

Mise en pratique du développement mobile iOS : Construction d'une application de A à Z

Réalisation d'une application météo (5 ème partie)


31 janvier 2022 - 17H30

Sommaire

01.Rappels

02.QCMS

03. Code

- 
- 01. Rappels
 - 02. QCMS
 - 03. Code

Rappels – live 1

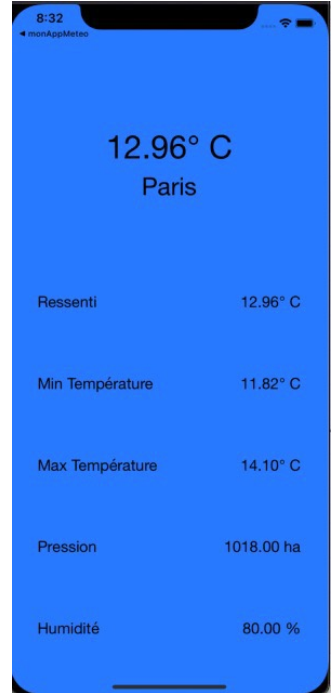
- Présentation de la chaine de travail pour développer une application mobile
- Présentation des choix des logiciels et frameworks
- Les prérequis : - réalisation d'une icone
- d'un launchscreen

Rappels – live 1

- Présentation de l'API : openweathermap
- Webservice pour récupérer des données météorologiques
- Gratuit pour une utilisation non-commercial
- Récupération du temps en fonction d'un ID de ville ou de sa localisation

Rappels – live 1

- Présentation de l'application météo :
 - un header (température – location)
 - un body : une liste minimaliste



Rappels – live 1

- Initialisation du projet dans Xcode
- Mise en place des éléments graphiques (Assets)
- Construction du header :
 - Mise en place des contraintes pour que les éléments graphiques soient correctement positionnés (autolayout)
 - Test de la mise en page

Rappels – live 2

- Connection des éléments graphiques avec le code :
 - liaison IBOutlets
- Réalisation de requête avec openWeatherMap
- Présentation de l'outil PostMan

Rappels – live 2

- Présentation du format JSON
- Système de clé/valeur
- Stockage :
 - chaînes de caractères
 - nombres
 - booleen
 - objets

```
{
  - coord: {
    lon: 2.3486,
    lat: 48.8534
  },
  - weather: [
    - {
      id: 800,
      main: "Clear",
      description: "clear sky",
      icon: "01d"
    }
  ],
  base: "stations",
  - main: {
    temp: 280.6,
    feels_like: 280.09,
    temp_min: 277.75,
    temp_max: 282.25,
    pressure: 1024,
    humidity: 73
  },
  visibility: 10000,
  - wind: {
    speed: 1.34,
    deg: 270,
    gust: 2.24
  },
  - clouds: {
    all: 0
  },
  dt: 1642259497,
  - sys: {
    type: 2,
    id: 2041230,
    country: "FR",
    sunrise: 1642232326,
    sunset: 1642263643
  },
  timezone: 3600,
  id: 2968815,
  name: "Paris",
  cod: 200
}
```

Rappels – live 2

- Réalisation de requête avec l'outil PostMan :
 - formulation de la requête à travers un url : endPoint
 - passage de paramètres « query »
 - différentes façons de récupérer des données : id ou lat et lon (pour longitude)

Rappels – live 2

- Présentation du protocol Decodable :
 - swift 3
 - décodage du json pour construire des objets
- Récupération des données depuis une api pour les ranger dans une structure de donnés
- Chargement et sauvegarde de données dans un fichier

Rappels – live 2

- Décodage d'un objet simple

```
let Personne1Json = """
{
  "nom": "Jeanne",
  "age": 35,
  "genre": "femme",
  "signe": "Lion",
  "travail": yes
}
"""
```

```
struct Personne : Decodable {
  let nom: String,
  let age: Int,
  let genre: String,
  let signe: String,
  let travail: Bool,
  let partenaire: String?
}
```

