



**universidad  
cenfotec\_**  
tecnologías digitales



universidad  
cenfotec\_

# iOS Development

By César Brenes Solano

# Información General

- Email: [cesarbreso@icloud.com](mailto:cesarbreso@icloud.com)
- Github: <https://github.com/cenfoiOS/iOSCenfotecCourseJanuary>

# ¿Qué es iOS?

- Sistema operativo móvil utilizado en algunos dispositivos Apple
- Interfaz basada en el concepto de manipulación directa

# Qué es Objective C

- Orientado a objetos
- Creado en 1986 como una extensión al lenguaje C, añadiéndole: clases, métodos, protocolos, excepciones, propiedades y categorías.

# ARC



# ARC

- Automatic Reference Counting
- Se incluyó a partir de Xcode 5
- Mantenimiento de la memoria automáticamente

# ¿Como funciona iOS?





# Herramientas





IDE que nos ofrece apple  
para desarrollar  
aplicaciones de iOS

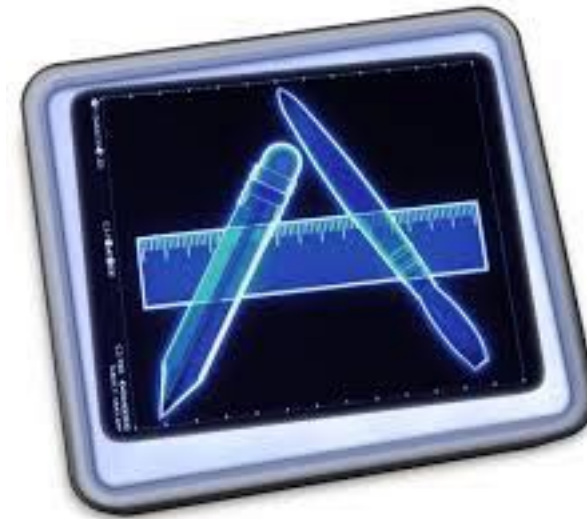
# iOS simulador

Permite simular tanto el  
iphone como el ipad en  
nuestro Mac



# iOS instruments

Herramienta de análisis que  
nos ayuda a optimizar y  
monitorear nuestra  
aplicación

