Universidad de El Salvador
Facultad de Ingeniería y Arquitectura
Escuela de Ingeniería en Sistemas

Programación en Dispositivos Móviles

Guía de Laboratorio N°04 B

"Introducción al manejo de Bases de Datos con SQLite"

Version IOS

Objetivo:

Que el alumno:

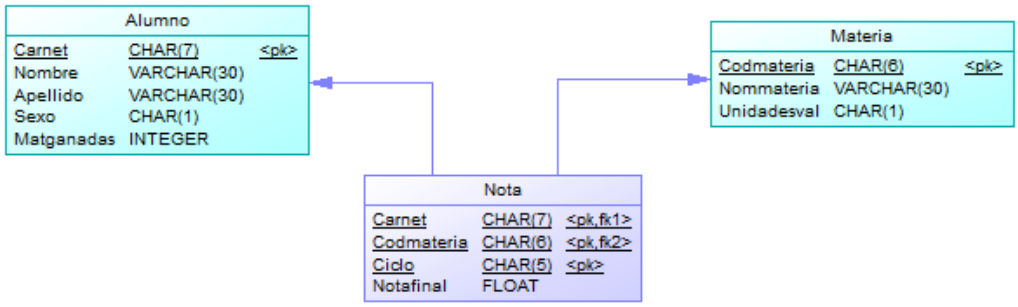
1- Implemente una base de datos a partir de su modelo físico en una aplicación IOS, usando el lenguaje SQLite.
2- Aprenda a crear una base de datos, y usar los métodos Update, Insert, Delete y Query para llevar su administración.

Descripción:

Esta práctica consistirá en realizar una aplicación que permita crear una base de datos o abrir una existente, conectarse a ella y que pueda manejarse desde la interfaz de usuario con sus respectivas pantallas.

Contenido

BASE DE DATOS ALUMNO, MODELO FISICO.

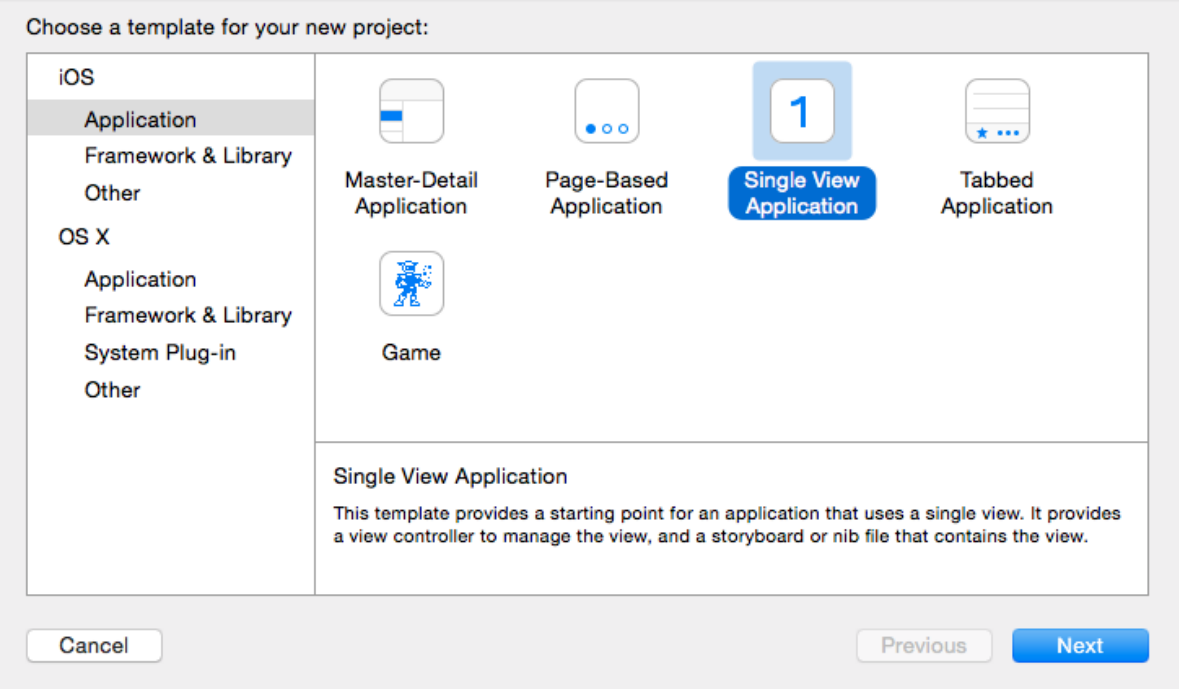


Creación del proyecto

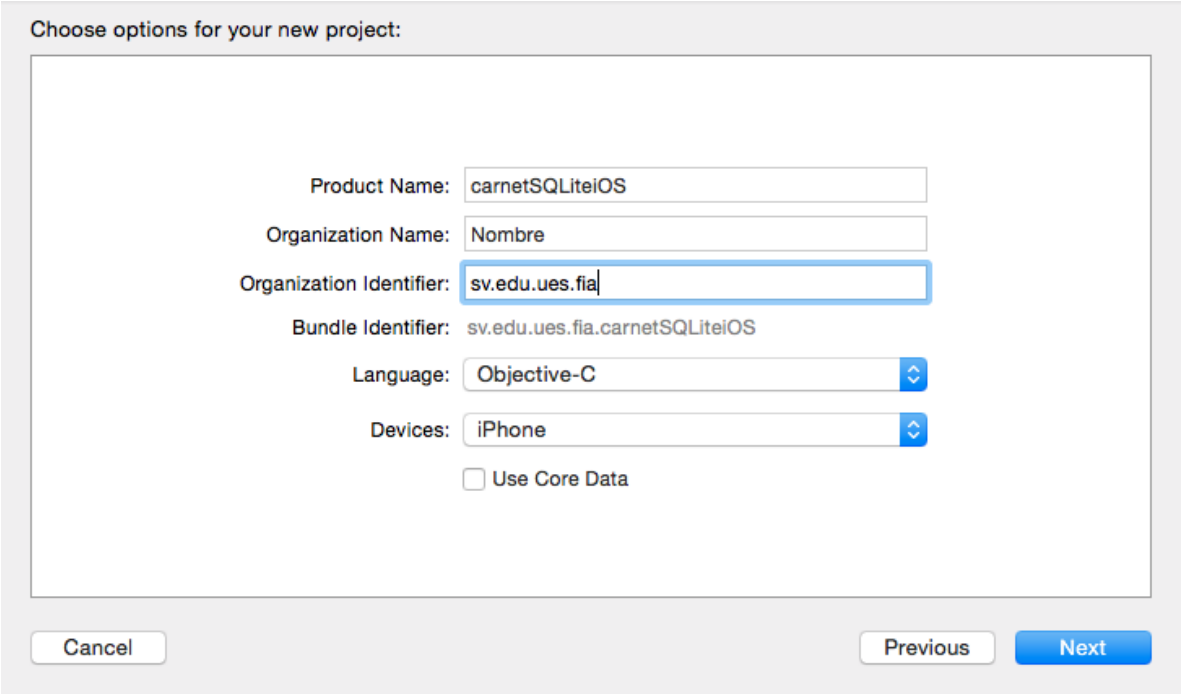
Se hara un mantenimiento por cada tabla del modelo de base de datos anterior

Main.storyboard

Al abrir Xcode, crear un nuevo proyecto de tipo “Single View Application”.

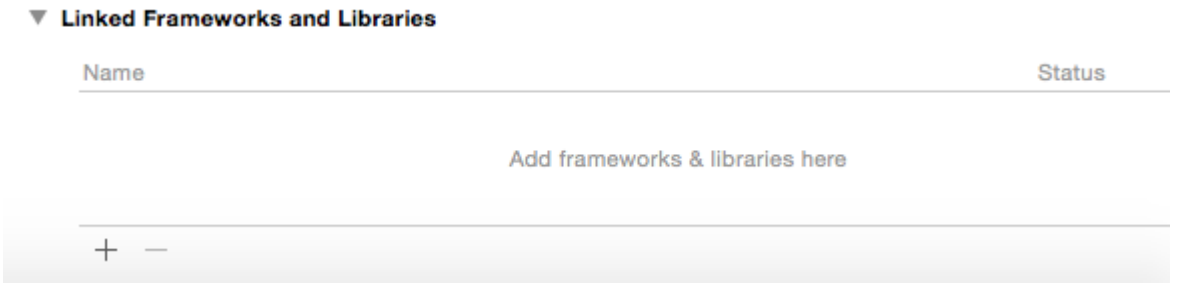


Al presionar Next , se presenta la ventana para ingresar el nombre del proyecto, como nombre colocar su carnet seguido de SQLiteiOS, en el nombre de organización colocar su nombre completo y las demás configuraciones realizarlas como se muestra en la imagen.



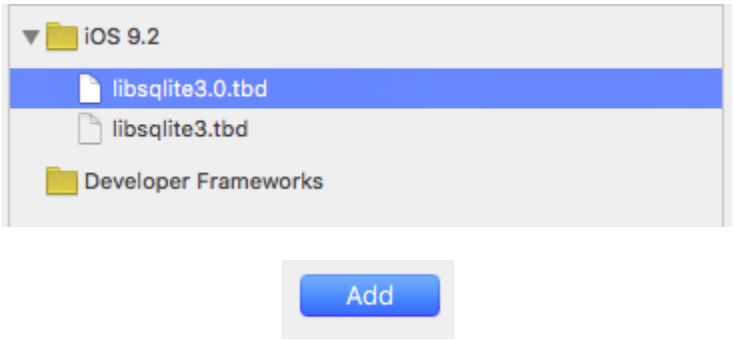
Cuando el proyecto es creado, se encuentra abierta la pestaña de opciones generales del proyecto, si no se encuentra seleccionarla.

En esta pestaña se encuentra una sección adonde se encuentran los frameworks y las librerías con la que se trabaja, como se muestra a continuación.

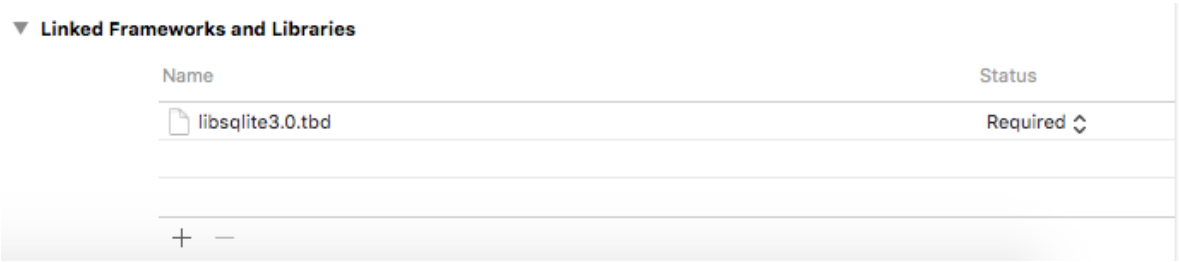


Debajo de la lista mostrada se encuentra un botón para agregar una librería o framework, hacer clic en el (+).

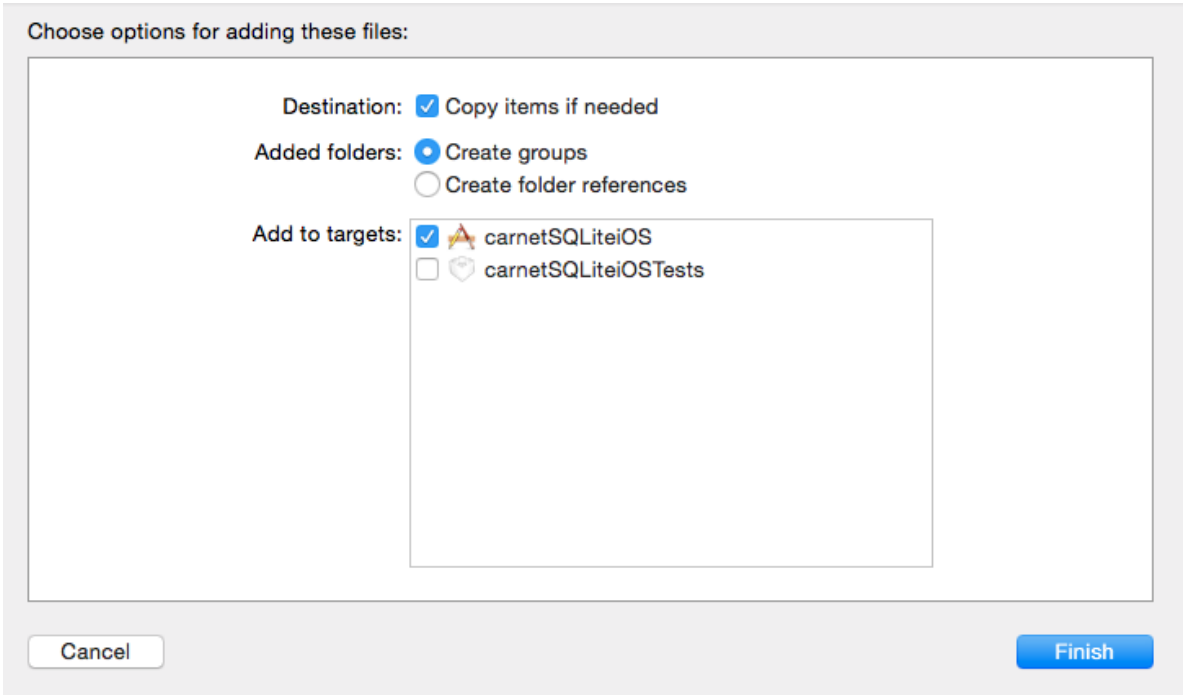
Al seleccionar se presentara una ventana como la que se muestra a continuación, en la cual se buscara la librería sqlite, se mostraran 2 opciones, seleccionar la librería “libsqlite3.0.dylib”, y hacer clic en Add.

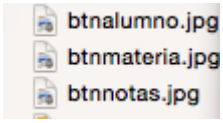


Al haber agregado la librería la sección de frameworks se observará como se muestra a continuación.



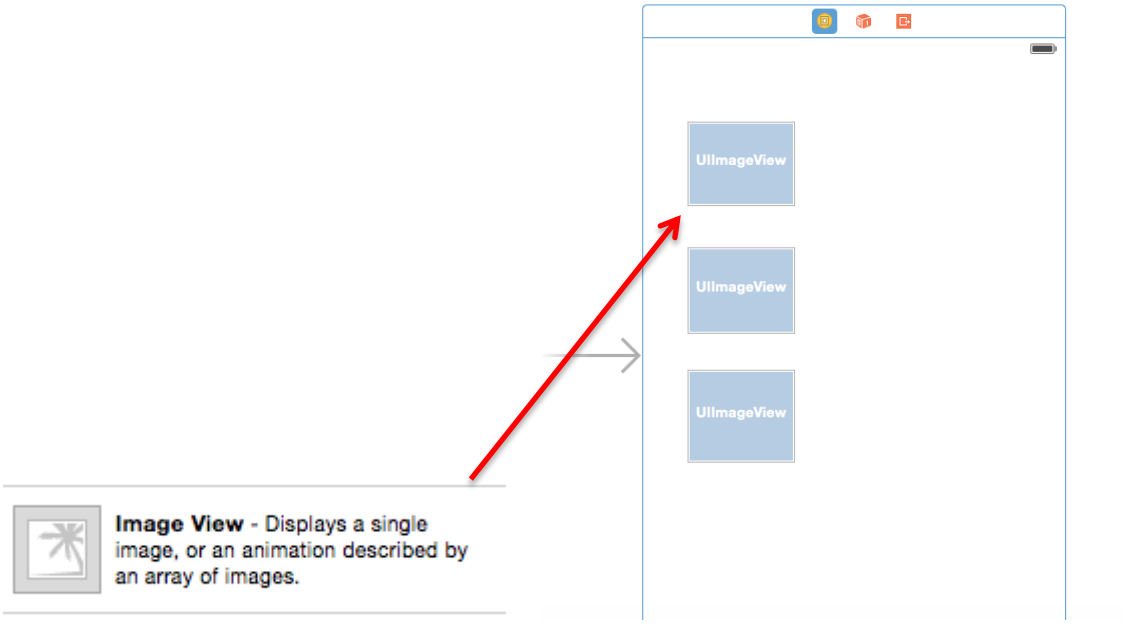
Descargar los archivos correspondientes a esta guía del aula virtual. Para que los archivos de imágenes descargados desde el aula puedan formar parte del proyecto, se debe de arrastrar dichos archivos hasta el proyecto en Xcode, al soltar el clic se presentara una ventana como la que muestra a continuación, configure dicha ventana como la imagen siguiente.





