TP 5
MapKit

À la fin de ce TP :

- Faire une archive contenant les projets Xcode des exercices
- Envoyer l'archive à ahumiliere@captaintrain.com avec l'objet : `[DANT] TP 5 – Prénom Nom`
- Si le TP est fait à plusieurs, préciser les noms et adresses mail de chacun

Le TP se base sur un nouveau projet Xcode à créer, de type « Single View Application ».

Exercice 1 : UIApplication

- Dans Interface Builder, ajouter un bouton « Ouvrir l'application Plans » à `ViewController`.
- Ajouter les contraintes pour positionner le bouton en haut de l'écran, centré horizontalement.
- Ajouter une action au touch du bouton. On initialise ici un objet de type `NSURL` à partir d'un `String`.

```
@IBAction func openMapsAppWithURL(sender: UIButton) {
    if let url =
        NSURL(string: "http://maps.apple.com/?q=Yosemite") {
        let app = UIApplication.sharedApplication()
        app.openURL(url)
    }
}
```

- Le paramètre `q=` représente la requête faite à Apple Maps. Il est parsé par l'application pour rechercher un emplacement sur la carte. Se documenter sur la classe `UIApplication`.
- Lancer l'application et observer le résultat.

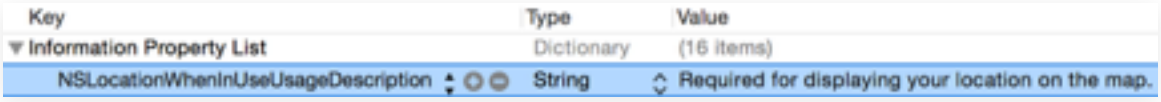
Exercice 2 : MapKit

- Explorer la documentation du framework `MapKit`. Il permet d'intégrer simplement des cartes dans une application iOS.
- Retirer le bouton de l'interface. Ajouter une MapView et la positionner sur tout l'écran avec AutoLayout. La vue devra s'adapter et occuper tout l'écran quelle que soit l'orientation.
- Lancer l'application. Observer le crash. L'application ne peut pas instancier la classe `MKMapView` car le framework `MapKit` n'est pas *linké*.
- Dans la barre latérale, sélectionner le projet et ajouter `MapKit.framework` à la liste des *Linked Frameworks and Libraries*.
- Lancer l'application.

Exercice 3 : CoreLocation

L'objectif est ici d'afficher la position de l'appareil sur la carte.

- Dans Interface Builder (IB), sélectionner la Map View et, avec l'Attributes Inspector, changer le type de carte à hybride, s'assurer que *Shows User Location* est coché.
- Lancer l'application, un message apparait dans la console.
  Les applications iOS doivent demander l'autorisation de l'utilisateur avant de pouvoir utiliser les informations de localisation.
- Se documenter sur le framework `CoreLocation`. Importer le framework dans `AppDelegate`.
- Toujours dans `AppDelegate`, ajouter une propriété avec une valeur par défaut à `CLLocationManager()` et explorer la classe `CLLocationManager` dans la documentation.
- Modifier l'implémentation de `application:didFinishLaunchingWithOptions:` pour ajouter un appel à la méthode `requestWhenInUseAuthorization()` sur l'instance de `CLLocationManager` ajoutée plus haut en propriété.
- Dans Info.plist, ajouter la clé `NSLocationWhenInUseUsageDescription` et indiquer comme valeur un texte expliquant pourquoi vous comptez utiliser la localisation dans votre application.



- Lancer l'application. Le simulateur choisit automatiquement Cupertino comme localisation par défaut. Il est possible de lui spécifier une autre position : *Debug* > *Location* > *Custom Location…*

Exercice 4

Nous allons maintenant faire en sorte que la carte affiche automatiquement la zone autour de la position courante.

- Explorer la documentation de `MKMapView`, et plus particulièrement la méthode `setUserTrackingMode:animated:` et l'enum `MKUserTrackingMode`.
- Importer le framework `MapKit` dans `ViewController`.
- Avec IB, ajouter une référence (`IBOutlet`) sur la map view dans `ViewController`.
- Dans la méthode `viewDidLoad`, appeler `setUserTrackingMode` en choisissant des paramètres adaptés en fonction de ce que vous aurez lu dans la documentation
- Lancer l'application et admirer le résultat.
- L'utilisateur pourrait vouloir un niveau de zoom différent de celui choisi par défaut. Etudier la documentation de `MKMapViewDelegate`, et particulièrement la méthode `mapView:didUpdateUserLocation:`.
- Dans la vue Document Outline (⧈), declarer `ViewController` come Delegate de la map view.
- Indicter que `ViewController` implement bien le protocole `MKMapViewDelegate`.
- Ajouter une implémentation `mapView:didUpdateUserLocation:` dans `ViewController`.

```
func mapView(mapView: MKMapView, didUpdateUserLocation userLocation:
MKUserLocation) {
    let center = CLLocationCoordinate2D(latitude:
       userLocation.coordinate.latitude,
       longitude: userLocation.coordinate.longitude)
    let width = 1000.0 // meters
    let height = 1000.0
    let region = MKCoordinateRegionMakeWithDistance(center, width,
       height)
    mapView.setRegion(region, animated: true)
}
```

- MapKit appelle automatiquement la méthode `mapView:didUpdateUserLocation:` de son `delegate` quand la position de l'utilisateur est mise à jour par l'appareil.
- Se documenter sur les fonctions `CLLocationCoordinate2D`, `MKCoordinateRegion` et `MKCoordinateRegionMakeWithDistance`.
- Lancer l'application.

Exercice 5

- Dans IB, ajouter une `UIToolbar` en base de l'écran de l'application.
- Ajouter les contraintes autolayout adaptées.
- Cliquer sur bouton de l'item par défaut et utiliser Attributes Inspector pour changer l'Identifier à Add.
- Lancer l'application.
- Créer un `IBOutlet dropPin:` pour le bouton.
- MapKit utilise le concept de `MKAnnotation` pour représenter les marqueurs sur la carte. Se documenter sur le protocole `MKAnnotation`. Il n'y a pas de class `MKAnnotation`, seulement un protocole. Nous allons devoir créer une classe implémentant le protocole `MKAnnotation`.
- Ajouter une nouvelle classe `Pin` au projet.
- Importer MapKit et indiquer que la classe implémente le protocole `NSObject`.
- Le protocole MKAnnotation requiert une propriété coordinate de type `CLLocationCoordinate2D`. Implémenter cette propriété et créer un initialiseur prenant en paramètre un `CLLocationCoordinate2D` pour initialiser coordinate.
- Explain how `CLLocationCoordinate2D` is a struct, and not an Swift object type.
- Dans la méthode `dropPin:`, créer un objet `Pin` à partir de la position au centre de la carte et ajouter l'annotation à la map view.
- Lancer l'application et tester.

Exercice 6 : Bonus

- En vous documentant par vous même, utiliser une image à la place de l'épingle standard. Ajouter une infobulle quand l'utilisateur clique sur l'épingle.