3.4 Migrations

Laravel facilita la creación de los archivos que realizan las migraciones con el comando: “php artisan make:migration name”. Estos archivos se vinculan con las clases del modelo y con las tablas de la base de datos, siempre y cuando tengan el mismo nombre. Al crear una migración a través de este comando, se crean automáticamente dos métodos dentro de este fichero, el método up se ejecutará al ejecutar esta migración, dentro de este se especificará que tabla se desea crear, y el método down especifica que hacer en caso de que se elimine la migración.

Dentro de cada archivo de migraciones en el método up, se especifica que atributos tendrá cada tabla de la base de datos y el tipo de estos.

Estos atributos pueden ser foreign keys y dentro de este fichero se especifica a que otro atributo de otra tabla hacen referencia, se ha especificado también en todas las foreign keys que, al eliminarse el objeto unitario de la relación, se eliminan todas las tablas que estaban relacionadas con este (*onDelete=CASCADE*). Un ejemplo de esto son los capítulos de un cómic, en caso de que se elimine el cómic, todos los capítulos con la foreign key apuntando a este cómic quedarán eliminados también.

Para ejecutar las migraciones definidas en Laravel usamos el comando: “php artisan migrate”. Este comando crea todas las tablas de las migraciones en la base de datos, en caso de encontrar una repetida o una con el mismo nombre, se ha especificado que elimine esta y la vuelva a subir de nuevo.

#### 4.3.5 Query

Laravel ofrece varios métodos para realizar queries y hacer peticiones a la base de datos. Estos métodos se pueden llamar desde la clase del modelo que está vinculada a la tabla o desde la tabla de la base de datos (DB::table(nombre))

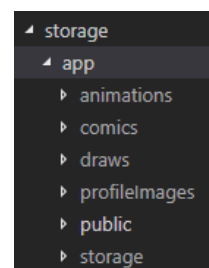
Se han usado los siguientes métodos para este proyecto:

- **Select:** Realizamos una operación select a la tabla especificada.
- **All:** Retorna todas las entradas de una tabla determinada.
- **Where:** Retorna todas las entradas de una tabla que tienen un atributo con un valor determinado.
- **Find:** Igual que where pero retorna un único objeto, no se especifica el atributo, ya que busca únicamente por clave primaria.
- **Get:** Retorna el resultado de la petición a la base de datos.
- **Delete:** Elimina los resultados de una petición determinada a la base de datos.
- **Save:** Guarda una entrada en una lista determinada.
- **Join:** Hace un join de los atributos especificados entre dos tablas determinadas, dentro del método.
- **Contains:** Se ha usado para las relaciones one to many o many to many, con este método se pregunta si la clase contiene alguna entrada determinada de otra clase.
- **Attach:** Se ha usado también para las mismas relaciones, con este método se agrega una entrada de otra clase a la relación.
- **Update:** Para cambiar el valor de un atributo determinado de una tabla.
- **First:** Retorna el primer objeto de una serie de objetos retornados en una query.

## 4.4 Storage

Cada vez que un usuario crea un dibujo, una animación, un cómic, un capítulo o un episodio, este sube un archivo o un conjunto de archivos desde su ordenador a la web, esto se gestiona de la siguiente manera:

En la parte de Frontend, se especifica qué tipo de archivos son válidos y se abre el explorador de archivos, una vez validados el resto de los campos del formulario, se lanza la request a la API desde el service, enviando el archivo y el resto de los campos del formulario como FormData en vez de JSON.



*Ilustración 18: Estructura de ficheros de Storage*

En la parte de Backend, se escucha la request enviada por el service, se vuelven a validar otra vez todos los campos y a comprobar que estamos logueados y se genera un path automático para el archivo con el método: `Storage::putfile()`. Posteriormente, se guarda este path en la base de datos y se recupera en la parte de Frontend para mostrarse en la vista de la aplicación.

Dentro de storage se han creado diferentes directorios adaptando el path de las entradas de cada tabla a estos directorios para poder tener organizados todos los archivos (fotos y videos) de la app.

## 4.5 Middleware

Los middleware permiten establecer un mecanismo para filtrar cualquier request HTTP que entre en la aplicación, Laravel permite la creación de estos middlewares a través del comando: `php artisan make:middleware name`.

A partir de este comando, se ha creado una clase CORS para poder tener acceso a la API desde el servidor de Angular. Dentro de esta clase se han especificado todas las cabeceras y métodos posibles en las requests que se enviarán desde el cliente. En un principio no es necesario este middleware CORS para Laravel, pero se tuvo que implementar ya que la API de Laravel rechazaba todas las requests que venían del servidor de Angular a no ser que se aplicara el middleware CORS

Para hacer uso de este middleware, se ha de especificar en cada ruta de la api el siguiente middleware CORS.

## 4.6 Log in y Sign up

El log in y el sign up de la app están en el topbar y el formulario de registro y login se muestra con una barra lateral al igual que con la barra de filtros solo que en este caso se superpone encima de la pantalla, de esta forma el usuario tiene más comodidad para loguearse y registrarse, ya que puede hacerlo en cualquier momento y en la pantalla donde esté sin ser redireccionado a otra ruta de la app.

