Trabajo Final
YourWindow

Diego Vicente Villagrasa

Junio 2021

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Aplicaciones y Usabilidad

Índice

1. Introducción

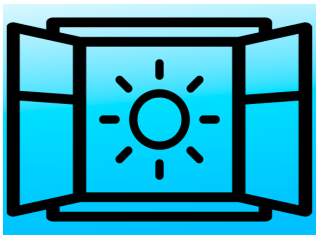

Figura 1: Logo YourWindow

Esta es la memoria explicativa de la práctica final de Aplicaciones y Usabilidad. En este documento se explica para que y como se utiliza la aplicación YourWindow, como ha sido programada, y se detalla algunas funciones especiales añadidas tanto a la UI como al propio código. Todo esto incluyendo diferentes Snippets e imágenes para complementar la explicación.

2. Manual explicativo

La aplicación YourWindow se trata de una app para la consulta del pronóstico del tiempo, separado por municipios de España. La experiencia que se quiere conseguir con la aplicación, como su propio nombre indica, es recibir la información del tiempo tan rápido y de forma tan simple como asomarse a la ventana. YourWindow es capaz de dar la temperatura, la máxima, la mínima, y diferentes datos de interés como probabilidad de precipitación, o velocidad del viento, a un sólo botón.

Lo primero que se observará al abrir la aplicación, es el entorno de búsqueda de municipio, donde el usuario podrá encontrar cualquier municipio dentro del territorio español. Tras encontrar la localidad, el usuario deberá seleccionar entre el mismo día en el que está realizando la búsqueda, el día siguiente, o dos días en adelante al presente. Por ejemplo, si el usuario busca el pronóstico un lunes, podrá ser capaz también de consultar el tiempo los días martes y miércoles.

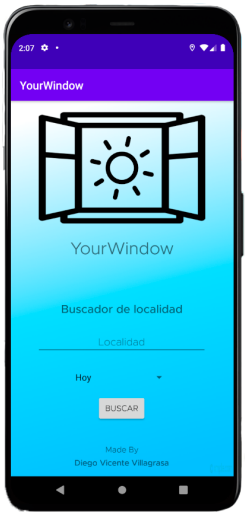

Figura 2: Buscador YourWindow

Cuando el usuario este conforme con su elección, podrá darle al botón de buscar. Entonces saltará a la actividad donde se da la previsión del tiempo de la localidad y el día seleccionados por el usuario. Desde esta misma interfaz, el usuario podrá pulsar el botón "Más información", para acceder a más detalles, como velocidad del viento, dirección del viento, y probabilidad de precipitación.

Figura 3: Main Activity YourWindow

3. Descripción algoritmica

El programa consta de dos actividades, una que se encarga de la tratado,búsqueda y envío a la actividad siguiente de la localidad elegida del usuario, y otra que se encarga de mostrar la información del tiempo. La primera se le refecrencia como Buscador, y la segunda como MainActivity, las cuales se explican brevemente a continuación.

3.1. Buscador

3.1.1. Buscador.java

La clase buscador es la encargada de leer todos los municipios de españa, mostrar al usuario las herramientas para que pueda seleccionar lo que se quiere buscar, y pasar esta información al MainActivity.

Para poder conseguir todos los municipios de España, se ha utilizado uno de los archivos proporcionados en la práctica 9 de la asignatura, el cual era un archivo excel el cual contenía todos los municipios del país, junto con su código el cual se utiliza para buscar los datos del tiempo.

Para conseguir utilizar este archivo, lo primero que se hizo fue convertirlo a JSON, y posterior a esto añadirlo a la aplicación en la carpeta assets. Ahora la aplicación lee el JSON a través de la función *loadJSONFromAsset()*, el cual devuelve el JSON en un string.

```
public String loadJSONFromAsset() {
    String json = null;
    try {
        InputStream is = getAssets().open("codMun.json");
        int size = is.available();
        byte[] buffer = new byte[size];
        is.read(buffer);
        is.close();
        json = new String(buffer, "UTF-8");
    } catch (IOException ex) {
        ex.printStackTrace();
        return null;
    }
    return json;
}
```

Listing 1: Función loadJSONFromAsset()