Teniendo las imágenes, acceder al archivo Main.storyboard, en este se podrá observar el ViewController principal, el cual se trabajara como menú principal, desde el cual se accederán a los ViewControllers que manejar los datos de Alumnos, Materias y Notas.

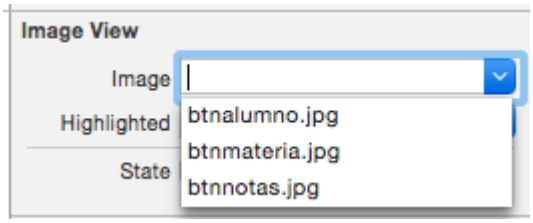
En este ViewController se agregar 3 archivos de tipo ImageView, cada uno de estos corresponderá a cada imagen descargada para la realización de la guía.



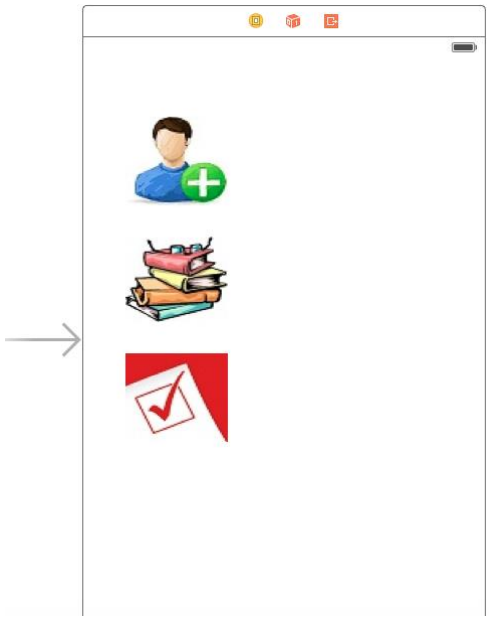
Al seleccionar un objeto ImageView, acceder a la pestaña de “Attributes inspector” ( ) ubicada en la barra de utilidades, al lado izquierdo.

En esta pestaña se encuentra una sección adonde se especifica la imagen que le corresponde al ImageView seleccionado.

Para el ImageView correspondiente a los alumnos seleccionar la imagen “btnalumno.jpg” y así sucesivamente para cada uno de los tipos de datos a gestionar en la aplicación.

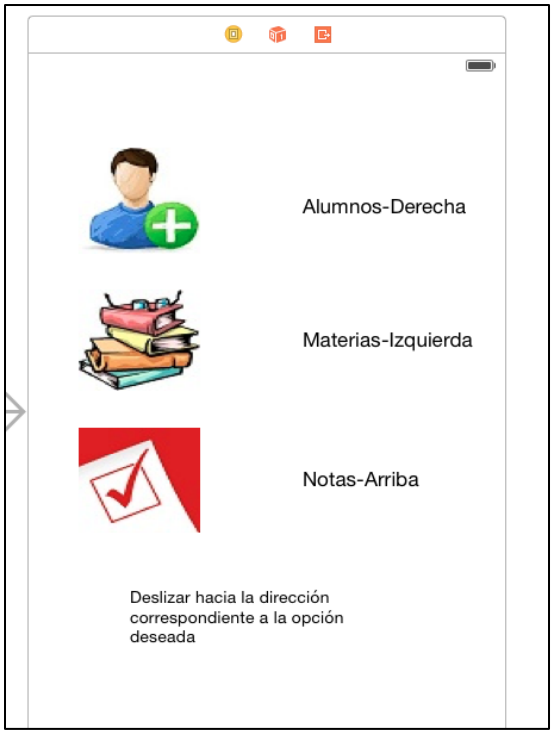


Al finalizar esta configuración se observará como se ven las imágenes en los “Image View”.



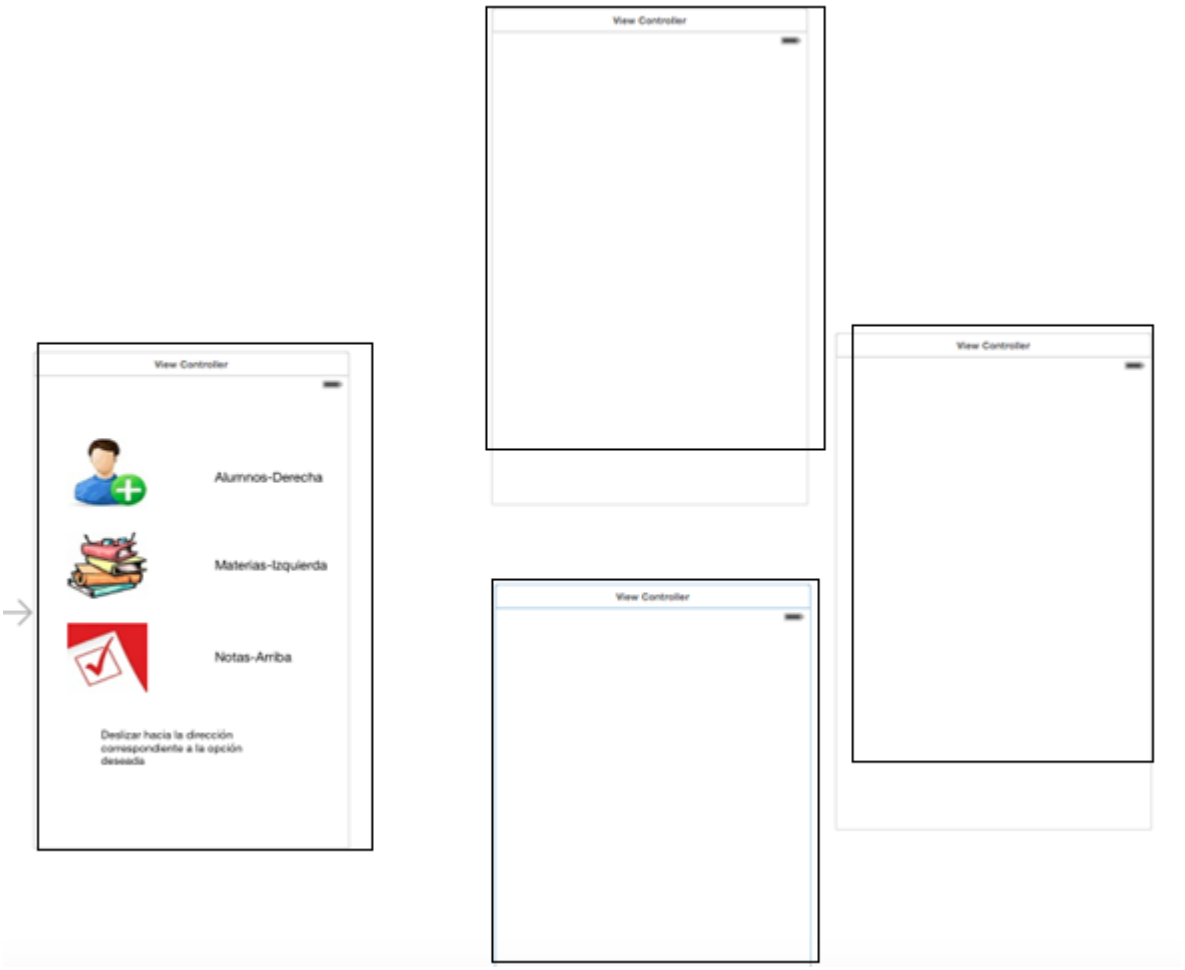
Siguiendo con la creación del ViewController principal, se agregarán 3 objetos Label y un objeto TextView, los cuales corresponden a indicaciones para acceder a las ventanas correspondientes.

Al tener todos los objetos creados, realizar los cambios necesarios para que el ViewController se vea de la siguiente manera.



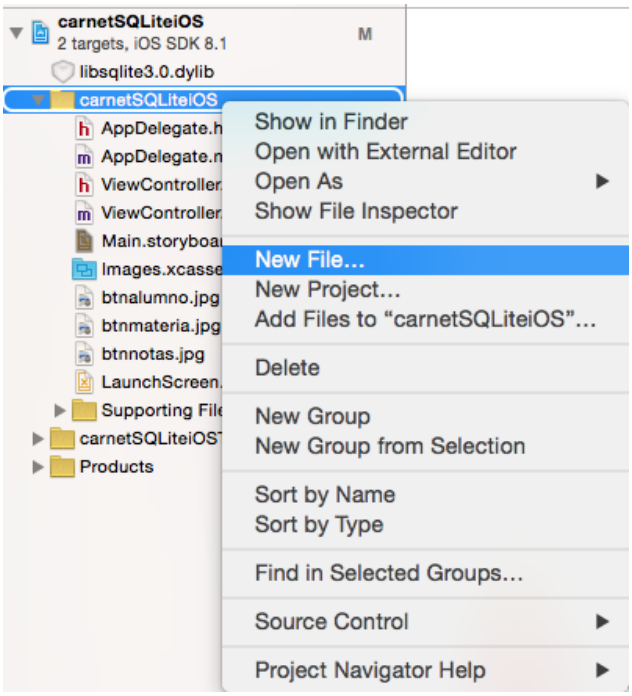
Agregar 3 objetos de tipo ViewController al Main.storyboard, los cuales corresponderán cada uno a Alumno, Materia y Nota.

Al agregar los View controller se verá el main storyboard de la siguiente manera:

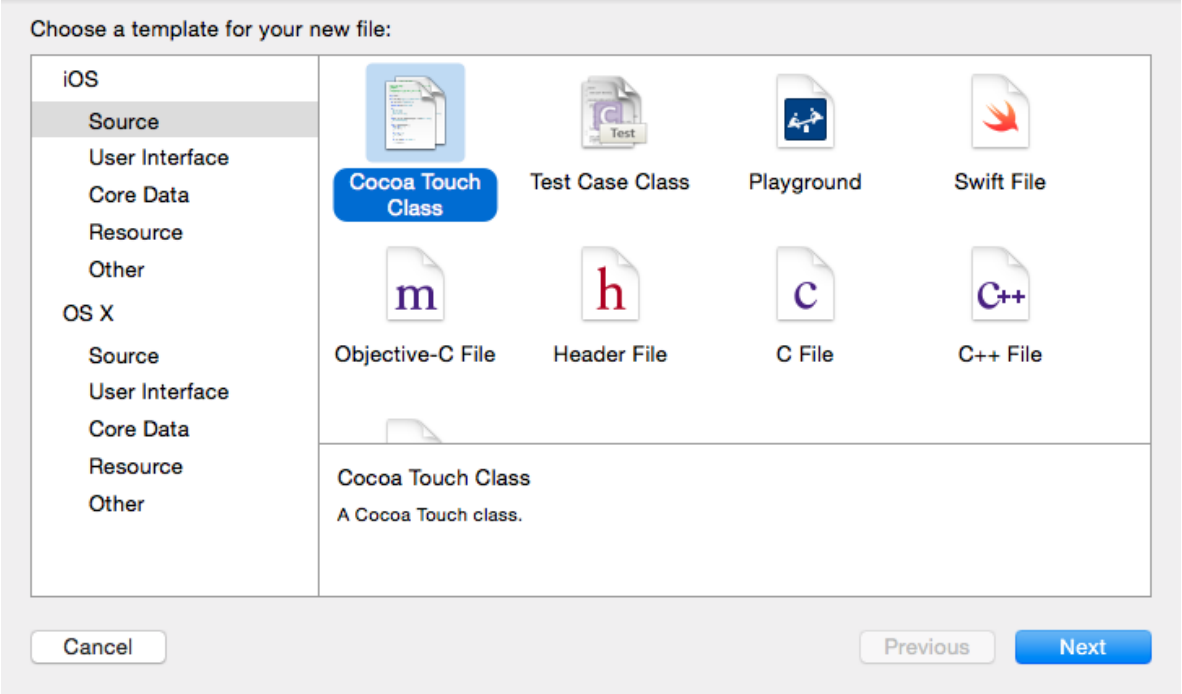


Se procede a crear archivos de tipo UIViewController para enlazar cada uno de los ViewControllers anteriormente creados.

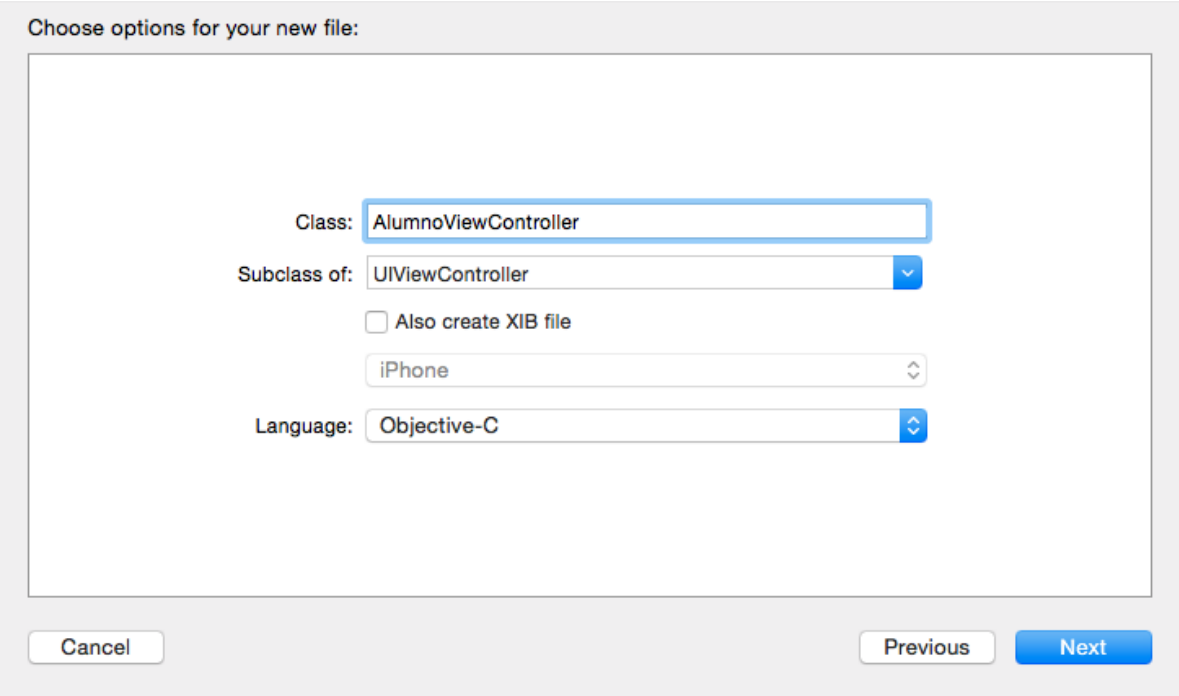
Seleccionar la carpeta del proyecto en la barra de navegacion, a la izquierda de la pantalla, hacer clic derecho sobre ella y seleccionar la opción “New File…”



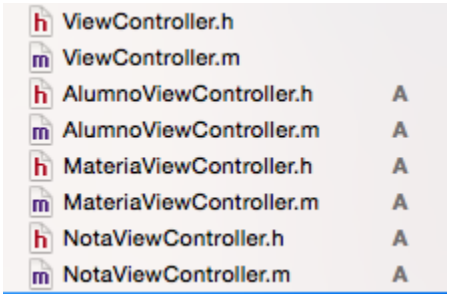
Seleccionar tipo de datos para iOS, una clase Cocoa Touch Class, Presionar Next



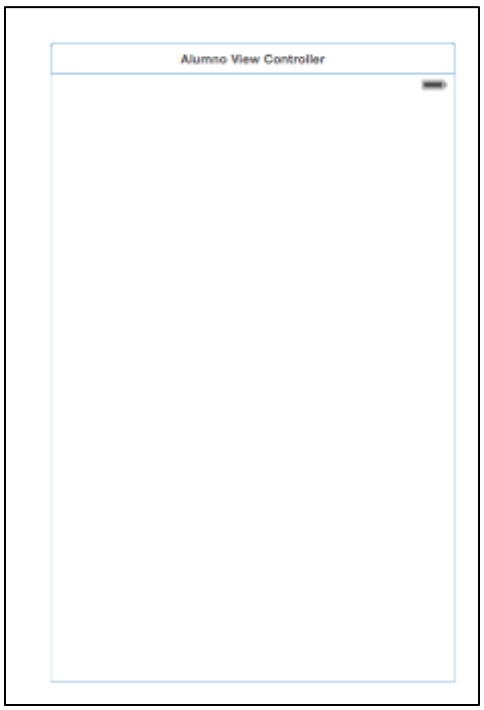
En la siguiente ventana seleccionar como subclase (Subclass Of), el tipo de objeto UIViewController y modificar el nombre de la clase como se muestra a continuación.



