

CIFP Ceuta



AnimeInfo

Carlos Atencia Ariza

2º DAW

Tutor

Rubén

Resumen

Esta aplicación web está basado en la muestra de Animes, recogiéndolas a través de una API.

Permite la creación de usuario, además de buscar y obtener información de los animes que les pueda interesar gracias a un buscador y filtros, también ofrece la capacidad de guardar en listas que el propio usuario puede crear y la capacidad de tanto compartir como comentar animes.

Palabras claves: animes, API, creación de usuario, buscador, filtros, listas, compartir, comentar

English Version

This web application is based on showcasing Animes, gathering them through an API.

It allows user creation, as well as searching for and obtaining information about the animes that may interest them thanks to a search engine and filters. It also offers the ability to save in lists that the user can create, as well as the ability to both share and comment on animes.

Keywords: animes, API, user creation, search engine, filters, lists, share, comment.

Índice

Contenido

| | |
|---|----|
| Resumen | 5 |
| English Version | 5 |
| Índice | 6 |
| Introducción | 9 |
| Definición del Producto/Servicio | 9 |
| Alternativas | 10 |
| Kitsu..... | 10 |
| Animedatos | 10 |
| Análisis DAFO..... | 11 |
| Requisitos Funcionales..... | 12 |
| Requisitos no Funcionales..... | 12 |
| Casos de uso | 13 |
| Usuario..... | 13 |
| Administrador | 13 |
| Problemas que soluciona respecto a otros proyectos..... | 14 |
| Objetivos..... | 14 |
| Diagrama de Gantt..... | 15 |
| Ciclo de desarrollo | 16 |
| Guía de estilos | 16 |
| Fuentes de colores | 16 |
| Colores Principales | 16 |
| Colores secundarios | 17 |
| Colores hover | 17 |
| Fuentes de texto | 18 |
| Chicle..... | 18 |
| Asap..... | 18 |

| | |
|---|----|
| Lato..... | 18 |
| Be vietnam pro..... | 18 |
| Iconos | 19 |
| Wireframe | 20 |
| Desktop | 20 |
| Mobile | 41 |
| Base de datos..... | 59 |
| Entidad Relación | 59 |
| Modelo relacional | 60 |
| Relaciones..... | 61 |
| Normalización..... | 61 |
| Diagrama de clases | 62 |
| Tecnologías implementadas..... | 63 |
| Visual studio code..... | 63 |
| HTML | 63 |
| CSS..... | 64 |
| Bootstrap..... | 64 |
| Javascript..... | 65 |
| Jquery..... | 65 |
| PHP..... | 66 |
| MYSQL..... | 66 |
| phpMyAdmin | 67 |
| GitHub..... | 67 |
| Dificultades | 68 |
| Obtención de animes por la API | 68 |
| Obtención de resultados y ver más con AJAX..... | 68 |
| Búsqueda y filtros con AJAX | 68 |
| Arquitectura MVC..... | 69 |

| | |
|--------------------------------|----|
| Requisitos mínimos..... | 70 |
| Entorno Cliente | 70 |
| Aclaración..... | 70 |
| aceptarCookie.js..... | 71 |
| ajaxBuscador.js..... | 72 |
| ajaxVerMas.js..... | 75 |
| accederAnime.js | 77 |
| validarInput.js | 79 |
| hamburguerMenu.js | 83 |
| obtenerURL.js..... | 83 |
| opcionesAdmin.js..... | 83 |
| opcionesListas.js | 84 |
| Base de datos..... | 85 |
| Consulta JOIN | 85 |
| Consulta con subconsulta | 85 |
| Entorno servidor | 86 |
| Aclaración..... | 86 |
| Futuras actualizaciones..... | 88 |
| Conclusión..... | 88 |

Introducción

Se ha decidido que la aplicación web estará basada en animes, debido a la inmensa popularidad y aceptación que se ha tenido a lo largo de los años y como ha ido evolucionando incluso en varios países.

Lo bueno es la gran variedad que puede tener, desde videojuegos, hasta mangas, series, películas e incluso moda y eventos que se organizan de dicha temática. Una buena opción de ingresos sería hacer publicidad y difundir por distintas redes sociales, también con la finalidad de atraer a gente nueva a este tipo de contenido.

Definición del Producto/Servicio

El servicio que se ofrece se basa en una aplicación web que ofrece una visualización de animes junto con su información a través de una API, facilitándole al usuario la búsqueda gracias a la implementación de un buscador y filtros creados con Ajax, ofreciéndole así al usuario una experiencia óptima; además, ofrece la capacidad de crear usuarios, permitiendo guardar los animes en listas que el propio usuario puede crear y permite también tanto compartir como comentar un anime.

Alternativas

Kitsu

Ventajas: Ofrece una buena cantidad de información y una sección donde la gente puede publicar y comentar contenido.

Desventajas: las listas donde se pueden guardar son solo 3 “completado”, “quiero verlo” y “empezado”, tampoco ofrece la capacidad de compartir.

Animedatos

Ventajas: Ofrece una sección donde la gente puede poner comentarios con respuestas.

Desventajas: Carece de creación de usuarios, capacidad de compartir animes y creación de listas.

Análisis DAFO

- Fortalezas (internas)
 - Amplia información de animes de varios géneros y tipos
 - Interacción sencilla entre el usuario y la web
 - Compartir animes, permitiendo atraer a más personas
 - Comentar animes, creando así una comunidad activa
 - Creación de Listas y me gusta
- Debilidades (internas)
 - Limitada a un público específico.
 - Dependencia con la API para la obtención de Animes.
 - Limitación a la hora de actualizar los animes al ser la primera versión.
- Oportunidades (externas)
 - Integración con redes sociales
 - Monetización a través de publicidad
 - Añadir otros idiomas
 - Implementación de actualizaciones en la información que se obtiene de la API y futuras opciones nuevas a añadir
- Amenazas (externas)
 - Competencia con plataformas con mayor reconocimiento
 - Posibles cambios en tecnologías que requieran modificar poco o mucho contenido implementado

Requisitos Funcionales

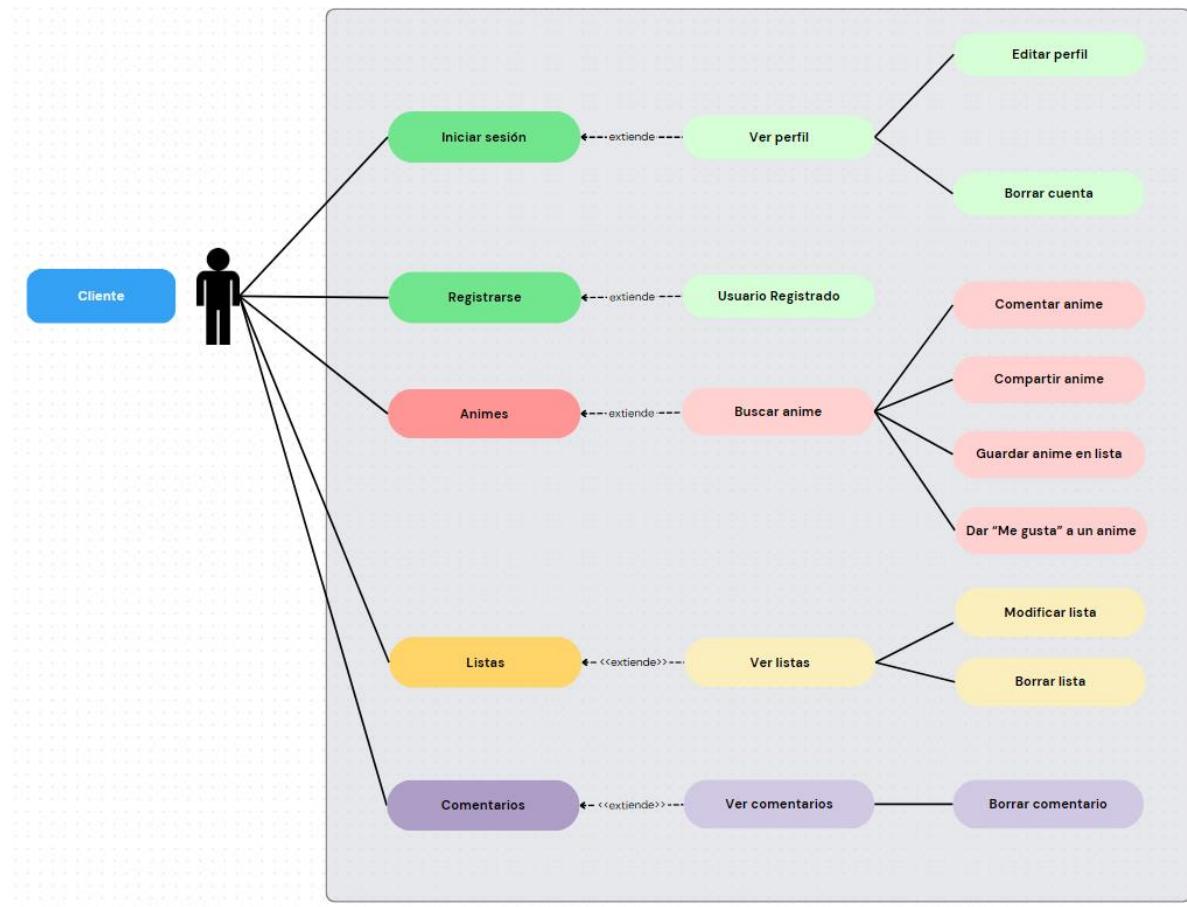
- RF-1: Gestión de Usuario
 - RF-1.1 Crear usuario
 - RF-1.2 Modificar Usuario
 - RF-1.3 Borrar Usuario
- RF-2: Gestión de animes
 - RF-2.1 Acceder al anime
 - RF-2.2 Buscador de animes
 - RF-2.3 Filtros de búsqueda
 - RF-2.4 Añadir anime a lista
 - RF-2.5 Compartir anime
 - RF-2.6 Comentar anime
 - RF-2.7 Borrar comentario
 - RF-2.8 Dar me gusta a un anime
- RF-3: Gestión de Lista
 - RF-3.1 Crear lista
 - RF-3.2 Editar Lista
 - RF-3.3 Borrar Lista

Requisitos no Funcionales

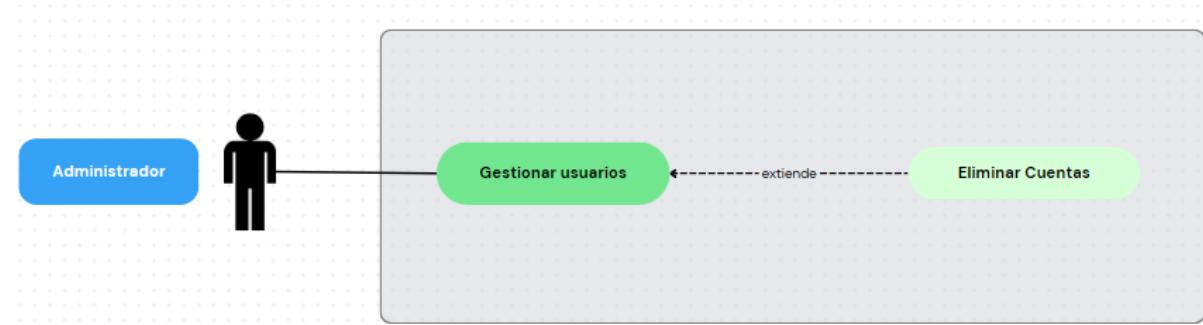
- RnF-1: Página Responsive
- RnF-2: Seguridad.
- RnF-3: Escalabilidad.
- RnF-4: Compatibilidad

Casos de uso

Usuario



Administrador



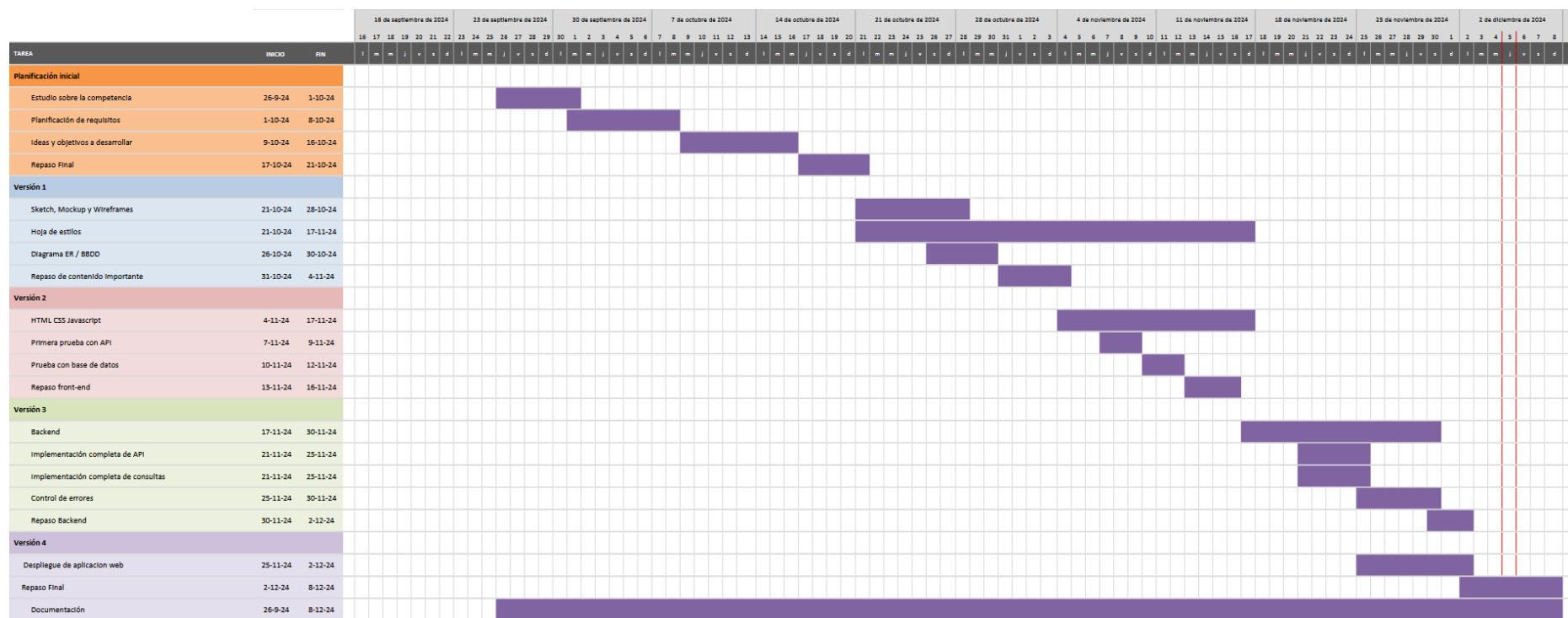
Problemas que soluciona respecto a otros proyectos

- Interacción Simple y eficaz: Evita distraer al usuario con demasiado contenido en pantalla, permitiendo así destacar lo más importante para el cliente.
- Capacidad de compartir y comentar: Los usuarios podrán compartir y comentar los animes que deseen.
- Buscador y filtros de búsqueda: Ofreciendo una búsqueda más concreta y fácil para el usuario
- Gestión de animes guardados: Gracias a la creación de listas y modificación de listas para el usuario

Objetivos

- Obtención correcta de animes a través de la API
- Creación y gestión de usuarios
- Mostrar información de animes
- Compartir animes
- Comentar animes
- Creación y gestión de listas
- Guardar animes en listas
- Buscador con Ajax y filtro para animes
- Dar me gusta a un anime

Diagrama de Gantt



Ciclo de desarrollo

Se ha optado por un ciclo ágil, permitiendo así realizar avances y poder hacer cambios y adaptar el desarrollo mientras voy avanzando para evitar con ello grandes errores a futuro.

Guía de estilos

Fuentes de colores

Colores Principales

| | | | | | | |
|------------------|-----------------------|-----------------|----------------------|-----------------------|-------------------------|------------------------|
| #000000 Black | F3F3F3 White smoke | FFFFFF White | 423A73 Delft Blue | 2D2850 Space cadet | FFC800 Mikado yellow | 53FB83 Spring green |
|------------------|-----------------------|-----------------|----------------------|-----------------------|-------------------------|------------------------|

#000000 - Black: Color de texto y sombras en la mayoría de los contenidos

#F3F3F3 - WhiteSmoke: Fondo en la mayoría de las secciones y divs ya que es un color neutro que combina bien con el tono blanco y el fondo morado oscuro.

#FFFFFF - White: Sombra de la mayoría de las secciones y divs.

#423A73 - Delft Blue: Color usado para header y footer

#2D2850 - Space Cadet: Color usado en los bodys de cada pagina

#FFC800 - Mikado Yellow: Color que se complementa bien con el morado usado para destacar botones e iconos importantes y animes con transiciones como ponerle una sombra de ese color.

#53FB83 - Spring Green: Color usado solo y exclusivamente para diferenciar a un admin de un cliente con un div con el rol en el header.

