Persistencia de datos en iOS: Ficheros y SQLite

Índice

1 Introducción....2
2 Ficheros plist (Property Lists)....2
2.1 Leyendo datos desde ficheros plist....2
2.2 Escribiendo datos en ficheros plist....5
3 SQLite....8
3.1 Herramientas disponibles....8
3.2 Lectura de datos....9
3.3 Mostrando los datos en una tabla....14

1. Introducción

La persistencia de datos en iOS se produce gracias a una pequeña memoria flash que tiene nuestro dispositivo, el cual es equivalente a un disco duro de tamaño limitado. Los datos que ahí se almacenen se conservarán aunque el dispositivo se apague. Por motivos de seguridad, el SO de iOS no permite que las aplicaciones accedan directamente a la memoria interna, pero a cambio existe una API completa de acceso a la porción de memoria que le corresponde a la app que desarrollemos.

En una aplicación iOS encontraremos básicamente 4 modos de almacenar nuestros datos de forma persistente, estos son:

- Ficheros de propiedades (*plist*)
- Bases de datos embebidas *SQLite*
- Datos de usuario (*User Defaults*)
- Core Data

En esta primera sesión veremos cómo usar los ficheros de texto (*Property Lists*) y el almacenamiento en base de datos de tipo *SQLite*.

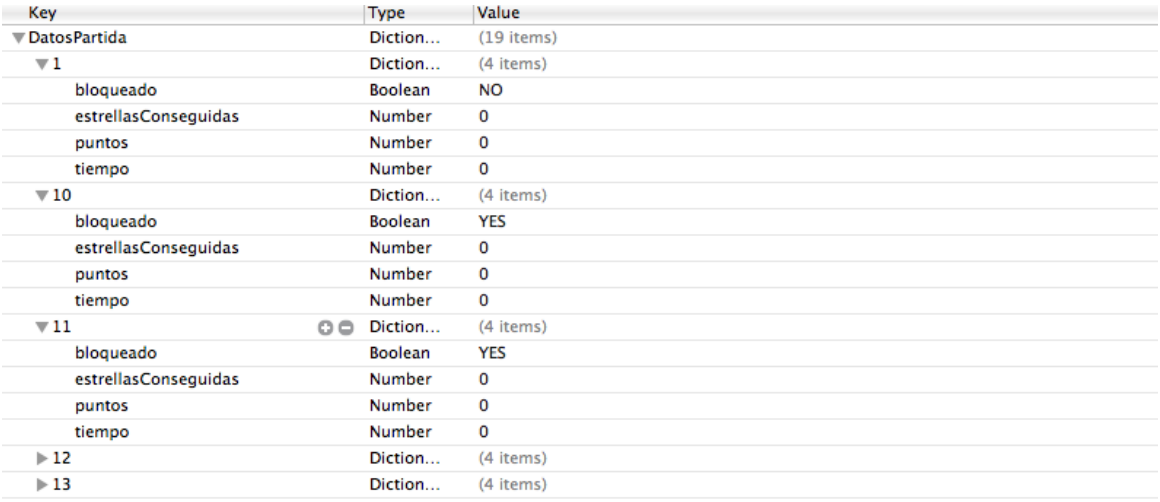
2. Ficheros plist (Property Lists)

El almacenamiento de datos en ficheros de propiedades es uno de los más usados en el desarrollo de las aplicaciones para **iOS**. Se usa principalmente para guardar datos de configuración de la aplicación y acceder a ellos de forma sencilla. Si estamos desarrollando un juego podemos usarlos para la definición de los niveles, almacenar las puntuaciones del jugador, etc.

2.1. Leyendo datos desde ficheros plist

Los ficheros plist (`Property Lists`) tienen formato **XML** y son *"amigables"* con xCode. Para entender el funcionamiento de este tipo de ficheros vamos a realizar un ejemplo en el que primero crearemos un fichero plist, lo completaremos con datos de ejemplo y después lo leeremos para mostrarlo por pantalla.

Un ejemplo de un fichero plist podría ser el siguiente:



Key	Type	Value
▼DatosPartida	Diction…	(19 items)
▼1	Diction…	(4 items)
bloqueado	Boolean	NO
estrellasConseguidas	Number	0
puntos	Number	0
tiempo	Number	0
▼10	Diction…	(4 items)
bloqueado	Boolean	YES
estrellasConseguidas	Number	0
puntos	Number	0
tiempo	Number	0
▼11	Diction…	(4 items)
bloqueado	Boolean	YES
estrellasConseguidas	Number	0
puntos	Number	0
tiempo	Number	0
▶12	Diction…	(4 items)
▶13	Diction…	(4 items)

Fichero plist

Antes de nada creamos un proyecto nuevo en xCode de tipo `Window-based Application` al que llamaremos `sesion03-ejemplo1`. Para crear un fichero plist debemos hacer click derecho sobre el directorio `Supporting Files` de nuestro proyecto y seleccionamos `iOS > Resource > Property List`. Lo guardaremos con el nombre `configUsuario`. Seguidamente ya podemos acceder al fichero recién creado haciendo click sobre el, en este momento xCode nos mostrará un pequeño editor totalmente vacio en el que iremos rellenando con datos. Podemos verlo también en formato *XML* si hacemos click derecho sobre el y seleccionamos `Open As > Preview`.