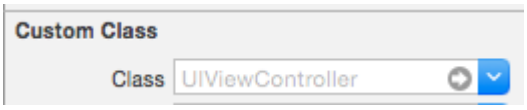
Crear los otros archivos correspondientes para Materia y Nota, cada uno de ellos posee la misma subclase que el creado anteriormente. Al finalizar esta creación los archivos creados quedarían como se muestra a continuación.



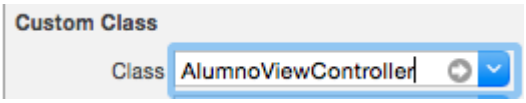
Para realizar la conexión de los archivos .h y .m creados, con cada uno de los ViewControlles agregados al Main.storyboard, acceder al Main.storyboard y seleccionar el ViewController correspondiente a Alumnos.



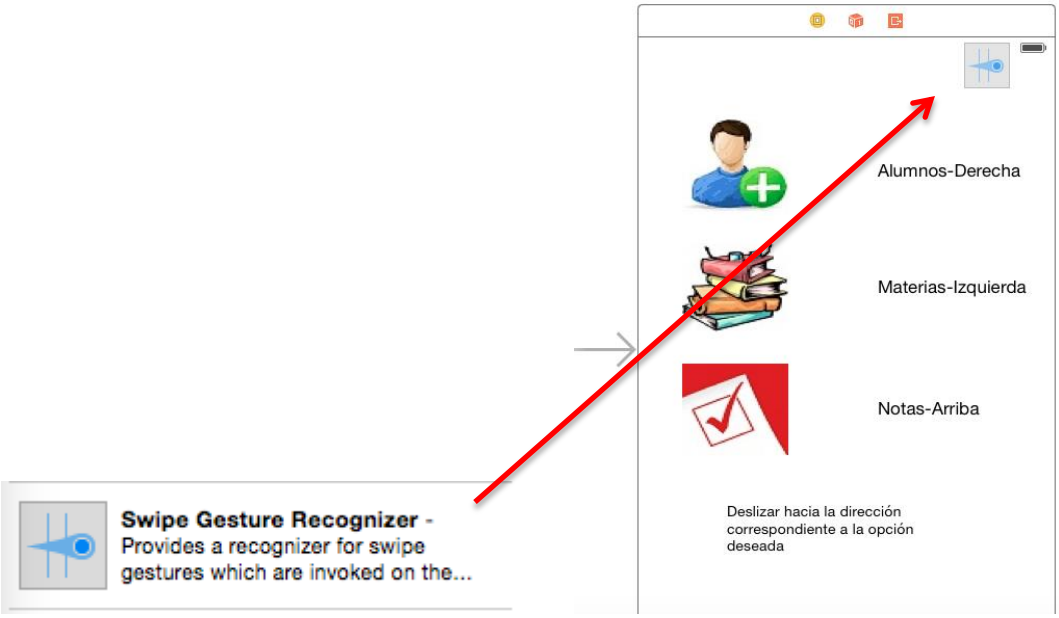
Ubicarse en la pestaña “Identity inscpector”  ubicada en la barra de utilidades de la derecha, en esta se encuentra una sección llamada Custom Class.



Para el ViewController correspondiente agregar la clase de los archivos .h y .m anteriormente creados para Alumnos, como se muestra a continuación. Realizar lo mismo para los demás ViewControllers con sus archivos correspondientes.

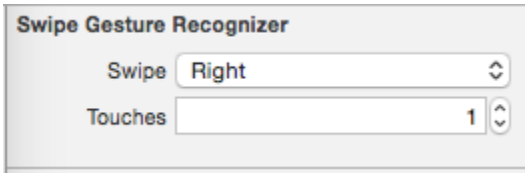


Para establecer la navegación entre pantallas se utilizara el reconocimiento de gestos, en este caso el “Swipe Gesture Recognizer”, por lo cual al ViewController que servirá de menú, se agregara un objeto de dicho tipo, arrastrándolo hasta el mismo.

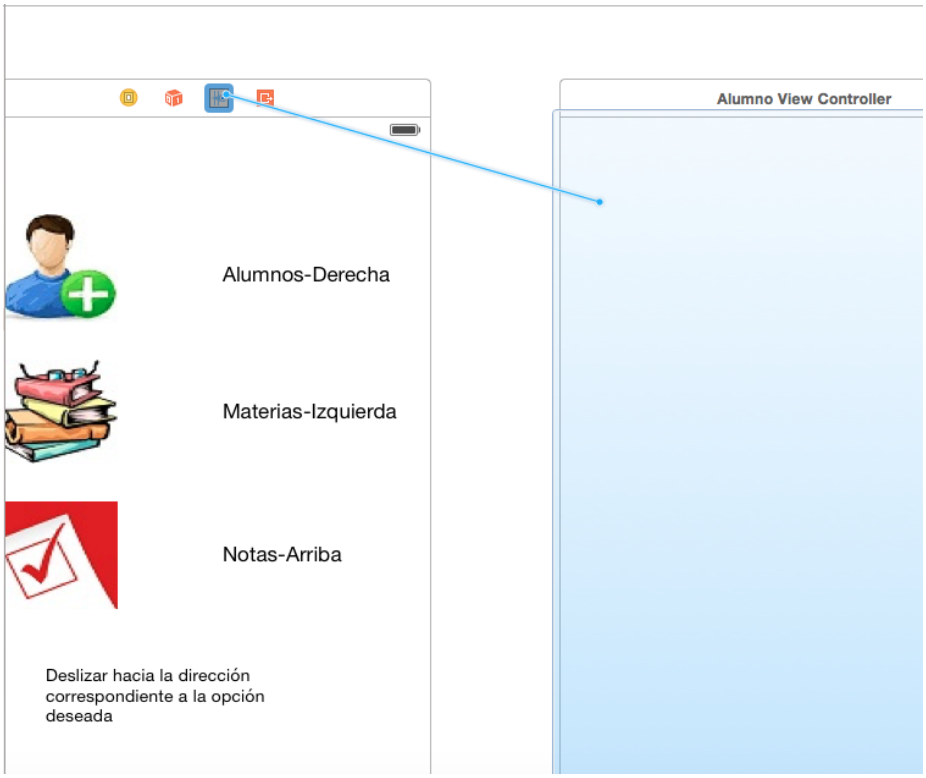


Al soltar el objeto este se verá agregado en la parte inferior del ViewController como se muestra a continucion, al seleccionar este elemento en la barra inferior se observa que se despliegan sus características en la barra de utilidades.

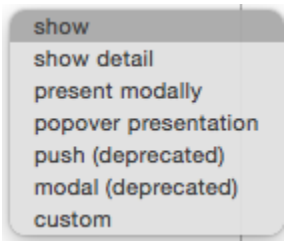
Acceder al "Attributes inspector" () en el cual se observara una sección llamada "Swipe Gesture Recongnizer", adonde se especifica hacia qué dirección se dirige el movimiento para realizar la acción deseada y cuantos toques se deben de realizar.



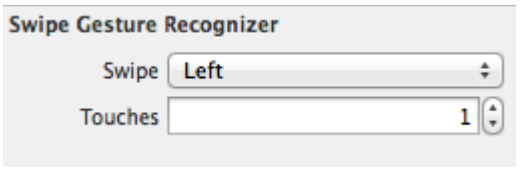
Para establecer la navegación de la pantalla de menú al ViewController de Alumnos, hacer clic derecho en el icono de "Swipe Gesture Recognizer" de la parte inferior del ViewController y arrastrarlo hasta el AlumnoViewController, como se muestra a continuación.

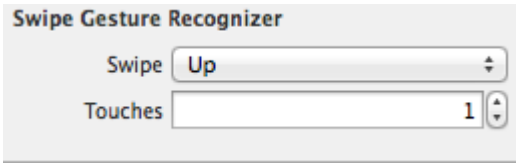


Al soltar la conexión se presenta una ventana como la que se muestra a continuación, seleccionar la opcion "modal", ya que esta permitirá la transición directa del menu principal a AlumnoViewController.

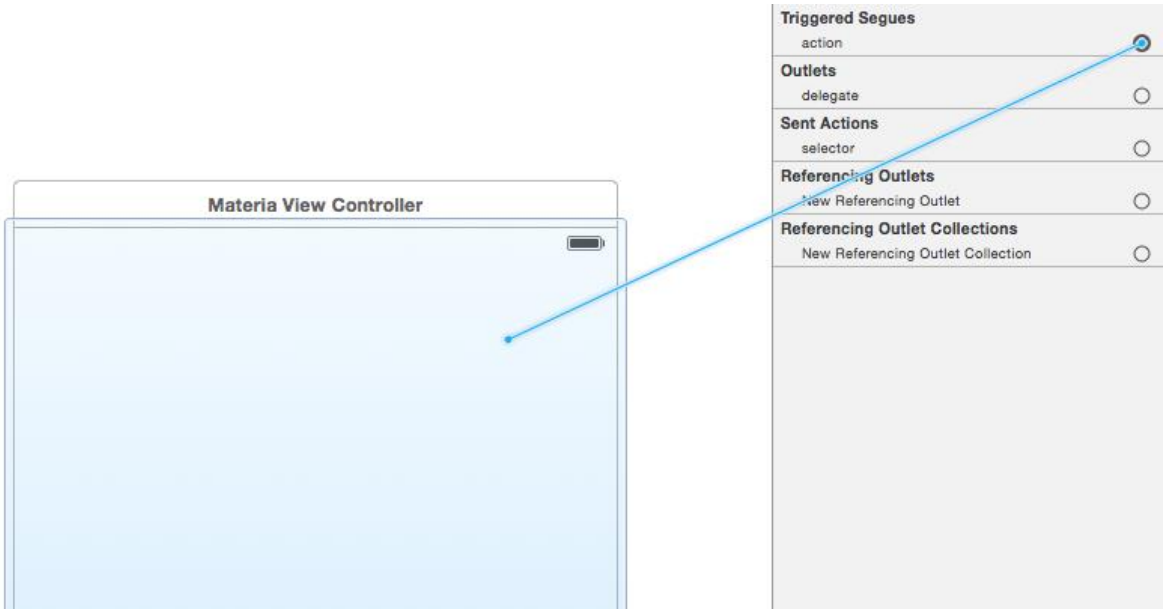


Realizar la misma configuración agregando 2 “Swipe Gesture Recognizer”, pero se realizarán los cambios en la dirección del “Swipe”, asignar según correspondan.

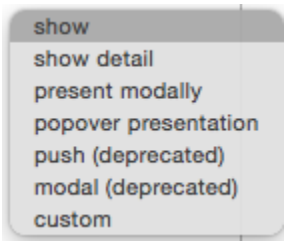




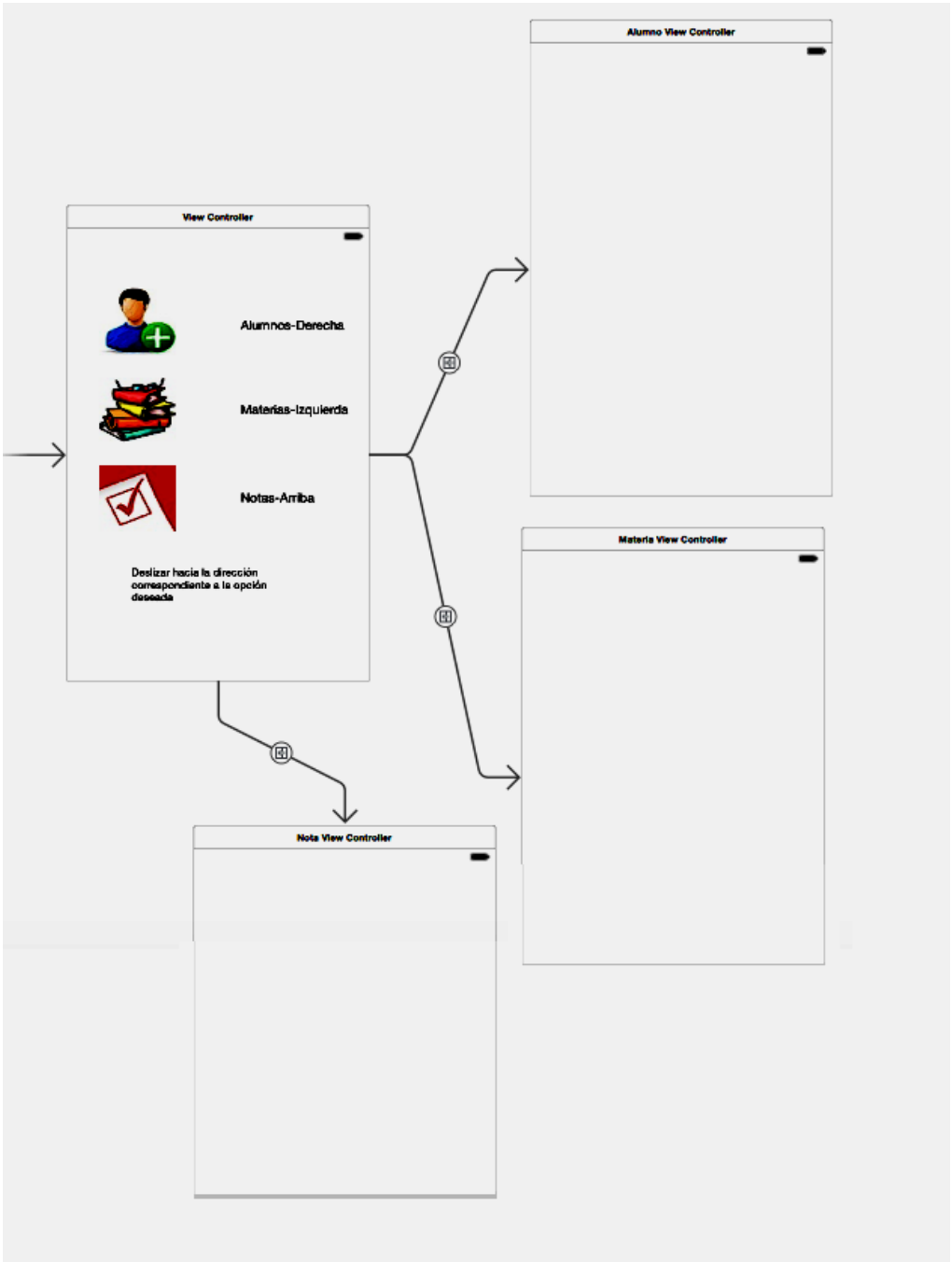
Si es difícil realizar la modalidad de arrastre, puede seleccionar el objeto “Swipe Gesture Recognizer” en la parte inferior del ViewController y acceder a la pestaña de “Connections Inspector” ( ), se observara una seccion de “Triggered Segues”, en la opción de “action”, hacer clic y arrastrar hasta el ViewController con el cual se quiere realizar la transición.