### 4.6.1 Sign up

La validación del Sign up se realiza en la parte del Backend con Laravel. En la parte Frontend únicamente se mira si ambas contraseñas coinciden, en caso afirmativo se envía la request al Backend donde se realiza la validación de los datos introducidos y

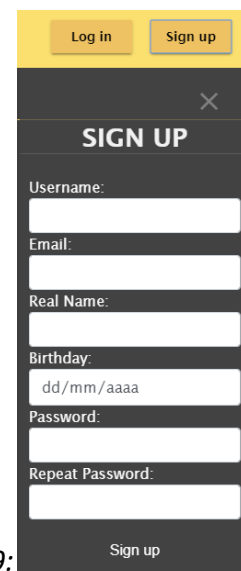


Ilustración 19:  
Vista del Sign Up

se crea el usuario en caso de que los datos estén bien, la validación del Backend consiste en lo siguiente:

- Username: Se valida que se haya introducido este campo y que el username introducido sea único en la tabla de Autores, la cual engloba usuarios y equipos y que pase la expresión regular que asegura que no puede empezar ni por . ni por \_ ni acabar por . ni \_ y tiene que tener entre 3 y 20 caracteres.
- Email: Se valida que se haya introducido este campo y que el texto introducido tenga formato de correo y no exista ningún usuario con este correo ya registrado.
- Real name: Se valida que se haya introducido este campo y que cumpla la misma expresión regular que usa el campo username
- Birthday: Se han realizado también expresiones regulares para confirmar que el formato de la fecha introducido es válido, que el campo de meses no supere 12, que el campo de días no supere el 31, que el año este entre el 1950 y el 2019 y que el campo no este vacío.
- Password: Se valida que el campo no este vacío y que cumpla la expresión regular donde debe contener mayúsculas y minúsculas y debe tener mínimo 8 caracteres. La contraseña del usuario se encripta con Laravel usando la función hash bcrypt() y se guarda en la base de datos de forma encriptada.

Si la validación de todos los campos es correcta, se crea el usuario con los datos introducidos, con la descripción de usuario vacía y se le asigna una foto de perfil aleatoria entre 6 diferentes que hay guardadas en el storage de la app. Si la validación es errónea, se muestra un alert al usuario con un mensaje de por qué no se ha podido crear la cuenta y el usuario no se crea en la base de datos.

## 4.6.2 Login

La validación del login se realiza en la parte del Backend con Laravel, esta validación consiste en lo siguiente:

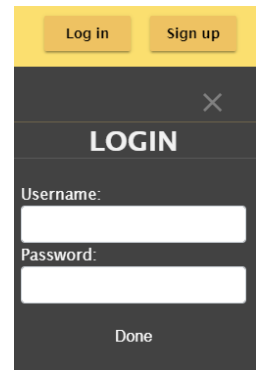
Primeramente, se verifica que ambos campos no estén vacíos y posteriormente se realiza una query a la base de datos donde se busca al usuario con el username introducido. Si encontramos el usuario seguimos con la validación, si no retornamos error.

Después de validar que existe el usuario, se hace uso de la librería **JWT-Auth token** en Laravel para verificar que el username y la contraseña coinciden, en caso afirmativo se retorna el usuario de la base de datos y el token creado por esta librería, con este



token se podrá mantener la sesión del usuario activa mientras navega por la app y se realizará el login automático al entrar en la app.

El token creado por la librería se guarda en el local storage de la web para poder hacer inicio de sesión automático, este token se recupera en el Frontend a través del servicio `auth.service` creado en angular, ya que todas las peticiones a la API que requieran permisos de usuario necesitarán este token en el cuerpo de la petición para poder verificar en la parte de la API que el token sigue siendo válido y que el usuario que hace la petición corresponde con el token enviado, para verificar el token en Laravel se usa el siguiente método de JWT: `JWTAuth::parseToken()->authenticate()`



*Ilustración 20:  
Vista del Log In*

Un ejemplo de petición GET a la API que requiera permisos de usuario sería así:

```
GET: localhost:8000/api/public-chats/?token=TOKEN_NUMBER
```

La duración de este token se ha especificado dentro del fichero de configuración del token que está localizado en `Backend/app/config/jwt.php`. Dentro de este fichero se ha cambiado el atributo `'ttl'` por un valor de un mes, es decir cuando pase un mes este token dejará de ser válido y el usuario deberá volver a loguearse en la app.

## 4.7 Routing

Dentro del módulo de angular se han definido las diferentes rutas a las que un usuario puede acceder dentro de la web, sin contar las rutas de la API que están definidas en Laravel. En estas rutas se especifica cuál es el camino hacia la ruta y el componente que debe de cargarse al acceder a esta ruta. La sintaxis para definir una ruta es la siguiente: `{path: '', component: nombre}`.

Estas rutas pueden ser estáticas si es siempre el mismo path o dinámicas si el path es diferente dependiendo de la acción del usuario, estas rutas cargan siempre el mismo componente, pero dentro de este se recupera la parte variable de la ruta y se muestra una cosa u otra dependiendo de esta parte variable. Por ejemplo, al abrir los detalles de un dibujo, se usa la ruta `/draw/{id}`, este id varía dependiendo del dibujo que el usuario abra, una vez abierto el componente, en el constructor de este recuperamos el id y mostramos el dibujo que tiene como id el de la ruta.

Todas las rutas definidas en el módulo de la app se gestionan desde `app.component` con el `router-outlet` y se muestra el componente de la ruta dentro de este `app.component`.

## 4.8 Barra de navegación lateral

Se ha implementado una barra de navegación lateral en la pantalla de Lists para poder buscar los elementos de la lista, filtrarlos y ordenarlos. Esta barra se oculta y se muestra a través de un botón en el ToolBar de Lists, de esta forma no se sobrecarga demasiado la pantalla, ya que una vez se ha realizado una búsqueda, no hay necesidad de seguir mostrando la navegación a no ser que se quiera buscar otra vez.

A continuación, se explicará en detalle cada zona de la barra de navegación lateral:

### 4.8.1 Filtros y Tags

Se ha creado una lista de tags en la base de datos donde se especifica el nombre del tag y el tipo de tag, es decir si corresponde a un tag de dibujo, de cómic o de animación.

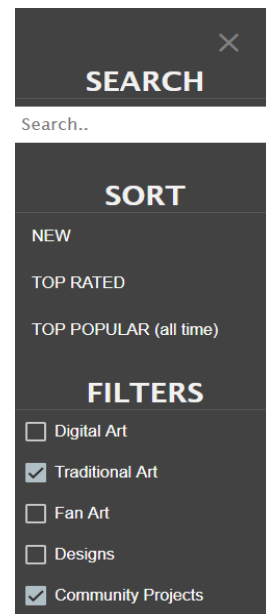
En Laravel se ha asociado esta lista de tags a la clase Tags y al tener una relación Many to Many entre tags y arts se ha creado también la tabla pivot art\_tag y se ha especificado en las clases este tipo de relación.

