

Documentación

MapaFI

Índice

1. Acerca de la aplicación.	p.2
2. Funciones.	p.3
• Inicio de sesión.	p.3
• Creación de usuario.	p.4
• Mapa Interactivo.	p.5
• Avisos y noticias de la facultad.	p.5
3. Desarrollo.	p.6
4. Vistas, código de la aplicación y base de datos.	p.9
○ Vistas	p.9
○ Código de la aplicación	p.10
○ Base de datos	p.16
5. Bibliografía.	p.16

1- Acerca de la aplicación

MapaFI

Propósito: Se creó la aplicación con el fin de ayudar al alumno perteneciente a la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, a poder identificar, por medio de un mapa interactivo, Edificios, Laboratorios, Bibliotecas, Auditorios y Tiendas dentro de la facultad, ya que estas muchas veces no se encuentran con mucha facilidad, así como también, dar noticia al alumno de eventos, avisos y alertas, ya que su distribución no suele ser muy buena dentro de la comunidad de la Facultad de Ingeniería.



2- Funciones

Al iniciar la aplicación, se mostrará una pantalla de carga, la cual tendrá el logo de la facultad de ingeniería, al terminar de cargar, la aplicación lo mandará a la primera pantalla, el inicio de sesión.

a. Inicio de sesión

Al terminar de iniciar la aplicación, se enviará a la pantalla de inicio de sesión, donde se pedirá un usuario y una contraseña para poder ingresar a la aplicación, en caso de no contar con una, o tener usuario/contraseña equivocada, no se permitirá el acceso a la aplicación.

En caso de no tener una cuenta: Si no se tiene una cuenta para ingresar a la aplicación, se puede crear una por medio del botón “Registrarse”.



b. Creación de usuario

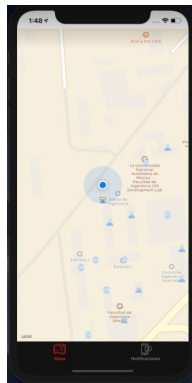
Al pedir registrarse por medio del botón, se enviará a una pantalla diferente, en la pantalla de registro, se le pedirá un correo que no esté en uso, una contraseña y una confirmación de ésta (para que no haya problema con escribir mal la contraseña deseada y no poder iniciar sesión después), una vez que todo esto esté escrito, apriete el botón “registrarse” y se creará su usuario, regresando a la pantalla de inicio de sesión (p.2), donde puede usar su nueva cuenta y contraseña para iniciar sesión, ya está listo para usar la aplicación!



Cancelar creación de usuario: si desea cancelar la creación de un nuevo usuario, puede apretar el botón “Cancelar” y lo regresará a la pantalla de inicio de sesión (p.2).

c. Mapa Interactivo

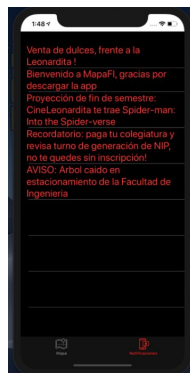
Al introducir correctamente su cuenta de correo y contraseña, la aplicación automáticamente desplegará una vista en la cual se mostrará un mapa con diversos iconos. Dichos iconos le ayudarán a identificar todos los edificios y laboratorios con los que cuenta la facultad. Debido a que al presionar alguno de estos iconos le mostrará el nombre del edificio o laboratorio donde se encuentra.



Regresar al mapa: Una vez ya iniciada sesión, se puede regresar al mapa, apretando el icono de mapa en la parte inferior de la pantalla principal.

d. Avisos y noticias de la FI

Al apretar el icono de notificaciones, en la barra que se encuentra en el extremo inferior de la pantalla principal, se abrirá la pestaña de notificaciones, la cual actualizará noticias y avisos de la facultad automáticamente, conectándose a una base de datos y obteniendo de ahí la información.



