Universidad Nacional

Escuela de Informática

EIF-411

 Diseño y programación de plataformas móviles

Reporte Front End Móvil

Profesor

Juan de Dios Murillo Morera

Estudiantes:

Djenane Hernández Rodríguez - 504210953

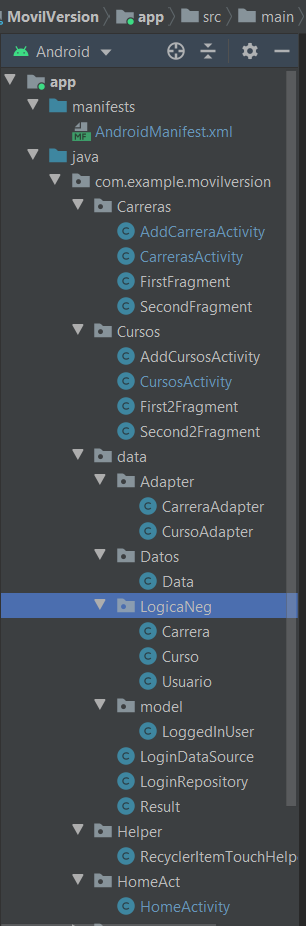
Diego Monterrey Benavides - 116810263

I ciclo 2020

Reporte Front End Móvil.

En este reporte, se va a contemplar características, componentes, actividades, navegación, etc. Representa la parte del Front End de una aplicación móvil.

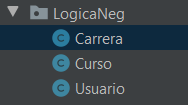
En sí, el contenido de las carpetas se estructura de la siguiente forma.



En este caso, se escogieron dos módulos, Carreras y Cursos, para esta parte del Font End solamente.

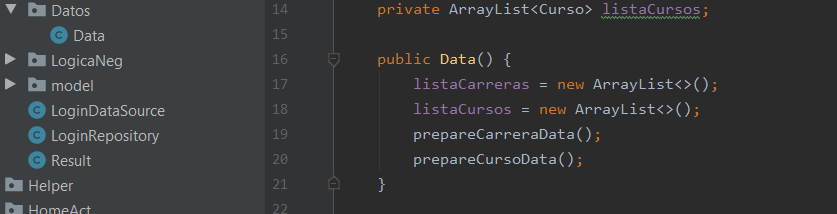
***Lógica de Negocio.***

Definidos diferentes clases para simular un tipo de base de datos, en este caso las clases de Carrera y Curso, además de Usuario que pueden entrar en la aplicación. Son clases Java comunes y corrientes.



***Data.***

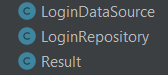
Carpeta en donde se va a simular un tipo de bases de datos, aquí vamos a crear diferentes arrays de las diferentes clases mencionadas arriba, para poder acceder a ellas cuando las necesitemos.



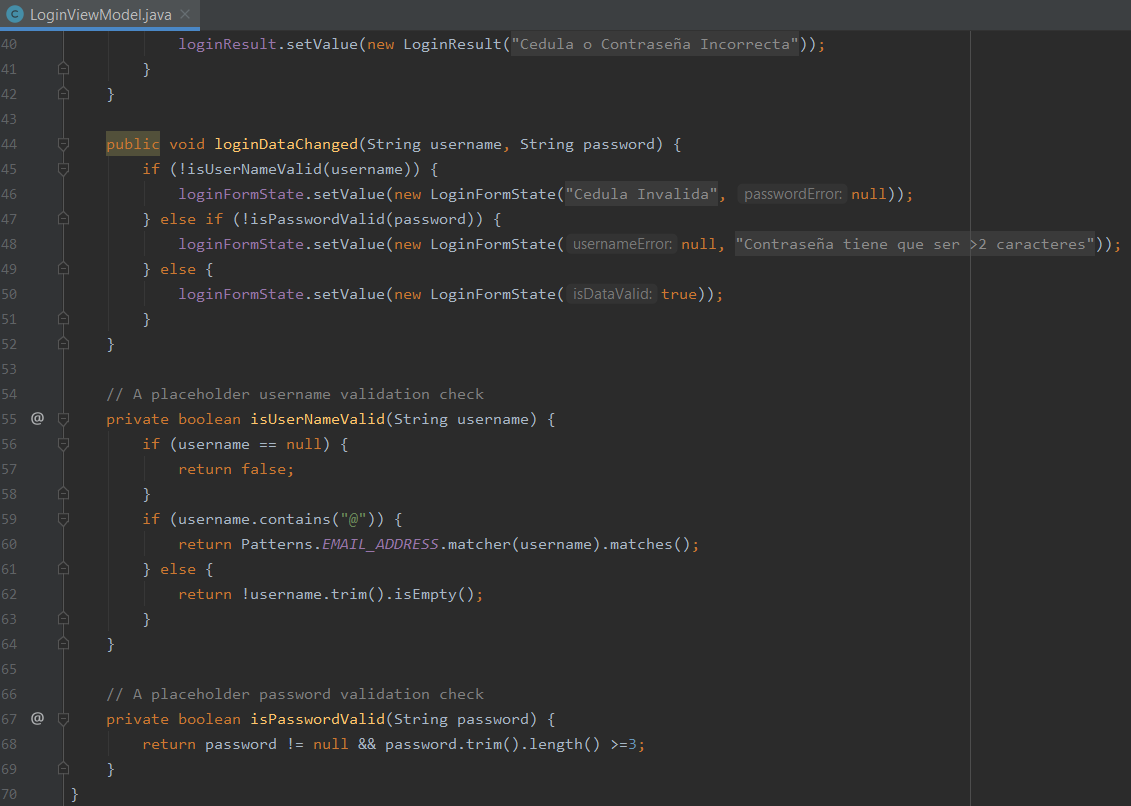
***Login Activity.***

Para comenzar el laboratorio se utilizó la plantilla de el de Login Activity, en esta actividad debemos hacer una comprobación de que el usuario exista, se ingrese la contraseña o usuario correcto.

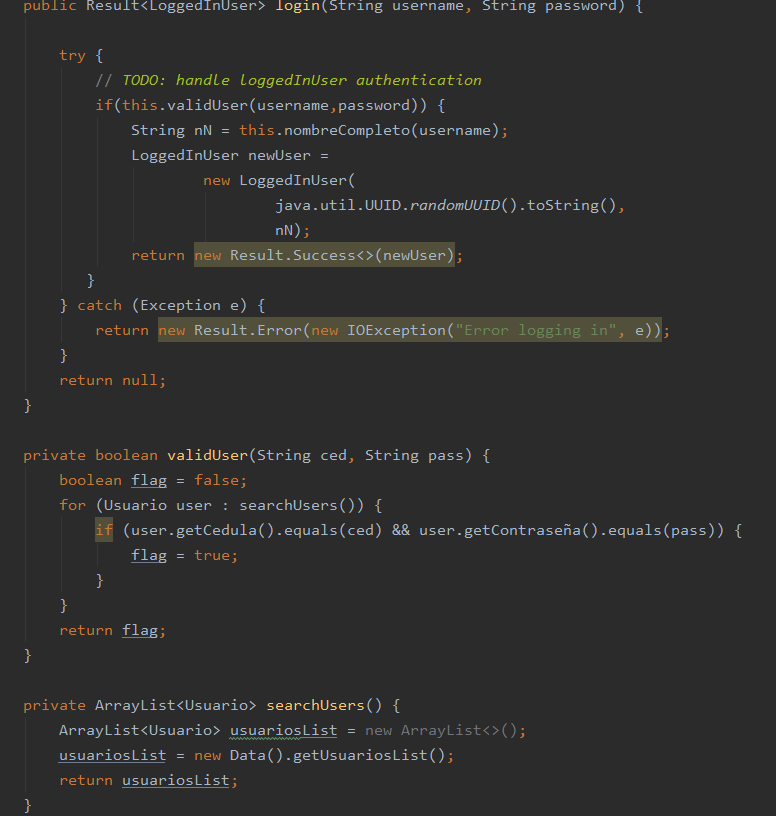
El Login activity nos genera muchas clases en java, por ejemplo.

En el userView, simplemente para hacer display del nombre de la persona, el siguiente es la actividad, FormSate, para identificar unos errores, Result, resultado de la operación, ya sea bueno o malo, ViewModel, para poder darle un modelo al Login, por ejemplo, caracteres de más de dos, si el usuario no se identificó, etc.

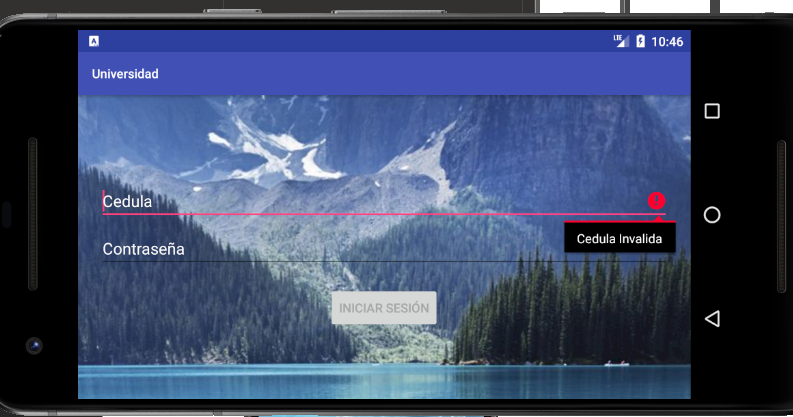
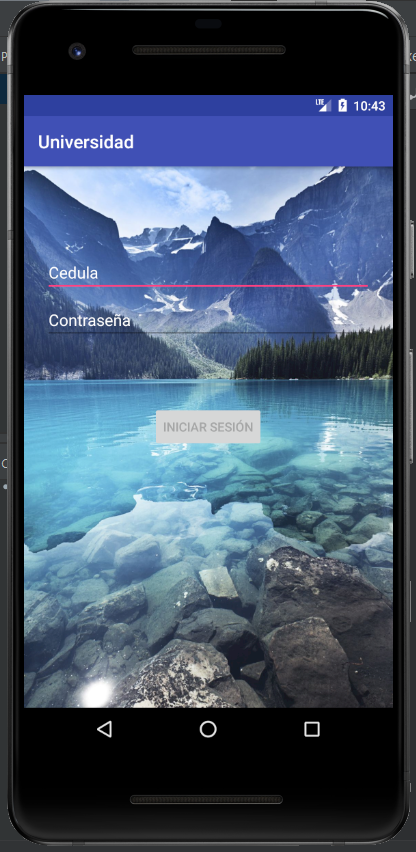


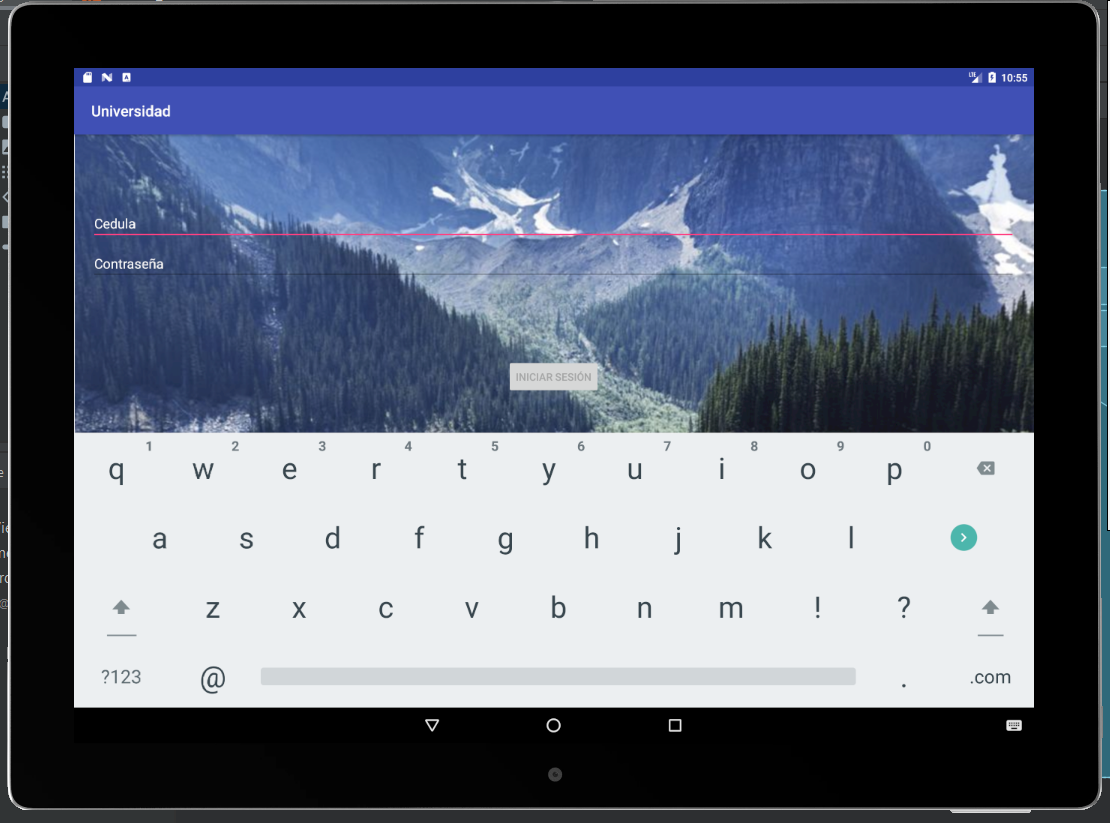
Por la parte del LoginDataSource, aquí es donde podemos hacer las operaciones de autentificación del usuario, por ejemplo.



