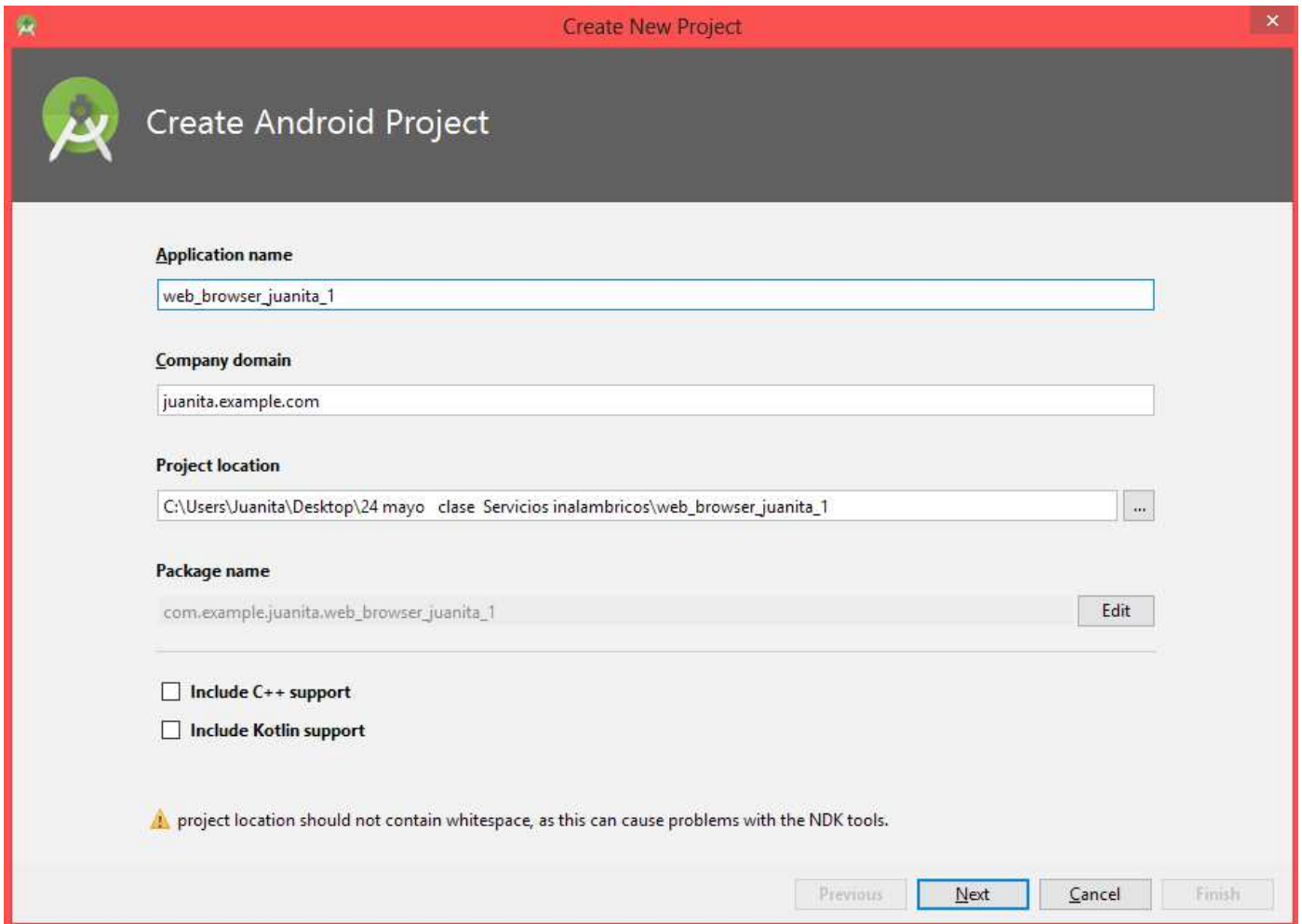


Lo primero que vamos a hacer es crear un nuevo proyecto en Android Studio. El proyecto tendrá las siguientes características:

- App Name: Ejemplo de Web_browser_juanita_1
- Min SDK: **API 14** (aunque podemos poner una inferior ya que las WebView llevan presentes desde API 5, Android 2.0)
- Activity: **Empty Activity**



Application name

web_browser_juanita_1

Company domain

juanita.example.com

Project location

C:\Users\Juanita\Desktop\24 mayo clase Servicios inalambricos\web_browser_juanita_1

Package name

com.example.juanita.web_browser_juanita_1

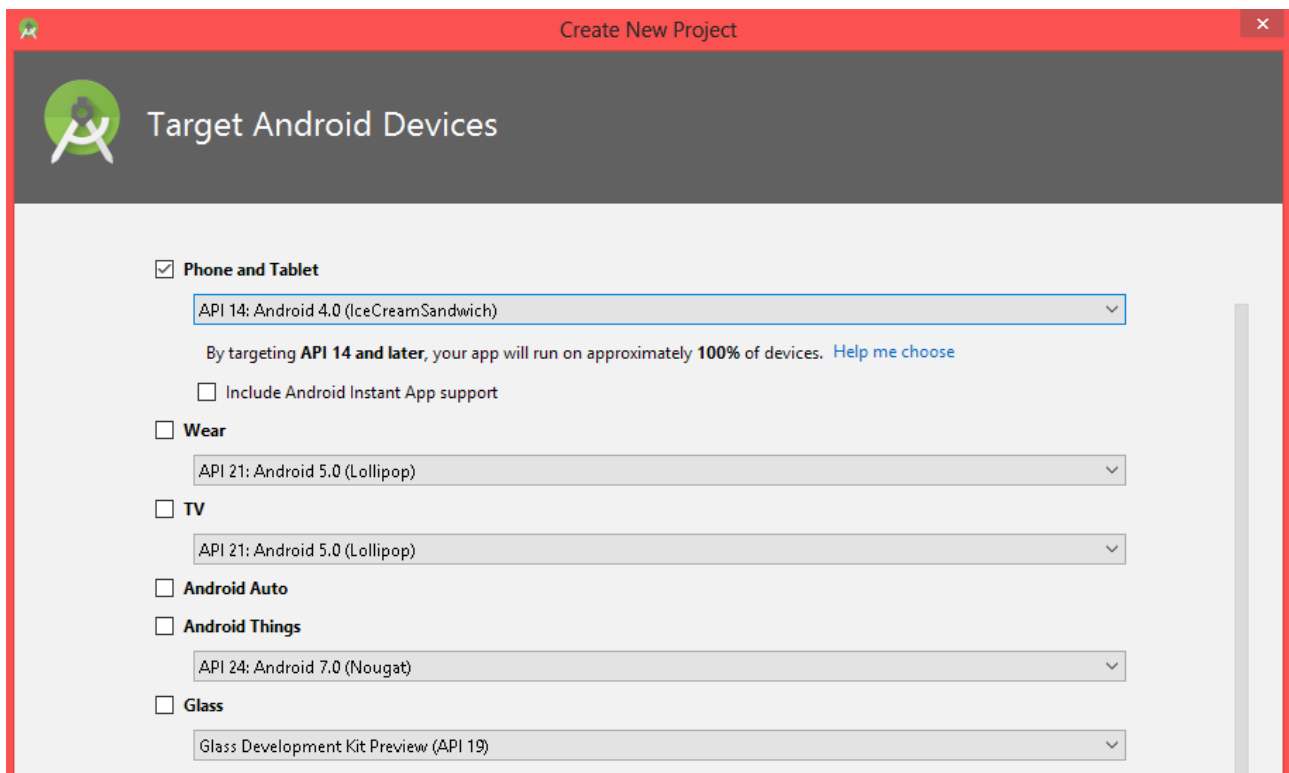
☐ Include C++ support

☐ Include Kotlin support

⚠ project location should not contain whitespace, as this can cause problems with the NDK tools.

Previous Next Cancel Finish

Si lo deseas, puedes cambiar la ubicación del proyecto, pero deja las demás opciones como están .Puedes poner el nombre que quieras a la aplicación.

