Vers des applications plus complexes…

camille.guinaudeau@u-psud.fr

Plan du cours

1. Manipuler plusieurs écrans
2. Communiquer avec l'utilisateur
3. Lire et enregistrer des données multimédia

Plan du cours

1. Manipuler plusieurs écrans
2. Communiquer avec l'utilisateur
3. Lire et enregistrer des données multimédia

Manipuler plusieurs écrans

Deux étapes importantes pour manipuler plusieurs écrans dans une application :

1. Ajouter un écran à l'application (vue + contrôleur de vue)
2. Ajouter des transitions (et transférer des données)

Ajouter un écran à l'application

Ajouter un nouveau contrôleur de vue

Chaque application iOS se décompose en une série d'écrans

Pour chaque écran, on implémente un contrôleur qui gère un écran et peut être chainé à d'autres

—> **Chaque vue à son contrôleur de vue**

Le storyboard dans Xcode facilite la visualisation de l'enchainement entre les différents écrans

Ajouter un écran à l'application

Ajouter un nouveau contrôleur de vue

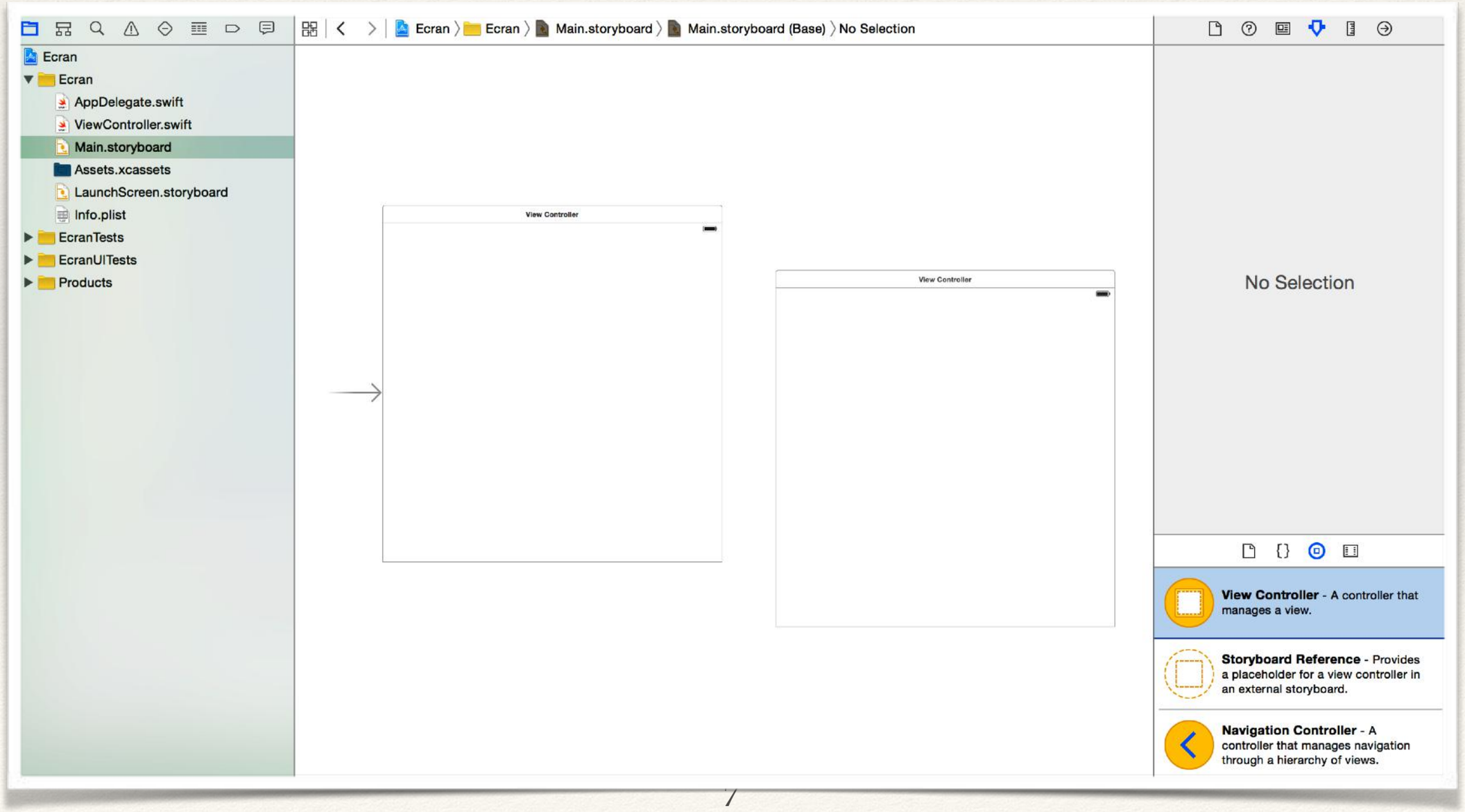
3 étapes nécessaires à l'ajout d'un nouveau contrôleur de vue :

1. ajouter l'élément graphique au storyboard
2. ajouter le fichier qui va contrôler cette vue
3. faire le lien entre l'élément graphique et les fichiers

Ajouter un écran à l'application

Ajouter un nouveau contrôleur de vue

L'ajout d'un nouvel objet de type `UIViewController` se fait par glisser-déposer de la bibliothèque d'objets vers le storyboard



Ajouter un écran à l'application

Création du nouveau fichier

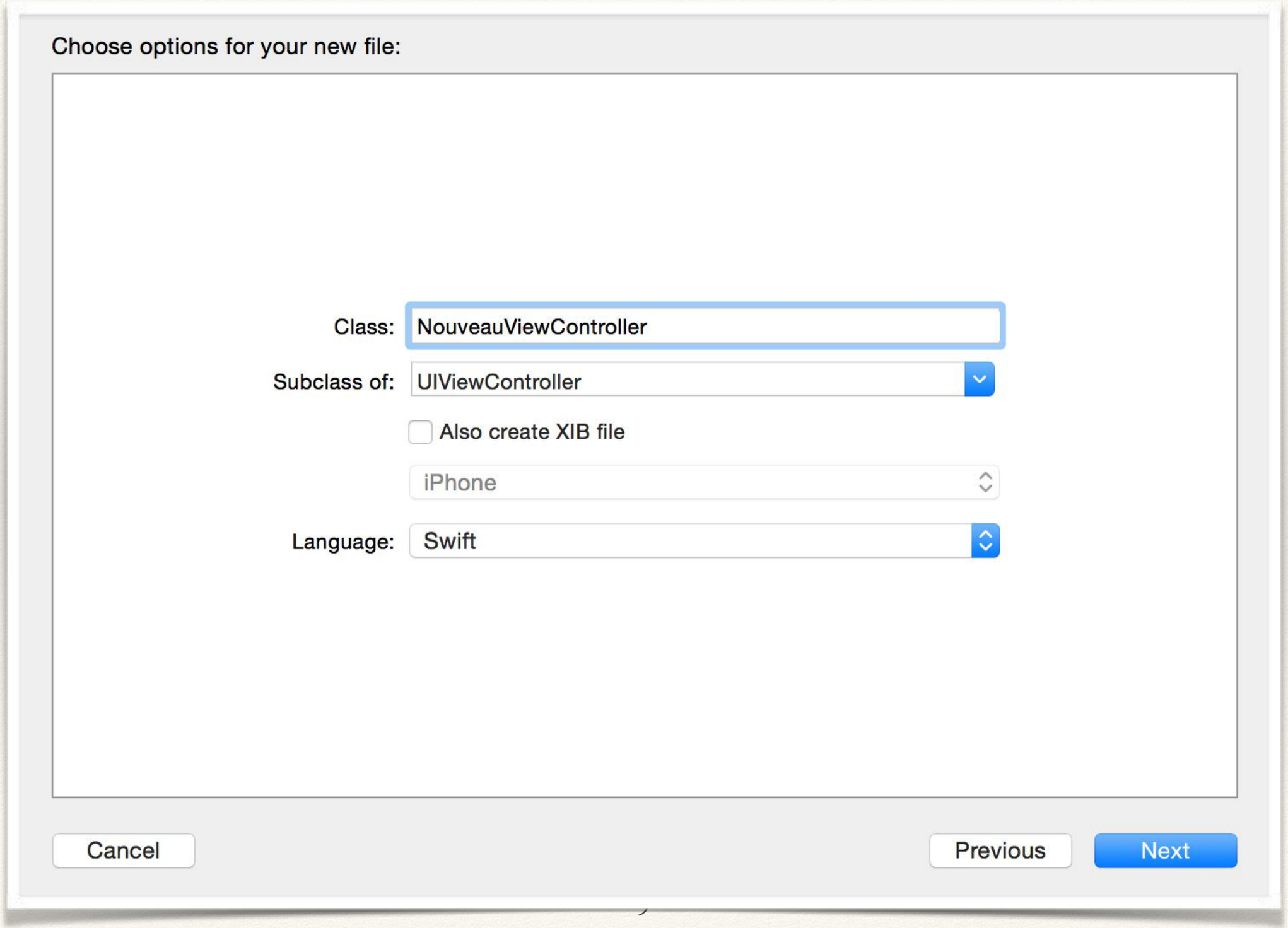
Xcode permet de créer rapidement un nouveau contrôleur de vue

Dans le menu *File*, *New File*, sélectionner *Cocoa Touch classes* et l'option subclass of *UIViewController*

Xcode ajoute un nouveau fichier d'implémentation d'une nouvelle classe dérivant de `UIViewController`

Ajouter un écran à l'application

Création du nouveau fichier



NouveauViewController.swift

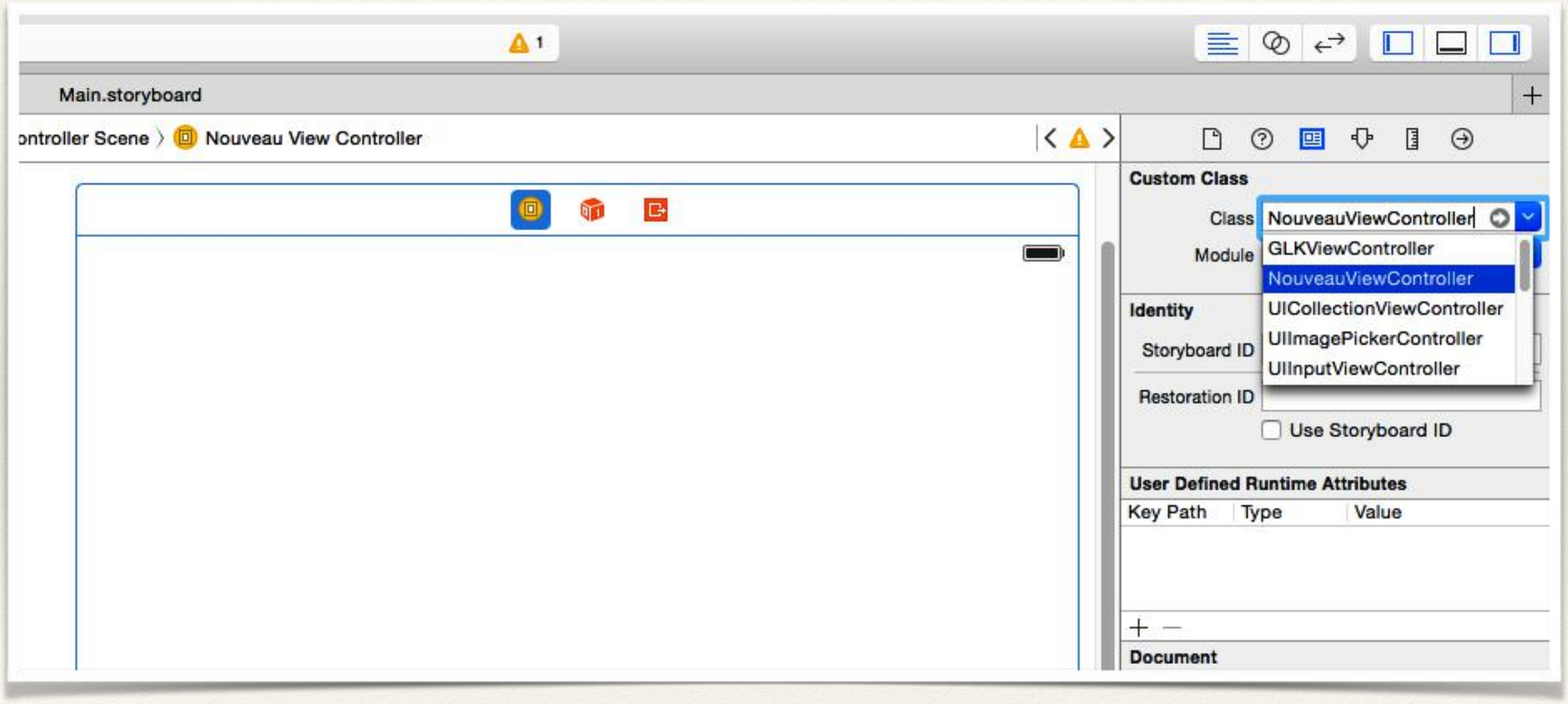
Ajouter un écran à l'application

Création du nouveau fichier

```
import UIKit

class NouveauViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little
preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destinationViewController.
        // Pass the selected object to the new view controller.
    }
    */
}
```