Cuando se le da al botón de iniciar sesión, este entra al método de Login, para por validar usuario, en este método, chequea si el usuario existe y si se escribió la contraseña correcta, si es success, pasara a la siguiente actividad identificándose con el nombre.

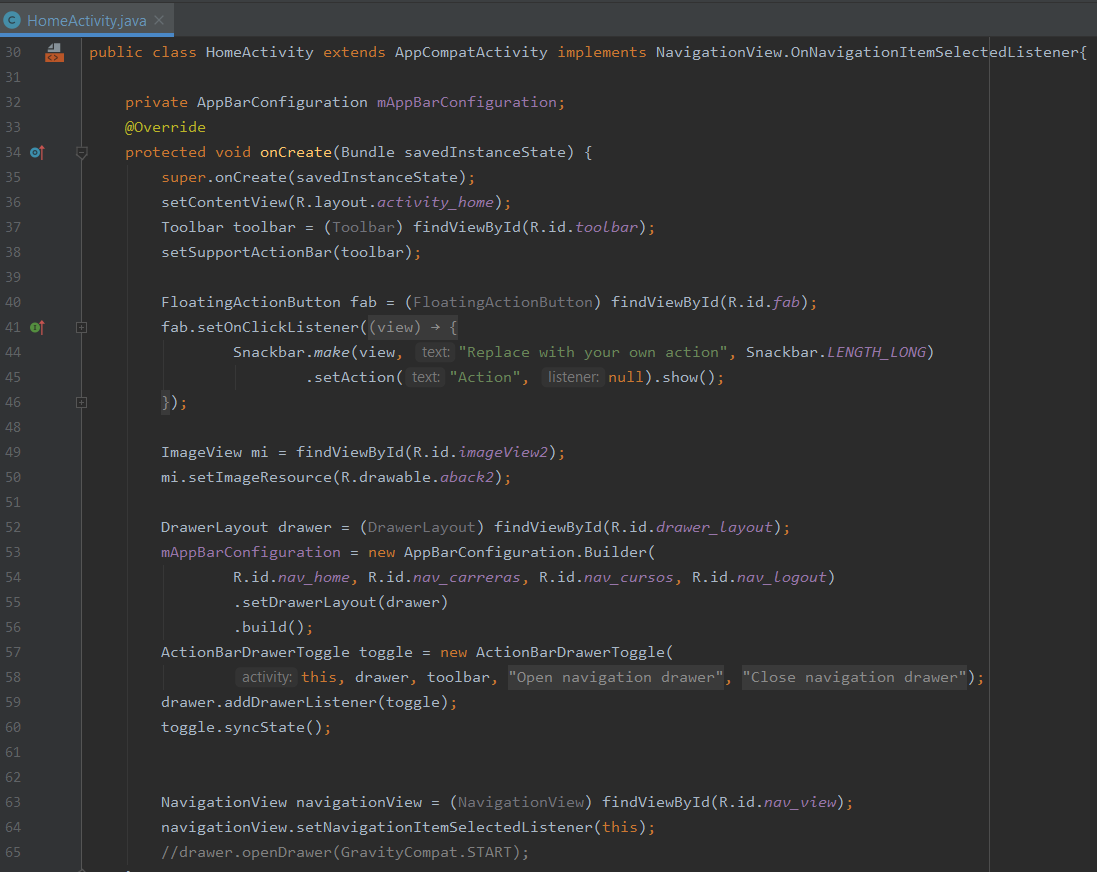
El layout estaría quedando como lo propone la plantilla además de que le asignaríamos una imagen de fondo y se estaría aplicando sus diferentes versiones, ya se landscape o tableta.





***HomeActivity.***

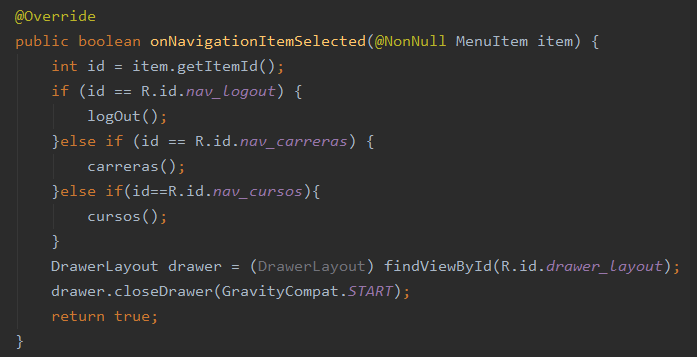
Esta actividad es el Navdrawer, es la aplicación principal donde, el su “drawer”, se colocarán los diferentes botones donde nos redirigirán a las diferentes actividades.



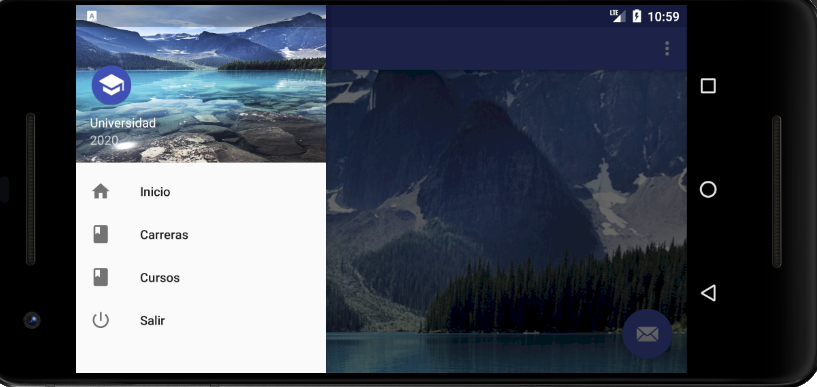
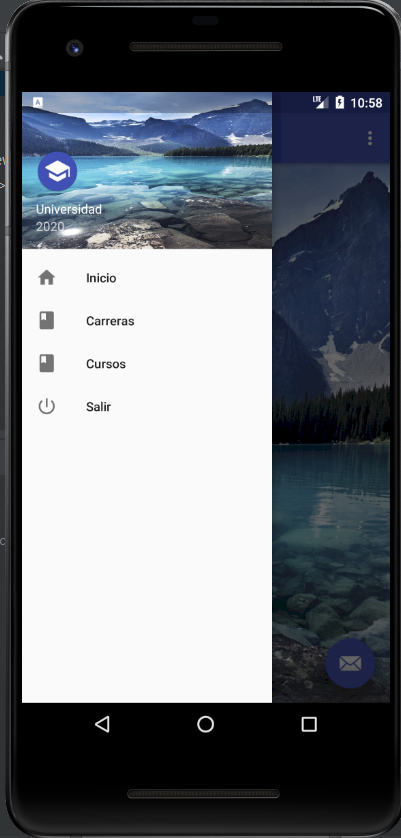
Viendo en la clase de actividad, además de la plantilla predeterminada, vemos que implementa el NavigationView.OnNavigationItemSelectedListener, este es el que nos va a permitir ejecutar el método de “onNavigationItemSelected(@NonNull MenuItem item)”, es el cual, si agregamos botones al drawer, este reconocería cual fue el que se seleccionó y nos dirigiríamos hacia allí, si es una actividad, con ayuda del listener.

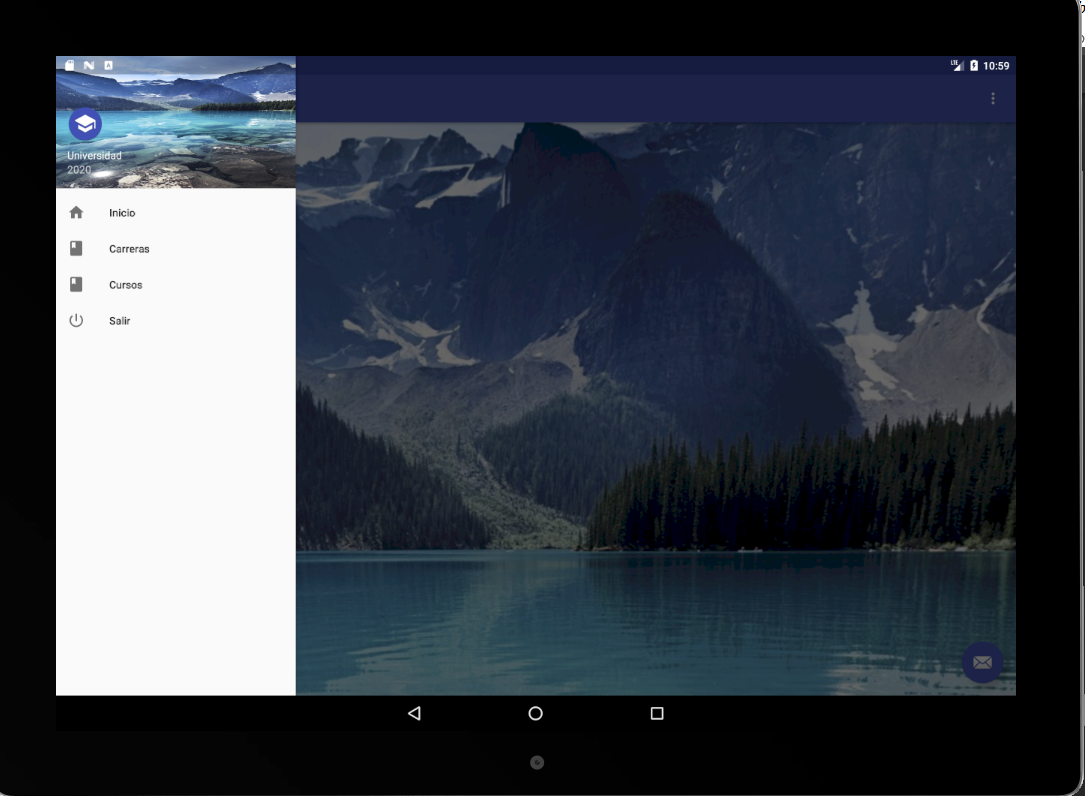
Además, tenemos que implementar en el oncreate el navigationview, y setearle el listener en esa actividad, además de una imagen como ImageView.

