Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

GRADO EN INGENIERÍA EN DISEÑO INDUSTRIAL Y DESARROLLO DEL PRODUCTO

Desarrollo de interfaz software para sistemas de evaluación Mago de Oz en dispositivos Android

Autor:

Blanco Sancha, Laura

Tutor:

Escudero Mancebo, David
Departamento de Informática
Área de Ciencia de la Computación e inteligencia Artificial

Valladolid, septiembre de 2020

El punto de partida es el desarrollo de una aplicación que tiene la intención de ser utilizada para hacer servicios tipo Mago de Oz para un videojuego Infantil de ordenador. La principal función de esta aplicación es la de ser utilizada como control remoto para sustituir el teclado físico del PC. Se realiza la aplicación utilizando Android Studio, centrándonos en un primer momento en la programación en lenguaje xml para programar en menor medida posteriormente en lenguaje java. Se realiza además un manual del desarrollador para que cualquier persona con un mínimo de conocimientos en programación puedo diseñar sus propios teclados según sus necesidades e incluirlos en la aplicación.

Desarrollo App en Android Studio – Mago de Oz – Programación Java

Diseño visual con xml – Teclado control remoto en Windows

The starting point is the development of an application that is intended to be used to make Wizard of Oz services for a Children's computer video game. The main function of this application is to be used as a remote control to replace the physical keyboard of the PC. The application is made using Android Studio, focusing at first on programming in xml language using later java language to a lesser extent. A developer's manual is also produced so that anyone with a minimum of programming knowledge can design their own keyboards according to their needs and include them in the application.

App Development in Android Studio - Wizard of Oz - Java Programming

Visual layout with xml - Remote control keyboard in Windows

ÍNDICE

En primer lugar, me gustaría agradecer a la Universidad de Valladolid por darme la oportunidad de realizar este Trabajo de Fin de Grado el cual me ha ayudado a darme cuenta de hacia dónde podría orientar mi futuro laboral.

Quiero dar las gracias a mis padre y a mi familia por apoyarme siempre e impulsarme a hacer lo que me gusta, gracias a ellos estoy ahora mismo escribiendo estas palabras. Gracias a mi padre por haberme ayudado a desarrollar mi espíritu creativo desde pequeña y a mi madre por tener la paciencia suficiente y decir siempre las palabras que necesitaba oír. Gracias también a mis amigos, a los de la infancia por soportarme en mis momentos de estrés y a los de la universidad por compartir tantos momentos de estudio y trabajo y hacerlo más ameno.

Quiero agradecer también a mi profesor y tutor, David Escudero, por confiar en mí para la realización de este proyecto y ayudarme incluso habiendo sido este un año atípico para todos. Gracias por haberme introducido en el mundo del diseño y desarrollo de aplicaciones y páginas web en la asignatura Técnicas de Presentación Multimedia, que ha sido el impulso necesario para atreverme a realizar este proyecto fuera de mi zona de confort.

Gracias también a Mario Corrales por haberme ayudado con sus conocimientos en Java haciendo que la aplicación salga adelante incluso a distancia y por haberme resuelto las dudas que me han ido surgiendo.

Me gustaría mencionar también a mi profesora de informática en 1º de carrera, Margarita Gonzalo, por introducirme y despertarme curiosidad por el mundo de la programación y hacerme ver que es una rama en la que puedo destacar.

Gracias a todas las personas con las que me he cruzado en estos últimos cuatro años y me han ayudado a ser quien soy hoy.

1. INTRODUCCIÓN

1.1. Motivación

La aplicación nace con la intención de utilizarla para hacer servicios tipo Mago de Oz para un videojuego infantil de ordenador, el cual se utiliza para mejorar las habilidades del habla de niños con síndrome de Down. Su principal función es utilizar el móvil como control remoto para sustituir el teclado del ordenador o para hacer alguna de sus funciones.

Además de la intención inicial para la que esta aplicación fue desarrollada, se puede utilizar para realizar más funciones. Por ejemplo, uno de los teclados desarrollados se podría utilizar para manejar una presentación de Power Point, permitiéndonos avanzar y retroceder por las diapositivas.

Otra de las intenciones de esta aplicación es que cualquier persona que tenga unos mínimos conocimientos de Java y Android, pueda agregar a la aplicación un teclado con sus propias necesidades. Por ejemplo, se podría crear un teclado para poder manejar la aplicación Netflix para Windows, teniendo control de play/pausa, el audio o los subtítulos.

Al poder controlar el teclado del ordenador con esta aplicación, otro posible uso podría ser para, conectando el ordenador a la tele, jugar a cualquier videojuego de ordenador que se maneje utilizando atajos de teclado, utilizando así el móvil como que fuera un mando.

1.2. Objetivos

- Desarrollo de la interfaz de una aplicación, programando la interfaz en lenguaje .xml y trabajando con distintos Layouts, para hacer una aplicación intuitiva y válida para distintos ámbitos.

- Realización de un Manual del Desarrollador para que, como he explicado anteriormente, cualquier persona con unos mínimos conocimientos en programación pueda crear sus propios teclados.

- Consecución de la conexión entre el dispositivo Android y el PC Windows y lectura correcta por parte del PC de lar órdenes mandadas por el smartphone.

1.3. Visión de conjunto

- Capítulo 2: Android.

Estudio el sistema operativo Android y de su entorno de desarrollo Android Studio, incluyendo sus diferentes Layouts y elementos gráficos necesarios para el desarrollo de la interfaz.

- Capítulo 3: Estado de la cuestión.

Estudio de mercado de aplicaciones con similitudes a nuestro objetivo. Estudio de la aplicación de partida, siendo la base sobre la que vamos a trabajar.

- Capítulo 4: Aplicación.

Desarrollo de la aplicación con los elementos necesarios para alcanzar nuestros objetivos. Explicación de los componentes utilizado. Desarrollo de un manual del desarrollador para que el usuario pueda realizar sus propios teclados. Manual de instalación explicando cómo realizar la conexión entre ambos dispositivos.

2. ANDROID

2.1. El Sistema Operativo Android

Android nació en 2003 de la mano de Rich Miner, Nick Sears, Chris White y Andy Rubin. Google compró Android en el año 2005, anunciándose dos años después, en 2007, la primera versión del sistema operativo, Android 1.0.[24]

Android es un sistema operativo, es decir, es el software que se ejecuta en un dispositivo y que nos permite usarlo y darle órdenes para que haga lo que le pedimos. Android administra el software y el hardware de los dispositivos en los que está instalado.

Fue diseñado para dispositivos móviles con pantalla táctil, como los teléfonos o tablets, aunque también se puede encontrar en muchos otros dispositivos.

En el año 2019, Android tenía una cuota de mercado del 90,9% y en el mundo había unos 2.500 millones de dispositivos activos.[10]

2.1.1. Arquitectura de Android

La estructura de Android está formada por 4 capas; el núcleo de Linux, las librerías nativas y el runtime de Android, el entorno de aplicación y las aplicaciones. Una de las características más importantes de Android es que estas capas están basadas en software libre.[20][26]

En la figura 2.1. se puede distinguir estas 4 capas.

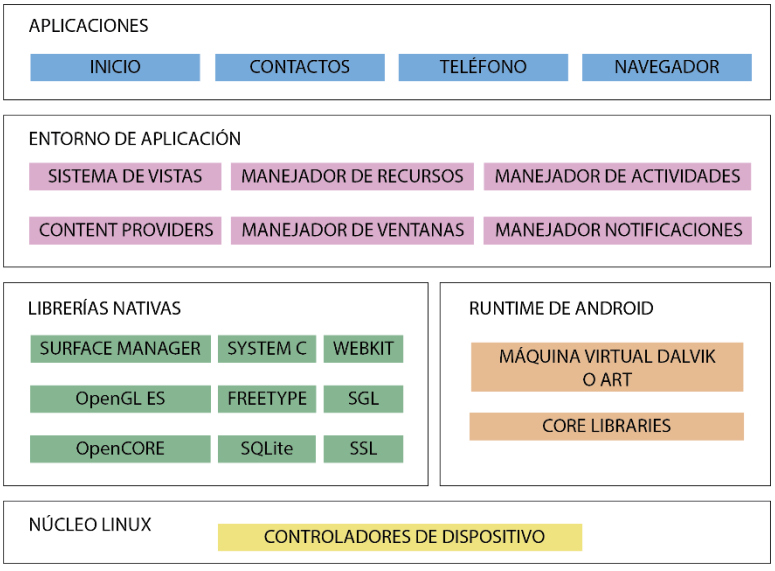


Figura 2.1. Gráfico capas de Android

- Núcleo Linux

El núcleo de Android está constituido por el sistema operativo Linux, concretamente por su versión 2.6.

Proporciona servicios como la seguridad o el manejo de la memoria, y además contiene los drivers, que son las instrucciones que permiten la interacción con el hardware del dispositivo.

La siguiente capa está compuesta por las librerías nativas y el runtime de Android.

- Librerías nativas

Esta parte contiene un conjunto de bibliotecas en C y C++, usadas en distintos componentes de Android. Cada librería tiene sus propias funciones.

- *Runtime* de Android

No se puede considerar exactamente una capa, ya que contiene librerías que se dividen en la de Java, con su funcionalidad habitual, y las de Android.

Aquí se encuentra la máquina virtual Dalvik, la cual es la utilizada en Android, siendo una máquina virtual un software que puede cargar en su interior otro sistema operativo.[47]

- Entorno de las aplicaciones

Contiene el conjunto de APIs[1] usadas por las aplicaciones base.

El desarrollador tiene acceso a la mayor parte de esta capa y es donde se encuentran la mayoría de las librerías que se usan en el día a día de un desarrollador Android.

- Aplicaciones

Esta capa contiene todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no.

Suele estar desarrolladas en Java, aunque es posible desarrollarlas en C/C++ y Kotlin.

2.1.2. Aplicaciones Android

Para escribir aplicaciones de Android, es posible, como he mencionado antes, utilizar los lenguajes Kotlin, Java y C/C++. Todos los contenidos de una aplicación Android están incluidos en un archivo de almacenamiento

[1]API – En inglés, *application programming interface.* Es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el software de las aplicaciones.

APK, que es el archivo que utilizan todos los dispositivos Android para instalar la aplicación.

Los componentes que forman una aplicación son una serie de elementos imprescindibles para su desarrollo. Podemos distinguir 4:[4]

- Actividades o *Activities*

Cada App de Android está formada por una serie de elementos que se van a visualizar en lo que coloquialmente conocemos como las pantallas de una aplicación. Estas pantallas son las actividades, cuya principal función es crear la interfaz de la aplicación utilizando un conjunto de ella.

- Servicio o *Service*

Es otro componente que se ejecuta en segundo plano y que no proporciona una interfaz de usuario. Por ejemplo, la aplicación Spotify reproduce música en segundo plano mientras el usuario se encuentra en otra aplicación.[11]

- Receptores de anuncios o *Broadcast Receive*

Este componente recibe anuncios (broadcast) y reacciona ante ellos. Puede ser por ejemplo una llamada entrante.

- Proveedores de Contenido o *Content Provider*

A través de este componente, las aplicaciones pueden consultar o modificar datos de otras. Por ejemplo, la aplicación Instagram puede utilizar imágenes guardadas en la galería de imágenes gracias a este componente.

2.1.3. Imagen Corporativa

El logotipo de Android se llama Andy, fue diseñado en 2007 por Irina Blok, una diseñadora gráfica de Google, y es un logotipo de código abierto, lo que significa que cualquier empresa del mundo puede personalizarlo.[25]

Color hexadecimal: #78C257

Pantone: 360 C

Tipografía: NORAD

Figura 2.2. Logotipo Android 2007

El pasado agosto del año 2019, con la nueva versión del sistema operativo Android 10, también se presentó un nuevo logotipo.[3][30]

En este nuevo logo, además de cambiar la forma se cambió el verde utilizado anteriormente, ya que se descubrió que era difícil de leer, y se eligió una paleta de colores que mejoraran el contraste visual.

Una vez seleccionado el tipo de activity que vamos a utilizar, se abrirá otra ventana donde elegiremos el nombre de la aplicación y la versión mínima de Android que va a soportar nuestra aplicación, apareciendo al lado de esta opción el porcentaje de dispositivos que la soportarían.

Color hexadecimal: #3DDC84

Pantone: 2412 C

Tipografía: AND Black

Figura 2.3. Logotipo Android actualidad

2.2. Android Studio

Android Studio fue presentado en el año 2013, es un entorno de desarrollo integrado para Android. Pertenece a Google y fue diseñado para ofrecer facilidades con nuevas herramientas para la creación de aplicaciones. Es una alternativa al programa Eclipse, con el que veremos notables diferencias al estar basado Android Studio en IntelliJ

En este caso, es el programa elegido para el desarrollo del programa, y cuenta con las siguientes características;

- Un entorno de desarrollo claro e intuitivo.
- Posibilidad de probar las aplicaciones desarrolladas en otros dispositivos.
- Elementos y plantillas de uso común en aplicaciones Android.
- Asistente para el código, con funciones como autocompletado del código.

2.3. Como crear un proyecto

Para crear un proyecto, lo primera que debemos hacer es abrir el programa Android Studio, se abrirá una ventana en la que seleccionaremos *"Start a new Android Studio Project".* Como se puede ver en el lado izquierdo

aparecerán los proyectos que has creado o con los que has trabajado recientemente.

En la siguiente ventana que se abrirá, tendremos que seleccionar para que dispositivo queremos hacer la aplicación, y una vez seleccionado esto deberemos seleccionar que tipo de activity queremos, pudiendo elegir entre ellas una activity vacía, con menú desplegable, de publicidad o que incluya Google maps.

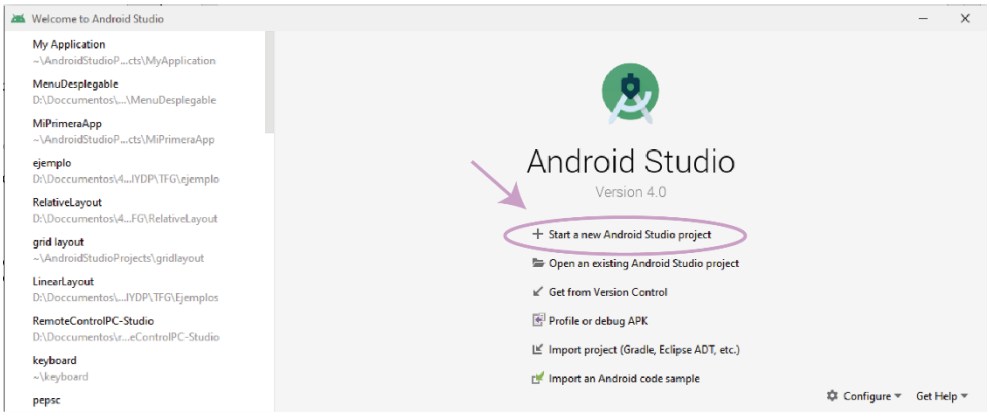


Figura 2.4. Como crear un proyecto 1.

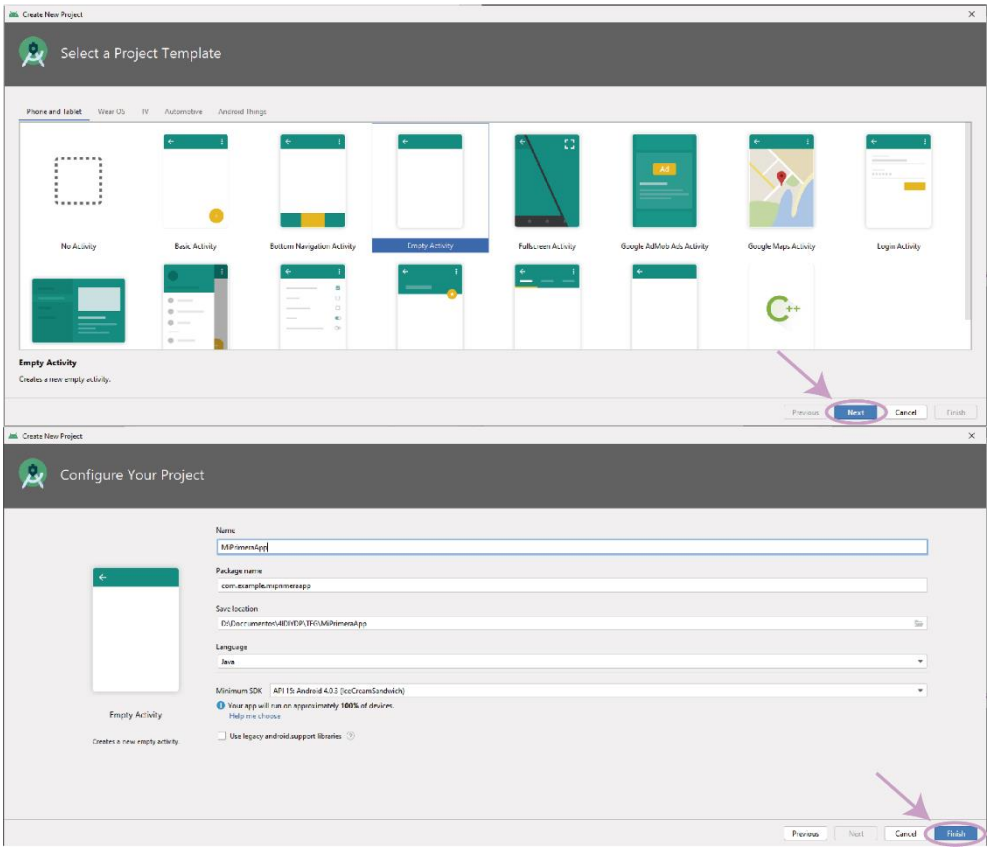


Figura 2.5. Como crear un proyecto 2.

Una vez abierto el proyecto podemos distinguir varias partes.

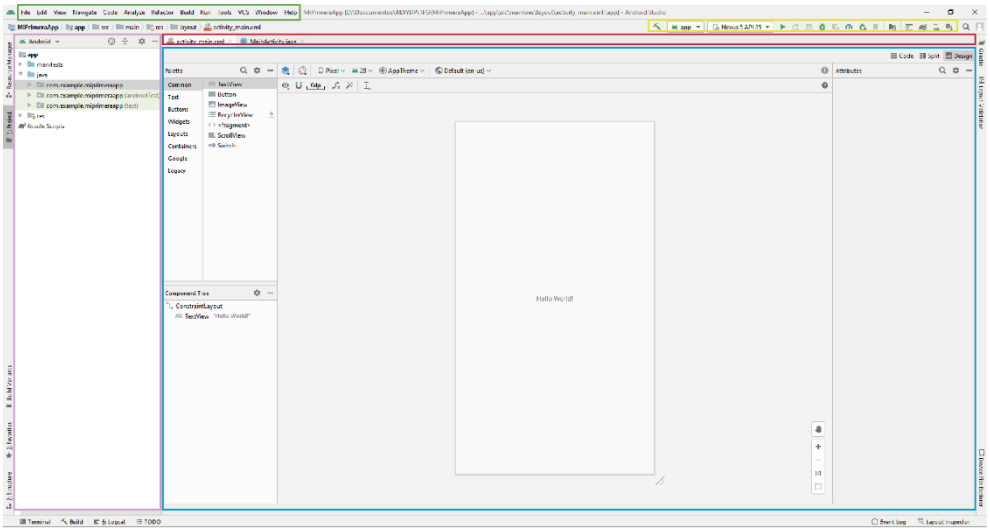


Figura 2.6. Pantalla Android Studio

A la izquierda, en la figura 2.6. , en el recuadro rosa, se encuentran los directorios. En el centro, en el recuadro azul, el espacio de trabajo, y encima en el recuadro rojo las ventanas abiertas dentro del proyecto. En el recuadro amarillo están las herramientas del emulador, y en el recuadro verde las ventanas con todas las funciones y herramientas de Android Studio

Dentro del directorio podemos encontrar varios subdirectorios.

- Directorio *Manifest*

Este directorio describe la información esencial de una aplicación, como es el nombre, o en el casi de haber varias *activities*, cual es la principal.

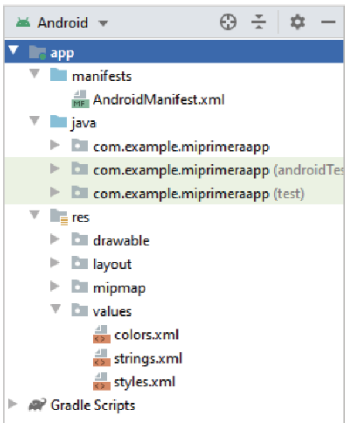


Figura 2.7. Directorios.

- Directorio *java*

En el directorio *java* encontramos todos los archivos .java relacionados con las activities.

- Directorio *res*

En este directorio encontramos todos los recursos que utiliza la aplicación. Dentro encontramos otros subdirectorios.

  - *Drawable,* donde se encuentran los archivos que son imágenes.
  - *Layout*, donde encontramos todos los archivos .xml que pertenecen a las interfaces de la aplicación.
  - *Mimap,* donde se guarda el icono de la aplicación.

- *Values,* donde encontramos diferentes valores como pueden ser colores, dimensiones, cadenas de texto o estilos.

En el espacio de trabajo podemos trabar con los archivos .java y .xml. Dentro de los archivos .java encontramos únicamente su código. En los archivos .xml podemos ver las interfaces que estamos diseñando, tanto gráficamente como en forma de código.

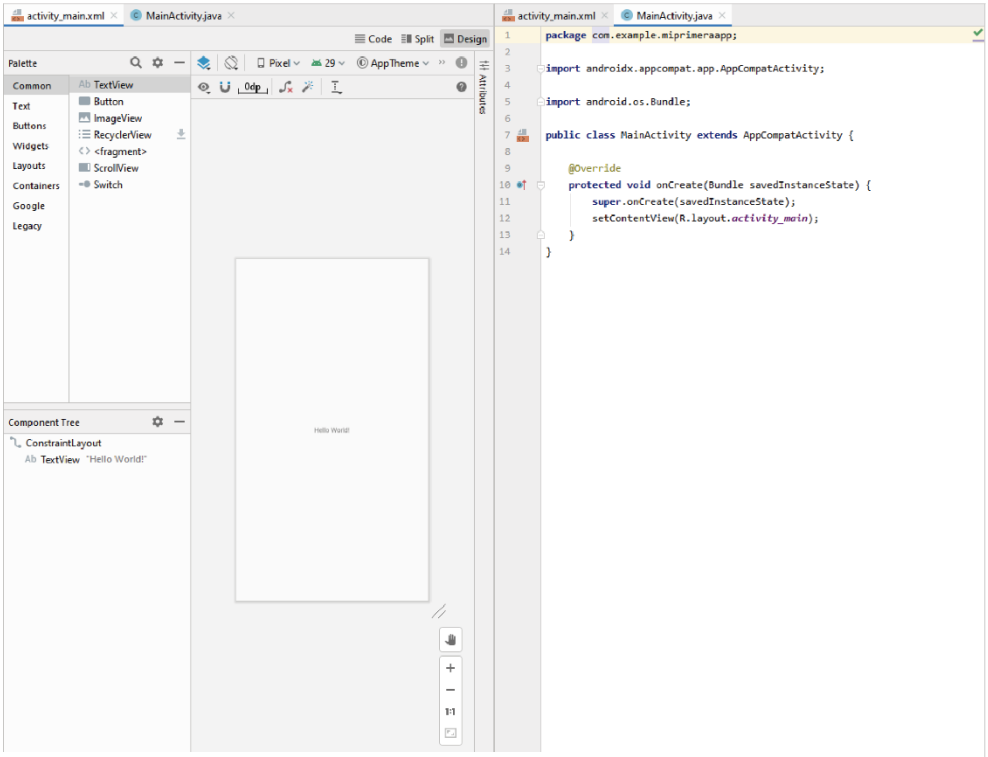


Figura 2.8. Espacio de trabajo.

2.4. Emulador

Esta herramienta es muy útil puesto que nos permite probar las app que hemos programado. Lo que hacemos con esta herramienta es reproducir con el software del programa, el hardware de un dispositivo físico.[29]

Es una herramienta muy sencilla de utilizar, lo primero que tenemos que hacer es abrir el AVD Manager desde la pestaña tools.

