iOS alapú szoftverfejlesztés - Labor 11

A laborsegédletet összeállította: Blázovics László, Kántor Tibor és Kelényi Imre

A labor témája:

* GPS Pozíció lekérdezés
  + CoreLocation
* Térkép megjelenítése
  + Map Kit
  + MKMapAnnotation

A laborhoz tartozó nagyobb kódrészletek a következő url-en érhetők el "copy-paste barát" formában:

https://gist.github.com/blazovics/aaa41932bf3b4d8e9bbc

# Messenger

Töltsük le az előző labor során kidolgozott némileg módosított projektet:

<https://drive.google.com/file/d/0B2dt4s5Q-kYsTjRXTmNYVmJqU0k/view?usp=sharing>

A főbb változások az előző verzióhoz képest, hogy a MessagesViewController alján megjelent egy toolBar rajta egy Map feiratú gombbal, illetve az üzenet létrehozó nézeten megjelent egy Label "Pending" felirattal. Ez a nézet fogja jelezni, hogy sikerült-e a feladó koordinátáját lekérdezni. Ezen kívül a ComposeMessageViewController-ben megjelent egy CLLocation property, ami a pozíciót fogja majd tárolni.

## Pozíció lekérdezése

Készítsünk el egy osztályt, mely segítségével le tudjuk kérdezni az aktuális pozíciónkat. Ehhez hozzuk létre a **LocationManager** nevű NSObject-ből származó osztályunkat.

Hogy az osztály értesítve legyen a pozícióval kapcsolatban, implementálni kell a CLLocationManagerDelegate protocolt!

Ahogy az a kódban is látható, az osztálynak importolnia kell a CoreLocation modult.

Hozzunk létre egy lastLocation nevű property-t, ami CLLocation típusú pozíciót fog eltárolni. (A legutolsó ismert pozíciót.) Továbbá hozzunk létre egy timeoutTimer nevű property-t mely egy NSTimer példányra fog hivatkozni.

Hogy értesíteni tudjuk a pozícióra várakozó objektumainkat, hozzunk létre egy **block** típusú változót, melynek a hívó fél ad "értéket" a startLocationManager meghívásakor.

Végül vegyünk fel egy property-t egy CLLocationManager példány tárolásához.

import CoreLocation

class LocationManager: NSObject, CLLocationManagerDelegate {

var lastLocation: CLLocation?

var timeoutTimer: NSTimer?

var locationUpdated: (()->Void)!

var locationManager: CLLocationManager!

}

Implementáljuk a szükséges függvényeket!

A pozíció lekérdezés indítását a következő függvény végzi. Ha nem történik érdemi esemény 15 másodperc után leáll a pozíció lekérdezése.

func startLocationManager(updated:(()->Void)?){

locationUpdated = updated!

locationManager = CLLocationManager()

locationManager.requestAlwaysAuthorization()

locationManager.delegate = self

locationManager.desiredAccuracy =

kCLLocationAccuracyNearestTenMeters

timeoutTimer = NSTimer.scheduledTimerWithTimeInterval(

15.0,

target: self,

selector:"timeout",

userInfo: nil,

repeats: false)

locationManager.startUpdatingLocation()

}

func timeout(){

stopLocationManager()

}

Ha időközben érkezik frissítés, akkor az alábbi függvény le fogja kezelni.

func locationManager(manager: CLLocationManager!,

didUpdateLocations locations: [CLLocation]) {

let newLocation = locations.last as! CLLocation

let locationAge = -newLocation.timestamp.timeIntervalSinceNow

if locationAge > 5.0{

return

}

if newLocation.horizontalAccuracy < 0{

return

}

if lastLocation == nil ||

lastLocation!.horizontalAccuracy > newLocation.horizontalAccuracy {

lastLocation = newLocation

if newLocation.horizontalAccuracy <= manager.desiredAccuracy{

stopLocationManager()

}

}

}

https://gist.github.com/blazovics/aaa41932bf3b4d8e9bbc#file-didupdatelocations-locationmanager-swift

Ezután adjuk írjuk meg a hibakezelő fügvényünket. Ha nem tudjuk meghatározni a pozíciónkat, akkor adjunk neki még lehetőséget, egyébként állítsuk meg a frissítést.

func locationManager(manager: CLLocationManager!,

didFailWithError error: NSError!) {

stopLocationManager()

print(error.description)

}

Majd implementáljuk azokat a metódust, mely ténylegesen leállítjA a frissítési folyamatot.

func stopLocationManager(){

if let timer = timeoutTimer{

timer.invalidate()

}

locationManager.stopUpdatingLocation()

