TheTvTracker

CIFP Juan de Colonia

Departamento de Informática

Desarrollo de Aplicaciones Multiplataforma

Óscar Ignacio López

Eduardo Liñares Fernández

08-05-2020

Índice

1. Introducción

1.1 - Descripción del proyecto

El nombre de mi proyecto es “TheTvTracker”, y es una aplicación de escritorio escrita en C# junto a .NET Core.

Permitirá a sus usuarios “seguir” distintas series y películas y llevará la cuenta de qué episodios el usuario ha visto, cuándo sale el próximo o, para el caso de las películas, avisará de su estreno.

1.2 - Justificación

Personalmente, muchas veces he comenzado a ver series con calendarios de publicación irregulares y simplemente me he olvidado de ellas, al no tener un recordatorio de cuándo se publica el siguiente episodio.

Asimismo, en algunos casos las series suspenden de forma temporal la publicación de nuevos episodios, y muchos seguidores pierden el hilo de lo que han visto y lo que no.

Tener una herramienta como esta facilitaría mucho estas situaciones.

1.3 - Cambios

Originalmente, la idea era realizar una aplicación móvil utilizando Xamarin y Xamarin.Forms, pero esta plataforma presenta numerosas limitaciones que impiden realizar muchas tareas básicas.

Además de esto, estas plataformas funcionan bajo Mono, en lugar de .NET Core. Las diferencias entre .NET y Mono se explicarán más adelante, así como las limitaciones de la plataforma Xamarin actualmente.

2. Planificación

2.1 - Requisitos Funcionales

Aquí trataré de definir las funciones que la aplicación debe poder realizar.

- Sistema basado en perfiles (usuarios).
- Cada usuario podrá ser administrado de forma individual.
- Cada usuario o perfil puede seguir distintas series o películas. En el caso de las series, poder registrar los episodios como "vistos" o no.
- En el caso de los episodios que aún no han sido estrenados, se mostrará el tiempo restante hasta ese día.

2.2 - Requisitos No Funcionales

Por el contrario, estos serán los aspectos de la aplicación que no están asociados necesariamente a su funcionalidad.

- TheTvTracker será una aplicación de escritorio.
- Sin instalación. "Portable".
- Multiplataforma.

2.3 - Recursos Hardware

Sólo es necesario un ordenador para el desarrollo como tal de la aplicación.

Puesto que es una aplicación de escritorio, se ejecutará dentro del ordenador de cada usuario, y no será necesario un servidor para la base de datos, ya que está también será local a cada ordenador.

2.4 - Recursos Software

Para el desarrollo de la aplicación, puesto que se ha optado por un ecosistema .NET, el IDE utilizado es Visual Studio 2019.

Se ha utilizado Git como sistema de control de versiones y más concretamente el servicio que ofrece GitHub de alojamiento de proyectos.

El resto de tecnologías y recursos del proyecto son de código abierto o bien de acceso gratuito.

En apartados posteriores se hablará con más detenimiento de cada tecnología utilizada por separado.

2.5 - Planificación Temporal

Tarea	Tiempo Máximo Esperado
Arquitectura base de datos y modelos	De 3 a 5 horas
Elegir stack de tecnologías	3 días (de 9 a 18 horas)
Diseño de pantallas	5 horas
Programación	Hasta entrega del proyecto, mín. 20h
TOTAL	28 horas + programación (aprox. 48h)

Estos tiempos, por supuesto, son orientativos y el desarrollo de la aplicación, probablemente, lleve mucho más tiempo.

2.6 - Planificación Económica

Para el desarrollo de la aplicación se han elegido tecnologías de código abierto o bien gratuitas.

Si bien es cierto que el IDE Visual Studio ofrece algunas opciones de pago, no he hecho uso de ninguna función restringida, osea que no podemos contabilizarlo.

Podemos tener en cuenta como gasto económico el gasto eléctrico del ordenador, pero es complicado de cuantificar.

Dentro de la planificación económica también hay que tener en cuenta el dinero que habríamos ganado por realizar esta aplicación dentro de un contexto laboral real.

Utilizando el convenio colectivo de Consultoría y Estudios de Mercado y de la Opinión Pública, podemos obtener los siguientes datos.

