

# iAd, AdMob e In Apps

## Índice

1 La publicidad de Apple: iAd.....	2
1.1 Ventajas y desventajas del uso de iAd.....	2
1.2 Integrando iAd en nuestras aplicaciones.....	3
2 La publicidad de Google: AdMob.....	12
2.1 AdMob: Ventajas y desventajas.....	12
2.2 Integrando AdMob.....	13
3 Otras plataformas de publicidad en iOS.....	23
4 Micropagos: In-Apps.....	24
4.1 Implementando los in-apps: App freemium.....	25

En esta primera sesión de **Servicios iOS** comenzaremos estudiando el sistema de publicidad de Apple: **iAd**. iAd está siendo actualmente uno de los servicios de publicidad más usados en aplicaciones iOS junto con otros como **AdMob**, el propio de *Google*. Durante esta sesión veremos las ventajas y desventajas usar iAd en nuestras aplicaciones, configuraremos el servicio de iAd desde el portal de desarrolladores, *iTunes Connect*, y realizaremos una aplicación de ejemplo que lo use.

Una vez visto iAd analizaremos el otro sistema de publicidad más usado en aplicaciones móviles: **AdMob**. Al igual que iAd, analizaremos en detalle las ventajas y desventajas de su uso y veremos un ejemplo paso a paso de integración en una aplicación.

Por último analizaremos el uso de las compras dentro de las aplicaciones, también llamados "micropagos" o como Apple los llama: **in-app**. El uso de in-apps está a día de hoy muy extendido como modelo de negocio en las aplicaciones para móviles. Su uso se está extendiendo cada vez más ya que permite explotar económicamente las aplicaciones gratuitas de una forma "limpia" y bastante atractiva al usuario final. En esta sesión veremos detenidamente como preparar nuestras aplicaciones para el uso de in-apps y realizaremos un ejemplo paso a paso.

## 1. La publicidad de Apple: iAd

Durante la *keynote* de 2010 Apple presentó algo muy novedoso y atractivo para el mundo de las aplicaciones iOS, una nueva plataforma de publicidad que denominó como **iAd**. El nuevo sistema de publicidad estuvo disponible junto con la salida del iPhone 4. La idea fundamental de iAd es proporcionar al usuario una publicidad de calidad y con un gran atractivo por la forma de integrarse en las aplicaciones.

### 1.1. Ventajas y desventajas del uso de iAd

iAd ofrece un sistema de publicidad **no intrusivo**: los anuncios se muestran en una ventana a parte (*popup*) dentro de la aplicación, lo que permite que el usuario acceda a esta publicidad sin salir de la aplicación, algo que no ocurre con el resto de modelos de publicidad. Este es un punto a favor.

Otra de las ventajas de iAd es el sistema de reparto de ganancias que tiene. En un sistema de publicidad común nosotros, como desarrolladores editores, recibimos unos beneficios basados en el ratio de CPC (Coste por click) o CPM (Coste por impresión). En iAd el sistema vuelve a cambiar y nosotros recibiremos ganancias de ambos ratios, eso sí, al igual que ocurre en las aplicaciones de pago, recibiremos un **60%** de las ganancias obtenidas. Es habitual que el desarrollador obtenga más beneficios de iAd que de otras plataformas de publicidad.

iAd está integrado en su totalidad con el sistema de APIs del SDK de iOS, así como en los portales de desarrolladores lo que supone una facilidad enorme a la hora de preparar la

aplicación para mostrar anuncios así como para gestionar las ganancias. No necesitaremos acceder a otro sistema y gestionar todo por otro lado, con iAd lo tenemos integrado al 100%.

Pero todo no son cosas buenas... iAd tiene una gran desventaja respecto a otras plataformas: la disponibilidad. Actualmente aún existen muchos países sin total cobertura de anuncios, lo que supone un problema grave para aquellos desarrolladores que basen su modelo de negocio únicamente en este sistema. El país que más disponibilidad tiene de iAd es, como no podría ser de otra manera, Estados Unidos. En España, por ejemplo, la disponibilidad es baja. Existen métodos alternativos que nos permiten detectar que en el caso de que nuestra aplicación no reciba anuncios aparezca otra plataforma como AdMob o nuestros propios anuncios.

La falta de disponibilidad de anunciantes es en gran medida debido a las altísimas tarifas que impone Apple a los anunciantes, los cuales se ven obligados a dejar a un lado iAd para pasarse a otras plataformas más baratas. En este sentido Apple está progresivamente bajando las tarifas, por lo que se espera que a corto-medio plazo existan muchos más anunciantes en iAd y por lo tanto, más disponibilidad.

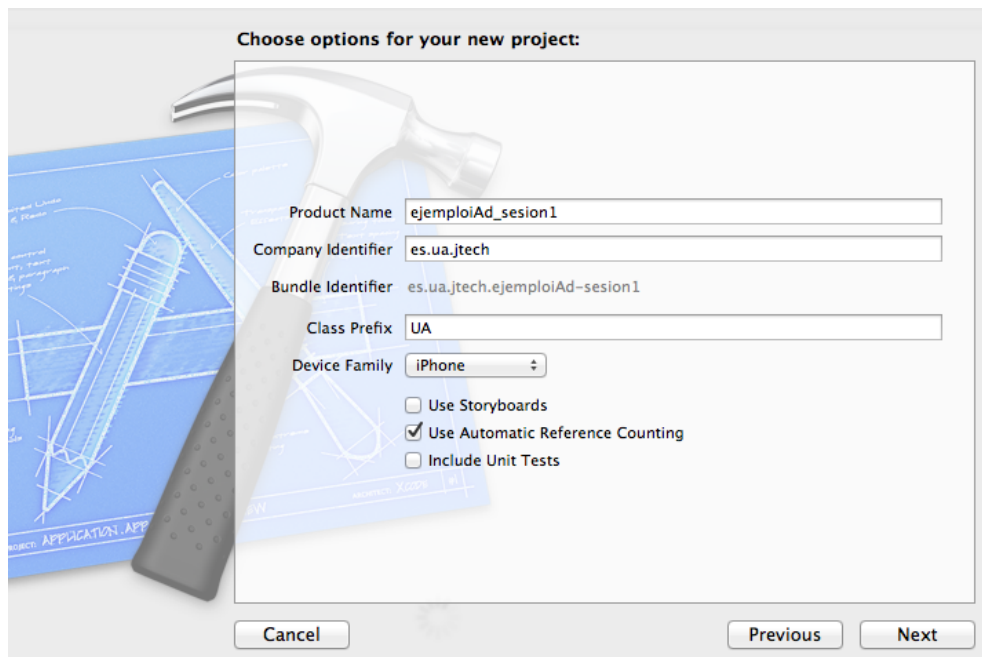
Por último comentar que una de las grandes desventajas del modelo de negocio de publicidad en aplicaciones móviles es la necesidad de disponer de una conexión activa a Internet para acceder a los anuncios. Como en el punto anterior, existen varios métodos alternativos que deberemos de contemplar en el caso de que el usuario no disponga de Internet, como por ejemplo mostrar anuncios propios que promocionen nuestros productos (otras aplicaciones...).

## 1.2. Integrando iAd en nuestras aplicaciones

En este punto veremos como integrar la plataforma de publicidad iAd en nuestras aplicaciones iOS. El banner de publicidad estará preparado para la rotación del dispositivo: este tiene que rotar junto con el dispositivo. Comenzaremos creando un proyecto desde cero en XCode el cual contendrá una vista únicamente y después incorporaremos iAd. ¡Comenzamos!

