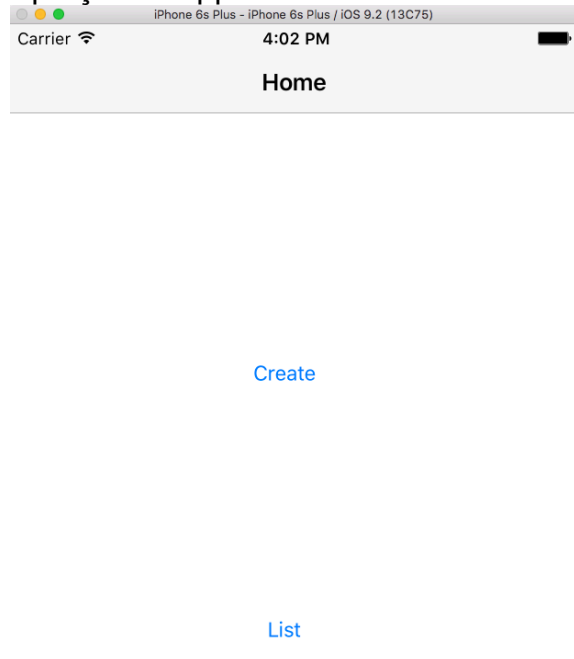


## TP2 GestionEleve

Cette application permet de créer et de lister des élèves (Student) et des étudiants (Undergraduate). Les élèves ont des caractéristiques propres, et les étudiants héritent des caractéristiques d'élève. Ce travail utilise l'héritage avec Core Data. Le stockage des données se fait sur le device par le biais d'une base SQLite. L'ORM CoreData est donc utilisé pour assurer la persistance des données.

Aperçu de l'application :



first Name

last Name

mail

phone

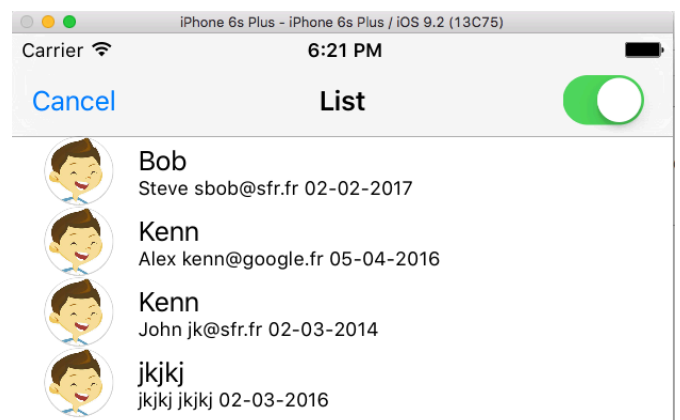
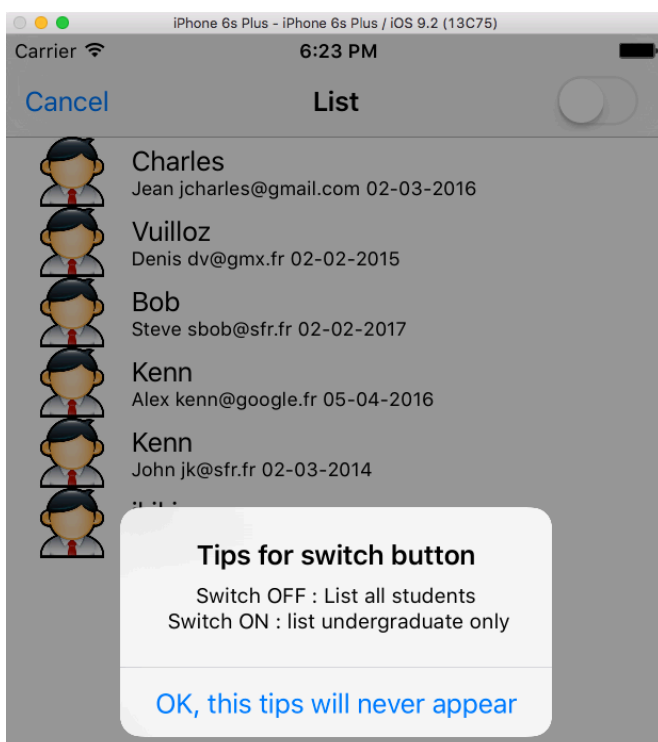
birthday

degree ☐

Scholarship ☐

serie

Mention



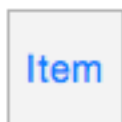
L'application utilise un Navigation Controller, afin d'avoir un menu de navigation en haut de l'écran.



## Partie 1 : le story board

Cette application **doit pouvoir supporter les différents formats d'écran.**

Elle permet d'utiliser tout un ensemble de composants utiles à une application de base :



**Bar Button Item** - Represents an item on a UIToolbar or UINavigationController object.

Button : bouton qui permet de créer un élève, étudiant ou annuler



**Navigation Item** - Represents a state of the navigation bar, including a title.

Item : permet d'afficher un bandeau de navigation au dessus de l'écran



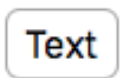
**Switch** - Displays an element showing the boolean state of a value. Allows tapping the control to toggle the value.

Switch : bouton qui permet de basculer d'un état à l'autre. Deux états possibles (élèves ou étudiants ici)



**Label** - A variably sized amount of static text.

Permet de placer un label



**Text Field** - Displays editable text and sends an action message to a target object when Return is tapped.

Zone de saisie d'un texte



**Date Picker** - Displays multiple rotating wheels to allow users to select dates and times.

Permet d'effectuer la saisie de la date de naissance d'un individu



**Picker View** - Displays a spinning-wheel or slot-machine motif of values.

Permet de faire la saisie de la série de l'étudiant ou de sa mention

**Button** - Intercepts touch events and sends an action message to a target object when it's tapped.

Button : bouton qui permet sélectionner la liste de classe

Le même Story Board peut être utilisé pour tous les devices qui utilisent iOS 9, sans écrire une seule ligne de code.

C'est l'intérêt de Universal Story Board.

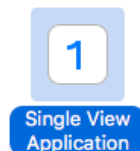
Vous allez devoir commencer par vous familiariser avec le story board pour générer les vues portraits et paysages (landscape) des appareils : iphone, ipad.

Tous les devices doivent pouvoir fonctionner en mode portrait et paysage.

iPhone 3.5-inch	iPhone 4S
iPhone 4-inch	iPhone 5, 5S
iPhone 4.7-inch	iPhone 6
iPhone 5.5-inch	iPhone 6 Plus
iPad	iPad 2, iPad Air (attention en Air = Retina x2) -> affichage point 7 du document

## 1. Setting the project

Pour créer le projet, la procédure habituelle est requise. Prendre un projet « single view application »



Penser à cocher Core Data :

Product Name:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Devices:

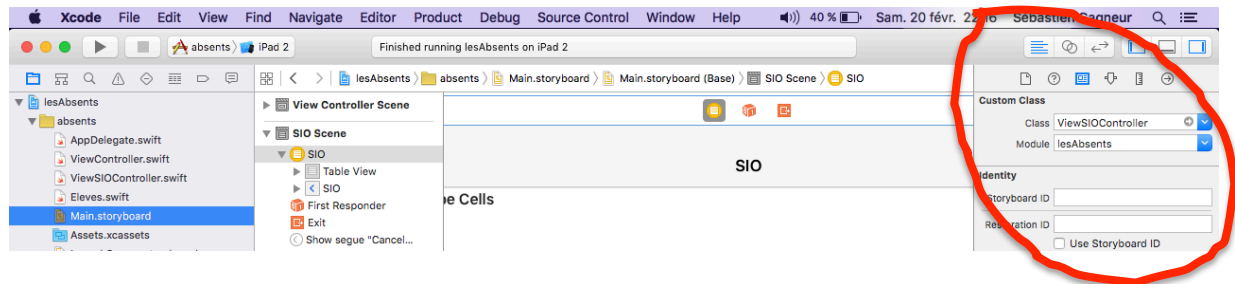
☒ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

## 2. Setting the story board

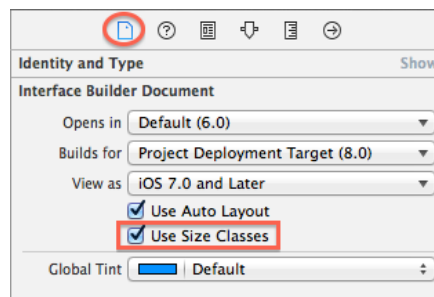
L'inspecteur est la fenêtre située à droite de Xcode :



Choisir le fichier du Main.storyboard, dans le projet, et aller dans le File Inspector :

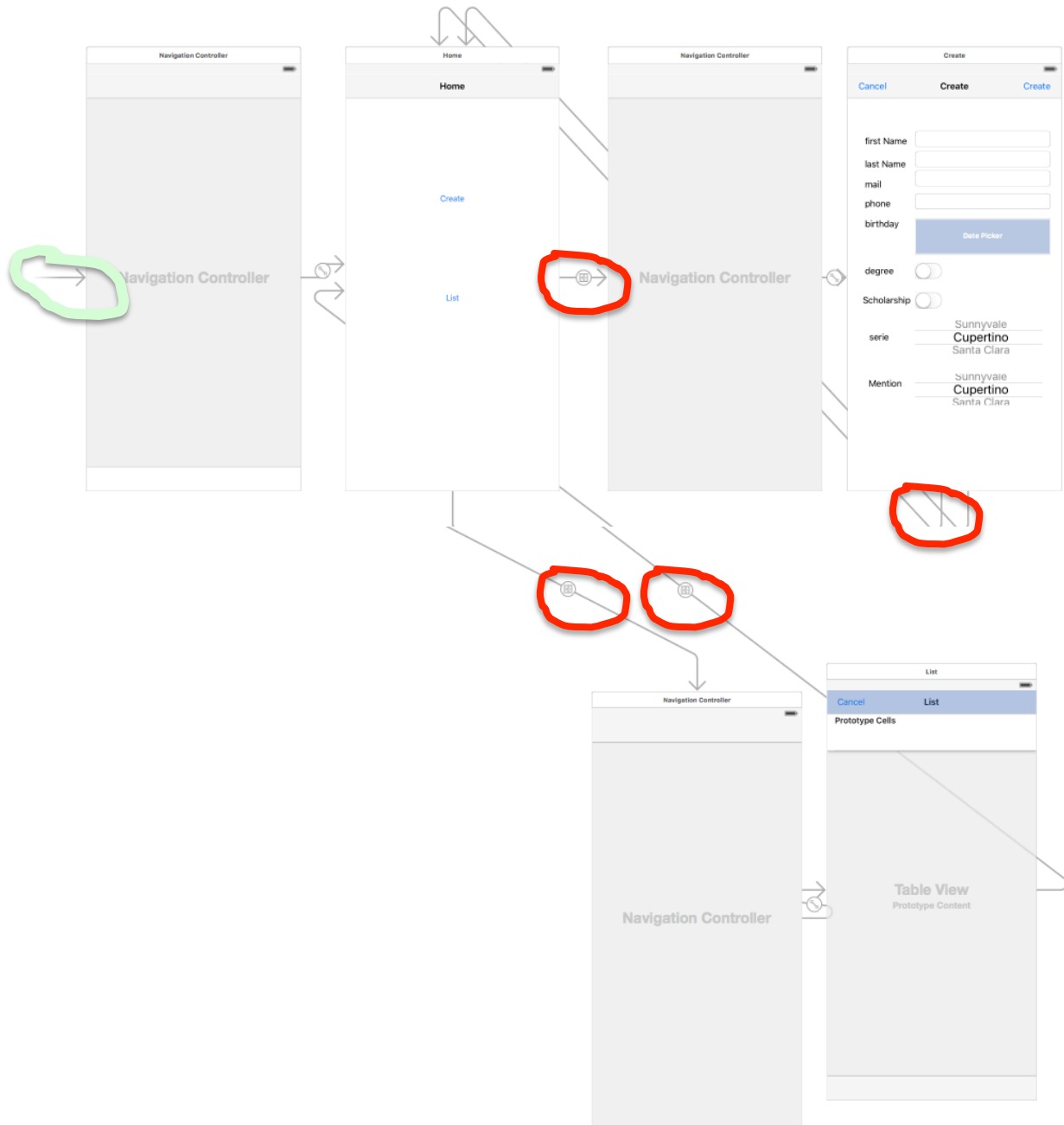


Il est nécessaire d'avoir activé, l'option suivante, qui en principe est activée par défaut.



### 3. Création du story board

Créer un Story Board qui sera toujours le même quelque soit le device iOS.