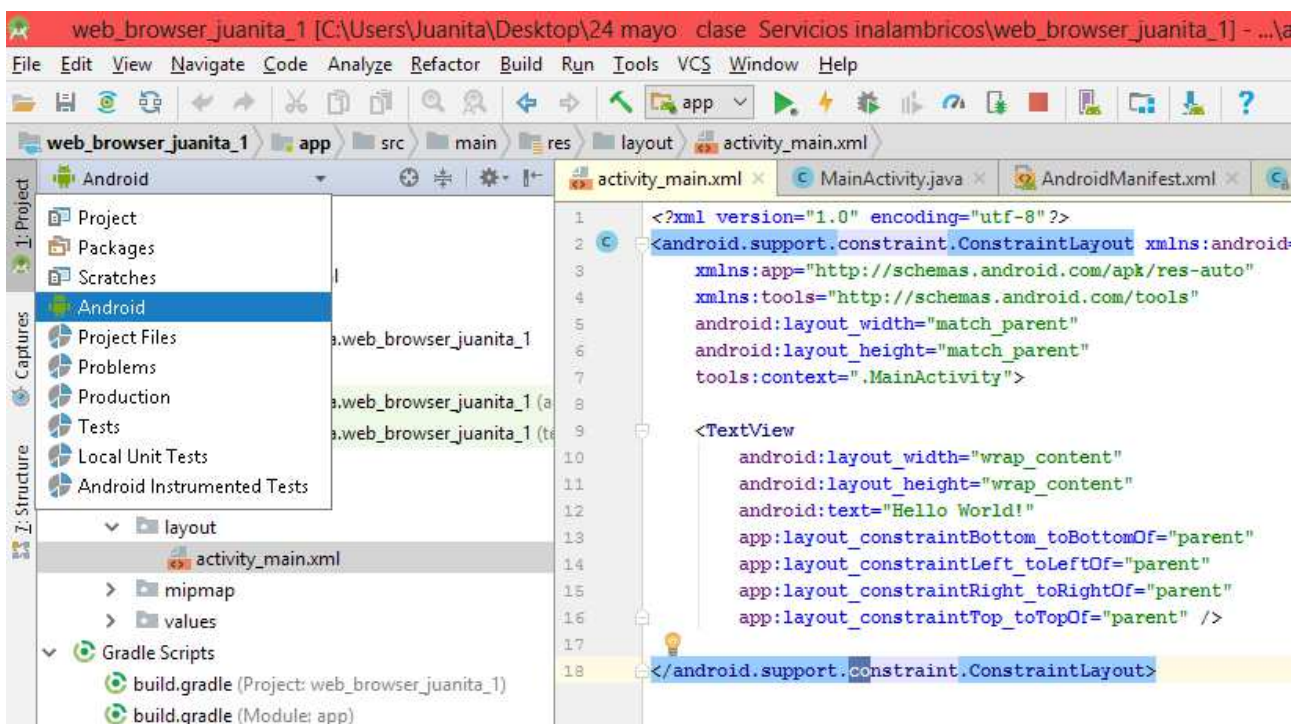


Cogemos la empty activity

La pantalla de configure activity la dejamos como está

Despues de un tiempo de procesamiento se abre el IDE de Android Studio y ya estamos en condiciones de comenzar a colocar el código en la aplicación.

Primero, asegúrate de que la ventana **Project** esté abierta (selecciona **View > Tool Windows > Project**) y la vista **Android** esté seleccionada en la lista desplegable de la parte superior de la ventana. Podrás ver los siguientes archivos:



app > java > com.example.myfirstapp > MainActivity.java

Esta es la activity principal (el punto de entrada para tu app). Cuando compilas y ejecutas la app, el sistema inicia una instancia de esta [Activity](#) y carga su diseño.

app > res > layout > activity_main.xml

Este archivo XML define el diseño para la IU de la activity. Puede contener [TextView](#) con un texto dentro por ejemplo “Hello world!”,

