TRiGlobe Part 3 – Tagging

We are now going to be creating our tags. Tags are what the users will exchange, digitally, to win points. Somet of these methods we will use later in the tutorial. In this part of the tutorial we will focus on creating a digital object to exchange, which will be our digital or virtual tags.

To create tags we need to have a class to contain our tag information. So create a new NSObject subclass called Tag and explore its .h file:

#import <Foundation/Foundation.h>

@interface Tag : NSObject {

NSNumber \*rglatitude;

NSNumber \*rglongitude;

NSString \*originUdid;

NSString \*destintyUdid;

NSString \*rgcountry;

NSString \*sender;

NSString \*receiver;

}

@property(nonatomic,copy)NSString \*destintyUdid;

@property(nonatomic,copy)NSNumber \*rglatitude;

@property(nonatomic,copy)NSNumber \*rglongitude;

@property(nonatomic,copy)NSString \*originUdid;

@property(nonatomic,copy)NSString \*rgcountry;

@property(nonatomic,copy)NSString \*sender;

@property(nonatomic,copy)NSString \*receiver;

-(id)initWithOriginUdid:(NSString\*)oudid

rglatitude:(NSNumber\*)lati

rglongitude:(NSNumber\*)longi;

-(id)initWithSender:(NSString\*)senderi

receiver:(NSString\*)receiveri

rglatitude:(NSNumber\*)lati

rglongitude:(NSNumber\*)longi

rgcountry:(NSString\*)rgcountri;

-(id)initWithOriginUdid:(NSString\*)oudid

destintyUdid:(NSString\*)dudid

rglatitude:(NSNumber\*)lati

rglongitude:(NSNumber\*)longi

rgcountry:(NSString\*)rgcountri;

@end

and its .m file:

#import "Tag.h"

@implementation Tag

@synthesize originUdid,destintyUdid,sender,receiver;

@synthesize rglatitude,rglongitude,rgcountry;

//unsused so far

-(id)initWithOriginUdid:(NSString\*)oudid

rglatitude:(NSNumber\*)lati

rglongitude:(NSNumber\*)longi

{

NSLog(@"TAG INIT");

if ( (self = [super init]) == nil )

return nil;

self.rglatitude = lati;

self.rglongitude = longi;

self.originUdid = oudid;

return self;

}

-(id)initWithSender:(NSString\*)senderi

receiver:(NSString\*)receiveri

rglatitude:(NSNumber\*)lati

rglongitude:(NSNumber\*)longi

rgcountry:(NSString\*)rgcountri

{

NSLog(@"TAG INIT OF TYPE SENDER RECEIVER");

if ( (self = [super init]) == nil )

return nil;

self.receiver = receiveri;

self.rglatitude = lati;

self.rglongitude = longi;

self.sender = senderi;

self.rgcountry = rgcountri;

return self;

}

-(id)initWithOriginUdid:(NSString\*)oudid

destintyUdid:(NSString\*)dudid

rglatitude:(NSNumber\*)lati

rglongitude:(NSNumber\*)longi

rgcountry:(NSString\*)rgcountri

{

NSLog(@"TAG INIT");

if ( (self = [super init]) == nil )

return nil;

self.destintyUdid = dudid;

self.rglatitude = lati;

self.rglongitude = longi;

self.originUdid = oudid;

self.rgcountry = rgcountri;

return self;

}

@end

This is the code that will create the individual tags when called. The ideal place to call a geotag is from a map view. That is precisely what we will create, a MapViewController. You will find its .h file below:

#import <UIKit/UIKit.h>

#import <MapKit/MapKit.h>

#import "SantiappsHelper.h"

#import "Tag.h"

#import <iAd/iAD.h>

#import "BumpClient.h"

@class CLGeocoder;

@class ModalViewController;

@interface MapViewController : UIViewController <MKMapViewDelegate,ADBannerViewDelegate>{

NSMutableData \*received\_data;

UIButton \*bumpButton;

}

@property (nonatomic, retain) IBOutlet MKMapView \*mapView;

@property (nonatomic, retain) CLGeocoder \*reverseGeocoder;

@property (nonatomic, retain) CLLocationManager \*locationManager;

@property (nonatomic, retain) IBOutlet UIBarButtonItem \*getAddressButton;

@property (nonatomic,retain) IBOutlet UIBarButtonItem \*plotMeButton;

@property (nonatomic,retain) IBOutlet UIButton \*buyMoreButton;

@property (nonatomic,retain) NSArray \*allUserTags;

@property (nonatomic, retain) NSNumber \*rglat;

@property (nonatomic, retain) NSNumber \*rglong;

