

PROYECTO FINAL

Jorge Zatón Pérez

PYTHONNN

Hombre con un cuchillo en la mano

Descripción generada automáticamente con confianza baja



JORGE ZATÓN PÉREZ

RESPONSABLE DEL DESARROLLO DE LA APLICACIÓN

**Fecha:** 14, FEBRERO, 2025

Índice de Contenidos

[DESCRIPCIÓN GENERAL DEL PROYECTO 4](#_Toc190599749)

[OBJETIVOS Y ALCANCE DEL PROYECTO 5](#_Toc190599750)

[Objetivos del Proyecto 5](#_Toc190599751)

[Alcance del Proyecto 6](#_Toc190599752)

[STACK TECNOLÓGICO Y ALTERNATIVAS EVALUADAS 8](#_Toc190599753)

[MODELO DE DATOS 10](#_Toc190599754)

[Resumen 14](#_Toc190599755)

[REQUISITOS DE LA APLICACIÓN 15](#_Toc190599756)

[Gestión de usuarios: 15](#_Toc190599757)

[Catálogo de películas y series. 19](#_Toc190599758)

[Gestión de la base de datos 28](#_Toc190599759)

[MANUAL DE INSTALACIÓN 30](#_Toc190599760)

[CONCLUSIONES 32](#_Toc190599761)

[EVOLUTIVOS DEL PROYECTO 32](#_Toc190599762)

[Conclusiones 34](#_Toc190599763)

[Conclusiones generales del proyecto 34](#_Toc190599764)

[Aprendizajes y experiencias adquiridas 35](#_Toc190599765)

[Evolutivos del proyecto 37](#_Toc190599766)

# DESCRIPCIÓN GENERAL DEL PROYECTO

El proyecto seleccionado ha sido una aplicación audiovisual tipo Netflix a la que hemos dado el nombre de Videoflix, diseñada para gestionar y mostrar información sobre series de televisión con sus temporadas y capítulos y visualización de las mismas, aunque en la construcción inicial se ha suplantado los completos por los trailers de los mismos, evitando así un mayor peso y problemas con derechos de imágenes y vídeos. La aplicación utiliza Reflex para la construcción de la interfaz de usuario y se conecta a una base de datos SQL para almacenar y recuperar información sobre dichas series y películas, así como el registro de usuarios.

Consta de una página principal, home, así como de sistema de registro e inicio de sesión, tanto para usuario como para el administrador. El usuario base, además de registrarse, puede acceder a la información de las series y películas, visualizarlas, marcarlas como favoritas o vistas y revisas las estadísticas de visualización de las mismas.

El administrador, además de tener acceso a todo ello, también puede modificar, añadir o eliminar tanto usuarios como series y películas. De igual manera, puede revisar las estadísticas de cada usuario y compararlos entre sí.

Se ha dotado de un contenido básico y minimalista, intentando que sea una aplicación sencilla de utilizar y visual, sin excesos que puedan distorsionar. Todas las páginas de la aplicación tienen una cabecera tipo *navbar* y un *footer* o pie de página.

Aunque se ha seguido un mismo patrón en el diseño y maquetación, se puede comprobar que distintas páginas similares, como el listado de series o películas, al igual que en los formularios, se han realizado cambios para identificar cuál resulta más atractivo y tomar una decisión final una vez puesta en marcha.

Este proyecto proporciona una plataforma completa para la gestión y visualización de series de televisión y sus capítulos, así como películas, ofreciendo una experiencia de usuario intuitiva y funcional, a la par de tener grandes posibilidades de desarrollo y crecimiento, tanto de cara al usuario como al aprovechamiento del mismo.

# OBJETIVOS Y ALCANCE DEL PROYECTO

## Objetivos del Proyecto

1. Gestión de Películas y Series:

- Proporcionar una plataforma para visualizar y obtener información sobre series de televisión y sus capítulos.

- Permitir la visualización de detalles de cada serie y película, incluyendo el título, número de temporadas, año de estreno, sinopsis y portada. En el caso de la película, permite también la visualización de la misma.

- Permitir la visualización de cada capítulo, incluyendo el título del capítulo, número de capítulo, número de temporada, sinopsis, duración y un enlace al video. En el caso de los capítulos así como de las temporadas, se ha obviado la inclusión de las portadas. Esto se ha realizado a propósito para liberar de carga al proyecto, aunque en la puesta en marcha final se puede modificar.

2. Interacción con la Base de Datos:

- Facilitar la inserción, actualización (en caso del administrador) y consulta de datos relacionados con las películas y series.

- Manejar de igual manera la inserción y actualización de usuarios de cara al administrador, así como el registro del usuario almacenando los datos en la misma.

- Utilizar consultas para recuperar y mostrar la información necesaria en la interfaz de usuario, bien sea para comparar los datos o mostrarlos.

3. Experiencia de Usuario:

- Ofrecer una interfaz de usuario intuitiva, ágil y atractiva utilizando Reflex.

- Permitir a los usuarios marcar las películas y capítulos como favoritos y visualizarlos en una sección dedicada.

- visualizar las estadísticas de películas vistas.