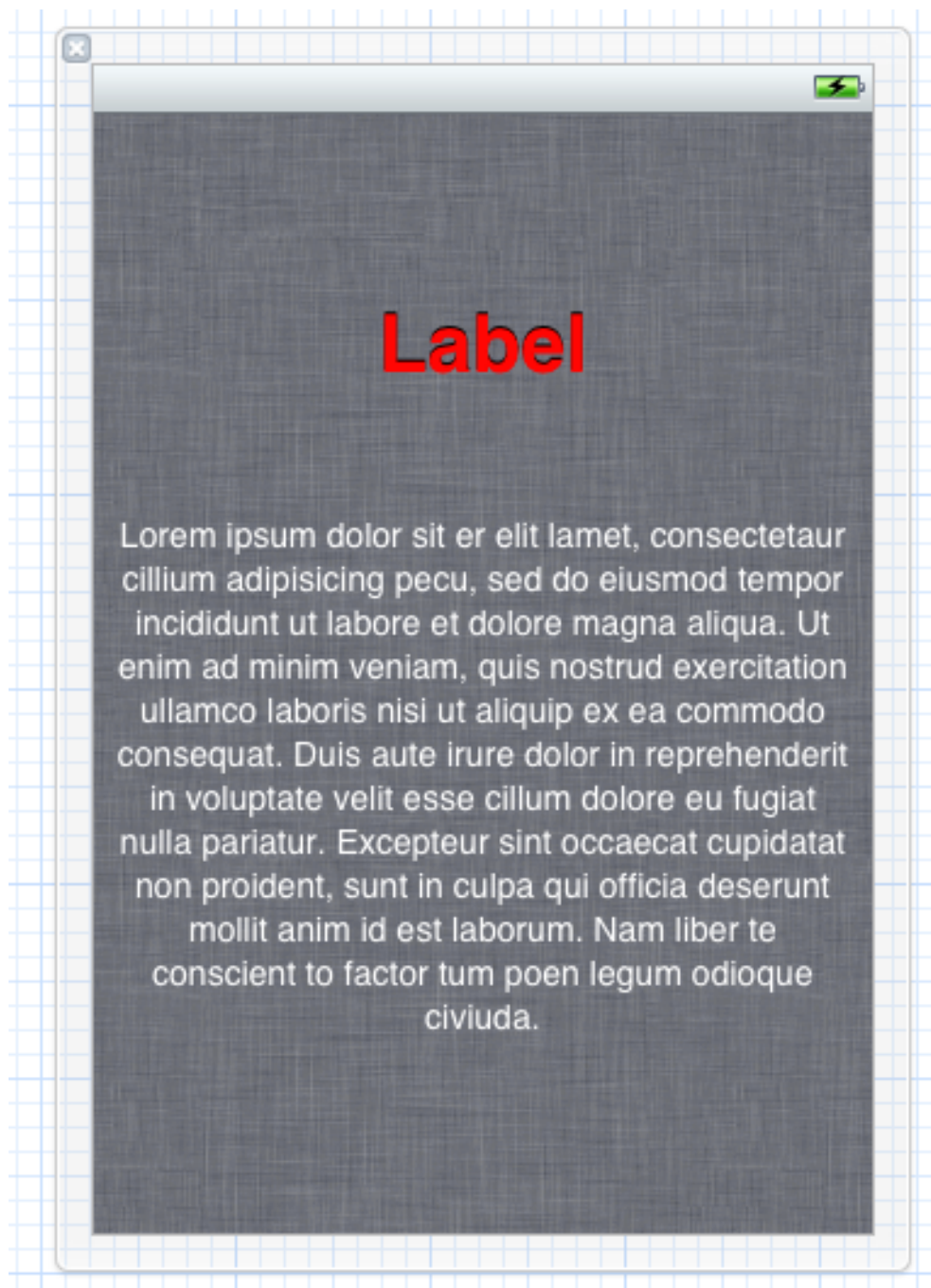
Empezamos creando un proyecto nuevo en XCode usando la plantilla Single View Application. Los datos del proyecto serán los siguientes:

- Product name: serviciosios\_sesion01\_iad
- Company Identifier: es.ua.jtech
- Class prefix: UA
- Device family: iPhone
- Seleccionar sólo "Use Automatic Reference Counting", el resto de opciones las dejamos desmarcadas.



#### Datos del proyecto

Una vez creado el proyecto en XCode vamos a modificar la vista que se crea por defecto, `UAViewController.xib` y vamos a añadirle dos campos y un fondo de color de texto quedando la vista de la siguiente forma:



UAViewController.xib

Una vez que hemos diseñado la vista principal de nuestra aplicación vamos a crear los *outlets* para relacionar los campos de texto y así poder acceder a ellos desde la controladora. Para hacer esto debemos abrir el fichero `UAViewController.h` y añadir las siguientes líneas justo antes de la etiqueta `@end`:

```

        @property (weak, nonatomic) IBOutlet UILabel
*labelEstadoiAd;
        @property (weak, nonatomic) IBOutlet UITextView *texto;

```

Ahora hacemos lo propio en el fichero de implementación `UAViewController.m`, añadimos los `@synthesize` para poder acceder a las propiedades:

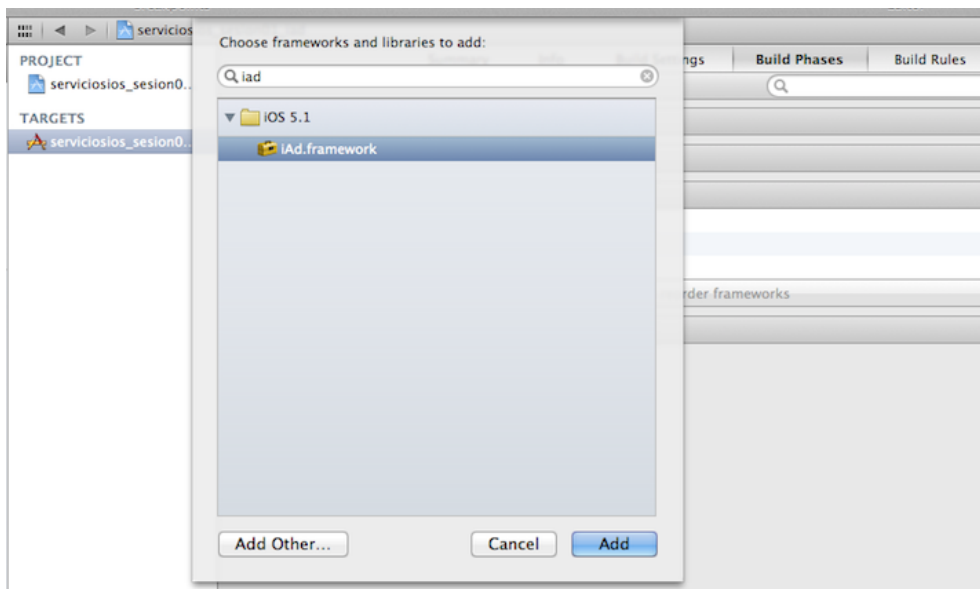
```

        @synthesize labelEstadoiAd = _labelEstadoiAd;
@synthesize texto = _texto;

```

Ahora sólo nos queda relacionar los componentes de la vista (los dos campos de texto) con los outlets. Una vez hecho esto ya tenemos la vista creada y lista para usar.

Con la vista diseñada y relacionada desde la controladora vamos a implementar el sistema de iAd. Comenzamos añadiendo el *framework* de iAd al proyecto. Esto lo hacemos desde la pestaña de Build Phases de la pantalla de información del proyecto. Deplegamos la pestaña Link Binary With Libraries y añadimos el framework `iAd.framework`.



Elegir el framework

Deberemos de modificar la controladora `UAViewController`: Añadimos la referencia a la clase `ADBannerView` que será la encargada de mostrar el banner en la vista. También creamos dos nuevas propiedades en la clase, un *flag* que nos indique si el banner está visible y otra para referenciar al propio banner. Por último debemos de indicar a la controladora que esta implementará los métodos del protocolo `ADBannerViewDelegate`, los cuales nos advertirán si se visualiza el banner, si se ha pulsado sobre el, etc. El archivo `UAViewController.h` quedaría por tanto de la siguiente manera:

```


```

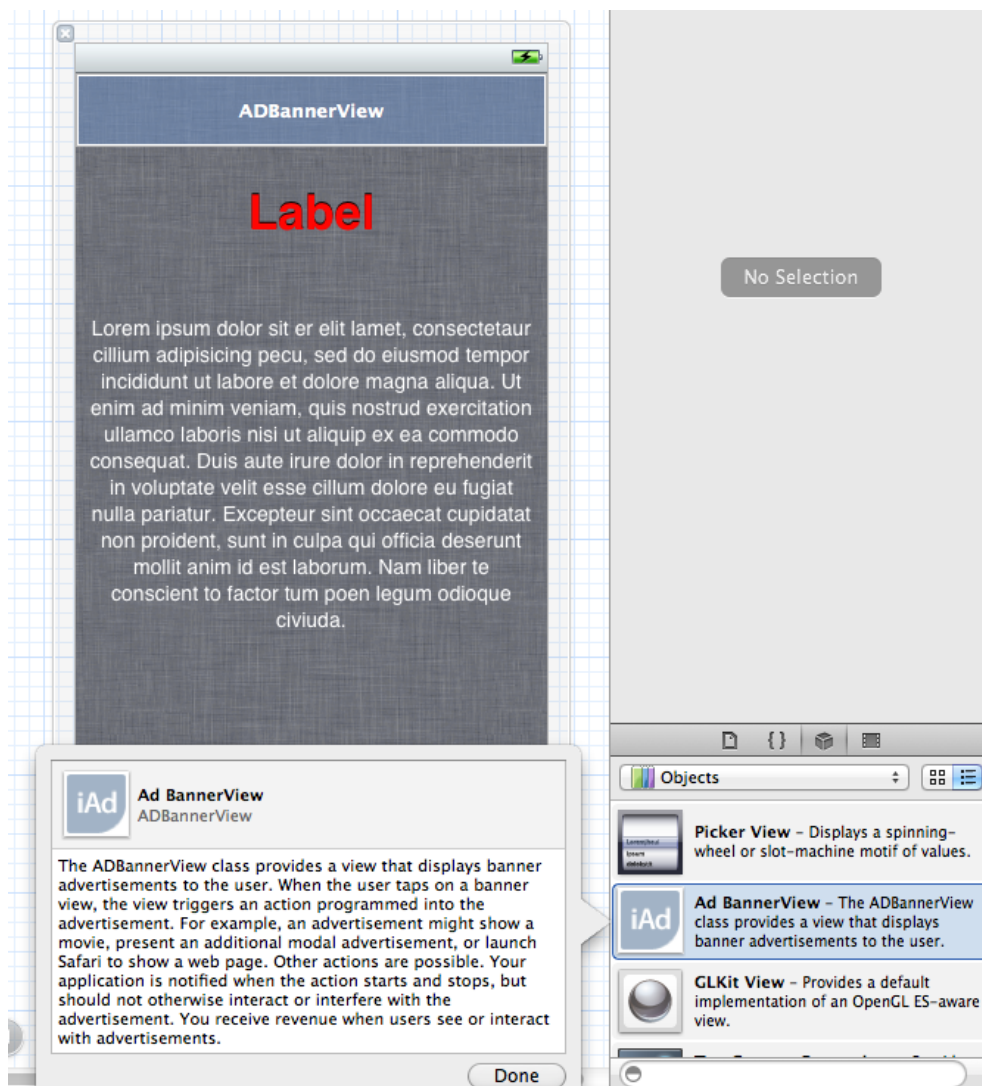
```
#import <UIKit/UIKit.h>
#import "iAd/ADBannerView.h"

@interface UAViewController : UIViewController <ADBannerViewDelegate>

@property (weak, nonatomic) IBOutlet UILabel *labelEstadoiAd;
@property (weak, nonatomic) IBOutlet UITextView *texto;
@property (nonatomic, strong) IBOutlet ADBannerView *adBannerView;
@property (nonatomic) BOOL bannerEsVisible;

@end
```