Rappels – live 2

- Déclaration et utilisation d'un objet JSON decodeur

```
let decoder = JSONDeconder()  
let person1JjsonData = person1JSON.data(using:.utf8)  
let person1 = try! decoder.decode(Person.self, from:person1JjsonData)  
print(person1)
```

Rappels – live 2

- Décodage d'un tableau d'objets

```
let personnesJSON = """
[
  {
    "name": "Pierre",
    "age": 25,
    "genre": "homme",
    "signe": "Taureau",
    "travail": true,
    "partenaire": "Emilie"
  },
  {
    "name": "Mary",
    "age": 45,
    "gender": "female",
    "sign": "Taurus",
    "partner": "James"
  },
]
"""
```

```
let decoder = JSONDecoder()
// conversion en jsonData
let personsJsonData = personnesJSON.data(using: .utf8)!
// on va placer les personn dans un tableau de personn
let personsArray = try! decoder.decode([Personne].self, from: personsJsonData)

for person in personsArray{
    print("\(person.name) 's partner \(person.partner ?? "none")")
}
```

Syntaxe : [Personne].self pour décoder un tableau d'objet

Rappels – live 2

- Décodage d'un JSON plus complexe

```
let familyJSON = """
{
  "nomdeFamille": "Dupond",
  "membres": [
    {
      "nom": "sophie",
      "age": 45,
      "genre": "femme",
      "signe": "Poisson",
      "travail": true,
      "partenaire": "Jacques"
    },
    {
      "nom": "Marie",
      "age": 45,
      "genre": "femme",
      "signe": "lion",
      "partenaire": "Franck",
      "travail": false,
    }
  ]
}
"""
```

```
struct famille: Decodable {
    let nomdeFamille: String
    let membres: [Personne]
}

let decoder = JSONDecoder()
let personJsonData = familyJSON.data(using: .utf8)
//print(personJsonData)
let famille1 = try decoder.decode(famille.self, from: personJsonData!)
```

Rappels – live 3

- Définition d'une énum de EndPoint avec des « case » pour construire une url en fonction des paramètres
- Construction de l'url en fonction du case avec des arguments
- Construction dynamique

Rappels – live 3

```
enum EndPoint{
    case cityId(path:String = "/data/2.5/weather", id:Int)

    var url:URL? {
        var components = URLComponents()
        components.scheme = "https"
        components.host = baseUrl
        components.path = path
        components.queryItems = queryParameters
        return components.url
    }

    private var path: String {
        switch self {
            case .cityId(let path, _):
                return path
        }
    }

    private var queryParameters : [URLQueryItem] {
        var queryParameters = [URLQueryItem]()
        switch self {
            case .cityId(_, let id):
                queryParameters.append(URLQueryItem(name:"id", value: String(id)))
        }
        // on ajoute le parametre de l'api Key
        queryParameters.append(URLQueryItem(name: "appid", value: apiKey))
        return queryParameters
    }
}
```

Rappels – live 3

- Réalisation d'une requête réseau sur l'api openweathermap
- Requête asynchrone
- Récupération des données du temps courant à partir d'un id
- Décodage du json à la réception des données
- Affichage des résultats

Rappels – live 4

- Création d'un HomeViewModel
- Récupération des données à partir du HomeViewModel
- Mise à jour des données quand on a reçu et décoder les données
- Création de variables calculées pour afficher les informations

Rappels – live 4

```
class HomeViewModel {  
  
    var weather:Weather?  
  
    func fetchWeather(for cityId: Int = ConfigManager.shared.cityID, _ completion: @escaping (() -> Void)){  
        NetWorkController.fetchWeather(for: cityId) { weather in  
            self.weather = weather;  
            //// NE PAS OUBLIER LE COMPLETION  
            completion()  
        }  
    }  
  
    var temperatureString:String {  
        return String(weather?.main.temp ?? 0)  
    }  
  
    var nameString: String {  
        return String(weather?.name ?? "")  
    }  
}
```

