Systèmes d'exploitation mobiles et applications

TP02 - To Do List + Multimedia

Master HES-SO

Émilie Gsponer, Grégory Emery

16 décembre 2015
version 1.0

Table des matières

1 Introduction

1.1 Cas pratiques

Ce projet est composé de deux cas pratiques distincts.

1.1.1 Cas pratique 1

Dans le premier cas pratique, il faut reprendre la To Do List du TP1. Elle va être complétée avec des options pour éditer les tâches et les supprimer.
L'édition se fait lors d'un clique sur l'une des tâches de la liste.
La suppression se fait avec un swipe ou à l'aide du bouton edit.
Une autre option à implémenter est de sauvegarder les tâches dans une base de donnée pour qu'elles soient rechargés lors du prochain lancement de l'application.

1.1.2 Cas pratique 2

Dans ce cas pratique, il va falloir réaliser une vue qui permet de lire une vidéo. Il faudra également implémenter le swipe pour changer de vue et une toolbar sur l'écran principal.
La partie concernant la liste de musique et sa lecture n'a pas pu être implémentée, car nous ne disposons pas d'IPhone et le simulateur ne permet pas de générer la liste de musiques.

1.2 Buts

1. Se familiariser avec le langage swift
2. Utiliser les identificateurs de segue
3. Utilisation de « gesture »
4. Créer une structure.
5. Utiliser le protocole NSCoding.
6. Utilisation de « MediaPlayer »

2 Travail réalisé

2.0.1 Cas pratique 1

Tous les points exigés par la donnée ont été réalisés. Voici un aperçu des deux différentes vues du projet, la tableView et la vue pour ajouter/éditer une tâche :