Le Story Board utilise dans une première fenêtre un navigation Controller (on aurait pu utiliser un ViewController), qui est la fenêtre d'accueil. La flèche à gauche constitue l'entry point (--->), qui désigne la première fenêtre à afficher.

Ici, l'association des Navigations Controller se fait avec un View Controller pour l'accueil (Home) et la saisie (Create)

Dans le cas de l'affichage de la liste (List), une table View est associée à un Navigation Controller.

De nombreux « segues » sont présents ils assurent la transition entre les différentes vues. Ils sont entourés en rouge sur le Story Board.

Pour créer les « segues » et régler la Table View, vous pouvez vous reporter au TP1 sur les absents qui utilisent les même principes.

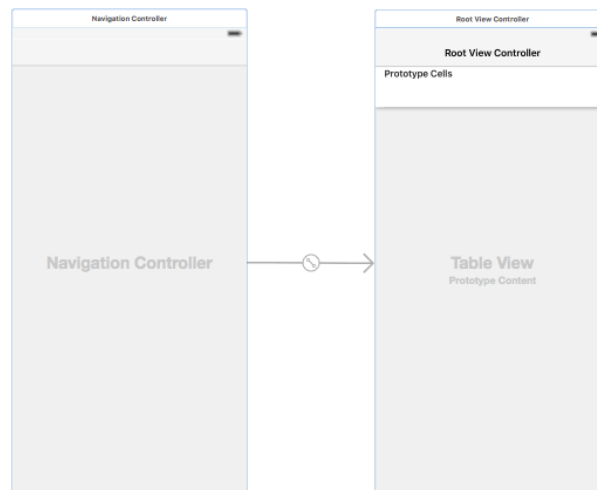
La seule petite nuance, et l'association Navigation Controller et View Controller. Celle-ci n'existe pas par défaut. Dans ce cas vous devez poser un navigation Controller sur le Story Board :



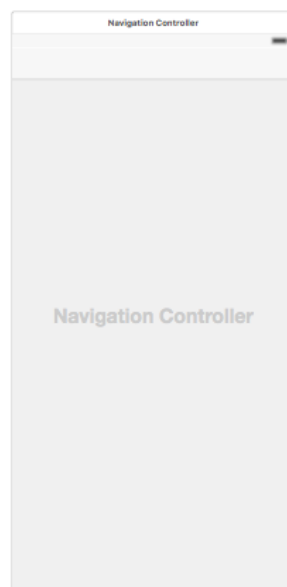
**Navigation Controller** - A controller that manages navigation through a hierarchy of views.

Permet de bénéficier d'une barre de navigation en haut de la vue

Vous devez ensuite supprimer la table View associée :



Le navigation Controller se retrouve seul :



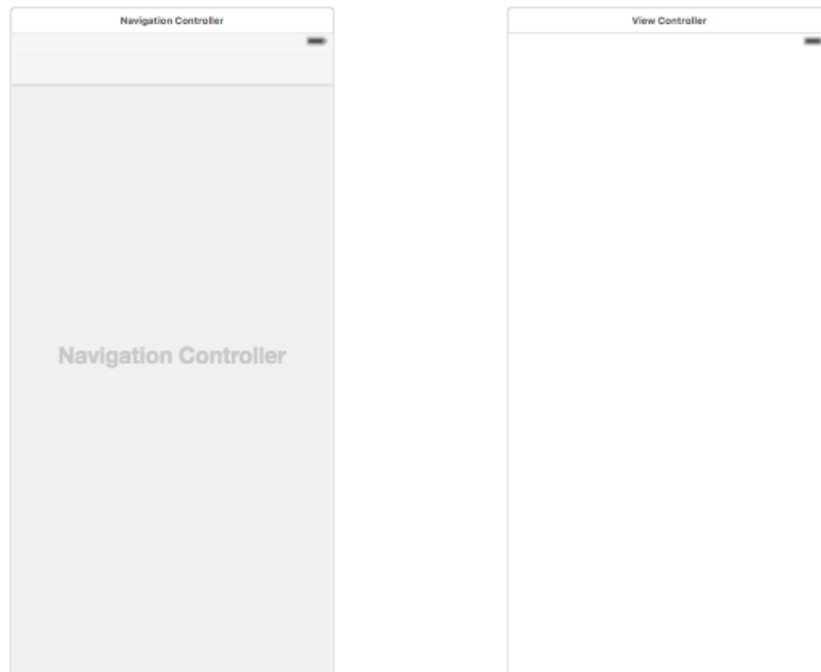
Prendre un View Controller et le relier au Navigation Controller :



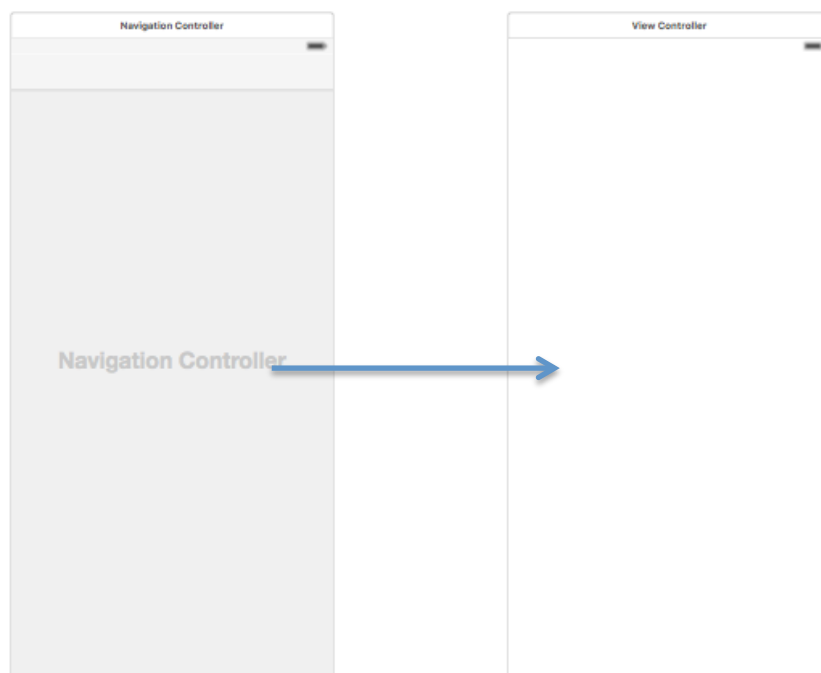
**View Controller** - A controller that manages a view.

Un View Controller permet de créer une vue vide sans barre de navigation placé en haut. Associé au Navigation Controller il permet d'avoir une barre en haut de la vue

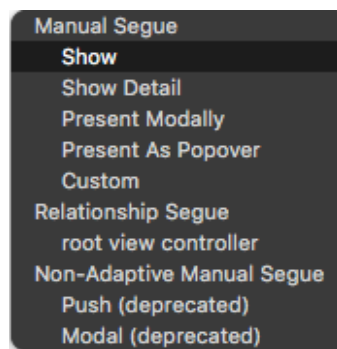
Le Navigation Controller et la vue ne sont pas liés avec un « segue »



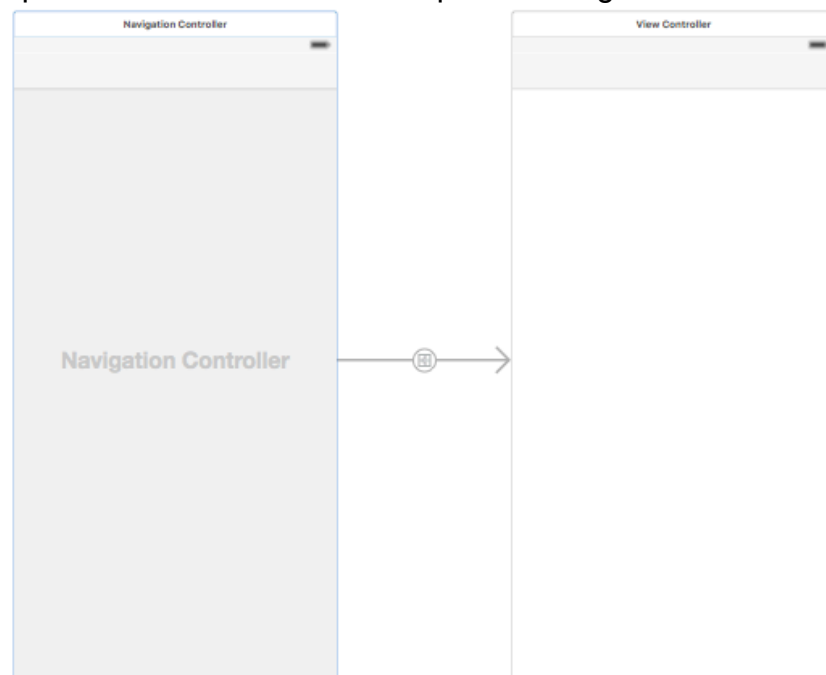
Pour lier les deux composants, appuyer sur ctrl, cliquer sur le Navigation Controller et glisser sur le View Controller.



Lorsque vous relâchez la souris le panel de type de « segue » s'ouvre, vous laissez show par défaut :








Les deux composants sont maintenant liés par un « segue »





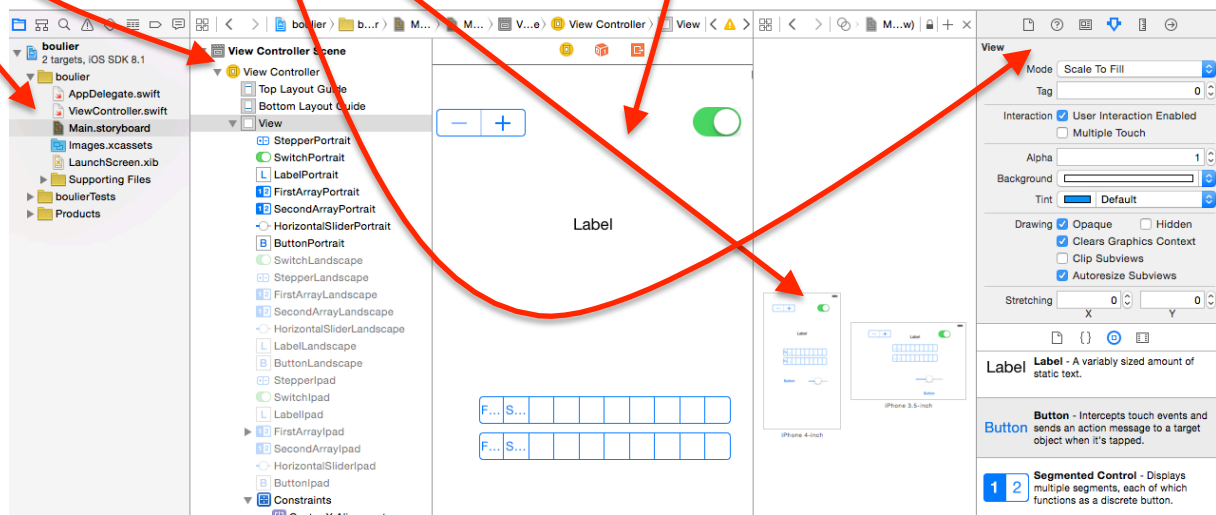
## 4. Xcode en mode story board

1. Choix du device  pour le lancement de l'application
2. Choix de la timeline  (passage à preview -> vue réelle au niveau story board)
3. Panneau gauche  (arborescence), panneau bas  (debug), panneau droit  (inspecteur))



L'écran de Xcode se compose de 5 zones au maximum.

