Systèmes d'exploitation mobiles et applications

TP01 - To Do List

Master HES-SO

Émilie Gsponer, Grégory Emery

26 novembre 2015
version 1.0

Table des matières

1 Introduction

1.1 Cas pratique

Dans ce travail pratique, nous devons créer une application permettant de gérer une To Do List. Deux vues seront créées. L'application pourra naviguer entre les deux.
Dans la première vue, on retrouve une représentation des tâches existantes. Chaque entrée est constituée d'une image (« ! » ou vide) ainsi que d'un descriptif de la tâche.
L'autre vue permet l'ajout d'une nouvelle tâche dans la liste. Un bouton sert à indiquer si la tâche est urgente ou non, un champ de text permet de recevoir le nom de la tâche.
L'image ci-dessous présente la solution de design proposée par la donnée.

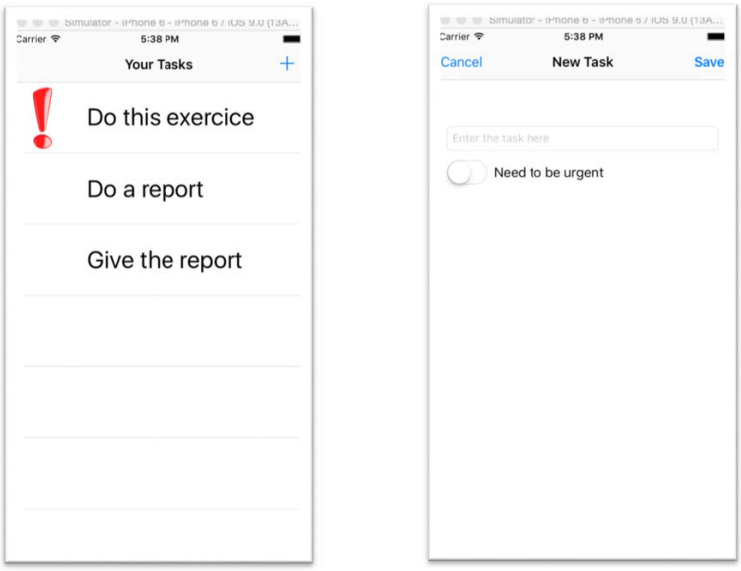

FIGURE 1 – Vues de l'application

1.2 Buts

1. Se familiariser avec le langage swift
2. Utilisation de "Navigation Controller"
3. Comprendre le développement et déploiement d'application iOS.

1.3 Objectifs

1. Création de la vue pour l'ajout d'une tâche
2. Connecter la vue avec le "view controller" pour l'ajout et implémenter le contrôleur.
3. Création du modèle qui représente une tâche
4. Création de la vue pour lister les tâches
5. Connecter la vue avec le "table view controller" pour la gestion de la table et implémenter le contrôleur
6. Mise en place de la navigation

2 Travail réalisé

Tous les points exigés par la donnée ont été réalisés. Les éléments suivants ont fait l'objet d'une recherche approfondie et ont été utilisés dans le projet :

1. Stackview : équivalent aux LinearLayout d'Android
2. View controller : gère la navigation entre les vues
3. Optional types : si une variable n'est pas initialisée, elle contient nil
4. Table view : équivalent aux ListView d'Android
5. Protocol UITextFieldDelegate : Interface proposant des méthodes gérant le champ de texte.
6. unwind segue : permet de quitter une vue et de revenir à une autre en appelant une méthode spécifique de cette dernière.

Cette image représente les deux vues de l'application développée.