4. Automatización de Datos:

- Utilizar técnicas de web scraping para extraer información de páginas web y poblar la base de datos con datos relevantes sobre series y capítulos. De igual manera se ha utilizado archivos csv para recopilar datos de películas y se han generado usuarios aleatorios para obtener una base de datos más completa.

## Alcance del Proyecto

En cuanto a la gestión de usuario, se ha limitado los datos a obtener. Tras valorar poder incluir fotos de perfil, algún tipo de biografía o similar, se ha descartado. De igual manera, la gestión de contraseñas es básica, no se ha contemplado la interacción avanzada (usuario, recomendaciones o calificaciones). Asimismo, no se ha activado la integración con Redes sociales, aunque se ha dejado la opción de cara al futuro.

En cuanto al contenido multimedia, como ya se ha indicado, se limita a la visualización de tráilers. El motivo principal es el peso que ganaría la aplicación al hacerlo y hasta la puesta en marcha no es necesario. Sí que se ha implementado de manera que el vídeo quede embebido ya sea a través de un enlace (tipo Youtube) o con código tipo html o similar.

Por otro lado, la gestión de administración es bastante completa, permitiendo cualquier modificación de los datos, bien sean de usuario, capítulos o películas.

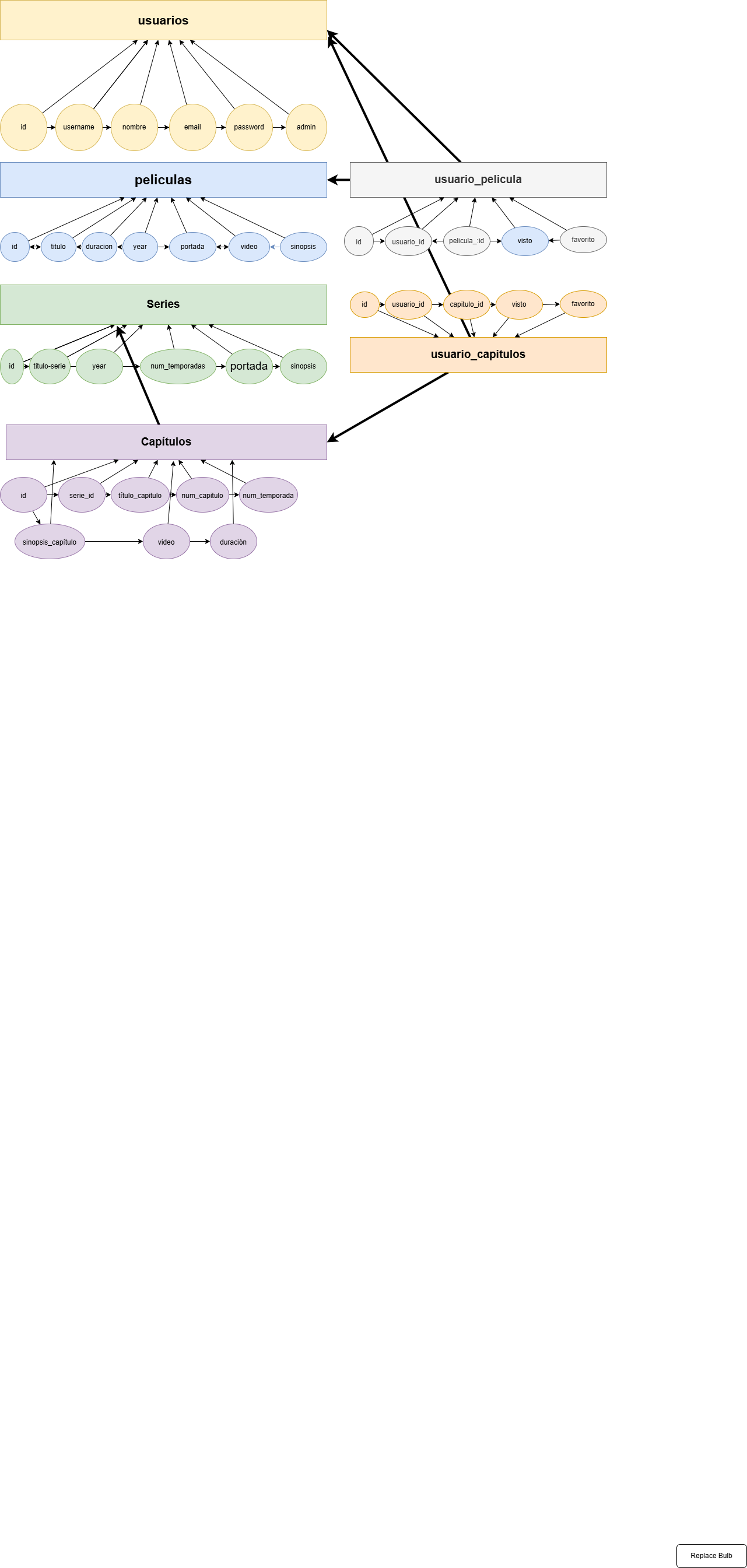
El proyecto Videoflix tiene como objetivo proporcionar una plataforma completa para la gestión y visualización de películas y series de televisión. Utiliza tecnologías modernas como Reflex para la construcción de la interfaz de usuario, pandas para la manipulación de datos, y BeautifulSoup para el web scraping. El alcance del proyecto incluye la creación de modelos y consultas para interactuar con la base de datos, la automatización de la extracción de datos y la presentación de una interfaz de usuario intuitiva y funcional, además de generar un sistema de administración sencillo con el que administrar fácilmente la creación y modificación de los datos.