1. L'arborescence du projet
2. L'arborescence du ViewController
3. La frame (l'écran que vous réalisez)
4. La vue réelle
5. L'inspecteur



Les boutons en bas sont assez importants :

1. Réglages Universal story board
2. + ajout de device en vue réelle



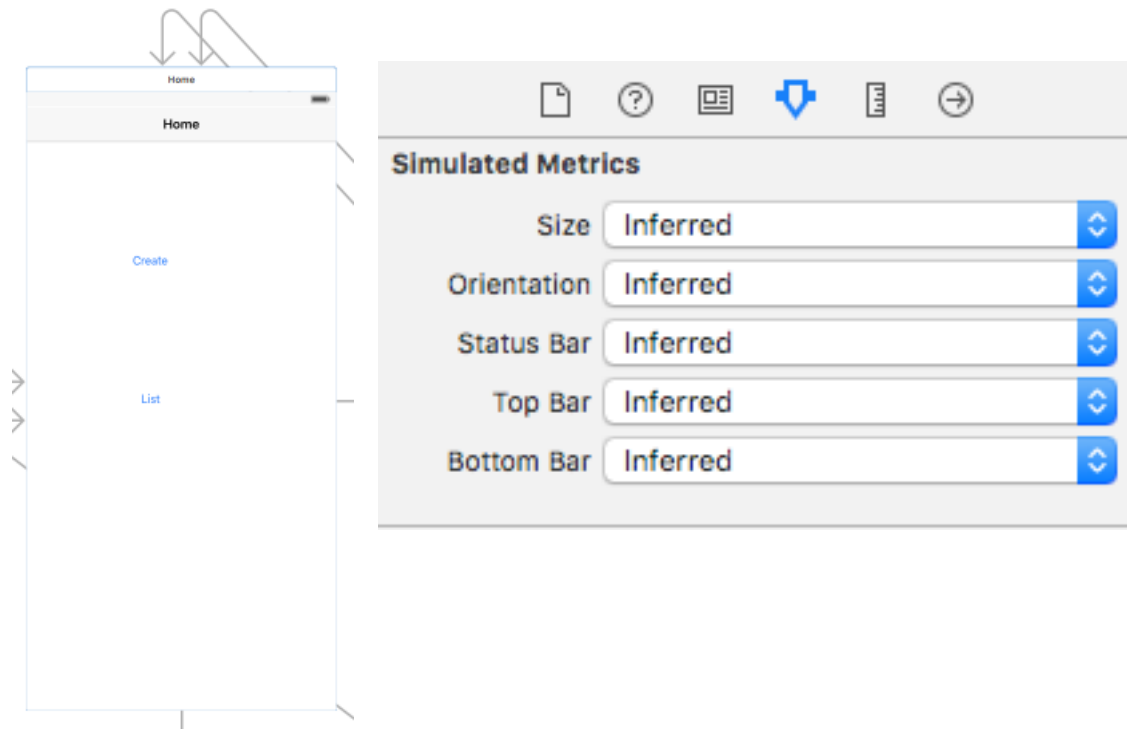
## 5. Universal story Board

Le système une fois compris est assez simple, voir même intuitif, et déconcertant.

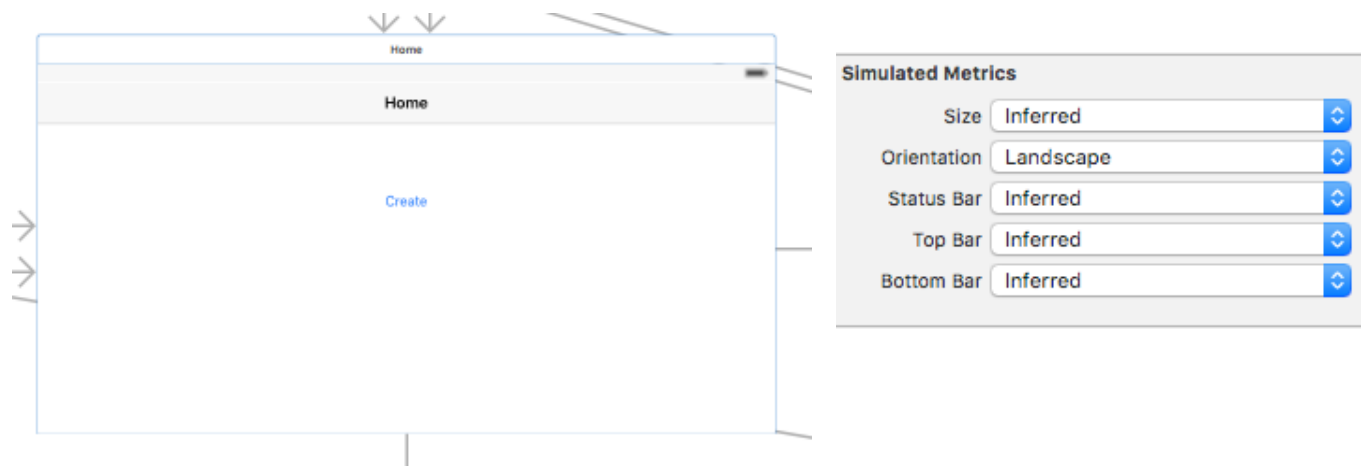
Le TP1 permet de réaliser l'ajustement aux différents écrans.

Voici ici d'autres fonctionnalités pour travailler avec des écrans en paysage ou portrait :

Par défaut le travail sur une vue du Story Board se fait avec « Inferred » (par défaut) dans l'inspector :



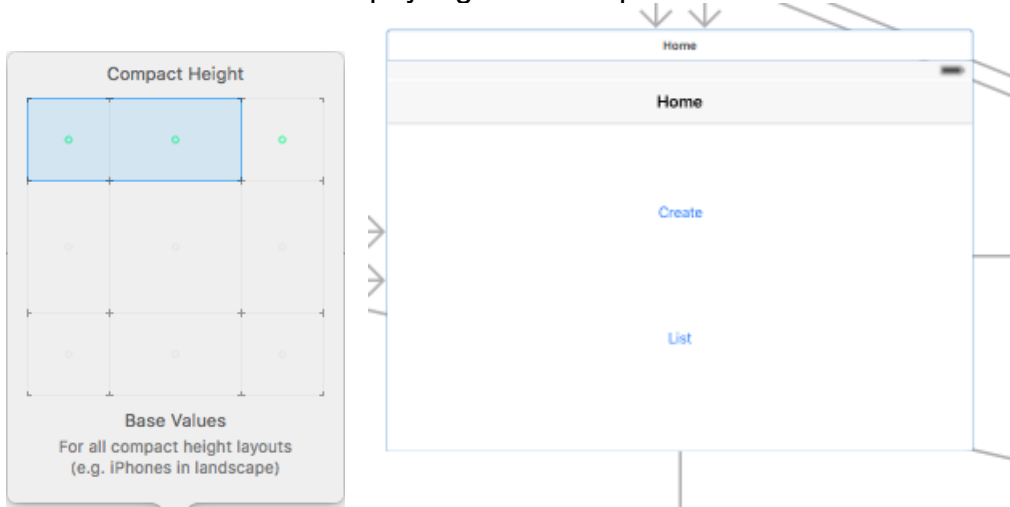
Si vous changez « Orientation » dans l'inspector, vous pouvez avoir la vue du device en question dans le story board :



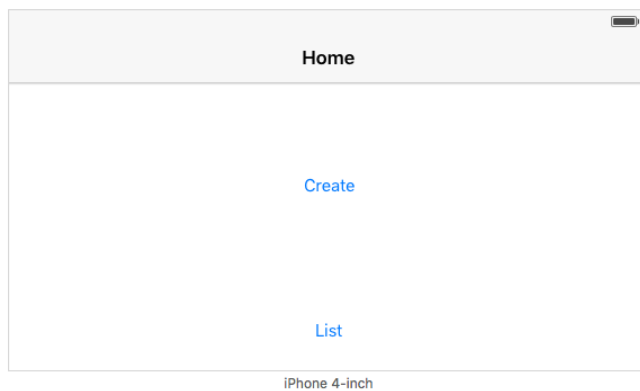
Cela permet de mieux gérer les vues en paysage.

**Exemple pour travailler un iphone 4 en vue paysage :**

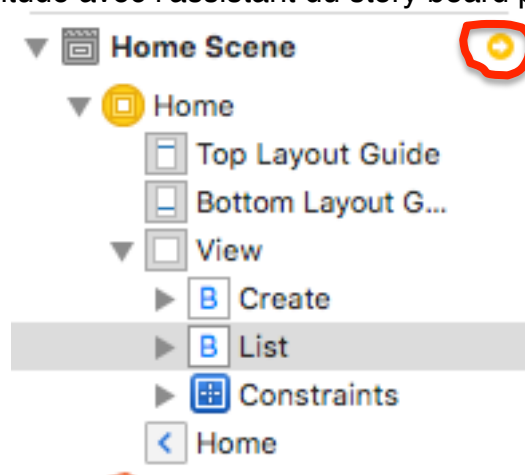
Régler le Layout pour la vue Iphone paysage, orienter le Story Board en paysage, avec « Orientation » en paysage dans inspector



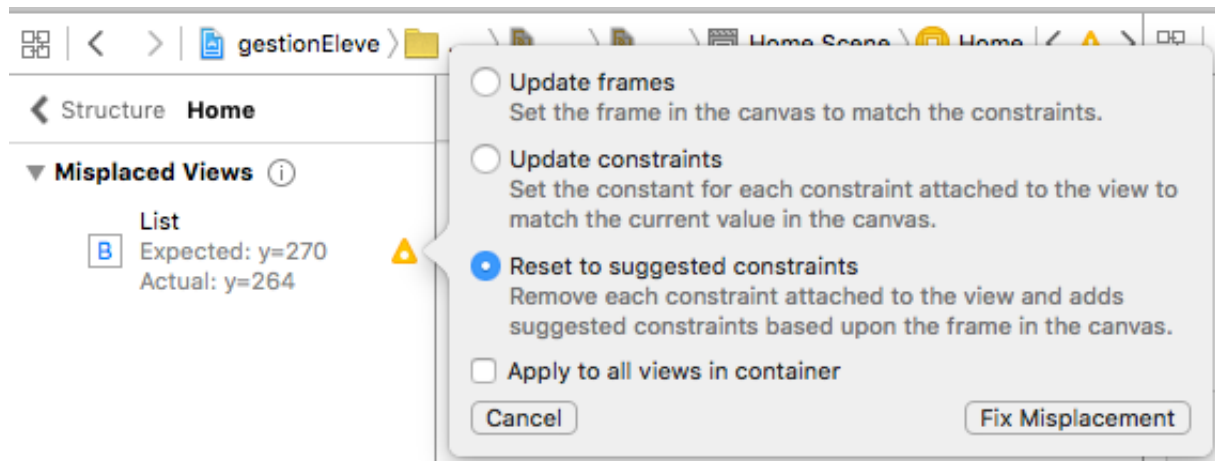
La vue en preview doit suivre les réglages, et être aussi en paysage (si vous l'utilisez)



Procéder comme d'habitude avec l'assistant du story board pour les contraintes.

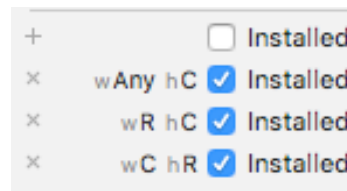


La couleur de l'assistant peut être jaune ou rouge suivant l'urgence dans le réglage des contraintes



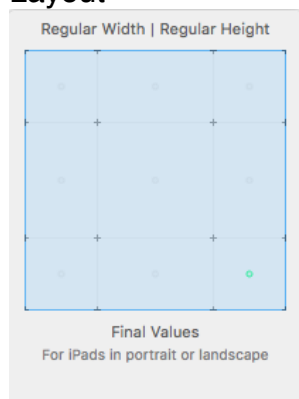
« Reset to suggested constraints » permet de travailler de manière automatique. Rien à calculer, il suffit de faire « Fix Misplacement » pour valider.