Abrimos de nuevo el archivo UAViewController.xib y *arrastramos* el objeto **Ad BannerView** a la vista situándolo en la parte superior. Una vez ajustado el banner lo relacionamos como hemos hecho con el resto de outlets quedando la vista principal de la siguiente forma. También deberemos indicar en la vista que la propiedad *delegate* del objeto Ad BannerView referencia a la controladora, para ello pulsamos la tecla *ctrl* sobre el banner y sin soltar la tecla arrastramos el cursor del ratón hasta *File's owner* y seleccionamos "delegate".



Ad BannerView

Por último nos queda completar la implementación de la controladora añadiendo los @synthesize que faltan y los métodos del protocolo ADBannerViewDelegate:

```
//synthesize despues de @end
@synthesize adBannerView = _adBannerView;
@synthesize bannerEsVisible = _bannerEsVisible;

//metodos de la clase delegada de ADBannerView
#pragma mark ADBannerViewDelegate

- (void)bannerViewDidLoadAd:(ADBannerView *)banner {
    if (!self.bannerEsVisible) {
        self.bannerEsVisible = YES;
    }
}
```



```

}

- (void)bannerView:(ADBannerView *)banner
didFailToReceiveAdWithError:(NSError *)error
{
    if (self.bannerEsVisible)
    {
        self.bannerEsVisible = NO;
    }
}

```

En primer método de los dos implementados se ejecutará cuando se cargue la vista del banner. El segundo método servirá para indicarnos si ha ocurrido algún error al recibir el banner de Apple. Para terminar modificaremos los dos métodos anteriores para que actualicen el campo de texto según el banner sea visible o no:

```

- (void)bannerViewDidLoadAd:(ADBannerView *)banner {
    if (!self.bannerEsVisible) {
        self.bannerEsVisible = YES;
        self.labelEstadoiAd.text = @"iAd funcionando";
    }
}

- (void)bannerView:(ADBannerView *)banner
didFailToReceiveAdWithError:(NSError *)error
{
    if (self.bannerEsVisible)
    {
        self.bannerEsVisible = NO;
        self.labelEstadoiAd.text = @"Error en iAd :(";
    }
}

```

Con esto último ya tenemos iAd integrado en una aplicación muy sencilla. Si arrancamos la aplicación veremos que aparecerá un banner de test el cual si pulsamos sobre el nos aparecerá una ventana emergente que con la publicidad (ahora nos aparecerá publicidad de pruebas). La publicidad real aparece únicamente cuando la aplicación está a la venta en la App Store.

Si rotamos el dispositivo podemos ver que el banner no se adapta bien a la rotación, para solucionar esto deberemos de completar la implementación añadiendo un par de métodos nuevos y modificando los del protocolo `ADBannerViewDelegate`. Comenzamos añadiendo dos métodos nuevos para gestionar la rotación del dispositivo y la de la vista del banner:

```

// Método que adapta el tamaño del banner según la rotación del
dispositivo
-
(void)rotaBannerPosicion:(UIInterfaceOrientation)toInterfaceOrientation {
    if (self.adBannerView != nil) {

```

```

        if (UIInterfaceOrientationIsLandscape(toInterfaceOrientation)) {
            [self.adBannerView setCurrentContentSizeIdentifier:
             ADBannerContentSizeIdentifierLandscape];
        } else {
            [self.adBannerView setCurrentContentSizeIdentifier:
             ADBannerContentSizeIdentifierPortrait];
        }
    }
}

// Método que se ejecuta automáticamente cuando el dispositivo rota
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
duration:(NSTimeInterval)duration {
    [self rotaBannerPosicion:toInterfaceOrientation];
}

```

Ahora añadimos una llamada al método `rotaBannerPosicion` desde los métodos del protocolo del `ADBannerView` quedando los métodos de la siguiente manera:

```

- (void)bannerViewDidLoadAd:(ADBannerView *)banner {
    if (!self.bannerEsVisible) {
        self.bannerEsVisible = YES;
        self.labelEstadoiAd.text = @"iAd funcionando";

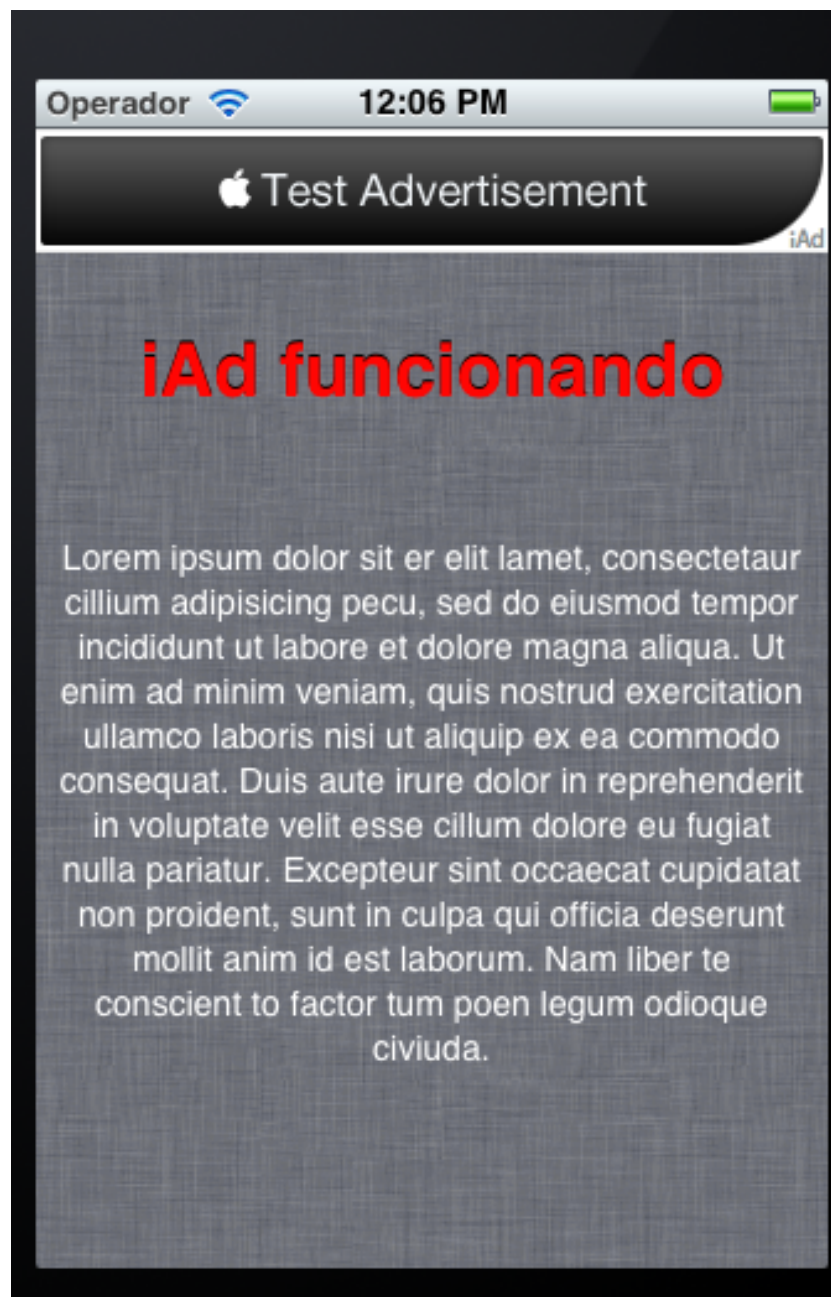
        [self rotaBannerPosicion:[UIDevice currentDevice].orientation];
    }
}

- (void)bannerView:(ADBannerView *)banner
didFailToReceiveAdWithError:(NSError *)error
{
    if (self.bannerEsVisible)
    {
        self.bannerEsVisible = NO;
        self.labelEstadoiAd.text = @"Error en iAd :(";

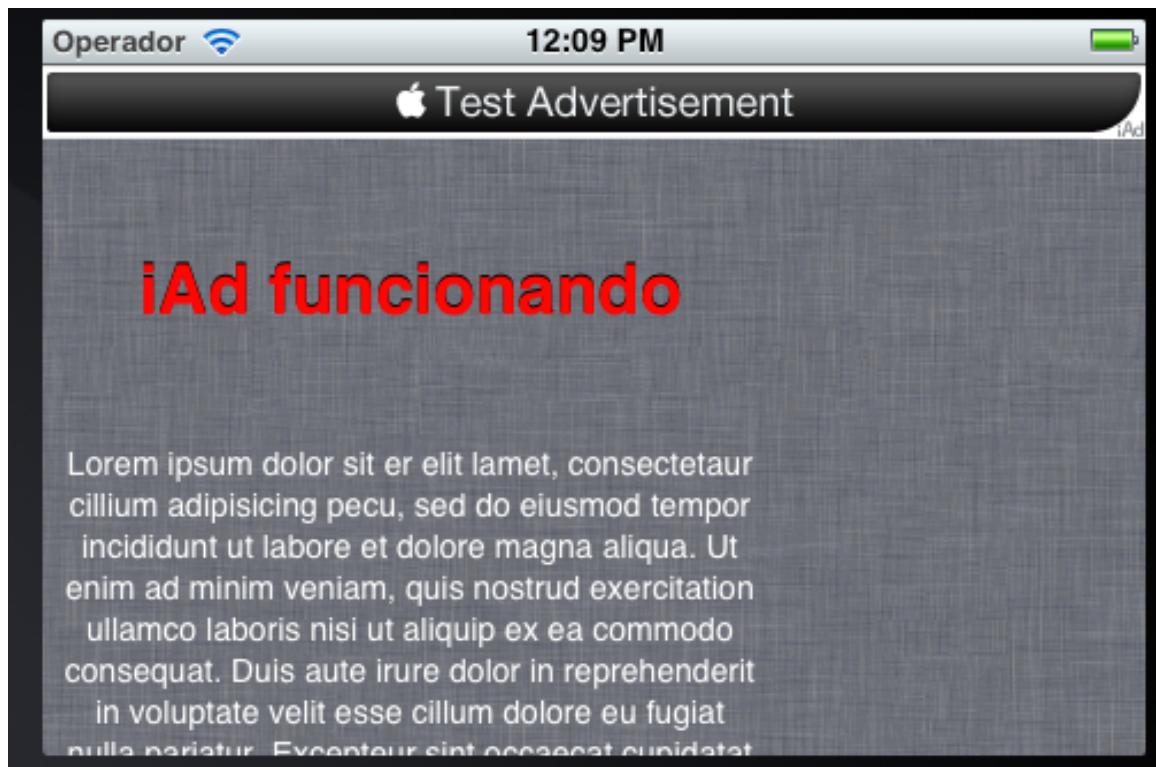
        [self rotaBannerPosicion:[UIDevice currentDevice].orientation];
    }
}