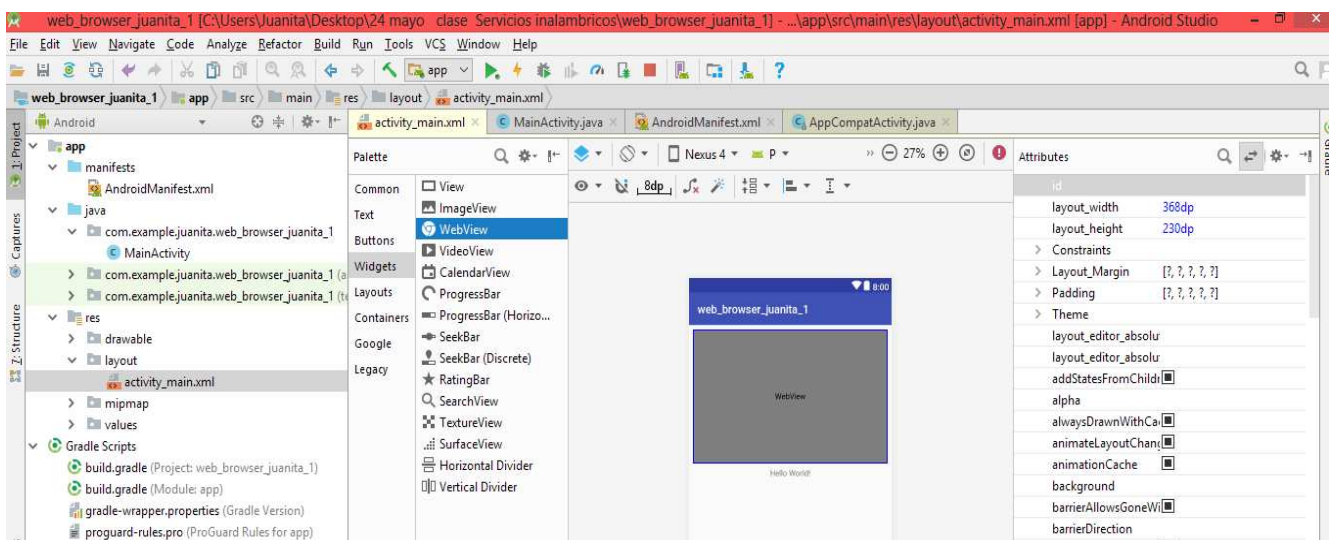
app > manifests > AndroidManifest.xml

El [archivo de manifiesto](#) describe las características fundamentales de la app y define cada uno de sus componentes.

Gradle Scripts > build.gradle

Verás dos archivos con este nombre: uno para el proyecto y otro para el módulo de la “app”. Cada módulo tiene su propio archivo `build.gradle`. Trabajarás principalmente con el archivo `build.gradle` del módulo para configurar la forma en que las herramientas de Gradle compilan y crean tu app.

Añadimos La WebView que se encuentra dentro de la sección “Widgets” de las vistas de Android Studio.



Una vez que hemos creado el proyecto y tenemos delante ya nuestra activity, lo único que debemos hacer es arrastrar dicho webView a la actividad de nuestra aplicación.

Configuraremos las siguientes propiedades de nuestro View:

- layout:width: fill_parent
- layout:height: fill_parent

También debemos rellenar el ID de la vista de manera que podamos hacer referencia a ella fácilmente.

En el apartado drag&drop todo listo. El resto que nos queda por hacer es programar el código Java para ejecutar la vista.

Podemos hacer que la webView funcione de dos formas diferentes. **La primera** de ellas, y un poco inútil, es que al llamarla Android abra nuestro navegador por defecto y cargue la URL especificada en el código. Esto se haría programando el código de la siguiente manera:

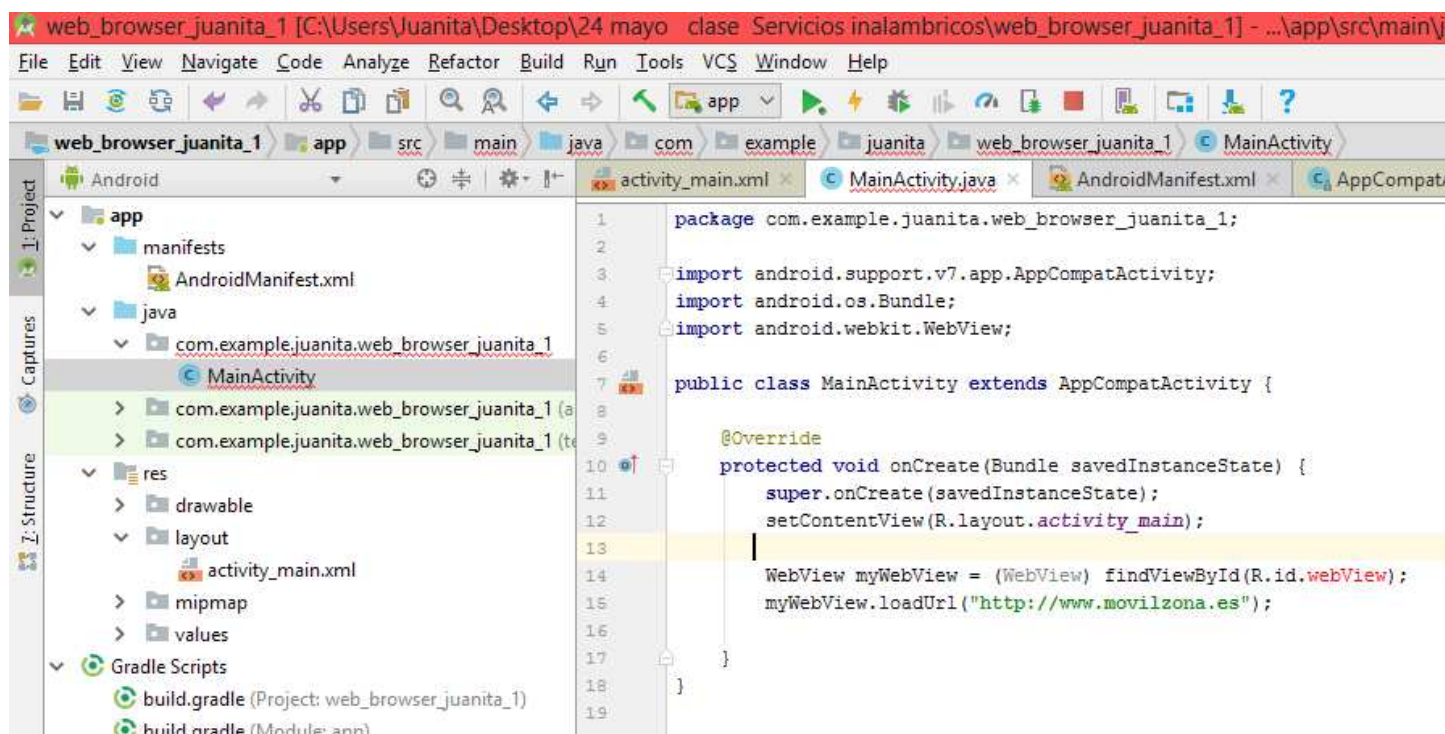
```
package es.movilzona.ejemplodewebview;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        WebView myWebView = (WebView) findViewById(R.id.webView);
        myWebView.loadUrl("http://www.uma.es");
    }

}
```