Ne pas oublier d'adapter les composants de la vue suivant les dimensions du Layout, sinon vos contraintes resteront sans effets :

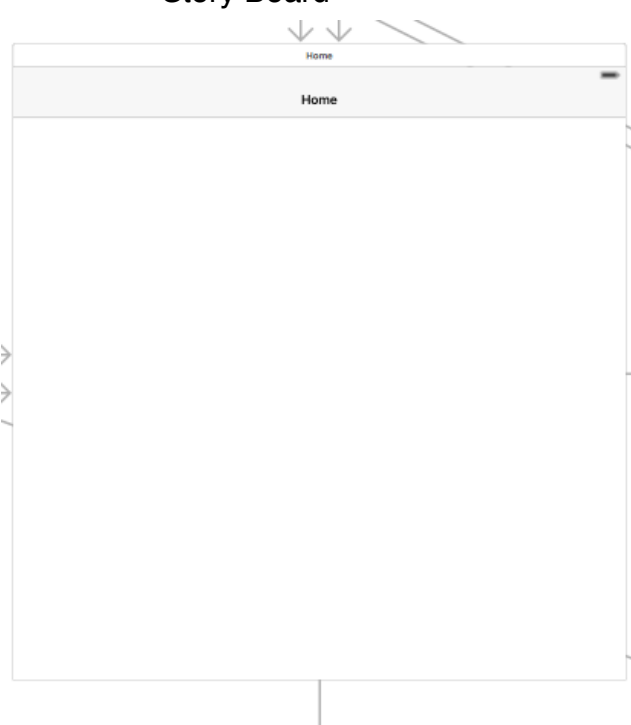


**Pour un ipad en mode portrait :**

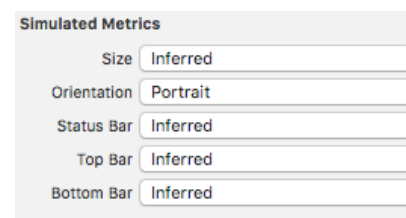
Layout



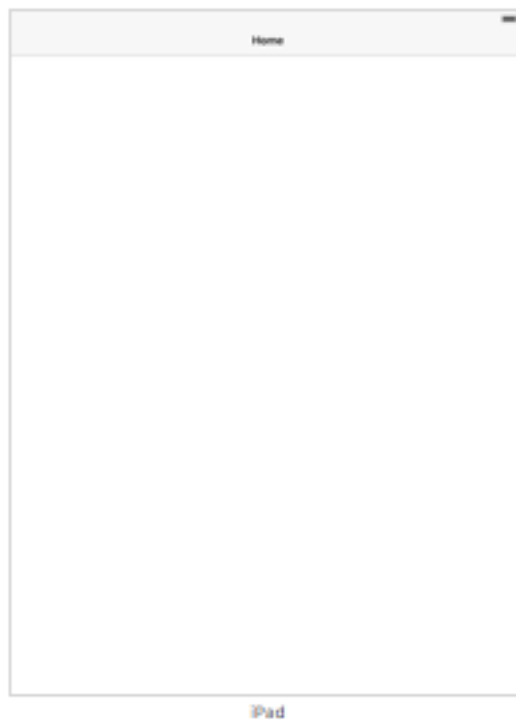
Storyboard



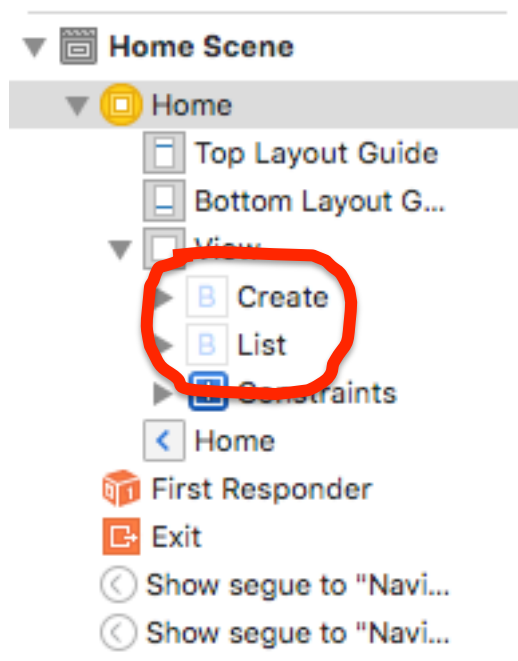
Inspector



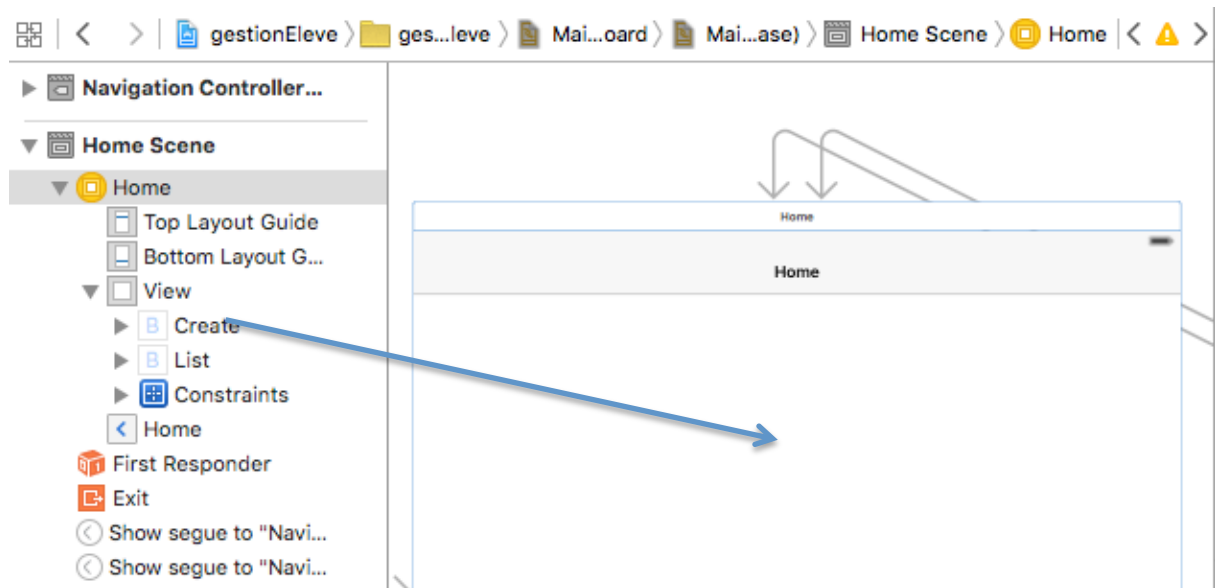
Preview :



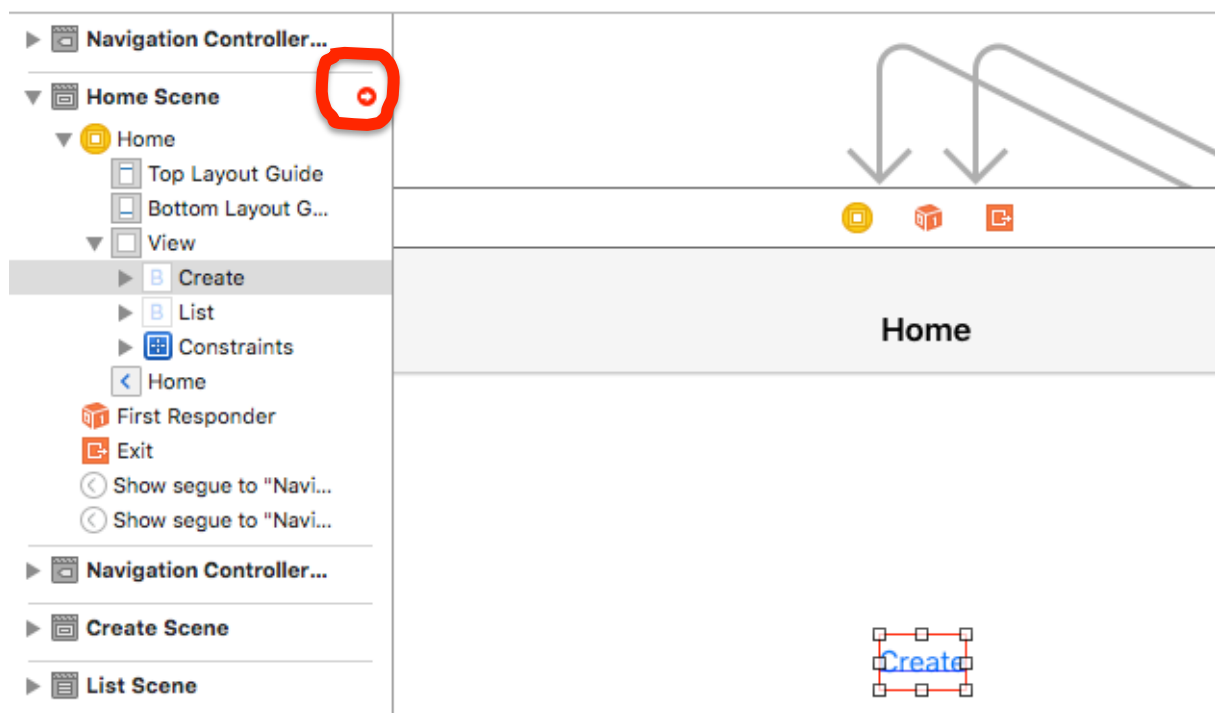
Pour l'instant aucun bouton n'est présent, en effet ils sont désactivés :



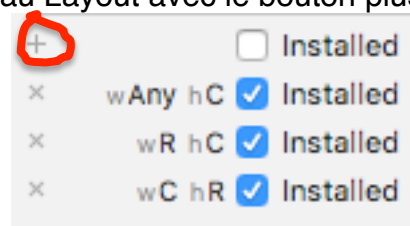
Je glisse le bouton « Create » sur le Story Board



L'assistant me signale des contraintes à ajouter :



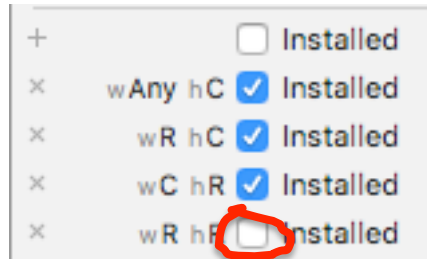
J'ajoute déjà le composant au Layout avec le bouton plus (dans l'inspecteur) :



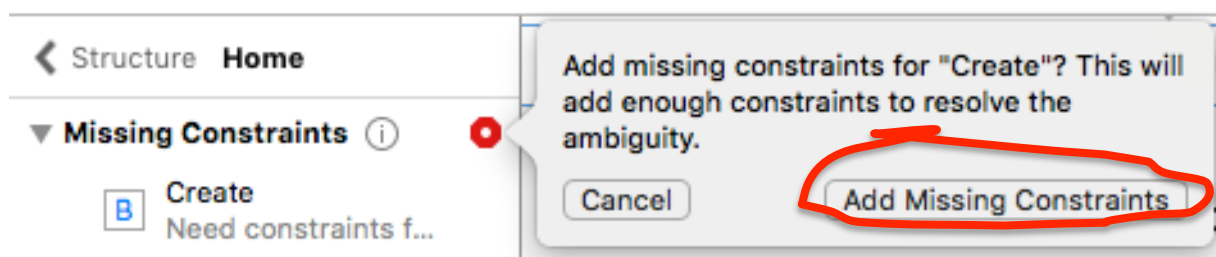
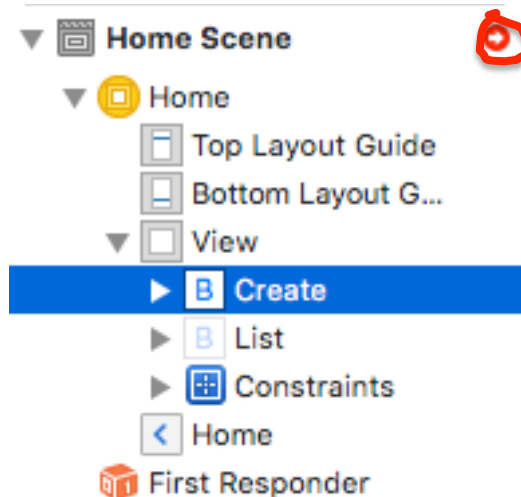
Par défaut la taille du Layout est proposée :



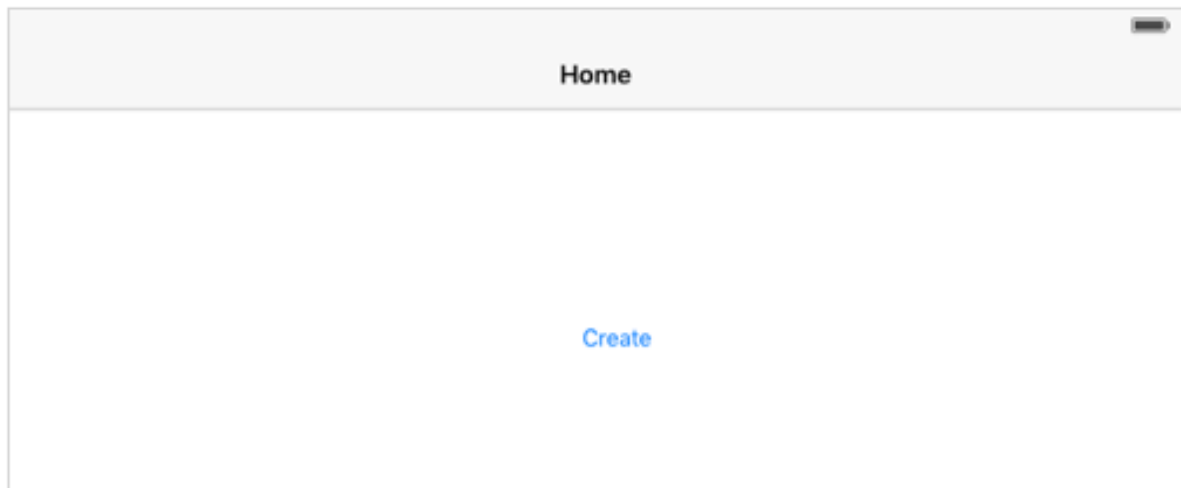
Je coche la nouvelle contrainte :



Je vais ensuite dans l'assistant faire les modifications attendues :



La contrainte est ajoutée, le bouton apparaît dans la « Preview » vide initialement :