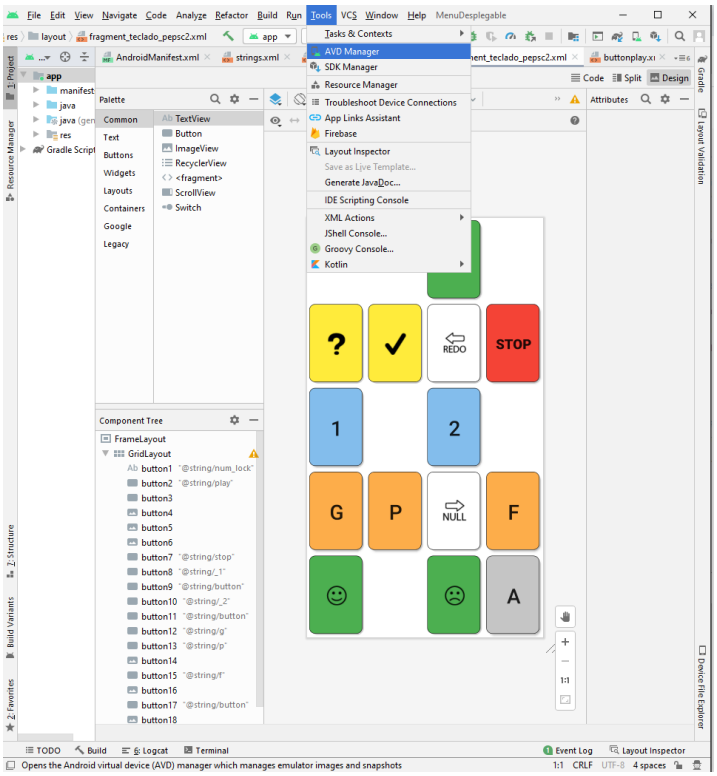


Figura 2.9. Instrucciones emulador 1

Una vez dentro de ese menú, se desplegará una ventana desde la que podremos crear un dispositivo nuevo. Una vez pulsamos en ese botón, se abrirá otra pestaña en la que podremos elegir el dispositivo en el que queremos probar nuestra aplicación, pudiendo elegir entre una televisión, un reloj, una Tablet o un móvil.

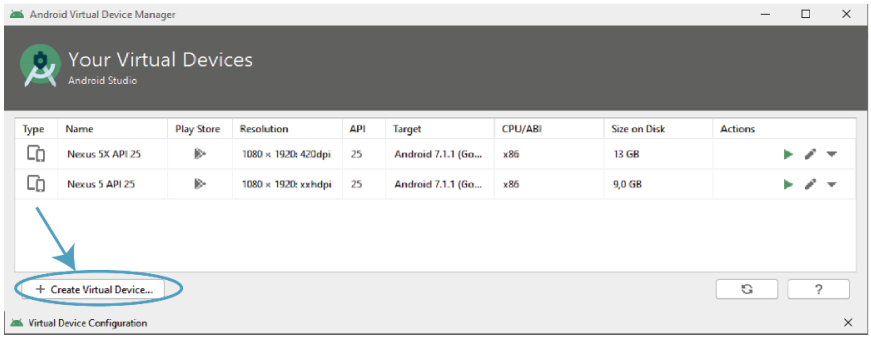


Figura 2.10. Instrucciones emulador 2.

Una vez elegido el teléfono, Android Studio nos va a solicitar que seleccionemos la versión de Android que va a tener nuestro dispositivo, la cual tendremos que descargar en el caso de que utilicemos el emulador por primera vez

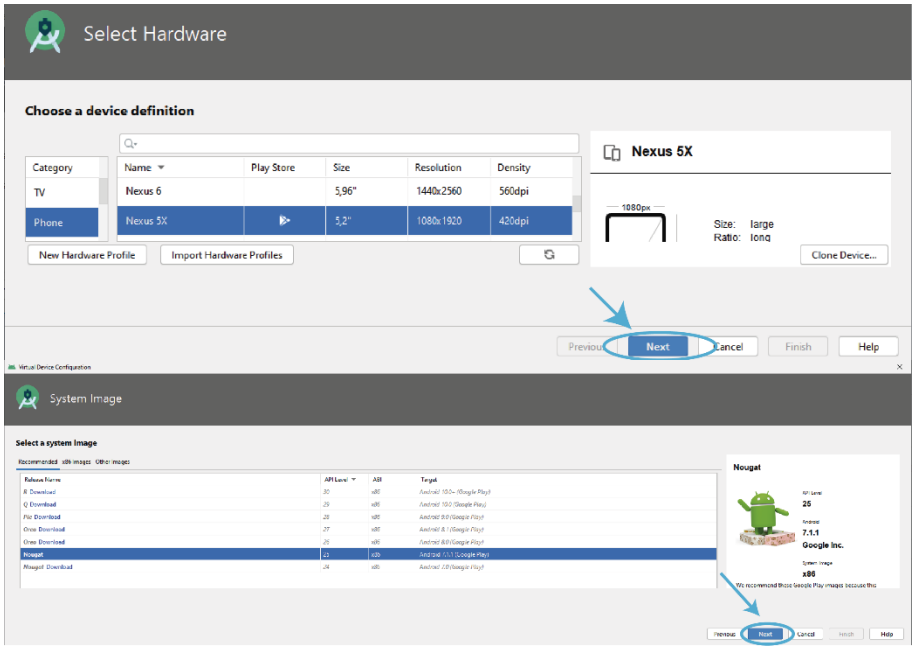


Figura 2.11. Instrucciones emulador 3.

Posteriormente elegiremos si queremos que el dispositivo esté en vertical u horizontal, y si queremos que solo se vea la pantalla o el móvil entero Una vez seleccionado todo esto, solo tendríamos que pulsar el triángulo verde que aparece en la parte superior y cargaríamos nuestro emulador.

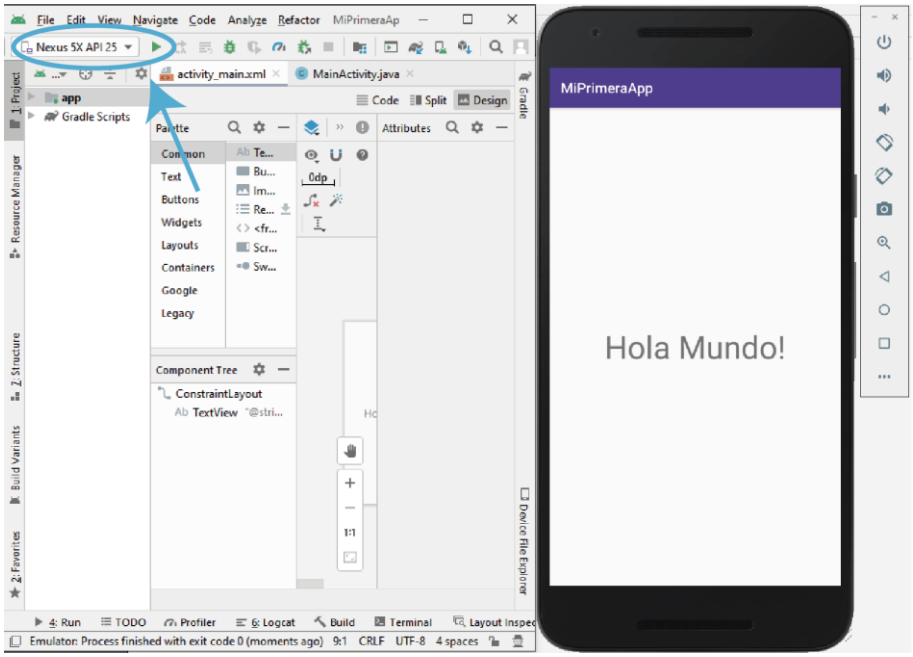


Figura 2.12. Instrucciones emulador 4.

2.5. Actividades

Como ya explicamos anteriormente, una actividad o *activity* es el lugar en el que se visualizan los elementos que permiten la interacción con el usuario, y que comúnmente es conocido como pantalla. Cada actividad está compuesta por su código y su interfaz para que funcione correctamente.[1]

2.5.1. Ciclo de una Actividad

Todas las actividades tiene un ciclo de vida compuesto por varios métodos, que son los siguientes.[2][7][23]

- *onCreate()*

Es el que se ejecuta cuando se crea una aplicación por primera vez. Aquí incluimos todo lo que tiene que ver con la interfaz, las variables, etc.

- *onStart()*

En este momento la aplicación se hace visible al usuario.

- *onResume()*

Aquí el usuario puede comenzar a interactuar con la aplicación al entrar en primer plano.

- *onPause()*

Se llama a este método cuando el usuario está abandonando la actividad, es decir, cuando deja de estar en primer plano.

- *onStop()*

La actividad deja de estar visible, ya sea porque la actividad ha finalizado o por otros motivos.

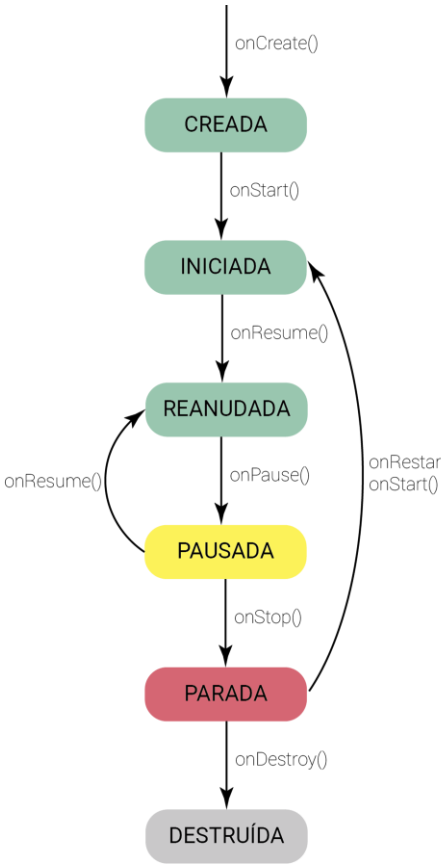


Figura 2.13. Ciclo de vida de un activity

- *onRestar()*

Se llama a este método cuando la actividad se ha detenido, antes de volver a iniciarla y de llamar al método onStart().

- *onDestroy()*

Se llama a este método antes de destruir la actividad, en este método limpiamos los recursos para mejorar el rendimiento.

2.6. Layouts

2.6.1. Tipos de Layouts

Un layout se utiliza para combinar en un mismo activity diferentes elementos, es decir, es un contenedor. Además nos permite asignar propiedades o características a estos elementos.

En Android Studio podemos encontrar varios Layouts, siendo algunos más utilizados que otros, y siendo algunos de los que se van a mencionar a continuación obsoletos. Entre estos encontramos:

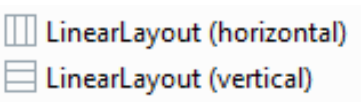
1. LinearLayout
2. RelativeLayout
3. AbsoluteLayout
4. TableLayout
5. GridLayout
6. FrameLayout

Los Layout más utilizados para las aplicaciones son el LinearLayout, el RelativeLayout y el FrameLayout.

2.6.2. LinearLayout

Es el Layout más utilizado ya que es el más sencillo y fácil de entender. La configuración dentro de este puede ser de forma vertical u horizontal, distribuyendo los elementos uno detrás de otro.

Si la forma de configuración elegida es la vertical, los elementos se irán colocando de arriba hacia abajo. Si en cambio la forma elegida es la horizontal, los elementos se colocarán de izquierda a derecha.

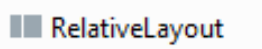


Podemos tomar como ejemplo de este tipo de configuración el de la pantalla de inicio de sesión de cualquier red social, en este caso Twitter, que utiliza un LinearLayout vertical (Figura 2.14..).

Figura 2.14. LinearLayout en Twitter

2.6.3. RelativeLayout

El RelativeLayout nos permite introducir cualquier elemento y añadir nuevos elementos en una posición relativa al anterior, es decir, cada elemento ocupa su posición utilizando como referencia a otro elemento.

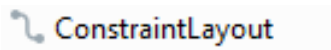


En este ejemplo de RelativeLayout (Figura 2.15.) podemos observar en el BluePrint las cotas, que señalan la posición que debe ocupar cada elemento, las cuales utilizan como referencia el ImageView.

2.6.4. ConstraintLayout

Es similar al RelativeLayout, aparece como una mejora en la versión 2.2 de Android Studio.

En este caso los elementos también se posicionan respecto a otros, o a los bordes del activity con cotas, tanto horizontales como verticales, que a diferencia del RelativeLayout pueden ser modificadas en los atributos del elemento, consiguiendo una mayor precisión.[5][31]



Desde la versión 2.2 de Android Studio en la que se introdujo es el Layout que aparece por defecto sustituyendo al RelativeLayout.
En este ejemplo (Figura 2.16.) podemos observar cómo aparecen las cotas de la misma manera que en el RelativeLayout.

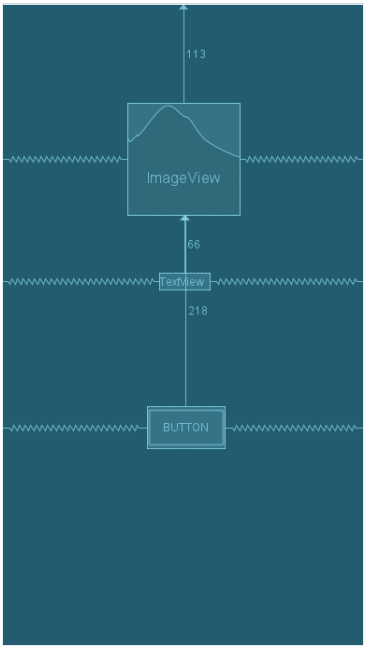


Figura 2.15. Ejemplo RelativeLayout

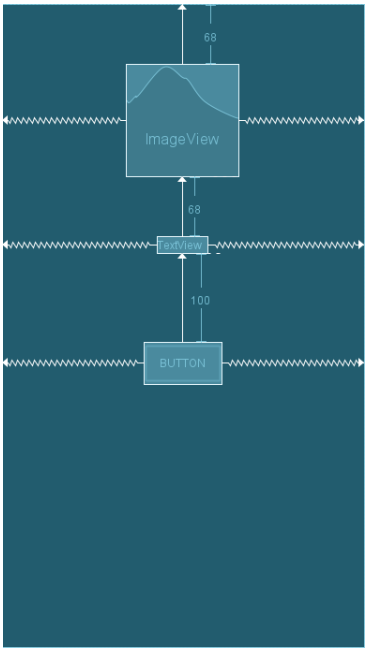


Figura 2.16. Ejemplo ConstraintLayout

2.6.5. AbsoluteLayout

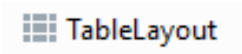
AbsoluteLayout permite establecer la posición de un elemento por sus coordenadas x e y. Esto implica que el elemento se mantendrá en esa posición independientemente de lo que suceda alrededor, por lo que no se a la pantalla si sus necesidades cambian.

Por este motivo este layout ya no está incluido en las actualizaciones de Android Studio.

2.6.6. TableLayout

En este caso los elementos de disponen en forma de tabla, colocándose en celdas, de manera similar a una hoja de cálculo, en filas y columnas.[13]

Los elementos no tienen por qué ocupar todas las celdas, y a su vez, también puede ocupar más de una celda cada uno.



En este sencillo ejemplo (Figura 2.17.), podemos observar una tabla formada por dos columnas y tres filas, en el que solo aparecen cuatro elementos, ya que dos de ellos ocupan más de una celda.

2.6.7. GridLayout

Es muy similar al TableLayout, permite distribuir los elementos en forma de tabla, pero aparece en Android Studio a partir de la versión 4.0.

Es Bastante más potente, pero tiene algunas incompatibilidades con versiones anteriores.

Un ejemplo de GridLayout que podemos encontrar en aplicaciones, es la forma en la que aparecen las fotos en el perfil de Instagram (Figura 2.18.).

2.6.8. FrameLayout

Este Layout posiciona los elementos utilizando todo el espacio, es decir, si se introducen varios elementos se colocarán unos encima de otros, lo cual es útil si por ejemplo para colocar fragments, que es una parte de la interfaz, como explicaré en el punto *2.9.2 Como agregar un Fragment a un activity*,

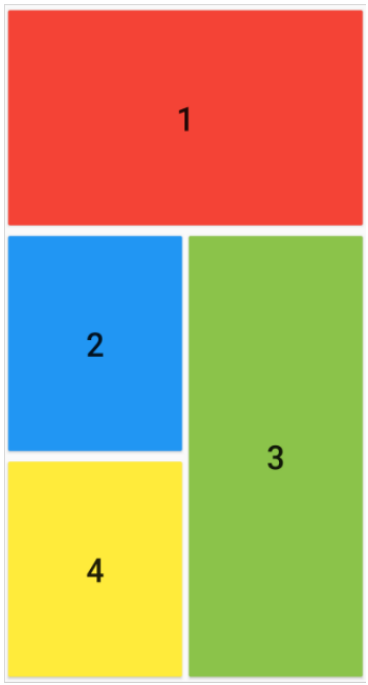


Figura 2.17. Ejemplo TableLayout

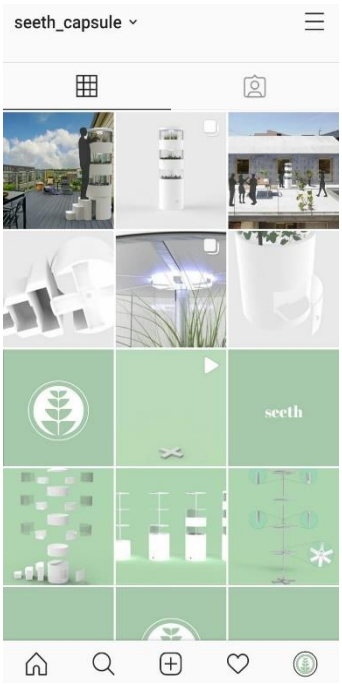


Figura 2.18.. Ejemplo GridLayout en App

2.7. Elementos gráficos

Los elementos gráficos son aquellos que sirven para crear la interfaz de nuestras aplicaciones.[15][19][27]

Todos estos elementos tiene dos propiedades en común, a anchura (*width*) y la altura (*height*). Estas dos propiedades pueden tener 3 valores preestablecidos además de sus valores numéricos. Estos 3 valores son los siguientes;

- Match_parent, lo que significa que el elemento va a tener la misma anchura o altura que el elemento que lo contiene, su "padre".
- Fill_parent, este valor hace que el elemento rellene al elemento que le contiene, se utiliza para lo mismo que match_parent, pero a partir de la versión API 8 se dejó de utilizar para dar paso a match_parent.
- Wrap_content, con el que el tamaño del elemento se adapta a lo que tiene en su interior.

2.7.1. Interfaz y pestañas

Todos estos elementos aparecen en las pestañas .xml en el espacio de trabajo que ya describimos anteriormente. Con la siguiente captura de pantalla, figura 2.19., pasamos a describir los elementos que aparecen en este espacio de trabajo.

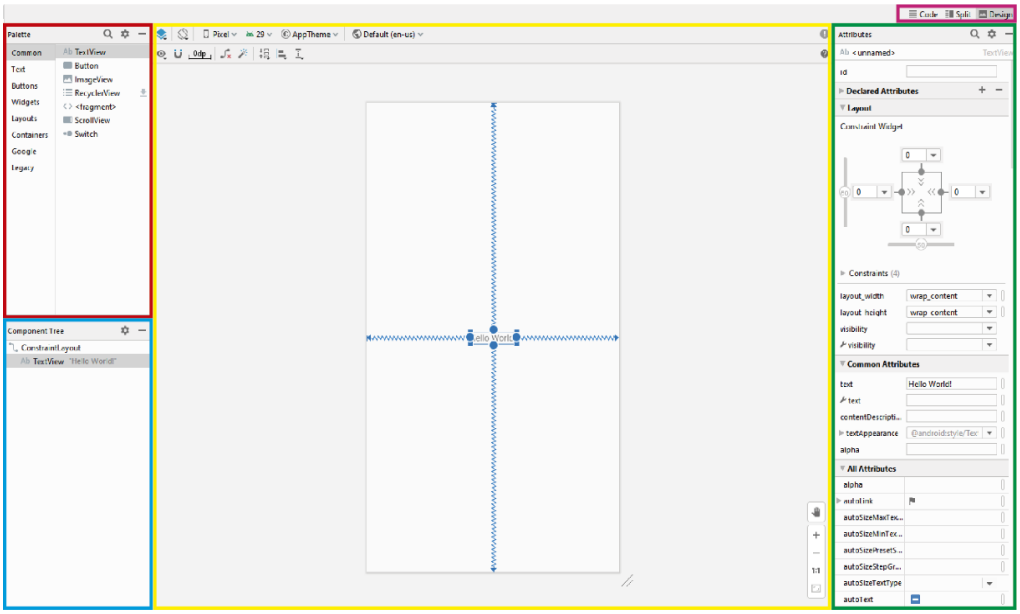


Figura 2.19. Espacio de trabajo

- Paleta

  En el recuadro rojo encontramos la paleta, en ella aparecen todos los elementos que podemos utilizar para dar forma a nuestra interfaz; botones, entradas de texto, mapas, etc. Y que explicaremos posteriormente.

- Árbol de componentes

  Debajo del anterior, en el recuadro azul, encontramos el árbol de componentes. En este aparecen todos los elementos que introduzcamos en nuestra interfaz, viéndose claramente en el árbol los que estas dentro de otros.

  En este caso podemos ver que el TextView con el texto “Hello World!” está dentro de un Layout de tipo ConstraintLayout.

- Atributos

  A la derecha del espacio de trabajo, en verde, nos encontramos con la ventana de atributos, donde aparecen todas las características que les podemos asignar a los elementos; desde su nombre o el id con el que nos vamos a referir a ellos en el código, hasta su tamaño, color, etc.

- Tipo de vista

  En la parte superior, en el recuadro rosa, nos encontramos con tres pestañas, la pestaña code, la pestaña split y la pestaña design.

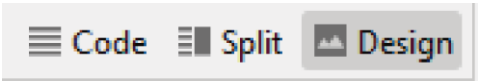


Figura 2.20. Tipo de vista

Estas pestañas tienen distintas características.

- Pestaña *Design*

  Esta pestaña nos permite ver la interfaz de forma gráfica, pudiendo arrastrar dentro de esta los diferentes elementos y colocarlos en la posición deseada

.

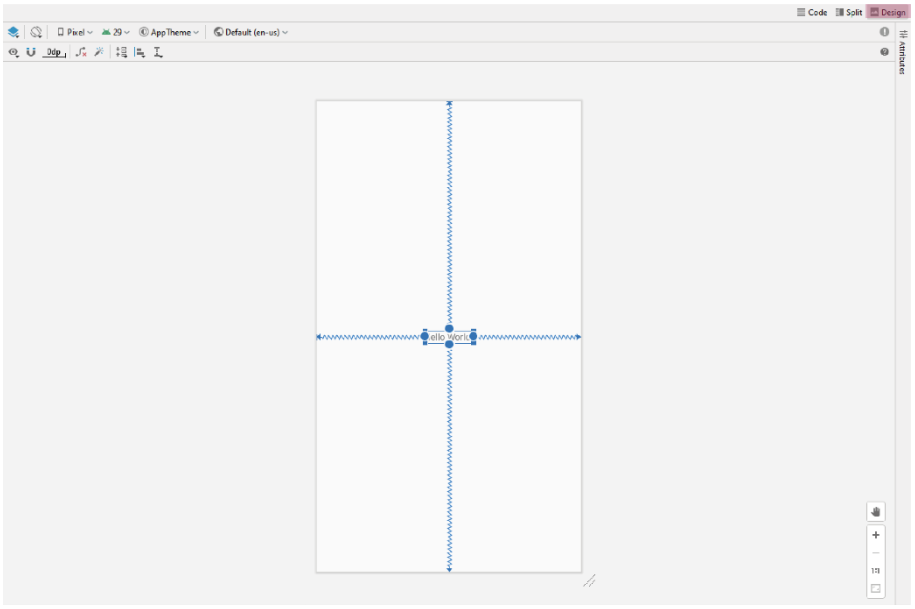


Figura 2.21. Pestaña Design

– Pestaña *Code*

En esta pestaña encontramos el código de la interfaz en XML. Podemos agregar más elementos escribiendo su código directamente, pero no podremos ver gráficamente dicha interfaz.

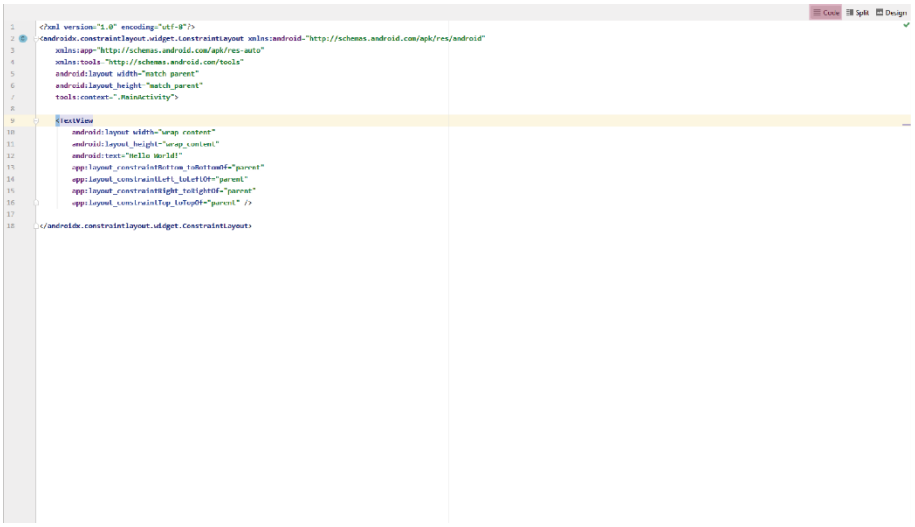


Figura 2.22. Pestaña Code

– Pestaña *Split*

Esta pestaña nos permite ver a la vez la interfaz en forma de código y en forma gráfica, lo que nos permite ver lo que estamos haciendo exactamente cuándo modificamos el código.

En esta pestaña sí que podemos recolocar los elementos que ya están en el lienzo arrastrándoles, pero no tenemos acceso al árbol de componentes ni a la paleta, por lo que solo podemos introducir elementos nuevos escribiendo su código, no arrastrándolos.