Desarrollo

La idea original, siempre fue hacer un mapa interactivo de la facultad, antes de proponerle al profesor la idea de la aplicación, se consideraron varias funciones adicionales, pero por cuestión de tiempo, se descartaron de la propuesta de ideas, sin embargo, se podrían añadir en un futuro, si se desea.

La aplicación se propuso en clase, como un mapa interactivo, al cual solo se puede entrar, si se crea una cuenta para la aplicación, la cual autentica que el usuario cuente con la cuenta y contraseña correcta, una vez que lo hace, se te envía a un mapa interactivo del Anexo y la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, donde también se podrán recibir noticias y avisos de sucesos dentro de la misma.

Después de presentar la aplicación, comenzó el desarrollo de la misma, el cual constó de 6 versiones, las cuales tomaron algunos días cada una en terminarse, ya que se consultaba la documentación de swift o se veía en clase y de ahí se aprendía lo necesario para continuar con el desarrollo.

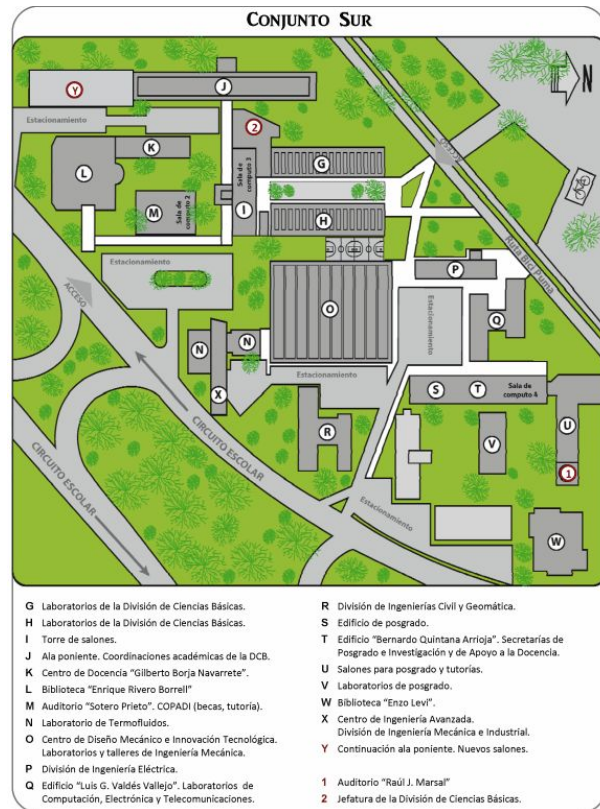
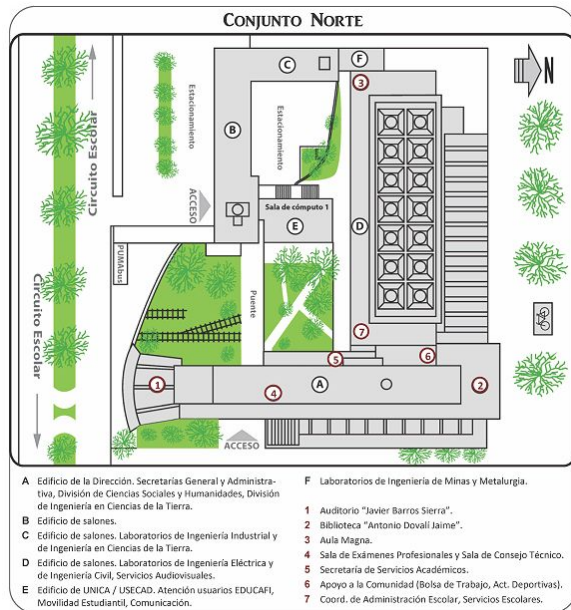
Ya que todo el código de cada actualización y la configuración de la base de datos se encuentra en la versión final, se explicará después de mencionar el avance por versiones de la aplicación

Versión 1.0

En la versión 1.0, se creó un mapa que pudiera mostrar tu ubicación mediante un pin, y la actualice en tiempo real, mostrando por donde avanzas utilizando el GPS del dispositivo, se probó utilizando el mapa normal y el mapa satelital pero por cuestiones de estética y ser más práctico, se optó por el primero.

