Filleule Quentin

Dandana Nasser

ESIREM
ÉCOLE SUPÉRIEURE D'INGÉNIEURS
NUMÉRIQUE ET MATÉRIAUX

Compte Rendu Projet Swift

TodoList

Introducing Swift

Dans ce compte rendu nous allons voir les étapes de la création de notre application. L'objectif est ici de créer une Todolist simple d'utilisation à l'aide de l'environnement XCode et du langage Swift.

La Vue Principale

Dans un premier temps, nous allons créer la vue principale de notre application à savoir la liste de chose à faire.

Pour cela nous allons créer un view contoller dans lequel nous allons mettre un TableView, l'objectif va ici être d'afficher à la suite les éléments de notre TodoListe.

Pour cela nous allons avoir besoin d'un tableau de données qui va être de type Data. Ce type est une classe que nous avons créé, comportant au minimum un nom et une description pour chaque élément.

Il faut maintenant gérer l'accès du TableView via le controller, pour cela nous créons un outlet de TableView vers un TableViewControlleur.

Pour avoir accès aux cellules, la manipulation est différente, il faut alors associer les cellules à une classe swift de type Cocoa Touch Class. C'est comme cela que nous obtenons TodoTableViewCell

Il ne nous reste plus qu'à associé cette classe à la cellule à l'aide du tableau d'identité, on peut ainsi créer un outlet des éléments de nos cellules (nom et description) .

Pour tester notre affichage, il nous faut créer une fonction addList qui vas nous permettre de créer la première tâche.

A chaque ajout d'un nouvel élément dans le tableau, nous avons besoin de faire deux choses :

- Utiliser la propriété « append » du tableau pour ajouter l'élément dans celui-ci
- Recharger la vue pour afficher le nouvel élément dans une nouvelle cellule

Notre Première tâche est maintenant afficher, cependant nous n'avons pas accès à la description et nous ne pouvons pas en créer directement depuis l'application. Ça va être l'objet des parties suivantes

La Vue de Création des Tâches

La première chose à faire est de rajouter une deuxième vue qui est aussi un ViewControlleur, cette deuxième vue va nous servir à créer une nouvelle tâche.

L'objectif va être ici d'ouvrir une nouvelle page (Edit View Controlleur) quand on appuis sur le bouton + situé en haut à droite de l'écran. Pour cela il suffit de cliquer sur le bouton en question puis de le drag and drop dans notre nouvelle vue, un menu de Segue s'affiche alors. Nous utiliserons la fonction Show pour afficher la nouvelle page que nous venons de personnaliser.

L’objectif est maintenant de rapatrier les informations du TextField et de la TextView sur la

la vue principale.

Pour cela nous allons avoir besoin de créer une fonction SaveBtnEdit, celle-ci sera relié directement au bouton Save. Cette fonction est en fait de type «UnwingSegue », c’est-à-dire les fonctions prévues par Swift pour transférer les informations d’une vue à une autre.

Sa source est donc bien le View Controller Edit dans lequel on récupère le texte contenu dans le TextField et la TextView, portant ici le nom de TodoTitle et TodoDesc.

Pour forcer la conversion en chaine de caractère nous utilisons un point d’exclamation à la fin de nos variables. On peut alors utiliser les informations de cette vue pour créer un nouvel élément dans notre tableau de TodoListe avec l’attribut « append ».

Enfin on recharge notre TableView pour afficher la nouvelle tache lorsqu’on revient à la vue principale.

La Vue de Modification des Tâches

Maintenant que nous pouvons créer des taches, nous avons besoins de voir la description, de la modifier et si besoin de la supprimer et cela en cliquant directement sur la tâche. C’est donc l’objet de notre troisième vue : DetailViewController.

Encore une fois la vue sera personnalisée avec un TextField et une TextView pour accueillir le nom et la description de la tâche.

De la même manière que pour le bouton de création des taches, nous avons drag and drop le TableView sur cette nouvelle vue puis sélectionner show dans le menue de Segue, on utilise ensuite la fonction « prepare » pour envoyer les données dessus.

Pour cela nous avons besoin d’obtenir le numéro de la tâche que nous devons afficher, c’est aussi l’objet de la fonction prepare dans laquelle nous utilisons l’attribut myIndexPathe.row pour l’obtenir.

Une fois la tache correctement récupérée et affichée, nous voulons la modifier. Pour cela il suffit de changer directement les éléments dans les champs texte correspondant. Une fois les modifications appliquées, il suffit d’appuyer sur le bouton Edit. Un unwindsegue se déclenche alors sur le même principe que pour la création des taches avec le bouton save. Seulement cette fois il faut également prendre en compte la tâche que l’on modifie, c’est donc ici que l’élément « row » entre en jeux.

On l’utilise pour remplacer les anciens éléments tel que :

myData[row].name (l’ancien texte) prend la valeur de name1 (nouveau texte).

Ce nouveau texte est le nom contenu dans la vue du DetailVueController que nous avons rapatrier avec la fonction EditBtnDetail précédente.

On utilise ensuite la méthode reloadData pour mettre à jour le tableau de tâche.

Nous allons maintenant nous attarder sur la fonction DelBtnDetail qui va nous permettre de supprimer nos tâches.

Le fonctionnement est globalement le même que la fonction précédente à l'exception que nous utilisons la méthode myData.remove(at : row). Celle-ci permet de supprimer un élément du tableau suivant son numéro, sachant que nous avons le numéro de la tache en question, la fonction est plutôt simple.

Il ne reste plus qu'à utiliser la méthode reloadData pour mettre à jour le tableau de tâche.

La méthode remove possède ici un avantage pour nous puisque le tableau se rééquilibre automatiquement pour ne pas laisser un élément vide. Dans notre cas, cela veut dire que les tâches sont toujours à la suite et qu'il n'y a pas de trous.

Il reste quelques fonctionnalités à rajouter par rapport à l'énoncé mais nous n'avons pas eu le temps de les faire en cours mais, dans la globalité, notre application au problème basic de la TodoListe.