IF26 - développement d'applications mobiles

TP09 - iOS 3 - Interfaces iOS

Préparations

➢ Objectifs

- L'objectif de ce TP est la réalisation d'applications mobiles iOS en utilisant Xcode et le langage Swift. Nous allons plus spécifiquement étudier les interfaces Tab Bar et Table View.

➢ Connaissances et compétences

- Comprendre les composants clés d'une Tab Bar et d'une Table View
- Utiliser un tableau pour stocker et travailler avec des données
- Afficher les données dynamiques dans une Table View

Exercice 1 : Tab Bar

1. Lancez l'application Xcode.
2. Créez un nouveau projet : file → new project → Single View App → next
3. Configuration de la fenêtre "choose options for your new project"
    a. Product Name : votreNom_TabBar (exemple Lemercier_TabBar)
    b. Team : Add account ... (pas de changement)
    c. Organisation Name : if26
    d. Organisation Identifier : fr.utt.if26
    e. Language : Swift
    f. Devices : Universal
    g. Cases
        i. Uses Code Data : oui
        ii. Include Unit Tests : non
        iii. Include Ui Tests : non
    h. Next
4. Nous allons utiliser un Tab Bar Controller qui gère l'affichage d'une barre d'onglets et présente l'un des onglets (ViewController) correspondant à l'icône sélectionnée dans la barre de sélection.
5. Ouvrez le Main.storyboard
6. Supprimez la scène View Controller Scene qui existe par défaut et toute son arborescence.
7. Glissez un Tab Bar Controller sur le StoryBoard à partir de la bibliothèque d'objets. Trois scènes sont ajoutées au storyboard.
    a. Item 2 Scene
    b. Item 1 Scene
    c. Tab Bar Controller Scene

8. Nous allons perdu le point d'entrée du projet en supprimant le View Controller initial. Remarque importante : sans cette information, l'application compile mais n'affiche rien à l'écran au démarrage. Nous allons définir le Initial View Controller en désignant le Tab Bar Controller. Dans le Main.storyboard, sélectionnez la scène du Tab Bar Controller, ouvrez l'inspecteur d'attribut et cochez la case "Is Initial View Controller". Une flèche de gauche à droite sur le Tab Bar Controller permet de valider cette configuration.
9. Ajoutez un label IF26 d'une taille de font de 30, en rouge dans la vue de l'Item 1.
10. Même chose dans l'item 2 avec un label LO07 (un copier-coller suffit).
11. Modifiez les labels dans la barre de sélection en bas pour la cohérence du projet (mettre IF26 et LO07).
12. Lancez l'exécution de votre application pour validation.
13. Ajoutez une nouvelle scène identique aux deux premières avec un label défini avec NF19. Pour cela :
    a. Glissez un nouveau View Controller sur le StoryBoard
    b. Copiez l'un des labels dans ce nouveau controller avec comme nouvelle valeur NF19
    c. Créez un segue de relation (Relationship Segue / view Controllers) qui modélise une relation parent/enfant entre le Tab Bar Controller en le nouveau View Controller
    d. Modifiez le label dans la barre de sélection.
14. Lancez l'exécution de votre application pour validation.

Exercice 2 : Liste d'éléments avec Table View Controller

1. Dans ce projet, nous allons utiliser une Table View qui est le contrôle le plus fréquemment utilisé pour afficher une liste d'éléments. La classe UITableView est une sous-classe de UIScrollView ce qui permettra un défilement des informations. L'utilisation d'un Table View Controller permet de régler plusieurs paramètres :
    a. L'affichage de la Table View
    b. La définition du modèle des cellules à afficher
    c. L'origine des données
    d. ...
2. Créez un nouveau projet (Single View Application) dont le product Name est votreNom_TableView (exemple Lemercier_TableView)
3. Ouvrez le Main.storyboard et supprimer la ViewControllerScene.
4. Glissez-déposez un Table View Controller sur le storyboard.
5. Définissez ce contrôleur comme point d'entrée de l'application en cochant la case "Is Initial View Controller" dans l'inspecteur des attributs.
6. Une nouvelle arborescence apparaît sous le Table View controller Scene. Dans la scène, sélectionnez le "Table View" et configurez la propriété Content avec la valeur Dynamic Prototypes.
7. Dans la scene, configurez le "Table View Cell" :
    a. Style : Subtitle
    b. Identifier : celluleModule (Cette valeur correspond à l'identifiant de réutilisation qui sera utilisé dans le code de la source de donnée)
    c. Selection : Default

   d. Accessory : Disclosure Indicator
   e. Editing Acc. : None
   f. Focus Style : Default