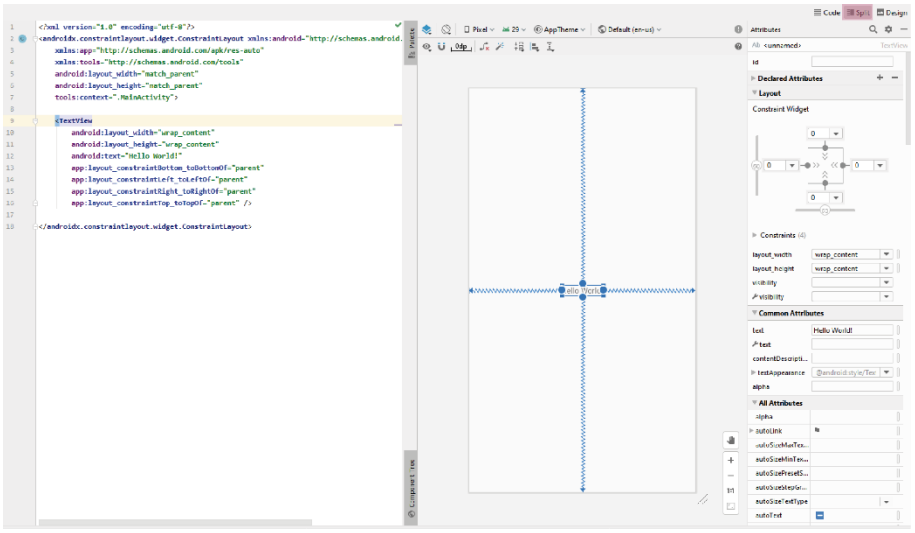


Figura 2.23. Pestaña Split

- Lienzo

El lienzo es la parte en la que diseñamos la interfaz, encuadrada con el rectángulo amarillo. Como he explicado antes, podemos ver el lienzo en forma de código, gráficamente o mixta.

Si visualizamos el lienzo en la pestaña *design* tenemos otras tres subpestañas para la forma de visualizarlo.

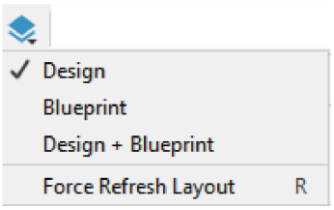


Figura 2.24. Lienzo

– *Design*

En la vista design vemos los elementos tal cual aparecerán en nuestro dispositivo, con todas sus características, colores, imágenes, tipografías, etc.

– *Blueprint*

En esta vista vemos como se relacionan los elementos de la interfaz entre ellos. En esta vista solo se ven los contornos y las líneas básicas de cada elemento, sin aparecer imágenes, colores, etc.

– *Design + blueprint*

Podemos interactuar con las dos vistas a la vez.

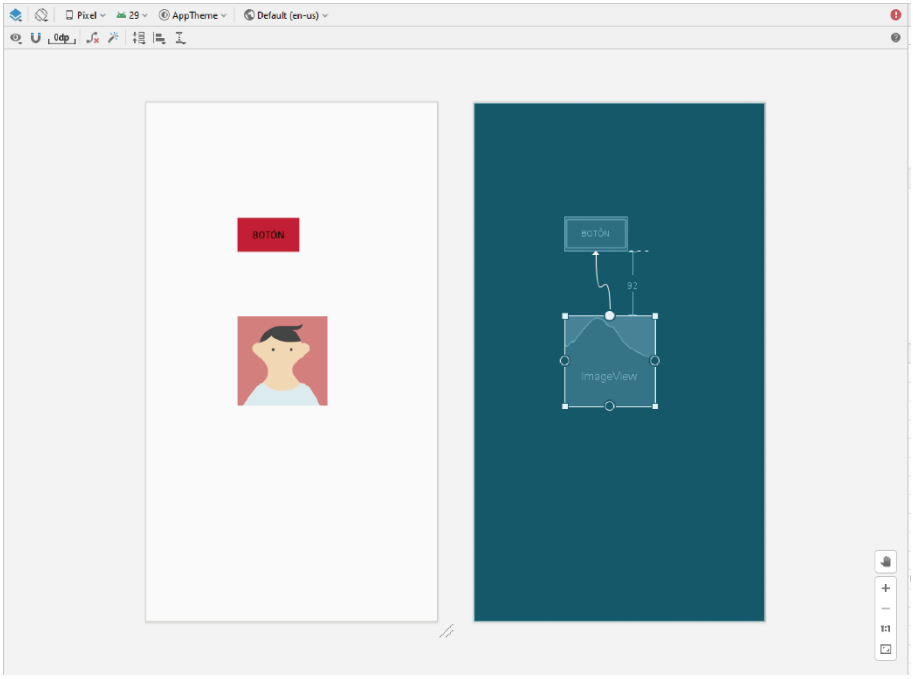


Figura 2.25. Design + blueprint

2.7.2. TextView

Es un elemento que sirve para mostrar un texto al usuario.[33]

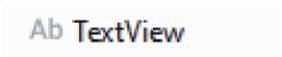


Figura 2.26. TextView

Algunas de las propiedades que podemos modificar en un TextView son las siguientes:

- *android: text*

Lo utilizamos para modificar el texto que va a visualizar el usuario. Si modificamos el texto directamente en la casilla text, nos saldrá un aviso en

el programa, que, a pesar de no interferir a la hora de ejecutar la aplicación, es mejor eliminar.

Esto lo conseguimos a través de la pestaña *strings,* que se encuentra dentro del directorio res con el siguiente código:

```
<string name="TextView">¡Hola Mundo!</string>
```

- *android: textColor*

Como se puede .observar en el nombre, lo utilizaremos para cambiar el color al texto

- *android: fontFamily*

Utilizamos este comando para cambiar la tipografía del texto, teniendo en cuenta que las tipografías por defecto son Droid Sans, Droid Serif y Droid Sans Mono.

Es posible utilizar una tipografía distinta descargando su archivo .ttf y pegándolo en un directorio que creado dentro del directorio *res* al que llamaremos *font.*

2.7.3. EditText

Un EditText es un elemento muy similar a un TextView, cuya principal diferencia es que en este caso es editable por el usuario.[14]

Hay muchos tipos de EditText como se puede ver, desde uno que te deje introducir solo texto, a otro que te deje introducir solo números, pasando por algunos especiales para introducir horas o fechas.

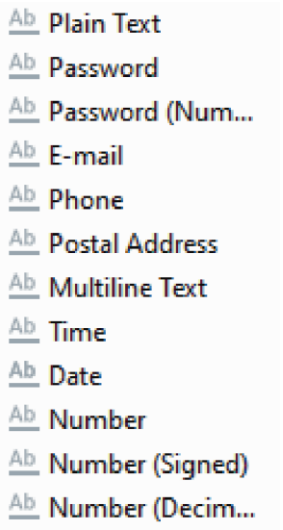


Figura 2.27. EditText

En este caso una de las propiedades que hemos señalado en el TextView, *android: text,* cambia ligeramente.

- *android: hint*

Es una propiedad muy similar a *android: text* en un TextView, pero en este caso, al ser un EditText editable, el texto que añadamos en *android: hint* desaparece al introducir un texto el usuario.

2.7.4. Button e ImageButton

Un botón es uno de los elementos que más se utilizan a la hora programar una aplicación. Su utilidad es realizar una acción cuando el usuario pulse sobre él Dentro de esta categoría tenemos dos tipos, Button e ImageButton.

- Button

En este caso dentro del botón solo tenemos texto. En este tipo de botón podemos modificar su color de fondo, la tipografía o el color del texto.

Por defecto el texto aparece escrito en mayúsculas, lo cual podremos desactivar introduciendo el siguiente código:

```
android:textAllCaps="false"
```

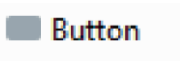


Figura 2.28. Button

- ImageButton

Al contrario que en un Button, dentro del botón tendremos una imagen, la cual podremos seleccionar entre las opciones de Android Studio o crear nosotros, pegándola en formato .png dentro del directorio *Drawable.*

En este tipo de botón podremos modificar el color de la imagen y el del fondo.

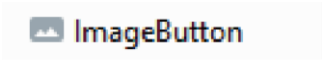


Figura 2.29. ImageButton

2.7.5. CheckBox

Este elemento es un tipo de botón que tiene dos estados, activo o inactivo, y que se comporta prácticamente igual que un botón.

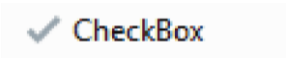


Figura 2.30. CheckBox

2.7.6. RadioGroup y RadioButton

Un RadioButton es otro tipo de botón muy similar a un CheckBox ya que también tiene dos estados, activo o inactivo.

En este caso un RadioButton no puede utilizarse de manera individual como un CheckBox, si no que tiene que ir contenido dentro de un RadioGroup.

Otra de las principales diferencias con un CheckBox es que este se activa con un clic y se desactiva pulsando otra vez, mientras que un RadioButton una vez activado no se puede desactivar volviendo a pulsar, sino que se desactiva al activar otros RadioButton que esté contenido en su mismo RadioGroup.

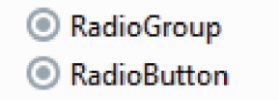


Figura 2.31. RadioGroup y RadioButton

2.8. Agregar un activity

Para poder navegar entre las distintas pantallas de una aplicación, debemos iniciar distintas activities

Como ya hemos mencionado anteriormente, una actividad o *activity* es el lugar en el que se visualizan los elementos que permiten la interacción con el usuario, y que comúnmente es conocido como pantalla.

Para crear un nuevo activity tendremos que hacer clic derecho en el directorio aplicación, para posteriormente desplegar new, activity y hacer clic en empty activity.

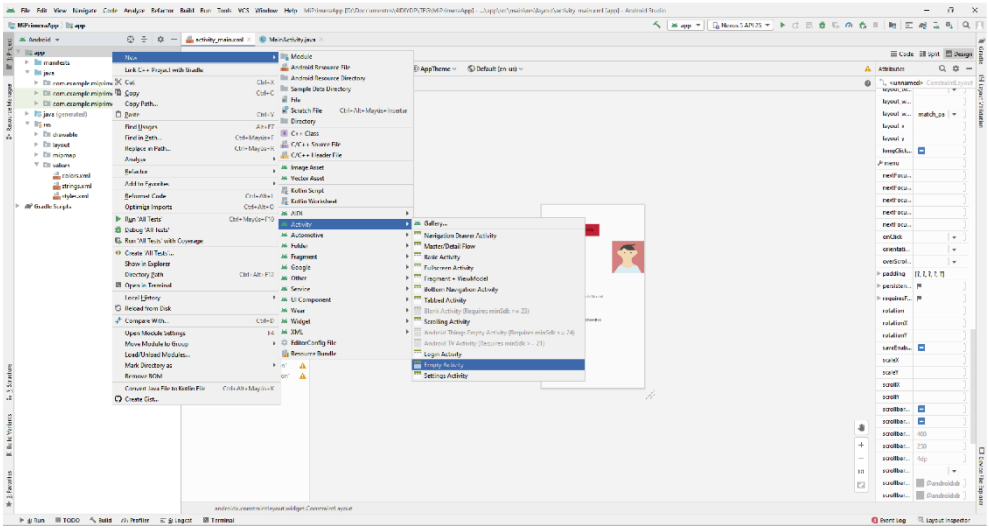


Figura 2.32. New activity

Una vez pulsamos en empty activity se abrirá la misma ventana que se abre al crear un nuevo proyecto y que ya explicamos anteriormente.

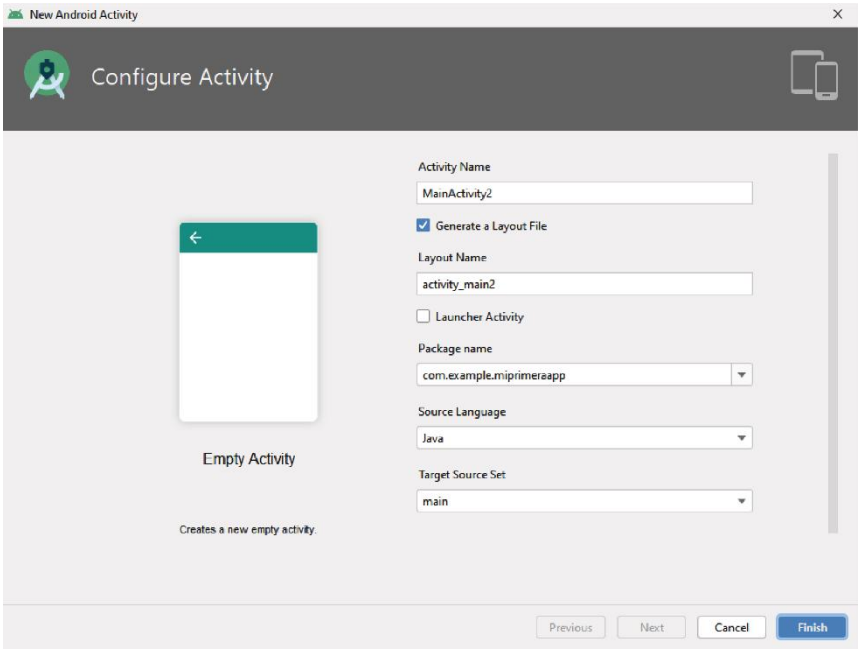


Figura 2.33. New Activity

En esta ventana tendremos que elegir un nombre para la nueva actividad, teniendo en cuenta que este nombre le utilizaremos posteriormente en el código java para llamar a esta nueva actividad

2.8.1. Llamar a otra activity

Una vez hemos creado esta nueva activity, deberemos realizar algunos cambios en la interfaz de la que queremos que sea nuestra activity principal.[21]

Para poder pasar de la pantalla principal a la nueva pantalla lo primero que deberemos hacer será introducir un botón en la interfaz, de esta manera al pulsar el botón podremos pasar de una actividad a otra.

Una vez hemos introducido el botón deberemos generar un método, que es un conjunto de instrucciones para realizar una acción, en este caso la de pasar a la siguiente pantalla. A este método le vamos a llamar *siguiente* y su código sería este:

```
public void siguiente (View view){
    }
```

Una vez hemos escrito este código en la pestaña .java de la actividad principal, tendremos que asociar el método al botón que hemos colocado

en la interfaz para que funcione. Esto lo haremos dentro de los atributos del botón, en la propiedad onClick, buscando el método *siguiente* que es el que hemos creado.

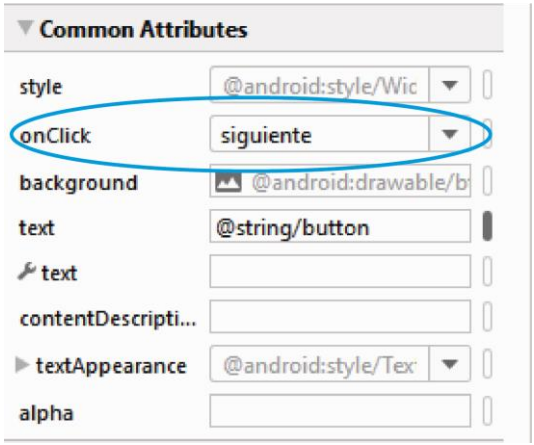


Figura 2.34. Método siguiente

Dentro de este método deberemos programar un Intent, que explicaré a continuación

2.8.2. Intent

Un Intent es un objeto que se utiliza para hacer que otro componente de la aplicación realice la acción que solicitemos.

El Intent representa una "intención" de hacer algo. Un Intent se puede utilizar para realizar muchas tareas pero en este caso nos vamos a centrar en los que inician otra activity.

Así, programando el Intent dentro del método que hemos creado anteriormente, el código quedará así:

```
public void siguiente(View view){
    Intent siguiente = new Intent(this, SegundoActivity.class);
    startActivity(siguiente);
}
```

Una vez realizado esto, si ejecutamos la aplicación, al pulsar el botón, este nos llevará directamente a la otra activity.

2.9. ¿Qué es un Fragment?

Un *Fragment* representa una parte de la interfaz, pudiéndose combinar varios *fragments* a la vez en un mismo *activity*.[34]

Un *Fragment* no puede visualizarse por sí solo, sino que siempre debe estar contenido en un *activity*. Además, un mismo *fragment* puede ser utilizado en varias *activities*.

2.9.1. Crear un nuevo *Fragment*

Crear un nuevo *fragment* es muy sencillo, y es muy similar a crear un nuevo *activity*.

Deberemos hacer clic derecho en el directorio *app*, para posteriormente dar a *new*, *fragment* y por último seleccionar el tipo de *fragment* que queremos añadir. El más utilizado es el *fragment*(*blank*).

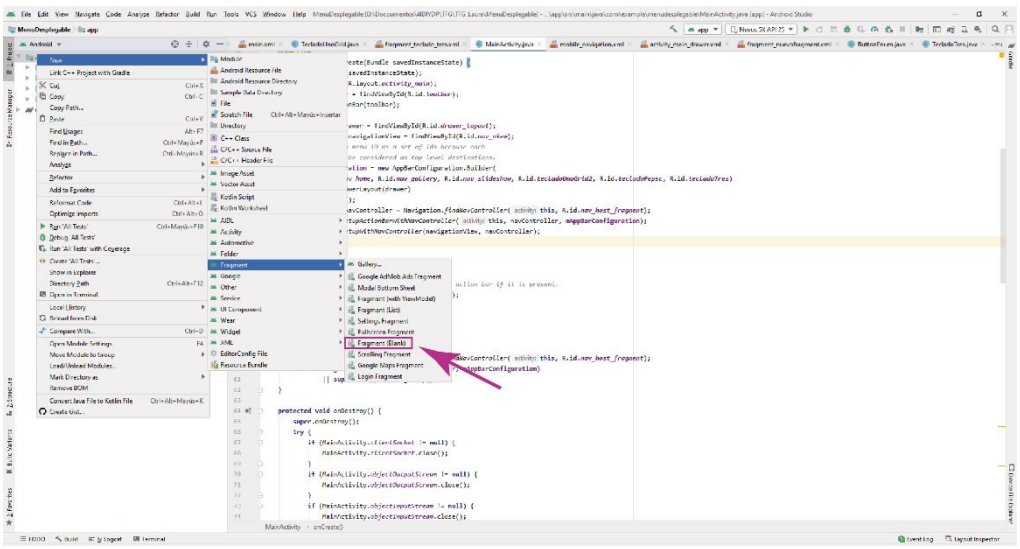


Figura 2.35. New Fragment

Se abrirá una ventana donde deberemos asignar un nombre al nuevo fragment.

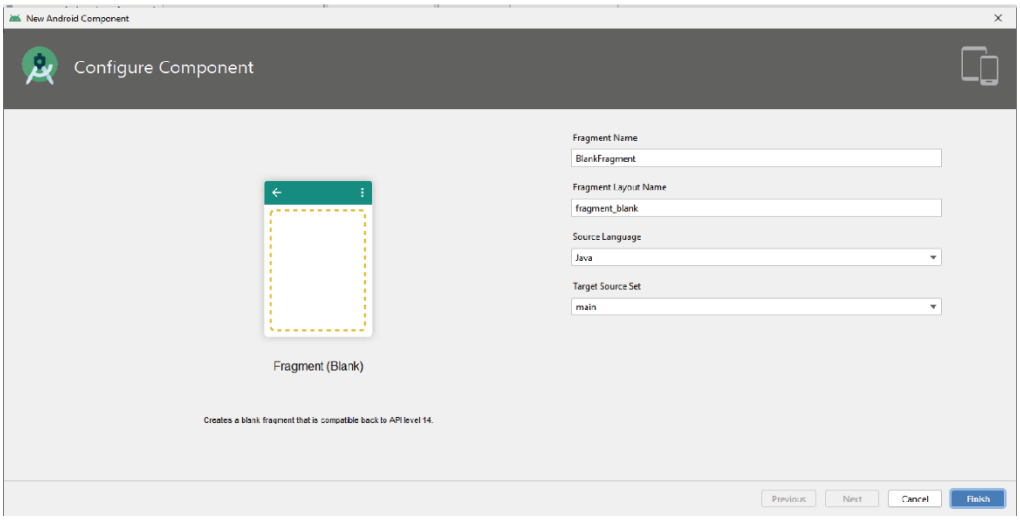


Figura 2.36. New Fragment

2.9.2. Como agregar un Fragment a un Activity

Para agregar un fragment a un activity tenemos que trabajar desde el archivo .xml del activity.

Agregaremos el siguiente código:

```
<fragment android:name="com.example.news.ArticleListFragment"
        android:id="@+id/list"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
```

Con el atributo android:name, lo que hacemos es llamar al fragment.
El resto de los atributos funcionan igual que en el resto de los elementos.

2.10. Navigation Drawer Activity

Un navigation drawer es el menú desplegable que aparece en muchas aplicaciones al deslizar el dedo desde el lado izquierdo hacia el centro.

Podemos tomar como ejemplo el menú principal de Twitter que utiliza este tipo de activity.

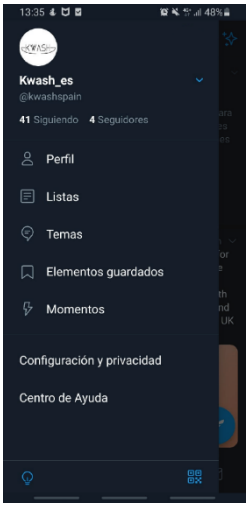


Figura 2.37. Ejemplo Navigation Drawer

Para iniciar un proyecto que contenga un navigation drawer tendríamos que seguir los mismos pasos que hemos descrito para crear un proyecto, pero en vez de abrir una *empty activity* (actividad vacía), deberemos abrir una *navigation drawer activity.*

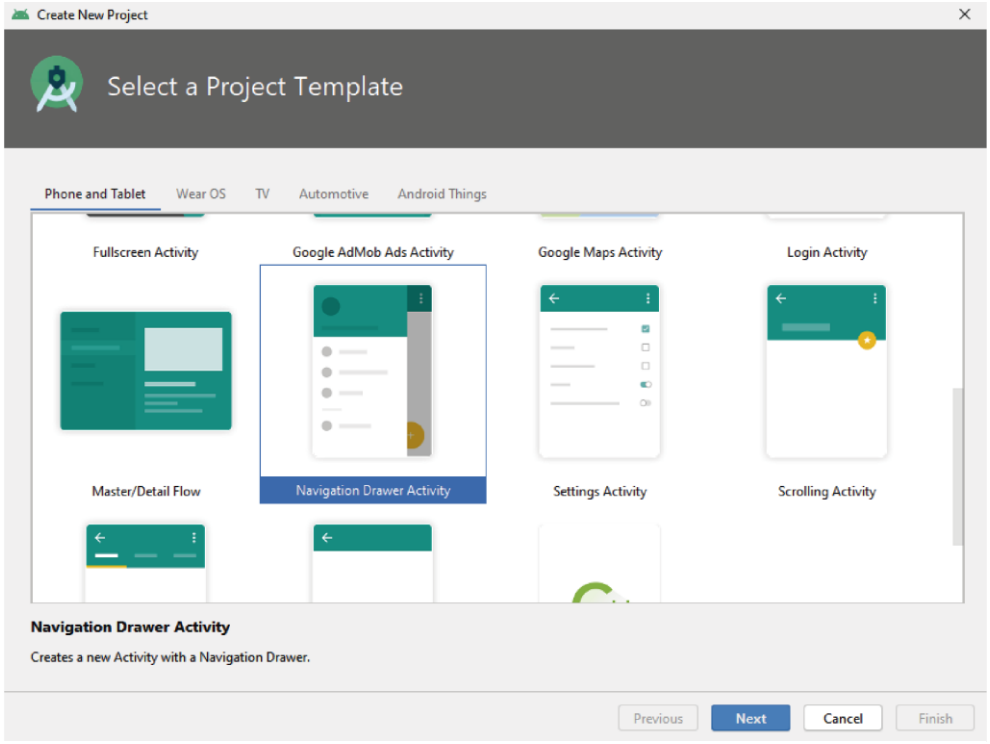


Figura 2.38. Abrir una Navigation Drawer Activity.

2.10.1. Componentes

Una vez abierto el navigation drawer activity vamos a pasar a analizar sus componentes.

Lo primero que aparece es el main activity, que en este caso, es el menú lateral desplegado donde podemos ver los diferentes activities a los que podemos acceder junto con un icono cada uno.

Otro de los cambios que se pueden observar fácilmente es que en el directorio *res*, se han añadido dos directorio nuevo, *menu* y *navigation*.