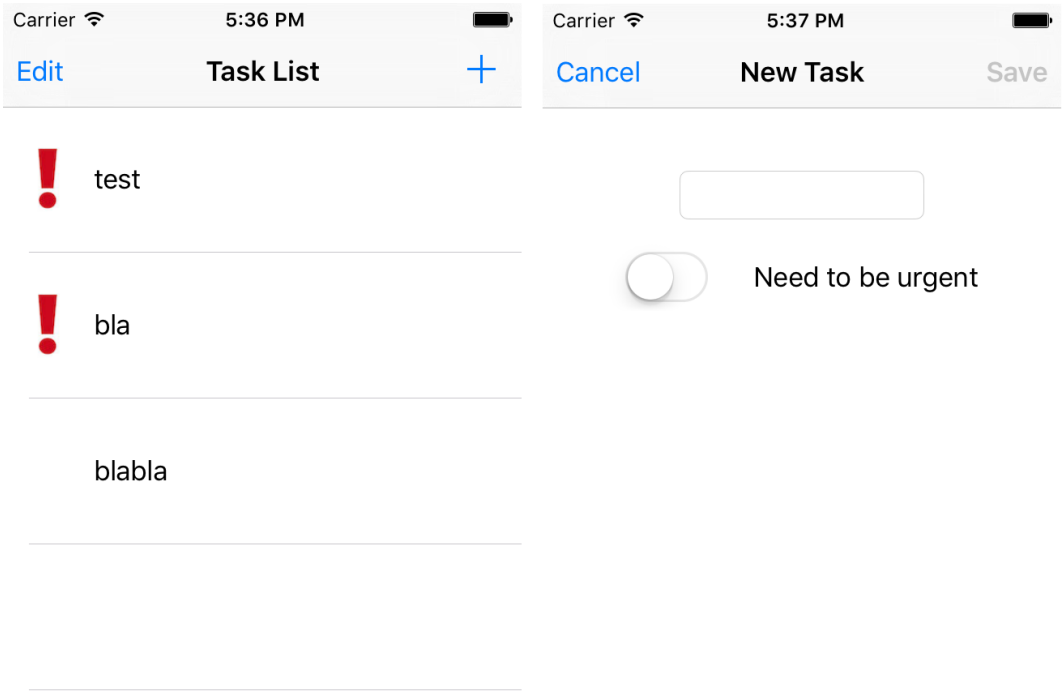

FIGURE 1 – Vues de l'application réalisée

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

Suppression d'une tâche :
Pour supprimer une tâche de la liste, deux procédés sont possibles. On peut soit cliquer sur le bouton Edit, qui affiche un symbole "-" à côté de chaque tâche, soit directement faire un swipe sur la gauche de la tâche. Les deux méthodes affichent le bouton delete qui enlèvera l'entrée de la liste.
Avec les deux méthodes, le bouton Edit se transforme en Done pour terminer l'action.

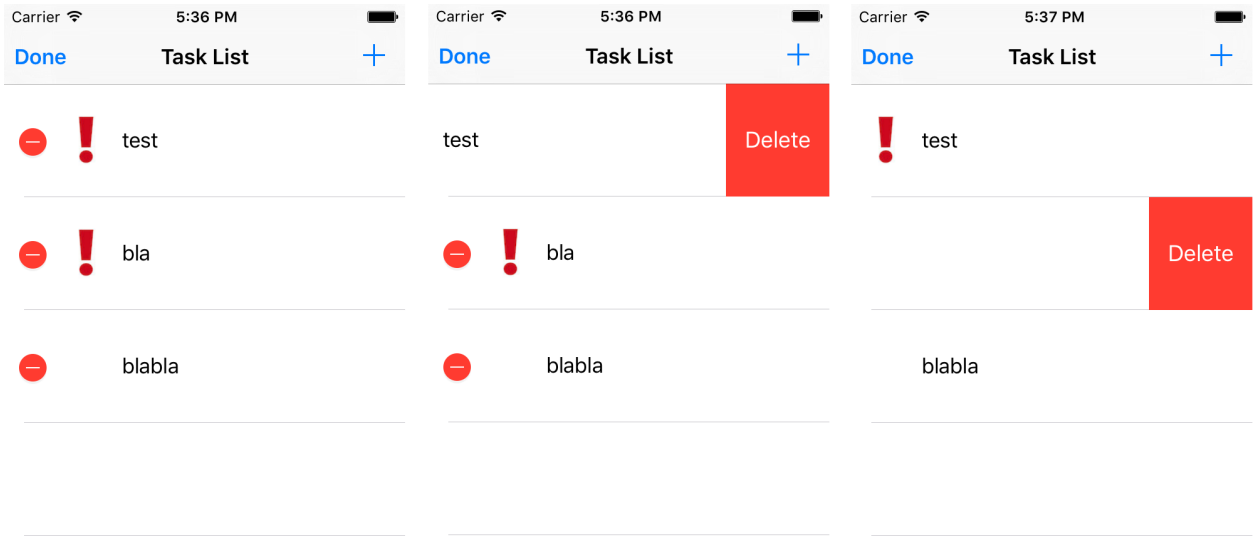

FIGURE 2 – Edit method (left and middle) - swipe method (right)

Edition/ Ajout d'une tâche :
Quand on édite une tâche, ses informations sont reportées dans la nouvelle vue, alors que quand on en ajoute une, la nouvelle vue est vide. La liste est ensuite mise à jour.