Rappels – live 4

```
override func viewDidLoad(_ animated: Bool) {
    viewModel.fetchWeather { [weak self] in
        print("mise à jour UI")
        DispatchQueue.main.async {
            self?.setupUI()
        }
    }
}

func setupUI() {
    temperatureLB.text = viewModel.temperatureString
    localiteLB.text = viewModel.nameString
}

override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(true)
    print("view will appear")
    print("la vue va apparaitre")
}
```

02. QCMS

QCMS

1. Dans quel format de fichier récupère-t-on les données ?
 - a. xml
 - b. json
 - c. csv
2. Comment peut-on qualifier les requêtes auprès du serveur ?
 - a. instantané
 - b. synchrone
 - c. asynchrone
3. Quel verbe du protocole 'http' permet de récupérer des données du server?
 - a. post
 - b. put
 - c. get

QCMS

4. Quel objet permet d'effectuer des requêtes réseaux ?
 - a. NSURL
 - b. URLSession
 - c. URL

5. Quel signe dans une url permet de chaîner les paramètres comme lat et lon par ex ?
 - a. !
 - b. ?
 - c. &

6. Quel protocole permet de décoder des json à la volée ?
 - a. JSONDecoder
 - b. Decodable
 - c. UIPickerViewDelegate

QCMS

9. Quelle « class » permet de définir de manière élégante les paramètres d'une requête http ?
- a. QueryItems
 - b. URLComponents
 - c. URLQueryItems
8. Quel objet permet de décoder les données une fois reçu ?
- a. URLSession
 - b. JsonDecodable
 - c. DataTask
9. Comment s'appelle les variables qui permet de mettre en forme les données avant de les afficher ?
- a. les variables observatrices
 - b. les variables calculées
 - c. les variables « tout court »

QCMS

10. Déclarer un tableau « MonTableau » de String de manière explicite, initialisé avec quelques valeurs, il pourra en accepter de nouvelles ?

- a. `let monTableau : String = [« pomme », « fraise »]`
- b. `var monTableau: [String] = [« pomme », « fraise »]`
- c. `Var monTableau = <String>[]`

11. Insérer la valeur « Cerise » au tableau ?

- a. `monTableau.append(« cerise »)`
- b. `monTableau.insert(« cerise »)`
- c. `monTableau.add(« cerise »)`

12. Quel méthode permet de retourner le nombre d'éléments de monTableau ?

- a. `size`
- b. `count`
- c. `length`

QCMS

13. Ou fait la récupération des données ?
- a. Dans la view
 - b. Dans le model
 - c. Dans le controleur
14. Quel mot clé permet de retourner les données retourner?
- a. complete
 - b. return
 - c. resume
15. Quel patron de conception permet
- a. JSDecoder
 - b. URLSession
 - c. NetWorkContrôler

QCMS

16. Quel mot clé fait référence à l'objet lui-même ?

- a. this
- b. self
- c. me

17. Quel mot clé est indispensable pour que notre requête réseau aboutisse ?

- a. start
- b. resume
- c. progress

18. Quel mot clé permet de faire référence à la class parent ?

- a. self
- b. super
- c. this

QCMS

19. Que faut-il faire avant de mettre à jour l'interface utilisateur ?
- a. refreshData
 - b. reloadData
 - c. DispatchQueue.main.async
20. De quel type doit être une variable pour qu'elle soit partagée à l'intérieur de la class ?
- a. static
 - b. global
 - c. late
21. Quel mot-clé permet de construire un objet à partir d'une class ?
- a. extension
 - b. init
 - c. construct

Exercice pratique

Soit 3 tableaux de 4 notes d'élèves (un tableau pour un élève)

Soit 1 tableau de coefficients à appliquer au 4 notes ?

1. Déclarer un tableau pour chaque étudiant.
2. Déclarer un tableau pour chaque matière.
3. Faire une fonction qui renvoie le nombre d'étudiants qui ont plus de 4 ?
4. Créer une structure Promo qui contiendra autant d'élèves que l'on veut ?



Code