Plataforma iOS

Sesión 8: Guías de estilo y personalizaciones avanzadas

Índice de contenidos

Guías de estilo en iOS

- Adaptar en la medida de lo posible las guías de estilo que plantea Apple a nuestras aplicaciones.
  - http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

- ¿Qué ocurre si no seguimos los puntos básicos que indica Apple?
  - Nuestra aplicación puede ser rechazada en el proceso de revisión.
  - Los usuarios pueden no estar familiarizados con nuestra aplicación y pueden llegar a no saber usarla adecuadamente.
  - Pocas posibilidades de promoción en iTunes (Apple).

    Apple tiene muy en cuenta a la hora de destacar una aplicación si ésta cumple con las guías de estilo propuestas.

Aspectos a tener en cuenta en iOS (1)

- **La pantalla**
  - Es el componente más importante.
  - Todo usuario interactúa con la aplicación mediante la pantalla.
  - Tener en cuenta el tamaño de la pantalla del dispositivo:

    iPhone 4, 4S (Retina Display): 960 x 640 px.

    iPad 1, 2: 1024 x 768 px.

    iPhone 3G, 3GS o iPod Touch: 480 x 320 px.

    iPad 3 (Nuevo iPad): 2048 x 1536 px
  - Si desarrollamos una aplicación universal deberemos tener en cuenta estas resoluciones, o al menos las más utilizadas.
  - Más resoluciones a tener en cuenta = más trabajo de diseño.

Aspectos a tener en cuenta en iOS (2)

- **Orientación del dispositivo.**
  - Apple **obliga** a que la aplicación esté preparada para funcionar en al menos dos posiciones opuestas:

    Landscape right & Landscape left / **horizontal**

    Portrait & Portrait UpsideDown / **vertical**
  - Se mejora la usabilidad de la aplicación.
  - Usamos el tipo enumerado: `UIInterfaceOrientation`
  - Método: `shouldAutorotateToInterfaceOrientation`
  - Ejemplo:

```
- (BOOL)shouldAutorotateToInterfaceOrientation:
  UIInterfaceOrientation)interfaceOrientation
{


    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
```

Aspectos a tener en cuenta en iOS (3)

- **Los gestos multitáctiles**
  - Gestos = movimientos de los dedos sobre la pantalla para realizar distintas acciones.
  - Actualmente iOS admite hasta 5 pulsaciones al mismo tiempo en pantalla.
  - Mejora la usabilidad y hace más atractiva la aplicación.

Aspectos a tener en cuenta en iOS (3)

- **Los gestos multitáctiles (dos o más dedos)**
  - Existen 6 tipos de gestos que podemos detectar y procesar en iOS 5 (`UIKit`)
    - Arrastre en 4 direcciones (*swipe*). `UISwipeGestureRecognizer`
    - Arrastre en cualquier dirección (*pan*) `UIPanGestureRecognizer`
    - Rotación (*rotation*) `UIRotationGestureRecognizer`
    - Pellizco (*pinch*) `UIPinchGestureRecognizer`
    - Pulsación larga (*long press*) `UILongPressGestureRecognizer`
    - Pulsación corta (*tap*) `UITapGestureRecognizer`
  - Existen otros gestos que están implementados por defecto en iOS, como el arrastre sencillo (*drag*).
  - Para detectar gestos usaremos la clase `UIGestureRecognizer`
  - Los gestos se deben de utilizar de una forma lógica, como indica la guía de estilo.

Aspectos a tener en cuenta en iOS (4)

- **La multitarea**
  - Multitarea = ejecución en segundo plano de varias aplicaciones.
  - Disponible a partir de iOS 4.
  - Apple “recomienda” que nuestra aplicación soporte multitarea.
  - Implementación muy sencilla (en un principio).

    Métodos de la clase *Delegate*:
    `applicationDidEnterBackground` y
    `applicationWillEnterForeground`
  - Varios puntos a tener en cuenta:
    - ¿La aplicación seguirá su proceso en segundo plano o se detendrá?
    - ¿Qué hacemos con los sonidos?
    - ¿Y si tenemos peticiones a Internet activas?

Aspectos a tener en cuenta en iOS (5)

- **Preferencias y ayuda.**
  - Si la aplicación usa preferencias de configuración propias, éstas deberán aparecer en el panel de configuración de los ajustes del dispositivo (App “Ajustes”).
  - Fichero “*Settings bundle*” de tipo *plist* (xml).