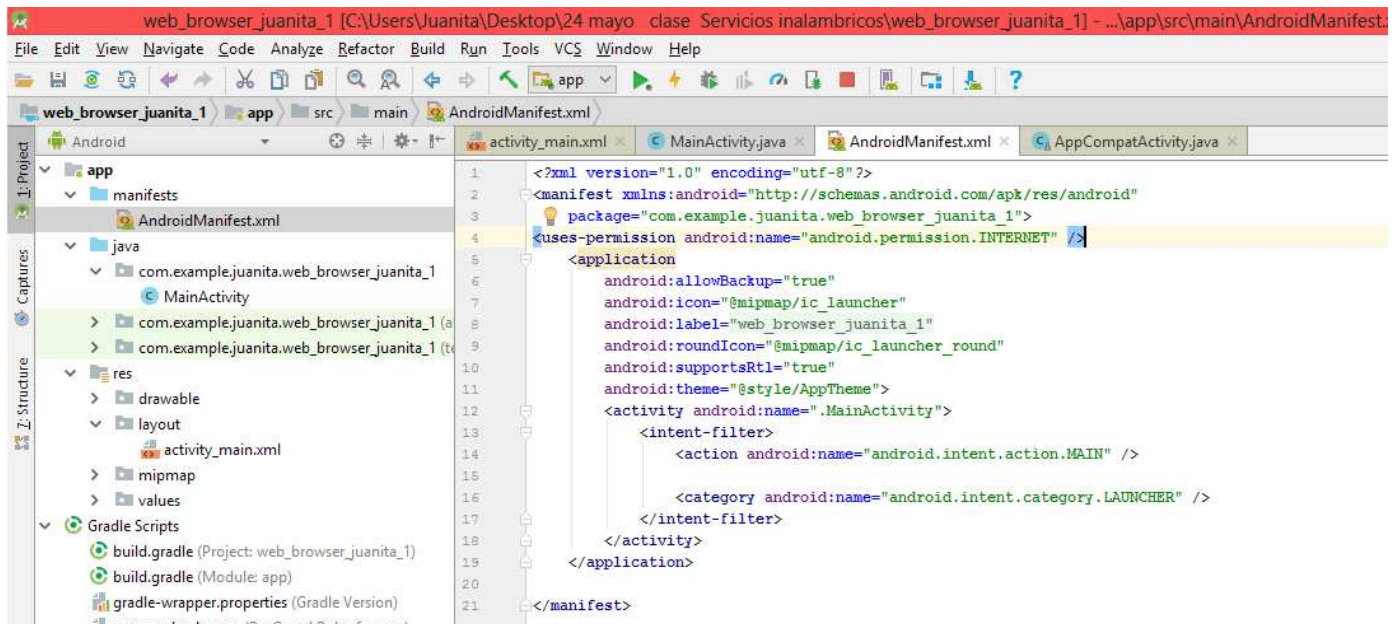


No ha funcionado ¿porque?

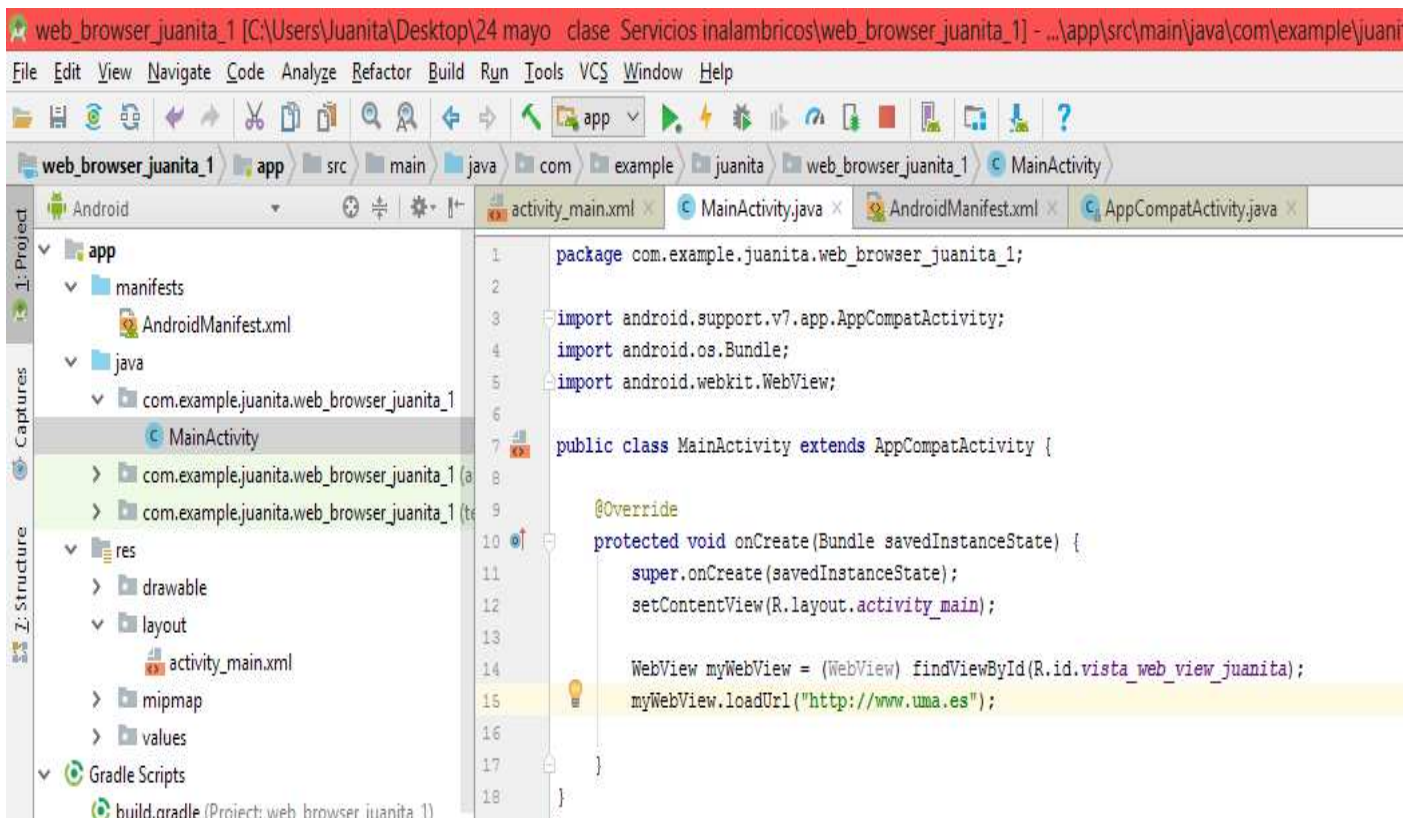
Antes de ejecutar la aplicación, recordamos que WebView necesita acceso a Internet, por lo que debemos añadir el permiso correspondiente en el archivo AndroidManifest.xml.