Al seleccionar y soltar el clic se presentara una ventana, en la cual seleccionar la opción “modal”.

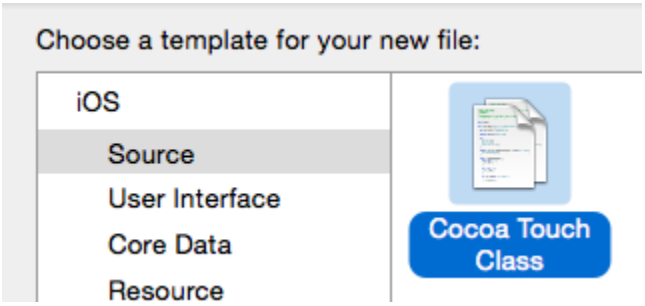


Al establecer las 3 conexiones el archivo Main.storyboard se observara de la siguiente manera.

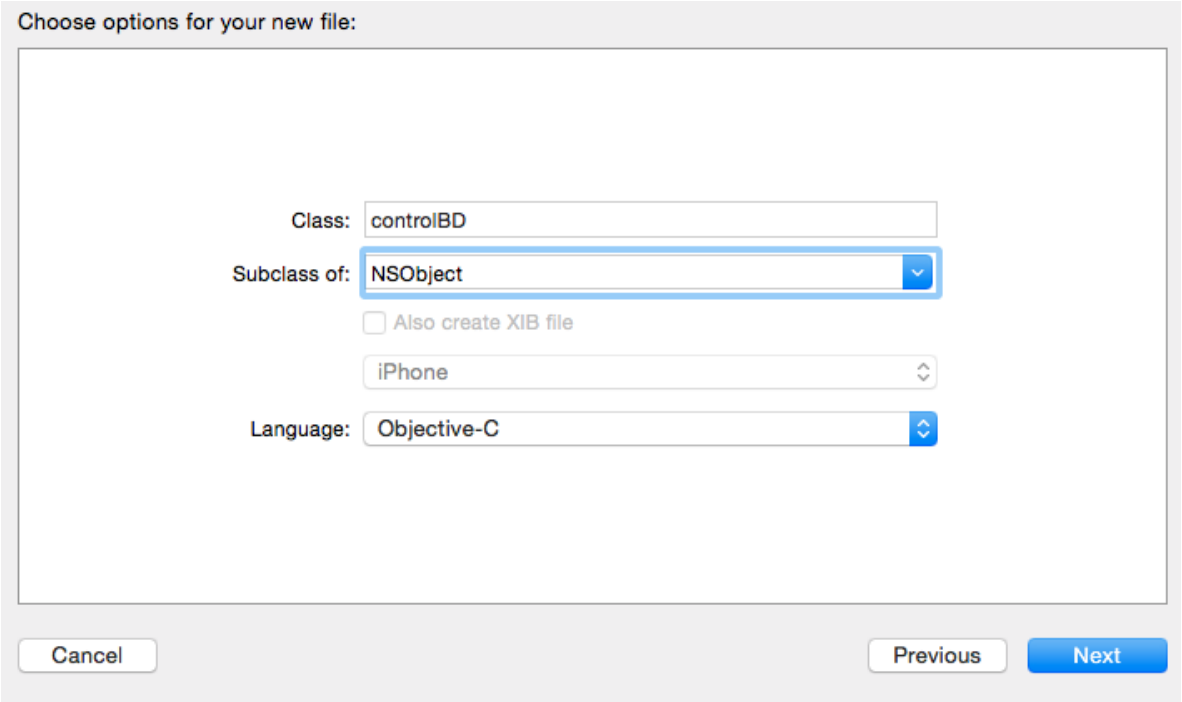


Clase Controladora de la Base de Datos.

Crear un nuevo archivo de tipo Cocoa Touch Class.



Colocar como nombre de la clase “controlBD” y elegir como subclase el tipo NSObject, dejar las configuraciones como la siguiente imagen:



Aparecerá en el proyecto los archivos creados:

Abrir controlBD.h, importar la librería sqlite3.h y agregar el siguiente código:

```
#import <Foundation/Foundation.h>
#import <sqlite3.h>

@interface controlBD : NSObject

    @property (strong,nonatomic)NSString *dbPath;
    +(controlBD *) sharedInstance;
    -(void) crearOabrirDB;

@end
```

Se define un NSString el cual permite establecer la dirección en la cual se guardara la base de datos, se crea además método con un objeto compartido sharedInstance para que pueda ser utilizado en cualquier parte del código y un método para crear o abrir la base de datos.

Abrir controlBD.m e importar controlBD.h, además modificar el interface de la siguiente manera:

```
#import "controlBD.h"

@interface controlBD()
{
    sqlite3 *alumnoDB;
    NSString *dbPathString;

}
@end
```

El objeto de tipo sqlite establece las conexiones y utiliza la librería controladora y el NSString que permite establecer la dirección en la cual se guardara la base de datos.

Agregar el método sharedInstance con el siguiente código:

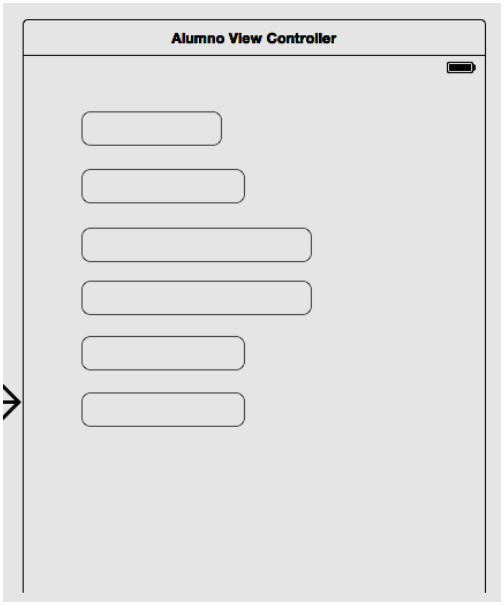
```
+(controlBD *)sharedInstance {
    static controlBD *myInstance = nil;

    // check to see if an instance already exists
    if (nil == myInstance) {
        myInstance  = [[[self class] alloc] init];
    }
    // return the instance of this class
    return myInstance;
}
```

Se creara el método crearOabrirDB, el cual crea la base desde cero si esta no se encuentra en la ruta especificada, de lo contrario se abre para trabajar con ella en la aplicación.

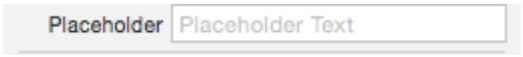
```
-(void) crearOabrirDB{
    NSArray *path =NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *docPath = [path objectAtIndex:0];
    dbPathString= [docPath stringByAppendingPathComponent:@"carnet.db"];
    char *error;
    NSFileManager *fileManager=[NSFileManager defaultManager];
    if (![fileManager fileExistsAtPath:dbPathString])
    {
        const char *dbPath = [dbPathString UTF8String];
        //crear la base de datos
        if (sqlite3_open(dbPath, &alumnoDB)==SQLITE_OK)
        {
            NSString *foreign=@"PRAGMA foreign_keys = ON";
            const char *foreign1=[foreign UTF8String];
            if (sqlite3_exec(alumnoDB, foreign1, NULL, NULL, &error) != SQLITE_OK)
            {
                NSLog(@"failed to set the foreign_key pragma");
                return ;
            }
            const char *sql_stmt="CREATE TABLE alumno (numalumno INTEGER NOT NULL PRIMARY KEY
                AUTOINCREMENT,carnet VARCHAR(7) NOT NULL UNIQUE, nombre VARCHAR(30) NOT NULL,apellido
                VARCHAR(30) NOT NULL ,sexo VARCHAR(1) NOT NULL,matganadas INTEGER ); ";
            sqlite3_exec(alumnoDB, sql_stmt, NULL, NULL, &error);
            sql_stmt="CREATE TABLE materia (nummateria INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,codmateria
                VARCHAR(6) NOT NULL UNIQUE,nommateria VARCHAR(30) NOT NULL, unidadesval VARCHAR(1) NOT
                NULL);";
            sqlite3_exec(alumnoDB, sql_stmt, NULL, NULL, &error);
            sql_stmt="CREATE TABLE nota (numnota INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,codmateria
                VARCHAR(6) NOT NULL,carnet VARCHAR(7) NOT NULL, ciclo VARCHAR(5) NOT NULL,notafinal float
                INTEGER,foreign key (carnet) references ALUMNO (CARNET) on delete restrict on update
                restrict,foreign key (codmateria) references MATERIA (CODMATERIA) on delete restrict on
                update restrict ); ";
            sqlite3_exec(alumnoDB, sql_stmt, NULL, NULL, &error);
            sqlite3_close(alumnoDB);
            _dbPath=dbPathString;
            return;
        }
    }
    const char *dbPath = [dbPathString UTF8String];
    //crear la base de datos

    if (sqlite3_open(dbPath, &alumnoDB)==SQLITE_OK)
    {
        _dbPath=dbPathString;

    }

}
```

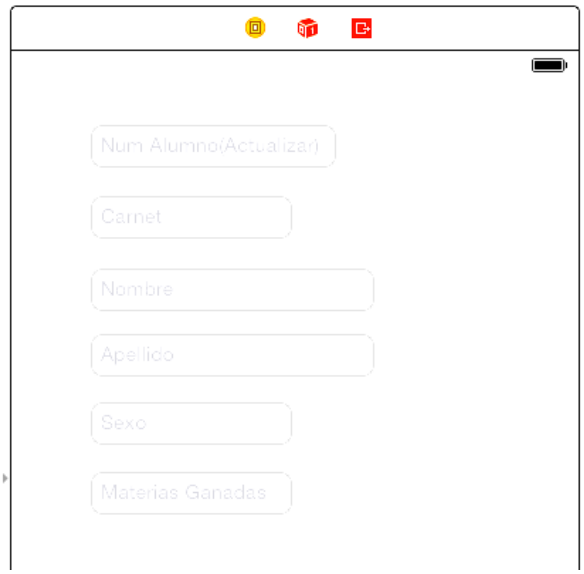
AlumnoViewController.

Para realizar la configuración del objeto AlumnoViewController, seleccionar el archivo Main.storyboard y seleccionar el ViewController correspondiente, se agregaran 6 objetos TextFields, que se organizaran de la siguiente manera mostrada.



Para mostrar una marca de agua especificando el contenido de cada TextField, ubicarse en la pestaña de “Attributes Inspector” ( ), ubicada en la barra de utilidades, esta presentara una opcion con el nombre de “Placeholder”, modificar cada atributo correspondiente a cada TextField, para que estos se muestren de la manera siguiente.



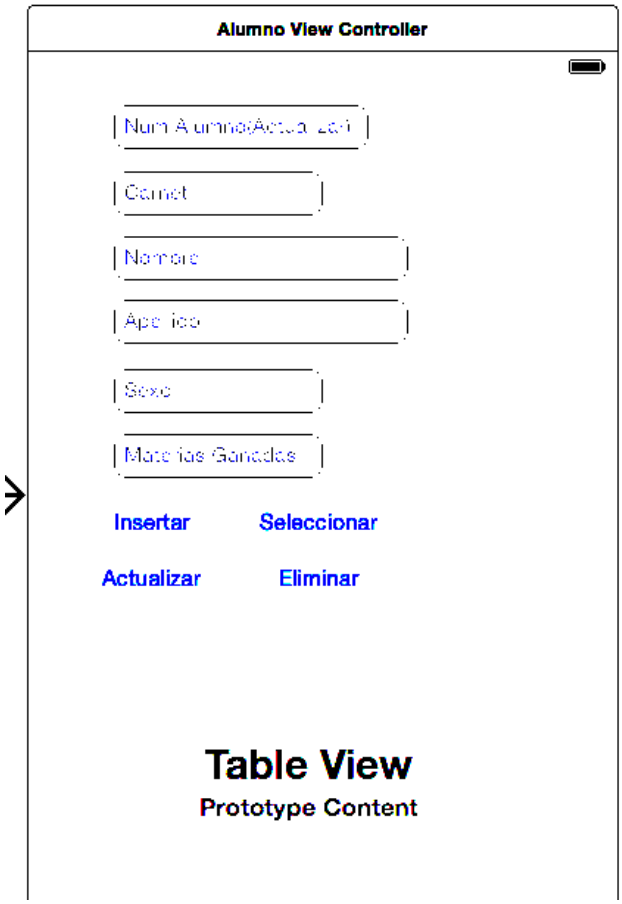


De igual manera agregar 4 objetos de tipo “Button” y cambiarles el texto como se muestra a continuacion.

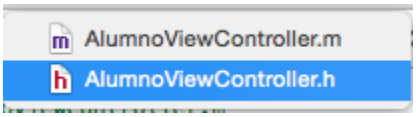
Alinearlos y modificar los TextFields si fuese necesario para que el AlumnoViewController se observe de la manera que se muestra a continuacion.

Para la presentación de los datos se incorporara un objeto de tipo “TableView”.

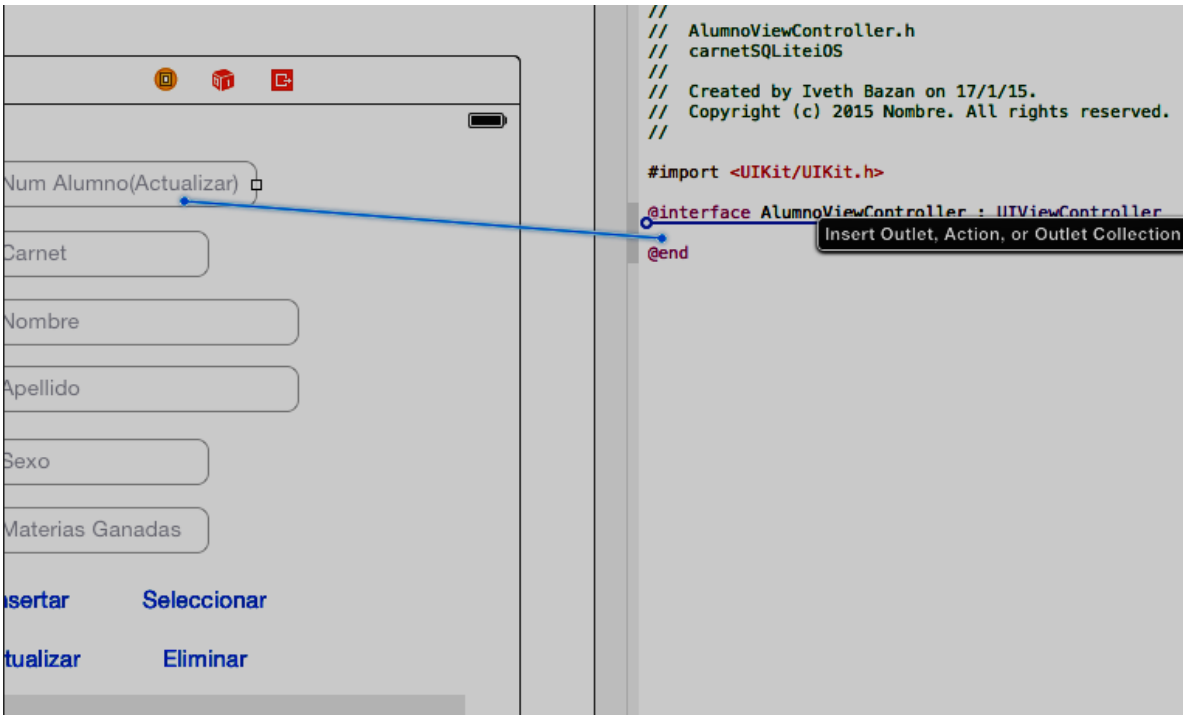
Colocar el objeto “TableView” en la parte inferior del ViewController como se muestra a continuación.



