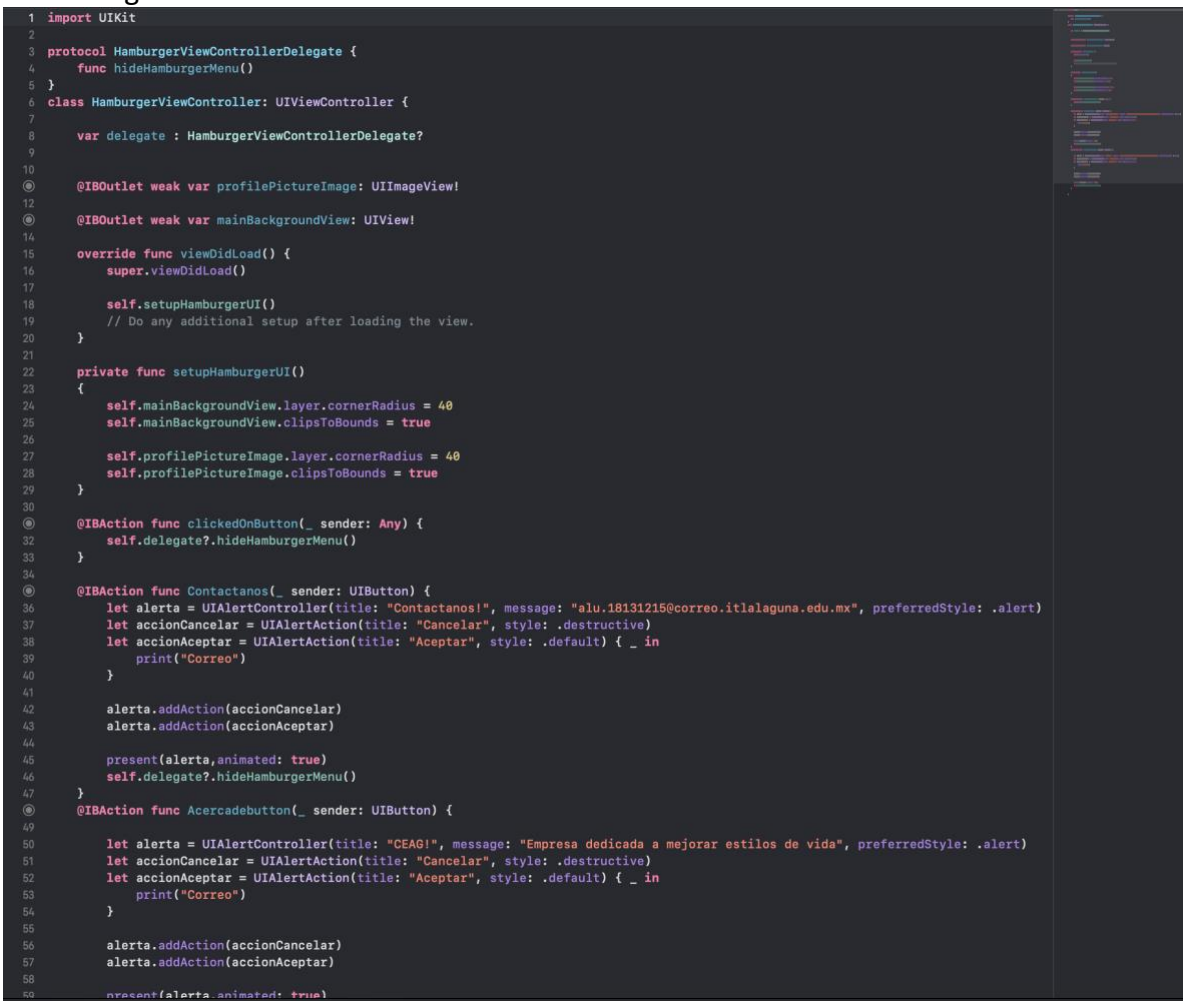
IMPLEMENTACION

Gestión de Proyectos de software

Instituto Tecnológico de la Laguna

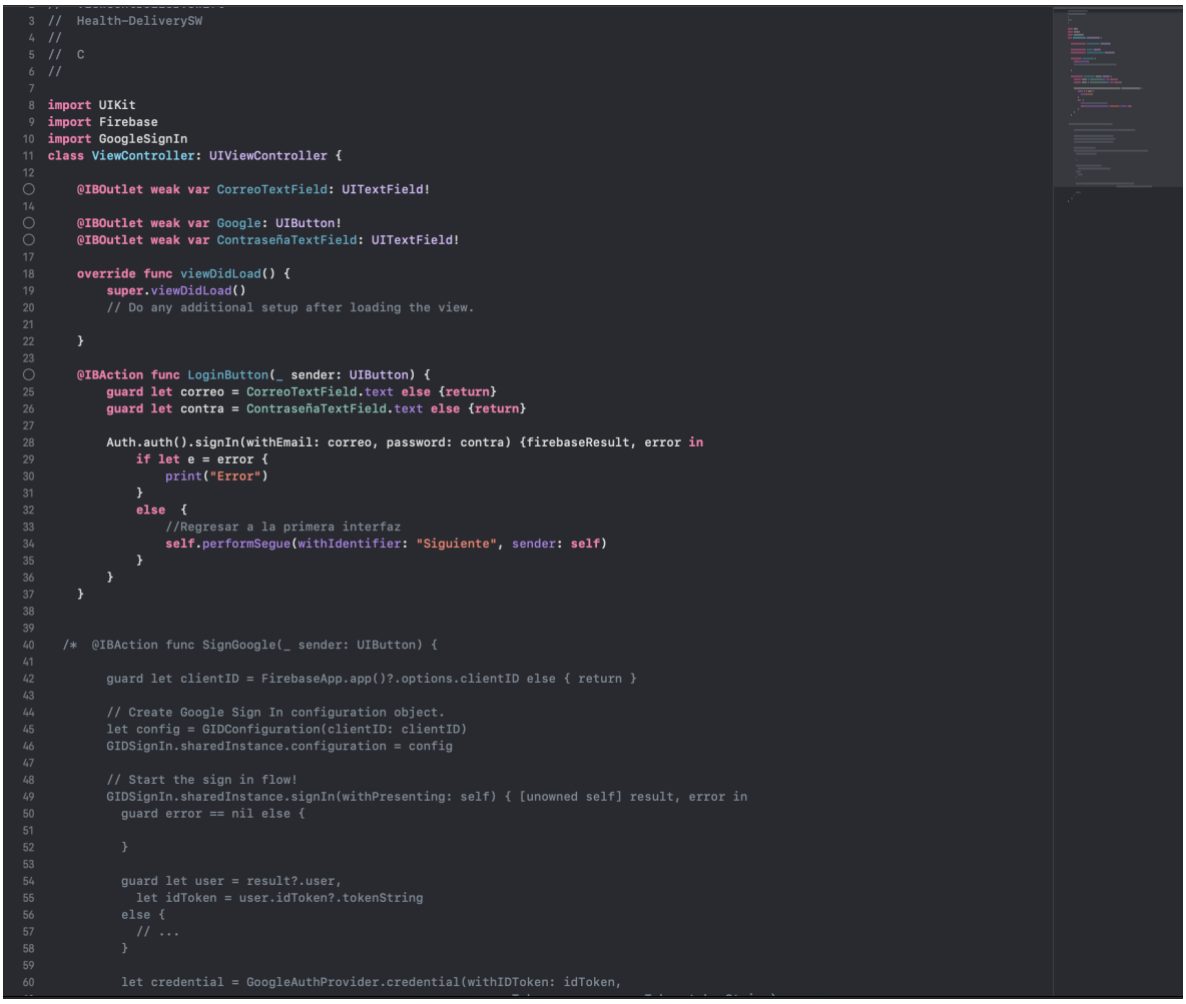
18131215	Gustavo Maximiliano Ambriz Zamarripa
18131395	Christian Emmanuel Escalera Cerda

Hambuerguer view controller



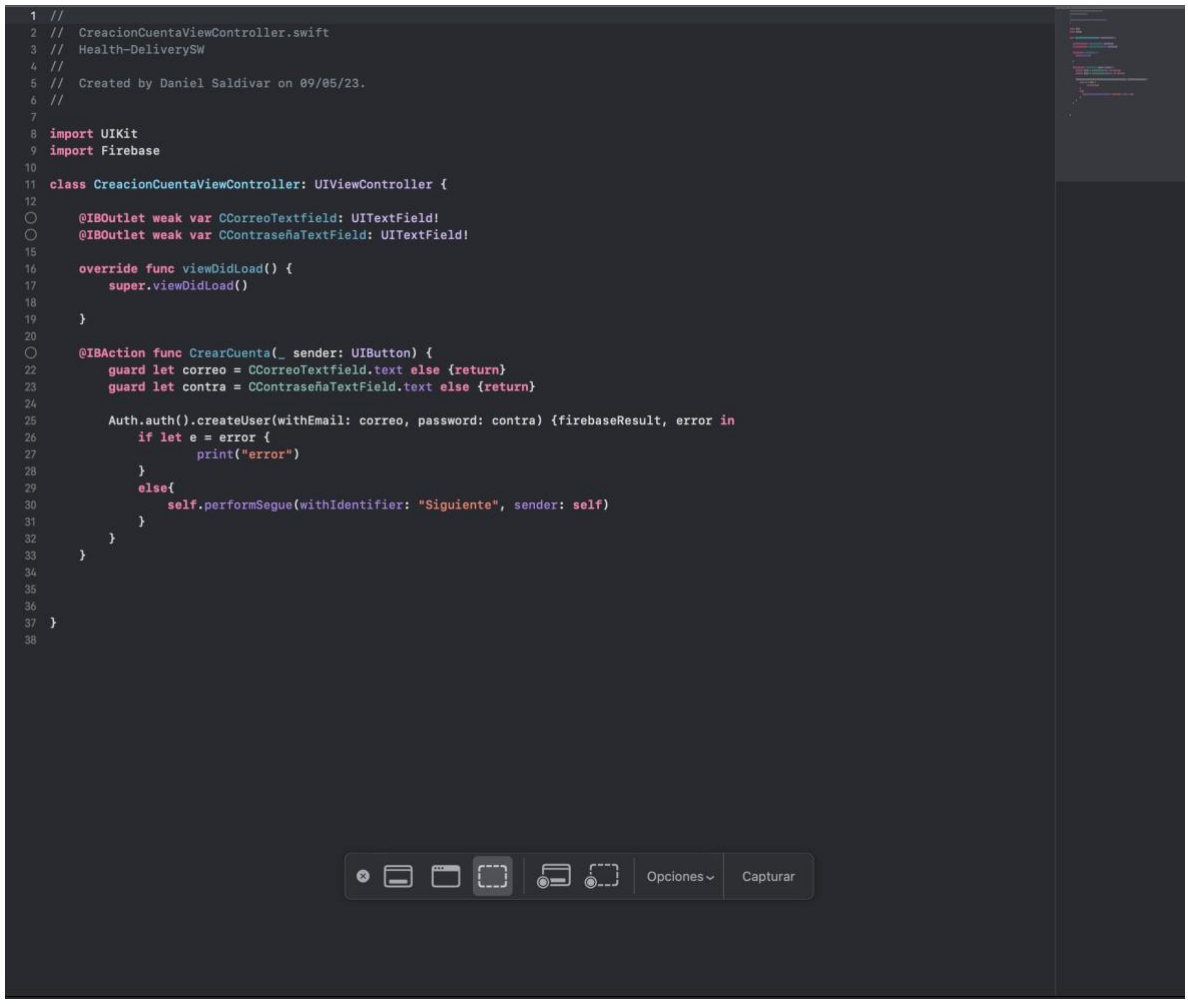
```swift
import UIKit

protocol HamburgerViewControllerDelegate {
    func hideHamburgerMenu()
}
class HamburgerViewController: UIViewController {

    var delegate : HamburgerViewControllerDelegate?


    @IBOutlet weak var profilePictureImage: UIImageView!

    @IBOutlet weak var mainBackgroundView: UIView!

    override func viewDidLoad() {
        super.viewDidLoad()

        self.setupHamburgerUI()
        // Do any additional setup after loading the view.
    }

    private func setupHamburgerUI()
    {
        self.mainBackgroundView.layer.cornerRadius = 40
        self.mainBackgroundView.clipsToBounds = true

        self.profilePictureImage.layer.cornerRadius = 40
        self.profilePictureImage.clipsToBounds = true
    }

    @IBAction func clickedOnButton(_ sender: Any) {
        self.delegate?.hideHamburgerMenu()
    }

    @IBAction func Contactanos(_ sender: UIButton) {
        let alerta = UIAlertController(title: "Contactanos!", message: "alu.18131215@correo.itlalaguna.edu.mx", preferredStyle: .alert)
        let accionCancelar = UIAlertAction(title: "Cancelar", style: .destructive)
        let accionAceptar = UIAlertAction(title: "Aceptar", style: .default) { _ in
            print("Correo")
        }

        alerta.addAction(accionCancelar)
        alerta.addAction(accionAceptar)

        present(alerta,animated: true)
        self.delegate?.hideHamburgerMenu()
    }
    @IBAction func Acercadebutton(_ sender: UIButton) {

        let alerta = UIAlertController(title: "CEAG!", message: "Empresa dedicada a mejorar estilos de vida", preferredStyle: .alert)
        let accionCancelar = UIAlertAction(title: "Cancelar", style: .destructive)
        let accionAceptar = UIAlertAction(title: "Aceptar", style: .default) { _ in
            print("Correo")
        }

        alerta.addAction(accionCancelar)
        alerta.addAction(accionAceptar)

        present(alerta,animated: true)
```

