

Tarea 7 para Programación Multimedia y Dispositivos Móviles



Diego Manuel Carrasco Castañares

Índice:

Creación del proyecto 7.1.....	06
Composición Activity_Main.....	06
Modificación tema en Android_Manifest.....	07
Modificación archivo Colors.....	08
Modificación archivo Strings.....	08
Composición Main_Activity.....	08
Contenido archivo numSecreto (archivo SharedPreferences).....	11
Contenido archivo Score (archivo .txt).....	12
Ubicación archivos Score y numSecreto en el proyecto.....	12
Acceso adb_shell.....	12
Creación del proyecto 7.2.....	13
Creación del menú.....	13
Creación de Activities.....	14
Contenido de los layout.....	16
Contenido Colors y String.....	21
Contenido archivos java.....	24

Detalles de la tarea de esta unidad.

Enunciado.

Tarea UT07.1. Adivina número con SharedPreferences y utiliza fichero para guardar el número. (4 puntos)

Crear una aplicación que permita adivinar un número generado aleatoriamente entre 1 y 25. A través del uso de la clase SharedPreferences, donde se irá acumulando el número de aciertos conseguidos por el usuario a lo largo de las distintas ejecuciones de la aplicación. Inicialmente se tendrá una puntuación de 0, pero conforme ejecutemos varias veces y vayamos adivinando, dicho contador de puntuación se irá incrementando.

Para ayudar al jugador se irá indicando si el número introducido es mayor o menor que el número que se pretende acertar generado aleatoriamente.

Además, debes saber acceder a la ubicación de los datos a nivel de fichero en las SharedPreferences mediante interfaz de comando adb shell y mediante el Device File Explorer.

El aspecto inicial de la aplicación será el mostrado en la captura de pantalla.

También se escribirá en un fichero el número que el usuario tiene que adivinar.

Y cuando se adivine el número se mostrará en la pantalla la información de fichero para comprobar que efectivamente ese era el número correcto.

Tarea UT07.2. Agenda de contacto con base de datos SQLite. (6 puntos)

Crear una aplicación que simule una agenda de contactos. La pantalla principal tendrá un aspecto como el de la imagen y en el título tendrá que poner obligatoriamente "Agenda de contacto de nombrealumn@":

La primera operación que hay que realizar es la de insertar el contacto en la base de datos SQLite. Para ello cuando se pulsa en el menú el símbolo + aparecerá la siguiente pantalla. Si se da en el botón cancelar volverá a la pantalla de inicio anterior y si se da a crear se mostrará un toast indicando si se ha insertado correctamente o ha habido algún error en la inserción. Hay que comprobar que todos los campos estén con datos para poder insertar en la BD. Cuando se inserte el contacto se volverá a la pantalla de inicio (pantalla anterior).

La segunda operación es la consulta. Cada vez que se produzca una inserción en la agenda se mostrará en la pantalla de inicio un ListView con todos los contactos de la agenda como se muestra en la siguiente imagen:

La tercera operación es la modificación. Cuando se seleccione un contacto de la agenda haciendo click en la ListView nos llevará a una nueva actividad donde me permita modificar los datos del contacto. Una vez modificado, se debe actualizar la lista de la pantalla principal. El botón cancelar no hará nada y volverá a la pantalla inicial y el botón modificar guardará los cambios en la BD y actualizará la lista de contactos de la pantalla inicial.

La cuarta operación es el borrado. Al hacer una pulsación larga sobre la lista, se borra el contacto seleccionado. El listado deberá actualizarse tras el borrado.

Criterios de calificación. Total 10 puntos.

La valoración total de la tarea es de 10 puntos repartidos del siguiente modo:

Tarea UT07.1 (4 puntos)

- Aspecto gráfico de la interfaz: 0,5 puntos.
- Funcionamiento de la app:
 - La aplicación va mostrando al usuario si el número introducido es mayor menor: 0,75 puntos.
 - La aplicación mantiene la información de los puntos obtenidos tras reiniciarse: 0,75 puntos.
- Muestras la ubicación y contenido de los archivos con la información con adb shell: 0,5 puntos.
- Muestras la ubicación y contenido de los archivos con la información con Device File Manager: 0,5 puntos.
- Operaciones sobre los ficheros: 1 punto.

Tarea UT07.2 (6 puntos)

- Aspecto gráfico de la interfaz: 0,5 puntos.
- Programación:
 - Creación de la base de datos y la tabla de contactos: 1 punto
 - Operación de inserción: 1,25 puntos.
 - Operación de consulta: 1 punto.
 - Operación de modificación: 1,25 puntos.
 - Operación de borrado: 1 punto.

Puesto que no existe una evaluación por unidades la evaluación se realiza en base a los criterios generales del módulo.

Recursos necesarios para realizar la Tarea.

- ?
- Ordenador con Android Studio instalado y suficientes recursos para ejecutar el emulador.
- ?
- Contenidos de la Unidad, muy importantes los ejercicios resueltos.
- ?
- Páginas web de los desarrolladores de los sistemas operativos para móviles.

Consejos y recomendaciones.

- ?
- Lee atentamente el enunciado y asegúrate de haber entendido lo que has de hacer.
- ?
- Intenta reproducir en tu Android Studio los ejercicios resueltos previamente.
- ?
- No dudes en comentarle a tu tutor o tutora cualquier duda que te pueda surgir.
- ?
- Envíasela a tu tutor o tutora a través del sistema establecido en la plataforma.
- ?
- Las capturas y el contenido de los ficheros deben aparecer en perfecto orden para que esté claro lo que deseas mostrar.
- ?
- Los diseños deben mostrarse lo más parecidos posibles a lo que se pide.
- ?
- Debe llevarse a cabo sobre una versión actual de Android.

Indicaciones de entrega.

Una vez realizada la tarea elaborarás un documento PDF.

- ?
- El documento pdf mostrará capturas de la app:
 - Captura del aspecto de tu aplicación al arrancar la primera vez.
 - Captura del aspecto de tu aplicación en juego con número mayor, menor y acierto.
 - Captura con arranque las siguientes ocasiones.
 - Captura que muestre el contenido del archivo de preferencias desde el Device File Explorer y adb shell.
- ?
- Añade el contenido de los ficheros editados: java, xml, archivo de manifiesto, etc.
- ?
- Las capturas que no sean de tu aplicación, sino de tu Android Studio, deben mostrar el fondo de pantalla y la fecha y hora de la barra de estado de tu sistema operativo para garantizar la originalidad del material.

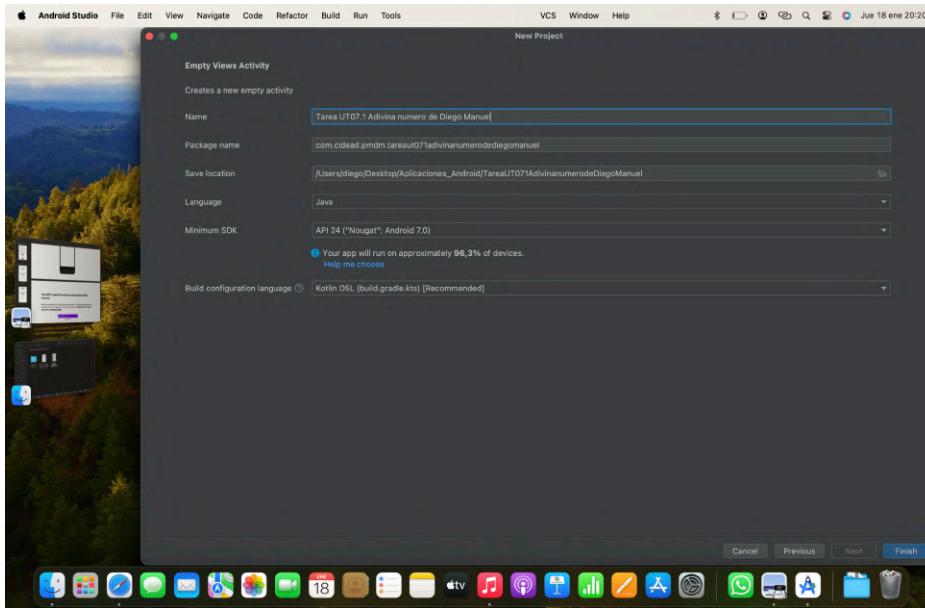
También se deberá entregar la carpeta comprimida del proyecto en formato .zip (File → Export → Export to Zip file...)

Entrega los archivos a través de la plataforma. Se nombrarán siguiendo las siguientes pautas:

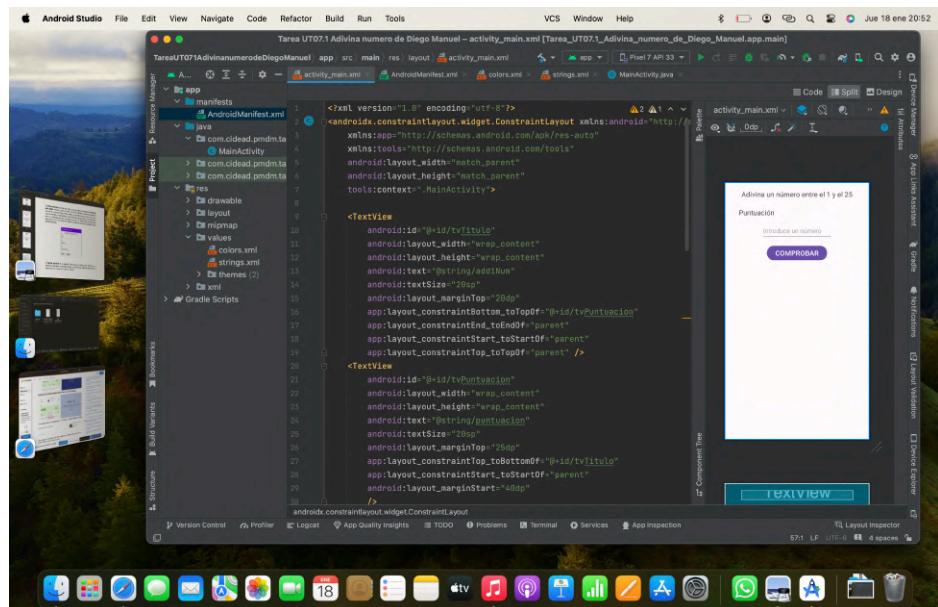
apellido1_apellido2_nombre_PMDM07_Tarea

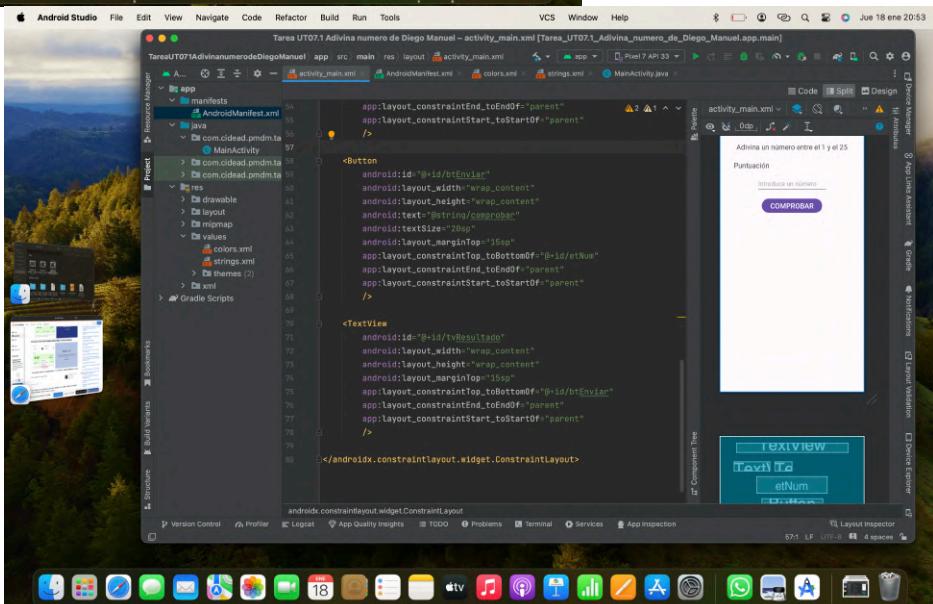
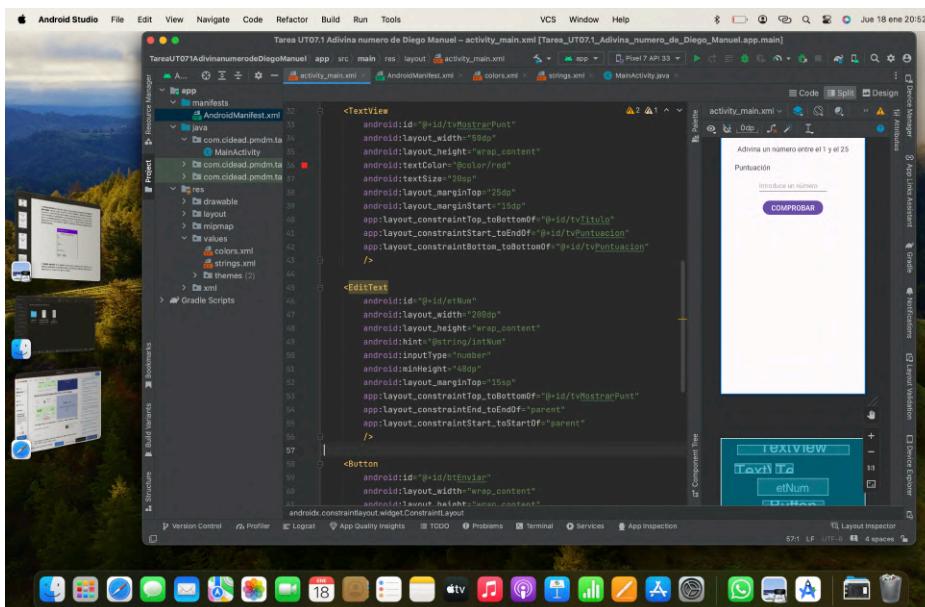
Tarea 7.1

Empezaremos creando el proyecto al cual llamaremos "Tarea UT07.1 Adivina numero de Diego Manuel".

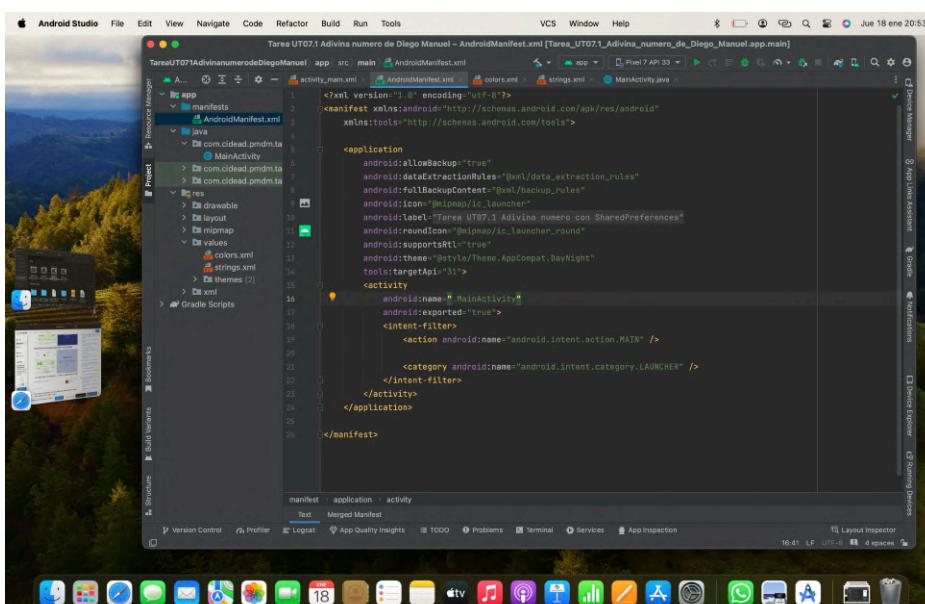


El activity_main contendrá un TextView para contener el texto "Adivina un número entre el 1 y el 25", otro TextView que contendrá el texto "Puntuacion", un EditText que albergará el número que introduzca el usuario, el cual será de tipo "number" para que no pueda introducir letras, un Button con el texto "COMPROBAR" y un último TextView que mostrara al usuario si el número oculto es mayor o menor al número introducido.