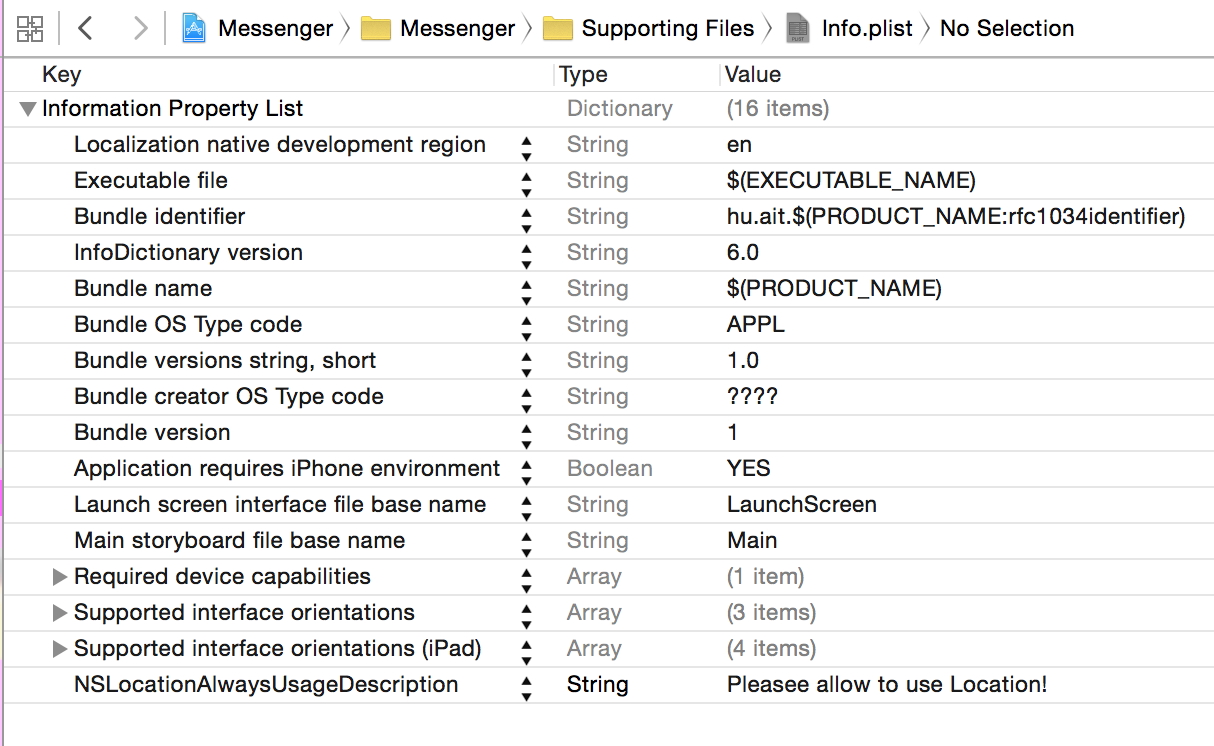
locationUpdated()

}

Végül adjuk hozzá az **info.plist** fájlban az alábbi kulcsot:

**NSLocationAlwaysUsageDescription**

Értéknek bármi megadható, de vigyázzunk, mert ezt fogja a felhasználó először elolvasni, amikor engedélyt kér az alkalmazás tőle.



## Pozíció csatolása az üzenethez

Térjünk rá a ComposeMessageViewController kiegészítésére.

Először vegyünk fel egy LocationManager típusú property-t az osztályban.

@IBOutlet weak var coordinatteLabel: UILabel!

var location: CLLocation?

var locationManager: LocationManager!

Ezután valósítsuk meg az aktuális pozíció tárolásáért felelős részeket.

Először inicializáljuk az LocationManager objektumunkat.

override func viewDidLoad() {

super.viewDidLoad()

locationManager = LocationManager()

}

Indítsuk el/állítsuk le a frissítést, amikor szükséges és implementáljuk a megfelelő block-ot a pozíció frissítéshez.

override func viewWillAppear(animated: Bool) {

locationManager.startLocationManager({ ()->Void in

if let location: CLLocation = self.locationManager.lastLocation{

self.location = location

self.coordinatteLabel.text = "\(location.coordinate.latitude)" + "

" + "\(location.coordinate.longitude)"

}

})

self.coordinatteLabel.text = "Pending"

}

override func viewWillDisappear(animated: Bool) {

locationManager.stopLocationManager()

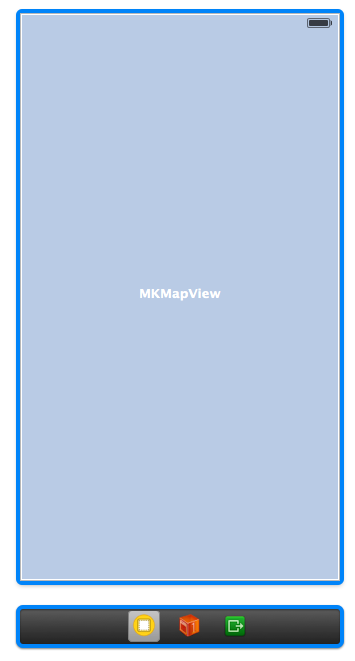
}

Teszteljük az alkalmazást!