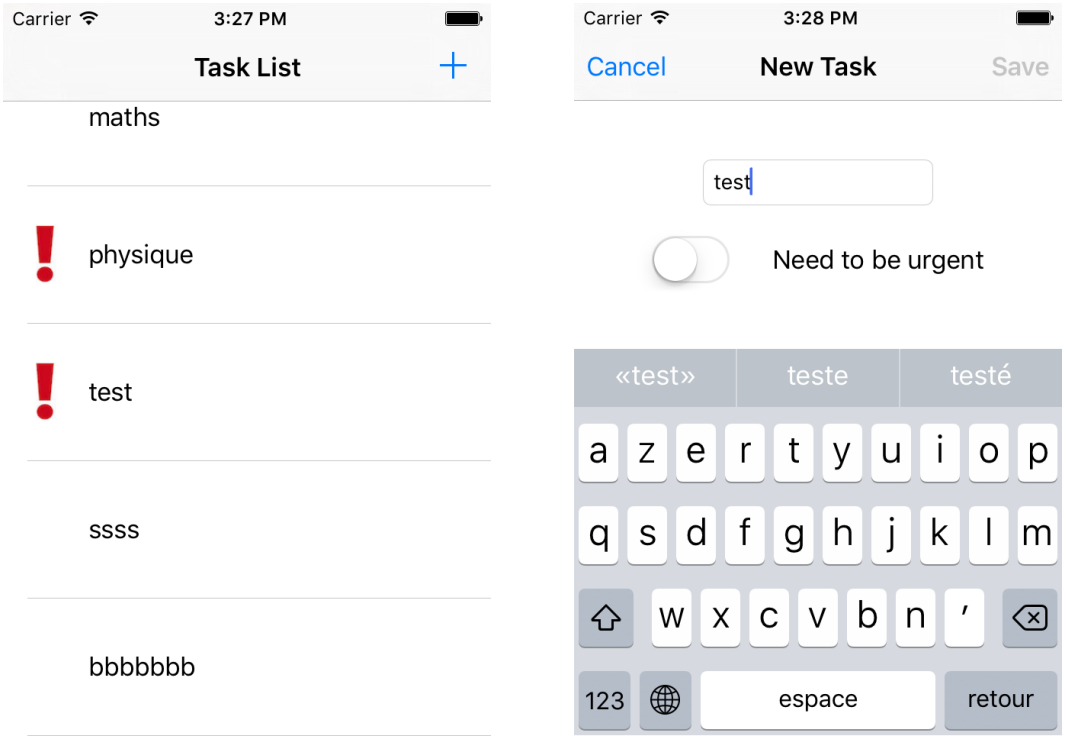

FIGURE 2 – Vues de l'application réalisée

Source de l'image ! :
http://static.guim.co.uk/sys-images/Guardian/Pix/pictures/2009/4/29/1240996556472/exclamation-001.jpg

N.B : A cause de l'implémentation du protocole UITextFieldDelegate, il est impératif d'appuyer sur la touche retour du clavier après avoir entrée le nom de la tâche. Sans cela, le bouton save ne devient pas actif.

3 Problèmes rencontrés

L'exercice 6 a posé quelques problème avec le "Navigation Controller" et plus spécifiquement la connexion des boutons cancel et save qui produisait une erreur.
Finalement, la navigation entre les vues a été reprise en suivant pas à pas le tutoriel à l'adresse suivante.
https://developer.apple.com/library/prerelease/ios/referencelibrary/GettingStarted/DevelopiOSAppsSwift/Lesson8.html

4 Réponse aux questions

4.1 Question 1

Donnée : Expliquez avec vos mots, quelle est l'utilité de la « segue ».

Une segue est l'équivalent d'un intent dans Android, elle permet de passer des données entre différentes vues lors d'un changement de vue.

4.2 Question 2

Donnée : Expliquez comment réagit le « navigation controller » si la vue appelée est de type « show » ou « Present Modally».

Show : La nouvelle vue contient automatiquement une bar de navigation avec un bouton back et est ajoutée par défaut au navigation controller. Le bouton back est déjà implémenté et permet de revenir à la vue précédente.

Present modally : affiche la nouvelle vue telle qu'elle est sans l'ajouter au navigation controller ou en lui ajoutant une barre de navigation. Si ce dernier élément est souhaité, il faudra également ajouter un navigation controller à la nouvelle vue.

4.3 Question 3

Donnée : Expliquez le cycle de vie des « view controller ».

L'image ci-dessous présente graphiquement le cycle de vie des view controller. Les méthodes viewDid(Dis)Appear et viewWill(Dis)Appear sont gérées automatiquement. Elles se charge d'effectuer les opérations pour qu'une vue puisse apparaître/disparaître et de rendre (in)visible tous les éléments qu'elle contient. L'utilisateur peut seulement agir quand le controller est dans l'état appeared à l'aide de la méthode viewDidLoad(). On peut par exemple y définir l'état d'un bouton ou le texte d'un label.

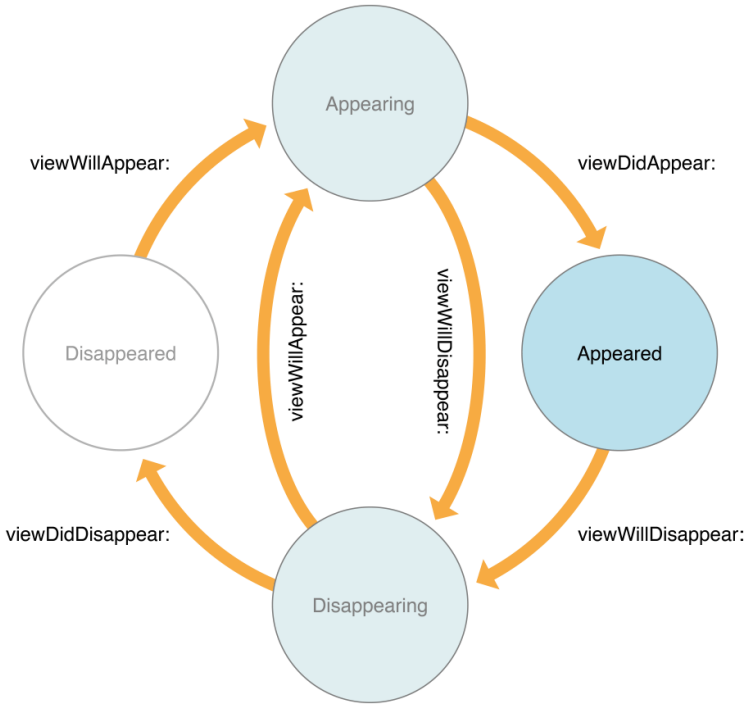

FIGURE 3 – Cycle de vie des view controller

5 Conclusion

Ce laboratoire a permis de mettre en avant la simplicité de création des TableView. Avec Android, ce n'est pas la même histoire, pour faire une ListView personnalisée, cela prend énormément de temps.
Une fois que l'on a compris le principe du navigation controller, on se rend compte que gérer les barres de navigation sous iOS est simple. On arrive à un résultat élégant sans y investir beaucoup de temps.