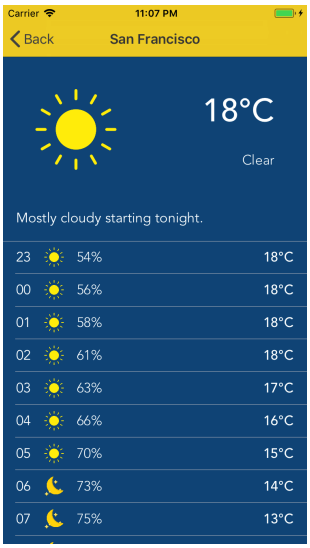
Weather App: Quel temps fait-il à San Francisco ?

Afin de terminer en beauté ce module iOS, vous allez devoir réaliser une application météo basique mais parfaitement utilisable.

UIViewController Map



UIViewController Detail 1/3

UIViewController Detail 2/3

UIViewController Detail 3/3

Pour d'aboutir à cela, une série d'étapes vous attend. L'ordre n'est pas anodin, il est fait pour vous aider. Il vous est donc conseillé de le respecter (mais vous n'êtes pas obligé et pouvez vous tenir à votre bon jugement)

Faites craquer vos doigts, échauffez vous les poignets, c'est parti !

Préparation:

Voici l'API que vous allez utiliser pour votre projet

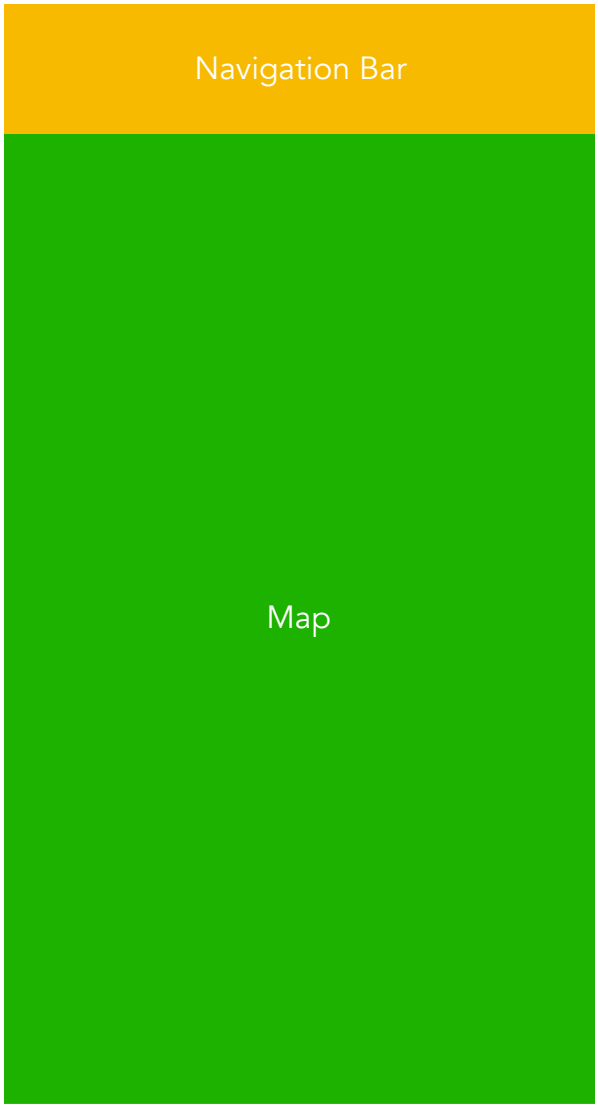
https://darksky.net/dev/docs

Vous devez vous inscrire et obtenir une secretKey. Celle-ci devra être utilisé à chaque appel de l'API. La documentation de l'API vous montre comment l'utiliser.

Vous devez aussi créer un dépôt Github au nom de votre projet.

La date limite du rendu est le Vendredi 24 Mai, 23h42

Étape 1: La Carte



Créez votre premier **UIViewController** avec une **MKMapView** et placez les points fournis dans le fichier *CitiesData.swift* sur cette carte.

Chaque annotation doit avoir un titre.

Étape 2: Sélection et navigation

Faites en sorte de pouvoir sélectionner un point et mettez en place la navigation vers votre second **UIViewController** qui affichera les détails.

La documentation de **MKMapViewDelegate** est la pour vous aider

Passez au **UIViewController** Detail les coordonnées et le titre de l'annotation (ou le model City associé)

Étape 3: Appel à l'API

Mettez en place votre appel à l'API en envoyant les bons paramètres depuis votre **UIViewController** *Detail*. Vous allez pour cela devoir étudier les documentation de celle-ci afin d'envoyer les bons paramètres.
Vous devez aussi faire en sorte que les informations renvoyés soit formatées en unité du système international (SI) et non pas impérial (tout ce qui est degrés Fahrenheit, miles, etc…)

Logguez bien votre retour de l'api, que celui-ci soit un success ou un error, afin d'avoir une bonne visibilité sur votre travail.