Se han creado diferentes rutas y métodos para poder editar, agregar o eliminar estos tags dentro de un arte en concreto, estas rutas son usadas por el servicio TagsService de la vista de la app y por el servicio getTags.

A continuación, se explicará en detalle los servicios implicados en los filtros y tags de la aplicación:

#### TagsService

Este Service recupera los tags de un arte concreto y añade o edita los tags de un arte en concreto. Se usa este Service en la vista a través de los componentes de creación de un nuevo cómic, dibujo o animación, donde el usuario selecciona los tags que quiere que tenga su arte y lo publica llamando al método de creación de un nuevo arte en ListsService y dentro de este se llamará al método de TagsService que añadirán los tags a este nuevo arte creado.



*Ilustración 21:  
Barra de  
navegación lateral*

## GetTagsService

Con este Service se recupera la totalidad de tags de la base de datos de un tipo de arte en concreto (animación, cómic o dibujo). Este Service se usa también al crear un nuevo arte, ya que se necesita mostrar al usuario el conjunto de tags disponibles para elegir en su nuevo arte. También se llama a este Service dentro del Service ChangeFilters.

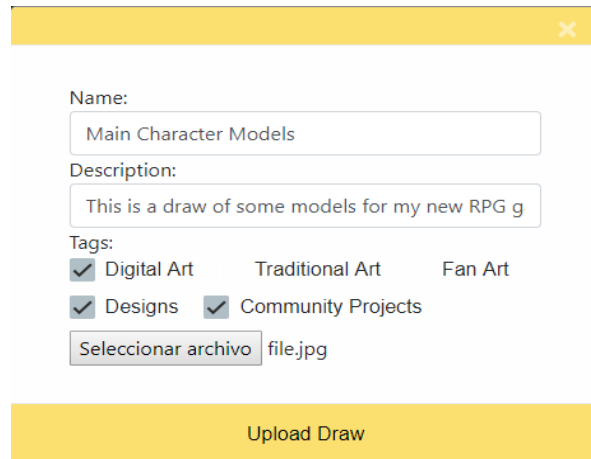


Ilustración 22: Template de creación de dibujos

## ChangeFiltersService

Se encarga de recuperar los filtros de cada tipo de arte a través del Service getTagsService y guardarlos en listas que pueden consultar cualquier componente que importe el servicio.

Este servicio se ha usado para mantener conectados el componente de la barra de filtros con el toolbar de la lista, ya que cuando se cambia de tipo de arte en el toolbar, se necesitaba alguna forma de avisar al componente de filtros que cargase los otros filtros. Con este servicio, ambos componentes se subscriben a estas listas de filtros, donde toolbar edita la lista y la barra de filtros se encarga de observar y mostrar esta lista recién editada.

Dentro del componente List se filtrarán los objetos de la lista aplicando los filtros disponibles que ha seleccionado el usuario a través de la barra de filtros.

## 4.8.2 Ordenar

El usuario puede elegir entre ordenar la vista por fecha de publicación, por más votados o por más visitas.

Una vez el usuario elige un tipo de orden, se llama a los métodos del servicio ListsService desde el componente List.

Para notificar cuando se ha seleccionado un tipo de orden al componente List desde el componente de la barra de filtros, se han usado los **EVENT EMITTERS**, estos event emitters envían un evento al producirse una acción determinada (en este caso clicar a una opción), este evento es escuchado por todo componente que esté suscrito a

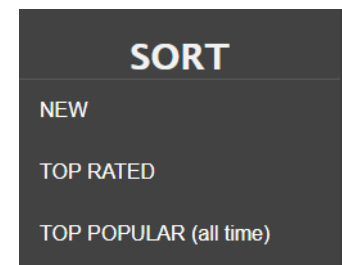


Ilustración 23: Opciones de ordenar la lista

este event emitter, en este caso list escucha el evento y llama al método adecuado del servicio ListsService donde se realiza una llamada a la API desde este servicio.

En la parte de la API, se ordena la lista recuperando los votos de cada publicación o recuperando las visitas o leyendo la fecha de publicación y se retorna al Frontend la misma lista, pero ordenada.

La lista se ordena respetando los filtros introducidos por el usuario, de la misma forma que se puede ordenar y posteriormente aplicar los filtros deseados.

### 4.8.3 Search

La barra de búsqueda de la app está disponible en todas las listas del apartado Lists de la app (Draws, Cómics, Animations, Users y Teams).

Esta barra de búsqueda no dispone de botón de búsqueda ya que escucha al evento keyup para realizar la búsqueda, es decir cada vez que se introduce un carácter en esta barra, se filtra la actual lista por los caracteres introducidos y se retornan todos los elementos que contengan esos caracteres en el título del elemento.

La búsqueda de elementos mediante esta función respeta cualquier filtro que se tenga activo en ese momento, es decir, solo se busca en los elementos filtrados anteriormente.

