un paseo por sus características

presentación de

José Emilio González Martínez

RESPONSIVE DESIGN

Responsive Design

Adaptándose a los recursos…

¿Qué es el *Responsive Design*?

Diseñar, organizar y adaptar

¿Qué es el Responsive Design?

Responsive Design = Diseño Adaptable

Es una técnica de diseño y desarrollo web que mediante el uso de una estructuras y diseño planificados consigue adaptar el sitio web al entorno del usuario, sea el que sea.

El Responsive Web Design trabaja bajo el concepto “One Web”, es decir, una web para todos que sea accesible y navegable desde cualquier tipo de dispositivo, ya sean dispositivos móviles, tabletas, pantallas widescreen o las diferentes resoluciones en los ordenadores.

¿Por qué molestarse en implementarlo?

¿Si ya tengo hecha mi web, para que implementar un diseño adaptable?

Las cifras hablan por sí solas.

¿Por qué adaptar nuestra web?

«Renovarse o morir»

Previsiones de ventas mundiales. (Millones de unidades)	2012 (Datos finales)	2013	2014
PCs y Notebooks	341	305	289
Ultramóviles	9	20	39
Tablets	120	201	276
Smartphones	1746	1821	1901
TOTAL	2217	2348	2506

*** Ventas de sistemas operativos**

Todos los sistemas operativos importantes, incluyendo Windows, crecerán en 2014. El aumento más espectacular es el de **Android**, que **aumenta en un 100% desde 2012**.

Previsiones de Ventas mundiales. (Millones de unidades)	2012 (Datos finales)	2013	2014
Android	505	866	1061
Windows	346	339	378
iOS/MacOS	212	296	354
RIM	34	25	22
Otros	1118	820	689
TOTAL	2217	2348	2506

Las cifras de ventas de dispositivos durante los últimos años hablan por sí solas.

El incremento de ventas de los dispositivos móviles se ha incrementado de forma espectacular, y por tanto no podemos obviar el diseño de nuestras páginas para éstos de ninguna manera.

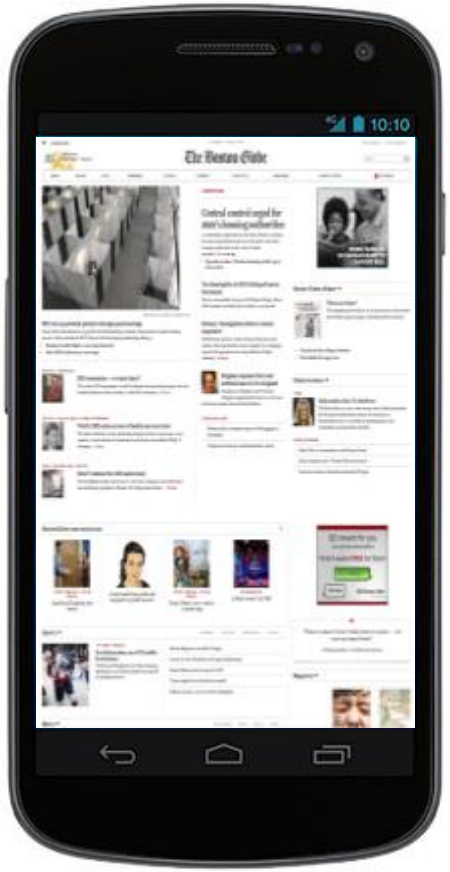
¿Ventajas al adaptar nuestra web?

Las ventajas que obtenemos usando un diseño adaptable son muchas pero sobre todo destacaría estas dos.

1. **Accesibilidad**. Desde el punto de vista del marketing la accesibilidad es una de las ventajas más relevantes del Responsive design, porque permite la visibilidad desde cualquier dispositivo y eso se traduce en visitas y ventas.