Colores secundarios

| | | | | | | | |
|----------------------|----------------------------------|----------------------------------|-------------------------------|----------------------------|-----------------------------|---------------------------|-------------------------------|
| 333333 Jet | 888888 Battleship gray | 3D642A Dark moss green | 135987 Lapis Lazuli | 2E729F UCLA Blue | AB0C00 Turkey red | 760800 Barn red | F481C0 Persian pink |
|----------------------|----------------------------------|----------------------------------|-------------------------------|----------------------------|-----------------------------|---------------------------|-------------------------------|

#333333 – Jet: Color usado para el header del section del login y el header.

#888888 – Battleship gray: Color usado para las sombras de las portadas de los animes en el index, listas y me gusta del usuario.

#3D642A – Dark moss green: Usado para indicar que un anime está en emisión.

#135987 Lapis Lazuli: Color usado para la mayoría de los botones que sean para aceptar, añadir o actualizar algo.

#2E729F – UCLA Blue: Color usado para el botón de cerrar sesión para así diferenciarlo del color “lapis lazuli”.

#760800 – Barn red: Color usado para el botón “Borrar Cuenta”.

#AB0C00 – Turkey red: Color usado para la mayoría de los botones que sean para borrar.

Colores hover

| | | | |
|-------------------------------|----------------------------|-------------------------------------|-------------------------------|
| 0862A1 Lapis Lazuli | 2274AB UCLA Blue | D10F00 Engineering orange | FF7ABF Persian pink |
|-------------------------------|----------------------------|-------------------------------------|-------------------------------|

#0862A1 – Lapis Lazuli: Color usado como hover con el color secundario Lapis Lazuli

#2274AB – UCLA Blue: Color usado como hover con el color secundario UCLA Blue

#D10F00 – Engineering Orange: Color usado como hover con el color secundario Turkey Red

#FF7ABF – Persian Pink: Color usado como hover con el color secundario Persian Pink

Fuentes de texto

Chicle

Título de animes en detalles de animes y título de página.

Whereas disregard and contempt for human rights have resulted

Asap

Título de animes en el index y título en otros apartados.

**Whereas disregard and contempt for human rights
have resulted**

Lato

Texto para botones y texto que no son dentro de detalles de anime.

**Whereas disregard and contempt for human rights
have resulted**

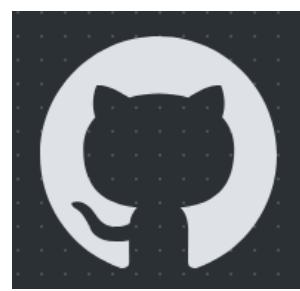
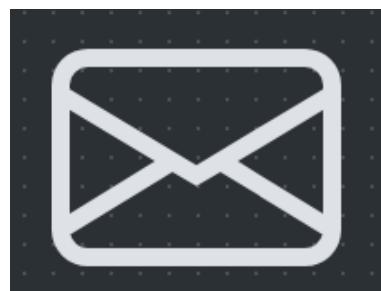
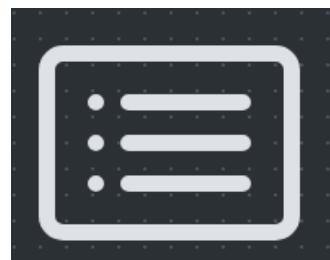
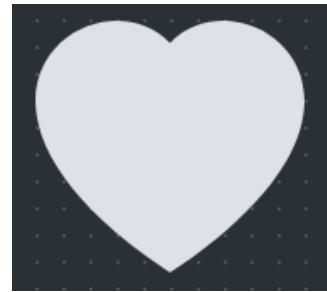
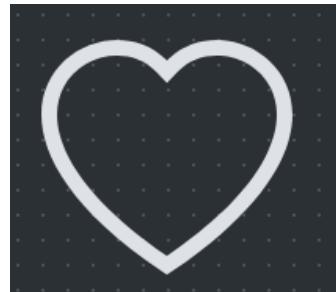
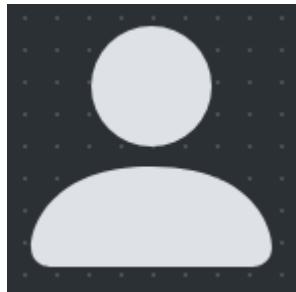
Be vietnam pro

Texto para detalles de anime.

**Whereas disregard and contempt for human
rights have resulted**

Iconos

Se han implementado los siguientes iconos obtenidos de [Bootstrap icons](#)



Wireframe

Desktop

HOME

Animelinfo

Login Sign Up

Search animes

Select last seen animes

Genre Format ID Ascent

Dragon Ball Z Dragon Ball Z Dragon Ball Z Dragon Ball Z Dragon Ball Z

Dragon Ball Z Dragon Ball Z Dragon Ball Z Dragon Ball Z Dragon Ball Z

Dragon Ball Z Dragon Ball Z Dragon Ball Z Dragon Ball Z Dragon Ball Z

See more

privacy policy © 2024 Carlos Atencia Ariza - Animelinfo



AnimelInfo

 Search animes

Select last seen animes

Genre

Format

ID Ascent



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z



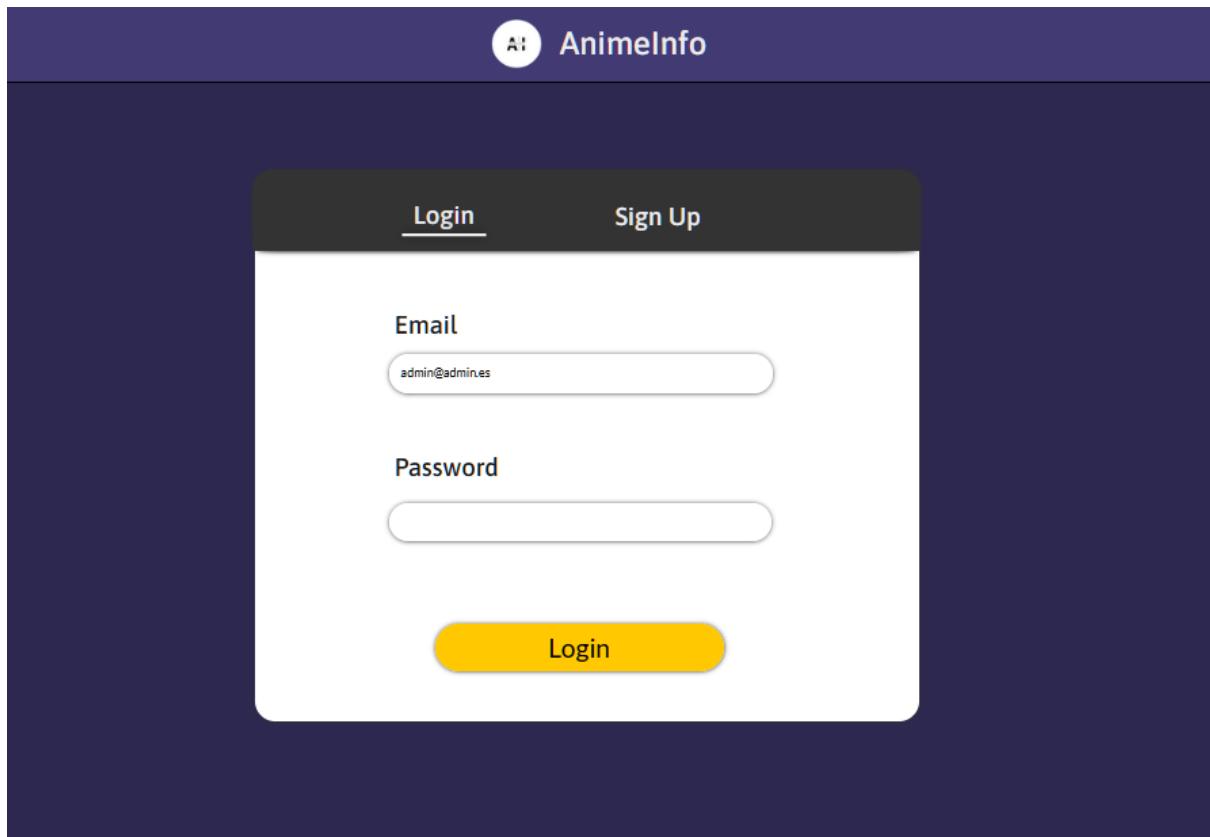
Dragon Ball Z

[See more](#)[privacy policy](#)

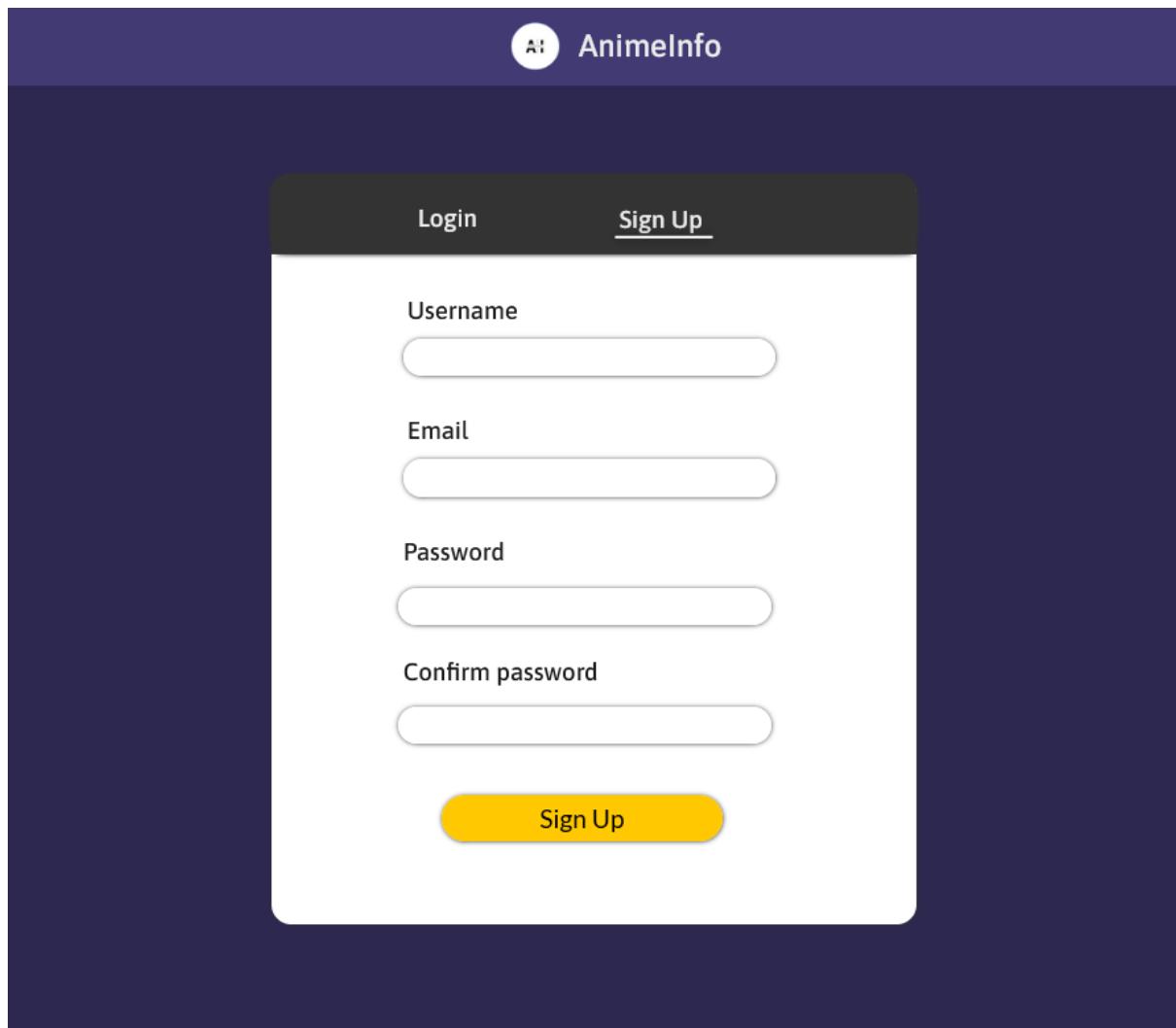
© 2024 Carlos Atencia Ariza - AnimelInfo



LOGIN



SIGN UP



**AnimelInfo**





Dragon Ball Z

[Copy URL](#)

Goku is back with his new son, Gohan, but just when things are getting settled down, the adventures continue. Whether he is facing enemies such as Freeza, Cell, or Boo, Goku is proven to be an elite of his own and discovers his race, Saiyan. He meets many new people, gaining allies as well as enemies, as he still finds time to raise a family and be the happy-go-lucky Saiyan he is.

RELEASING

[Save on list](#)

 271098

 See comments

- Format : TV
- Episodes : 300
- Genres : Action, Adventure, Comedy, Fantasy, Supernatural
- Start date : 1989-04-26
- End date : 1996-31-01

[privacy policy](#)

© 2024 Carlos Atencia Ariza - AnimelInfo



Página | 24



Dragon Ball Z

[Copy URL](#)

Goku is back with his new son, Gohan, but just when things are getting settled down, the adventures continue. Whether he is facing enemies such as Freeza, Cell, or Boo, Goku is proven to be an elite of his own and discovers his race, Saiyan. He meets many new people, gaining allies as well as enemies, as he still finds time to raise a family and be the happy-go-lucky Saiyan he is.

RELEASING

[Save on list](#)

271099

- **Format:** TV
- **Episodes:** 291
- **Genre :** Action, Adventure, Comedy, Fantasy, Supernatural
- **Start date:** 1989-04-26
- **End date:** 1996-31-01

See comments

[Post](#)

Anonimo 20-10-2020

[privacy policy](#)

© 2024 Carlos Atencia Ariza - AnimeInfo





RELEASING

Save on list

271098

▶ See comments

Dragon Ball Z

[Copy URL](#)

Goku is back with his new son, Gohan, but just when things are getting settled down, the adventures continue. Whether he is facing enemies such as Freeza, Cell, or Boo, Goku is proven to be an elite of his own and discovers his race, Saiyan. He meets many new people, gaining allies as well as enemies, as he still finds time to raise a family and be the happy-go-lucky Saiyan he is.

Choose the list in which you want to save the anime :

[Save](#)[Cancel](#)

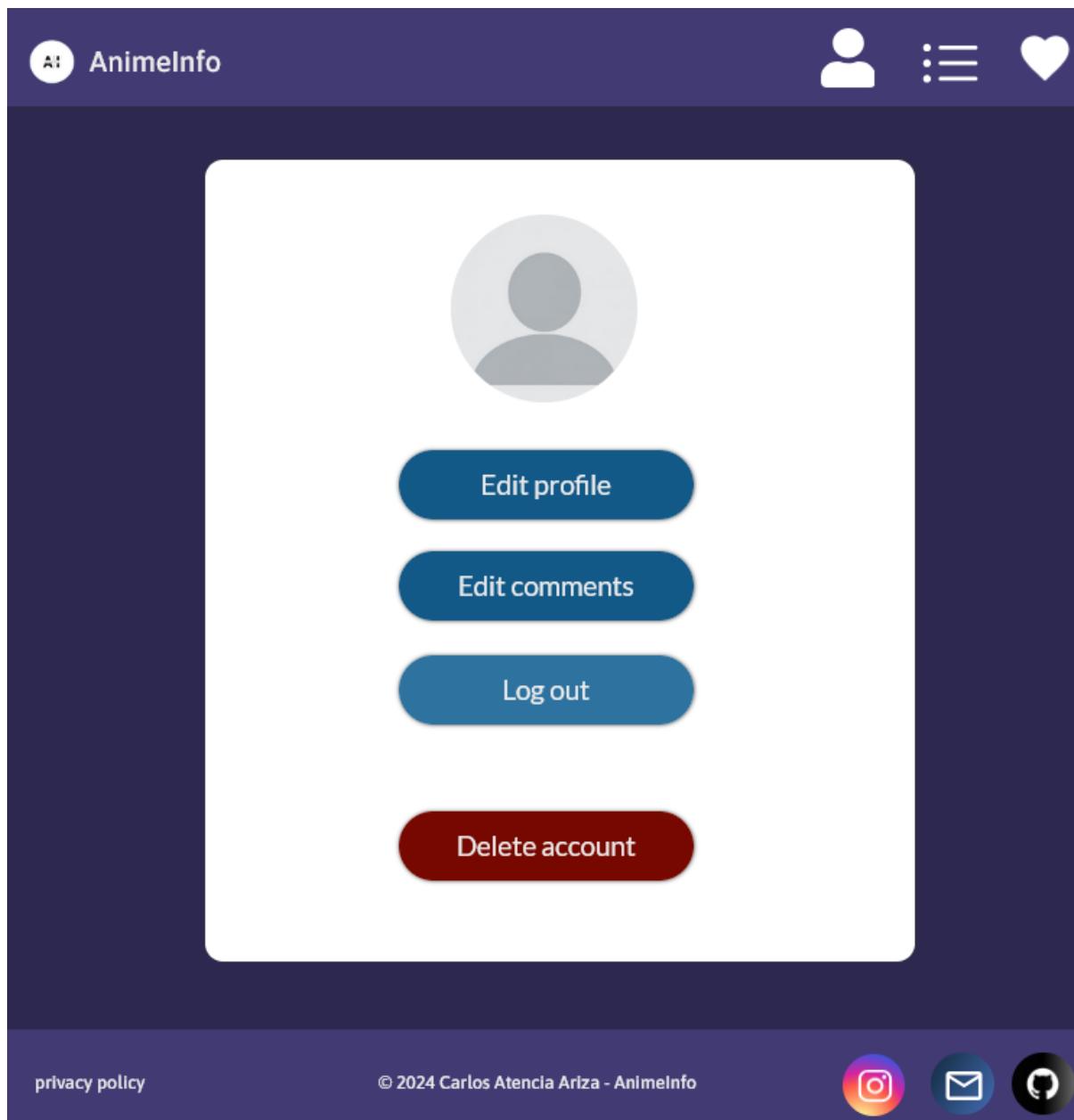
○ End date : 1996-31-01

[privacy policy](#)

© 2024 Carlos Atencia Ariza - AnimelInfo



PROFILE





¿Are you sure you want to delete your account?

