Intefaz de usuario - Ejercicios

Índice

1 Formulario..... 2
2 Recoger datos del formulario mediante SharedPreferences..... 2
3 Validación del DNI..... 3
4 Visor de Google Maps..... 4
5 Barra de progreso en el título..... 6
6 Menú de preferencias..... 8

1. Formulario

Vamos a crear un formulario para pedir cita para la renovación del DNI. (La cita es ficticia, y el programa no va a realizar ninguna petición ni va a transmitir los datos introducidos). Seguiremos los siguientes pasos:

- En Eclipse creamos un nuevo proyecto de Android que se llame `ADJM_S10_1`. Seleccionamos el nivel de API 8, nombre de la aplicación el mismo que el del proyecto, el nombre del paquete `es.ua.jtech.ajdm.s10`, y el nombre de la actividad principal, `Formulario`.
- En el layout `main.xml` tenemos que modificar el XML para que soporte la barra de desplazamiento vertical. Esto se puede conseguir con un `ScrollView` y dentro el layout deseado, por ejemplo el `LinearLayout`. Dentro de él creamos el siguiente formulario:

Formulario (parte superior e inferior).

- Usar el atributo `android:ems` de los `TextView` para hacer que las etiquetas de nombre y apellidos tengan el mismo ancho y así los campos de texto empiecen alineados.
- Añadir `hints` a algunos campos de texto.

2. Recoger datos del formulario mediante SharedPreferences

Al pulsar "Siguiente", vamos a pasar a otra pantalla que muestra un resumen con la información del formulario.

- Crear la nueva `Activity` con el nombre `Resumen`. Registrarla en el `AndroidManifest.xml`.
- Crear un nuevo layout en `res/layout`, llamado `resumen.xml` (no puede haber mayúsculas) y añadir el `TextView` donde se mostrará el resumen.
- Crear el `Intent` con la actividad al pulsar el botón "Siguiente" e iniciar la actividad.
- Sobrecargar el método `Resumen.onCreate(...)` y programar la puesta del resumen en el campo de texto. Un ejemplo sería: "Nombre Apellido1 Apellido2 con DNI 999999999-F solicita cita para el día 00/00/00 a las 00:00." Para ello hay que recoger los datos de la anterior actividad, `Formulario`.

¿Cómo pasar los datos de una actividad a otra?

- Aunque existen diferentes maneras (Intent extras, campos estáticos de la clase, getters o setters públicos, clase Singleton, Preferences, grabar en archivo, proveedores de contenidos, base de datos SQLite, etc), en este ejercicio vamos a utilizar las `SharedPreferences`.
- Al pulsar "Siguiente" tendremos que grabar las preferencias, con etiqueta `preferenciasDNI`. Nos hará falta usar también un `PreferencesEditor`.
- Al crearse la nueva actividad `Resumen` tendremos que leer las preferencias compartidas. Una vez leídas, podremos mostrarlas en layout `resumen.xml`.

3. Validación del DNI

El campo del DNI podría validarse o bien durante la escritura, o bien sólo cuando se pulse el botón "Siguiente". Vamos a hacer las dos cosas:

- Durante la introducción del DNI en el `EditText` vamos a validar con dos filtros:
  - Un filtro nuevo que compruebe que lo introducido cumpla la siguiente expresión regular: `"^([0-9]{0,7}|[0-9]{8,8}[a-zA-Z]{0,1})$"`.
  - El filtro `InputFilter.AllCaps()` que convertirá la letra en mayúscula.
- La anterior expresión regular no comprueba que la letra sea la correcta. Esto lo haremos sólo cuando se pulse "Siguiente", y la funcion que lo compruebe se llamará `boolean compruebaLetraDNI(String dni)`, listada a continuación.

```
                private boolean compruebaLetraDNI(String dni) {
                        try{
                                int num =
Integer.parseInt(dni.substring(0, 8));
                                char letra =
"TRWAGMYFPDXBNJZSQVHLCKE".charAt(num % 23);
                                if(dni.charAt(8)!=letra)
                                        return false;
                        }catch(Exception e){
                                Log.i("ValidacionDNI",
```

```
e.getClass().toString());
                                return false;
                        }
                        return true;
                }
```

En caso de que la letra no sea correcta, mostraremos un `Toast` con el texto "Letra del DNI incorrecta" y no pasaremos a la siguiente actividad sino que, quedándonos en la misma, pondremos el foco sobre el `EditText` del DNI.