```

Si volvemos a ejecutar la aplicación veremos que el banner rota junto con el dispositivo y este se adapta al tamaño adecuado.



Vista vertical



Vista horizontal

## 2. La publicidad de Google: AdMob

En este apartado analizaremos las ventajas y desventajas de usar la plataforma de publicidad de Google para móviles **AdMob** y veremos cómo implementarla en una aplicación propia.

### 2.1. AdMob: Ventajas y desventajas

Se podría decir que, en iOS, AdMob es el complemento perfecto de iAd y es que AdMob tiene varios puntos a favor bastante destacables, vamos a analizarlos uno a uno:

Por un lado, en contra de lo que ocurre con iAd, la publicidad de AdMob está disponible en la mayoría de países y de ellos las impresiones son casi al 100%, por lo que en el caso de implementarlo nos aseguraremos que siempre o casi siempre nos aparecerá el banner, algo que no podemos asegurar a día de hoy con iAd.

Google es el encargado de la plataforma AdMob, lo que nos asegura una fiabilidad y estabilidad bastante importante. También disponemos de una API bastante documentada y diversos tutoriales para implementar AdMob en nuestras aplicaciones, lo que nos facilita el trabajo.

Otro punto a favor tiene que ver con la compatibilidad de la plataforma, y es que AdMob es, aparte de iOS, compatible con Android, por lo que podremos tener asociada una misma cuenta para distintas plataformas de móviles.

Por otro lado, una desventaja que tiene AdMob es que no está integrada con el sistema de desarrolladores de Apple, en concreto con el portal de *iTunes Connect*, por lo que todas las estadísticas de clicks, ganancias, etc. funcionan de otra forma y deberemos de gestionar dos cuentas distintas, una para AdMob y otra para Apple. Respecto a las ganancias y al reparto de beneficios de la publicidad, en AdMob se suele obtener algo menos de ganancias ya que los ratios de CPC y CPM son algo menores.

Por último comentar que la publicidad de AdMob en versiones anteriores era intrusiva para la experiencia del usuario. En la versión actual han mejorado este punto y ya funciona de manera similar a iAd: al pulsar sobre el banner de publicidad no saldrá de la aplicación y se abrirá a cambio una ventana emergente. Esto es un punto a favor, sin duda alguna.

## 2.2. Integrando AdMob

---

Con el fin de aprender a integrar AdMob en nuestras aplicaciones iOS vamos a realizar un sencillo ejemplo en el que mediante una serie de pasos sencillos nos registraremos como desarrolladores, nos descargaremos el SDK de AdMob para iOS y finalmente lo implementaremos.

Para poder hacer uso de AdMob tanto en iOS como en Android deberemos de registrarnos previamente en su sitio web, para ello abrimos un navegador y entramos en <http://www.admob.com>. Una vez dentro pulsamos sobre *Register* en la parte superior derecha, rellenamos todos los campos del formulario y pulsamos en *Submit*.

AdMob Home
Register | Log In

## AdMob

### Enriching Mobile

AdMob provides app developers tools to promote, measure and monetize mobile apps.

#### Looking to advertise on mobile?

Reach targeted audiences across platforms and devices with Google Mobile Ads. Grow online sales, send more customers to your store, or build your brand across top websites and apps with innovative mobile ad formats. [Learn more.](#)

#### Solutions for mobile web publishers

Mobile web publishers can now generate revenue from their mobile webpages with Google AdSense. Show targeted, mobile-specific ads from advertisers looking to reach your mobile audience. [Learn more.](#)

#### THE GUIDE TO THE APP GALAXY

Travel through the App Galaxy to learn how to build a business on mobile. Read top app developer success stories or create your own journey to share with others.

#### CURRENT STATS

Global Impressions Served:  
**1,270,912,091,103**

[Get Started](#)

[Log In](#)

#### News & Announcements

- SDK improvements for iOS**  
Support for additional Rich Media Interstitial ad formats and more. [Learn More](#)
- The Power of Search Ads, Now in Mobile Apps**  
Show relevant search ads to users, when they are looking, from within your app. [Learn More](#)

© 2005-2012 AdMob Google Inc | [Privacy Policy](#) | [Terms of Service](#) | [About AdMob](#) | [Contact](#) | [Blog](#) | [Site Map](#)

Language: English

## AdMob

Una vez registrados en el portal, pulsamos sobre *Sitios y aplicaciones* > *Agregar sitio/aplicación*. De nuevo deberemos de rellenar todos nuestros datos de la empresa y de la cuenta personal del banco para los ingresos.

The screenshot shows the AdMob account setup interface. At the top, the AdMob logo is visible with the text 'by Google'. Below the logo is a navigation bar with tabs: 'Campañas', 'Sitios y aplicaciones', 'Informes', 'Herramientas', and 'Cuenta'. The 'Cuenta' tab is selected. Below the navigation bar, there are two sub-tabs: 'Cuenta' and 'Detalles de pago'. The 'Detalles de pago' tab is active. A message box at the top of the 'Detalles de pago' section states: 'Utilice solo caracteres ingleses para rellenar la información sobre pagos de la cuenta.' Below this, a red banner reads: 'Antes de crear un sitio, rellene la información de contacto y preferencias de pago.' The main form is divided into two sections: 'Información impositiva' and 'Detalles de pago'. The 'Información impositiva' section includes fields for 'País' (set to 'España'), 'Tipo de cuenta' (with a dropdown menu), 'Nombre de empresa', and 'ID impositiva local'. The 'Detalles de pago' section includes radio buttons for 'Pago por ACH/transferencia' (selected) and 'Pagar a través de PayPal', followed by fields for 'Nombre del titular', 'Nombre del banco', 'Dirección de la entidad bancaria', 'Número de cuenta del titular/IBAN', and 'Código SWIFT'. At the bottom of the form are 'Enviar' and 'Cancelar' buttons.