<uses-permission

android:name="android.permission.INTERNET" />



Ver donde está puesto el id del del WebView



Primera Modificación

Aunque funciona, no es lo que queremos. Nosotros queremos que Android utilice su motor de navegación dentro de la propia aplicación sin tener que depender de ningún navegador más. Esto nos lleva a la segunda forma, la más útil, profesional y recomendada.

El código para que la web cargue integrada en la propia aplicación sería:

```
package ejemplodewebview;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends Activity {
    WebView browser;

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

        // Definimos el webView
        browser=(WebView)findViewById(R.id.webView);

        browser.setWebViewClient(new WebViewClient() {

            @Override
            public boolean shouldOverrideUrlLoading(WebView view, String url) {
                view.loadUrl(url);
                return true;
            }
        });
        // Cargamos la web
        browser.loadUrl("http://www.uma.es");
    }
}
```

Segunda Modificación :Mejorando la aplicación con funciones avanzadas de WebView

Aunque nuestra aplicación funciona, podemos hacer que funcione aún mucho mejor habilitando al navegador ciertas funciones adicionales y añadiendo unos elementos para facilitar la navegación.

Extras que podemos añadir a nuestra WebApp

Por ejemplo, una función que no debemos olvidar añadir a nuestro código es la que nos permite habilitar JavaScript ya que, de lo contrario, muchas webs cargarán mal.

Para ello añadiremos en MainActivity.java:

```
browser.getSettings().setJavaScriptEnabled(true);
```

También podemos habilitar el Zoom mediante la llamada, útiles si nuestra web no es de diseño responsive:

```
browser.getSettings().setBuiltInZoomControls(true); // Habilita el Zoom
```

```
browser.getSettings().setDisplayZoomControls(false); // Oculta los botones de zoom, haciendo que solo funcione con gestos.
```