1492.6€ ---------- 30 días

X€---------- 4/5 días

X = 5 * 1492.6 / 30 = 249€ aprox.

Obtendríamos, aproximadamente, 250€ si tomamos las 30h del proyecto y las dividimos en jornadas de 8h.

3. Tecnologías

3.1 - Tecnologías

Aquí se especificarán todas las librerías y/o tecnologías usadas para el desarrollo del proyecto. Se incluirán enlaces de acceso a la página o documentación de cada librería.

TheTvTracker es una aplicación de escritorio, programada en C# y utilizando .NET Core.

.NET Core (enlace)

.NET Core es el futuro de la plataforma .NET, creada por Microsoft. A diferencia de su predecesor, permite programación multiplataforma, de forma que una aplicación programada en Windows usando esta tecnología puede funcionar también bajo Linux y Mac.

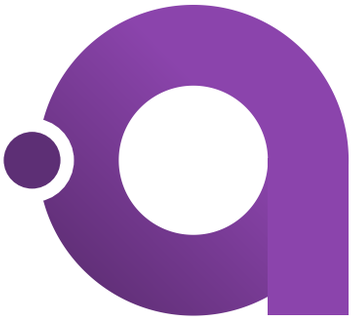
Es una tecnología de código abierto, lo que la hace extremadamente atractiva para los desarrolladores, particularmente aquellos que antes "desconfiaban" de Microsoft y las herramientas que proporcionaban y aquellos a los que les gusta controlar hasta el último de los detalles de las aplicaciones que desarrollan.

Avalonia (enlace)

Avalonia es un conjunto de librerías de código abierto que permiten desarrollar interfaces gráficas que funcionen tanto en Windows como en Linux y Mac sin ningún trabajo adicional por parte del programador.

Se basa en WPF, de Microsoft, y será muy familiar para aquellos desarrolladores que hayan trabajado previamente con este.

Incluye además ciertas facilidades respecto a WPF en lo referente a su sistema de estilos, similar al propuesto para la web con CSS, y su sistema de bindings que permiten seleccionar más fácilmente otros controles y sus propiedades.



LiteDB (enlace)

Para la base de datos, se ha utilizado LiteDB, una pequeña base de datos NoSQL preparada para el desarrollo multiplataforma.

Esta base de datos estructura sus datos en documentos, de forma similar a como lo hace MongoDB.

RestSharp (enlace)

RestSharp es una librería que facilita la comunicación con APIs REST desde C#.

Permite realizar peticiones, tanto de forma asíncrona como de forma síncrona a un servidor y recoger los datos que devuelva.

Esta librería es clave para el proyecto, ya que es la que se encargará de realizar las consultas para obtener los datos de las series y las películas.

Newtonsoft JSON.NET (enlace)

Esta librería nos permitirá transformar los datos obtenidos por la API mediante RestSharp. Actúa como una especie de ORM que convierte el JSON en objetos de C#, aunque en el proyecto solo lo he utilizado para obtener valores sueltos de la abundante información.

4. Desarrollo y Secuencia Temporal

4.1 - Estructura de la Aplicación

La aplicación sigue el modelo MVVM, utilizando la librería ReactiveUI, que viene incluida en Avalonia.

Esto quiere decir que cada vista tiene asociada a su vez un "ViewModel", que es una clase específica que contendrá todas las operaciones que podrá realizar esa vista en concreto.

Además de esto, dentro de lo posible, la aplicación está dividida en capas capaces de realizar sus funciones de forma independiente las unas de las otras.

4.2 - Estructura de la Base de Datos

La base de datos está estructurada de forma intrínseca en documentos BSON (similar a JSON, pero en formato binario), que es el mismo tipo de documento que utiliza MongoDB.

Cada Usuario representa un documento, que a su vez dentro tiene otros documentos (Series y Películas).

En lo respectivo a las imágenes, la base de datos sólo contendrá la ruta de las imágenes, aunque LiteDB ofrece opciones de almacenaje de ficheros.

