// Firebase do the realtime data better than parse <ls

Back4App.com aws.amazon.com

single view app pod init,

cocoapods ⇒ pod 'Parse'

↳ pod install

// if download examples online ⇒ run pod install to use the code

⚠ missing the navigation bar ⇒ set the segue to the navigation Controller

// AppDelegate.swift

```
import UIKit
import Parse

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
        [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        let configuration = ParseClientConfiguration {
            $0.applicationId = "....."
            $0.clientKey = "....."          ⇒ Get From the back4App dashboard
            $0.server = "....."
        }
        Parse.initialize(with: configuration)

        return true
    }
```

// SceneDelegate.swift

```
import UIKit
import

class SceneDelegate: UIResponder, UIWindowSceneDelegate {
    var window: UIWindow?
    func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions: UIScene.ConnectingOptions) {

        let currentUser = PFUser.current()
        if currentUser != nil {
            let board = UIStoryboard(name: "Main", bundle: nil)
            let navigationController = board.instantiateViewController(identifier: "navigationController") as!
                UINavigationController      ⇒ give identifier to navigationcontroller in main.storyboard
            window?.rootViewController = navigationController
        }
        guard let _ = (scene as? UIWindowScene) else { return }
    }
```

// new File -> swift file -> PlaceModel.swift

```
import Foundation

class PlaceModel{

    static let sharedInstance = PlaceModel()

    var placeName = ""
    var placeType = ""
    var placeAtmosphere = ""
    var placeImage = UIImage()
    var placeLatitude = ""
    var placeLongitude = ""

    private init(){}
}
```

// example:

```
class AccountManager{
    static let sharedInstance = AccountManager()
    var userInfo = (ID: "bobthedev", password: "01030946")
    func network(){

    }
    private init(){}
}
```

// ViewController.swift ⌘ right click -> refactor -> rename


```
import UIKit
import Parse

class ViewController: UIViewController{
    @IBOutlet weak var userNameText : UITextField!
    @IBOutlet weak var passwordText : UITextField!

    override func viewDidLoad(){
        super.viewDidLoad()

    }
    @IBAction func signInClicked (_ sender:Any){
        if userName.text != "" && passwordText.text != ""{
            PFUser.logInWithUsername(inBackground: userNameText.text!, password: passwordText.text!){ (user, error) in
                if error != nil{
                    self.makeAlert(titleInput: "Error", messageInput: error?.localizedDescription ?? "Error!")
                }else{
                    self.performSegue(withIdentifier: "toPlacesVC", sender: nil)

                }
            }
        }else{
            makeAlert(titleInput: "Error", messageInput: "Username/Password??")
        }
    }

    @IBAction func signUpClicked(_ sender:Any){
        if userName.text != "" && passwordText.text != ""{
            let user = PFUser()
            user.username = userNameText.text!
            user.password = passwordText.text!

            user.signUpInBackground(block{(success, error) in
                if error != nil{
                    self.makeAlert(titleInput: "Error", messageInput: error?.localizedDescription ?? "Error!")
                }else{
                    self.performSegue(withIdentifier: "toPlacesVC", sender: nil)
                }
            }

        }else{
            makeAlert(titleInput: "Error", messageInput: "Username/Password??")
        }
    }
}
```

```
//let parseObject = PFObject(className: "Fruits")
parseObject["name"] = "Banna"
parseObject["calories"] = 100
parseObject.saveInBackground{ (success, error) in
    if error != nil{
        print(error?.localizedDescription)
    }else{
        print("uploaded")
    }
}
```

```
//let query = PFQuery(className: "Fruits")
query.whereKey("name", equalTo: "Apple")
query.findObjectsInBackground{ (objects, error) in
    if error != nil{
        print(error?.localizedDescription)
    } else{
        print(objects)
    }
}
```