# STACK TECNOLÓGICO Y ALTERNATIVAS EVALUADAS

* Como lenguaje de programación base se utiliza Python 3.11.8, elegido dado que es el aprendido durante este curso.
* El IDE utilizado es VSC. La razón es que, tras realizar las prácticas tanto en Jupyter como PyCharm, se ha considerado interesante realizar la programación en un IDE distinto y para familiciarse con diferentes alternativas. Además, aunque quizás no esté optimizado a Python como PyCharm, gracias a su amplitud y a la posibilidad de instalar plugins, es a día de hoy uno de los más completos y utilizados en programación.
* Como Framework para la creación de la web, me he decantado por Reflex 0.6.6.. Las alternativas valoradas fueron Flask, descartada dado que ya se utilizó en una práctica anterior y la intención en nuestro caso ha sido aprender a programar con un framework nuevo y Django. A pesar de ser este último posiblemente el más usado para realizar aplicaciones web en Python y ser una herramienta muy bien valorada por la comunidad por su enorme potencial, me decanté por Reflex puesto que, aunque es muy reciente (nació en 2022), tiene una gran inversión detrás y mucho potencial de crecimiento (en 2023 se han realizado más de 15000 webs). Además, es compatible con otros lenguajes de programación como Javascript y permite la maquetación con lenguajes de etiquetas como HTML y también CSS (lo que se ha hecho en el proyecto, con la particularidad de que, siguiendo las buenas prácticas de Python, utiliza el guion bajo en lugar del medio habitual en CSS). No obstante, puede realizarse sin conocimientos de estos, usando únicamente Python puro y tiene, entre otros, configuradas por defecto librerías como SqlAlchemy o Matplotlib. Por otro lado, están creando un hosting propio en el que, simplemente ejecutando la instrucción “reflex deploy”, podemos tener nuestra web operativa y subida a un servidor de forma gratuita o previo paga añadiendo dominio propio.
* Para la realización de los logos y Favicon se ha utilizado la web <https://designs.ai/>, con la que, mediante inteligencia artificial es relativamente sencillo crear diseños de casi todo tipo. En este caso, se ha valorado también hacerlo con Canva, pero nos pareció mejor esta opción dado que crea en diversos formatos los logos con las indicaciones que se le da de una manera muy automatizada.
* En cuanto a la base de datos, nos hemos decantado por realizarla con Sqlite y se ha utilizado DB Browser SQlite 3.12 para las comprobaciones, así como el ORM SQLAlchemy 2.0.36 junto a sqlmodel 0.0.22. No se han evaluado otras opciones, dado que Reflex está preconfigurado para trabajar con este modelo de BBDD, a pesar de que puede funcionar a la perfección con cualquier BBDD, ya sea SQL o NoSQL compatible con Python.
* Se ha utilizado la biblioteca randomuser 1.6 para generar usuarios aleatorios y añadirlos a la base de datos. De igual manera, se ha utilizado la librería random 0.0.1 para generar números aleatorios.
* Para las fechas, nos hemos decantado por Datetime.
* Para dotar de un mayor contenido de una manera rápida, aprender su funcionamiento e iniciarnos en el webscrapping, se ha utilizado BeautifulSoup 4.13.3 y request 2.32.3
* También se ha utilizado pandas 1.3.3 tanto para extraer películas de un archivo csv, como para insertar estos datos y los extraídos a través del webscrapping a la base de datos.

# MODELO DE DATOS

Como se ha explicado anteriormente, el modelo de datos se ha creado con SqlAlchemy y Sqlmodel, integradas en Reflex. De igual manera, se ha utilizado DB browser Sqlite para las comprobaciones. A continuación, dejamos un esquema (también adjunto junto al resto del proyecto) y pasamos con su explicación. Como singularidad, cabe destacar que en la creación de las clases del archivo models.py (del que hablaremos más adelante, no es necesario la inserción del primary key, dado que Reflex lo hace de forma automática.