Ajouter un écran à l'application

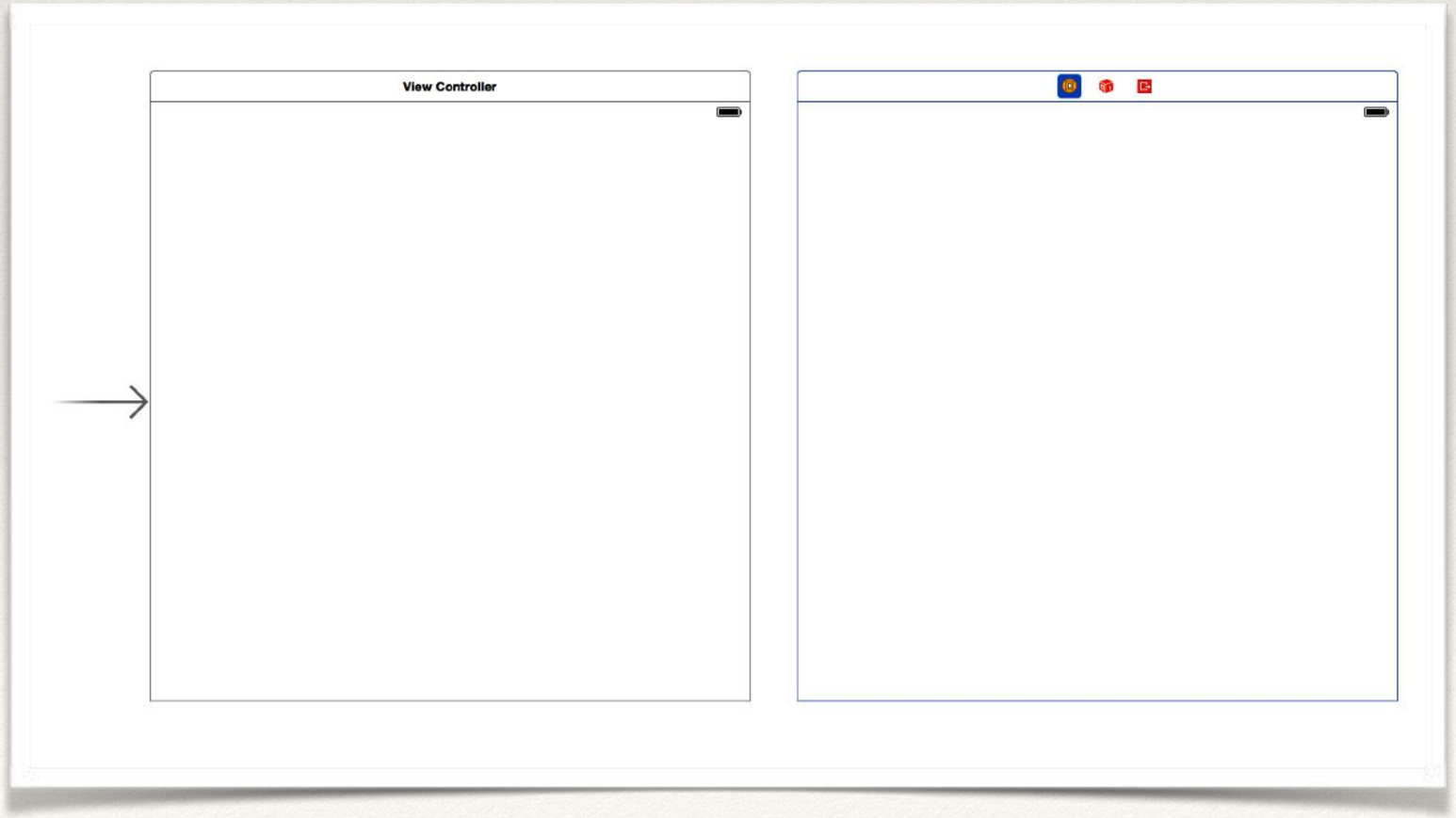
Création du lien entre l'interface et les fichiers



Ajouter des transitions et transférer des données

Les transitions entre les vues sont représentées par des flèches

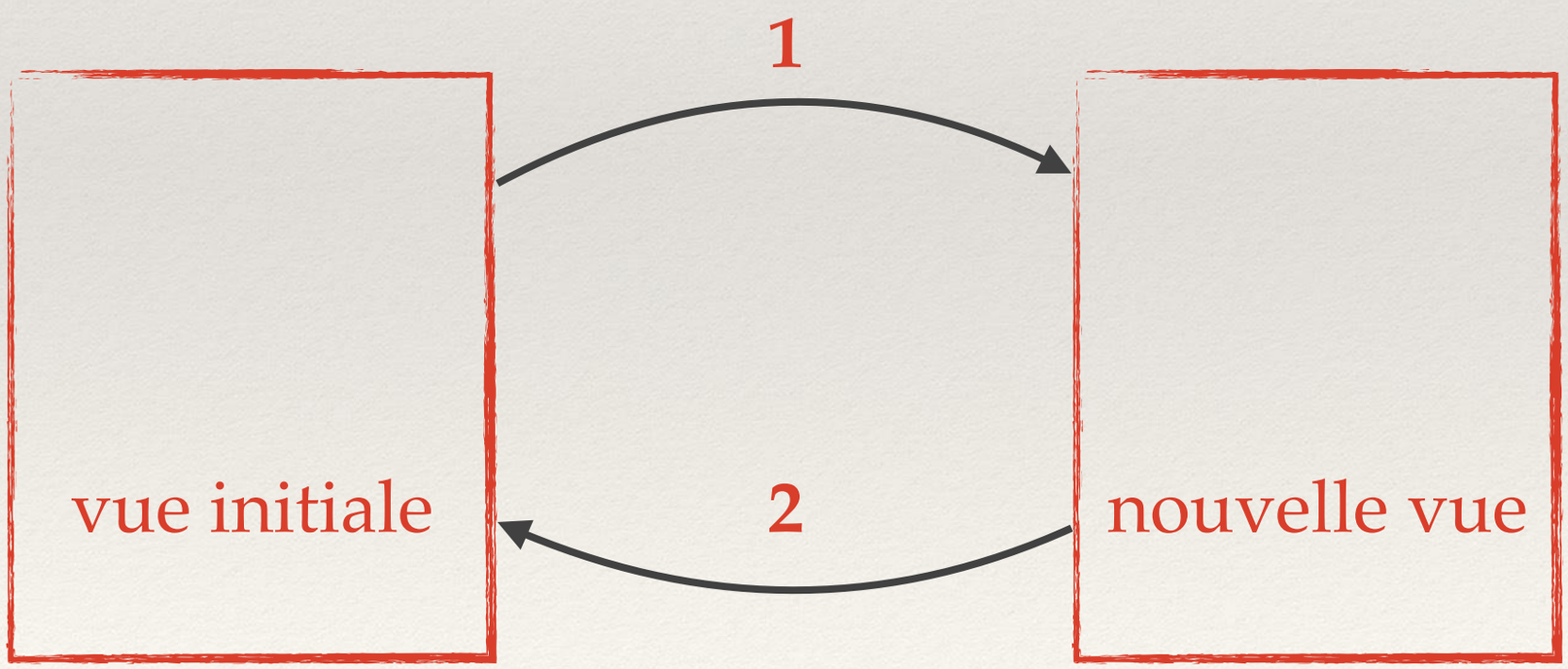
La première flèche entrante symbolise l'appel du programme au storyboard



Ajouter des transitions et transférer des données

Deux types de transitions :

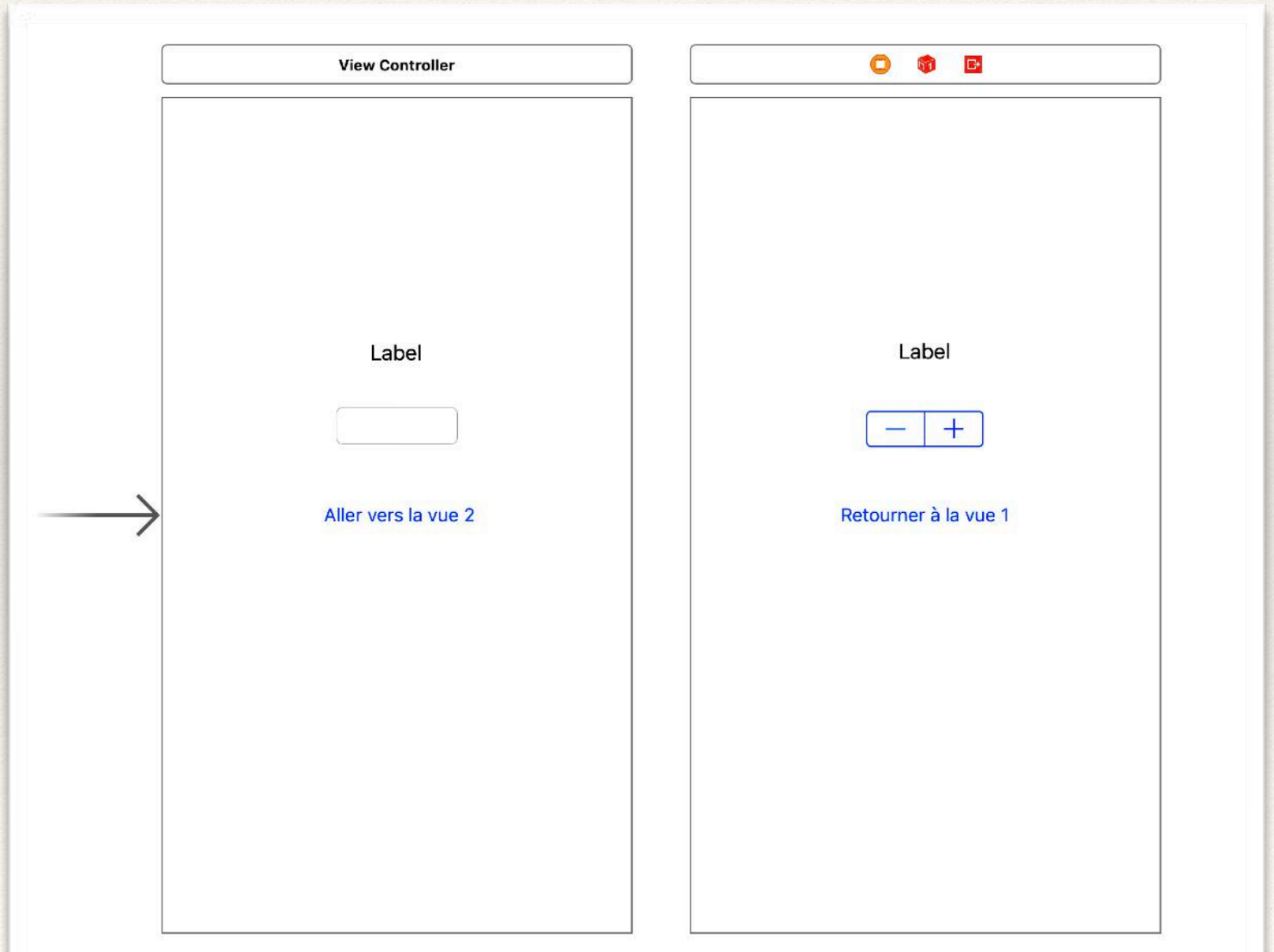
1. Une transition pour aller vers une nouvelle vue
2. Une transition pour revenir à la vue initiale