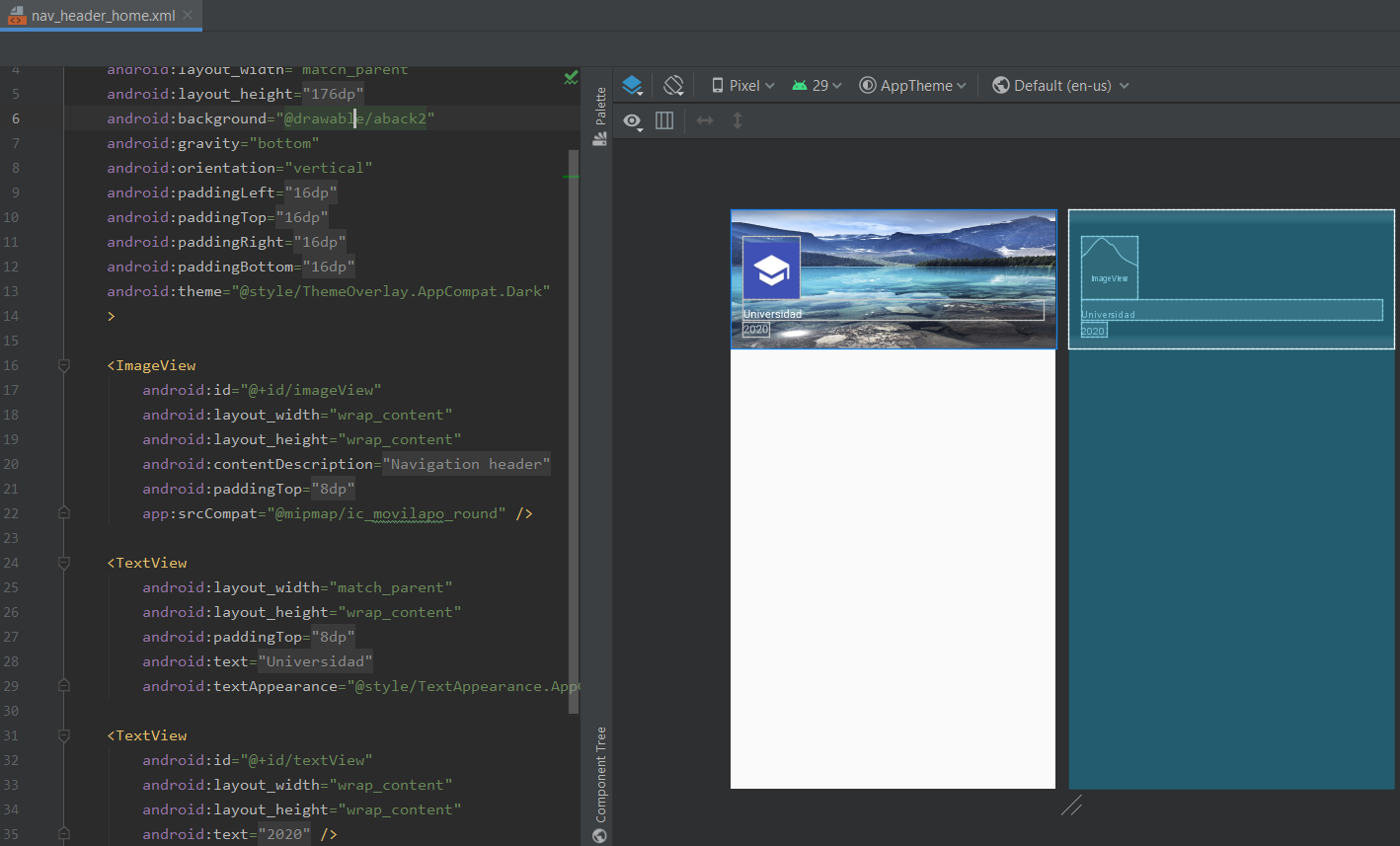
Revisando el método, onNavigationItemSelected, podemos ver es un método que nos pasa por parámetros un item del menú del drawer, con esto, podemos agarrar ese item, que es un int, compararlo con un id, que R ya lo tienen y se ha creado previamente como un layout, y si es alguno respectivo, se dirige a el método, que este, finalizara la actividad donde esta, y creara un nuevo Intent con la actividad que se eligió y se iniciaría dicha actividad.



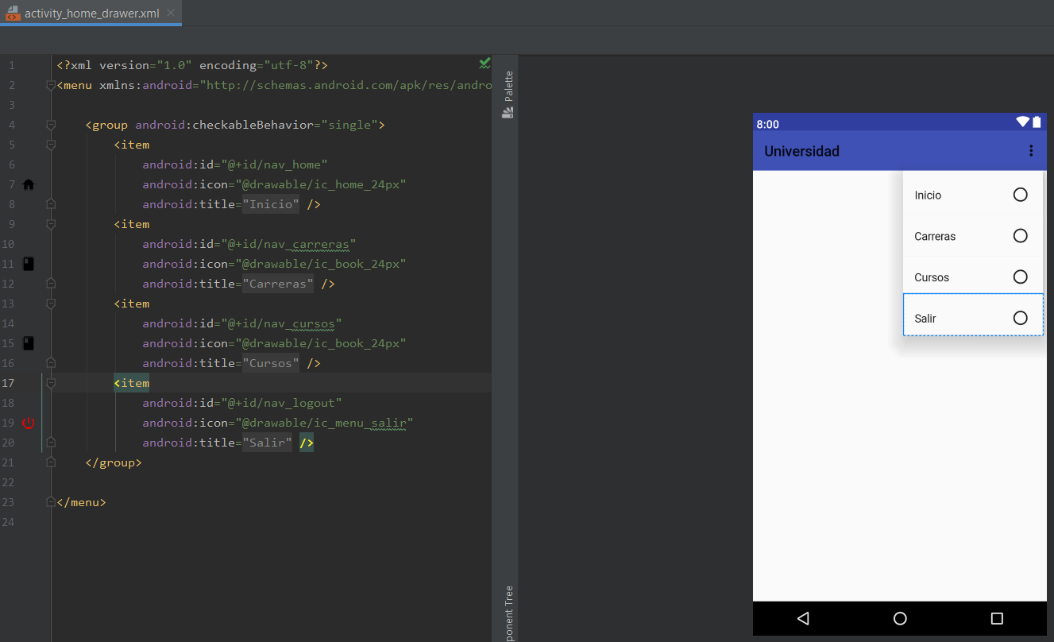
El layout, estaría quedando de la siguiente manera, además de tener sus respectivas variaciones en landscape y en una tableta.





El NavHeader del HomeActivity, también se modificó, en la parte de Strings, además se selecciono una imagen como de perfil y el background de este se utiliza el mismo que el del Home. 

Siguiendo en la parte del HomeActivity, en la parte del drawer se modificaron dos de los botones que traía por defecto(Camera y Slideshow), además se agrego uno más, el cual es el de LogOut, cada uno con sus respectivos id´s, estos son los que van a ser llamados en el OnNavigationItemSelected, en el HomeActivity, que nos redirige a sus respectivas actividades.



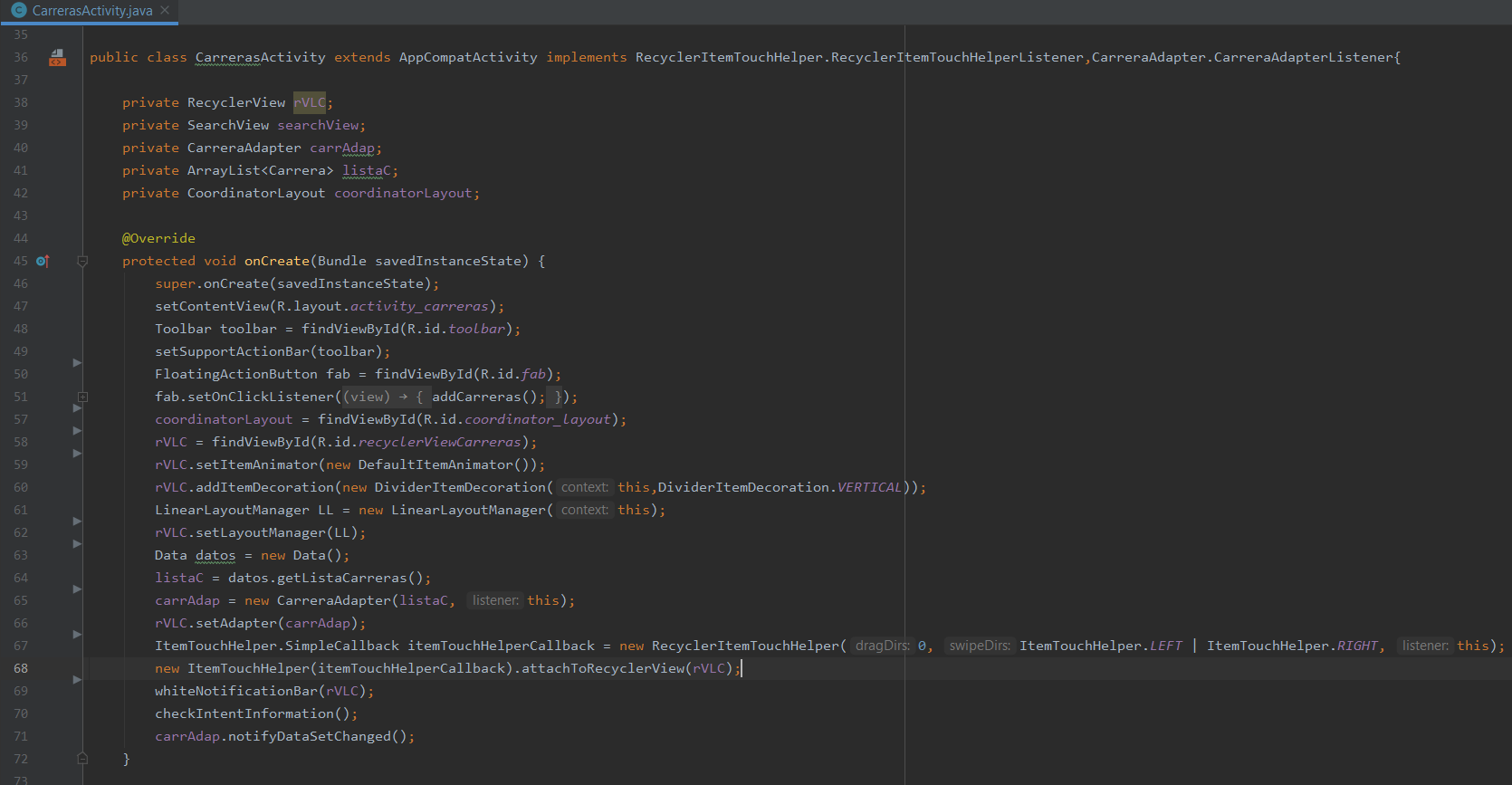
Luego de explicar la parte del Home y Login, quedan las partes de Carreras y Cursos, estas dos actividades, tienen en su cuerpo, una lista de los diferentes carreras o cursos, para ello, se utiliza un RecyclerView, que este se explicara dentro de poco.

***Carreras y Cursos Activities.***

Estas dos actividades, son las que vamos a utilizar, ya que aquí podremos hacer diferentes operaciones, ya sea, agregar una nueva carrera o curso, editarlo, eliminarlo y listas todos de donde las tengamos almacenadas, ya sea una base de datos, o unas listas temporales.

Cuando se crea la actividad básica, se crearán dos fragmentos de Java y dos fragmentos de layout, por el momento los eliminaremos, ya que no se trabajará con estos, además de la actividad Java principal y dos layouts importantes, el layout principal y el layout de contenido.

En la actividad principal Java, vamos a tener diferentes métodos, el más importante oncreate, aquí seteamos layouts, botones, inicializamos listas, etc.