Delete

Cancel

Edit comments

Log out

Delete account

[privacy policy](#)

© 2024 Carlos Atencia Ariza - AnimelInfo



UPDATE PROFILE

The screenshot shows the 'Update Profile' screen of the AnimelInfo app. At the top, there is a purple header bar with the 'AnimelInfo' logo on the left and three icons on the right: a user profile icon, a menu icon, and a heart icon. Below the header is a large white rectangular form area. On the left side of this area is a placeholder circular profile picture with a grey silhouette of a person. Below it is a blue rounded rectangular button labeled 'Upload photo'. To the right of the placeholder are three input fields: 'Username' (empty), 'Email' (empty), and 'Password' (empty). At the bottom of the form is a yellow rounded rectangular button labeled 'Update profile'.

privacy policy

© 2024 Carlos Atencia Ariza - AnimelInfo

UPDATE COMMENTS

The screenshot shows the AnimelInfo app interface. At the top, there's a purple header bar with the 'AnimelInfo' logo, a user profile icon, a menu icon, and a heart icon. Below the header, there are two separate comment sections for the anime 'Dragon Ball Z'. Each section includes a thumbnail image, the title 'Dragon Ball Z', a 'See comments' button, and a red '(datetime)' placeholder for a comment. The bottom of the screen features a dark footer bar with links for 'privacy policy', copyright information ('© 2024 Carlos Atencia Ariza - AnimelInfo'), and social media icons for Instagram, Email, and a circular profile.

Comments

Dragon Ball Z

See comments

(datetime)

Delete

(datetime)

Delete

privacy policy

© 2024 Carlos Atencia Ariza - AnimelInfo

LISTS

The screenshot shows the 'Lists' screen of the AnimeInfo app. At the top, there is a header with the 'AnimeInfo' logo and three icons: a user profile, a menu, and a heart. Below the header, the word 'Lists' is displayed in bold. To the right of 'Lists' is a blue button labeled 'Create list'. The main content area is a table with three rows, each representing a list. The columns are 'Title', 'Date creation', and 'Options'. The first row contains 'List 1' (created on 2024-10-10), the second row contains 'List 2' (created on 2024-09-07), and the third row contains 'List 3' (created on 2024-12-09). Each row has a set of three icons in the 'Options' column: a folder, a pencil, and a trash can. At the bottom of the screen, there are links for 'privacy policy' and '© 2024 Carlos Atencia Ariza - AnimeInfo', along with social media icons for Instagram, Email, and a dark circular icon.

| Title | Date creation | Options |
|--------|---------------|---------|
| List 1 | 2024-10-10 | |
| List 2 | 2024-09-07 | |
| List 3 | 2024-12-09 | |

privacy policy © 2024 Carlos Atencia Ariza - AnimeInfo



Lists

Title

List 1

List 2

List 3

Insert list name

Create

Cancel

Create list

Options

[privacy policy](#)

© 2024 Carlos Atencia Ariza - AnimelInfo





Lists

[Create list](#)

Title

List 1

Insert new list name

[Edit](#)[Cancel](#)

List 2

Options



List 3

2024-12-09

[privacy policy](#)

© 2024 Carlos Atencia Ariza - AnimelInfo





Lists

[Create list](#)

Title

Date creation

Options

List 1

Are you sure you want to remove this list?

[Delete](#)[Cancel](#)

List 2



List 3

2024-12-09

[privacy policy](#)

© 2024 Carlos Atencia Ariza - AnimeInfo



LIST

AI AnimeInfo

nameList

Dragon Ball Z

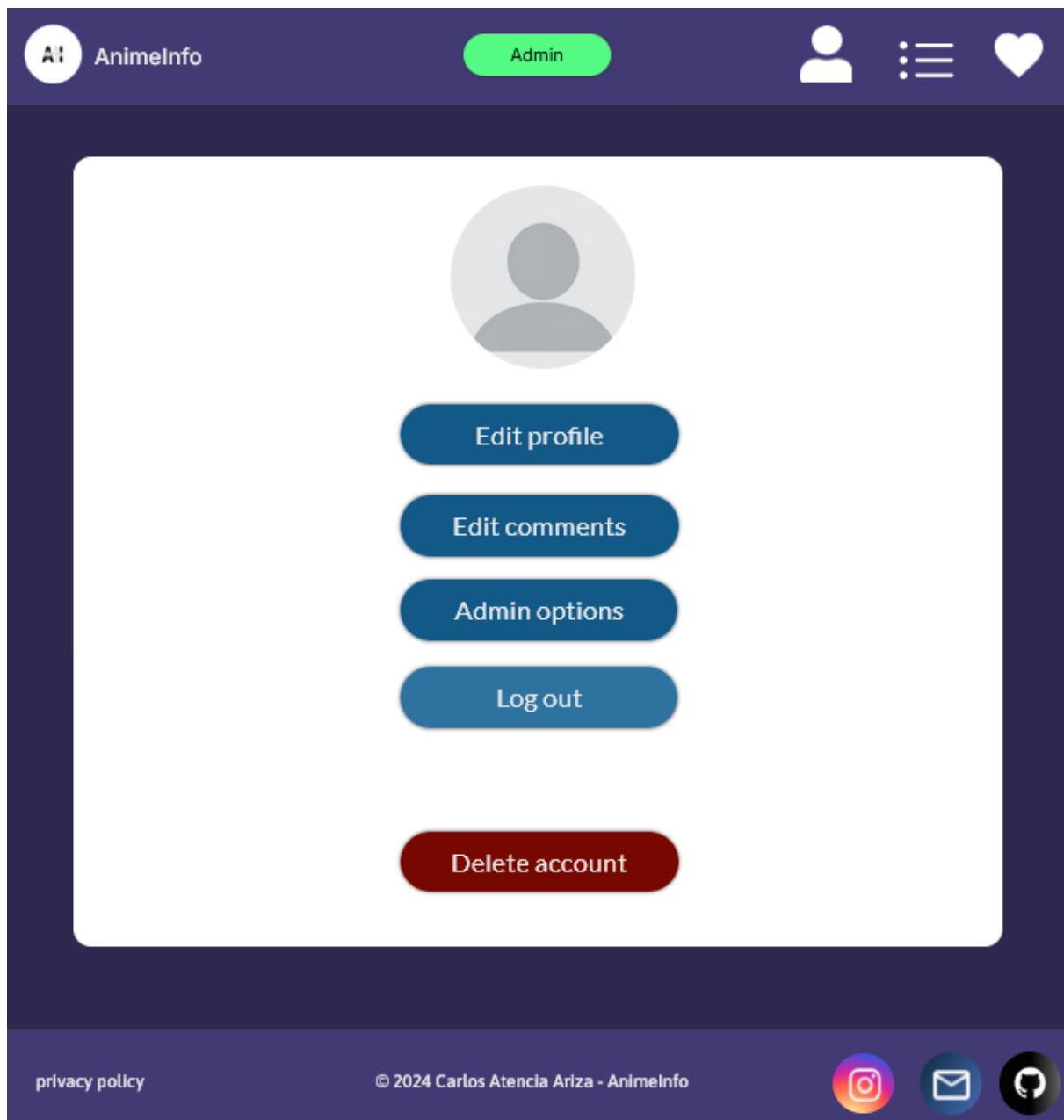
privacy policy

© 2024 Carlos Atencia Ariza - AnimeInfo

FAVORITES

The screenshot shows the AnimelInfo app interface. At the top left is the logo "AnimelInfo". At the top right are three icons: a user profile, a menu, and a heart. Below the header is a white button labeled "Favorites". The main area displays a 3x5 grid of 15 image thumbnails. Each thumbnail features a colorful illustration of the Dragon Ball Z characters standing together, with a red circular icon containing a white play button symbol in the top right corner. Below each thumbnail, the text "Dragon Ball Z" is visible. At the bottom of the screen, there are links for "privacy policy" and "© 2024 Carlos Atencia Ariza - AnimelInfo", along with social media icons for Instagram, Email, and WhatsApp.

ADMIN



OPCIONES ADMIN

The screenshot shows the AnimeInfo mobile application interface. At the top, there is a purple header bar with the "Admin" role indicator. Below the header, the main content area has a white background and a rounded rectangular border. The title "Clients" is centered at the top of this area. A table is displayed with three columns: "ID", "Username", and "Options". The "Options" column contains a red trash can icon for each row. The data in the table is as follows:

| ID | Username | Options |
|---------|----------|---------|
| d56s7f3 | carlos | |
| fs67wr3 | kkivan | |
| fhseui4 | xjorgex | |

At the bottom of the screen, there is a dark blue footer bar with the following elements from left to right: "privacy policy", the copyright notice "© 2024 Carlos Atencia Ariza - AnimeInfo", and three circular icons for social media or contact: Instagram, Email, and a profile icon.



Clientes

ID

Username

Options

d56s7f3

¿Are you sure delete this user?

Delete

Cancel

fs67wr3

fhseui4 xjorgex

[privacy policy](#)

© 2024 Carlos Atencia Ariza - AnimelInfo



PRIVACY POLICY

The screenshot shows the AnimeInfo mobile application interface. At the top, there is a dark blue header bar with the "AnimeInfo" logo on the left and three icons on the right: a user profile, a menu, and a heart. Below the header is a white content area with rounded corners. The title "PRIVACY POLICY" is centered at the top of this area. The content is organized into several sections with bold headings: "Introduction", "What information do we collect?", "Cookies and why we use them", "Why accept cookies?", "Your control over cookies", "How do we protect your information?", "Changes to this privacy policy", and "Contact us". Each section contains descriptive text and a small note at the bottom. At the very bottom of the screen, there is a dark blue footer bar with the text "privacy policy" on the left, the copyright notice "© 2024 Carlos Atencia Ariza - AnimeInfo" in the center, and three circular icons on the right representing social media links (Instagram, Email, and a generic icon).

PRIVACY POLICY

Introduction

At AnimeInfo, we take your privacy seriously. This Privacy Policy explains how we collect, use, and protect your personal information when you visit our website. By using our services, you agree to the terms of this policy.

What information do we collect?

We may collect personal information such as your id, role, and browsing data, including cookies, to improve your experience on our website. This helps us provide personalized content and ensure the functionality of certain features.

Cookies and why we use them

Cookies are small files stored on your device that allow us to remember your preferences. By accepting cookies, you help us offer you lost animes visited, enhance your user experience in login and sign up, and improve the performance of our site.

Why accept cookies?

- **Personalized Experience:** By accepting cookies, we can remember your preferences and modify backgrounds in login and sign up, so you don't have to re-enter them each time you visit our site.
- **Recent Anime History:** We use cookies to store the last 5 animes you have viewed, so you can easily revisit them at any time.
- **Website Analytics:** Cookies help us understand how visitors interact with our site.

Your control over cookies

While cookies enhance your experience, you have full control over them. You can accept or decline cookies in your browser. However, please note that disabling cookies may affect some features on our website and limit your browsing experience.

How do we protect your information?

We take appropriate measures to safeguard your personal information. Your data is stored securely.

Changes to this privacy policy

We may update this Privacy Policy from time to time. Any changes will be posted.

Contact us

If you have any questions or concerns about this Privacy Policy or how we handle your personal information, feel free to contact us at this [email](#).