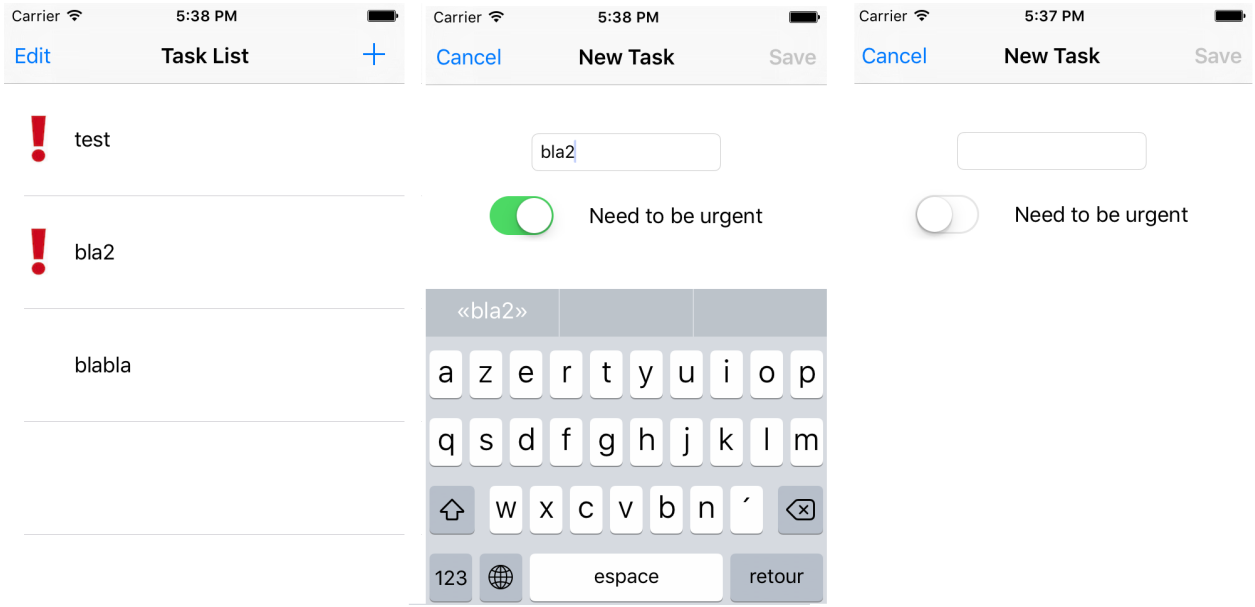

FIGURE 3 – Edit task (left and middle) - Add task (right)

Sauvegarde des tâches :
Lorsque l'on relance l'application, les tâches que l'on a créées précédemment sont présentes.

2.0.2 Cas pratique 2

Tous les points exigés par la donnée ont été réalisés sauf la partie concernant la musique. Voici un aperçu des différentes vues du projet :