8. Nous allons maintenant créer une classe dérivée d'un Table View Controller.
   a. Supprimez le fichier ViewController.swift existant.
   b. Créez un nouveau fichier de type iOS > Cocoa Touch Class
      i. nom : TestTableViewController
      ii. SubClass of : UITableViewController
   c. Ouvrez le Main.storyboard, sélectionnez la scene du Table View Controller et associez le via le champ Class avec la classe TestTableViewController.
9. Lancez l'exécution de votre application pour validation.
10. Nous allons maintenant définir la source des données. C'est l'élément TestTableViewController qui joue le rôle de source de données pour la Table View. Commençons par un test avec des valeurs statiques. Nous allons créer une liste de 3 sections contenant 5 cellules.
    a. Ouvrez le fichier TestTableViewController.swift
    b. Ajoutez au début de la classe un attribut nommé identifiantModuleCellule = "celluleModule". Attention, cet identifiant doit correspondre à celui défini dans le Table View Cell.
    c. Modifiez le code pour que la méthode numberOfSections retourne la valeur 3.
    d. Modifiez le code pour que la méthode tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) retourne la valeur 5.
    e. Ajoutez une méthode override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
    f. Complétez cette méthode avec le code suivant :

```
let cell = tableView.dequeueReusableCell(withIdentifier: identifiantModuleCellule, for: indexPath)
cell.textLabel?.text = "Cellule \(indexPath.row)"
cell.detailTextLabel?.text = "Section \(indexPath.section)"
return cell
```

    g. Quel est le rôle de cette méthode ?
    h. Remarque : l'opérateur "?" permet un chaînage optionnel, si la valeur est nil alors aucune opération n'a lieu.
11. Lancez l'exécution de votre application pour validation.
12. Nous souhaiterions que les en-têtes de section soient visibles maintenant. Ajoutez la méthode suivante :

```
override func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
  return "Section \(section)"
}
```

13. Lancez l'exécution de votre application pour validation.

Exercice 3 : Table View Controller avec une liste de Modules de l'UTT

1. L'objectif de cet exercice est de créer une liste d'objets Module en utilisant les éléments définis dans l'exercice 2.
2. Récupérez les fichiers Resultat.swift, Module.swift et Curus.swift sur le site elearning.

3. Créez un nouveau fichier de type iOS > Cocoa Touch Class
    a. nom : ModuleTableViewController
    b. SubClass of : UITableViewController
4. Nous allons remplacer le TestTableViewController. Ouvrez le Main.storyboard, sélectionnez la scene du Table View Controller et associez le via le champ Class avec la classe ModuleTableViewController.
5. Dans le code de la classe ModuleTableViewController :
    a. Ajoutez un attribut cursus correspondant à un tableau d'objet Module initialisé à un tableau vide : var cursus: [Module] = []
    b. Dans la méthode viewDidLoad() vous allez initialiser l'attribut cursus à l'aide de la méthode getModules() de la classe Cursus.
    c. La méthode numberOfSections retourne 1
    d. La méthode tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) retourne le nombre de Module contenu dans l'attribut cursus.
    e. La méthode tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? retourne nil
    f. Nous devez modifier cette méthode pour que le textLabel corresponde au sigle et que le detailTextLabel à une concaténation de la catégorie, du nombre de crédit et du résultat.
6. Lancez l'exécution de votre application pour validation.
7. Proposez des fonctionnalités supplémentaires à cette application.

Exercice final : devoir

1. Pour chaque étudiant, proposez soit une question supplémentaire à l'un des exercices de ce TD/TP, soit un commentaire, soit un QCM ou soit une ressource disponible sur Internet.
2. Vous transmettrez votre proposition via le devoir moodle sur le site e-learning du module.