Ajouter des transitions et transférer des données

Exemple

Application avec deux contrôleurs de vue



Ajouter des transitions et transférer des données

Exemple

ViewController.swift

```
import UIKit

class ViewController: UIViewController {
    var valeurVue1 : Int = 0
    @IBOutlet weak var labelVue1: UILabel!
    @IBOutlet weak var champTextVue1: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    @IBAction func actionBoutonVue1(_ sender: Any) {
        labelVue1.text = champTextVue1.text
        valeurVue1 = Int(champTextVue1.text!)!
    }
}
```

Ajouter des transitions et transférer des données

Exemple `NouveauViewController.swift`

```
import UIKit

class NouveauViewController: UIViewController {
    var valueVue2 : Int = 0
    @IBOutlet weak var labelVue2: UILabel!
    @IBOutlet weak var stepperVue2: UIStepper!

    override func viewDidLoad() {
        super.viewDidLoad()
        labelVue2.text = String(valueVue2)
        stepperVue2.minimumValue = Double(valueVue2)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    @IBAction func actionStepperVue2(_ sender: Any) {
        labelVue2.text = String(stepperVue2.value)
        valeurVue2 = Int(stepperVue2.value)
    }
}
```

Ajouter des transitions et transférer des données

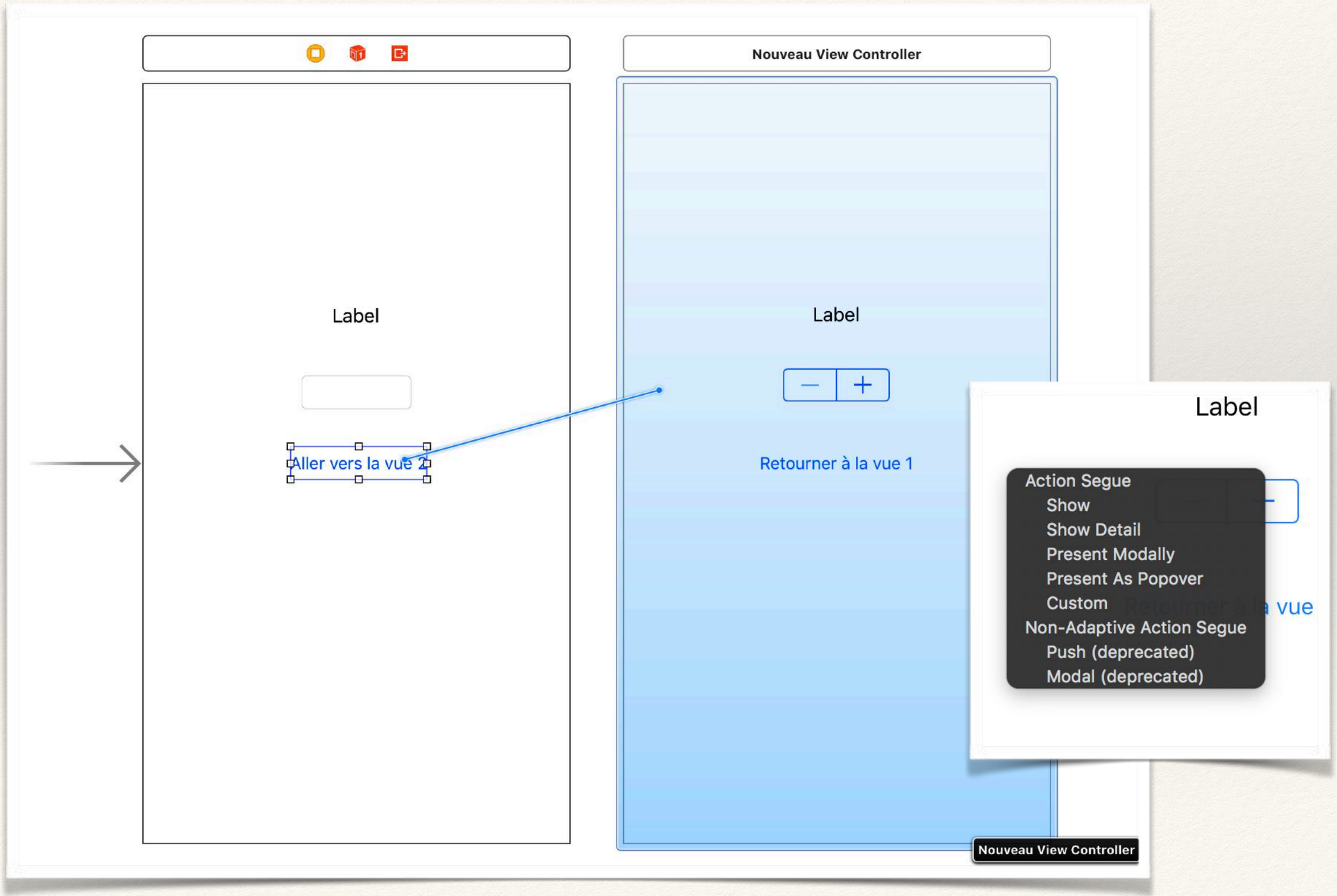
Aller vers une nouvelle vue

Pour aller vers une nouvelle vue

1. Créer la transition (**le segue**) via le storyboard
2. Identifier la transition
3. Implémenter la méthode appelée lors de la transition

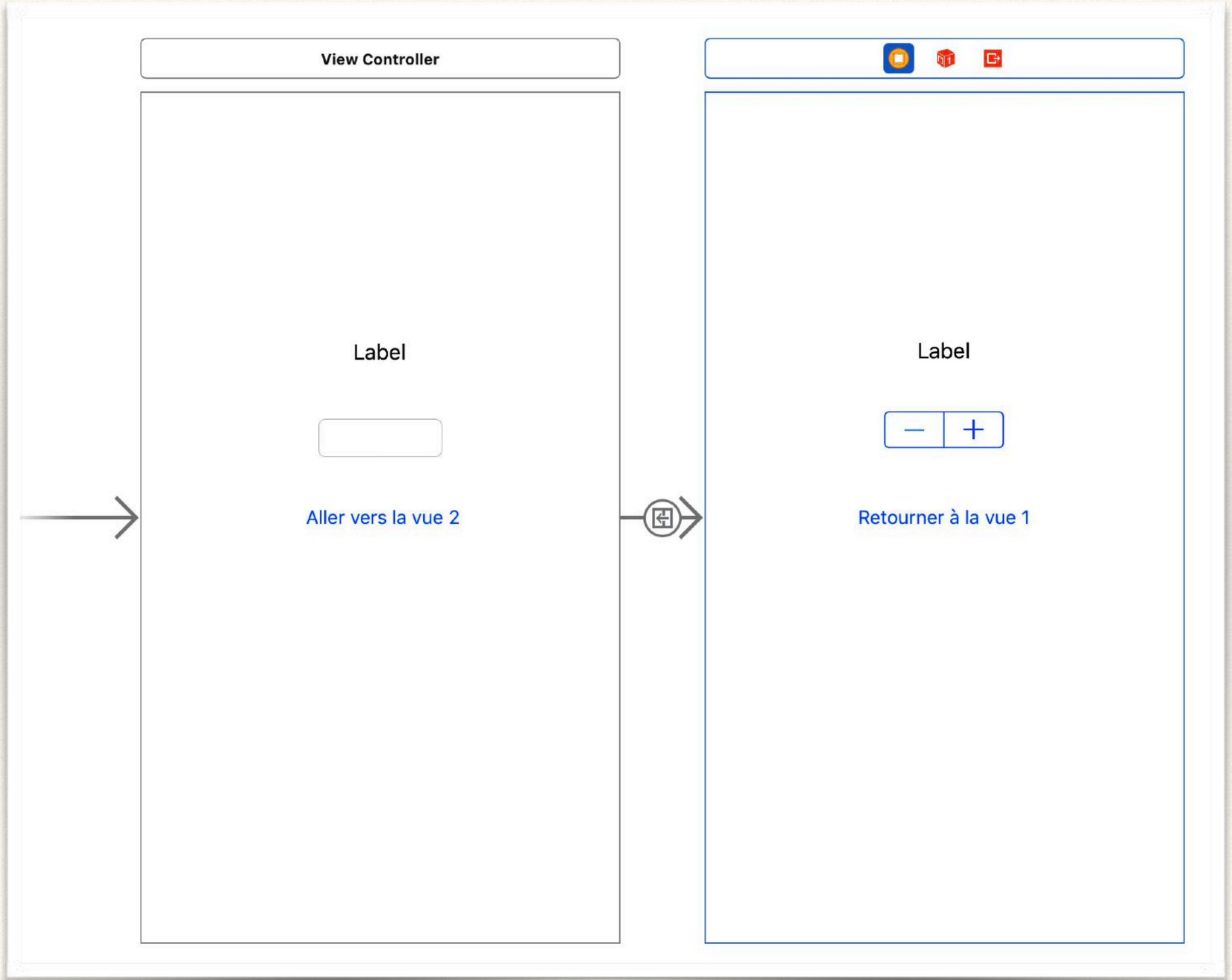
Ajouter des transitions et transférer des données