Mobile

HOME

AI

Animelinfo

≡

Search animes

Select last seen animes

Genre Format ID Ascent



Dragon Ball Z



Dragon Ball Z



Dragon Ball Z

See more

privacy policy

© 2024 Carlos Atencia Ariza - Animelinfo

AI

Animelinfo

X

Search animes

Select last seen animes

Genre Format

Login

Sign Up



Dragon



Dragon



Dragon

See n

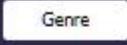
privacy policy

© 2024 Carlos Atencia Ariza - Animelinfo

 AnimelInfo 

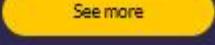



Dragon Ball Z


Dragon Ball Z


Dragon Ball Z



[privacy policy](#) © 2024 Carlos Atencia Ariza - AnimelInfo 

 AnimelInfo 



 Home
 Profile
 Lists
 Favorites


Dragon Ball Z

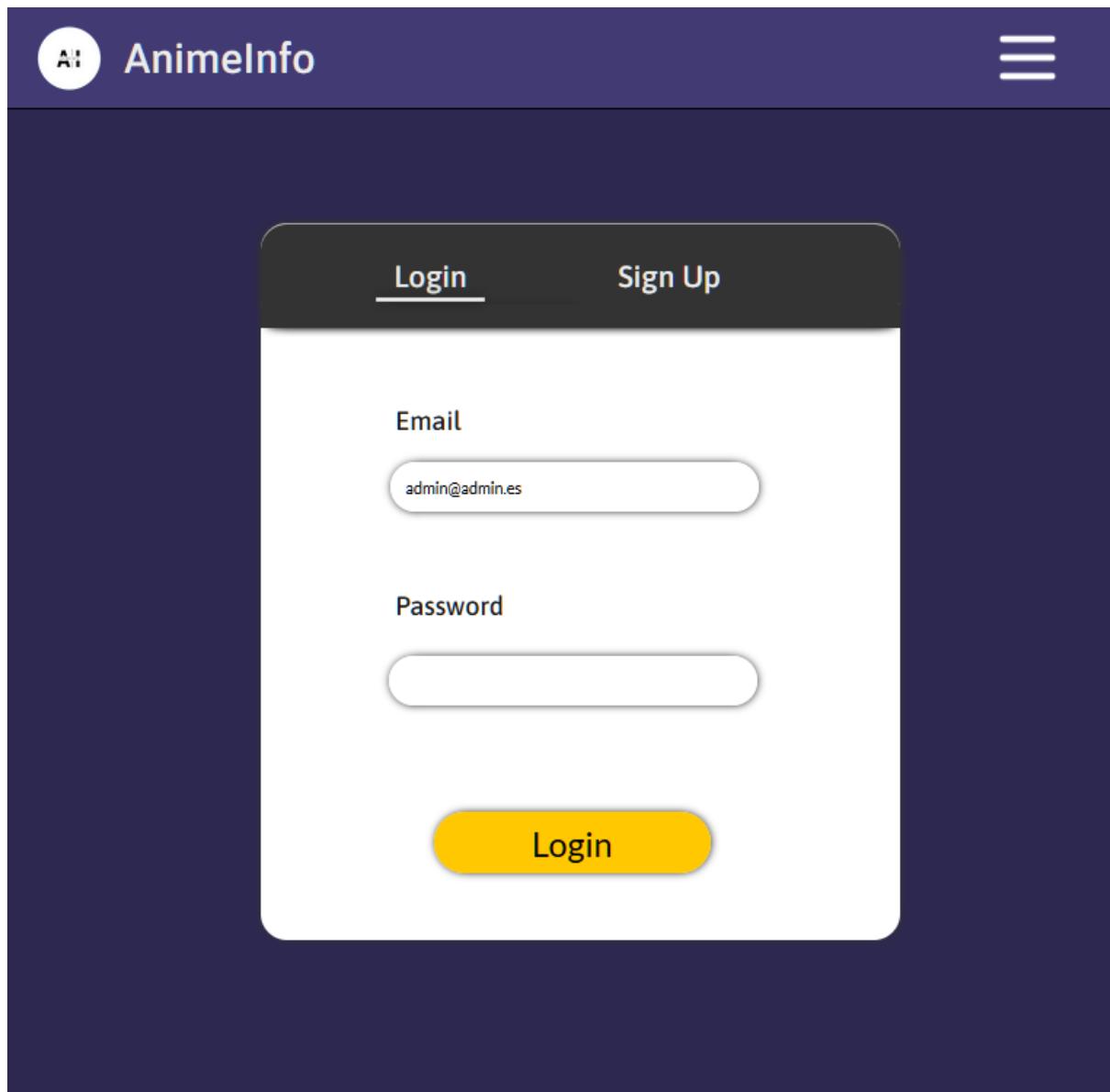

Dragon Ball Z


Dragon Ball Z

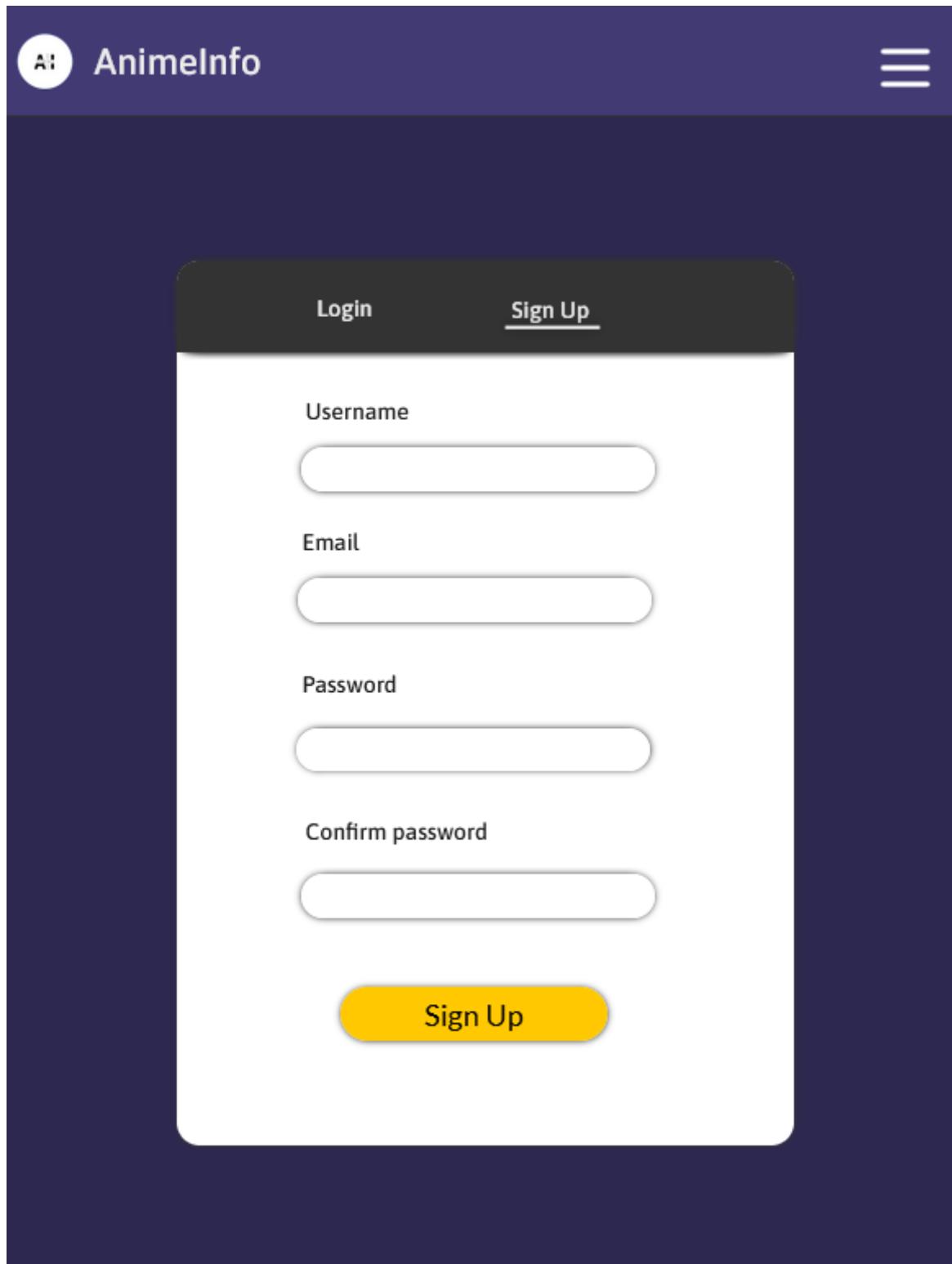


[privacy policy](#) © 2024 Carlos Atencia Ariza - AnimelInfo 

LOGIN



SIGN UP



ANIME

AI AnimelInfo



Copy URL

Dragon Ball Z

RELEASING

Save on list

271098

Goku is back with his new son, Gohan, but just when things are getting settled down, the adventures continue. Whether he is facing enemies such as Freeza, Cell, or Boo, Goku is proven to be an elite of his own and discovers his race, Saiyan. He meets many new people, gaining allies as well as enemies, as he still finds time to raise a family and be the happy-go-lucky Saiyan he is.

- Format: TV
- Episodes: 291
- Genre: Action, Adventure, Comedy, Fantasy, Supernatural
- Start date: 1989-04-26
- End date: 1996-31-01

See comments

privacy policy

© 2024 Carlos Atencia Ariza - AnimelInfo

AI Animelinfo



Copy URL

Dragon Ball Z

FINISHED

Save on list

271099

Goku is back with his new son, Gohan, but just when things are getting settled down, the adventures continue. Whether he is facing enemies such as Freeza, Cell, or Boo, Goku is proven to be an elite of his own and discovers his race, Saiyan. He meets many new people, gaining allies as well as enemies, as he still finds time to raise a family and be the happy-go-lucky Saiyan he is.

Format: TV
Episodes: 291
Genre: Action, Adventure, Comedy, Fantasy, Supernatural
Start date: 1989-04-26
End date: 1996-31-01

See comments

Anonimo 2020-10-20

privacy policy

AI AnimeInfo

Copiar URL

Choose the list in which you want to save the anime :

Save Cancel

Goku is back with his new son, Gohan, but just when things are getting settled down, the adventures continue. Whether he is facing enemies such as Freeza, Cell, or Boo, Goku is proven to be an elite of his own and discovers his race, Saiyan. He meets many new people, gaining allies as well as enemies, as he still finds time to raise a family and be the happy-go-lucky Saiyan he is.

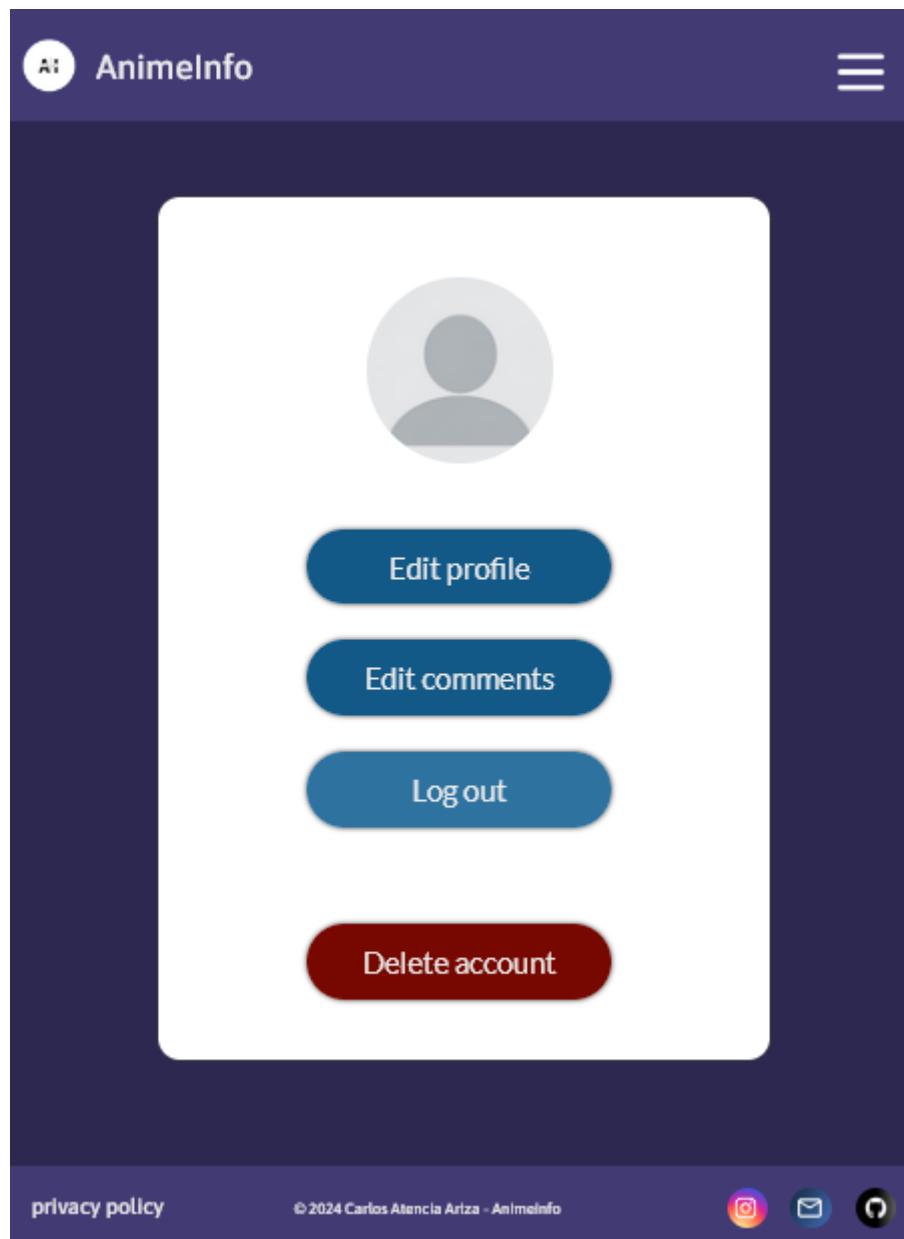
- Format : TV
- Episodes : 291
- Genre : Action, Adventure, Comedy, Fantasy, Supernatural
- Start date : 1989-04-26
- End date : 1996-31-01

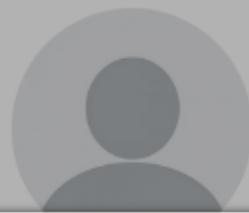
▶ See comments

privacy policy

© 2024 Carlos Atencia Ariza - AnimeInfo

PROFILE





Are you sure you want to delete your account?

Delete

Cancel

Edit comments

Log out

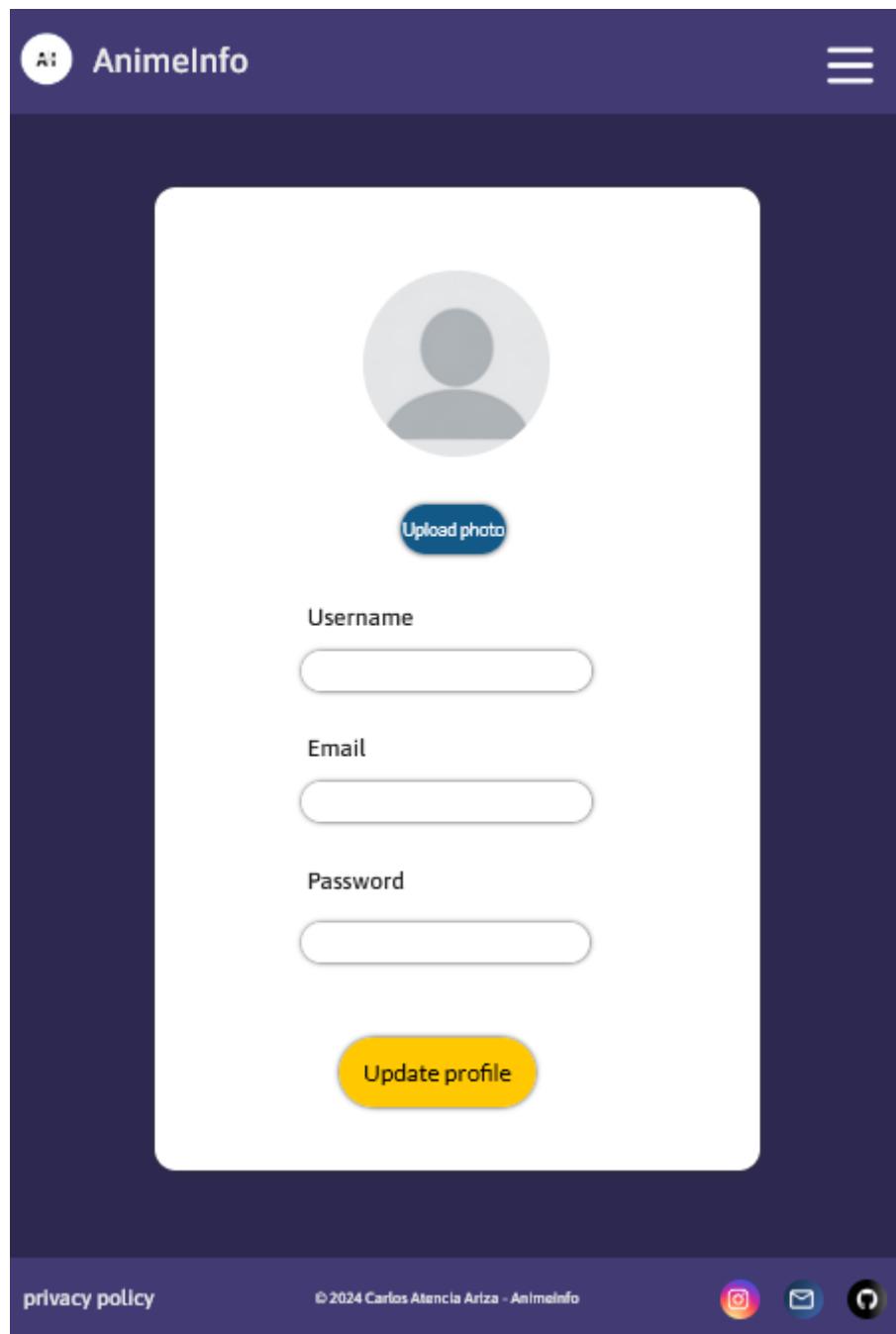
Delete account

[privacy policy](#)

© 2024 Carlos Atencia Ariza - Animeinfo



EDIT PROFILE



EDIT COMMENTS

AI AnimeInfo

Comments



Dragon Ball Z

▶ See comments



Dragon Ball Z

▼ See comments

(datetime)

Delete

privacy policy

© 2024 Carlos Alenda Ariza - AnimeInfo



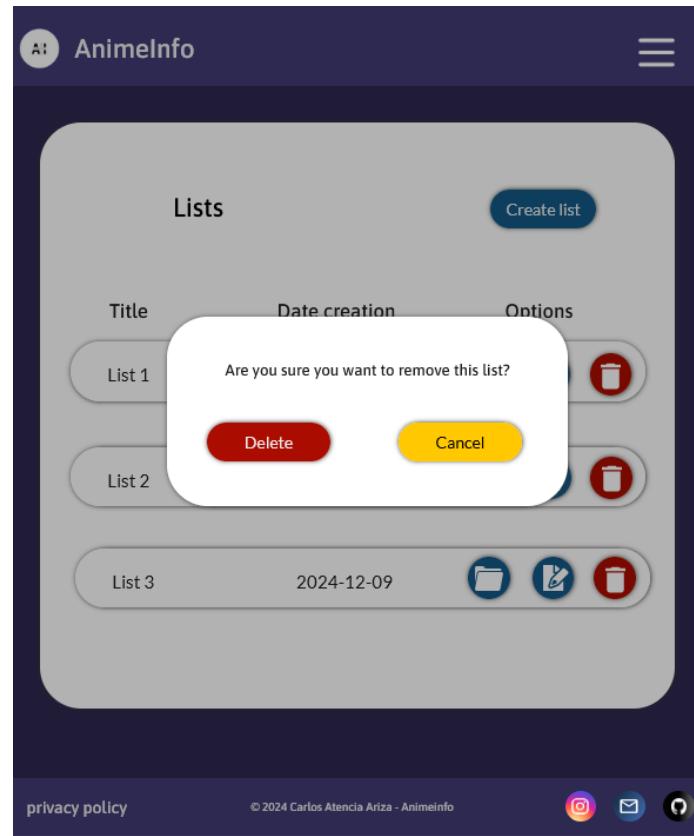
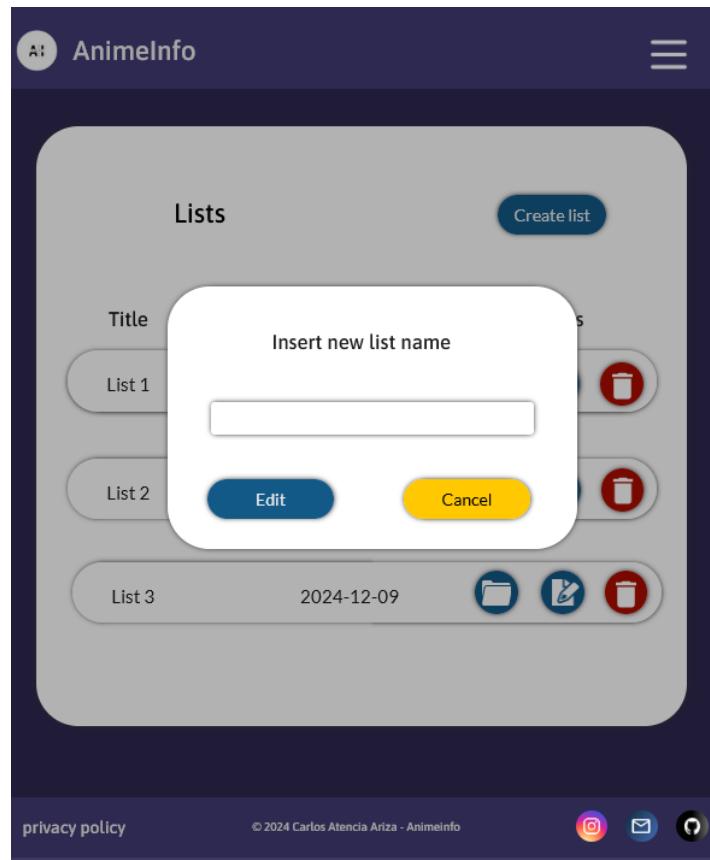
LISTS

