Universidad Nacional Autónoma de México

Facultad de Ingeniería

Gómez Cid José Antonio
Román Cuevas Quetzali Andrea

Modelos de Programación Orientada a Objetos

Red - Up

Documentación

Contexto: En conjunto y basados en nuestras experiencias previas estando en una fila esperando formados por largos periodos de tiempo, desarrollamos la idea de “Red - Up”, gracias a que la población de la Ciudad de México es muy grande, esto deriva en la insuficiencia de abastecimiento de algunos servicios.

Introducción: “Red -Up” es una aplicación que proporciona servicios. Para el nombre tomamos como referencia los elementos sobre los cuales la aplicación trabaja, en este caso las colas o filas en las que las personas se forman, escogimos la conjugación del verbo “formar” (formarse) traducido primeramente al inglés (queue up), de igual manera lo relacionamos con el concepto de programación “colas” (queues en inglés) que vimos en semestres previos, posteriormente traducimos la palabra “queue” en croata (red).

Inspiración: Nos basamos en algunas aplicaciones existentes que proveen servicios similares, en este caso tomamos como referencia: Uber con su familia de aplicaciones (Uber drivers, Uber y Uber Eats) de la cual obtuvimos el layout para la app, Rappi de la cual obtuvimos la idea principal de la aplicación,

específicamente de los “Rappi favores”, y por último Netflix que nos brindó la idea sobre el manejo de las vistas de nuestra aplicación.

Objetivo:

Proveer una oportunidad de trabajo mientras que el usuario invierte el tiempo que podría estar desperdiciando en una fila en otras actividades.

Conceptos a implementar y desarrollar:

- Uso del tiempo: La aplicación busca que el usuario invierta el tiempo que normalmente gastaría estando en una fila en otras actividades de su conveniencia.
- Oportunidades de trabajo: La aplicación provee un forma fácil y sencilla de obtener un ingreso.
- Servicio: La meta del servicio de la aplicación es generar una economía circular con el tiempo y el dinero.

Mercado Meta y Segmentación del Mercado: Tras un breve análisis se determinó que el mercado meta es la Clase Media y Media Alta de la Ciudad de México, puesto a que ellos, primeramente tienen acceso a un smartphone y los recursos necesarios para obtener el servicio.

CÓDIGO.

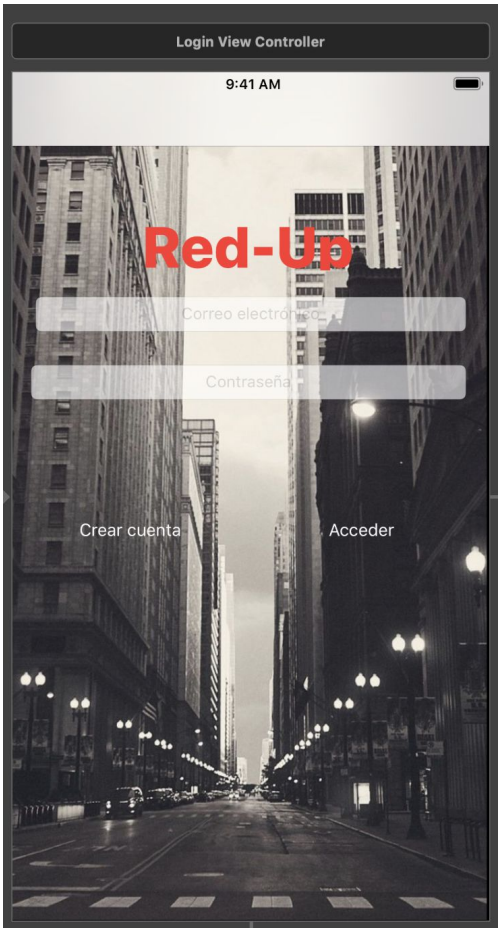
Ícono de la aplicación.

Pods:

- Firebase/Core: Pod principal de Firebase, el resto se derivan de este.
- Firebase/Database: Proporciona el servicio de base de datos en tiempo real.
- Firebase/Auth: Proporciona el servicio de autenticación del usuario.
- FirebaseUI/Storage: Proporciona y complementa a Firebase/Auth en el flujo de acceso y administración de cuentas.
- Firebase/Firestore: Proporciona el servicio de una base de datos flexible y escalable para la programar en servidores.
- Firebase/Storage: Proporciona el servicio de almacenamiento.