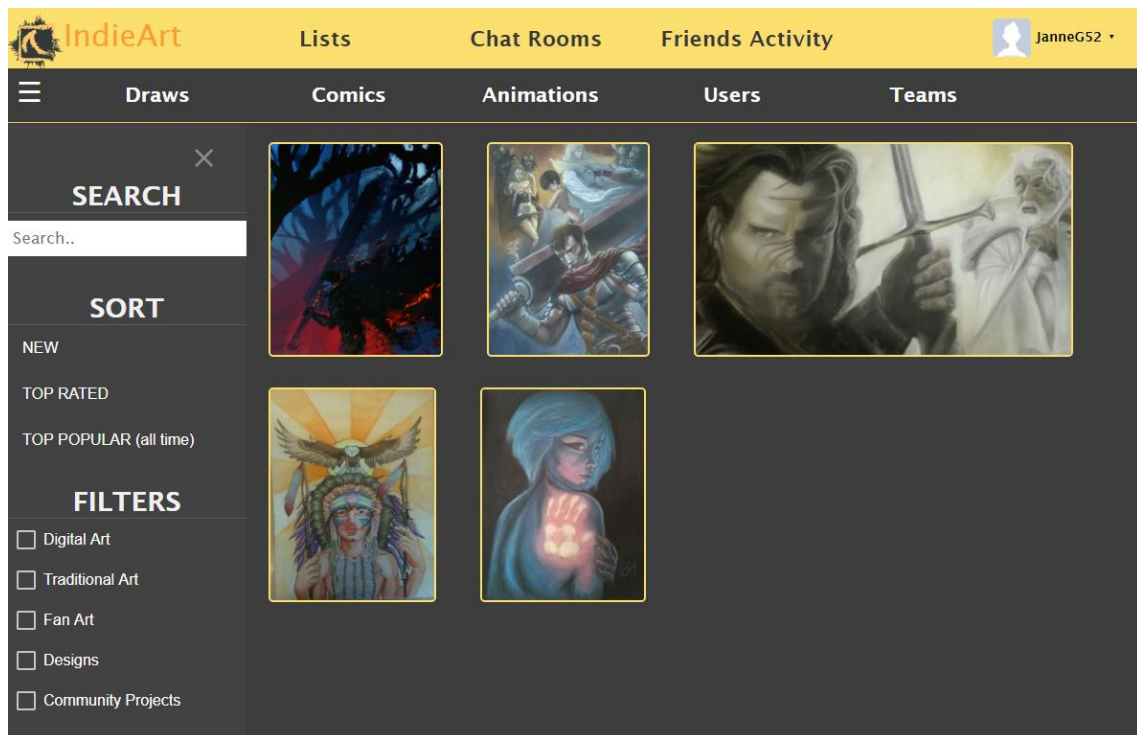


Ilustración 24: Vista de Lists (Draws)

## 4.9 Chats

Dentro de la aplicación, en el apartado de chats rooms se han implementado tres tipos de listas, estas listas están separadas por tipos de chats, los chats públicos que son accesibles para cualquier usuario, pero solo pueden ser eliminados por el creador; los chats privados, donde solamente los miembros de este pueden acceder al chat (los miembros los añade el creador una vez creado el chat introduciendo el nombre de usuario del nuevo miembro) y únicamente el creador puede eliminar el chat y los chats de equipo que únicamente tienen acceso los miembros de un equipo determinado (estas salas se crean automáticamente al crearse el equipo).

Al igual que con el arte y los autores, con chats también se ha tenido que realizar una herencia en la base de datos para poder asociar los IDs de los chats en la tabla pivot que especifica que usuario pertenece a que chat, así pues, las propiedades compartidas entre public chats, private chats y team chats están definidas en la tabla chats.

Para implementar estos chats, se ha creado un socket desde angular que emite un evento de nuevo mensaje con el id del chat donde se ha producido el nuevo mensaje. Este evento es escuchado por el componente de angular chat-viewer que verifica que el id del evento es el del chat donde está el usuario y actualiza la lista de mensajes. Al actualizar esta lista, valida si hay más de 50 mensajes, en caso de que se supere este límite, se eliminan los mensajes más antiguos, de esta forma no llenamos la memoria de la base de datos. El evento que emite el socket es producido cada vez que un usuario hace un post de un mensaje nuevo en ese chat.

Para recuperar los mensajes, eliminarlos, publicarlos y lanzar eventos al socket se usa el servicio chat-service definido en la parte de angular de la app.

El socket utilizado para realizar esta funcionalidad es el socket.io ya que funciona con Node.js al igual que el servidor del Frontend y se ejecutará en el puerto 3000. Para comprobar el correcto funcionamiento del socket, de los eventos emitidos y los chats se ha hosteado la aplicación en la red local y se ha creado un chat entre dos cuentas de usuario una en cada ordenador conectado a la misma red local.



Ilustración 25: Vista de Chat Rooms (Team Chats)

## 4.10 Friend's Activity

En esta pantalla recuperamos las últimas 30 publicaciones de los usuarios o equipos que sigue el actual usuario, se muestran ordenadas de más reciente a más antigua. Se han usado las collections de Laravel para poder ordenar las publicaciones.

Para mostrar estas noticias se ha creado la clase Notifications en el modelo de la app. Estas Notifications pueden ser de tipo nuevo dibujo, nuevo episodio o nuevo capítulo, dependiendo del tipo se mostrarán unos atributos u otros en la vista de la app.

A través de una notificación podemos acceder directamente al perfil del usuario autor de la notificación o al dibujo, capítulo o episodio publicado.

