TP Développement iOS 1 - Vos premières applications pas à pas.

IMT Nord Europe – UV AMSE

Lors de ces TP, vous avez accès (à distance...) à des Mac mini, avec XCode installé pour que vous puissiez développer et tester vos applications. Nous allons voir la réalisation d'une première application.

1 Votre première application

1.1 Création du projet

Pour ce premier projet, vous allez sélectionner un projet de type « App » dans la partie « iOS ». Dans les paramètres, à la place de SwiftUI, nous allons choisir StoryBoard et choisir comme langage Swift. Pour "organization identifier", vous pouvez écrire par exemple "com.imtld.".

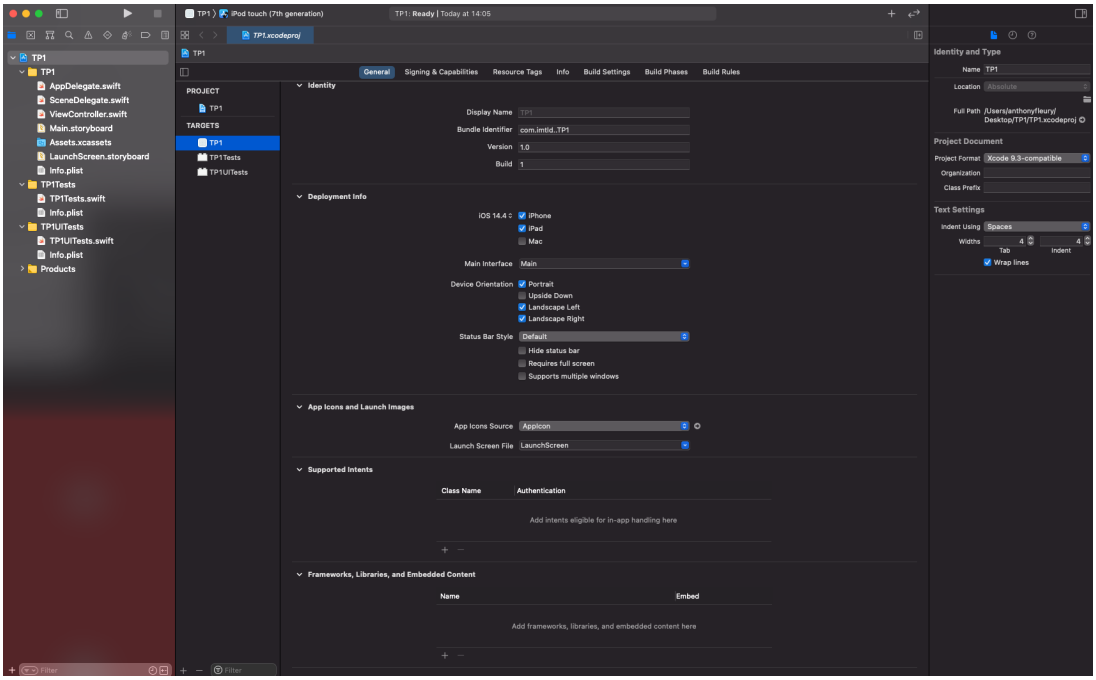

FIGURE 1 – Votre projet et sa configuration

Vous pouvez tout d'abord regarder la configuration de votre projet en cliquant, sur l'explorateur de gauche, sur le nom de celui-ci (la première ligne). Vous aurez ainsi les détails sur le nom de votre programme, son nom affiché sur l'iPhone, les orientations supportées, etc. (voir Fig. 1)
Votre projet au départ contient plusieurs fichiers, dont certains qui vous intéressent plus en détail :

— Le fichier Main.storyboard qui contient les vues pour iPhone et pour iPad (différentes tailles d'écran) pour lequel la Fig. 2 vous montre la version iPhone SE 2 (en bas vous avez un View As qui vous montre quel device vous avez),

— Le fichier ViewController.swift qui décrit la classe de la vue principale générée par XCode (que vous voyez dans le storyboard).

1.2 Votre première vue

Vous allez d'abord travailler sur le fichier Storyboard.