- Si el DNI es incorrecto al pulsar "Siguiente", enfocarlo usando el método `requestFocus()` de su `EditText` correspondiente. Así llamaremos la atención al usuario sobre el campo que está fallando.

Toast: Letra del DNI incorrecta.

4. Visor de Google Maps

Vamos a añadir un visor de Google Maps al layout `resumen.xml` de los ejercicios anteriores. Después añadiremos un selector de oficinas para que, según la oficina seleccionada, cambien las coordenadas del mapa. Vamos a seguir los siguientes pasos:

- Para usar el MapView desde nuestra aplicación necesitamos obtener una clave para la API de Google Maps. Ésta se genera a partir de la huella digital MD5 del certificado

digital que usamos para firmar nuestras aplicaciones. Para el desarrollo será suficiente con que utilicemos el certificado de debug que se crea por el Android SDK para desarrollar. Sin embargo éste no nos servirá para producción. Para generar la clave para la API debemos seguir los siguientes pasos:

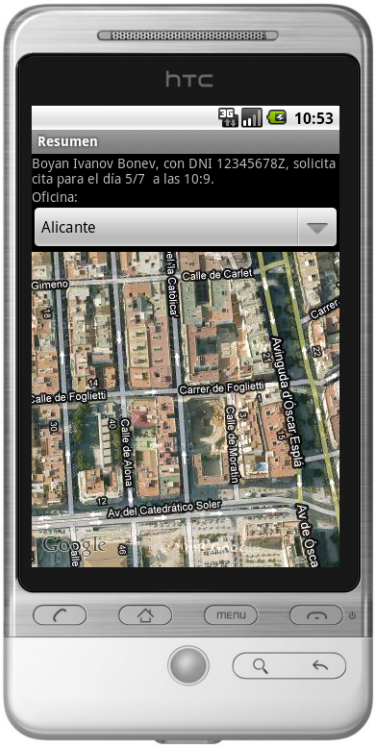
- Generar una huella digital MD5 para el certificado de debug. Encuéntralo en un fichero llamado debug.keystore, cuya ruta está indicada en Eclipse en las preferencias de Android Build. Desde el directorio indicado en la ruta, ejecutamos:

```
keytool -list -keystore debug.keystore
```

- Entramos en http://code.google.com/android/add-ons/google-apis/maps-api-signup.html y rellenamos el formulario. Para ello tenemos que autenticarnos con nuestra cuenta de Google. Es ahí donde pegaremos nuestra huella digital. Finalmente nos muestra la clave de la API, la copiamos y nos la guardamos.

Más información en http://code.google.com/android/add-ons/google-apis/mapkey.html

- Editar el `AndroidManifest.xml` e indicar que la aplicación usa la librería de google maps, y que requiere permisos para Internet.
- Añadir el mapa en el layout `resumen.xml`, debajo del campo de texto, ocupando todo el ancho posible de la pantalla. En la declaración del componente introduciremos la clave para la API que hemos obtenido.
- En `Resumen.onCreate()` obtener la referencia al mapa y crear un controlador `MapController` a partir del `MapView` para poder moverlo. Poner zoom por defecto 18, y permitir que el usuario cambie el nivel de zoom.
- Añadir un `Spinner` entre el campo de texto y el mapa. Asociarle un `ArrayAdapter` pasándole el `String[] oficinas={"Alicante","Elche","San Vicente"}`.
- Asignar al `Spinner` un nuevo `OnItemSelectedListener` que, según si la ciudad seleccionada, moverá el mapa a las siguientes coordenadas:
  - Alicante: longitude = -0.495232; latitude = 38.340688
  - Elche: longitude = -0.685809; latitude = 38.26189
  - San Vicente: longitude = -0.52081; latitude = 38.401739



Visor de Google Maps en nuestra aplicación.

- Opcional: Investigar cómo colocar una marca de Google Maps en las coordenadas correspondientes.