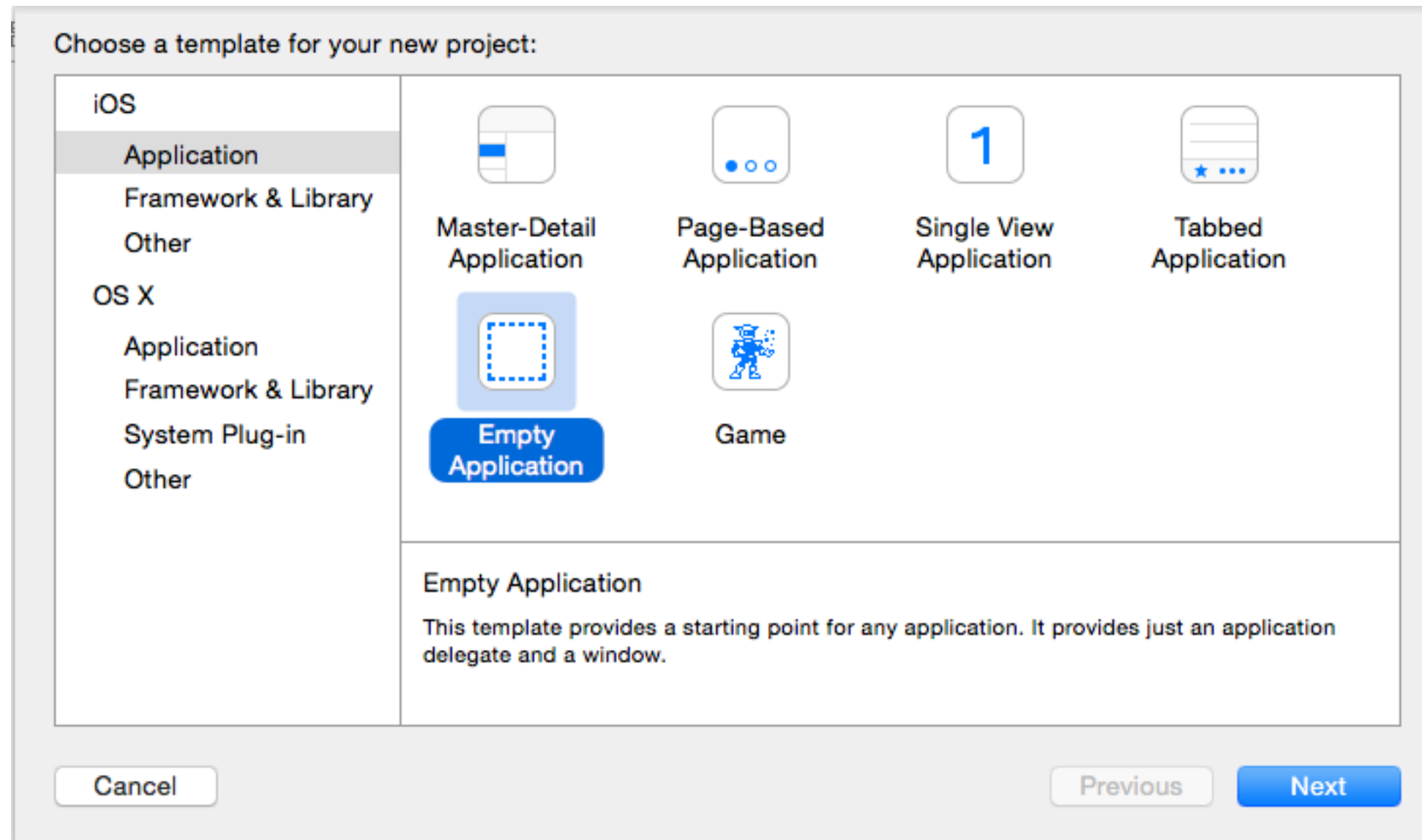


- **Nuestro primer proyecto**

Vamos a crear nuestro primer proyecto en Xcode para conocer la herramienta



# Tipos de Proyectos



# Master-Detail Application

Aplicaciones de iPad que muestren más de una vista en la pantalla al mismo tiempo. Suele tener una vista de control(listado) y una vista en detalle de los elementos





# Paged-Based Application

Aplicaciones que utilizan paginación, como revistas o libros.





