04. Mostrar distintos tipos de vistas en el RecyclerView

En esta iteración se quiere que la lista de películas mostradas a través del RecyclerView estén ordenadas y agrupadas por el año en que estrenaron.

Ahora el adapter del RecyclerView no recibirá un List<Film> como dato, sino que recibirá una List<Item>. Item es un *interface* del que existen dos posibles implementaciones: FilmItem y YearItem. FilmItem es un item que representa una película. YearItem es un item que representa un año y el nº de películas estrenadas ese año. De esta forma, el List<Item> contendrá dos tipos de ítems: los de tipo película y los de tipo año. La idea es que después de un item de tipo YearItem le sigan items de tipo FilmItem correspondientes a las películas estrenadas ese año.

Al haber dos tipos de ítems, cada tipo se visualizará en pantalla de forma distinta a través de dos tipos de layouts (R.layout.film_item y R.layout.year_item) y por tanto dos tipos de view holders (FilmItemViewHolder y YearItemViewHolder).

Para hacer esto posible, es necesario que el adapter redefinia el método getItemViewType(position: Int): Int. Este método deberá devolver un valor entero que represente el tipo de view a utilizar para mostrar el item de la lista con posición position. El valor del entero devuelto lo define el programador. Aquí se ha determinado devolver el identificador del layout a utilizar para mostrar cada tipo de item.

El método onCreateViewHolder() recibe el entero que representa el tipo de vista (el valor devuelto por el método getItemViewType()) y en función de este valor crea el view y view holder correspondientes.

Cada view holder tendrá su propia versión del método bind() puesto que el dato se mostrará de forma distinta según sea el tipo del view holder.

Puesto que los ítems de la lista ya no son objetos Film sino objetos Item, el DiffUtil.ItemCallback<Item> del ListAdapter también será distinto.

Por su parte, el view model también sufre modificaciones. La lista de películas recibida desde el repository habrá que transformarla en una List<Item>. Para ello, en el MutableStateFlow se realizarán una serie de operaciones que realicen esta tranaformación:

1. Ordenar las películas por el año de estreno
2. Convertir cada película en un objeto FilmItem: pasamos así de un List<Film> a un List<FilmItem>)
3. Agrupar FilmItem por el año de emisión de la película: de esta forma se pasa de un List<FilmItem> a un Map<Int, List<FilmItem>>. En este mapa cada entrada representa las películas estrenadas en el mismo año. El año año será el valor del clave y el List<FilmItem> será el valor vinculado a la clave.

4. Crear un List<Item> a partir del Map<Int, List<FilmItem>>. Por cada entrada del mapa se añadirá a la lista resultante en primer lugar un item de tipo YearItem correspondiente al valor de la clave del mapa (es decir, al año de estreno) y seguido a él todos los FilmItem contenidos en la List<FilmItem> (valor de la entrada del mapa).

El valor resultante de aplicar estas operaciones es un Flow<List<Item>> que a través de la operación stateIn() se "convertirá" en un StateFlow