5. Barra de progreso en el título

El caso más común que requiere indicar el progreso es el de operaciones de transferencia por red. En este ejercicio vamos a cargar una página web en el lugar del mapa (eliminando dinámicamente el mapa y añadiendo un visor web) y vamos a aprovechar el título de la ventana Resumen para indicar en éste el progreso de carga de la web. Seguiremos los siguientes pasos:

- Añadimos un botón "Web..." al lado del `Spinner` y asociamos a ese botón un `onClickManager` que se encargará de eliminar del Layout `resumen.xml` todos los componentes salvo el `TextView` que contiene el resumen. Para eliminarlos necesita una referencia al Layout correspondiente, pero éste no tiene identificador. Se lo añadimos:

```
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:id="@+id/layout_resumen" >
```

```
  ...
```

De esta manera podemos obtener una referencia a este objeto:

```
LinearLayout resumenLayout =
(LinearLayout)findViewById(R.id.layout_resumen);
```

y eliminar los `View` que queramos con:

```
resumenLayout.removeView(mapView);
```

- Una vez eliminados los `View` innecesarios, tenemos que añadir el `WebView` (que debemos crear) con el método:

```
WebView webView = new WebView(getApplicationContext());
resumenLayout.addView(webView);
```

Nota:

En este caso, en lugar de encontrar el componente buscándolo por su identificador, lo hemos creado porque no lo tenemos definido en ningún Layout. Podíamos haber creado otro `.xml` que definiera el WebView con un identificador, y obtenerlo de ahí.

- Configuramos el `WebView` y le indicamos qué URL cargar:

```
webView.getSettings().setJavaScriptEnabled(true);
                    webView.setInitialScale(60);
                    webView.loadUrl(
"http://www.dnielectronico.es/obtencion_del_dnie/renovar.html");
```

- Habilitamos la barra de progreso en el título de la ventana. Debemos solicitar esta característica al principio del método `onCreate()`, antes de ejecutar el `setContentView()`. Además vamos a guardarnos una referencia a la actividad actual (Resumen) para acceder de forma más fácil a los métodos del progreso.

```
public class Resumen extends MapActivity {
    ...
        Activity a;

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

            requestWindowFeature(Window.FEATURE_PROGRESS);
                a = this;  //Guardarme una referencia a la
                                    //actividad para el progress bar

            setContentView(R.layout.resumen);
        ...
```

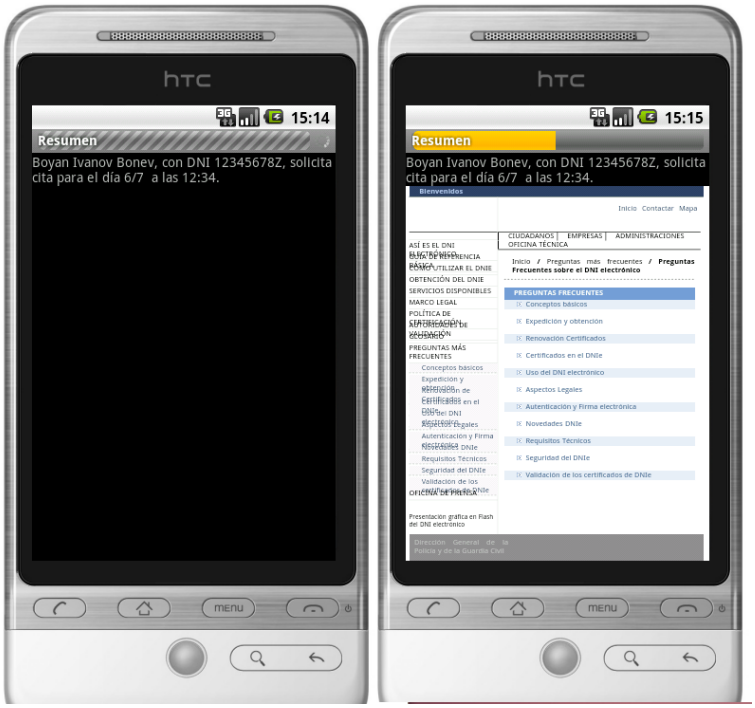
- Una vez habilitada la barra de progreso necesitamos, en primer lugar, ponerla visible cuanod se vaya a iniciar la carga, con:

```
                a.setProgress(0);
```

```
                a.setProgressBarVisibility(true);
```

y en segundo lugar añadir al `WebView` manejadores que la actualicen cada vez que cambie el progreso de carga de la página y cuando la carga termine:

```
            webView.setWebChromeClient(new WebChromeClient() {
                public void onProgressChanged(WebView view, int
progress) {
                    a.setProgress(progress*100);
                }
              });
            webView.setWebViewClient(new WebViewClient(){
                    public void onPageFinished(WebView view, String
url){
                            super.onPageFinished(view, url);
                            a.setProgressBarVisibility(false);
                    }
            });
```



Barra de progreso del visor web cargando.

