Compte rendu TP de programmation mobile IOS

Fait sur Xcode en Swift

Marcelo Lopes

1. Conception du modèle de données

Pour cette partie je n'ai rien rajouter de spécial à part deux attribut en plus qui sont statut et duTo qui permettent respectivement de savoir si une tache est fait ou pas et pour quand elle est à faire.

2. Création du Storyboard et Programmation de l'application

Pour la création de la première vue avec les taches, j'ai utilisé la même méthode que nous avons vue en TD en mettant dans ma cellule un switch qui permettra de modifier l'attribut statut de la classe ToDo. De plus sur cette vue, j'ai rajouté un bouton permettant d'afficher la vue qui permet d'ajouter une nouvelle tâche.

Pour faire apparaître la vue qui affiche les détails des taches, j'ai créé un segue qui est attaché à ma cellule et qui va vers ma vue où j'affiche les détails de ma tâche. Pour envoyer les données vers cette vue, j'utilise la méthode prepare ou je mets dans mon segue de destination la tâche qui a été cliquée. Dans cette vue, j'ai aussi créé un bouton permettant de supprimer une tache en utilisant un unwindSegue ou je vérifie que la tâche qu'on souhaite supprimer existe toujours dans notre liste de tâches.

Dans la vue qui permet d'ajouter des taches, j'ai mis deux textField permettant de renseigner le nom et la description, ces valeurs sont enregistrées seulement lorsqu'on retire le clavier visuel ce qui signifie que si l'on ne retire pas le clavier au moins une fois avant d'appuyer sur save, les attributs nom et description correspondront à des « nil ». Ensuite, lorsqu'on appuie sur le bouton save, la méthode unwindSegue vérifie que les données reçues sont conformes a ce que l'on souhaite et si cela n'est pas le cas elle met a true l'attribut erreur qui permet d'activer une alerte dans la méthode viewWillAppear (méthode qui s'active dès que la vue apparaît.) qui affiche une erreur indiquant qu'on ne peut pas ajouter la tache en question.

3. Ajout de la date de réalisation

Pour ajouter la date de réalisation, j'ai rajouté un UIDatePicker dans ma vue qui permet de créer une tache et je récupère la date du UIDatePicker pour la mettre dans l'attribut duTo de mon objet ToDo.

Pour trier les taches en fonction des dates, j'utilise un bouton qui lorsqu'on appuie dessus, trie le tableau toDoList en utilisant la méthode sorted(by :) avec une closure qui compare les attributs date et qui permet de les trier par ordre croissant des dates. Ensuite, pour que cela apparaisse sur la vue, j'utilise la méthode tableView.reloadData() qui permet de recharger les données affichées dans la table.

4. Ajout de catégories

Pour ajouter la fonctionnalité de catégorie, j'ai dû modifier mon système principal, mais en gardant le principe de base que j'ai fait.

Tout d’abord, j’ai créé une nouvelle vue qui devient ma vue principale où j’ai une liste des catégories de taches et qui ressemble quasiment à la vue permettant d’afficher les taches (cette vue utilise aussi un TableView). J’ai aussi rajouté un bouton pour ajouter une catégorie qui lorsqu’on appuie dessus, affiche une alerte permettant de donner le nom à notre nouvelle catégorie.

Lorsqu’on appuie sur une catégorie, on affiche la liste contenant les taches de cette catégorie qui correspond à notre ancienne vue principal avec la liste des taches, mais où j’ai rajouté des fonctionnalités en plus qui sont :

-un label pour voir dans quelle catégorie nous somme

-un bouton pour supprimer la catégorie actuelle

- un bouton retour en arrière personnalisé qui lorsqu’on appuie dessus (et donc qu’on retourne à la vue principale) modifie le tableau d’attribut toDoList de la classe ToDoList avec le tableau d’attribut toDoList de la vue permettant d’afficher toutes les taches. Ce qui permet de stocker tous les changements effectués sur cette catégorie.

De plus, j’ai aussi rajouté dans la méthode viewDidLoad des données permettant de plus facilement manipuler l’application.

5. Ajout d'un moteur de recherche

Pour cette partie, j'ai rajouté un UISearchBar dans notre vue permettant d’afficher nos taches de notre catégorie. De plus, nous avons rajouté dans la classe correspondant à cette vue un nouvel attribut qui est filteredToDoList et qui correspond maintenant au tableau de taches qu’on affichera dans notre tableView.

Maintenant que cela est fait pour que notre barre de recherche fonctionne, nous utilisons la méthode searchBar qui s’active dès que nous modifions la valeur de la barre de recherche. À l’intérieur de cette méthode, nous filtrons les taches de l’attribut filteredToDoList en regardant si la chaîne de caractères de la barre de recherche est contenue dans le nom de notre tâche, si c’est le cas alors la tache reste dans l’attribut sinon elle est supprimée. Pour faire cela, j’utilise la méthode filter du tableau qui permet de créer un nouveau tableau qui contient uniquement les éléments de notre tableau d'origine qui satisfont notre critère.