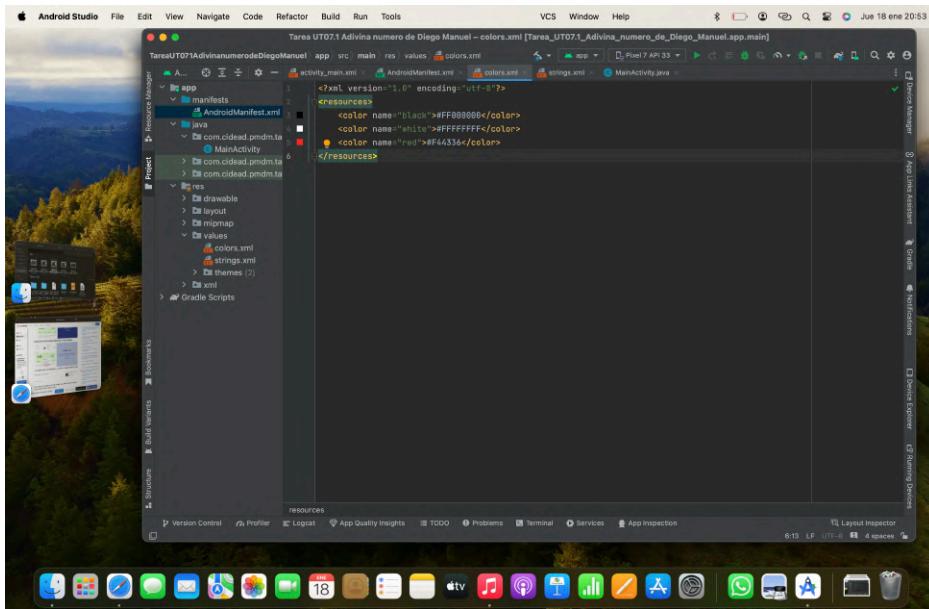




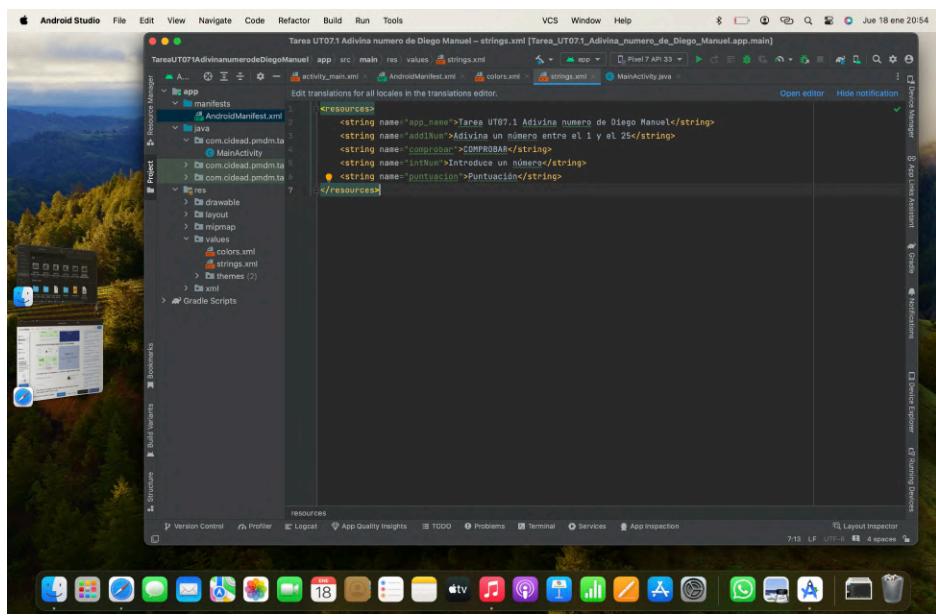
En el `Android_manifest` modificaremos el theme para que muestre la actionbar superior con el nombre de la app.



El archivo colors contendrá el color rojo, llamado red, además del blanco y negro que trae por defecto.

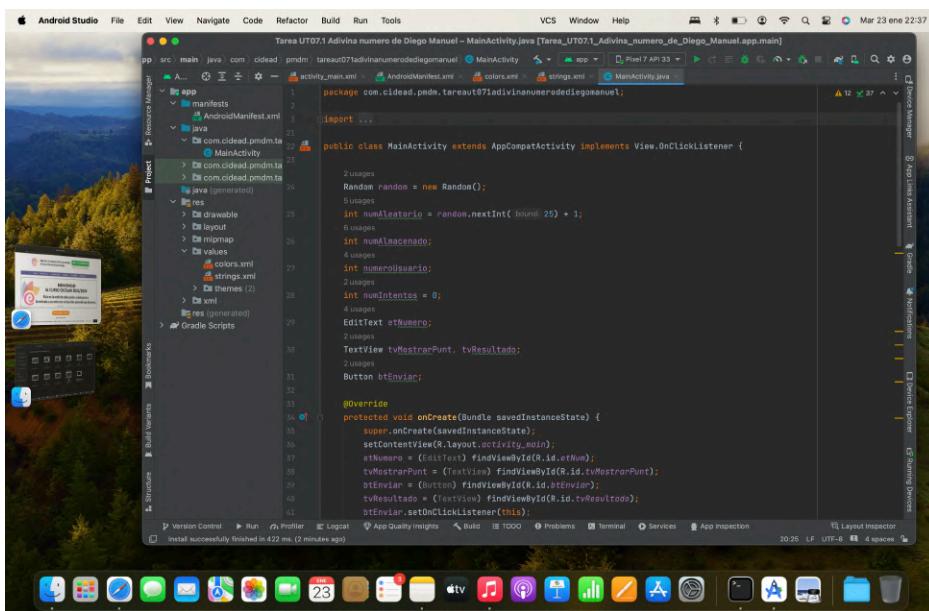


El archivo strings contendrá los textos de los TextView y del Button creados en el layout.



El Main_Activity contendrá una variable de tipo Random, una variable de tipo int llamada "numAleatorio", que contendrá el número creado de forma aleatoria entre el 1 y el 25, una variable de tipo int llamada numAlmacenado que será la encargada de guardar el número de aciertos que lleva el usuario (número que esta guardado en un archivo de tipo xml en shared_prefs), otra variable de tipo int llamada "numUsuario" que contendrá el número que ha introducido el usuario, otra variable de tipo int llamada "numIntentos" que la iniciaremos a 0 y será la encargada de guardar el número de intentos que va necesitando el usuario para adivinar el número secreto, un EditText llamada "etNúmero" que almacenará el número que el usuario ha introducido, dos de tipo TextView llamadas "tvMostrarPunt" y "tvResultado" que contendrán

las veces que ha acetado el usuario y el número secreto respectivamente y una de tipo Button llamada "btEnviar".



En el método OnClick crearemos el SharedPreferences al que llamaremos "sharedPreferences" que recuperara (o creará en caso de no existir) el archivo con nombre "Score" en modo privado.

Tras ello, mediante le método Editor del SharedPreferences editara dicho archivo.

Posteriormente instanciaremos la variable numAlmacenado con el sharedPreferences creado anteriormente, la llave será "Ganado" y el valor por defecto 0.

A continuación crearemos un if que comprobará si el usuario no a introducido ningun número (es decir, el EditText esta vacío) y si es así le mostrara un toast con el mensaje "Debes introducir un número", en caso contrario, obtendremos el número que ha introducido el usuario y lo guardaremos en una variable de tipo string para, posteriormente pasarlo a tipo int mediante le método Integer.parseInt.

Tras ello, dentro del else que nos ocupa, crearemos varios if para comprobar si el número secreto es menor, mayor o igual al introducido por el usuario y mostrara el mensaje correspondiente en el TextView creado a tal fin.

En el caso de ser igual ambos número, además de lo anterior, aumentara en uno la variable numAlmacenado, mostrara un toast con los aciertos que lleva y otro toast con los intentos que ha necesitado para adivinarlo, mediante el método putInt aplicado al editor creador anteriormente le pasaremos la variable numAlmacenado al fichero del SharedPreferences para que lo almacene, aplicaremos el método commit al editor y crearemos una

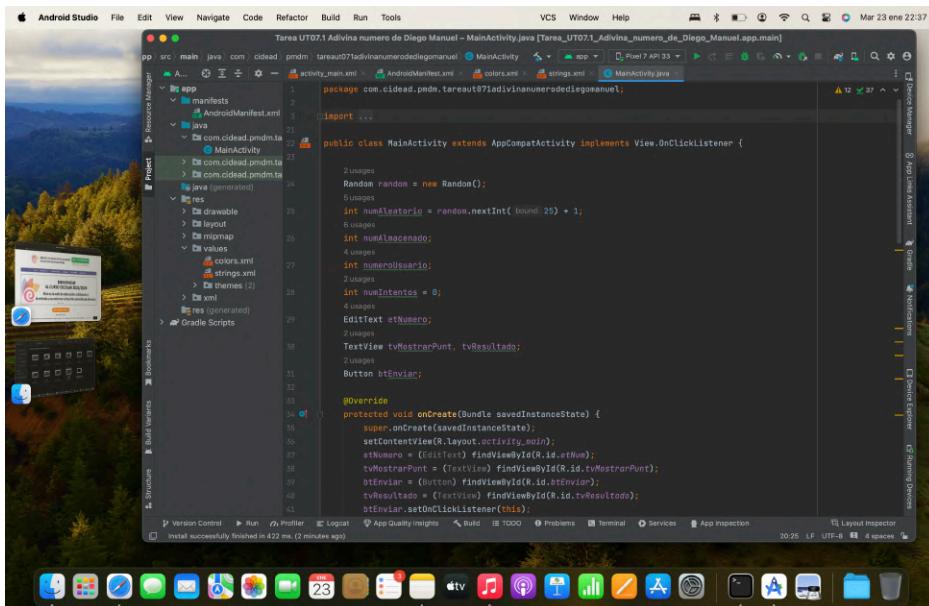
variable de tipo OutputStreamWriter llamada archivo y la instanciaremos a null.

A continuación, también dentro del if que nos ocupa, abriremos (o crearemos en caso de que no exista) un nuevo fichero de tipo txt llamado numSecreto que almacenará el número aleatorio que hemos generado.

Mediante el método write aplicado al archivo le pasaremos el numAleatorio para que lo guarde en archivo, pasandolo a string, le aplicaremos el método flush al archivo y posteriormente lo cerraremos mediante el método close.

Tras ello, crearemos una instancia del fichero, pasandole la ruta, crearemos una variable de tipo ImputStreamReader llamada archivoRecivido al que le pasaremos el nombre del fichero, un BufferedReader al que le pasaremos la variable archivoRecibido, un string llamado linea, que leera la primera linea del archivo recibido, mostraremos en un toast el número secreto almacenado en la variable linea y volveremos a generar el número aleatorio nuevamente para que pueda seguir jugando el usuario.

A continuación, ya fuera el if anterior, sumaremos uno a la variable numIntentos y vaciaremos la variable etNumero.



```

        btEnviar.setOnClickListener(this);
        SharedPreferences sharedpreferences = getSharedPreferences( name: "Score", Context.MODE_PRIVATE);
        SharedPreference.Editor editor = sharedpreferences.edit();
        numAlmacenado = sharedpreferences.getInt( key: "Guardado", defaultValue: 0);
        tvMostrarPunt.setText(String.valueOf(numAlmacenado));

        @Override
        public void onClick(View v) {
            SharedPreferences sharedpreferences = getSharedPreferences( name: "Score", Context.MODE_PRIVATE);
            SharedPreference.Editor editor = sharedpreferences.edit();
            numAlmacenado = sharedpreferences.getInt( key: "Guardado", defaultValue: 0);

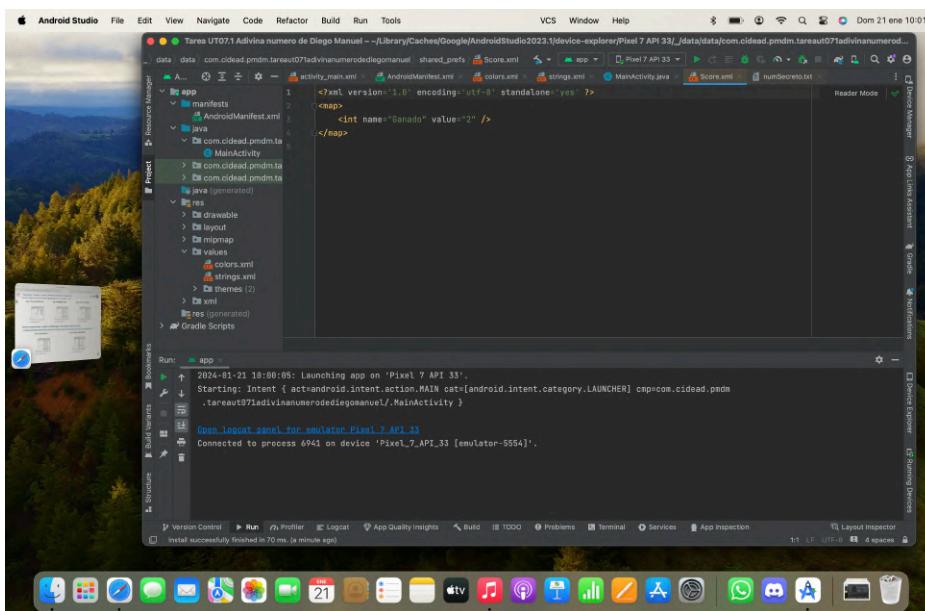
            if(etNumeros.getText().toString().isEmpty()) {
                Toast.makeText( context: this, text: "Debes introducir un numero", Toast.LENGTH_LONG).show();
            } else {
                String textoEtNumeros = etNumero.getText().toString();
                numeroUsuario = Integer.parseInt(textoEtNumero);
                if (numeroUsuario < numAleatorio) {
                    tvResultado.setText("El numero es menor");
                }
                if (numeroUsuario > numAleatorio) {
                    tvResultado.setText("El numero es mayor");
                }
                if (numeroUsuario == numAleatorio) {
                    tvResultado.setText("Has acertado el numero");
                    numero++;
                    numAlmacenado++;
                    Toast.makeText( context: this, text: "Llevas " + numAlmacenado + " aciertos", Toast.LENGTH_LONG).show();
                    Toast.makeText( context: this, text: "Has necesitado " + numIntentos + " intentos", Toast.LENGTH_LONG).show();
                    editor.putInt("Guardado", numAlmacenado);
                    editor.commit();
                }
            }
            OutputStreamWriter archive = null;
        }
    
```

El contenido del archivo numSecret.txt podemos verlo en la siguiente imagen.