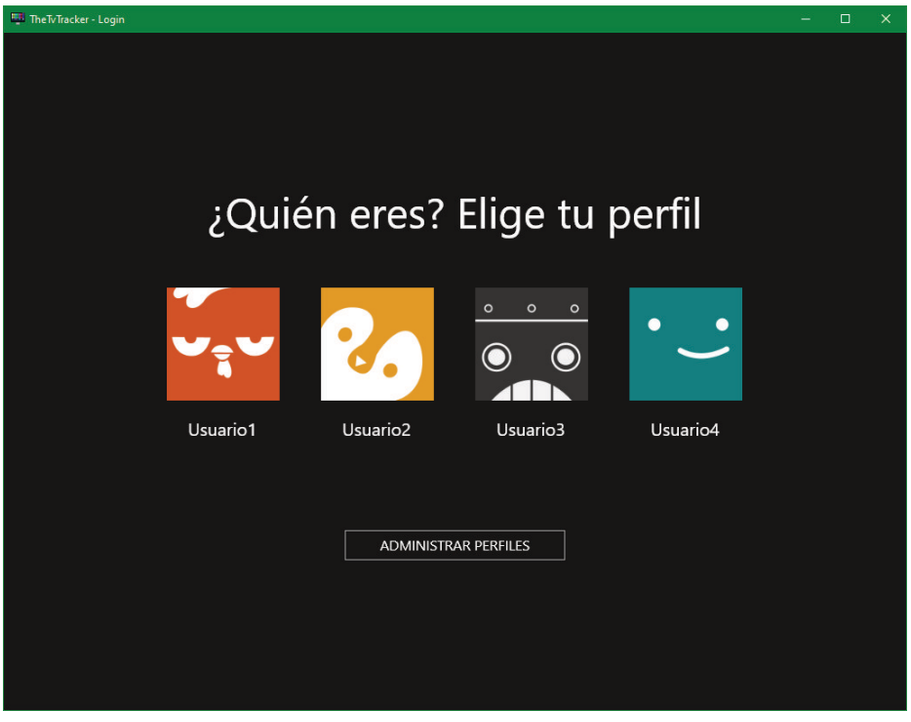
4.3 - Diseño de la Aplicación

Cada framework de interfaces de usuario define un estilo propio asociado al entorno de destino.

Con Avalonia, es complicado conseguir esto, ya que una misma app se puede ejecutar en distintos SO.

Debido a esto, he aprovechado dentro de lo posible el sistema de estilos de Avalonia para darle a la aplicación un diseño similar al propuesto por Netflix. Un diseño que se basa en destacar lo más importante: el contenido multimedia (en nuestro caso, imágenes).

4.4 - Ejemplos del Diseño



4.5 - Control de Errores

Es complicado hacer una gestión de errores de cosas que escapan a nuestro control, como la disponibilidad de una API externa, o que los datos que nos proporcione esa API tengan siempre la misma estructura.

Durante el desarrollo, varios errores referentes a estas situaciones han surgido.

Algunas películas no tienen un póster asociado, o el archivo no está donde la API indica. También hay algunas que no poseen una descripción, entradas duplicadas, etc.

Esta situación es comprensible, pero no hay mucho que podamos hacer al respecto excepto utilizar try catch en nuestra aplicación.

Si una imagen concreta no existe, el programa simplemente no la mostrará. Esto afecta a la estructura de la interfaz, pero al menos el resto de datos, si están disponibles, se muestran.

Avalonia tiene su propio procesamiento de errores. Por ejemplo, si un binding falla, no saltará ningún error, pero ningún valor será mostrado.

4.6 - Limitaciones de las Tecnologías

Como ya he mencionado, al principio este proyecto iba a ser una aplicación móvil hecha con Xamarin junto a Xamarin.Forms, pero este conjunto de tecnologías presentan varias limitaciones.

No hay, por ejemplo, numerosos controles que serían necesarios para el desarrollo del proyecto, y el rendimiento deja bastante que desear, incluso con un proyecto básico.

Las cuestiones de rendimiento se deben a que por debajo de una aplicación Xamarin se ejecuta Mono, un entorno similar a .NET Core que fue desarrollado por la comunidad antes de que .NET Core fuese concebido.