****

La base de datos del proyecto está diseñada para almacenar y gestionar información sobre series de televisión, películas y usuarios. A continuación, se describen las principales tablas y sus relaciones:

Tablas Principales

1. Series
   * Descripción: Esta tabla almacena información sobre las series de televisión.
   * Campos:
     + id: Identificador único de la serie (clave primaria).
     + titulo: Título de la serie.
     + num\_temporadas: Número de temporadas de la serie.
     + year: Año de estreno de la serie.
     + sinopsis: Sinopsis de la serie.
     + portada: Ruta de la imagen de portada de la serie.
2. Capitulos
   * Descripción: Esta tabla almacena información sobre los capítulos de las series añadidas a la base de datos.
   * Campos:
     + id: Identificador único del capítulo (clave primaria).
     + serie\_id: Identificador de la serie a la que pertenece el capítulo (clave foránea que referencia a Series.id).
     + titulo\_capitulo: Título del capítulo.
     + num\_capitulo: Número del capítulo dentro de la temporada.
     + num\_temporada: Número de la temporada a la que pertenece el capítulo.
     + sinopsis\_capitulo: Sinopsis del capítulo.
     + video: Enlace al video del capítulo.
     + duracion: Duración del capítulo en minutos.
3. Peliculas
   * Descripción: Esta tabla almacena información sobre las películas.
   * Campos:
     + id: Identificador único de la película (clave primaria).
     + titulo: Título de la película.
     + year: Año de estreno de la película.
     + sinopsis: Sinopsis de la película.
     + portada: Ruta de la imagen de portada de la película.
     + duracion: Duración de la película en minutos.
     + video: Enlace al video de la película. (o al código embebido)
4. Usuarios
   * Descripción: Esta tabla almacena información sobre los usuarios de la plataforma.
   * Campos:
     + id: Identificador único del usuario (clave primaria).
     + nombre: Nombre del usuario.
     + email: Correo electrónico del usuario.
     + password: Contraseña del usuario (almacenada de forma segura).
     + Admin: booleano que indica que un usuario es a su vez administrador de la aplicación web, representado con valores numéricos 0 (False, por defecto) o True en caso de ser administrador**.**
5. Usuario\_Capitulo
   * Descripción: Esta tabla almacena la relación entre los usuarios y los capítulos que han marcado como favoritos.
   * Campos:
     + usuario\_id: Identificador del usuario (clave foránea que referencia a Usuarios.id).
     + capitulo\_id: Identificador del capítulo (clave foránea que referencia a Capitulos.id).
6. Usuario\_Pelicula
   * Descripción: Esta tabla almacena la relación entre los usuarios y las películas que han marcado como favoritas.
   * Campos:
     + usuario\_id: Identificador del usuario (clave foránea que referencia a Usuarios.id).
     + pelicula\_id: Identificador de la película (clave foránea que referencia a Peliculas.id).

Relaciones

* Relación Serie-Capítulo: Existe una relación uno a muchos entre la tabla Series y la tabla Capitulos. Esto significa que una serie puede tener múltiples capítulos, pero cada capítulo pertenece a una sola serie. Esta relación se establece mediante el campo serie\_id en la tabla Capitulos, que referencia al campo id en la tabla Series.
* Relación Usuario-Capítulo: Existe una relación muchos a muchos entre las tablas Usuarios y Capitulos, gestionada a través de la tabla intermedia Usuario\_Capitulo. Esto permite que un usuario pueda marcar múltiples capítulos como favoritos y que un capítulo pueda ser marcado como favorito por múltiples usuarios.
* Relación Usuario-Película: De igual manera, existe una relación muchos a muchos entre las tablas Usuarios y Peliculas, gestionada a través de la tabla intermedia Usuario\_Pelicula. Esto permite que un usuario pueda marcar múltiples películas como favoritas y que una película pueda ser marcada como favorita por múltiples usuarios.

Funcionalidades de la Base de Datos

1. Inserción de Datos:
   * Los datos sobre series, capítulos, películas y usuarios se insertan en la base de datos utilizando sesiones de Reflex y consultas definidas en los módulos de consultas. El administrador es quien gestiona las películas y series, así como los usuarios, eliminando añadiendo o modificando los datos,
2. Consulta de Datos:
   * Las consultas a la base de datos permiten recuperar información sobre series, capítulos, películas y usuarios para mostrarlas en la interfaz de usuario. Estas consultas se gestionan a través de funciones definidas en las distintas páginas de consultas.
3. Gestión de Favoritos:
   * Los usuarios pueden marcar capítulos y películas como favoritos, y estas relaciones se gestionan a través de las tablas Usuario\_Capitulo y Usuario\_Pelicula.

## Resumen

La base de datos está estructurada para gestionar eficientemente la información sobre series de televisión, películas y usuarios. Con tablas bien definidas y relaciones claras, permite almacenar, consultar, actualizar y eliminar datos de manera efectiva. La integración con Reflex y las consultas definidas en cada ventana de la aplicación web, aseguran una interacción fluida entre la base de datos y la interfaz de usuario**.**

# REQUISITOS DE LA APLICACIÓN

## Gestión de usuarios:

Los usuarios y administradores pueden registrarse e iniciar sesión con credenciales seguras. Para ello, desde la página principal o Home, se han añadido enlaces para poder iniciar sesión o registrarse.

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.**

Desde los botones que se aprecian, así como desde los que aparecen en el header, un usuario puede acceder a las páginas de registro e inicio de sesión que se detallan a continuación.

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.** Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

En cuanto al administrador, la ventana de inicio de sesión no se ha añadido enlace directo, para una mayor seguridad. No obstante, se puede acceder directamente desde la ruta <http://localhost:3000/admin_login>, siendo la visualización exacta a la de cualquier otro usuario. Se ha diseñado el programa de manera que si un usuario que no es administrador accede a dicha página, no reconoce su inicio de sesión.