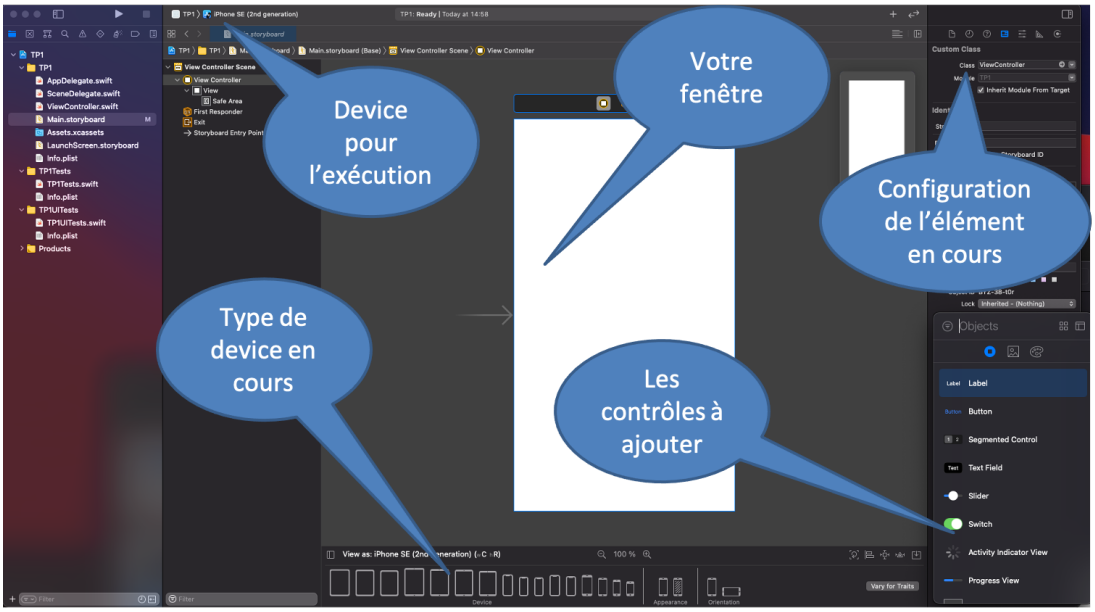

FIGURE 2 – Votre fenêtre de départ.

1.2.1 Remplissage de votre vue

La première étape est de remplir votre vue avec différents contrôles. Pour afficher les différents contrôles disponibles, vous allez dans View puis Show Library. Lorsque vous êtes vraiment sur le mac, en appuyant en plus sur la touche Option (en ouvrant la librairie) celle ci va rester affichée... A distance par contre c'est plus compliqué (car le mapping clavier n'est pas simple). Parmi ces contrôles, vous trouverez des labels (du texte), des zones de saisie (text edit), des contrôles multisegments, des boutons, etc.

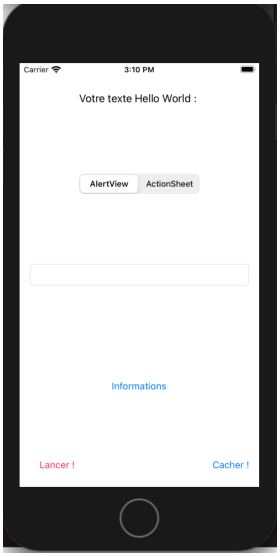

FIGURE 3 – Fenêtre de votre application

La figure 3 vous donne un exemple de présentation de votre vue, exécutée sur un iPhone SE 2 (sur simulateur pour être conforme à votre environnement à vous), avec 3 boutons, un label, une zone de texte et un sélecteur multisegment. **Travail de cette partie : Ajoutez un à un les contrôles proposés à votre vue puis réglez les paramètres de la vue puis de chacun des contrôles pour donner à votre application le style que vous voulez. Nous nous concentrerons pour l'instant sur le fait de faire une vue iPhone. Peut importe son aspect sur iPad.**