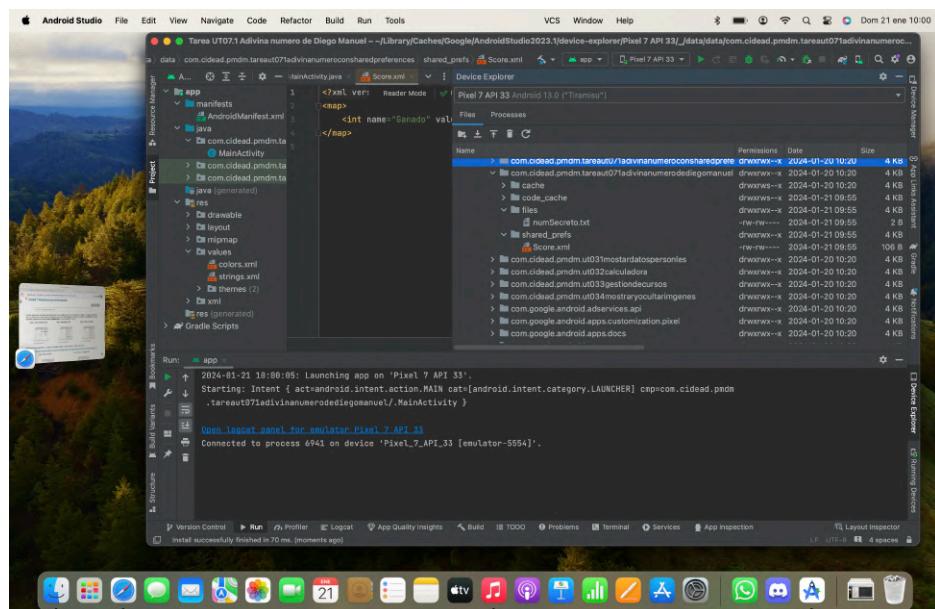
```

        2024-01-21 18:00:05: Launching app on 'Pixel 7 API 33'.
        Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.cidead.pndm.tareaut071adivinarnumerodemanuel/.MainActivity }
        .tareaut071adivinarnumerodemanuel/.MainActivity
        Open logcat panel for emulator Pixel 7 API 33
        Connected to process 6941 on device 'Pixel_7_API_33 [emulator-5554]'.
    
```

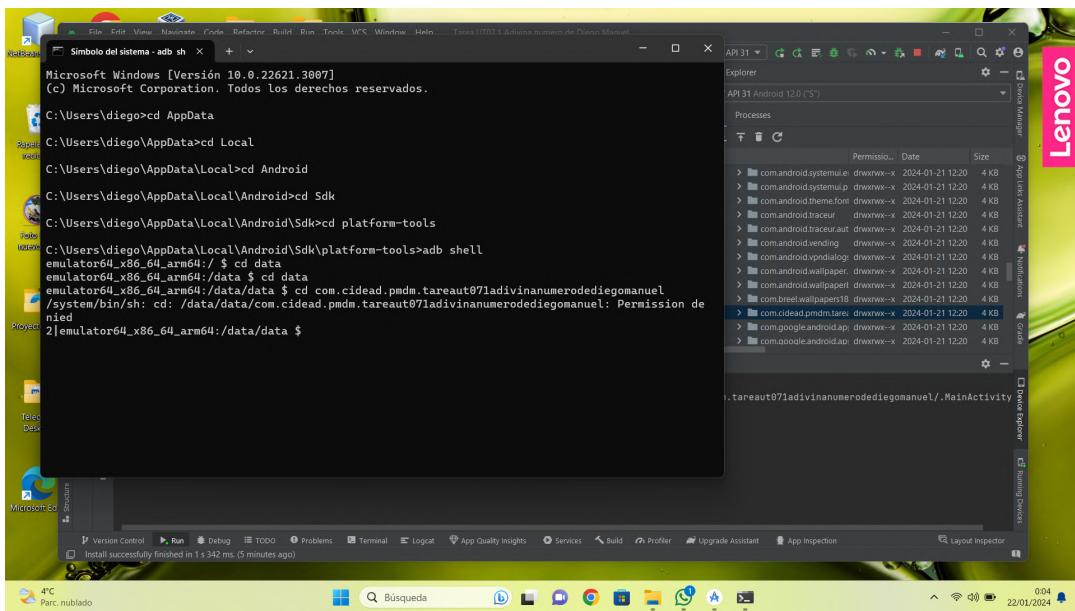
El contenido del archivo Score podemos verlo en la siguiente imagen.



En la siguiente imagen podemos ver donde están ubicados los dos archivos anteriores dentro del paquete de la propia aplicación.

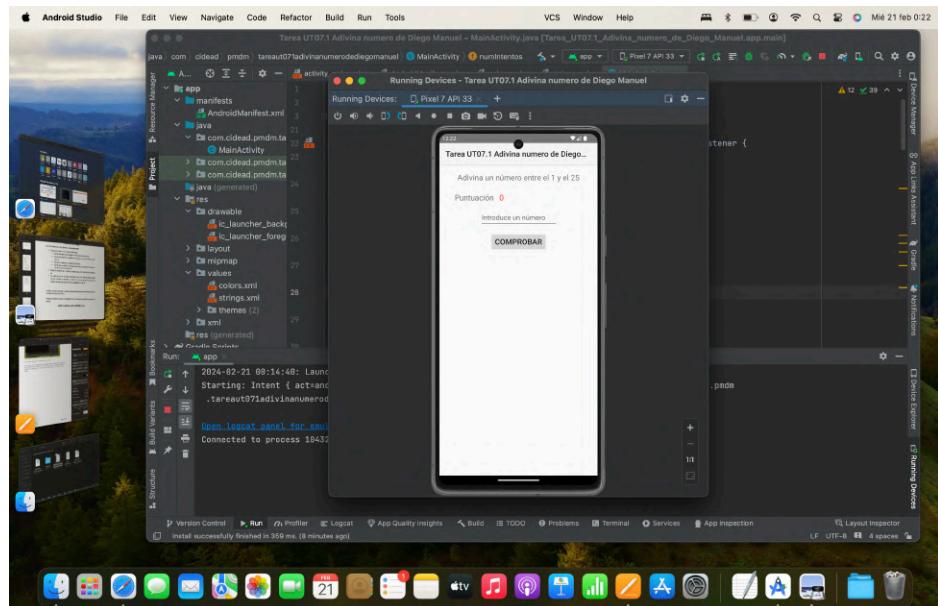


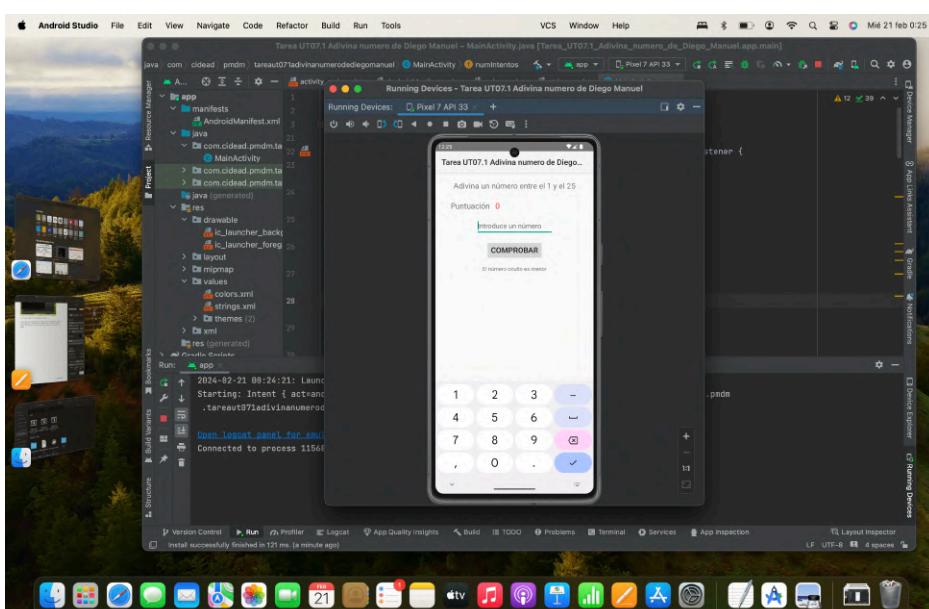
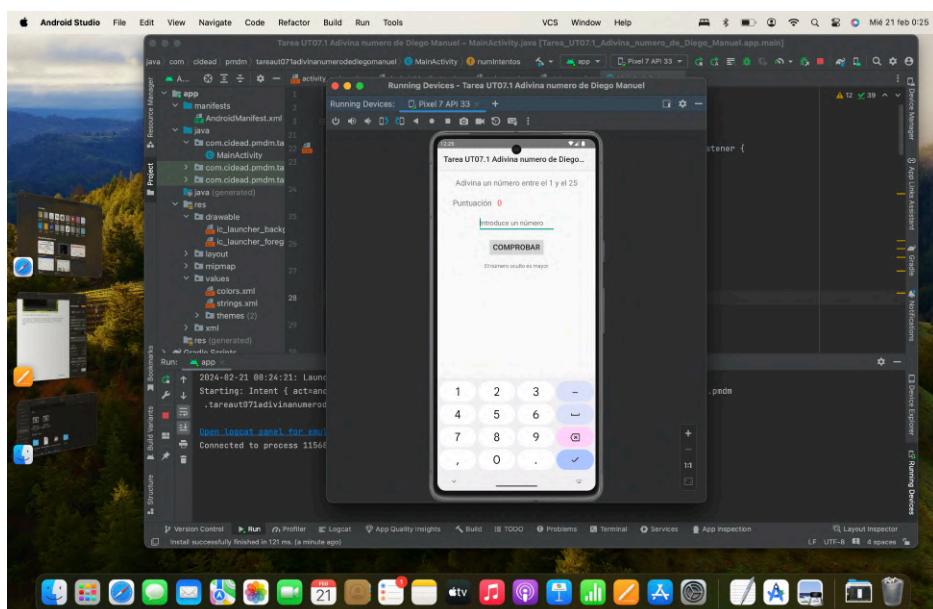
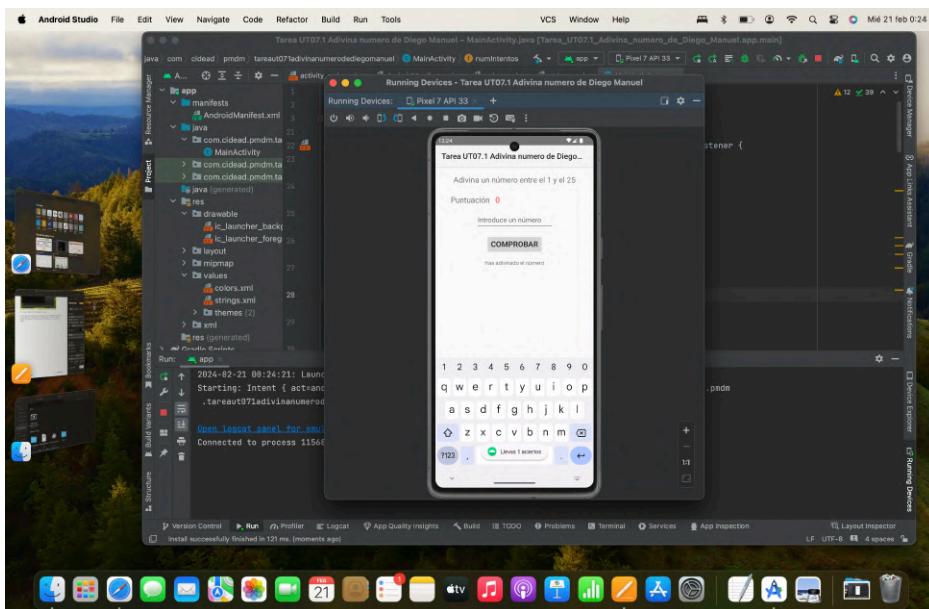
Para acceder mediante el adb Shell he tenido que cambiar de pc a uno con Windows ya que en el mac no he conseguido acceder. He pasado el proyecto al otro pc y podemos ver en la siguiente imagen como puedo llegar hasta el paquete que contiene la aplicación pero no soy capaz de otorgar los permisos para acceder al archivo en cuestión.

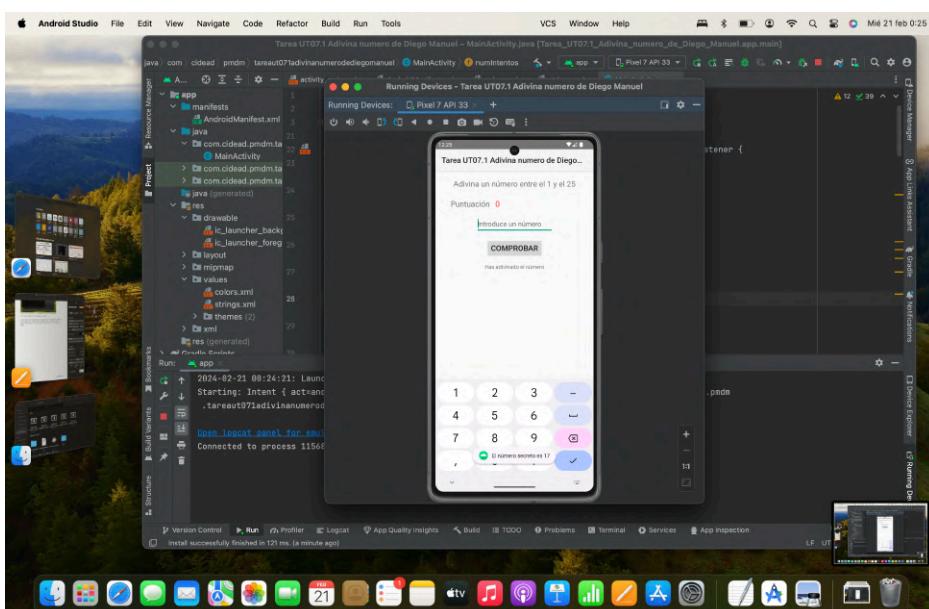
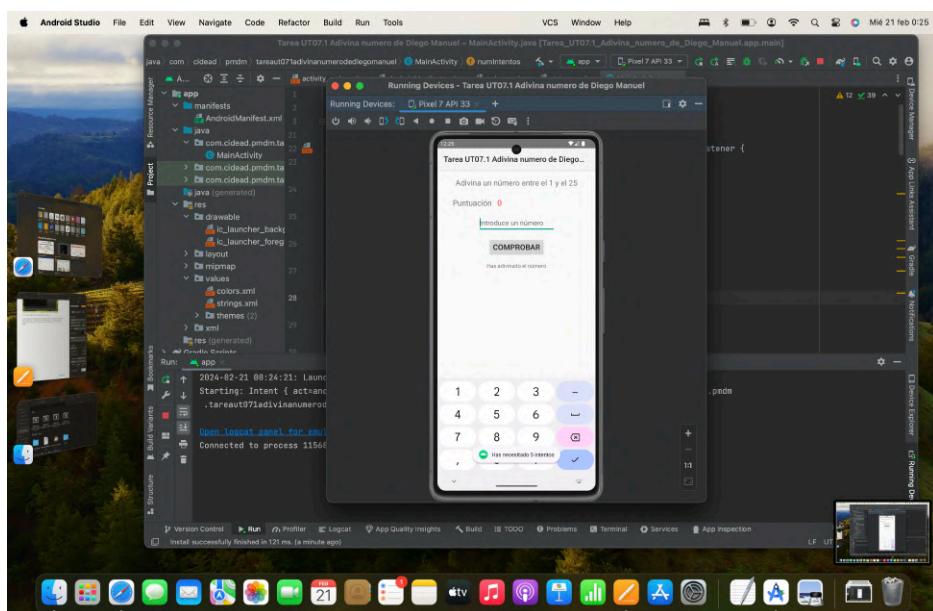
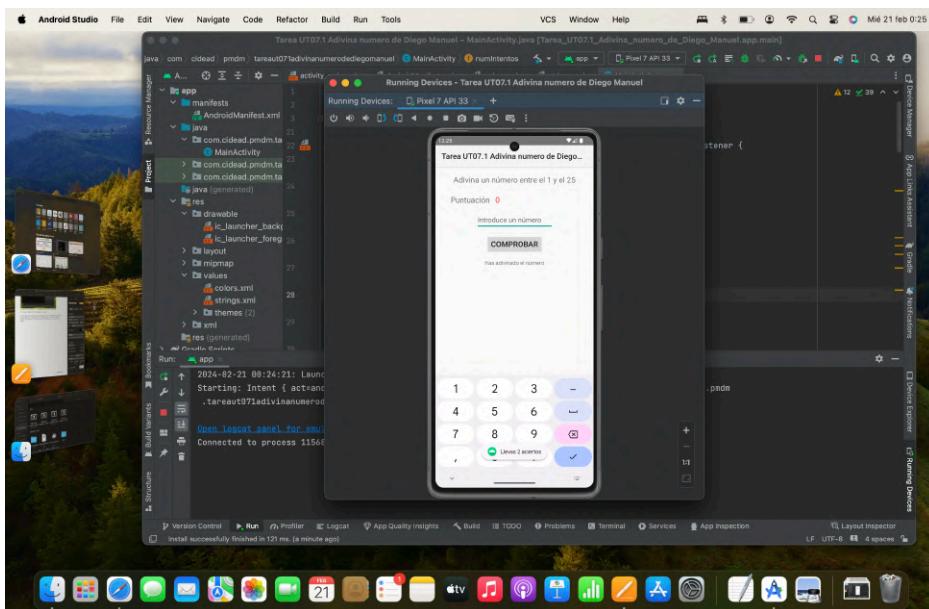