El siguiente paso es la función *readCode*, la cual se ocupa de leer la raiz del JSON, y de crear un diccionario para facilitar el uso de los datos. El diccionario (HashMap), está formado por las claves (los municipios) y los valores (los códigos de los municipios). De esta forma, se puede buscar fácilmente el código del municipio elegido por el usuario, para entonces pasarlo a MainActivity y mostrar los datos del tiempo.

```
public void readCode(){
    try {
        JSONArray raiz = new JSONArray(loadJSONFromAsset());
        munList = new HashMap<String,String>();

        for(int ind=0; ind<= raiz.length()-2 ; ind++){
            JSONObject codMunlist = raiz.getJSONObject(ind);
            String ciudad = codMunlist.getString("NOMBRE");
            municipios.add(ciudad);
            String cpro = codMunlist.getString("CPRO");
            String cmun = codMunlist.getString("CMUN");
            String codigo = cpro+cmun;
            munList.put(ciudad,codigo);
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Listing 2: Función readCode()

Por último, la funcionalidad que permite que el usuario pueda elegir el municipio y día, se ha incluido un AutoCompleteTextView, y un Spinner, respectivamente.
Cuando el usuario pulse el botón buscar, se activará el OnClickListener, el cual se ocupa de enviar a través de un Intent los datos del código de localidad y el día previamente seleccionados por el usuario.

```
            localidad = atv.getText().toString();
            if (munList.containsKey(localidad)){
                codigoLocalidad = munList.get(localidad);
                Log.i("DIA DE LA SEMANA", dia);
                Intent intent = new Intent(Buscador.this,MainActivity.class);
                intent.putExtra("codigoLocalidad", codigoLocalidad);
                intent.putExtra("diaSemana", numeroDia);
                startActivity(intent);
```

Listing 3: Contenido dentro del onClicl()

3.1.2. activity buscador.xml

En este archivo xml se define la UI del buscador (Figura 2). Lo primero destacable es la inclusión de un fondo, que se encuentra en la carpeta *drawable*. Después de esto, se encuentran dos ImageView, los cuáles forman el logo de la aplicación, además dos simples TextView, uno con el nombre de la app, y otro señalando el Buscador de localidad.

Después de los elementos estéticos, el primer elemento funcional es el AutoCompleteTextView, elemento el cuál permitirá al usuario introducir la localidad que desean buscar. Además, a medida que se vaya escribiendo, dará la opción de autocompletar el nombre del pueblo. Esto es realizado gracias a la lista *municipios*, rellenada en la misma función que *munList* (el diccionario con las claves y valores de ciudad/código).

Tras esto encontramos el Spinner con el cual el usuario elegirá el día del cual quiere consultar el tiempo. El resultado de este se enviará a través del Intent anteriormente mencionado con la variable *dia*.

Por último, tendremos el botón de buscar, el cuál activará el Intent para pasar a la siguiente actividad, y mandará los datos de la localidad y día elegidos.

3.2. MainActivity

3.2.1. MainActivity.java

La clase MainActivity es la encargada de mostrar los resultados de la búsqueda anteriormente realizada.

Para ello lo primero a realizar será un *getIntent()*, con el cuál se recibirán los datos del código del municipio y el día seleccionados por el usuario. Ahora con el código del municipio, podremos construir la variable URL, la cual permite acceder al primer JSON necesario para conseguir los datos del tiempo deseados.

```
1         Intent intent = getIntent();
2         cLocalidad = (String) getIntent().getStringExtra("codigoLocalidad");
3         dSemana = (Integer) getIntent().getIntExtra("diaSemana",0);
4         Log.i("NUMERO DIA SEMANA", String.valueOf(dSemana));
```

Listing 4: Recepción de datos del Intent

Los pasos seguidos para conseguir esto, es idéntico al de la práctica 9. Primero se pasará esta URL a la función *ServiciosWebEncadenados()*, la cuál utilizará a su vez la función *API REST()*. El resultado de todo esto, es el JSON necesario para poder consultar los datos.

Se pasa entonces a la función *printParametros()*, la cuál si que se ha modificado ligeramente. La base es la misma que se hizo en la práctica 9, pero ahora no es necesario leer las franjas horarias que si se necesitaron a lo largo del curso. En YourWindow, se lee la raiz del JSON, y de este pasamos al objeto "prediccion". En el siguiente paso, se encuentra uno de los cambios incluidos. Para leer el día, se pasa la variable dSemana, la cual proviene del Intent realizado en la clase Buscador. Entonces después de esto, se extraen todos los datos necesarios, y se realizan diferentes *setText* para que se muestren los datos en la UI.

3.2.2. activity main.xml

Los primeros elementos que se encontrará el usuario son las variables *resPueblo* y *resTempe*, los cuáles son el resultado de la búsqueda para la localidad (conseguida a través del Intent) y la temperatura.

Después se encuentra la temperatura máxima y mínima de ese día, junto a dos ImageView de una flecha roja, y otra azul, para que se pueda comprender facilmente la información que está siendo vista.

Los siguientes dos elementos están relacionados con el estado del cielo. Uno es un ImageView, el cual cambiará dependiendo del estado del mismo, y el propio resultado conseguido en la función *printParametros()*.

Por último, el usuario encontrará el botón de más información. Cuando este botón sea pulsado, desvelará la probabilidad de precipitación, la direccción del viento, y la velocidad del viento. Para realizar esto, se ha creado un ConstraintLayout dentro del ConstraintLayout principal. También se han puesto todos los elementos de este ConstraintLayout en gone, menos el botón. Cuando el botón se pulse, se activará su OnClickListener, el cuál hará que si los elementos están en gone, aparezcan o viceversa.