1.2.2 Associer un élément de la vue à une variable ou une action

L'association d'éléments de la vue à du code se fait en deux étapes. La première étape est d'insérer, dans le code de la vue, (le fichier ViewController.swift) une nouvelle variable qui décrit le contrôle. De la même manière, sont inclues aussi des méthodes d'instances qui vont réaliser des actions lors d'un évènement. La figure 4 montre la modification

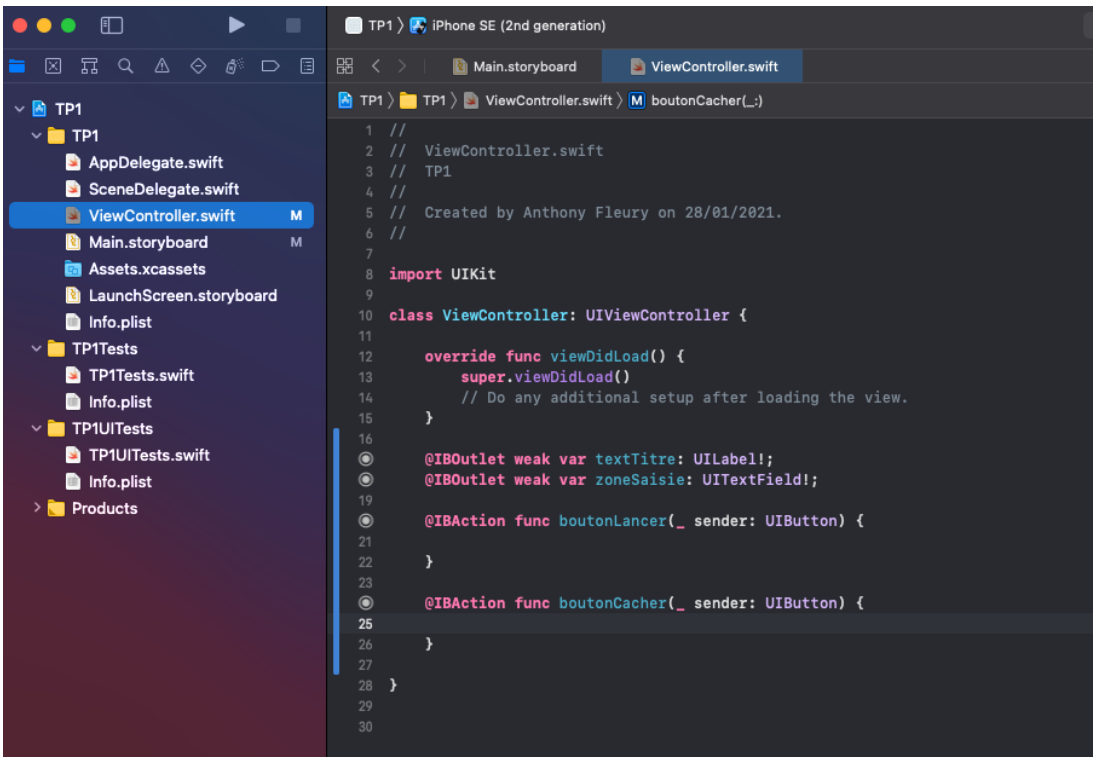

FIGURE 4 – Code de votre vue

de la classe associée à la vue pour ajouter un label. IBOutlet est un mot particulier qui permet de désigner un élément d'interface graphique, IBAction est un ajout particulier au prototype de la méthode désignant une méthode associée à un évènement graphique. . Ces méthodes reçoivent en général l'envoyeur (déclaration de la forme @IBAction **func** boutonLancer(_ sender: UIButton)) mais celui-ci peut-être ignoré lorsque ce n'est pas nécessaire, i.e. lorsque vous n'avez pas besoin de connaître le contrôle envoyant le message (vous en aurez besoin si une même méthode gère plusieurs évènements). Autre chose, vous noterez le modificateur de type **weak** dans le type de la variable. Celui-ci indique que la gestion mémoire de la variable est faite ailleurs. Vous noterez enfin que la variable est un optionnel (présence du ! à la fin de la déclaration), ceci parce que, lorsque la fenêtre n'est pas encore créée, la variable n'existe pas...