View controller



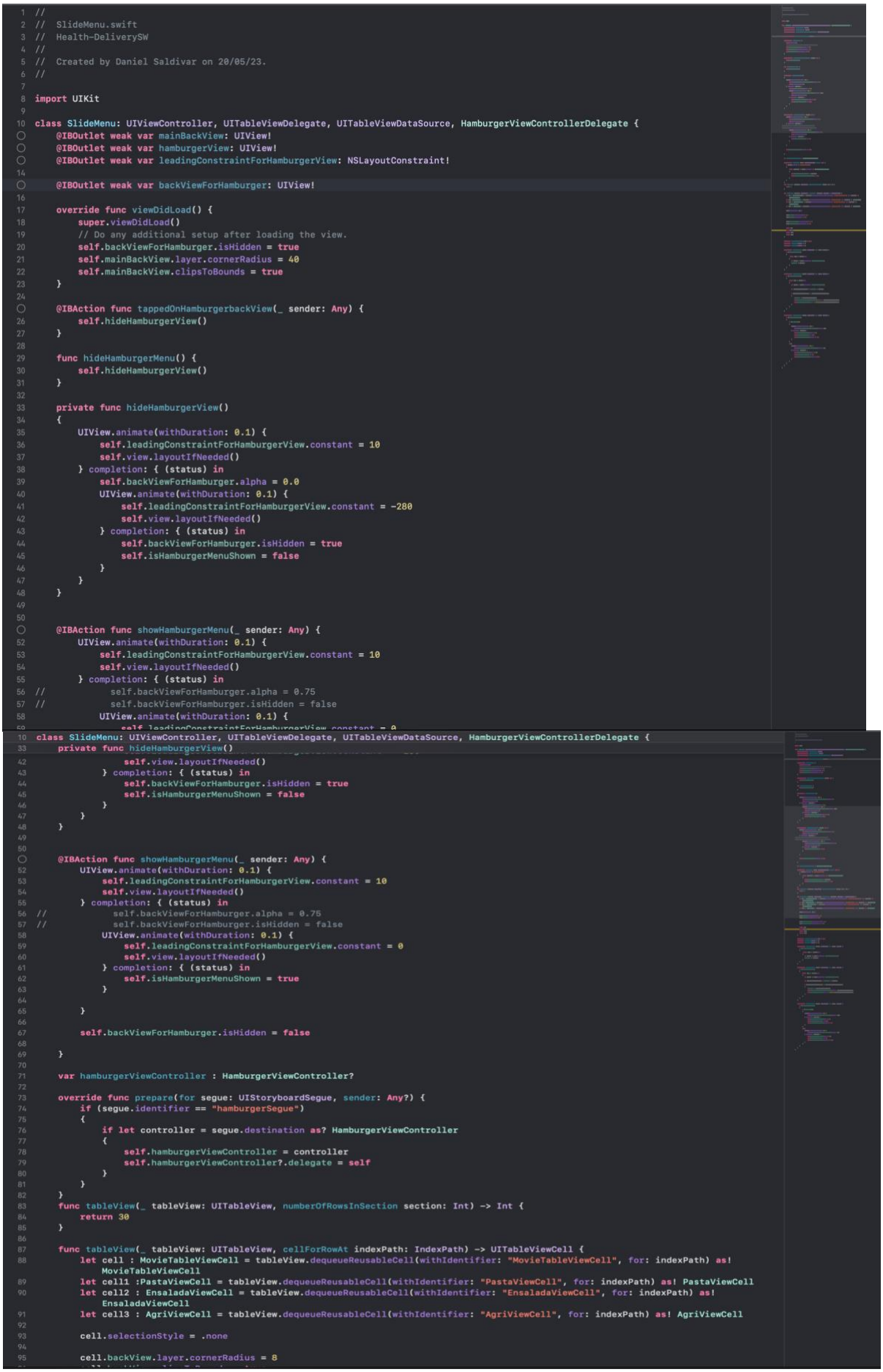
```swift
//  Health-DeliverySW
//
//  C
//

import UIKit
import Firebase
import GoogleSignIn
class ViewController: UIViewController {

    @IBOutlet weak var CorreoTextField: UITextField!

    @IBOutlet weak var Google: UIButton!
    @IBOutlet weak var ContraseñaTextField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

    }

    @IBAction func LoginButton(_ sender: UIButton) {
        guard let correo = CorreoTextField.text else {return}
        guard let contra = ContraseñaTextField.text else {return}

        Auth.auth().signIn(withEmail: correo, password: contra) {firebaseResult, error in
            if let e = error {
                print("Error")
            }
            else  {
                //Regresar a la primera interfaz
                self.performSegue(withIdentifier: "Siguiente", sender: self)
            }
        }
    }


  /*  @IBAction func SignGoogle(_ sender: UIButton) {

        guard let clientID = FirebaseApp.app()?.options.clientID else { return }

        // Create Google Sign In configuration object.
        let config = GIDConfiguration(clientID: clientID)
        GIDSignIn.sharedInstance.configuration = config

        // Start the sign in flow!
        GIDSignIn.sharedInstance.signIn(withPresenting: self) { [unowned self] result, error in
          guard error == nil else {

          }

          guard let user = result?.user,
            let idToken = user.idToken?.tokenString
          else {
            // ...
          }

          let credential = GoogleAuthProvider.credential(withIDToken: idToken,
```