Para empezar a rellenar el fichero que acabamos de crear, hacemos click derecho dentro de su contenido (ahora vacio) y seleccionamos `"Add row"`. Entonces nos aparecerá una fila vacía. Dentro del campo `"Key"` escribimos la clave, un nombre descriptivo que deberemos utilizar más adelante para acceder a los datos. En nuestro caso vamos a escribir `"config"` como clave. Dentro de la columna `"Type"` seleccionamos `"Dictionary"`. Veremos que aparece una flecha en el lado izquierdo que indica que pueden haber subniveles dentro de nuestro diccionario.

Ahora añadimos un nuevo item que cuelgue del diccionario que acabamos de crear, para ello hacemos click sobre la flecha de la izquierda, esta se girará hacia abajo, entonces hacemos click ahora sobre el símbolo `+ (más)`. Nos aparecerá una nueva línea que cuelga del diccionario.

En esta nueva línea, dentro del campo `"key"` escribimos `"nombre"`, el campo `"type"` lo dejamos como de tipo `"String"` ya que va a ser una cadena de texto y en la columna de `"Value"` escribimos nuestro nombre, por ejemplo `"Javi"`.

Ahora vamos a añadir otro campo nuevo en nuestra configuración, para ello hacemos click sobre el símbolo `+` y nos aparecerá una nueva línea justo al mismo nivel que la anterior y colgando del diccionario. Dentro del campo de `"Key"` escribimos `"ciudad"` y en `"Value"` escribimos `"Alicante"`. El campo de `"Type"` lo dejamos como `"String"`,

Ahora dentro de nuestro fichero *plist* vamos a indicar los dispositivos que dispone el usuario, para ello creamos una nueva linea dentro del diccionario y le pondremos como clave `"dispositivos"`. Seleccionamos el tipo `"Array"` y seguimos los mismos pasos que al crear el diccionario, para ello hacemos click sobre la flecha de la izquierda y después sobre el símbolo `+ (más)`. Ahora nos aparecerá una nueva fila con clave `Item 0`. Seleccionamos el tipo `"String"` y en el campo `"Value"`: `"iPhone"`.

Ahora repetimos el paso 3 veces, indicando como `values`: `"iPad"`, `"Android"` y `"Blackberry"`.

Una vez seguidos todos estos pasos tendremos nuestro fichero de propiedades listo. Debe de quedar como se muestra a continuación:

Key	Type	Value
▼ config	Diction…	(3 items)
▼ dispositivos	Array	(4 items)
Item 0	String	Blackberry
Item 1	String	Android
Item 2	String	iPhone
Item 3	String	iPad
ciudad	String	Alicante
nombre	String	Javi

Fichero plist terminado

Este fichero se almacenará dentro del directorio de *bundle* cuando compilemos la aplicación. En el caso de que queramos escribir en el, debermos almacenar el fichero dentro del directorio de *"Documents"* del binario.

Ahora desde nuestro código vamos a acceder a él y a mostrarlo por pantalla, para ello escribimos el siguiente código dentro del método `didFinishLaunchingWithOptions` de la clase delegada:

```
        // Cargamos el fichero PLIST
            NSString *mainBundlePath = [[NSBundle mainBundle] bundlePath];
            NSString *plistPath = [mainBundlePath
            stringByAppendingPathComponent:@"configUsuario.plist"];
            NSDictionary *diccionario = [[NSDictionary alloc]
            initWithContentsOfFile:plistPath];

            // Cargamos el Diccionario inicial que esta en el raiz del
fichero
            NSDictionary *dicConfig = [diccionario
objectForKey:@"config"];

            // Cargamos los campos que estan dentro del diccionario del
raiz
            NSString *nombre = [dicConfig objectForKey:@"nombre"];
```

```
            NSString *ciudad = [dicConfig objectForKey:@"ciudad"];
            NSArray *dispositivos = [dicConfig
objectForKey:@"dispositivos"];

            // Mostramos por consola los datos obtenidos
            NSLog(@"Nombre: %@", nombre);
            NSLog(@"Ciudad: %@", ciudad);

            for (NSString *dispositivo in dispositivos){
                NSLog(@"Dispositivo: %@", dispositivo);
            }


            [self.window makeKeyAndVisible];
            return YES;
```