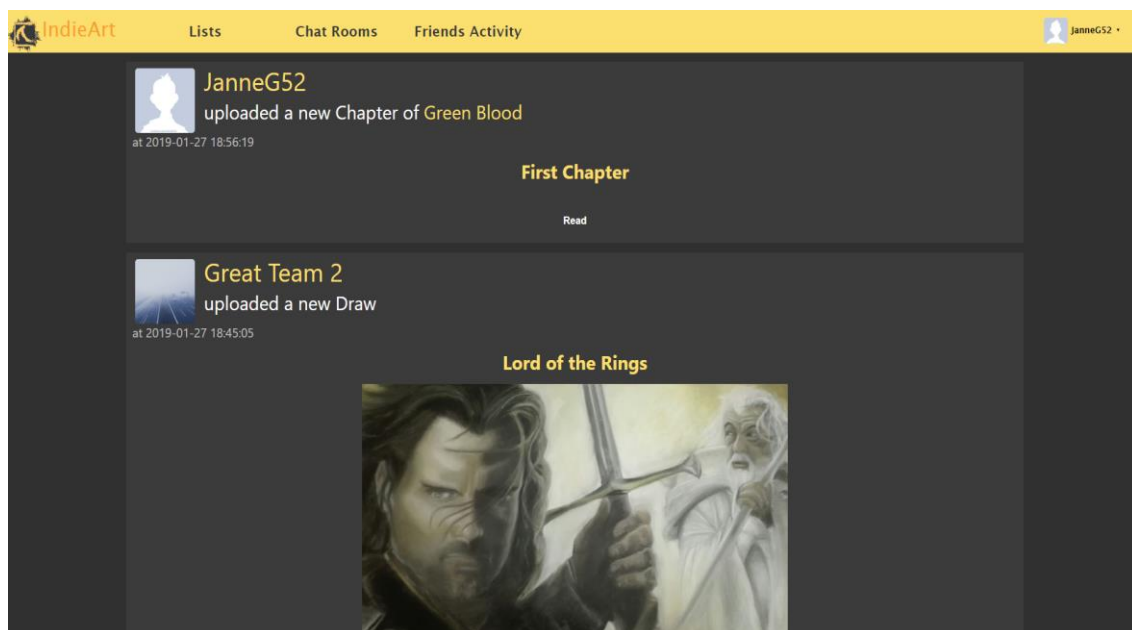


Ilustración 27: Vista de Friend's Updates

Los seguidores y seguidos de un usuario o un equipo se mostrarán en la parte lateral izquierda del perfil de este mostrando únicamente la foto del usuario o equipo, esta foto es un link al perfil del usuario.

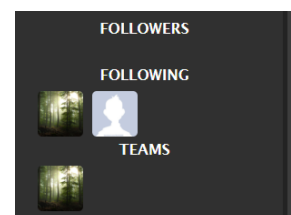


Ilustración 26: Lista de miembros, seguidores y seguidos en el perfil

## 4.11 Equipos

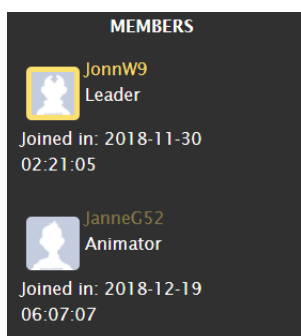
Los equipos consisten básicamente en cuentas compartidas por varios usuarios donde los miembros de esta cuenta pueden subir publicaciones, ascender otros miembros a administradores, establecer su rol en el equipo, participar en el chat de equipo o añadir más miembros al equipo. A continuación, se explica en detalle cómo llevar a cabo estas acciones:

### 4.11.1 Creación de un equipo

Cualquier usuario puede crear un equipo a través del botón que aparece al clicar a su perfil en el topbar. Una vez dentro, aparece un formulario donde debe rellenar el nombre (el cual no debe estar cogido, es decir, ningún usuario o equipo debe llamarse así ya), la descripción del equipo, la foto de perfil del equipo y el rol desempeñado por el usuario en el equipo.

Una vez rellenado el formulario, se valida y se crea el equipo con un único miembro y administrador, que sería el usuario que ha rellenado el formulario. Al crearse el equipo, se crea también el chat de equipo en el apartado Team Chats, al cual solo pueden acceder los miembros de este.

Dentro del perfil de un equipo se pueden identificar los miembros en el lado lateral derecho, dentro de este apartado, cada miembro aparece con un marco amarillo en caso de que sea administrador y con su foto, su nombre de usuario y su rol en este equipo.



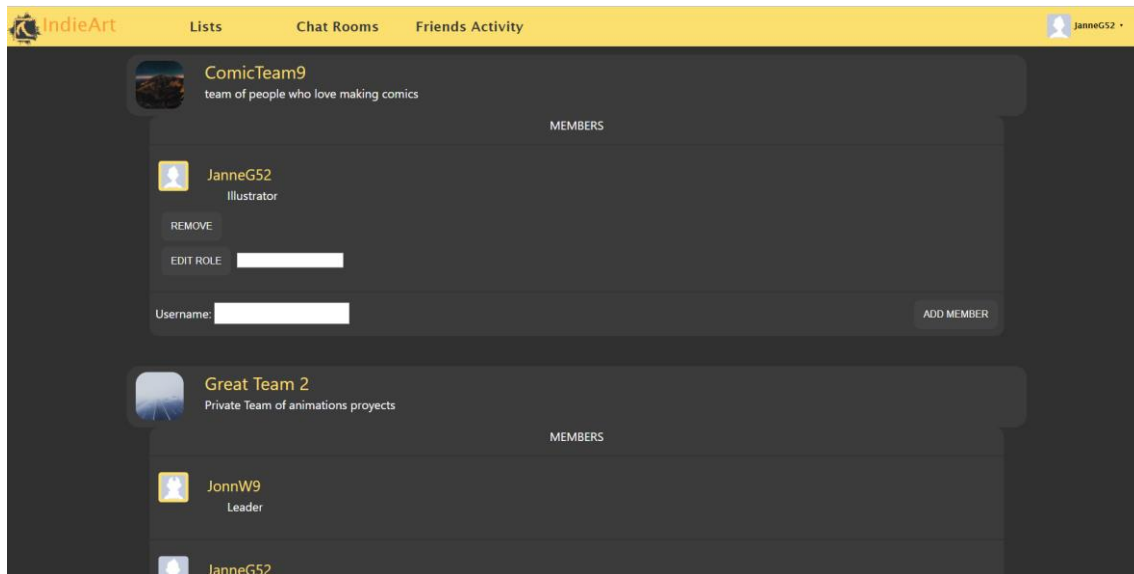
*Ilustración 28: Lista de miembros de un equipo*

### 4.11.2 Gestionar equipos

En el topbar podemos acceder a la opción Manage Teams, esta opción muestra la lista de equipos de los cuales el usuario actual es miembro, dentro de cada equipo de la lista se muestra la información de este y la lista de miembros del equipo.

A través de esta pantalla, el usuario puede editar su rol en el equipo o salir del equipo, en caso de que el usuario sea administrador del equipo, también tiene las opciones de añadir otro miembro al equipo o de promocionar otro usuario miembro a administrador.

Si un equipo se queda sin miembros o administradores, este es eliminado de la base de datos automáticamente (esta es la única forma de poder eliminar un equipo).



