03. Incluir Flow en el ViewModel para gestionar el estado de la Activity

Esta aplicación es una iteración de la aplicación 2. Ahora se quiere que la aplicación Android sea reactiva en lugar de imperativa que era como estaba programada la versión anterior. Para conseguir este objetivo ello se emplean los Flow.

En `FilmsRepository`, la funcionalidad `queryFilm()` devuelve un `Flow<List<Film>>`. Con esto se pretende representar que la función devolverá un flujo de objetos de tipo `List<Film>`. Esto implica que, de forma asíncrona, se podrá recibir a través de dicho flow objetos `List<Film>` a lo largo del tiempo. La idea es preparar la aplicación para el caso de uso en el que se desee estar permanentemente actualizado cuando haya cualquier alteración en las películas almacenadas en el repository. Es decir, más adelante, cuando se añadan, se eliminen o se modifiquen películas, el Flow devuelto por `queryFilm()` emitirá un nuevo `List<Film>` con los nuevos datos.

En `MainViewModel` se define un `StateFlow< List<Film> >` (atributo `films`). Este flow será al que se suscriba `MainActivity` para recibir a través de él la lista de películas. Podemos pensar en un `StateFlow` como una variable. Esta variable puede cambiar de valor a lo largo del tiempo. En lo que se diferencia de otras variables normales es que tiene la capacidad de permitir que otros objetos se suscriban a ella para ser notificados cuando el `StateFlow` cambie de valor. En ese momento el `StateFlow` notificará a todos sus suscriptores enviándoles su nuevo valor. Es más, si en cualquier momento se suscribiera a él un nuevo suscriptor, automáticamente le enviaría una notificación con el valor que en ese momento tuviera. De la misma forma que otros objetos se pueden suscribir al `StateFlow`, también pueden cancelar su suscripción (momento en el que dejarían de recibir notificaciones)

Será `MainActivity` quien se suscriba al `StateFlow`. Lo hará cuando su ciclo de vida sea STARTED: `repeatOnLifecycle(Lifecycle.State.STARTED)`. Que su ciclo de vida sea STARTED implica que recibirá las notificaciones del StateFlow mientras la Activity esté visible. Además, automáticamente se cancelará la suscripción cuando el estado de la Activity deje de ser visible, esto es, cuando el ciclo de vida de la Activity transite al estado no visible (STOPPED) o cuando se finalice la activity. Lo que hace la Activity con las notificaciones que recibe del `StateFlow` (cada notificación representa el nuevo valor que tenga el StateFlow) es actualizar la lista de películas del *adapter* del `RecyclerView` y, por tanto, visualiza inmediatamente en pantalla la nueva lista de películas.

De cara al Activity, el `StateFlow` es un *flow* de sólo lectura. Esto implica que Activity no tiene la capacidad de modificar el valor del `StateFlow`.

Por otra parte, `MainViewModel` define un atributo `MutableStateFlow` (el atributo _films). Este tipo de dato es una versión de `StateFlow` al que sí se le podrá modificar su valor. Hay que tener en cuenta que al haberse declarado el atributo con visibilidad `private`, sólo los métodos del view model serán quienes puedan modificar el valor del `MutableStateFlow`.

En `MainViewModel` el atributo `films` (de tipo `StateFlow`) representa una "vista" de sólo lectura del `MutableStateFlow` _films. Así, cuando se modifique el valor de `_films`, automáticamente se "habrá modificado" el valor del atributo films y, por tanto, notificará a sus suscriptores de este hecho.

Pues bien, el método `queryFilms()` de `MainViewModel` invoca al método `queryFilms()` del repository. Este le devuelve un Flow a través del que, de forma asíncrona, recibirá listas de películas. Al suscribirse al dicho flow (operación `collect`) se indica que en el momento en el que reciba una notificación con un nuevo `List<Film>`, se modificará el valor del `_films` (el `MutableStateFlow`) con dicho valor (`_films.emit( it )`). Esta modificación será recibida inmediatamente por el Activity a través de la notificación que le envíe el `StateFlow`. El activity reaccionará ante esta notificación mostrando los datos en el `RecyclerView`.