Si todo ha ido correctamente, deberemos obtener la siguiente salida por la consola:

```
2011-09-01 20:10:25.981 sesion03-ejemplo0[2311:207] Nombre: Javi
2011-09-01 20:10:25.983 sesion03-ejemplo0[2311:207] Ciudad: Alicante
2011-09-01 20:10:25.984 sesion03-ejemplo0[2311:207] Dispositivo: Blackberry
2011-09-01 20:10:25.985 sesion03-ejemplo0[2311:207] Dispositivo: Android
2011-09-01 20:10:25.986 sesion03-ejemplo0[2311:207] Dispositivo: iPhone
2011-09-01 20:10:25.986 sesion03-ejemplo0[2311:207] Dispositivo: iPad
```

Lista de dispositivos

2.2. Escribiendo datos en ficheros plist

Partiendo del ejemplo anterior vamos a escribir ahora en el fichero, para ello primero debemos comprobar que el fichero plist se encuentre dentro del directorio de `Documents` del dispositivo, en el caso de que no lo esté (por ejemplo, si es la primera vez que arranca la aplicación), debemos copiarlo a ese directorio. Una vez que tenemos el fichero dentro del directorio de documentos ya podemos escribir en el.

Para realizar todo el proceso de comprobación de la existencia del fichero dentro del directorio de documentos, copiarlo en ese directorio y leer los datos debemos de sustituir el código que hay dentro del método `didFinishLaunchingWithOptions` de la clase delegada por este otro:

Atención

Para que funcione la escritura en un fichero plist, este debe de estar dentro del directorio de `Documents` de nuestro dispositivo iOS, no en el boundle.

```
                NSDictionary *diccionarioRaiz;

            // Ruta del directorio de documentos de nuestro dispositivo
            NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
            NSUserDomainMask, YES);

            if ([paths count] > 0)
```

```
            {
                NSString *documentsDirectory = [paths objectAtIndex:0];
                NSString *documentsFilename = [documentsDirectory
                stringByAppendingPathComponent:@"configUsuario.plist"];

                // Primero comprobamos si existe el fichero en el
directorio de documentos
                BOOL fileExists = [[NSFileManager defaultManager]
                fileExistsAtPath:documentsFilename];
                if (fileExists)
                {
                    NSLog(@"Fichero encontrado en directorio de documentos
OK: %@!",
                    documentsFilename);

                    // Si existe ya, cargamos los datos desde aqui
directamente
                    diccionarioRaiz = [[NSDictionary alloc]
                    initWithContentsOfFile:documentsFilename];

                }
                else
                {
                    NSLog(@"No se encuentra el fichero configUsuario.plist
en
                    el directorio de documentos->Lo creamos.");

                    // Si no existe, primero cargamos el fichero PLIST
desde el Boundle
                    NSString *mainBundlePath = [[NSBundle mainBundle]
bundlePath];
                    NSString *plistPath = [mainBundlePath
stringByAppendingPathComponent:@"configUsuario.plist"];
                    diccionarioRaiz = [[NSDictionary alloc]
                    initWithContentsOfFile:plistPath];

                    // Y despues escribimos los datos cargados (el
diccionario completo)
                    // a un fichero nuevo de la carpeta de documentos
                    [diccionarioRaiz writeToFile:documentsFilename
atomically:YES];

                    NSLog(@"plist de configUsuario creado!");


                }


                // Cargamos el Diccionario inicial que esta en el raiz del
fichero
                NSDictionary *dicConfig = [diccionarioRaiz
objectForKey:@"config"];

                // Cargamos los campos que estan dentro del diccionario
del raiz
                NSString *nombre = [dicConfig objectForKey:@"nombre"];
                NSString *ciudad = [dicConfig objectForKey:@"ciudad"];
                NSArray *dispositivos = [dicConfig
objectForKey:@"dispositivos"];

                // Mostramos por consola los datos obtenidos
                NSLog(@"Nombre: %@", nombre);
                NSLog(@"Ciudad: %@", ciudad);

                for (NSString *dispositivo in dispositivos){
```

```
                    NSLog(@"Dispositivo: %@", dispositivo);
                }

                // Por ultimo liberamos la memoria del diccionario raiz
                [diccionarioRaiz release];
            }



            [self.window makeKeyAndVisible];
            return YES;
```

Para comprobar el funcionamiento arrancamos la aplicación y veremos que la primera vez se creará el fichero dentro de la carpeta de `Documents` de nuestro dispositivo y en las siguientes ya no se creará y simplemente lo leerá desde la carpeta de `Documents`.