### AdMob

Una vez rellenado el formulario anterior nos aparecerá una nueva ventana para elegir la plataforma de la aplicación que vamos a realizar. Ahi seleccionamos *Aplicación de iPhone*



## AdMob

Ahora rellenamos los detalles de la aplicación (nombre, categoría, descripción, estilo de los banners...). En el nombre de la aplicación escribiremos `sesion01 admob`. Completamos el resto de datos y pulsamos en *Continuar*.





**Detalles**

Nombre de aplicación:

URL de App Store:

Categoría:

Descripción de aplicación:


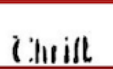
Tema:  El texto del anuncio se colocará aquí. 


☒ Negro ☐ Personalizado

Fondo: # (eg. #0066FF)

Texto: # (eg. #0066FF)

☐ Mi aplicación utiliza información de ubicación



**Actualización del SDK de AdMob**

Todas las nuevas aplicaciones para Android, iPad, iPhone y Windows Phone 7 necesitan un SDK de AdMob cuya fecha de lanzamiento sea el 15 de marzo de 2011 o una posterior.

Si la versión del SDK de AdMob que está utilizando es anterior al 15 de marzo de 2011, su nueva aplicación no recibirá ninguna impresión de anuncio.

## AdMob

Una vez completados los datos básicos de la aplicación vamos a descargarnos el SDK de AdMob. Para descargar el SDK debemos pulsar en *Descargar AdMob iOS SDK*. También podemos acceder a la documentación entrando en el enlace que hay justo debajo del botón de descarga.

Sitios y aplicaciones ☒ Agregar sitio/aplicación nuevos

**Agregar sitios/aplicaciones nuevos**

Información del sitio ☒ Obtener código del sitio

**Código de instalación - sesion01 admob**

 Esta aplicación necesita un SDK de AdMob cuya fecha de lanzamiento sea igual al 15 de marzo de 2011 o una posterior. Antes de enviar solicitudes con esta aplicación, descargue la última versión del SDK de AdMob.

**AdMob iOS SDK incluye:**

1. **LÉAME:** Empezar a usar los anuncios de AdMob iOS.
2. **AdMob SDK:** Necesario para publicar anuncios. Colóquelo en el proyecto de Xcode.

<http://code.google.com/mobile/ads/docs/ios/>

## AdMob

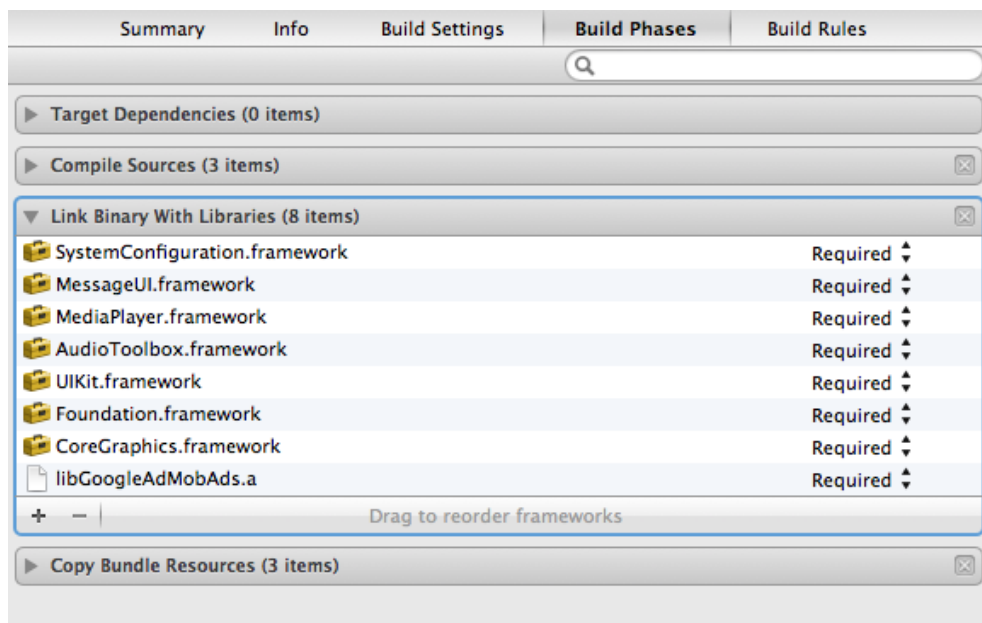
Con esto ya nos hemos creado una cuenta en AdMob y nos hemos descargado la última

versión del SDK para iOS. Ahora vamos a crear un nuevo proyecto en XCode usando la plantilla Single View Application y que tendrá los siguientes datos:

- Product name: serviciosios\_sesion01\_admob
- Company Identifier: es.ua.jtech
- Class prefix: UA
- Device family: iPhone
- Seleccionar sólo "Use Automatic Reference Counting", el resto de opciones las dejamos desmarcadas.

Ahora, con el proyecto abierto, arrastramos la carpeta completa del SDK de AdMob a la raíz del proyecto seleccionando la opción Copy items into destination group's folder (if needed). También deberemos de añadir al proyecto los siguientes frameworks:

AudioTollbox.framework,  
MediaPlayer.framework, MessageUI.framework y  
SystemConfiguration.framework.



AdMob

Una vez añadidos los frameworks necesarios y el SDK de AdMob vamos a "comenzar" con el desarrollo del contenido de la aplicación. Vamos a modificar la vista principal `UINavigationController.xib` añadiéndole, al igual que en el ejemplo de iAd, dos campos de texto y un fondo, la vista deberá quedar de la siguiente manera:



AdMob

Ahora vamos a modificar la controladora `UAViewController` para añadirle el código necesario del banner. El archivo `UAViewController.h` debe quedar como sigue:

```

        #import <UIKit/UIKit.h>
#import "GADBannerView.h"

@interface UAViewController : UIViewController

@property (strong, nonatomic) GADBannerView *AdMob;

@end

```

Ahora modificamos el archivo `UAViewController.m` añadiendo el siguiente código al método `viewDidLoad`:

```

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from
    a nib.

    // Posicionamos el banner en la parte inferior de la vista
    self.AdMob = [[GADBannerView alloc] initWithFrame:
        CGRectMake(0.0, self.view.frame.size.height -
                    GAD_SIZE_320x50.height,
                    GAD_SIZE_320x50.width,
                    GAD_SIZE_320x50.height)];

    self.AdMob.adUnitID = AdMob_ID;

    // Añadimos el banner a la vista
    self.AdMob.rootViewController = self;
    [self.view addSubview:self.AdMob];

    // Hacemos una petición de test para el banner
    GADRequest *r = [[GADRequest alloc] init];
    r.testing = YES;
    [self.AdMob loadRequest:r];
}

```

También deberemos definir nuestro AdMob ID y el `@synthesize` del banner:

```

// Debajo del import
#define AdMob_ID @"a14f52699ef091a" // Nuestro Publisher
ID (www.admob.com)

// Debajo de la etiqueta implementation
@synthesize AdMob = _AdMob;

```

Una vez hechos estos últimos cambios ya podemos compilar y arrancar nuestra aplicación. Si todo ha ido bien veremos el banner de prueba en la parte inferior de la vista.



AdMob

Si queremos que el banner aparezca en la parte superior de la pantalla, al igual que hicimos en el ejemplo de iAd, deberemos de modificar ligeramente el código de posicionamiento del método `viewDidLoad` de la siguiente manera:

```
self.AdMob = [[GADBannerView alloc]
initWithFrame:CGRectMake(0.0, 0.0,
GAD_SIZE_320x50.width,
GAD_SIZE_320x50.height)];
```





AdMob

Si queremos que el banner rote al mismo tiempo que giramos el dispositivo deberemos de implementarlo nosotros. Podemos acceder a más información sobre la API de AdMob así como a código de ejemplo en esta dirección: <https://developers.google.com/mobile-ads-sdk/docs/ios/fundamentals>.

