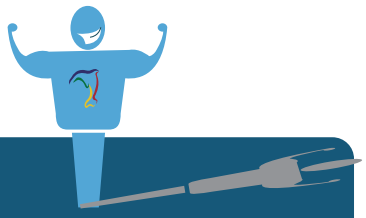
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

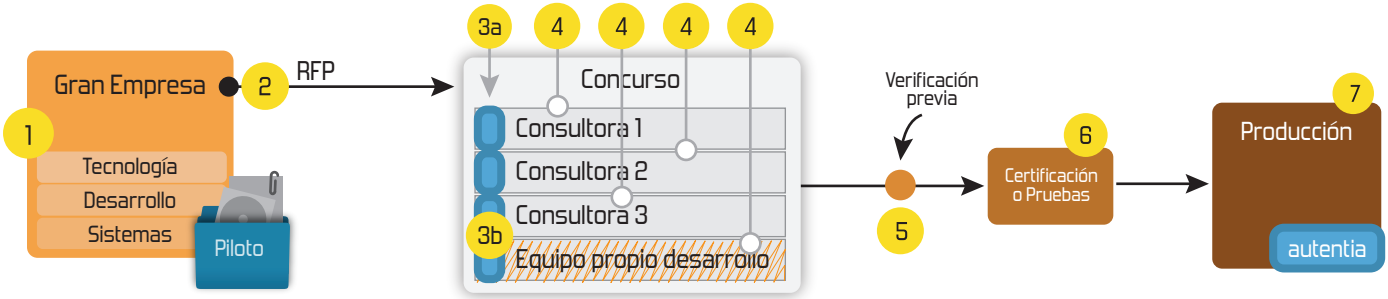
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

AdictosAlTrabajo
Temporada Completa de Terrakas
terrakas.com
autentia Soporte a desarrollo informático
Hosting patrocinado por enredados

Entra en Adictos a través de
E-mail
Contraseña
Entrar Deseo registrarme Olvidé mi contraseña

Inicio Quiénes somos Formación Comparador de salarios Nuestros libros Más

» Estás en: Inicio Tutoriales Desarrollando una aplicación de detección de iBeacons



Sara Subijana Gracia

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos de servicios soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2014-05-19 Tutorial visitado 2 veces Descargar en PDF

Desarrollando una aplicación de detección de iBeacons

Índice de contenidos

1. Introducción
2. Entorno
3. Desarrollando la aplicación
3.1. Creación de un nuevo proyecto con XCode
3.2. Código para monitorización de los iBeacons
3.3. Lanzando nuestra aplicación
4. Conclusiones
5. Referencias

1. Introducción

iBeacon es el nombre comercial que da Apple a un sistema basado en Bluetooth Low Energy (BLE, Bluetooth 4.0, Bluetooth LE, Bluetooth Smart) que extiende los servicios de localización de iOS (Core Location) y permite la detección de unos dispositivos conocidos como beacons cuando nos aproximamos a ellos.



Bajo este sistema **los beacons se limitan a emitir unas señales sobre Bluetooth 4.0 de forma continua**. Estas señales contienen información relativa al dispositivo, localización y nuestra proximidad. De este modo una aplicación perteneciente a un negocio podría ser lanzada cuando se detectara la proximidad a uno de sus establecimientos con un sistema de beacons establecido.

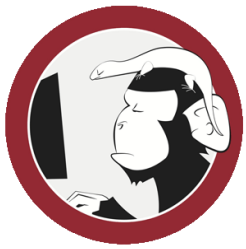
Aunque este sistema tiene un nombre inventado por Apple, no está limitado a dispositivos iOS u OS X. De hecho, siempre y cuando se disponga de una API para tratar la información de los beacons, los dispositivos compatibles con Bluetooth 4.0 y Android 4.3 o superior ya pueden utilizarlos.

iOS 5 ya tenía soporte para trabajar con Bluetooth 4.0 a través de Core Bluetooth pero no ha sido hasta iOS 7 que Apple ha puesto a disposición de los desarrolladores una API de alto nivel para trabajar directamente con el sistema iBeacon. El mecanismo consistió en extender la clase **Core Location** añadiendo una API nueva para manejar los iBeacons. En el siguiente apartado veremos qué métodos y clases deben utilizarse para la monitorización y acceso a la información desde una sencilla app.