The screenshot shows the 'Lists' screen of the AnimeInfo app. At the top, there is a header with the 'AnimelInfo' logo and a three-line menu icon. Below the header, the word 'Lists' is centered above a button labeled 'Create list'. A table displays three lists:

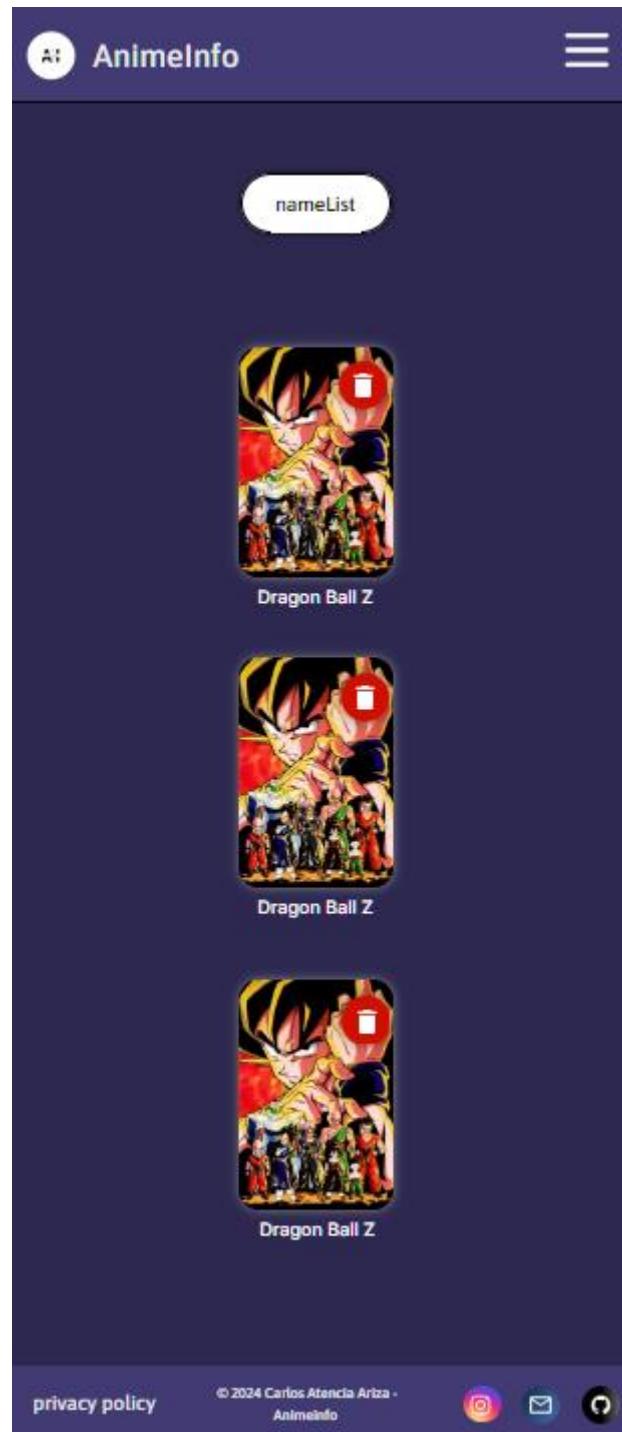
| Title | Date creation | Options |
|--------|---------------|---------|
| List 1 | 2024-10-10 | |
| List 2 | 2024-09-07 | |
| List 3 | 2024-12-09 | |

At the bottom of the screen, there are links for 'privacy policy' and '© 2024 Carlos Atencia Ariza - Animeinfo', along with social media icons for Instagram, Email, and GitHub.

The screenshot shows a modal dialog box titled 'Insert list name' overlaid on the 'Lists' screen. The modal contains a text input field labeled 'Title' and two buttons: 'Create' (blue) and 'Cancel' (yellow). In the background, the list items from the previous screenshot are visible: List 1 (2024-10-10), List 2 (2024-09-07), and List 3 (2024-12-09). The bottom navigation bar with 'privacy policy', copyright information, and social media links is also present.



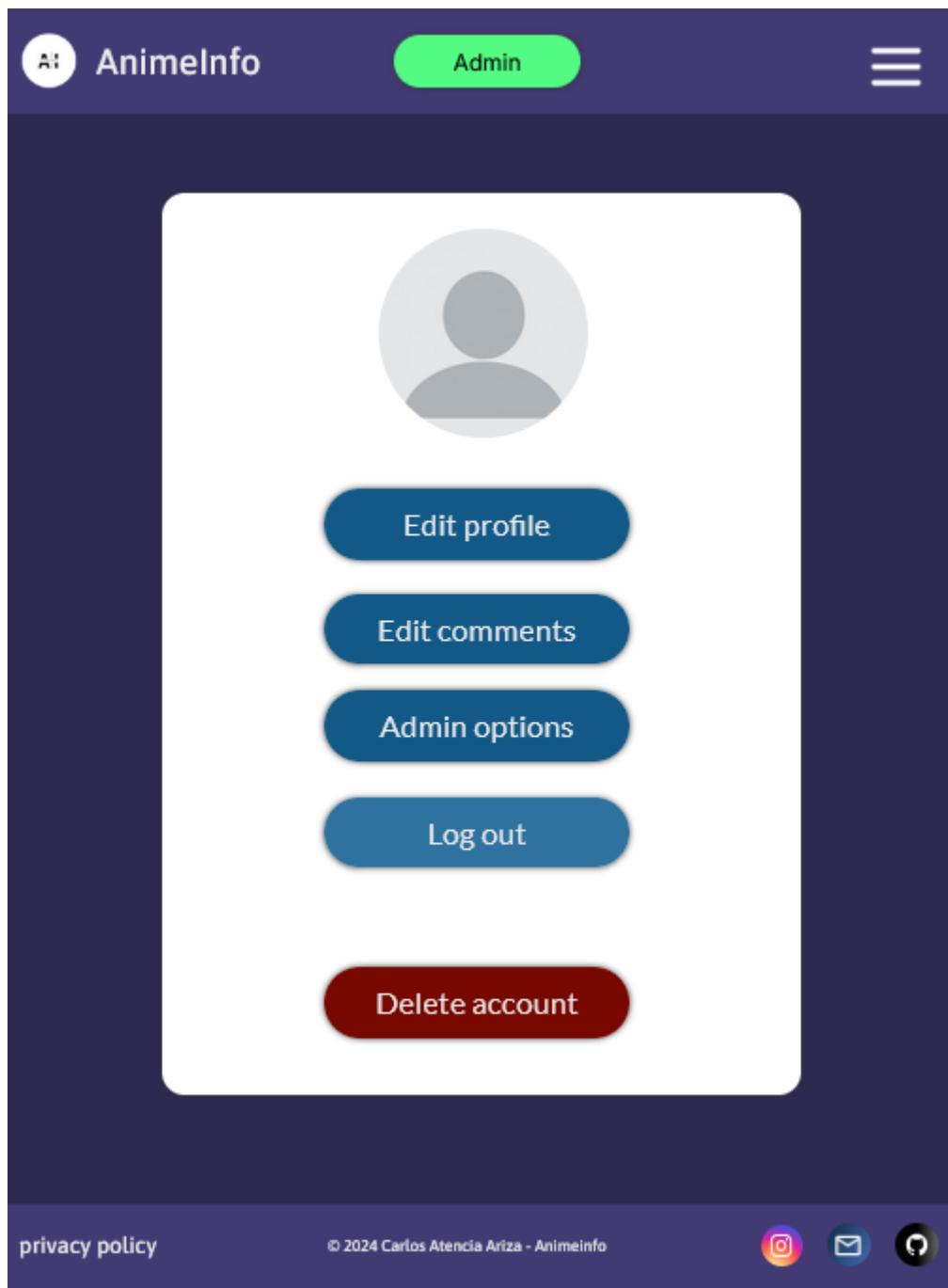
LIST



FAVORITES

The screenshot shows the AnimelInfo mobile application's Favorites screen. At the top, there is a header with the logo "AnimelInfo" and a menu icon. Below the header, a button labeled "Favorites" is visible. The main content area displays three items, each consisting of a thumbnail image and the text "Dragon Ball Z". Each thumbnail features a red circular icon with a white trash symbol, indicating a delete function. The thumbnails appear to be identical, showing a scene from Dragon Ball Z. At the bottom of the screen, there is a footer bar containing links for "privacy policy", copyright information ("© 2024 Carlos Atencia Ariza - AnimelInfo"), and social media icons for Instagram, Email, and a circular profile icon.

ADMIN

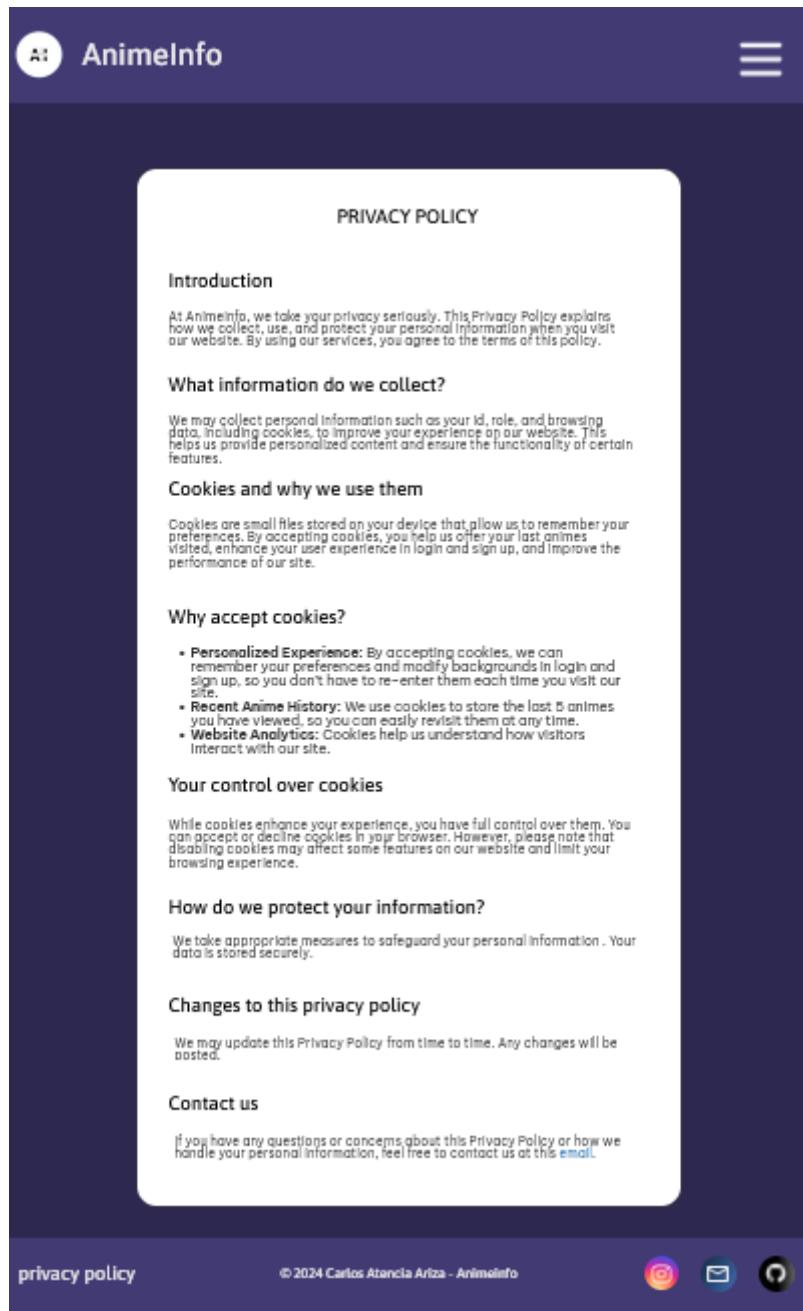


OPCIONES ADMIN

The screenshot shows the Admin section of the AnimeInfo application. At the top, there's a navigation bar with the 'Admin' button highlighted. Below it, a large white rounded rectangle contains the heading 'Clients' and a table with three rows. The table has columns for 'ID', 'Nombre', and 'Opciones'. Each row contains an ID (d56s7f3, fs67wr3, fhseui4), a name (carlos, kkivan, xjorgex), and a red trash can icon in the 'Opciones' column. At the bottom of the screen, there are links for 'privacy policy' and '© 2024 Carlos Atencia Ariza - AnimeInfo', along with social media icons for Instagram, Email, and GitHub.

This screenshot shows the same Admin interface as the previous one, but with a modal dialog box overlaid on the 'Clients' table. The dialog box contains the text '¿Are you sure delete this user?' and two buttons: 'Delete' (red) and 'Cancel' (yellow). The background of the table rows is grayed out to indicate they are not interactive while the dialog is open.

PRIVACY POLICY



The screenshot shows the 'PRIVACY POLICY' page from the AnimeInfo website. The page has a dark blue header with the 'AnimeInfo' logo and a menu icon. The main content area is white with a rounded bottom. At the top of the content area, it says 'PRIVACY POLICY'. Below that is a section titled 'Introduction' with a paragraph of text. Following that is a section titled 'What information do we collect?' with a paragraph of text. Next is a section titled 'Cookies and why we use them' with a paragraph of text. Then there's a section titled 'Why accept cookies?' with a bulleted list of three reasons: 'Personalized Experience', 'Recent Anime History', and 'Website Analytics'. Below that is a section titled 'Your control over cookies' with a paragraph of text. Underneath that is a section titled 'How do we protect your information?' with a paragraph of text. At the bottom of the content area, there's a section titled 'Changes to this privacy policy' with a paragraph of text. Finally, at the very bottom, there's a section titled 'Contact us' with a paragraph of text.

PRIVACY POLICY

Introduction

At AnimeInfo, we take your privacy seriously. This Privacy Policy explains how we collect, use, and protect your personal information when you visit our website. By using our services, you agree to the terms of this policy.

What information do we collect?

We may collect personal information such as your ID, role, and browsing data, including cookies, to improve your experience on our website. This helps us provide personalized content and ensure the functionality of certain features.

Cookies and why we use them

Cookies are small files stored on your device that allow us to remember your preferences. By accepting cookies, you help us offer you last animes visited, enhance your user experience in login and sign up, and improve the performance of our site.

Why accept cookies?

- Personalized Experience:** By accepting cookies, we can remember your preferences and modify backgrounds in login and sign up, so you don't have to re-enter them each time you visit our site.
- Recent Anime History:** We use cookies to store the last 5 animes you have viewed, so you can easily revisit them at any time.
- Website Analytics:** Cookies help us understand how visitors interact with our site.

Your control over cookies

While cookies enhance your experience, you have full control over them. You can accept or decline cookies in your browser. However, please note that disabling cookies may affect some features on our website and limit your browsing experience.

How do we protect your information?

We take appropriate measures to safeguard your personal information. Your details are stored securely.

Changes to this privacy policy

We may update this Privacy Policy from time to time. Any changes will be posted.

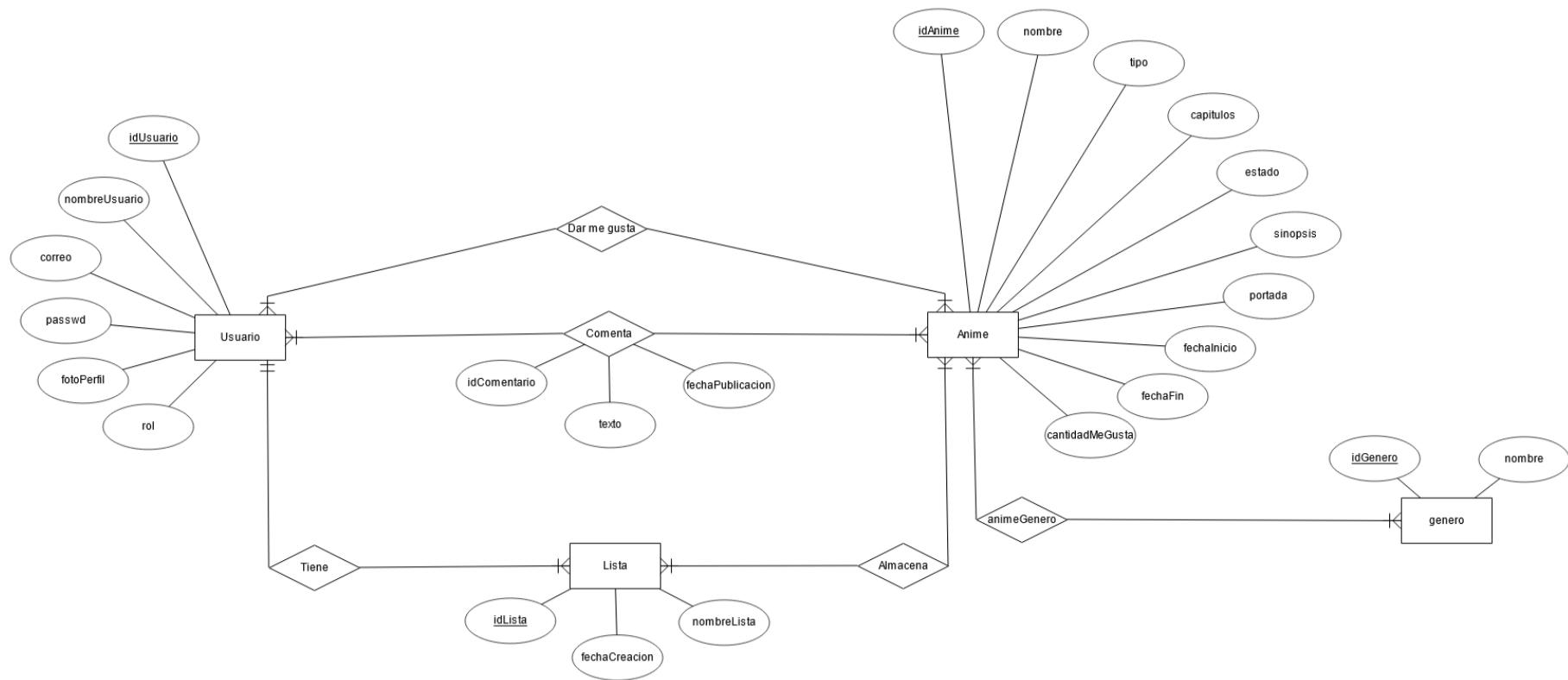
Contact us

If you have any questions or concerns about this Privacy Policy or how we handle your personal information, feel free to contact us at this [email](#).

