iOS Development

- Swift Fundamentals

Swift fundamentals - I

➔ Xcode => iOS Development tool
➔ Download Xcode
    ➔ Use the app store application
    ➔ Or Use below url
https://developer.apple.com/xcode

iOS Development

- Swift Fundamentals

Swift fundamentals - II

➔ iOS programming language
➔ Start Playground (File -> New -> Plaground -> Blank)
➔ Variables (let or var, standard types)
➔ Optional => nullable variable
➔ Loops and condition statements
➔ Lists and Maps

```
// variables
var total: Double
total = 150
let discount = 0.05
total -= total * discount
print(total)
// optional
var name: String?
print(name?.first)
if let realName = name {
    print(realName.first)
}
// list maps, loops, condition statements
var paragraph = [String]()
paragraph.append("ios")
paragraph.append("android")
for word in paragraph {
    print(word)
}
for index in 0...paragraph.count-1 {
    print(paragraph[index])
}
var i = 0
while i < paragraph.count {
    print(paragraph[i])
    i += 1
}
if paragraph.count > 1 {
    print(paragraph[0])
}
var traductions = [String:String]()
traductions["hello"] = "bonjour"
print(traductions.values)
```

iOS Development

- Swift Fundamentals

Swift fundamentals - III

➔ OOP
- ➔ Functions
- ➔ Initializers
- ➔ Properties
- ➔ Inheritance
- ➔ Delegate pattern => Interface

```
//POO
class Product {
    private var name : String
    private var price : Double

    init(name: String, price : Double) {
        self.name = name
        self.price = price
    }

    func toString() -> String {
        return "\(self.name) - price \(self.price)"
    }
}
let bread = Product(name: "bread", price: 2.5)
print(bread.toString())
protocol PerishableProductDelegate {
    func conditioning() -> String
}
class PerishableProduct : Product, PerishableProductDelegate {
    func conditioning() -> String {
        return "Ice"
    }
}
let fish = PerishableProduct(name: "fish", price: 12)
print(fish.toString())
print(fish.conditioning())
```

iOS Development

- Swift Fundamentals

Exercice SF-1
Soit la liste d'entiers suivants : -12; 23; -54; 89; -90; 67; 87; -21
Afficher les entiers positifs de cette liste

Exercice SF-2
Soit la liste d'entiers suivants : liste (12; 23; 54; 89; 90; 67; 87)

1. Calculer la somme des éléments de cette liste
2. Calculer la moyenne des valeurs de la liste
3. Calculer le maximum des valeurs de la liste

Exercice SF-3
Soit une liste de voitures : voitures(citroen c4, mazda, citroen picasso, mercedez classe c)

1. Afficher le nombre de voitures dans la liste
2. Afficher les voitures de la marque citroen
3. Afficher le nombre de voiture citroen

iOS Development

- Swift Fundamentals

Exercice SF-4

Soit un objet Personne défini par son **nom** de type chaine de caractères et son **âge** de type entier.

1. Créer la classe Personne avec ses attributs et les méthodes de base (getter et setter) y compris un constructeur .
2. Créer une méthode display() permettant d'afficher une Personne
3. Créer une méthode lifeStatus() qui en fonction de l'âge de la personne retourne une chaine de caractère indiquant si la personne est un enfant, un adolescent ou bien un adulte :
   - âge >= 10 => «Enfant »
   - âge >10 et âge <18 => « Adolescent »
   - âge >= 18 => « Adulte »

4. Créer les personne suivantes : personneUn(«Nico», 9), personneDeux(« Lyne», 23)
5. Afficher les personnes créées
6. Indiquer le « lifeStatus » de chacune des personnes créées.
7. Créer une liste de personnes composée de 10 personnes et afficher la personne la plus agée

iOS Development

- Swift Fundamentals

Exercice SF5

Gestion de notes

Etudiants			
Matricule	**Nom**	**Prénoms**	**Moyenne**
M0001	Klein	Calvin	
M0002	Nadal	Rafael	
M0003	Federer	Roger	
M0004	Maradoza	Diedo	

Matières	
Libellé	**Coefficient**
Network	2
Developmen	3
Cloud	4

Notes		
Matricule	**Matière**	**Note**
M0001	Network	8
M0002	Network	15
M0004	Network	7
M0001	Development	8
M0002	Development	1
M0004	Development	7
M0001	Cloud	2
M0002	Cloud	5
M0004	Cloud	19