Mono permite ejecutar en cualquier plataforma/arquitectura

código C#, pero el rendimiento es inferior. En el caso de los dispositivos móviles, esta situación se ve exacerbada porque varios entornos de ejecución se ejecutan el uno dentro del otro.

5. Conclusiones

5.1 - Grado de Cumplimiento

Hacia el final del desarrollo, diversos problemas con el sistema de acceso a datos a la API me han forzado a acotar la funcionalidad de la aplicación.

Al principio, mi intención era que pudieses tener en tu librería tanto series como películas, y poder controlar la fecha de salida de los episodios, etc.

Finalmente, la parte de las series ha quedado descartada como posible futura mejora de la aplicación por falta de tiempo.

Asimismo, el hecho de que la estructura de los datos recibidos de la API no sea siempre la misma me ha forzado a elimintar también el objetivo de controlar las fechas de estreno de las películas.

Ver que la API ofrecía también las imágenes de forma gratuita ha llevado a poder visualizar en la aplicación estas imágenes, descargándolas previamente. Esta no era una funcionalidad pensada desde el principio, pero hace más dinámica la aplicación.

5.2 - Agradecimientos

La API de TheMovieDatabase (TMDB) ha sido esencial para el desarrollo de la aplicación y me gustaría utilizar este pequeño espacio para agradecerles la oferta de su API de forma gratuita a cualquier desarrollador.

6. Guías

6.1 - Guía de Instalación

La aplicación es portable, ya que actualmente es imposible crear un instalador multiplataforma sin usar herramientas de pago.

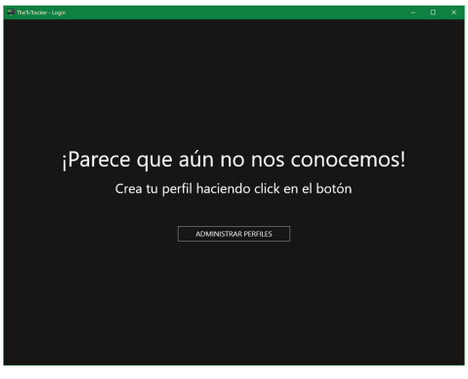
Simplemente se descarga el archivo zip del proyecto y se abre el archivo exe ejecutable.

El proyecto está compilado de forma que no es necesario tener .NET Core instalado en el ordenador para su funcionamiento.

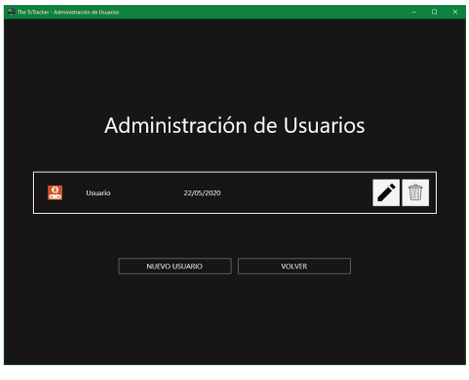
6.2 - Guía de Uso

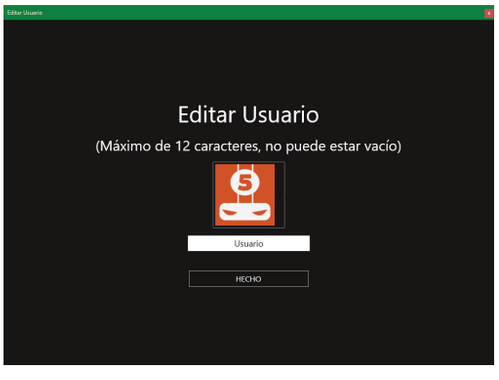
El uso de la aplicación es sencillo. Dentro de lo posible se han utilizado pocos controles que puedan distraer al usuario.

Al abrir la aplicación por primera vez se inicializará la base de datos y se informará al usuario de que debe crear un perfil.