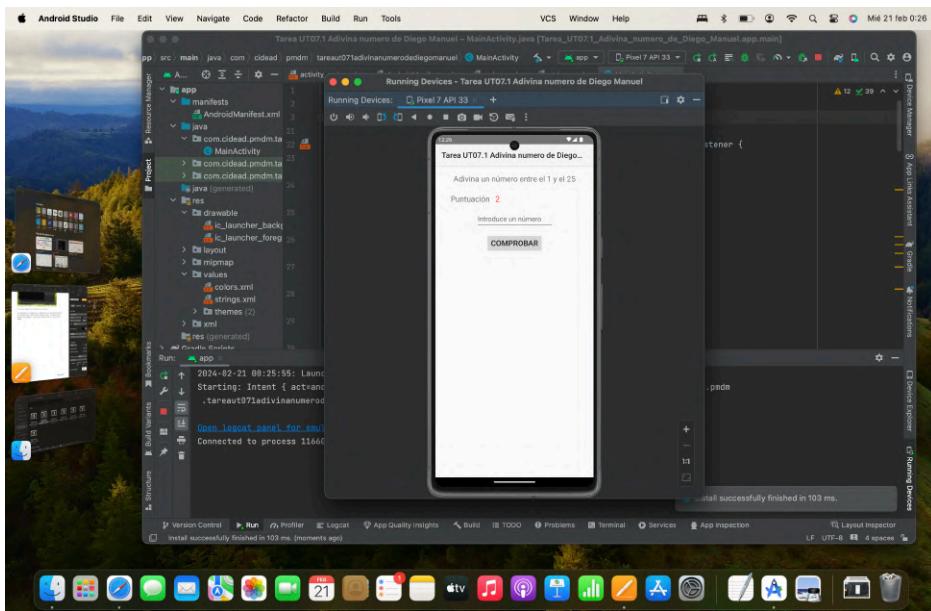
Y con esto damos por finalizada la tarea 7.1

A continuación se muestran las capturas de la aplicación abierta por primera vez, cuando pide un número mayor, un número menor, cuando ha acertado y, tras jugar varias veces, el inicio de la misma.





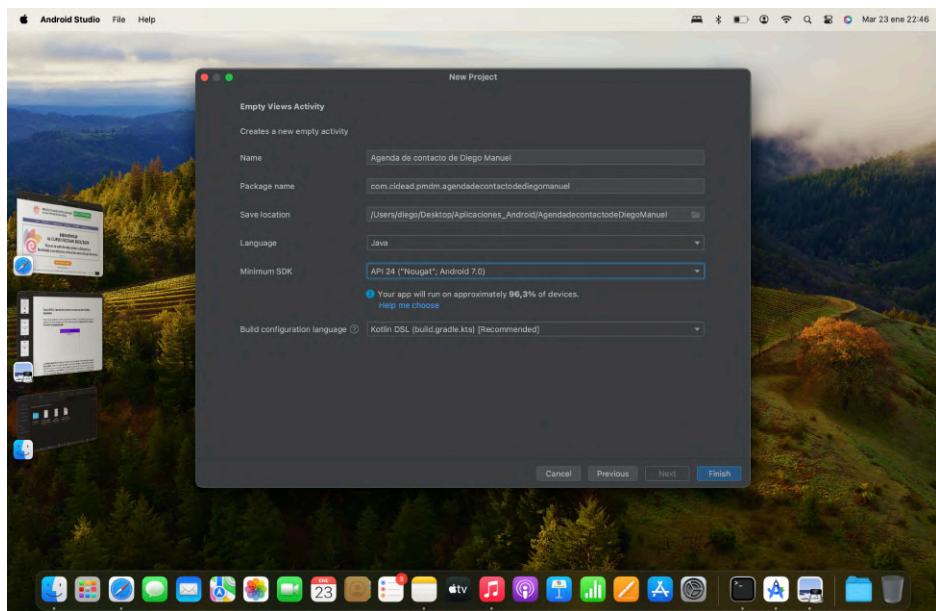




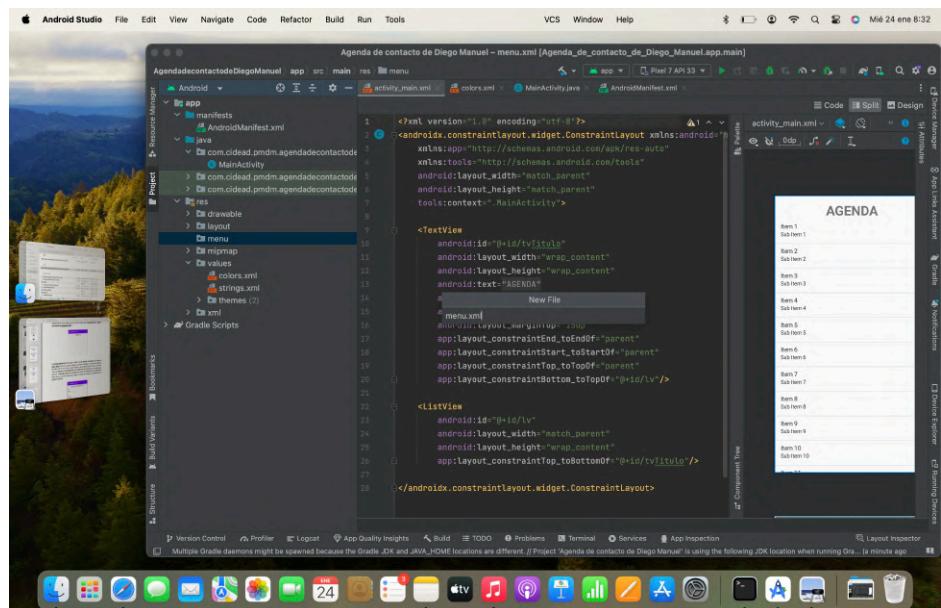
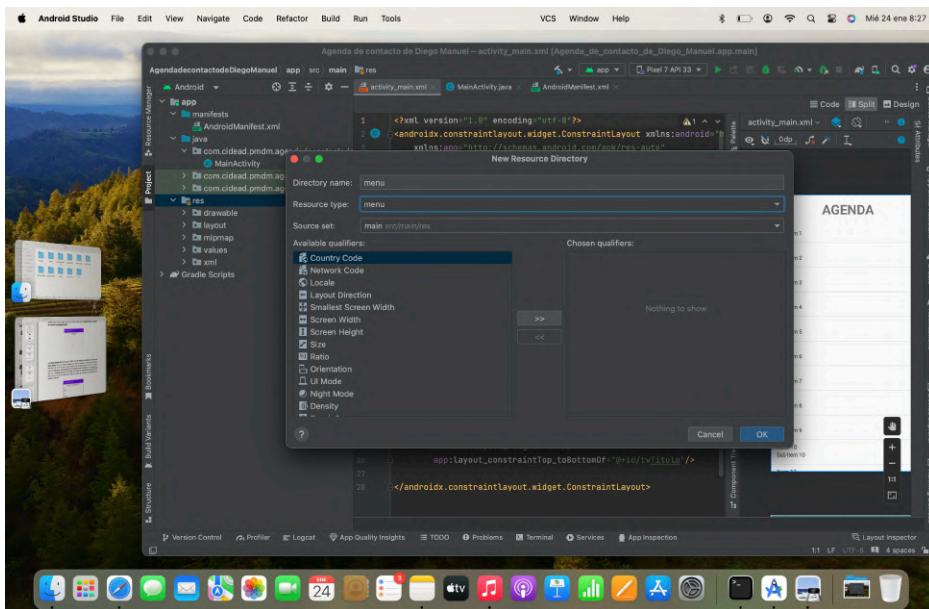
Y con esto damos por finalizada esta tarea.

Tarea 7.2 Agenda de contacto.

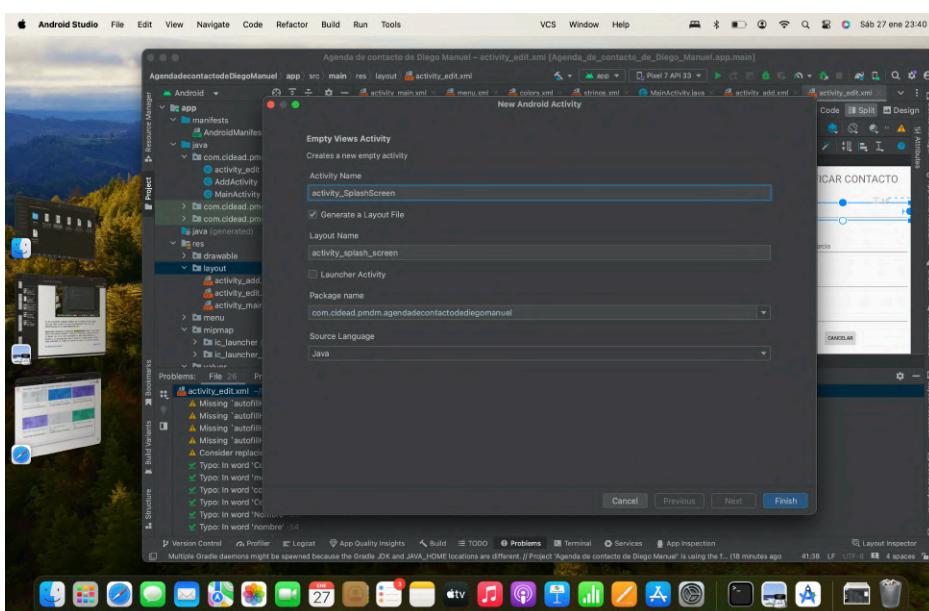
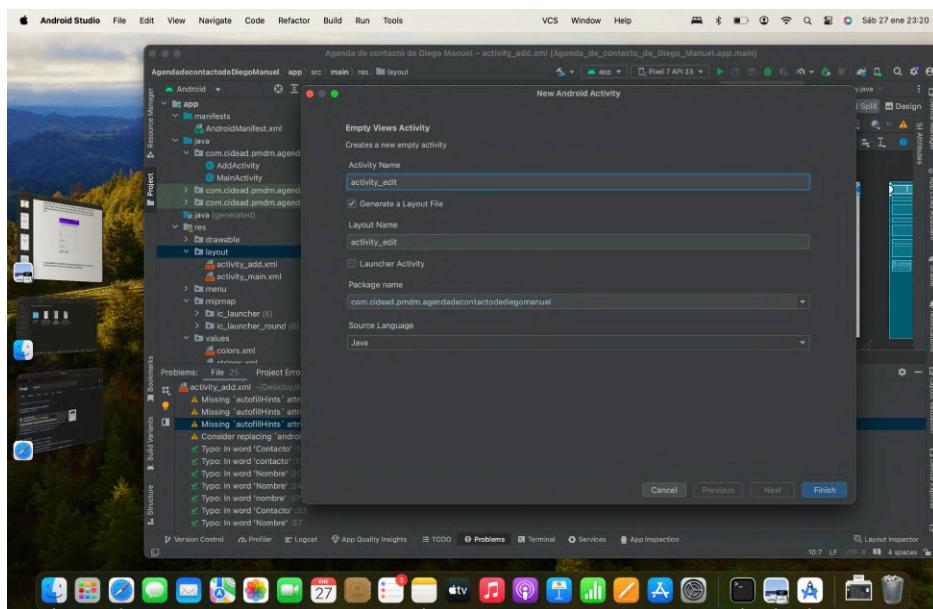
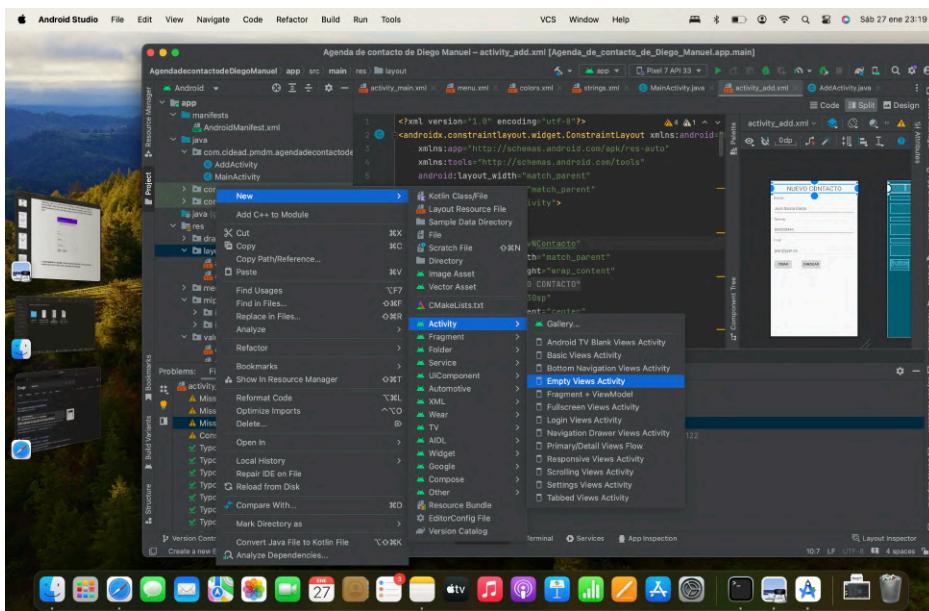
Empezaremos creando el proyecto, que en nuestro caso se llamará "Agenda de contacto de Diego Manuel".



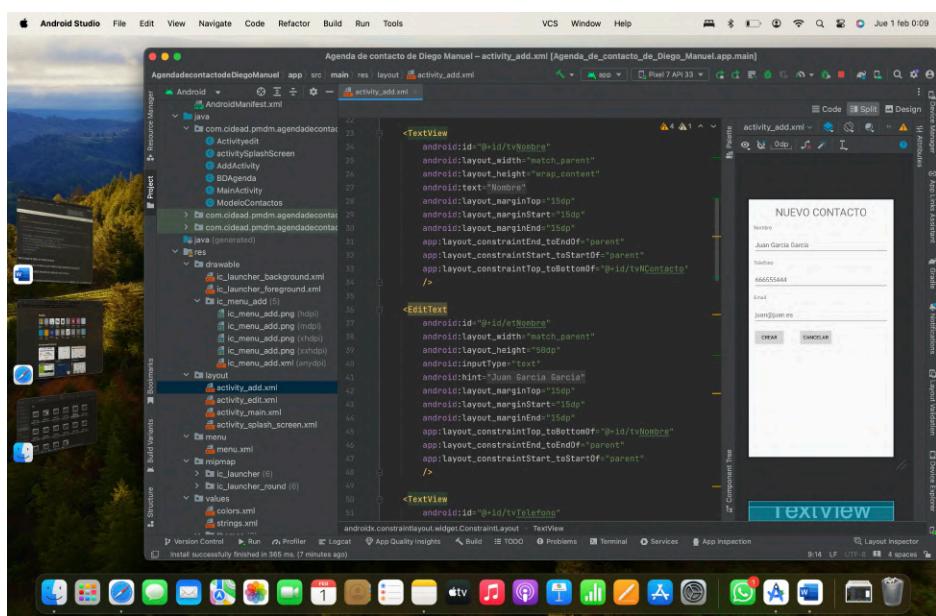
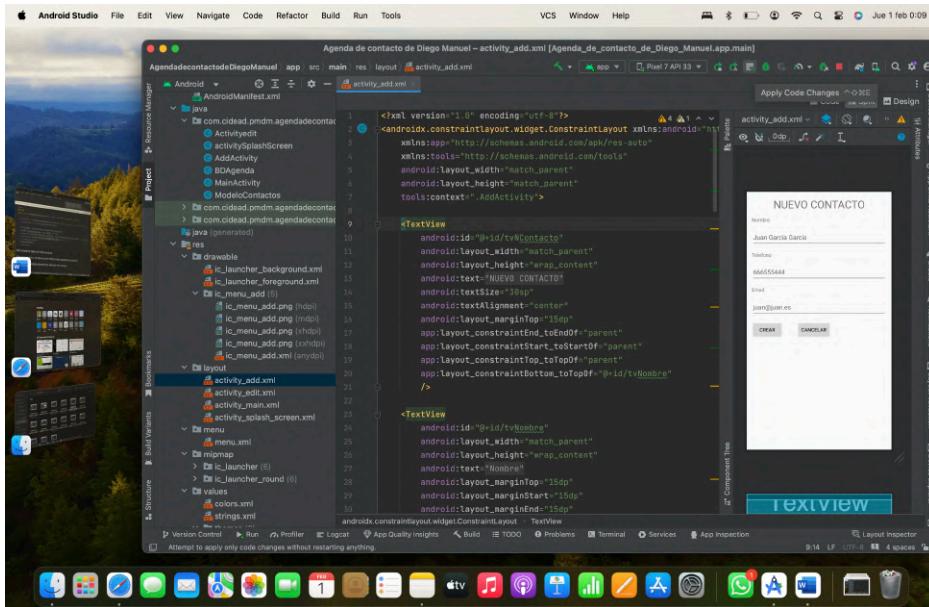
Tras ello crearemos un recurso de tipo menú llamado menú que albergara nuestro menú un fichero xml llamado del mismo modo.