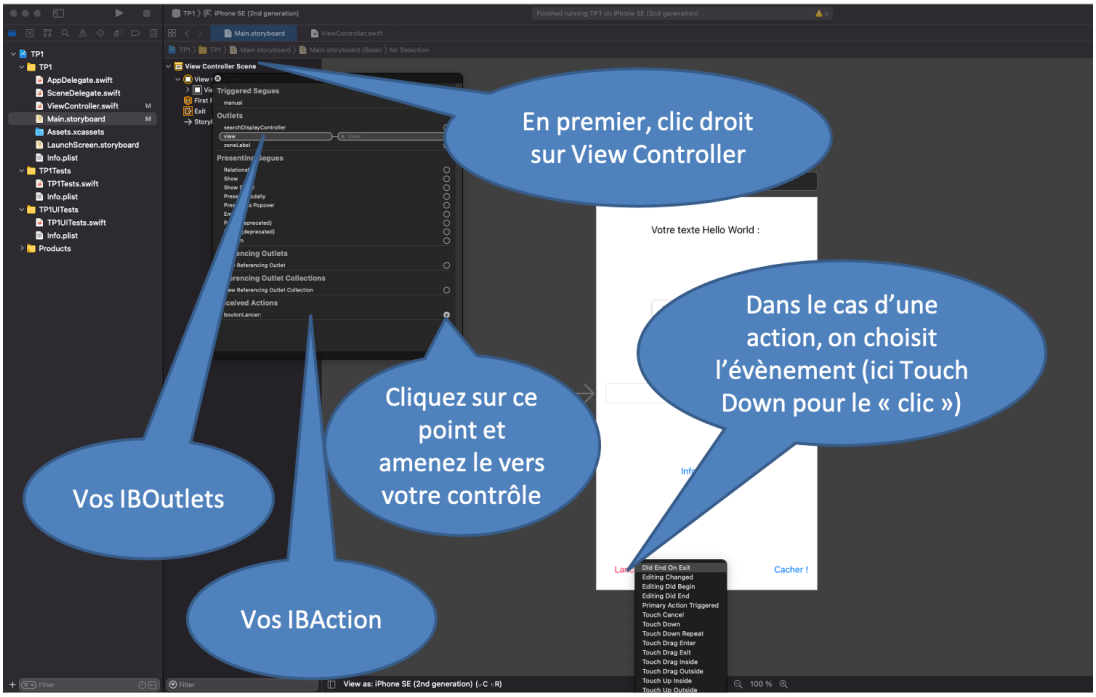

FIGURE 5 – Association variables ou méthodes avec un contrôle

La figure 5 vous montre l'association de vos contrôles avec les variables et les méthodes de votre classe gérant la vue. Un clic droit sur le contrôleur de vue du storyboard permet d'avoir accès aux Outlets et aux Actions.
Une fois ceci fait, dans le code de votre vue, vous pouvez alors avoir accès aux éléments de vos contrôles et aussi modifier la manière dont ils sont affichés. Vos contrôles ont en effet des propriétés membres, avec les accesseurs qui sont créés, et qui permettent d'avoir des informations sur celui-ci ou modifier ces informations. Ainsi, tous les contrôles auront une propriété *isHidden* permettant de cacher, montrer ou de savoir si le contrôle est caché ou non. De manière plus particulière, un UITextField (zone de saisie de texte) aura une propriété text permettant d'avoir accès au texte tapé et de modifier le texte de la zone. Un UIButton aura lui un title alors qu'un UISegmentedControl aura une propriété selectedSegmentIndex renvoyant un entier ou permettant de modifier celui qui est sélectionné. Par exemple, pour un contrôle UITextField champTexte, écrire champTexte.text = "Nouveau␣Texte" changera le texte affiché par le contrôle.