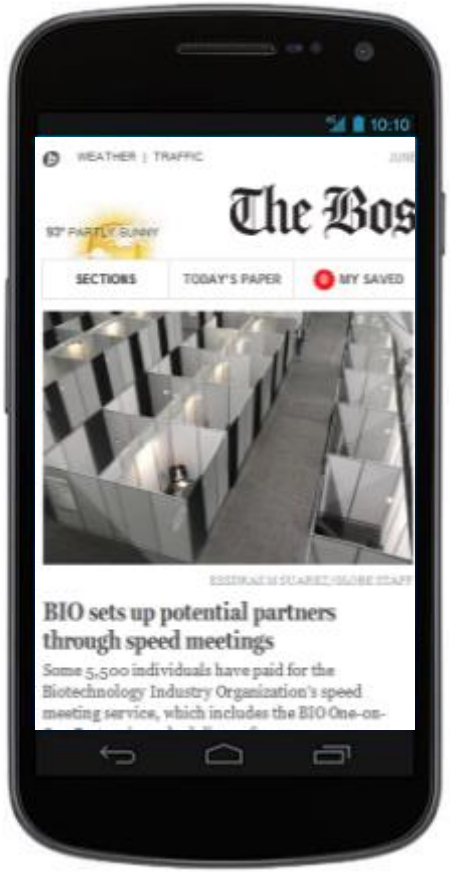
2. **Experiencia de usuario**. Mayor visibilidad y mejor experiencia de usuario.

Visitando una web NO RESPONSIVA en dispositivo móvil.



Página con un ancho normal de 960px.

Visitando una web NO RESPONSIVA en dispositivo móvil.




```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

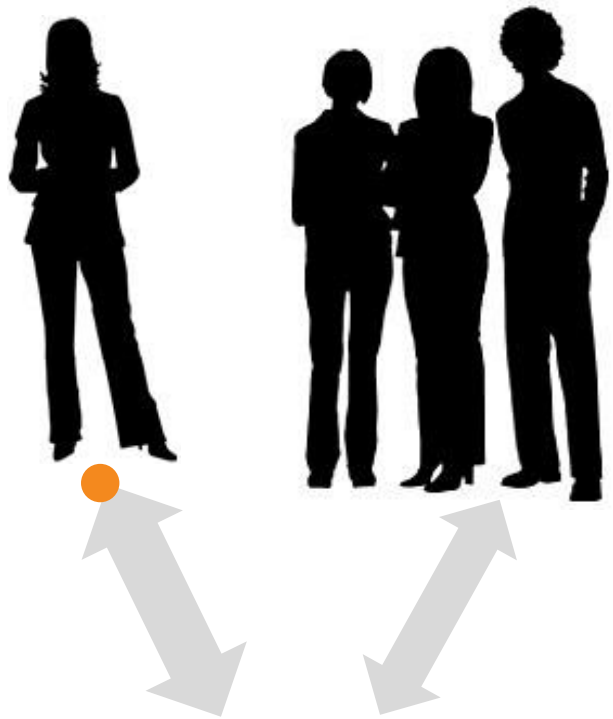
Quitamos el escalado por defecto y descubrimos que la web se convierte en un sitio muy poco manejable con barras de desplazamiento tanto horizontal como vertical.

El ancho de un dispositivo móvil puede ser de unos 320px.

¿Cómo implementar un diseño Adaptable a una web?

Los objetivos

- Una sola versión de HTML y CSS reducen costes tanto de creación como de mantenimiento.
- Un solo diseño es válido para cualquier formato.
- Desde el punto de vista SEO (posicionamiento), no se duplica el contenido y se evitan las redirecciones, ya que se muestra una sola URL en los buscadores.



Trabajando en un ejemplo

RESPONSIVE DESIGN.

Los principales conceptos que se deben tener en cuenta a la hora de realizar un responsive design son:

- **CSS Media Queries**. Permiten activar y desactivar partes del CSS del diseño según el tamaño de la pantalla y rediseñar las reglas de diseño CSS.

- **Trabajar con porcentaje**s. En lugar de trabajar con pixels, mejor trabajar con porcentajes para ganar la flexibilidad y adaptabilidad.

- **Imágenes flexibles**: Las imágenes no deben tener anchos fijos sino un máximo, para que las imágenes ajusten a todas las pantallas o resoluciones de navegador.

Trabajando en un ejemplo

Modelo de cajas

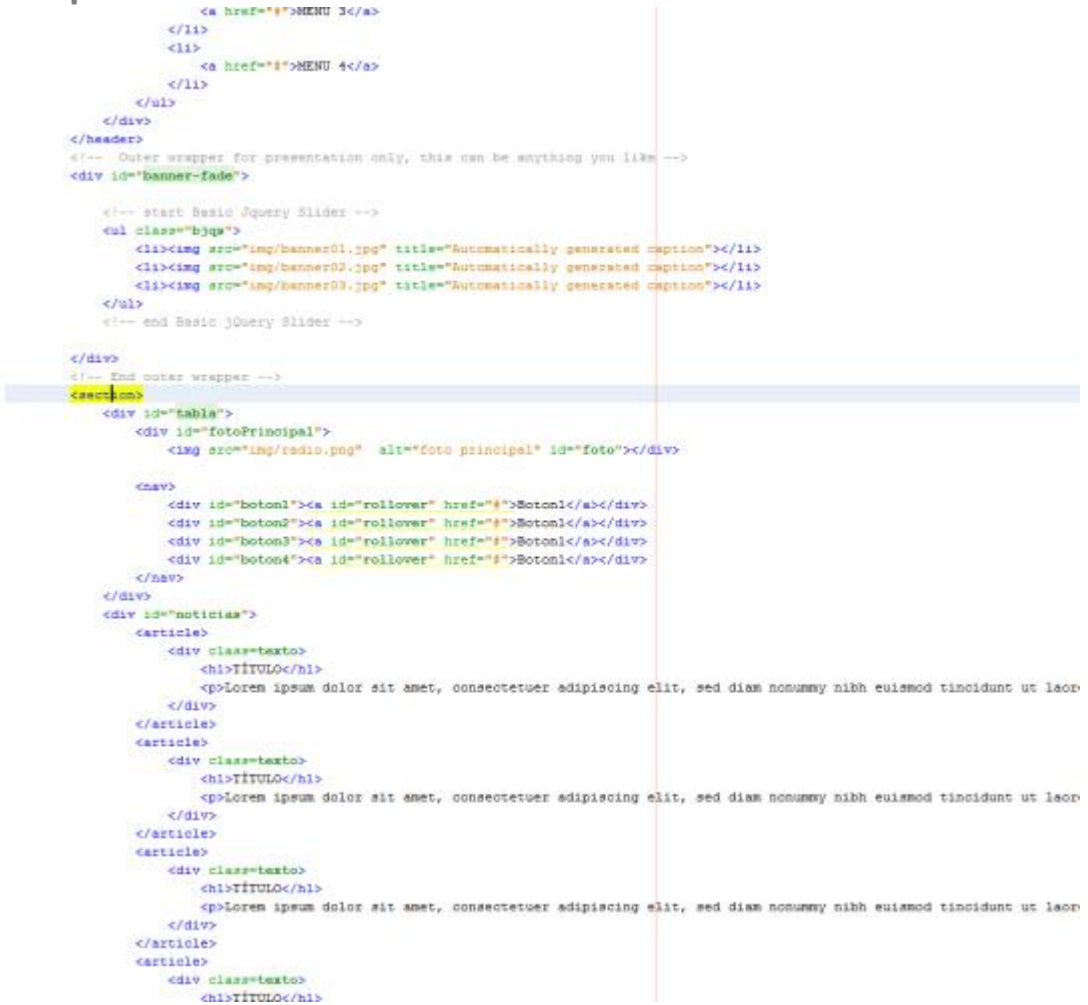
```
-->
<!DOCTYPE html>
<html>
    <head>
        <title>Ejemplo de Web Responsiva</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="stylesheet" type="text/css" href="css/MediaQueries.css" title="default">
    </head>
    <body>
        <div id="contenedor">
            <div id="cabecera">
                <div id="logo">
                </div>
                <div id="menu">
                </div>
            </div>
            <div id="cuerpo">
                <div id="portada">

