

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

Réalisation d'une application météo (6 ème partie)


7 février 2022 - 17H30

# Sommaire

01.Rappels

02.QCMS

03. Code

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

## Rappels – live 1

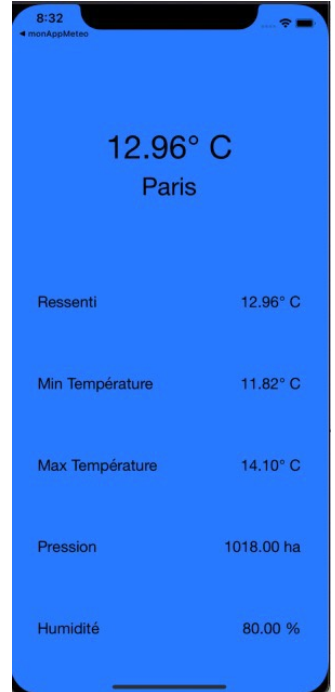
- Présentation de la chaine de travail pour développer une application mobile
- Présentation des choix des logiciels et frameworks
- Les prérequis : - réalisation d'une icone  
- d'un launchscreen

## 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 protocole Decodable :
  - swift 3
  - décodage du json pour construire des objets
- Récupération des données depuis une api pour les ranger dans une structure de données
- 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 personnn dans un tableau de personnn
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 apparaître")
}
```

## Rappels – live 5

- Explication de [weak self] à l'intérieur d'une closure
- Cycle retain => fuite de mémoire => crash
- Sans weak, le self capture une instance de type « strong »
- Si on a un nil, cela va être propager
- Possibilité d'utilisé aussi [unowned self]

## Rappels – live 5

- Cas pratique de mise en place d'un weak

```
class Personn {  
    let name:String  
    init(name:String){  
        self.name = name;  
    }  
    deinit {  
        print("\(name) is being deinitialisezed")  
    }  
}
```

```
var bob:Personn? = Personn(name: "Bob")
```

```
class Appartement {  
    let number:Int  
    // un résident  
    weak var resident: Personn?  
    init(number:Int){  
        self.number = number  
    }  
}
```

# Rappels – live 5

```
protocol WebService_ {
    func fetchData(_ completionHandler: @escaping (Float) -> Void)
}

/* Mise en place de weak*/
class viewController2 : UIViewController {

    var service:WebService_?
    let temperatureLB = UILabel()
    func toString(_ rawData: Float) -> String {
        return "\(rawData)"
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        service?.fetchData({ [weak self] res in
            self?.temperatureLB.text = self?.toString(res) ?? ""
        })
    }
}
```



## Rappels – live 5

- Révision d'algorithmique : création d'une fonction qui prend en paramètres une structure Promo
- Cette structure contient un tableau d'étudiant
- Chaque étudiant contient un tableau de notes
- La fonction renvoie le nombre d'étudiant dont la moyenne est supérieur à 16 après avoir appliqué un tableau de coefficient aux notes de chaque étudiant
- Utilisation de la structure de contrôle « For » pour parcourir les étudiants
- Utilisation de la structure de contrôle « while » pour parcourir les notes
- Utilisation de la structure de contrôle « if » pour incrémenter un cpt si la moyenne est supérieure à 16

# Rappels – live 5

```
struct Etudiant_ {  
    var notes:[Float]  
}
```

```
struct Promo2 {  
    var etudiants:[Etudiant_] = [Etudiant_(notes: [18,19,2]),Etudiant_(notes:  
        [15,13,18]),Etudiant_(notes: [12,14,17])]  
}
```

## Rappels – live 5

```
func noteEtudiant2(promo: Promo2) -> Int {  
    var cpt = 0;  
    var moyenne:Float = 0  
    var cumul:Float = 0  
    for etudiant in promo.etudiants {  
        var i=0  
        while i < etudiant.notes.count {  
            cumul = cumul + etudiant.notes[i] * coeff[i]  
            i = i + 1  
        }  
        moyenne = cumul / (coeff[0] + coeff[1] + coeff[2])  
        print(moyenne)  
        if moyenne > 16 {  
            cpt = cpt + 1  
        }  
        cumul = 0  
    }  
    return cpt;  
}
```

## 02. QCMS

## QCMS

1. Quelle structure de contrôle permet d'accéder à chaque élément par itération ?
  - a. while
  - b. repeat while
  - c. for each
  
2. Déclarer une variable « maMoyenne », de type double, initialisé à 12.0?
  - a. var maMoyenne = 12.0
  - b. let maMoyenne: double = 12.0
  - c. var maMoyenne : double = 12.0
  
3. Déclarer un tableau de « mesNotes » de manière explicite, avec possibilité d'ajouter des éléments ?
  - a. var mesNotes = [12.5, 13, 17, 18.0]
  - b. let mesNotes = [12.5, 13, 17, 18.0]
  - c. var mesNotes: double = [12.5, 13, 17, 18.0]

## QCMS

4. On parcourt ce tableau et on souhaite appliquer un coeff contenu dans un tableau de coeffArray, quelle structure de contrôle va - ton utiliser ? (2 réponses)
- a. for each
  - b. repeat while
  - c. while
5. Quel mot clé définit une liaison faible?
- a. weak
  - b. self
  - c. String
6. Que peut provoquer une fuite de mémoire ?
- a. rien
  - b. un crash
  - c. un warning

## QCMS

7. Quel signe permet de débiller un optionnel ?
- a. ?
  - b. !
  - c. .
8. Dans quel format on convertit le json avant de le parser ?
- a. utf8
  - b. ansi
  - c. Latin-1
9. Quel composant UIKit sert d'afficher du texte ?
- a. Text
  - b. TextField
  - c. Label

## QCMS

10. Quelle « class » permet de créer une instance pour effectuer une requête réseau ?
- a. JSONDecoder
  - b. URLSession
  - c. NetworkController
11. Comment s'appelle la connexion d'un bouton avec une action ?
- a. IBaction
  - b. Iboutlet
  - c. Segue
12. Quel mot clé il ne faut pas oublier pour qu'une requête aboutisse ?
- a. start
  - b. resule
  - c. start



## QCMS

13. Quel mot clé permet de renvoyer les données reçues ?
- a. func
  - b. completion
  - c. escaping
14. Quelle propriété permet de changer la couleur d'un sfSymbol?
- a. color
  - b. tintColor
  - c. backgroundColor:
15. Pour centraliser nos « endPoint » ou point de connexion on utilise ?
- a. Struct
  - b. Enum
  - c. Class

## QCMS

16. Quel objet permet d'envoyer des données d'un controller à un autre?
- a. segue
  - b. tag
  - c. sender
17. Quel signe permet de chaîner les paramètres d'une requête http(s) ?
- a. ?
  - b. :
  - c. &
18. Quel patron de conception permet de rendre accessible les données dans l'application?
- a. Delegate
  - b. Observer
  - c. Singleton

## QCMS

- 19 . Quelle class permet définir de manière propre les paramètres d'une requête http ?
- a. URLComponents
  - b. URLQueryItems
  - c. QueryItems
20. Quel mot clé est nécessaire pour passer une fonction en completion handler?
- a. completion
  - b. escaping
  - c. return
21. Quel class gère les écrans de base dans l'application ?
- a. TableViewController
  - b. PickerViewController
  - c. ViewController



Code