Aller vers une nouvelle vue : 1 - créer la transition



Ajouter des transitions et transférer des données

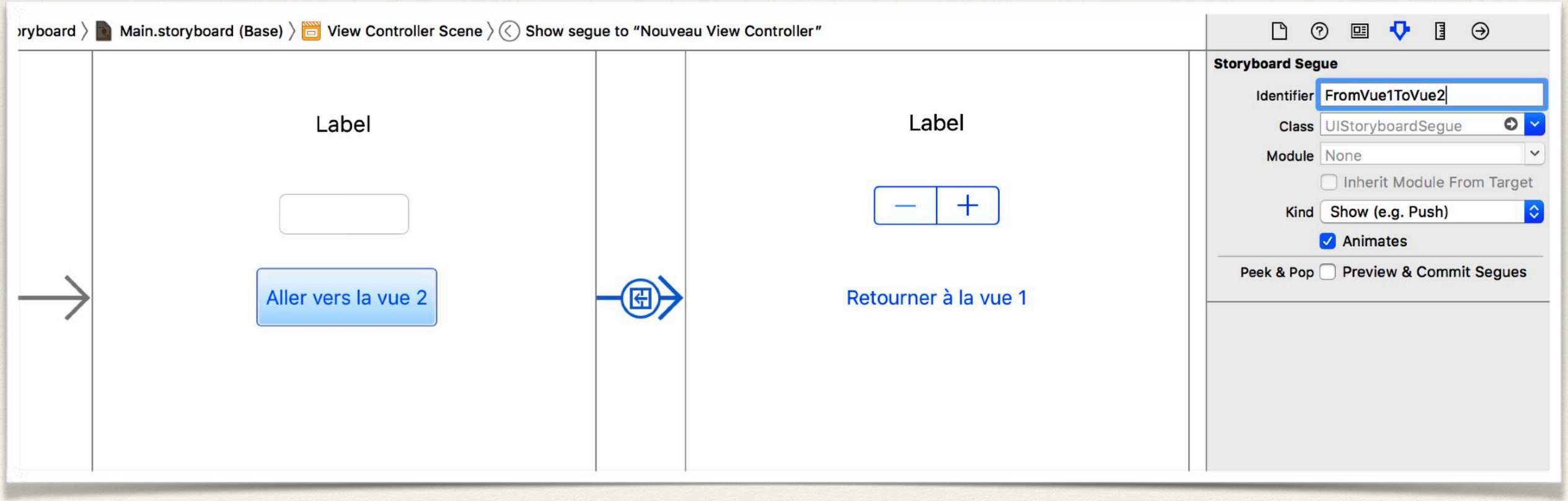
Aller vers une nouvelle vue : 1 - créer la transition



Ajouter des transitions et transférer des données

Aller vers une nouvelle vue : 1 - identifier la transition

L'identification du segue se fait via le storyboard



Ajouter des transitions et transférer des données

Aller vers une nouvelle vue : 3 - implémenter la méthode appelée lors de la transition

La méthode `prepare(for segue: UIStoryboardSegue, sender: Any?)` permet d'implémenter les actions à mettre en place lors du changement de la vue
—> Utile pour le transfert des données

Sa définition se trouve dans le fichier de la vue d'origine (ici le fichier `ViewController.swift`)

Ajouter des transitions et transférer des données

Aller vers une nouvelle vue : 3 - implémenter la méthode appelée lors de la transition

```swift
override func prepare(for segue: UIStoryboardSegue,
                                                sender: Any?){
    if segue.identifier == "FromVue1ToVue2" {

        // Definition du controler de destinatation
        let destinationVC = segue.destination as!
                                        NouveauViewController

        // Transfert de données
        destinationVC.valeurVue2 = valeurVue1
    }
}
```

ViewController.swift

Ajouter des transitions et transférer des données

Retourner à la vue d'origine

Mécanisme **Unwind segue**

Permet de revenir vers le contrôleur de vue qui a créé le segue initial (`ViewController`)

1. Ajouter une méthode de type IBAction dans le fichier `ViewController.swift` (contrôleur de la vue initiale)
2. Écrire une méthode `prepare` dans le fichier `NouveauViewController.swift` (contrôleur de vue qu'on veut quitter)
3. Créer la transition dans le storyboard
4. Identifier la transition dans le storyboard

Ajouter des transitions et transférer des données

Retourner à la vue d'origine - 1

```swift
// Méthode appelée lors du retour vers le contrôleur de vue à l'origine du segue initial

@IBAction func retourDansLaVue1(segue : UIStoryboardSegue) {
    champTextVue1.text = ""
    labelVue1.text = String(valeurVue1)
}
```

ViewController.swift

Ajouter des transitions et transférer des données

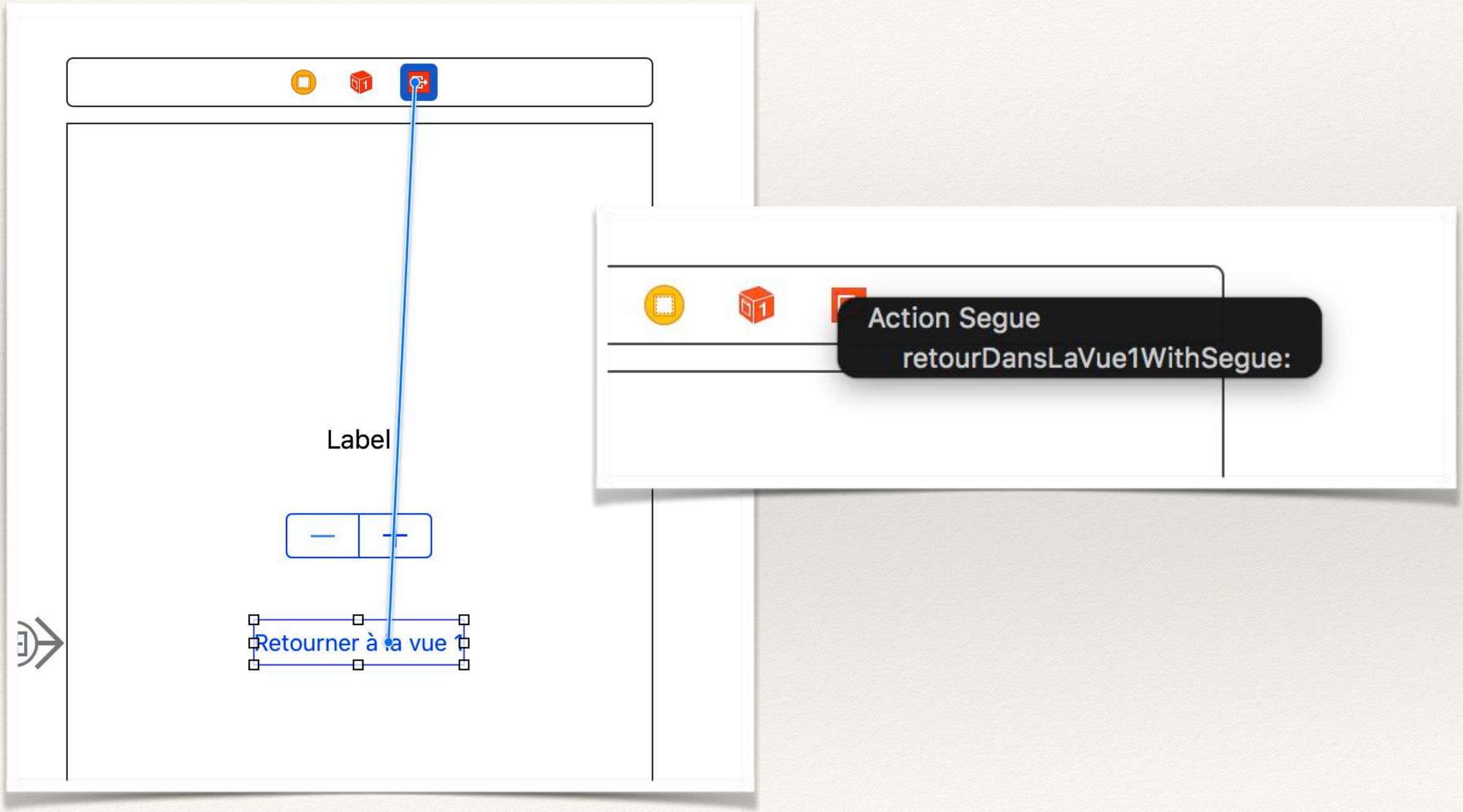
Retourner à la vue d'origine - 2

NouveauViewController.swift

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?){
    if segue.identifier == "FromVue2ToVue1" {

        // Definition du controler de destinatation
        let destinationVC = segue.destination as! ViewController