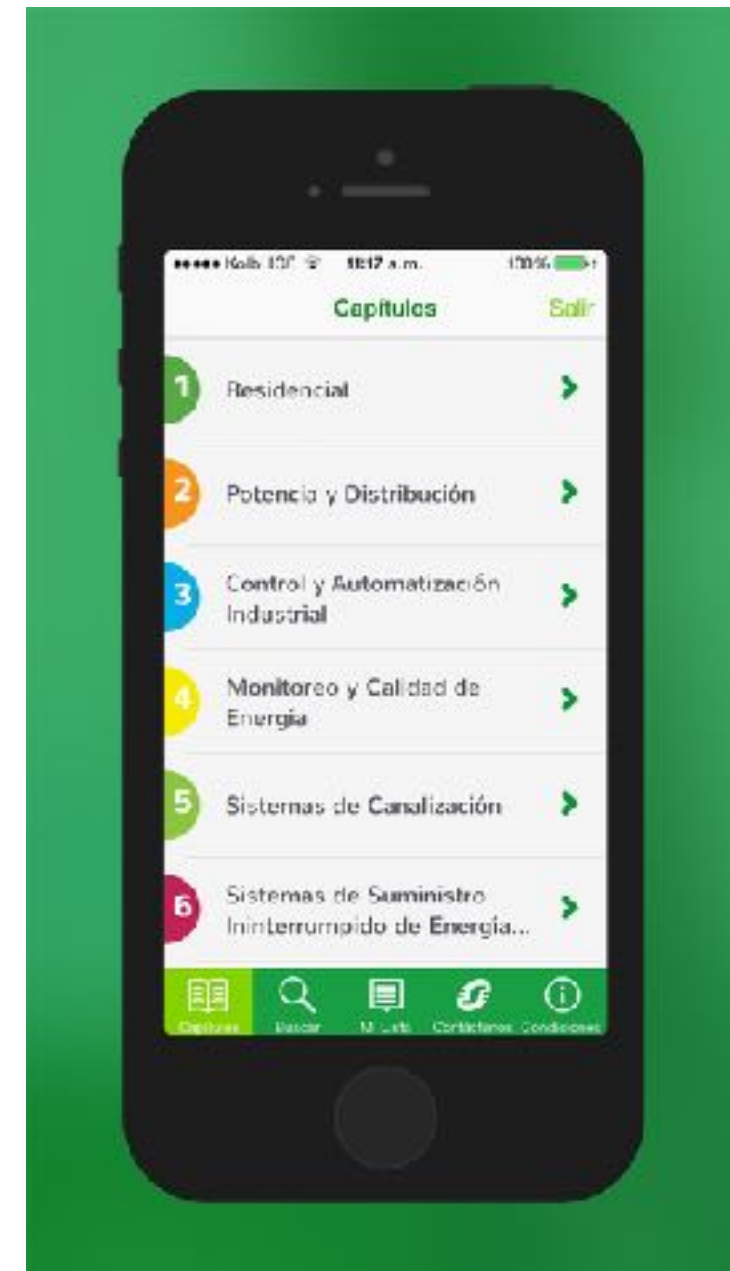
# Single View Application

Aplicaciones que utilizan una vista simple para implementar su interfaz



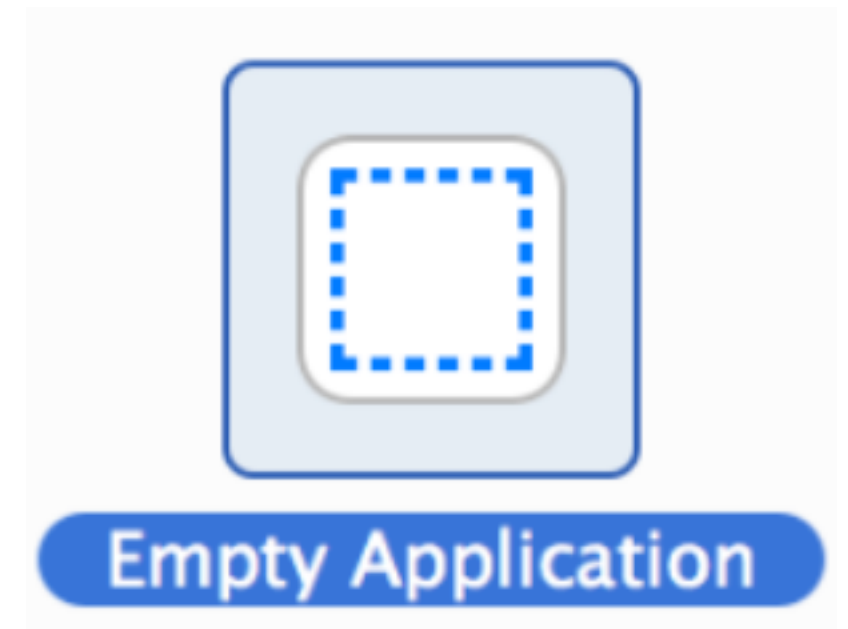
# Tabbed Application

Aplicaciones que permiten al usuario elegir entre varias opciones (Tabs)



