# PuntoGPSQR

Aplicación iOS creada por Daniel Barros López y Yoji Sargent Harada

## Requerimientos

Se facilitarán las coordenadas de una serie de puntos GPS (latitud/longitud) mediante códigos QR a la appGPSQR que iniciará automáticamente la navegación GPS hacia dicho punto, debe guardar el recorrido realizado y mostrarlo en un mapa al finalizar el mismo. En el anexo I tenéis ejemplos de los códigos QR que se deben detectar, en la defensa serán distintos.

## Tutoriales

En el desarrollo de esta aplicación aprendimos a manejar parte de la librería CoreLocation para la obtención de la localización del usuario (previa autorización) y MapKit, para el cálculo de rutas y gestión de elementos de la interfaz de un MKMapView (un mapa). La lectura de códigos QR la hacemos con un código de terceros.

## 1. Obtención de autorización para uso de localización

Lo primero que necesitamos es un objeto CLLocationManager, mediante el cual recibimos actualizaciones de la localización del usuario.

```
let locationManager = CLLocationManager()
```

Antes de poder recibir estas actualizaciones necesitamos pedir permiso al usuario.

```
if CLLocationManager.authorizationStatus() != .AuthorizedAlways {
      locationManager.requestAlwaysAuthorization()
}
```

Aquí pedimos una autorización de tipo .AuthorizedAlways, lo que significa que recibiremos actualizaciones de localización incluso cuando nuestra app está en corriendo en el background. Es conveniente hacer esto en la función viewDidLoad() de nuestro UIViewController principal. Hay que recordar asignar el delegate de nuestro locationManager al view controller

```
locationManager.delegate = self
```

e implementar la función locationManager(_:didChangeAuthorizationStatus:) para que se nos notifique cuando hay algún cambio en la autorización.

```
func locationManager(manager: CLLocationManager, didChangeAuthorizationStatus
status: CLAuthorizationStatus) {
       if status == .AuthorizedAlways {
            // Tenemos autorización!
      }
}
```

## 2. Obtención de la localización del dispositivo

Una vez hemos obtenido la autorización podemos empezar a recibir actualizaciones de localización mediante el location manager. Es importante especificar el distanceFilter, que determina la mínima distancia que debe recorrer el dispositivo antes de generar una nueva actualización de localización.

```
locationManager.distanceFilter = 5
locationManager.startUpdatingLocation()
```

El delegate del location manager (nuestro view controller principal) debe implementar la función locationManager(_:didUpdateLocations:) y ya empezará a recibir actualizaciones.

```
func locationManager(manager: CLLocationManager, didUpdateLocations
locations: [CLLocation]) {
      // locations contiene las localizaciones visitadas desde la última
actualización
}
```

Hay que recordar llamar al método stopUpdatingLocation() en algún momento para dejar de recibir actualizaciones.

```
locationManager.stopUpdatingLocation()
```

## 3. Mostrar un punto en el mapa

Una vez obtenemos una localización mediante nuestro lector de códigos QR debemos mostrarla en el map view. Para ello existe la función addAnnotation(_:) de MKMapView. Toma como parámetro un objeto que adopte el protocolo MKAnnotation, que simplemente especifica que debe tener una propiedad coordinate.
Debemos crear una clase que adopte el protocolo y que podamos inicializar a partir de un objeto localización.

```
class MapAnnotation: NSObject, MKAnnotation {
    var coordinate: CLLocationCoordinate2D

    init(location: CLLocation) {
        coordinate = location.coordinate
        super.init()
    }
}
```

Una vez hecho esto podemos crear nuestro objeto MapAnnotation y usarlo para llamar a la función addAnnotation(_:) de nuestro map view.

```
mapView.addAnnotation(MapAnnotation(location: location))
```

# 4. Mostrar una ruta en el mapa

Una ruta está representada por un objeto MKRoute de MapKit. Para mostrarla en el mapa usamos un método de MKMapView llamado addOverlay(_:level:). El primer parámetro que toma es un objeto MKOverlay. MKRoute tiene una propiedad llamada polyline que es de este tipo y representa una línea que pasa por los puntos de la ruta.

```
mapView.addOverlay(route.polyline, level: .AboveRoads)
```