### 3. Otras plataformas de publicidad en iOS

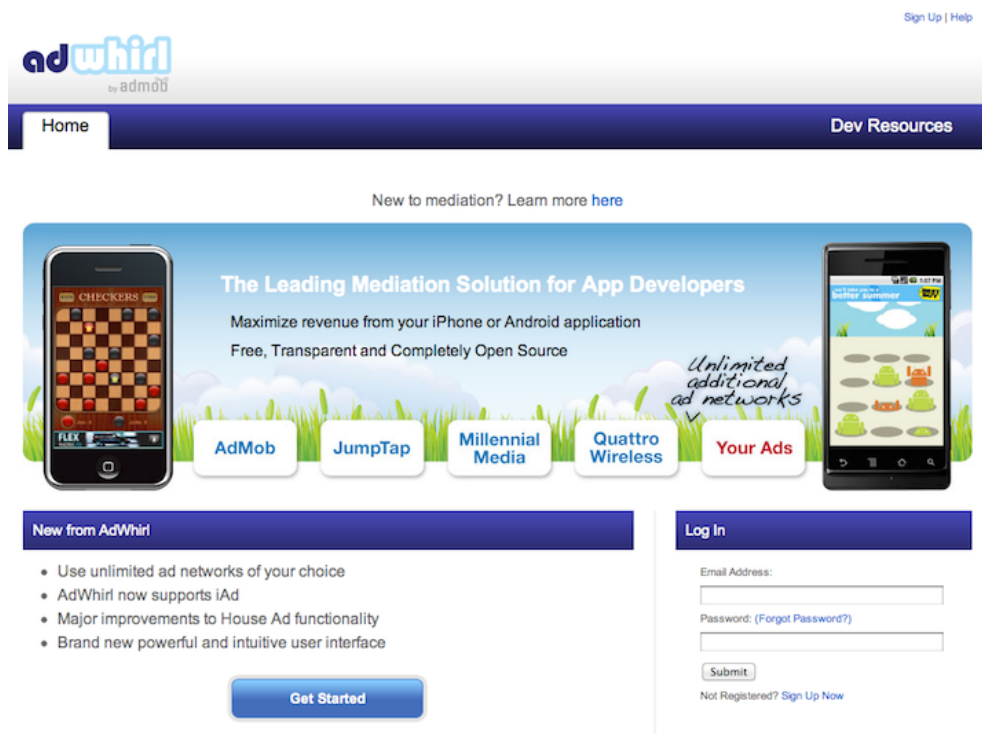
Además de iAd (Apple) y de AdMob (Google) existen otras plataformas de publicidad de las cuales podemos hacer uso en nuestras aplicaciones. Una de las más conocidas es AdSense (también de Google) y MobClix. Esta última se suele usar como complemento a iAd o AdMob ya que sus ratios de CPM y CPC son bastantes pequeños por lo que los beneficios que obtenemos son pocos.

AdSense en cambio ofrece unos porcentajes de ganancias algo mayores pero tiene el inconveniente que la publicidad que ofrece es muy poco atractiva: banners con fondo blanco o negro y con texto, sin imágenes. Esta plataforma suele ser muy poco efectiva y, por lo tanto, poco recomendada para usar como sistema de publicidad principal.

Como hemos visto en los puntos anteriores, muchas veces tenemos el problema de la falta de conexión a Internet o falta de disponibilidad de banners. Esto provoca que la publicidad no se muestre en la aplicación y por lo tanto, no obtengamos ganancias de esta. Para evitar esto deberemos de implementar un "plan alternativo" que nos asegure que siempre se va a mostrar la publicidad.

Tenemos dos opciones para implementar este plan alternativo: programarlo en nuestra aplicación nosotros mismos o utilizar una API que nos ayude a hacerlo. Esta última opción es la que vamos a analizar con un poco más de profundidad.

Existen varios sistemas online que ofrecen un servicio de "rotación" de publicidad entre distintas plataformas, entre ellas destaca para móviles **AdWhirl** (<https://www.adwhirl.com/>). AdWhirl un sistema que se encarga dinámicamente y de forma totalmente automática de cambiar entre redes de publicidad según la preferencia que indiquemos nosotros y la disponibilidad de estas.



Imagen

AdWhirl además destaca porque podemos incorporar nuestros propios banners y disponemos de toda la gestión de las plataformas de publicidad desde un mismo panel de control, lo que nos facilita enormemente la tarea de gestión de estas.

En AdWhirl podemos añadir la inmensa mayoría de redes de publicidad, entre ellas están: iAd, AdMob, JumTab, Millennial Media, Quattro Wireless, Mobclix, AdSense, etc... Una vez registrados en el sistema (totalmente gratuito) deberemos escoger las redes que queremos usar en nuestra aplicación y asignarles una preferencia. Esta preferencia se tendrá en cuenta a la hora de mostrar la publicidad, si una red no tiene disponibilidad de banners pasará a la siguiente con más preferencia.

Obviamente deberemos registrarnos previamente en aquellas plataformas (o redes) de publicidad que vayamos a utilizar en AdWhirl, y desde el panel de control añadiremos las claves de cada una.

#### 4. Micropagos: In-Apps

Una vez visto el modelo de negocio de publicidad vamos a analizar en profundidad otro modelo bastante usado entre los desarrolladores de Apple: **in-apps**.





In-app

Los in-apps o *micropagos* son compras que se realizan dentro de la aplicación. Una aplicación de iOS puede tener tantos in-apps como quiera del desarrollador y cada uno de estos al precio que se desee (según las tarifas de Apple). Existen dos tipos de in-apps: consumibles y no consumibles.

- **Consumibles:** Aquellos que el usuario puede comprar más de una vez, por ejemplo: dinero virtual en un juego social.
- **No consumibles:** Aquellos que el usuario sólo puede comprar una vez, por ejemplo: desbloquear un nivel en un juego.

Lo más habitual es implementar el sistema de in-apps en aplicaciones gratuitas ya que de este modo conseguiremos muchos más descargas y, por lo tanto, más clientes potenciales de in-apps.

La implementación de in-apps en una aplicación iOS es algo compleja y requiere de una cuenta activa como desarrollador en Apple. En el siguiente apartado realizaremos un ejemplo paso a paso de incorporación de un in-app en una aplicación para iPhone.

#### 4.1. Implementando los in-apps: App freemium

En este apartado veremos en más detalle como incorporar los in-apps en nuestras aplicaciones. Para ello realizaremos un ejemplo de una aplicación *freemium* o lo que es lo mismo: aplicación reducida con opción a compra para desbloquear la versión completa). La aplicación la implementaremos paso a paso durante los cuales veremos como crear los in-apps dentro del portal de *iTunes Connect* y como programarlos en modo de test o *sandboxing* en una aplicación sencilla. ¡Empezamos!

El ejemplo que vamos a realizar consistirá en una sencilla aplicación iPhone la cual estará formada por dos vistas: una principal a la que todo el mundo tendrá acceso y otra a la que sólo se podrá acceder si hemos comprado un in-app que costará 0.79 euros, una pantalla "secreta". Este in-app será de tipo **no consumible**, es decir, sólo podremos comprarlo una vez y tendremos que tener en cuenta la compra para que cada vez que el usuario arranque la aplicación compruebe si ya ha comprado el in-app.

#### 4.1.1. Preparación: Apple Developer y iTunes Connect

Antes de empezar con la programación de la aplicación en XCode vamos a crear un **App ID** para nuestra aplicación. El App ID es una cadena de texto única que identifica cada una de nuestras aplicaciones, en este caso crearemos un App ID dentro del portal de desarrolladores (<https://developer.apple.com>) con el nombre Ejemplo in app y el identificador (*Bundle ID*) es.ua.jtech.ejemploinapp.

Provisioning Portal

Antes de empezar con la programación de la aplicación en XCode vamos a crear los in-apps dentro del portal iTunes Connect. Conviene recordar que para acceder a este portal deberemos de tener una cuenta activa como desarrollador Apple. Accedemos al portal (<https://itunesconnect.apple.com>) usando nuestro Apple ID y contraseña.

Una vez dentro entraremos dentro del apartado Manage Your Applications y creamos una nueva aplicación pulsando en Add New App.



## Manage Your Applications

Add, view, and manage your applications in the iTunes Store.

### Manage Your Applications

El formulario que nos aparece lo completamos según la siguiente imagen y pulsamos en "Continue".

**App Information**

Enter the following information about your app.

Default Language: Spanish

App Name: Ejemplo In App

SKU Number: 99

Bundle ID: Ejemplo in app - es.ua.jtech.ejemploinapp2

You can register a new Bundle ID [here](#).

**Note:** Note that the Bundle ID cannot be changed if the first version of your app has been approved or if you have enabled Game Center or the iAd Network.

Does your app have specific device requirements? [Learn more](#)

[Cancel](#) [Continue](#)

### App Informaion

Seguimos completando el resto de formularios básicos de creación de una aplicación nueva en iTunes Connect... Una vez hecho esto (nos llevará un tiempo ya que debemos de escribir una descripción, subir unas imágenes básicas, etc.) pasaremos a crear los in-apps.

Una vez que hayamos guardado el formulario anterior pulsamos sobre el botón "Manage In-App Purchases" para comenzar con la creación de los in-apps.

**Ejemplo In App**

App Information [Edit](#)

Identifiers	Links
SKU: 99	<a href="#">View in App Store</a>
Bundle ID: es.ua.jtech.ejemploinapp2	
Apple ID: 507883571	
Type: iOS App	
Default Language: Spanish	

- [Rights and Pricing](#)
- [Manage In-App Purchases](#)
- [Manage Game Center](#)
- [Set Up iAd Network](#)
- [Newsstand](#)
- [Delete App](#)

### Manage In-App Purchases

En el formulario que nos aparece ahora deberemos de escoger entre cuatro tipos distintos de in-apps, los dos primeros son los que hemos comentado anteriormente (consumibles y no consumibles) y los otros dos son exclusivos para aplicaciones del quiosco (*Newstand*). Nosotros escogeremos el segundo tipo, **Non Consumable** ya que una vez que el usuario compre el in-app este se bloqueará y no tendrá que comprarlo más veces.