## MapView megjelenítés

Hogy meg is tudjuk nézni az egyes helyhez kötött üzeneteket, hozzunk létre egy **MapViewController** nevű osztályt, ami a UIViewController-ből származik.

A storyboard-ban tegyünk a ViewController view-ja fölé egy teljes nézetet betöltő MKMapView-t, majd kössük be egy outlet-tel a ViewController-be mapView néven.

Ne felejtsük el az újonnan hozzáadott MapView-nak beállítani a tartalmazó viewController-t, mint delegate-et!



Miután ez megvan, térjünk vissza az osztály forrásához, importáljuk a MapKit modult! Kössük be a térképet a Storyboardból egy Outlettel, továbbá adjunk hozzá egy property-t, ami majd a megjelenítendő üzeneteinket fogja tartalmazni. Illetve jelezzük, hogy meg fogjuk valósítani az MKMapViewDelegate-et!

import MapKit

class MapViewController: UIViewController, MKMapViewDelegate {

@IBOutlet weak var mapView: MKMapView!

var messages = [AnyObject]()

A storyboardban hozzunk létre egy **MapSegue** nevű push segue-t, amivel a Map gombra kattintva betölthetjük a térképet. A navigation bar title-je legyen Map!

Ezután menjünk át a MEMessagesViewController-be készítsük fel a térkép betöltésére: egészítsük ki a prepareForSegue metódust!

override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {

if segue.identifier == "ComposeMessageSegue" {

let navVc = segue.destinationViewController as!

UINavigationController

let vc = navVc.viewControllers.first as?

ComposeMessageViewController

vc?.delegate = self

}

else if segue.identifier == "MapSegue" {

let vc = segue.destinationViewController as! MapViewController

vc.messages = messages

}

}

Teszteljük az alkalmazást!

Látható, hogy a térkép betöltődött, ugyanakkor nincs rajta semmi.

Ezt megoldandó, hozzunk létre egy új NSObjectből leszármazó osztályt, MessageAnnotation névvel, ami megvalósítja az **MKAnnotation** protocol-t:

import UIKit

import MapKit

class MessageAnnotation: NSObject, MKAnnotation{

var coordinate: CLLocationCoordinate2D

var title: String?

var subtitle: String?

init(coordinate: CLLocationCoordinate2D, title: String, subtitle:

String) {

self.coordinate = coordinate

self.title = title

self.subtitle = subtitle

}

}

Ha ez sikerült, akkor térjünk vissza a MapViewControllerbe és tegyük lehetővé, hogy a térképen megjelenjenek az üzenetek.

Először is adjuk hozzá az üzeneteket a térképhez annotation formában.

override func viewDidLoad() {

super.viewDidLoad()

for message in messages{

let toUser = message["to\_user"] as! String

let fromUser = message["from\_user"] as! String

let title = "\(toUser) \(fromUser)"

let coordinate = CLLocationCoordinate2DMake(

message["latitude"] as! Double,

message["longitude"] as! Double)

let annotation = MessageAnnotation(coordinate: coordinate,

title:title,

subtitle:message["topic"] as! String)

self.mapView.addAnnotation(annotation)

}

}

https://gist.github.com/blazovics/aaa41932bf3b4d8e9bbc#file-viewdidload-mapviewcontroller-swift

Futtassuk az alkalmazást!

Az Annotation-ök mellett adjunk hozzá egy törtvonalat (**MKPolyLine**) is a térképhez úgy, hogy minden üzenet legyen egymás után sorban összekötve. Ehhez először hozzunk létre egy koordinátákat tartalmazó tömböt, amivel létrehozzuk a törtvonalat, majd adjuk hozzá a törtvonalat a mapView-hoz, mint overlay.

override func viewDidLoad() {

super.viewDidLoad()

var coordinates = [CLLocationCoordinate2D]()

for message in messages{

let toUser = message["to\_user"] as! String

let fromUser = message["from\_user"] as! String

let title = "\(toUser) \(fromUser)"

let coordinate = CLLocationCoordinate2DMake(

message["latitude"] as! Double,

message["longitude"] as! Double)

coordinates.append(coordinate)

let annotation = MessageAnnotation(coordinate: coordinate,

title:title,

subtitle:message["topic"] as! String)

self.mapView.addAnnotation(annotation)