@property (nonatomic, retain) NSString \*postUDID;

@property (nonatomic, retain) NSDate \*postDate;

@property (nonatomic, retain) NSString \*country;

@property (nonatomic, retain) NSString \*post1;

@property (nonatomic, retain) NSString \*post2;

@property (readwrite) BOOL ceroIsSingle;

@property (nonatomic, retain) Tag \*newlyTag;

//iAds

@property (nonatomic,retain) ADBannerView \*adView;

@property BOOL bannerIsVisible;

// BUMP API

@property (nonatomic, retain) IBOutlet UIImageView \*bumpFourLogo;

@property (nonatomic, retain) IBOutlet UIBarButtonItem \*bumpToConnectButton;

-(IBAction) startBumpButtonPress;

-(void)postToFacebook;

-(IBAction)reverseGeocodeCurrentLocation;

-(IBAction)plotMeMethod;

-(IBAction)buyMoreSelector;

@end

#import "MapViewController.h"

#import "PlaceMarkViewController.h"

#import "MapViewAnnotation.h"

#import "ModalViewController.h"

#import "InAppPurchaseManager.h"

@implementation MapViewController

//In App Purchase method call

-(IBAction) buyMoreSelector{

InAppPurchaseManager \*purchase = [[InAppPurchaseManager alloc] init];

[purchase loadStore];

}

# pragma CREATE SINGLE OR EXCHANGED TAGS

//Exchanges and creates tag with value 2---

-(IBAction)startBumpButtonPress{

self.ceroIsSingle = 1;

[self configureBump];

CLLocation \*userLoc = self.mapView.userLocation.location;

CLLocationCoordinate2D userCoordinate = userLoc.coordinate;

self.rglat = [NSNumber numberWithDouble:self.mapView.userLocation.location.coordinate.longitude];

self.rglong = [NSNumber numberWithDouble:self.mapView.userLocation.location.coordinate.longitude];

// get the stored NSUserPrefs email identifier...

NSUserDefaults \*prefs = [NSUserDefaults standardUserDefaults];

NSString \*storedUserName = [prefs objectForKey:@"storedUser"];

self.newlyTag = [[Tag alloc] initWithSender:storedUserName receiver:nil rglatitude:self.rglat rglongitude:self.rglong rgcountry:nil];

NSLog(@"newTag incomplete:%@,%@,%@,%@,%@",self.newlyTag.sender, self.newlyTag.rglatitude, self.newlyTag.rglongitude, self.newlyTag.receiver, self.newlyTag.rgcountry);

[self postToFacebook];

}

# pragma SOCIAL NETWORKS OR PLOT

//Plots your tags

-(IBAction)plotMeMethod{

// When called loop thru array from vDL and plot...

NSLog(@"Array allUserTags:%@", self.allUserTags);

NSDictionary \*anEntry;

for (anEntry in self.allUserTags){

//create a location variable

CLLocationCoordinate2D location;

location.latitude = [[anEntry objectForKey:@"latitude"] doubleValue];

location.longitude = [[anEntry objectForKey:@"longitude"] doubleValue];

// added to see if locations are printed

NSLog(@"coordlocation:%f, %f",location.latitude, location.longitude);

//Add the annotation to our map view

MapViewAnnotation \*newAnnotation = [[MapViewAnnotation alloc] initWithTitle:@"Tag" andCoordinate:location];

[self.mapView addAnnotation:newAnnotation];

[newAnnotation release];

}

}

- (void)presentLogin{

NSLog(@"login");

NSUserDefaults \*prefs = [NSUserDefaults standardUserDefaults];

if (![prefs stringForKey:@"storedUser"] && ![prefs stringForKey:@"storedPass"]) {

NSLog(@"No user prefs stored");

// BUT WAIT, before all this, lets pop up a view controller for user registration

UIStoryboard\* sb = [UIStoryboard storyboardWithName:@"Storyboard" bundle:nil];

ModalViewController \*popupController = [sb instantiateViewControllerWithIdentifier:@"ModalViewController"];

[self presentViewController:popupController animated:NO completion:nil];

} else {

NSString \*storedUser = [NSString stringWithFormat:@"User:%@",[prefs stringForKey:@"storedUser"]];

NSString \*storedPass = [NSString stringWithFormat:@"User:%@",[prefs stringForKey:@"storedPass"]];

UIAlertView \*internetAlert = [[UIAlertView alloc] initWithTitle:storedUser

message:storedPass

delegate:self

cancelButtonTitle:@"Cancel"

otherButtonTitles:@"Ok", nil];

[internetAlert show];

}

}

-(void)viewDidAppear:(BOOL)animated{

[super viewDidAppear:YES];

[self performSelector:@selector(presentLogin) withObject:nil afterDelay:1.5];

}

// Implement viewDidLoad to do additional setup after loading the view, typically from a nib.