        // Transfert de données
        destinationVC.valeurVue1 = valeurVue2
    }
}
```

Ajouter des transitions et transférer des données

Retourner à la vue d'origine - 3



Ajouter des transitions et transférer des données

Retourner à la vue d'origine - 4

Ajouter des transitions et transférer des données

Bilan

Dans le storyboard:

- Créer et identifier deux transitions : `segue` et `unwind segue`

Dans le code :

- Écrire une méthode `prepare` et une méthode retournant une `IBAction` dans le fichier `ViewController.swift`
- Écrire une méthode `prepare` dans le fichier `NouveauViewController.swift`

Assembler les écrans de l'application

L'utilisateur d'un iPhone ou d'un iPad s'attend à trouver les mêmes règles d'ergonomie dans toutes les applications
—> aide en ligne ou mode d'emploi rares

Le SDK fournit les composants logiciels permettant d'implémenter les principes ergonomiques de l'interface Apple

Assembler les écrans de l'application

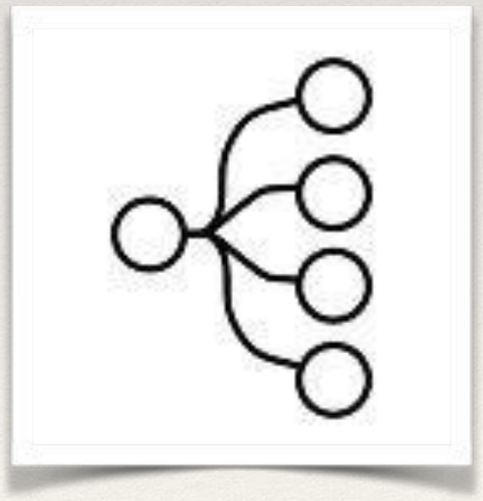
Design Pattern d'interface iOS - Listes hiérarchiques

Navigation dans des listes hiérarchiques

- Une liste hiérarchique est un arbre dont les noeuds sont des vues
- L'utilisateur ne voit que le noeud courant
- Le noeud racine de la liste est toujours le plus à gauche

Ce design pattern repose sur trois éléments

1. une barre de navigation en haut de l'écran
2. une vue sous la barre de navigation
3. les animations qui passent d'une vue à l'autre



Assembler les écrans de l'application

Design Pattern d'interface iOS - Listes hiérarchiques

La barre de navigation

Elle est divisée en trois éléments

1. Un bouton en haut à gauche qui revient à la vue parente (pas affiché s'il n'y a pas de vue parente)
2. Un titre décrivant la vue actuelle
3. Un bouton contextuel à droite

La vue de contenu

C'est le noeud que l'utilisateur est en train de consulter.

Si ce noeud à des fils, ils sont souvent représentés par une nouvelle liste d'éléments que l'utilisateur peut sélectionner

Assembler les écrans de l'application

Design Pattern d'interface iOS - Listes hiérarchiques

Principes de navigation à respecter

- L'utilisateur doit toujours pouvoir revenir en arrière
- L'emplacement en haut à gauche est réservé au bouton retour
- Les animations montrent à l'utilisateur la conséquence de son action sur un élément de la liste

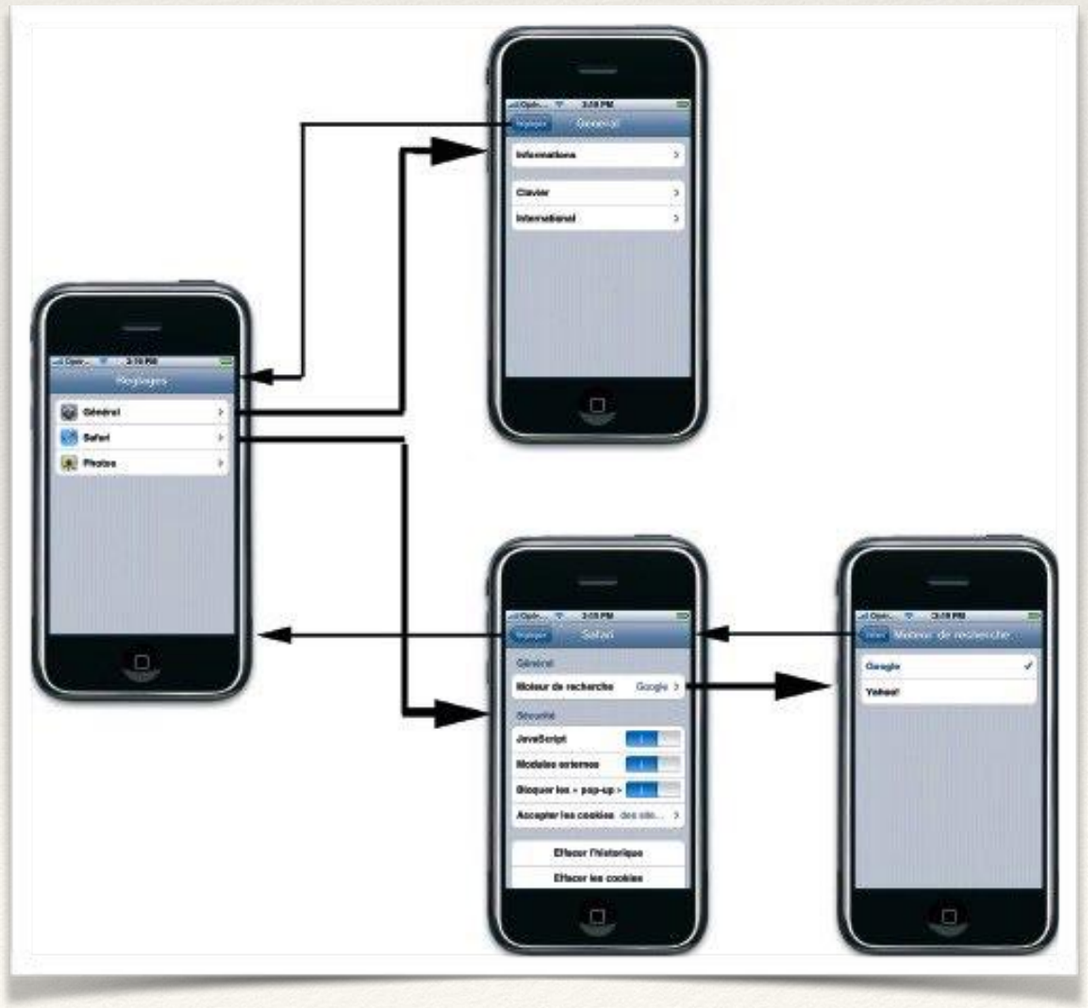
Assembler les écrans de l'application

Le contrôleur de navigation

La classe `UINavigationController` fournit toute la logique nécessaire pour mettre en place une navigation dans une liste hiérarchique

Elle prend en charge :

- l'affichage de la barre de navigation (titre et boutons)
- les animations entre les écrans



Assembler les écrans de l'application

Le contrôleur de navigation

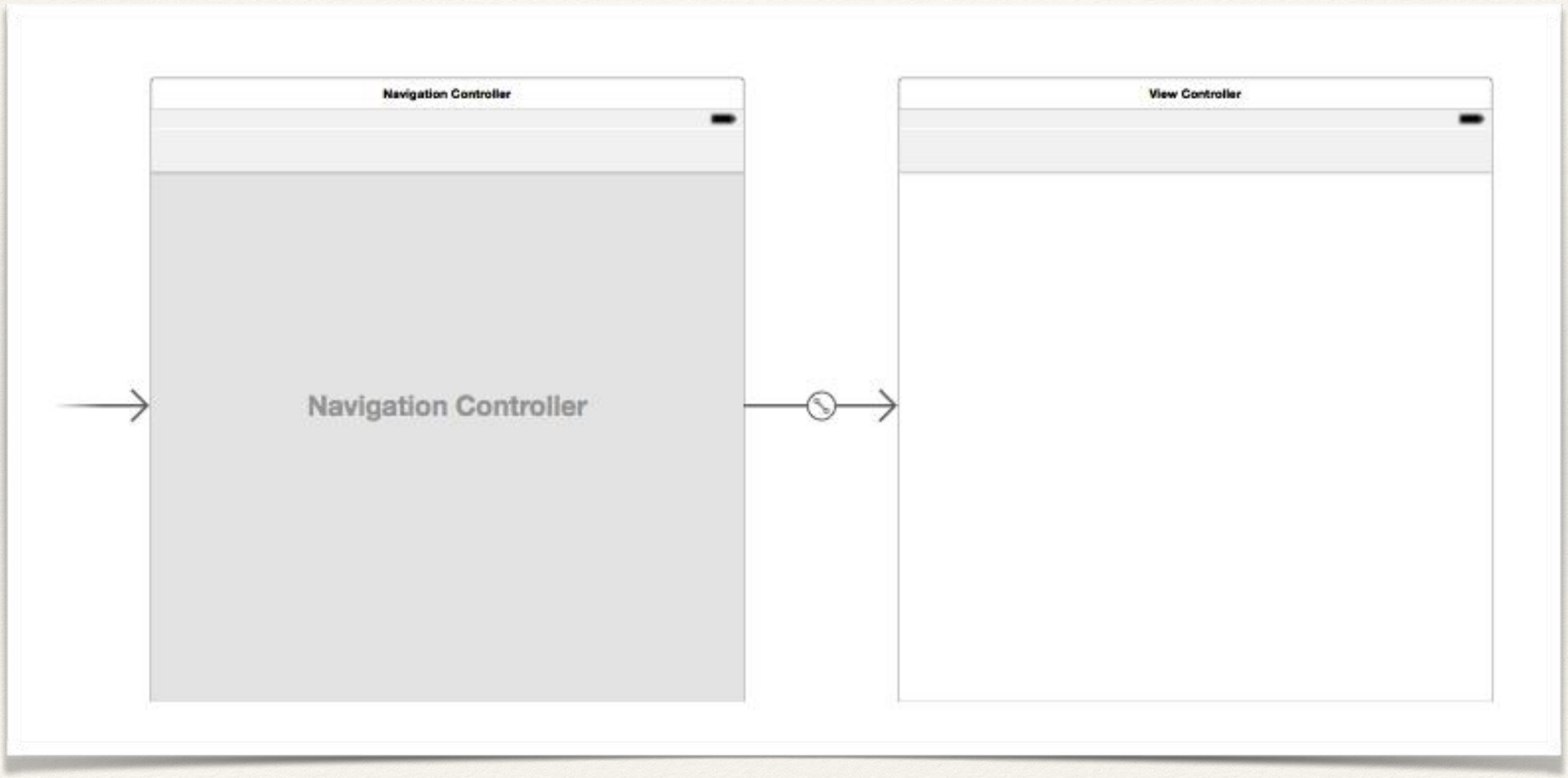
Ajouter un contrôleur de navigation à votre application

1. Sélectionnez une vue dans votre storyboard
2. Dans le menu « Editor », cliquez sur « Embed in » puis sur « Navigation controller »

—> Un « Navigation controller » doit apparaitre dans votre storyboard et se placer devant votre vue

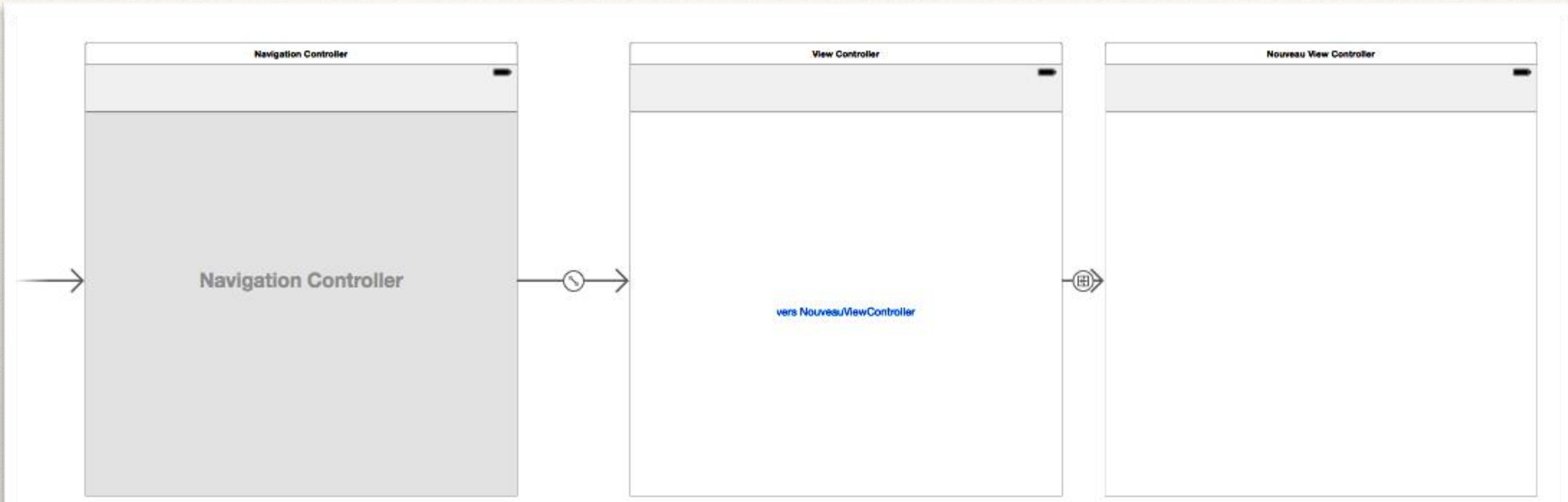
Assembler les écrans de l'application

Le contrôleur de navigation

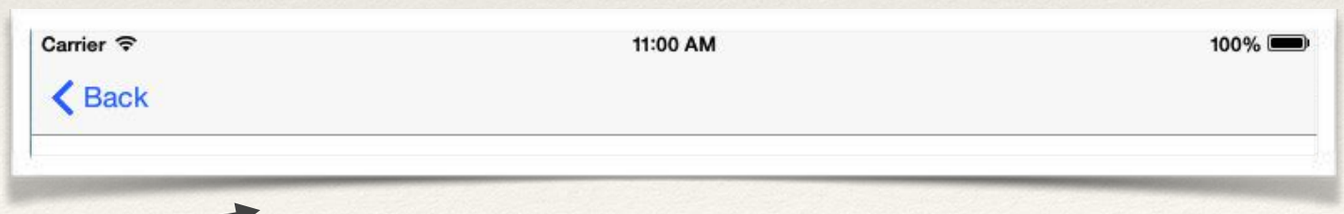