*Ilustración 29: Vista de Manage Teams*

### 4.11.3 Publicar como equipo y eliminar publicaciones

Si un usuario miembro de un equipo desea publicar en nombre de un equipo conjunto, deberá hacerlo desde el perfil del equipo, donde rellenará el mismo formulario de creación de arte y se publicará el arte en nombre del equipo, así pues, solo los usuarios que sean miembros de ese equipo podrán eliminar ese arte o seguir publicando capítulos o episodios dentro de este en caso de que sea un cómic o una animación.

La comprobación de si un usuario puede eliminar o no un arte se realiza tanto en el Frontend como en el Backend para garantizar la máxima seguridad de la web. La comprobación consiste en verificar a través del token si el usuario que hace la petición de eliminar es el autor de esta publicación, en caso de que no sea así se comprueba si el autor de la publicación es un equipo, en caso de que así sea, se buscan los miembros de este y se mira si coincide algún miembro de estos con el actual usuario que se ha obtenido del token. Si el usuario está dentro de los miembros, se le deja eliminar la publicación, si no saltará un mensaje diciendo que no tiene autorización para eliminar. Además de hacer esta validación, el botón de eliminar publicación no aparece en la vista si no se pasa la validación del Frontend.



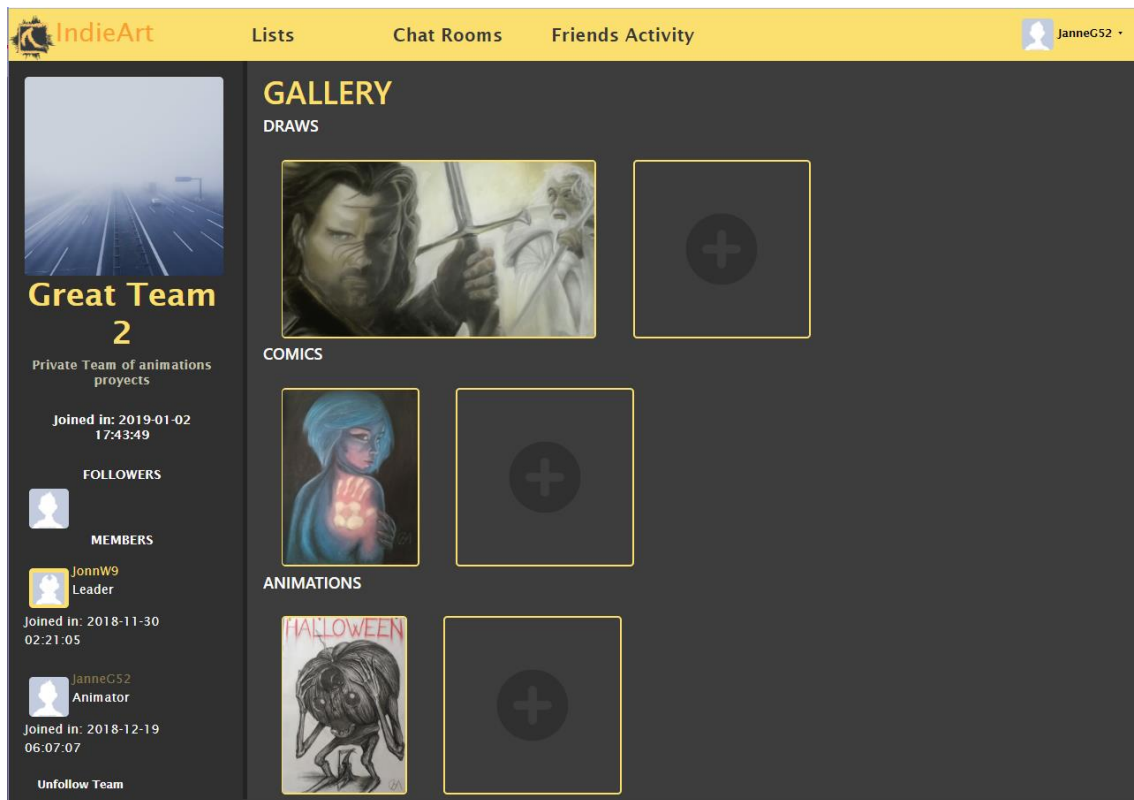


Ilustración 30: Vista del perfil de un equipo

## 4.12 Comentarios, votos y visitas

La web permite al usuario comentar cualquier publicación de la web, tanto animación, cómic o dibujo. Estos comentarios están asociados a estas publicaciones en una relación de many to one y constan de un autor y un mensaje.

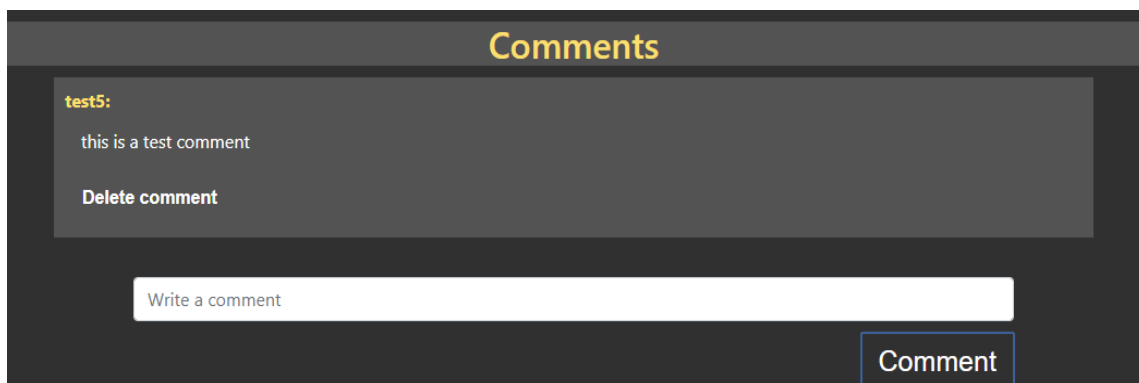


Ilustración 31: Vista de comentarios de una publicación

El usuario también tiene la opción de eliminar sus propios comentarios.