                </div>
                <div id="portada_movil">

                </div>
                <div id="noticias">
                    <div id="noticia_principal">
                        <div class="titular">
                        </div>
                        <div class="cuerpo_noticia">
                        </div>
                    </div>
                    <div id="cuadro_mininoticias">
                        <div id="noticia1">
                            <div class="titular">
                            </div>
                            <div class="cuerpo_noticia">
                            </div>
                        </div>
                        <div id="noticia2">
                            <div class="titular">
                            </div>
                            <div class="cuerpo_noticia">
                            </div>
                        </div>
                        <div id="noticia3">
                            <div class="titular">
                            </div>
                            <div class="cuerpo_noticia">
                            </div>
                        </div>
                    </div>
                </div>
            </div>
```

Trabajando en un ejemplo

Web ejemplo con
algunas etiquetas HTML5



```
                <a href="#">MENU 3</a>
            </li>
            <li>
                <a href="#">MENU 4</a>
            </li>
        </ul>
    </div>
</header>
<!--  Outer wrapper for presentation only, this can be anything you like -->
<div id="banner-fade">

    <!-- start Basic Jquery Slider -->
    <ul class="bjqs">
        <li><img src="img/banner01.jpg" title="Automatically generated caption"></li>
        <li><img src="img/banner02.jpg" title="Automatically generated caption"></li>
        <li><img src="img/banner03.jpg" title="Automatically generated caption"></li>
    </ul>
    <!-- end Basic jQuery Slider -->

