

Application Miawouf



Objectifs pédagogiques:

- Utiliser une barre de navigation
- Créer des segues
- Utiliser une barre d'onglet
- Comprendre le cycle de vie du contrôleur
- Utiliser les principaux composants d'un formulaire
- Gérer le clavier
- Basse des données entre contrôleur
- Présenter des alertes
- Utiliser les extensions

Contexte

Dans les cours d'application iOS, on développe une application de rencontre pour Chien et Chat. Cette application est baptisée Miawouf.

Miawouf est divisée en deux, une inscription pour les chiens et une pour les chats. Et l'inscription se fait en trois pages :

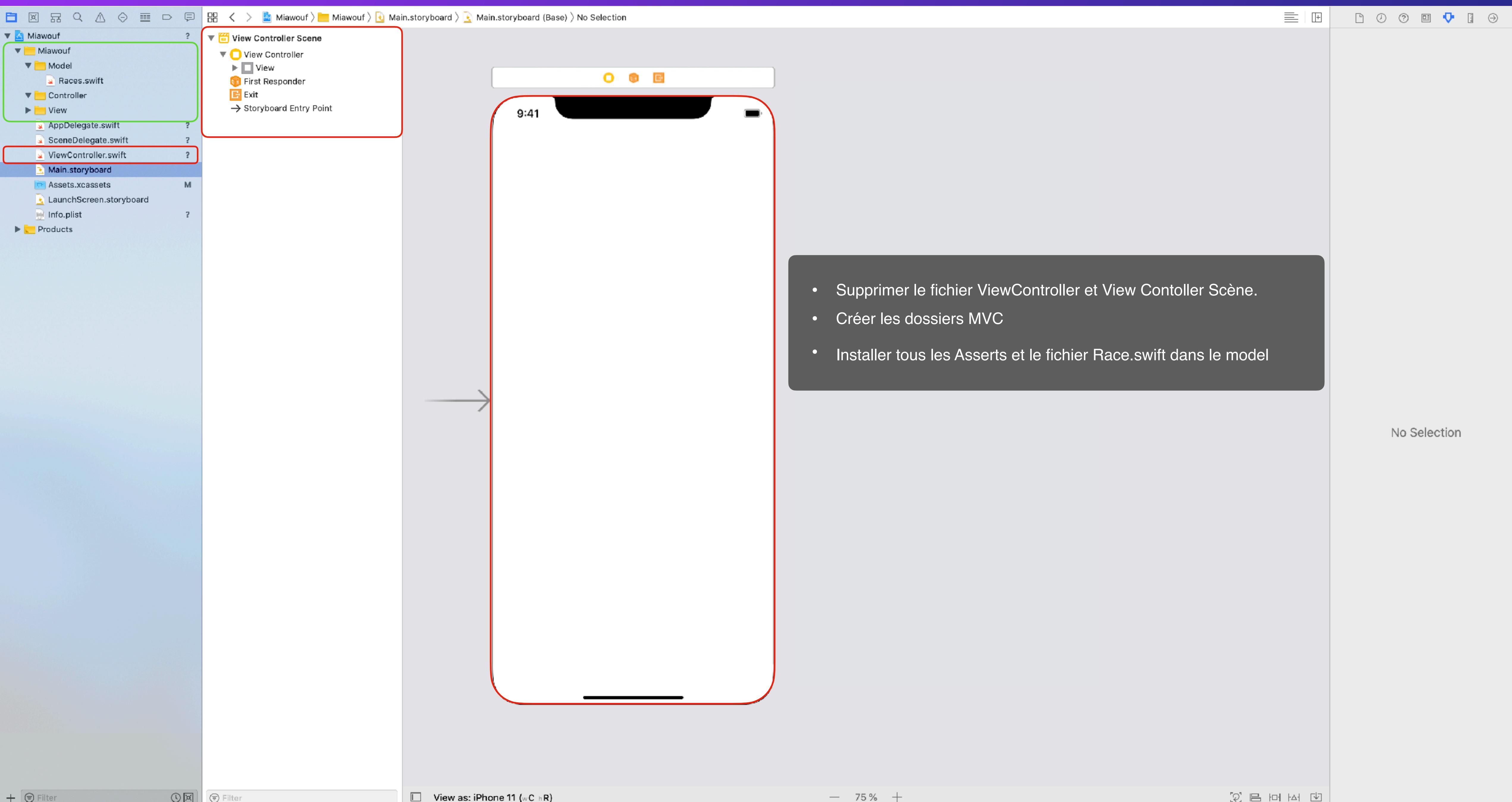
- une page d'accueil
- un formulaire d'inscription
- une page de confirmation