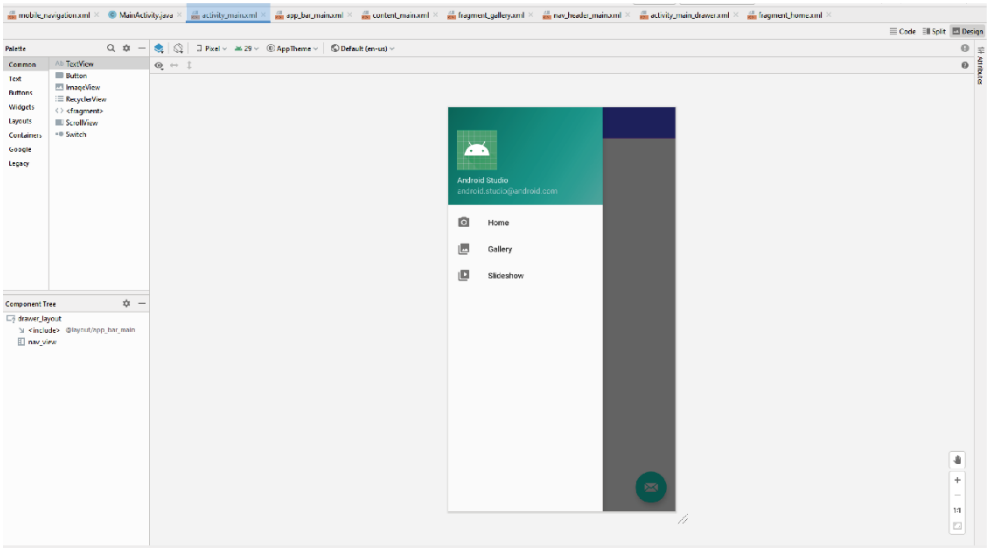


Figura 2.39. Main Activity

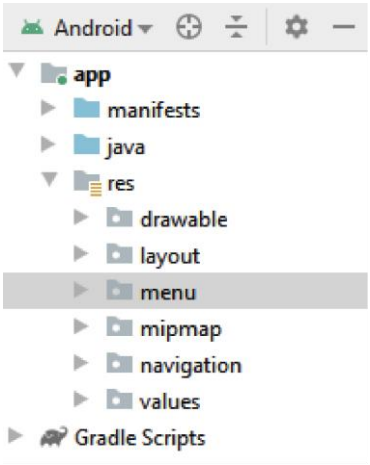


Figura 2.40. Directorio menu.

Como he explicado antes, el menú desplegable que te lleva al resto de activities se encuentra en el main activity, pero para poder cambiar los nombres y los iconos tenemos que recurrir al *activity_main_drawer*, que se encuentra en el directorio *menu.*

En este menú desplegable también observamos que hay un encabezado, donde aparece un icono, un nombre y una dirección de correo. Todos estos elementos se pueden modificar, añadir o quitar en el *nav_header_main*.

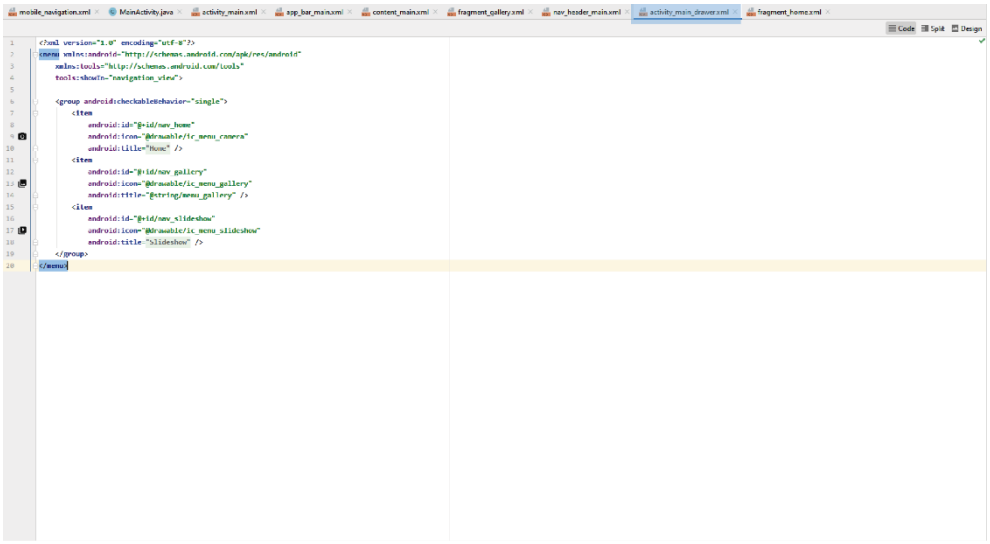


Figura 2.41. activity main drawer

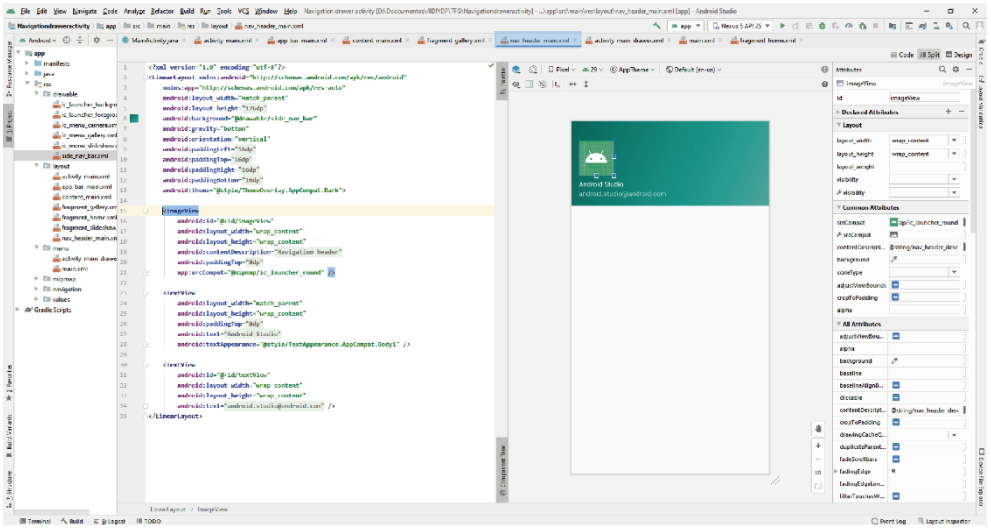


Figura 2.42. Nav_header_main

Además, en el navigatión activity drawer, aparece por defecto un floating action button, que es un botón que aparece en todas las activities de la aplicación.[8]

Si queremos modifciar este botón, cambiarlo de color o incluso eliminarlo debemos acceder a app_bar_main, donde también podemos modificar el color de la barra donde aparece el desplegable del menú.

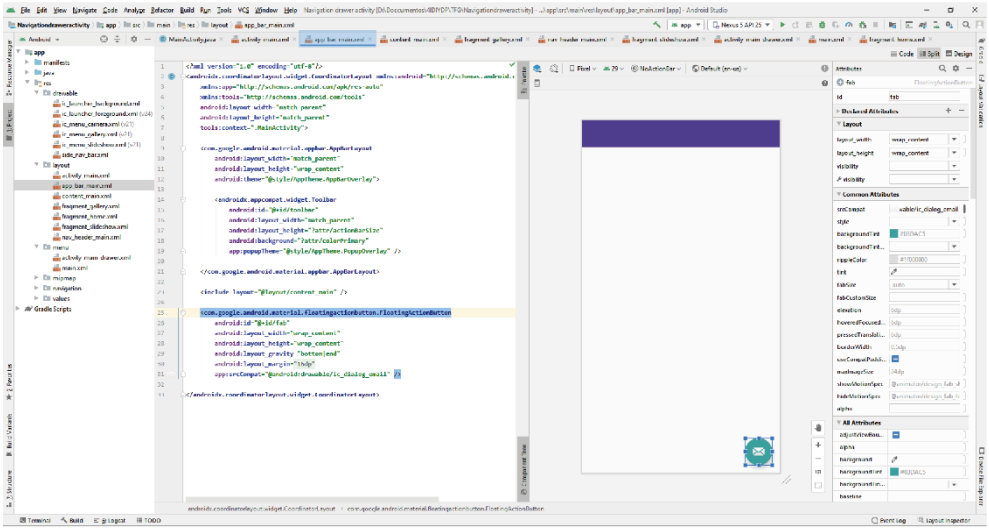


Figura 2.43. App_bar_main

2.10.2. Navegación

Programar la navegación en un navegación drawer activity es relativamente sencillo, ya que puedes hacer todo desde una única pestaña, *mobile_navigation*, que encontramos en el directorio *navigation.*

Lo primero que tenemos que hacer es acceder a esa pestaña, y veremos que aparecerán los tres *fragments* que aparece creados por defecto.

Como podemos observar, en uno de ellos, aparece al lado del nombre una casita, lo cual significa que es el *fragment* que va a aparecer al abrir la aplicación.

Para crear un *fragment* nuevo, lo único que tenemos que hacer es pulsar el rectángulo con una cruz verde y se nos abrirá una ventana donde seleccionaremos "*create new destination*". Seleccionaremos un *fragment blank*, al que cambiamos el nombre y cómo podemos ver su archivo .xml aparecerá en el directorio layout y su archivo .java en el directorio java.

A continuación pulsaremos en el rectángulo con la cruz verde y añadiremos el fragment que hemos creado.

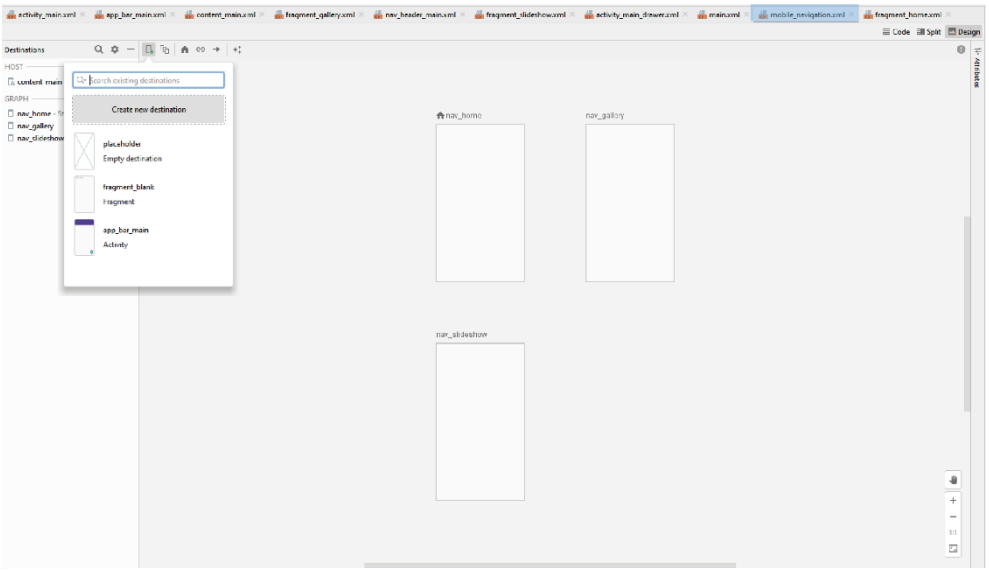


Figura 2.44. Mobile_navigation

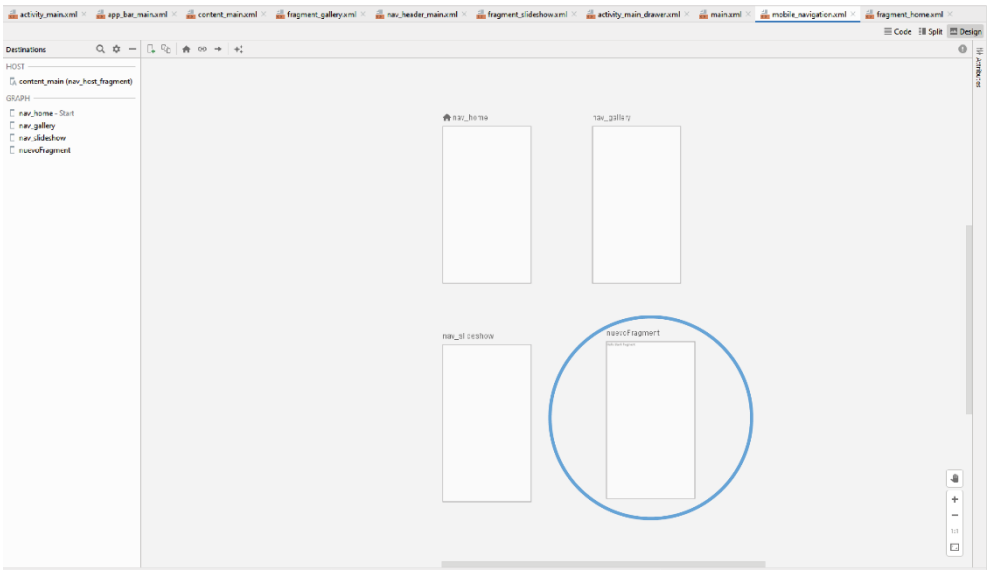


Figura 2.45. Nuevo fragment

Para mostrar el *fragment* desde el *navigation drawer*, entraremos dentro del *activity_main_drawer*, y añadiremos un nuevo *item* copiando el código de los anteriores, pero cambiando el id al nombre que hemos asignado al nuevo *fragment.*

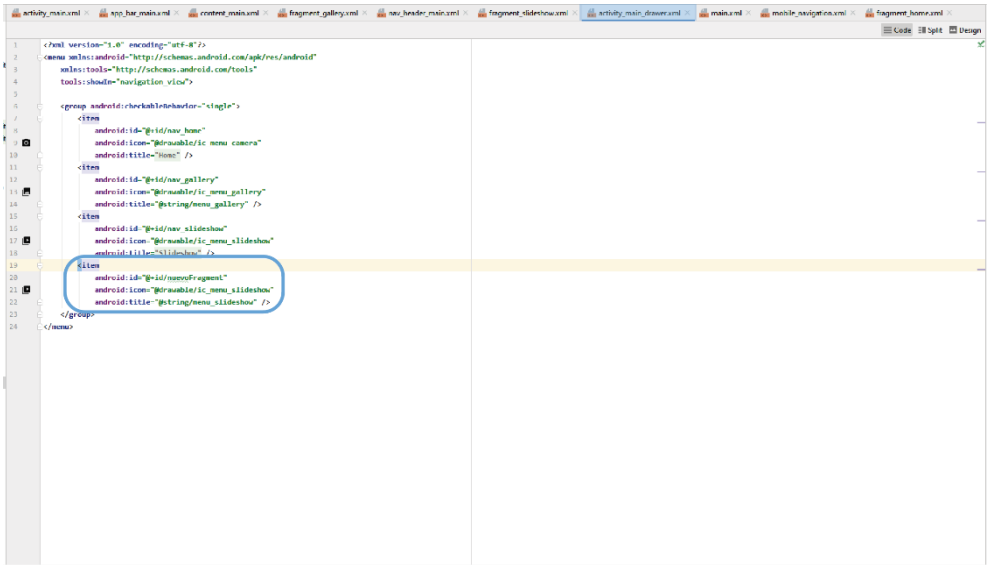


Figura 2.46. Nuevo item

Además, para que nos salgan las tres rayas que nos permiten abrir el menu desplegable y no solo una fecha de volver atrás, deberemos incluir en el código de *MainActivity*, en el lugar que indicaré ahora, el nombre del fragment que aparece *en el mobile_navigation.xml.*

```
mAppBarConfiguration = new AppBarConfiguration.Builder(
        R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow,
R.id.tecladoUnoGrid2, R.id.tecladoPepsc, R.id.tecladoTres)
        .setDrawerLayout(drawer)
        .build();
```

3. ESTADO DE LA CUESTIÓN

3.1. ¿Qué esperamos de la aplicación?

La aplicación nace con la intención de utilizarla para hacer servicios tipo Mago de Oz para un videojuego infantil de ordenador. Su principal función es utilizar el móvil como control remoto para sustituir el teclado del ordenador o para hacer alguna de sus funciones.

Además de la intención inicial para la que esta aplicación fue desarrollada, se puede utilizar para realizar más funciones. Por ejemplo, uno de los teclados desarrollados se podría utilizar para manejar una presentación de Power Point, permitiéndonos avanzar y retroceder por las diapositivas.

Otra de las intenciones de esta aplicación es que cualquier persona que tenga unos mínimos conocimientos de Java y Android, pueda agregar a la aplicación un teclado con sus propias necesidades. Por ejemplo, se podría crear un teclado para poder manejar la aplicación Netflix para Windows, teniendo control de play/pausa, el audio o los subtítulos.

Al poder controlar el teclado del ordenador con esta aplicación, otro posible uso podría ser para, conectando el ordenador a la tele, jugar a cualquier videojuego de ordenador que se maneje utilizando atajos de teclado, utilizando así el móvil como que fuera un mando.

3.2. Análisis de aplicaciones que funcionan como controles remotos en TV

En este caso, las únicas aplicaciones que he podido encontrar funcionan como mandos a distancia para televisiones o para proyectores, pero nos puede servir para analizar sus interfaces.

3.2.1. LG TV Plus

Esta aplicación es la desarrollada por LG para utilizar con los Smart TV de su marca.[38]

Pasaremos a analizar algunas de sus funciones y su interfaz.

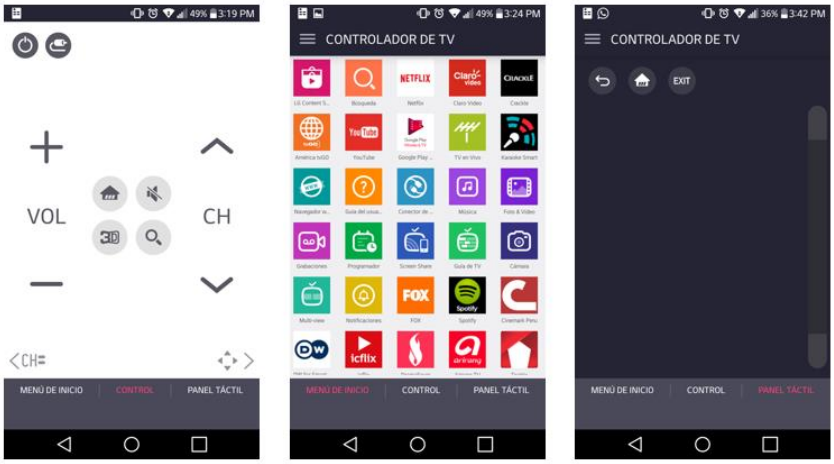


Figura 3.1. Controlador de TV LG TV Plus

Como podemos ver en la figura 3.1., el primer uso que podemos dar a esta aplicación es el de controlador de TV. Dentro de esta función, hay distintas subfunciones.

Lo que podemos ver en la foto de la izquierda es un mando a distancia al uso, con los controles de volumen y canal. Además dentro de este mando se puede ver en la parte inferior una flecha hacia la izquierda donde pone CH y una flecha hacia la derecha. En este caso, la interfaz podría utilizar un RelativeLayout o ConstraintLayout.

Con la flecha de la derecha pasamos a un menú con la lista de canales a los que podemos acceder, que podría estar utilizando para su interfaz un RelativeLayout de posición horizontal. Con la flecha de la izquierda podríamos acceder a un mando numérico para poder ir directamente al número de canal que queramos, utilizando probablemente en este caso un GridLayout.

Figura 3.2. Teclado numérico y lista de canales

En la imagen del medio (*figura 3.1.)* podemos observar que la segunda utilidad es una pantalla con los iconos de las distintas aplicaciones a las que podrías acceder en la Smart TV. En cuanto a la interfaz de esta pantalla, lo más probable es que se utilice un GridLayout.

En la imagen de la derecha del todo (*figura 3.1*.), nos encontramos con una pantalla táctil a través de la cual podríamos manejar un cursor que aparecería en la pantalla de la televisión. Es posible que esté utilizando un RelativeLayout o un ConstraintLayout.

Como podemos ver la aplicación utiliza un navigation drawer, ya que aparecen las tres líneas del menú desplegable en la parte superior derecha.

3.2.2. Epson IProjection

Esta aplicación, al igual que la anterior, es la oficial de la marca Epson y solo es compatible con los proyectores de esta marca.[35][36]

En esta aplicación también tenemos varias utilidades, que analizaremos a continuación.

En la pantalla principal podemos observar las diferentes utilidades que tenemos a través de ImageButton.

Podemos ver que en este caso lo más probable es que se esté utilizando un GridLayout, ya que aparece en forma de tabla.

También podemos ver que también utiliza un Navigation Drawer, ya que vemos las tres rayas del menú desplegable en la esquina superior izquierda.

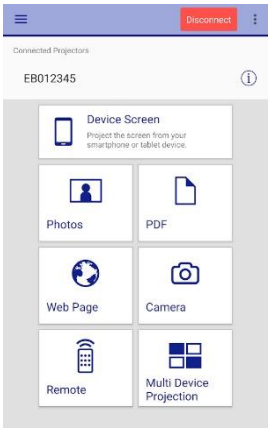


Figura 3.3. Pantalla principal

Dentro de esta pantalla principal, la primera utilidad que encontramos es la de fotografías. Aquí se pueden ver las fotos que queremos proyectar con los botones necesarios para navegar entre ellas. Lo más posible es que se esté utilizando un RelativeLayout o un ConstraintLayout.

En la opción de pdf, nos encontramos con una pantalla desde la que podemos escribir o dibujar sobre el pdf que se está proyectando en pantalla. En este caso también es muy posible que se utilice un RelativeLayout o un ConstraintLayout.

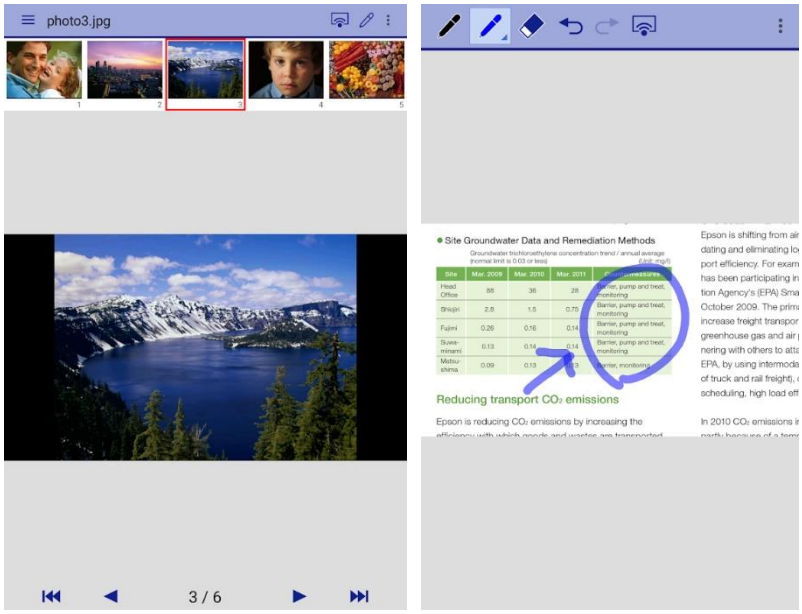


Figura 3.4. Photos y PDF

3.2.3. Mando a distancia para la TV

En esta caso, la aplicación también es un mando a distancia para la televisión, pero no es el mando oficial de una marca en concreto, si no que sirve para cualquier Smart TV.[37]

En este caso tenemos tres activities, pero solo tienen una utilidad, la de un mando a distancia físico, en control de volumen, de canal, teclado numérico y botón de on/off. La interfaz puede estar hecha en forma de GridLayout o de ConstraintLayout.

En la primera activity tenemos un único botón, utilizado para pasar a la siguiente activity donde elegiremos el modelo de nuestro televisor.

Figura 3.5. Mando a distancia para la TV

En la segunda activity aparece un listado de modelos de TV, utilizando probablemente un RelativeLayout de posición horizontal.

3.3. Análisis de aplicaciones que funcionan como controles remotos en PC

3.3.1. Remote mouse

Esta aplicación sirve para todo tipo de dispositivos android e IOS, tanto smartphones como tablets, convirtiéndolos en mandos a distancia para el PC.[48]

Dentro de esta aplicación tenemos varios controles como podemos ver en la figura 3.6. Por un lado encontramos un panel táctil que haría las funciones de ratón. Por otro lado, podemos utilizar el teclado del móvil de forma remota para poder escribir en el ordenador.

En cuanto a la interfaz, lo más probable es que se esté utilizando un FrameLayout para poder utilizar distintos fragments.

Figura 3.6. Control remoto para PC

3.3.2. MouseMote AirRemote

Esta aplicación solo puede ser utilizada con sistema operativo Android, ya sea un smartphone o una Tablet, pudiéndose controlar remotamente dispositivos Windows.[49]

Al igual que en la aplicación anterior tenemos dos controles, por un lado encontramos el panel táctil que funcionaría como ratón, y por otro lado encontraríamos el teclado.

Con respecto a la interfaz, es probable que, al igual que la aplicación anterior, esté utilizando un FrameLayout para poder introducir distintos fragments, utilizando probablemente el fragment del teclado un GridLayout.

Figura 3.7. MouseMote AirRemote

3.4. Aplicación de partida

Para realizar la aplicación, nos basamos en una aplicación de código abierto, que nos permitía reutilizar algunas partes de su código, como el utilizado para enlazar los botones a su respectiva tecla en el teclado del PC, pero introduciendo algunos cambios. [46] Su funcionamiento es muy sencillo, la aplicación instalada en nuestro dispositivo Android envía un código a la aplicación instalada en Windows. Esta a su vez anda un código al controlador de teclado que interrumpe la tecla que hemos ordenado.