2. Entorno

Catálogo de servicios Autentia





Síguenos a través de:

Últimas Noticias

» Concurso del Día de la Madre:

» Aprende gratis ReactiveCocoa

» Checklist de Scrum de Autentia

» Autentia estrena web y celebra el X Cycling Day

» Buscamos personal para Autentia y nuestros clientes (10-Marzo-2014)

Histórico de noticias

Últimos Tutoriales

» SCRATCH, aprende a programar desde pequeño

» Final Cut Pro X vs iMovie 10.1

» Implementar una función UDF de Apache Pig

» Primeros pasos con Bonita BPM Community 6.2.6

» Lanzamos nuevo site en Autentia.com

El tutorial está escrito usando el siguiente entorno:

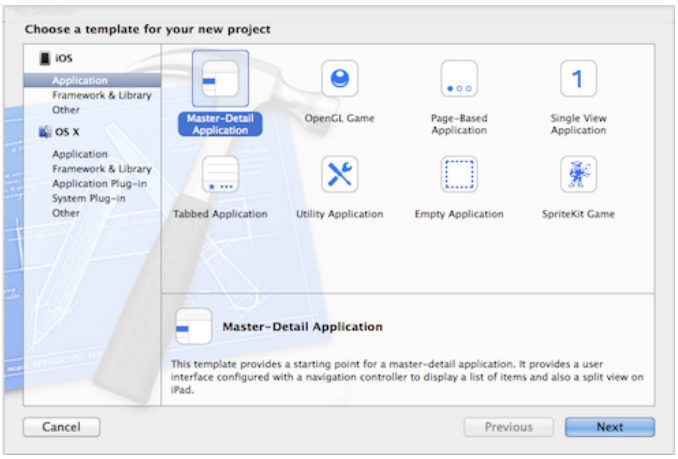
- Hardware: Portátil MacBook Pro 15 pulgadas (2.4 GHz Intel i7, 8GB 1333 Mhz DDR3, 500GB Flash Storage).
- Sistema Operativo: Mac OS X Lion 10.9.1
- XCode 5.1 (versión beta)
- iBeacons Proveedor Kontakt.io
- iPod Touch 5 generación (instalada versión 7.1 iOS)

3. Desarrollando la aplicación

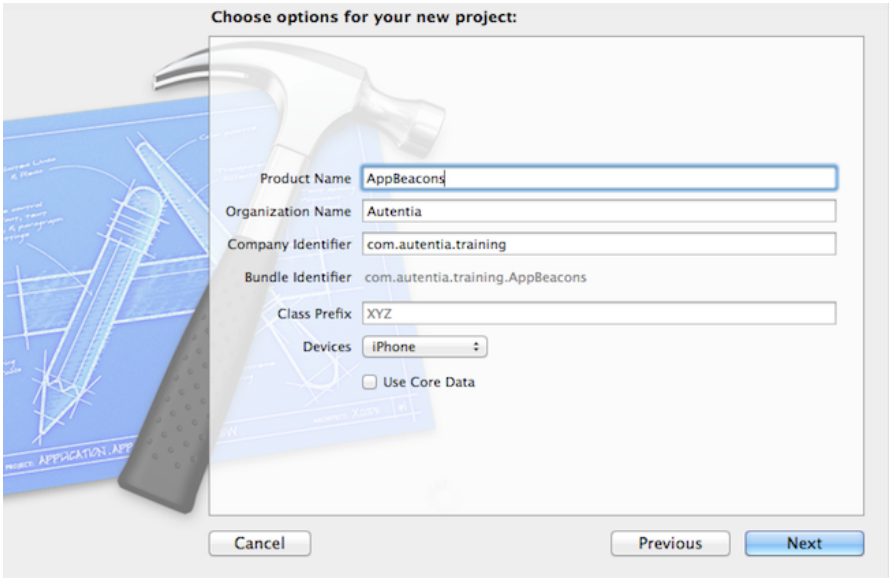
En este apartado construiremos una pequeña app que detecte iBeacons y muestre un listado de ellos. Se podrá acceder al detalle de cada uno para ver la información desglosada.

3.1. Creación de un nuevo proyecto con XCode

Creamos un proyecto desde *New -> Project -> Master-Detail Application.* Hemos elegido este tipo de aplicación por comodidad ya que implementa por defecto un table view. Este table view servirá para mostrar nuestro listado de iBeacons.



El nombre de nuestra aplicación será AppBeacons.