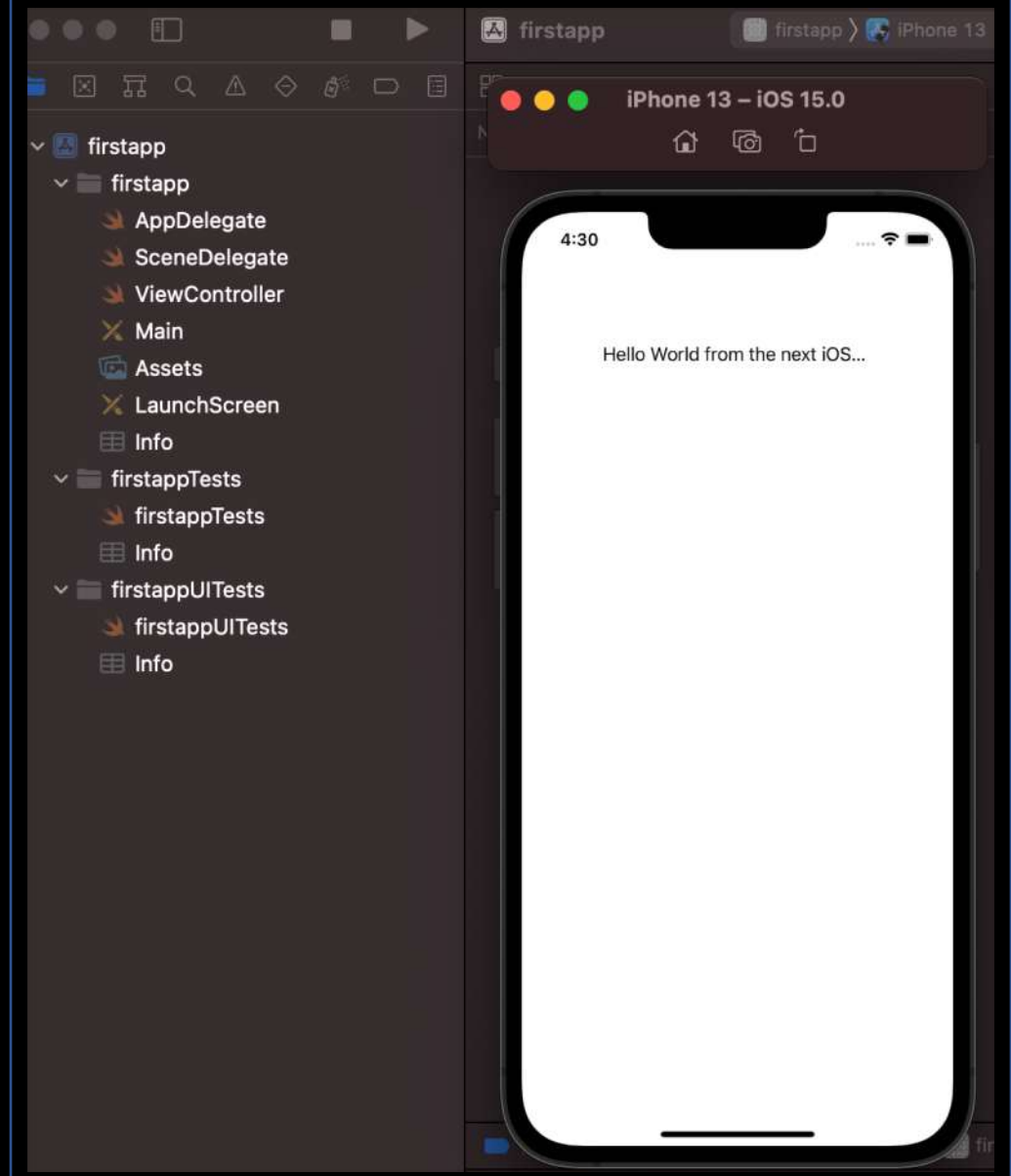
	Travail à faire
1	Identifier et créer les classes
2	Saisir les données
3	Afficher les étudiants n'ayant aucune note
4	Calculer, enregistrer la moyenne de chaque étudiant
5	Afficher les moyennes des étudiants
6	Calculer et afficher la moyenne de la classe

Mobile App Development

- Download or clone the project below
  https://github.com/dovene/first-app-ios/tree/helloworld

- Open the downloaded project using Xcode (open the firstapp.xcodeproj file)

- Select a simulator and run the project



iOS Development

- Mobile App Development

iOS Mobile App Essentials

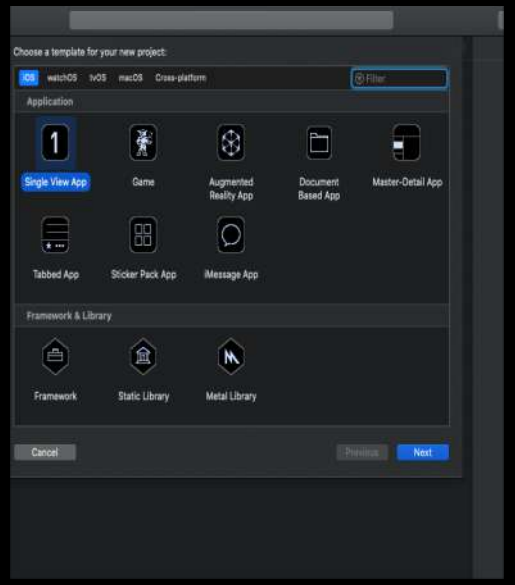
- Swift and Objective C programming languages
- View Controller => Swift file that handles all the interactions between the user and the application
- Storyboard => represents the UI (the view layer in MVC architecture)
- Autolayout => guidelines of view items positioning
- AppDelegate or SceneDelegate => Swift files that hold application main configuration
- Info.plist => XML file for application configuration

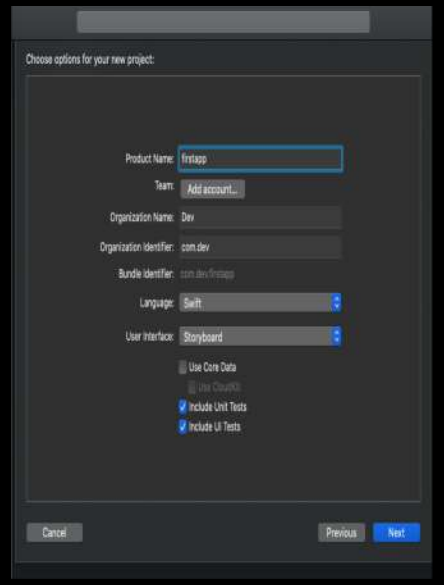
iOS Development

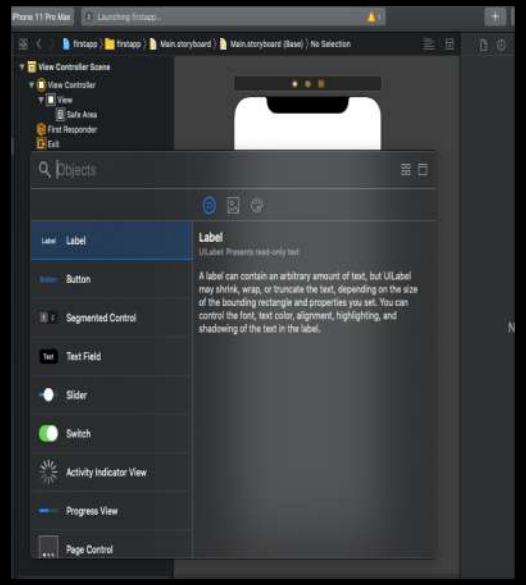
- Mobile App Development

Create your first iOS app – Part 1 Getting Started

1. Start Xcode -> Create Project -> Single View App -> Select storyboard and input app product name
2. Open the Main.storyboard file and add a label (make sure it is well positioned using Autolayout) to display Hello World
3. Run the application