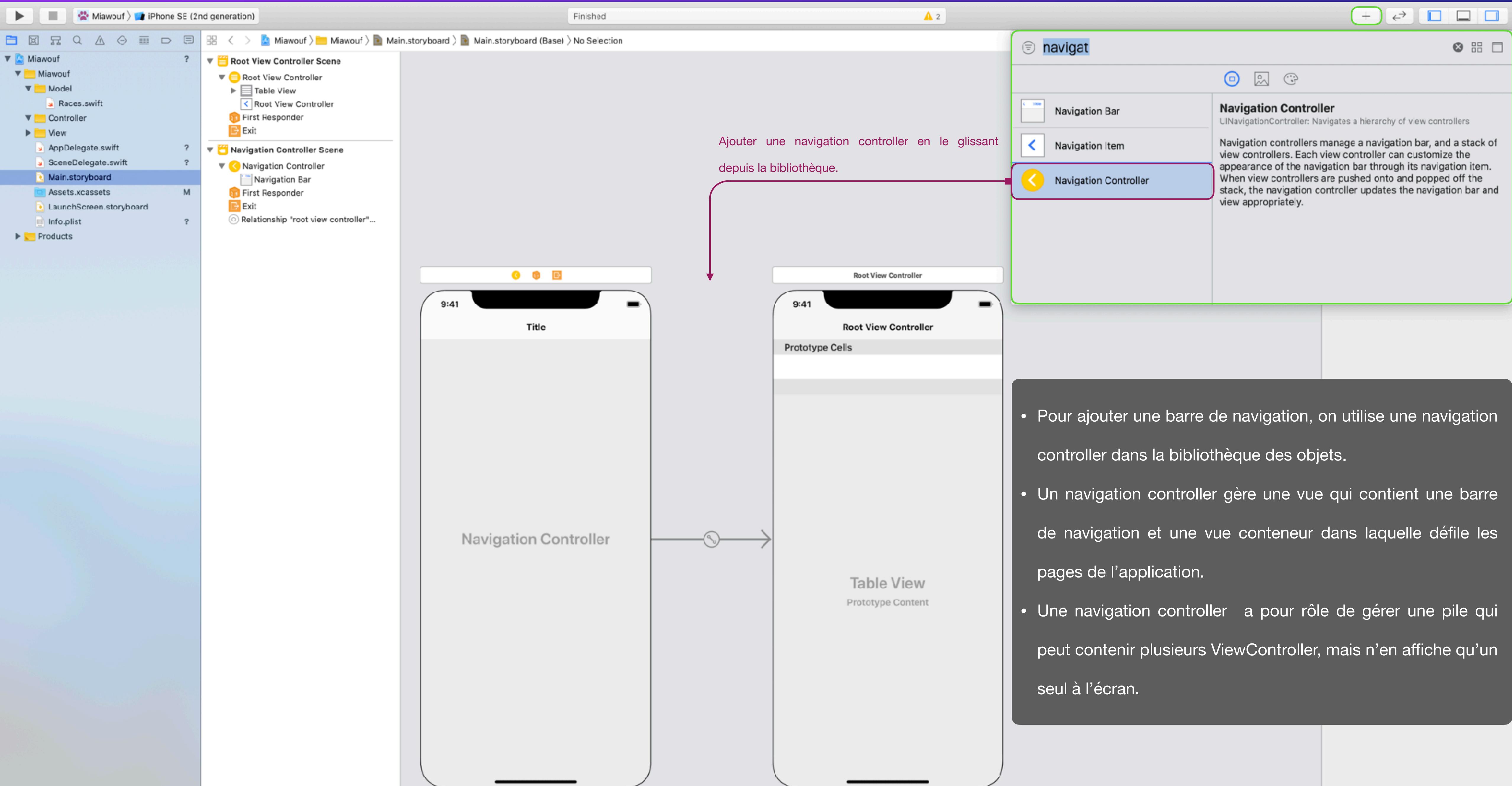
Schéma Main.storyboard



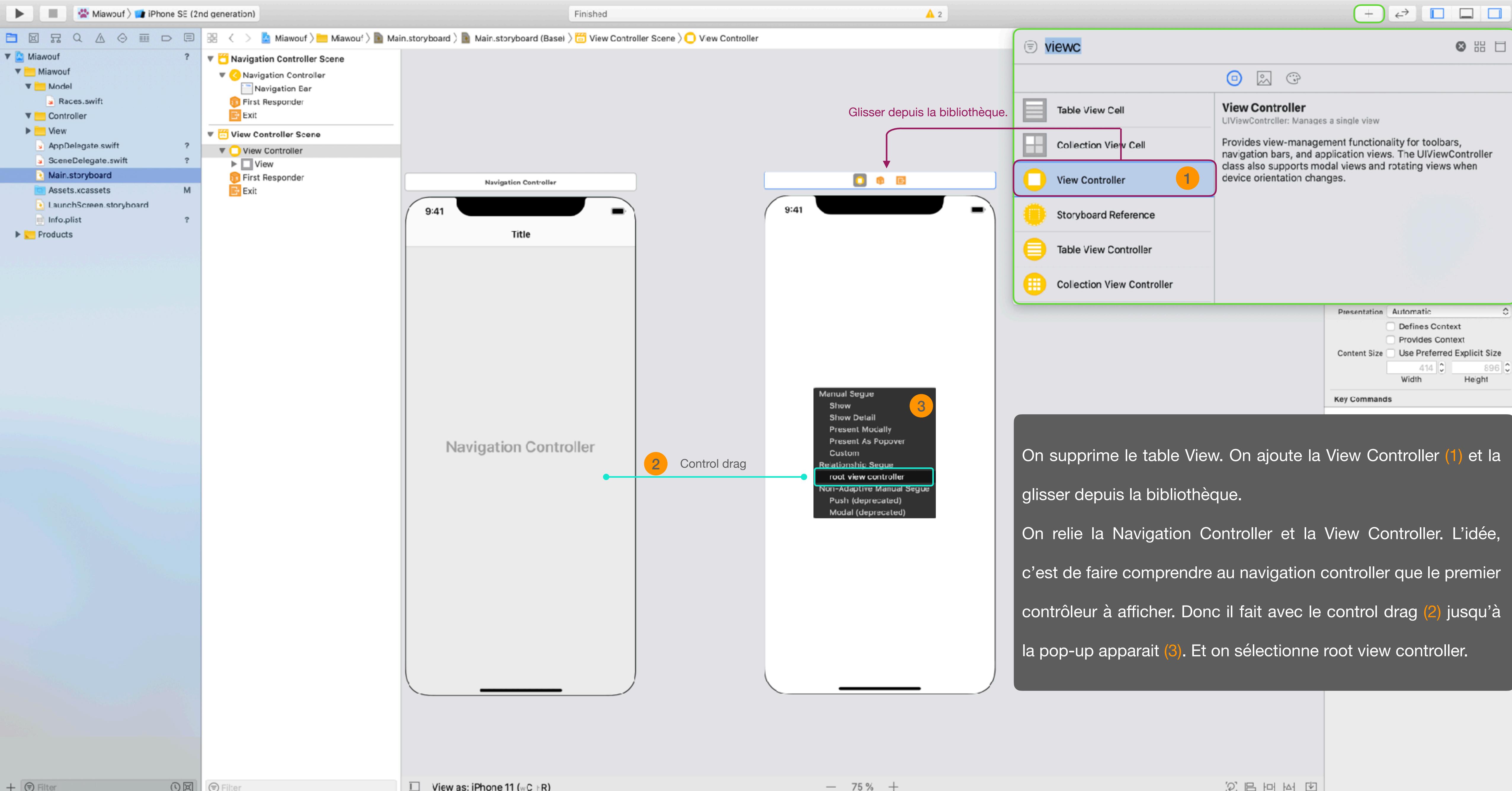
1.Organisation



2. Créer la barre de navigation: Navigation Controller



3. Ajouter la page dans storyboard: View Controller



4. Personnaliser la barre de navigation controller

The screenshot shows the Xcode interface with the project 'Miawouf' open. The storyboard 'Main.storyboard' is displayed, showing a 'Navigation Controller Scene' containing a 'Welcome Scene'. The 'Welcome' view has a title 'Welcome' and a large teal text 'Miawouf'. A pink paw print button with the text 'Go !' is also visible. The navigation bar at the top is white. The right side of the screen shows the 'Navigation Bar' settings in the Attributes Inspector, which are currently set to 'Default' style with a translucent background.

Navigation Bar

- Style: Default
- Translucent: checked
- Bar Tint: Custom (pink)
- Shadow: Shadow Image
- Back: Back Indicator Image
- Back Mask: Back Indicator Mask

Title Text Attributes

- Title Font: No Font
- Title Color: Custom (green)
- Title Shadow: Default
- Default Position

Large Title Text Attributes

- Title Font: No Font
- Title Color: Default
- Title Shadow: Default
- Default Position