6. Menú de preferencias

En este ejercicio vamos a crear para el "Formulario" el menú que aparece al pulsar el botón "menu" que el móvil tiene por hardware. Habrá dos opciones: "Preferencias" y "Acerca de...". Para el primer caso crearemos un menú de preferencias. Seguiremos los siguientes pasos.

- Crearemos un nuevo Android XML de tipo menu, llamado `menu.xml` y lo editaremos con el IDE para añadirle dos items. A cada uno de los items le tendremos que asignar un título: "Preferencias" y "Acerca de...".
- Para que el menú se infle al pulsar el botón debemos sobrecargar la función `Formulario.onCreateOptionsMenu(Menu m)` que deberá inflar el menú con la función `getMenuInflater().inflate(R.menu.menu, menu);` y devolver `true`.
- Para que las opciones del menú se ejecuten con la pulsación por parte del usuario tenemos que sobrecargar la función `Formulario.onOptionsItemSelected( ... )` y comprobar qué item del menú ha sido seleccionado. Para el de "Acerca de..." mostraremos un `Dialog` que debemos 1) crear como objeto, 2) ponerle título, 3) ponerle un nuevo campo de texto pasándoselo como parámetro a la función `dialog.setContentView(...)` y 4) mostrarlo con `dialog.show()`.
- Para el caso de la opción "Preferencias" vamos a crear una nueva `Activity` para el menú de Preferencias:

```
                Intent i = new Intent(Formulario.this,
Preferencias.class);
                startActivity(i);
```

  La actividad Preferencias la debemos crear y en su método `onCreate(...)`, tras la llamada al `super`, cargamos las preferencias

```
                addPreferencesFromResource(R.xml.preferences);
```

  a partir del recurso xml que debemos, una vez más, crear, como se indica en el siguiente paso.
- Podemos crear el recurso XML con la opción de crear nuevo Android XML de tipo Preferences. Utilizando el editor del IDE podemos añadir separadores de categorías e items. El XML utilizado para generar el menú de la imagen que se muestra más abajo, es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
<PreferenceCategory android:title="Validar DNI en:">
        <CheckBoxPreference
        android:title="en el campo"
        android:summary="Validará la introducción de números y una
letra"
        android:key="validacampo"></CheckBoxPreference>
        <CheckBoxPreference
        android:title="al pulsar"
        android:summary="Comprobará también que la letra sea la
correcta"
        android:key="validaboton"></CheckBoxPreference>
</PreferenceCategory>
<PreferenceCategory android:title="Otras preferencias:">
        <CheckBoxPreference android:enabled="false"
        android:title="Otra, deshabilitada"
        android:key="otra"></CheckBoxPreference>
```

```
</PreferenceCategory>
</PreferenceScreen>
```

Menú de preferencias

- Una vez creada la pantalla de preferencias, éstas se guardarán automáticamente en las preferencias por defecto del contexto de la aplicación,

```
                SharedPreferencias prefValidacion =
PreferenceManager.
getDefaultSharedPreferences(getApplicationContext());
```

  Los valores de las preferencias se pueden recoger con el método `prefValidacion.getBoolean("validacampo",true)`, donde el el primer parámetro debe coincidir con la clave del campo, y el segundo parámetro es el valor por defecto que se obtendría en caso de que esta preferencia todavía no tenga un valor.
- Podemos declarar como campos del Formulario dos variables booleanas, `validacampo` y `validaboton`, cuyos valores tendremos que actualizar con el valor de las preferencias, cada vez que el usuario las cambie. Para escuchar cuándo cambian las preferencias tenemos que implementar un `OnSharedPreferenceChangeListener` como se lista a continuación:

```
prefValidacion.registerOnSharedPreferenceChangeListener(
        new OnSharedPreferenceChangeListener() {
```

```
                @Override
                public void onSharedPreferenceChanged(
                  SharedPreferences sharedPreferences, String key)
                {
                        //actualizar mis campos con las nuevas
preferencias
                }
        });
```

- No debemos olvidar cambiar la lógica de validación, tanto en el campo como en el botón, para que no valide en caso de que así se indique en los campos booleanos correspondientes.