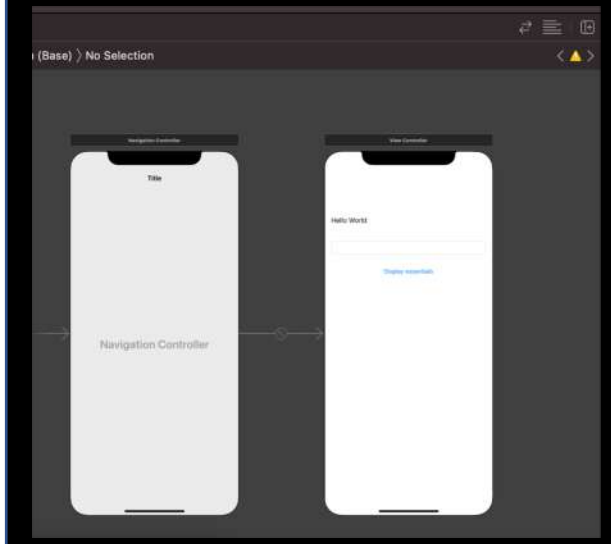


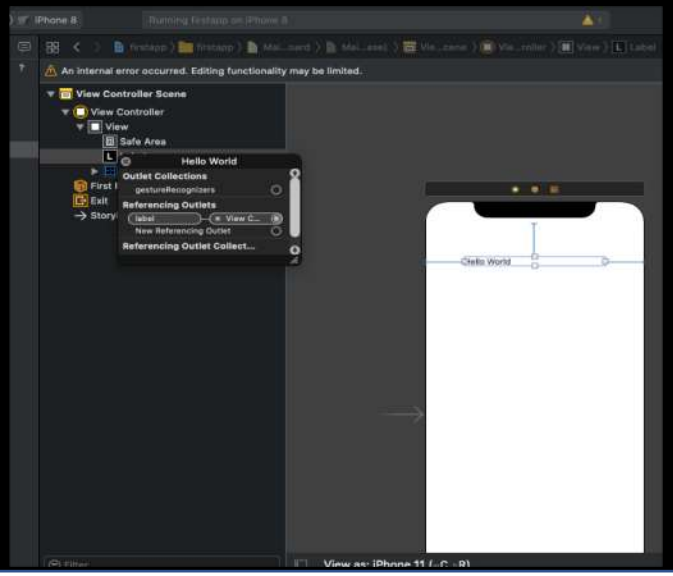


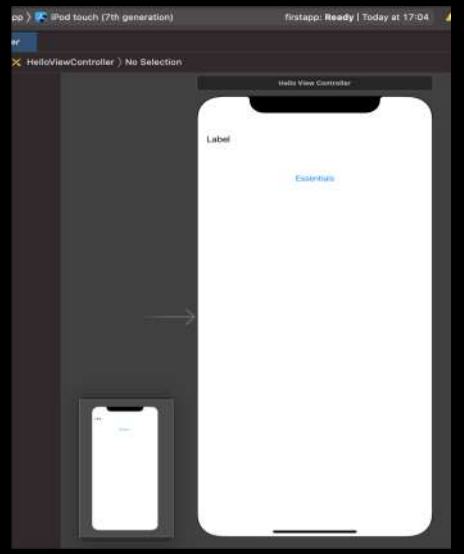


Create your first iOS app – Part 2 View items and navigation

1. Add an input field below the « Hello World » text and a button => our goal is to display Hello World and the value of the input field on a second screen
2. Embed the view controller within a navigation controller
3. Create a new storyboard file and its related view controller to be displayed on the click of the previously added button
4. Create IBOutlets in the ViewController class for all the view items and set the binding in the storyboard







Create your first iOS app – Part 3 View items, forms and navigation

1. Use addTarget method of the button to handle click events
2. Dispay another viewcontroller using pushViewController method of NavigationController
3. Use an instance variable from the second view controller to pass the information to be displayed

iOS Development

- Mobile App Development

```swift
import UIKit

class ViewController: UIViewController {

    // MARK: - Properties

    @IBOutlet private weak var label: UILabel!
    @IBOutlet private weak var button: UIButton!
    @IBOutlet private weak var usernameTextField: UITextField!


    // MARK: - View Life Cycle

    override func viewDidLoad() {
        super.viewDidLoad()
        title = "iOS Essentials"
        label.text = "Hello World from the next iOS rock star"
        button.addTarget(self, action: #selector(launchHelloVC), for: .touchUpInside)
    }

    override func viewWillDisappear(_ animated: Bool) {
        usernameTextField.endEditing(true)
    }

    // MARK: - Actions

    @objc
    func launchHelloVC(){
        let storyBoard = UIStoryboard(name: "HelloViewController", bundle: nil)
        let helloViewController = storyBoard.instantiateViewController(withIdentifier:
            "HelloViewController") as! HelloViewController
        helloViewController.welcomeMessage = usernameTextField.text
        self.navigationController?.pushViewController(helloViewController, animated: true)
    }
```

```swift
import Foundation
import UIKit

class HelloViewController: UIViewController {

    // MARK: - Properties

    public var welcomeMessage : String?
    @IBOutlet private weak var welcomeLabel: UILabel!
    @IBOutlet private weak var essentialsButton: UIButton!

    // MARK: - View Life Cycle

    override func viewDidLoad() {
        super.viewDidLoad()
        title = "Welcome"
        welcomeLabel.text = "Welcome \(welcomeMessage ?? "")"
        essentialsButton.addTarget(self, action: #selector(launchEssentialsListVC), for: .touchUpInside)
    }



    // MARK: - Actions
    @objc
    func launchEssentialsListVC(){
        let storyBoard = UIStoryboard(name: "EssentialsListViewController", bundle: nil)
        let essentialsListViewController = storyBoard.instantiateViewController(withIdentifier:
            "EssentialsListViewController") as! EssentialsListViewController
    self.navigationController?.pushViewController(essentialsListViewController, animated: true)
    }

}
```