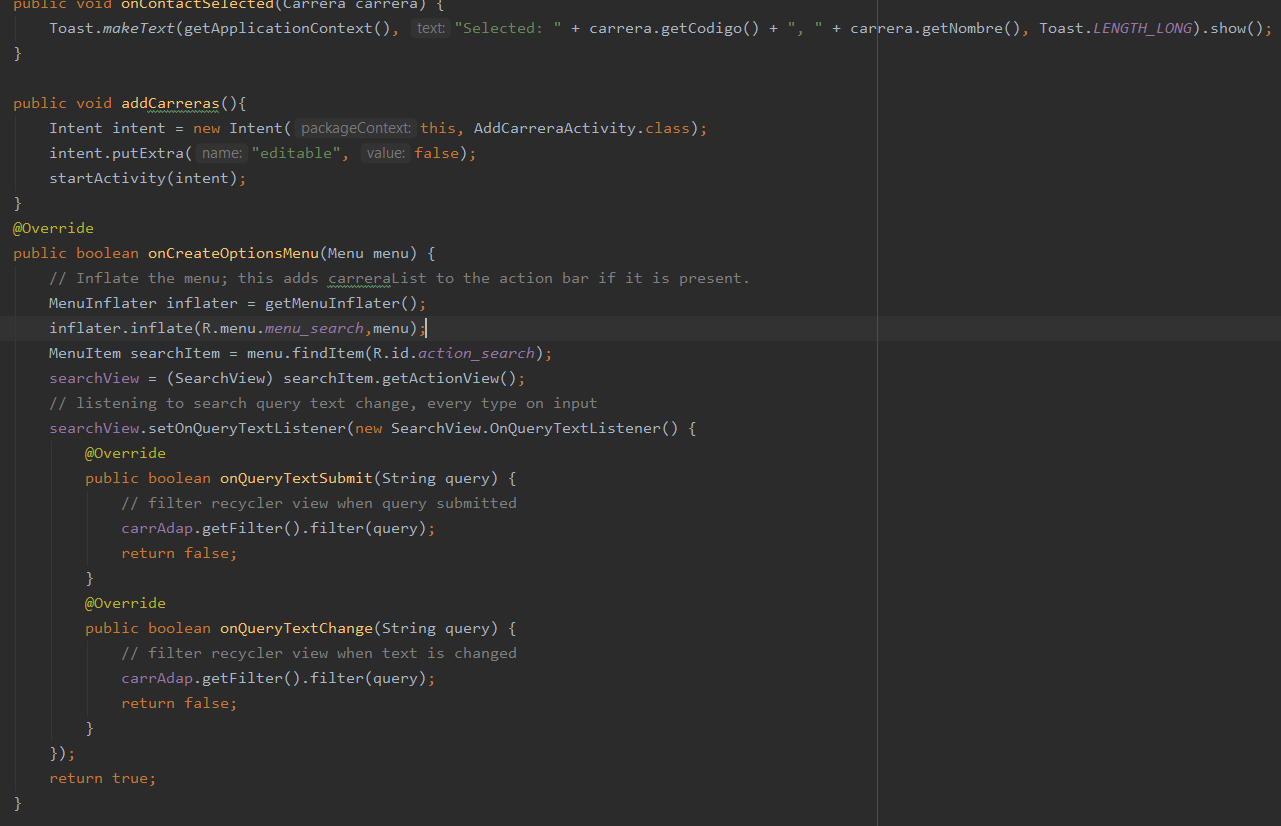


Como se puede observar, las cuatro primeras líneas la plantillan la carga por defecto, lo demás lo vamos agregando. Al botón que trae la plantilla, le vamos a asignar un Intent para dirigirnos a otra actividad para agregar nuevas carreras o cursos, le asignamos un coordinatorlayout, en el layout donde estemos, creamos una variable de tipo RecyclerView, y le seteamos el id donde la creamos (contenet\_carreras o cursos), agregamos una animación, además de una decoración de división, además de un nuevo manager para el layout, que utilizaremos para el Helper, donde podemos hacer Swipe de cada uno de los elementos del del RecyclerView.

Por el momento cargamos los datos de una lista en la clase de datos, y los asignamos a una lista que creamos como atributo en la actividad, seteamos el adaptador, para poder mostrar el RecyclerView, pasándole la lista con los datos, además del listener que implementamos, que en este caso es esta actividad.

Creamos un nuevo ItemTouchHelper, para hacer los swipes, pasando el mismo listener, además de dos id’s, un para izquierda y otro para la derecha.

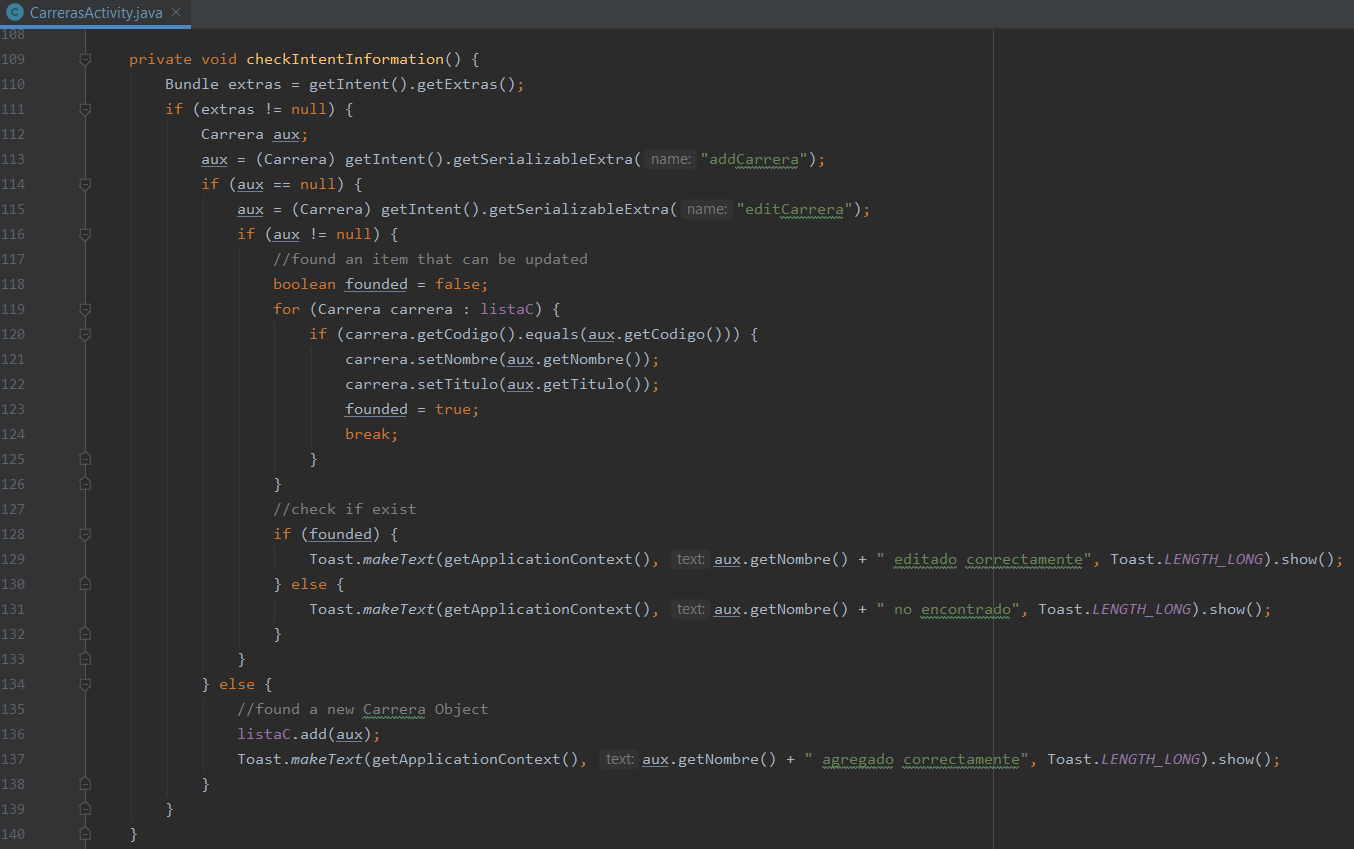
Se llama a la whitenotificacionbar, para poder ver la barra de notificaciones correctamente, después el checkIntenetInformation, en el cual vamos a verificar si se agrego una nueva carrera o curso o si mas bien se edito una que ya está en la lista.



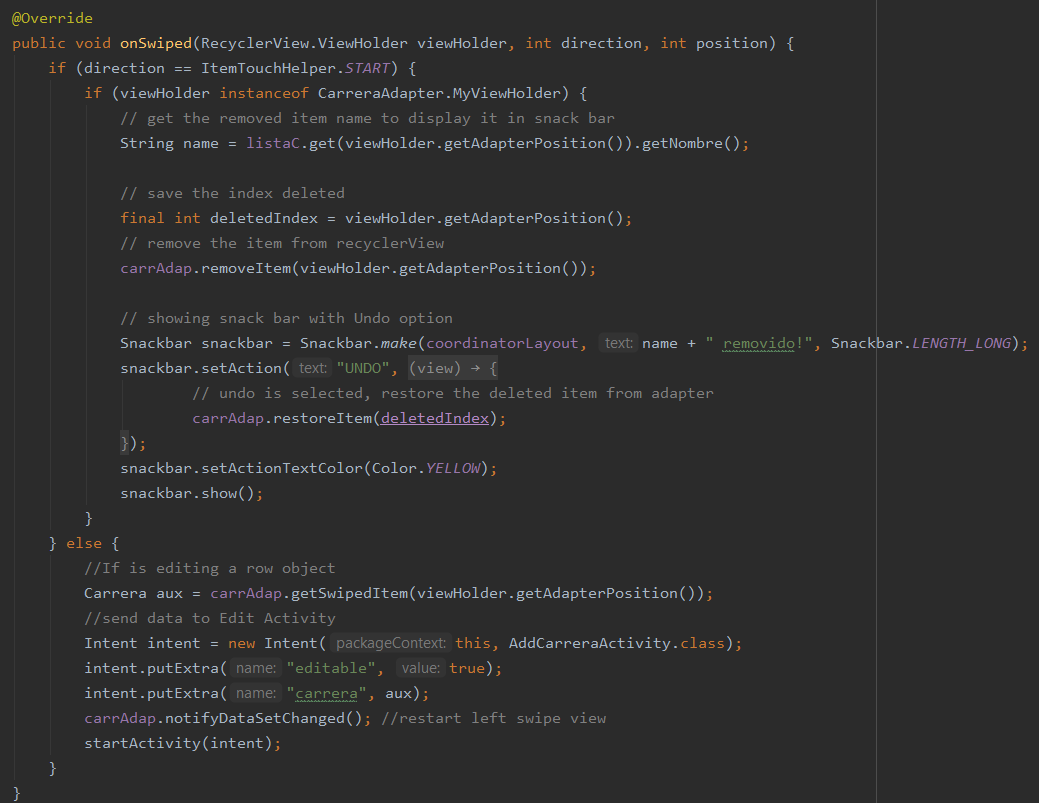
Siguiendo con los métodos, onContactSelected, resulta un Toast, el el cual podemos ver cual objeto se selecciono de la lista.

El addCarreras, es el método que va a utilizar el botón fab.

onCreateOptionMenu, en el cual vamos a poder realizar los filtros cuando busquemos uno en particular, esto en el icono de buscar en la toolbar de cada actividad.



El checkIntenetInformation, este método podemos verificar si fue agregado un nuevo elemento o fue editado, esto con ayuda del extra, que también en la actividad de crear Curso o Carrera, cuando lo mandamos, elegimos en el extra, dependiendo si es editable o no, y este lo filtra, para ver si solo edito o tienen que agregar una nueva. En resumen, verifica, si fue editado, o si es nuevo lo agrega al RecyclerView.