// present Modally -> full screen
editor -> embeded in navigation controller

// PlacesVC.swift

```
import UIKit
import Parse

class PlacesVC : UIViewController, UITableViewDelegate, UITableViewDataSource{
    @IBOutlet weak var tableView : UITableView!
    var placeNameArray = [String]()
    var placeIdArray = [String]()
    var selectedPlaceId = ""

    override func viewDidLoad(){
        super.viewDidLoad()

        navigationController?.navigationBar.topItem?.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem:
            UIBarButtonItem.SystemItem.add, target:self, action:#selector(addButtonClicked))
        navigationController?.navigationBar.topItem?.leftBarButtonItem = UIBarButtonItem(title:"Logout", style:UIBarButtonItem.Style.plain,
            target:self, action:#selector(logoutButtonClicked))

        tableView.delegate = self
        tableView.dataSource = self
        getDataFromParse()
    }

    func getDataFromParse(){
        let query = PFQuery(className:"Places")                    8:17
        query.findObjectsInBackground{(objects, error) in
            if error != nil {
                self.makeAlert(titleInput:"Error", message:error?.localizedDescription)
            }else{
                if objects != nil{
                    self.placeIdArray.removeAll(keepingCapacity:False)
                    self.placeNameArray.removeAll(keepingCapacity:False)
                    for object in objects!{
                        if let placeName=object.object(forKey:"name") as? String{
                            if let placeId=object.objectId as? String{
                                self.placeNameArray.append(placeName)
                                self.placeIdArray.append(placeId)
                            }
                        }
                    }
                    self.tableView.reloadData()
                }
            }
        }
    }

    @objc func addButtonClicked(){

        self.performSegue(withIdentifier:"toAddPlaceVC", sender:nil)
```

```
}

@objc func logoutButtonClicked(){
    PFUser.logOutInBackground{(error) in
        if error != nil{
            self.makeAlert(titleInput: "Error", message: error?.localizedDescription)
        }else{
            self.performSegue(withIdentifier: "toSignUpVC", sender: nil)
        }
    }
}

func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int{
    return placeNameArray.count

}

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell{
    let cell = UITableViewCell()
    cell.textLabel?.text = placeNameArray[indexPath.row]
    return cell
}

func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath){
    selectedPlaceId = placeIdArray[indexPath.row]

    self.performSegue(withIdentifier: "toViewController", sender: nil)
}

override func prepare(for segue: UIStoryboardSegue, sender: Any?){
    if segue.identifier == "toDetailsVC"{
        let destinationVC = segue.destination as! DetailsVC
        destinationVC.chosenPlaceId = selectedPlaceId

    }
}

func makeAlert(titleInput: String, messageInput: String){
    let alert = UIAlertController(title: titleInput, message: messageInput, preferredStyle:
        UIAlertController.Style.alert)
    let okButton = UIAlertAction(title: "Ok", style: UIAlertAction.Style.default, handler: nil)
    alert.addAction(okButton)
    self.present(alert, animated: true, completion: nil)
}
```

```
//AddPlaceVC.swift
import UIKit

class AddPlaceVC: UIViewController, UIImagePickerControllerDelegate, UINavigationControllerDelegate{
    @IBOutlet weak var placeNameText: UITextField!
    ↓ similiar ones

    override func viewDidLoad(){
        super.viewDidLoad()

        imageView.isUserInteractionEnabled = true
        let imageTapRecognizer = UITapGestureRecognizer(target: self, action: #selector(selectImage))
        imageView.addGestureRecognizer(imageTapRecognizer)
    }

    @objc func selectImage(){
        let picker = UIImagePickerController()
        picker.delegate = self
        picker.sourceType = .photoLibrary
        picker.allowEditing = true
        present(picker, animated: true, completion: nil)
    }

    func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info:
        [UIImagePickerController.InfoKey: Any]){