Se utiliza la herramienta proporcionada por Firebase debido a que al ser una aplicación de servicios es necesario tener siempre conocimiento sobre el proveedor del servicio y el cliente.

LOGIN VIEW CONTROLLER.



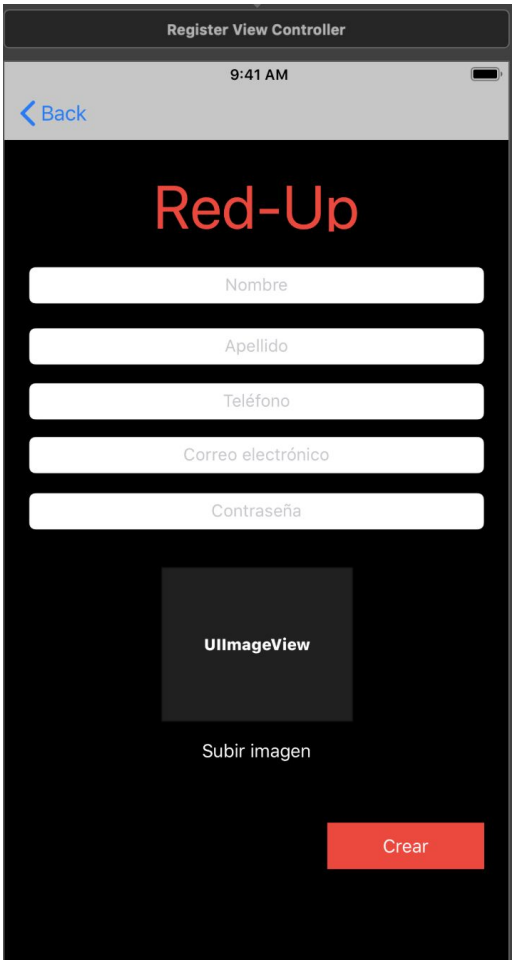
Es la vista principal, es en la que se accede a la aplicación en caso de tener una cuenta previamente creada, en caso contrario pasará a la vista correspondiente.

Elementos:

- Label
- Button
- Text Field

Cocoa Touch Class: Login View Controller.
Swift File: Usuario.swift

REGISTER VIEW CONTROLLER.



En caso de tener una cuenta previamente creada, en esta vista se preguntan la información necesaria para que usuario genere una cuenta.

Elementos:

- Label
- Button
- Text Field
- ImageView
- Navigation Controller

Cocoa Touch Class: Register View Controller.
Swift File: Usuario.swift

TAB BAR CONTROLLER.

Este es el Tab Bar Controller, controla y maneja el funcionamiento de las cuatro vistas representadas por los nombres: Perfil, Pedido, Mapa y Chat.

Elementos:

- Tab Bar Controller
- Items (4)

PEDIDO.

En esta vista, te será posible escoger el tipo de establecimiento buscas para obtener el servicio, para eso se muestran cuatro íconos presentando las opciones: Restaurante, Banco, Entretenimiento y Otros; posteriormente en la parte de abaja se indica el precio por minuto referido a cada opción.

Elementos:

- ImageView
- Label
- Button
- Text Field

Cocoa Touch Class: Pedido View Controller
Swift File: Orden.swift

PedidoViewController.swift

```
9  import UIKit
10
11 class PedidoViewController: UIViewController {
12     var costo = 0
13
◉      @IBOutlet weak var otro: UIButton!
◉      @IBOutlet weak var banco: UIButton!
◉      @IBOutlet weak var restaurante: UIButton!
◉      @IBOutlet weak var entretenimiento: UIButton!
18
19
```

Se importa la librería UIKit. Aparece la clase en la cual estamos trabajando. Se realiza la conexión de los **cuatro botones**, referentes a los **íconos**. Se inicializa la variable **costo** en **0**.

```
20
21      override func viewDidLoad() {
22          super.viewDidLoad()
23
24          // Do any additional setup after loading the view
25
26
27
28      }
29
◉       @IBOutlet weak var precio: UILabel!
31
◉       @IBAction func didSelectButton(_ sender: UIButton) {
33
34          var message = ""
35
36          print(sender.tag)
37          switch sender.tag {
38          case 0:
39              message = "Costo por minuto: $2"
40          case 1:
41              message = "Costo por minuto: $1"
42          case 2:
43              message = "Costo por minuto: $3"
44          case 3:
45              message = "Costo por minuto: $4"
46
47          default:
48              print("opción invalida")
49         }
50
51          precio.text = message
52
53      }
```