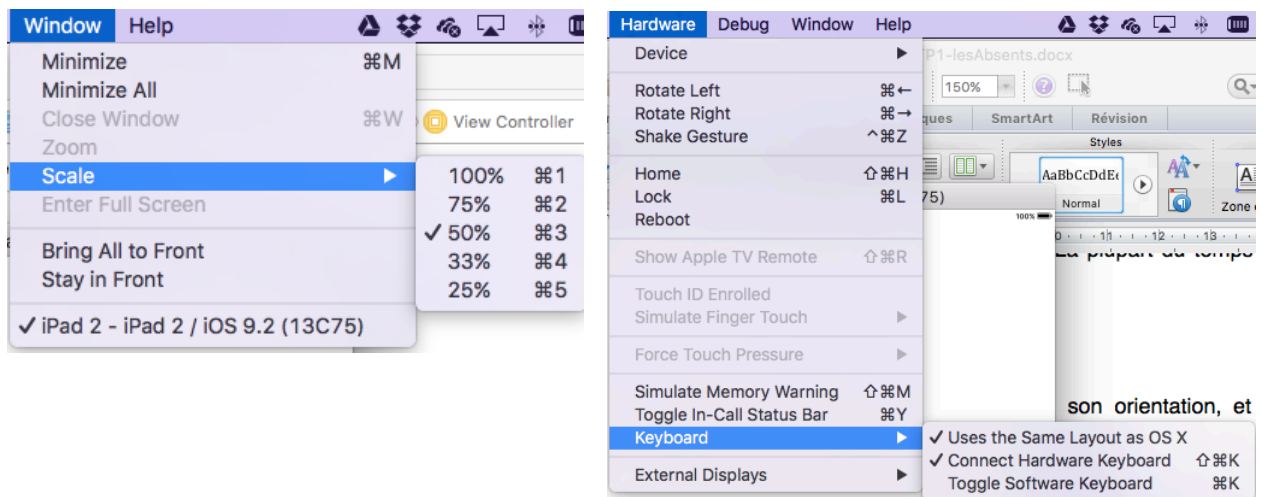


Le double clic dans la « Preview » permet de grossir ou diminuer la taille des fenêtres de manière limitée sur le niveau de zoom. Avec un mac book pro et le pad vous pouvez grossir ou diminuer de manière plus précise (agrandissement ou réduction avec deux doigts sur le pad)

Ensuite même manipulation pour le bouton List.

## 7. iOS simulator

Le simulateur permet de régler la taille de l'écran, son orientation, et le clavier connecté :



## Partie 2 : le codage de l'application

Cette version de l'application utilise Core Data, il est nécessaire de cocher la case Core Data à la création du projet.

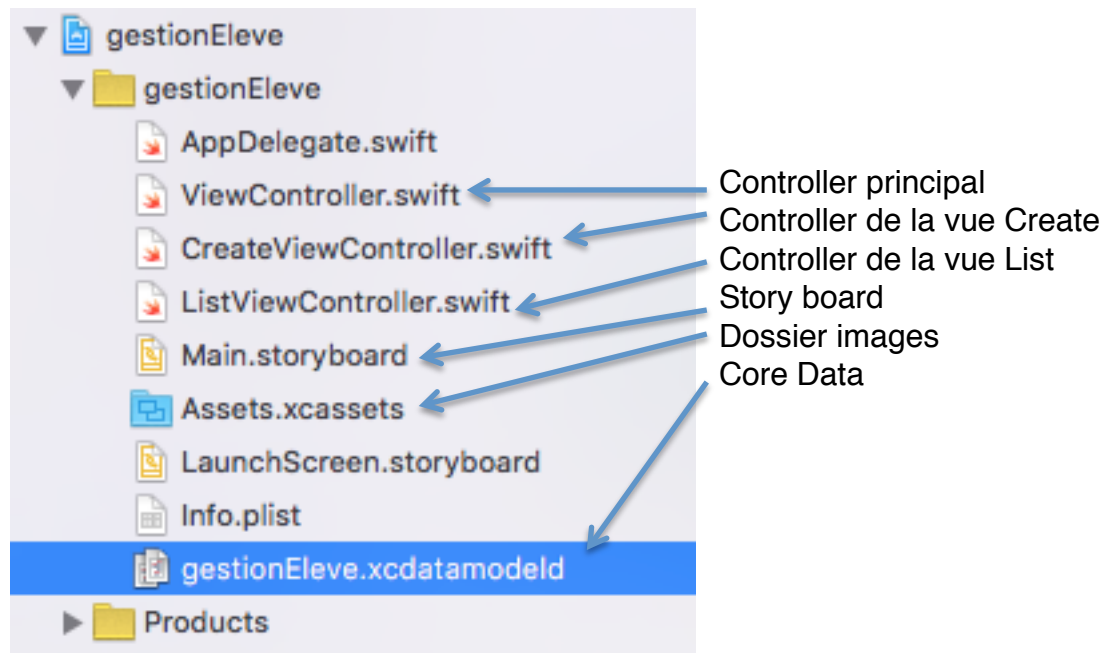
### 2.1. Core Data

C'est l'ORM de Xcode.

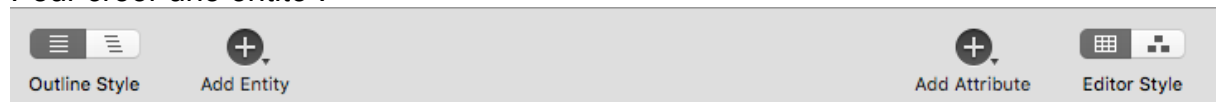


Un **mapping objet-relationnel** (en anglais **object-relational mapping** ou **ORM**) est une technique de programmation informatique qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle en définissant des correspondances entre cette base de données et les objets du langage utilisé. On pourrait le désigner par « correspondance entre monde objet et monde relationnel ».

L'arborescence du projet :



Pour créer une entité :



L'entité Student :

ENTITIES	
<b>E</b> Student	
<b>E</b> UnderGraduate	
FETCH REQUESTS	
CONFIGURATIONS	
<b>C</b> Default	

Attributes		
Attribute ^	Type	
<b>S</b> birthday	String	↕
<b>S</b> firstname	String	↕
<b>S</b> lastname	String	↕
<b>S</b> mail	String	↕
<b>S</b> number	String	↕
<b>S</b> phone	String	↕
+ -		

L'entité UnderGraduate :

ENTITIES

E

 Student

E

 UnderGraduate

FETCH REQUESTS

CONFIGURATIONS

C

 Default

▼

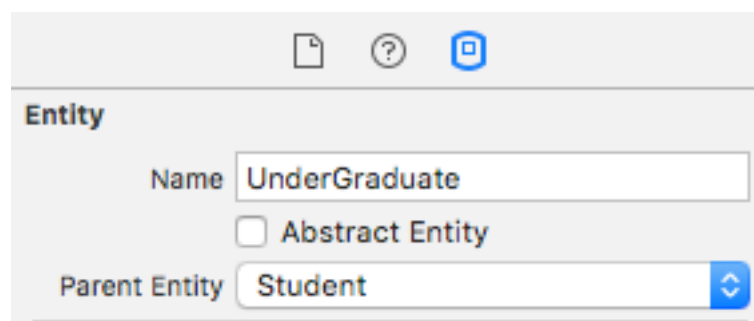
 Attributes

Attribute ^	Type	
<div>B</div> degree	Boolean	↕
<div>S</div> mention	String	↕
<div>B</div> scholarship	Boolean	↕
<div>S</div> series	String	↕

+

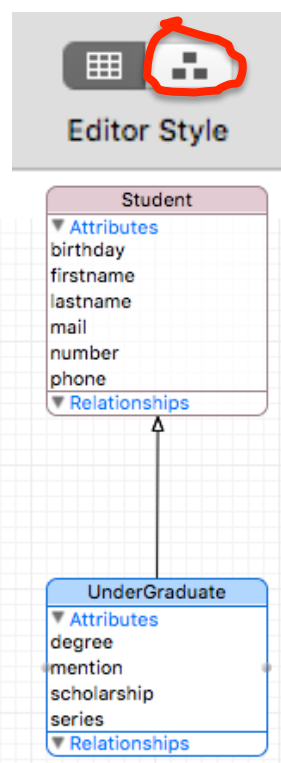
—

Attention, UnderGraduate hérite de Student, il faut régler cela dans l'inspecteur :



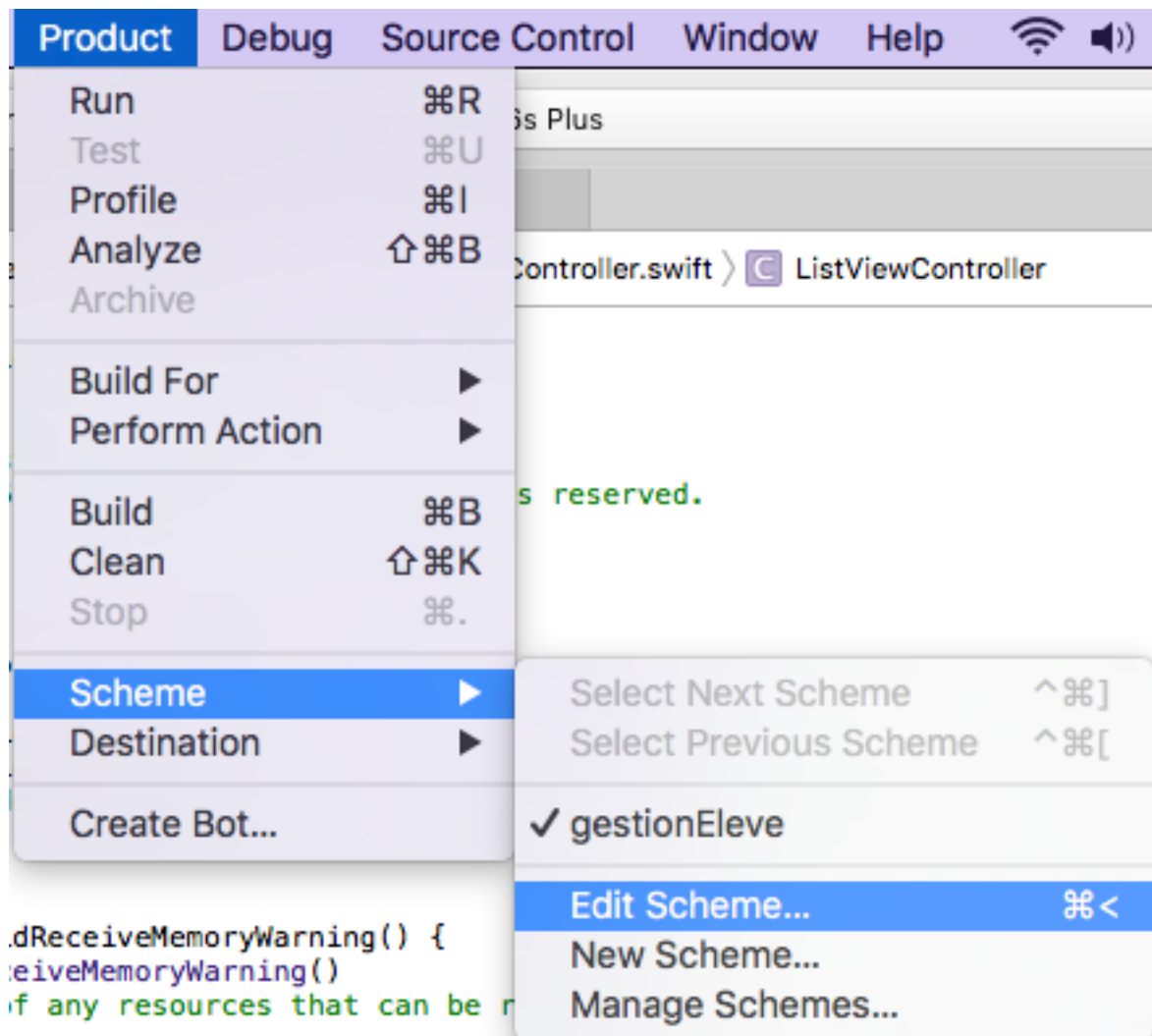
The Inspector window shows the configuration for the 'UnderGraduate' entity. The 'Name' field is set to 'UnderGraduate'. The 'Abstract Entity' checkbox is unchecked. The 'Parent Entity' dropdown menu is set to 'Student'.