- (void)viewDidLoad {

[super viewDidLoad];

self.mapView.delegate = self;

self.mapView.showsUserLocation = YES;

//GET STORED USER IN ORDER TO LOAD-FETCH DATA

NSUserDefaults \*prefs = [NSUserDefaults standardUserDefaults];

// check if location enabled

BOOL locationAllowed = [CLLocationManager locationServicesEnabled];

if (locationAllowed==NO) {

UIAlertView \*alert = [[UIAlertView alloc] initWithTitle:@"Location Service Disabled"

message:@"To re-enable, please go to Settings and turn on Location Service for this app."

delegate:nil

cancelButtonTitle:@"OK"

otherButtonTitles:nil];

[alert show];

[alert release];

}

// iAds

self.adView = [[ADBannerView alloc] initWithFrame:CGRectZero];

self.adView.frame = CGRectOffset(self.adView.frame, 0, -50);

self.adView.requiredContentSizeIdentifiers = [NSSet setWithObjects:ADBannerContentSizeIdentifierPortrait,ADBannerContentSizeIdentifierLandscape,nil];

self.adView.currentContentSizeIdentifier = ADBannerContentSizeIdentifierPortrait;

[self.view addSubview:self.adView];

self.adView.delegate=self;

self.bannerIsVisible = NO;

}

#pragma iAd Delegate Methods

- (void)bannerViewDidLoadAd:(ADBannerView \*)banner{

if (!self.bannerIsVisible)

{

[UIView beginAnimations:@"animateAdBannerOn" context:NULL];

// banner is invisible now and moved out of the screen on 50 px

banner.frame = CGRectOffset(banner.frame, 0, 50);

[UIView commitAnimations];

self.bannerIsVisible = YES;

}

}

- (void)bannerView:(ADBannerView \*)banner didFailToReceiveAdWithError:(NSError \*)error{

if (self.bannerIsVisible)

{

[UIView beginAnimations:@"animateAdBannerOff" context:NULL];

// banner is visible and we move it out of the screen, due to connection issue

banner.frame = CGRectOffset(banner.frame, 0, -50);

[UIView commitAnimations];

self.bannerIsVisible = NO;

}

}

- (BOOL)bannerViewActionShouldBegin:(ADBannerView \*)banner willLeaveApplication:(BOOL)willLeave{

NSLog(@"Banner view is beginning an ad action");

BOOL shouldExecuteAction = YES;

if (!willLeave && shouldExecuteAction)

{

// stop all interactive processes in the app

}

return shouldExecuteAction;

}

- (void)bannerViewActionDidFinish:(ADBannerView \*)banner{

// resume everything you've stopped

}

- (void)viewDidUnload {

[super viewDidUnload];

self.mapView = nil;

self.getAddressButton = nil;

//[bumpConn stopBump];

}

- (void)didReceiveMemoryWarning {

// Releases the view if it doesn't have a superview.

[super didReceiveMemoryWarning];

// Release any cached data, images, etc that aren't in use.

}

#pragma MAPVIEW DELEGATE METHODS

- (void)mapView:(MKMapView \*)mapView didUpdateUserLocation:(MKUserLocation \*)userLocation {

// We have location, do your logic of setting the map region here.

CLLocationCoordinate2D zoomLocation;

zoomLocation = self.mapView.userLocation.coordinate;

//NSLog(@"self.ciudad is nil - a city was not picked, using %g,%g", zoomLocation.latitude, zoomLocation.longitude);

CLLocationDistance visibleDistance = 5000; // 5 kilometers

MKCoordinateRegion adjustedRegion = MKCoordinateRegionMakeWithDistance(zoomLocation, visibleDistance, visibleDistance);

[\_mapView setRegion:adjustedRegion animated:YES];

}

- (void)mapView:(MKMapView \*)mapView didAddAnnotationViews:(NSArray \*)views{

// we have received our current location, so enable the "Get Current Address" button

[self.getAddressButton setEnabled:YES];

}

#pragma BUMP METHODS

-(void) quickAlert:(NSString \*)titleText msgText:(NSString \*)msgText{

UIAlertView \*alert = [[UIAlertView alloc] initWithTitle:titleText message:msgText delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];

[alert show];

[alert release];

}

- (void) configureBump {

NSLog(@"configureBump>");

[BumpClient configureWithAPIKey:@"9ed7d2fa665f462eb05d5f87d4c07124" andUserID:[[UIDevice currentDevice] name]];