# Empty Application

La plantilla funciona como punto de partida de cualquier aplicación

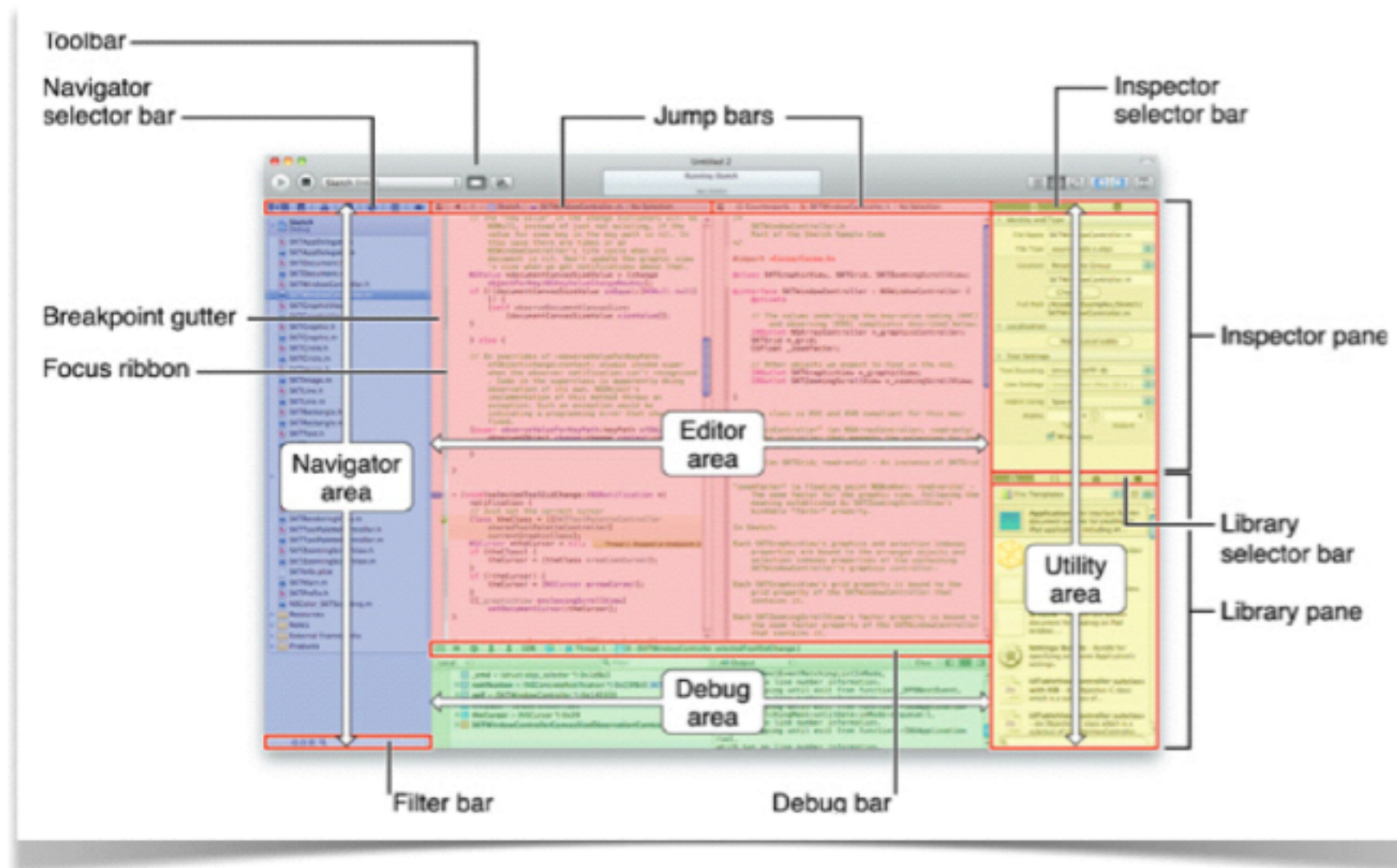


# Game Application

La plantilla funciona como punto de partida para todas aquellas aplicaciones que sean un juego



