09. Ejecución de la acción edit del swipe y de la creación de nuevas películas.

En esta iteración se quiere implementar la acción "Edit" del swipe y además se quiere añadir también la funcionalidad que permita añadir nuevas peliculas. Estas dos funcionalidades las se realizarán en la nueva activity FilmActivity.

Puesto que los view models de las activities MainActivity y FilmActivity deben acceder al repository, la instanciación de este se realiza en el objeto *application* `FilmsApplication`. Esta clase, además de extender de `Application`, también debe registrarse en el fichero AndroidManifest.xml (atributo `name` del elemento `<application>`) La clase *application* actúa como un *singleton*.

En los *view models* se ha modificado su objeto `factory` que es el que facilita la creación de éstos. Ahora, desde el parámetro `extras` se obtiene el objeto *application* y de él el objeto repository que se utilizará para la creación de los view models.

Se ha cambiado por completo la implementación de FilmsRepository. Se incluye todas las funcionalidades que permitirían realizar un CRUD:

- queryFilms(): consulta todas las películas.
- queryFilmById(id: Long): consulta una pelicula identificada por su id
- removeFilm(id: Long): elimina la película identificada por id
- insertFilm(film: Film): añade una nueva película
- updateFilm(film: Film): actualiza los datos de una película.

En la implementación de estas funcionalidades no habría que descartar que se produjesen excepciones mientras se ejecutan (piénsese por ejemplo que los datos puedan residir en una base de datos o que se deba acceder a un servicio web). Habría que evitar que estas excepciones se propagasen al view model. Para ello, todos los métodos van a devolver ahora un Flow<Result>.

Result es una interface que se ha definido de forma genérica. Esta interface tiene dos posibles implementaciones: Result.Ok y Result.Error. Se quiere que las funcionalidades del repository devuelvan un flow conteniendo un objeto Result.Ok si la ejecución se ha realizado sin problemas o bien contenga un Result.Error en caso de que se haya producido un error (se haya propagado una excepción mientras se procesa el método).

Como se ve en la implementación de los métodos del repository, todos emiten a través de flow {...} el dato que, de alguna forma, se esperaría obtener (un List<Film>, o un Film, etc) Pues bien, a cada uno de esos flows se le aplica una nueva función de extensión que convierte dichos flows en otros de tipo Flow<Result>. Esta función de extensión es Flow<T>.asResult(). Esta función, convierte el dato notificado por el flow a un objeto Result.Ok y, en el caso de

que el flow propagase una excepción, emitiría un objeto Result.Error conteniendo el mensaje de la excepción.

De esta forma, el view model siempre recibirá un Flow<Result>. Los subscriptores (operaciones collect {}) tendrán que determinar si el tipo del Result es Result.Ok o Result.Error pudiendo en cada caso obtener el dato subyacente (List<Film>, Film, etc o bien el mensaje del error) y proceder en consecuencia.

Por otra parte, para que desde MainActivity se abra FilmActivity, se invocará al método startActivity(). Esta operación se realiza en el suscritor de events cuando el evento procesado sea un Event.OnEditFilmRequested o un Event. OnInsertFilmRequested. Al método startActivity() se le pasa como parámetro un objeto Intent que representará la actividad a arrancar. Para ello, cuando se crea el objeto Intent, es necesario especificarle en su constructor la clase del activity a arrancar (FilmActivity::class.java). Opcionalmente se puede agregar más datos extras que se considere necesarios (piénsese que el intent contiene un Bundle al que se le pueden agregar datos extras):

- En caso de editar la película, se le pasan los extras FilmActivity.EXTRA_ACTION y FilmActivity.EXTRA_ID que contendrán respectivamente los valores FilmActivity.ACTION_UPDATE (indicando que la acción a realizar es una actualización) y el identificador de la película.
- En caso de insertar una nueva película, al Intent se le pasa el extra FilmActivity.EXTRA_ACTION con el valor FilmActivity.ACTION_INSERT.

Cuando arranque FilmActivity, su view model FilmActivityViewModel podrá acceder a los extras de estos intents mediante el objeto SavedStateHandle.

El estado de FilmActivity se representa por medio de un objeto FilmUI que contendrá o bien los mismos datos de la película a editar o bien los valores por defecto (`defaultFilm`) si la película en lugar de editarse se estuviera creando. Los atributos de la clase serán de tipo var (variables) ya que en el layout de la activity (R.layout.activity_film) se empleará two-way data binding. Además, los atributos poster, voteAverage, tags y releasedate son de tipo MutableSstateFlow ya que se quiere que al modificarse su valor se desencadene la actualización en la UI.

En Bindings.kt se definen *binding adapters* e *inverse binding adapters* para los widgets ImageView, ChipGroup y RatingBar del layout.