Le mode graphique confirme l'héritage :



## 2.2. SQL mode in console

Pour activer le mode SQL verbeux dans la console, il faut aller dans les propriétés du projet :



Et ajouter la ligne **-com.apple.CoreData.SQLDebug 1** pour le mode debug :



Plus d'informations : <https://lukabratos.me/blog/2014/08/24/debugging-core-data/>

La console affiche les ordres SQL :

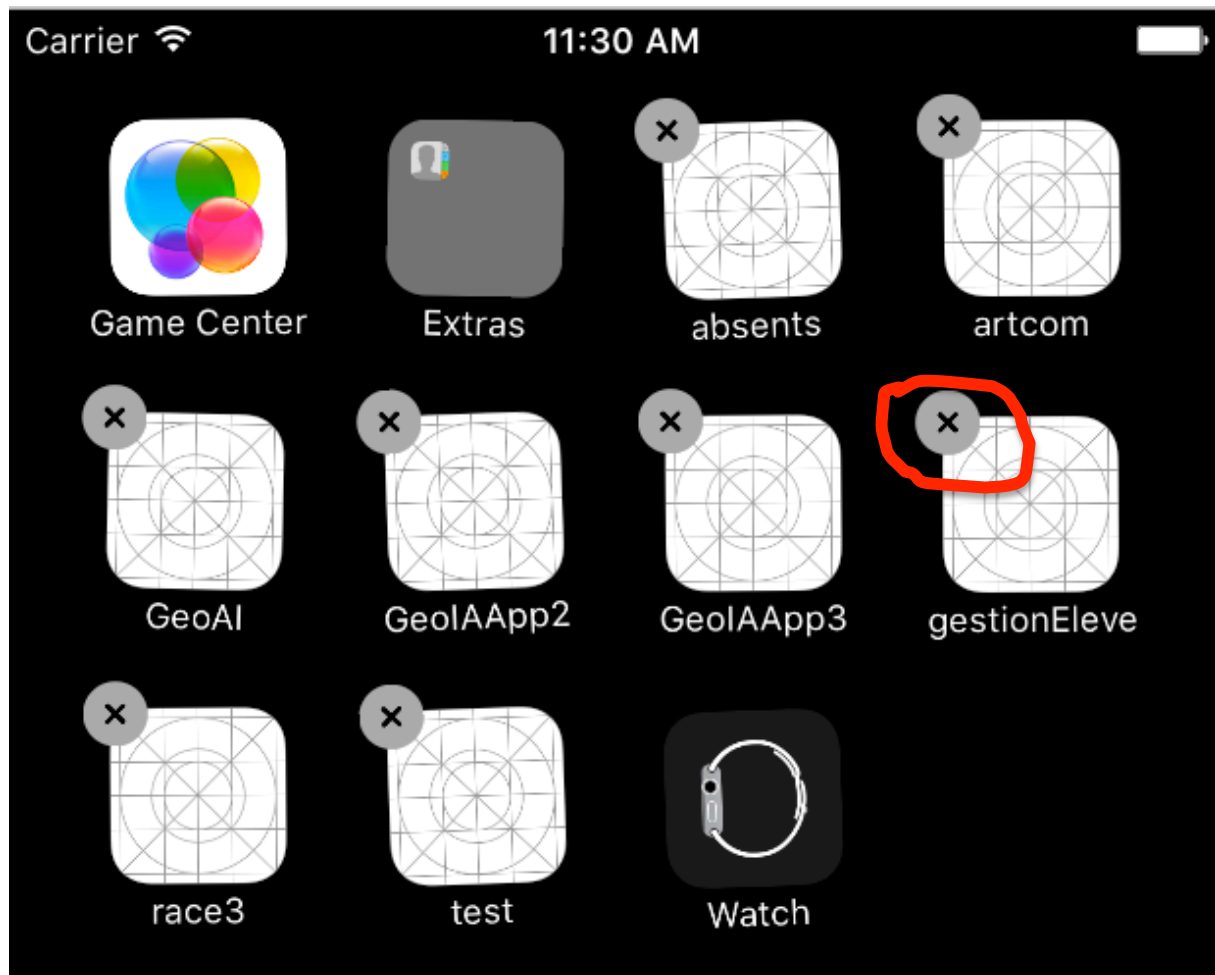
```

CoreData: sql: SELECT t0.Z_ENT, t0.Z_PK, t0.Z_OPT,
t0.ZBIRTHDAY, t0.ZFIRSTNAME, t0.ZLASTNAME, t0.ZMAIL,
t0.ZNUMBER, t0.ZPHONE, t0.ZDEGREE, t0.ZMENTION,
t0.ZSCHOLARSHIP, t0.ZSERIES FROM ZSTUDENT t0 WHERE
t0.ZNUMBER <> ?
2016-03-05 18:23:07.672 gestionEleve[1201:760036]
CoreData: annotation: sql connection fetch time:
0.0009s
2016-03-05 18:23:07.673 gestionEleve[1201:760036]
CoreData: annotation: total fetch execution time:
0.0028s for 10 rows.

```

### 2.3. Access to sqlite file

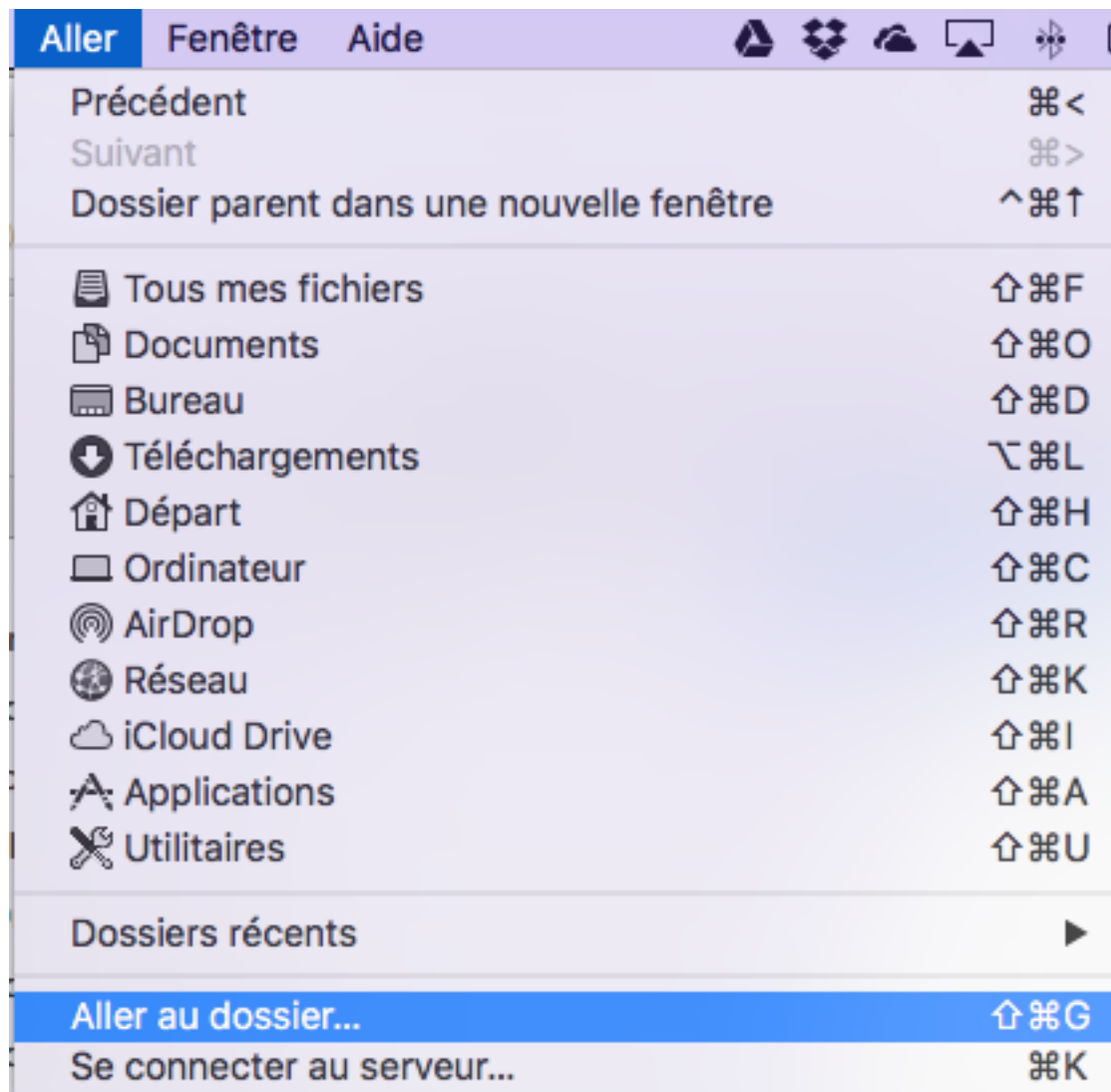
Si vous avez changé la structure de votre base de données, changement de type d'une colonne ou changement de nom d'une colonne, ajout d'une table, ..., il faut supprimer l'IPA du simulateur, et relancer l'application qui provoquera l'installation d'une nouvelle base :



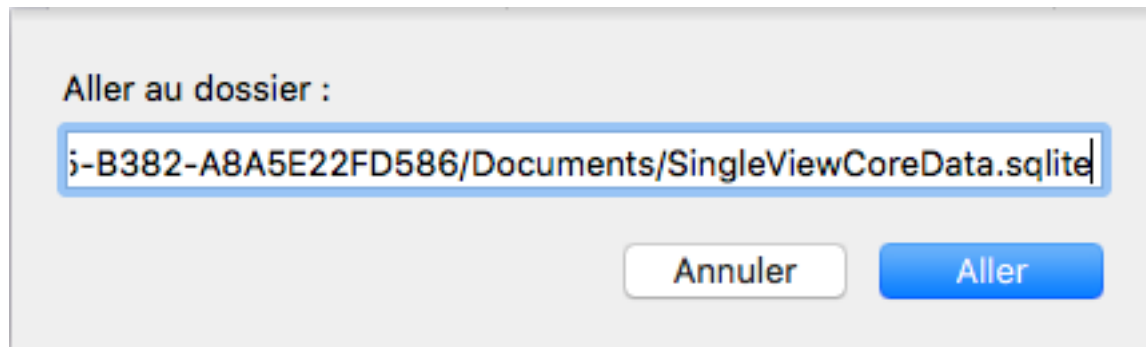
Pour accéder à la base de données, le mode SQL verbeux permet de faire apparaître le chemin de la base de données :

```
2016-03-01 11:15:17.270 gestionEleve[1015:408674]
CoreData: annotation: Connecting to sqlite database
file at "/Users/seb/Library/Developer/CoreSimulator/
Devices/840F88EE-D555-455D-BAFE-0ABB22509F80/data/
Containers/Data/Application/DF661958-C907-4A05-B382-
A8A5E22FD586/Documents/SingleViewCoreData.sqlite"
2016-03-01 11:15:17.272 gestionEleve[1015:408674]
```

Avec le Finder choisissez l'option aller au dossier :

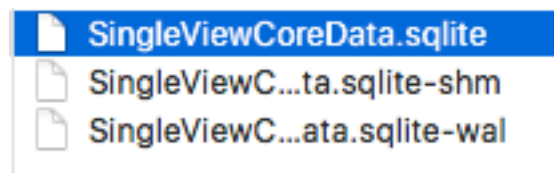


Copier/coller le chemin :

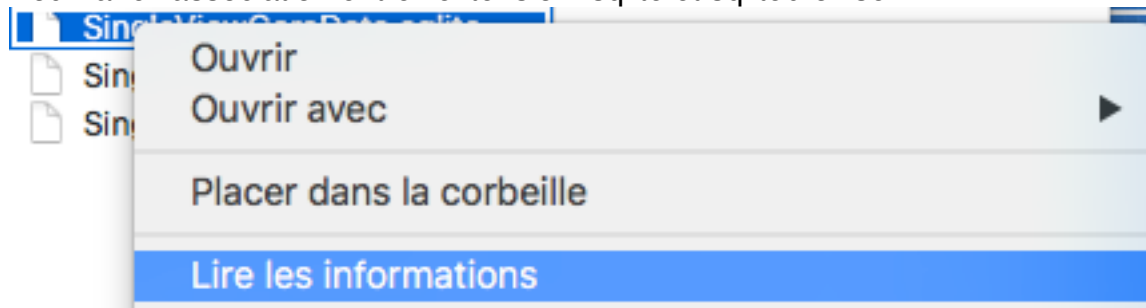


Go !