Creacion cuenta view controller

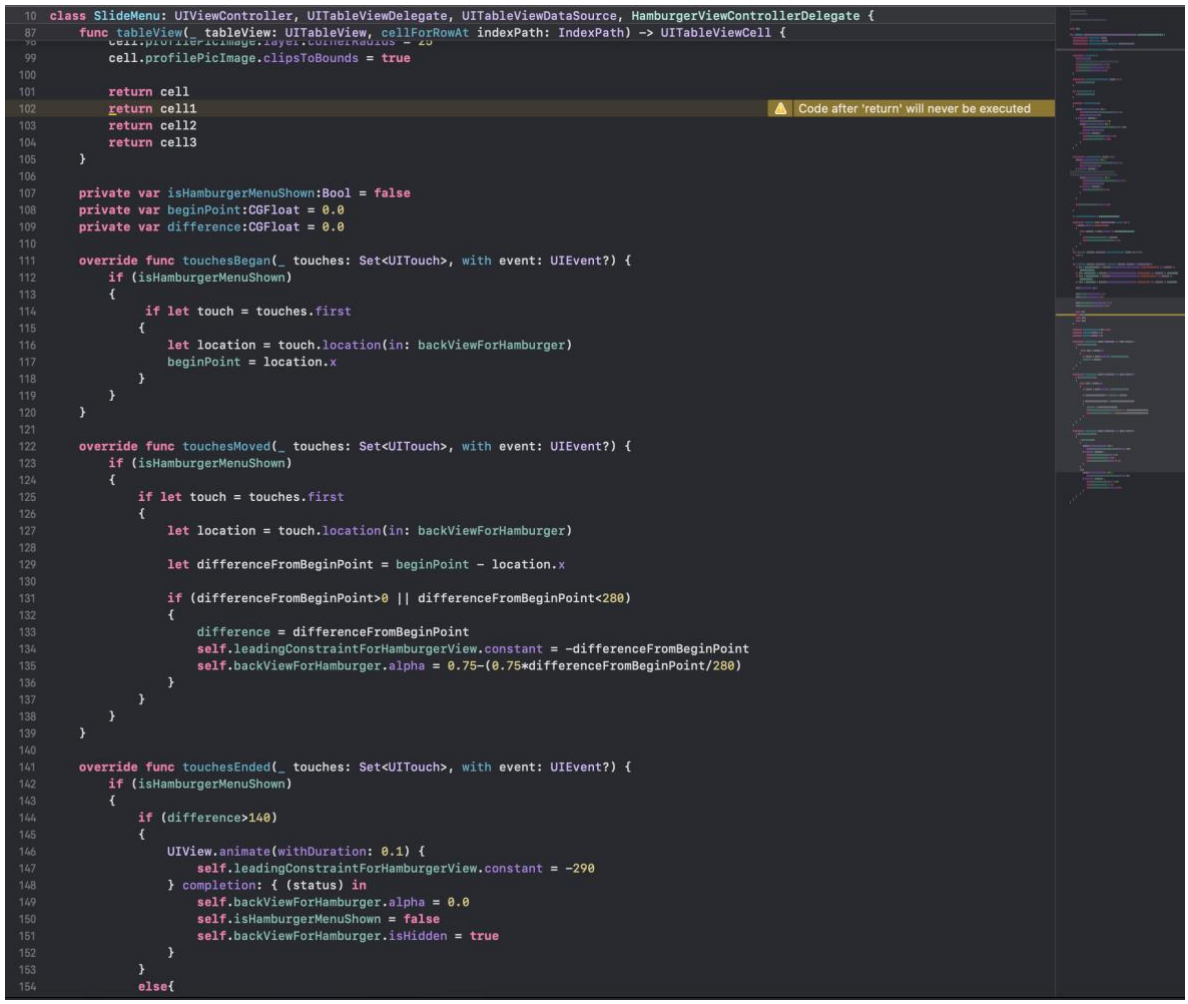


```swift
//
//  CreacionCuentaViewController.swift
//  Health-DeliverySW
//
//  Created by Daniel Saldivar on 09/05/23.
//

import UIKit
import Firebase

class CreacionCuentaViewController: UIViewController {

    @IBOutlet weak var CCorreoTextfield: UITextField!
    @IBOutlet weak var CContraseñaTextField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()

    }

    @IBAction func CrearCuenta(_ sender: UIButton) {
        guard let correo = CCorreoTextfield.text else {return}
        guard let contra = CContraseñaTextField.text else {return}

        Auth.auth().createUser(withEmail: correo, password: contra) {firebaseResult, error in
            if let e = error {
                    print("error")
            }
            else{
                self.performSegue(withIdentifier: "Siguiente", sender: self)
            }
        }
    }


}
```