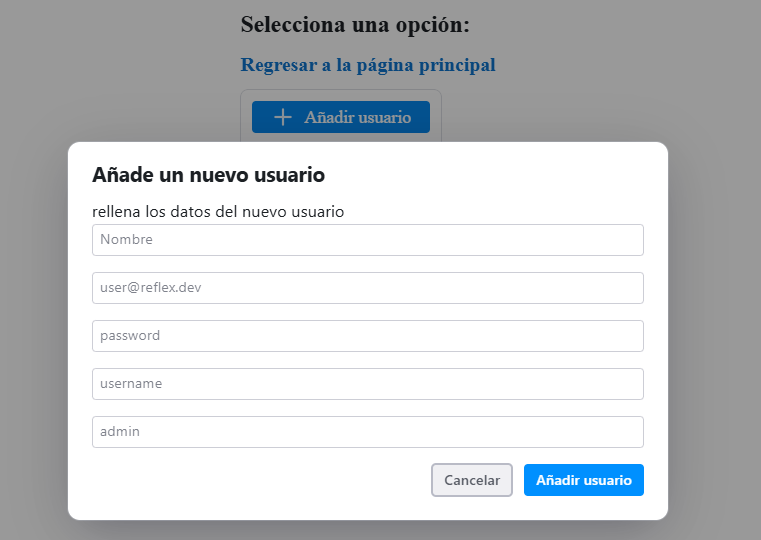
Dicho administrador tiene la capacidad de añadir, editar y eliminar usuarios.

Desde el panel de administración, se presenta un cuadro para que el mismo acceda a sus distintas funcionalidades de forma ágil y sencilla.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Como se aprecia, en este caso nos hemos decantado por la inserción de datos mediante ventanas emergentes. Cuando se edita, elimina o añade un usuario, una alerta confirma que se ha llevado a cabo con éxito o nos avisa si se produce algún tipo de error.

## Catálogo de películas y series.

El administrador también puede añadir, editar y eliminar series y películas.

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.** Interfaz de usuario gráfica, Texto, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Para estos casos, nos hemos decantado por diversas “tarjetas” tipo formulario convencional. Como ya se ha explicado, si no se ha seguido el mismo patrón es para probar distintas opciones, aprender más sobre los distintos componentes de Reflex y poder comprobar con más detalle cuál es la opción preferida tanto como por los administradores como por los usuarios llegado el caso.

En cuanto a los usuarios, también tienen un panel sobre el que elegir la opción deseada. En este caso, las opciones pasan por visualizar los catálogos de películas y series disponibles, sus favoritos y vistos, así como un apartado de gráficas estadísticas en el que nos centraremos más adelante.

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.**

De igual manera, se indica de cuántas temporadas consta la serie y cuál de ellas está disponible en la plataforma. **Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.**

Por último, se accede a la página de visualización, desde donde, además de ver el capítulo o película, se puede marcar el mismo como favorito o visto. En caso de que ya se haya hecho anteriormente, lanzará un aviso comunicándolo.

Como se ve en las capturas, cada película tiene una página en la que se describen el año de estreno, duración y sinopsis. En el caso de la serie, tienen sinopsis tanto cada serie como cada uno de los capítulos.

Por último, cada usuario tiene acceso a gráficas estadísticas donde consultar los minutos vistos en películas y series:

**Gráfico, Gráfico de barras

El contenido generado por IA puede ser incorrecto.**

En el caso del administrador, puede comparar a dos usuarios entre sí.

**Gráfico de barras

El contenido generado por IA puede ser incorrecto.**

## Gestión de la base de datos

Todos los datos de usuarios, películas y series, así como los visualizados y favoritos quedan guardados en la base de datos, denominada Reflex.db

De forma segura, quedan registrados los distintos tipos de datos en cada una de las tablas. Usuarios, Películas, Series, Capítulos (vinculada a series) UsuariosCapitulos y UsuarioPeliculas, estas últimas relacionadas con las tablas Usuarios, Películas y series para recoger los datos de visualización y favoritos del usuario.

**Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Aplicación, Tabla, Excel

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Imagen que contiene grande, reloj, gabinete, luz

El contenido generado por IA puede ser incorrecto.Imagen que contiene computer, grande, luz, lado