La arquitectura de nuestro proyecto se compone de los siguientes ficheros:

- *Main.storyboard*: Definiremos gráficamente nuestras pantallas y las asociaremos con los contorladores
- *AppDelegate*: Delegado prinicipal de la aplicación.
- *MasterViewController*: Controlador de la pantalla de inicio y que visualizará una tabla con los iBeacons detectados.
- *DetailViewController*: Controlador de la pantalla de detalle al seleccionar una celda de la tabla. En el detalle se podrá acceder a información relevante del iBeacon.

El proyecto por defecto se compone de una vista principal que muestra una tabla editable y una segunda vista que muestra el detalle de una celda al seleccionarla. Para nuestro tutorial esta aplicación nos vale, pero necesitamos "limpiar" el controlador principal de nuestra aplicación.

Abrimos el fichero MasterViewController.m y eliminamos el código de edición de filas, quedando un esqueleto como el que sigue:

```
1  @implementation MasterViewController
2
3
4  - (void)insertNewObject:(id)sender
5  {
6      if (!self.detectedBeacons) {
```

```
        self.detectedBeacons = [[NSMutableArray alloc] init];
    }
    [self.detectedBeacons insertObject:[NSDate date] atIndex:0];
    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:0 inSection:0];
    [self.tableView insertRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAn
}

#pragma mark - CLLocationManagerDelegate

- (void)locationManager:(CLLocationManager *)manager
      didDetermineState:(CLRegionState)state
              forRegion:(CLRegion *)region
{
    if (state == CLRegionStateInside) {
        NSLog(@"inside region");
        [self.locationManager startRangingBeaconsInRegion:(CLBeaconRegion *)region];
    } else {
        NSLog(@"not in region");
        [self.locationManager stopRangingBeaconsInRegion:(CLBeaconRegion *)region];
    }
}


- (void)locationManager:(CLLocationManager *)manager
        didRangeBeacons:(NSArray *)beacons
               inRegion:(CLBeaconRegion *)region
{
    self.detectedBeacons = [NSMutableArray new];
    for (CLBeacon *beacon in beacons) {
        NSLog(@"%@",beacon);
        [self.detectedBeacons addObject:beacon];
    }
    [self.tableView reloadData];
}


#pragma mark - Table View and data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return self.detectedBeacons.count;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPa
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell" forInde

    CLBeacon *object = self.detectedBeacons[indexPath.row];


    NSLog(@"Beacon %@ monitorized with Major %ld and Minor: %ld",
          [object.proximityUUID UUIDString],
          (long)object.major.integerValue,
          (long)object.minor.integerValue);

    NSString *beaconLabel = [NSString stringWithFormat:
                             @"Beacon monitorized with Major %ld and Minor: %ld ",
                             (long)object.major.integerValue, (long)object.minor.integerV
    cell.textLabel.text = beaconLabel;
    return cell;
}

- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPat
{
    // Return NO if you do not want the specified item to be editable.
    return YES;
}

- (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingSty
{
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        [self.detectedBeacons removeObjectAtIndex:indexPath.row];
        [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAn
    } else if (editingStyle == UITableViewCellEditingStyleInsert) {
        // Create a new instance of the appropriate class, insert it into the array, and
    }
}


#pragma mark - UIViewController

- (void)viewDidLoad
{
    [super viewDidLoad];


    NSUUID *proximityUUID = [[NSUUID alloc] initWithUUIDString:kIBeaconUUID];
    self.beaconRegion = [[CLBeaconRegion alloc] initWithProximityUUID:proximityUUID iden
    self.beaconRegion.notifyEntryStateOnDisplay = YES;

    self.locationManager = [CLLocationManager new];
    self.locationManager.delegate = self;

    if (![CLLocationManager isRangingAvailable]) {
        NSLog(@"Couldn't turn on ranging: Ranging is not available.");
        return;
    }

    if (self.locationManager.rangedRegions.count > 0) {
        NSLog(@"Didn't turn on ranging: Ranging already on.");
    }

    [self.locationManager startMonitoringForRegion:self.beaconRegion];

```

```
}

- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if ([[segue identifier] isEqualToString:@"showDetail"]) {
        NSIndexPath *indexPath = [self.tableView indexPathForSelectedRow];
        NSDate *object = self.detectedBeacons[indexPath.row];
        [[segue destinationViewController] setDetailItem:object];
    }
}


@end
```

El delegado AppDelegate viene con unos métodos implementados por defecto que no necesitamos. Únicamente dejaremos implementado *didFinishLaunchingWithOptions* para comprobar que en background la aplicación se refresca:

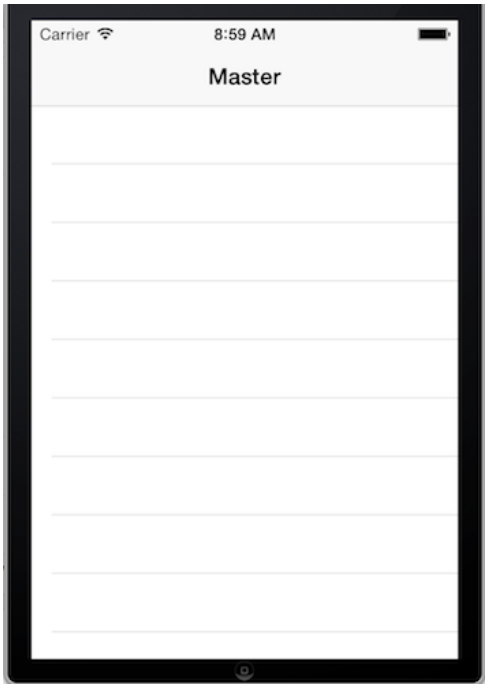
```
#import "AppDelegate.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictiona
{
    if ([[UIApplication sharedApplication] backgroundRefreshStatus] == UIBackgroundRefresh
        NSLog(@"Background updates are available for the app.");
    }else if([[UIApplication sharedApplication] backgroundRefreshStatus] == UIBackgroundRe
    {
        NSLog(@"The user explicitly disabled background behavior for this app or for the
    }else if([[UIApplication sharedApplication] backgroundRefreshStatus] == UIBackgroundRe
    {
        NSLog(@"Background updates are unavailable and the user cannot enable them again.
    }
    return YES;
}

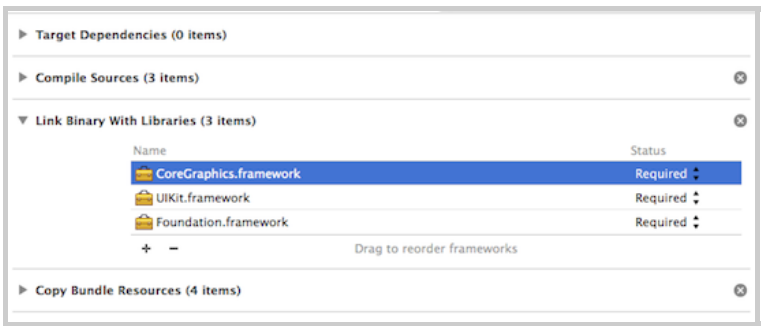
@end
```

Ya tenemos nuestra aplicación preparada. Hasta ahora debería mostrar una tabla vacía:

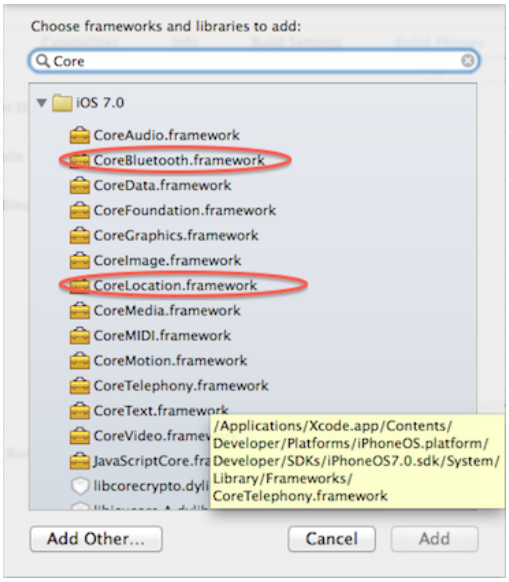


3.2. Código para monitorización de los iBeacons

Para la detección de iBeacons se hace uso de la librería CoreLocation y CoreBluetooth. Accedemos a la sección *Build Phases* de nuestro target.



Añadimos los enlaces a las librerías.



Necesitamos definir unas constantes y propiedades privadas en nuestro controlador. Estas propiedades contendrán toda la información de localización y array auxiliar donde almacenaremos los beacons detectados para asociarlo al table view.

```
#import "MasterViewController.h"
#import "DetailViewController.h"

#pragma mark - Constants iBeacons info
static NSString * const kIBeaconUUID = @"F7826DA6-4FA2-4E98-8024-BC5B71E0893E";
static NSString * const kRegionIdentifier = @"Kontakt.io";


@interface MasterViewController ()
    @property (nonatomic, strong) NSMutableArray *detectedBeacons;
    @property (nonatomic, strong) CLLocationManager *locationManager;
    @property (nonatomic, strong) CLBeaconRegion *beaconRegion;

@end
```

Notar que para acceder al array *detectedBeacons* se accede previamente a la instancia a través de self. Esta notación es una buena práctica ya que aporta legibilidad.