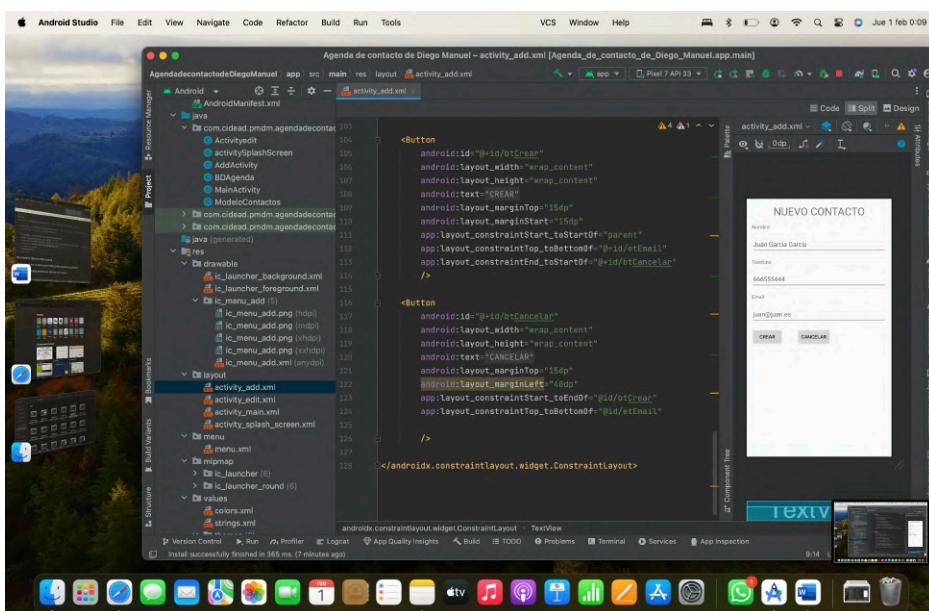
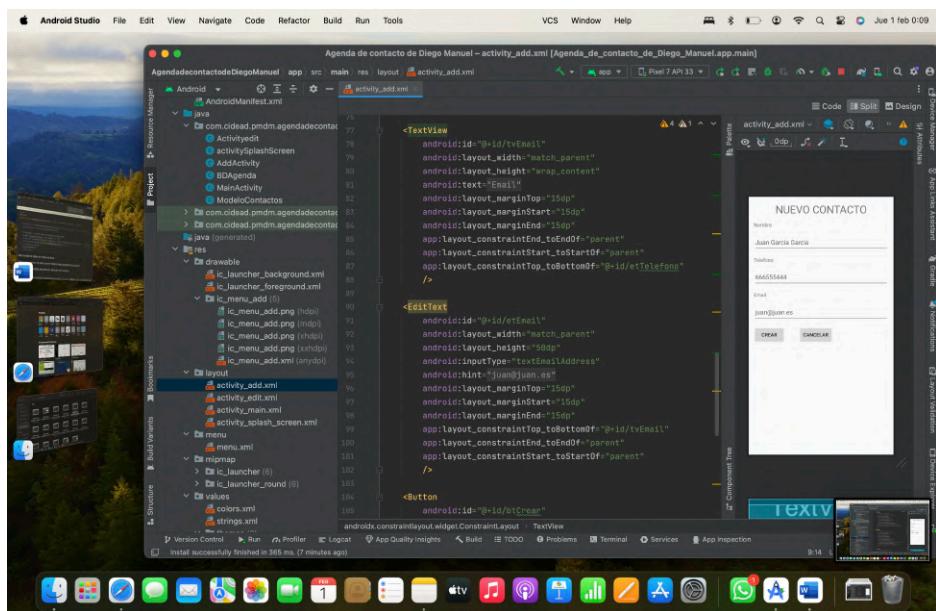
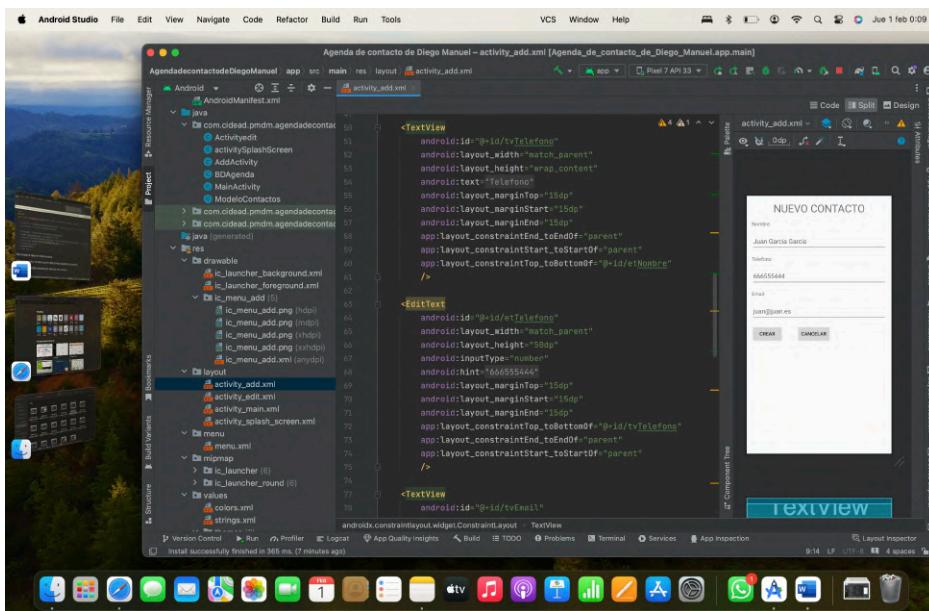


A continuación crearemos una serie de actividades vacías llamadas "activity_edit" que albergara la actividad para editar el contacto seleccionado, "activity_SplashScreen" que será la encargada de albergar nuestro SplashScreen y "addActivity" que será la encargada de albergar la actividad para crear un nuevo contacto

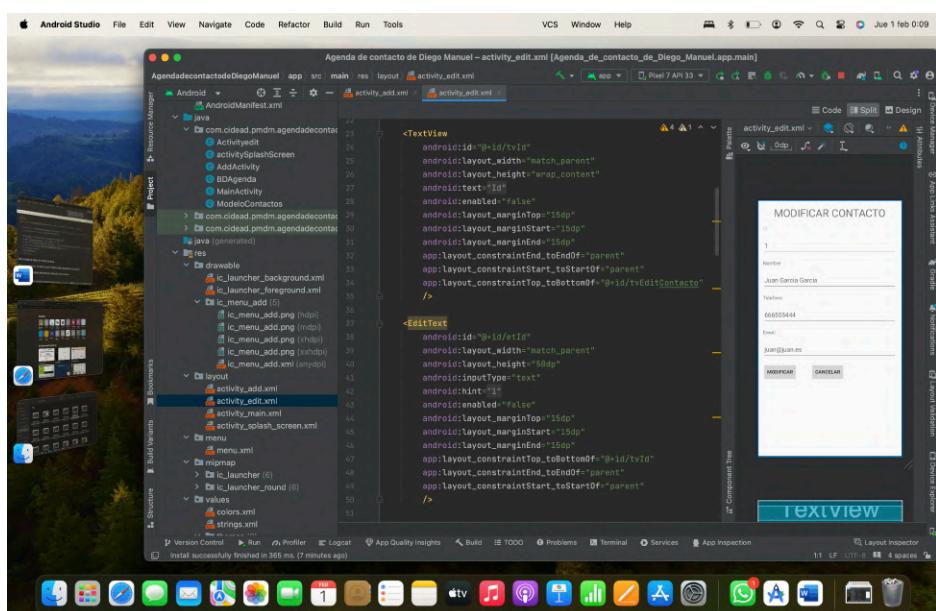
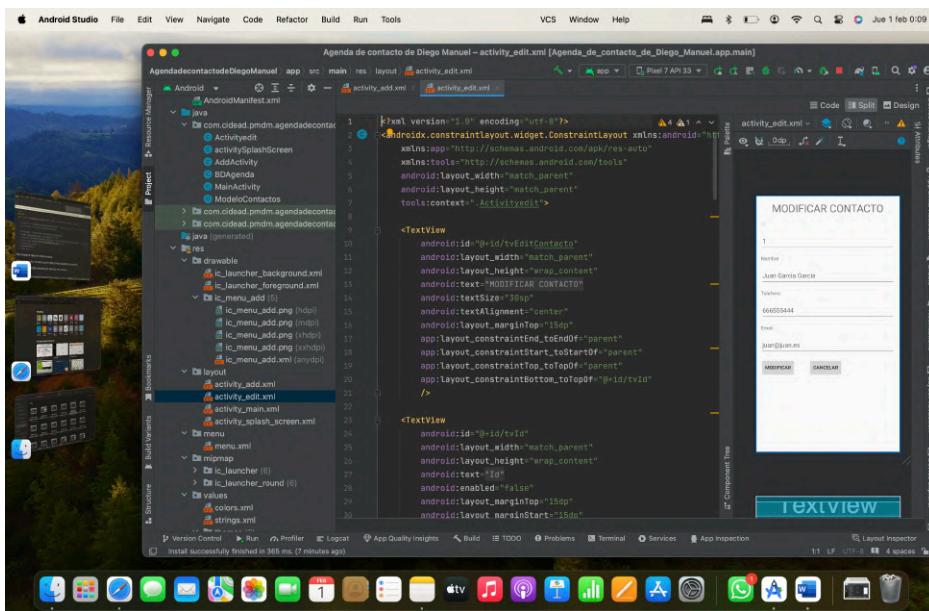


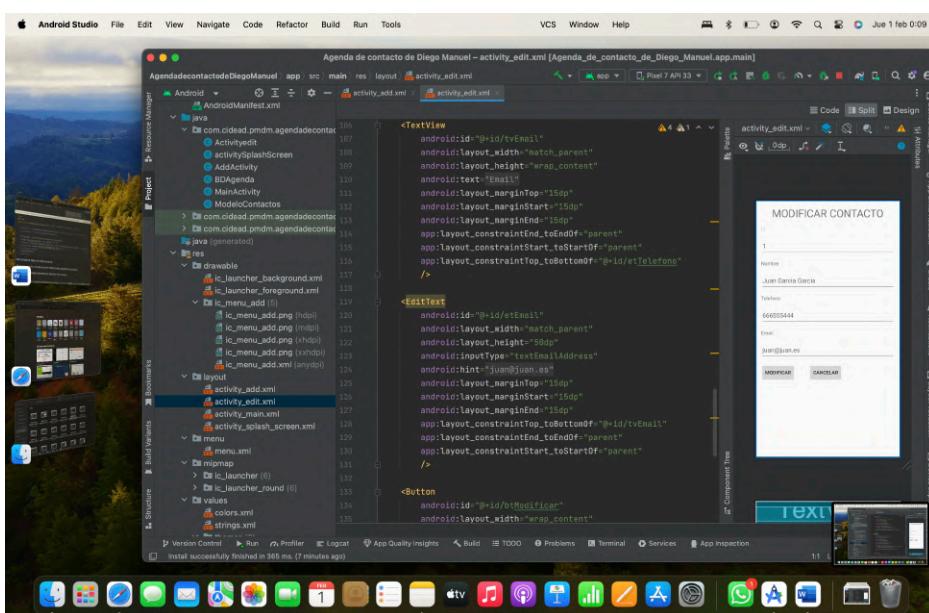
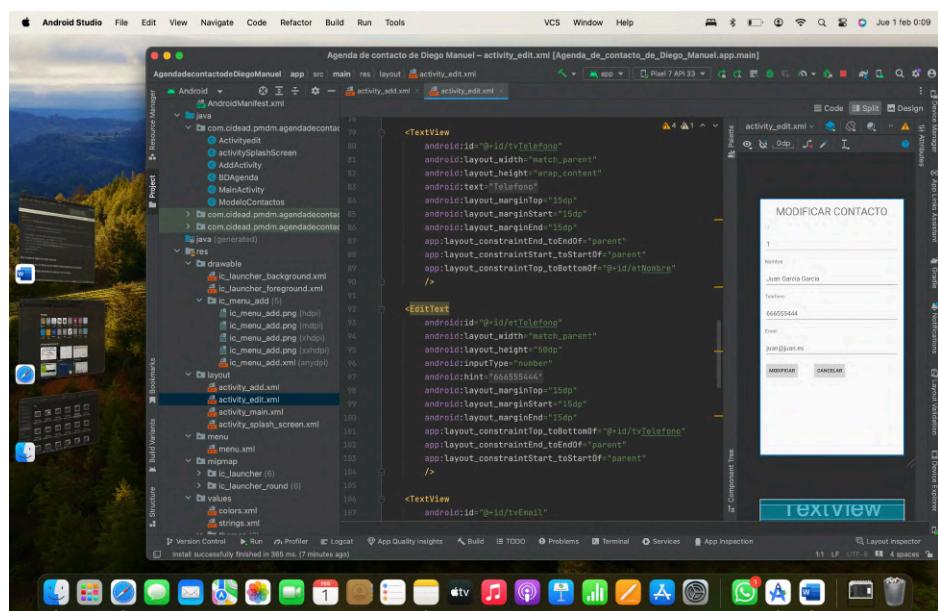
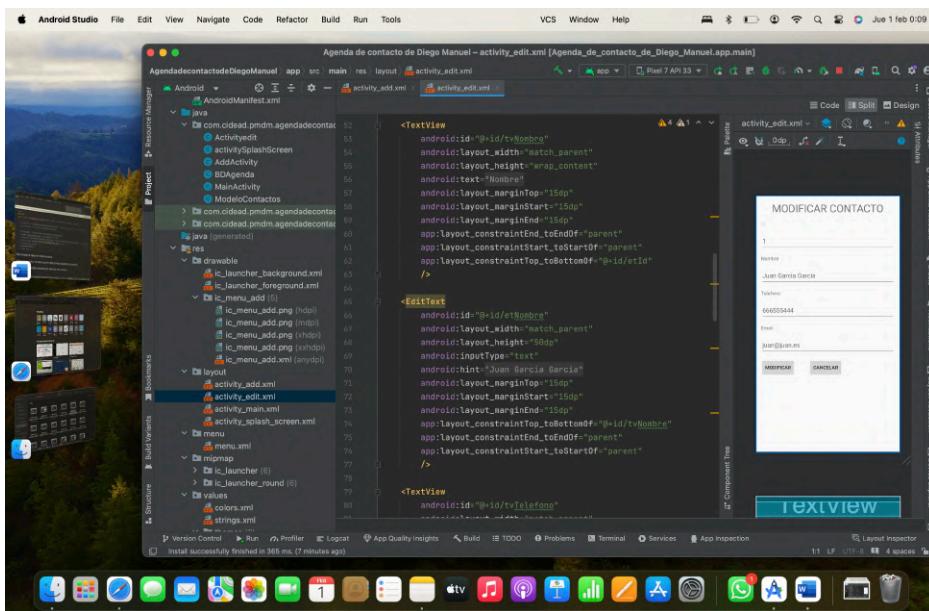
El archivo activity_add.xml contendrá un TextView para guardar el título de la actividad cuyo texto será "NUEVO CONTACTO", un TextView con el texto "Nombre", un EditText que será usado para introducir el usuario el nombre del contacto, otro TextView con el texto "Telefono", un EditText que será usado para que el usuario introduzca el teléfono del contacto, otro TextView con el texto "Email", un EditText que será utilizado para que el usuario introduzca el email del contacto y dos botones, unos con el texto "CREAR" y otro con el texto "CANCELAR" respectivamente.