El contenido generado por IA puede ser incorrecto.**

# MANUAL DE INSTALACIÓN

**IMPORTANTE! No puede haber espacios en la ruta, puesto que entonces nos dará error al ejecutar Reflex.**

**A continuación, explicamos cómo arrancar el proyecto desde Visual Studio Code en Windows.**

* + abrimos VSC y desde el menú archivo pinchamos en abrir carpeta. Seleccionamos la carpeta de nombre Videoflix.
  + Desde el menú Ver, selleccionamos paleta de comandos y pinchamos en crear ambiente Python, posteriormente en Venv. Seleccionamos la versión de Python, para evitar problemas de compatibilidad, se recomienda versiones anteriores a 3.12. En nuestro caso, se ha usado la versión 3.11.8, seleccionamos también el archivo requirements.txt para que se instale con las versiones de trabajo.
  + Una vez creado el entorno virtual, debemos acceder a él a través de la terminal (en el menú superior aparece Terminal para acceder al mismo). 
  + Ya desde el terminal, en el caso de no tener privilegios, se debe ejecutar el siguiente comando: Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process
  + Para arrancar el entorno virtual: venv\Scripts\activate
  + En el caso de que se nos haya olvidado o de error la instalación de requirements, podemos ejecutar el siguiente comando: pip install -r requirements.txt de lo contrario, debemos instalarlo manualmente, copiando los siguientes: pip install reflex==0.6.6.post3 –
  + Para el funcionamiento del proyecto no es necesario, pero si se quieren utilizar y verificar los archivos data\_random.py y subir\_películas.py con el que se han creado datos, se instalarán los siguientes: pip install requests==2.26.0 – pip install pandas – pip instal randomuser – pip install bs4
  + Una vez instalados todos los paquetes, ejecutaremos el comando Reflex run
* No debería dar error, pero en el caso de que lo haga al conectar la base de datos, debemos escribir en el terminal la instrucción: reflex db init, seguido de reflex db makemigrations y reflex db migrate

Con esto realizado, ya podremos acceder a la aplicación a través de la ruta:

[http://localhost:3000](http://localhost:3000/).

Si Reflex detecta que dicho puerto está ocupado, se inicializará en otro y nos lo indicará.

Si por algún motivo el puerto del backend no es el 8000, se debe modificar la constante para que se vean las imágenes que se suben. Esto es debido a que, aunque la forma automática es a través de la función rx.get\_upload\_url(filename), esta no funciona con objetos ni Reflex permite convertirlos a str, por lo que he utilizado la forma menos recomendada (aunque aceptada), que es la ruta de la imagen (<http://localhost:8000//_upload/filename>). Esto aparentemente está corregido para la versión actualizada de Reflex 7.0, pero nuestro proyecto se ha realizado con la anterior.

En el archivo constants.py está, entre otras, la constante:

BACKEND="http://localhost:8000" #cambiar por la url del backend

**Texto

El contenido generado por IA puede ser incorrecto.**

# CONCLUSIONES

## EVOLUTIVOS DEL PROYECTO

Este proyecto es una aplicación web llamada Videoflix, diseñada para gestionar y mostrar información sobre series de televisión y sus capítulos. La aplicación utiliza Reflex para la construcción de la interfaz de usuario y se conecta a una base de datos para almacenar y recuperar información sobre las series y capítulos. A continuación, se presenta una descripción general de los componentes y funcionalidades del proyecto:

Componentes Principales

1. Modelos:
   * [**Series**](vscode-file://vscode-app/c:/Users/zaton/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)**: Representa una serie de televisión** con atributos como título, número de temporadas, año de estreno, sinopsis, y portada.
   * Capitulos: Representa un capítulo de una serie con atributos como título del capítulo, número de capítulo, número de temporada, sinopsis del capítulo, duración, y un enlace al video.
2. Consultas:
   * [Series\_queries](vscode-file://vscode-app/c:/Users/zaton/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Contiene funciones para interactuar con la base de datos y realizar consultas relacionadas con las series.
   * [multiples\_queries](vscode-file://vscode-app/c:/Users/zaton/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Contiene funciones para realizar consultas múltiples, como consultar la base de datos y gestionar los capítulos favoritos.
3. Componentes de la Interfaz de Usuario:
   * [listar](vscode-file://vscode-app/c:/Users/zaton/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Función que recibe un objeto [serie](vscode-file://vscode-app/c:/Users/zaton/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) y genera una tarjeta con la información de la serie, incluyendo la portada, título, número de temporadas, año de estreno y sinopsis.
   * [listar\_cap](vscode-file://vscode-app/c:/Users/zaton/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Función que recibe un objeto [cap](vscode-file://vscode-app/c:/Users/zaton/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") y genera una tarjeta con la información del capítulo, incluyendo el título del capítulo, duración, número de temporada, número de capítulo y sinopsis del capítulo.
   * [mostrar\_serie](vscode-file://vscode-app/c:/Users/zaton/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Función que muestra los títulos de las series en la web utilizando un estado de Reflex.
   * [mostrar\_favourite\_caps](vscode-file://vscode-app/c:/Users/zaton/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Función que muestra los capítulos favoritos en la web utilizando un estado de Reflex.

Funcionalidades

* Visualización de Series: La aplicación permite listar todas las series disponibles en la base de datos, mostrando información relevante como la portada, título, número de temporadas, año de estreno y sinopsis.
* Visualización de Capítulos: La aplicación permite listar los capítulos de una serie específica, mostrando información como el título del capítulo, duración, número de temporada, número de capítulo y sinopsis.
* Gestión de Favoritos: Los usuarios pueden marcar capítulos como favoritos y visualizarlos en una sección dedicada a los capítulos favoritos.
* Interacción con la Base de Datos: La aplicación se conecta a una base de datos para almacenar y recuperar información sobre las series y capítulos. Las consultas a la base de datos se gestionan a través de funciones definidas en los módulos de consultas.