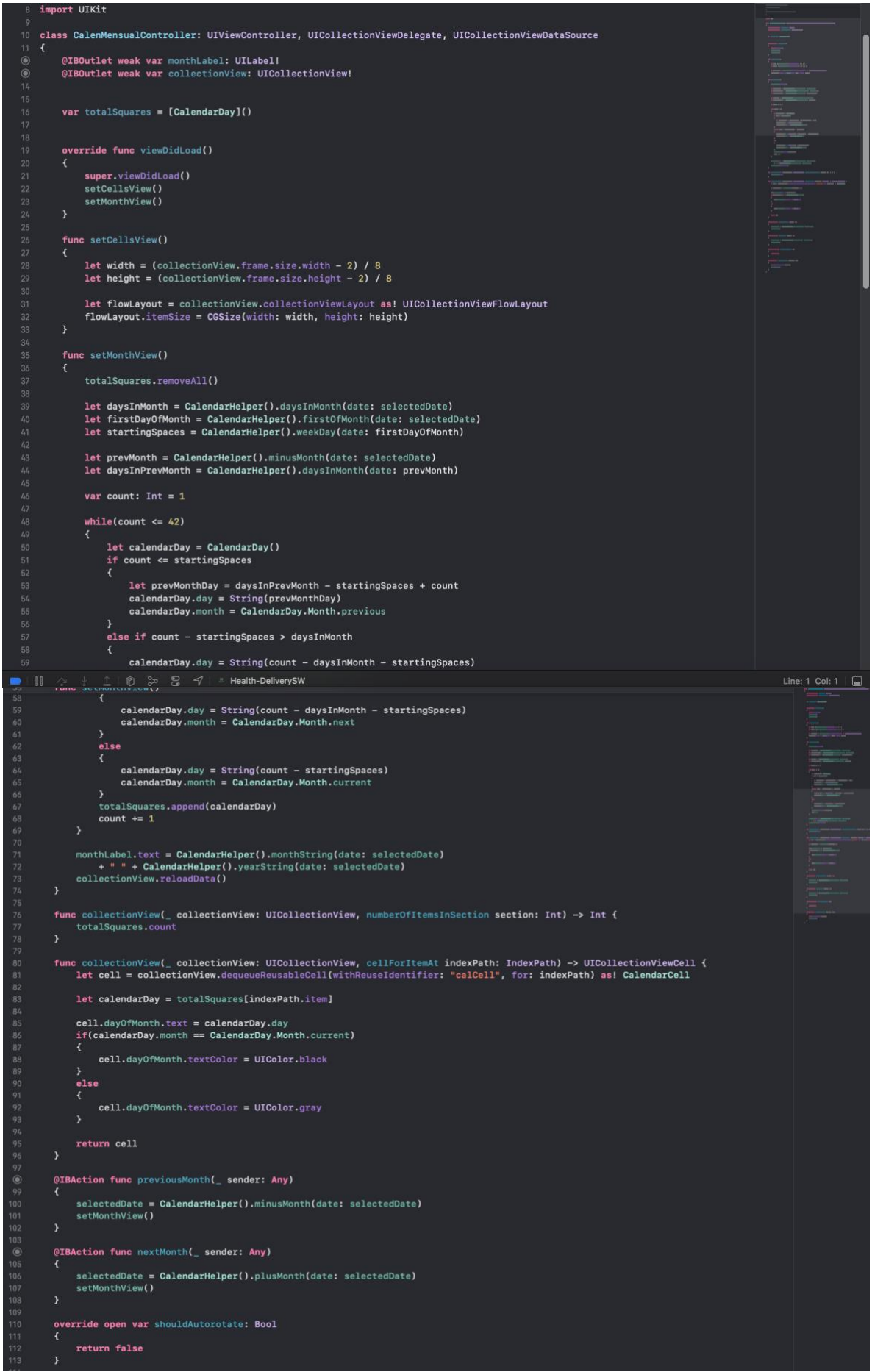
Slide menú view controller



```swift
//
//  SlideMenu.swift
//  Health-DeliverySW
//
//  Created by Daniel Saldivar on 20/05/23.
//

import UIKit

class SlideMenu: UIViewController, UITableViewDelegate, UITableViewDataSource, HamburgerViewControllerDelegate {
    @IBOutlet weak var mainBackView: UIView!
    @IBOutlet weak var hamburgerView: UIView!
    @IBOutlet weak var leadingConstraintForHamburgerView: NSLayoutConstraint!

    @IBOutlet weak var backViewForHamburger: UIView!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        self.backViewForHamburger.isHidden = true
        self.mainBackView.layer.cornerRadius = 40
        self.mainBackView.clipsToBounds = true
    }

    @IBAction func tappedOnHamburgerbackView(_ sender: Any) {
        self.hideHamburgerView()
    }

    func hideHamburgerMenu() {
        self.hideHamburgerView()
    }

    private func hideHamburgerView()
    {
        UIView.animate(withDuration: 0.1) {
            self.leadingConstraintForHamburgerView.constant = 10
            self.view.layoutIfNeeded()
        } completion: { (status) in
            self.backViewForHamburger.alpha = 0.0
            UIView.animate(withDuration: 0.1) {
                self.leadingConstraintForHamburgerView.constant = -280
                self.view.layoutIfNeeded()
            } completion: { (status) in
                self.backViewForHamburger.isHidden = true
                self.isHamburgerMenuShown = false
            }
        }
    }


    @IBAction func showHamburgerMenu(_ sender: Any) {
        UIView.animate(withDuration: 0.1) {
            self.leadingConstraintForHamburgerView.constant = 10
            self.view.layoutIfNeeded()
        } completion: { (status) in
//            self.backViewForHamburger.alpha = 0.75
//            self.backViewForHamburger.isHidden = false
            UIView.animate(withDuration: 0.1) {
                self.leadingConstraintForHamburgerView.constant = 0
                self.view.layoutIfNeeded()
            } completion: { (status) in
                self.backViewForHamburger.isHidden = true
                self.isHamburgerMenuShown = false
            }
        }
    }


    @IBAction func showHamburgerMenu(_ sender: Any) {
        UIView.animate(withDuration: 0.1) {
            self.leadingConstraintForHamburgerView.constant = 10
            self.view.layoutIfNeeded()
        } completion: { (status) in
//            self.backViewForHamburger.alpha = 0.75
//            self.backViewForHamburger.isHidden = false
            UIView.animate(withDuration: 0.1) {
                self.leadingConstraintForHamburgerView.constant = 0
                self.view.layoutIfNeeded()
            } completion: { (status) in
                self.isHamburgerMenuShown = true
            }

        }

        self.backViewForHamburger.isHidden = false

    }

    var hamburgerViewController : HamburgerViewController?

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if (segue.identifier == "hamburgerSegue")
        {
            if let controller = segue.destination as? HamburgerViewController
            {
                self.hamburgerViewController = controller
                self.hamburgerViewController?.delegate = self
            }
        }
    }
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return 30
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell : MovieTableViewCell = tableView.dequeueReusableCell(withIdentifier: "MovieTableViewCell", for: indexPath) as!
            MovieTableViewCell
        let cell1 :PastaViewCell = tableView.dequeueReusableCell(withIdentifier: "PastaViewCell", for: indexPath) as! PastaViewCell
        let cell2 : EnsaladaViewCell = tableView.dequeueReusableCell(withIdentifier: "EnsaladaViewCell", for: indexPath) as!
            EnsaladaViewCell
        let cell3 : AgriViewCell = tableView.dequeueReusableCell(withIdentifier: "AgriViewCell", for: indexPath) as! AgriViewCell

        cell.selectionStyle = .none

        cell.backView.layer.cornerRadius = 8
```