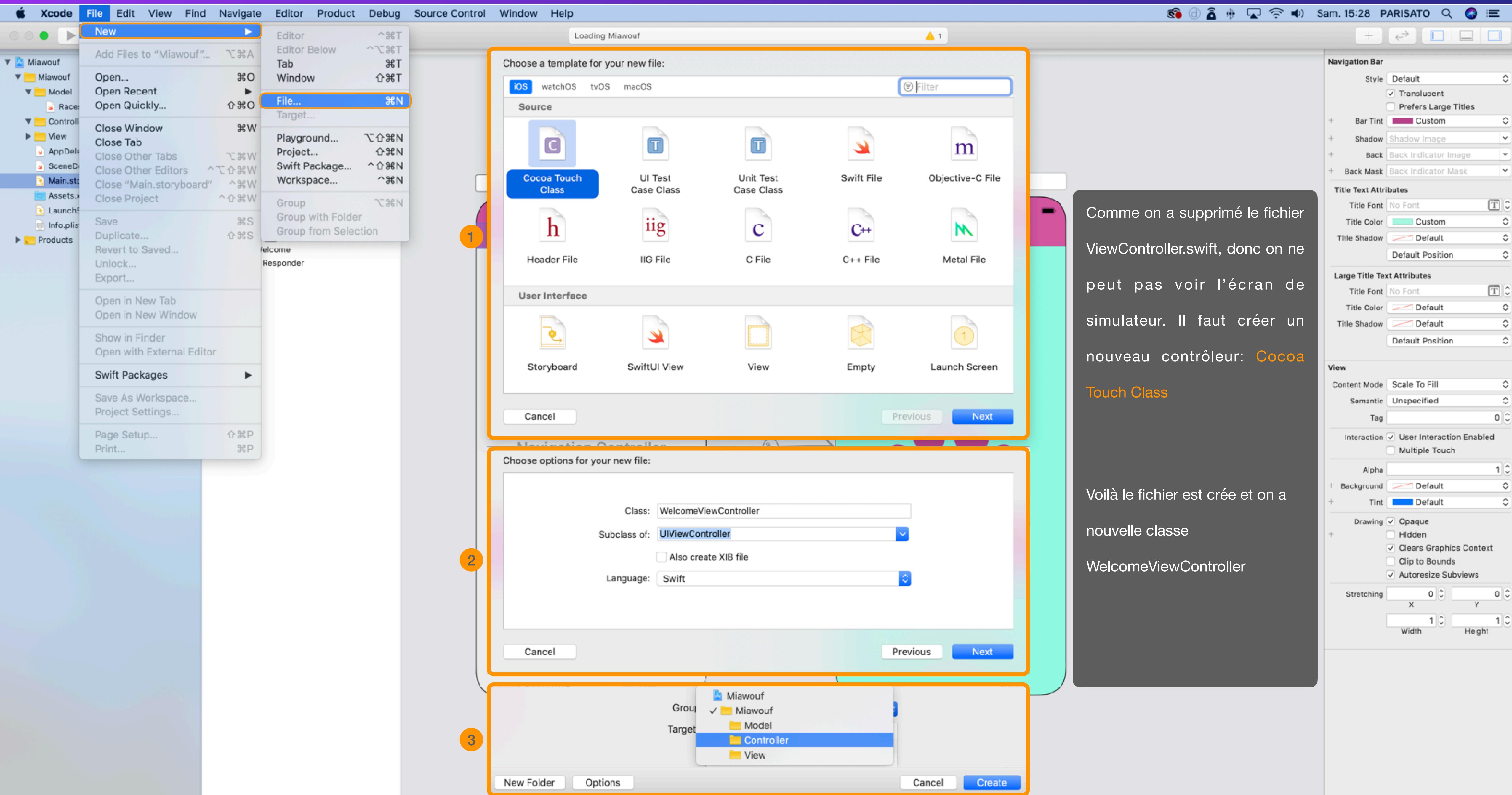
View

- Content Mode: Scale To Fill
- Semantic: Unspecified
- Tag: 0
- Interaction: User Interaction Enabled (checked)
- Alpha: 1
- Background: Default
- Tint: Default
- Drawing: Opaque (checked)
- Hidden: unchecked
- Clears Graphics Context: checked
- Clip to Bounds: unchecked
- AutoresizingMask Subviews: checked
- Stretching: X: 0, Y: 0
- Width: 1, Height: 1