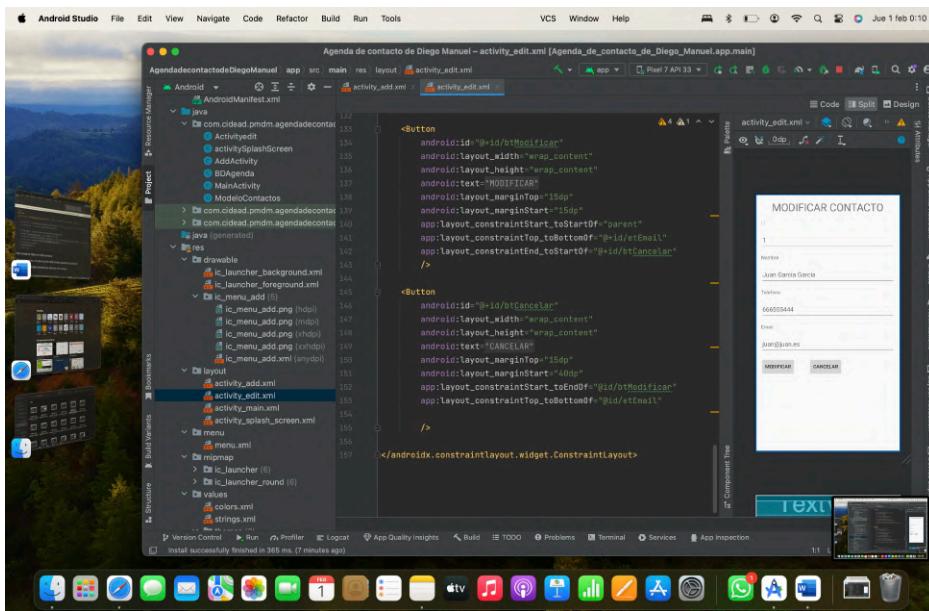




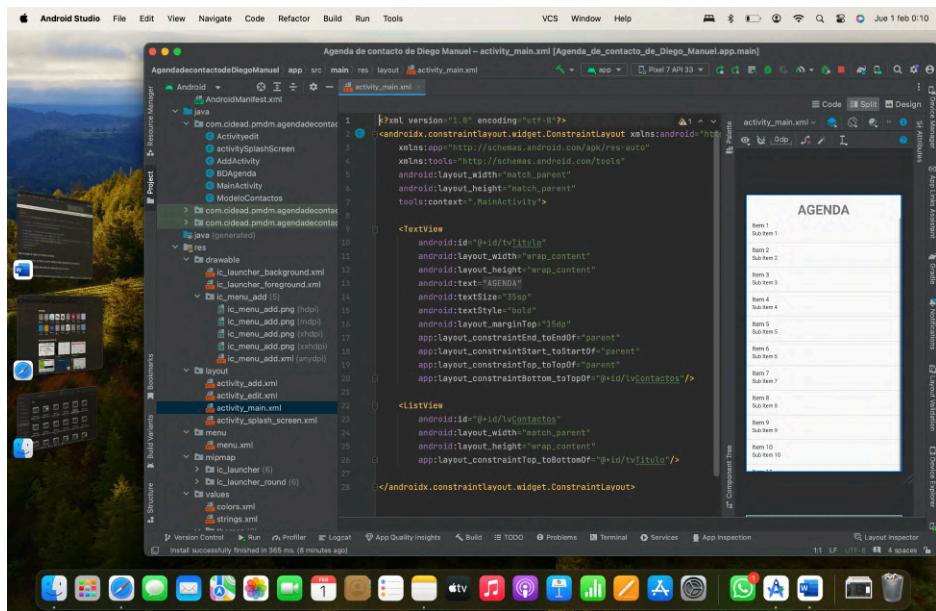
El archivo activity_edit.xml contendrá un TextView para guardar el título de la actividad cuyo texto será "MODIFICAR CONTACTO", un TextView con el texto "Nombre", un EditText que será usado para recuperar el nombre e introducir el usuario el nombre del contacto, otro TextView con el texto "Telefono", un EditText que será usado para recuperar el teléfono y que el usuario introduzca el teléfono del contacto, otro TextView con el texto "Email", un EditText que será utilizado para recuperar el email y que el usuario introduzca el email del contacto y dos botones, unos con el texto "MODIFICAR" y otro con el texto "CANCELAR" respectivamente.



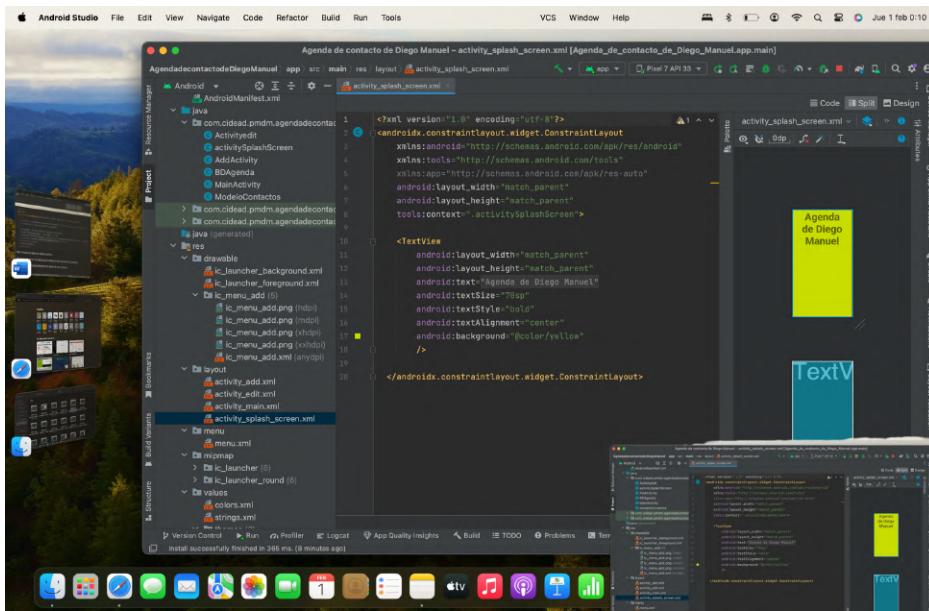




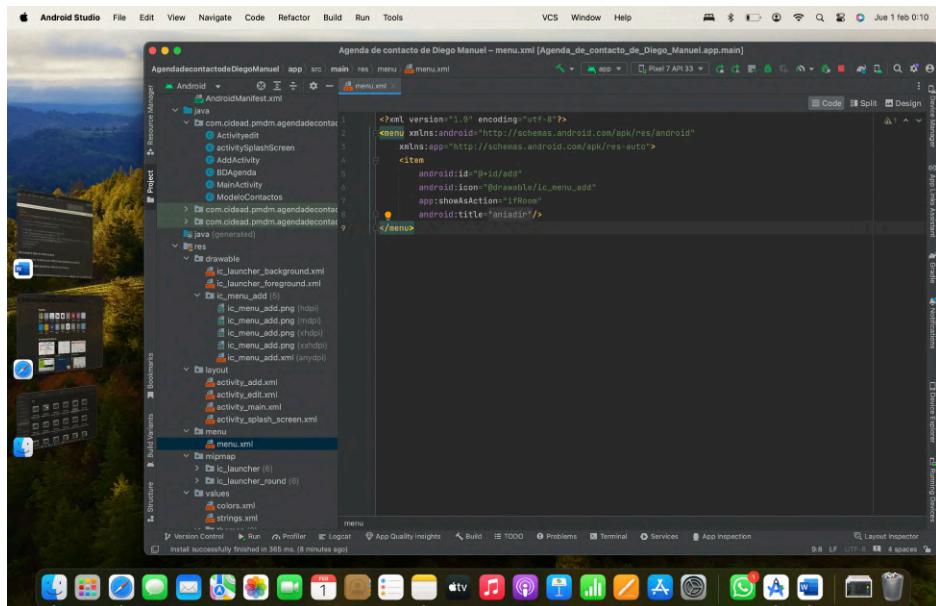
El archivo `activity_main.xml` dispondrá de un `TextView` para guardar el título de la actividad cuyo texto será "AGENDA" y un `ListView` para mostrar los contactos guardados.



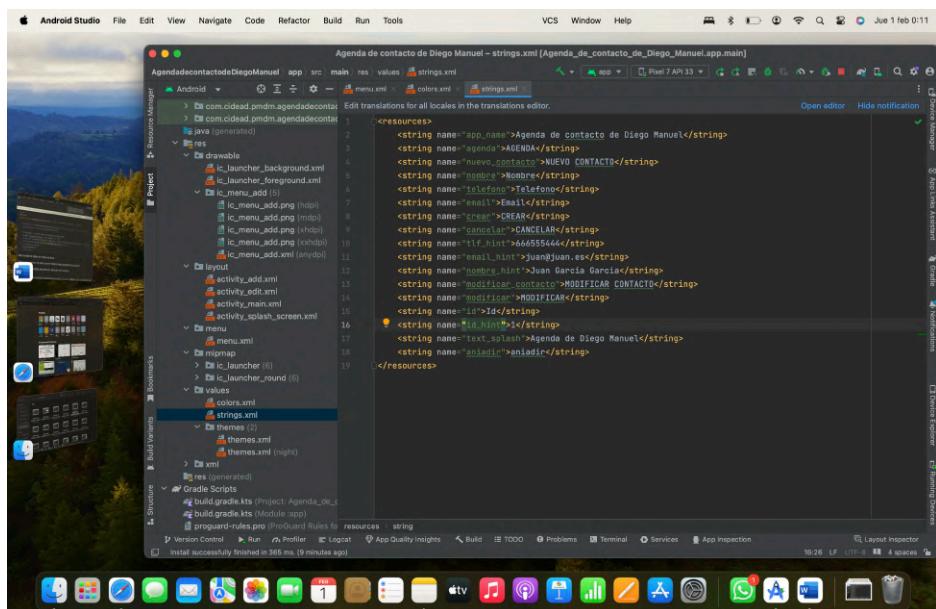
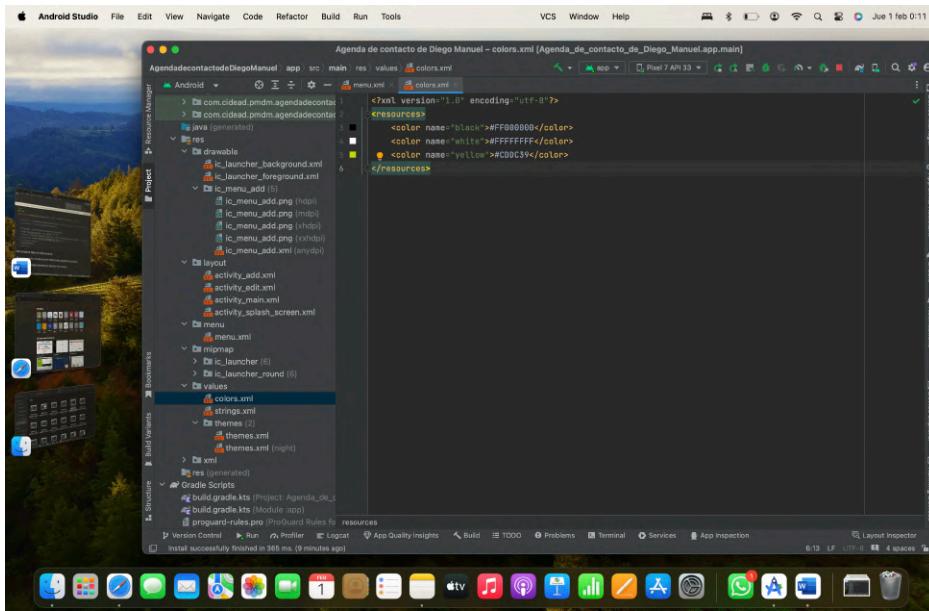
El archivo `activity_splash_screen.xml` contendrá la composición del SplashScreen que vamos a mostrar al usuario.



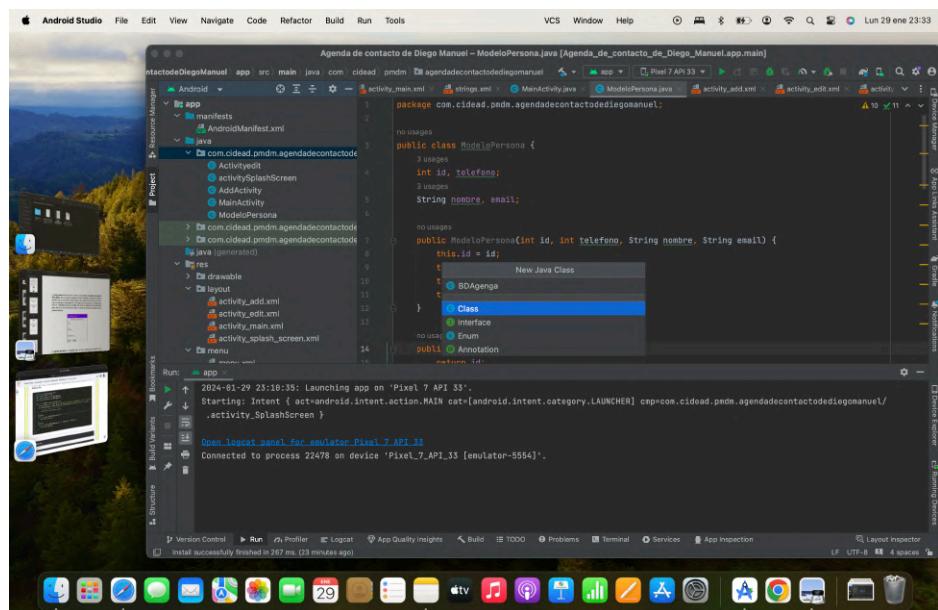
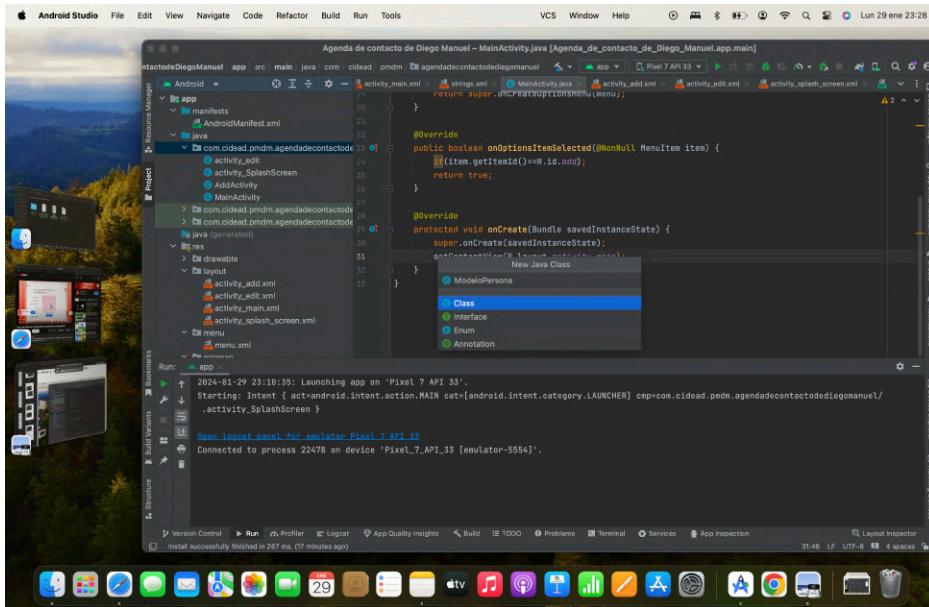
El archivo menú.xml contendrá un ítem con la imagen que elegimos (símbolo +).