Ahora que ya tenemos la gestión de ficheros básica completada vamos a escribir sobre el algún dato. Para ello simplemente escribimos el siguiente fragmento de código justo antes de la línea `[self.window makeKeyAndVisible]`:

```
        //***********************************************
            // Cambiamos el nombre y lo guardamos en el fichero
            // Ruta del directorio de documentos de nuestro dispositivo

            if ([paths count] > 0)
            {
                    // Cargamos el fichero desde el directorio de
documentos del dispositivo
                NSString *documentsDirectory = [paths objectAtIndex:0];
                NSString *documentsFilename = [documentsDirectory
                stringByAppendingPathComponent:@"configUsuario.plist"];

                diccionarioRaiz = [[NSDictionary alloc]
                initWithContentsOfFile:documentsFilename];

                NSDictionary *dicConfig = [diccionarioRaiz
objectForKey:@"config"];

                // Cambiamos el valor del nombre en el diccionario
                [dicConfig setValue:@"Otro nombre" forKey:@"nombre"];

                // Escribimos todo el diccionario de nuevo al fichero del
                // directorio de documentos
                [diccionarioRaiz writeToFile:documentsFilename
atomically:YES];

                        // Por ultimo liberamos el diccionario de la
memoria
                [diccionarioRaiz release];
            }
```

Lo que hemos hecho en el código anterior es cargar todo el diccionario de nuevo desde el directorio de documentos, modificar los datos que queremos cambiar o incluso añadir y después guardar el diccionario en el fichero.

Volvemos a arrancar la aplicación y veremos que ha cambiado el campo de nombre.

Nota

La lectura y escritura de datos en un fichero plist se debe de hacer de una sóla vez, nunca por partes. Siempre se debe de cargar o escribir un diccionario `NSDictionary` como elemento principal del fichero.

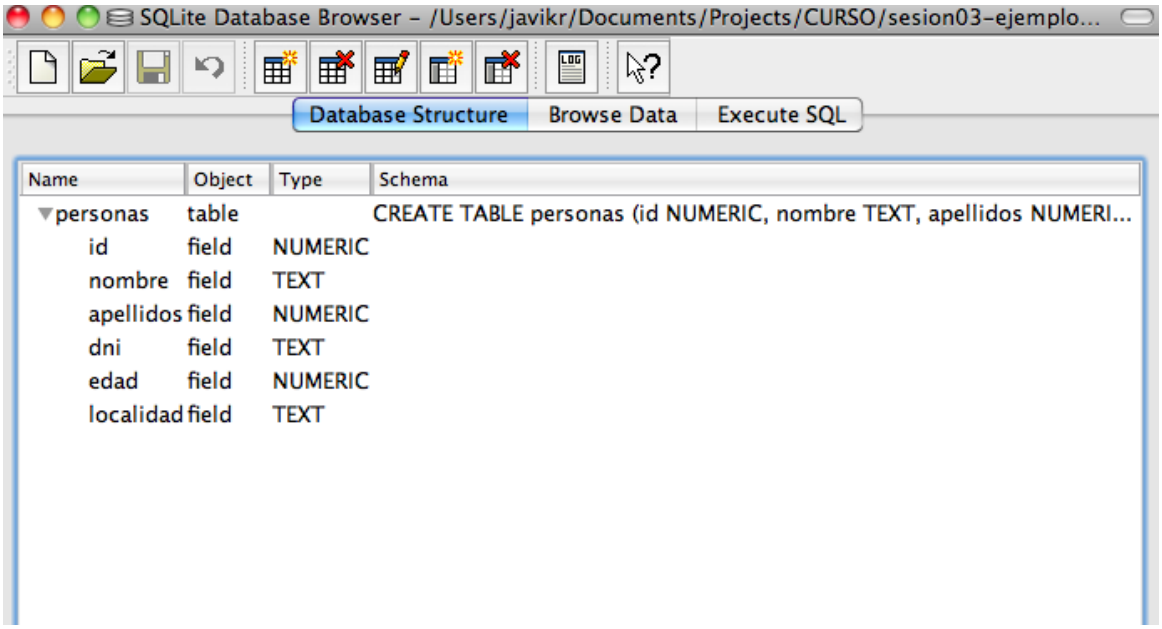
3. SQLite

SQLite es una base de datos relacional ligera con la que se puede trabajar de forma sencilla mediante sentencias **SQL**. La base de datos se almacenará en un fichero dentro de nuestra aplicación. Este método de persistencia puede ser el adecuado cuando los datos a almacenar estén en un formato de tablas, filas y columnas y se necesite que los datos sean accesibles de forma rápida.