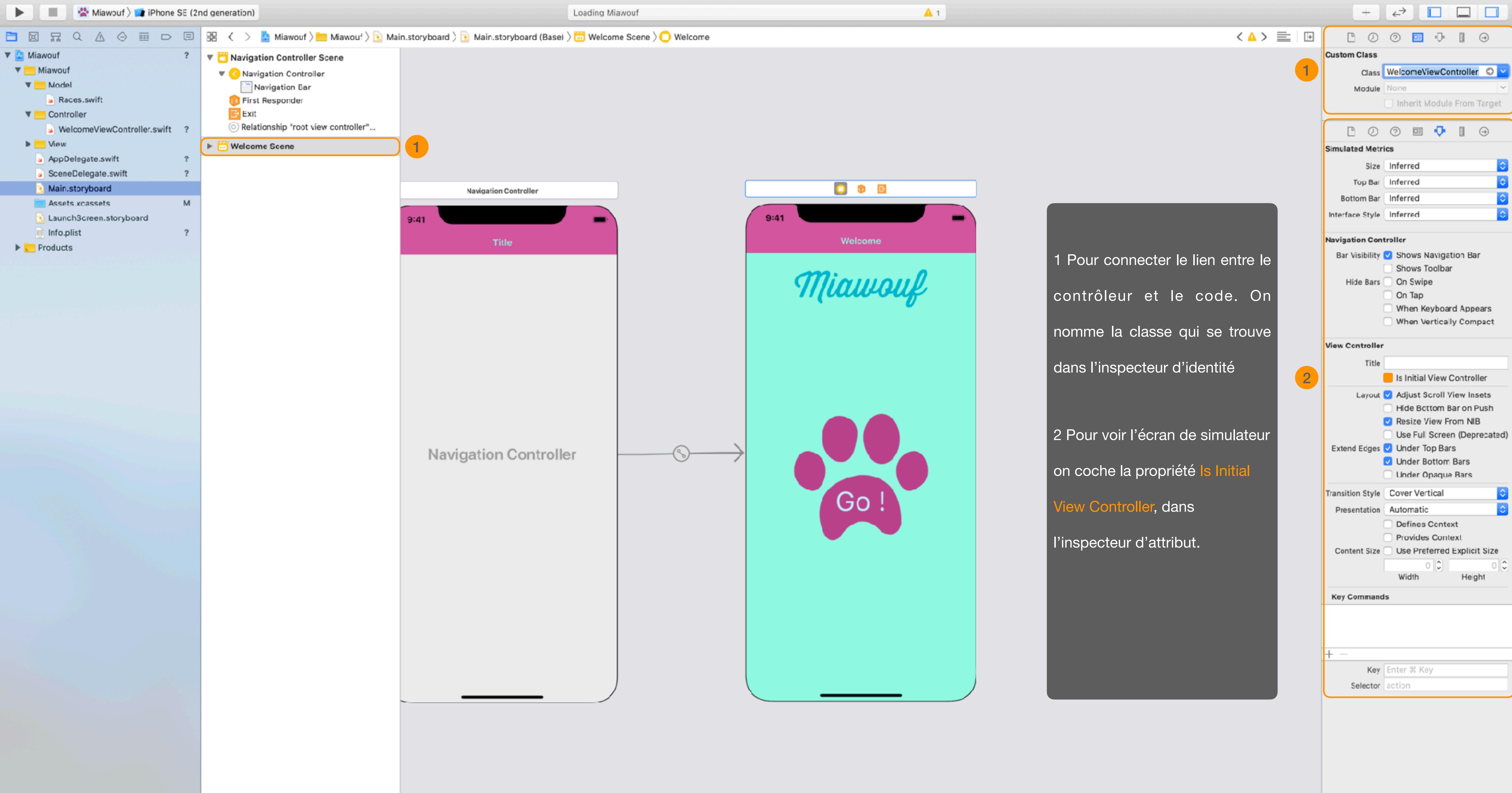
Text

- 1 créer l'image Miawouf, le bouton Go et on met la couleur du fond vert.
- 2 pour ajouter un titre dans la barre, on sélectionne dans le ViewController: navigation item. Et on le nomme Welcome.
- 3 pour choisir la couleur rose pour la barre, on sélectionne dans la navigation controller.

5. Créer une sous-classe de ViewController



6. Nommer le nom de classe dans le storyboard



1 Pour connecter le lien entre le contrôleur et le code. On nomme la classe qui se trouve dans l'inspecteur d'identité

2 Pour voir l'écran de simulateur on coche la propriété **Is Initial View Controller**, dans l'inspecteur d'attribut.

7. Utiliser un segue

The screenshot shows the Xcode storyboard editor with three scenes: Welcome Scene, Form View Controller Scene, and Succes View Controller Scene.

- Welcome Scene:** Displays a button labeled "Go!".
- Form View Controller Scene:** Displays a button labeled "VALIDER". A control drag from the "VALIDER" button to the "Form View Controller" in the storyboard triggers an **Action Segue**. The segue's **Show** option is selected.
- Succes View Controller Scene:** Displays a message: "Wouf! Vous avez bien été ajouté à notre base de données! Nous vous enverrons un SMS dès que nous avons une âme soeur à vous proposer!".

Custom Class: Set to `FormViewController`.

Identity: Storyboard ID is empty. Restoration ID is empty. Use Storyboard ID is checked.

User Defined Runtime Attributes: Key Path, Type, and Value are empty.

Document: Label is "Xcode Specific Label". Object ID is `rEW-7u-Lqb`. Lock is set to "Inherited - (Nothing)". Localizer Hint is "Comment For Localizer".

Text (right side):

- On ajoute la deuxième page pour appeler le contrôleur correspondant `FormViewController`. Cette manipulation est la même chose que `WelcomeViewController` (ajouter la page View Controller, créer le sous-classe de `ViewController` et nommer de la classe dans l'inspecteur d'identité).
- Le control drag doit se faire entre le bouton qui initialise la transition (bouton Go et `FormViewController`). Et on sélectionne `Segue Show`.
- On ajoute la troisième page `SuccesViewController`. Cette page sera affichée lorsqu'on appuie sur le bouton VALIDER et elle informera l'utilisateur que son inscription a été prise en compte avec succès. On utilise le control drag et on sélectionne `Segue Présent Modally`.

8. Revenir en arrière

The screenshot shows the Xcode interface with the storyboard and code side-by-side.

Storyboard: The storyboard displays three screens: **Welcome Scene**, **Form View Controller Scene**, and **Succes View Controller Scene**. A segue connects the Welcome screen to the Form View Controller. Another segue connects the Form View Controller to the Succes View Controller. On the Succes View Controller screen, there is a button labeled "Inscrire un autre chien".

Text Callout: A callout box contains the following text:

Dans Succes View Controller Scene, il n'y a pas de bouton retour. Car on a choisi Segue Present Modally pour s'afficher uniquement. Alors on rajoute un bouton avec une croix en haut à gauche sur l'interface. Ensuite, on crée une action nommée **dismiss** (1), car son rôle va être de faire disparaître la page. Enfin, on déclare une méthode de SuccesViewController, s'appelle **dismiss** (2).

Code: The code for **SuccesViewController** is shown in the right panel:

```
import UIKit  
class SuccesViewController: UIViewController {  
    @IBAction func dismiss() {  
        dismiss(animated: true, completion: nil)  
    }  
}  
  
The code for WelcomeViewController is also shown:

```
import UIKit
class WelcomeViewController: UIViewController {
 @IBAction func unwindToWelcome(segue:UIStoryboardSegue) {}
}
```


```

9. Ajouter une barre d'onglet: Tab Bar Controller

On ajoute Tab Bar Controller et la glisser depuis la bibliothèque. Et on supprime les deux Items Scene.

1. On rajoute le ViewController = Chat Scene
2. On crée le sous classe nommé **CatWelcomeViewController.swift**.
3. On connecte le lien entre le contrôleur et le code.

On va faire le control drag de **Tab Bar Controller** à **Chat Scene**. Ensuite une pop-up apparaît, on choisit **ViewController**.