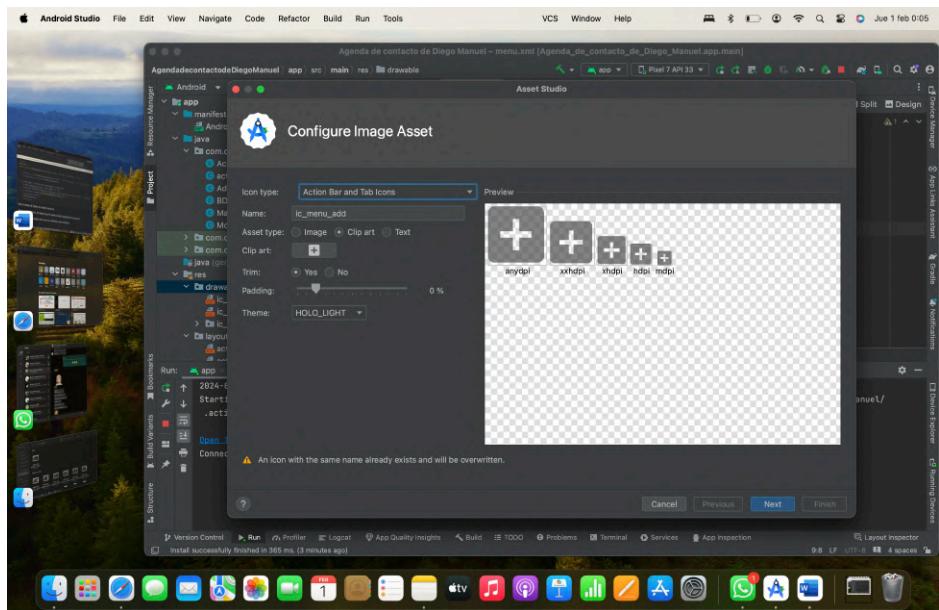
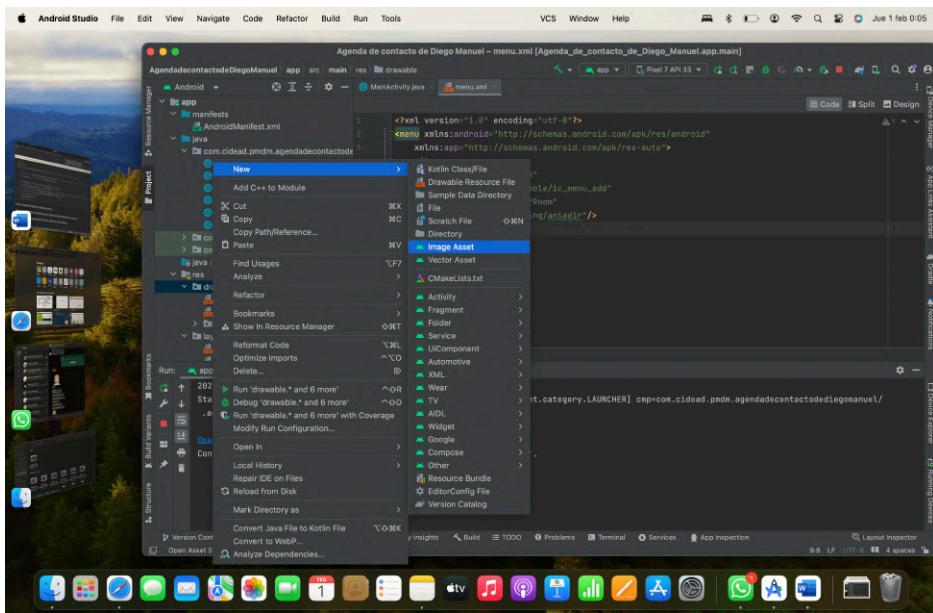
El contenido de los archivos colors.xml y strings.xml podemos verlo en las siguientes imágenes.



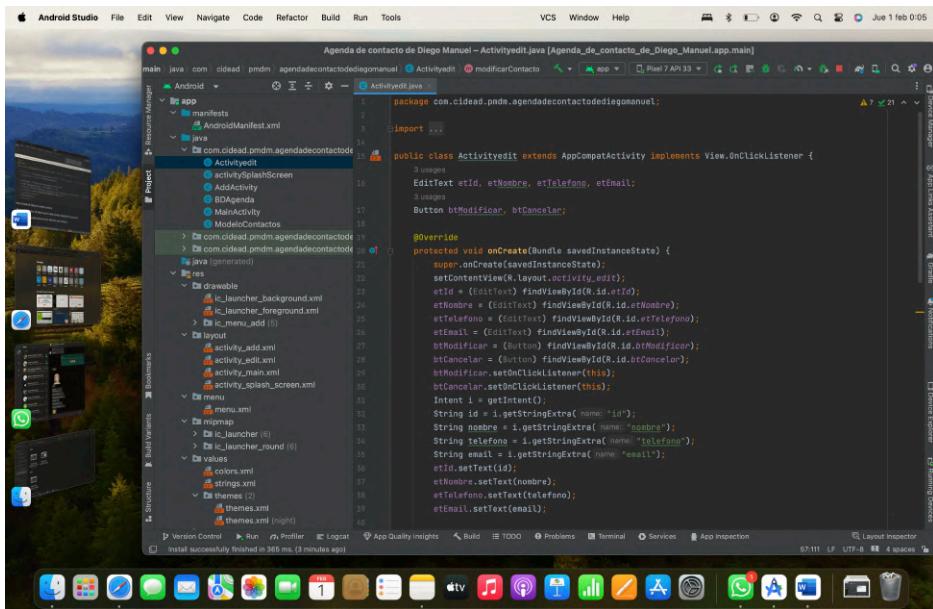
Tras ello crearemos una clase java nueva llamada "ModeloPersona" que albergará objetos de cada uno de los contactos y otra llamada "BDAgenda" (al principio la llamé "BDAgenga" y luego refactorice el nombre, por eso en la imagen sale con "BDAgenga") que realizara la conexión a la base de datos.



Posteriormente crearemos un nuevo recurso de tipo Image Asset con la imagen del símbolo + que hará la función de pasar a la actividad para añadir un nuevo contacto.

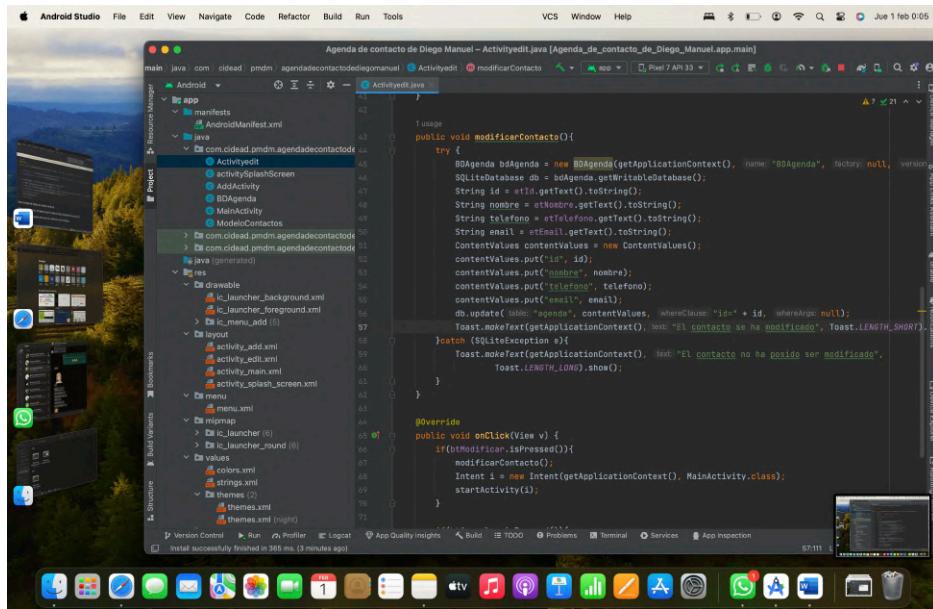


A continuación crearemos el código del Activityedit.java, el cual tendrá cuatro atributos de tipo EditText llamados "etId", "etNombre", "etTelefono" y "etEmail" respectivamente y dos de tipo Button llamados "btModificar" y "btCancelar" respectivamente.



Este activity albergará un método llamado modificarContacto, que creara una nueva conexión a la base de datos, la abrirá en modo escritura, extraerá el texto que el usuario ha introducido en los editText correspondientes, creará un nuevo contentValues y los pasará a la base de datos mediante el método put aplicado al contentValues pasándole como parámetros los datos extraídos.

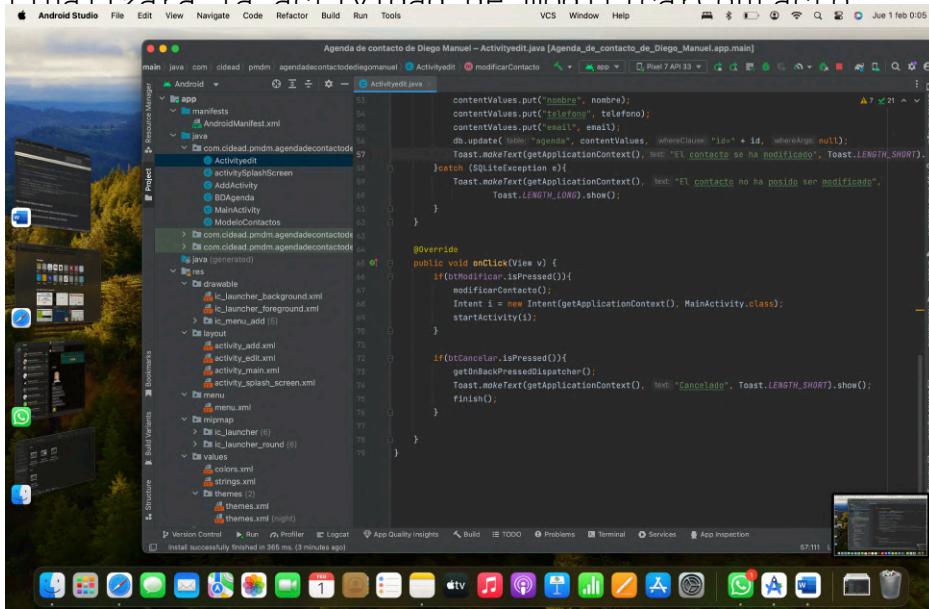
Tras ellos mostrara en toast si todo ha ido bien o por el contrario no se ha podido modificar el contacto.



En el método `onClic` verificaremos si el botón de modificar ha sido pulsado, en cuyo caso llamará al método `modificarContacto`, creará un nuevo intent que volverá a mostrar la actividad principal (`MainActivity`) e iniciará dicho intent.

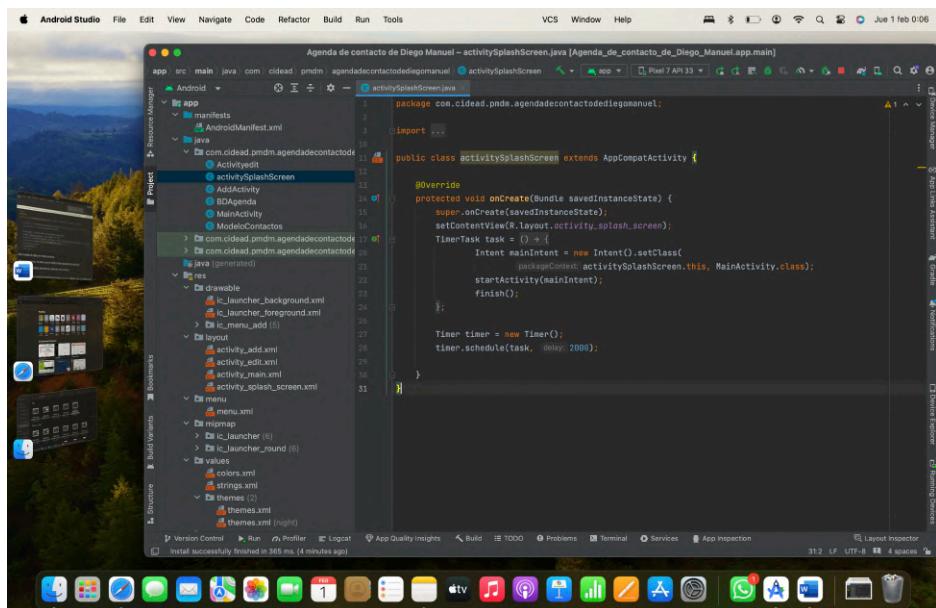
En caso de pulsar el botón cancelar llamará al método `getOnBackPressedDiespatcher`, navegar hacia atrás en la app,

mostrará en un toast que se ha cancelado la modificación y finalizará la actividad de modificarContacto.



La clase activitySplashScreen contendrá, dentro del método onCreate, un setContentView, un TimerTask que a su vez creará un intent que te devuelve a la actividad principal (MainActivity), iniciará dicho intent y finalizará la actividad actual.

A continuación creará un nuevo Timer, que constará de 2 segundos, tiempo que será el que se muestre dicho SplashScreen.

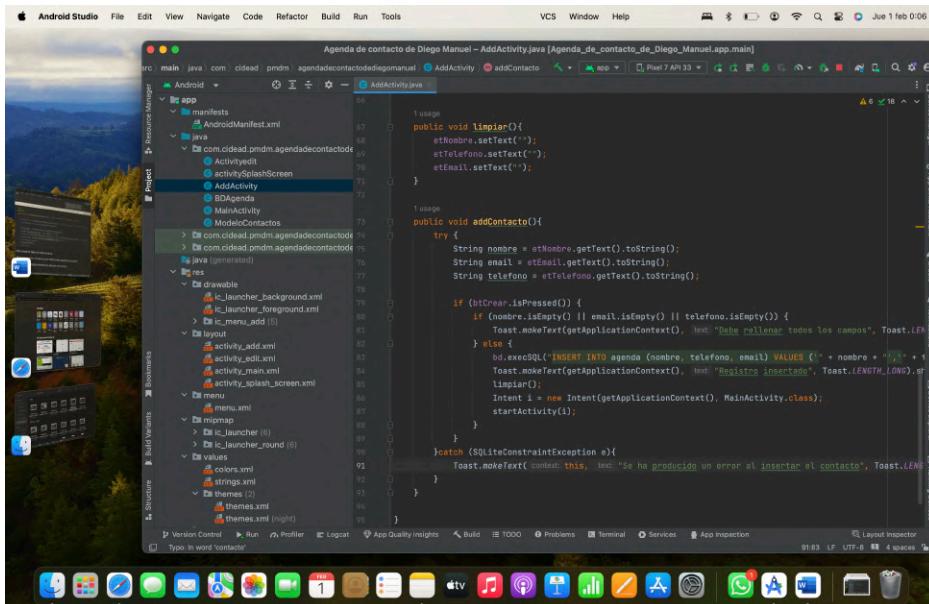


La clase addActivity contendrá un método llamado limpiar, que será el encargado de vaciar los editText.

