5. Tu segunda Activity

En este capítulo ha sido creada la segunda activity de la app GeoQuiz. Esta segunda actividad servirá al usuario para hacer trampas al contestar el cuestionario y poder saber de cada pregunta su respuesta antes de contestarla.

La llamaremos CheatActivity, y tendrá el siguiente aspecto.

Cuando apretemos el botón de Show Answer nos mostrará el valor correcto de la respuesta de la pregunta, encima del botón. Y deberemos utilizar el botón de Atrás propio de Android para volver a la pantalla de la pregunta.

Si el usuario clica en Show Answer, además, cuando conteste la pregunta, el Toast que aparece diciendo si la respuesta ha sido correcta o no, mostrará un mensaje diciendo que está mal hacer trampas, tanto si la pregunta es correcta como si no.

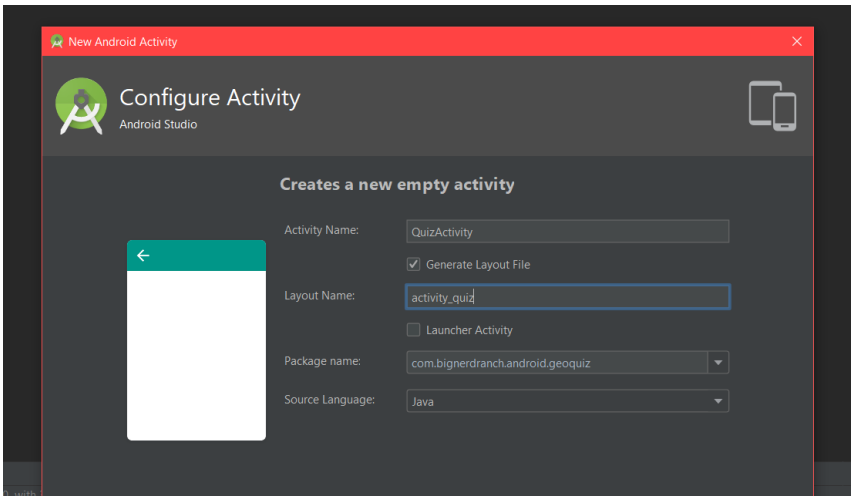
Este capítulo es de los más importantes del curso ya que en este aprenderemos a:

- Crear una nueva Activity y un nuevo Layout para ella.
- Abrir una nueva Activity desde otra Activity.
- Transferir datos entre una Activity y otra.

Creación de la nueva Activity

Hay mucho que hacer en este capítulo, pero el primer paso será definir en el archivo *res/values/strings.xml* TODAS las Strings que utilizaremos en la nueva Activity.

Una vez definidas todas las Strings, crearemos la nueva Activity. Para ello clicaremos con el botón derecho en el paquete donde tenemos la QuizActivity, y seleccionamos **New → Activity → Empty Activity.**



Le ponemos nombre y seleccionamos la opción de generar un archivo para su layout.

Por vuestra cuenta inspeccionar el código para ver como se ha diseñado el layout de la CheatActivity o intentad implementar el layout de esta segunda Activity por vuestra cuenta.

Como hemos seleccionado crear una nueva Empty Activity, la CheatActivity hereda de Activity, y por lo tanto también tiene el lifecycle callback **onCreate(Bundle)**, en el cual se construye la Activity y donde inflamos el layout de ésta.

Declaración de Activities en el Manifest

En el archivo AndroidManifest encontramos metadata que describe nuestra aplicación al sistema operativo Android.

Nota: **TODAS** las Activities de nuestra aplicación deben estar declaradas en el archivo Manifest.

```
<activity android:name=".QuizActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>

        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
<activity android:name=".CheatActivity">
</activity>
```

Se recomienda al alumno que investigue y lea con detenimiento el archivo AndroidManifest por su cuenta para adquirir algunos conocimientos destacables de la información principal de una aplicación Android.