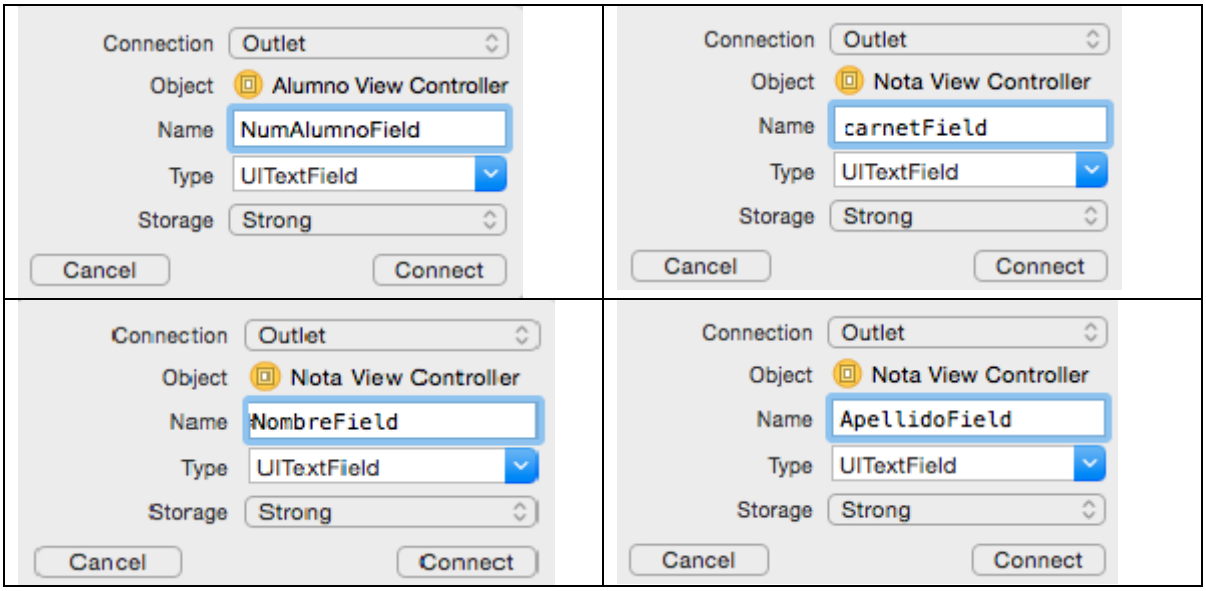
Ahora para enlazar los objetos con el archivo AlumnoViewController.h, accedemos a la ventana con asistente  , para poder visualizar tanto el Main.storyboard y el archivo de AlumnoViewController.h.

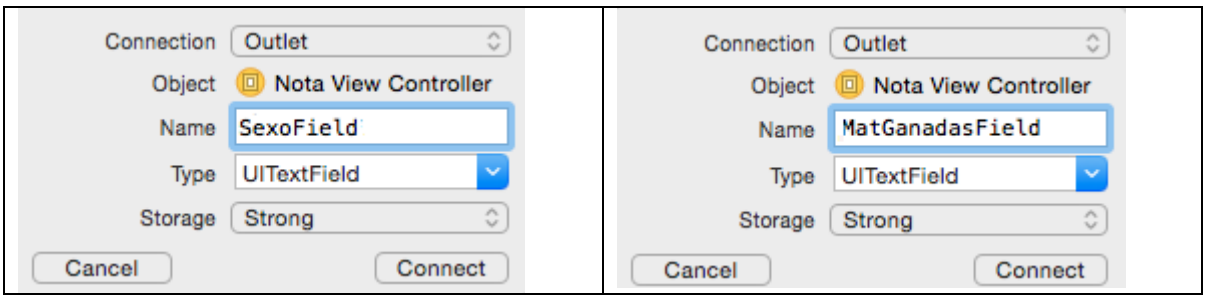


Para crear la conexión, hacer clic derecho desde el TextField deseado y arrastrar hacia el archivo .h, como se muestra a continuacion.



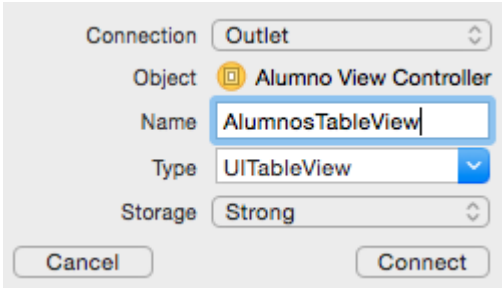
Al soltar la conexión, se presentara una ventana emergente, en la cual se especificara el tipo de conexión a establecer, en cada una de ellas que se realice con los TextFields, se tomara el tipo de conexión “Outlet” y el tipo de almacenamiento (”Storage”) de tipo “Strong”. Se muestra a continuacion todas las ventanas correspondientes a cada uno de los TextFields.



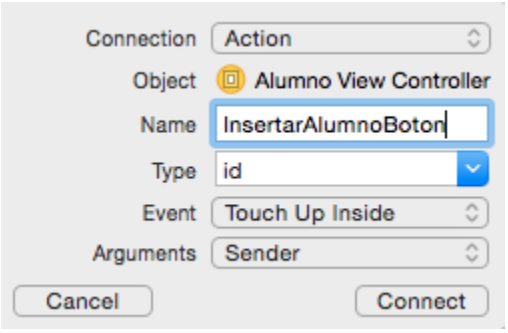


También se realizará la conexión con el objeto TableView.

Al presentarse la ventana emergente se cofigura como se muestra en la siguiente imagen.



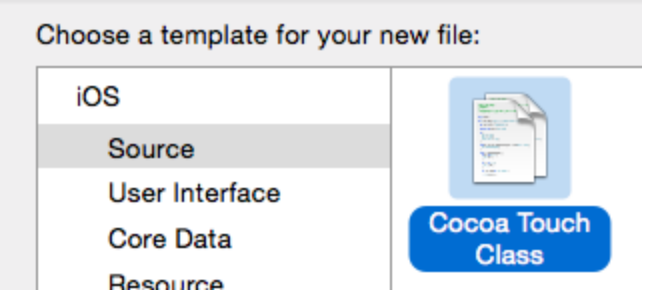
De igual manera se establecen las conexiones para los botones correspondientes, pero para estos objetos se creara una conexión de tipo “Action”, todas las conexiones se establecerán de igual manera con los diferentes nombres. InsertarAlumnoBoton, ConsultarAlumnoBoton, ActualizarAlumnoBoton y EliminarAlumnoBoton correspondiente.



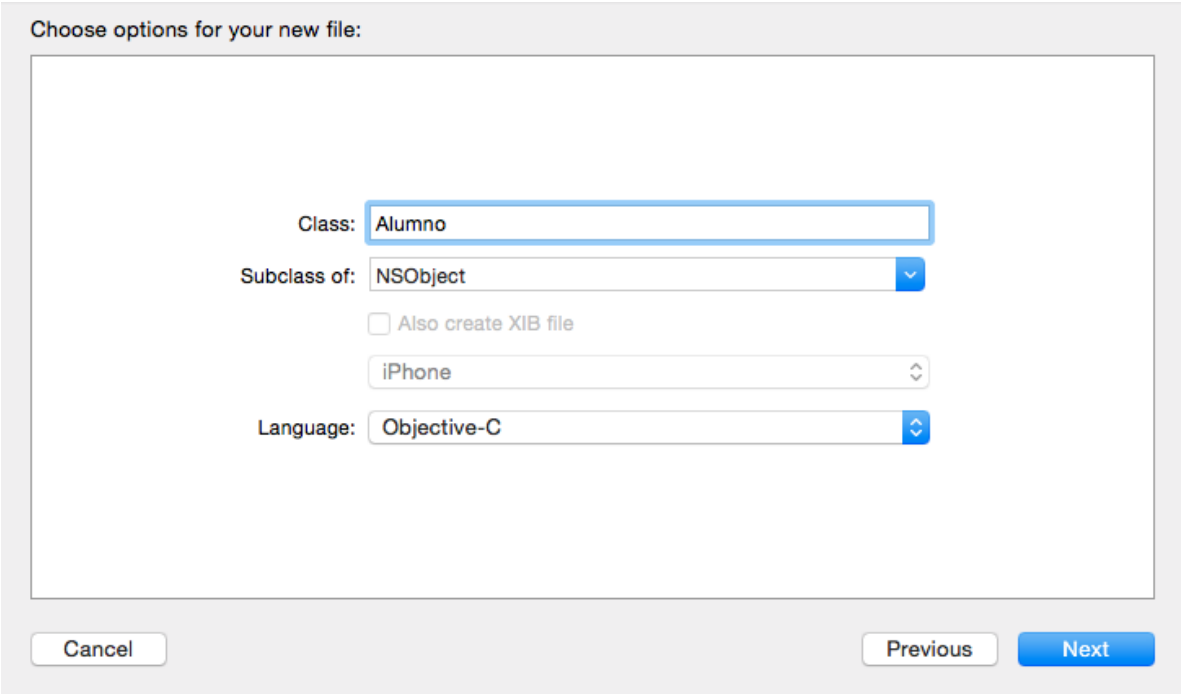
Al finalizar todas las conexiones el archivo AlumnoViewController.h deberá observarse de la siguiente manera.

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

@interface AlumnoViewController : UIViewController
@property (strong, nonatomic) IBOutlet UITextField *NumAlumnoField;
@property (strong, nonatomic) IBOutlet UITextField *carnetField;
@property (strong, nonatomic) IBOutlet UITextField *NombreField;
@property (strong, nonatomic) IBOutlet UITextField *ApellidoField;
@property (strong, nonatomic) IBOutlet UITextField *SexoField;
@property (strong, nonatomic) IBOutlet UITextField *MatGanadasField;
@property (strong, nonatomic) IBOutlet UITableView *AlumnosTableView;
- (IBAction)InsertarAlumnoBoton:(id)sender;
- (IBAction)ConsultarAlumnoBoton:(id)sender;
- (IBAction)ActualizarAlumnoBoton:(id)sender;
- (IBAction)EliminarAlumnoBoton:(id)sender;

@end
```

Para gestionar los atributos que comprenden al tipo de dato Alumno, se creara una clase de tipo Cocoa Touch Class.



Crear una clase con subclase de tipo NSObjetc y colocarle el nombre de Alumno



En esta clase ingresar el siguiente código, en el cual se crear los atributos correspondientes al alumno, el cual consta de 4 atributos de tipo NSString y 2 atributos de tipo int.

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

@interface Alumno : NSObject

@property (assign) int numAlumno;
@property (nonatomic, strong)NSString *carnet;
@property (nonatomic, strong)NSString *nombre;
@property (nonatomic, strong)NSString *apellido;
@property (nonatomic, strong)NSString *sexo;
@property (assign) int matganadas;
@end
```

Se modificara el archivo AlumnoViewController.h con los siguientes cambios, estos presentan la importación de diferentes librerías a utilizar y la especificación del uso de una fuente de datos del TableView modificado personalmente, como también los métodos abstractos de esta subclase son heredados para ser modificados individualmente, para la necesidad del usuario.

```
#import <UIKit/UIKit.h>
#import <sqlite3.h>
#import "Alumno.h"

@interface AlumnoViewController : UIViewController<UITableViewDataSource,UITableViewDelegate>
@property (strong, nonatomic) IBOutlet UITextField *NumAlumnoField;
@property (strong, nonatomic) IBOutlet UITextField *carnetField;
@property (strong, nonatomic) IBOutlet UITextField *NombreField;
@property (strong, nonatomic) IBOutlet UITextField *ApellidoField;
@property (strong, nonatomic) IBOutlet UITextField *SexoField;
@property (strong, nonatomic) IBOutlet UITextField *MatGanadasField;
@property (strong, nonatomic) IBOutlet UITableView *AlumnosTableView;
- (IBAction)InsertarAlumnoBoton:(id)sender;
- (IBAction)ConsultarAlumnoBoton:(id)sender;
- (IBAction)ActualizarAlumnoBoton:(id)sender;
- (IBAction)EliminarAlumnoBoton:(id)sender;

@end
```

Acceder al archivo AlumnoViewController.m, en este se especificara todas las funcionalidades del manejo de datos para la tabla Alumno.

Como se observa se encuentran los métodos creados con los botones anteriormente, estos se modificaran posteriormente.

```
- (IBAction)InsertarAlumnoBoton:(id)sender {
}

- (IBAction)ConsultarAlumnoBoton:(id)sender {
}

- (IBAction)ActualizarAlumnoBoton:(id)sender {
}
- (IBAction)EliminarAlumnoBoton:(id)sender {
}
```

En el archivo .m se agregara el siguiente texto, el cual corresponde a un arreglo para controlar los objetos tipo Alumno que se controlaran, el objeto de tipo sqlite, el cual permite establecer las conexiones y utilizar la librería controladora y finalmente un NSString el cual permite establecer la dirección en la cual se guardara la base de datos.

```
@interface AlumnoViewController ()
{
    NSMutableArray *arrayAlumno;
    sqlite3 *alumnoDB;
    NSString *dbPathString;
}
@end
```

El método viewDidLoad, será modificado con el siguiente código, en el cual se establece que el objeto AlumnoTableView, se trabajara de manera modificada en este archivo y se llama al método antes creado para acceder a la base de datos.

```
- (void)viewDidLoad {
    [super viewDidLoad];
    [[controlBD sharedInstance] crearOabrirDB];
    arrayAlumno =[[NSMutableArray alloc]init];
    [[self AlumnosTableView]setDelegate:self];
    [[self AlumnosTableView]setDataSource:self];


    // Do any additional setup after loading the view.
}
```

Se crean los métodos heredados a modificar del objeto TableView, en los cuales se especifican la cantidad de secciones en la tabla y la cantidad de filas en dicha sección, la cual esta especificada por el número de elementos en el arreglo antes creado.

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

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{
    return [arrayAlumno count];
}
```

Se agrega el siguiente método, el cual se encarga de especificar la presentación de las filas en el TableView, como se observa se incorpora a la fila el texto del TextField correspondiente a carnet, como también se incorpora el texto correspondiente a nombre y apellido, utilizando un método de los objetos NSString llamado “stringWithFormat”, el cual especifica que la cadena de caracteres está compuesta por 2 cadenas, las cuales se agregan posteriormente.

```
-(UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath{

    static NSString *CellIdentifier=@"Cell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (!cell) {
        cell=[[UITableViewCell alloc]initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:
            CellIdentifier];
    }
    Alumno *aAlumno=[arrayAlumno objectAtIndex:indexPath.row];
    cell.textLabel.text=aAlumno.carnet;
    cell.detailTextLabel.text=[NSString stringWithFormat:@"%@ %@",aAlumno.nombre, aAlumno.apellido] ;
    return cell;
}
```

Modificar el método InsertarAlumnoBoton, como se muestra a continuación, el cual inserta el alumno obteniendo los datos de cada TextField correspondiente.