Siempre que el usuario elimine un comentario o publique un comentario, el componente de la vista comment envía un event emitter al componente comments list

para que actualice esta lista de comentarios desde la base de datos, sin tener que refrescar la página.

Los votos se han implementado de forma que cuando se entra en una publicación en concreto, el programa recupera la nota del usuario en caso de que lo haya puntuado y la nota media de todas las puntuaciones. Cuando el usuario vota esta publicación, se envía un POST de la nota al servidor sobrescribiendo la nota actual del usuario.

Respecto a las visitas, solo se tienen en cuenta si el usuario que ha visualizado una publicación esta logueado, en caso de que este logueado, se guarda en una tabla que publicación ha sido visitada y por quien en caso de que no la haya visitado antes, de esta forma se evita que suba el contador de visitas al visitar la publicación más de una vez y de que no aumente las visitas si no está logueado.

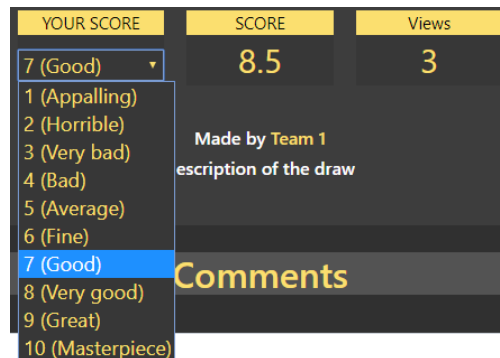


Ilustración 32: Vista de la sección de votos y visitas en una publicación

## 4.13 Visores

### 4.13.1 Visor de cómics

Dentro de los detalles de un cómic, se puede acceder a la lista de capítulos de este, ordenadas por fecha de publicación. La creación de cada capítulo consiste en rellenar un formulario donde se especifica el nombre del capítulo, el número de este y se van agregando o eliminando páginas a través de un input file; una vez el usuario ha subido todas las páginas que quería desde su pc, confirma el formulario y se sube el capítulo con todas las páginas.

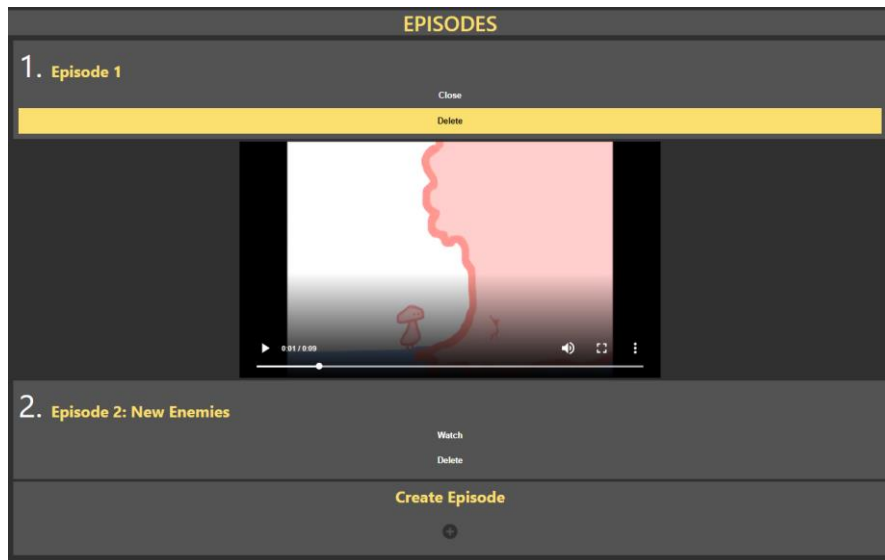
Si el usuario hace clic en el capítulo, se abrirá un visor de imágenes donde se muestra página por página el capítulo, reiniciando el scroll al pasar de página y con un contador de páginas arriba a la derecha. El usuario también puede usar las flechas de dirección del teclado para cambiar de página y la tecla Escape para salir del visor, de esta forma se hace la lectura del capítulo más cómoda.

### 4.13.2 Visor de episodios

Dentro de los detalles de una animación, el usuario puede acceder a la lista de episodios de esta, dentro de esta lista el usuario puede crear un nuevo episodio (siempre que sea autor de la animación), el formulario de creación del episodio consiste en rellenar el título del episodio, el número y el video.

Se ha usado el visor por defecto de HTML5 para implementar los visores de video de cada episodio. Este visor es un desplegable, que se abre dentro de la misma página de la lista de episodios cuando el usuario clicca a un episodio de la lista.

Solo puede haber un visor activo a la vez, así que cuando se abre otro episodio, se cierra el desplegable anterior.



*Ilustración 33: Lista de episodios con el visor desplegado*

## 5 Conclusiones

En este proyecto, se ha diseñado e implementado una aplicación web que funciona como red social entre artistas. Este trabajo se ha realizado siguiendo la lista de requisitos mencionada en la introducción y siguiendo un diseño y análisis premeditado antes de la implementación.

En lo que respecta a los objetivos, se considera que se han cumplido todos los objetivos específicos descritos en la introducción del proyecto gracias a los conocimientos adquiridos en el grado, así como la experiencia en el desarrollo de webs que se ha conseguido en todo el proceso de implementación de la web.

También se considera que se han cumplido los objetivos generales del proyecto, ya que la app implementada, ofrece al usuario las herramientas necesarias para poder contactar con otros artistas, crear equipos para publicar conjuntamente y escalar en el ranking de la web con las publicaciones del usuario.

Por otro lado, se ha realizado una web facilitando al usuario las tareas más pesadas como en el caso del login y el registro, y haciendo la navegación por la web rápida e intuitiva gracias a Angular que permite la creación de Single Page Applications dando a los usuarios una experiencia mucho más fluida de la web ya que todos los códigos de HTML, CSS Y JS se cargan de una vez.