3.1. Herramientas disponibles

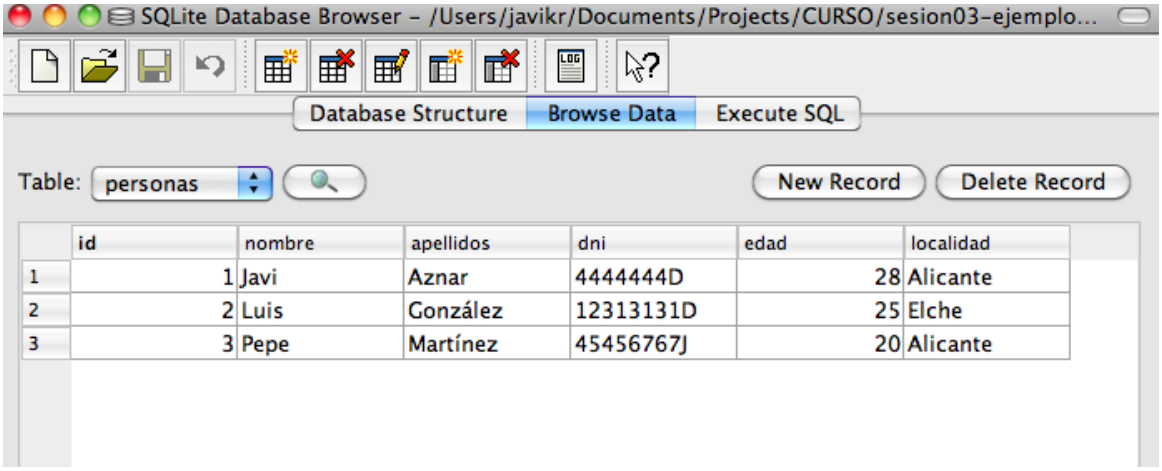
Para explicar el funcionamiento de las librerias de *SQLite* de *Cocoa Touch* vamos a seguir un ejemplo completo. Comenzamos creando la base de datos, para ello podemos utilizar uno de los frontends que facilitan la gestión del *SQLite*: uno bastante sencillo y recomendado es el SQLite Database Browser y otro es fmdb.

Nosotros utilizaremos el primero (*SQLite Database Browser*). Abrimos la aplicación y hacemos click sobre el botón de crear nueva base de datos que llamaremos: `personasDB.sqlite`. Seguidamente creamos las tablas, para simplificar en nuestro ejemplo crearemos una sóla tabla llamada `"personas"`. Esta tabla tendrá las siguientes columnas: `id (NUMERIC), nombre (TEXT), apellidos (TEXT), dni (TEXT), edad (NUMERIC)` y `localidad (TEXT)`.



Estructura de la Base de Datos

Una vez creada la tabla pasamos a insertar datos dentro de ella, para ello abrimos la pestaña de `Browse Data` y hacemos click en `New Record`. Cuando tengamos al menos 3 registros metidos dentro de la tabla ya podemos guardar la base de datos. Hacemos click sobre *guardar*, le ponemos el nombre que queramos y elegimos cualquier directorio del MAC.