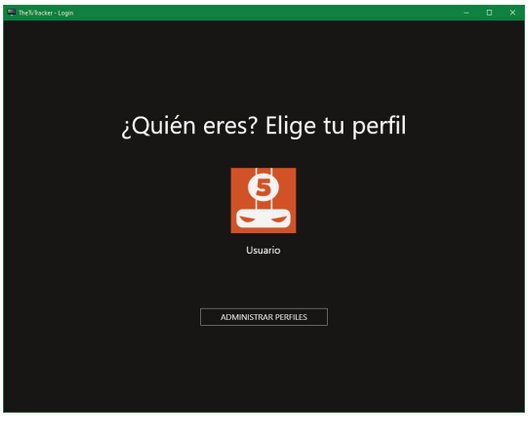


Si entra a la pantalla de administración de usuarios podrá crear el suyo propio, eligiendo uno de los avatares disponibles.



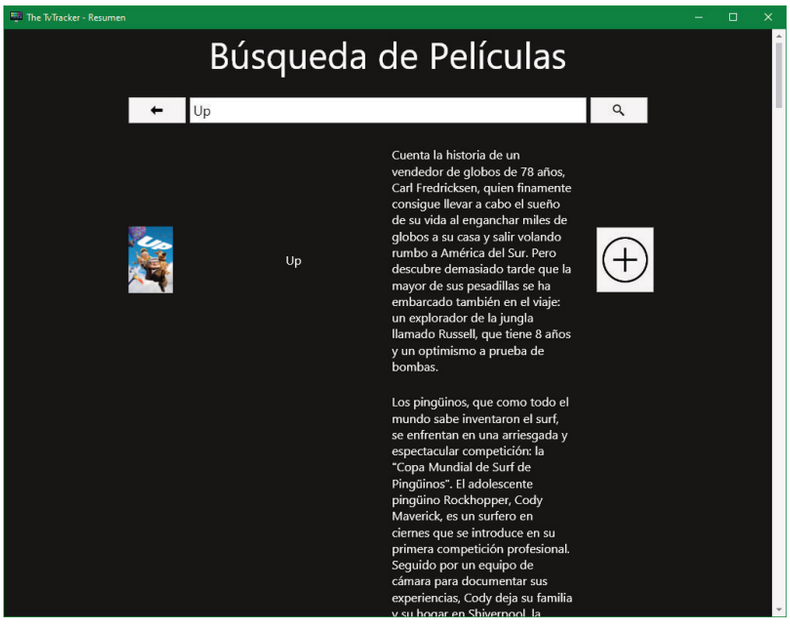


Hecho esto, podrá acceder con su perfil a la aplicación y ver su pantalla de resumen.

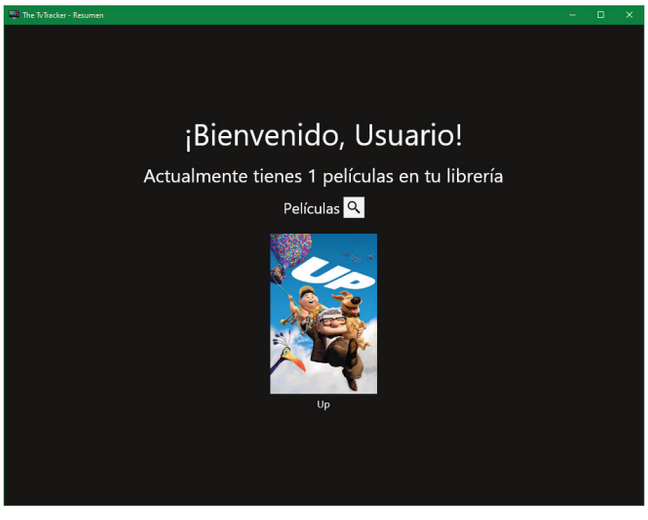


Desde la pantalla de resumen podrá navegar a la pantalla de búsqueda de películas.

Los resultados aparecerán al buscar y podrá añadir películas a su colección.



Una vez añadidas, aparecerán en su pantalla de resumen.



7. Bibliografía

7.1 - Recursos Investigados

Los enlaces de cada librería dirigen a su documentación, que ha sido esencial para el desarrollo.

Además de estos recursos, también he visto algunos vídeos de YouTube para ver cómo desarrollar con Avalonia, y ejemplos de aplicaciones en funcionamiento.

https://www.youtube.com/watch?v=rho26Ik30D4

https://www.youtube.com/watch?v=G0iIATBWAkw