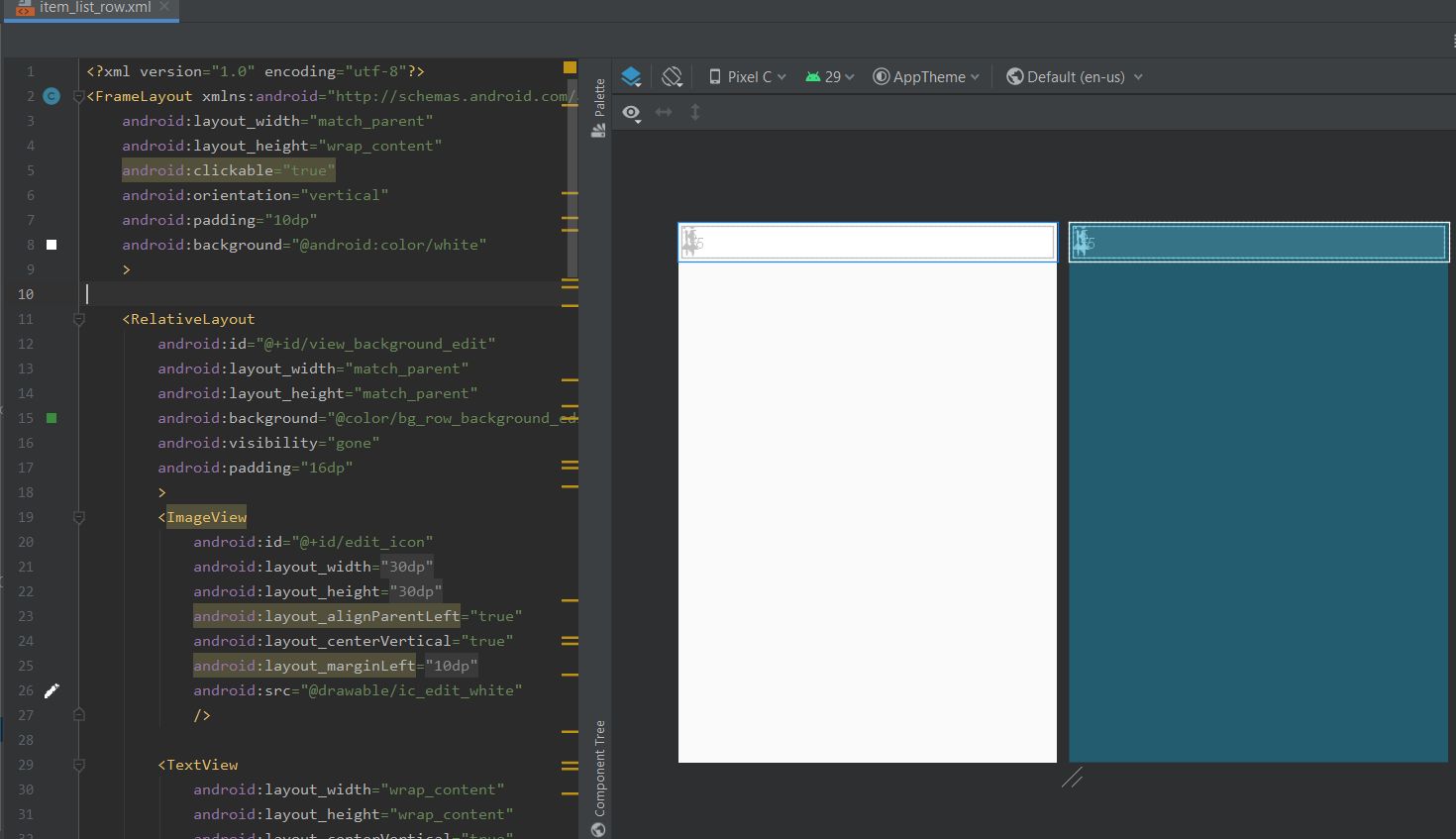


El ultimo método importante, el de onSwipe, en el cual vamos a poder realizar la operación de Swipe ya se izquierda o derecha, si es izquierda, este elimina al elemento del RecyclerView, y muestras un toast de que fue removido, sino, para le otro lado, este, con ayuda del adaptador, agarra el item, y lo agrega como un nuevo Curso o carrera, hacemos el intento de pasar a la actividad de crear un nuevo curso o carrera, además de poner un extra, que es decir, una información adicional que podemos trasmitir, como por ejemplo uno de tipo URI. Lo definimos como editable, le mandamos el item, le decimos al adaptador que hubo cambios, y empezamos la actividad.

***RecyclerView.***

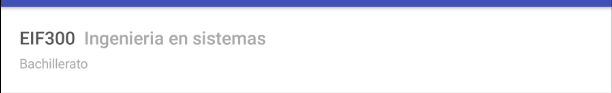
Para mostrar las listas de diferentes clases en una actividad, el RecyclerView ayuda bastante para cumplir con esta operación, entonces vamos paso a paso.

***Primero*** debemos de crear un layout de diferentes textos que vamos a tener en cada una de las celdas de la lista, por ejemplo, en este caso una fila de la lista en el RecyclerView dentro de un layout se implementara un texto principal, otro texto debajo de este mas pequeño, y otro mas a la derecha de este pequeño, puede darles color a las celdas, etc.



Corriéndolo en el móvil estarían quedando de la siguiente manera.

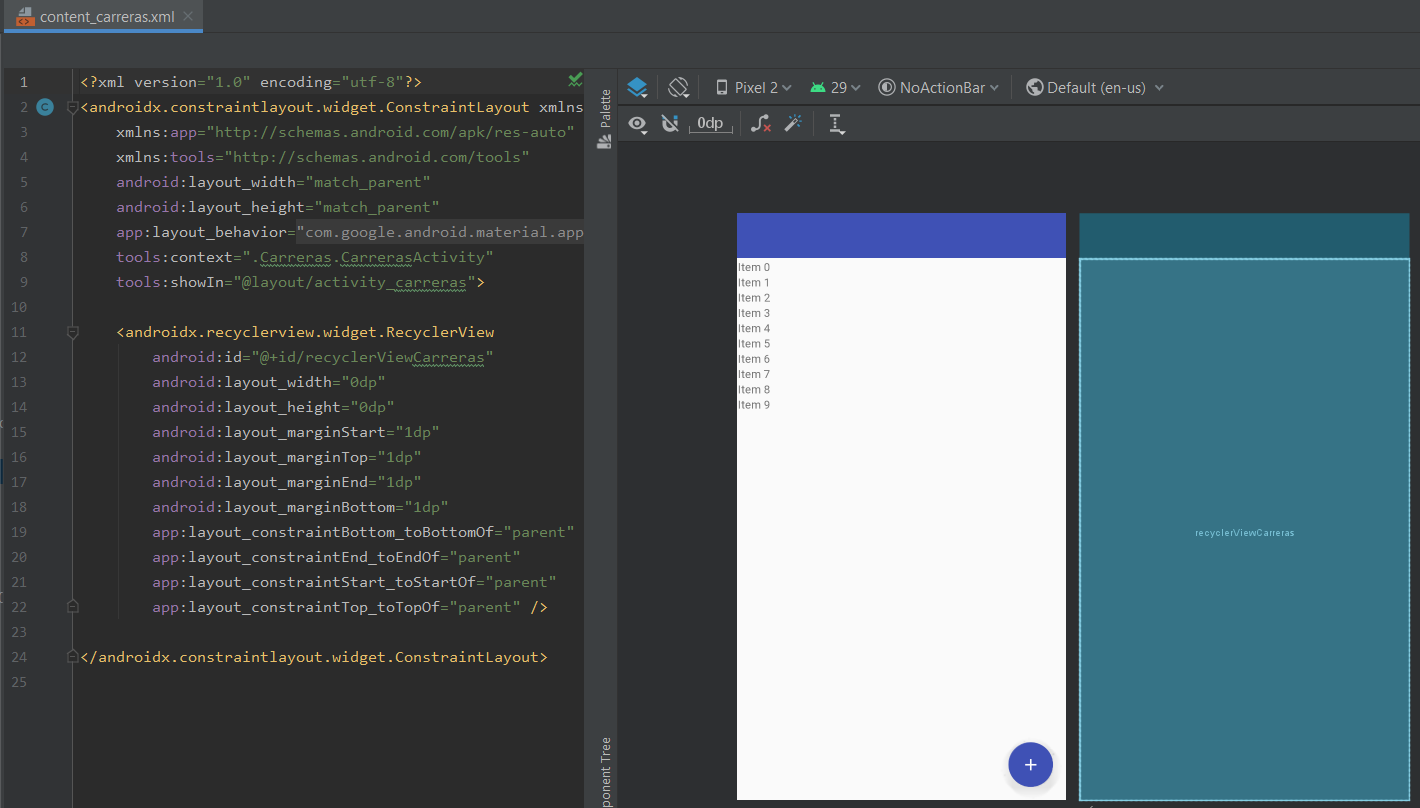
Carreras.



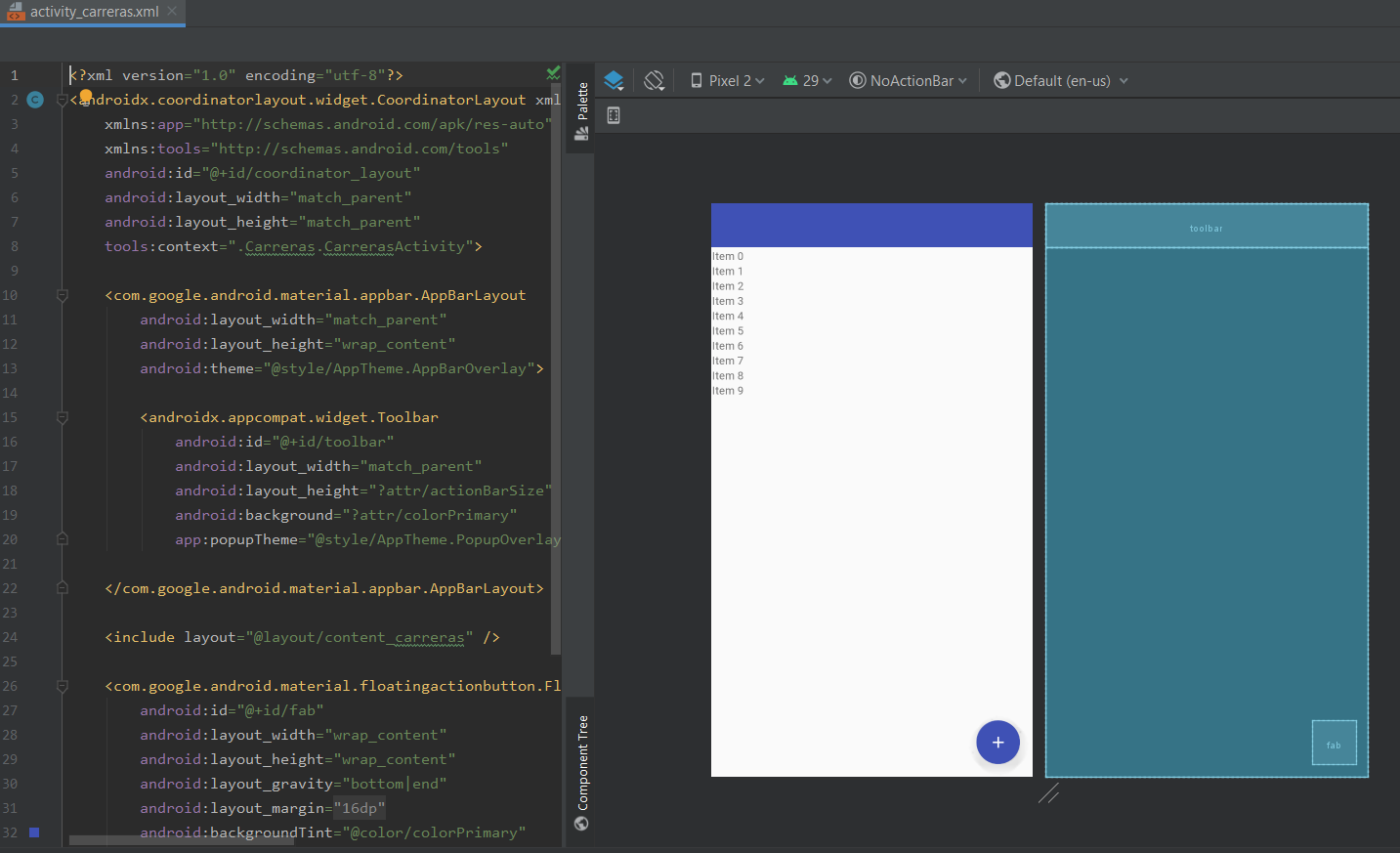
Cursos.



***Segundo*** es utilizar el layout “Content\_activity”, en el cual se le agregaría un elemento dentro de tipo RecyclerView, y darle la forma que necesitemos, además de asignarle un tipo de ShowIn, en donde la vamos a ver representada, en este caso, en el layout por defecto de Carreras y Cursos, así este nos trae la plantilla en este espacio, para que nosotros le agreguemos mas cosas utilizando este layout con el RV.



***Tercero*** en el layout por defecto de la Actividad de Carrera y Curso, tenemos que agregarle el nuevo layout que se ha creado, para que este ya quede linkeado a el layout y así poder visualizar la lista completa de las Carreras y los Cursos.



Teniendo los layout listos, podemos seguir al siguiente paso, que es crear un adaptador para las Carreras y Cursos, esto para que se acoplen y asignar variables, es como un tipo de filtro que pasan los datos.

***Adapter Carrera.***

Para empezar, hay que extenderlo de un RecyclerView. Adapter, ya que el Android no lo trae por defecto, con este adaptador, podemos obtener la data, de donde la tenemos almacenada, pasársela al Layout como lo necesitemos, además de in Filterable, para poder filtrar que necesitamos.

En la clase, se va a crear otra clase, que es el ViewHolder, que igual se extiende de RecyclerView.ViewHolder, aquí vamos a definir atributos de tipo TextView, además de otros atributos de tipo RelativeLayout, para la implementación del Swipe que se explicara más adelante; en cada uno de estos, vamos a asignar los id´s que se crearon la lista de fila del layout, para ir linkeando cuando obtenemos los datos.