**Ejemplo In App — In-App Purchases**

**Select Type**

Select the In-App Purchase type you want to create. If there is a type that appears to be missing, it is likely that you have not signed the most recent contract(s). Go to the [Contracts, Tax, and Banking](#) module in iTunes Connect and accept the latest Paid Applications agreement. Note that to access the Paid Applications agreement, you will first need to accept the latest version of the [Developer Program License Agreement](#) (if you haven't already).

**Consumable**

A consumable In-App Purchase must be purchased every time the user downloads it. One-time services, such as fish food in a fishing app, are usually implemented as consumables.

[Select](#)

**Non-Consumable**

A non-consumable In-App Purchase only needs to be purchased once by the user. Services that do not expire or decrease with use, such as a new race track for a game app, are usually implemented as non-consumables.

[Select](#)

### In app

Ahora deberemos de completar todos los campos del formulario sobre el in-app. Escribiremos un nombre único para identificarlo, una breve descripción en al menos un idioma, un precio (elegimos la primera tarifa, *tier 1*) y una imagen. Rellenaremos el formulario tal y como se muestra en la siguiente imagen:

### Ejemplo In App — In-App Purchases

**Details**

Enter a reference name and a product ID for this In-App Purchase. You must also add at least one language, along with a display name and a description in that language.

Reference Name  ?

Product ID  ?

[Add Language](#)

Language	Display Name	Description	
Spanish	Desbloquear pantalla sorpresa	Ejemplo de in-apps	<a href="#">Delete</a>

**Pricing and Availability**

Enter the pricing and availability details for this In-App Purchase below.

Cleared for Sale ☒ Yes ☐ No

Price Tier  ?


[View Pricing Matrix](#)

App Store	U.S.*	Mexico	Canada	U.K.	European Union*	Sweden	Denmark	Norway	Switzerland	Australia	New Zealand	Japan	China
Customer Price	US\$0.99	\$12.00	CA\$0.99	£0.69	0,79 €	7,00kr	6,00kr	7,00kr	1.00Fr	AU\$0.99	NZ\$1.29	JP¥85	CN ¥6.00
Your Proceeds	US\$0.70	MX \$8.40	CA\$0.70	£0.42	0,48 €			3,92kr	0.65Fr	AU\$0.63	NZ\$0.90	JP¥60	CN ¥4.20

\*The U.S. price applies to all countries where apps are sold in U.S. dollars. The European Union price applies to all countries where apps are sold in euros. [See Details.](#)

**Screenshot for Review**

Before you submit your In-App Purchase for review, you must upload a screenshot. This screenshot will be for review purposes only. It will not be displayed on the App Store. Screenshots must be at least 640x920 pixels and at least 72 DPI.



[Choose File](#)

[Cancel](#)
[Save](#)

### In-App configuración

Una vez completado el formulario anterior ya tenemos el in-app creado y configurado en iTunes Connect. Ahora deberemos de implementarlo en nuestra aplicación, pero antes vamos a diseñar toda la estructura básica.

#### 4.1.2. Implementación de toda la estructura básica

Comenzamos creando un proyecto nuevo en XCode con la plantilla Single View Application al que llamaremos **serviciosios\_sesion01\_inapps** y tendrá los siguientes datos:

- Product name: serviciosios\_sesion01\_inapps
- Company Identifier: es.ua.jtech
- Class prefix: UA
- Device family: iPhone
- Seleccionar sólo "Use Automatic Reference Counting", el resto de opciones las dejamos desmarcadas.

Una vez creado el proyecto en XCode vamos a modificar la vista principal `UAViewController.xib` añadiendo una etiqueta de texto y dos botones, uno para comprar el in-app y otro para abrir la pantalla "secreta":



Vista

Ahora programamos la controladora principal de la aplicación. Abrimos el archivo `UAViewController.h` y lo dejamos de la siguiente manera:

```
#import <UIKit/UIKit.h>

@interface UAViewController : UIViewController
```

```

@property (weak, nonatomic) IBOutlet UIButton *botonCompra;
@property (weak, nonatomic) IBOutlet UIButton *botonAbreVentanaSecreta;

-(IBAction)clickBotonCompra:(id)sender;
-(IBAction)clickBotonVentanaSecreta:(id)sender;

@end

```

Ahora modificamos el archivo `UAViewController.m` añadiendo los `@synthesize` y las acciones de los botones:

```

//Debajo de @implementation
@synthesize botonCompra = _botonCompra;
@synthesize botonAbreVentanaSecreta = _botonAbreVentanaSecreta;

// Añadimos los siguientes métodos
-(IBAction)clickBotonCompra:(id)sender {
    NSLog(@"clickBotonCompra");
}

// Código de lanzamiento del in-app

-(IBAction)clickBotonVentanaSecreta:(id)sender {
    NSLog(@"clickBotonVentanaSecreta");
}

// Código para abrir la ventana secreta

```

Ahora vamos a diseñar la "ventana secreta", para ello vamos a crear un nuevo objeto de tipo `UIViewController Subclass` que llamaremos `VentanaSecretaViewController`. Abrimos la vista y la editamos a nuestro gusto, por ejemplo:





VentanaSecretaViewController

Una vez hecho esto ahora nos queda relacionar los *Outlets* y las acciones de la vista principal, esto lo hacemos desde la propia vista, y completar la acción del botón que abre la ventana secreta:

```
-(IBAction)clickBotonVentanaSecreta:(id)sender {
```

```
NSLog(@"clickBotonVentanaSecreta");

VentanaSecretaViewController *ventanaSecreta =
[[VentanaSecretaViewController alloc]
 initWithNibName:@"VentanaSecretaViewController" bundle:nil];

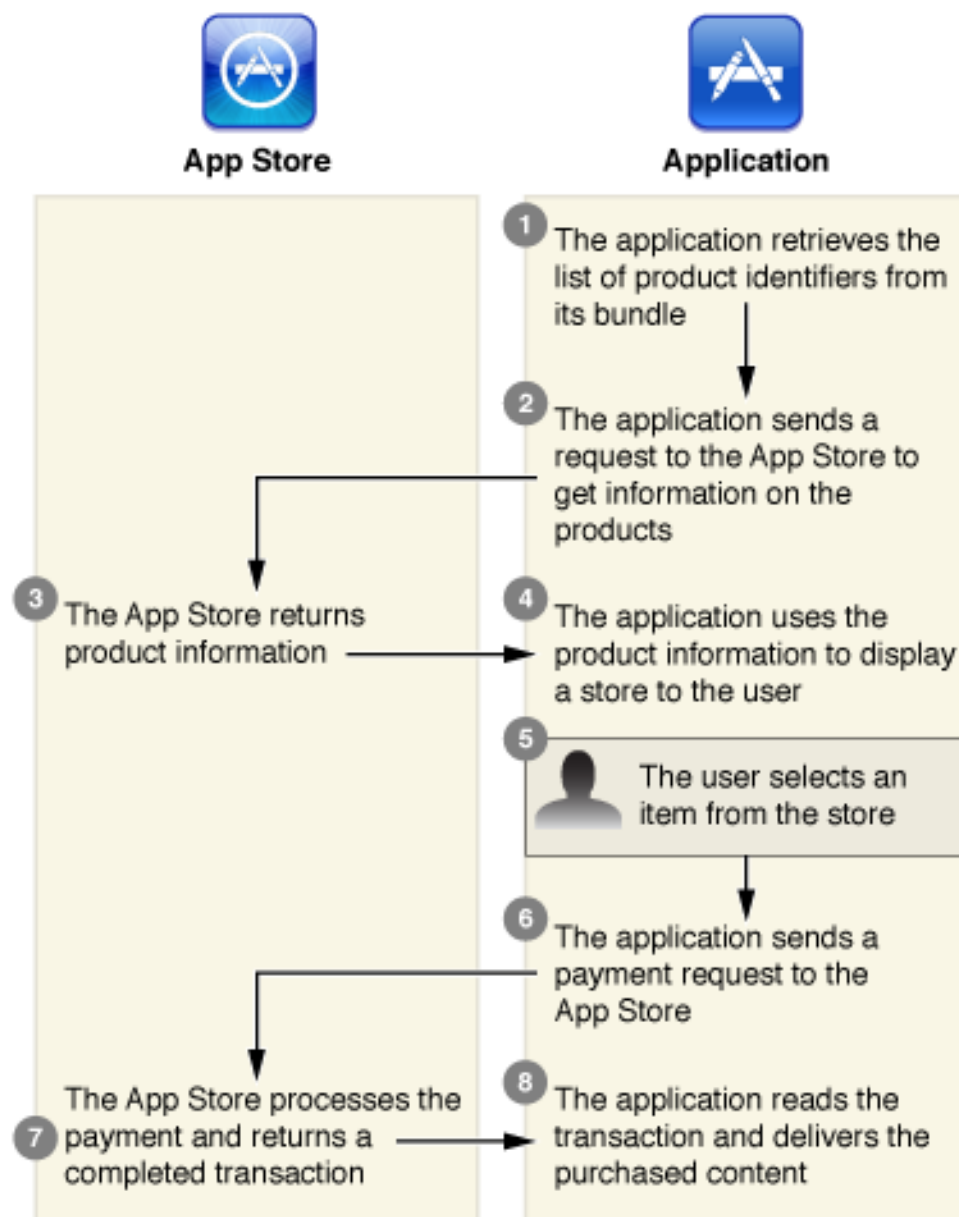
[self presentViewController:ventanaSecreta animated:YES];
}
```

Comprobamos que la aplicación funciona correctamente.