```
        animBtn.setOnClickListener(new View.OnClickListener() {
            boolean visible;
            @RequiresApi(api = Build.VERSION_CODES.KITKAT)
            @Override
            public void onClick(View view) {
                TransitionManager.beginDelayedTransition(tContainer);
                visible = !visible;

                infPrecipitacionT.setVisibility(visible ? View.VISIBLE: View.GONE)
                    ;
                infdirVientoT.setVisibility(visible ? View.VISIBLE: View.GONE);
                infvelVientoT.setVisibility(visible ? View.VISIBLE: View.GONE);

                probPrecipitacionT.setVisibility(visible ? View.VISIBLE: View.GONE
                    );
                dirVientoT.setVisibility(visible ? View.VISIBLE: View.GONE);
                velVientoT.setVisibility(visible ? View.VISIBLE: View.GONE);
            }
        });
```

Listing 5: Código en MainActivity.java para el funcionamiento de la animación

3.3. Aspectos de la aplicación a mencionar

Para el desarrollo y funcionamiento de la aplicación se han tenido que crear y modificar diferentes directorios. Por ejemplo, se ha creado la carpeta *anim*, dentro de la carpeta res, la cual contiene el archivo *translate anim*, un archivo xml que contiene la animación que se aplica a los símbolos mostrados en MainActivity (menos el símbolo del sol, el cual se le aplica una *RotateAnimation*).
Otro de los directorios modificados, ha sido la carpeta drawable, donde se han incluido todas las imágenes utilizadas, y el fondo utilizado en la aplicación.

También es destacable el icono dinámico del estado del cielo de la actividad ActivityMain. Para poder lograrlo, se ha clasificado todas las posibilidades de estado del cielo en 5 tipos; sol, nuboso, lluvia, tormenta y nieve. Cuando la aplicación ejecute *printParametros()*, se realizará el siguiente if, donde se elegirá el símbolo adecuado, aparte de añadir una animación.

```
            final ImageView iconoTiempo = (ImageView) findViewById(R.id.iv_image);
            translateAnim = AnimationUtils.loadAnimation(this,R.anim.
                translate_anim);
            //Animacion
           if(Arrays.asList(tiempoSol).contains(estCielo)){
               RotateAnimation anim = new RotateAnimation(0.0f, 360.0f, Animation.
                   RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF, 0.5f);
               anim.setInterpolator(new LinearInterpolator());
               anim.setRepeatCount(Animation.INFINITE);
               anim.setDuration(15000);

               // Start animating the image

               iconoTiempo.startAnimation(anim);

           }else if (Arrays.asList(tiempoNuboso).contains(estCielo)){
                iconoTiempo.setImageResource(R.drawable.cloudy);
                iconoTiempo.startAnimation(translateAnim);
```

```
17              }else if (Arrays.asList(tiempoLluvia).contains(estCielo)){
18                  iconoTiempo.setImageResource(R.drawable.rain);
19                  iconoTiempo.startAnimation(translateAnim);
20              }else if (Arrays.asList(tiempoTormenta).contains(estCielo)){
21                  iconoTiempo.setImageResource(R.drawable.heavyrain);
22                  iconoTiempo.startAnimation(translateAnim);
23              }else if (Arrays.asList(tiempoNieve).contains(estCielo)){
24                  iconoTiempo.setImageResource(R.drawable.snowflake);
25                  iconoTiempo.startAnimation(translateAnim);
26              }
```

Listing 6: Código en MainActivity.java para el funcionamiento del icono dinámico