Aspectos a tener en cuenta en iOS (6)

- **Principales tecnologías anteriores a iOS 5**
  - Si la aplicación que desarrollemos permite hacer uso de cualquiera de las tecnologías disponibles en iOS: ¡Usadlas!
  - Multitarea.
  - Imprimir (AirPrint).
  - iAD (publicidad)
  - Vista rápida de documentos.
  - Sonidos.
  - Control por voz (VoiceOver)
  - Menú contextual de edición en textos (copiar/pegar/seleccionar).
  - Teclado (personalizaciones, tipos).
  - Servicios de localización (GPS, Brújula, acelerómetro, giroscopio...).
  - Notificaciones push, in-apps (micro-pagos).

Aspectos a tener en cuenta en iOS (7)

- **Principales tecnologías aparecidas en iOS 5**
  - iCloud (compartir en la nube).

  - iMessage (mensajería instantánea).
  - Core Image (modificación avanzada de imágenes).
  - Airplay (mirroring, Apple TV).
  - Integración con Twitter
  - Siri, Facebook, Mapas Apple... ¿iOS 6?

Índice de contenidos

- Guías de estilo en iOS
  - Características principales de la plataforma iOS
  - Estrategias de diseño de aplicaciones
  - Principales tecnologías disponibles en iOS

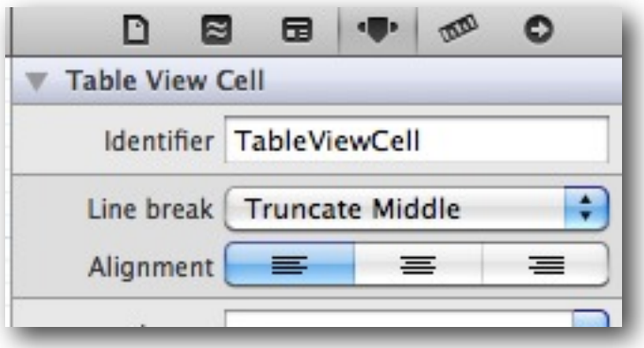
- **Personalizaciones avanzadas**
  - **Celdas**

Personalizaciones avanzadas: Celdas

- Muy habitual en el desarrollo de aplicaciones iOS.
- Objetivos: conseguir un aspecto atractivo al usuario, mejorar la usabilidad y distinguirnos del resto de aplicaciones.
- Ejemplo: App Twinkle (cliente Twitter).

Programando celdas personalizadas (1)

- **Pasos a realizar (*sin usar Story Boards*):**
  - **1)** Creamos la clase de la celda. Heredará de `UITableViewCell` y contendrá todos los elementos que queremos que tenga la celda (etiquetas, imágenes, campos de formulario, etc...).
  - **2)** Diseñamos la celda mediante Interface Builder: uso del componente `Table View Cell`
  - **3)** Pestaña *Identity Inspector*: escribimos el nombre de la clase que referencia a la celda.
  - **4)** Pestaña *Attributes Inspector*: escribimos un identificador para la celda, por ejemplo: `TableViewCell`



Programando celdas personalizadas (2)

- **5)** En el método `cellForRowAtIndexPath` deberemos de inicializar la celda con el fichero Nib (*Interface Builder*) que hemos diseñado anteriormente:

```
- (UITableViewCell *)tableView:(UITableView *)tableView
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
        static NSString *CellIdentifier = @"TableViewCell";
        TableViewCell *cell = (TableViewCell*)[tableView
                    dequeueReusableCellWithIdentifier: CellIdentifier];
         if (cell == nil) {
            // Cargamos el Nib de la celda
            NSArray *array = [[NSBundle mainBundle] loadNibNamed:@"TableViewCell"
                                        owner:self options:nil];
            cell = [array objectAtIndex:0]; // TODO: Faltan comprobaciones...
        }
        // Configuramos la celda
        cell.labelTitulo.text = [NSString stringWithFormat:
                            @"Título noticia número %d", indexPath.row+1];
        cell.labelTexto.text = @"Texto de pruebas...";
        cell.labelAutor.text = @"Autor/es de la noticia";
        cell.imagen.image = [UIImage imageNamed:@"logo_mvl.png"];

        return cell;
}
```

¿Preguntas?