</div>
<!-- End outer wrapper -->
<section>
    <div id="tabla">
        <div id="fotoPrincipal">
            <img src="img/radio.png"  alt="foto principal" id="foto"></div>

        <nav>
            <div id="boton1"><a id="rollover" href="#">Boton1</a></div>
            <div id="boton2"><a id="rollover" href="#">Boton1</a></div>
            <div id="boton3"><a id="rollover" href="#">Boton1</a></div>
            <div id="boton4"><a id="rollover" href="#">Boton1</a></div>
        </nav>
    </div>
    <div id="noticias">
        <article>
            <div class=texto>
                <h1>TÍTULO</h1>
                <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laor
            </div>
        </article>
        <article>
            <div class=texto>
                <h1>TÍTULO</h1>
                <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laor
            </div>
        </article>
        <article>
            <div class=texto>
                <h1>TÍTULO</h1>
                <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laor
            </div>
        </article>
        <article>
            <div class=texto>
                <h1>TÍTULO</h1>
```

Trabajando en un ejemplo

Aspecto de la web de ejemplo una vez hemos terminado su diseño CSS.

Vista desde el móvil del resultado de nuestro ejemplo

Página con un ancho normal de 960px.

Trabajando en un ejemplo

Así que es hora de implementar las reglas que debe seguir nuestra interfaz para adaptarse a la resolución que hemos decidido que debe tener nuestro dispositivo móvil

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Práctica HTML5+CSS3</title>
        <link href="estilo.css" rel="stylesheet" type="text/css">
        <meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1.0;">

        <!-- SCRIPTS PARA SLIDERk -->
        <!-- bjqs.css contains the *essential* css needed for the slider to work -->
        <link rel="stylesheet" href="bjqs.css">
        <!-- some pretty fonts for this demo page - not required for the slider -->
        <link href='http://fonts.googleapis.com/css?family=Source+Code+Pro|Open+Sans:300' rel='stylesheet' type='text/css'>
        <!-- demo.css contains additional styles used to set up this demo page - not required for the slider -->
        <link rel="stylesheet" href="demo.css">
        <!-- load jQuery and the plugin -->
        <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
        <script src="js/bjqs-1.3.min.js"></script>
        <script class="secret-source">
            jQuery(document).ready(function($) {

                $('#banner-fade').bjqs({
                    height: 320,
                    width: 960,
                    responsive: true
                });

            });
        </script>

        <link href="media_queries.css" rel="stylesheet" type="text/css">

    </head>

    <body>
```

Trabajando en un ejemplo

```
@media only screen and (max-width: 400px){
    body{width:380px;}
    #banner-fade{display: none;}

    #nombre{

        font-size:30px;
        margin-top:10px;
        margin-left:-10px;
        float:left;
    }

    #idiomas{
        margin-top:0px;
        margin-left:13%;
        float: left;
        clear: both;
    }
    #logo{

        height: 60%;
        width: 60%;
    }
    #fotoPrincipal{
        width: 99%;
        height: 280px;

    }
    #foto {
        margin-top: 10px;
        height: 100%;
        width: 100%;
    }
```

Archivo media_queries.css

Redefinimos el comportamiento que deben tener los elementos HTML dentro de la resolución seleccionada.

Vista desde el móvil del resultado de nuestro ejemplo

Página totalmente adaptada.

Otras utilidades a tener en cuenta

Es importante saber que hay otras utilidades que se pueden usar que pueden complementar lo usado anteriormente:

Javascript
JQuery
CSS3

Complementos ya programados:

Bootstrap
Plantillas…

¿Qué resoluciones debería tener en cuenta en media queries?...

Cuantas más MEJOR

Aunque depende del caso...

- Smartphone- 320 x 480
- Tablet - 768 x 1024
- Desktop - 1024 x 1280
- Constante cambio...

¿Qué resoluciones debería tener en cuenta en media queries?...

Probar en diferentes dispositivos

O usar utilidades similares a

http://www.responsinator.com/

¿Te interesa?

¿Por qué debería molestarme en saber yo todo esto si yo soy desarrollador **Android?** ¿Qué tengo que ver con todo esto?

«Adaptarse o morir…»

Miles de páginas por adaptar

- Hay miles de páginas por adaptar.
- Gran demanda de dispositivos móviles
- Las empresas buscan el facilitar el acceso a sus usuarios y complementar lo que ya tienen a las nuevas tecnologías.
- Los usuarios desean un acceso sencillo e intuitivo.