}

let polyline = MKPolyline(coordinates: &coordinates,

count: coordinates.count)

mapView.addOverlay(polyline)

}

Ezután implementáljuk a következő delegate metódust, hogy ki is legyen rajzolva a törtvonal.

func mapView(mapView: MKMapView, rendererForOverlay overlay: MKOverlay) -> MKOverlayRenderer {

if overlay is MKPolyline{

var line = MKPolylineRenderer(overlay: overlay)

line.strokeColor = UIColor.blueColor()

line.lineWidth = 2

return line

}

return MKPolylineRenderer()

}

Látható, hogy most már szépen megjelennek az üzenetek és az azokat összekötő vonal, ugyanakkor nem tudjuk betölteni a tartalmukat. Ehhez valósítsuk meg a következő delegate metódust.

func mapView(mapView: MKMapView, viewForAnnotation annotation:

MKAnnotation) -> MKAnnotationView? {

if let \_ = annotation as? MessageAnnotation

{

let reusableId = "MessengerAnnotationID"

var pinAnnotationView =

mapView.dequeueReusableAnnotationViewWithIdentifier(reusableId)

as? MKPinAnnotationView

if pinAnnotationView == nil{

pinAnnotationView = MKPinAnnotationView(annotation:

annotation, reuseIdentifier: reusableId)

pinAnnotationView!.pinColor = MKPinAnnotationColor.Green

pinAnnotationView!.canShowCallout = true

let callOutButton = UIButton(type:.DetailDisclosure)

pinAnnotationView!.rightCalloutAccessoryView =

callOutButton

}

else{

pinAnnotationView!.annotation = annotation

}

return pinAnnotationView

}

return nil

}

https://gist.github.com/DjCantor/774af10415fcd2ae3a95#file-mkannotationview-mapview-mkmapview-themapview-viewforannotation-id-mkannotation-annotation

Ezután implementáljuk az előbb hozzáadott gomb eseménykezelőjét.

A gomb megnyomására először lekérdezzük az adott üzenet koordinátáját, majd a beépített reverse geocoding szolgáltatás segítségével megkapjuk a pontos címét a küldés helyének.

Ezután Létrehozunk egy két MKMapItem-ből álló tömböt. Az első elem az előbb meghatározott helyet fogja tartalmazni, a második pedig az aktuális koordinátánk lesz.

Végül ezt és egy megfelelő kulcsokat tartalmazó collection-t átadva meghívjuk a beépített Map alkalmazást, ami a paramétereknek megfelelően megtervezi az útvonalat az átadott pontok között.

func mapView(mapView: MKMapView, annotationView view: MKAnnotationView,

calloutAccessoryControlTapped control: UIControl) {

let coordinate = view.annotation!.coordinate

let geocoder = CLGeocoder()

let location = CLLocation(latitude: coordinate.latitude,

longitude: coordinate.longitude)

geocoder.reverseGeocodeLocation(location, completionHandler: {

(placemarks, error) -> Void in

if (error != nil) {

print("Error occured \(error!.localizedDescription)")

return

}

if placemarks != nil && placemarks!.count > 0 {

let clPlacemark = placemarks!.first!

let placemark = MKPlacemark(placemark: clPlacemark)

let mapItem = MKMapItem(placemark: placemark)

mapItem.name = view.annotation!.title!

var mapItems = [MKMapItem]()

mapItems.append(mapItem)

mapItems.append(MKMapItem.mapItemForCurrentLocation())

let launchOptions: [String:AnyObject] =

[MKLaunchOptionsMapTypeKey :

MKMapType.Hybrid.rawValue,

MKLaunchOptionsDirectionsModeKey :

MKLaunchOptionsDirectionsModeDriving]

MKMapItem.openMapsWithItems(mapItems, launchOptions:

launchOptions)

}

})

}

https://gist.github.com/blazovics/aaa41932bf3b4d8e9bbc#file-calloutaccessorycontroltapped-mapviewcontroller-swift

## Önálló feladatok

1. Módosítsuk a vonal kirajzolásán úgy, hogy azok az üzenetek legyenek összekötve, akiknek a szerzői üzentek már a másik félnek. (Több vonalra lesz szükség.)
2. Az Annotation leftCalloutAccessoryView-jában jelenítsük meg az üzenethez tartozó képet!
3. Az üzenetek mellett jelenítsük meg a saját pozíciónkat is!