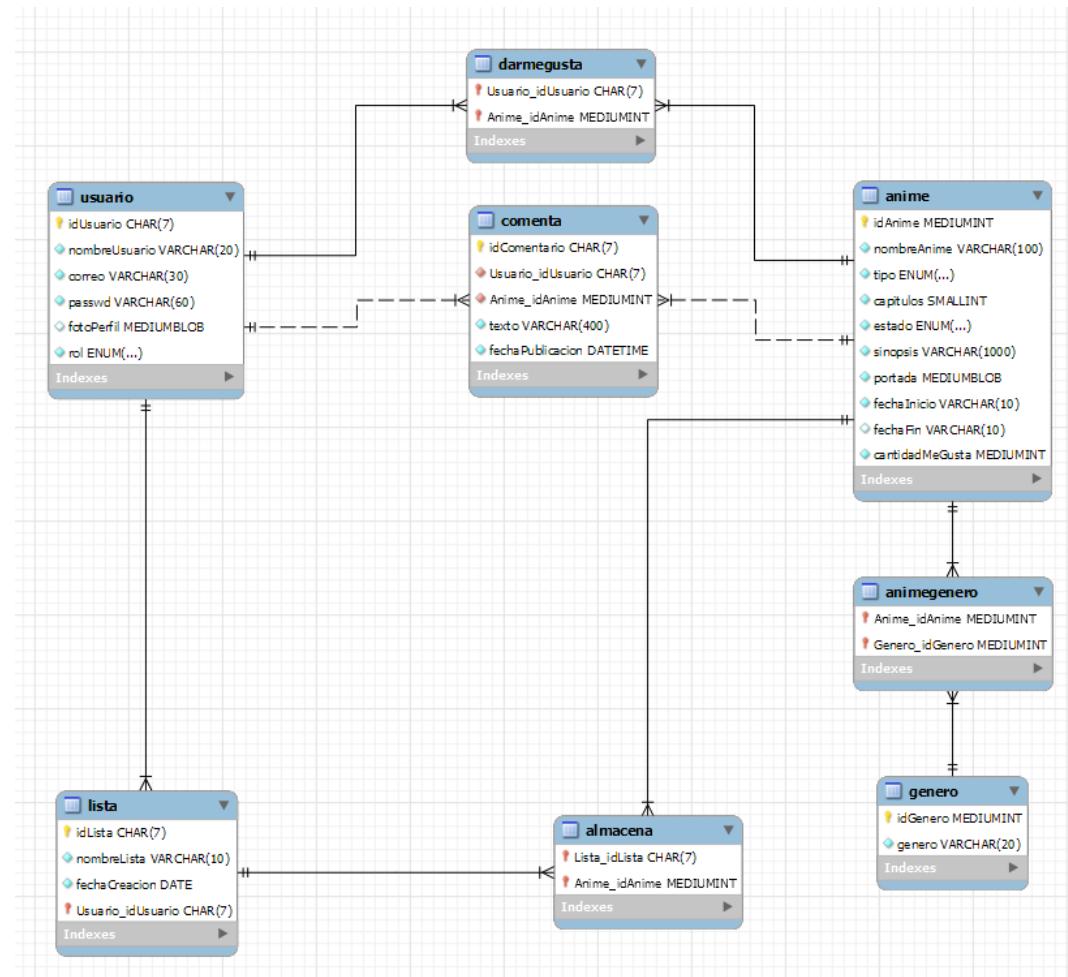
[privacy policy](#) © 2024 Carlos Abanca Ariza - AnimeInfo 

Base de datos

Entidad Relación



Modelo relacional



Relaciones

- **Relaciones 1:N**

- usuario -> lista (Un usuario puede tener múltiples listas).
- usuario -> comenta (Un usuario puede hacer múltiples comentarios).
- anime -> comenta (Un anime puede tener múltiples comentarios).

- **Relaciones N:M**

- lista - anime (“almacena” como tabla intermedia).
- anime - genero (“animegenero” como tabla intermedia).
- usuario - anime (“darmegusta” como tabla intermedia)

Todas las relaciones son identificativas excepto en la tabla comenta, debido a que tenemos un **idComentario** para identificar cada y no depende de las claves foráneas, se ha implementado así para identificar los distintos comentarios que pueda realizar un único usuario y/o varios comentarios en un único anime.

Normalización

La base de datos cumple con las tres formas normales por las siguientes razones:

1. Primera Forma Normal (1FN):

Todas las tablas tienen valores atómicos en sus columnas, es decir, cada campo tiene un solo valor.

2. Segunda Forma Normal (2FN):

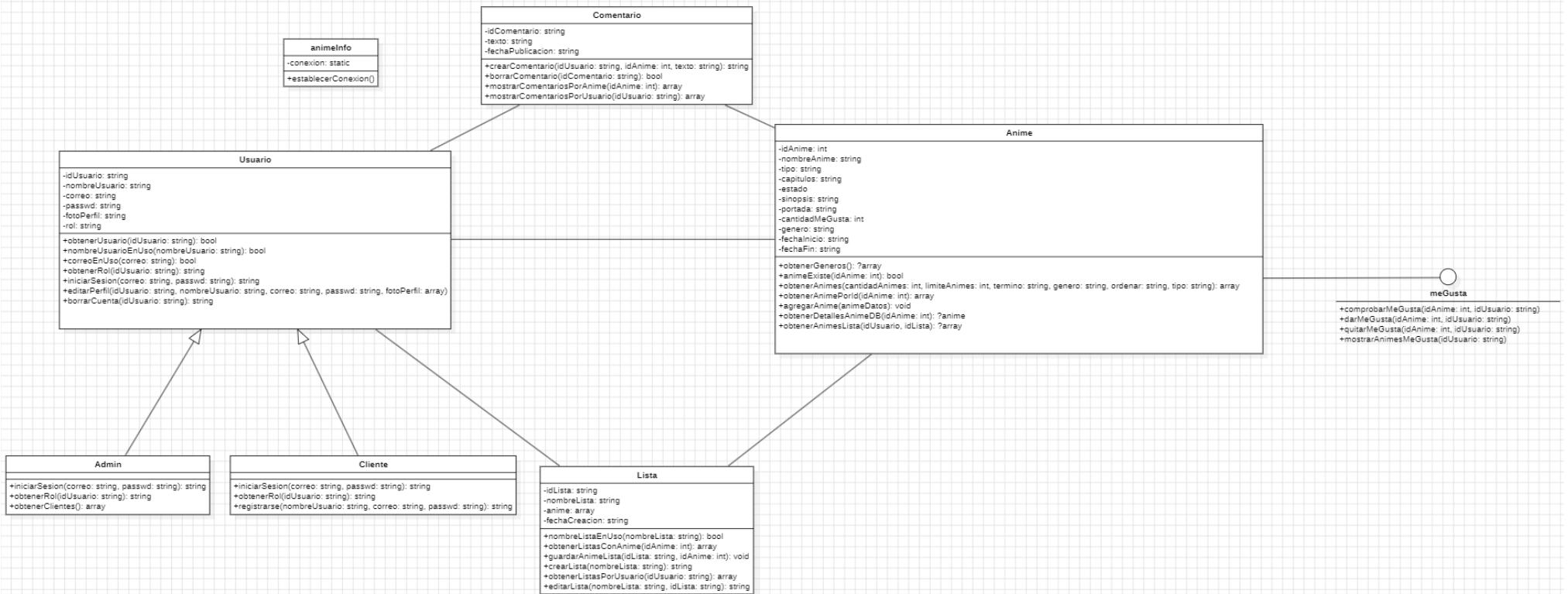
En las tablas con claves compuestas, todas las columnas dependen completamente de la clave primaria compuesta y no de una parte de ella.

3. Tercera Forma Normal (3FN):

No existen dependencias transitivas entre las columnas. Es decir, las columnas dependen solo de la clave primaria y no de otras columnas de manera indirecta.

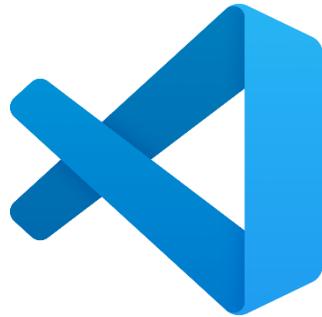
La base de datos está normalizada, asegurando la eficiencia, evitando redundancias y garantizando que las dependencias sean claras y adecuadas en cada tabla

Diagrama de clases



Tecnologías implementadas

Visual studio code



Un editor de código fuente ligero y altamente extensible desarrollado por Microsoft. Soporta múltiples lenguajes de programación y cuenta con características como depuración, control de versiones integrado y un amplio ecosistema de extensiones.

HTML



Lenguaje de marcado estándar para crear la estructura y contenido de páginas web, como encabezados, párrafos, enlaces, imágenes y formularios.

CSS



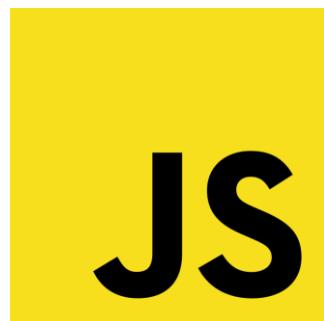
Lenguaje de diseño que controla la presentación visual de los elementos HTML. Permite definir estilos, colores, fuentes, márgenes, y diseños responsivos.

Bootstrap



Un framework de diseño CSS y JavaScript que facilita la creación de sitios web responsivos y con estilos modernos mediante el uso de componentes predefinidos como botones, formularios y barras de navegación.

Javascript



Lenguaje de programación dinámico que permite agregar interactividad y funcionalidad avanzada a las páginas web, como animaciones, validaciones de formularios y comunicación con servidores.

Jquery



Una biblioteca de JavaScript que simplifica la manipulación del DOM, el manejo de eventos, las animaciones y las solicitudes AJAX, reduciendo el código necesario.

PHP



Lenguaje de programación del lado del servidor diseñado para el desarrollo web. Permite crear aplicaciones dinámicas y conectarse con bases de datos para manejar contenido interactivo.

MYSQL



Sistema de gestión de bases de datos relacional (RDBMS) popular y de código abierto, ideal para almacenar y gestionar datos en aplicaciones web.

phpMyAdmin



Herramienta web que proporciona una interfaz gráfica para administrar bases de datos MySQL, permitiendo realizar tareas como crear tablas, insertar datos y ejecutar consultas SQL.

GitHub



Plataforma basada en Git para el control de versiones y la colaboración en proyectos de software. Permite a los desarrolladores trabajar en equipo, realizar seguimiento de cambios y alojar repositorios de código.

Dificultades

Obtención de animes por la API

He tenido varios problemas como por ejemplo los rankings, debido a que puede haber 2 categorías los cuales son mayor valorado o más popular.

A la hora de obtener esa información es un array, pero el orden en el que salía el mayor valorado o más podía ser distinto, además de que podía ser por ejemplo más popular del 2004 o más popular de verano, lo cual complica un poco su obtención y muestra de este en la API.

También con las fechas ya que hay algunas que no tienen fecha de fin y otros que sí, tuve que almacenarlos como **varchar** en la base de datos para mostrar solo año y mes o año, mes y día o nada.

Obtención de resultados y ver más con AJAX

He tenido varios problemas como que salían duplicados, o no salían, o los obtenía de mala manera, considero que ha sido de las cosas más complejas durante el proceso en términos de implementar contenido.

Búsqueda y filtros con AJAX

Además de tener varios problemas como comenté anteriormente, con las búsquedas y los filtros tenía problemas como que realiza doble o triple petición a la vez, lo cual llegaba al límite de peticiones en poco tiempo, limitando la interacción del usuario con la página.

Arquitectura MVC

```
> controller  
> model  
> view  
🐘 autoload.php  
🐘 index.php
```

```
▽ model  
🐘 admin.php  
🐘 anime.php  
🐘 animeinfodb.php  
🐘 cliente.php  
🐘 comentario.php  
🐘 funciones.inc.php  
🐘 lista.php  
🐘 megusta.php  
🐘 usuario.php
```

```
▽ view  
> assets  
> templates  
🐘 anime.php  
🐘 editarComentarios.php  
🐘 editarPerfil.php  
🐘 index.php  
🐘 lista.php  
🐘 listas.php  
🐘 login.php  
🐘 meGusta.php  
🐘 opcionesAdmin.php  
🐘 perfil.php  
🐘 politicaPrivacidad.php  
🐘 signUp.php
```

```
▽ controller  
🐘 ajaxBuscador.php  
🐘 ajaxVerMas.php  
🐘 animeController.php  
🐘 cerrarSesion.php  
🐘 editarComentariosController.p...  
🐘 editarPerfilController.php  
🐘 indexController.php  
🐘 listaController.php  
🐘 listasController.php  
🐘 loginController.php  
🐘 meGustaController.php  
🐘 opcionesAdminController.php  
🐘 perfilController.php  
🐘 politicaPrivacidadController.php  
🐘 signUpController.php
```

```
▽ view  
> assets  
▽ templates  
🐘 footer.php  
🐘 head.php  
🐘 header.php
```

```
▽ view  
▽ assets  
▽ css  
# anime.css  
# editarComentarios.css  
# editarPerfil.css  
# footer.css  
# hamburguerMenu.css  
# header.css  
# index.css  
# lista.css  
# listas.css  
# login.css  
# meGusta.css  
# opcionesAdmin.css  
# perfil.css  
# politicaPrivacidad.css  
# signUp.css  
# style.css  
▽ fonts  
▲ Asap-VariableFont_wdth,wght...  
▲ BeVietnamPro-Bold.ttf  
▲ BeVietnamPro-Regular.ttf  
▲ Chicle-Regular.ttf  
▲ Lato-Bold.ttf  
▲ Lato-Regular.ttf  
▽ images  
▽ scripts
```

```
▽ view  
▽ assets  
▽ css  
▽ fonts  
▽ images  
🖼 default.jpg  
🖼 fondo1.jpg  
🖼 fondo2.jpg  
🖼 fondo3.jpg  
🖼 fondo4.jpg  
🖼 fondo5.jpg  
🖼 logoFinal2.png  
▽ scripts  
JS accederAnime.js  
JS aceptarCookie.js  
JS ajaxBuscador.js  
JS ajaxVerMas.js  
JS comprobarComentario.js  
JS comprobarListas.js  
JS comprobarLogin.js  
JS comprobarSignUp.js  
JS editarPerfil.js  
JS hamburguerMenu.js  
JS obtenerURL.js  
JS opcionesAdmin.js  
JS opcionesListas.js  
JS validarInput.js
```

Requisitos mínimos

Entorno Cliente

Aclaración

Dejo los archivos donde implementé ciertos requisitos

- DOM: todos los archivos
- Expresiones regulares:
 - validarInput.js
 - comprobarComentario.js
 - comprobarLogin.js
 - comprobarSignUp.js
 - comprobarListas.js
 - editarPerfil.js
- Cookies:
 - aceptarCookie.js
 - accederAnime.js
- Eventos: todos excepto validarInput.js
- AJAX:
 - ajaxBuscador.js
 - ajaxVerMas.js
- Try catch: editarPerfil.js
- FileReader: editarPerfil.js
- Clipboard: obtenerURL.js

aceptarCookie.js

En caso de no tener las cookies aceptadas

```
// Comprobamos si el usuario ya ha aceptado las cookies
if (!obtenerCookie("cookiesAceptadas")) {
    mostrarCookies();
}
```