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return self.detectedBeacons.count;
}
```

Ahora debemos indicar que nuestro controlador implementa el protocolo para la detección de beacons. Abrimos *MasterViewController.h* e incluimos lo siguiente:

```
#import <corebluetooth corebluetooth.h="">
#import <corelocation corelocation.h="">

@interface MasterViewController : UITableViewController <cllocationmanagerdelegate>

@end
</cllocationmanagerdelegate></corelocation></corebluetooth>
```

Para comenzar la monitorización de iBeacons hay que definir la región e inicializar el manager de localización. Actualizamos el evento del view controller viewDidLoad con el siguiente código:

```
- (void)viewDidLoad
{
    [super viewDidLoad];


    NSUUID *proximityUUID = [[NSUUID alloc] initWithUUIDString:kIBeaconUUID];
    self.beaconRegion = [[CLBeaconRegion alloc] initWithProximityUUID:proximityUUID ident
    self.beaconRegion.notifyEntryStateOnDisplay = YES;

    self.locationManager = [CLLocationManager new];
    self.locationManager.delegate = self;

    if (![CLLocationManager isRangingAvailable]) {
        NSLog(@"Couldn't turn on ranging: Ranging is not available.");
        return;
    }

    if (self.locationManager.rangedRegions.count > 0) {
        NSLog(@"Didn't turn on ranging: Ranging already on.");
    }

    [self.locationManager startMonitoringForRegion:self.beaconRegion];

}
```

Hay que tener en cuenta que al definir la región de nuestros iBeacons **es necesario indicar el ProximityUUID del proveedor y un identificador para la región**. Para los beacons de kontakt el ProximityUUID es *F7826DA6-4FA2-4E98-8024-BC5B71E0893E*.

El siguiente paso es implementar los métodos del protocolo que detectan cuándo está el dispositivo dentro de una regi&oacite;n para monitorizar.

El método *didDetermineState* es el encargado de detectar si el dispositivo se encuentra dentro de una región definida y en ese caso comenzar a monitorizar los beacons:

```
- (void)locationManager:(CLLocationManager *)manager
      didDetermineState:(CLRegionState)state
              forRegion:(CLRegion *)region
{
    if (state == CLRegionStateInside) {
        NSLog(@"inside region");
        [self.locationManager startRangingBeaconsInRegion:(CLBeaconRegion *)region];
    } else {
        NSLog(@"not in region");
        [self.locationManager stopRangingBeaconsInRegion:(CLBeaconRegion *)region];
    }
}
```

El método *didRangeBeacons* es el que nos proporciona los beacons detectados en nuetra región. Aquí será donde refresquemos la tabla con los nuevos beacons detectados:

```
- (void)locationManager:(CLLocationManager *)manager
        didRangeBeacons:(NSArray *)beacons
               inRegion:(CLBeaconRegion *)region
{
    self.detectedBeacons = [NSMutableArray new];
    for (CLBeacon *beacon in beacons) {
        NSLog(@"%@",beacon);
        [self.detectedBeacons addObject:beacon];
    }
    [self.tableView reloadData];
}
```

Terminamos modificando el método del datasource que muestra el contenido en la tabla para que nos muestre el major y minor de los beacons detectados:

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPa
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell" forIndex

    CLBeacon *object = self.detectedBeacons[indexPath.row];


    NSLog(@"Beacon %@ monitorized with Major %ld and Minor: %ld",
          [object.proximityUUID UUIDString],
          (long)object.major.integerValue,
          (long)object.minor.integerValue);

    NSString *beaconLabel = [NSString stringWithFormat:
                             @"Beacon monitorized with Major %ld and Minor: %ld ",
                             (long)object.major.integerValue, (long)object.minor.integerV
    cell.textLabel.text = beaconLabel;
    return cell;
}
```

Podemos mostrar más en detalle la información del iBeacon seleccionado. Para ello, adaptamos el controlador de detalle, DetailViewController.