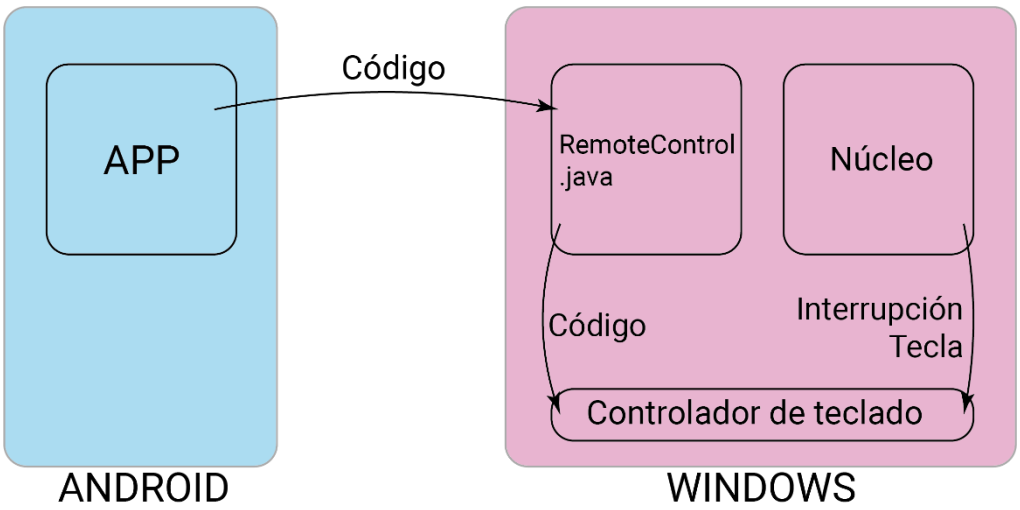


Figura 3.8. Funcionamiento Aplicación.

En el código aplicación también encontramos el código para poder incluir un panel táctil que haga las funciones de ratón, aunque en este caso no le utilizamos.

También tenemos otras prestaciones como la posibilidad de descargar archivos del ordenador en el móvil o enviar archivos del móvil al ordenador, pero en este caso estas funcionalidades tampoco eran necesarias.

Otra de las pantallas que podemos encontrar contiene los controles para el encendido/apagado del PC, incluyendo también un botón para poner el Pc en suspensión y otro para bloquearlo.

En este caso la parte del código que queríamos aprovechar es la utilizada para el teclado. En el caso de esta aplicación nos encontramos con un teclado básico como el que aparece en las aplicaciones que he analizado anteriormente, pero nosotros queremos conseguir realizar una aplicación donde se puedan incluir teclados para todo tipo de usos, y no el teclado estándar.

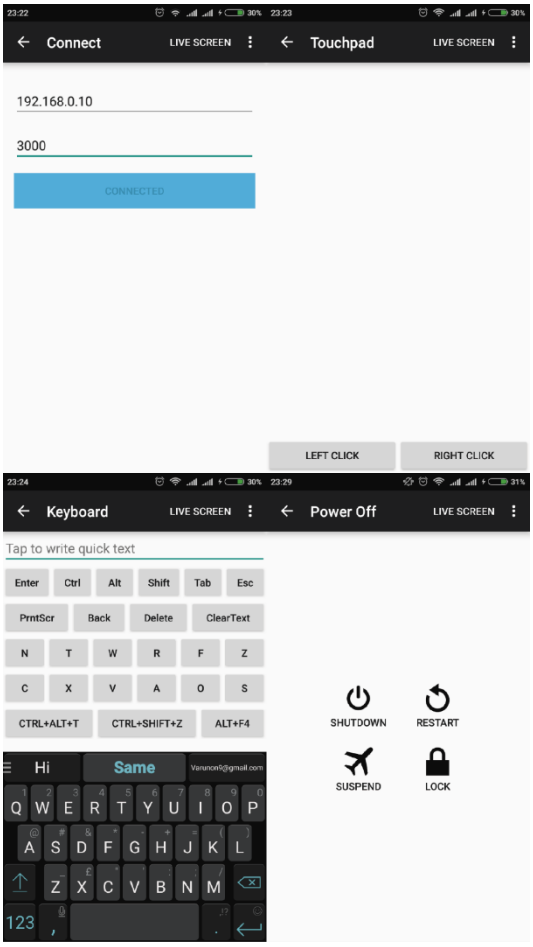


Figura 3.9. Aplicación de partida

4. APLICACIÓN

4.1. Diseño de la Interfaz

4.1.1. Primeras Ideas

Para el desarrollo de la aplicación, lo primero que queríamos conseguir era que existieran diferentes teclados ya creados, y el usuario pudiera navegar a través de ellos.

Para hacer eso la primera idea que surgió fue trabajar con fragments, que como ya explicamos antes son un parte de la interfaz, pudiendo utilizar varios fragments en un mismo activity.[40]

Hicimos un proyecto de prueba con botones para poder acceder a tres teclados diferentes, asignando a cada teclado un color, para comprobar su correcto funcionamiento.

Esta idea fue descartada, puesto que no quedaba limpio a la vista y, en caso de que el usuario con el tiempo quisiera añadir más teclados, el espacio del teclado en sí se iría reduciendo al ser ocupado por los botones.

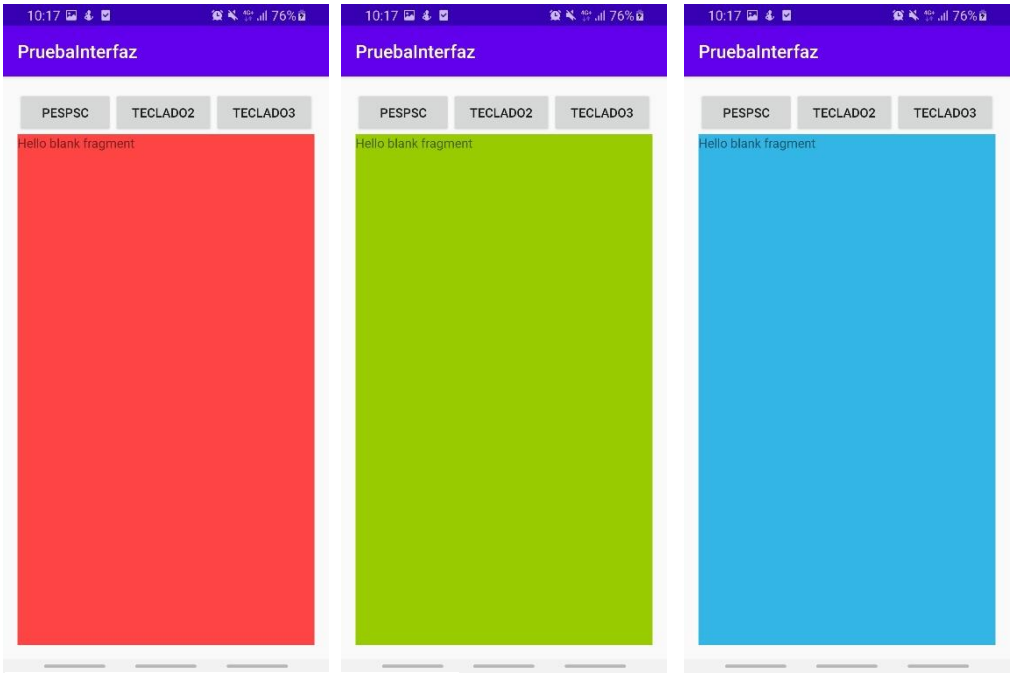


Figura 4.1. Prueba Interfaz con Fragments

Por lo que pensamos que sería una mejor opción desarrollar un menú desplegable, que como ya he mencionado antes se llama *Navigation Drawer,* de tal manera que los botones para acceder a los diferentes teclados estuvieran ocultos hasta el momento en que el usuario quiera cambiar a otro.

En este punto, a la vez que desarrollamos el *Navigation Drawer*, empezamos a trabajar en los distintos teclados, obteniendo un primer diseño de cada uno.

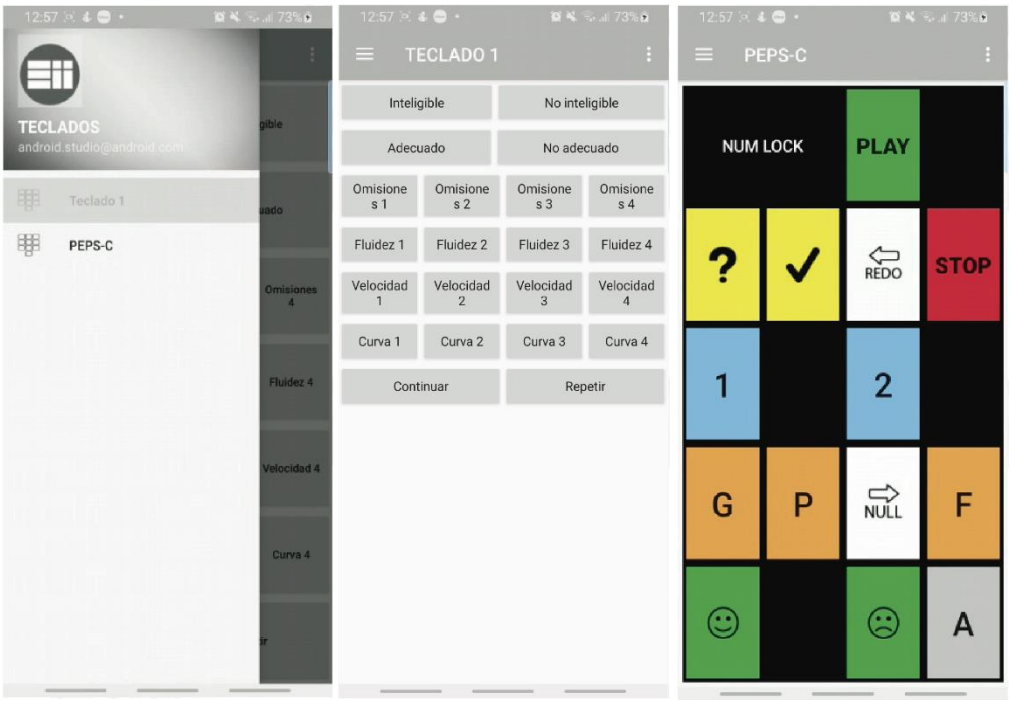


Figura 4.2. Primera versión Navigation Drawer

Esta primera versión se acerca más a la idea de la aplicación. Establezco el uso del *navigation drawer* para la navegación entre los teclados. Los teclados también sufrirán en versiones posteriores notables cambios, aunque mantendrán el uso del GridLayout que observamos en el teclado PEPS-C y que posteriormente se implementará en el teclado 1.

4.1.2. Splash Activity

Una vez elegido el *navigation drawer* para la navegación entre los distintos teclados, tenemos que empezar a pensar en la aplicación de forma global y en las distintas pantallas que va a tener a parte de los distintos teclados.

La gran mayoría de las aplicaciones tiene un primer *activity* que inicia al abrir la aplicación, y que suele ser a pantalla completa, con algún color de fondo y el icono de la aplicación en blanco.[41][43][44]

Decidimos introducir esto en la aplicación. En un primer momento, utilizamos el logotipo de la escuela para realizar una prueba de cómo podría programar esto. Para realizarlo utilizamos un botón que ocupa toda la pantalla con un pequeño texto abajo indicando que para pasar a la siguiente pantalla hay que pulsar.

Una vez conseguido esto, desarrollo un icono básico, en el que se puede ver un teclado rodeado por ondas que tienen la misma forma que las teclas.

Seguimos desarrollando esta pantalla, puesto que lo que queremos conseguir es que el *activity* esté unos segundos y después pase a la siguiente pantalla sin necesidad de que el usuario interaccione con ella, lo cual consigo con un *Splash,* quitando el texto de pulse para continuar.

Un Splash es una ventana o una pantalla que se muestra durante muy pocos segundos.

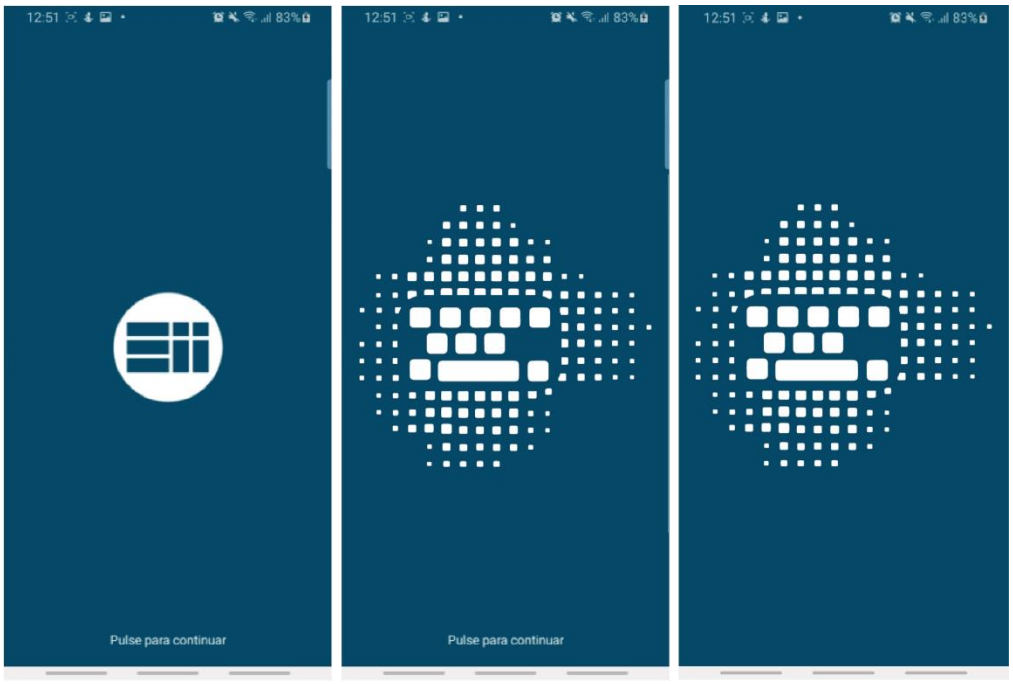


Figura 4.3. Splash Activity

4.1.3. Pantalla de acceso

Para poder enlazar en móvil con el ordenador y que este funcione como control remoto, necesitaremos introducir en la aplicación un IP que nos dará el ordenador, lo cual explicaremos en más profundidad más adelante.

Debido a esto es necesario incluir otro activity en el que te pida el IP y lo valide para poder crear la conexión con el ordenador, y una vez realizado esto continúe al menú desplegable donde aparecerán los teclados.

Dentro de esta pantalla nos encontramos con un TextView, que muestra el texto “Introduzca su IP”.

También aparece un EditText donde introduciremos el IP del ordenador y otro en el que introduciremos el número de puerto.

También tenemos un Button con el que validaremos los datos y pasaremos a la siguiente pantalla.

Figura 4.4. Pantalla de acceso

4.1.4. Menú desplegable

Como hemos explicado antes, el menú desplegable es la forma elegida para seleccionar los teclados, y se llama *navigation drawer.*

En un principio el color elegido para el navigation drawer era el gris, aunque decidimos cambiarlo, puesto que a pesar de ser un color elegante y serio, era demasiado apagado, por lo que nos decidimos por un azul.

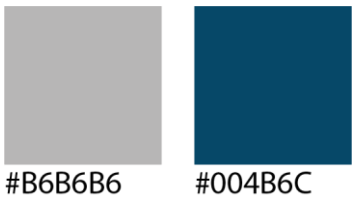


Figura 4.5.. Colores

El menú desplegable viene con unos iconos por defecto para cada uno de los botones de acceso a los fragments. decidimos que la mejor opción era utilizar un icono de un teclado.

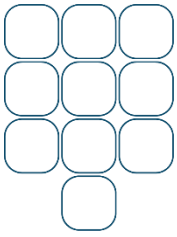


Figura 4.6. Icono teclado

También cambia con respecto a la primera versión la imagen que aparecía en la parte superior, que en este caso desaparece para aparecer solo el nombre de la aplicación.

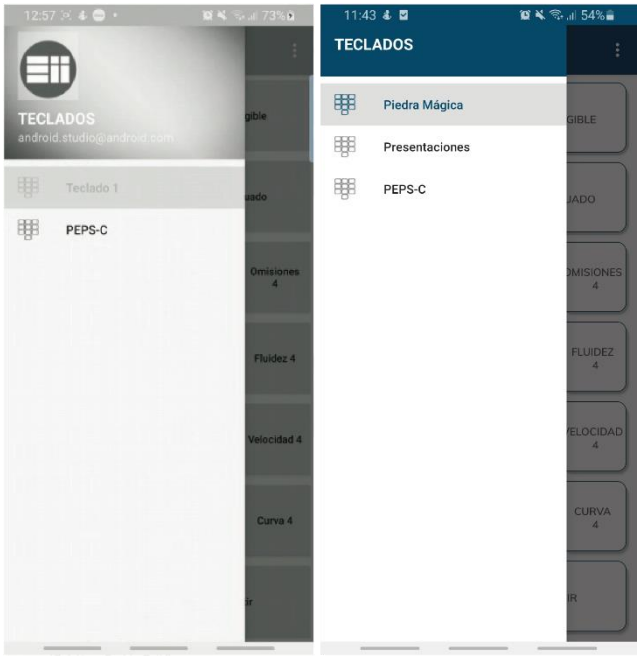


Figura 4.7.. Menu desplegable.

4.1.5. Teclados

En cuanto a los teclados, en un primer momento realizamos el diseño de dos, para posteriormente añadir uno más con un diseño más sencillo.

Para el diseño de la interfaz del teclado Piedra Mágica y del teclado PEPS-C he utilizado el GridLayout y para el teclado Presentaciones un LinearLayout.

En cuanto al primer teclado, es el que sufre el cambio más significativo respecto a las primeras ideas.

En las primeras ideas utilizábamos un relative layout, lo que suponía que el teclado no ocupaba la pantalla completa, cosa que en el último diseño sí que ocurre.

Otro de los cambios más significativos es el de los botones, que pasan de ser de un color gris oscuro a ser blancos con una fina línea del azul utilizado para la aplicación que los delimita.

Figura 4.8. Interfaz teclado 1

En cuanto al teclado PEPS-C, es en el que más cambios podemos notar desde las primeras ideas. En un principio las teclas eran cuadradas y ahora hemos redondeado las esquinas.

El fondo ha pasado de ser negro a ser blanco para dar mayor sensación de limpieza y de unidad entre los diferentes teclados. Las teclas, además, tienen una pequeña separación entre ellas y están bordeadas para definir mejor cada una. En este teclado también utilizamos un GridLayout cuyo código se encuentra en los anexos.

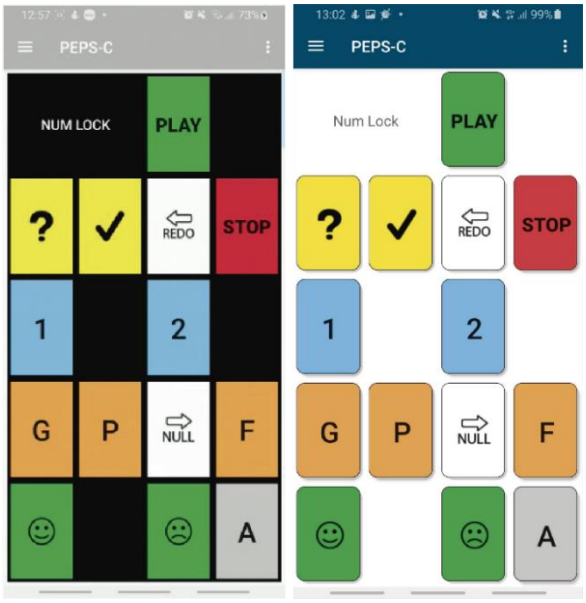


Figura 4.9. Interfaz teclado PEPS-C

El último teclado creado es mucho más sencillo que los anteriores. En él solo encontramos dos botones; anterior y siguiente. El estilo de estos botones es el mismo que el utilizado en el teclado piedra mágica, con las teclas en color blanco y el bordeado en el color azul utilizado en la aplicación.

Figura 4.10. Interfaz teclado 2

Como hemos explicado antes, estos teclados son los diseñados en un principio, pero la idea de la aplicación es que cualquier persona que sepa un poco de programación en Java y en Android, y que siga el manual del desarrollador que veremos más adelante, pueda incluir sus propios diseños de teclado.

4.1.6. Colores

Como ya hemos mencionado antes al hablar del menú desplegable, el color principal de la aplicación es el azul, que también se utiliza para el icono de la aplicación, debido a que simboliza fidelidad y seriedad. Muchas marcas emplean el color azul ya que se vincula con la credibilidad.[39]

En cuanto al texto, en lugar de utilizar el negro, utilizamos un gris oscuro ya que utilizar el negro sobre blanco en pantallas, ya sea aplicaciones, páginas web u otros, no es muy recomendable, y el gris mejora la legibilidad.[45]

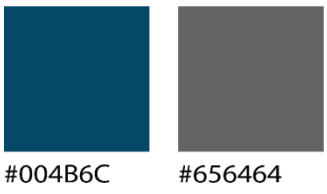


Figura 4.11. Colores de la aplicación

En el teclado PEPS-C, por ejemplo, que es un teclado que ya había sido desarrollado anteriormente, aparecen otros colores aparte de estos.

4.2. Componentes

Pasamos a desarrollar a continuación, con un poco más de detalle, los componentes utilizados para el diseño de la interfaz.

4.2.1. Utilización del GridLayout

Como ya explicamos en el apartado anterior, el GridLayout permite distribuir los elementos en forma de tabla en la interfaz. En este caso, al estar desarrollando teclados, es el Layout más adecuado para utilizar, aunque en el caso del Teclado Presentaciones al solo tener dos botones, no es necesario y serviría con un RelativeLayout.

Para poder entender mejor el funcionamiento y utilización de este Layout, vamos a explicar más en profundidad los controles utilizados en el código.

Adjuntamos a continuación una parte del código donde podemos observar estos controles:

```
<GridLayout
    android:useDefaultMargins="true"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="4"
    android:rowCount="7"
    android:padding="5dp">

    <Button
        android:id="@+id/intButton"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:layout_columnSpan="2"
        android:layout_gravity="fill"
        android:background="@drawable/sombratres"
        android:text="@string/inteligible"
        android:textSize="12sp"
        android:textColor="#656464"
        android:fontFamily="@font/mulishsemibold"
        />
```

En el rectángulo azul encontramos los controles que corresponden a la configuración de la tabla en su totalidad, mientras que en el rectángulo

verde encontramos los controles que configuran cada botón individualmente.

Pasamos a analizar para que sirve cada control para la configuración de la tabla:

- *useDefaultMargins;* esta orden hace que las celdas de la tabla tengan una separación entre ellas.
- *layout_width y layout_height;* sirven para establecer la anchura y altura de la tabla respectivamente, en este caso estamos diciendo que se adapte a las dimensiones de la pantalla.
- *columCount y rowCount;* determinan el número de columnas y de filas que tendrá la tabla respectivamente.
- *padding;* establece los márgenes respecto al borde de la pantalla.

En cuanto a los controles utilizados para la configuración de cada botón individualmente:

- *id;* es el nombre que se da a cada elemento y que luego se utilizará para hacer referencia a este en el código .java.
- *layout_width y layout_height;* como he explicado antes sirven para establecer la altura y anchura de cada botón. Este control siempre tiene que estar en los elementos que introduzcamos en la interfaz, pero en este caso es 0 porque la altura y la anchura lo va a establecer el control que voy a explicar a continuación.
- *layout_columnWeight, layout_rowWeight y layout_gravity;* con la combinación de estos tres botones establecemos la anchura y altura de cada botón, en este caso ordenándole que se adapte a la medida de cada celda.
- *layout_columnSpan y layout_rowSpan;* estos dos controles se utilizan para indicar el número de columnas y filas que ocupará cada botón respectivamente. Solo se deberá utilizar en caso de que el botón ocupe más de una fila o una columna, que es lo establecido por defecto.
- *background;* sirve para establecer el estilo de cada botón, lo cual también programamos y explicaré en el apartado siguiente.
- *text; establecemos el texto que va a aparecer en el botón*
- *textSize; sirve para seleccionar el tamaño de texto*
- *textColor; con el seleccionamos el color del texto*
- *fontFamily, sirve para seleccionar una tipografía si queremos que sea distinta a las establecidas por defecto.*

4.2.2. Diseño de los botones

Para establecer el estilo de los botones utilizaremos el control *android:background* como he dicho anteriormente.

En este caso hay varios tipos de botones, por un lado tenemos los utilizados en los teclados 1 y 2, los cuales son blancos con un delineado del color azul empleado para el resto de la aplicación. Este botón cambia de estado cuando lo pulsas, apareciendo una sombra azul.

Para los botones del teclado PEPS-C tenemos más variedad, ya que las teclas son de distintos colores, aunque todos tienen el mismo estilo, tiene el fondo de un color y están delineados en negro y al pulsar aparece un sombreado de su color de fondo oscurecido.

Los botones de todos los teclados tienen una sobra gris por detrás para que dé la sensación de que tienen un poco de relieve.