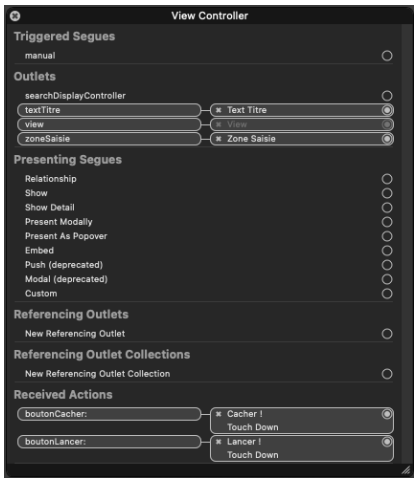

FIGURE 6 – La fenêtre une fois les associations faites

Une fois ceci fait, vous voyez, comme sur la Figure 6, les différentes associations entre les variables et vos contrôles.
Travail de cette partie : Ajoutez les différentes variables pour vos contrôles, et une méthode d'instance que vous associerez au bouton « Cacher ». Cette méthode cachera l'ensemble des éléments de la fenêtre graphique (sauf le bouton cacher et le bouton informations) puis changera le texte pour afficher Montrer dans le bouton. Un autre clic fera l'inverse (montrer tous les contrôles et rechanger le texte du bouton pour Cacher).

1.2.3 Votre bouton lancer

Nous voulons que ce bouton affiche une boîte de dialogue qui répète le message inscrit. Pour ceci, nous allons utiliser les Alertes, avec deux types d'affichage que nous allons voir (les AlertView et ActionSheet).
Pour utiliser ceux-ci, vous allez utiliser le code suivant :

Listing 1 – Code dans votre fichier Swift

```
let controller = UIAlertController(title: "Titre␣du␣message", message: "
    Contenu␣du␣message", preferredStyle: .alert)

let action1 = UIAlertAction(title: "Premier␣Bouton", style: .default) { (
    action:UIAlertAction) in
    // code première action
}

let action2 = UIAlertAction(title: "Annuler", style: .cancel) { (action:
    UIAlertAction) in
    // code annuler
}

let action3 = UIAlertAction(title: "Bouton␣type␣destructif", style: .
    destructive) { (action:UIAlertAction) in
    // code destructif
}
```

```
        controller.addAction(action1)
        controller.addAction(action2)
        controller.addAction(action3)
        self.present(controller, animated: true, completion: nil)
```

Dans le .preferredStyle, si vous mettez .action, vous allez changer le type d'Alerte (message au milieu et bloquant) à Action (en bas et non bloquant).
Dans ce code, vous allez lister les "actions" à faire, avec du code à l'intérieur pour chacune. En fonction de ce qui va faire que la fenêtre se ferme, la bonne action sera faite.
Un exemple pour construire et appeler chacune d'entre-elle vous est aussi montré dans ce code. Selon ce que vous voulez faire avec vos fenêtres, vous mettrez du code différent dans celles-ci.
Dernier petit détail, le type "destructive" affichera le texte du bouton en rouge (associé en général à une action qui a une conséquence possiblement néfaste).
Travail de cette partie : Adaptez le code donné pour que lorsque vous allez cliquer sur le bouton lancer, selon si AlertView ou ActionView est sélectionné, vous chargez le bon type de fenêtre, avec comme message le texte que vous avez entré et deux boutons : un bouton qui va uniquement fermer la fenêtre (ne rien faire), un autre bouton qui va supprimer le texte de la zone d'édition.

1.2.4 Comportement spécifique : l'iPhone et le clavier...

Vous avez donc maintenant écrit le code de votre bouton Lancer et celui de votre bouton Cacher/Montrer. Lorsque vous tapez du texte, le clavier apparait automatiquement dans la fenêtre. Dans le cas où vous êtes sur iPad, pas de soucis, un bouton permettant de cacher le clavier sur iPad est disponible. Par contre, sur iPhone ce n'est pas le cas. Nolus voudrions faire disparaitre le clavier en appuyant sur retour.
Pour faire ceci, nous allons définir une méthode @IBAction **func** clavier(_ sender: UIButton) contenant comme seule instruction : sender.resignFirstResponder().
Travail de cette partie : Insérez ce code dans votre vue puis associez cette action à l'évènement « Did End On Exit » de votre zone de texte. Vérifiez alors que le clavier disparait après la saisie.

1.3 Ajout d'une vue

Maintenant que vous avez écrit votre première vue et bien décrit son comportement, nous allons voir comment construire et appeler une seconde vue.