Función para obtener las cookies

```
// Funcion para obtener la cookie insertando el nombre de este
function obtenerCookie(nombre) {
    let cookies = document.cookie.split(";");
    for (let i = 0; i < cookies.length; i++) {
        let cookie = cookies[i].trim();
        if (cookie.indexOf(nombre + "=") === 0) {
            return cookie.substring(nombre.length + 1);
        }
    }
    return "";
}
```

Función para borrar la cookie que tenga el nombre como parámetro

```
// Funcion para eliminar la cookie
function eliminarCookie(nombre) {
    document.cookie = nombre + "=; path=/; max-age=0";
}
```

Función para mostrar el cuadro de aceptar o cancelar cookies

```
function mostrarCookies() {
    // Generamos un fondo que bloquee el contenido que haya detrás
    let fondoCookies = document.createElement("div");
    fondoCookies.classList.add("cookie-fondoCookies");

    // Generamos el div de la cookie con el contenido
    let divCookies = document.createElement("div");
    divCookies.classList.add("cookie-divCookies");

    // Generamos el mensaje que tendrá la cookie
    let mensajeCookies = document.createElement("p");
    mensajeCookies.textContent = "This site uses cookies to enhance the user experience. By continuing to browse, you accept their use.";

    // Generamos los botones aceptar y cancelar
    let btnAceptar = document.createElement("button");
    btnAceptar.classList.add("acceptButton", "btnAcceptCancel");
    btnAceptar.textContent = "Accept";

    let btnCancelar = document.createElement("button");
    btnCancelar.classList.add("cancelButton", "btnAcceptCancel");
    btnCancelar.textContent = "Cancel";

    divCookies.appendChild(mensajeCookies);
    divCookies.appendChild(btnAceptar);
    divCookies.appendChild(btnCancelar);

    document.body.appendChild(fondoCookies);
    document.body.appendChild(divCookies);

    // Eventos cuando acepte o cancele la cookie
    btnAceptar.addEventListener("click", function() {
        document.cookie = "cookiesAceptadas=true; path=/; max-age=" + 60 * 60 * 24 * 30 + "; SameSite=Lax; Secure";
        document.body.removeChild(divCookies);
        document.body.removeChild(fondoCookies);

        location.reload();
    });

    btnCancelar.addEventListener("click", function() {
        document.cookie = "cookiesAceptadas=false; path=/; max-age=" + 600; // 1 hora
        document.body.removeChild(divCookies);
        document.body.removeChild(fondoCookies);

        // Eliminamos las cookies en caso de tener si las cookies fueron rechazadas
        eliminarCookie("ultimosAnimesVistos");
        eliminarCookie("fondo");
        location.reload();
    });
}
```

ajaxBuscador.js

Empezamos nombrando las variables que usaremos a continuación

```
$(document).ready(function () {
    let ultimaBusqueda = "";
    let ultimoGenero = "";
    let ultimoOrdenado = "";
    let ultimoTipo = "";
    let delayBusqueda = null; // Variable para almacenar el timeout
```

Función para realizar la búsqueda de animes con una llamada a AJAX

```
// Función para realizar la búsqueda
function realizarBusqueda(cantidadAnimes = 0, limiteAnimes = 20) {
    // Obtenemos los valores insertados en los inputs
    let buscar = document.getElementById('buscar').value.trim();
    let genero = document.getElementById('genero').value;
    let ordenar = document.getElementById('ordenar').value;
    let tipo = document.getElementById('tipo').value;

    // Si la búsqueda no ha cambiado, no se hace nada
    if (buscar === ultimaBusqueda && genero === ultimoGenero && ordenar === ultimoOrdenado && tipo === ultimoTipo) {
        return;
    }

    // Realizamos la petición AJAX
    $.get('ajaxBuscador.php', {
        buscar: buscar,
        genero: genero,
        ordenar: ordenar,
        tipo: tipo,
        cantidadAnimes: cantidadAnimes,
        limiteAnimes: limiteAnimes
    })
    .done(function(response) {
        // En caso de funcionar se hace la petición
        let animes = JSON.parse(response);
        let resultadosAnime = document.getElementById('resultadosAnime');

        // Limpiamos los resultados anteriores de manera segura
        while (resultadosAnime.firstChild) {
            resultadosAnime.removeChild(resultadosAnime.firstChild);
        }

        // Eliminamos la sección de "no resultados" en caso de existir
        let noResultadosAnime = document.querySelector('[section.noResultadosAnime]');
        if (noResultadosAnime) {
            noResultadosAnime.remove();
        }

        // Si obtenemos animes los muestra, sino, nos muestra un error
        if (animes.length > 0) {
            // Añadimos los animes al DOM
            animes.forEach(function(anime) {
                generarAnimeDOM(anime, resultadosAnime);
            });
        } else {
            // Mostrar mensaje de no resultados
            generarNoResultadosDOM(resultadosAnime);
        }

        // Actualizamos las últimas búsquedas
        ultimaBusqueda = buscar;
        ultimoGenero = genero;
        ultimoOrdenado = ordenar;
        ultimoTipo = tipo;
    })
}
```

Eventos para alternar los animes que encuentre dependiendo de la búsqueda y/o filtro insertado

```
// Evento con timeout para comprobar cambios en los filtros
$('#buscar').on('input', function () {
    clearTimeout(delayBusqueda); // Cancelamos la búsqueda anterior si no se ejecutó
    delayBusqueda = setTimeout(function () {
        realizarBusqueda(); // Hacemos la búsqueda con un pequeño retraso
    }, 500);
});

// Controlamos si se realiza algún cambio, llamando así a la función que realiza la búsqueda
$('#genero, #ordenar, #tipo').on('change', function () {
    clearTimeout(delayBusqueda); // Cancelamos la búsqueda anterior si no se ejecutó
    delayBusqueda = setTimeout(function () {
        realizarBusqueda(); // Hacemos la búsqueda con un pequeño retraso
    }, 300);
});
```

Funciones para generar contenidos con DOM

```
// Función para generar el DOM de un anime
function generarAnimeDOM(anime, resultadosAnime) {
    let animeDiv = document.createElement('div');
    animeDiv.classList.add('anime');
    animeDiv.setAttribute('data-id', anime.id);
    animeDiv.setAttribute('data-nombre', anime.nombreAnime);

    let img = document.createElement('img');
    img.setAttribute('src', anime.portada);
    img.setAttribute('alt', anime.nombreAnime);

    let h5 = document.createElement('h5');
    h5.classList.add('my-3');
    h5.textContent = anime.nombreAnime;

    animeDiv.appendChild(img);
    animeDiv.appendChild(h5);
    resultadosAnime.appendChild(animeDiv);
}

// Función para generar el DOM de resultados no encontrados
function generarNoResultadosDOM(resultadosAnime) {
    let noResultadosAnime = document.createElement('section');
    noResultadosAnime.classList.add('noResultadosAnime', 'text-center');

    let p = document.createElement('p');
    p.classList.add('text');
    p.textContent = 'No results found.';

    noResultadosAnime.appendChild(p);
    resultadosAnime.append(noResultadosAnime);
}
```

ajaxVerMas.js

Empezamos declarando las variables

```
$(document).ready(function () {
    let cantidadAnimes = 0;
    let limiteAnimes = 20;
    let cargando = false;
```

función para cargar más animes cuando se pulsa el botón

```
function cargarMasAnimes() {
    if (cargando) return;

    cargando = true;

    // Obtenemos los valores de los filtros
    let termino = document.getElementById("buscar").value.trim();
    let genero = document.getElementById("genero").value;
    let ordenar = document.getElementById("ordenar").value;
    let tipo = document.getElementById("tipo").value;

    $.get("ajaxVerMas.php", [
        action: "cargarAnimes",
        cantidadAnimes,
        limiteAnimes,
        buscar: termino,
        genero,
        ordenar,
        tipo,
    ])
    .done(function (response) {
        let animes = JSON.parse(response);
        let resultadosAnime = document.getElementById("resultadosAnime");
        let verMasButton = document.getElementById("verMas");

        // Quitamos el mensaje de error si existe
        let errorSection = document.querySelector('section.errorSection');
        if (errorSection) {
            errorSection.remove();
        }

        // Si no hay animes, desabilitamos el botón y mostramos el mensaje de error solo cuando se entra a la pagina
        if (!animes || animes.length === 0) {
            if (cantidadAnimes === 0) {
                generarError();
            }
            verMasButton.disabled = true;
            return;
        }

        // Mostramos Los animes
        animes.forEach((anime) => {
            generarAnimeDOM(anime, resultadosAnime);
        });

        // Aumentamos la cantidad de animes para poder mostrar los siguientes 20
        cantidadAnimes += limiteAnimes;

        // En caso de que los animes sean menores que el límite, desabilitamos el botón
        if (animes.length < limiteAnimes) {
            verMasButton.disabled = true;
        }

        cargando = false;
    })
    .fail(function () {
        console.error("Error al cargar los animes");
        cargando = false;
    });
}
```

función para cargar los primeros 20 animes con el botón

```
// Función usada se cargan los primeros animes al entrar y el botón
function cargarAnimes() {
    let verMasButton = document.getElementById("verMas");

    // Si el botón no existe, lo creamos
    if (!verMasButton) {
        verMasButton = generarVerMasButton();
    }

    // Para evitar errores por si tiene el evento de click se lo quitamos y se lo añadimos de nuevo
    verMasButton.removeEventListener("click", cargarMasAnimes);
    verMasButton.addEventListener("click", cargarMasAnimes);

    // Cargamos Los primeros 20 animes
    setTimeout(function () {
        cargarMasAnimes();
    }, 500);
}

cargarAnimes();
```

Evento para alterar si añadir o no el botón de ver mas dependiendo si se ha buscado algo o no

```
// Evento para añadir o eliminar el botón "Ver más" según la búsqueda
let busquedaInput = document.getElementById("buscar");
busquedaInput.addEventListener("input", function () {
    let termino = this.value;

    let verMasButton = document.getElementById("verMas");

    // En caso de que el campo de búsqueda esté vacío y no hay botón "Ver más" entonces lo creamos
    // En caso de que el campo de búsqueda tenga texto y exista el botón "Ver más", lo eliminamos
    if (termino === "" && !verMasButton) {
        verMasButton = generarVerMasButton();
        verMasButton.addEventListener("click", cargarMasAnimes);
    } else if (termino !== "" && verMasButton) {
        verMasButton.remove();
    }
});
```

Funciones para generar contenidos con DOM

```
// Función para generar el DOM de cada anime
function generarAnimeDOM(anime, resultadosAnime) {
    let animeDiv = document.createElement("div");
    animeDiv.classList.add("anime");
    animeDiv.setAttribute("data-id", anime.id);
    animeDiv.setAttribute("data-nombre", anime.nombreAnime);

    let img = document.createElement("img");
    img.setAttribute("src", anime.portada);
    img.setAttribute("alt", anime.nombreAnime);

    let h5 = document.createElement("h5");
    h5.classList.add("my-3");
    h5.textContent = anime.nombreAnime;

    animeDiv.appendChild(img);
    animeDiv.appendChild(h5);
    resultadosAnime.appendChild(animeDiv);
}

// Función para generar el DOM del botón "Ver más"
function generarVerMasButton() {
    let verMasButton = document.createElement("button");
    verMasButton.classList.add("verMas");
    verMasButton.id = "verMas";
    verMasButton.textContent = "Ver más";
    document.querySelector("section.text-center.m-4").appendChild(verMasButton);

    return verMasButton;
}

// Función para generar el DOM del mensaje de error
function generarError() {
    let errorSection = document.createElement('section');
    errorSection.classList.add('errorSection', 'text-center', 'm-4');

    let p = document.createElement('p');
    p.classList.add('text');
    p.textContent = 'An error has occurred or you have exceeded the request limit. Please wait a moment before trying again.';

    errorSection.appendChild(p);

    // Insertamos el mensaje de error en el DOM, al principio de la página
    let container = document.querySelector("section.text-center.m-4");
    container.insertBefore(errorSection, container.firstChild);
}
```

accederAnime.js

Función para redirigir a los detalles del anime seleccionado y guardar los últimos animes vistos

```
// Función que usamos para redirigir al usuario al anime seleccionado y guardarlo en la cookie de los 5 últimos animes vistos
$(document).on('click', '.anime', function() {
    let idAnime = $(this).data('id');
    let nombreAnime = $(this).data('nombre');

    // Verificamos si las cookies han sido aceptadas para guardar el anime
    if (obtenerCookie("cookiesAceptadas") === "true") {
        guardarUltimoAnimeVisto(idAnime, nombreAnime); // Lo almacenamos en la cookie
    }

    window.location.href = 'animeController.php?id=' + idAnime; // Redirigimos al anime
});
```

Función para redirigir al anime seleccionado entrado a los últimos vistos

```
// Función que usamos para redirigir al usuario al anime seleccionado desde el select y guardarla en la cookie de los 5 últimos animes vistos
$(document).on('change', '#animesRecientesVistos', function() {
    let idAnime = $(this).val();

    window.location.href = 'animeController.php?id=' + idAnime; // Redirigimos al anime
});
```

Función para guardar los últimos animes vistos a la cookie

```
// Función para guardar el anime en las cookies solo si las cookies han sido aceptadas
function guardarUltimoAnimeVisto(idAnimeObtenido, nombreAnimeObtenido) {

    if (obtenerCookie("cookiesAceptadas") !== "true") {
        return;
    }

    let animesRecientes = obtenerCookie("ultimosAnimesVistos");

    if (animesRecientes) {
        animesRecientes = JSON.parse(animesRecientes);
    } else {
        animesRecientes = [];
    }

    // En caso de repetirse el anime se elimina y se hace un push con el actual
    animesRecientes = animesRecientes.filter(anime => anime.idAnime !== idAnimeObtenido);
    animesRecientes.push({ idAnime: idAnimeObtenido, nombreAnime: nombreAnimeObtenido });

    // Si hay más de 5 animes, eliminamos el más antiguo
    if (animesRecientes.length > 5) {
        animesRecientes.shift();
    }

    // Guardamos la cookie con los últimos animes vistos
    document.cookie = "ultimosAnimesVistos=" + JSON.stringify(animesRecientes) + "; path=/; max-age=" + 60 * 60 * 24 * 30 + "; SameSite=Lax; Secure";
}
```

Función para mostrar los últimos animes vistos en un select

```
// Función que usamos para cargar los últimos animes desde la cookie y mostrarlos en el select
function obtenerAnimesRecientes() {

    let animesRecientesVistos = obtenerCookie("ultimosAnimesVistos");

    if (animesRecientesVistos) {
        animesRecientesVistos = JSON.parse(animesRecientesVistos);

        let select = document.getElementById("animesRecientesVistos");

        // Creamos un option por cada anime en la cookie
        animesRecientesVistos.forEach(function(anime) {
            let option = document.createElement("option");
            option.value = anime.idAnime;
            option.classList.add("option");
            option.textContent = anime.nombreAnime;
            select.appendChild(option);
        });
    }

    // Cargamos los animes de la cookie en el select cuando la página se cargue
    $(document).ready(function() {
        obtenerAnimesRecientes();
    });
}
```

validarInput.js

```
// Expresiones regulares para validación
const nombreUsuarioRegex = /^[A-z0-9]{5,20}/; // Entre 5 y 20 caracteres alfanumericos
const correoRegex = /^(?=.{6,30}$)([w]+@[a-z]+\.[a-zA-Z]{2,})$/; // Correo que tenga entre 6 y 30 caracteres
const passwordRegex = /^(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9]).{8,20}/; // Contraseña entre 8 y 20 caracteres que tenga al menos una mayúscula, una minúscula y un número
const comentarioRegex = /^(?=>S)([A-zñÑ]{1,-99}){10,400}(?<=S)$/ // Comentario entre 10 y 400 caracteres con alfanumericos, simbolos y que no empiece ni acabe con espacios
const listasRegex = /^(?=>[A-z])[A-z0-9]{4,10}$/ // Lista debe tener entre 4 y 10 caracteres alfanuméricos con al menos una letra
```

Función para validar cada input

```
// Funcion para validar los inputs con expresiones regulares
function validarInput(input, regex, elementoError, mensajeError) {
    // En caso de no cumplir con la expresión regular mostramos el mensaje de error y añadimos el css de error
    if (!regex.test(input.value) && input.value) {
        elementoError.textContent = mensajeError;
        elementoError.classList.add("errorMessage");
        input.classList.add("inputError");
        return false;
    }
    if (elementoError) {
        // Si cumple la expresión regular quitamos el mensaje de error y el css
        elementoError.textContent = "";
    }
    input.classList.remove("inputError");
    return true;
}
```

función para validar si las contraseñas coinciden

```
// Función para validar la confirmación de la contraseña
function validarConfirmacion(input, inputConfirmar, elementoError, mensajeError) {
    // En caso de no cumplir con la expresión regular mostramos el mensaje de error y añadimos el css de error
    if (input.value !== inputConfirmar.value) {
        elementoError.textContent = mensajeError;
        elementoError.classList.add("errorMessage");
        inputConfirmar.classList.add("inputError");
        return false;
    }
    // Si cumple la expresión regular quitamos el mensaje de error y el css
    elementoError.textContent = "";
    inputConfirmar.classList.remove("inputError");
    return true;
}
```