Assembler les écrans de l'application

Le contrôleur de navigation



Lors du lancement de l'application



Assembler les écrans de l'application

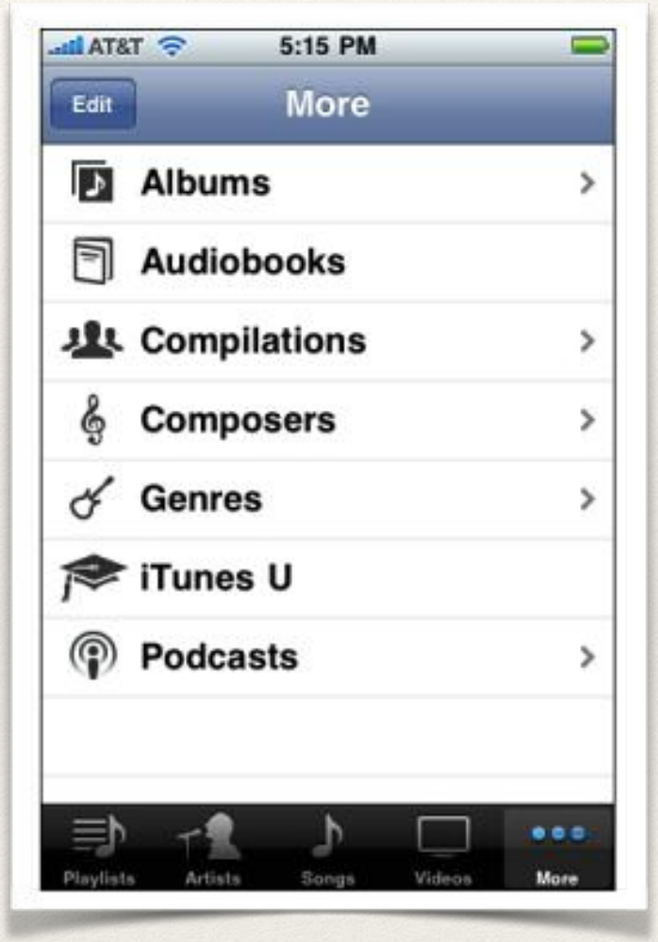
Design Pattern d'interface iOS - Onglet

Navigation par onglet

Permet de mettre au même niveau plusieurs fonctionnalités équivalentes (iTunes, Horloge)

La barre d'onglets doit respecter les principes suivants :

- L'application ne doit jamais forcer un changement d'onglet
- Quand l'utilisateur change d'onglet et revient au précédent, il doit retrouver la même vue



Les onglets servent uniquement à passer d'une vue à l'autre

La barre d'onglets affiche au plus cinq icônes

Assembler les écrans de l'application

Le contrôleur d'onglets

La classe `UIBTabBarController` gère

- la barre d'onglets affichée en bas de l'écran
- l'apparition de l'onglet « Autre »
- la personnalisation de la barre d'onglets

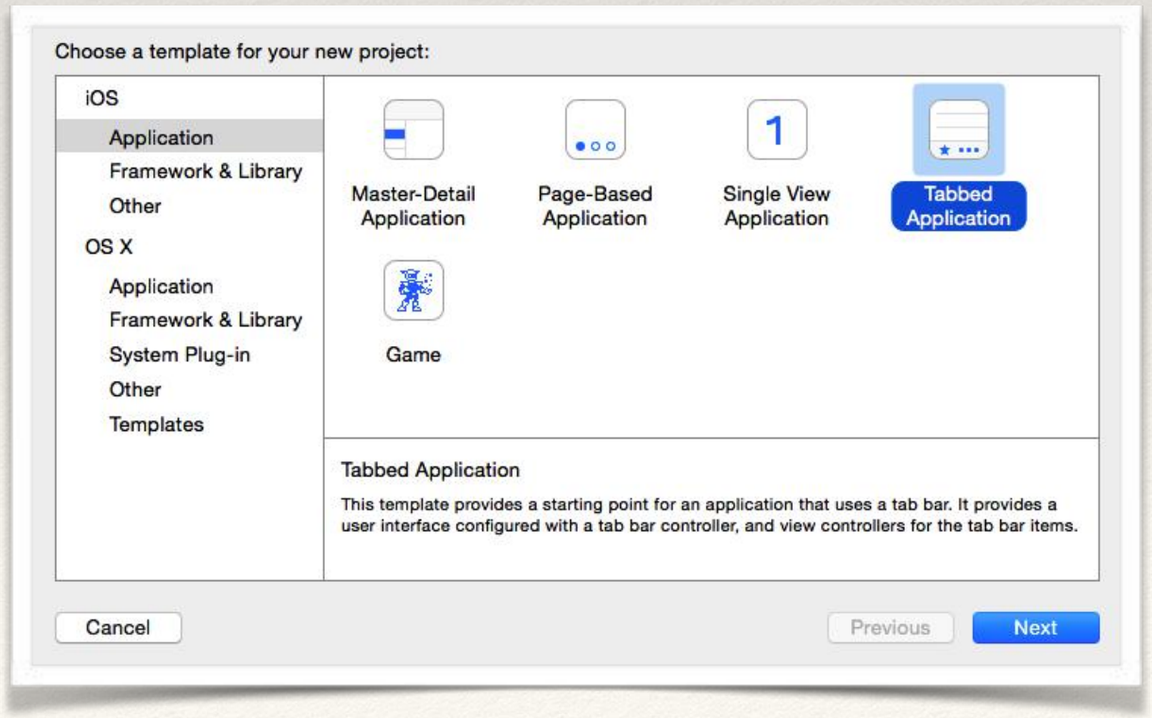
Assembler les écrans de l'application

Le contrôleur d'onglets

Création d'un contrôleur d'onglets dans votre application

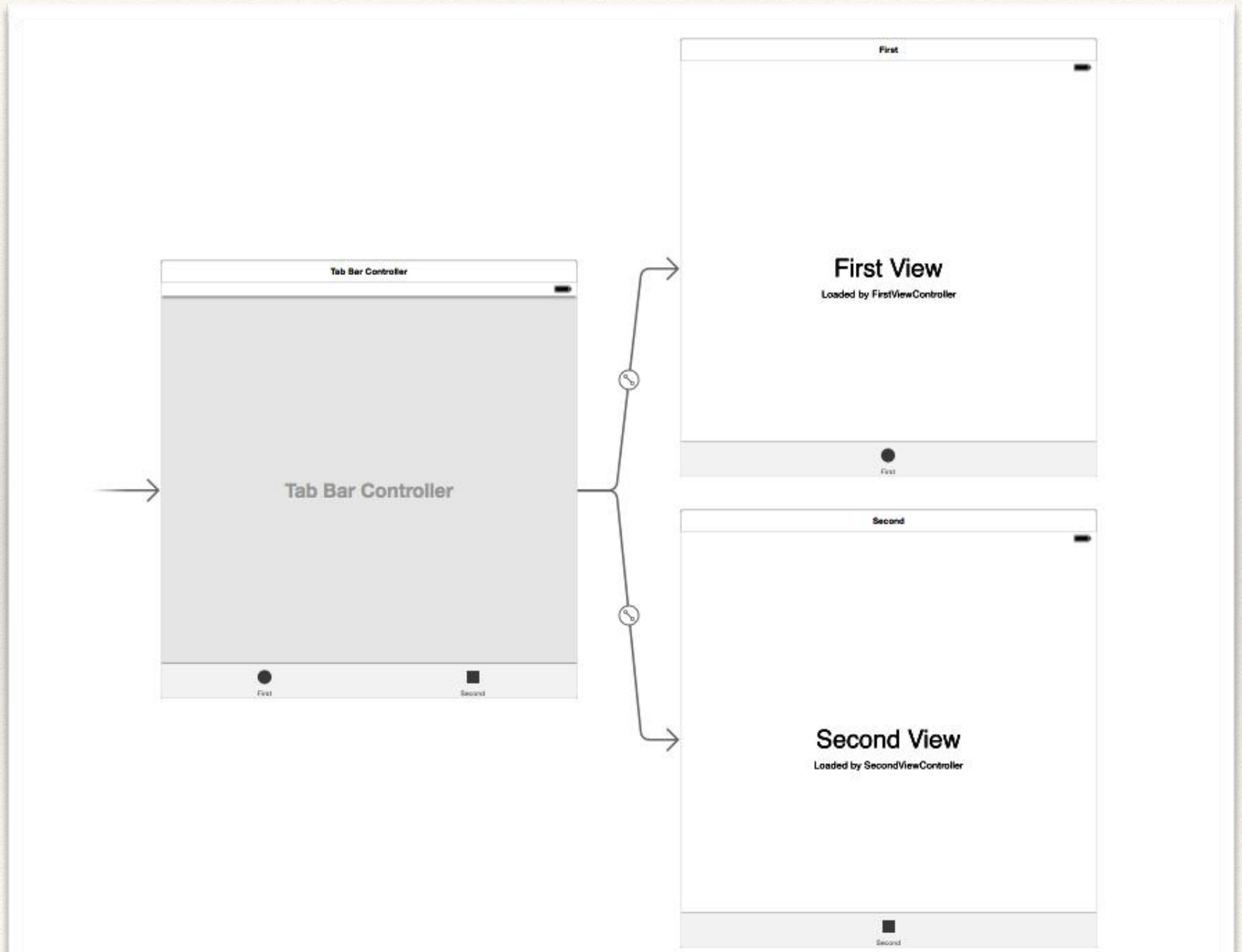
Utilisation d'un template « Tabbed Application » lors de la création d'un projet

—> Création de deux onglets



Assembler les écrans de l'application

Le contrôleur d'onglets

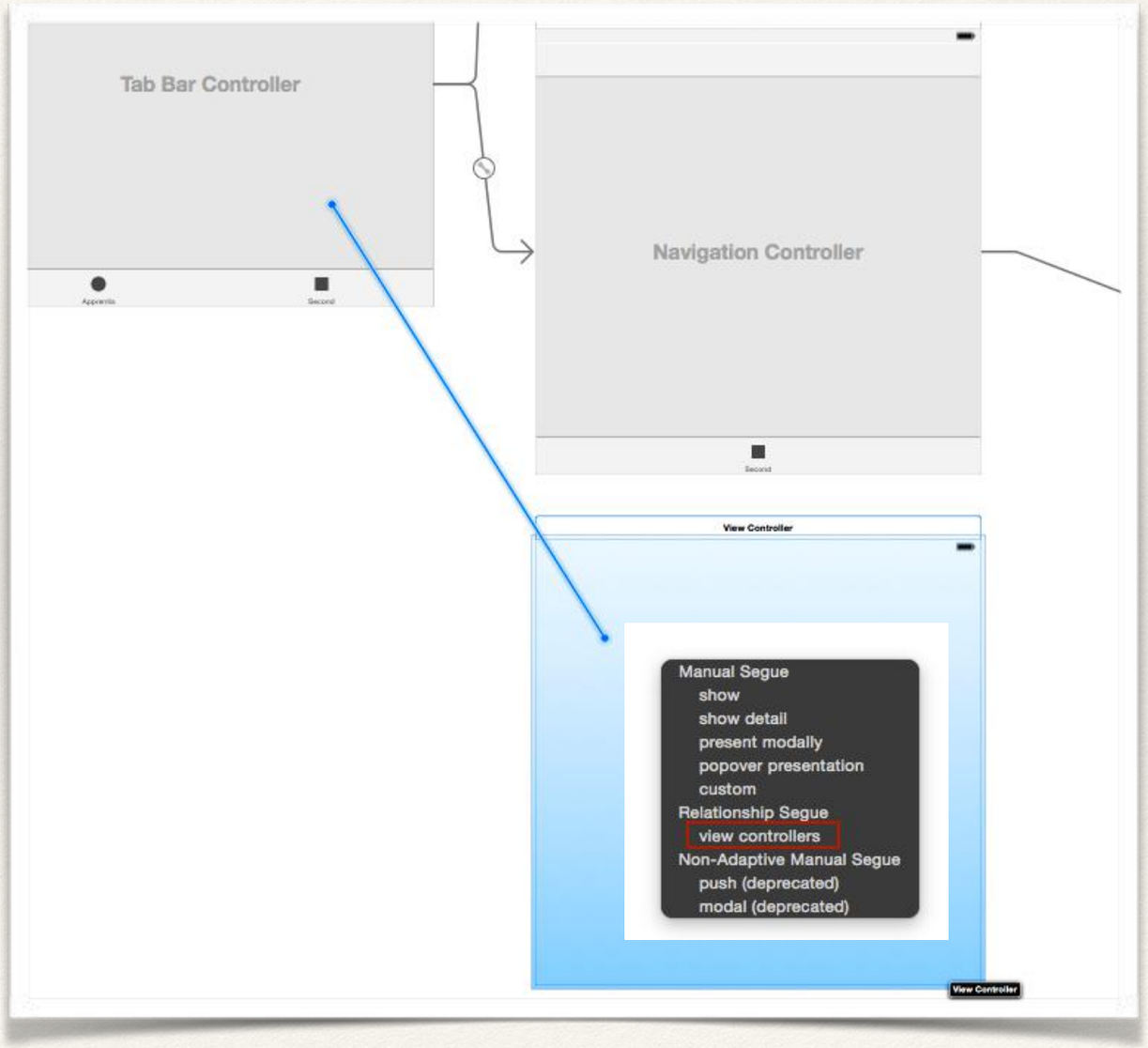


Assembler les écrans de l'application

Le contrôleur d'onglets

Ajout d'un onglet supplémentaire :

1. Ajouter un nouveau contrôleur de vue (cf diapo 6)
2. Clic-droit enfoncé du « Tab Bar Controller » vers le nouveau viewController, sélectionner « Relationship segue - viewController »



Assembler les écrans de l’application

Design Pattern d’interface iOS - Combinaison

La barre d’onglet est souvent utilisée en combinaison avec les listes hiérarchiques

—> Une liste hiérarchique est contenue dans un onglet

⚠️ Ne jamais intégrer une barre d’onglets comme élément dans une liste hiérarchique

Plan du cours

1. Manipuler plusieurs écrans
2. Communiquer avec l'utilisateur
3. Lire et enregistrer des données multimédia

Plan du cours

1. Manipuler plusieurs écrans
2. Communiquer avec l'utilisateur
3. Lire et enregistrer des données multimédia

Envoi de messages

Frameworks MessageUI et Social

Frameworks `MessageUI` ~~et `Social`~~

- Envoi de mail : `MFMailComposeViewController`
- Envoi de SMS : `MFMessageComposeViewController`
- ~~Envoi de Post : `SLComposeViewController`~~
- Envoi d'alerte : `UIAlertController`

Envoi de messages

Mail - MFMailComposeViewController

L'envoi d'un mail se fait en trois étapes

1. Ajout du framework et du délégué
2. Configuration du message
3. Présentation de l'interface d'envoi

Il est nécessaire d'implémenter le protocole `MFMailComposeViewControllerDelegate` et d'ajouter le framework `MessageUI` au fichier `.swift`

Envoi de messages

Mail - MFMailComposeViewController

La configuration du message se fait grâce aux méthodes

- `setSubject:`
- `setToRecipients:  / setCcRecipients: / setBccRecipients:`
- `setMessageBody:isHTML:`
- `addAttachmentData:mimeType:fileName:`

Le deuxième paramètre est une chaîne de caractères qui précise le type de données à transmettre

Exemple : `"image/png"`, `"text/html"`

Envoi de messages

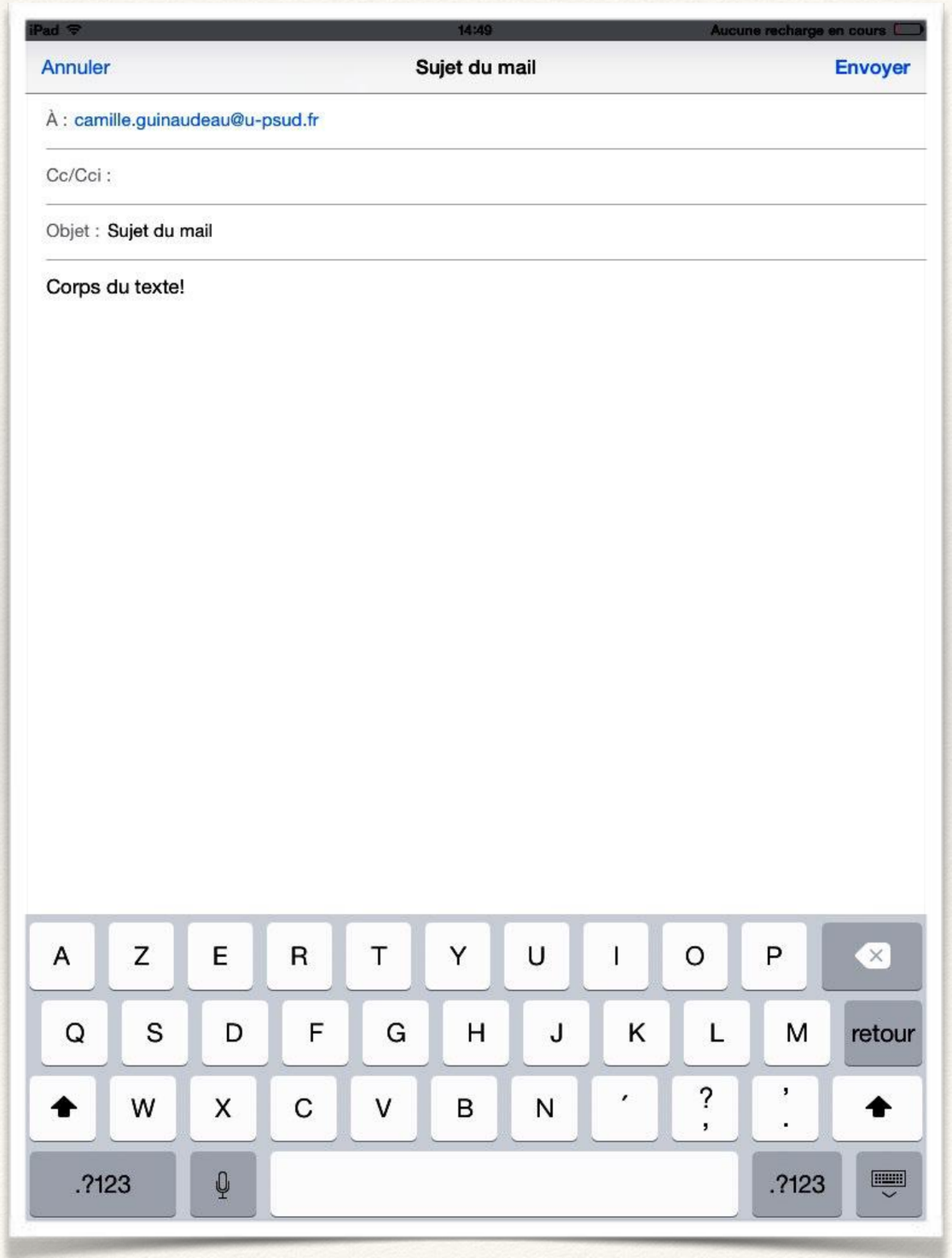
Mail - MFMailComposeViewController