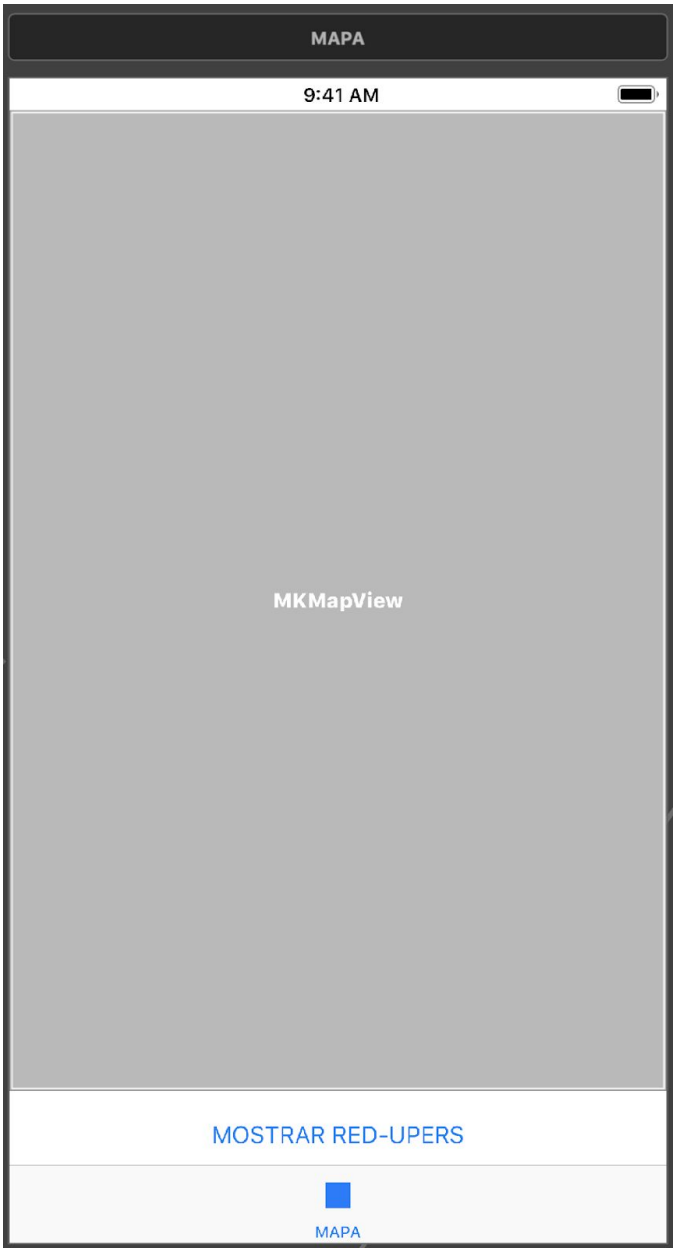
Se conecta el label de **precio**. Se hace una función anclada a los cuatro botones; se crea la variable **message** de tipo String. El **tag** es un identificador que se le asigna, en este caso a los botones y se usa en un **switch case**, se asigna a cada **tag** un mensaje que será el que la variable **message** obtenga según sea el caso, se finaliza el switch con un default. Se imprime el **tag**. Se asigna el el mensaje que obtuvo la variable **message** en el espacio del **Text Field**.

Orden.swift

```
8
9  import Foundation
10
11 var USERid = ""
12
```

Se importa la librería **Foundation**. Se crea la variable **USERid** de tipo String.

MAPA.



En esta vista te será posible ver en un mapa tu ubicación, así como de ver a los **Red - Upers** cerca de ti.

Elementos:

- MKMapView
- Button

Cocoa Touch Class: Mapa View Controller
Swift File: Ubicación.swift

MapaViewController.swift

```swift
import UIKit
import MapKit
import CoreLocation


class MapaViewController: UIViewController, CLLocationManagerDelegate {

    @IBOutlet weak var map: MKMapView!

    var manager = CLLocationManager()
    var longitud: CLLocationDegrees!
    var latitud: CLLocationDegrees!

    override func viewDidLoad() {
        super.viewDidLoad()

        map.showsUserLocation = true
        manager.delegate = self
        manager.requestWhenInUseAuthorization()

    }
```

Se importan las librerías **UIKit**, **MapKit**, **CoreLocation**, las últimas dos para el uso de mapas. Se muestran las clases con las que se trabajará. Se crean tres variables para el control de los mapas. En el **viewDidLoad** se muestran tres funciones, para mostrar el mapa al usuario, para que el mapa se controle así mismo y para que se lance una ventana preguntando al usuario si permite a la aplicación acceder a su ubicación.