Empezar una nueva Activity

La forma más simple de abrir una nueva Activity desde otra es mediante el método **startActivity(Intent)**. Este método es reenviado desde la Activity hacia el sistema operativo para que este cree la nueva Activity, ya que se tratará de un contexto distinto y debe ser controlado por el dispositivo.

La parte del sistema operativo que trata estas acciones es el **ActivityManager**.

En este punto, podríamos preguntarnos: ¿Cómo sabe el ActivityManager que Activity debe lanzar? Fácil, ya que ese dato lo encontraremos en el **Intent** que recibe el método startActivity.

El constructor del Intent tiene como parámetros:

- **Context packageContext**: Refiere en que "Application Package" la clase de la Activity puede ser encontrada.
- **Class<?> cls**: Clase (subclase de Activity) que lanzaremos.

Por lo tanto, cuando el usuario clique en el botón de hacer trampas, deberemos hacer lo siguiente:

```
mCheatButton = (Button) findViewById(R.id.cheat_button);
mCheatButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        boolean answerIsTrue = mQuestionBank[mCurrentIndex].isAnswerTrue();
        Intent intent = CheatActivity.newIntent( packageContext: QuizActivity.this, answerIsTrue);
        startActivityForResult(intent, REQUEST_CODE_CHEAT);
    }
});
```

De esta forma conseguimos crear un Intent mediante el método *newIntent* declarado en la clase CheatActivity:

```
public static Intent newIntent(Context packageContext, boolean answerIsTrue) {
    Intent intent = new Intent(packageContext, CheatActivity.class);
    intent.putExtra(EXTRA_ANSWER_IS_TRUE, answerIsTrue);
    return intent;
}
```

Como vemos, en el método **newIntent**, se crea un Intent de la clase recibida como primer parámetro (en este caso es la clase QuizActivity) hacia la clase CheatActivity.

Además, se transfiere información de una Activity a la otra mediante el método **putExtra** propio de la clase Intent. Con ella podremos transmitir datos de cualquier tipo entre dos Activities. El primer parámetro de este método es el TAG con el que identificaremos los datos, y el segundo parámetro son los datos en si.

En este punto, ya hemos conseguido lanzar una segunda Activity transmitiéndole datos mediante un Intent, pero **¿Cómo podemos obtener estos datos en la nueva Activity?**

Para poder obtener los datos que hemos introducido en el Intent mediante su método putExtra, deberemos utilizar la función **getBooleanExtra** (podemos sustituir "Boolean" por cualquier tipo de datos) en el método onCreate() de la CheatActivity.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cheat);

    mAnswerIsTrue = getIntent().getBooleanExtra(EXTRA_ANSWER_IS_TRUE, defaultValue: false);
```

¡Ya hemos conseguido obtener los datos transmitidos desde la QuizActivity!

Para saber si el usuario ha hecho trampas y ha mostrado la respuesta, hemos utilizado anteriormente la función **startActivityForResult** en vez de la función **startActivity**, ya que la diferencia entre ambas es que podemos devolver un código de error o de suceso, para así saber como ha sido la ejecución de la segunda Activity.

El primer parámetro de esta nueva función es el mismo Intent que enviábamos anteriormente, y el segundo parámetro es el código definido por nosotros REQUEST_CODE_CHEAT.

```
        startActivityForResult(intent, REQUEST_CODE_CHEAT);
```

Si nos fijamos en el código de ejecución de la CheatActivity, veremos que en el método onClick del Listener del botón de mostrar la respuesta, hacemos una llamada a una función que nos hemos declarado, llamada setAnswerShownResult.