Étape 4: Création des models et parsing JSON

Comme vous avez pu le constater, l'API est relativement complète et renvoie beaucoup d'informations. Nous n'allons toutefois pas tout utiliser. Voici ce dont vous aurez besoin:

- L'icon du temps
- La température
- Le résumé (summary) actuel
- La force du vent
- La pression
- L'humidité
- l'indice UV
- Le texte du hourly forecast
- Le texte du daily forecast
- Un tableau d'éléments contenant les informations heures par heures (hourly) suivantes:
  - Le temps UNIX (nombre de secondes écoulées depuis le 1 janvier 1970)
  - L'icon du temps
  - L'humidité
  - La température
- Un tableau d'éléments contenant les informations jour par jour suivantes (daily) suivantes:
  - Le temps UNIX
  - L'icon du temps
  - La température minimum
  - La température maximum

Vous devez créer les models adéquats afin de stocker ces informations de façon logique et ordonnée (pensez que ces models seront utilisés par la **UITableView** dans votre **UIViewController** Detail)

En cas de succès du retour de l'API, vous décoderez la data reçue grâce au parsing natif de Swift comme vu en cours

Étape 5: Construction de la UI de la vue détail

Vous utiliserez une **UITableView** avec plusieurs cellules differentes:
- Une cellule *Header*
- Une cellule pour afficher le texte du hourly forecast et du daily forecast (voir documentation API)
- Une cellule pour afficher les informations du hourly forecast
- Une cellule pour afficher les informations du daily forecast
- Une cellule pour afficher des extras informations (pressure, wind speed, etc…)

Aussi, les cellules ne doivent pas être sélectionnables.

Tips: Tout comme le schéma, mettez une couleur de background différente pour chaque type de cellule afin de vous y retrouver pendant la construction de votre UI

Étape 6: Liage des données à la UI

Maintenant que vos models sont crées et qu'ils se remplissent au moment du retour success de l'API, vous allez pouvoir afficher leur contenu sur la UI.

Navigation Bar title: Le nom de la ville

Header: Affiche les informations actuelles suivantes:
- Icon
- Température (En degrée celsius, avec le symbole de l'unité, arrondie)
- Le texte du current forecast

Hourly Forecast: Affiche le texte du hourly forecast

Hourly Forecast infos: Chaque cellule affiche les informations suivantes:
- Le chiffre de l'heure (Formaté avec deux chiffres, même pour les heures en dessous de 10… Google est votre ami)
- L'icon
- L'humidité (Si et seulement si celle-ci est supérieure à 0)
- La température (En degrée celsius, avec le symbole de l'unité, arrondie)

Daily Forecast Affiche le texte du daily forecast

Daily Forecast infos: Chaque cellule affiche les informations suivantes:
- Le jour de la semaine (En lettre et en anglais)
- L'icon
- La température maximum (En degrée celsius, avec le symbole de l'unité, arrondie)
- La température minimum (En degrée celsius, avec le symbole de l'unité, arrondie)

Extra Informations: 2 cellules, 4 informations à afficher:
Sur la première:
- L'humidité (Formatée en pourcentage, avec le symbole %)
- La vitesse du vent (En kilomètre par heure, avec le symbole de l'unité)

Sur la seconde:
- La pression (En hectopascal, avec le symbole de l'unité)
- L'indice UV

Étape 7: Ajout d'un mode liste sur le premier ViewController

Afin d'enrichir votre application, vous allez rajouter une **UITableView** sur votre **UIViewController** *Map*. Celle ci devra afficher la liste des villes que vous avez précédemment placez sur la carte.

Au clique d'une des cellules, vous devrez naviguer vers le **UIViewController** *Detail*

Toutefois, nous ne voulons pas nous débarrasser de la carte. Les deux doivent cohabiter. Vous devez donc créer un bouton dans la **Navigation Bar** et une *@IBAction* reliée à celui-ci qui fera un "switch" entre la **UITableView** et la **MKMapView**. Chaque clique doit intervertir les deux. Il y a plusieurs façons de faire cela, vous être libre d'utiliser celle de votre choix.

Vous avez à votre disposition une icône "hamburger" mais vous êtes libre d'en ajouter une autre (ou même d'en mettre une différente selon le mode d'affiche sur lequel vous vous trouvez. Cela sera bien sûr valoriser si vous le faites)

Étape 8: Un bon coup de pinceau

Affiner le design de votre UI avec des polices et des couleurs de votre choix. Tout est de votre choix mais la condition est que votre application doit être agréable à utiliser.

Vérifier vos espacements et homogénéiser les si besoins (Essayer de respecter au maximum des espacements d'une valeur d'un multiple de 8 ou d'un multiple de 4)

Si besoin, intégrer des icons sur les boutons (Celui du switch ou le back du **UIViewController** *Detail*)

Étape 9: Refactoring

Repassez en review votre code et essayer de refactorer les bouts de code qu'il vous semble nécessaire (un conseil: Si votre projet est fonctionnel avant cet étape, créez une autre branche sur Github qui part de votre branche projet et faites votre refactoring sur cette nouvelle branche. Cela vous enlevera le stress de tout casser et vous n'aurez qu'à faire un merge sur votre branche projet quand vous aurez fini le refactoring)

Étape 10: Like a boss

Pieds sur la table, lunettes de soleil sur le museau, mojito à la main, vous avez bien bossé. Profitez, vous avez raison

Idées de bonus:

- Loader sur le UIViewController Detail pendant le chargement des données + Affichage d'un message d'erreur dans la UI
- Couleur du fond différent selon le temps qu'il fait actuellement