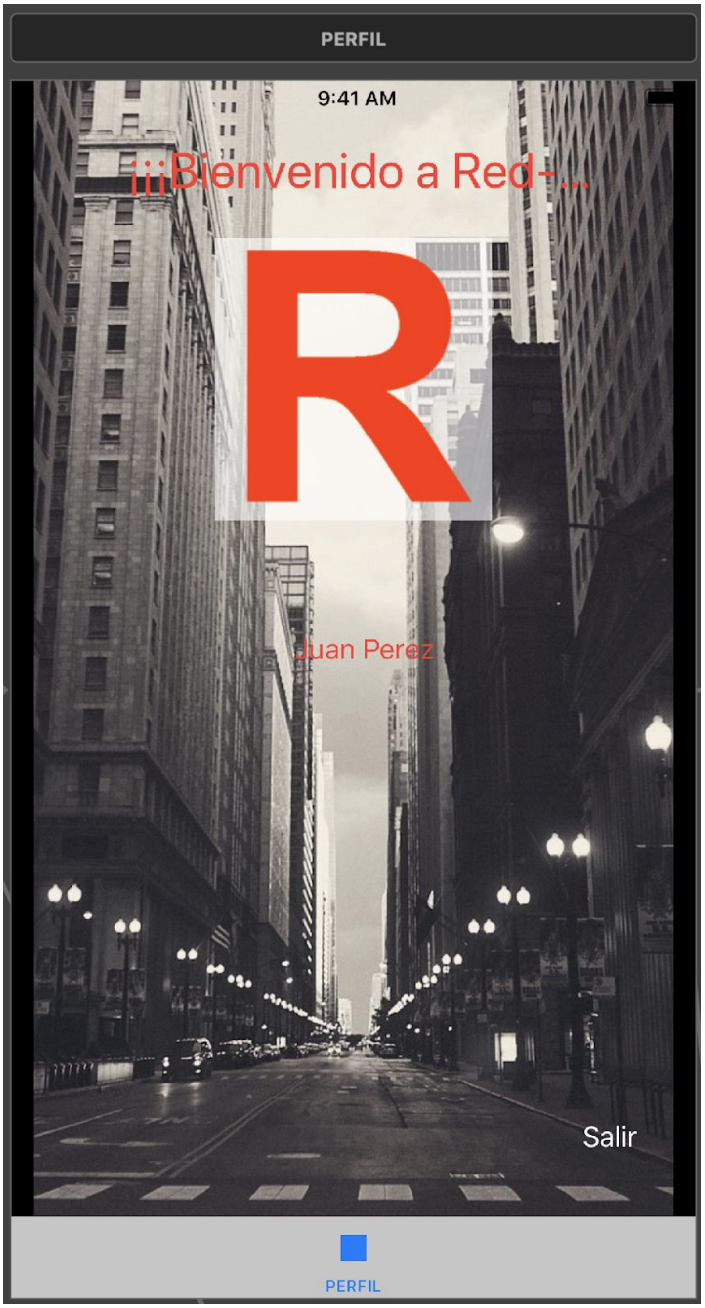
```swift
func locationManager( _ manager:CLLocationManager, didUpdateLocations location: [CLLocation]) {

    if let location = location.last{

        self.longitud = location.coordinate.longitude
        self.latitud = location.coordinate.latitude

    }
}
```

En esta función se establece la última ubicación del usuario.

```swift
func locationManager(_ manager: CLLocationManager, didChangeAuthorization status: CLAuthorizationStatus) {
    if status == .authorizedWhenInUse {
        manager.startUpdatingLocation()
    }
}

@IBAction func getLocation(_ sender: UIButton) {

    print(latitud, longitud)                                   2 ⚠ Expression implicitly coerced from 'Cl

    let localizacion = CLLocationCoordinate2D(latitude: latitud!, longitude: longitud!)

    let span = MKCoordinateSpan(latitudeDelta: 0.01, longitudeDelta: 0.01)

    let region = MKCoordinateRegion(center: localizacion, span: span)

    map.setRegion(region, animated: true)
    map.showsUserLocation = true
    map.mapType = .hybrid

    let anotacion = MKPointAnnotation()
    anotacion.title = "Red- Uper"
    anotacion.subtitle = "servicio"
    anotacion.coordinate = CLLocationCoordinate2D(latitude: 19.435477, longitude: -99.1364789)
    map.addAnnotation(anotacion)

    let anotacion2 = MKPointAnnotation()
    anotacion2.title = "Red- Uper"
    anotacion2.subtitle = "servicio"
    anotacion2.coordinate = CLLocationCoordinate2D(latitude: 19.435102, longitude: -99.138123)
    map.addAnnotation(anotacion2)

    let anotacion3 = MKPointAnnotation()
    anotacion3.title = "Red- Uper"
    anotacion3.subtitle = "servicio"
    anotacion3.coordinate = CLLocationCoordinate2D(latitude: 19.435744, longitude: -99.143957)
    map.addAnnotation(anotacion3)

    let anotacion4 = MKPointAnnotation()
    anotacion4.title = "Red- Uper"
    anotacion4.subtitle = "servicio"
    anotacion4.coordinate = CLLocationCoordinate2D(latitude: 19.434647, longitude: -99.139335)
    map.addAnnotation(anotacion4)


}
```