class SlideMenu: UIViewController, UITableViewDelegate, UITableViewDataSource, HamburgerViewControllerDelegate {
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
cell.profilePicImage.clipsToBounds = true
return cell
return cell1
Code after 'return' will never be executed
return cell2
return cell3
}
private var isHamburgerMenuShown:Bool = false
private var beginPoint:CGFloat = 0.0
private var difference:CGFloat = 0.0
override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
if (isHamburgerMenuShown)
{
if let touch = touches.first
{
let location = touch.location(in: backViewForHamburger)
beginPoint = location.x
}
}
}
override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
if (isHamburgerMenuShown)
{
if let touch = touches.first
{
let location = touch.location(in: backViewForHamburger)
let differenceFromBeginPoint = beginPoint - location.x
if (differenceFromBeginPoint>0 || differenceFromBeginPoint<280)
{
difference = differenceFromBeginPoint
self.leadingConstraintForHamburgerView.constant = -differenceFromBeginPoint
self.backViewForHamburger.alpha = 0.75-(0.75*differenceFromBeginPoint/280)
}
}
}
}
override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
if (isHamburgerMenuShown)
{
if (difference>140)
{
UIView.animate(withDuration: 0.1) {
self.leadingConstraintForHamburgerView.constant = -290
} completion: { (status) in
self.backViewForHamburger.alpha = 0.0
self.isHamburgerMenuShown = false
self.backViewForHamburger.isHidden = true
}
}
else{