```
- (IBAction)InsertarAlumnoBoton:(id)sender {
    char *error;
    if (sqlite3_open([[controlBD sharedInstance].dbPath UTF8String], &alumnoDB)==SQLITE_OK) {
        NSString *insert_Stmt=[NSString stringWithFormat:@"INSERT INTO ALUMNO
            (CARNET,NOMBRE,APELLIDO,SEXO,MATGANADAS) values ('%s ','%s','%s','%s','%d')",[self.carnetField.
            text UTF8String],[self.NombreField.text UTF8String],[self.ApellidoField.text UTF8String],[self.
            SexoField.text UTF8String],[self.MatGanadasField.text intValue]];
        const char *insert_stmt=[insert_Stmt UTF8String];

        if (sqlite3_exec(alumnoDB, insert_stmt, NULL, NULL, &error)==SQLITE_OK) {
            NSLog(@"Alumno Insertado correctamente");

            Alumno *alumno =[[Alumno alloc]init];
            [alumno setCarnet:self.carnetField.text];
            [alumno setNombre:self.NombreField.text];
            [alumno setApellido:self.ApellidoField.text];
            [alumno setSexo:self.SexoField.text];
            [alumno setMatganadas:[self.MatGanadasField.text intValue]];
            [arrayAlumno addObject:alumno];
        }
        else{
            NSLog(@"Alumno no Insertado");
        }
        sqlite3_close(alumnoDB);
    }


}
```

Modificar el método ConsultarAlumnoBoton con el código que se presenta a continuación, este código genera una consulta en la base de datos obteniendo todas las filas encontradas en la tabla Alumno y obteniendo información de dichos elementos, los ingresa al arreglo y este genera una fila en el objeto TableView.

```
- (IBAction)ConsultarAlumnoBoton:(id)sender {
    sqlite3_stmt *statement;
    if (sqlite3_open([[controlBD sharedInstance].dbPath UTF8String], &alumnoDB)==SQLITE_OK) {
        [arrayAlumno removeAllObjects];
        NSString *querySql =[NSString stringWithFormat:@"SELECT * FROM ALUMNO"];
        const char *querysql=[querySql UTF8String];

        if (sqlite3_prepare(alumnoDB, querysql,-1,&statement,NULL)==SQLITE_OK) {
            while (sqlite3_step(statement)==SQLITE_ROW) {
                NSString *numalumno1=    [[NSString alloc]initWithUTF8String:(const char *)
                    sqlite3_column_text(statement, 0)];
                NSString *carnet1=    [[NSString alloc]initWithUTF8String:(const char *)sqlite3_column_text
                    (statement, 1)];
                NSString *nombre1=    [[NSString alloc]initWithUTF8String:(const char *)sqlite3_column_text
                    (statement, 2)];
                NSString *apellido1=  [[NSString alloc]initWithUTF8String:(const char *)sqlite3_column_text
                    (statement, 3)];
                NSString *sexo1=      [[NSString alloc]initWithUTF8String:(const char *)sqlite3_column_text
                    (statement,4)];
                NSString *matganadas1=[[NSString alloc]initWithUTF8String:(const char *)sqlite3_column_text
                    (statement, 5)];

                Alumno *alumno=[[Alumno alloc]init];
                [alumno setNumAlumno:[numalumno1 intValue]];
                [alumno setCarnet:carnet1];
                [alumno setNombre:nombre1];
                [alumno setApellido:apellido1];
                [alumno setSexo:sexo1];
                [alumno setMatganadas:[matganadas1 intValue]];
                [arrayAlumno addObject:alumno];

            }
        }
        else{
            NSLog(@"Lista vacia");
        }
        sqlite3_close(alumnoDB);
    }

    [[self AlumnosTableView]reloadData];

}
```

Para el método actualizar se realizan los siguientes cambios, se muestran a continuación, se observa que se realizan cambios en el registro exceptuando numalumno y carnet, ya que estos son primarios.

```
- (IBAction)ActualizarAlumnoBoton:(id)sender {
    static sqlite3_stmt *statement=nil;
    if (sqlite3_open([[controlBD sharedInstance].dbPath UTF8String], &alumnoDB)==SQLITE_OK) {
        char *update_Stmt="UPDATE ALUMNO SET NOMBRE=?, APELLIDO=?, SEXO=?, MATGANADAS=? WHERE NUMALUMNO=? ";
        if (sqlite3_prepare_v2(alumnoDB, update_Stmt, -1, &statement, NULL)==SQLITE_OK){
            sqlite3_bind_text(statement,1,[self.NombreField.text UTF8String], -1, SQLITE_TRANSIENT);
            sqlite3_bind_text(statement,2,[self.ApellidoField.text UTF8String], -1, SQLITE_TRANSIENT);
            sqlite3_bind_text(statement,3,[self.SexoField.text UTF8String], -1, SQLITE_TRANSIENT);
            sqlite3_bind_int(statement, 4, [self.MatGanadasField.text intValue]);
            sqlite3_bind_int(statement, 5, [self.NumAlumnoField.text intValue]);
            sqlite3_step(statement);
            sqlite3_finalize(statement);
            Alumno *alumno =[[Alumno alloc]init];
            [alumno setMatganadas:[self.MatGanadasField.text intValue]];
            [alumno setCarnet:self.carnetField.text];
            [alumno setNombre:self.NombreField.text];
            [alumno setApellido:self.ApellidoField.text];
            [alumno setSexo:self.SexoField.text];
            [alumno setNumAlumno:[self.NumAlumnoField.text intValue]];
            [arrayAlumno addObject:alumno];
            NSLog(@"Alumno modificado");
        }
        else
        {
            NSLog(@"Alumno no modificado");

        }
        sqlite3_close(alumnoDB);
    }

}
```

El método EliminarAlumnoBoton, habilita la edición de todas las filas del elemento TableView.

```
- (IBAction)EliminarAlumnoBoton:(id)sender {
    [[self AlumnosTableView]setEditing:!self.AlumnosTableView.editing animated:YES];
}
```

Se agrega el siguiente método, con el cual se pretende realizar la eliminación correspondiente en la base de datos, cuando la eliminación en el TableView se lleve a cabo.

```
-(void)deleteData:(NSString *)deleteQuery{
    char *error;
    if (sqlite3_open([[controlBD sharedInstance].dbPath UTF8String], &alumnoDB)==SQLITE_OK) {

        if (sqlite3_exec(alumnoDB, [deleteQuery UTF8String],NULL, NULL, &error)==SQLITE_OK) {
            NSLog(@"Alumno Eliminado");
        }
        else
        {
            NSLog(@"Alumno no Eliminado");

        }
        sqlite3_close(alumnoDB);

    }
}
```

Se agrega el siguiente método, para especificar el momento que se ejecuta la edición con éxito y así realizar la eliminación directa en la base de datos, como también la actualización del arreglo y del objeto TableView.

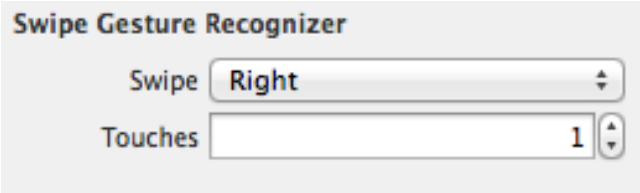
```
-(void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
    forRowAtIndexPath:(NSIndexPath *)indexPath{
    if (editingStyle==UITableViewCellEditingStyleDelete){
        Alumno *alum =[arrayAlumno objectAtIndex:indexPath.row];
        [self deleteData:[NSString stringWithFormat:@"DELETE FROM ALUMNO WHERE CARNET IS '%s'",[alum.carnet
            UTF8String]]];
        [arrayAlumno removeObjectAtIndex:indexPath.row];
        [tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath] withRowAnimation:
            UITableViewRowAnimationFade];

    }
}
```

El siguiente método se agrega para permitir la perdida de foto de cada TextField y así ocultar de manera dinámica el teclado.

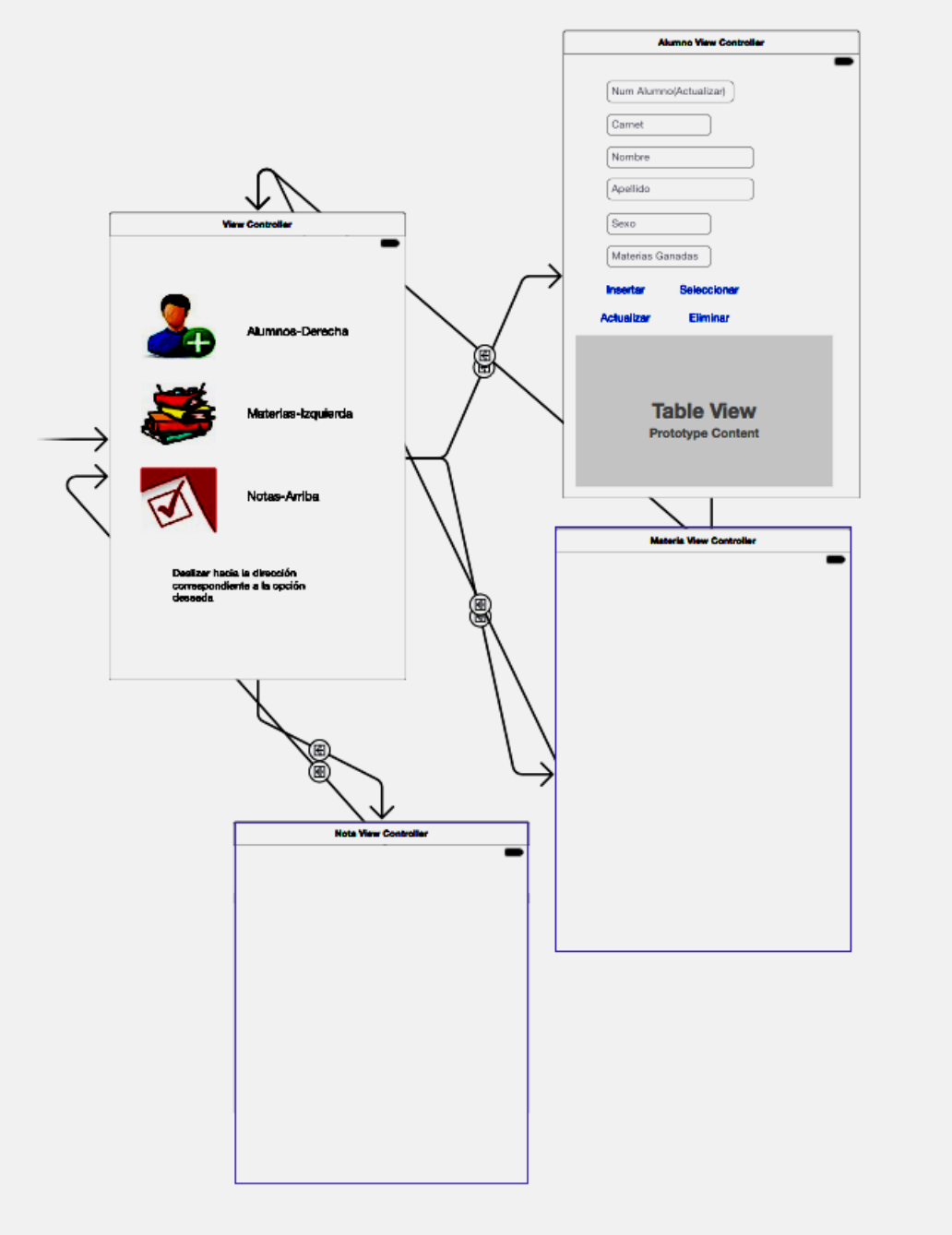
```
-(void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    [super touchesBegan:touches withEvent:event];
    [[self NumAlumnoField]resignFirstResponder];
    [[self carnetField]resignFirstResponder];
    [[self NombreField]resignFirstResponder];
    [[self ApellidoField]resignFirstResponder];
    [[self SexoField]resignFirstResponder];
    [[self MatGanadasField]resignFirstResponder];

}
```

Para realizar la navegacion completa se incorporara un “Swipe Gesture Recognizer” en cada uno de los demás ViewControllers, en este caso al elemento AlumnoViewController.

Se configurara de la siguiente manera como anteriormente se explicó, para que el deslice del evento sea hacia la derecha.

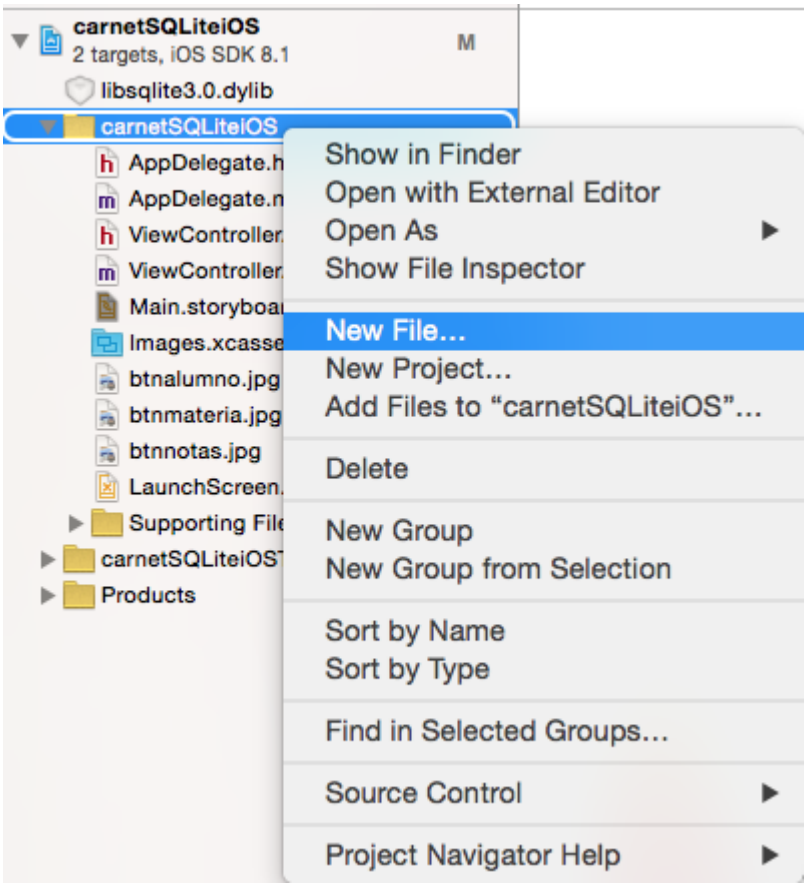


Realizar esto para todos los demás elementos ViewControllers, obteniendo así el Main.storyboard, con una navegación completa, este se observara de la manera siguiente mostrada.

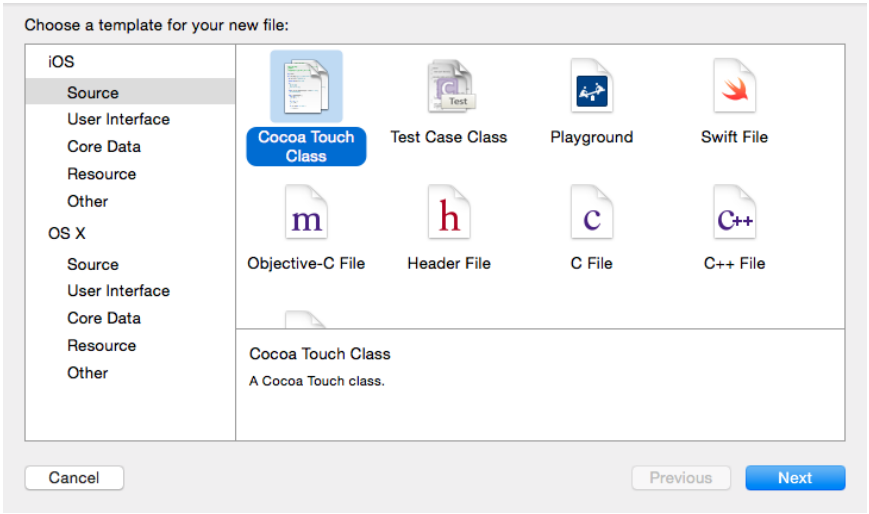


NotaViewController

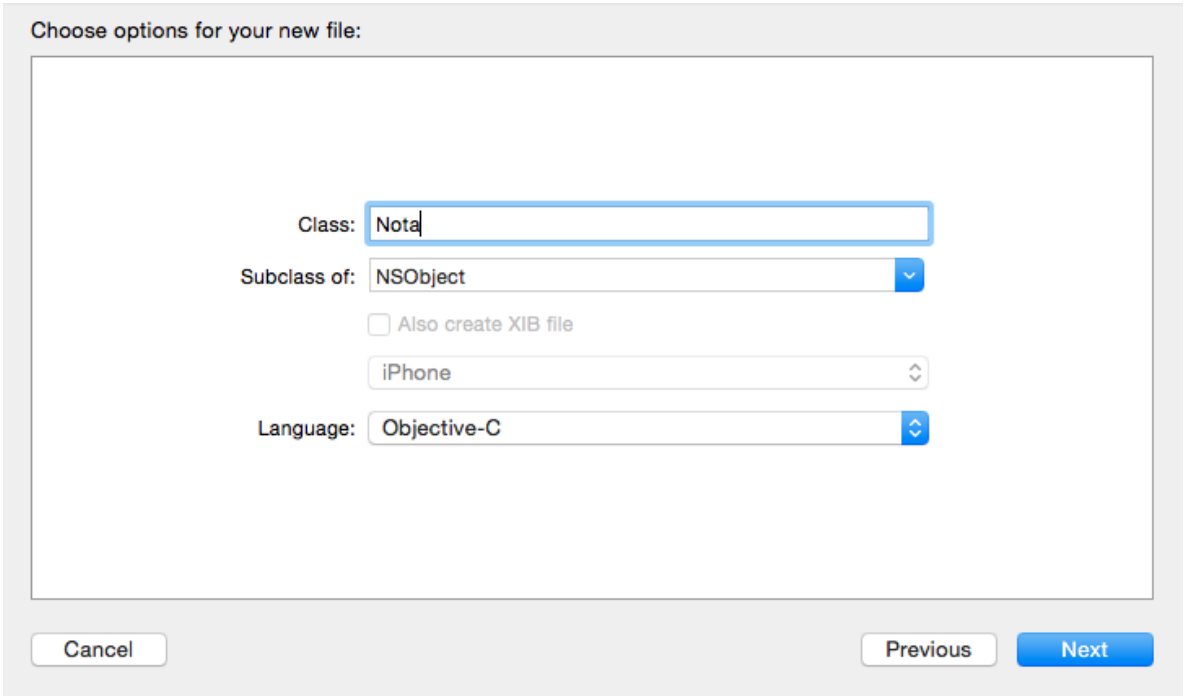
Para continuar con el siguiente ViewController, y crear todos los controladores para la clase Nota, primeramente se crean los archivos de clase.



Al seleccionar agregar nuevo archivo, se presentara una pantalla en la cual se debe de seleccionar el tipo de archivo que se desea crear, este archivo será de tipo "Objective-C class"



Esta clase a crear tendrá como subclase la clase NSObjetc, y el nombre de la clase será Nota, como se muestra a continuación.



Al crear los archivos en la barra de navegación de la izquierda se podrán observar los dos archivos creados, se muestra a continuación.

Primeramente acceder al archivo Nota.h y este modificarlo como se muestra a continuación, modificado este código ya se tendrán todos los atributos de la clase Nota.

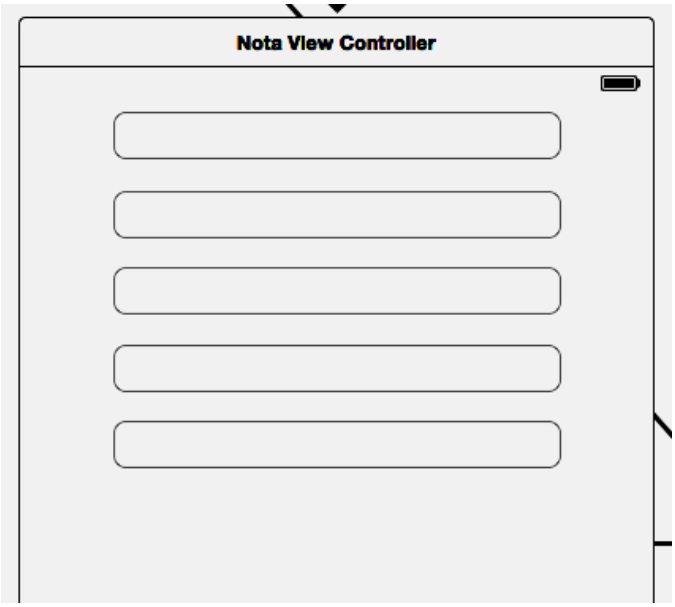
```
@interface Nota : NSObject