La función **locationManager** permite obtener la ubicación del usuario cuando éste lo permita. La función **getLocation** en donde se crea las variables **span**, **localización**, **región** para posteriormente crear la región en el mapa, la ubicación del usuario y el tipo de mapa que se muestra en la pantalla. Se ponen tres pines en el mapa.

PERFIL.



En esta vista el usuario podrá ver su foto y su nombre completo.

Elementos:

- ImageView
- Label
- Text Field
- Button

Cocoa Touch Class: Profile View Controller
Swift File: Usuario.swift

```swift
import UIKit
import FirebaseAuth
import Firebase

class ProfileViewController: UIViewController {



    @IBOutlet weak var nombrerUser: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
        Firestore.firestore().collection("users").addSnapshotListener{(snapshot, error) in
            if let error = error{
            debugPrint(error)
            }else{
                for document in (snapshot?.documents)!{
                    //print(document.data())
                    let data = document.data()
                    if (data["Nombre"] != nil){
                        let nombre = data["Nombre"] as! String
                        print(nombre)

                       if (data["Apellido"] != nil){
                        let apellido = data["Apellido"] as! String
                            print(apellido)
                        }

                        if (data["correo"] != nil){
                            let correo = data["correo"] as! String
                            print(correo)
                            if USERid == correo {
                                self.nombrerUser.text = nombre
                                 print(nombre)
                            }
                        }


                    }

                }
            }
    }
    }


        func logout(_ sender: Any) {

    }

    @IBAction func signOut(_ sender: Any) {
        try! Auth.auth().signOut()
        self.dismiss(animated: true, completion: nil)
    }
```