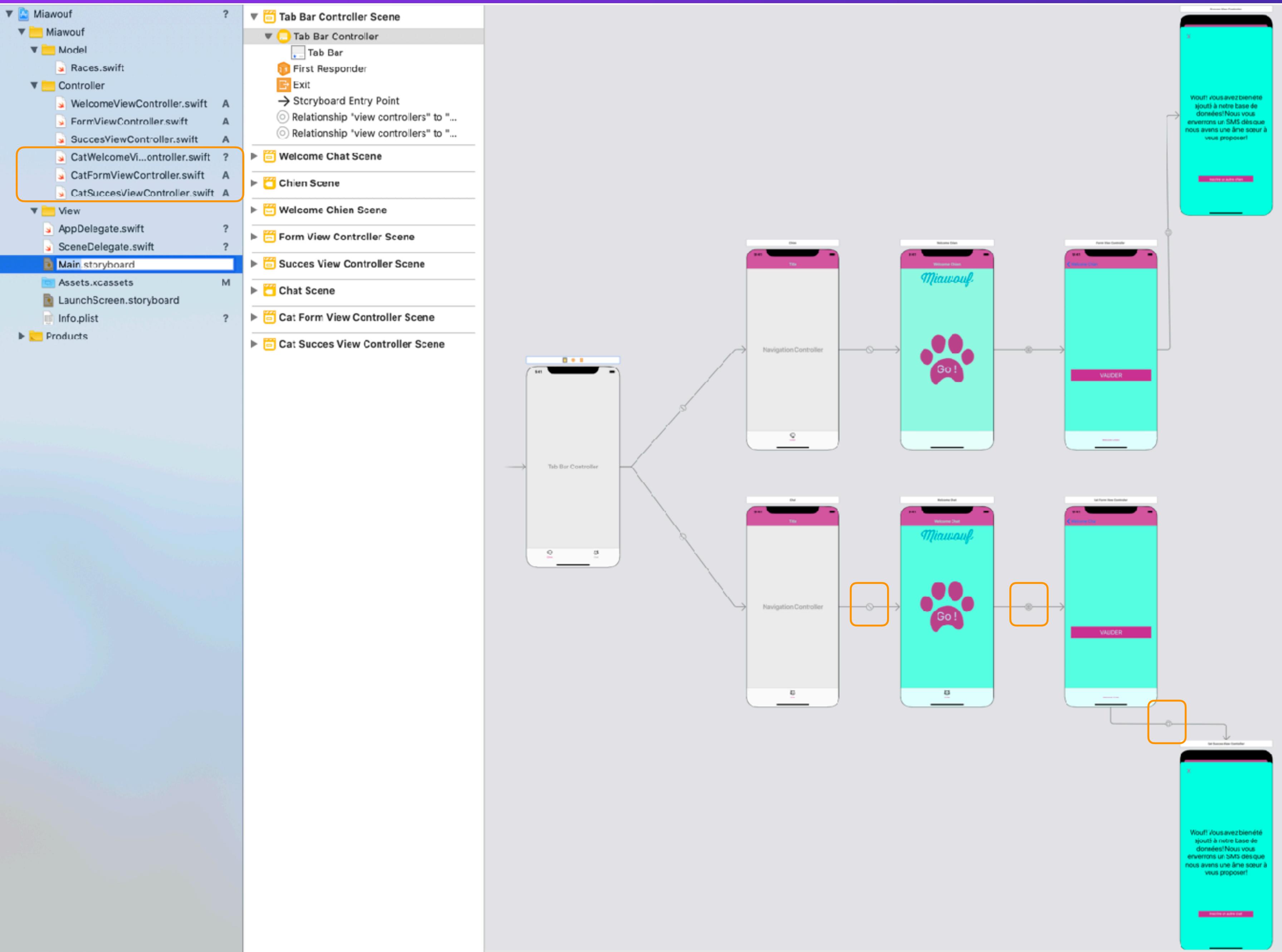
Custom Class
Class **CatWelcomeViewController.swift**
Module **Miaouf**
Inherit Module From Target

Identity
Storyboard ID
Restoration ID
Use Storyboard ID

User Defined Runtime Attributes
Key Path | Type | Value

Document
Label Xcode Specific Label
Object ID PrB-yC-7Bb
Lock Inherited - (Nothing)
Localizer Hint Comment For Localizer

10. Finir la navigation d'une application pour les chats



1) On va donc créer le navigation controller qu'il faudra relier au Tab Bar Controller. Puis on va ajouter trois contrôleurs qui correspondent au trois même étapes que pour les chiens.

2) Ensuite on crée les sous classes pour:

- CatWelcomeViewController.swift
- CatFormViewController.swift
- CatSuccesViewController.swift

Et on fait les control drags pour les segues:

- Root
- segue Show
- Segue Modally

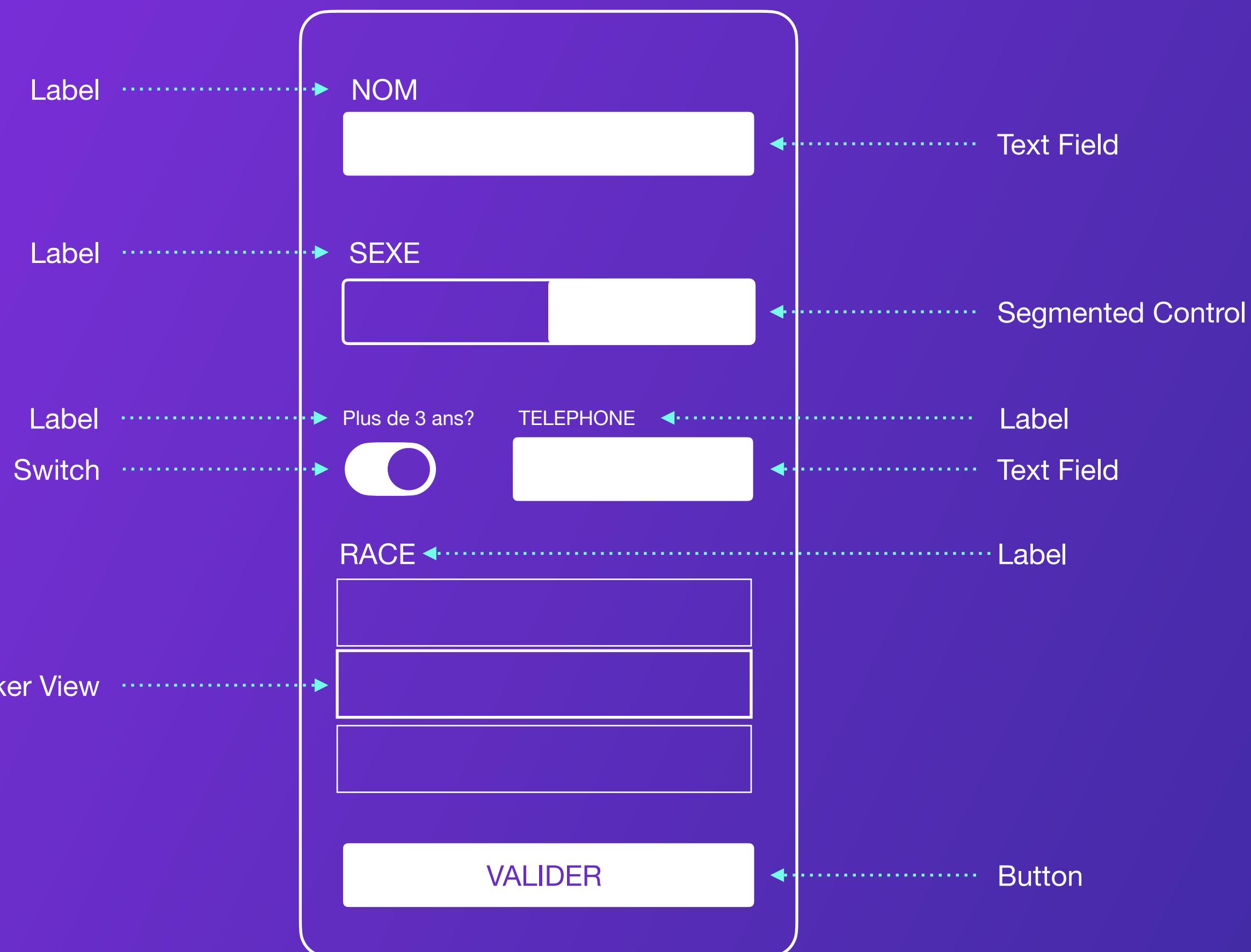
3) Enfin on crée la cible , on déclare dans le contrôleur:

-> @IBAction func unwinToWelcomCat ()

On déclare aussi pour disparaître la page:

-> @IBAction func dismissCat ()

Schéma Main.storyboard



1. Nom avec textField

textField: c'est un champ de texte

Il se matérialise par une barre blanche dans laquelle on peut écrire du texte.

- 1 La propriété **Text** permet de remplir le champ de texte avec le texte de son choix.
- 2 La propriété **Placeholder** de préremplir le champ de texte avec une indication grisée qui s'effacera dès que l'utilisateur commence à écrire.
- 3 La propriété **Border Style** permet de changer le style de la bordure autour du champ de texte.
- 4 La propriété **Clear Button** permet d'afficher un bouton qui supprime le contenu complet du champ de texte pour que l'utilisateur puisse revenir à zéro.
- 5 Les propriétés sous **Text Input Traits** concernent le clavier. Elles permettent de customiser ce dernier. En effet, le clavier en iOS est géré par le champs de texte. Parmi les propriétés intéressantes:

- Content type
- Capitalization
- Correction
- Spell Checking
- Keyboard Type
- Keyboard Look
- Return Key

2. Segment Control

The screenshot shows the Xcode interface with a storyboard project named "Miawouf". The storyboard scene is "Form View Controller Scene". On the right, the "Segmented Control" inspector is open, showing settings for a segmented control with two segments: "Segment 0 - Femelle" (selected) and "Segment 1 - Mâle". The "Title" is set to "Femelle" and "Image" is set to "Image". The "Behavior" is set to "Enabled" and "Selected". The storyboard preview shows a form with fields for "NOM" (containing "Medor"), "SEXE" (with a segmented control showing "Femelle" selected), and a "VALIDER" button at the bottom.

1 On rajoute le **label** nommé « sexe » pour organiser le formulaire.

2 On choisit la propriété **Segmented Control**. Cette propriété permet de choisir le nombre de segment du composant. Ici, on prend 2 options Mâle et Femelle.

3 **Titre**: on peut modifier le texte affiché dans le segment. On vous propose à écrire Femelle en premier et le second Mâle.

4 **Image**: à la place du texte, on peut choisir de mettre une icône.

5 **TintColor**: cette couleur définit la teinte générale d'une vue. Elle est souvent utilisée par les sous-classes d'UIView pour définir.