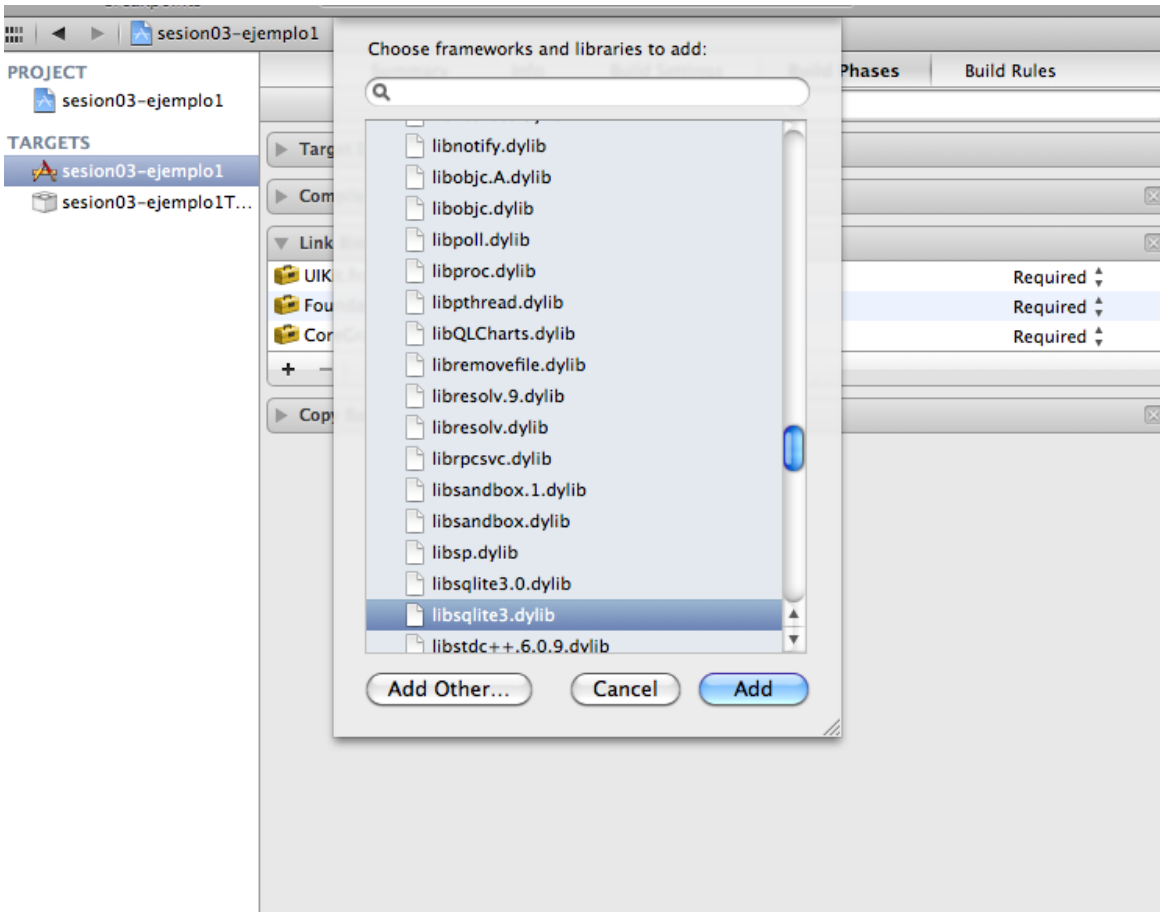


Datos de la Base de Datos

3.2. Lectura de datos

Una vez que tenemos la base de datos *SQLite* creada ya podemos empezar a escribir el código para leer y mostrar los datos dentro de nuestra aplicación:

Primero empezamos creando un proyecto nuevo en xCode usando la plantilla basada en vistas `View-based application` y lo llamaremos `sesion03-ejemplo2`. Ahora añadimos el framework de sqlite a nuestro proyecto, para ello hacemos click sobre el raiz del proyecto y seleccionamos la pestaña `Build Phases`. Desplegamos el listado de `Link Binary With Libraries` y hacemos click sobre el botón `(+)`. El el listado seleccionamos el framework `libsqlite3.dylib` y hacemos click en `Add`.



Añadiendo el framework libsqlite3.dylib

Ahora añadimos a nuestro proyecto la base de datos que hemos creado anteriormente, para ello arrastramos el fichero `personasDB.sqlite` a nuestro directorio `Supporting Files`. Asegurate de seleccionar la opción de `"Copy items to destination group's folder (if needed)"` para que se copie el fichero dentro de la carpeta del proyecto.

Para acelerar el proceso de lectura de la base de datos y ahorrar memoria vamos a cargar primero todos los datos en objetos, para ello vamos a crear una clase que se encargue de almacenar los datos. Hacemos click derecho sobre el raiz del proyecto *"New file > Objective-C class"*, seleccionamos `"NSObject"` como clase principal y le damos a *"Next"*. Escribimos como nombre de la clase: `"Persona"` y hacemos click en *"save"*.

Persona.h

```
//imports iniciales

@interface Persona : NSObject {
    int _uId;
    NSString *_nombre;
    NSString *_apellidos;
    NSInteger _edad;
    NSString *_localidad;

}

@property (nonatomic, assign) int uId;
@property (nonatomic, copy) NSString *nombre;
@property (nonatomic, copy) NSString *apellidos;
@property (nonatomic) NSInteger edad;
@property (nonatomic, copy) NSString *localidad;

- (id)initWithUid:(int)uId nombre:(NSString *)nombre
apellidos:(NSString *)apellidos edad:(NSInteger)edad
localidad:(NSString *)localidad;

@end
```

Persona.m

```
#import "Persona.h"


@implementation Persona

@synthesize uId = _uId;
@synthesize nombre = _nombre;
@synthesize apellidos = _apellidos;
@synthesize edad = _edad;
@synthesize localidad = _localidad;

- (id)initWithUid:(int)uId nombre:(NSString *)nombre
apellidos:(NSString *)apellidos
edad:(NSInteger)edad localidad:(NSString *)localidad {

    if ((self = [super init])) {
        self.uId = uId;
        self.nombre = nombre;
        self.apellidos = apellidos;
        self.edad = edad;
        self.localidad = localidad;

    }
    return self;
}

- (void) dealloc {
    self.nombre = nil;
    self.apellidos = nil;
    self.localidad = nil;
    [super dealloc];
}
```

```
@end
```

Una vez creada la clase `"Persona"` vamos a crear la clase encargada de manejar la base de datos *sqlite3*. Otra opción sería cargar todos los datos directamente desde la clase delegada al arrancar la aplicación pero no es recomendable ya que si en algún momento tenemos que cambiar de motor de base de datos lo tendremos más complicado.