```swift
import UIKit
import MessageUI

class ViewController: UIViewController, MFMailComposeViewControllerDelegate {

    @IBAction func envoiMail(_ sender: Any) {
        if MFMailComposeViewController.canSendMail() {
            let mail = MFMailComposeViewController()
            mail.mailComposeDelegate = self
            mail.setToRecipients(["camille.guinaudeau@u-psud.fr"])
            mail.setSubject("Sujet du mail")
            mail.setMessageBody("Corps du texte!", isHTML: true)

            present(mail, animated: true)
        } else {
            // show failure alert
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

ViewController.swift

Envoi de messages

Mail - MFMailComposeViewController



Envoi de messages

Mail - MFMailComposeViewController

```
func mailComposeController(_ controller:
            MFMailComposeViewController, didFinishWith result:
            MFMailComposeResult, error: Error?) {
    switch result {
    case MFMailComposeResult.sent:
        print("Mail envoyé")
    case MFMailComposeResult.saved:
        print("Mail sauvergadé")
    case MFMailComposeResult.cancelled:
        print("Mail annulé")
    case MFMailComposeResult.failed:
         print("Erreur lors de l'envoi du mail")
    }
    dismiss(animated: true, completion: nil)
}
```

ViewController.swift

Envoi de messages

SMS - MFMessageComposeViewController

Comme pour les mails, la configuration d'un SMS se fait grâce aux méthodes :

- `setRecipients:`
- `setBody:`
- `setSubject:`

Il est nécessaire d'implémenter le protocole `MFMessageComposeViewControllerDelegate` et d'ajouter le framework `MessageUI` au fichier `.swift`

Envoi de messages

SMS

```swift
import UIKit
import MessageUI

class ViewController: UIViewController, MFMessageComposeViewControllerDelegate {

    @IBAction func envoiMessage(_ sender: Any) {
        if MFMessageComposeViewController.canSendText() {
            let message = MFMessageComposeViewController()
            message.messageComposeDelegate = self
            message.recipients = ["0123456789", "0987654321"]
            message.body = "Corps du message!"
            if MFMessageComposeViewController.canSendSubject() {
                message.subject = "Sujet du message"
            }
            present(message, animated: true)
        } else {
            // show failure alert
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

ViewController.swift

Envoi de messages

SMS

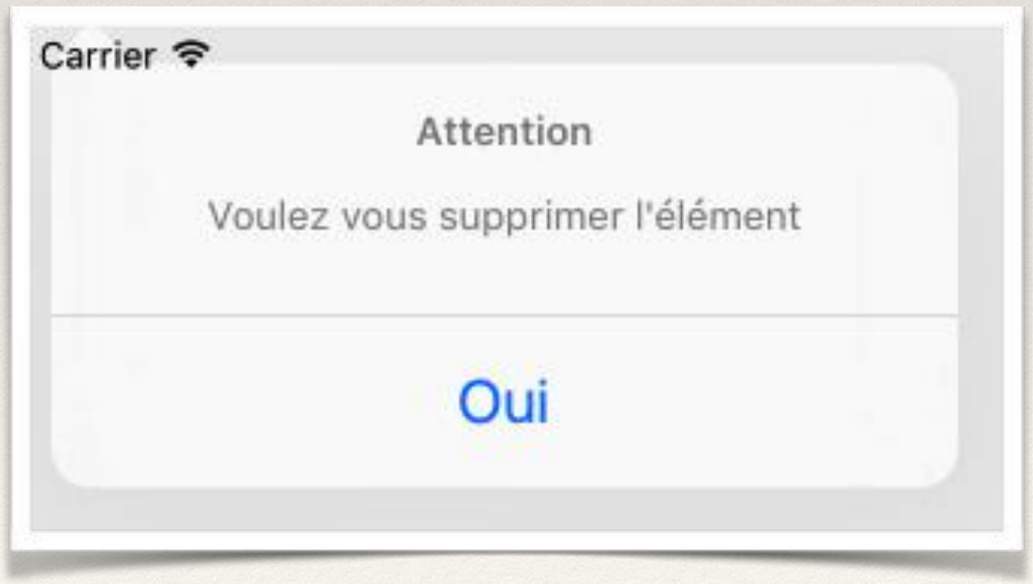
```
func messageComposeViewController(_ controller:
          MFMessageComposeViewController, didFinishWith result:
          MessageComposeResult) {
    switch result {
    case MessageComposeResult.sent:
        print("Message envoyé")
    case MessageComposeResult.cancelled:
        print("Message annulé")
    case MessageComposeResult.failed:
        print("Erreur lors de l'envoi du message")
    }
    controller.dismiss(animated: true, completion: nil)
}
```

ViewController.swift

Message d'alerte

UIAlertController

La classe `UIAlertController` permet d'afficher un message à l'utilisateur, pour confirmer une action par exemple ou lui fournir une information ponctuelle



Message d'alerte

UIAlertController