            imageView.image = info[.originalImage] as? UIImage

            self.dismiss(animated: true, completion: nil)
    }

                                                                    ← explaination
//before IOS12 → info.plist ⇒ plus button ⇒ Privacy - Photo Library Usage Description → Value: to access photos


    @IBAction func nextButtonClicked(_ sender: Any){
        if placeNameText.text! != "" && placeTypeText.text! != "" && placeAtmosphere.text != ""{
            if let chosenImage = placeImageView.image{
                let placeModel = PlaceModel.sharedInstance
                placeModel.placeName = placeNameText.text!
                    ↓ similar ones
                placeModel.placeImage = chosenImage

                performSegue(withIdentifier: "toMapVC", sender: nil)
            }
        }else{
            //Alert
        }
    }
```

```
// MapVC.swift          // embed in another navigation controller for the MapVC

import UIKit
import MapKit
import Parse

class MapVC: UIViewController, MKMapViewDelegate, CLLocationManagerDelegate{
    @IBOutlet weak var mapView: MKMapView!
    var locationManager = CLLocationManager()


    override func viewDidLoad(){
        super.viewDidLoad()

        navigationController?.navigationBar.topItem?.rightBarButtonItem = UIBarButtonItem(title: "Save", style:
            UIBarButtonItem.Style.plain, target: self, action: #selector(saveButtonClicked))

        navigationController?.navigationBar.topItem?.rightBarButtonItem = UIBarButtonItem(title: "< back", style:
            UIBarButtonItem.Style.plain, target: self, action: #selector(backButtonClicked))

        mapView.delegate = self
        locationManager.delegate = self          // p-list > privacy-Location when in use ..... > "to use ym..."
        locationManager.desiredAccuracy = kCLLocationAccuracyBest
        locationManager.requestWhenInUseAuthorization()
        locationManager.startUpdatingLocation()

        let gestureRecognizer = UILongPressGestureRecognizer(target: self, action: #selector(chooseLocation(gestureRecognizer:)))
        gestureRecognizer.minimumPressDuration = 3
        mapView.addGestureRecognizer(gestureRecognizer)
    }

    @objc func chooseLocation(gestureRecognizer: UILongPressGestureRecognizer){
        if gestureRecognizer.state == .began{
            let touchedPoint = gestureRecognizer.location(in: self.mapView)
            let touchedCoordinates = self.mapView.convert(touchedPoint, toCoordinateFrom: self.mapView)

            let annotation = MKPointAnnotation()
            annotation.coordinate = touchedCoordinates
            annotation.title = PlaceModel.sharedInstance.placeName
            annotation.subtitle = PlaceModel.sharedInstance.placeType
            self.mapView.addAnnotation(annotation)

            PlaceModel.sharedInstance.placeLatitude = String(touchedCoordinates.latitude)
            PlaceModel.sharedInstance.placeLongitude = String(touchedCoordinates.longitude)
        }
}
```

```
func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]){
    locationManager.stopUpdatingLocation() // so the user can pan around while moving
    let location = CLLocation(coordinate2D(latitude: locations[0].coordinate.latitude,
                                           longitude: locations[0].coordinate.longitude)
    let span = MKCoordinateSpan(latitudeDelta: 0.05, longitudeDelta: 0.05)
    let region = MKCoordinateRegion(center: Location, span: span)
    mapView.setRegion(Region, animated: true)

}

@objc func saveButtonClicked(){
    let placeModel = PlaceModel.sharedInstance
    let object = PFObject(className: "places")
    object["name"] = placeModel.placeName
    ↓ similar ones

    if let imageData = placeModel.placeImage.jpegData(compressionQuality: 0.5){
        object["image"] = PFFileObject(name: "image.jpg", data: imageData)

        Object.saveInBackground{ (sucess, error) in
            if error != nil{
                //Alert
            } else{
                self.performSegue(withIdentifier: "fromMapVC-toPlacesVC", sender: nil)
            }
        }
    }
}


@objc func backButtonClicked(){
    // navigationController?.popViewController(animated: true) ← not work for this situation
    self.dismiss(animated: true, completion: nil)
}
```