#### 4.1.3. Implementación del in-app

---

Una vez implementado el funcionamiento básico de la aplicación (sin in-apps) vamos a incorporar el in-app para desbloquear la ventana secreta. Comenzamos añadiendo el framework `StoreKit` al proyecto y modificando el fichero `UAViewController.h` importando la clase `StoreKit` y añadiendo el protocolo `SKProductsRequestDelegate` en la definición de la controladora para que esta implemente sus métodos delegados. Este sería el flujo de información entre la aplicación nuestra y la App Store que vamos a usar:



Flujo de información con la App Store

El código que implementará el in-app quedaría como sigue:

```
#import <UIKit/UIKit.h>
#import <StoreKit/StoreKit.h>

@interface UAViewController : UIViewController
<SKProductsRequestDelegate, SKPaymentTransactionObserver>

@property (weak, nonatomic) IBOutlet UIButton *botonCompra;
```

```

@property (weak, nonatomic) IBOutlet UIButton *botonAbreVentanaSecreta;

@property(n nonatomic, retain) NSMutableArray *productIdentifierList;
@property(n nonatomic, retain) NSMutableArray *productDetailsList;

- (IBAction)clickBotonCompra:(id)sender;
- (IBAction)clickBotonVentanaSecreta:(id)sender;

@end

```

Ahora tenemos que modificar el archivo `UAViewController.m` añadiendo las clases delegadas del protocolo `SKProductsRequestDelegate` y del *observer* `SKPaymentTransactionObserver`. El código final quedaría de la siguiente manera:

```

#import "UAViewController.h"
#import "VentanaSecretaViewController.h"

@implementation UAViewController
@synthesize botonCompra = _botonCompra;
@synthesize botonAbreVentanaSecreta = _botonAbreVentanaSecreta;
@synthesize productIdentifierList = _productIdentifierList;
@synthesize productDetailsList = _productDetailsList;

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    self.productIdentifierList = [[NSMutableArray alloc] init];
    self.productDetailsList = [[NSMutableArray alloc] init];

    [self.productIdentifierList addObject:@"ejemplo3"];

    // Cargamos la lista de productos
    SKProductsRequest *request = [[SKProductsRequest alloc]
    initWithProductIdentifiers:[NSSet
    initWithArray:self.productIdentifierList]];
    request.delegate = self;
    [request start];

    // Comprobamos si hemos comprado antes el in-app
    if ([[NSUserDefaults standardUserDefaults]
    boolForKey:@"inappComprado"] == YES){
        self.botonAbreVentanaSecreta.hidden = NO;
    }
    else {
        self.botonAbreVentanaSecreta.hidden = YES;
    }

    // Reinicia cualquier transacción si esta se interrumpió
    [[SKPaymentQueue defaultQueue] addTransactionObserver:self];

    [super viewDidLoad];
}

```

```

}

- (void)viewDidUnload
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
}

- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
}

- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
}

- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
}

- (BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
    return (interfaceOrientation !=
    UIInterfaceOrientationPortraitUpsideDown);
}

#pragma mark Acciones de los botones

- (IBAction)clickBotonCompra:(id)sender {
    NSLog(@"clickBotonCompra");

    // Código de lanzamiento del in-app
    if ([self.productDetailsList count] > 0 && [SKPaymentQueue
canMakePayments]){
        SKProduct *thisProduct = [self.productDetailsList
objectAtIndex:0];

        SKPayment *payment = [SKPayment paymentWithProduct:thisProduct];
        [[SKPaymentQueue defaultQueue] addPayment:payment];

        NSLog(@"Comprando...");
    }
    else {
        NSLog(@"No existen productos");
    }
}

- (IBAction)clickBotonVentanaSecreta:(id)sender {
    NSLog(@"clickBotonVentanaSecreta");

    VentanaSecretaViewController *ventanaSecreta =
[[VentanaSecretaViewController alloc]
initWithNibName:@"VentanaSecretaViewController" bundle:nil];

```

```

        [self presentViewController:ventanaSecreta animated:YES];
    }

#pragma mark - InApps métodos

/*
 * METODO QUE SE ACTIVA CUANDO RECIBIMOS LOS PRODUCTOS QUE TENEMOS EN
 IN-APP
 */
-(void)productsRequest:(SKProductsRequest *)request
didReceiveResponse:(SKProductsResponse *)response
{
    NSLog(@"didReceiveResponse. Total productos: %d", [response.products
count]);

    [self.productDetailsList addObjectsFromArray: response.products];

    for (NSString *invalidProductId in response.invalidProductIdentifiers)
    {
        NSLog(@"Producto invalido id: %@", invalidProductId);
    }
}

/*
 * CARGA DE PRODUCTOS TERMINADA
 */
-(void)requestDidFinish:(SKRequest *)request
{
}

/*
 * FALLO CON LA CARGA DE LOS PRODUCTOS
 */
-(void)request:(SKRequest *)request didFailWithError:(NSError *)error
{
    NSLog(@"Failed to connect with error: %@", [error
localizedDescription]);
}

/*
 * METODO QUE DESBLOQUEA EL BOTÓN
 */
- (void)desbloqueaVentanaSecreta {
    NSLog(@"Actualiza flag de in app comprado");
    [[NSUserDefaults standardUserDefaults] setBool:NO
forKey:@"inappComprado"];
    [[NSUserDefaults standardUserDefaults] synchronize];

    // Activamos el boton
    self.botonAbreVentanaSecreta.hidden = NO;
}

/*
 * METODO QUE SE ACTIVA CUANDO SE COMPLETA UNA COMPRA
 */
- (void)completeTransaction:(SKPaymentTransaction *)transaction {
    NSLog(@"completeTransaction...");

    [self desbloqueaVentanaSecreta];
}

```

```
        [[SKPaymentQueue defaultQueue] finishTransaction: transaction];
    }

    /*
     * METODO QUE SE ACTIVA CUANDO SE CONTINUA CON UNA COMPRA
     */
    - (void)restoreTransaction:(SKPaymentTransaction *)transaction {
        NSLog(@"restoreTransaction...");

        [self desbloqueaVentanaSecreta];
        [[SKPaymentQueue defaultQueue] finishTransaction: transaction];
    }

    /*
     * METODO QUE SE ACTIVA CUANDO UNA COMPRA FALLA
     */
    - (void)failedTransaction:(SKPaymentTransaction *)transaction {
        if (transaction.error.code != SKErrorPaymentCancelled)
        {
            NSLog(@"Transaction error: %@",
transaction.error.localizedDescription);
        }

        [[SKPaymentQueue defaultQueue] finishTransaction: transaction];
    }

    /*
     * METODO QUE SE ACTIVA CUANDO REALIZAMOS UNA TRANSACCIÓN
     */
    - (void)paymentQueue:(SKPaymentQueue *)queue
updatedTransactions:(NSArray *)transactions
    {
        NSLog(@"updatedTransactions...");

        for (SKPaymentTransaction *transaction in transactions)
        {
            switch (transaction.transactionState)
            {
                case SKPaymentTransactionStatePurchased:
                    [self completeTransaction:transaction];
                    break;
                case SKPaymentTransactionStateFailed:
                    [self failedTransaction:transaction];
                    break;
                case SKPaymentTransactionStateRestored:
                    [self restoreTransaction:transaction];
                default:
                    break;
            }
        }
    }

@end
```

### Atención

El bundle ID de la aplicación en XCode debe de coincidir con el Bundle identifier que hemos especificado al crear la aplicación en iTunes Connect.

Si ejecutamos el código nos debería de aparecer el botón de la ventana secreta desactivado. Cuando pulsemos sobre el botón de comprar in-app nos pedirá un nombre de usuario y contraseña, ahí tendremos que escribir nuestro usuario de pruebas que previamente hemos creado dentro de la sección *Manage Users* del portal iTunes Connect.

En el momento de confirmar la compra se ejecutará el método `updatedTransactions` el cual activará, si todo ha ido bien, el botón para mostrar la ventana secreta.





Confirmación in-app