Tecnologías Utilizadas

* Reflex: Utilizado para la construcción de la interfaz de usuario y la gestión de estados.
* Pandas: Utilizado para la manipulación y análisis de datos.
* BeautifulSoup: Utilizado para el web scraping y la extracción de datos de páginas web.
* SQLAlchemy: Utilizado para la interacción con la base de datos.
* Python: Lenguaje de programación principal utilizado en el proyecto.

Este proyecto proporciona una plataforma completa para la gestión y visualización de series de televisión y sus capítulos, ofreciendo una experiencia de usuario intuitiva y funcional.

# Conclusiones

## Conclusiones generales del proyecto

Trabajar con Reflex ha sido todo un reto. A pesar de tener una documentación bastante amplia y llena de ejemplos, hay muy pocos tutoriales. Además, al ser un framework de reciente creación, cuenta con diversos bugs y durante la realización del proyecto se han producido varias actualizaciones que han complicado su elaboración. Es comprensible, dado que, como se ha comentado, está un proceso inicial, pero el hecho de que la documentación se actualizase a la última versión y, a diferencia de otros frameworks, no guarde un archivo con documentación de versiones anteriores ha dificultado su elaboración.

Por poner un ejemplo, la clase AuthState, con la que se manejan los logados de la web no funciona con la nueva versión de Reflex 7.0, debido a los últimos cambios en la configuración de las variables. Además, la documentación para el manejo de la autenticación es casi inexistente en Reflex, hasta el punto de que para crear administradores se ha optado por crear un parámetro booleano admin para validarlo, sin que se haya encontrado ni en la web oficial, reflex.dev, ni en ninguno de los proyectos de ejemplo a los que referencia en GitHub ninguna manera de realizarlo, aunque ya en febrero de 2025 se ha lanzado una librería, reflex-local-auth, que resuelve dichos problemas, no obstante, al tener finalizado el proyecto no se ha utilizado. Por otro lado, los manejos de los estados, para recoger información de las series, películas, etc, es relativamente compleja, más que por su dificultad, dado que una vez aprendida la manera no lo es tanto, como por la ausencia de ejemplos, hasta el punto de que la propia Inteligencia Artificial insertada en la web oficial no ha sabido dar una respuesta clara, teniendo que recurrir a ensayos de prueba y error además de a consultar en el foro de Discord oficial de Reflex.

No obstante, la parte positiva en nuestra opinión supera con creces los aspectos negativos. Reflex es tremendamente adaptativa, está llena de componentes y es perfectamente compatible con componentes de Radiux o Chakra, entre otros, admite html, css, JS y es muy completa en cuanto a opciones de maquetación. Aunque tiene algunas peculiaridades, dado que trabaja al 100% con Python, permite a cualquier programador de dicho lenguaje la creación de una web con su fronted y backend, sin conocimientos previos de css o html, aunque es muy recomendable familiarizarse con las reglas, al menos básicas, de css para que sea de ayuda en la elaboración.

Una vez compilado, trabaja de manera ágil en local y es de esperar que lo haga también una vez puesto en producción, dadas las comprobaciones realizadas de webs creadas con Reflex. Para este proyecto, se ha intentado realizar un código claro, limpio y reutilizable en la mayoría de los casos, aunque para probar diversos componentes se ha optado también en ocasiones, como en algunos formularios, por pegar los que aparecen en Reflex sin mayores modificaciones. Por otro lado, en lo que al manejo de errores se refiere, se ha utilizado para la modificación por parte del administrador de usuarios, series y películas, aunque dado que Reflex de forma predeterminada cuenta con su propio manejo de errores y excepciones y ha sido en ocasiones de gran utilidad para detectar fallos en el código se ha decidido en determinadas clases prescindir de su utilización, no por pereza o falta de desarrollo, sino para no caer en la redundancia dentro del código y centrarnos más en la parte de aprendizaje. Como comentaremos más adelante, de cara a su puesta en marcha cabe la posibilidad de que esa parte sea mejorable, pero desde nuestro punto de vista hemos dado prioridad al crecimiento del proyecto y a la parte de aprendizaje.

## Aprendizajes y experiencias adquiridas

Durante el proyecto, además de poner en práctica lo aprendido durante el curso de programación de Python, se han adquirido varios conocimientos.

Por un lado, a trabajar con un IDE diferente a los vistos hasta ahora como es Visual Studio Code y a familiarizarse con el uso de la terminal para la ejecución de comandos.

También se ha adquirido una grata experiencia en el uso del lenguaje utilizado, Python y a trabajar en proyectos de un tamaño superior al visto hasta ahora, organizando el código en diversos archivos y trabajando constantemente con importaciones.

En cuanto al manejo de las bases de datos, más allá de los conceptos básicos aprendidos hasta ahora, nos hemos familiarizado con estructuras relacionadas más complejas, aprendiendo a realizar operaciones básicas como la suma de elementos y la relación entre las diversas tablas.