En el constructor de la clase del adaptador, vamos a asignar una lista y un listener.

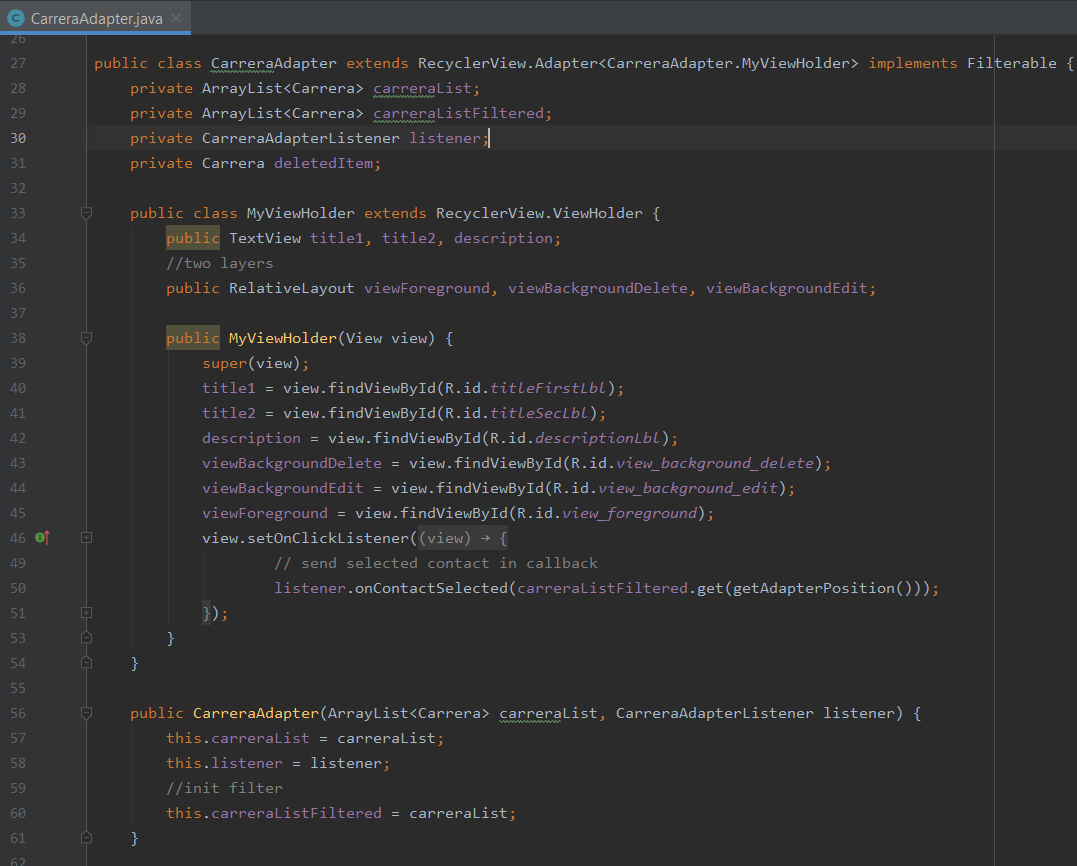
Siguiendo el código, en el onCreateViewHolder, es donde vamos a asignar el layout de las listas de ítems, en este caso el de item\_row\_list, que se ha creado, la celda de la lista, el layout donde le dimos la forma a la celda, con los diferentes textos.

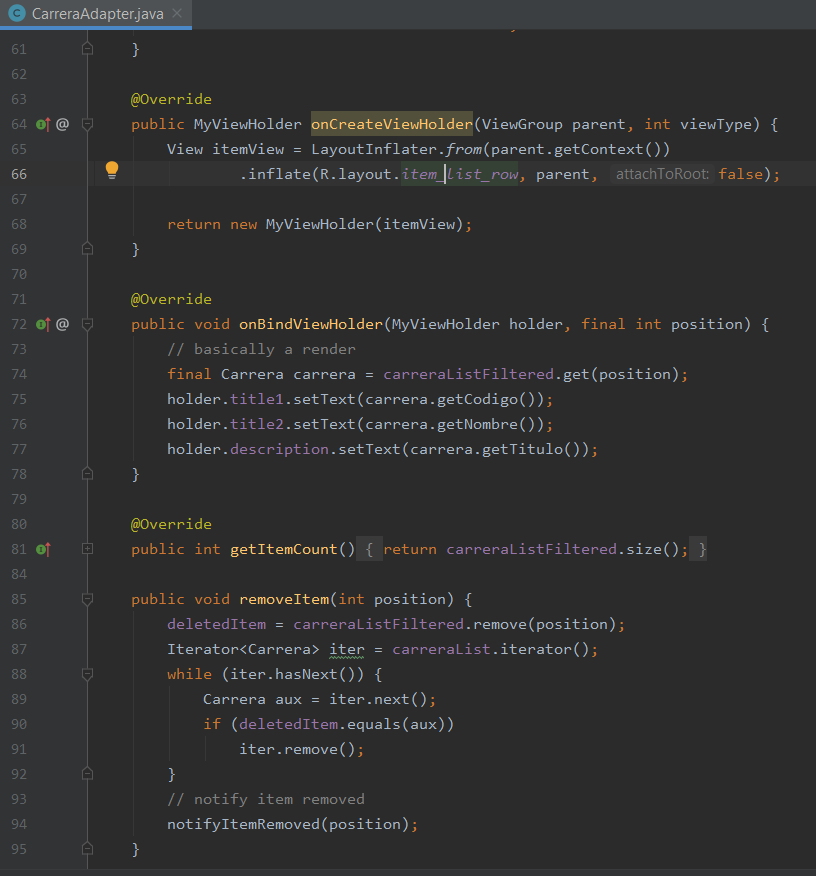
El OnBindViewHolder, en este podemos bindear los datos de cada lista, luego tenemos el getSwiped, en el cual se retorna el item en el cual el usuario esta en la lista, luego el onItemMove, cuando se va moviendo la lista en su respectivo layout.

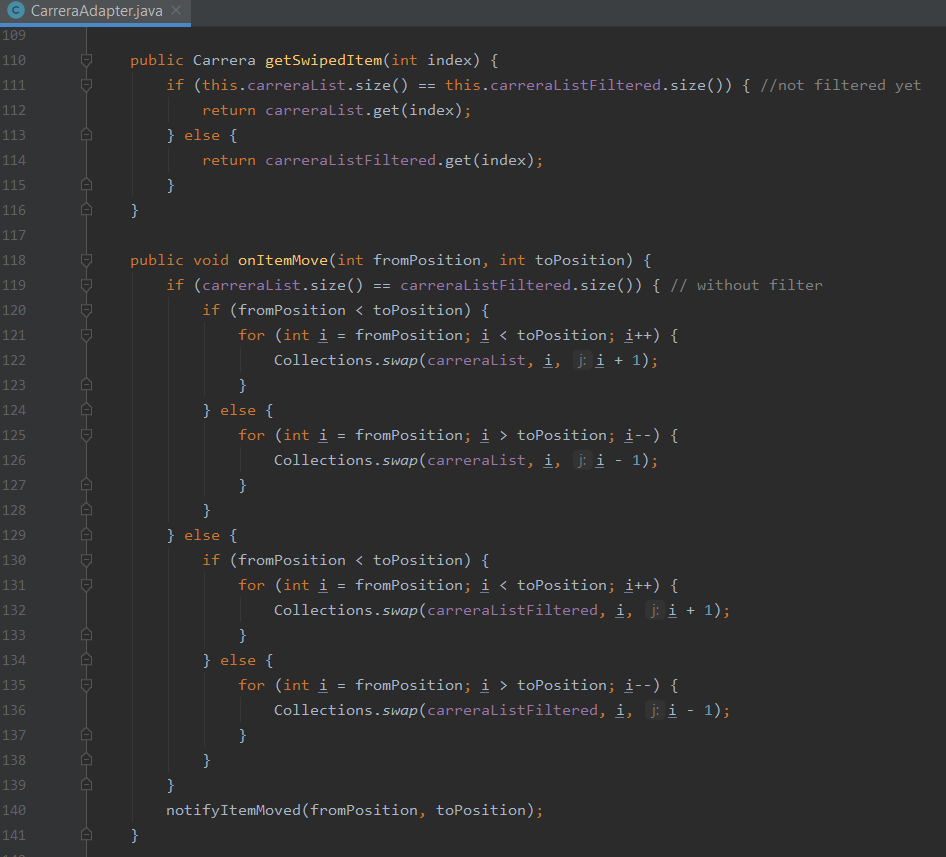
Los últimos, getFilter, es el método que se utiliza para poder filtrar los elementos de la lista, en el mismo RecyclerView, por ultimo se crea una interfaz que va a definir el listener.

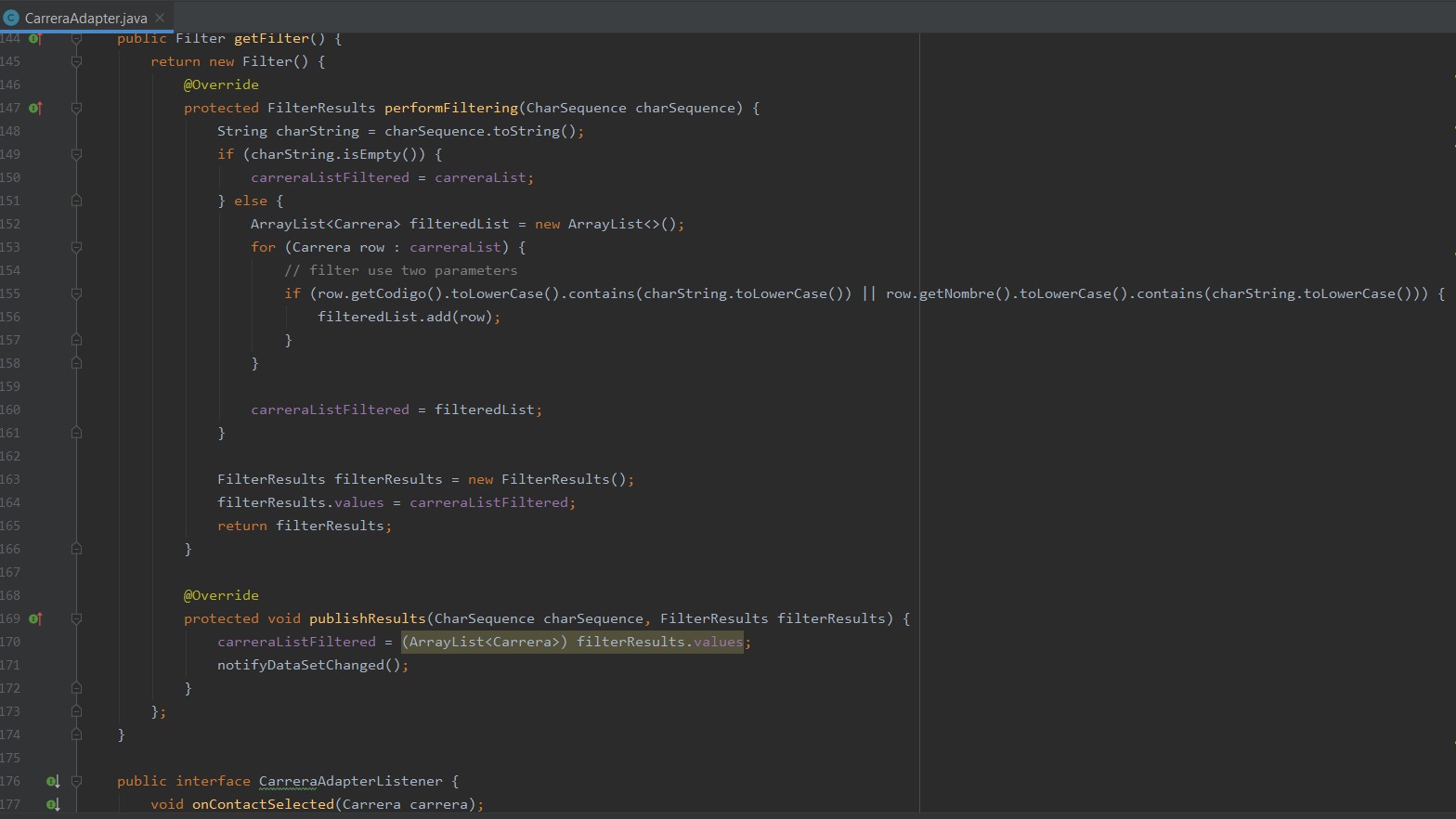
***Adapter Curso.***

Es el mismo que el adaptador de la carrera, a excepción de un adicional más, que es un espacio de texto que va a cuál carrera el curso este linkeado.