DetailViewController.h

```
#import <uikit uikit.h="">
#import <corelocation corelocation.h="">

@interface DetailViewController : UIViewController

@property (strong, nonatomic) id detailItem;

@property (weak, nonatomic) IBOutlet UILabel *detailProxUUIDLabel;
@property (weak, nonatomic) IBOutlet UILabel *detailMajorLabel;
@property (weak, nonatomic) IBOutlet UILabel *detailMinorLabel;
@property (weak, nonatomic) IBOutlet UILabel *detailProximityLabel;
@end
</corelocation></uikit>
```

DetailViewController.m

```
#import "DetailViewController.h"

...

- (void)configureView
{
    // Update the user interface for the detail item.

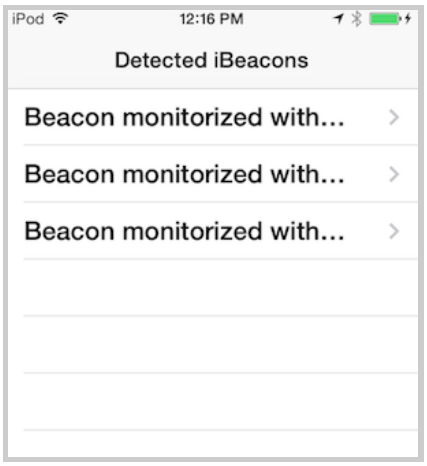
    if (self.detailItem) {
        CLBeacon *detailBeacon = (CLBeacon *)self.detailItem;
        NSString *beaconProxUUIDLabel = [NSString stringWithFormat:@"ProximityUUID: %@",
        self.detailProxUUIDLabel.text = beaconProxUUIDLabel;
        NSString *beaconMajorLabel = [NSString stringWithFormat:@"Major: %ld", (long)deta
        self.detailMajorLabel.text = beaconMajorLabel;
        NSString *beaconMinorLabel = [NSString stringWithFormat:@"Minor: %ld", (long)deta
        self.detailMinorLabel.text = beaconMinorLabel;
        NSString *beaconProximityLabel = nil;
        switch (detailBeacon.proximity) {
            case CLProximityImmediate:
                beaconProximityLabel = @"Proximity: inmediate (0 - 20 cm)";
                break;
            case CLProximityNear:
                beaconProximityLabel = @"Proximity: near (20cm - 2m)";
                break;
            case CLProximityFar:
                beaconProximityLabel = @"Proximity: far (2m - 70m)";
                break;
            case CLProximityUnknown:
            default:
                beaconProximityLabel = @"Proximity: unknown";
                break;
        }
        self.detailProximityLabel.text = beaconProximityLabel;
    }
```

```
35  }
36
37  ...
```

3.3. Lanzando nuestra aplicación

El resultado será una app que en su pantalla principal muestra los iBeacons detectados. Recordar que la aplicación necesita que el dispositivo tenga el bluetooth activado.

Listado de iBeacons detectados:



Detalle de iBeacon:



4. Conclusiones

iOS ofrece a través de Core Location una API sencilla para poder detectar estos dispositivos desde sencillas apps. Los iBeacons todaví necesitan ser explotados, pero son potencialmente mucho más precisos que detección de ubicaciones por GPS.

El dispositivo como tal no tiene mucha complejidad y se limita a enviar continuamente una información. La complejidad del desarrollo recae en la aplicación que tenga que explotar los datos enviados por el iBeacon. Por ejemplo, para saber la ubicación geográfica concreta, deberíamos desarrollar una app que a partir de los datos del iBeacon detectado recuperara esta información a través de la red. No obstante, todav´a hay mucha diferencia entre proveedores y encontrar la información necesaria y una API potente no es algo trivial.

5. Referencias

Este pequeño ejercicio lo he realizado apoyándome en varias fuentes:

- http://maniacdev.com/2013/10/example-an-app-using-the-new-ios-7-ibeacon-api
- http://www.punteroavoid.com/blog/2014/02/18/ibeacon-101/
- http://www.cocoaosx.com/2014/01/30/ibeacons-ios7/

Puedes descargarte el código del tutorial desde mi repositorio de github pinchando aquí.

Un saludo.

Sara

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» Registrate y accede a esta y otras ventajas **«**

SOME RIGHTS RESERVED Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

IMPULSA Impulsores Comunidad ¿Ayuda?

---- sin clicks

0 personas han traído clicks a esta página

powered by karmacracy

Copyright 2003-2014 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto

XHTML 1.0 CSS XML RSS XML ATOM