CODIGO COMPLETO

MainActivity

```
package com.example.juanita.web_browser_juanita_1;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends AppCompatActivity {
    WebView browser;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // version anterior
        // WebView myWebView = (WebView)
        findViewById(R.id.vista_web_view_juanita);
        //myWebView.loadUrl("http://www.uma.es");
        // Definimos el webView
        browser=(WebView)findViewById(R.id.vista_web_view_juanita);
        browser.getSettings().setBuiltInZoomControls(true); //hace zoom
        browser.getSettings().setJavaScriptEnabled(true); //habilita java
        browser.getSettings().setDisplayZoomControls(false); //oculta botones
        zoom
        browser.setWebViewClient(new WebViewClient() {
```

```
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            view.loadUrl(url);
            return true;
        }
    });
    // Cargamos la web
    browser.loadUrl("http://www.uma.es");
}
}
```


Más Modificaciones

Hacer que el botón “back” de nuestro smartphone vuelva a la página anterior

Por defecto, si pulsamos sobre el botón de volver atrás de nuestro smartphone, la aplicación volverá a la actividad anterior que, al no existir, cerrará el navegador y volverá al escritorio.

Es posible modificar este comportamiento simplemente añadiendo el MainActivity.java el siguiente código:

```
@Override
public void onBackPressed()
{
    if (browser.canGoBack())
    {
        browser.goBack();
    }
    else
    {
        super.onBackPressed();
    }
}
```

De esta manera, cuando pulsemos el botón “atrás” de Android volveremos a la página anterior, en lugar de cerrar la aplicación.

Añadiendo una barra de progreso

En muchas ocasiones es útil saber cómo está cargando una determinada web para saber si ha terminado o si aún le quedan elementos que descargar. Para dotar a nuestra app de una apariencia más profesional vamos a añadirle una barra de progreso.

Arrastrado desde la lista de views una ProgressBar (Horizontal) a la parte inferior de la pantalla.

Una vez listo, en ActivityMain.java añadiremos el código que le indicará a la barra cargar en función lo que cargue de la web:

```
import android.webkit.WebChromeClient;
import android.widget.ProgressBar;

...

//Sincronizamos la barra de progreso de la web
progressBar = (ProgressBar) findViewById(R.id.progressbar);

browser.setWebChromeClient(new WebChromeClient() {
    @Override
    public void onProgressChanged(WebView view, int progress) {
        progressBar.setProgress(0);
        progressBar.setVisibility(View.VISIBLE);
        MainActivity.this.setProgress(progress * 1000);

        progressBar.incrementProgressBy(progress);

        if (progress == 100) {
            progressBar.setVisibility(View.GONE);
        }
    }
});
```

Cuando volvamos a ejecutar la aplicación veremos en la parte inferior una barra de progreso mientras carga la web.

Podemos Modificar las propiedades de dicha barra (por ejemplo, ajustándola horizontalmente, cambiando el color, grosor, etc).

Código fuente de MainActivity.java con todo lo anterior implementado.

```
package es.movilzona.ejemplodewebview;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.webkit.WebChromeClient;
import android.widget.ProgressBar;

public class MainActivity extends Activity {
    private WebView browser;
    private ProgressBar progressBar;

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

        // Definimos el webView
        browser=(WebView)findViewById(R.id.webView);

        //Habilitamos JavaScript
        browser.getSettings().setJavaScriptEnabled(true);

        //Habilitamos los botones de Zoom
        browser.getSettings().setBuiltInZoomControls(true);

        browser.setWebViewClient(new WebViewClient() {

            @Override
            public boolean shouldOverrideUrlLoading(WebView view, String url) {
                view.loadUrl(url);
                return true;
            }
        });

        // Cargamos la web
        browser.loadUrl("http://www.uma.es");

        //Sincronizamos la barra de progreso de la web
        progressBar = (ProgressBar) findViewById(R.id.progressbar);

        browser.setWebChromeClient(new WebChromeClient() {
            @Override
            public void onProgressChanged(WebView view, int progress) {
                progressBar.setProgress(0);
                progressBar.setVisibility(View.VISIBLE);
                MainActivity.this.setProgress(progress * 1000);

                progressBar.incrementProgressBy(progress);

                if (progress == 100) {
                    progressBar.setVisibility(View.GONE);
                }
            }
        });
    }

    @Override
```

```
public void onBackPressed()  
{  
    if (browser.canGoBack())  
    {  
        browser.goBack();  
    }  
    else  
    {  
        super.onBackPressed();  
    }  
}  
}
```