1.3.1 Ajout d'une vue au storyboard

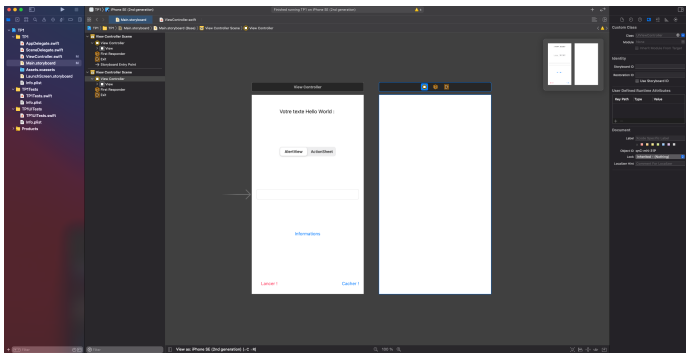

FIGURE 7 – Création et Association de votre vue

Dans votre storyboard, vous pouvez ajouter un objet de type View Controller (voir Figure 7). Cet objet va gérer une nouvelle fenêtre dans laquelle vous pouvez ajouter des contrôles.
Travail de cette partie : Faites ceci pour votre storyboard.

1.3.2 Création de la classe associée

Dans Files, New, File, sélectionnez un type Cocoa Touch pour iOS. Dans la fenêtre suivante, donnez un nom à votre classe, et mettez comme subclass of UIViewController. Sélectionnez le langage Swift. Cliquez sur next. Sélectionnez l'endroit de votre projet et Create.
Travail de cette partie : Regardez la figure 8 pour associer cette classe nouvellement créée à votre vue dans le storyboard.

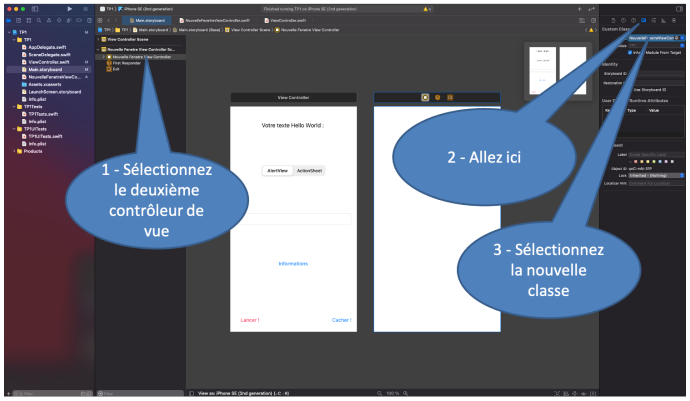

FIGURE 8 – Création et Association de votre vue

1.3.3 Association de l'ouverture de la vue à un bouton

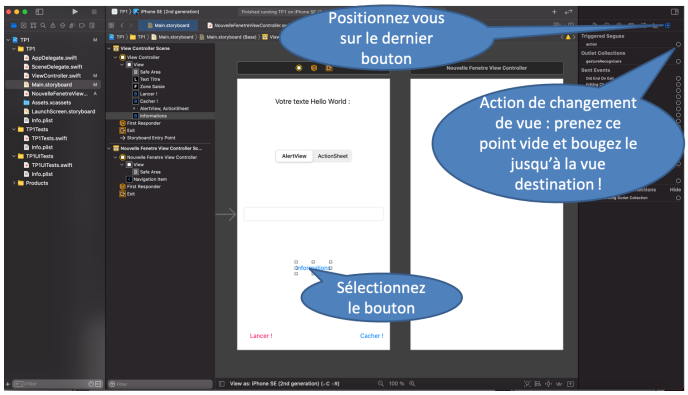

FIGURE 9 – Changement de vue au clic sur le bouton !

Suivez la figure 9 pour associer l'ouverture de votre nouvelle vue au clic du bouton information. Lorsque cela vous est demandé, vous pouvez choisir un mode d'ouverture « Present Modally ».
Travail de cette partie : Faites cette association avec votre bouton information pour votre storyboard iPhone puis iPad.