```
// DetailsVC.swift
import UIKit
import MapKit
import Parse

class DetailsVC: UIViewController, MKMapViewDelegate, CLLocationManagerDelegate{
    @IBOutlet weak var detailsImageView: UIImageView!
    ↓ similar ones

    var chosenPlaceId = ""
    var chosenLatitude = Double()
    var chosenLongitude = Double()

    override func viewDidLoad(){
        super.viewDidLoad()
        getDataFromParse()
        detailsMapView.delegate = self
    }

    func getDataFromParse(){
        let query = PFQuery(className: "Places")
        query.whereKey("objectId", equalTo: chosenPlaceId)
        query.findObjectsInBackground{(object, error) in
            if error != nil{
                // Alert
            }else{
                if objects != nil{
                    if objects.count > 0{
                        let chosenPlaceObject = objects[0]
                        if let placeName = chosenPlaceObject(forKey: "name") as? String{
                            self.detailsNameLabel.text = placeName
                        }
                        ↓ similar ones
                        if let placeLatitude = chosenPlaceObject(forKey: "latitude") as? String{
                            if let placeLatitudeDouble = Double(placeLatitude){
                                self.chosenLatitude = placeLatitudeDouble
                            }
                        }
                        ↓ similar for longitude

                        if let imageData = chosenPlaceObject.object(forKey: "image") as? PFFileObject{
                            imageData.getDataInBackground{(data, error) in
                                if error != nil{
                                    if data != nil{
                                        self.detailsImageView.image = UIImage(data: data!)
                                    }
                                }
                            }
                        }

                        let location = CLLocation(coordinate2D(latitude: self.chosenLatitude,
                                                               longitude: self.chosenLongitude)
                        let span = MKCoordinateSpan(latitudeDelta: 0.05, longitudeDelta: 0.05)
                        let region = MKCoordinateRegion(center: Location, span: span)
```

```
                self.detailsMapView.setRegion(region, animated: true)

                let annotation = MKPointAnnotation()
                annotation.coordinate = location
                annotation.title = self.detailsNameLabel.text!
                annotation.subtitle = self.detailsTypeLabel.text!
                self.detailsMapView.addAnnotation(annotation)

            }
        }
    }
}

func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView?{
    if annotation is MKUserLocation{
        return nil
    }
    let reuseId = "myAnnotation"
    var pinView = mapView.dequeReusableAnnotationView(withIdentifier: reuseId) as? MKPinAnnotationView
    if pinView == nil{
        pinView = MKPinAnnotationView(annotation: annotation, reuseIdentifier: reuseId)
        pinView?.canShowCallout = true
        pinView?.tintColor = UIColor.black

        let button = UIButton(type: UIButton.ButtonType.detailDisclosure)
        pinView?.rightCalloutAccessoryView = button
    }else{
        pinView?.annotation = annotation
    }
    return pinView
}

func mapView(_ mapView: MKMapView, annotationView view: MKAnnotationView, calloutAccessoryControlTapped control: UIControl){
    if self.chosenLongitude != 0.0 && self.chosenLatitude != 0.0{
        var requestLocation = CLLocation(latitude: self.chosenLatitude, longitude: self.chosenLongitude)
        CLGeocoder().reverseGeocodeLocation(requestLocation){ (placemarks, error) in
            if let placemark = placemarks{
                if placemark.count > 0{
                    let newPlacemark = MKPlacemark(placemark: placemark[0])
                    let item = MKMapItem(placemark: newPlacemark)
                    item.name = self.detailsNameLabel.text
                    let launchOptions = [MKLaunchOptionsDirectionsModeKey: MKLaunchOptionsDirectionsModeDriving]
                    item.openInMaps(launchOptions: launchOptions)
                }
            }
        }
    }
}
```