A continuación, muestro los archivos donde hago las comprobaciones con estas funciones

comprobarComentario.js

```
document.addEventListener('DOMContentLoaded', () => {
    // Obtenemos el formulario, el input y el mensaje de error
    const form = document.getElementById('formularioComentario');
    const comentarioInput = document.getElementById('comentario');
    const ComentarioError = document.getElementById('comentarioError');

    let comentarioCorrecto = false; // Variable para comprobar el estado de la validacion

    // Validamos el campo cuando el usuario escribe y cuando accede a la pagina
    comentarioInput.addEventListener('input', () => {
        comentarioCorrecto = validarInput(comentarioInput, comentarioRegex, ComentarioError, 'Only allow from 10 to 400 letters or numbers and this symbols : ?\!-_\\\'');
    });

    comentarioCorrecto = validarInput(comentarioInput, comentarioRegex, ComentarioError, 'Only allow from 10 to 400 letters or numbers and this symbols : ?\!-_\\\'');

    form.addEventListener('submit', (e) => {
        // Si es incorrecto deshabilitamos el boton
        if (!comentarioCorrecto) {
            e.preventDefault();
        }
    });
});
```

comprobarListas.js

Crear lista

```
// CREAR LISTA
document.addEventListener('DOMContentLoaded', () => {
    // Obtenemos el formulario, el input y el mensaje de error
    const form = document.getElementById('crearListaForm');
    const crearListaInput = document.getElementById('nombreLista');
    const crearListaError = document.getElementById('crearListaError');

    let crearListaCorrecta = false; // Variable para comprobar el estado de la validacion

    // Validamos el campo cuando el usuario escribe y cuando accede a la pagina
    crearListaInput.addEventListener('input', () => {
        crearListaCorrecta = validarInput(crearListaInput, listasRegex, crearListaError, 'Between 4 and 10 alphanumeric characters and must have at least one letter.');
    });

    crearListaCorrecta = validarInput(crearListaInput, listasRegex, crearListaError, 'Between 4 and 10 alphanumeric characters and must have at least one letter.');

    form.addEventListener('submit', (e) => {
        // Si es incorrecto deshabilitamos el boton
        if (!crearListaCorrecta) {
            e.preventDefault();
        }
    });
});
```

Editar lista

```
// EDITAR LISTA
document.addEventListener('DOMContentLoaded', () => {
    // Obtenemos el formulario, el input y el mensaje de error
    const form = document.getElementById('editarListaForm');
    const editarListaInput = document.getElementById('editarNombreLista');
    const editarListaError = document.getElementById('editarListaError');

    let editarListaCorrecta = false; // Variable para comprobar el estado de la validacion

    // Validamos el campo cuando el usuario escribe y cuando accede a la pagina
    editarListaInput.addEventListener('input', () => {
        editarListaCorrecta = validarInput(editarListaInput, listasRegex, editarListaError, 'Between 4 and 10 alphanumeric characters and must have at least one letter.');
    });

    editarListaCorrecta = validarInput(editarListaInput, listasRegex, editarListaError, 'Between 4 and 10 alphanumeric characters and must have at least one letter.');

    form.addEventListener('submit', (e) => {
        // Si es incorrecto deshabilitamos el boton
        if (!editarListaCorrecta) {
            e.preventDefault();
        }
    });
});
```

comprobarLogin.js

```
document.addEventListener('DOMContentLoaded', () => {
    // Obtenemos el formulario, los inputs y los mensajes de error
    const form = document.getElementById('formularioIniciarSesion');

    const correoInput = document.getElementById('correo');
    const passwdInput = document.getElementById('passwd');

    const correoError = document.getElementById('correoError');
    const passwdError = document.getElementById('passwdError');

    // Variables para comprobar el estado de las validaciones
    let correoCorrecto = false;
    let passwdCorrecto = false;

    // Validamos cada campo cuando el usuario escribe y cuando accede a la página
    correoInput.addEventListener('input', () => {
        correoCorrecto = validarInput(correoInput, correoRegex, correoError, 'Please enter a valid email that has between 6 and 30 characters.');
    });

    passwdInput.addEventListener('input', () => {
        passwdCorrecto = validarInput(passwdInput, passwdRegex, passwdError, 'The password must be between 8 and 30 characters long and include an uppercase letter, a lowercase letter, and a number.');
    });

    correoCorrecto = validarInput(correoInput, correoRegex, correoError, 'Please enter a valid email that has between 6 and 30 characters.');
    passwdCorrecto = validarInput(passwdInput, passwdRegex, passwdError, 'The password must be between 8 and 30 characters long and include an uppercase letter, a lowercase letter, and a number.');

    form.addEventListener('submit', () => {
        // Si es incorrecto desabilitamos el botón
        if (!correoCorrecto || !passwdCorrecto) {
            e.preventDefault();
        }
    });
});
```

comprobarSignUp.js

```
document.addEventListener('DOMContentLoaded', () => {
    // Obtenemos el formulario, los inputs y los mensajes de error
    const form = document.getElementById('formularioRegistrarse');

    const nombreUsuarioInput = document.getElementById('nombreUsuario');
    const correoInput = document.getElementById('correo');
    const passwdInput = document.getElementById('passwd');
    const passwdConfirmarInput = document.getElementById('passwdConfirmar');

    const nombreUsuarioError = document.getElementById('nombreUsuarioError');
    const correoError = document.getElementById('correoError');
    const passwdError = document.getElementById('passwdError');
    const passwdConfirmarError = document.getElementById('passwdConfirmarError');

    // Variables para comprobar el estado de las validaciones
    let nombreUsuarioCorrecto = false;
    let correoCorrecto = false;
    let passwdCorrecto = false;
    let confirmarPasswdCorrecto = false;

    // Validamos cada campo cuando el usuario escribe y cuando accede a la página
    nombreUsuarioInput.addEventListener('input', () => {
        nombreUsuarioCorrecto = validarInput(nombreUsuarioInput, nombreUsuarioRegex, nombreUsuarioError, 'It must have between 5 and 30 alphanumeric characters.');
    });

    correoInput.addEventListener('input', () => {
        correoCorrecto = validarInput(correoInput, correoRegex, correoError, 'Please enter a valid email that has between 6 and 30 characters.');
    });

    passwdInput.addEventListener('input', () => {
        passwdCorrecto = validarInput(passwdInput, passwdRegex, passwdError, 'The password must be between 8 and 30 characters long and include an uppercase letter, a lowercase letter, and a number.');
    });

    passwdConfirmarInput.addEventListener('input', () => {
        confirmarPasswdCorrecto = validarConfirmacion(passwdInput, passwdConfirmarInput, passwdConfirmarError, "The passwords do not match.");
    });

    nombreUsuarioCorrecto = validarInput(nombreUsuarioInput, nombreUsuarioRegex, nombreUsuarioError, 'It must have between 5 and 30 alphanumeric characters.');
    correoCorrecto = validarInput(correoInput, correoRegex, correoError, 'Please enter a valid email that has between 6 and 30 characters.');
    passwdCorrecto = validarInput(passwdInput, passwdRegex, passwdError, 'The password must be between 8 and 30 characters long and include an uppercase letter, a lowercase letter, and a number.');
    confirmarPasswdCorrecto = validarConfirmacion(passwdInput, passwdConfirmarInput, passwdConfirmarError, "The passwords do not match.");

    form.addEventListener('submit', () => {
        // Si es incorrecto desabilitamos el botón
        if (!nombreUsuarioCorrecto || !correoCorrecto || !passwdCorrecto || !confirmarPasswdCorrecto) {
            e.preventDefault();
        }
    });
});
```

editarPerfil.js

```
document.addEventListener('DOMContentLoaded', () => {
    // Obtenemos el formulario
    const form = document.getElementById('formularioEditarPerfil');

    // Obtenemos los inputs
    const nombreUsuarioInput = document.getElementById('nombreUsuario');
    const correoInput = document.getElementById('correo');
    const passwdInput = document.getElementById('passwd');

    // Obtenemos los mensajes de error de los inputs
    const nombreUsuarioError = document.getElementById('nombreUsuarioError');
    const correoError = document.getElementById('correoError');
    const passwdError = document.getElementById('passwdError');

    // Variables para comprobar el estado de las validaciones
    let nombreUsuarioCorrecto = false;
    let correoCorrecto = false;
    let passwdCorrecto = false;

    // Validamos cada campo cuando el usuario escribe
    nombreUsuarioInput.addEventListener('input', () => {
        nombreUsuarioCorrecto = validarInput(nombreUsuarioInput, nombreUsuarioRegex, nombreUsuarioError, 'It must have between 5 and 30 alphanumeric characters.');
    });

    correoInput.addEventListener('input', () => {
        correoCorrecto = validarInput(correoInput, correoRegex, correoError, 'Please enter a valid email has between 6 and 30 characters.');
    });

    passwdInput.addEventListener('input', () => {
        passwdCorrecto = validarInput(passwdInput, passwdRegex, passwdError, 'The password must be between 8 and 30 characters long and include an uppercase letter, a lowercase letter, and a number.');
    });

    // Los seteamos tambien aqui en caso de estar el campo vacio o al fallar cuando seteamos el correo y contraseña que haya introducido
    nombreUsuarioCorrecto = validarInput(nombreUsuarioInput, nombreUsuarioRegex, nombreUsuarioError, 'It must have between 5 and 30 alphanumeric characters.');
    correoCorrecto = validarInput(correoInput, correoRegex, correoError, 'Please enter a valid email has between 6 and 30 characters.');
    passwdCorrecto = validarInput(passwdInput, passwdRegex, passwdError, 'The password must be between 8 and 30 characters long and include an uppercase letter, a lowercase letter, and a number.');

    // Preventir envío si alguna validación es incorrecta
    form.addEventListener('submit', (e) => {
        if (!nombreUsuarioCorrecto || !correoCorrecto || !passwdCorrecto) {
            e.preventDefault();
        }
    });
});
```

En este archivo se ha implementado **fileReader** para mostrar la imagen que inserte el usuario a la hora de modificar el perfil, controlado con un try catch en caso de error.

```
// Función que se ejecuta cuando el usuario selecciona un archivo
function verImagen(e) {
    try {
        // Obtenemos el archivo seleccionado por el usuario
        const imagen = e.target.files[0];

        // Comprobamos si el archivo es una imagen
        if (imagen && imagen.type.startsWith("image/")) {
            // Si el archivo es una imagen entonces creamos un objeto FileReader para Leer el archivo
            const reader = new FileReader();

            // Función que se ejecuta cuando la imagen ha sido leída correctamente
            reader.onload = function (e) {
                // Obtenemos el elemento <img> donde vamos a mostrar la previsualización
                const img = document.getElementById("verFotoPerfil");

                // Establecemos el src de la imagen con el resultado que nos da FileReader en base64
                img.src = e.target.result;

                // Limpiamos el mensaje de error, si estaba visible
                const imagenError = document.getElementById("imagenError");
                imagenError.style.display = "none";
            };

            // Leemos el archivo como una URL en formato base64
            reader.readAsDataURL(imagen);
        } else {
            // Si el archivo no es una imagen, mostramos el mensaje de error en el DOM
            throw new Error("Por favor, selecciona solo una imagen.");
        }
    } catch (error) {
        // Capturamos cualquier error y mostramos el mensaje correspondiente
        const imagenError = document.getElementById("imagenError");
        imagenError.textContent = error.message; // Mostramos el mensaje de error en el DOM
        imagenError.style.display = "block"; // Hacemos visible el mensaje de error

        // Limpiamos el campo de archivo y la imagen previsualizada
        document.getElementById("img").value = "";
        document.getElementById("verFotoPerfil").src = "";
    }
}
```

hamburguerMenu.js

```
// Evento para alternar el menú hamburguesa en mobile al hacer click en el elemento con id menuIcono

document.getElementById('menuIcono').addEventListener('click', function() {
    const menuIcono = document.getElementById('menuIcono');
    const navegadorMovil = document.getElementById('navegadorMovil');

    // Cambiar entre activado y no activado el menu hamburguesa

    menuIcono.classList.toggle('active');
    navegadorMovil.classList.toggle('active');
});
```

obtenerURL.js

Se ha implementado **clipboard** para copiar al portapapeles al hacer click en un botón la URL del anime que este visualizando el usuario.

```
// Función para copiar la URL y cambiar el texto del botón
document.getElementById('botonCopiarURL').addEventListener('click', function(e) {
    e.preventDefault();

    // Obtenemos la url con la API de clipboard, con writetext copiamos dicha información que haya dentro de este
    navigator.clipboard.writeText(window.location.href).then(function() {
        // Cambiamos el texto del botón a "URL copiada"
        document.querySelector('#botonCopiarURL span').textContent = 'URL copied';

        // Despues de unos segundos vuelve a tener el texto anterior
        setTimeout(function() {
            document.querySelector('#botonCopiarURL span').textContent = 'Copy URL';
        }, 2000);
    });
});
```

opcionesAdmin.js

Al querer borrar un usuario se abre un modal donde obtenemos por hidden el id del usuario con la siguiente función:

```
// Función para obtener el id del usuario a eliminar
$(document).ready(function () {
    // Asignamos el evento de click a todos los botones con la clase "deleteUser"
    $('.deleteUser').on('click', function () {
        // Obtenemos el ID del usuario del atributo "data-id"
        const idUsuario = $(this).data('id');

        // Pasamos el ID al input hidden del formulario
        $('#obtenerIdUsuario').val(idUsuario);
    });
});
```

opcionesListas.js

Al querer borrar una lista se abre un modal donde obtenemos por hidden el id de la lista y como value por input el nombre de este con la siguiente función:

```
$document).ready(function () {
    // Evento para editar el nombre de la lista
    $('.modify').on('click', function () {
        const nombreLista = $(this).data('nombre');
        const idLista = $(this).data('id');

        // Asignamos los valores a los inputs correspondientes
        $('#editarNombreLista').val(nombreLista);
        $('#idListaEditar').val(idLista);
    });
}
```

Al querer borrar una lista se abre un modal donde obtenemos por hidden el id del usuario con la siguiente función:

```
// Evento para borrar una lista
$('.deleteList').on('click', function () {
    const idLista = $(this).data('id');

    // Pasamos el ID al input hidden del formulario
    $('#idLista').val(idLista);
});
```

Base de datos

Consulta JOIN

```
// Consulta para obtener los datos del anime en la BBDD
$sql = "
SELECT a.idAnime, a.nombreAnime, a.tipo, a.capitulos, a.estado, a.sinopsis, a.fechaInicio, a.fechaFin, a.portada, a.cantidadMeGusta, GROUP_CONCAT(g.genero) AS generos
FROM Anime a
JOIN AnimeGenero ag ON a.idAnime = ag.Anime_idAnime
JOIN Genero g ON ag.Genero_idGenero = g.idGenero
WHERE a.idAnime = :idAnime
GROUP BY a.idAnime
";
```

Consulta con subconsulta

```
// Consulta con subconsulta para añadir a me gusta e incrementar el contador de me gusta +1
$sql = "
INSERT INTO darmegusta (Anime_idAnime, Usuario_idUsuario)
VALUES (:idAnime, :idUsuario);

UPDATE Anime
SET cantidadMeGusta = (
    SELECT COUNT(*) FROM darmegusta WHERE Anime_idAnime = :idAnime
)
WHERE idAnime = :idAnime;
";
```

Entorno servidor

Aclaración

Se ha implementado las siguientes características:

- POO
- Herencia: Usuario tiene de hijo a cliente y admin
- MVC
- Clase abstracta: Usuario
- Interfaz: Me gusta, se implementa en la clase Anime
- Cookies: fondo aleatorio en login y sign up
- Sesiones:
 - ID de sesión para seguridad
 - Id del usuario
 - Rol del usuario
 - Tiempo inactivo (controlado en caso de estar ausente cerrar sesión)
- Perfiles de usuario y zonas privadas inaccesibles para usuarios que no pertenezcan a dicho rol o no hayan iniciado sesión
- Control Try-catch para la base de datos
- Operaciones CRUD
- Uso de una API: [Anilist](#)
- Cumplimiento de todos los requisitos funcionales

Diseño de interfaces web

Aclaración

Implementando Bootstrap en ciertas clases en mi css modifco algunas de bootstrap como por ejemplo container-fluid en anime.css para ciertos casos.

Futuras actualizaciones

- Actualización de animes cada cierto tiempo
- Vista con próximos animes
- Mejorar funcionalidad de compartir añadiendo redes sociales directamente
- Comentarios con valoración y respuestas a comentarios

Conclusión

Después de todo el proceso para realizar la página, requisitos a cumplir, funciones a implementar y por todos los procesos por los que se tuvo que pasar para dar por finalizado la aplicación web, ha sido todo un reto para cumplir todo lo propuesto, completar todos los requisitos propuestos y finalizar a tiempo la aplicación web.

He aprendido tecnologías como **jQuery** para las peticiones en AJAX y mayor facilidad en algún función y evento, también **Bootstrap** que fue implementado por ejemplo para estructuras ciertos contenidos y para los modales.

En general estoy satisfecho con lo que he podido realizar hasta esta versión, aunque se puede mejorar bastante de cara al futuro.