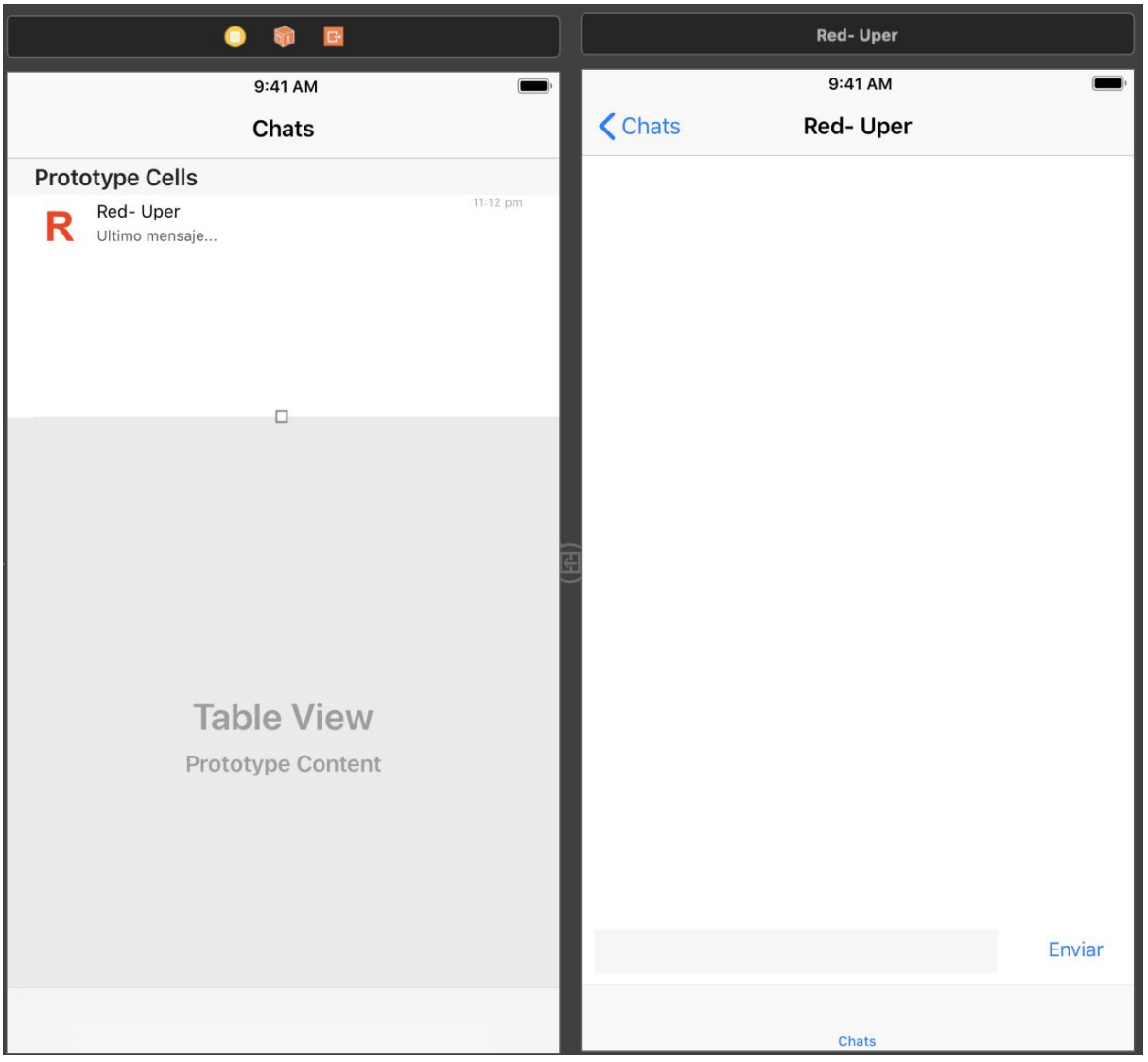
Se importan las librerías **UIKit, FirebaseAuth** y **Firebase.** Se crea la clase **ProfileViewController**. Se crea un label; se crea un documento y se se conecta con Firebase y Firestore. En el documento están los datos de nombre, apellido y teléfono.

Usuario.swift

```swift
import UIKit

class User{

    var id: String
    var nombre: String
    var apellido: String
    var telefono: Int

    init(id: String, nombre: String, apellido: String, telefono: Int){

        self.id = id
        self.nombre = nombre
        self.apellido = apellido
        self.telefono = telefono


    }
}
```

Se importa la librería **UIKit** y se crea la clase **User** la cual tiene cuatro variables y se inicializa la clase.

CHAT.



En esta vista se encuentra el chat con los **Red - Upers.**

Elementos:

- TableView
- TableViewCell
- Button

Cocoa Touch Class: ChatViewController

Swift File:

```swift
import UIKit

class ChatTableViewCell: UITableViewCell {

    @IBOutlet weak var profileImageVIew: UIImageView!
    @IBOutlet weak var nameOfPerson: UILabel!
    @IBOutlet weak var lastMessageLabel: UILabel!
    @IBOutlet weak var timeLabel: 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
    }

}
```

Se importa las librerías y se crea la clase del chat. Se insertan los ImageView y los label.

Chat Table View Controller.

```swift
import UIKit

class ChatsTableViewController: UITableViewController {

    let cellChat = "cellChat"

    override func viewDidLoad() {
        super.viewDidLoad()

        // Uncomment the following line to preserve selection between presentations
        // self.clearsSelectionOnViewWillAppear = false

        // Uncomment the following line to display an Edit button in the navigation bar for this view controller.
        // self.navigationItem.rightBarButtonItem = self.editButtonItem
    }

    // MARK: - Table view data source

    override func numberOfSections(in tableView: UITableView) -> Int {
        // #warning Incomplete implementation, return the number of sections
        return 1
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        // #warning Incomplete implementation, return the number of rows
        return 5
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: cellChat , for: indexPath) as! ChatTableViewCell

       cell.nameOfPerson.text = "Red- Uper\(indexPath.row)"
        cell.lastMessageLabel.text = "hola, mucho gusto \(indexPath.row)"
        cell.timeLabel.text = "11:11 am"

        return cell
    }

    override func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
        return 70
    }
```

Se inserta un Navigation Controller para manejar las vistas del chat.