Create your first iOS app – Part 4 List Screen

1. We are going to add a third screen to diplay our iOS essentials list
2. Add a new stoyboard file holding a tableview and the related swift viewcontroller
3. Add a xib file representing a cell of the list and the related swift class that extends a uitablecellview
4. Configure tableview delegate and datasource, register the cellview
5. Make all necessary implementation to display the essentials list

iOS Development

- Mobile App Development

```swift
protocol EssentialTableViewCellDelegate {
    func getSelectedEssential(essential: Essential)
}

class EssentialTableViewCell: UITableViewCell {

    @IBOutlet private weak var titleLabel: UILabel!
    @IBOutlet private weak var descriptionLabel: UILabel!
    private var essential = Essential()
    public var essentialTableViewCellDelegate: EssentialTableViewCellDelegate?


    override func awakeFromNib() {
        super.awakeFromNib()
        let tapCell = UITapGestureRecognizer(target: self, action: #selector(handleCellTap))
        addGestureRecognizer(tapCell)
    }

    @objc
    func handleCellTap(){
        essentialTableViewCellDelegate?.getSelectedEssential(essential: essential)
    }

    public func setEssential(_ essential: Essential) {
        self.essential = essential
        descriptionLabel.text = essential.description
        titleLabel.text = essential.title
    }
```

```swift
12  class EssentialsListViewController: UIViewController, UITableViewDelegate, UITableViewDataSource,
        EssentialTableViewCellDelegate {
    @IBOutlet private weak var tableView: UITableView!
14      private var essentials = [Essential]()
15      private let viewCellName: String = "EssentialTableViewCell"
16
17      // MARK: - View Life Cycle
18      override func viewDidLoad() {
19          super.viewDidLoad()
20          essentials.append(Essential("mock tile", "mock description"))
21          essentials.append(Essential("mock dev title", "mock dev description"))
22
23          tableView.register(UINib(nibName: viewCellName, bundle: nil), forCellReuseIdentifier: viewCellName)
24          tableView.dataSource = self
25          tableView.delegate = self
26          tableView.tableFooterView = UIView()
27      }
28
29      func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
30          essentials.count
31      }
32
33      func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
34          let cell = tableView.dequeueReusableCell(withIdentifier: viewCellName) as! EssentialTableViewCell
35          cell.setEssential(essentials[indexPath.row])
36          cell.essentialTableViewCellDelegate = self
37          return cell
38      }
39
40      func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
41          return 120
42      }
43
44      func getSelectedEssential(essential: Essential) {
45          let essentialDetailSB = UIStoryboard(name: "EssentialDetailViewController", bundle: nil)
46          let essentialDetailViewController = essentialDetailSB.instantiateViewController(identifier:
                "EssentialDetailViewController") as! EssentialDetailViewController
47          essentialDetailViewController.essential = essential
48          // display it modally for variety sake !
49          present(essentialDetailViewController, animated: true, completion: nil)
```

Create your first iOS app – Part 5 Detail Screen

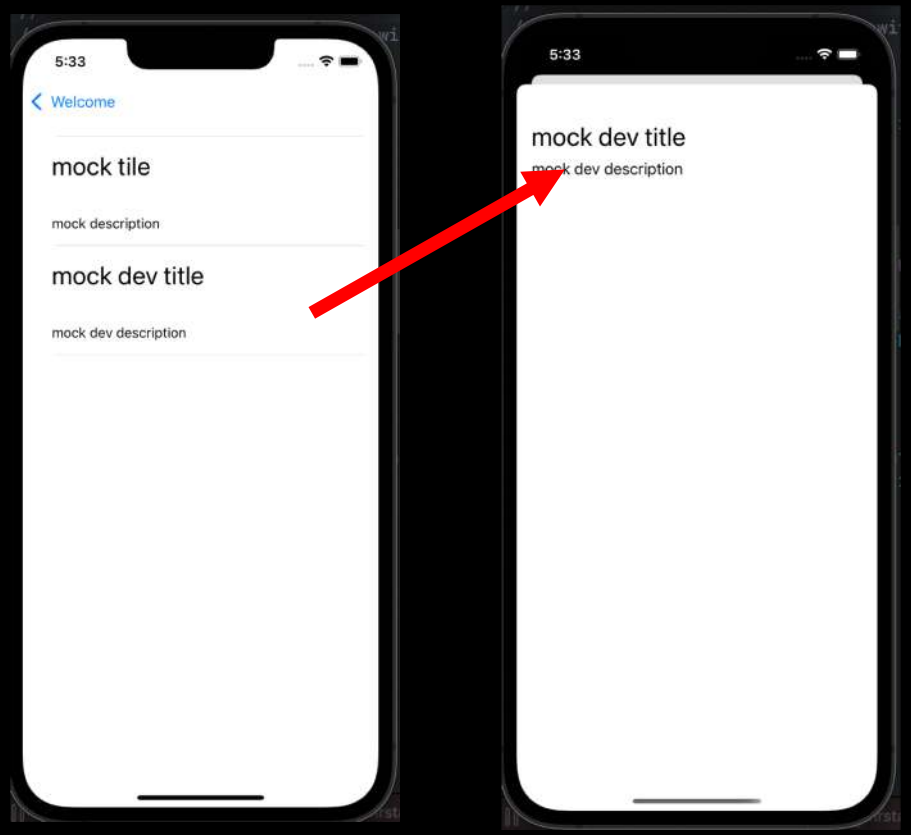
iOS Development

- Mobile App Development

1. We are going to add a fourth screen to diplay details of an essential cell view on user selection from the essentials list
2. Add the corresponding storyboard and view controller

