08. Ejecución de acciones cuando se realiza el swipe de los ítems del RecyclerView.

En esta iteración de la aplicación se ha cambiado la implementación del *repository*. Ahora, además de consultar todas las películas (`queryFilms()`), también se puede eliminar una película a la que identificaremos por su id (`removeFilm(id: Long)`) así como también podremos añadir una nueva (`addFilm(film: Film)`).

No sólo se ha añadido más funcionalidad, también se consigue que permanentemente esté actualizada la información de las películas. Ahora la funcionalidad `queryFilms()` devuelve en forma de *flow* un `MutableFlow` que actualizaremos cada vez que eliminamos o agregamos una nueva película. Esto implica que a través del *flow* devuelto por `queryFilms()` llegará en forma de notificación la lista actualizada de películas al *view model*.

En el *view model* también se ha alterado la definición del Item de tipo `FilmItem` de forma que tenga métodos `edit()` y `remove()` que representen las acciones a realizar cuando el usuario selecciona una de las opciones que aparecen al hacer *swipe* sobre un item del RecyclerView. Estos métodos a su vez invocan, respectivamente, a los métodos `onEditFilm()` y `onRemoveFilm()` del *view model*:

- `onEditFilm()` envía a través de `_events` un evento de tipo `Event.OnFilmEdit`. Este evento incluye la información de la película que se quiere editar. El suscriptor de `events` (Flow que representa una vista de sólo lectura de `_events`) será la Activity, quien mostrará un mensaje en pantalla a través de un Snakebar.

- `onRemoveFilm()` invoca a `FilmsRepository#removeFilm(id: Long)` para eliminar la película. Esto provocará que se elimine la película y que automáticamente llegue una notificación a través del *flow* al que nos hemos suscrito en `MainViewModel#queryFilms()` por lo que también se actualizará los datos del RecyclerView.
  Por otra parte, en `onRemoveFilm()` también se emite a través de `_events` un evento de tipo `Event.OnFilmRemoved`. Este evento incluye la información de la película eliminada. Este evento, cuando lo procese la Activity mostrará en pantalla un Snakebar que informará de la eliminación de la película. A este Snakebar se le añadirá una acción «*Undo*» con el propósito de revertir (o deshacer) el efecto de la eliminación. Para ello, en caso de ser aceptada por el usuario, provocará que se invoque al método `MainViewModel#addFilm()` pasándole como parámetro la película que fue originalmente eliminada. `MainViewModel#addFilm()` invocará a `FilmsRepository#addFilm()` que, al agregar la película, forzará la

emisión nuevamente del listado de películas y, consecuentemente, su actualización en el RecyclerView.

Por otra parte, se quiere que al seleccionar el usuario una de las opciones que aparecen al hacer *swipe* sobre los ítems del RecyclerView, además de desencadenar la acción correspondiente (eliminación o edición de la película) también se fuerce al *view model* a realizar un reset() a fin de forzar al Motionlayout a volver al estado inicial de su *transition* en curso (se cierre el swipe). A tal fin se ha definido la interface Action.
Esta interface define una funcionalidad que permite la ejecución (método perform) de una acción (definida como una expresión lambda que no recibe parámetros y devuelve Unit).
En el `layout R.id.film_item` se ha definido la variable «action» de tipo `Action`. Además, a los widgets `R.id.btRemove` y `R.id.btEdit` (los botones que representan las opciones del swipe) se les asigna a través de su atributo `android:onClick` una *binding expression* que representa lo que debe ejecutarse al ser pulsados. En ambos casos, se utiliza la variable «`action`» para indicar que la acción que debe ser ejecutada sea, respectivamente, `filmItem.onRemove` y `filmItem.onEdit` (atributos de `Item.FilmItem`).

En el *view holder* `FilmItemViewHolder` se fija el valor de la variable «`action`». Este será una implementación de la interface `Action` que invocará la acción recibida como parámetro (los valores `filmItem.onRemove` y `filmItem.onEdit` enviados por las *binding expressions* anteriores) y además forzará el `reset()` del MotionLayout.