CalenMensualController

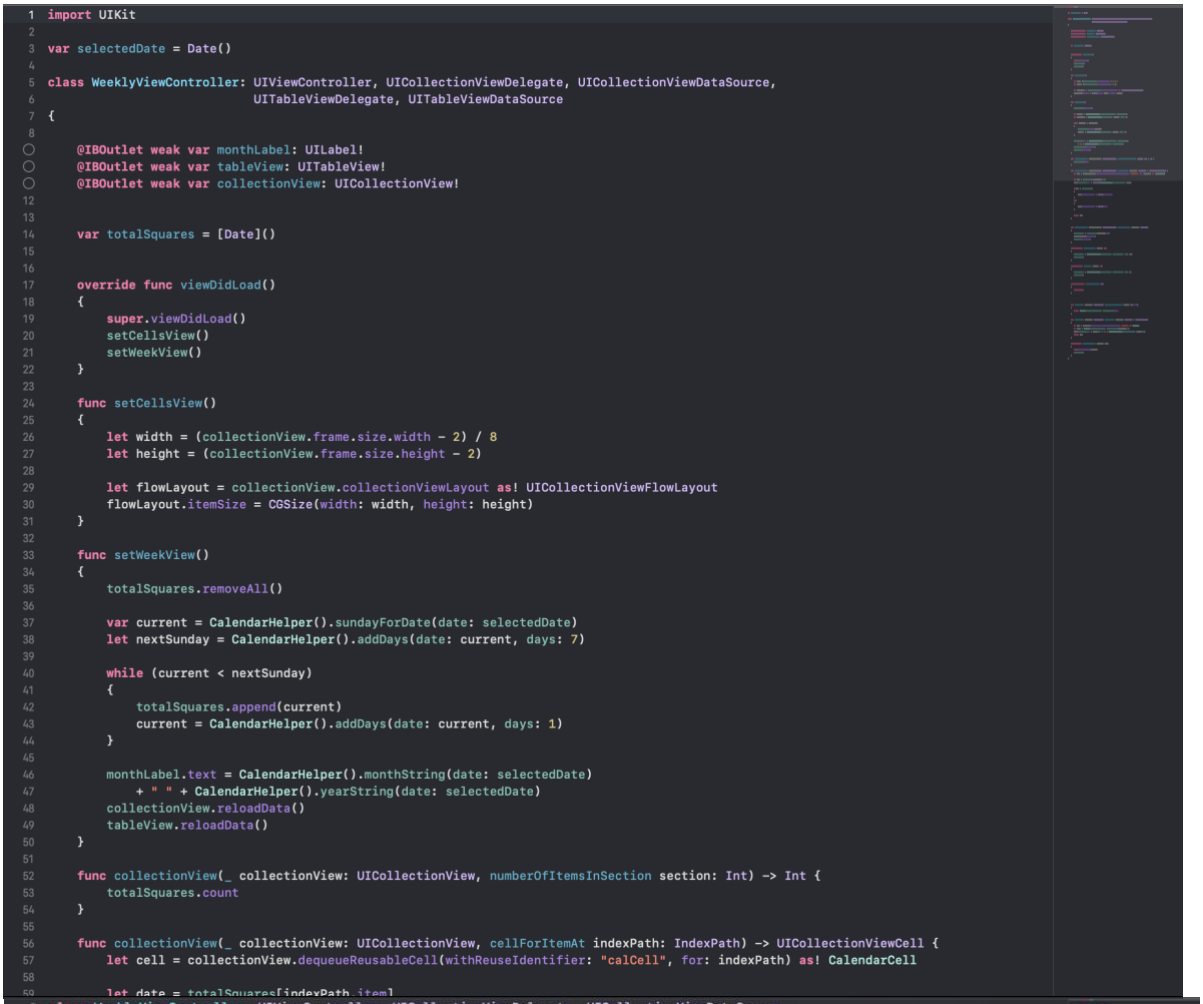


```swift
import UIKit

class CalenMensualController: UIViewController, UICollectionViewDelegate, UICollectionViewDataSource
{
    @IBOutlet weak var monthLabel: UILabel!
    @IBOutlet weak var collectionView: UICollectionView!


    var totalSquares = [CalendarDay]()


    override func viewDidLoad()
    {
        super.viewDidLoad()
        setCellsView()
        setMonthView()
    }

    func setCellsView()
    {
        let width = (collectionView.frame.size.width - 2) / 8
        let height = (collectionView.frame.size.height - 2) / 8

        let flowLayout = collectionView.collectionViewLayout as! UICollectionViewFlowLayout
        flowLayout.itemSize = CGSize(width: width, height: height)
    }

    func setMonthView()
    {
        totalSquares.removeAll()

        let daysInMonth = CalendarHelper().daysInMonth(date: selectedDate)
        let firstDayOfMonth = CalendarHelper().firstOfMonth(date: selectedDate)
        let startingSpaces = CalendarHelper().weekDay(date: firstDayOfMonth)

        let prevMonth = CalendarHelper().minusMonth(date: selectedDate)
        let daysInPrevMonth = CalendarHelper().daysInMonth(date: prevMonth)

        var count: Int = 1

        while(count <= 42)
        {
            let calendarDay = CalendarDay()
            if count <= startingSpaces
            {
                let prevMonthDay = daysInPrevMonth - startingSpaces + count
                calendarDay.day = String(prevMonthDay)
                calendarDay.month = CalendarDay.Month.previous
            }
            else if count - startingSpaces > daysInMonth
            {
                calendarDay.day = String(count - daysInMonth - startingSpaces)
```

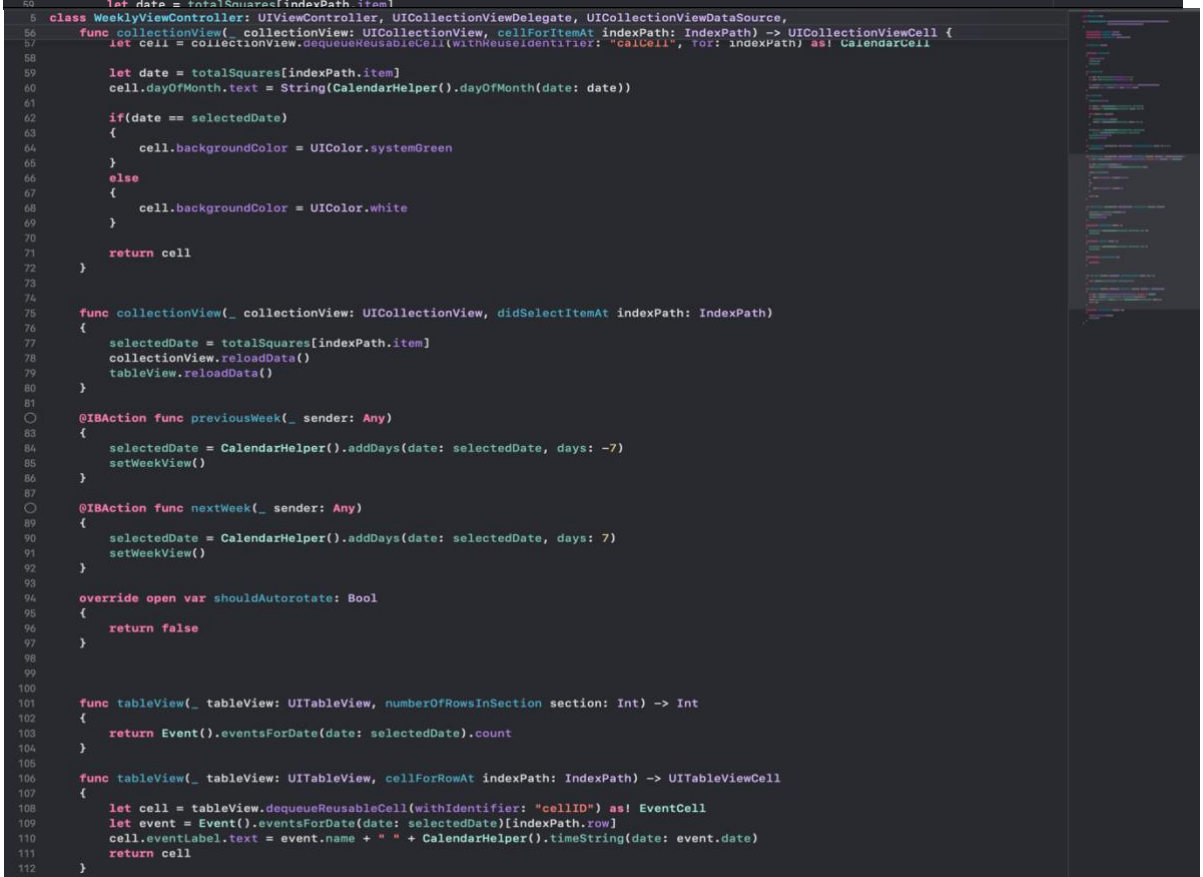
Health-DeliverySW — Line: 1 Col: 1

```swift
                calendarDay.day = String(count - daysInMonth - startingSpaces)
                calendarDay.month = CalendarDay.Month.next
            }
            else
            {
                calendarDay.day = String(count - startingSpaces)
                calendarDay.month = CalendarDay.Month.current
            }
            totalSquares.append(calendarDay)
            count += 1
        }

        monthLabel.text = CalendarHelper().monthString(date: selectedDate)
            + " " + CalendarHelper().yearString(date: selectedDate)
        collectionView.reloadData()
    }

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        totalSquares.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "calCell", for: indexPath) as! CalendarCell

        let calendarDay = totalSquares[indexPath.item]

        cell.dayOfMonth.text = calendarDay.day
        if(calendarDay.month == CalendarDay.Month.current)
        {
            cell.dayOfMonth.textColor = UIColor.black
        }
        else
        {
            cell.dayOfMonth.textColor = UIColor.gray
        }

        return cell
    }

    @IBAction func previousMonth(_ sender: Any)
    {
        selectedDate = CalendarHelper().minusMonth(date: selectedDate)
        setMonthView()
    }

    @IBAction func nextMonth(_ sender: Any)
    {
        selectedDate = CalendarHelper().plusMonth(date: selectedDate)
        setMonthView()
    }

    override open var shouldAutorotate: Bool
    {
        return false
    }
```