Creamos entonces una clase dentro de nuestro proyecto llamada `"BDPersistencia"` que herede de `NSObject` y escribimos el siguiente código:

`BDPersistencia.h`

```
//imports iniciales

@interface BDPersistencia : NSObject {
    sqlite3 *_database;
}

+ (BDPersistencia*)database;
- (NSArray *)arrayPersonas;


@end
```

`BDPersistencia.m`

```
#import "BDPersistencia.h"
#import "Persona.h"

@implementation BDPersistencia

static BDPersistencia *_database;


+ (BDPersistencia*)database {
    if (_database == nil) {
        _database = [[BDPersistencia alloc] init];
    }
    return _database;
}


- (id)init {
    if ((self = [super init])) {
        NSString *sqLiteDb = [[NSBundle mainBundle]
        pathForResource:@"personasDB"
ofType:@"sqlite"];

        if (sqlite3_open([sqLiteDb UTF8String],
&_database) != SQLITE_OK)
        {
            NSLog(@"Fallo al abrir la BD!");
        }
    }
    return self;
}
```

```
                - (void)dealloc {
                    sqlite3_close(_database);
                    [super dealloc];
                }

                - (NSArray *)arrayPersonas {

                    NSMutableArray *arraySalida = [[[NSMutableArray alloc]
init]
                    autorelease];

                    NSString *query = @"SELECT id, nombre, apellidos,
localidad, edad
                    FROM personas ORDER BY apellidos DESC";

                    sqlite3_stmt *statement;
                    if (sqlite3_prepare_v2(_database, [query UTF8String],
-1,
                    &statement, nil)
                        == SQLITE_OK) {
                        while (sqlite3_step(statement) == SQLITE_ROW) {
                            int uniqueId = sqlite3_column_int(statement,
0);
                            char *nombreChar = (char *)
sqlite3_column_text(statement, 1);
                            char *apellidosChar = (char *)
sqlite3_column_text(statement, 2);
                            char *localidadChar = (char *)
sqlite3_column_text(statement, 3);
                            int edad = sqlite3_column_int(statement, 4);

                            NSString *nombre = [[NSString alloc]
                            initWithUTF8String:nombreChar];
                            NSString *apellidos = [[NSString alloc]
                            initWithUTF8String:apellidosChar];
                            NSString *localidad = [[NSString alloc]
                            initWithUTF8String:localidadChar];
                            Persona *persona = [[Persona alloc]
                            initWithUid:uniqueId nombre:nombre
                            apellidos:apellidos edad:edad
localidad:localidad];

                            [arraySalida addObject:persona];
                            [nombre release];
                            [apellidos release];
                            [localidad release];
                            [persona release];
                        }
                        sqlite3_finalize(statement);
                    }
                    return arraySalida;

                }
                @end
```

Con esto ya podemos leer de nuestra base de datos las personas. Para ver que funciona todo a la perfección escribimos el siguiente código dentro de la clase delegada en el método `applicationDidFinishLaunching`. Antes debemos incluir en la parte superior las clases creadas:

```
                #import "BDPersistencia.h"
```

```
                #import "Persona.h"
```

```
                NSArray *personas = [BDPersistencia
database].arrayPersonas;
                for (Persona *persona in personas) {
                    NSLog(@"%d: %@, %@, %@, %d", persona.uId,
persona.nombre,
                    persona.apellidos, persona.localidad, persona.edad);
                }
```

Si ejecutamos el proyecto, obtendremos la siguiente salida por consola:

```
2011-09-02 17:57:11.520 sesion03-ejemplo1[4567:207] Carga OK BD
2011-09-02 17:57:11.523 sesion03-ejemplo1[4567:207] DATOS:
2011-09-02 17:57:11.524 sesion03-ejemplo1[4567:207] 3: Pepe, Martínez, Alicante, 20
2011-09-02 17:57:11.524 sesion03-ejemplo1[4567:207] 2: Luis, González, Elche, 25
2011-09-02 17:57:11.525 sesion03-ejemplo1[4567:207] 1: Javi, Aznar, Alicante, 28
2011-09-02 17:57:11.531 sesion03-ejemplo1[4567:207] total: 3
```

Listado de personas

3.3. Mostrando los datos en una tabla

Una vez que tenemos el código para obtener todas las filas de la tabla de personas de la base de datos queda lo más fácil, mostrarlas dentro de una vista de la aplicación. En este caso utilizaremos la vista de tabla `"UITableView"`.