```
@IBAction func createAlert(_ sender: Any) {
    let alert = UIAlertController(title: "Attention",
                  message: "Voulez vous supprimer l'élément",
                  preferredStyle: .alert)
    alert.addAction(UIAlertAction(title: "Oui",
                                  style: .default) { action in
        // Implémenter la suppression de l'élément
    })
    alert.addAction(UIAlertAction(title: "Annuler",
                                  style: .cancel) { action in
        self.dismiss(animated: true, completion: nil)
     })
     alert.popoverPresentationController?.sourceView = self.view
     self.present(alert, animated: true, completion: nil)
}
```

ViewController.swift

Message d'alerte

UIAlertController

```
@IBAction func createAlert(_ sender: Any) {
    let alert = UIAlertController(title: "Attention",
                  message: "Voulez vous supprimer l'élément",
                  preferredStyle: .actionSheet)
    alert.addAction(UIAlertAction(title: "Oui",
                                  style: .default) { action in
        // Implémenter la suppression de l'élément
    })
     alert.popoverPresentationController?.sourceView = self.view
     self.present(alert, animated: true, completion: nil)
}
```

ViewController.swift

Message d'alerte

UIAlertController - Ajouter un champ de texte

```swift
@IBAction func createAlert(_ sender: Any) {
    let alert = UIAlertController(title: "Entrer votre nom",
                                  message: nil, preferredStyle: .alert)
    alert.addTextField()

    let submitAction = UIAlertAction(title: "OK",
                                     style: .default) { action in
        let answer = alert.textFields![0]
        print(answer.text!)
    }

    alert.addAction(submitAction)

    alert.popoverPresentationController?.sourceView = self.view
    self.present(alert, animated: true, completion: nil)
}
```

ViewController.swift

Message d'alerte

UIAlertController - Ajouter un champ de texte

Plan du cours

1. Manipuler plusieurs écrans
2. Communiquer avec l'utilisateur
3. Lire et enregistrer des données multimédia

Plan du cours

1. Manipuler plusieurs écrans
2. Communiquer avec l'utilisateur
3. Lire et enregistrer des données multimédia

Le son

Formats audio

Format	Compression	Description
MP3	Oui	Format de compression audio avec perte
AAC	Oui	Advanced Audio Coding : compression audio avec pertes offrant une meilleure qualité que le MP3 à débit égal
ALAC	Oui	Apple Lossless Audio Codec : format de compression sans perte
IMA4	Oui	Format de compression avec perte —> il est facilement décodé par le processeur d'un appareil mobile
Linear PCM	Non	Format audio non compressé le plus répandu

Le son

Formats audio

Un seul fichier peut être décodé à la fois par le hardware de l'appareil
—> plusieurs fichiers c'est la CPU qui fait le décodage (impact sur les performances de l'application)
—> privilégier le format IMA4 ou format PCM

La conversion peut s'effectuer avec l'outil `afconvert` disponible sur tous les ordinateurs Mac sous OS X

Exemple pour convertir au format AAC :

```
afconvert – f acc <fichier source> <fichier destination>
```

Le son

Lancer la lecture de son

La lecture de son dans une application s'appuie sur la classe `AVAudioPlayer` du framework `AVFoundation`

Cette classe permet d'initialiser le player avec l'URL d'un fichier audio et de régler le volume en lecture

Le protocole `AVAudioPlayerDelegate` permet d'implémenter les méthodes informant que la lecture s'est terminée

- avec succès `audioPlayerDidFinishPlaying`
  `(_ player: AVAudioPlayer, successfully flag: Bool)`
- avec une erreur `audioPlayerDecodeErrorDidOccur`
  `(_ player: AVAudioPlayer, error: Error?)`

Le son

Lancer la lecture de son

ViewController.swift

```
import UIKit
import AVFoundation

class ViewController: UIViewController, AVAudioPlayerDelegate {

    override func viewDidLoad() {
        super.viewDidLoad()
        let path : URL = NSURL.fileURL(withPath:Bundle.main.path(
                                        forResource: "son", ofType: "mp3")!)
        let player : AVAudioPlayer

        do {
            try player = AVAudioPlayer(contentsOf: path)
            player.prepareToPlay()
            player.play()
        }
        catch {
            print("Erreur lors de la création de l'audioPlayer")
        }
    }
}
```

Le son

Lancer la lecture de son

ViewController.swift

```
@IBAction func lancerMusique(_ sender: Any) {
    let path : URL = NSURL.fileURL(withPath:Bundle.main.path(
                        forResource: "son", ofType: "mp3")!)
    let player : AVAudioPlayer
    do {
        try player = AVAudioPlayer(contentsOf: path)
        if player.isPlaying {
            player.stop()
        }
        else {
            player.setVolume(2.5, fadeDuration: TimeInterval(10))
            player.prepareToPlay()
            player.numberOfLoops = 2
            player.play()
        }
    }
    catch {
       print("Erreur lors de la création de l'audioPlayer")
    }
}
```

La vidéo

Lecture de vidéos

La lecture de vidéo dans une application s'appuie sur les classes `AVPlayer` et `AVPLayerViewController` des frameworks `AVKit` et `AVFoundation`

Lorsque l'application lance la lecture, le lecteur s'affiche et joue la vidéo

Les fichiers .mov, .mp4 et .mpv peuvent être lus.

La vidéo

Intégrer le lecteur vidéo dans une application

Comme pour le son, il faut initialiser le `AVPlayerController` puis appeler la méthode `play`

La fenêtre du `AVPlayerController` doit également être ajoutée à la vue courante de l'application pour être visible

La vidéo

Intégrer le lecteur vidéo dans une application

ViewController.swift

```
import UIKit
import AVKit
import AVFoundation

class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func lectureVideo(_ sender: Any) {
        let movieURL : URL = NSURL.fileURL(withPath:Bundle.main.path(
                                           forResource: "video",
                                           ofType: "mp4")!)

        let player = AVPlayer(url:movieURL)

        let playerController = AVPlayerViewController()
        playerController.player = player
        self.showDetailViewController(playerController, sender: self)

        player.play()
    }
}
```

La vidéo

Détecter la fin de la lecture d'une video

Pas de protocole.

Pour connaitre les évènements liés au cycle de vie du lecteur de vidéo, il faut s'inscrire à des notifications.

Les principales notifications sont les suivantes :

- `AVPlayerItemDidPlayToEndTime`
- `AVPlayerItemFailedToPlayToEndTime`
- `AVPlayerItemNewAccessLogEntry:`
- `AVPlayerItemNewErrorLogEntry`
- `AVPlayerItemPlaybackStalled`
- `AVPlayerItemTimeJumped`

La vidéo

Détecter la fin de la lecture d'une video

ViewController.swift

```
import UIKit
import AVKit
import AVFoundation

class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func lectureVideo(_ sender: Any) {
        let movieURL : URL = NSURL.fileURL(withPath:Bundle.main.path(
                                            forResource: "video", ofType: "mp4")!)

        let player = AVPlayer(url:movieURL)

        NotificationCenter.default.addObserver(self,
                           selector:#selector(self.playerDidFinishPlaying(note:)),name:
          NSNotification.Name.AVPlayerItemDidPlayToEndTime, object: player.currentItem)
    }

    @objc func playerDidFinishPlaying(note: NSNotification){
        //Called when player finished playing
    }
}
```

Les photos

Prendre et utiliser des photos

La classe `UIImagePickerController` permet d'utiliser l'album photo et la caméra de l'appareil mobile

Elle permet de :

- vérifier la présence d'une caméra,
- d'initialiser l'objet `UIImagePickerController`
- de préciser quels contrôles sont affichés

Le protocole `UIImagePickerControllerDelegate` doit être implémenté pour annuler ou traiter une prise de vue

Les photos

Vérification l'appareil photo

Tous les appareils mobiles ne possèdent pas le même matériel

Il est nécessaire de vérifier ce qu'autorise le terminal

La méthode `isSourceTypeAvailable:` permet de vérifier la présence de :

- `UIImagePickerControllerSourceTypeCamera`
- `UIImagePickerControllerSourceTypePhotoLibrary`
- `UIImagePickerControllerSourceTypeSavedPhotoAlbum`

Les photos

Paramétrer l'interface de prise de vue

Pour afficher à l'écran une interface de prise de vue, il faut initialiser un objet de type `UIImagePickerController` et préciser que sa source est de type caméra

On peut également préciser quelle caméra utiliser

Il est aussi possible de permettre l'édition de la prise de vue grâce à l'attribut `allowsEditing`

Les photos

Paramétrer l'interface de prise de vue

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        let picker = UIImagePickerController()

        picker.allowsEditing = false
        picker.sourceType = UIImagePickerControllerSourceType.camera
        picker.cameraDevice =
                            UIImagePickerControllerCameraDevice.rear
        picker.cameraCaptureMode = .photo
        picker.modalPresentationStyle = .fullScreen
        present(picker,animated: true,completion: nil)
    }
}
```

ViewController.swift

Les photos

Implémentation du délégué

```
import UIKit

class ViewController: UIViewController,
                      UIImagePickerControllerDelegate,
                      UINavigationControllerDelegate {

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

        picker.delegate = self

    }
}
```

ViewController.swift

Les photos

Implémentation du délégué

Implémentation de la méthode permettant de gérer l'annulation d'une prise de vue

ViewController.swift

```
func imagePickerControllerDidCancel(_ picker:
                                    UIImagePickerController) {
    picker.dismiss(animated: true, completion: nil)
}
```

Les photos

Implémentation du délégué

Implémentation de la méthode permettant de traiter une photo ou une vidéo prise par l'utilisateur

`imagePickerController:didFinishPickingMediaWithInfo:`

Le paramètre `info` contient les informations sur le média sélectionné par l'utilisateur

- Si l'utilisateur a pris une photo, `info` est un dictionnaire contenant
  - l'image originale `UIImagePickerControllerOriginalImage`
  - l'image éditée `UIImagePickerControllerEditedImage`
- Si l'utilisateur a pris une vidéo, `info` contient l'url de la vidéo

Les photos

Implémentation du délégué

```
func imagePickerController(_ picker: UIImagePickerController,
     didFinishPickingMediaWithInfo info: [String : Any]) {

    let imageChoisie = info[UIImagePickerControllerEditedImage]
                                                    as! UIImage

    // Utilisation de l'image récupérée

    picker.dismiss(animated: true, completion: nil)
}
```

ViewController.swift

Les photos

Lecture d'un fichier contenant une photo

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        let fileURL : URL =
NSURL.fileURL(withPath:Bundle.main.path(
                                    forResource: "image",
                                         ofType: "png")!)
        if let imageData = NSData(contentsOf: fileURL) {
            let image = UIImage(data: imageData as Data)
        }
    }
}
```

ViewController.swift

Les photos

Enregistrement d'une photo dans un fichier .png

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        if let image = UIImage(named: "example.png") {
            if let data = UIImagePNGRepresentation(image) {
                let documents =
                        NSSearchPathForDirectoriesInDomains(.documentDirectory,
                        .userDomainMask, true)[0]
                let filename = documents.appending("/image.png")
                let url : URL = URL(fileURLWithPath: filename)
                do {
                    try data.write(to: url)
                }
                catch {
                    print("Erreur au moment de l'enregistrement")
                }
            }
        }
    }
}
```

ViewController.swift

Les photos

Enregistrement d'une photo dans un fichier .jpg

```swift
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        if let image = UIImage(named: "example.jpg") {
            if let data = UIImageJPEGRepresentation(image, 1.0) {
                let documents =
                        NSSearchPathForDirectoriesInDomains(.documentDirectory,
                        .userDomainMask, true)[0]
                let filename = documents.appending("/image.jpg")
                let url : URL = URL(fileURLWithPath: filename)
                do {
                    try data.write(to: url)
                }
                catch {
                    print("Erreur au moment de l'enregistrement")
                }
            }
        }
    }
}
```

ViewController.swift