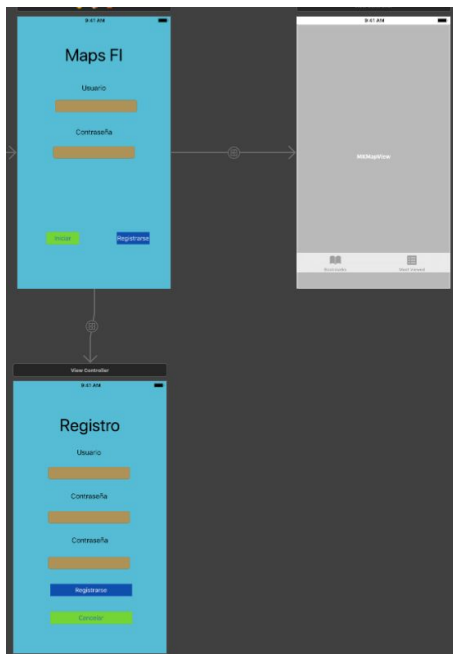
Versión 1.1

En la versión 1.1, se crearon las anotaciones del mapa, las cuales se fueron colocando con la ayuda de los mapas de la facultad, ir a buscar los edificios en persona y compararlos con el mapa que ya teníamos completo, se consideró que los pines se actualicen de la base de datos, pero como el mapa se refresca continuamente para obtener y mostrar tu ubicación en tiempo real, se optó por ponerlos dentro del código, al no ser tantos como para necesitar ahorrar líneas de código pero también considerando que los estaría descargando continuamente y eso utiliza datos.



Versión 1.2

En la versión 1.2, se crearon las pantallas de inicio de sesión y registro de usuario, pero sin funcionalidad, ya que aún no se creaba la base de datos de Firebase, y por lo mismo no se había implementado, también se implementó el mapa dentro de una TabBar, la cual permite cambiar entre las pantallas del mapa y la nueva creada pantalla de notificaciones, la cual en esta versión, solamente mostraba notificaciones escritas dentro del código, para ver si funciona y como se ve, aún no contaban con funcionalidad, como las pantallas de autenticación y registro, se podría decir que ésta versión es el demo de la aplicación completa.



Versión 1.3

En la versión 1.3 se instalaron los pods, los cuales se necesitaban para poder utilizar la base de datos de Firebase en la aplicación, los pods utilizados fueron:

- pod 'Firebase/Core' se utiliza para poder utilizar Firebase
- pod 'Firebase/Firestore' se utiliza para poder almacenar datos
- pod 'Firebase/Auth' se utiliza para poder autenticar y crear usuarios

Dato: hubo algún error con los pods en la versión 1.5, el cual solamente pudimos corregir creando un archivo nuevo e instalando los pods ahí y volviendo a hacer el proyecto en el nuevo archivo, lo cual no tomó mucho tiempo ya que se contaba con el código del archivo anterior.

Versión 1.4

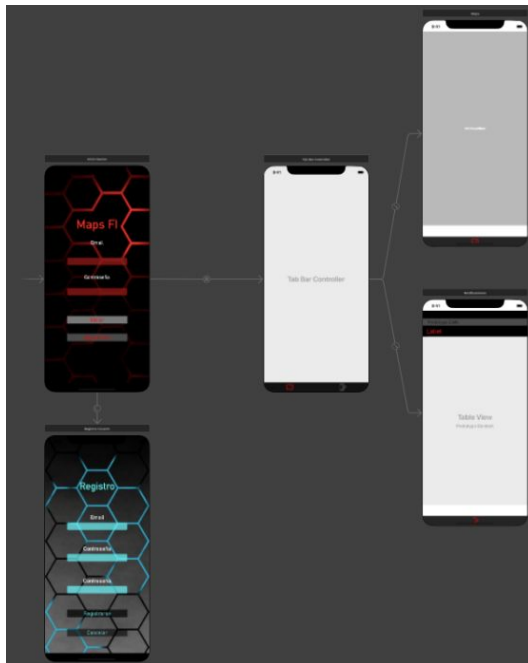
En la versión 1.4, se implementó la base de datos, utilizando Firebase, haciendo posible la creación de cuentas de usuarios y el inicio de sesión de éstos si utilizan la cuenta/contraseña correcta, también se actualiza la tabla de notificaciones por medio de firebase, agregando y quitando en tiempo real las nuevas noticias.