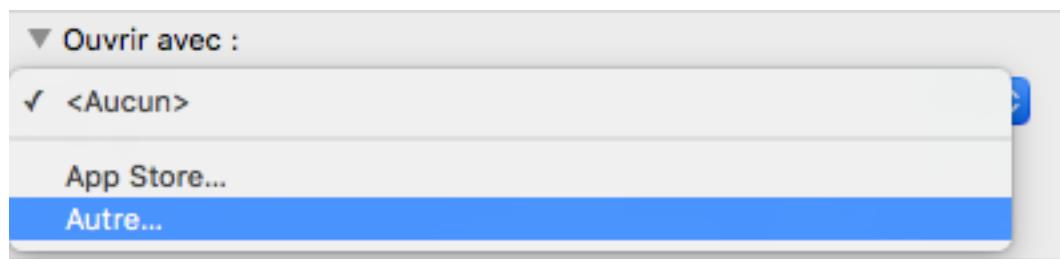
Pour ouvrir le fichier SQLite vous pouvez utiliser sqlitebrowser.dmg qui est une application open source gratuite : <http://sqlitebrowser.org>



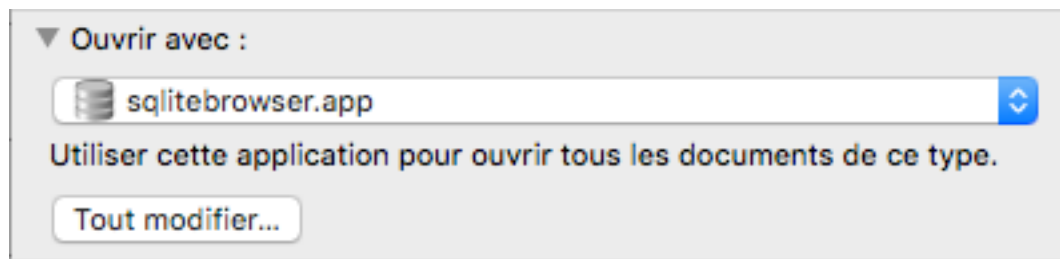
Pour faire l'association entre l'extension .sqlite et sqlitebrowser :



Aller à la rubrique ouvrir avec :



Choisir sqlitebrowser pour ouvrir :



Tout modifier permet d'associer tous les fichiers .sqlite de votre machine à cet utilitaire.

**ATTENTION** : sqlitebrowser doit être quitté à chaque utilisation, sinon un verrou empêche votre application d'accéder à la base de données SQLite.

## 2.4. SQLiteBrowser

Au final, il n'y a qu'une seule table ZStudent qui enregistre tous les attributs des deux entités.

La structure est la suivante :

Nom	Type	Schéma
▼ Tables (4)		
▼ ZSTUDENT		CREATE TABLE ZSTUDENT ( Z_PK INTEGER PRIMARY KE
Z_PK	INTEGER	'Z_PK' INTEGER
Z_ENT	INTEGER	'Z_ENT' INTEGER
Z_OPT	INTEGER	'Z_OPT' INTEGER
ZDEGREE	INTEGER	'ZDEGREE' INTEGER
ZSCHOLARSHIP	INTEGER	'ZSCHOLARSHIP' INTEGER
ZBIRTHDAY	VARCHAR	'ZBIRTHDAY' VARCHAR
ZFIRSTNAME	VARCHAR	'ZFIRSTNAME' VARCHAR
ZLASTNAME	VARCHAR	'ZLASTNAME' VARCHAR
ZMAIL	VARCHAR	'ZMAIL' VARCHAR
ZNUMBER	VARCHAR	'ZNUMBER' VARCHAR
ZPHONE	VARCHAR	'ZPHONE' VARCHAR
ZMENTION	VARCHAR	'ZMENTION' VARCHAR
ZSERIES	VARCHAR	'ZSERIES' VARCHAR
▶ Z_METADATA		CREATE TABLE Z_METADATA (Z_VERSION INTEGER PRIM
▶ Z_MODELCACHE		CREATE TABLE Z_MODELCACHE (Z_CONTENT BLOB)
▶ Z_PRIMARYKEY		CREATE TABLE Z_PRIMARYKEY (Z_ENT INTEGER PRIMA
▼ Index (1)		
ZSTUDENT_Z_ENT_INDEX		CREATE INDEX ZSTUDENT_Z_ENT_INDEX ON ZSTUDENT
Vues (0)		
Déclencheurs (0)		

Les données sont les suivantes :

Table : ZSTUDENT

Nouvel Enregistrement Supprimer l'enregistrement

OLARSHIP	ZBIRTHDAY	ZFIRSTNAME	ZLASTNAME	ZMAIL	ZNUMBER	ZPHONE	ZMENTIC
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	02-03-2016	Jean	Charles	jcharles@gmail.c...	8EggZBybtbFwT...	123456	NULL
2	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	02-02-2015	Denis	Vuiloz	dv@gmx.fr	L1laPZZ3DmWYl...	123456789	NULL
4	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	02-02-2017	Steve	Bob	sbob@sfr.fr	TCKmq6fRIQDb5...	0987654	NULL
6	02-02-2017	Steve	Bob	sbob@sfr.fr	TCKmq6fRIQDb5...	0987654	No mention
7	05-04-2016	Alex	Kenn	kenn@google.fr	dQV8phyQZHVE...	09876543	NULL
8	05-04-2016	Alex	Kenn	kenn@google.fr	dQV8phyQZHVE...	09876543	No mention
9	02-03-2014	John	Kenn	jk@sfr.fr	SOtJfJlJgAfLpPh...	12234567	NULL
10	02-03-2014	John	Kenn	jk@sfr.fr	SOtJfJlJgAfLpPh...	12234567	No mention
11	02-03-2016	jkjkj	jkjkj	jkjkj	UXlQ70fAzvenyP...		NULL

Chaque ajout d'une ligne crée un élève de base (student) et un étudiant (undergraduate) (deux lignes à chaque saisie). Si la ligne concerne juste une élève (student), une ligne à blanc (null) est présente pour l'étudiant (undergraduate). Si la ligne ajoutée, concerne un étudiant (undergraduate), une ligne concerne l'étudiant en tant qu'élève (student), la seconde, c'est l'étudiant lui-même (undergraduate).

- Une ligne Student, ne concerne que les attributs de Student
- Une ligne de UnderGraduate concerne les attributs de Students (héritage) et les siens.

A l'affichage de la liste de tous les individus présents, il faut faire un `SELECT DISTINCT * FROM ZSTUDENT`, cependant le `DISTINCT` qui existe reste sans effet dans l'ORM. Il faut alors régler le problème par un `DISTINCT` en Swift.

## 2.2. Le ViewController

Le View controller de la vue « Home » ne comprend aucun code particulier, puisque les bouton « Create » et « List » sont associés à des « segues » qui lancent les vues « Create » et « List ». Le code spécifique est donc présent dans ses vues.



```

//
// ViewController.swift
// gestionEleve
//
// Created by Sébastien Gagneur on 29/02/2016.
// Copyright © 2016 Sébastien Gagneur. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

}

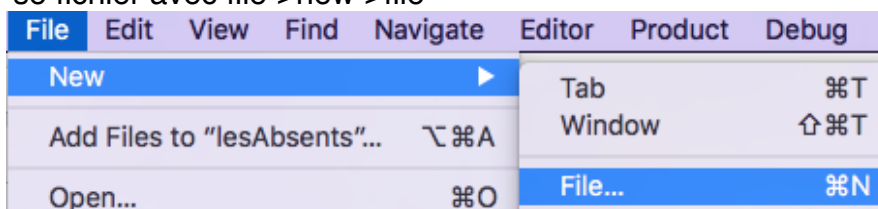
```

Nous n'utiliserons pas de classe métier codée en Swift, elles seront implémentées dans core Data.

## 2.3 CreateViewController

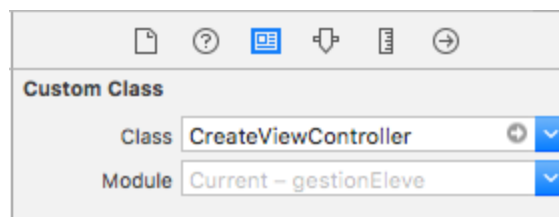
C'est le code du controller de la seconde fenêtre qui s'affiche lorsque vous cliquez sur le bouton « Create ». Elle permet de saisir les caractéristiques d'un élève.

Il faut créer ce fichier avec file->new->file



Il faut aussi associer ce controller à la vue « Create » :

L'inspecteur permet là encore de régler la chose :



Au chargement de la vue, l'association des zones de texte, pickers divers, bouton de création doit être faite avec le contrôleur.

Le chargement de la vue permet d'initialiser certains champs :

```
import UIKit
import CoreData

class CreateViewController: UIViewController, UIPickerViewDataSource, UIPickerViewDelegate {

    var dateOfBirth : String = " "
    var mention : String = " "
    var serie : String = " "

    @IBOutlet weak var firstNameTextField: UITextField!

    @IBOutlet weak var lastNameTextField: UITextField!
    @IBOutlet weak var mailTextField: UITextField!

    @IBOutlet weak var phoneTextField: UITextField!

    @IBOutlet weak var birthDatePicker: UIDatePicker!

    @IBOutlet weak var degreeSwitch: UISwitch!

    @IBOutlet weak var scholarshipSwitch: UISwitch!

    @IBOutlet weak var pickerViewSerie: UIPickerView!
    // those values could be download from a table in SQLite
    var pickerViewSerieDataSource = ["STI2D", "STMG", "BacPro", "S", "ES", "L"]

    @IBOutlet weak var pickerViewMention: UIPickerView!
    // those values could be download from a table in SQLite
    var pickerViewMentionDataSource = ["No mention", "Good enough", "Very good"]

    override func viewDidLoad() {
        super.viewDidLoad()
        self.pickerViewSerie.dataSource = self
        self.pickerViewSerie.delegate = self
        self.pickerViewMention.dataSource = self
        self.pickerViewMention.delegate = self

        // default values for all picker !

        serie = pickerViewSerieDataSource[0]
        mention = pickerViewMentionDataSource[0]
        let dateFormatter = NSDateFormatter()
        // format in your string
        dateFormatter.dateFormat = "dd-MM-yyyy"
        dateOfBirth = dateFormatter.stringFromDate(birthDatePicker.date)
    }
}
```

Les différents pickers se gèrent de la manière suivante :

```

@IBAction func birthDatePickerAction(sender: AnyObject) {
    // Don't forget to set limits and features for your date picker in inspector panel through story
    board
    let dateFormatter = NSDateFormatter()
    // format in your string
    dateFormatter.dateFormat = "dd-MM-yyyy"
    dateOfBirth = dateFormatter.stringFromDate(birthDatePicker.date)
}

func numberOfComponentsInPickerView(pickerView: UIPickerView) -> Int {
    // number of columns for a picker view
    // All my pickers View have only one column
    return 1
}

func pickerView(pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
    // Don't forget to set tag value to make difference between each picker view -> Story board +
    inspector
    var count : Int = 0
    if pickerView.tag == 1
    {
        count = pickerViewSerieDataSource.count
    }
    if pickerView.tag == 2
    {
        count = pickerViewMentionDataSource.count
    }
    return count
}

func pickerView(pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) ->
String? {
    // Don't forget to set tag value to make difference between each picker view -> story board +
    inspector
    var selectedRow : String = ""
    if pickerView.tag == 1
    {
        selectedRow = pickerViewSerieDataSource[row]
        serie = pickerViewSerieDataSource[row]
    }
    if pickerView.tag == 2
    {
        selectedRow = pickerViewMentionDataSource[row]
        mention = pickerViewMentionDataSource[row]
    }
    return selectedRow
}

```

Le bouton « Create » permet de créer un individu. Ce n'est pas l'action sur le bouton (@IBAction) mais le déclenchement du « segue » associé qui va faire la création dans la base de données :

```

override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    if (segue.identifier == "Create")
    {
        // create rows in SQLite with CoreData

        // let student : Student
        // let underGraduate : UnderGraduate

        let number = randomStringWithLength(26)
        print("OK \(number)")
        //1
        let appDelegate =
        UIApplication.sharedApplication().delegate as! AppDelegate

        let managedContext = appDelegate.managedObjectContext

        //2
        let entityStudent = NSEntityDescription.entityForName("Student",
            inManagedObjectContext:managedContext)

        let studentRow = NSManagedObject(entity: entityStudent!,
            insertIntoManagedObjectContext: managedContext)

        let entityUnderGraduate = NSEntityDescription.entityForName("UnderGraduate",
            inManagedObjectContext:managedContext)

        let undergraduateRow = NSManagedObject(entity: entityUnderGraduate!,
            insertIntoManagedObjectContext: managedContext)

        //3
        //student = Student(last: lastNameTextField.text!, first: firstNameTextField.text!, mail:
        mailTextField.text!, phone: phoneTextField.text!, day: dateOfBirth, num: number as
        String)

        //4
        studentRow.setValue(number, forKey: "number")
        studentRow.setValue(lastNameTextField.text, forKey: "lastname")
        studentRow.setValue(firstNameTextField.text, forKey: "firstname")
        studentRow.setValue(mailTextField.text, forKey: "mail")
        studentRow.setValue(phoneTextField.text, forKey: "phone")
        studentRow.setValue(dateOfBirth, forKey: "birthday")
    }
}

```

Suivant la valeur du switch, pour un bachelier ou pas, on fait une création différente. Le switch est off, c'est un élève, le switch est on c'est un étudiant avec niveau bac. L'élève ne possède que des caractéristiques de base : numéro, nom, prénom, mail, téléphone, anniversaire.