[[BumpClient sharedClient] setMatchBlock:^(BumpChannelID channel) {

NSLog(@"Matched with user: %@", [[BumpClient sharedClient] userIDForChannel:channel]);

[[BumpClient sharedClient] confirmMatch:YES onChannel:channel];

}];

NSLog(@"setMatchBlock>");

[[BumpClient sharedClient] setChannelConfirmedBlock:^(BumpChannelID channel) {

NSLog(@"Channel with %@ confirmed.", [[BumpClient sharedClient] userIDForChannel:channel]);

//1.SEND USERNAME STORED as DATA>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

NSUserDefaults \*prefs = [NSUserDefaults standardUserDefaults];

NSData \*moveChunk = [NSKeyedArchiver archivedDataWithRootObject:[prefs stringForKey:@"storedUser"]];

[[BumpClient sharedClient] sendData:moveChunk toChannel:channel];

}];

NSLog(@"setChannelConfirmedBlock>");

[[BumpClient sharedClient] setDataReceivedBlock:^(BumpChannelID channel, NSData \*data) {

//2.RECEIVE DATA>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

NSString \*dataReceived = [NSKeyedUnarchiver unarchiveObjectWithData:data];

NSLog(@"Data received from %@: %@", [[BumpClient sharedClient] userIDForChannel:channel], dataReceived);

[self quickAlert:@"Data Got :)" msgText:dataReceived];

}];

NSLog(@"setDataReceivedBlock>");

[[BumpClient sharedClient] setConnectionStateChangedBlock:^(BOOL connected) {

if (connected) {

NSLog(@"Bump connected...");

} else {

NSLog(@"Bump disconnected...");

}

}];

NSLog(@"setConnectionStateChangedBlock>");

[[BumpClient sharedClient] setBumpEventBlock:^(bump\_event event) {

switch(event) {

case BUMP\_EVENT\_BUMP:

NSLog(@"Bump detected.");

break;

case BUMP\_EVENT\_NO\_MATCH:

NSLog(@"No match.");

break;

}

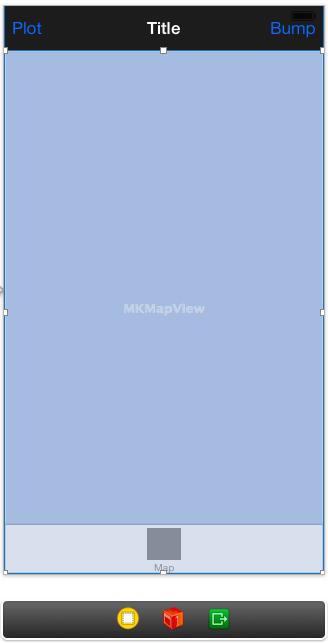
}];

NSLog(@"setBumpEventBlock>");

}

@end

To complement this class you must create a UI by creating a scene of subclass UIViewController and add a MapView into it between the navigation bar and the tab bar like so:



It is time to step through the code step by step. You will see some extra code that we are throwing in but we will use in the final section of this tutorial where we add iAds and Social.

Remember to add MapKit and iAdKit to the project in the Target Settings > Build Phases > Link Libraries. As you can see, the MapViewController imports the MapKit framework, the SantiappsHelper class, the Tag class, the iAD framework and a class called BumpClient. This is the BumpAPI which we will add later.

On the other hand, in the MapViewController you see that we use 2 new classes; InAppPurchaseManager and MapViewAnnotation. The InAppPurchaseManager class we will not use until later.

So let’s cover the classes we will use in this part of the tutorial. First, the MapViewAnnotation:

#import <Foundation/Foundation.h>

#import <MapKit/MapKit.h>

@interface MapViewAnnotation : NSObject <MKAnnotation>{

NSString \*title;

CLLocationCoordinate2D coordinate;

}

@property (nonatomic, copy) NSString \*title;

@property (nonatomic, readonly) CLLocationCoordinate2D coordinate;

- (id)initWithTitle:(NSString \*)ttl andCoordinate:(CLLocationCoordinate2D)c2d;

@end

and its implementation looks like:

#import "MapViewAnnotation.h"

@implementation MapViewAnnotation

@synthesize title, coordinate;

- (id)initWithTitle:(NSString \*)ttl andCoordinate:(CLLocationCoordinate2D)c2d {

[super init];

title = ttl;

coordinate = c2d;

return self;

}

@end

This simply creates an MKAnnotation by subclassing it and adding a title and an actual coordinate. In the end we create a MapViewAnnotation with a custom initializer.

That’s it for now so see you next time!