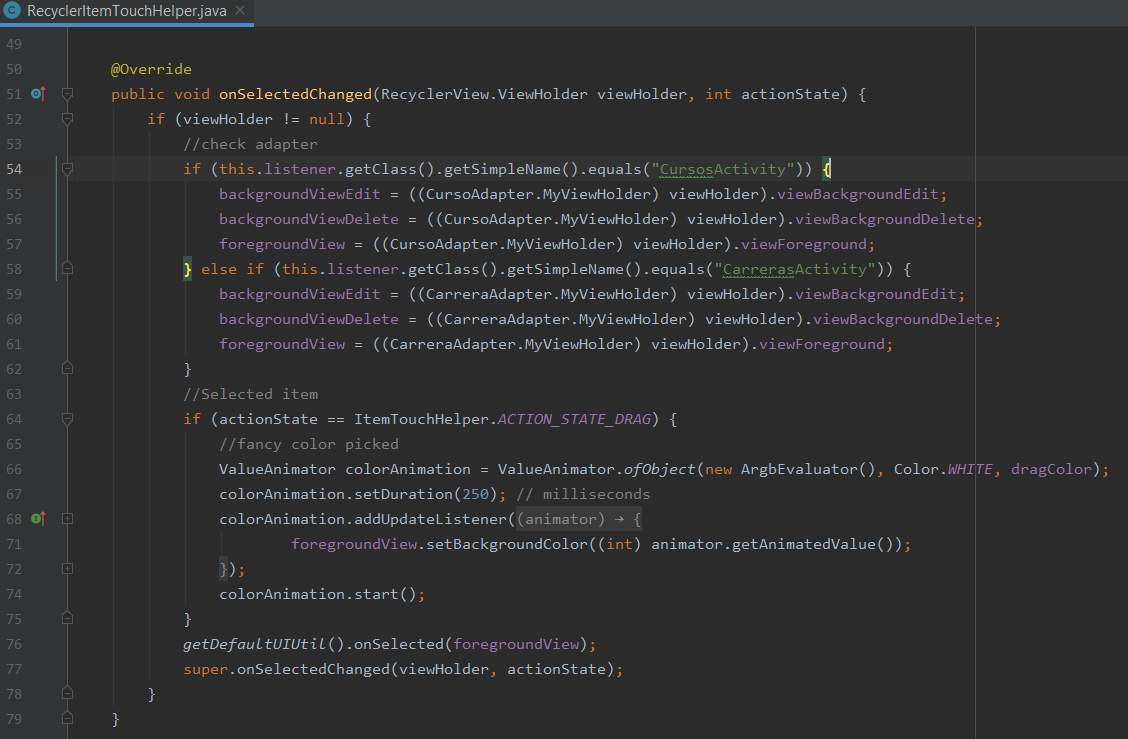




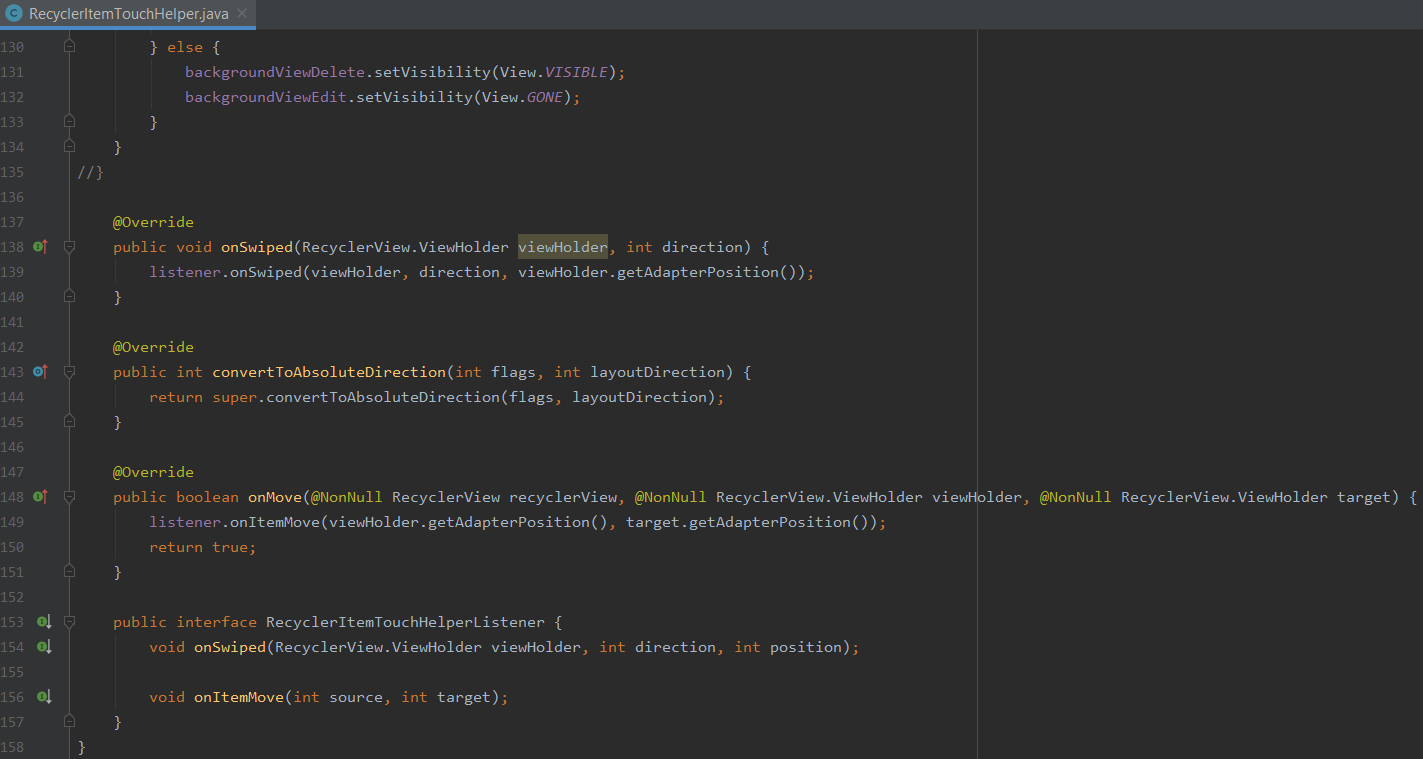
***ReclyclerItemTouchHelper.***

Con el layout de las Carreras y Cursos listos, además de los layout del RecyclerView para las Carreras y Cursos también, además de sus respectivos Adapter, con esto podemos ver la lista de cada uno de ellos, además de poder hacer los filtros en cada una, pero faltaría el Swipe para poder editar o eliminar, además de implementar las dos actividades de Carreras y Cursos. necesitaríamos una “ayuda” en el RecyclerView, y por ello, existe el Helper. Este nos ayudara a realizar las operaciones de hacer el Swipe en el RV, para eso, necesitamos un listener del mismo tipo de la clase, que se ira por parámetro en el constructor, además de sus direcciones de drags y swipes.

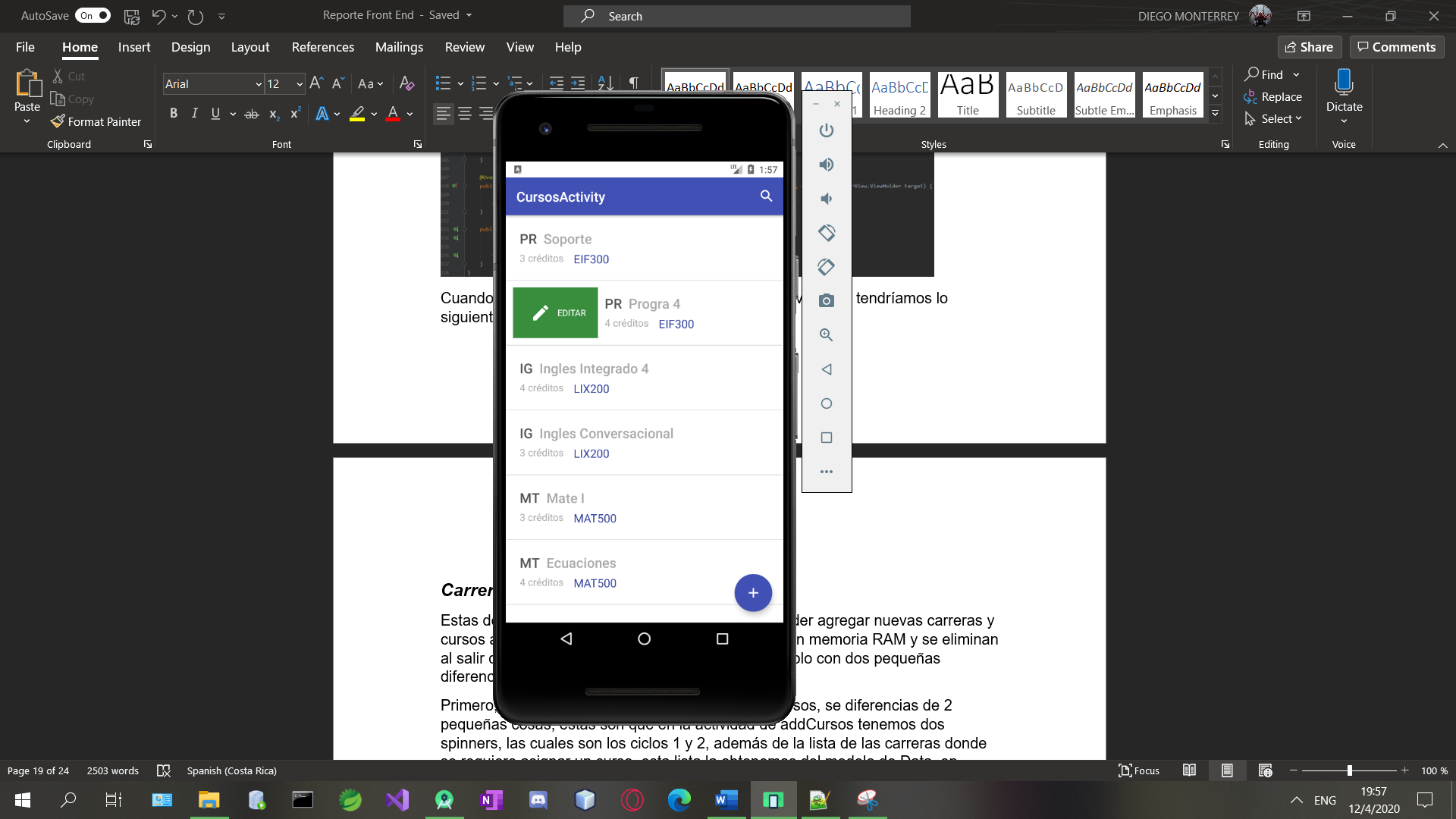
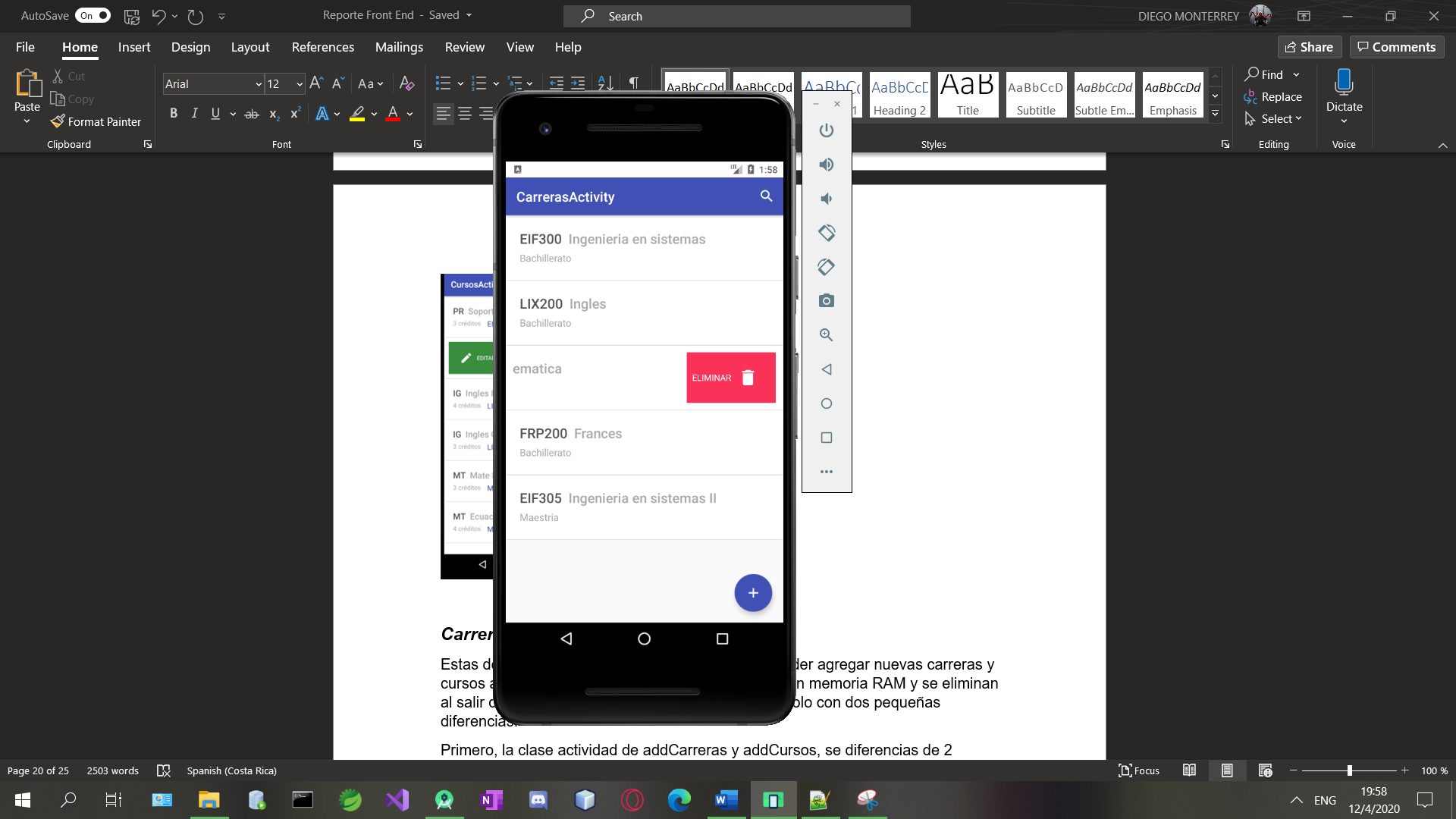
Uno de los métodos mas importantes, es el de onSelectecChanged, con este, le vamos a pasarle ViewHolder de tipo RV, en el cual vamos a tener listado los ítems, además de un estado de acción para cambiar algunos colores y su duración.