Get a working copy from https://github.com/dovene/first-app-ios/tree/feature/multi-screen-app

```
import Foundation
import UIKit

class EssentialDetailViewController: UIViewController {

    @IBOutlet private weak var titleLabel: UILabel!
    @IBOutlet private weak var descriptionLabel: UILabel!
    public var essential = Essential()

    override func viewDidLoad() {
        super.viewDidLoad()
        descriptionLabel.text = essential.description
        titleLabel.text = essential.title
    }
}
```



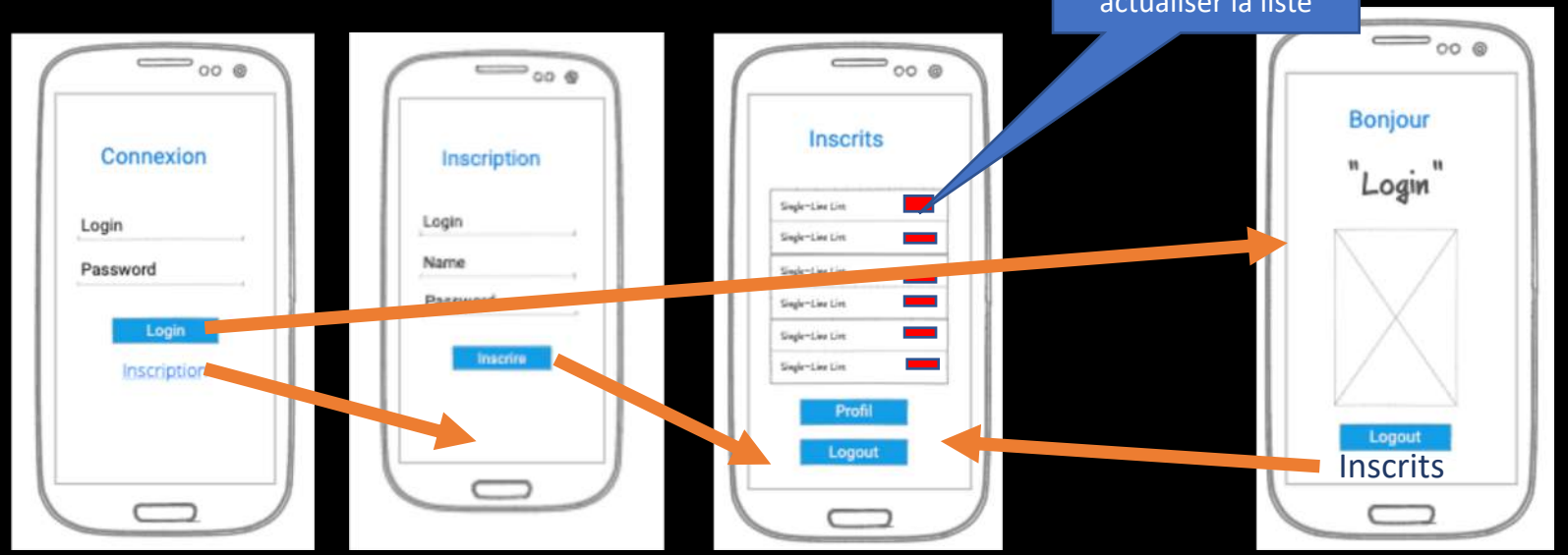
iOS Development

- iOS – TP

iOS TP

TP2 Authentication System

Vous êtes chargé d'intégrer un système d'authentification à une application existante. Les fonctionnalités attendus sont l'inscription, la connexion, l'affichage de la liste des utilisateurs et l'écran de profil d'un utilisateur.
Ci-dessous les maquettes du nouveau module.
L'image de profil est ajoutée via les Assets de l'application



iOS Development

- iOS – TP

iOS TP

TP1 - Mes Notes

L'objectif de cet exercice est de créer une mini application de calcul de moyenne.

1ère Partie

- Créer une classe ***Matiere*** caractérisée par un libellé et un coefficient.
- Créer une classe ***MoyenneParMatiere*** caractérisée par le libellé de la matière et la moyenne sur 20.
- Créer des formulaires de saisie des matières et des moyennes
- Afficher la liste des matières et des moyennes avec la possibilité de supprimer les saisies
- Ajouter un bouton sur l'écran des moyennes permettant de calculer et d'afficher la moyenne générale

L'écran d'accueil est composé d'un titre et de deux boutons permettant d'accéder aux formulaires de saisie des matières et des moyennes . Les écrans de liste apparaissent à la validation des formulaires de saisie.

2ème Partie

Ajouter la classe Etudiant caractérisée par un matricule, un nom et un prénom

Créer les écrans de saisie et d'affichage de la liste des étudiants

Faire les modifications nécessaires pour enregistrer les moyennes par étudiant.