1.3.4 Comportement de cette vue

Cette seconde vue vous est montrée sur la Figure 10. Dans cette vue nous avons :

- Une barre de navigation avec le titre informations,
- Un texte donnant les informations sur l'application,
- Un texte et un switch qui va permettre de valider le fait que le texte précédent a été compris,
- Un bouton retour dans la barre de navigation.

Ce que l'on veut est que lorsque le switch est sur on (faites une méthode qui se déclenche sur un évènement changement de valeur du bouton de barre de navigation), alors le bouton est actif. Par défaut dans la vue, le bouton est inactif (enabled décoché) et le switch sur off.
L'appuie sur le bouton retour va lancer le code qui permet de fermer la fenêtre en cours et qui est : **self**.dismiss(animated: **true**, completion: **nil**).
Attention, pour accéder à la valeur du switch, si celui-ci s'appelle monSwitch, alors vous devez utiliser monSwitch.on.
Pour le bouton de barre de navigation, bouton.enabled=**true** l'active alors que bouton.enabled=**false** le désactive.
Travail de cette partie : Vérifiez le comportement global de toute votre application et si vous voulez, vous pouvez également changer les différents paramètres de celle-ci (icône, splash window, etc.).

2 Votre seconde application : un petit jeu

Laissez maintenant parler votre imagination et écrivez un petit jeu, pour lequel vous allez définir quelques questions dans des tableaux avec des bonnes et des mauvaises réponses, puis une validation pour chaque question (qui sera tirée au hasard).

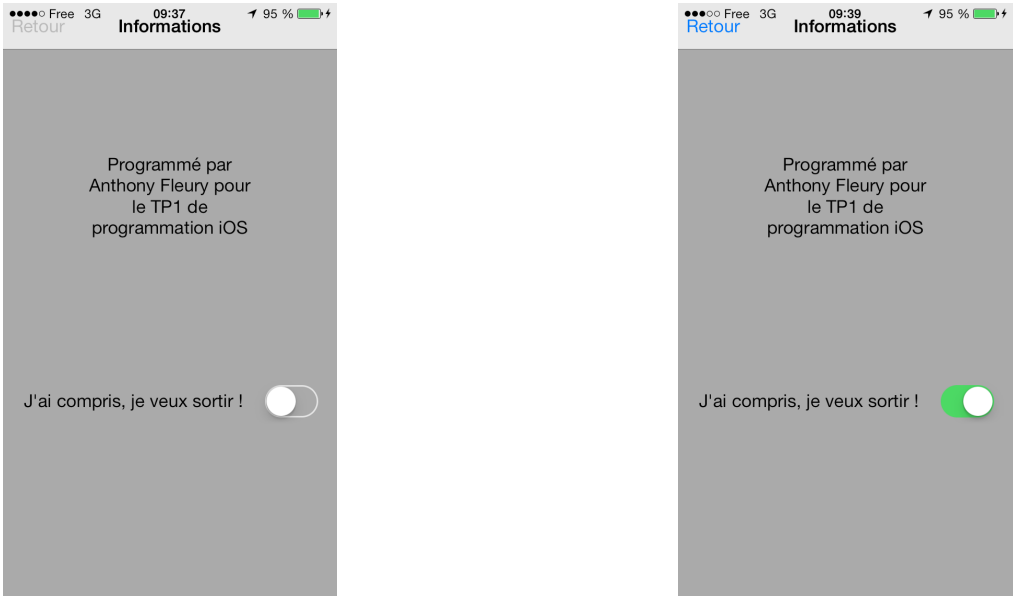

FIGURE 10 – La seconde vue.

Faites un jeu avec un système de score (prenez exemple sur des jeux TV ou autre).
À vous de jouer et de coder ! N'hésitez pas à poser des questions pour réaliser certaines fonctionnalités que vous ne sauriez pas écrire.

3 Références

Le site d'Apple contient de très nombreuses documentations (en ligne et sous forme de PDF) pour l'ensemble du SDK iOS. Vous les trouverez à l'adresse `http://developer.apple.com/library/ios/index.html`.