Versión 1.5

Para la versión 1.5 se tenían planeadas dos cosas, agregar la función para el mapa, que si no se encuentra dentro de la facultad o el anexo, daría aviso de que se encuentra fuera del perímetro para el que trabaja la aplicación, la segunda cosa planeada, fue una actualización estética de la aplicación, lamentablemente, se adelantó la fecha de entrega del proyecto y ocurrió un error con los pods, mencionado previamente, por lo cual solo se pudo hacer la parte estética de la actualización.

Vistas, Código de la aplicación y Base de datos

Vistas: el código tiene cuatro vistas, la pantalla de inicio de sesión, la pantalla de creación de cuenta, la vista del mapa y la vista de notificaciones, estas dos últimas son controladas por una tab bar, permitiendo cambiar fácilmente entre una y la otra.



Código: El código se divide en distintas secciones; Esta parte del código está específicamente diseñada para el inicio de sesión de los usuarios, en este apartado el código se encarga de rectificar de que el correo y la contraseña que el usuario introdujo se encuentren almacenados en la base de datos.

```
import UIKit
import Firebase

class InicioSesion: UIViewController {

    @IBOutlet weak var CorreoTextBox: UITextField!
    @IBOutlet weak var ContraseñaTextBox: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    @IBAction func InicioSesion(_ sender: UIButton) {

        guard let correo = CorreoTextBox.text,
              let password = ContraseñaTextBox.text else { return }

        Auth.auth().signIn(withEmail: correo, password: password) { (data, error) in
            if let error = error {
                debugPrint(error.localizedDescription)
            } else {
                self.performSegue(withIdentifier: "VistaMapa", sender: self)
            }
        }
    }
}
```

En este otro apartado del código se encuentra el registro de usuarios por medio de una base de datos, en esta sección del código los usuarios crean una cuenta proporcionando su correo y una contraseña, al finalizar esta acción, se crea un dato con dichos campos.

```
import UIKit
import Firebase

class RegistroUsuario: UIViewController {

    @IBOutlet weak var CorreoTextBox: UITextField!
    @IBOutlet weak var ContraseñaTextBox1: UITextField!
    @IBOutlet weak var ContraseñaTextBox2: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    func salir() {
        dismiss(animated: true, completion: nil)
    }

    @IBAction func registrarse(_ sender: UIButton) {

        guard let correo = CorreoTextBox.text, let password1 = ContraseñaTextBox1.text, let password2 = ContraseñaTextBox2.text else {return}

        if (correo != "" && password1 != "" && password2 != "") {

            Auth.auth().createUser(withEmail: correo, password: password1) { (data, error) in
                if let error = error {
                    debugPrint(error.localizedDescription)
                }

                let user = data?.user

                guard let userId = user?.uid else { return }

                Firestore.firestore().collection("users").document(userId).setData(["email": correo], completion: { (error)
                    in
                        if let error = error {
                            debugPrint(error)
                        } else { self.salir() }
                    })
            })
        }
    }

    @IBAction func cancelar(_ sender: UIButton) {
        salir()
    }
}
```