Calculer et afficher la moyenne générale d'un étudiant connaissant son matricule.

iOS Development

- iOS – TP

iOS TP

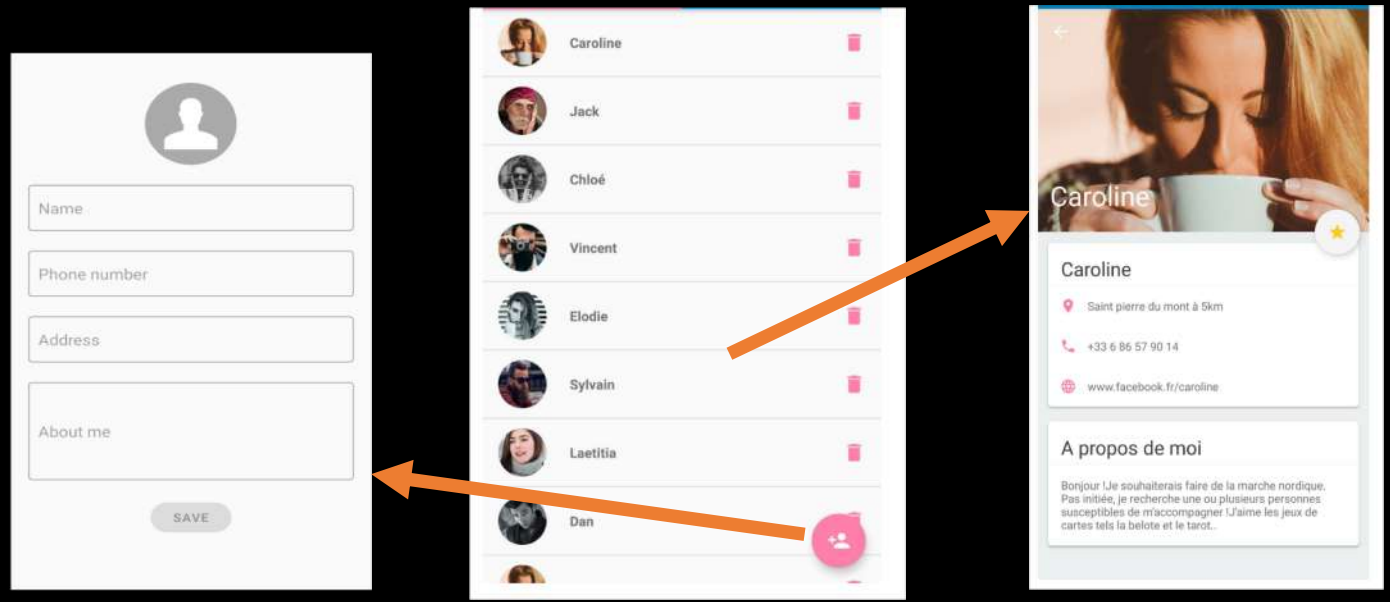
TP3 Mes Voisins

La mairie de votre ville vous confie les maquettes suivantes pour développer une première version de l'application Mes Voisins dont le MVP est le suivant :

- Ajout/Suppression de voisins
- Mise en favoris de voisins

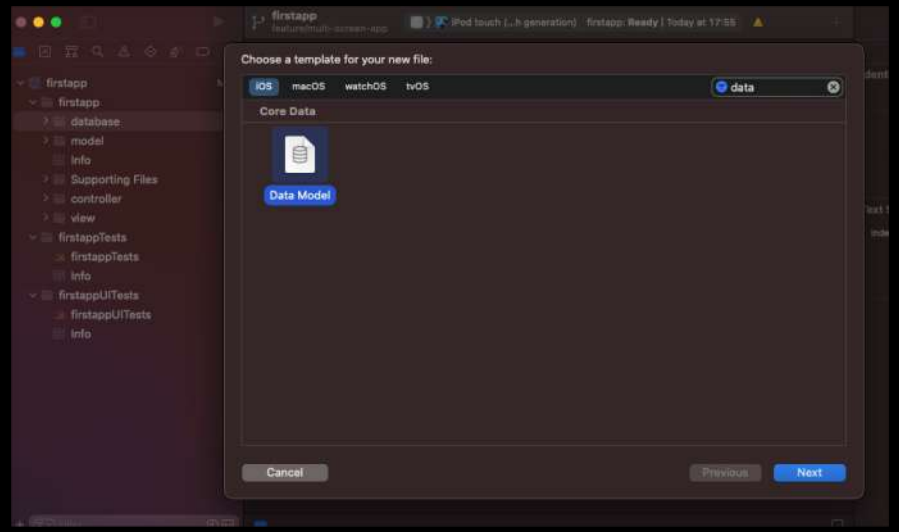
Chaque voisin est caractérisé par un id, un nom complet, un avatar, une description/A propos, un tel, une adresse, un site web éventuellement.

PS : Vous pouvez récupérer des avatars en ligne https://i.pravatar.ccdu du genre https://i.pravatar.cc/150?u=a042581f4e29026703d



iOS Data Persistence - CoreData

- → Persistence framework
- → Serialize data into SQL Lite database
- → Add Coredata to the project (File ->New -> File -> Data Model)



iOS Data Persistence - CoreData

→ Update App Delegate to get a PersistenceContainer

```
lazy var persistentContainer: NSPersistentContainer = {
    let container = NSPersistentContainer(name: "firstapp")
    container.loadPersistentStores(completionHandler: { (_, error) in
        if let error = error as NSError? {
            print("Unresolved error \(error), \(error.userInfo)")
        }
    })
    return container
}()

static var persistentContainer: NSPersistentContainer {
    return (UIApplication.shared.delegate as! AppDelegate).persistentContainer
}
```

iOS Data Persistence – CoreData

Save and retrieve iOS essentials using CoreData

- → Use a ManagedObjectContext to save and fetch a ManagedObject
- → Use ManagedObject method setValue(_ value: **Any**?, forKey key: String) to change and method value(forKey key: String) to get persisted object value

- → Get a working copy from https://github.com/dovene/first-app-ios/tree/feature/add-core-data