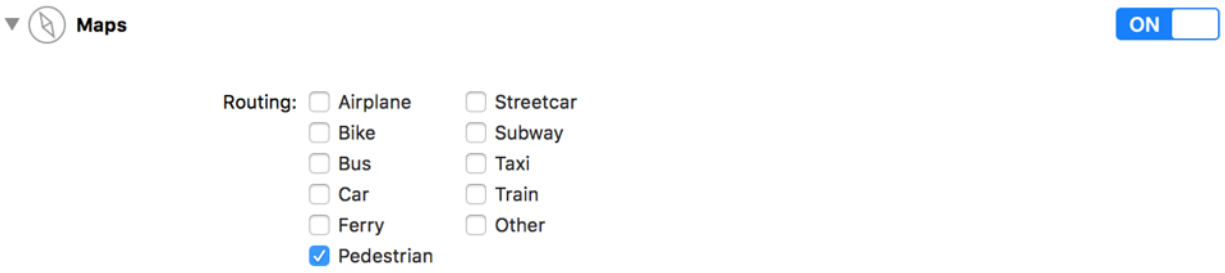
Obtener el objeto MKRoute es más complicado. Primero debemos crear un objeto MKDirectionsRequest en el que especificaremos detalles de la ruta tales como el punto de partida y de finalización.

```
let request = MKDirectionsRequest()
request.source = MKMapItem.mapItemForCurrentLocation()
request.destination = MKMapItem(placemark: MKPlacemark(coordinate:
                        location.coordinate, addressDictionary: nil))
request.requestsAlternateRoutes = false
request.transportType = .Walking
```

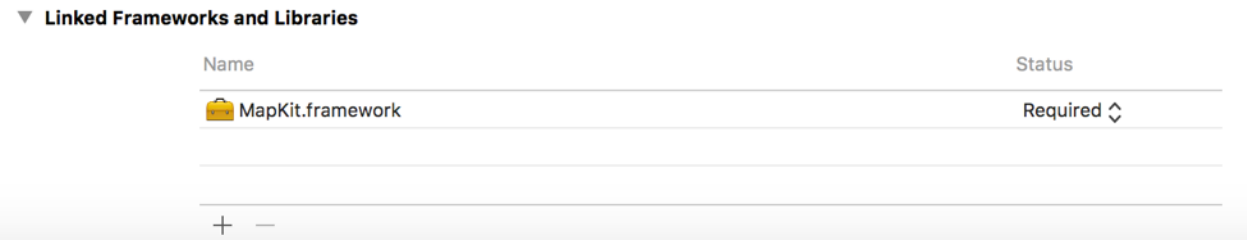
Después crearemos un objeto MKDirections a partir del request y llamaremos al método calculateDirectionsWithCompletionHandler(_:), que se ejecutará asíncronamente y al que pasamos un bloque de código que se encargará de manejar la ruta obtenida una vez hayan terminado los cálculos.

```
let directions = MKDirections(request: request)
directions.calculateDirectionsWithCompletionHandler {
      [weak self] response, error in
      if let route = response?.routes.first {
            self?.mapView.addOverlay(route.polyline, level: .AboveRoads)
      }
}
```

Para que lo anterior funcione debemos indicar en la configuración de nuestro proyecto que nuestra aplicación hace uso de rutas a pie. Para ello vamos a nuestro Target, vamos a la sección Capabilities y marcamos la casilla Pedestrian dentro de Maps.




Tampoco se nos debe olvidar incluir el framework MapKit a la lista de librerías enlazadas de nuestro proyecto. Para ellos vamos a nuestro Target, a la sección General y añadimos MapKit en la subsección Linked Frameworks and Libraries.




Como último debemos implementar el método mapView(_:rendererForOverlay) en el delegate de nuestro map view (en nuestro caso el view controller principal). Este método devuelve un objeto MKOverlayRenderer que especifica como se han de dibujar los overlays en el mapa.

```
func mapView(mapView: MKMapView, rendererForOverlay overlay: MKOverlay) ->
MKOverlayRenderer {
        let renderer = MKPolylineRenderer(overlay: overlay)
        let color = UIColor(red: 0.12, green: 0.56, blue: 1, alpha: 0.8)
        renderer.strokeColor = color
        renderer.lineWidth = 5
        return renderer
    }
```

## 5. Abrir la aplicación de Mapas del sistema con direcciones paso a paso a una localización

Primero creamos un objeto MKMapItem a partir de la localización de destino

```
let mapItem = MKMapItem(placemark: MKPlacemark(coordinate:
location.coordinate,
addressDictionary: nil))
```

Después usamos el método openInMapsWithLaunchOptions(_:), en el que incluimos un diccionario con la clave MKLaunchOptionsDirectionsModeKey y valor MKLaunchOptionsDirectionsModeWalking para indicar que queremos que se inicie el proceso de direcciones paso a paso al abrirse la aplicación de Mapas.

```
mapItem.openInMapsWithLaunchOptions([MKLaunchOptionsDirectionsModeKey:
MKLaunchOptionsDirectionsModeWalking])
```