Forma sencilla de realizar una APP

ANDROID a partir de una web

En pocos y sencillos pasos podremos adaptar una página web para hacer creer al usuario que se encentra ante una aplicación móvil.

Crear una APP que tenga el aspecto de nuestra web

¡Creando una APP!

Vamos a crear una APP que muestre nuestra WEB usando :

JDK

Eclipse

ADT

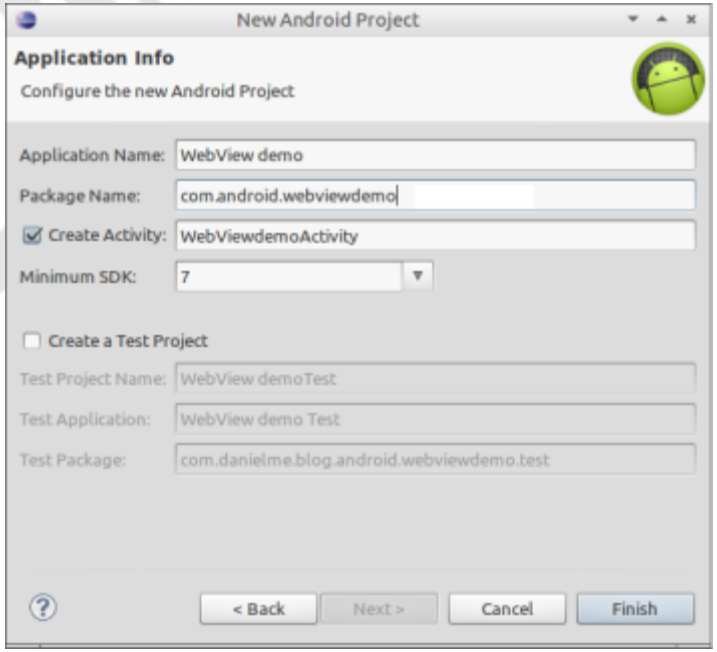
Y dentro de estos los componentes:

WebView

ProgressBar

SlidingDrawer

Creando la aplicación:



Creamos el WebView en el Layout creado por defecto

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <WebView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/webkit"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />

</LinearLayout>
```

Creando la aplicación:

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

public class WebViewdemoActivity extends Activity
{
    private WebView browser;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        browser = (WebView)findViewById(R.id.webkit);

                //habilitamos javascript y el zoom
        browser.getSettings().setJavaScriptEnabled(true);
        browser.getSettings().setBuiltInZoomControls(true);

        //habilitamos los plugins (flash)
        browser.getSettings().setPluginsEnabled(true);

        browser.loadUrl("http://mipaginaweb.com");
    }
}
```

En nuestra Activity dejamos por defecto la página que queremos mostrar en nuestra aplicación. Y se añade el permiso de internet en el manifiesto.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Creando la aplicación:

Si se pulsa un enlace, Android lo abrirá con la aplicación por defecto para tal fin, esto sería, el propio navegador. Si queremos permanecer en nuestra APP hay que especializar la clase WebViewClient y sobrescribir el método shouldOverrideUrlLoading para que devuelva false. Puesto que en este método se recibe como parámetro la URL solicitada se podría utilizar también para filtrar las direcciones a las que se permite el acceso.

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

public class WebViewdemoActivity extends Activity
{
    private WebView browser;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        browser = (WebView)findViewById(R.id.webkit);

                //habilitamos javascript y el zoom
        browser.getSettings().setJavaScriptEnabled(true);
        browser.getSettings().setBuiltInZoomControls(true);

                //habilitamos los plugins (flash)
        browser.getSettings().setPluginsEnabled(true);

                browser.loadUrl("http://mipaginaweb.com");

        browser.setWebViewClient(new WebViewClient()
        {
                        // evita que los enlaces se abran fuera nuestra app en el n
            @Override
            public boolean shouldOverrideUrlLoading(WebView view, String url)
            {
                return false;
            }

        });
    }

}
```

Creando la aplicación:

Para que el usuario no esté confuso si una web deja de cargar, vamos a añadir una barra de progreso de carga.