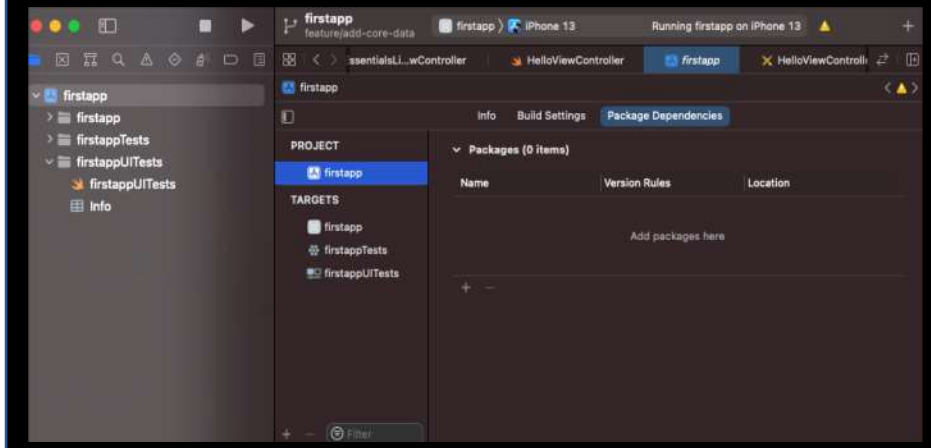
```
func saveEssential() {
    let managedContext =
    AppDelegate.persistentContainer.viewContext
    let essentialEntity =
    NSEntityDescription.entity(forEntityName: "EssentialEntity",
                               in: managedContext)!
    let essentialObject = NSManagedObject(entity: essentialEntity,
                                          insertInto: managedContext)
    essentialObject.setValue(nameTextField.text, forKey: "title")
    essentialObject.setValue(descriptionTextField.text, forKey: "comments")
    do {
        try managedContext.save()
    } catch let error as NSError {
        print("Could not save. \(error), \(error.userInfo)")
    }
}
```

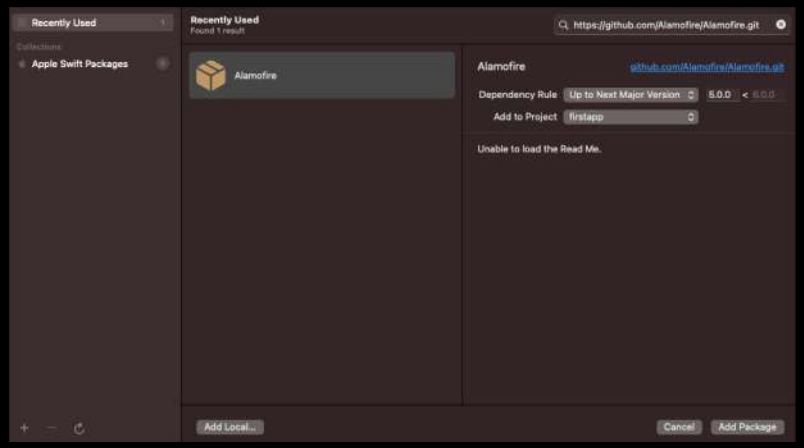
```
private var essentials = [NSManagedObject] ()

override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated)
      let managedContext =
        AppDelegate.persistentContainer.viewContext
      let fetchRequest =
        NSFetchRequest<NSManagedObject>(entityName: "EssentialEntity")
      do {
        essentials = try managedContext.fetch(fetchRequest)
      } catch let error as NSError {
        print("Could not fetch. \(error), \(error.userInfo)")
      }
}
```

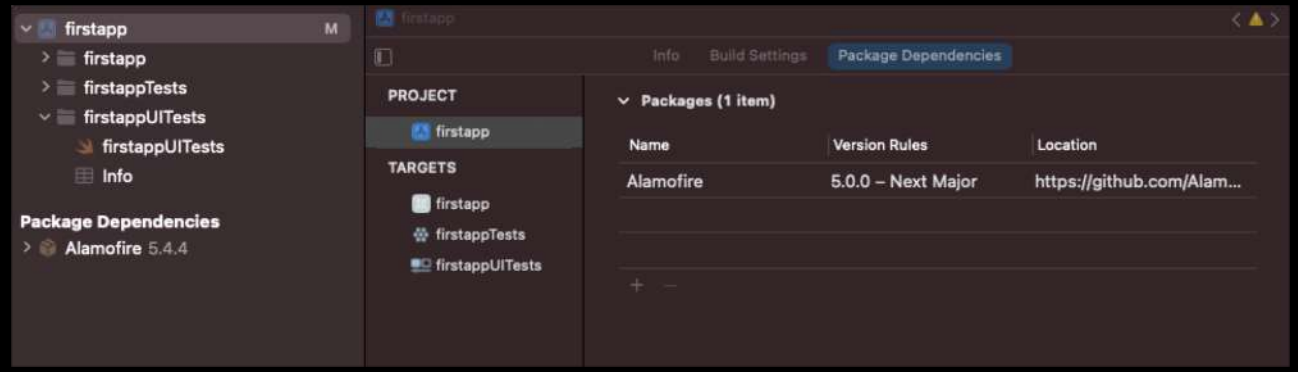
```
let essential = Essential(essentials[indexPath.row].value(forKey: "title")
    as! String, essentials[indexPath.row].value(forKey: "comments") as!
    String)
cell.setEssential(essential)
```

Networking - Alamofire

- → Alamofire => framework that handle http requests
- → Add Alamofire with Swift Package Manager