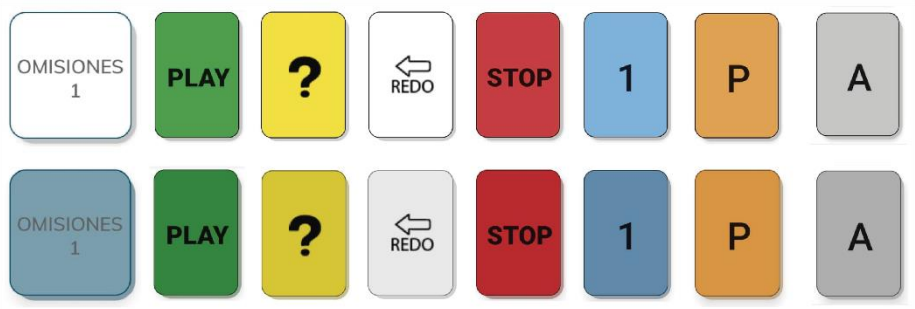


Figura 4.12. Botones y sus cambios de estado

Para crear un estilo nuevo para los botones hay que seguir los siguiente pasos:

Lo primero que deberemos hacer es dar clic derecho en la carpeta drawable, pulsar new y en *Drawable Resource File.*

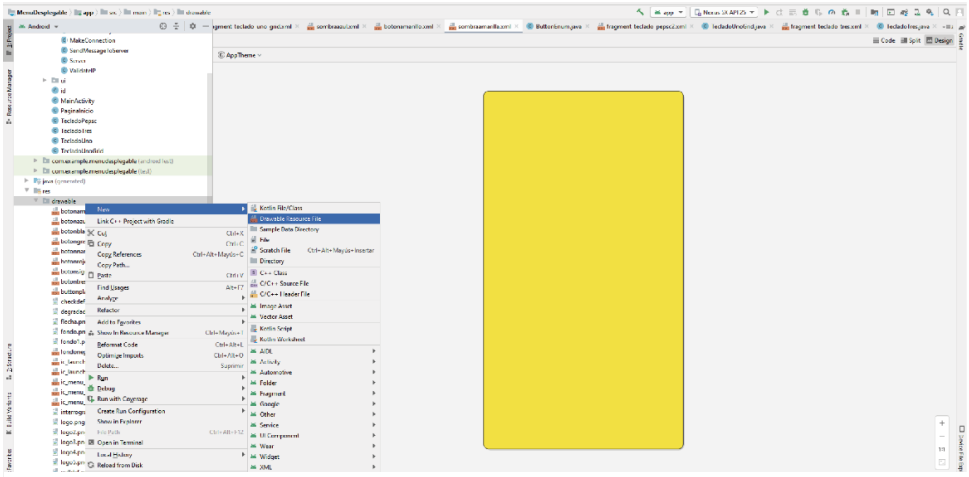


Figura 4.13. Nuevo Drawable Resource File

Una vez pulsamos en *Drawable Resource File* se abrirá una pestaña llamada *new Resource File*, donde deberemos seleccionar un nombre, igual que hemos hecho al crear fragments o activities.

Como podemos ver en la siguiente imagen, la pantalla desde la que creamos un estilo para un botón tiene la misma configuración que la pantalla de .xml, con una ventana de código, otra de diseño y una combinada.

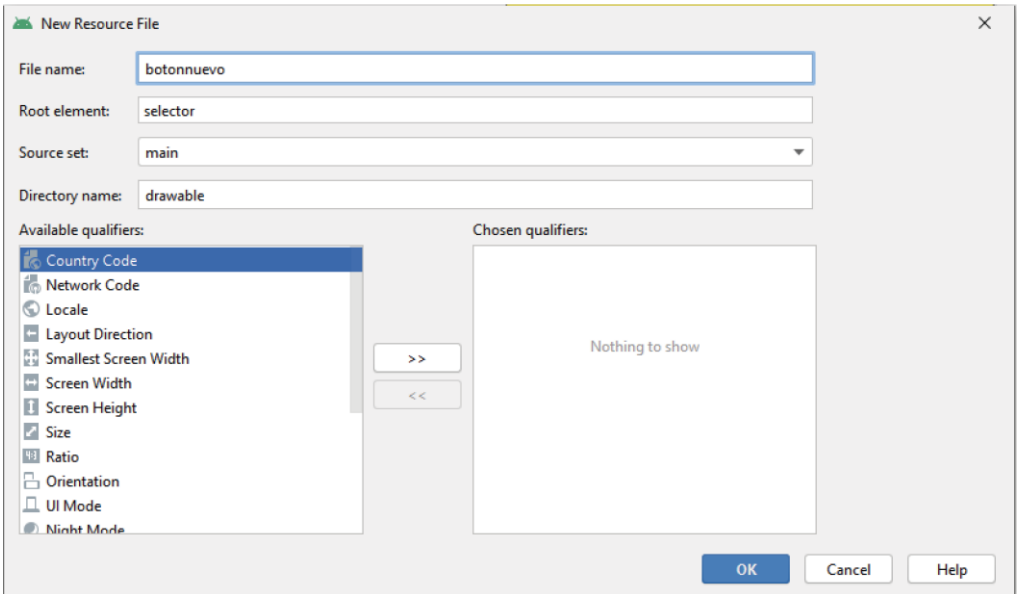


Figura 4.14. New Resource File

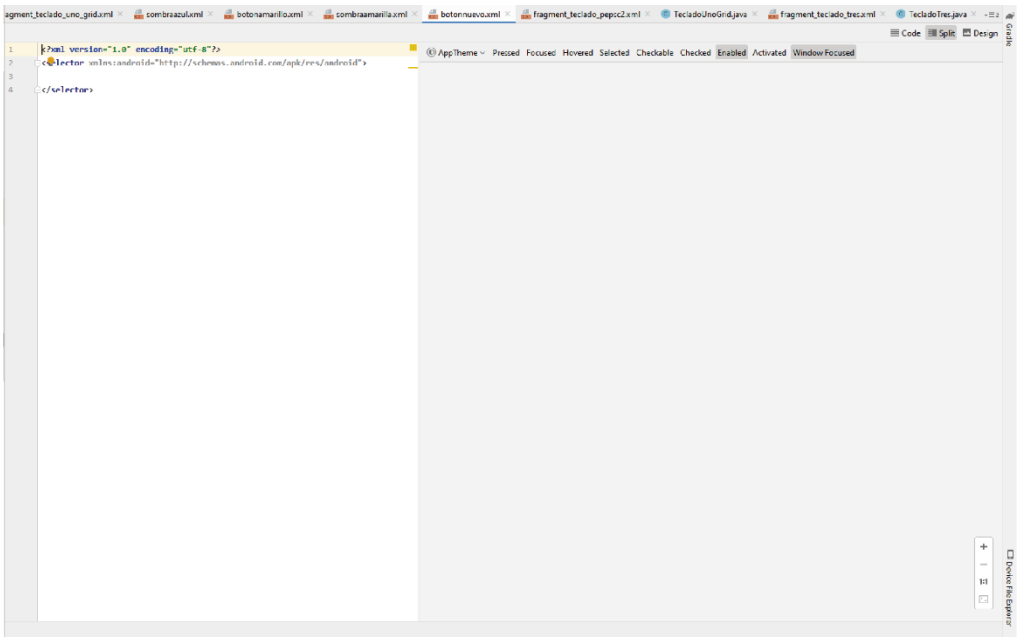


Figura 4.15. New Resource File

En esta ventana incluimos el siguiente código, el cual procedo a explicar:

```
<layer-list
xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:left="3dp" android:top="3dp">
   <shape>
      <solid android:color="#A9B8B8B8" />
      <corners android:radius="13dp" />
   </shape>
   </item>
   <item android:bottom="3dp" android:right="3dp">
   <shape>
   <corners
      android:bottomLeftRadius="10dp"
      android:bottomRightRadius="10dp"
      android:radius="0.1dp"
      android:topLeftRadius="10dp"
      android:topRightRadius="10dp" />
   <solid android:color="#FFEB3B" />
   <stroke
      android:width="2px"
      android:color="#000000"></stroke>
   </shape>
   </item>
</layer-list>
```

Como podemos observar hay dos *items*, es decir, dos partes que conforman el botón, por un lado tenemos el *item* verde, que es el que representa la sombra gris y por otro lado tenemos el que está contenido por el recuadro azul que representa el botón propiamente dicho.

Con el comando *shape* lo que estamos programando es la forma que queremos que tenga nuestro botón, en este caso estamos dando un color de fondo con el comando *solid* y estamos diciendo que tenga las esquinas redondeadas con el comando *corners*.

Además en el *item* del botón tenemos el comando *stroke*, con el que hacemos que se vea la línea de contorno.

Para hacer que el botón se sombree al pulsar tenemos que crear un nuevo *Drawable Resource File* en el que encontramos el siguiente código:

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
android:color="#C1B22A">
   <item android:drawable="@drawable/botonnuevo"></item>
</ripple>
```

En este caso, lo que estamos haciendo con el control ripple es precisamente que aparezca la sombra al pulsar, estableciendo el color de esa sombra, y a su vez anexamos esto al botón que habíamos creado anteriormente.

Para dar este estilo al botón lo único que tendremos que hacer será escribir su nombre en background en el archivo .xml.

4.2.3. Imágenes para los ImageButton

Para las imágenes que aparecen en los ImageButton he tenido que realizar gráfico vectoriales, de tal manera que no se pixelen al visualizarse en pantallas de distinto tamaño.

Todas las imágenes han sido desarrolladas en Illustrator con línea negra sobre fondo blanco, ya que posteriormente en Android Studio es posible cambiar el color.

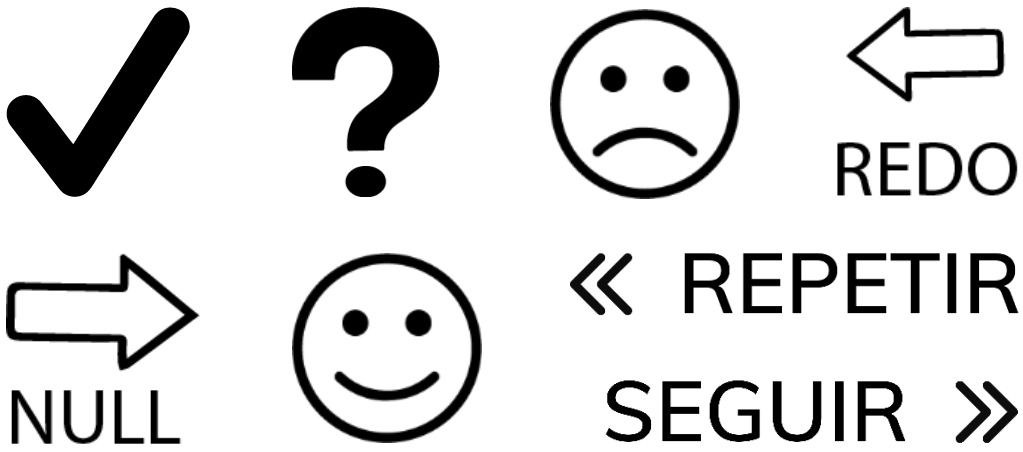


Figura 4.16. Imágenes para los ImageButton

4.3. Manual del desarrollador

Desarrollo este apartado para que cualquier persona que tengo unos conocimientos mínimos de programación pueda modificar la aplicación y añadir sus propios teclados.

Lo primero que deberías saber es como funciona esta aplicación. A la vez que te descargas el código de la aplicación android, en la misma carpeta aparecer un ejecutable *RemoteControlPC y el archivo apk.*

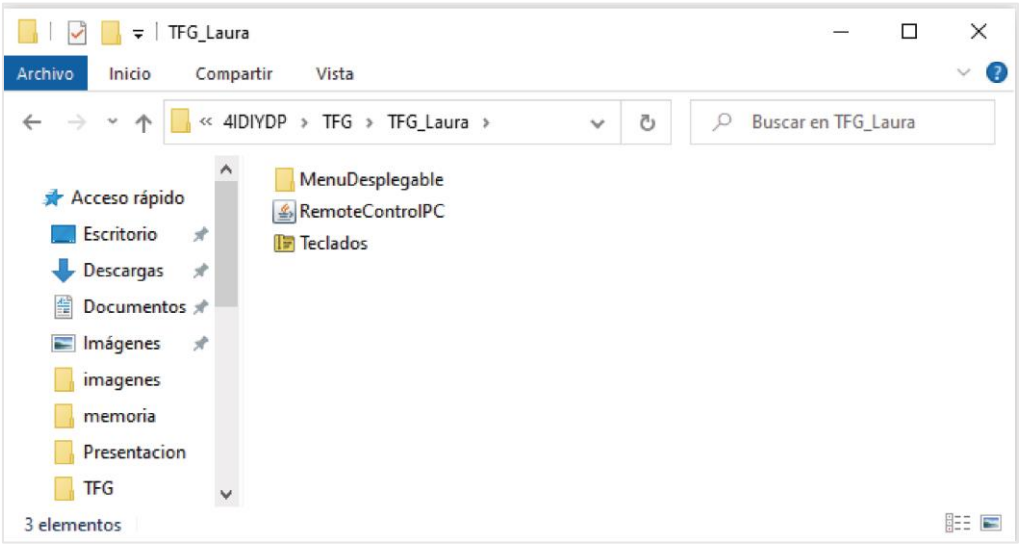


Figura 4.17. Carpeta descargable.

Para que te hagas una idea rápida, la carpeta *MenuDesplegable* es el proyecto de Android Studio, Teclados es la aplicación para Android y el programa *RemoteControlPC* es para Windows.

El funcionamiento es muy simple, a la vez que ejecutamos el *MenuDesplegable* en el móvil, deberemos ejecutar *RemoteControlPC* en el ordenador, debiendo estar ambos dispositivos conectados a la misma red. De esta manera, *RemoteControlPC* nos dará una dirección IP y un puerto que nos pedirá la aplicación móvil para poder conectar ambos dispositivos. De esta manera, cuando tu realices una acción con la aplicación móvil, se enviará una instrucción al ordenador, que será traducida por el código fuente de java de *RemoteControlPC* y que mandará una instrucción al propio ordenador.

Una vez entendido el funcionamiento podemos proceder a explicar cómo modificar el archivo en Android Studio.

4.3.1. Instalar Android Studio

Para poder desarrollar en Android necesitaremos tener varias cosas instaladas en el ordenador. Por un lado deberemos tener instalado Java y por otro lado deberemos tener instalado Android Studio.

Para poder descargar e instalar Java simplemente tendremos que entrar en su página oficial java.com/es/download.

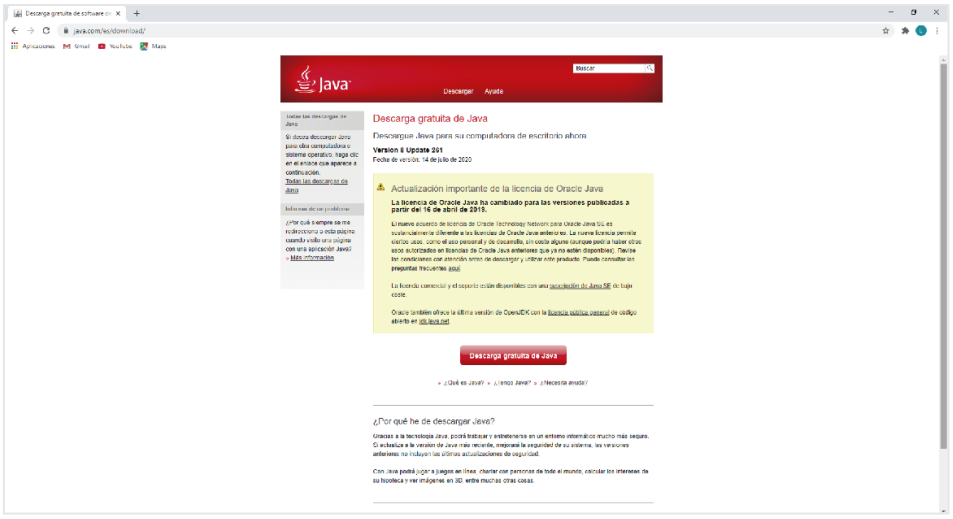


Figura 4.18. Descargar Java.

Una vez descargado e instalado Java descargaremos Android Studio que también podemos encontrar en su página oficial

https://developer.android.com/studio?hl=es y posteriormente lo instalaremos.

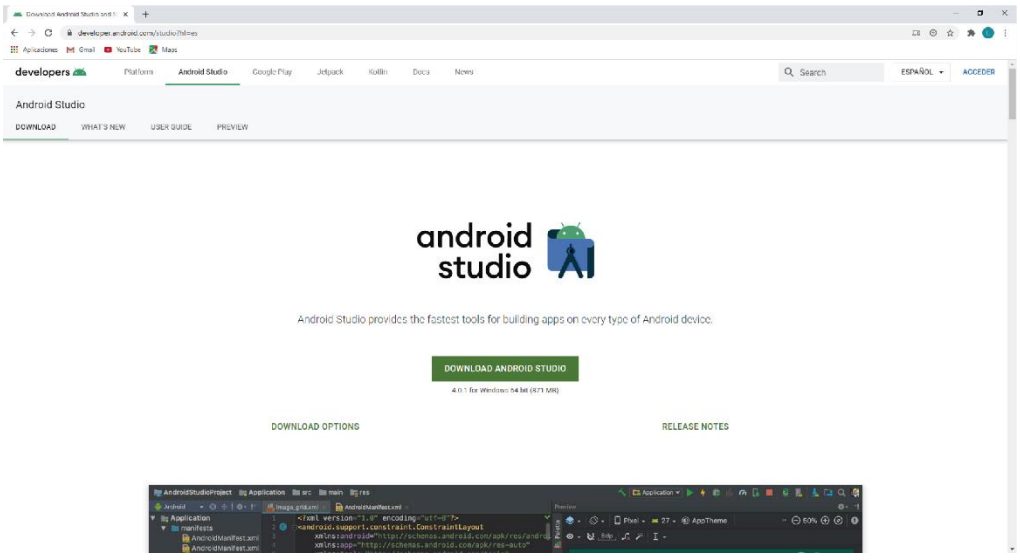


Figura 4.19. Descargar Android Studio.

Una vez obtenidos los programas podemos empezar a realizar los cambios o añadidos deseados.

Al abrir Android Studio nos aparecerá la siguiente pantalla y deberemos seleccionar “Open an existing Android Studio project”.

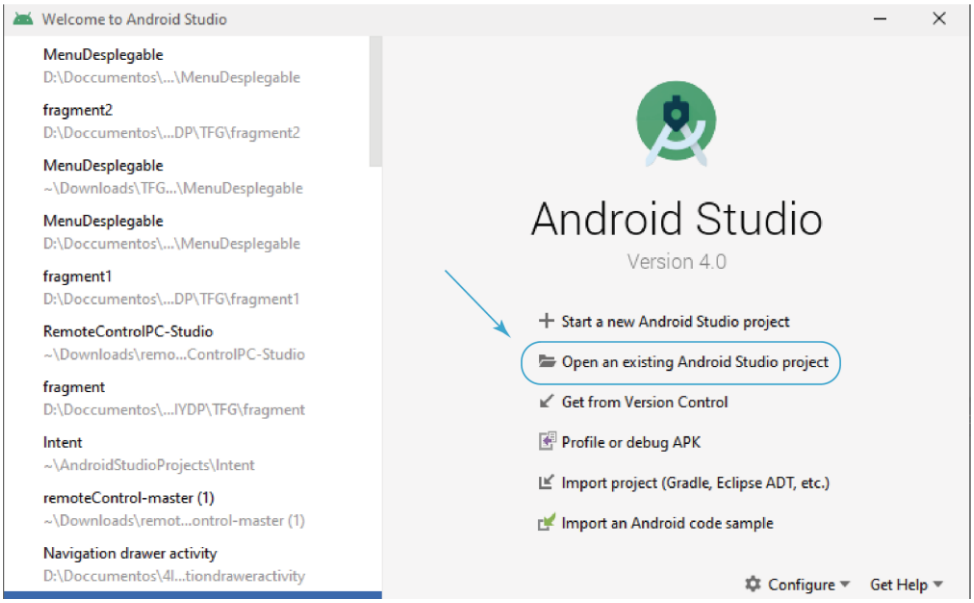


Figura 4.20. Abrir proyecto en Android Studio

Al abrir el proyecto encontraremos la interfaz de la aplicación con los componentes que ya expliqué anteriormente en el punto *2.3. Como crear un proyecto* y en el punto *2.7.1. Interfaz y pestañas*.

Deberemos desplegar el directorio *res*, dentro de este directorio encontraremos muchos subdirectorios y deberemos desplegar *navigation*. Dentro de esta carpeta encontraremos el archivo *mobile_navigation.xml,* el cual deberemos abrir haciendo doble clic porque es donde crearemos los teclados que queramos añadir.

4.3.2. Crear el Fragment del interfaz

Cuando abramos este archivo, encontraremos la pantalla que muestro a continuación, donde aparecerán los tres teclados que ya han sido creados. Para poder empezar a crear el nuevo teclado, tendremos que añadir un nuevo *fragment,* para hacer esto observamos que en la parte superior izquierda hay un icono de un rectángulo con un más en verde, como aparece en la figura 4.21., deberemos clicar sobre él y dar a “*create new destination”.*

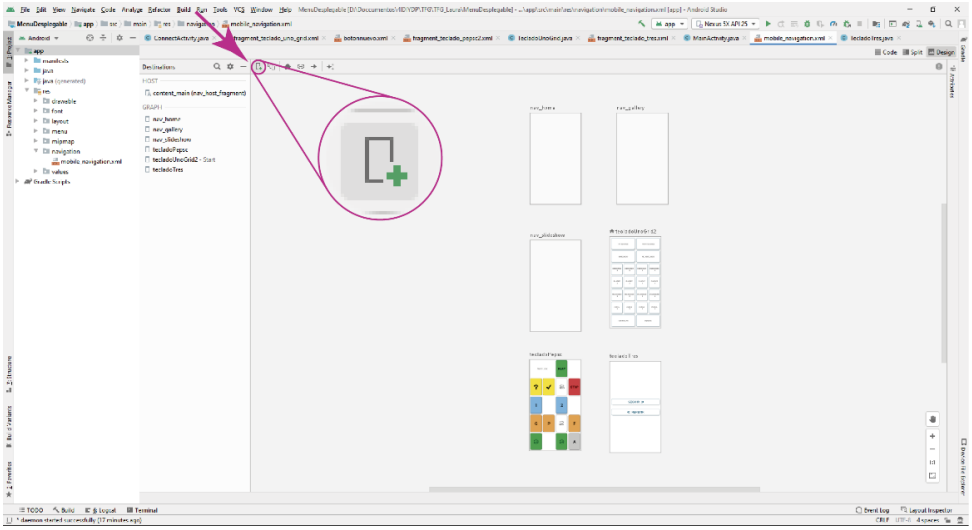


Figura 4.21. Crear un nuevo Fragment

Se desplegará una pantalla en la que deberemos elegir un *Fragment (Blank)* y dar a *next*. En la siguiente pantalla tendremos que asignarle un nombre y clicar en *finish.*

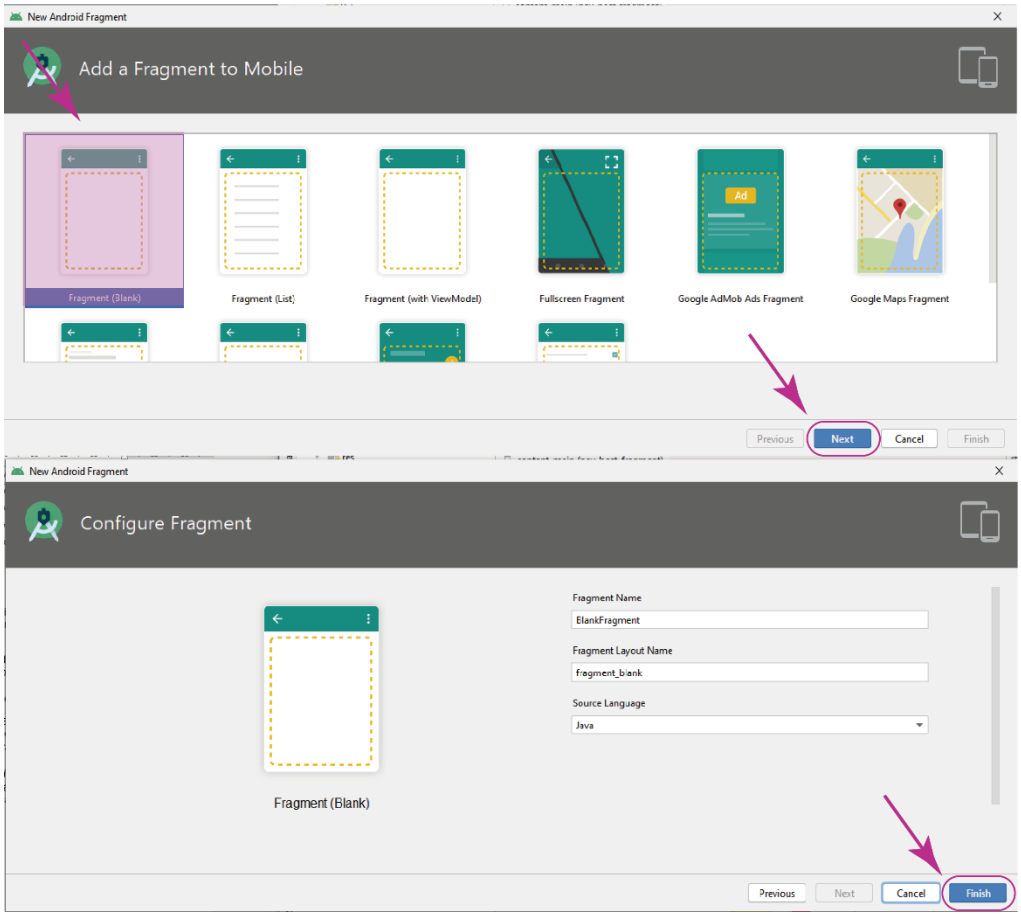


Figura 4.22. Crear nuevo Fragment

Cuando realicemos estos pasos, el nuevo *fragment* aparecerá junto con el resto de los teclados en el espacio de trabajo del *mobile_navigation*, y lo único que deberemos hacer es dar doble clic sobre él para acceder a su archivo *.xml* y poder configurarlo.

En cuanto al diseño de la interfaz, aunque puedas añadir algún TextView si queremos hacer alguna aclaración que pueda ver el usuario que utilice la aplicación, lo más importante es introducir botones, tanto *Button* como *ImageButton*.

Una vez tengamos realizada tu interfaz, pudiendo utilizar las instrucciones y las descripciones de los elementos realizados previamente en los puntos *2.6. Layouts, 2.7. Elementos gráficos* y *3.4. Componentes*, deberemos pasar a programar en el archivo *.java*, con el que mandaremos las instrucciones al ordenador para poder manejar el teclado desde el móvil. Es importante que todos los botones tengan asignado un *id* reconocible del que poderse acordar fácilmente ya que tendremos que declarar estos botones en la programación en *java*.

4.3.3. Enlazar el botón a teclas concretas

En el directorio *java/com.example.menudesplegable/connection* encontraremos un archivo *.java* llamado *ButtonEnum*, deberemos hacer doble clic ya que en este archivo aparecen declaradas todas las teclas con su respetivo código, de tal manera que para que un botón este enlazado con la tecla A, deberemos utilizar el objeto K_A.

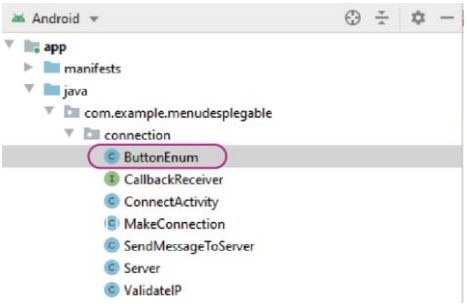


Figura 4.23. Archivo ButtonEnum.java

Para aprender a programar el enlace de un botón vamos a hacer un ejemplo muy sencillo de un teclado con un solo botón cuyo id será boton y que estará enlazado a la letra B.

Lo primero que deberemos hacer será abrir el archivo .java del nuevofragment que encontrarás en *java/com.example.menudesplegable* y abrirlo haciendo doble clic.

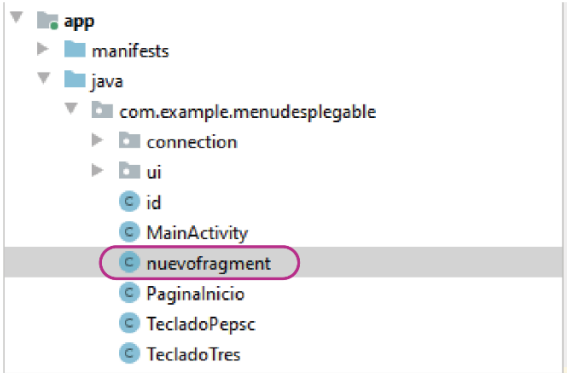


Figura 4.24.. Archivo nuevofragment.java

Una vez abierto el archivo .java, deberemos introducir los siguientes fragmentos de código. Lo primero que hay que introducir es el fragmento implements View.OnClickListener en el public class. A continuación realizaremos la declaración de variables, en este caso solo es una, pero si existieran más botones deberíamos declarar tantas variables como número de botones. Este código deberá ir después de las declaraciones private String mParam1 y private String mParam2.