@property (assign) int NumNota;
@property (nonatomic, strong) NSString *CodMateria;
@property (nonatomic, strong) NSString *Carnet;
@property (nonatomic, strong) NSString *Ciclo;
@property (assign) float NotaFinal;

@end
```

Acceder al archivo NotaViewController.h para poder utilizar los métodos para la gestión sql y hacer uso de la clase antes creada de Nota, se importaran ambos archivos .h, como también se heredara de las super clases de “TableView”, para lo cual se coloca junto a UIViewController, las clases a heredar. El archivo NotaViewController.h quedaría de la manera siguiente.

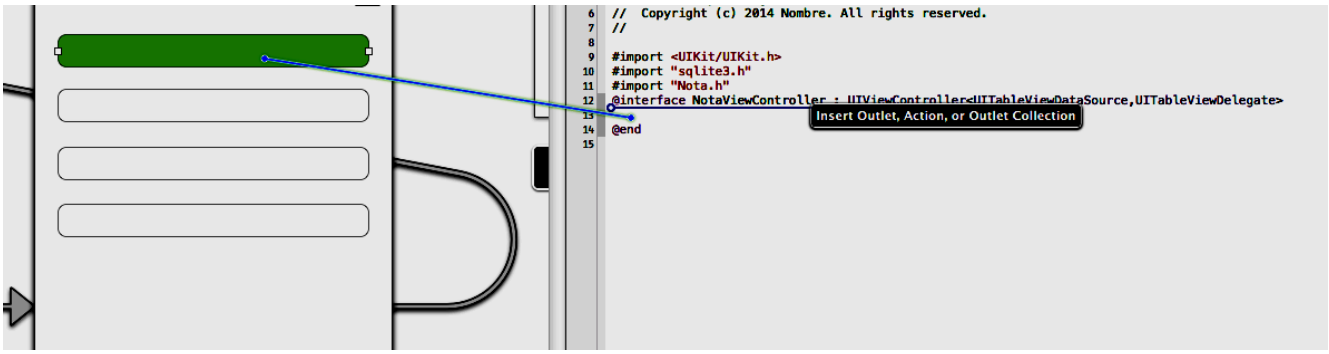
```
#import <UIKit/UIKit.h>
#import <sqlite3.h>
#import "Nota.h"

@interface NotaViewController : UIViewController <UITableViewDataSource, UITableViewDelegate>


@end
```

Para modificar el ambiente grafico, se debe de seleccionar el ViewController correspondiente, y a este se debe de agregar objetos de tipo TextField.

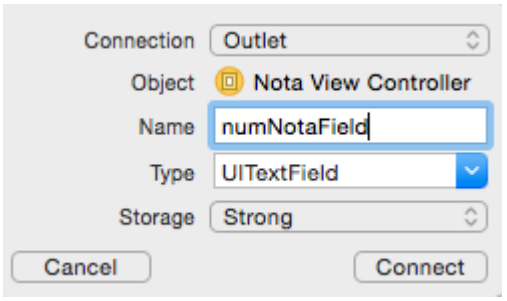
Agregar 5 objetos antes mostrados y ordenarlos como se muestra a continuación.

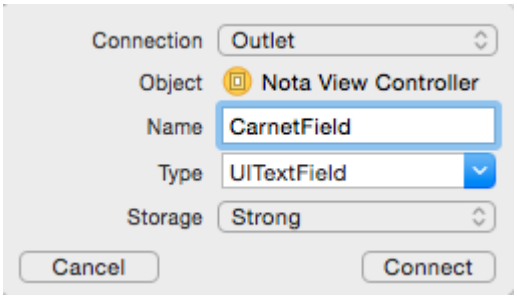


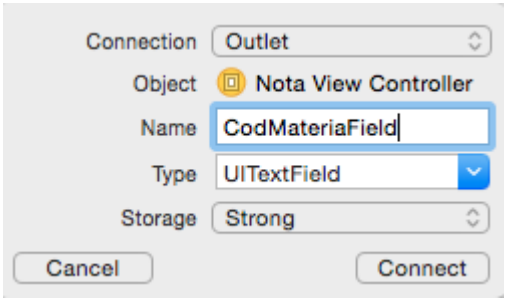
Utilizar el método de arrastre para realizar la conexión de cada uno de los objetos agregados con su respectiva declaración en el archivo “NotaViewController.h”, esto se realiza utilizando la vista de asistente, arrastrando con clic derecho desde el objeto con el que se desea realizar la conexión hasta el archivo .h correspondiente.

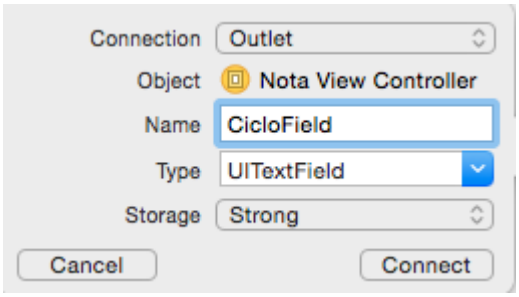


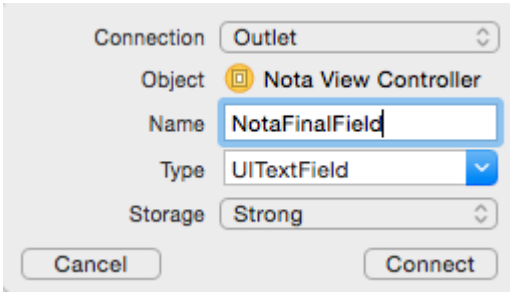
Al soltar en el archivo .h correspondiente, se presenta una ventana emergente, esta sirve para poner nombre a la conexión y seleccionar que tipo de conexión sera, a continuacion se presentan la descripcion de las conexiones correspondientes a cada uno de los TextField agregados.



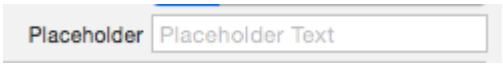




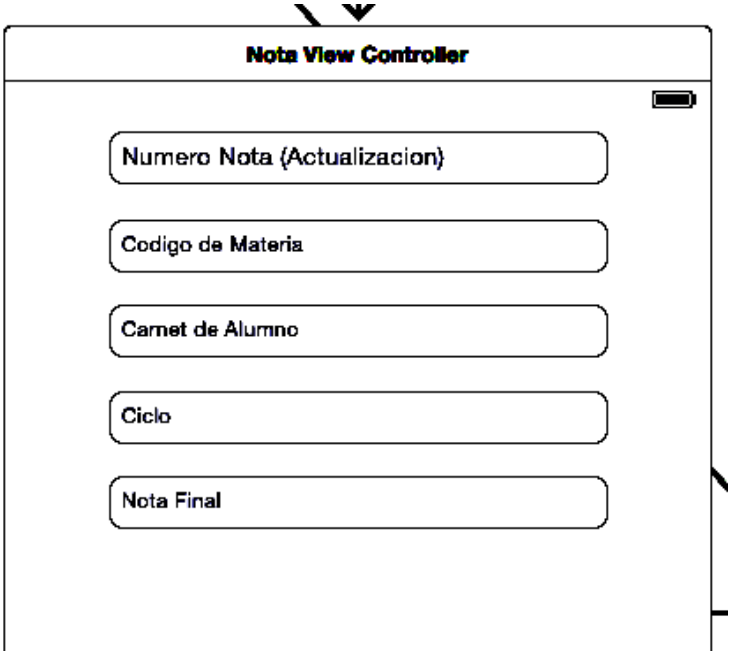




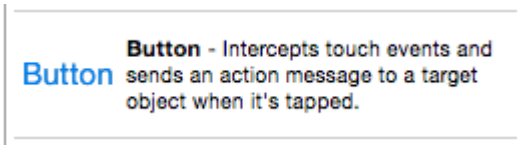
Cuando se realizan las conexión se procederá a modificar el texto descriptivo presentado por cada uno de los TextFields, seleccionar el objeto que se desea modificar, acceder a la pestaña de atributos () en el lado derecho de la pantalla, en la primera sección se observara el atributo llamado ‘’Placeholder”, el cual agregara un texto descriptivo al objetoseleccionado.



Al modificar todos los objetos, la pantalla se debe de observar como se muestra a continuación.



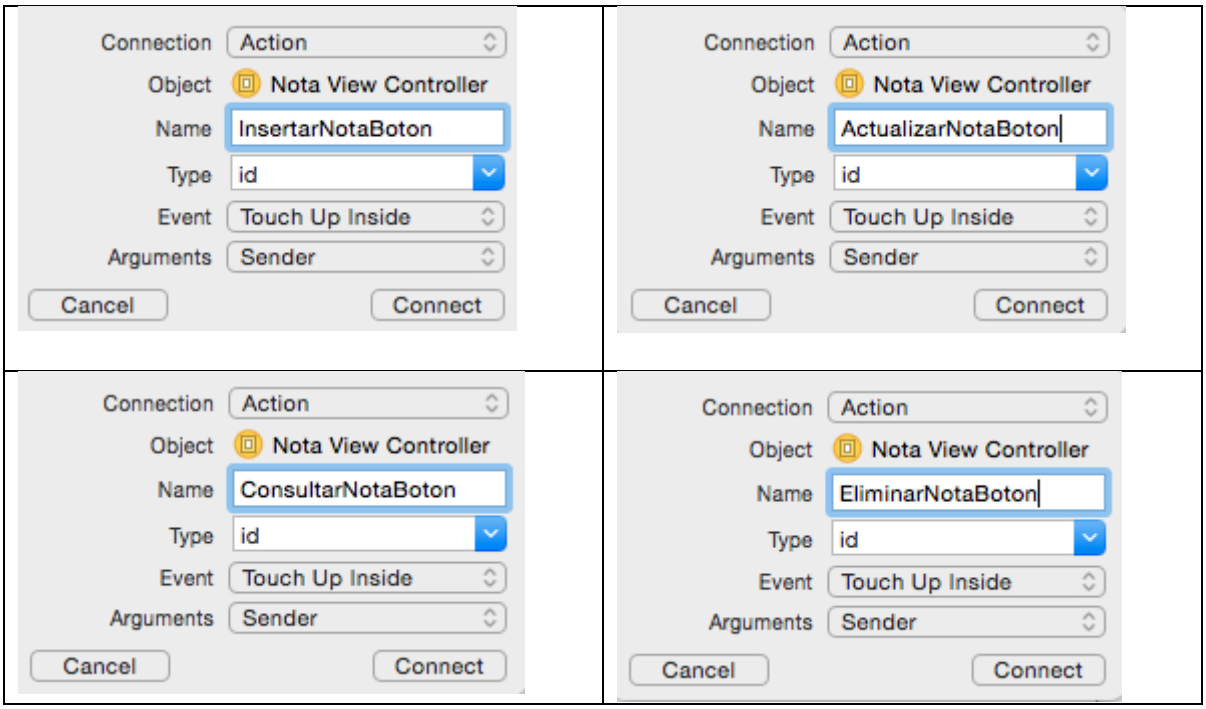
A continuación se debe de agregar los objetos de tipo Button.



Agregar 4 objetos de tipo Button, y modificar el texto de cada uno de ellos, para que estos se observen como se muestra en la imagen siguiente.

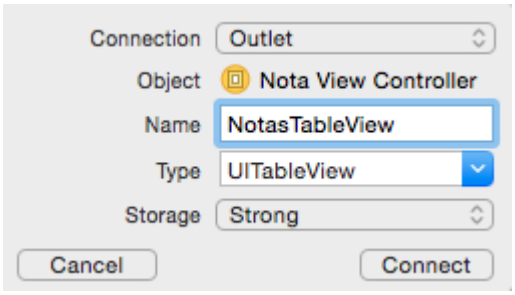
Para establecer la conexión de los botones antes creados, se debe de realizar la misma metodología que con los objetos anteriores.

Las conexiones se crean como se muestra a continuación, cada una corresponde a cada uno de los botones creados.



Para mostrar la informacion se utilizara un objeto de tipo TableView.

Agregar este objeto a la parte inferior del ViewController y realizar a conexión de igual manera que se hizo con los TextFields, a continuación se muestra la configuración a utilizar para dicha conexión.



Al finalizar todas las conexiones el archivo “NotaViewController.h” debe de quedar como se muestra a continuación.