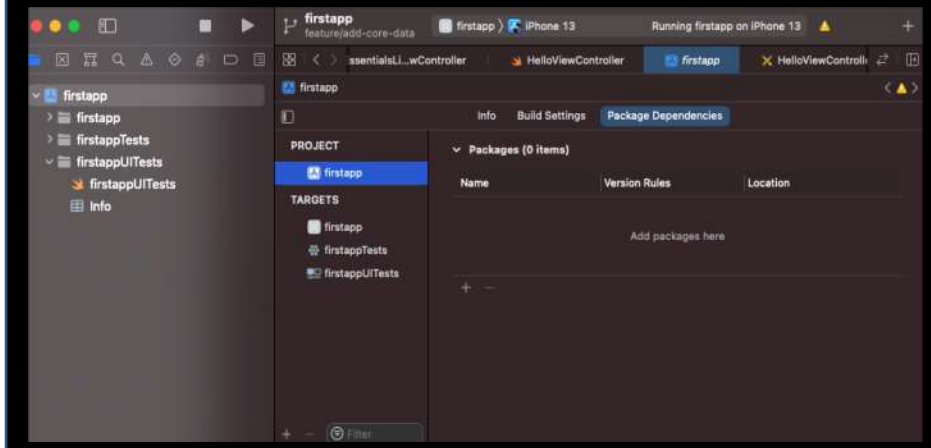


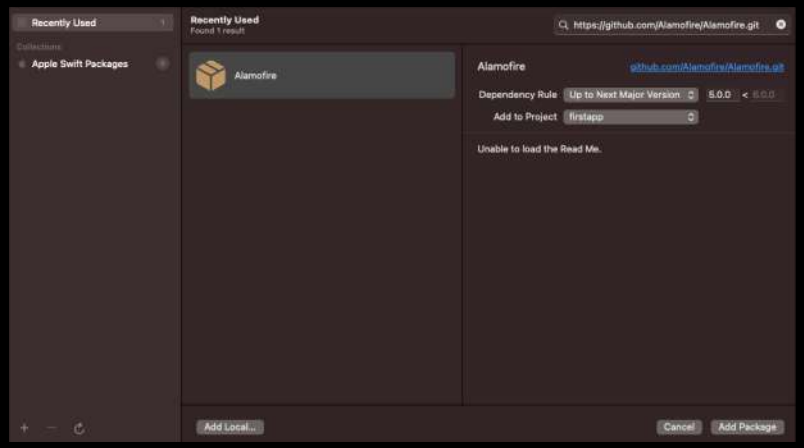


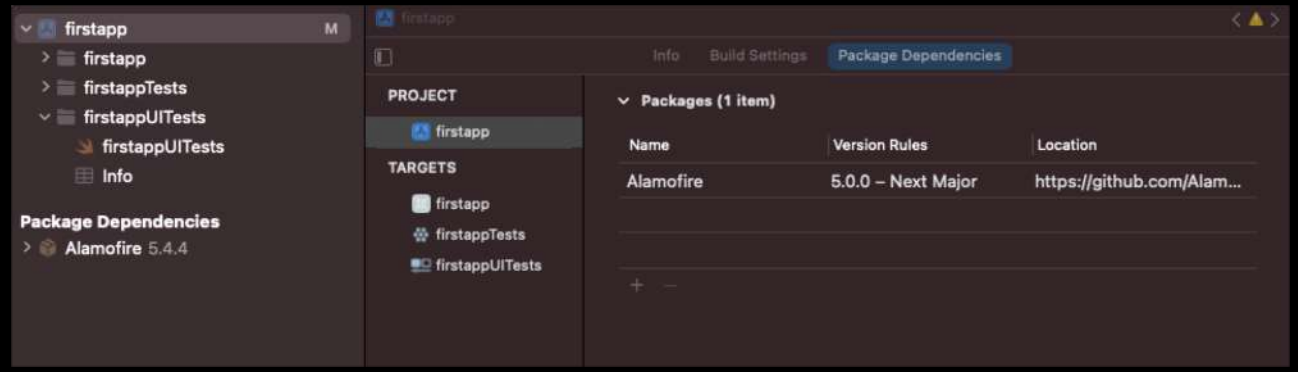


Networking - Alamofire

- → Alamofire => framework that handle http requests
- → Add Alamofire with Swift Package Manager







Networking - Alamofire

iOS Development

- Networking - Alamofire

Request Github API and display a user repositories statistics

- → Create a Codable Structure
- → Add a new button on the first screen to trigger the API call using the input name as a Github user
- → Import, instantiate Alamofire and use its methods to make API Calls

- → Get a working copy from https://github.com/dovene/first-app-ios/tree/feature/add-networking

```swift
import Foundation

struct GithubRepository: Codable {
    let name: String
    let url: String
}
```

```swift
@objc
func displayGithubRepositories() {
    guard let username = usernameTextField.text else {
        return
    }
    let request = AF.request("https://api.github.com/users/\(username)/repos")
    request.responseDecodable(of: [GithubRepository].self) {response in
        switch response.result {
        case .success:
            if let repositories = response.value {
                self.displayAlert(title: "Vous avez \(repositories.count)
                    repositories publics", message: "\(repositories[0].name) est
                    l'un de vos repositories")
            }
        case .failure:
            self.displayAlert(title: "Error", message: "\(String(describing:
                response.error))")
        }
    }
}

func displayAlert(title: String, message: String){
  let alertConroller = UIAlertController(title: title, message: message,
      preferredStyle: .alert)
    alertConroller.addAction(UIAlertAction(title: "Ok", style: .default ))
    present(alertConroller, animated: true, completion: nil)
}
```

iOS http Requests & CoreData

iOS Web API & Local Storage

TP - Google API Book

En s'inspirant de la maquette ci-dessous proposer une application permettant de rechercher un livre en saisissant l'auteur et le titre du livre. On souhaite également pouvoir enregistrer les livres favoris en local pour une consultation offline.
Voici l'url d'accès à l'API https://www.googleapis.com/books/v1/
Format de requête :
https://www.googleapis.com/books/v1/volumes?q=titreDuLivre+inauthor:auteurDuLivre

N.B. Vous pouvez utiliser AlamofireImage pour la récupération des images