L'étudiant, enregistre en plus (il hérite) l'état bachelier, l'état boursier, sa série de bac, et sa mention.

Il suffit ensuite d'enregistrer la ligne qui correspond au cas à traiter dans la base de données

```

if degreeSwitch.on
{
    // here, we have got one underGraduate instance
    //underGraduate = UnderGraduate(last: lastNameTextField.text!, first:
    firstNameTextField.text!, mail: mailTextField.text!, phone: phoneTextField.text!,
    day: "", num: number as String, deg: degreeSwitch.on, men: "", scholar:
    scholarShipSwitch.on, serie: "")

    //All of these values are come from inheritance with Student entity !
    undergraduateRow.setValue(number, forKey: "number")
    undergraduateRow.setValue(lastNameTextField.text, forKey: "lastname")
    undergraduateRow.setValue(firstNameTextField.text, forKey: "firstname")
    undergraduateRow.setValue(mailTextField.text, forKey: "mail")
    undergraduateRow.setValue(phoneTextField.text, forKey: "phone")
    undergraduateRow.setValue(dateOfBirth, forKey: "birthday")

    // Properties of UnderGraduate entity only !
    undergraduateRow.setValue(degreeSwitch.on, forKey: "degree")
    undergraduateRow.setValue(scholarShipSwitch.on, forKey: "scholarship")
    undergraduateRow.setValue(serie, forKey: "series")
    undergraduateRow.setValue(mention, forKey: "mention")
}

//5
do {
    try managedContext.save()
    //6
    //Students.append(studentRow)
    //underGraduates.append(undergraduateRow)
} catch let error as NSError {
    print("Could not save \(error), \(error.userInfo)")
}
}
}

```

## 2.4 ListViewController

C'est le code du controller de la fenêtre qui liste les élèves et étudiants lorsque vous cliquez sur le bouton « List ». Elle permet de lister tout les élèves (students) enregistrés, ou seulement les étudiants (undergraduate) avec le bouton switch.

```

import UIKit
import CoreData

class ListViewController: UITableViewController {

    var Students = [NSManagedObject]()
    var underGraduates = [NSManagedObject]()
    var switchButton : UISwitch = UISwitch()
    var checkbox : Bool = false
    var distinctNumber : String = ""

    override func viewDidLoad() {
        super.viewDidLoad()

        switchButton.on = false
        switchButton.addTarget(self, action: "switchButtonAction:", forControlEvents: .ValueChanged)
        self.navigationItem.rightBarButtonItem = UIBarButtonItem(customView: switchButton)

        self.tableView.tableFooterView = UIView(frame: CGRectZero)
    }
}

```

Au chargement de la vue, l'ajout du bouton switch se fait dans le programme. Pas par le Story Board, parce que cela ne permet pas d'avoir un type UISwitch à manipuler, même un transtypage ne règle pas le problème.

Il faut associer, le bouton switch à une action (méthode), encore une fois cela se fait par le programme, il n'y pas d'IBAction ici !

La méthode switchButtonAction est présentée plus loin.

Dans le cas d'un élève (student), la gestion du DISTINCT se fait par le programme.

```
// In case of Student
if switchButton.on == false
{
    //1
    let appDelegate =
    UIApplication.sharedApplication().delegate as! AppDelegate

    let managedContext = appDelegate.managedObjectContext

    //2
    let fetchRequest = NSFetchRequest(entityName: "Student")
    fetchRequest.predicate = NSPredicate(format: "(number != 'null')")

    //3
    do {
        let results =
        try managedContext.executeFetchRequest(fetchRequest)
        Students = results as! [NSManagedObject]
    } catch let error as NSError {
        print("Could not fetch \(error), \(error.userInfo)")
    }

    // Select distinct number from students
    distinctNumber = ""
    var index : Int = 0
    for oneStudent in Students
    {
        let n = ((oneStudent.valueForKey("number")) as? NSString) as? String
        if (distinctNumber == n)
        {
            Students.removeAtIndex(index)
            index--
        }
        else
        {
            distinctNumber = n!
        }
        index++
    }
}
```

Dans le cas d'un étudiant :

```

// In case of undergraduate
else
{
    //1
    let appDelegate =
        UIApplication.sharedApplication().delegate as! AppDelegate

    let managedContext = appDelegate.managedObjectContext

    //2
    let fetchRequest = NSFetchRequest(entityName: "UnderGraduate")
    fetchRequest.predicate = NSPredicate(format: "(number != 'null') AND (degree != 'null')")

    //3
    do {
        let results =
            try managedContext.executeFetchRequest(fetchRequest)
        undergraduates = results as! [NSManagedObject]
    } catch let error as NSError {
        print("Could not fetch \(error), \(error.userInfo)")
    }
}

```

La méthode `switchButtonAction` associée au bouton `switch`, reprend les éléments précédents :

---

```

func switchButtonAction(sender : UISwitch)
{
    if sender.on
    {
        print("on")
    }
    else
    {
        print("off")
    }
    // In case of Student
    if switchButton.on == false
    {
        //1
        let appDelegate =
            UIApplication.sharedApplication().delegate as! AppDelegate

        let managedContext = appDelegate.managedObjectContext

        //2
        let fetchRequest = NSFetchRequest(entityName: "Student")
        fetchRequest.predicate = NSPredicate(format: "(number != 'null')")
        fetchRequest.returnsDistinctResults = true

        //3
        do {
            let results =
                try managedContext.executeFetchRequest(fetchRequest)
            Students = results as! [NSManagedObject]
        } catch let error as NSError {
            print("Could not fetch \(error), \(error.userInfo)")
        }
        // Select distinct number from students
        distinctNumber = ""
        var index : Int = 0
        for oneStudent in Students
        {
            let n = ((oneStudent.valueForKey("number")) as? NSString) as? String
            if (distinctNumber == n)
            {
                Students.removeAtIndex(index)
                index--
            }
            else
            {
                distinctNumber = n!
            }
            index++
        }
    }
}

```

Mais surtout elle recharge la Table View à la fin :



```

// In case of undergraduate
else
{
    //1
    let appDelegate =
        UIApplication.sharedApplication().delegate as! AppDelegate

    let managedContext = appDelegate.managedObjectContext

    //2
    let fetchRequest = NSFetchRequest(entityName: "UnderGraduate")
    fetchRequest.predicate = NSPredicate(format: "(number != 'null') AND (degree != 'null')")

    //3
    do {
        let results =
            try managedContext.executeFetchRequest(fetchRequest)
        undergraduates = results as! [NSManagedObject]
    } catch let error as NSError {
        print("Could not fetch \(error), \(error.userInfo)")
    }
}

tableView.reloadData()
}

```

Protocol pour la Table View :

On compte !

```

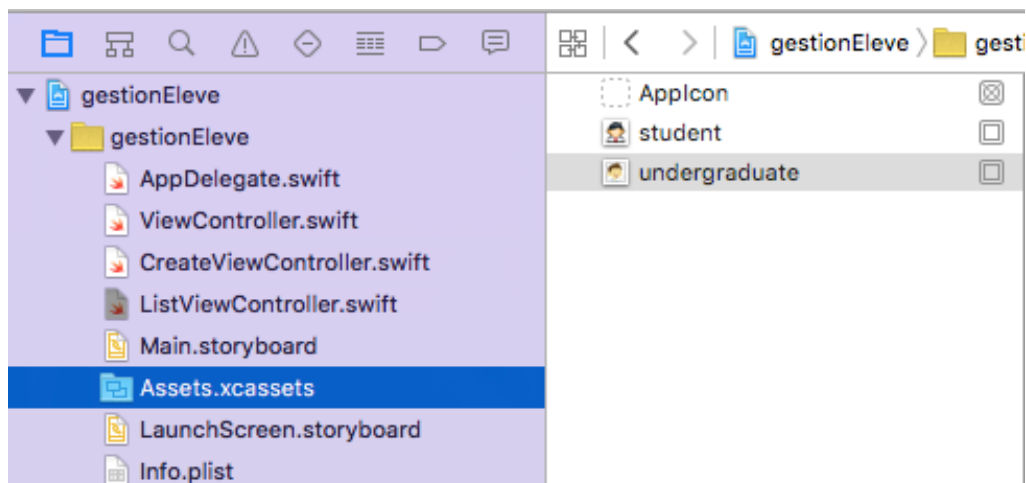
//COUNT
override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    var count : Int = 0
    if switchButton.on == false
    {
        count = Students.count
        print("élèves : \(count)")
    }
    else
    {
        count = undergraduates.count
        print("étudiants : \(count)")
    }
    return count
}

```

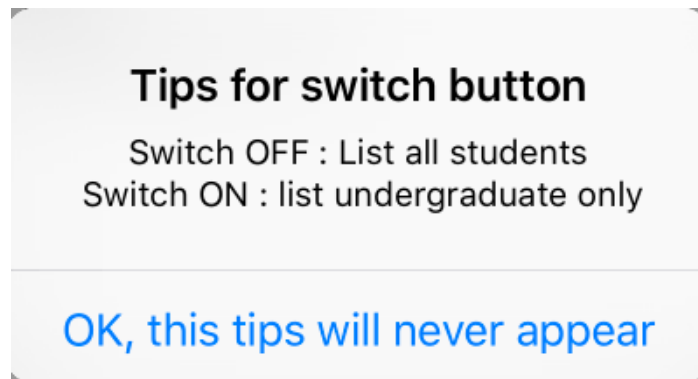
On insère dans la Table View avec les images :

```
//INSERT
override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
    UITableViewCell {
    if checkbox == false
    {
        let alert = UIAlertController(title: "Tips for switch button", message: "Switch OFF : List
all students \n Switch ON : list undergraduate only ", preferredStyle:
UIAlertControllerStyle.Alert)
        alert.addAction(UIAlertAction(title: "OK, this tips will never appear", style:
UIAlertActionStyle.Default, handler: nil))
        self.presentViewController(alert, animated: true, completion: alertCheck)
    }
    let cell: UITableViewCell = UITableViewCell(style: UITableViewCellStyle.Subtitle,
reuseIdentifier: "Cell")
    print("Ligne = \(indexPath.row)")
    if switchButton.on == false
    {
        let lastname = (Students[indexPath.row].valueForKey("lastname") as? NSString) as? String
        let firstname = (Students[indexPath.row].valueForKey("firstname") as? NSString) as? String
        let mail = (Students[indexPath.row].valueForKey("mail") as? NSString) as? String
        let birth = (Students[indexPath.row].valueForKey("birthday") as? NSString) as? String
        if (lastname != nil && firstname != nil && mail != nil && birth != nil)
        {
            let image : UIImage = UIImage(named: "student")!
            cell.imageView!.image = image
            cell.textLabel?.text = lastname
            let subString = firstname! + " " + mail! + " " + birth!
            cell.detailTextLabel?.text = subString
        }
    }
    else
    {
        let lastname = (underGraduates[indexPath.row].valueForKey("lastname") as? NSString) as?
String
        let firstname = (underGraduates[indexPath.row].valueForKey("firstname") as? NSString) as?
String
        let mail = (underGraduates[indexPath.row].valueForKey("mail") as? NSString) as? String
        let birth = (underGraduates[indexPath.row].valueForKey("birthday") as? NSString) as? String
        if (lastname != nil && firstname != nil && mail != nil && birth != nil)
        {
            let image : UIImage = UIImage(named: "undergraduate")!
            cell.imageView!.image = image
            cell.textLabel?.text = lastname
            let subString = firstname! + " " + mail! + " " + birth!
            cell.detailTextLabel?.text = subString
        }
    }
    return cell
}
}
```

Les images sont présentes dans Xcassets, il faudra penser à les glisser depuis le Finder :



Le message qui permet au début de préciser le fonctionnement du bouton Switch s'efface une fois qu'il a été affiché une fois :



Pour cela l'alerte est mappée sur la méthode `alerCheck()`, qui fait la chose suivante :

```
func alertCheck()  
{  
    checkbox = true  
}
```

Cette dernière méthode se passe de commentaire.