En esta parte del código donde implementamos la visualización del mapa, utilizando un arreglo de pines para guardarlos dentro del código y los implementamos por medio de un for.

```
import MapKit
import CoreLocation

class ViewController: UIViewController, CLLocationManagerDelegate, MKMapViewDelegate {

    @IBOutlet weak var mapView: MKMapView! // Es el objeto mapa

    let locationManager = CLLocationManager() //Determina si el usuario tiene habilitados los servicios de
    localización. Si no se le indicará al usuario la advertencia

    override func viewDidLoad() {
        super.viewDidLoad()
        mapView.delegate = self // el mapa se hace responsable de si mismo

        mapView.showsUserLocation = true //La anotación que representa la ubicación del usuario.

        if (CLLocationManager.locationServicesEnabled() == true){ //Devuelve SI si el dispositivo admite el servicio
        de lo contrario NO

            if(CLLocationManager.authorizationStatus() == .restricted || CLLocationManager.authorizationStatus() ==
            .denied || CLLocationManager.authorizationStatus() == .notDetermined ){

                locationManager.requestWhenInUseAuthorization()//Llamar a este método activará un mensaje para
                solicitar Autorización del usuario.

            }

            locationManager.desiredAccuracy = 1.0 // Especifica que las actualizaciones de ubicación pueden
            pausarse automáticamente cuando sea posible.

            locationManager.delegate = self //La determinación de cuándo se pueden pausar automáticamente las
            actualizaciones de ubicación.
            locationManager.startUpdatingLocation() // activa la función locationManager

            for i in notaciones{
                let annotation = PersonalAnnotation()
                annotation.coordinate = CLLocationCoordinate2D(latitude: i.latitud, longitude: i.longitud)
                annotation.title = i.title
                annotation.subtitle = i.subtitle
                annotation.imagePin = i.imagen

                mapView.addAnnotation(annotation)
            }

        }else{
            print("Activa tu localización")
        }
    }
}
```

```
func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
    print(locations[0])

    let region = MKCoordinateRegion(center: CLLocationCoordinate2D(latitude: locations[0].coordinate.latitude,
longitude: locations[0].coordinate.longitude) , latitudinalMeters: 250 , longitudinalMeters: 250)

    self.mapView.setRegion(region, animated: true)
}

func locationManager(_ manager: CLLocationManager, didFailWithError error: Error) {
    print("no puede acceder a su ubicación actual")
}

func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView? {
    //print("function")
    if annotation is MKUserLocation{
        return nil
    }
    var newAnnotation = mapView.dequeueReusableAnnotationView(withIdentifier: "PersonalAnnotation")
    if newAnnotation == nil{
        newAnnotation = MKAnnotationView(annotation: annotation, reuseIdentifier: "PersonalAnnotation")
        newAnnotation?.canShowCallout = true
    }else{
        newAnnotation?.annotation = annotation
    }

    if let newAnnotationView = annotation as? PersonalAnnotation{
        newAnnotation?.image = UIImage(named: newAnnotationView.imagePin)
    }
    return newAnnotation
}
}
```

Clase de anotación en el mapa, para poder asignarle el nombre de la imagen que va a tener el pin dentro del mapa:

```
import Foundation
import MapKit
class PersonalAnnotation : MKPointAnnotation {

    var imagePin : String!
}
```

El arreglo de los pines funciona de la siguiente manera:

```
import Foundation
import UIKit

struct pines {

    var title: String
    var subtitle: String
    var latitud: Double
    var longitud: Double
    var imagen: String

}

var notaciones : [pines] = [
    pines(title: "G", subtitle: "Laboratorio de la División de Ciencias Básicas", latitud: 19.3269, longitud: -99.1828,
imagen: "lab"),
    pines(title: "H", subtitle: "Laboratorio de la División de Ciencias Básicas", latitud: 19.3270, longitud: -99.1826,
imagen: "lab"),
    pines(title: "I", subtitle: "Torre de salones", latitud: 19.3264, longitud: -99.1826, imagen: "salon"),
    pines(title: "J", subtitle: "Ala poniente, Coordinaciones académicas de la DCB", latitud: 19.3263, longitud: -
99.1832, imagen: "salon"),
    pines(title: "K", subtitle: "Centro de Docencia Gilberto Borja Navarrete", latitud: 19.3260, longitud: -99.18285,
imagen: "salon"),
    pines(title: "L", subtitle: "Biblioteca Enrique Rivero Borrell", latitud: 19.3256, longitud: -99.1826, imagen:
"biblioteca"),
    pines(title: "M", subtitle: "Auditorio Sotero Prieto, COPADI(becas, tutorías)", latitud: 19.3260, longitud: -
99.1824, imagen: "auditorio"),
]
```