# Herramientas en Xcode



# Interfaz Gráfica



# Interfaz Gráfica

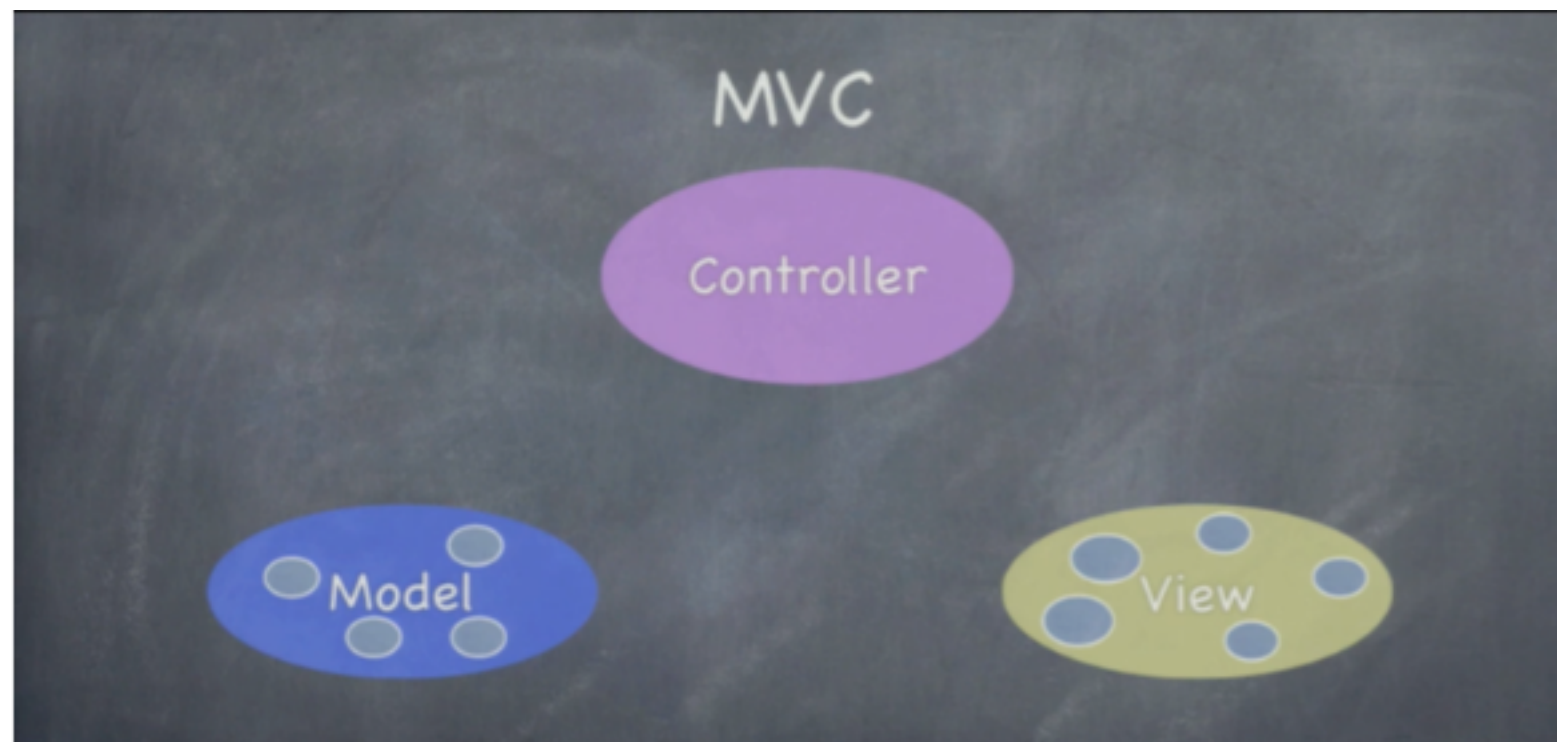
Existen tres formas para poder crearla:

1-Storyboard

2- Xib File

3- Código

# Modelo Vista Controlador

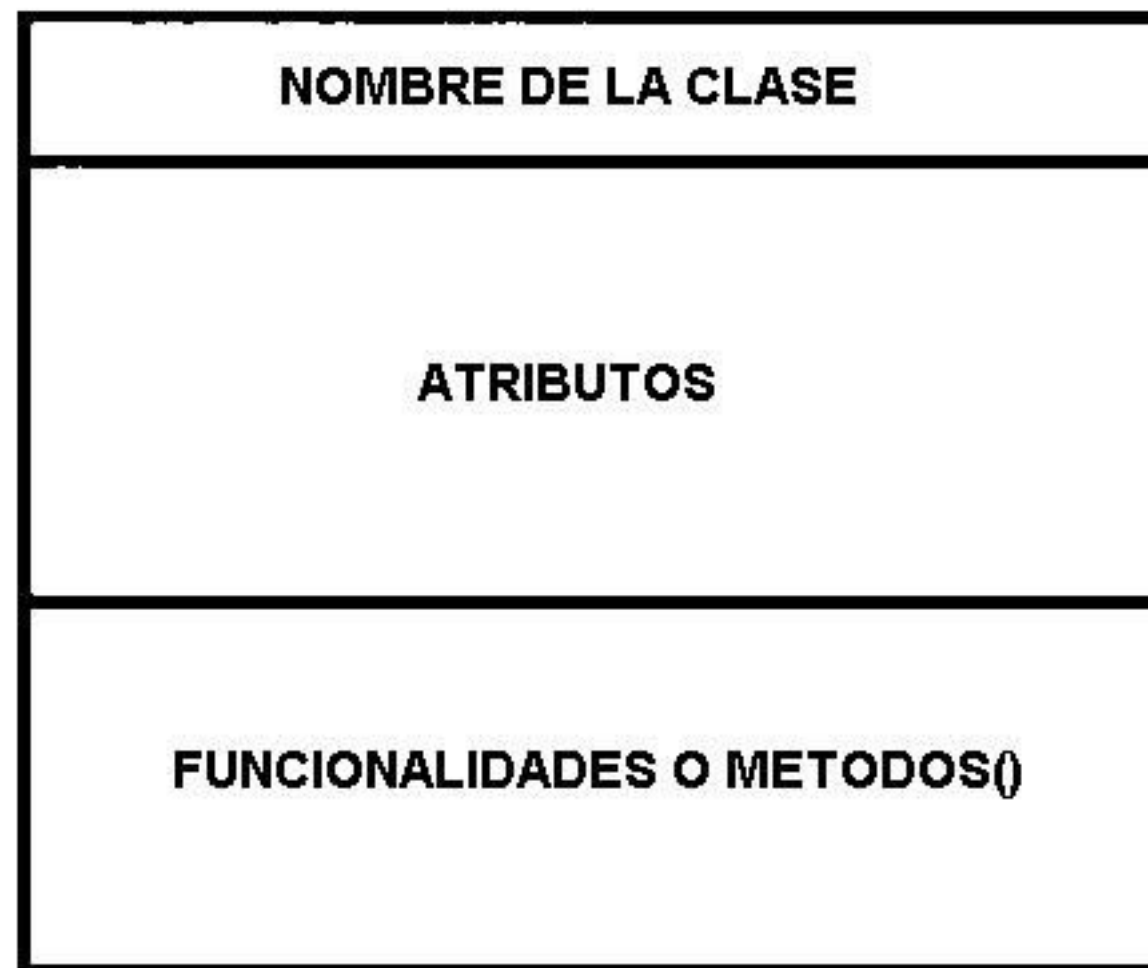




# MVC

- Modelo: Es el encargado de manejar los datos de nuestra aplicación así también como ellos serán manipulados
- Vista: Es el responsable de la representación visual del modelo y del control que el usuario pueda tener con esta
- Controlador: Este funciona como un mediador que coordina todo el trabajo. El accede a los datos desde el modelo y lo despliega en la vista.

# {clases}



# Sintaxis

- La sintaxis para la creación de clases consta de dos partes:
  - MyClase.h (header file)
  - MyClase.m (implementation file)