```
private Button boton;
```

Nota: En caso de que el botón sea un ImageButton, deberemos poner eso y no Button

Justo después de esto introduciremos el siguiente código, que es el que designa las claves para cada tecla.

```
private HashMap<Integer,Integer> buttonMap=new
HashMap<Integer,Integer>();
```

Lo siguiente que tendremos que hacer será relacionar los variables que hemos declarado con el código de la tecla que queramos, en este caso la B.

Incluyo el código que ya aparece en el archivo java encuadrándolo en verde, el cuál deberemos sustituir por el que aparece recuadrado en azul.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_nuevo_fragment, container,
false);
}
```

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View rootView=inflater.inflate(R.layout.fragment_nuevo_fragment,
container, false);
    //return inflater.inflate(R.layout.fragment_teclado_tres, container,
false);
    initialization(rootView);
    return rootView;
}
```

En lugar de *fragment_nuevo_fragment,* deberemos poner el nombre del archivo xml del fragment que hemos creado.

A continuación de esto añadiremos el siguiente código.

```
private HashMap<Integer,Integer> getButtonMap(){
    HashMap<Integer,Integer> map=new HashMap<Integer,Integer>();
    map.put(R.id.boton, ButtonEnum.K_B);

   return map;
   }
   private void initialization (View rootView){
      boton =(Button) rootView.findViewById(R.id.boton);
      boton.setOnClickListener(this);
      buttonMap=getButtonMap();
   }
   private void buttonClick (int id){
      String action = "TYPE_KEY";
      int KeyCode=buttonMap.get(id);
      sendKeyCodeToServer(action,KeyCode);
   }
   private void sendKeyCodeToServer(String action, int keyCode){
      MainActivity.sendMessageToServer(action);
      MainActivity.sendMessageToServer(keyCode);
   }
   @Override
   public void onClick(View view) {
      int id = view.getId();
      buttonClick(id);
   }
```

Nota: si estas utilizando un ImageButton, sustituir en todas las partes del código donde aparezca la palabra Button por ImageButton.

Con este código, ya tendríamos programado el botón de nuestro teclado y sería completamente funcional.

4.3.4. Incluir el teclado en el menú desplegable

Lo siguiente que deberemos hace para que este teclado aparezca en nuestro menú desplegable será acceder a *activity_main_drawer.xml*, que encontraremos en el directorio *res/menu.*

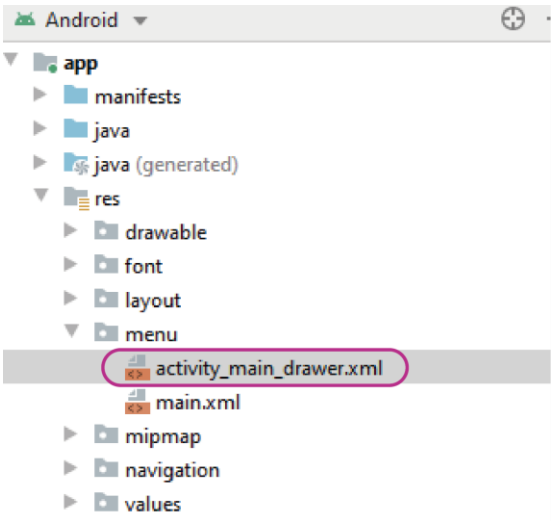


Figura 4.25. Activity_main_drawer

Cada elemento que aparece en el menu desplegable es un item en el *activity_main_drawer,* por lo que lo único que deberemos hacer será copiar la estructura de los anteriores cambiando el *id* por el de nuestro nuevo teclado y el icono si no queremos que sea un teclado.

```
<item
   android:id="@+id/nuevofragment"
   android:icon="@drawable/teclado"
   android:title="PEPS-C"
   android:iconTint="#004B6C"/>
```

A continuación deberemos entrar en el *MainActivity.java* el cual encontraremos en el directorio java y añadir el nombre de nuestro teclado en el siguiente código.

```
mAppBarConfiguration = new AppBarConfiguration.Builder(
      R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow,
```

R.id.*tecladoUnoGrid2*, R.id.*tecladoPepsc*, R.id.*tecladoTres,*
R.id.nuevofragment)
.setDrawerLayout(drawer)
.build();

4.4. Instalación

En este punto, el nuevo teclado ya debería funcionar y lo siguiente que debemos hacer es probarlo.

En el caso de que tengamos antivirus instalado en el ordenador, lo primero que deberemos hacer es desactivar el firewall, en caso contrario no funcionará.

En este punto tenemos dos posibilidades, que queramos instalar directamente la aplicación sin introducir cambios, o que queramos instalar la aplicación habiendo realizado algún cambio en Android Studio.

Lo primero que debemos hacer en cualquiera de los dos casos es activar en nuestro smartphone *"opciones de desarrollador"*, que conseguimos entrando en ajustes, acerca del teléfono, información de software y pulsando siete veces en *Número de compilación.*

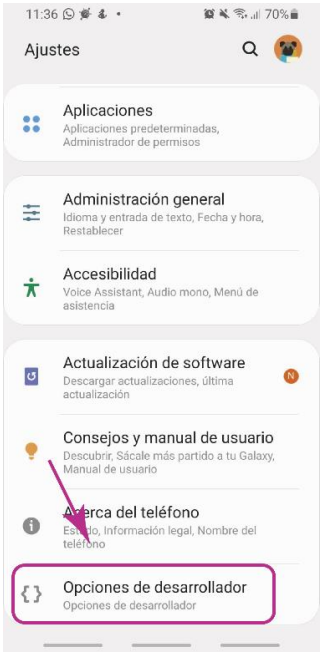


Figura 4.26. Opciones de desarrollador

Una vez esta opción este activada, conectaremos el móvil y el ordenador a la misma red, conectaremos el móvil al ordenador y ejecutaremos el programa como expliqué en el apartado *2.4. Emulador,* en caso de haber modificado el código en Android Studio, o directamente ejecutaremos la APK en el móvil si no hemos modificado nada.

Mientras la aplicación se ejecuta en el móvil, ejecutaremos en el ordenador el otro archivo de la carpeta, *RemoteControlPC*, y este nos dará una dirección IP y un puerto que deberemos introducir en la aplicación del móvil.

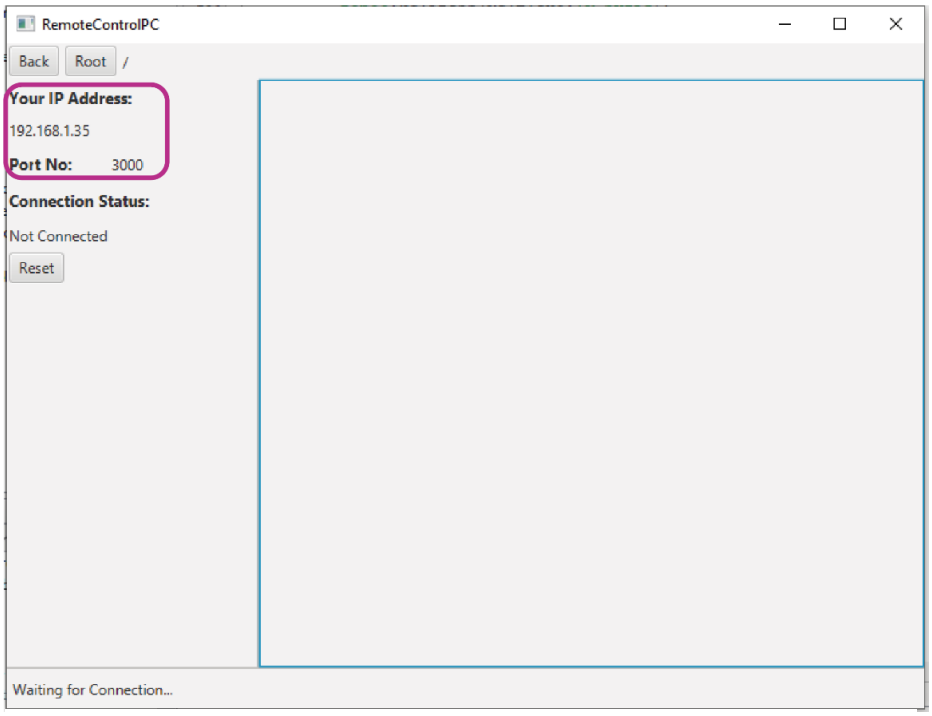


Figura 4.27. Dirección IP

En este momento el ordenador y el móvil ya estarán enlazados y para comprobarlo, podemos abrir el bloc de notas en el ordenador y probar a escribir desde el nuestro teclado.

5. CONCLUSIONES

En este proyecto nos habíamos planteado el objetivo de realizar una aplicación con la intención de utilizarla para hacer servicios tipo Mago de Oz para un videojuego infantil de ordenador, utilizando el dispositivo Android como control remoto del teclado físico del PC. Después de haber analizado una gran variedad de aplicaciones existentes en el mercado con similitudes a la nuestra y de haber introducir todas las mejoras que habíamos planteado en un primer momento podemos concluir que hemos alcanzado nuestro objetivo inicial con respecto a la creación y desarrollo de la aplicación.

El proyecto nos ha permitido introducirnos de lleno en el sistema Android y en su entorno de desarrollo para tener la base necesaria para poder empezar a desarrollar aplicaciones para este sistema operativo. Nos hemos documentado con respecto a todos los elementos y funciones que incluye su entorno de desarrollo, Android Studio, para tener una mayor facilidad a la hora de empezar a desarrollar la aplicación.

Se ha realizado el diseño de la interfaz, consiguiendo que sea intuitiva y fácil de utilizar, además de conseguir un aspecto atractivo. Hemos cumplido los objetivos respecto a los teclados que queríamos desarrollar, consiguiendo teclados prácticos para el servicio tipo Mago de Oz para videojuegos. Para conseguir esto se ha requerido un proceso de formación y mucho trabajo de ensayo error al no haber tenido anteriormente ningún contacto con estos lenguajes, pero siendo un proceso muy enriquecedor personalmente.

Además, la aplicación desarrollada incluye un manual del desarrollador muy completo con toda la información necesaria para que el usuario pueda desarrollar sus propios teclados, desde los requerimientos necesarios para poder programar en Android Studio con sus respectivos manuales de instalación, hasta el proceso de instalación y prueba de la aplicación en dispositivos Android, pasando por la implementación del código y la explicación de los posibles elementos gráficos que desee utilizar.

Personalmente ha sido una experiencia muy enriquecedora, por un lado he adquirido conocimientos en diferentes lenguajes de programación que pueden ser muy útiles en vistas al futuro laboral, por otro lado he conseguido realizar un buen trabajo de síntesis y asimilación de la información, muy necesario en todos los ámbitos de la vida y especialmente útiles para la continuación de mi formación.

6. BIBLIOGRAFÍA

[1] Activity. (2018). Android Developers. https://developer.android.com/reference/android/app/Activity

[2] Activity: creación y ciclo de vida. (2016, 30 marzo). Activity: creación y ciclo de vida. https://academiaandroid.com/activity-creacion-y-ciclo-de-vida/#:%7E:text=Como%20explicamos%20en%20la%20anterior,para%20interaccionar%20con%20la%20aplicaci%C3%B3n.

[3] Android. (2019, 22 agosto). *Say hello to Android's new brand identity*. YouTube. https://www.youtube.com/watch?v=-2w7RKAlKTs&feature=youtu.be

[4] Aspectos fundamentales de la aplicación | Desarrolladores de Android. (2019). Android Developers. https://developer.android.com/guide/components/fundamentals?hl=es-419

[5] Build a Responsive UI with ConstraintLayout |. (2017). Android Developers. https://developer.android.com/training/constraint-layout/

[6] Castillo, L. J. D. (2020). *ANDROID STUDIO. Aprende a desarrollar aplicaciones* (1.a ed.). Alfaomega.

[7] Cómo interpretar el ciclo de vida de una actividad. (2018). Android Developers. https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es

[8] Creando un FloatingActionButton en Android Vía @LibreDesarrollo. (2015, 24 agosto). DesarrolloLibre. https://www.desarrollolibre.net/blog/android/creando-un-floatingactionbutton-en-android-lib-de-soporte#.X0jBuMgzaUk

[9] Cristian Henao. (2020, 25 abril). *Como crear un NAVIGATION DRAWER en Android | 2020*. YouTube. https://www.youtube.com/watch?v=5BiQZ01-H9U

[10] ¿Cuántos dispositivos Android existen? (2020, 10 febrero). https://www.nts-solutions.com/. https://www.nts-solutions.com/blog/dispositivos-android.html

[11] Descripción general de los servicios | Desarrolladores de Android. (2019). Android Developers. https://developer.android.com/guide/components/services?hl=es-

419#:%7E:text=Un%20Service%20es%20un%20componente,usuario%20cambie%20a%20otra%20aplicaci%C3%B3n.

[12] Descripción general del manifiesto de una app. (2019). Android Developers. https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419

[13] Diseños Android básicos: TableLayout. (2012, 5 noviembre). La columna 80. https://columna80.wordpress.com/2012/11/05/diseos-android-bsicos-tablelayout/

[14] EditText - Aprendo programación. (2018). Aprendo Programación. https://sites.google.com/site/aprendoprogramacion/android/5-widgets/edittext

[15] El Baúl del Programador. (2016, 1 enero). *ProgramaciÃ<sup>3*n Android: Interfaz grÃ¡fica - Componentes grÃ¡ficos y Eventos</i>. El Baúl del programador. https://elbauldelprogramador.com/programacion-android-interfaz-grafica_25/

[16] Fernández, S. (2019, 16 abril). *Android supera el 90% de cuota en España mientras que iOS cae por debajo del 9%, según Kantar*. Xataka Móvil. https://www.xatakamovil.com/mercado/android-supera-90-cuota-espana-ios-cae-debajo-9-kantar

[17] Galisteo, A. (2017). *Android. Quitar las mayúsculas por defecto en un Button*. Android. Quitar las mayúsculas por defecto en un Button. https://www.galisteocantero.com/android-quitar-las-mayusculas-por-defecto-en-un-button/

[18] Girones, J. T. (2012). *EL GRAN LIBRO DE ANDROID* (2.a ed.). ALFAOMEGA.

[19] Introducción al diseño de interfaces gráficas en Android. (2013). Introducción al diseño de interfaces gráficas en Android. http://www.jtech.ua.es/dadm/restringido/android/sesion03-apuntes.pdf

[20] J. (2020, 4 julio). *CURSO de ANDROID #2 [CONOCE sus FUNDAMENTOS (2/2)] ► Arquitectura, Dispositivos y Curiosidades*. Webipedia HD - Tu Mundo Virtual. http://webipedia.es/tecnologia/cursos/introduccion-a-android-22/

[21] La Geekipedia De Ernesto. (2018, 23 febrero). *Curso Android desde cero #23 | Cómo pasar de una Activity a otra - Intent en Android*. YouTube. https://www.youtube.com/watch?v=VQcJRM6ZgHw&list=PLyvsggKtwbLX06iMtXnRGX5lyjiiMaT2y&index=24

[22] Leiva, A. (2016, 6 octubre). *Android Studio 2.2 lleva el desarrollo de Android a un nuevo nivel*. Genbeta. https://www.genbeta.com/desarrollo/android-studio-2-2-lleva-el-desarrollo-de-android-a-un-nuevo-nivel

[23] Lopez, E. (2018, 13 enero). *Ciclo de vida de un Activity en Android*. UnProgramador. https://unprogramador.com/ciclo-de-vida-de-un-activity-en-android/

[24] M. (2017, 11 septiembre). *Análisis de Android, el sistema operativo para móviles de Google*. Blog de tecnología e información. https://blog.ibertronica.es/tutoriales/android-sistema-operativo/

[25] Marca Android. (2019). Marca Android. https://www.creativosonline.org/blog/android-marca.html

[26] Máster en Desarrollo de Aplicaciones Android - Arquitectura de Android. (2020). Máster en desarrollo de aplicaciones Android. http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de%0candroid#:%7E:text=El%20n%C3%BAcleo%20de%20Android%20est%C3%A1,soporte%20de%20drivers%20para%20dispositivos.

[27] Máster en Desarrollo de Aplicaciones Android - Componentes de una aplicación. (2020). Máster en Desarrollo de Aplicaciones Android - Componentes de una aplicación. http://www.androidcurso.com/index.php/tutoriales-android-fundamentos/31-unidad-1-vision-general-y-entorno-de-desarrollo/149-componentes-de-una-aplicacion

[28] Métodos en Java. (2020). Métodos en java. http://puntocomnoesunlenguaje.blogspot.com/2012/04/metodos.html

[29] MovilZona. (2016, 3 marzo). *Curso de Desarrollo Android. Tema 3: Cómo crear y configurar un emulador Android con Android Studio*. https://www.movilzona.es/tutoriales/android/desarrollo/como-crear-y-configurar-un-emulador-android-con-android-studio/

[30] Otriz, R. (2019, 29 agosto). *Nuevo logo de Android con aspecto moderno y más accesible*. Nuevo logo de Android con aspecto moderno y más accesible. https://www.roc21.com/2019/08/23/nuevo-logo-android/

[31] PicÓN, A. (2019, 21 marzo). *Android — Explorando ConstraintLayout (1) - cornershop-tech*. Medium.

https://tech.cornershop.io/android-explorando-constraintlayout-1-4f39beef9f82

[32] Steiner | dafont.com. (2006). Steiner. https://www.dafont.com/es/steiner.font

[33] TextView - Aprendo programación. (2017). TextView - Aprendo programación. https://sites.google.com/site/aprendoprogramacion/android/5-widgets/textview

[34] Fragmentos | Desarrolladores de Android |. (2019). Android Developers. https://developer.android.com/guide/components/fragments?hl=es-419

[35] Epson iProjection - Apps en Google Play. (2019, 12 septiembre). Play Store. https://play.google.com/store/apps/details?id=com.epson.iprojection&hl=es_419

[36] Epson iProjection - Para proyecciones inalámbricas. (2018). Scotch scotch. https://www.proyector24.es/es/epson-application-iprojection

[37] Mando a distancia para la TV - Apps en Google Play. (2019, 18 noviembre). Play Store. https://play.google.com/store/apps/details?id=com.vsray.remote.control&hl=es_419

[38] ¿Qué puedo hacer con la aplicación LG TV PLUS? (2016, 25 noviembre). Experiencias LG. https://www.experienciaslg.com.pe/appsjuegos/que-puedo-hacer-con-la-aplicacion-lg-tv-plus/

[39] Castillero Mimenza, O. (2020, 6 septiembre). *¿Qué significa el color azul en Psicología?* Psicología y Mente. https://psicologiaymente.com/psicologia/que-significa-el-azul#:%7E:text=Otros%20conceptos%20que%20una%20gran,la%20inteligencia%20y%20al%20deporte.

[40] Cristian Henao. (2017, 23 junio). *65. Como Usar Fragments Dinamicos en Android*. YouTube. https://www.youtube.com/watch?v=OOKLRTa4fHw

[41] El Arte de la Programación. (2018, 24 junio). *Cómo crear un SPLASH SCREEN en Android Studio*. YouTube. https://www.youtube.com/watch?v=GM8sc0ZjopU

[42] Fragmentos | Desarrolladores de Android |. (2019). Android Developers. https://developer.android.com/guide/components/fragments?hl=es-419

[43] García, I. (2018, 13 marzo). *Crear un Splash Activity en Android*. Cable Naranja. https://www.cablenaranja.com/crear-un-splash-activity-en-android/

[44] J. (2018). *Mostrar pantalla splash en Android durante unos segundos al iniciar la aplicación | Jon Segador*. Jon Segador. http://jonsegador.com/2012/11/mostrar-pantalla-splash-android-durante-unos-segundos-iniciar-aplicacion/

[45] Valenzuela, V. (2015, 7 mayo). *Combinaciones básicas de color entre texto y color de fondo •*. Silo Creativo. https://www.silocreativo.com/combinaciones-basicas-de-color-entre-texto-y-color-de-fondo/

[46] Kumar, V. (2017). *varunon9/Remote-Control-PC*. GitHub. https://github.com/varunon9/Remote-Control-PC

[47] Ramírez, I. (2020, 31 enero). *Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas*. Xataka. https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas

[48] Turn iPhone, iPad and Android into wireless mobile mouse / trackpad / keyboard with Remote Mouse. (2020). Remote Mouse. https://www.remotemouse.net/

[49] Wafer, A. (2020, 26 agosto). *How to use Android devices as Windows 10 PC keyboard*. Windows Report | Error-free Tech Life. https://windowsreport.com/use-android-pc-keyboard-usb/

ANEXO: CÓDIGOS

1. Menú Desplegable

.

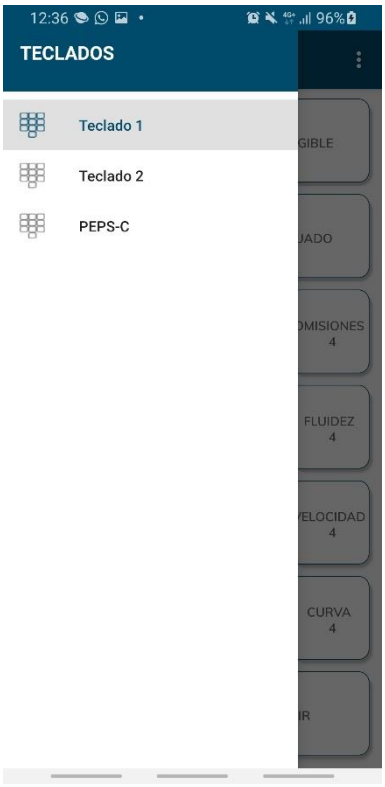


Figura 1 anexos. Menú desplegable

Con esta pantalla, lo que conseguimos es un menú desplegable en el que poder navegar por todos los teclados

- MainAcitivty.java