Comenzamos creando una nueva vista en nuestro proyecto, para ello hacemos click derecho sobre el raiz y seleccionamos: *"New File > UIVIewController Subclass"*. Asegurate de marcar `"Subclass of UITableViewController"` y la opcón de `"With XIB for user interface"`. Hacemos click en *"Next"* y guardamos el fichero como `"PersonasViewController"`, por ejemplo.

Ahora abrimos el fichero `"PersonasViewController.h"` y añadimos un `NSArray` que será el que almacene las personas de la base de datos:

```
                //imports iniciales

                @interface PersonasViewController : UITableViewController
{
                    NSArray *_listadoPersonas;
                }
                @property (nonatomic, retain) NSArray *listadoPersonas;

                @end
```

Ahora abrimos el fichero `"PersonasViewController.m"`, añadimos el `@syntetize`, importamos los ficheros de gestión de la base de datos, la clase `Persona` y todo el código básico para la gestión de la vista de la tabla:

```
#import "PersonasViewController.h"
#import "Persona.h"
#import "BDPersistencia.h"

@implementation PersonasViewController

@synthesize listadoPersonas = _listadoPersonas;

- (id)initWithStyle:(UITableViewStyle)style
{
    self = [super initWithStyle:style];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)dealloc
{
    [super dealloc];

    self.listadoPersonas = nil;
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in
use.
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    [super viewDidLoad];

    self.title = @"Listado de personas";

    self.listadoPersonas = [BDPersistencia
database].arrayPersonas;
}

- (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 ==
UIInterfaceOrientationPortrait);
                }

                #pragma mark - Table view data source

                - (NSInteger)numberOfSectionsInTableView:(UITableView
*)tableView
                {
                    // Return the number of sections.
                    return 1;
                }

                - (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:
                (NSInteger)section
                {
                    // Return the number of rows in the section.
                    return [self.listadoPersonas count];
                }

                - (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:
                (NSIndexPath *)indexPath
                {
                    static NSString *CellIdentifier = @"Cell";

                    UITableViewCell *cell = [tableView
                    dequeueReusableCellWithIdentifier:CellIdentifier];
                    if (cell == nil) {
                        cell = [[[UITableViewCell alloc]
initWithStyle:UITableViewCellStyleSubtitle
                        reuseIdentifier:CellIdentifier] autorelease];
                    }

                    // Configure the cell...
                    Persona *persona = (Persona *)[self.listadoPersonas
objectAtIndex:indexPath.row];
                    cell.textLabel.text = [NSString stringWithFormat:@"%@
%@ (%d)", persona.nombre,
                    persona.apellidos, persona.edad];
                    cell.detailTextLabel.text = persona.localidad;

                    return cell;
                }



                #pragma mark - Table view delegate

                - (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:
                (NSIndexPath *)indexPath
```

```
                {
                    // Navigation logic may go here. Create and push
another view controller.

                }

                @end
```

Ahora sólo nos queda incorporar la vista que acabamos de crear dentro de nuestra aplicación. Para ello vamos a llamarla desde el Delegate y a meterla dentro de un "Navigation Controller". Primero debemos de añadir un "Outlet" en la clase delegada hacia un "Navigation Controller":

```
                //imports iniciales


                @interface sesion03_ejemplo1AppDelegate : NSObject
<UIApplicationDelegate> {
                    UIWindow *_window;
                    UINavigationController *_navController;
                }

                @property (nonatomic, retain) IBOutlet UIWindow *window;
                @property (nonatomic, retain) IBOutlet
UINavigationController *navController;

                @end
```

Ahora hacemos click sobre la vista "MainWindow.xib" y arrastramos un "Navigation Controller" desde el listado de la columna de la derecha hacia la columna de la izquierda donde pone "Objects". Nos aparecerá el "Navigation Controller" dibujado en pantalla. Ahora desplegamos el "Navigation Controller" y hacemos click sobre el "View Controller" que hay dentro. Nos vamos a la pestaña de "Identity Inspector" de la columna de la derecha y escribimos dentro de "Class" el nombre de la vista del listado de personas: "PersonasViewController".

Finalmente hacemos click sobre el objeto *"Delegate"*, vamos a la pestaña de "Show Connections Inspector" y arrastramos desde "navController" hasta el objeto "Navigation Controller" de la columna de la izquierda. Ya tenemos los controladores y las vistas conectadas.

Una vez hecho esto nos queda indicar dentro de nuestro código que la aplicación arranque con el Navigation Controller, para ello escribimos lo siguiente al final del método "didFinishLaunchingWithOptions" de la implementación de la clase delegada:

```
                self.window.rootViewController = self.navController;
                [self.window makeKeyAndVisible];
                return YES;
```

Ya está listo todo y preparado para compilar. Si ejecutamos el proyecto veremos que nos

aparece el listado de personas en la vista de tabla.