IF26 - développement d'applications mobiles

TP10 - iOS 4 - 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 la persistance des données en local sur le terminal.

➢ Connaissances et compétences

- Maîtriser l'utilisation de boîtes de dialogue Alert (UIAlertController)
- Apprendre à construire une connexion à une base SQLite embarquée.
- Apprendre à exécuter les commandes classique pour manipuler une base de données relationnelle : création d'une table, insertion de tuples, mise à jours, suppression d'un tuple et requêtes SQL.
- Connaître l'utilisation d'un framework (SQLite.swift)

Exercice 1 : Préparation de l'interface

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_Users (exemple Lemercier_Users)
    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. Ouvrez le Main.storyboard
5. Ajoutez cinq boutons au ViewController par défaut :
    a. Create Table
    b. Insert User
    c. List User
    d. Update User (mail)
    e. Delete User
6. Il n'est pas nécessaire de disposer des variables associées aux boutons. Nous allons uniquement créer les 5 méthodes associées aux boutons sachant que :
    a. Les noms des méthodes sont : createTable, insertUser, listUser, ...

   b. Les méthodes n'ont pas de paramètres, pour cela il suffit de définir le paramètre "argument" à la valeur "none"

7. Ajoutez un affichage à toutes les méthodes comme sur l'exemple suivant :

```
@IBAction func createTable() {
  print ("Bouton create")
}
```

8. Une façon simple pour récupérer des données des utilisateurs est d'utiliser une boîte de dialogue Alert. L'extrait de code suivant (méthode insertUser) permet de proposer une fenêtre contenant 2 zones de saisies de type texte :

```
@IBAction func insertUser() {
   print ("Bouton insert")
   let alert = UIAlertController(title: "Insert User", message: nil, preferredStyle: .alert)
   alert.addTextField { (tf) in tf.placeholder = "Name"}
   alert.addTextField { (tf) in tf.placeholder = "Email"}
   let action = UIAlertAction(title: "Submit", style: .default) { (_) in
      guard
         let name = alert.textFields?.first?.text,
         let email = alert.textFields?.last?.text
      else { return }
      print (name)
      print (email)
   }
   alert.addAction(action)
   present(alert, animated: true, completion: nil)
}
```

9. En utilisant l'exemple de la méthode insertUser(), complétez les méthodes updateUser() et deleteUser() sachant que l'on souhaite récupérer que le champ ID (de type texte) d'un user.

Exercice 2 : Intégration du projet GitHub SQLite.swift

1. Récupérez le projet SQLite.swift sur GitHub : https://github.com/stephencelis/SQLite.swift
2. Pour information, nous allons travailler avec une table users définie par :

```
CREATE TABLE "users" (
        "id" INTEGER PRIMARY KEY NOT NULL,
        "name" TEXT,
        "email" TEXT NOT NULL UNIQUE
)
```

3. En bas de la page du projet, vous trouverez un exemple d'utilisation de ce framework qui permet de gagner beaucoup de temps.
   a. Repérez comment définir une connexion
   b. Identifiez la déclaration d'une table et de ses champs (attributs)
   c. Identifiez l'instruction de création d'une table
   d. Notez la syntaxe : let insert = users.insert(name <- "Alice", email <- "alice@mac.com")
4. Téléchargez le projet (SQLite.swift-master) sur votre Mac et dézippez-le.
5. Recherchez le fichier SQLite.xcodeproj et glissez-déposez ce fichier dans votre projet en le plaçant sous le répertoire Products.
6. A gauche dans l'arborescence des fichiers, sélectionnez votre projet (premier élément).
   a. Recherchez en bas de la page, la section "Linked Frameworks and Libraries"
   b. Faire "+"

   c. Ajoutez le dossier SQlite.framework from 'SQLite iOS' (c'est normalement le premier de la liste)
   d. Recherchez la section "Embedded Binaries"
   e. Faire "+"
   f. Sélectionnez dans "Products" le dossier SQLite.framework iOS
7. Au final vous devez avoir un framework pour la section "Embedded ..." et 2 pour la section "Linked ...".

Exercice 3 : Ajout des instructions SQLite au projet

1. Recherchez sur Google la définition d'une closure en swift. Quelle est son rôle ?
2. Recherchez sur Google la définition de l'instruction guard en swift. Quelle est son rôle ?
3. La ressource YouTube suivante permet de finir le projet. Vous pouvez commencer le film à 5m45s du début.
   a. Nom de la ressource : SQLite: Local Database | Swift 4, Xcode 9 de Kilo Loco
   b. https://youtu.be/c4wLS9py1rU

Exercice 4 : Finir les projets iOS

1. Mettrez à profit le temps restant pour finaliser les projets précédents.
2. Terminez l'ensemble des devoirs restants (8, 9, et 10).

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.