Otro aspecto por destacar es el tema de la seguridad de la web, se han usado tokens para las peticiones a la API, se han realizado validaciones tanto dentro del Backend como dentro del Frontend con expresiones regulares y con consultas a la base de datos. Se ha dedicado bastante tiempo en esta zona, ya que considera un tema muy importante por el hecho de que un usuario pueda eliminar por ejemplo una publicación de otro usuario, esto causaría demasiadas consecuencias negativas para el usuario afectado y es algo que no se debe permitir bajo ningún concepto.

Finalmente, se ha de destacar el correcto funcionamiento de los equipos de la web, donde se puede publicar conjuntamente, gestionar los miembros asignando roles y promoviéndolos y tener un chat dedicado únicamente a los miembros del equipo. Esta es la funcionalidad más importante de la web, ya que ninguna otra web de arte ofrece esta posibilidad y el hecho de que exista esta funcionalidad hace que la web se acerque más al objetivo principal que es que la relación entre artistas sea posible.

## 5.1 Futuras ampliaciones

Una posible ampliación para la web podría ser, añadir una nueva sección de competiciones sobre una temática en concreto, donde artistas y equipos pudieran presentar su proyecto y votar a los candidatos de la competición. Estas competiciones tendrían una duración de una semana y se recompensaría a los usuarios que han votado y se entregarían premios a los usuarios ganadores. Estos premios podrían ser o bien medallas para su perfil o logros o algún marco en el proyecto ganador que dé más reconocimiento público a este usuario o equipo, de esta forma se consigue hacer la web más social y más competitiva.

Otra futura mejora, sería aprovechar el apartado de chats rooms para introducir otro tipo de chat donde las empresas puedan colgar sus anuncios y ofertas de trabajo con todos los detalles y los usuarios puedan ver estas ofertas y abrir un chat con esta empresa dentro de la oferta para presentarse como candidato para el puesto de trabajo que ofrezcan, de esta forma se daría un enfoque algo más profesional a la web y los usuarios cuidarían mucho más las publicaciones colgadas por ellos, ya que las empresas tendrían total permiso para visualizar sus perfiles.

Otra posible mejora para la web sería poder notificar al usuario cuando reciben comentarios en sus publicaciones. Estas notificaciones se irían sumando en una lista a la cual podría acceder el usuario a través de un botón. Antes de entrar en esta lista, el usuario puede ver el número de notificaciones que tiene pendientes por mirar. Estas notificaciones contienen una breve descripción y un link que redireccionaría al usuario a dicho comentario.

## Glosario

**Framework:** Es un entorno de trabajo que permite desarrollar software de manera más sencilla, puede incluir librerías, soporte de programas y lenguaje interpretado para ayudar a unir los diferentes componentes de un proyecto.

**TypeScript:** Es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases.

**LESS:** Es un lenguaje dinámico de programación para dar estilo a los diferentes componentes de la web.

**HTML:** Es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Se trata de las siglas que corresponden a *HyperText Markup Language*.

**PHP:** Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

**Laravel:** Es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7.

**Angular:** Es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

**Request:** En un modelo cliente-servidor, un request es un requerimiento, petición o solicitud que le hace un cliente a un servidor.

**Hostear:** Almacenar una página web o aplicación web en un servidor, el cual tiene las características necesarias para atender las peticiones.

**Token JWT:** Es una cadena de caracteres que permite la propagación de identidad y privilegios, pueden ser utilizados para propagar la identidad de usuarios como parte del proceso de autenticación entre un proveedor de identidad y un proveedor de servicio.

**API:** Se trata de las siglas que corresponden a *application programming interface*, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software.

**Query:** Es la parte de una URL que especifica a una aplicación web que datos quiere retornar de la base de datos a través de esta petición.

**Frontend:** Es la parte del software que interactúa con los usuarios.

**Backend:** Es la parte del software que procesa la entrada desde el Frontend.

## Bibliografía

Denys Vuika. (2018). Dynamic Routes with Angular. [en línea]. Recuperado el 9 de noviembre de 2018 de <https://medium.com/@DenysVuika/dynamic-routes-with-angular-6fda03b7fa2c>

Google. (2010). Angular Cli, A command line interface for Angular. [en línea]. Recuperado el 13 de septiembre de 2018 de <https://cli.angular.io/>

Google. (2010). Angular Material Components. [en línea]. Recuperado el 21 de octubre de 2018 de <https://material.angular.io/components/categories>

Google. (2010). What is Angular?. [en línea]. Recuperado el 10 de septiembre de 2018 de <https://angular.io/docs>

Jan Goyvaert. (2003). Regular Expresions Info. [en línea]. Recuperado el 19 de octubre de 2018 de <https://www.regular-expressions.info/>

Luis Aviles. (2018). Real Time Apps with TypeScript: Integrating Web Sockets, Node & Angular. [en línea]. Recuperado el 6 de diciembre de 2018 de <https://medium.com/dailyjs/real-time-apps-with-typescript-integrating-web-sockets-node-angular-e2b57cbd1ec1>

Maximilian Schwarzmüller. (2016). Laravel + Angular 2 / Vue.js. [en línea]. Recuperado el 15 de octubre de 2018 de <https://github.com/mschwarzmueller/laravel-ng2-vue>

Sean Tymon. (2014). JSON Web Token Authentication for Laravel & Lumen. [en línea]. Recuperado el 29 de octubre de 2018 de <https://github.com/tymondesigns/jwt-auth/wiki>

Stack Exchange Inc. (s.f). Stack Overflow. [en línea]. Recuperado el 28 de septiembre de 2018 de <https://stackoverflow.com>

The core Less Team. (2009). Less In-Depth Guide. [en línea]. Recuperado el 15 de noviembre de 2018 de <http://lesscss.org/features>

Taylor Otwell. (s.f). Documentación oficial Laravel 5.7. [en línea]. Recuperado el 27 de septiembre de 2018 de <https://laravel.com/docs/5.7>

Typescriptlang. (2012). Typescript Documentation. [en línea]. Recuperado el 4 de noviembre de 2018 de <https://www.typescriptlang.org/docs/home.html>

W3Schools. (1999). HTML, The language for building web pages. [en línea]. Recuperado el 2 de noviembre de 2018 de <https://www.w3schools.com>