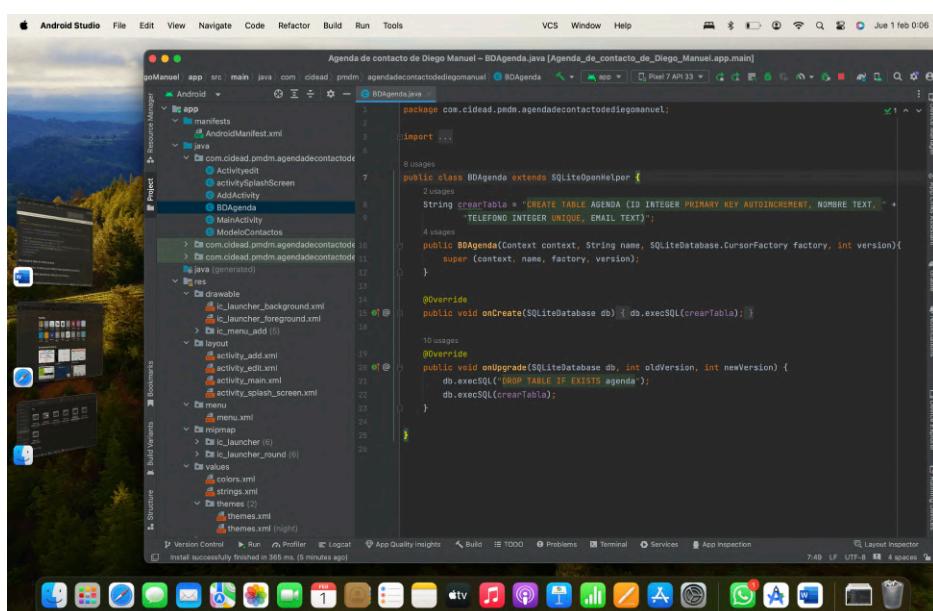
Dispondrá de un método llamado addContacto, el cual será el encargado de obtener los datos introducidos por el usuario en los editText y guardarlos en la base de datos.

Para ello contendrá tres atributos de tipo String llamados nombre, email y teléfono respectivamente.

Mediante un if comprobará si el usuario ha pulsado el botón crear, en caso de hacerlo, mediante otro if, comprobará si todos los campos han sido rellenos por el usuario, si no es así mostrará un mensaje mediante un Toast indicando que todos los campos deben ser rellenados, en caso de estar todos los campos rellenos, insertará en la base de datos, mediante el método execSQL, la sentencia necesaria para guardar los datos de contacto, mostrará en un toast el mensaje correspondiente, llamará al método limpiar, creará un nuevo intent que devolverá al usuario a la actividad principal y lo iniciará.



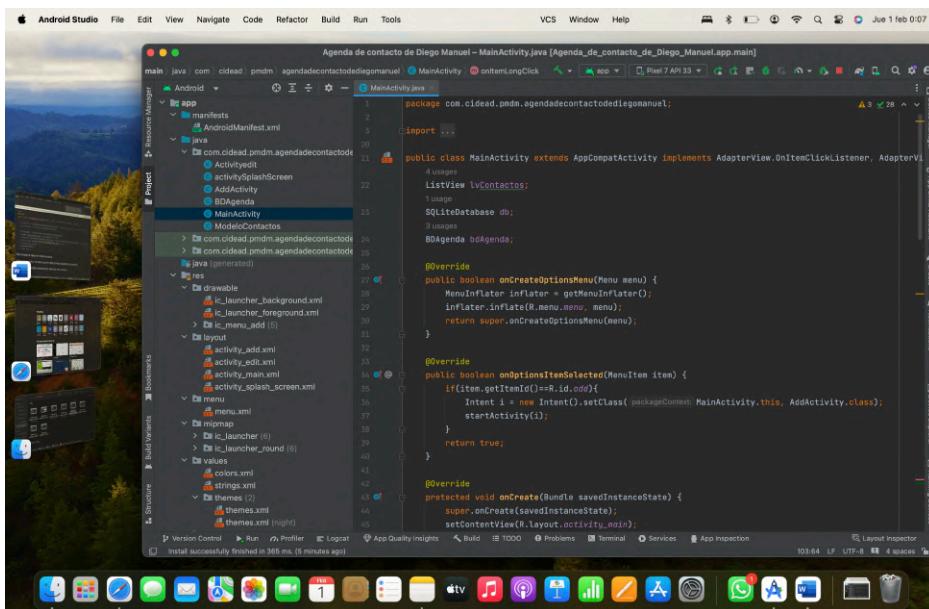
La clase BDAgenda contendrá la sentencia para crear la tabla Agenda con sus campos correspondientes, su método constructor, el método onCreate y el método onUpgrade, el cual será el encargado de borrar la tabla, si existe, y crear la tabla nueva.



El MainActivity contendrá tres atributos, uno de tipo ListView llamado lvContactos, otro de tipo SQLiteDatabase llamado db y otro de tipo BDAGenda llamado bdAgenda.

Tendrá un método llamado onCreateOptionsMenu, que creará un menú (el creado en los pasos anteriores).

Dispondrá de otro método llamado onOptionsItemSelected, que mediante un if comprobará si se ha pulsado el botón add del menú y si es así, creara un nuevo intent que llevará al usuario a la actividad addActivity e iniciará dicho intent.

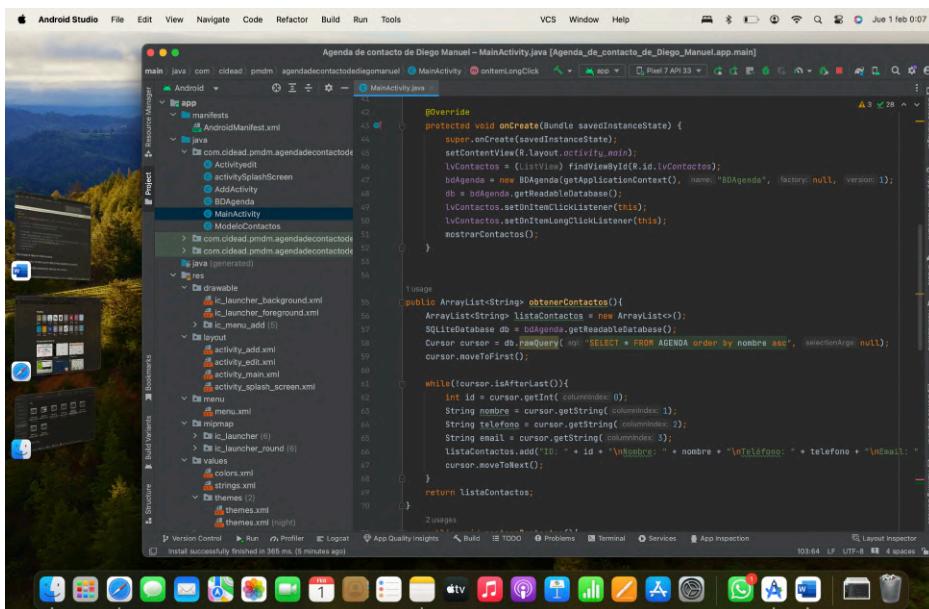


Dispondrá de un método llamado obtenerContactos, que será el encargado de extraer de la base de datos los contactos existentes y guardarlos en un ArrayList.

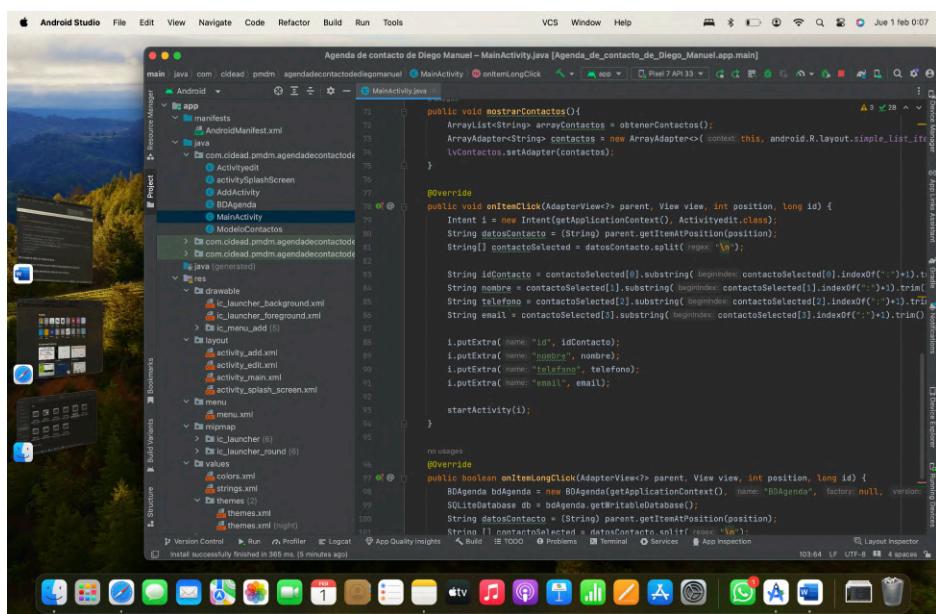
En dicho método crearemos un ArrayList de tipo string, una nueva conexión a la base de datos, la cual abriremos en modo lectura, un cursor que ejecutará la sentencia correspondiente para obtener todos los contactos y moveremos el cursor al primer resultado.

Dispondrá de un while, que mientras el cursor no este en el último resultado, mediante los métodos get (int, string, etc) obtendrá los resultados de cada columna y fila, los añadirá al ArrayList creado anteriormente y moverá el cursor al siguiente resultado.

Este método retornara el ArrayList.

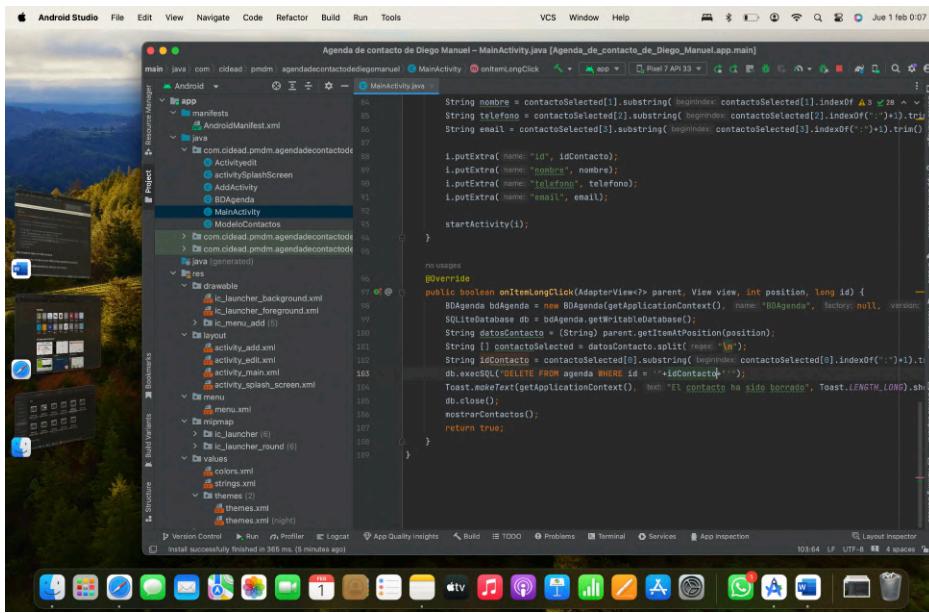


Contendrá un método onItemClick, que será el encargado de capturar, si se ha pulsado sobre algún item, el item seleccionado por el usuario, crear un nuevo intent hacia la actividad ActivityEdit, capturar del item el contenido del mismo y, mediante los métodos putExtra pasar dichos datos a la nueva actividad mediante el intent creado y por último, iniciará dicha actividad.



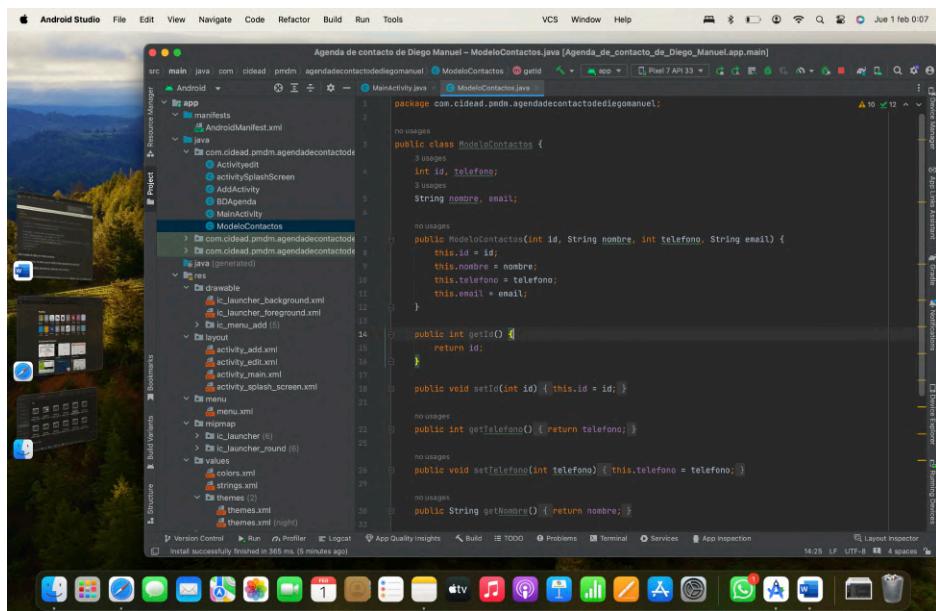
Dispondrá de un método onItemLongClick, que será el encargado de borrar el contacto sobre el cual se ha mantenido la pulsación.

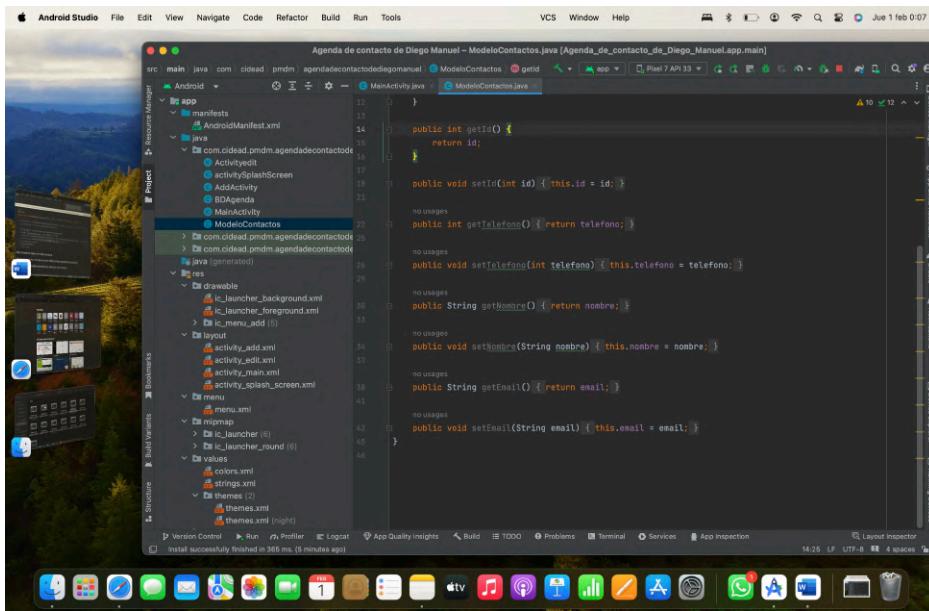
Para ello dispondrá de una nueva conexión a la base de datos, la cual abrirá en modo escritura, obtendrá los datos del item sobre el cual se ha mantenido la pulsación, los guardará en un array de tipo string, ejecutará la sentencia correspondiente para borrar dicho contacto, mostrará en un toast el mensaje correspondiente, cerrará la conexión a la base de datos y llamará al método mostrarContactos, para que muestre de nuevo los contactos que guarda la base de datos de forma actualizada.



La clase `ModeloContactos` dispondrá de dos atributos de tipo `int` llamados `id` y `telefono` respectivamente y otros dos de tipo `String` llamados `nombre` y `email` respectivamente.

Dispondrá de su método constructor con todos los parámetros y los `getter` and `setter`.





Y con esto damos por finalizada la tarea 7.2