```
private AppBarConfiguration mAppBarConfiguration;
    public static Socket clientSocket = null;
    public static ObjectInputStream objectInputStream = null;
    public static ObjectOutputStream objectOutputStream = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
```

```
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow,
R.id.tecladoUnoGrid2, R.id.tecladoPepsc, R.id.tecladoTres)
                .setDrawerLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }

    protected void onDestroy() {
        super.onDestroy();
        try {
            if (MainActivity.clientSocket != null) {
                MainActivity.clientSocket.close();
            }
            if (MainActivity.objectOutputStream != null) {
                MainActivity.objectOutputStream.close();
            }
            if (MainActivity.objectInputStream != null) {
                MainActivity.objectInputStream.close();
            }
```

```
        } catch(Exception e) {
            e.printStackTrace();
        }
        Server.closeServer();
    }

    //this method is called from fragments to send message to server
(Desktop)
    public static void sendMessageToServer(String message) {
        if (MainActivity.clientSocket != null) {
            new SendMessageToServer().execute(String.valueOf(message),
"STRING");
            /*try {
                MainActivity.objectOutputStream.writeObject(message);
                MainActivity.objectOutputStream.flush();
            } catch (Exception e) {
                e.printStackTrace();
                socketException();
            }*/
        }
    }

    public static void sendMessageToServer(int message) {
        if (MainActivity.clientSocket != null) {
            new SendMessageToServer().execute(String.valueOf(message),
"INT");
            /*try {
                MainActivity.objectOutputStream.writeObject(message);
                MainActivity.objectOutputStream.flush();
            } catch (Exception e) {
                e.printStackTrace();
                socketException();
            }*/
        }
    }

    public static void socketException() {
        //Toast.makeText(thisActivity, "Connection Closed",
Toast.LENGTH_LONG).show();
        if (MainActivity.clientSocket != null) {
            try {
                MainActivity.clientSocket.close();
                MainActivity.objectOutputStream.close();
```

```
                MainActivity.clientSocket = null;
            } catch(Exception e2) {
                e2.printStackTrace();
            }
        }
    }

    public static void sendMessageToServer(float message) {
        if (MainActivity.clientSocket != null) {
            new SendMessageToServer().execute(String.valueOf(message),
"FLOAT");
            /*try {
                MainActivity.objectOutputStream.writeObject(message);
                MainActivity.objectOutputStream.flush();
            } catch (Exception e) {
                e.printStackTrace();
                socketException();
            }*/
        }

    public static void sendMessageToServer(long message) {
        if (MainActivity.clientSocket != null) {
            new SendMessageToServer().execute(String.valueOf(message),
"LONG");
            /*try {
                MainActivity.objectOutputStream.writeObject(message);
                MainActivity.objectOutputStream.flush();
            } catch (Exception e) {
                e.printStackTrace();
                socketException();
            }*/
        }
    }
}
```

- activity_main_drawer.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:showIn="navigation_view">

    <group android:checkableBehavior="single">
```

```
        <item
            android:id="@+id/tecladoUnoGrid2"
            android:icon="@drawable/teclado"
            android:title="Teclado 1"
            android:iconTint="#004B6C"/>
        <item
            android:id="@+id/tecladoTres"
            android:icon="@drawable/teclado"
            android:title="Teclado 2"
            android:iconTint="#004B6C"/>
        <item
            android:id="@+id/tecladoPepsc"
            android:icon="@drawable/teclado"
            android:title="PEPS-C"
            android:iconTint="#004B6C"/>

    </group>
</menu>
```

2. Pantalla de Inicio (Splash)

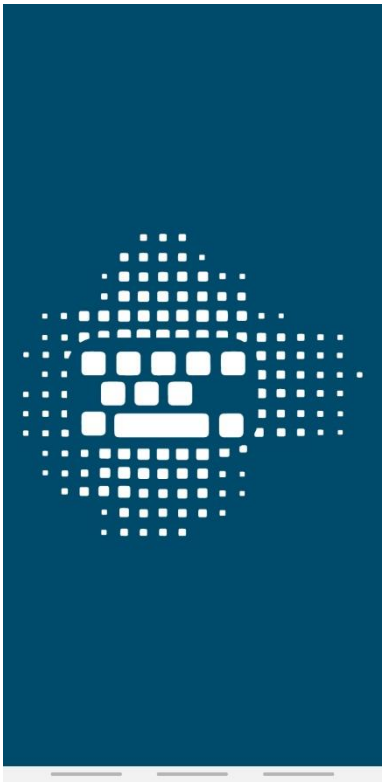


Figura 2 Anexos. Splash Activity

Con este activity conseguimos que la pantalla de inicio se muestre durante pocos segundos antes de pasar al siguiente activity.

- activity_pagina_inicio.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".PaginaInicio"
    android:background="#004B6C">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="323dp"
        android:layout_height="645dp"
        android:contentDescription="@string/imagen"
        android:tint="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/logo5" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

- PaginaInicio.java

```
public class PaginaInicio extends AppCompatActivity {

    // Duración en milisegundos que se mostrará el splash
    private final int DURACION_SPLASH = 1500; // 1,5 segundos

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCR
EEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_pagina_inicio);
        new Handler().postDelayed(new Runnable() {
            public void run() {
                Intent intent = new Intent (PaginaInicio.this,
ConnectActivity.class);
```

```
                startActivity(intent);
                finish();

            };
        }, DURACION_SPLASH);
    }
}
```

3. Página conexión

Figuras 3 Anexo. Página conexión

Con este activity lo que conseguimos es enlazar el móvil al ordenador al introducir la dirección IP.

- fragment_connect.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="30dp"
    android:orientation="vertical"
    android:background="#FFFFFF">
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Introduzca su dirección IP"
        android:fontFamily="@font/mulishsemibold"
        android:textSize="18sp"
        android:textColor="#656464"/>

    <EditText
        android:id="@+id/ipAddress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:layout_marginTop="15dp"
        android:hint="Dirección IP" >

        <requestFocus />
    </EditText>

    <EditText
        android:id="@+id/portNumber"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:layout_marginTop="15dp"
        android:hint="Número de puerto"
        android:inputType="number" />

    <Button
        android:id="@+id/connectButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dp"
        android:text="Conectar"
        android:background="@drawable/sombratres"
        android:textColor="#656464"
        android:fontFamily="@font/mulishsemibold"
        android:textSize="14sp"/>

</LinearLayout>
```

4. Teclado Piedra Mágica

Figura 4 Anexo. Teclado Piedra Mágica

- Fragment_teclado_uno_grid.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".TecladoUnoGrid"
    android:background="#ffffff">

    <GridLayout
        android:useDefaultMargins="true"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:columnCount="4"
        android:rowCount="7"
        android:padding="5dp">
        <Button
            android:id="@+id/intButton"
            android:layout_width="0dp"
```

```
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnSpan="2"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/inteligible"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"
    />

<Button
    android:id="@+id/nointButton"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnSpan="2"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/no_inteligible"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/adeButton"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnSpan="2"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/adecuado"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/noadeButton"
    android:layout_width="0dp"
```

```
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnSpan="2"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/no_adecuado"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/o1Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/omisiones_1"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/o2Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/omisiones_2"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/o3Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
```

```
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/omisiones_3"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/o4Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/omisiones_4"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/f1Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/fluidez_1"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/f2Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/fluidez_2"
    android:textSize="12sp"
```

```
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/f3Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/fluidez_3"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/f4Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/fluidez_4"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/v1Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/velocidad_1"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
```

```
    android:id="@+id/v2Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/velocidad_2"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/v3Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/velocidad_3"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/v4Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/velocidad_4"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>


<Button
    android:id="@+id/c1Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
```

```
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/curva_1"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/c2Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/curva_2"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/c3Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
    android:text="@string/curva_3"
    android:textSize="12sp"
    android:textColor="#656464"
    android:fontFamily="@font/mulishsemibold"/>

<Button
    android:id="@+id/c4Button"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombratres"
```

```
        android:text="@string/curva_4"
        android:textSize="12sp"
        android:textColor="#656464"
        android:fontFamily="@font/mulishsemibold"/>

    <Button
        android:id="@+id/contButton"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_rowWeight="1"
        android:layout_columnSpan="2"
        android:layout_columnWeight="1"
        android:layout_gravity="fill"
        android:background="@drawable/sombratres"
        android:text="@string/continuar"
        android:textSize="12sp"
        android:textColor="#656464"
        android:fontFamily="@font/mulishsemibold"/>

    <Button
        android:id="@+id/repButton"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_rowWeight="1"
        android:layout_columnSpan="2"
        android:layout_columnWeight="1"
        android:layout_gravity="fill"
        android:background="@drawable/sombratres"
        android:text="@string/repetir"
        android:textSize="12sp"
        android:textColor="#656464"
        android:fontFamily="@font/mulishsemibold"/>

  </GridLayout>
</FrameLayout>
```

- TecladoUnoGrid.java

```
public class TecladoUnoGrid extends Fragment implements
View.OnClickListener {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
```

```
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    private Button intButton;
    private Button nointButton;
    private Button adeButton;
    private Button noadeButton;
    private Button o1Button;
    private Button o2Button;
    private Button o3Button;
    private Button o4Button;
    private Button f1Button;
    private Button f2Button;
    private Button f3Button;
    private Button f4Button;
    private Button v1Button;
    private Button v2Button;
    private Button v3Button;
    private Button v4Button;
    private Button c1Button;
    private Button c2Button;
    private Button c3Button;
    private Button c4Button;
    private Button contButton;
    private Button repButton;

    private HashMap<Integer,Integer> buttonMap=new
HashMap<Integer,Integer>();

    public TecladoUnoGrid() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment TecladoUnoGrid.
     */
    // TODO: Rename and change types and number of parameters
    public static TecladoUnoGrid newInstance(String param1, String
param2) {
        TecladoUnoGrid fragment = new TecladoUnoGrid();
        Bundle args = new Bundle();
```

```
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View
rootView=inflater.inflate(R.layout.fragment_teclado_uno_grid,
container, false);
        //return inflater.inflate(R.layout.fragment_teclado_uno_grid,
container, false);
        initialization(rootView);

        return rootView;
    }

    private HashMap<Integer,Integer> getButtonsMap(){
        HashMap<Integer,Integer> map=new
HashMap<Integer,Integer>();
        map.put(R.id.intButton, ButtonEnum.N_1);
        map.put(R.id.nointButton,ButtonEnum.N_2);
        map.put(R.id.adeButton,ButtonEnum.N_3);
        map.put(R.id.noadeButton,ButtonEnum.N_4);
        map.put(R.id.o1Button,ButtonEnum.K_A);
        map.put(R.id.o2Button,ButtonEnum.K_B);
        map.put(R.id.o3Button,ButtonEnum.K_C);
        map.put(R.id.o4Button,ButtonEnum.K_D);
        map.put(R.id.f1Button,ButtonEnum.K_E);
        map.put(R.id.f2Button,ButtonEnum.K_F);
        map.put(R.id.f3Button,ButtonEnum.K_G);
        map.put(R.id.f4Button,ButtonEnum.K_H);
        map.put(R.id.v1Button,ButtonEnum.K_I);
        map.put(R.id.v2Button,ButtonEnum.K_J);
        map.put(R.id.v3Button,ButtonEnum.K_K);
        map.put(R.id.v4Button,ButtonEnum.K_L);
```

```
        map.put(R.id.c1Button,ButtonEnum.K_M);
        map.put(R.id.c2Button,ButtonEnum.K_N);
        map.put(R.id.c3Button,ButtonEnum.K_O);
        map.put(R.id.c4Button,ButtonEnum.K_P);
        map.put(R.id.contButton,ButtonEnum.A_RIGHT);
        map.put(R.id.repButton,ButtonEnum.A_LEFT);

        return map;
    }

    private void initialization(View rootView){
        intButton = (Button) rootView.findViewById(R.id.intButton);
        nointButton=(Button) rootView.findViewById(R.id.nointButton);
        adeButton=(Button) rootView.findViewById(R.id.adeButton);
        noadeButton=(Button) rootView.findViewById(R.id.noadeButton);
        o1Button=(Button) rootView.findViewById(R.id.o1Button);
        o2Button=(Button) rootView.findViewById(R.id.o2Button);
        o3Button=(Button) rootView.findViewById(R.id.o3Button);
        o4Button=(Button) rootView.findViewById(R.id.o4Button);
        f1Button=(Button) rootView.findViewById(R.id.f1Button);
        f2Button=(Button) rootView.findViewById(R.id.f2Button);
        f3Button=(Button) rootView.findViewById(R.id.f3Button);
        f4Button=(Button) rootView.findViewById(R.id.f4Button);
        v1Button=(Button) rootView.findViewById(R.id.v1Button);
        v2Button=(Button) rootView.findViewById(R.id.v2Button);
        v3Button=(Button) rootView.findViewById(R.id.v3Button);
        v4Button=(Button) rootView.findViewById(R.id.v4Button);
        c1Button=(Button) rootView.findViewById(R.id.c1Button);
        c2Button=(Button) rootView.findViewById(R.id.c2Button);
        c3Button=(Button) rootView.findViewById(R.id.c3Button);
        c4Button=(Button) rootView.findViewById(R.id.c4Button);
        contButton=(Button) rootView.findViewById(R.id.contButton);
        repButton=(Button) rootView.findViewById(R.id.repButton);

        intButton.setOnClickListener(this);
        nointButton.setOnClickListener(this);
        adeButton.setOnClickListener(this);
        noadeButton.setOnClickListener(this);
        o1Button.setOnClickListener(this);
        o2Button.setOnClickListener(this);
        o3Button.setOnClickListener(this);
        o4Button.setOnClickListener(this);
        f1Button.setOnClickListener(this);
        f2Button.setOnClickListener(this);
        f3Button.setOnClickListener(this);
        f4Button.setOnClickListener(this);
        v1Button.setOnClickListener(this);
        v2Button.setOnClickListener(this);
        v3Button.setOnClickListener(this);
```

```
        v4Button.setOnClickListener(this);
        c1Button.setOnClickListener(this);
        c2Button.setOnClickListener(this);
        c3Button.setOnClickListener(this);
        c4Button.setOnClickListener(this);
        contButton.setOnClickListener(this);
        repButton.setOnClickListener(this);

        buttonMap=getButtonsMap();
    }

    private void buttonClick(int id){
        String action = "TYPE_KEY";
        int keyCode=buttonMap.get(id);
        sendKeyCodeToServer(action,keyCode);
    }

    private void sendKeyCodeToServer(String action, int keyCode) {
        MainActivity.sendMessageToServer(action);
        MainActivity.sendMessageToServer(keyCode);
    }

    @Override
    public void onClick(View view) {
        int id = view.getId();
        buttonClick(id);
    }
}
```

5. Teclado Presentaciones

Figura 5 Anexo. Teclado Presentaciones

- Fragment_teclado_tres.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TecladoTres"
    android:background="#FFFFFF">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="10dp"
        android:gravity="center">

        <ImageButton
            android:id="@+id/seguir"
            android:layout_width="match_parent"
            android:layout_height="50dp"
```

```
            android:layout_marginBottom="32dp"
            android:background="@drawable/sombratres"
            android:src="@drawable/seguir6"
            android:text="button"
            android:tint="#004B6C" />

        <ImageButton
            android:id="@+id/repetir"
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:background="@drawable/sombratres"
            android:src="@drawable/repetir3"
            android:text="button"
            android:tint="#004B6C" />
    </LinearLayout>
</FrameLayout>
```

- TecladoTres.java

```
public class TecladoTres extends Fragment implements
View.OnClickListener {
    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    private ImageButton seguir;
    private  ImageButton repetir;

    private HashMap<Integer,Integer> buttonMap=new
HashMap<Integer,Integer>();

    public TecladoTres() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
```

```
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment TecladoTres.
     */
    // TODO: Rename and change types and number of parameters
    public static TecladoTres newInstance(String param1, String param2) {
        TecladoTres fragment = new TecladoTres();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View rootView=inflater.inflate(R.layout.fragment_teclado_tres,
container, false);
        //return inflater.inflate(R.layout.fragment_teclado_tres, container,
false);
        initialization(rootView);
        return rootView;
    }

    private HashMap<Integer,Integer> getButtonMap(){
        HashMap<Integer,Integer> map=new HashMap<Integer,Integer>();
        map.put(R.id.seguir, ButtonEnum.A_RIGHT);
        map.put(R.id.repetir, ButtonEnum.A_LEFT);
        return map;
    }
    private void initialization (View rootView){
        seguir =(ImageButton) rootView.findViewById(R.id.seguir);
        repetir=(ImageButton) rootView.findViewById(R.id.repetir);
```

```
        seguir.setOnClickListener(this);
        repetir.setOnClickListener(this);
        buttonMap=getButtonMap();
    }
    private void buttonClick (int id){
        String action = "TYPE_KEY";
        int KeyCode=buttonMap.get(id);
        sendKeyCodeToServer(action,KeyCode);
    }

    private void sendKeyCodeToServer(String action, int keyCode){
        MainActivity.sendMessageToServer(action);
        MainActivity.sendMessageToServer(keyCode);
    }

    @Override
    public void onClick(View view) {
        int id = view.getId();
        buttonClick(id);
    }
}
```

6. Teclado PEPS-C

Figura 6 Anexo. Teclado PEPS-C

- Fragment_teclado_pepsc2.xml

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TecladoPepsc"
    android:background="#FFFFFF">

    <GridLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FFFFFF"
        android:columnCount="4"
        android:rowCount="5"
        android:useDefaultMargins="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/button1"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_rowWeight="1"
            android:layout_columnSpan="2"
            android:layout_columnWeight="1"
            android:layout_gravity="fill"
            android:background="#FFFFFF"
            android:gravity="center_horizontal|center_vertical"
            android:text="@string/num_lock"
            android:textColor="#656464"
            android:textSize="18sp"
            tools:targetApi="lollipop" />

        <Button
            android:id="@+id/button2"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_rowWeight="1"
```

```
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombraverde"
    android:text="@string/play"
    android:textSize="24sp"
    android:textStyle="bold"
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button3"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="#000000"
    android:text=""
    android:visibility="invisible"
    tools:targetApi="lollipop" />

<ImageButton
    android:id="@+id/button4"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombraamarilla"
    android:contentDescription="@string/interrogaci_n"
    android:padding="2dp"
    android:src="@drawable/interrograciondef"
    android:text="@string/button"
    android:tint="#000000"
    tools:targetApi="lollipop" />

<ImageButton
    android:id="@+id/button5"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombraamarilla"
```

```
    android:contentDescription="@string/check"
    android:padding="2dp"
    android:src="@drawable/checkdef"
    android:text="@string/button"
    tools:targetApi="lollipop" />

<ImageButton
    android:id="@+id/button6"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombrablanca"
    android:contentDescription="@string/redo"
    android:padding="2dp"
    android:src="@drawable/redodef"
    android:text="@string/button"
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button7"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombraroja"
    android:text="@string/stop"
    android:textSize="24sp"
    android:textStyle="bold"
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button8"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombraazul"
    android:text="@string/_1"
    android:textSize="36sp"
```

```
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button9"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="#020202"
    android:text="@string/button"
    android:visibility="invisible"
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button10"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombraazul"
    android:text="@string/_2"
    android:textSize="36sp"
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button11"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="#000000"
    android:text="@string/button"
    android:visibility="invisible"
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button12"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
```

```
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombranaranja"
    android:text="@string/g"
    android:textSize="36sp"
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button13"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombranaranja"
    android:text="@string/p"
    android:textSize="36sp"
    tools:targetApi="lollipop" />

<ImageButton
    android:id="@+id/button14"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombrablanca"
    android:contentDescription="@string/nulo"
    android:padding="2dp"
    android:src="@drawable/nulldef"
    android:text="@string/button"
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button15"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombranaranja"
    android:text="@string/f"
    android:textSize="36sp"
```

```
    tools:targetApi="lollipop" />

<ImageButton
    android:id="@+id/button16"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombraverde"
    android:contentDescription="@string/sonriente"
    android:padding="2dp"
    android:src="@drawable/sonrientedef"
    android:text="@string/button"
    tools:targetApi="lollipop" />

<Button
    android:id="@+id/button17"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="#000000"
    android:text="@string/button"
    android:visibility="invisible"
    tools:targetApi="lollipop" />

<ImageButton
    android:id="@+id/button18"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_rowWeight="1"
    android:layout_columnWeight="1"
    android:layout_gravity="fill"
    android:background="@drawable/sombraverde"
    android:contentDescription="@string/triste"
    android:padding="2dp"
    android:src="@drawable/tristedef"
    android:text="@string/button"
    tools:targetApi="lollipop" />

<Button
```

```
        android:id="@+id/button19"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:layout_gravity="fill"
        android:background="@drawable/sombra"
        android:text="@string/a"
        android:textSize="36sp"
        tools:targetApi="lollipop" />
    </GridLayout>
</FrameLayout>
```

- TecladoPepsc.java

```
public class TecladoPepsc extends Fragment implements
View.OnClickListener{

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    private Button PLB;
    private ImageButton IIB;
    private ImageButton OIB;
    private ImageButton RIB;
    private Button SB;
    private Button UB;
    private Button DB;
    private Button GB;
    private Button PB;
    private ImageButton NIB;
    private Button FB;
    private ImageButton SIB;
    private ImageButton TIB;
    private Button AB;

    private HashMap<Integer,Integer> buttonMap=new
HashMap<Integer, Integer>();

    public TecladoPepsc() {
        // Required empty public constructor
    }
```

```
    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment TecladoPepsc.
     */
    // TODO: Rename and change types and number of parameters

    public static TecladoPepsc newInstance(String param1, String
param2) {
        TecladoPepsc fragment = new TecladoPepsc();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View rootView=inflater.inflate(R.layout.fragment_teclado_pepsc2,
container, false);
        //return inflater.inflate(R.layout.fragment_teclado_uno_grid,
container, false);
        initialization(rootView);
        return rootView;
    }

    private HashMap<Integer,Integer> getButtonsMap(){
        HashMap<Integer,Integer> map=new
HashMap<Integer,Integer>();
        map.put(R.id.PLB, ButtonEnum.K_A);
        map.put(R.id.IIB,ButtonEnum.K_B);
        map.put(R.id.OIB,ButtonEnum.K_C);
        map.put(R.id.RIB,ButtonEnum.A_LEFT);
```

```
        map.put(R.id.SB,ButtonEnum.K_D);
        map.put(R.id.UB,ButtonEnum.N_1);
        map.put(R.id.DB,ButtonEnum.N_2);
        map.put(R.id.GB,ButtonEnum.K_G);
        map.put(R.id.PB,ButtonEnum.K_P);
        map.put(R.id.NIB,ButtonEnum.A_RIGHT);
        map.put(R.id.FB,ButtonEnum.K_F);
        map.put(R.id.SIB,ButtonEnum.N_3);
        map.put(R.id.TIB,ButtonEnum.N_4);
        map.put(R.id.AB,ButtonEnum.N_5);

        return map;
    }
    private void initialization(View rootView){
        PLB = (Button) rootView.findViewById(R.id.PLB);
        IIB=(ImageButton) rootView.findViewById(R.id.IIB);
        OIB=(ImageButton) rootView.findViewById(R.id.OIB);
        RIB=(ImageButton) rootView.findViewById(R.id.RIB);
        SB=(Button) rootView.findViewById(R.id.SB);
        UB=(Button) rootView.findViewById(R.id.UB);
        DB=(Button) rootView.findViewById(R.id.DB);
        GB=(Button) rootView.findViewById(R.id.GB);
        PB=(Button) rootView.findViewById(R.id.PB);
        NIB=(ImageButton) rootView.findViewById(R.id.NIB);
        FB=(Button) rootView.findViewById(R.id.FB);
        SIB=(ImageButton) rootView.findViewById(R.id.SIB);
        TIB=(ImageButton) rootView.findViewById(R.id.TIB);
        AB=(Button) rootView.findViewById(R.id.AB);


        PLB.setOnClickListener(this);
        IIB.setOnClickListener(this);
        OIB.setOnClickListener(this);
        RIB.setOnClickListener(this);
        SB.setOnClickListener(this);
        UB.setOnClickListener(this);
        DB.setOnClickListener(this);
        GB.setOnClickListener(this);
        PB.setOnClickListener(this);
        NIB.setOnClickListener(this);
        FB.setOnClickListener(this);
        SIB.setOnClickListener(this);
        TIB.setOnClickListener(this);
        AB.setOnClickListener(this);


        buttonMap=getButtonsMap();
    }
```

```
    private void buttonClick(int id){
        String action = "TYPE_KEY";
        int keyCode=buttonMap.get(id);
        sendKeyCodeToServer(action,keyCode);
    }
    private void sendKeyCodeToServer(String action, int keyCode) {
        MainActivity.sendMessageToServer(action);
        MainActivity.sendMessageToServer(keyCode);
    }

    @Override
    public void onClick(View view) {
        int id = view.getId();
        buttonClick(id);
    }
}
```