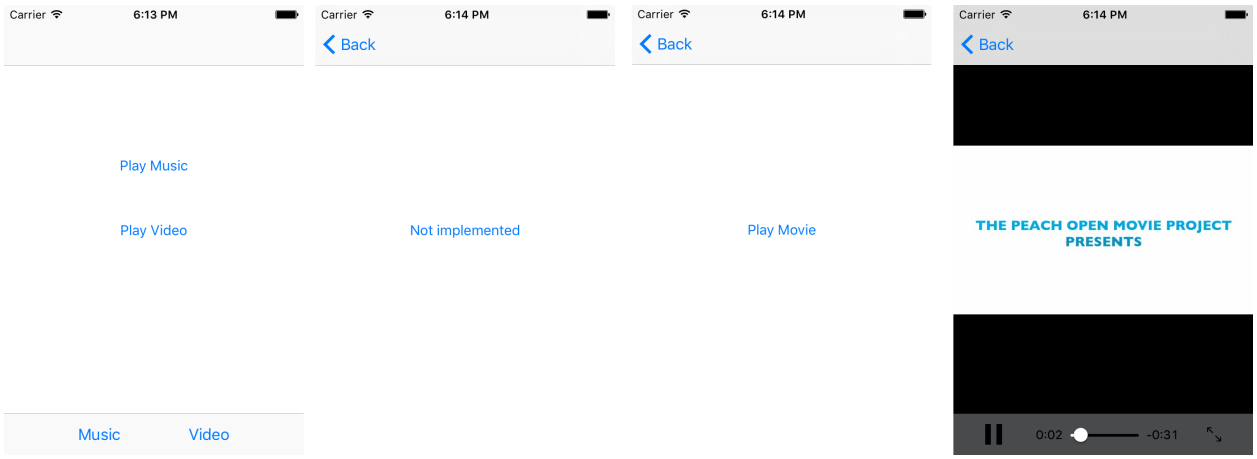

FIGURE 4 – Vues de l'application réalisée

Source de la vidéo :
http://jplayer.org/video/m4v/Big_Buck_Bunny_Trailer.m4v

Le bouton Play Music, le Music de la toolbar ainsi qu'un swipe à droite ouvre la liste des musique. Cette vue n'étant pas implémentée, un simple message est affiché.
Le bouton Play Video, le Video de la toolbar ainsi qu'un swipe à gauche ouvre la vue pour lancer une vidéo.
Cette dernière contient un simple bouton qui va lancer la vidéo dans une autre vue. Le lecteur a été implémenté pour lire la vidéo en boucle. Le bouton stop et play fonctionnent correctement. Le bouton done par contre ne revient pas à la vue précédente, il stop simplement la lecture. Cette particularité est due au fait que le lecteur n'est pas implémenté directement dans la vue au bouton Play movie, mais dans une vue séparée.
Pour la navigation entre les vues, chacune possède un bouton back qui permet de retourner à la précédente.

3 Problèmes rencontrés

Un "bug" s'est produit avec XCode. Il y avait deux versions différentes du même fichier .swift dans le projet pour une raison inconnue. Le storyboard en affichait une et le projet en compilait une autre. Ce problème nous a fait perdre beaucoup de temps. Il a été découvert en introduisant une erreur dans le fichier et le projet compilait quand même. Le problème a été résolu en copiant le même code dans les deux versions.

4 Réponse aux questions

4.1 Question 1

Donnée : Expliquez avec vos mots, quelle est l'utilité d'identifier la segue.

Dans le projet, on a utilisé des segues "identifiées" dans les méthodes preparForSegue. Cela a permis en fonction de quelle segue a déclenché l'action de passer ou non des informations à la prochaine vue.

4.2 Question 2

Donnée : Expliquez l'utilité d'une « unwind segue ».

Elle permet de quitter une vue et de revenir simplement à une autre en appelant une méthode spécifique de cette dernière. L'avantage est que l'on sait exactement à quel endroit du code on revient dans l'autre vue, cela permet de faire une action spécifique comme par exemple mettre à jour le contenu d'une TableView.

4.3 Question 3

Donnée : Expliquez avec vos mots, qu'est-ce qu'une structure.

Une structure est semblable à une énumération. Ses propriétés sont semblables aux classes. Le différence est que la struct fonctionne par copie alors que les classes utilisent des références.

4.4 Question 4

Donnée : Expliquez le principe de « Convenience » avec vos propres mots.

Le mot clé convenience est utilisé pour les initialisations de classes. Cela permet d'offrir à l'utilisateur plusieurs manière d'initialiser la classes en passant un nombre de paramètres variables et définir des valeurs par défaut. Voici un exemple plus parlant :

```
init(content: String, sender: String, recipient: String)
{
  self.content = content
  //Rule 1: Calling designated Initializer from immediate superclass
  super.init(sender: sender, recipient: recipient)
}
convenience init()
{
  //Rule 2: Calling another initializer in same class
  self.init(content: "")
}
convenience init(content: String)
{
  //Rule 2: Calling another initializer in same class
  self.init(content: content, sender: "Myself")
}
```

```
17 convenience init(content: String, sender: String)
18 {
19   //Rule 2 and 3:  Calling the Designated Initializer in same class
20   self.init(content: content, sender: sender, recipient: sender)
21 }
```

5 Conclusion

Ce projet a permis de constaté que le support en ligne pour le développement sur iOS avec Swift est moins bien fournie qu'avec Android, il y a moins de tutoriels. Cela vient sûrement du fait que la programmation était auparavant en Objective C. La plupart des exemples trouvés sont dans ce langague et non en Swift.
Une autre observation est que l'implémentation du MediaPlayer est relativement simple et ne demande pas beaucoup de code. Les boutons play,stop... sont déjà implémentés.