# Ejemplo

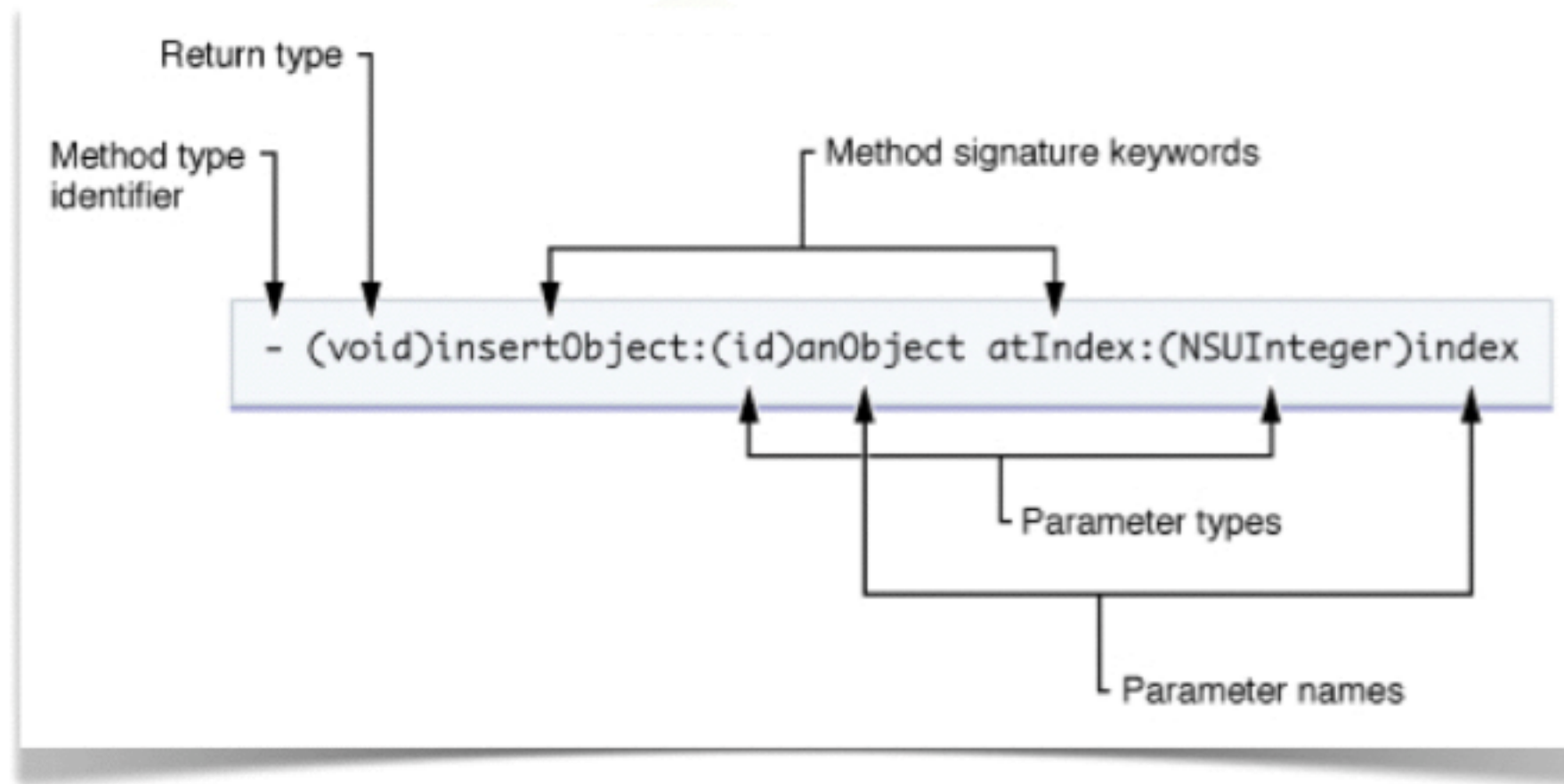
```
//  
// CenfotecClass.h  
// dsd  
//  
// Created by Cesar Brenes on 2/1/15.  
// Copyright (c) 2015 Cesar Brenes. All rights reserved.  
//  
  
#import <Foundation/Foundation.h>  
  
@interface CenfotecClass : NSObject  
// Alls declarations  
@end
```

```
//  
// CenfotecClass.m  
//  
// Created by Cesar Brenes on 2/1/15.  
// Copyright (c) 2015 Cesar Brenes. All rights reserved.  
//  
  
#import "CenfotecClass.h"  
  
@implementation CenfotecClass  
//Attributes  
}  
  
@end
```

# Métodos



# Sintaxis



# Llamadas a métodos

- Invocación sin parámetros:

```
[object method];
```

- Con un parámetro:

```
[object methodWithInput:input];
```

- Con valor de retorno:

```
output = [object methodWithOutput];
```

- Con valor de retorno y parámetro de entrada:

```
output = [object methodWithInputAndOutput:input];
```

# Ejemplos

```
- (int) multiplica: (int)a por:(int)b {  
    return a * b;  
}
```

```
- (NSString *) cadenaResultado: (int)resultado {  
    NSString *res = [NSString stringWithFormat:@"El resultado es: %d", resultado];  
    return res;  
}
```

```
int resultado = [self multiplica:2 por:3];  
NSString *cadena = [self cadenaResultado:resultado];  
NSLog(@"%@", cadena);
```



# Properties

- Las properties permiten generar automáticamente métodos de acceso para los atributos de las clases:

```
@interface person : NSObject {  
    NSString *name;  
    NSString *surname;  
}  
  
@property (retain) NSString* name;  
@property (retain) NSString* surName;  
  
@end
```

person.h

# Tipos

- ReadOnly: Solo garantiza el get del atributo y bloquea el set
- Atomic: Se utiliza en ambientes donde hay multi-threaded, si al declararse solo se usa el identificar @property la propiedad por default es atomic
- Nonatomic: Para ambientes donde no se utilice multi-threaded o nosotros mismos creemos un ambiente seguro para el mismo

# Tipos

- Strong: Crea una referencia fuerte que permite la vida del objeto a través de la ejecución del app, solo será destruido cuando sea cambiado a nil
- Weak: Significa que usted no quiere tener el control sobre la vida de ese objeto

# Diferencias

- `self.nombrePropiedad`: Mantiene set y get standards
- `@synthesize nombrePropiedad`: Custom set y get
- `_nombrePropiedad`: Custom set y get

# Ciclo de vida