Además de ese, existen otro método que ayudan, onSwipe, es el que le pasamos por parámetro el ViewHolder, además de la dirección que en ese momento se escogió, el onMove, cuando se llama para mover de posiciones el item del RV, pero de momento no se va a utilizar, y la interface, donde seleccionamos el listener del Helper para el RV.



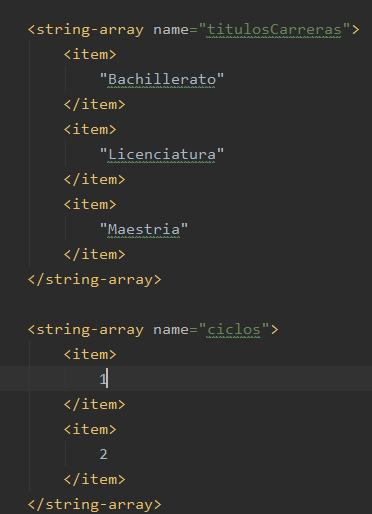
Cuando terminemos de implementar todas estas actividades, tendríamos lo siguiente.

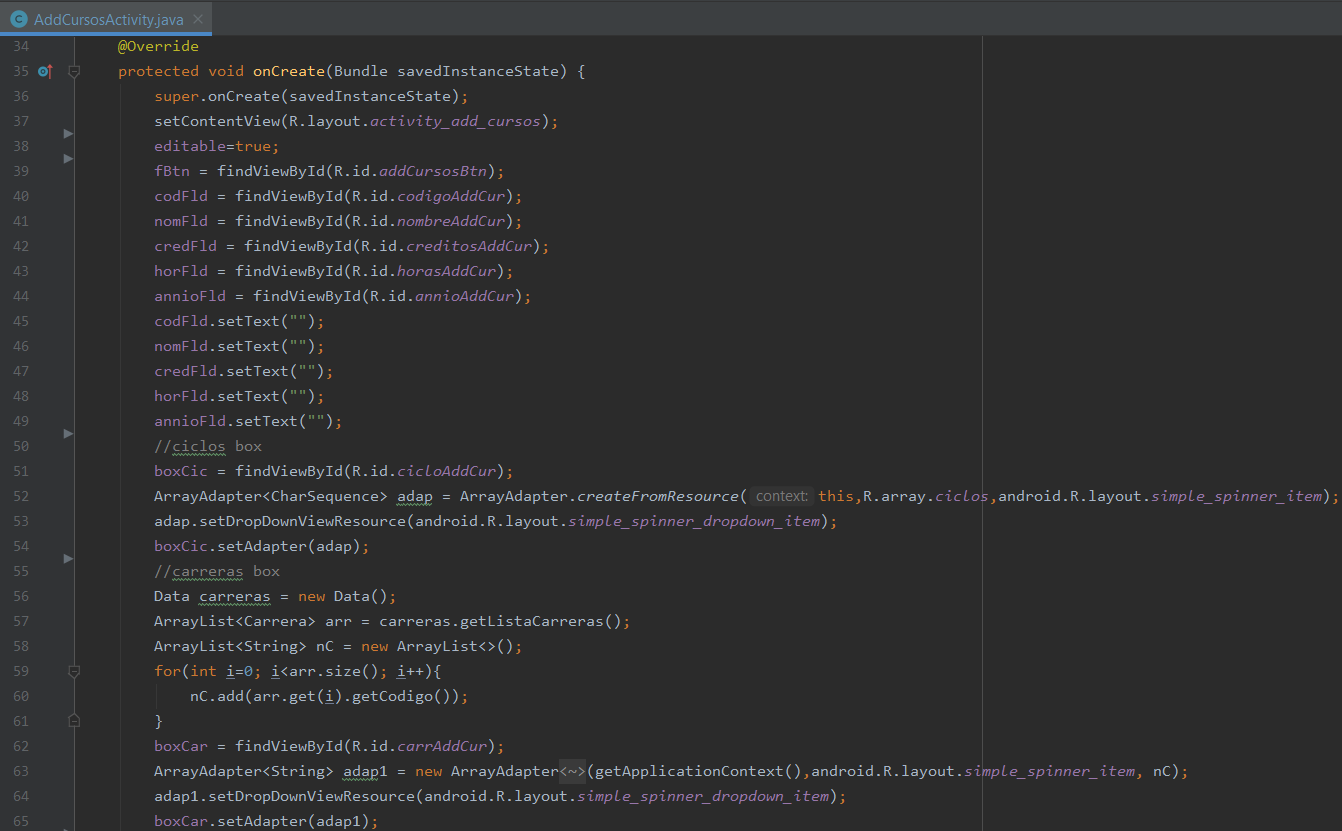
 

***Carreras y Cursos addActivities.***

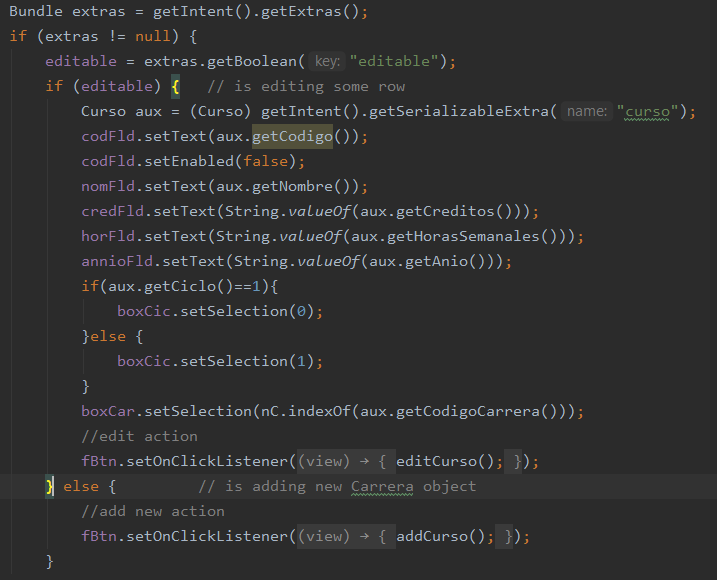
Estas dos actividades son en las cuales vamos a poder agregar nuevas carreras y cursos a la lista del RV (por el momento solo carga en memoria RAM y se eliminan al salir de actividades), estas dos son casi iguales, solo con dos pequeñas diferencias.

Primero, la clase actividad de addCarreras y addCursos, se diferencias de 2 pequeñas cosas, estas son que en la actividad de addCursos tenemos dos spinners, las cuales son los ciclos 1 y 2, además de la lista de las carreras donde se requiere asignar un curso, esta lista la obtenemos del modelo de Data, en addCarreras, tenemos la lista de títulos ya definidos, estas listas están en la parte de Strings, de aquí en donde las podemos modificar y acceder, por medio de R.



Para empezar, en el oncreate, empezamos haciendo el editable como verdadero, seteamos los espacios en blanco, para que cuando entremos no haya datos quemados, si no es editable, como decimos, tenemos spinners, y a estos spinners les vamos a pasar las diferentes listas, le creamos un adaptador de Arrays, y se lo seteamos al Spinner. 

Luego, en Cursos, en el segundo Spinner, vamos a obtener la lista de todas las carreras, y agregarlas al Spinner con solo el código, para el linkeo.

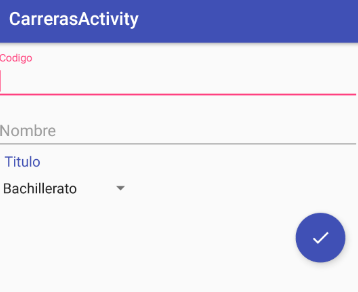


Luego, para terminar el oncreate, vamos a requerir el uso de los extras, ya que, si un elemento del RecyclerView se mandó para esta actividad, entonces viene con contenido en el extra, si no, simplemente es para agregar una nueva Carrera o Curso. Si viene como editable, entonces, seteamos los campos de esta actividad con la información que viene en el extra.

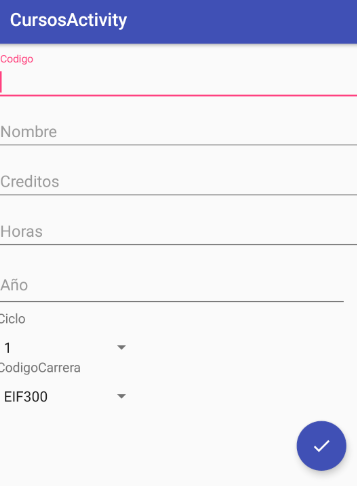


Los ultimos metodos importantes, son lo de editar Curso y agregr Curso, en el de editar, volvemos a crear un Curso nuevo con los espacios editados, y le asignamos al extra un string de editCurso, para que sepa que fue editado, y asi no lo agregue como uno mas, y el de agregar Curso, lo mismo, pero con el extra de agregar, para que simple,ente, cuando llegue al check de la actividad Pirncipal, de Carreras o Cursos, este lo agreggue sin nada mas, el de validatefrom, es para que complete todos los campos del formulario, quedando dicha actividad de la siguiente manera.

Carreras.



Cursos.



Para finalizar, contemplar que se utilizo la librería de AndoidX, que es la mas reciente actualmente, el SearchView que se utilizo, con la librería actual, esta disponible para la API 7 en adelante, el minimo que se utilizara es Android 24.

Hay que saber que se debe implementar el RecyclerView en el Buil.Gradle.