Aquí se encuentra el código de las celdas de la tabla de notificaciones:

```
import UIKit

class CustomTableViewCell: UITableViewCell {

    @IBOutlet weak var labelCell: UILabel!

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
        super.setSelected(selected, animated: animated)

        // Configure the view for the selected state
    }

}
```

Y finalmente, aquí se encuentra el código utilizado para la barra de notificaciones:

```
import UIKit
import Firebase

class NotificacionesViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {

    var noticias = [String]()

    @IBOutlet weak var tableVista: UITableView!
    override func viewDidLoad() {
        super.viewDidLoad()
        tableVista.delegate = self
        tableVista.dataSource = self
        tableVista.estimatedRowHeight = 200

        Firestore.firestore().collection("Noticias").addSnapshotListener{(snapshot, error) in
            if let error = error{
                debugPrint(error)
            }else{
                self.noticias.removeAll()
                for document in (snapshot?.documents)!{
                    //print(document.data())
                    let data = document.data()
                    if (data["aviso"] != nil){
                        let resiveNotificacion = data["aviso"] as! String

                        self.noticias.append(resiveNotificacion)
                    }
                }

                print(self.noticias)

                self.tableVista.reloadData()
            }
        }

    }

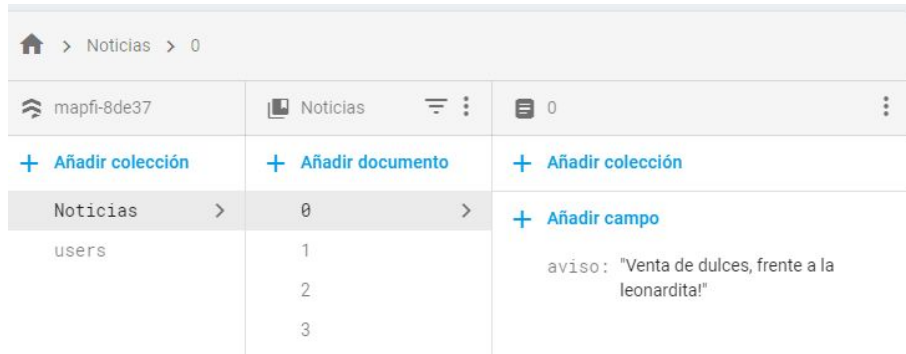
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return noticias.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        var cell = tableVista.dequeueReusableCell(withIdentifier: "cell", for: indexPath) as! CustomTableViewCell

        cell.labelCell.text = noticias[indexPath.row]
        return cell
    }
}
```


Base de datos

Colecciones: nuestra base de datos cuenta con dos colecciones, una de noticias, que se encarga de los avisos que se muestran en la aplicación, y otro de usuarios, que se encarga de almacenar los usuarios.



🏠 > Noticias > 0		
mapfi-8de37	Noticias	0
+ Añadir colección	+ Añadir documento	+ Añadir colección
Noticias >	0 >	+ Añadir campo
users	1	aviso: "Venta de dulces, frente a la leonardita!"
	2	
	3	

Bibliografía

- Apple Education (2017) Everyone Can Code App Development with Swift. Recuperado el 1 de diciembre de 2018, de <https://books.apple.com/mx/book/app-development-with-swift/id1219117996>
- Introducción al desarrollo de apps con Swift
[Apple Education](#)
- SWIFT: DESARROLLO DE APLICACIONES IOS
ENRIQUE BLASCO BLANQUER