Se ha aprendido de igual manera a crear una aplicación web relativamente compleja con un framework nuevo, como es Reflex, adaptándonos a sus cambios y diferencias con los visto anteriormente, como pueden ser tkinter o Flask. Además, nos hemos familiarizado con la búsqueda intensiva de documentación, visualización de tutoriales e incluso al uso productivo de la inteligencia artificial para la resolución de dudas. Es importante destacar en este punto que no se ha pedido en ningún caso a la IA la creación de código, pero sí en algunos puntos la explicación del mismo, o una revisión cuando algo no acababa de funcionar. No obstante, se ha comprobado que la inteligencia artificial es bastante más errática de lo que la gente pueda pensar en un primer momento, aunque nos ha sido en ocasiones de ayuda para salir de ciertas situaciones comprometidas.

Aunque de una manera un tanto distinta a lo que realmente realizan Seaborn o Matplotlib,, dado que se ha utilizado el entorno nativo de Reflex, nos hemos familiarizado con la visualización de estadísticas y datos a través de gráficas.

También se ha aprovechado el proyecto para adentrarnos aún más en el uso de una de las librerías más utilizadas en Python, sobre todo en lo que a la manipulación y análisis de datos se refiere: Pandas.

Para que el proyecto fuese más completo y dotar de más contenido al mismo y a la base de datos, se ha usado por la librería randomuser para la creación de usuarios, de muy fácil manejo para la creación de usuarios, entre otros.

Por otro lado, nos hemos iniciado en el webscrapping, con requests y BeautifulSoup para recuperar datos de webs, concretamente de las tablas de Wikipedia, todo ello junto a pandas para el almacenamiento de los datos junto a sql.

En definitiva, nos hemos adentrado en la creación de páginas webs, su maquetación, control de estados, manejo del backend a nivel principiante, así como a la manipulación de datos con herramientas como pandas y sql o herramientas de visualización de datos como Matplotlib.

# Evolutivos del proyecto

De cara al desarrollo fututo del proyecto, proponemos, entro otras, las siguientes mejoras:

1. Modificar, ahora que está disponible, la autenticación, utilizando la librería reflex-local-auth mencionada anteriormente. De igual manera, habilitar el registro y autenticación con Google o diversas RRSS puede ser una manera de facilitar el registro a los usuarios. Además, proponemos dotar al sistema de registro de la autenticación de dos factores y el uso obligatorio de contraseñas de alto nivel.
2. Habilitar el reseteo de contraseña, así como la configuración del usuario a través de un perfil personalizado.
3. En el apartado de constantes y etilos, se han habilitado una serie de colores. Para la fase inicial se ha optado por un diseño minimalista casi en blanco y negro en su totalidad, con la posibilidad de cambiar a modo claro/oscuro a través del botón a la derecha del header. Proponemos dos opciones:
   1. Utilizar los colores propuestos u otros en caso de que sea deseable su modificación. Es una tarea sencilla a realizar a través del archivo style.py, gracias a las constantes BASE\_STYLE, HEADER\_STYLE y FOOTER\_STYLE
   2. Habilitar (reflex lo trae de manera nativa), una paleta para que sea el usuario quien modifique los colores y estilo de la interfaz a su gusto.
4. En el caso de las series y películas, no se han organizado por categorías, en un primer momento fue un error por nuestra parte, dado que no se tuvo en cuenta para la creación de la base de datos. Se pensó en incluirla posteriormente, pero dado que el proyecto estaba bastante avanzado y que para el aprendizaje consideramos dicha creación redundante, se ha decidido dejarlo tal cual, centrando esfuerzos en otros aspectos. No obstante, para la puesta en marcha definitiva de la web, consideramos interesante valorar dicha organización.
5. La mayor parte del proyecto sigue la misma pauta en lo que al diseño se refiere. No obstante, hay determinados aspectos de maquetación, como pueden ser los alineamientos o tamaños (align o width), que se repiten varias veces. Esto es debido a que se han copiado componentes, “tal cual”, de la web oficial de Reflex. Aunque no afecta en sí, a su visualización, puede ser recomendable ampliar el archivo de constantes para que toda la maquetación quede unificada, además de para simplificar futuras modificaciones.
6. Dotar de la posibilidad al usuario de valorar las distintas películas y series. De esta manera tendremos más información sobre los gustos del usuario, además de ofrecerle de forma personalizada nuevos contenidos. Cabe reseñar que un usuario puede marcar como favorita una película, pero tras la visualización de la misma, puede que el resultado no sea el esperado. Además, no está habilitada la opción de desmarcar de favoritos, lo cual también puede ser aconsejable.
7. A nivel de programación, de igual manera sea aconsejable ampliar el manejo de errores, aunque, como se ha comentado anteriormente, Reflex es muy completo en ese aspecto, las alertas lanzadas al usuario pueden ser confusas y sin sentido para él, aunque de gran valor de cara al programador, razón por la cual antes de su entrada en producción se ha decidido dejar como está.
8. De igual manera, se indicó que se han utilizado distintos tipos de formularios y componentes, de cara a comprobar cuales se valoran de una manera más positiva. Antes de la puesta en marcha definitiva, es recomendable decantarse por un único formato una vez consensuado entre las partes.
9. Por último, en caso de que se detecte algún error o mejora imprescindible, se agradecen sus indicaciones para modificar aquellas partes que se deseen

# Estructura del código (capturas)