```
#import <UIKit/UIKit.h>
#import <sqlite3.h>
#import "Nota.h"

@interface NotaViewController : UIViewController <UITableViewDataSource, UITableViewDelegate>
@property (strong, nonatomic) IBOutlet UITextField *numNotaField;
@property (strong, nonatomic) IBOutlet UITextField *CarnetField;
@property (strong, nonatomic) IBOutlet UITextField *CodMateriaField;
@property (strong, nonatomic) IBOutlet UITextField *CicloField;
@property (strong, nonatomic) IBOutlet UITextField *NotaFinalField;

- (IBAction)InsertarNotaBoton:(id)sender;
- (IBAction)ConsultarNotaBoton:(id)sender;
- (IBAction)ActualizarNotaBoton:(id)sender;
- (IBAction)EliminarNotaBoton:(id)sender;

@property (strong, nonatomic) IBOutlet UITableView *NotasTableView;

@end
```

Ya establecido el archivo .h, proceder a modificar el archivo .m, modificar la cabecera del archivo agregando las siguientes llaves y su contenido interno, se establece el arreglo en el cual se guardaran las notas, como también el archivo sqlite y la dirección de dicho archivo.

```
@interface NotaViewController ()
{
    NSMutableArray *arrayNota;
    sqlite3 *alumnoDB;
    NSString *dbPathString;
}
@end
```

Agregar el código correspondiente para el botón InsertarNotaBoton, como se muestra a continuación. Se observa una sección del código con comentario, este código habilita las foreign key, mientras permanezca la conexión abierta, este código se debe de escribir siempre que se desee que sqlite evalue las foreign key, estas automáticamente se deshabilitan.

```
- (IBAction)InsertarNotaBoton:(id)sender {
    char *error;
    if (sqlite3_open([[controlBD sharedInstance].dbPath UTF8String],  &alumnoDB)==SQLITE_OK) {
        NSString *insert_Stmt=[NSString stringWithFormat:@"INSERT INTO NOTA
            (CODMATERIA,CARNET,CICLO,NOTAFINAL) values ('%s','%s','%s','%f')",[self.CodMateriaField.text
            UTF8String],[self.CarnetField.text UTF8String],[self.CicloField.text UTF8String],[self.
            NotaFinalField.text floatValue]];
        const char *insert_stmt=[insert_Stmt UTF8String];

        if (sqlite3_exec(alumnoDB, insert_stmt, NULL, NULL, &error)==SQLITE_OK) {
            NSLog(@"Nota Insertada correctamente");

            Nota *nota =[[Nota alloc]init];
            [nota setNumNota:[self.numNotaField.text intValue]];
            [nota setCodMateria:self.CodMateriaField.text];
            [nota setCarnet:self.CarnetField.text];
            [nota setCiclo:self.CicloField.text];
            [nota setNotaFinal:[self.NotaFinalField.text floatValue]];
            [arrayNota addObject:nota];
        }
        else{
            NSLog(@"Nota no Insertada1");
        }
        sqlite3_close(alumnoDB);
    }

}
```

En el método viewDidLoad reservar memoria e inicializar el arreglo, y establecer el objeto NotasTableView como la tabla que heredara los métodos de las superclases.

```
- (void)viewDidLoad {
    [super viewDidLoad];
    [[controlBD sharedInstance] crearOabrirDB];
    arrayNota =[[NSMutableArray alloc]init];
    [[self NotasTableView]setDelegate:self];
    [[self NotasTableView]setDataSource:self];


    // Do any additional setup after loading the view.
}
```

Agregar los siguientes métodos que sobreescriben los métodos heredados. El primero especifica el número de secciones en cada fila, el segundo específica cuantas filas posee la tabla, esto se obtiene del número total de objetos en el arreglo de notas y el último crea la tabla en base al array de notas y establece el texto como también los detalles de la tabla.

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

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{
    return [arrayNota count];
}


-(UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath{

    static NSString *CellIdentifier=@"Cell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (!cell) {
        cell=[[UITableViewCell alloc]initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:
            CellIdentifier];
    }
    Nota *aNota=[arrayNota objectAtIndex:indexPath.row];
    cell.textLabel.text=[NSString stringWithFormat:@"%f", aNota.NotaFinal];
    cell.detailTextLabel.text=[NSString stringWithFormat:@"%@ %@",aNota.Carnet, aNota.CodMateria] ;
    return cell;
}
```

Modificar el método ConsultarNotaBoton, con el código que se muestra a continuación, este realiza una consulta en la base y agrega objetos al arreglo con los registros obtenidos, para agregar los datos a la tabla de notas.

```
- (IBAction)ConsultarNotaBoton:(id)sender {
    sqlite3_stmt *statement;
    if (sqlite3_open([[controlBD sharedInstance].dbPath UTF8String], &alumnoDB)==SQLITE_OK) {
        [arrayNota removeAllObjects];
        NSString *querySql =[NSString stringWithFormat:@"SELECT * FROM NOTA"];
        const char *querysql=[querySql UTF8String];

        if (sqlite3_prepare(alumnoDB, querysql,-1,&statement,NULL)==SQLITE_OK) {
            while (sqlite3_step(statement)==SQLITE_ROW) {
                NSString *numnota1=    [[NSString alloc]initWithUTF8String:(const char *)sqlite3_column_text
                    (statement, 0)];
                NSString *codmateria1=    [[NSString alloc]initWithUTF8String:(const char *)
                    sqlite3_column_text(statement, 1)];
                NSString *carnet1=    [[NSString alloc]initWithUTF8String:(const char *)sqlite3_column_text
                    (statement, 2)];
                NSString *ciclo1=  [[NSString alloc]initWithUTF8String:(const char *)sqlite3_column_text
                    (statement, 3)];
                NSString *notafinal1=  [[NSString alloc]initWithUTF8String:(const char *)sqlite3_column_text
                    (statement, 4)];

                Nota *nota=[[Nota alloc]init];
                [nota setNumNota:[numnota1 intValue]];
                [nota setCodMateria:codmateria1];
                [nota setCarnet:carnet1];
                [nota setCiclo:ciclo1];
                [nota setNotaFinal:[notafinal1 floatValue]];
                [arrayNota addObject:nota];
            }
        }
        else{
            NSLog(@"Lista vacia");
        }
        sqlite3_close(alumnoDB);
    }

    [[self NotasTableView]reloadData];

}
```

Modificar el método ActualizarNotaBoton, este utiliza el TextField numnotaField, para realizar la modificación de una nota específica, solo modifica el valor de la nota final, ninguno de los otros valores puede ser modificado.

```
- (IBAction)ActualizarNotaBoton:(id)sender {
    static sqlite3_stmt *statement=nil;
    if (sqlite3_open([[controlBD sharedInstance].dbPath UTF8String], &alumnoDB)==SQLITE_OK) {
        char *update_Stmt="UPDATE NOTA SET NOTAFINAL=? WHERE NUMNOTA=? ";
        if (sqlite3_prepare_v2(alumnoDB, update_Stmt, -1, &statement, NULL)==SQLITE_OK){
            sqlite3_bind_int(statement, 1, [self.NotaFinalField.text floatValue]);
            sqlite3_bind_int(statement, 2, [self.numNotaField.text intValue]);
            sqlite3_step(statement);
            sqlite3_finalize(statement);
            Nota *nota=[[Nota alloc]init];
            [nota setNumNota:[self.numNotaField.text intValue]];
            [nota setCodMateria:self.CodMateriaField.text];
            [nota setCarnet:self.CarnetField.text];
            [nota setCiclo:self.CicloField.text];
            [nota setNotaFinal:[self.NotaFinalField.text floatValue]];
            [arrayNota addObject:nota];

            NSLog(@"Nota modificado");
        }
        else
        {
            NSLog(@"Nota no modificado");

        }
        sqlite3_close(alumnoDB);
    }

}
```

Modificar el metodo EliminarNotaBoton, el cual habilita la edicion de la tabla. Agregar el metodo deleteData, el cual realiza la eliminacion en la base.

```
- (IBAction)EliminarNotaBoton:(id)sender {
    [[self NotasTableView]setEditing:!self.NotasTableView.editing animated:YES];
}


-(void)deleteData:(NSString *)deleteQuery{
    char *error;
    if (sqlite3_open([[controlBD sharedInstance].dbPath UTF8String], &alumnoDB)==SQLITE_OK) {

        if (sqlite3_exec(alumnoDB, [deleteQuery UTF8String],NULL, NULL, &error)==SQLITE_OK) {
            NSLog(@"Nota Eliminada");
        }
        else
        {
            NSLog(@"Nota no Eliminada");

        }
        sqlite3_close(alumnoDB);

    }
}
```

Agregar y modificar el método heredado como se muestra a continuación, este realiza la eliminación cuando se termina de modificar.

```
-(void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
    forRowAtIndexPath:(NSIndexPath *)indexPath{
    if (editingStyle==UITableViewCellEditingStyleDelete){
        Nota *not =[arrayNota objectAtIndex:indexPath.row];
        [self deleteData:[NSString stringWithFormat:@"DELETE FROM NOTA WHERE CARNET IS '%s' AND CODMATERIA
            IS '%s' AND CICLO IS '%s'",[not.Carnet UTF8String],[not.CodMateria UTF8String],[not.Ciclo
            UTF8String]]];
        [arrayNota removeObjectAtIndex:indexPath.row];
        [tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath] withRowAnimation:
            UITableViewRowAnimationFade];

    }
}
```

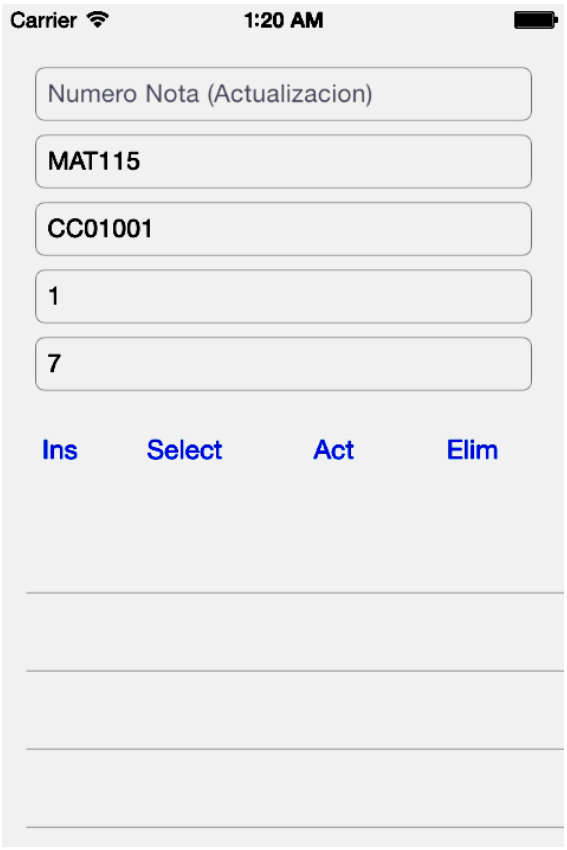
Establecer el siguiente método para que el teclado se oculte al seleccionar otro objeto.

```
-(void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    [super touchesBegan:touches withEvent:event];
    [[self CodMateriaField]resignFirstResponder];
    [[self CarnetField]resignFirstResponder];
    [[self CicloField]resignFirstResponder];
    [[self numNotaField]resignFirstResponder];
    [[self NotaFinalField]resignFirstResponder];

}
```

Pruebas

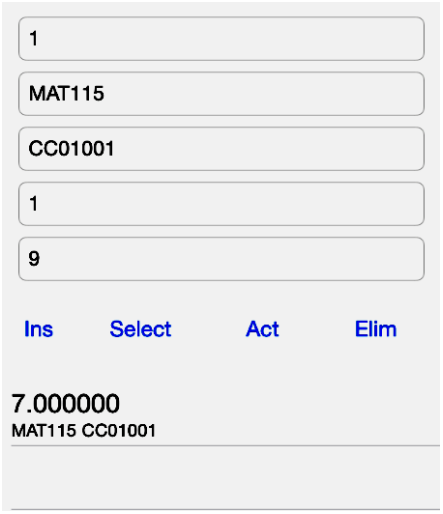
A continuación se muestran ejemplos de la como utilizar la aplicación.

a) Insertar: escriba los datos excepto el numero de nota, presione insert

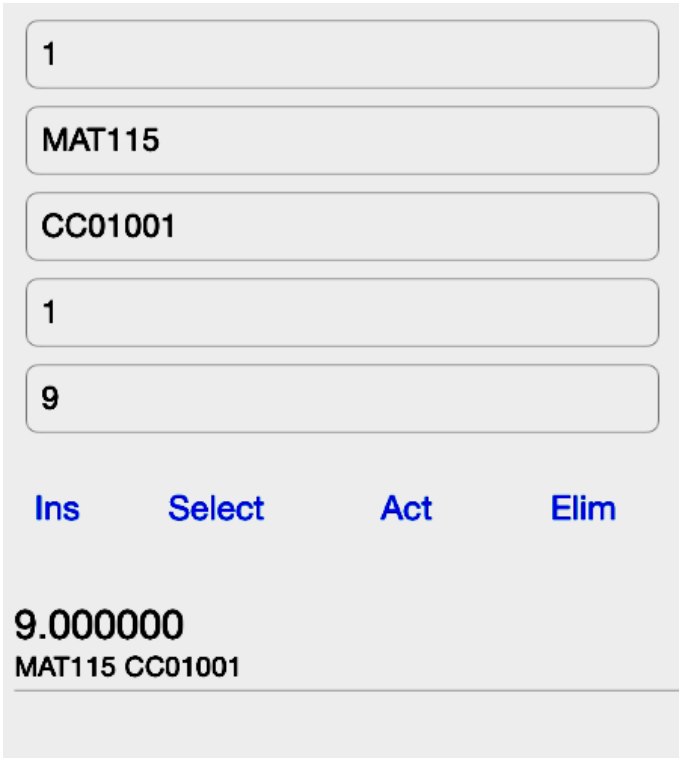


b) Seleccionar(consulta): Presione clic en Select para ver los datos en la Tableview(tabla inferior)

c) Actualizacion: Como el registro anterior tiene id =1(autogenerado), digitar los datos para ubicar el registro, y los cambios(nueva nota=9), presionar **Act**



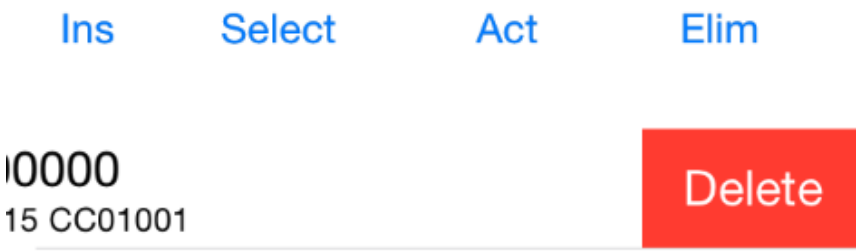
Luego **Select**



d) Eliminar:Primero se presiona **Elim**

Luego el signo menos del registro

Y clic en Delete



Finalice el mantenimiento para la tabla Materias y suba el archivo al link respectivo.