```
private void setAnswerShownResult(boolean isAnswerShown) {
    Intent data = new Intent();
    data.putExtra(EXTRA_ANSWER_SHOWN, isAnswerShown);
    setResult(RESULT_OK, data);
}
```

En esta función vemos que:

1) **Creamos un Intent** al que llamamos data.
2) **Le introducimos** la **información** referente a si el usuario ha clicado en mostrar el resultado o no.
3) Hacemos una llamada a la función **setResult**, a la que le pasamos como parámetros:
    - **RESULT_OK**: Código estándar para informar a la Activity que ha levantado la Activity actual que se ha ejecutado todo correctamente.
    - **Data**: El Intent que hemos creado, **para informar a la QuizActivity** que el usuario ha hecho trampas.

Esta es la única opción que hemos programado para finalizar la CheatActivity, pero si el usuario clica el botón de atrás propio de Android, el sistema realizará un **setResult(RESULT_CANCELED)**.

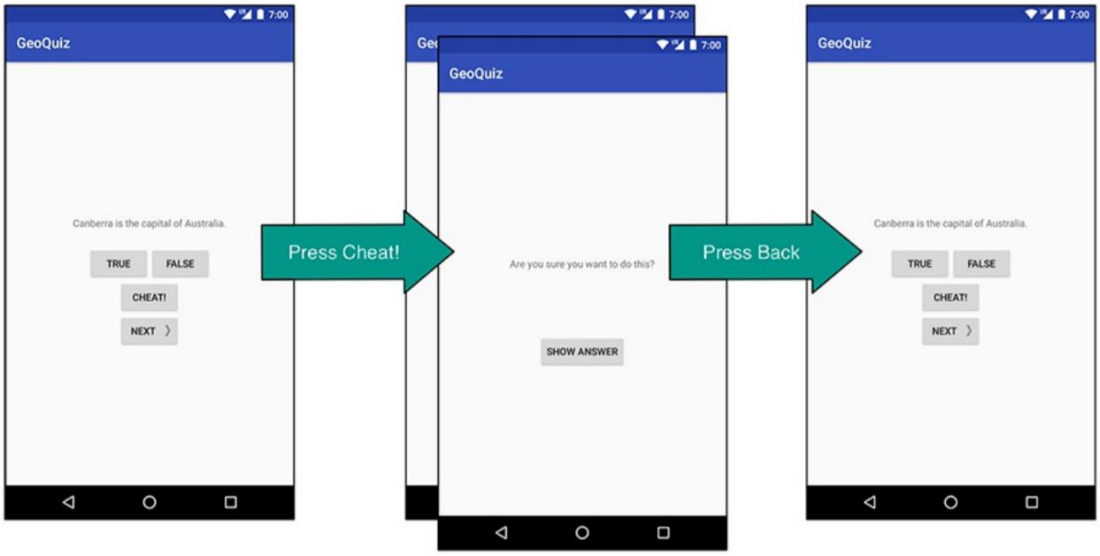
Para recoger la información que la CheatActivity nos ha devuelto, en la QuizActivity deberemos sobrescribir el método **onActivityResult**, que será llamado cuando una Activity lanzada por la QuizActivity termine.

```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode != Activity.RESULT_OK) {
        return;
    }

    if (requestCode == REQUEST_CODE_CHEAT) {
        if (data == null) {
            return;
        }
        mIsCheater = CheatActivity.wasAnswerShown(data);
    }
}
```

Ahora, en la QuizActivity, ya sabemos si el usuario ha hecho trampas o no. Tan sólo deberemos modificar los Toasts, que mostramos al responder una pregunta, para dar por terminado este capítulo.

Por último, un pequeño esquema del estado del **stack** en el momento de mostrar cada Activity.



Nota: Una llamada al método **Activity.finish()** en la CheatActivity también finalizaría la actividad y, lógicamente, también la sacaría del stack. Es otra forma de terminar una actividad.

Challenges

Para los alumnos más curiosos se les anima a solucionar los siguientes bugs, que permiten a los usuarios más tramposos de la aplicación salirse con la suya, ya que éstos son unos grandes tramposos. A continuación, se detallan los bugs en orden ascendente de dificultad.

- El tramposo puede rotar la pantalla en la CheatActivity después de haber hecho trampas para así limpiar el resultado de si es tramposo o no.
- El tramposo, una vez ha vuelto a la pantalla QuizActivity, puede rotar la pantalla para limpiar el atributo de si es un tramposo o no.
- El tramposo puede clicar en NEXT hasta que la pregunta en la que ha hecho trampas vuelva a aparecer.