3. Switch

The screenshot shows the Xcode interface with the storyboard file 'Main.storyboard' open. The 'Welcome Chien Scene' is selected. In the storyboard preview, there is a text field containing 'Mecor', a gender selection section with 'Femelle' and 'Male' buttons, and a section for age with a switch labeled 'Plus de 3 ans ?'. The switch is currently set to 'ON' (green). The Xcode sidebar shows the project structure and the current file's contents.

« Avez-vous plus de 3 ans ? », ce genre de questions qui se répondent par oui ou non. On utilise la propriété **Switch**.

Dans l'inspecteur d'attributs, on a trois propriétés simples:

State: cette propriété permet de décider de l'état par défaut du bouton avec OFF ou ON.

On Tint: on choisit la couleur du composant lorsqu'il est dans l'état ON

Thumb Tint: on change la couleur du rond.

The Xcode interface includes the following panels:

- Project Navigator**: Shows the project structure with files like 'Races.swift', 'WelcomeViewController.swift', etc.
- File Inspector**: Shows basic file properties.
- Object Library**: Shows various UI components like buttons, labels, and switches.
- Attributes Inspector**: Shows the properties of the selected 'Switch' component, including 'State' (set to 'On'), 'On Tint' (set to 'Custom' with a green color), and 'Thumb Tint' (set to 'Custom' with a blue color).
- Document Outline**: Shows the hierarchical structure of the storyboard scene.
- Assistant Editor**: Shows the code for the 'WelcomeViewController.swift' file.

4. Telephone avec TextField

The screenshot shows the Xcode interface with the storyboard open. The left sidebar displays the project structure, including files like `Races.swift`, `WelcomeViewController.swift`, and `FormViewController.swift`. The storyboard preview shows a form titled "Welcome Chien" with fields for "NOM" (containing "Mecor"), "SEXE" (with "Femelle" selected), and "Plus de 3 ans ?" (with a switch turned on). A "Telephone" text field is present, which is highlighted with a yellow selection rectangle. The right side of the screen shows the "Text Field" inspector, where the "Content Type" is set to "Phone Number" and the "Keyboard Type" is set to "Phone Pad". Other properties like "Text" (Plain), "Font" (System 14.0), and "Placeholder" (Placeholder Text) are also visible.

Pour pouvoir contacter les chiens de les mettre en relation, on va leur demander leur numéros de téléphone. Pour cela on va choisir le champ de texte et on va utiliser la propriété Content Type à Phone Number et la propriété Keyboard Type à Phone Pad.

4. Picker View

Picker View prend la forme d'une roulette que l'on peut diviser en plusieurs colonnes qui vont chacune monter des listes différentes.

On sélectionne Picker View dans la bibliothèque des objets et glisser le dans l'interface.

Schéma Main.storyboard & ViewController



```
class FormViewController: UIViewController, UIPickerViewDataSource, UIPickerViewDelegate {  
  
    func numberOfComponents(in pickerView: UIPickerView) -> Int {  
        return 1  
    }  
  
    func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {  
        return dogRaces.count  
    }  
  
    func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? {  
        return dogRaces[row]  
    }  
}
```

1. Delegate, Datasource

The screenshot shows the Xcode interface with the storyboard open. On the left, the file navigator displays various files including `Races.swift`, `WelcomeViewController.swift`, and `FormViewController.swift`. The storyboard scene is titled "Welcome Chien". Inside, there's a `Form View Controller` containing fields for "NOM" (with value "Medor"), "SEXE" (with button bar items "Femelle" and "Male", where "Femelle" is highlighted), "Plus de 3 ans ?" (with switch), "RACE" (with picker view showing options: Mountain View, Sunnyvale, Cupertino, Santa Clara, San Jose), and a "VALIDER" button. A callout bubble over the `Form View Controller` points to outlets `dataSource` and `delegate`. A number "1" is circled near the `dataSource` outlet. A second callout bubble over the `Picker View` points to its `dataSource` outlet, with a number "2" circled near it. The right side of the screen shows the attributes inspector for the `Picker View`, which is set to `Scale To Fill` content mode and `Unspecified` semantic.

On a crée l'interface avec Picker View, mais le sélecteur n'apparaît pas au simulateur (à l'écran).

- la vue nomme le contrôleur son **datasource** et son **delegate** avec deux control drag.
- le contrôleur s'engage à répondre aux questions de la vue. Pour cela, le contrôleur va implémenter avec le protocole (voir les codes ci-dessous).

Ligne 11 -> on ajoute le nom de protocole dans la classe

Ligne 12 -> la méthode renvoie un entier Int. Car cet entier correspond au nombre de composant pour la races de chiens. Donc on va répondre à la question est return 1.

Ligne 16 -> la méthode retourne aussi un entier. C est le nombre d'éléments présents dans chaque colonne. On va répondre par le nombre de races de chiens à afficher: `dogRaces.count`

Ligne 20 -> la méthode renvoie un String optionnel qui correspond au titre que l'on veut mettre pour chaque élément. On va utiliser le paramètre `row` en index de tableau `dogRaces`.

```
8 import UIKit
10
11 class FormViewController: UIViewController, UIPickerViewDataSource, UIPickerViewDelegate {
12     func numberOfComponents(in pickerView: UIPickerView) -> Int {
13         return 1
14     }
15
16     func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
17         return dogRaces.count
18     }
19
20     func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? {
21         return dogRaces[row]
22     }
23 }
24 }
```