```
<ProgressBar
    android:id="@+id/progressbar"
    style="@android:style/Widget.ProgressBar.Horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_margin="3dp"
    android:visibility="gone"/>
```

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

public class WebViewdemoActivity extends Activity
{
        private WebView browser;

    private ProgressBar progressBar;

        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);
                WebViewdemoActivity.this.setProgress(progress * 1000);

                progressBar.incrementProgressBy(progress);

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

Creando la aplicación:

Ahora cambiaremos el título según la página que visitemos gracias al método onReceivedTitle:

```
@Override
public void onReceivedTitle(WebView view, String title)
{
    WebViewdemoActivity.this.setTitle("WebView demo: " + WebViewdemoActivity.this.browser.getTitle());
}
```

También podremos añadir el slidingDrawer para los siguientes elementos de navegación:

```
<SlidingDrawer
    android:id="@+id/slidingDrawer1"
    android:layout_width="match_parent"
    android:layout_height="100dp"
    android:layout_alignLeft="@+id/nav"
    android:layout_alignParentBottom="true"
    android:content="@+id/content"
    android:handle="@+id/handle" >
    <Button
        android:id="@+id/handle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Opciones de Navegación" />
    <LinearLayout
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:gravity="center"
        android:orientation="horizontal" >
        <Button
```

```
<Button
    android:id="@+id/buttonAnt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/url"
    android:minWidth="85dp"
    android:onClick="anterior"
    android:text="Atrás" />

<Button
    android:id="@+id/buttonDetener"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minWidth="85dp"
    android:onClick="recargar"
    android:text="Recargar" />

<Button
    android:id="@+id/buttonSig"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/url"
    android:layout_toRightOf="@+id/buttonAnt"
    android:minWidth="5dp"
    android:onClick="siguiente"
    android:text="Adelante" />
```

Creando la aplicación:

Algunos métodos que podríamos asociar a los botones de navegación de nuestro **SlidingDrawer** :

```
public void anterior(View view)
{
    browser.goBack();
}

public void siguiente(View view)
{
    browser.goForward();
}

public void detener(View view)
{
    browser.stopLoading();
}
```

Creando la aplicación:

Según lo que queramos hacer en nuestra APP podríamos usar los métodos:

onPageStarted: Se ejecuta al iniciarse la carga de una página, una vez por cada frame.

onPageFinished: Se invocará cuando haya terminado la carga de la página.

Hacer que el botón físico «atrás» no nos saque de la aplicación, sino que funcione como el botón «atrás»:

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

Creando la aplicación:

Para controlar la gestión de errores que se pudieran producir en el webview (página no encontrada,… etc) podemos implementar el método onReceivedError para actuar en consecuencia:

```
@Override
public void onReceivedError(WebView view, int errorCode, String description, String failingUrl)
{
    AlertDialog.Builder builder = new AlertDialog.Builder(WebViewdemoActivity.this);
    builder.setMessage(description).setPositiveButton("Aceptar", null).setTitle("onReceivedError");
    builder.show();
 }
```

Siempre que en una APP necesitemos conectividad, es una buena práctica comprobar que la tenemos disponible antes de solicitar cualquier operación y esperar a recibir un timeout o error similar. Para ello hay que hacer uso de ConnectivityManager y NetworkInfo. (Dar permisos en el manifiesto)

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

```
private boolean checkConnectivity()
        {
            boolean enabled = true;

            ConnectivityManager connectivityManager = (ConnectivityManager) this.getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo info = connectivityManager.getActiveNetworkInfo();

            if ((info == null || !info.isConnected() || !info.isAvailable()))
            {
                enabled = false;
                Builder builder = new Builder(this);
                builder.setIcon(android.R.drawable.ic_dialog_alert);
                builder.setMessage(getString(R.string.noconnection));
                builder.setCancelable(false);
                builder.setNeutralButton(R.string.ok, null);
                builder.setTitle(getString(R.string.error));
                builder.create().show();
            }
            return enabled;
        }
```

Creando la aplicación:

Para controlar la gestión de errores que se pudieran producir en el webview (página no encontrada,… etc) podemos implementar el método onReceivedError para actuar en consecuencia:

```
@Override
public void onReceivedError(WebView view, int errorCode, String description, String failingUrl)
{
    AlertDialog.Builder builder = new AlertDialog.Builder(WebViewdemoActivity.this);
    builder.setMessage(description).setPositiveButton("Aceptar", null).setTitle("onReceivedError");
    builder.show();
 }
```

Tenga acceso en cualquier lugar

- Ya tenemos nuestra WEB
- Y nuestra APP Android
- **Con RESPONSIVE DESIGN**

José Emilio González Martínez

IZV 2013