Calen Semanal



```swift
import UIKit

var selectedDate = Date()

class WeeklyViewController: UIViewController, UICollectionViewDelegate, UICollectionViewDataSource,
                            UITableViewDelegate, UITableViewDataSource
{

    @IBOutlet weak var monthLabel: UILabel!
    @IBOutlet weak var tableView: UITableView!
    @IBOutlet weak var collectionView: UICollectionView!


    var totalSquares = [Date]()


    override func viewDidLoad()
    {
        super.viewDidLoad()
        setCellsView()
        setWeekView()
    }

    func setCellsView()
    {
        let width = (collectionView.frame.size.width - 2) / 8
        let height = (collectionView.frame.size.height - 2)

        let flowLayout = collectionView.collectionViewLayout as! UICollectionViewFlowLayout
        flowLayout.itemSize = CGSize(width: width, height: height)
    }

    func setWeekView()
    {
        totalSquares.removeAll()

        var current = CalendarHelper().sundayForDate(date: selectedDate)
        let nextSunday = CalendarHelper().addDays(date: current, days: 7)

        while (current < nextSunday)
        {
            totalSquares.append(current)
            current = CalendarHelper().addDays(date: current, days: 1)
        }

        monthLabel.text = CalendarHelper().monthString(date: selectedDate)
            + " " + CalendarHelper().yearString(date: selectedDate)
        collectionView.reloadData()
        tableView.reloadData()
    }

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        totalSquares.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "calCell", for: indexPath) as! CalendarCell

        let date = totalSquares[indexPath.item]
```



```swift
class WeeklyViewController: UIViewController, UICollectionViewDelegate, UICollectionViewDataSource,
    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "calCell", for: indexPath) as! CalendarCell

        let date = totalSquares[indexPath.item]
        cell.dayOfMonth.text = String(CalendarHelper().dayOfMonth(date: date))

        if(date == selectedDate)
        {
            cell.backgroundColor = UIColor.systemGreen
        }
        else
        {
            cell.backgroundColor = UIColor.white
        }

        return cell
    }


    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath)
    {
        selectedDate = totalSquares[indexPath.item]
        collectionView.reloadData()
        tableView.reloadData()
    }

    @IBAction func previousWeek(_ sender: Any)
    {
        selectedDate = CalendarHelper().addDays(date: selectedDate, days: -7)
        setWeekView()
    }

    @IBAction func nextWeek(_ sender: Any)
    {
        selectedDate = CalendarHelper().addDays(date: selectedDate, days: 7)
        setWeekView()
    }

    override open var shouldAutorotate: Bool
    {
        return false
    }



    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
    {
        return Event().eventsForDate(date: selectedDate).count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
    {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cellID") as! EventCell
        let event = Event().eventsForDate(date: selectedDate)[indexPath.row]
        cell.eventLabel.text = event.name + " " + CalendarHelper().timeString(date: event.date)
        return cell
    }
```

DailyEvent

```swift
import UIKit


class DailyViewController: UIViewController, UITableViewDelegate, UITableViewDataSource
{
    @IBOutlet weak var hourTableView: UITableView!
    @IBOutlet weak var dayOfWeekLabel: UILabel!
    @IBOutlet weak var dayLabel: UILabel!

    var hours = [Int]()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        initTime()
        setDayView()
    }

    func initTime()
    {
        for hour in 0...23
        {
            hours.append(hour)
        }
    }

    func setDayView()
    {
        dayLabel.text = CalendarHelper().monthDayString(date: selectedDate)
        dayOfWeekLabel.text = CalendarHelper().weekDayAsString(date: selectedDate)
        hourTableView.reloadData()
    }

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

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cellDailyID") as! DailyCell

        let hour  = hours[indexPath.row]
        cell.time.text = formatHour(hour: hour)

        let events = Event().eventsForDateAndTime(date: selectedDate, hour: hour)
        setEvents(cell, events)

        return cell
    }

    func setEvents(_ cell: DailyCell, _ events: [Event])
    {
        hideAll(cell)
        switch events.count
        {
        case 1:
            setEvent1(cell, events[0])
        case 2:
            setEvent1(cell, events[0])
            setEvent2(cell, events[1])
```

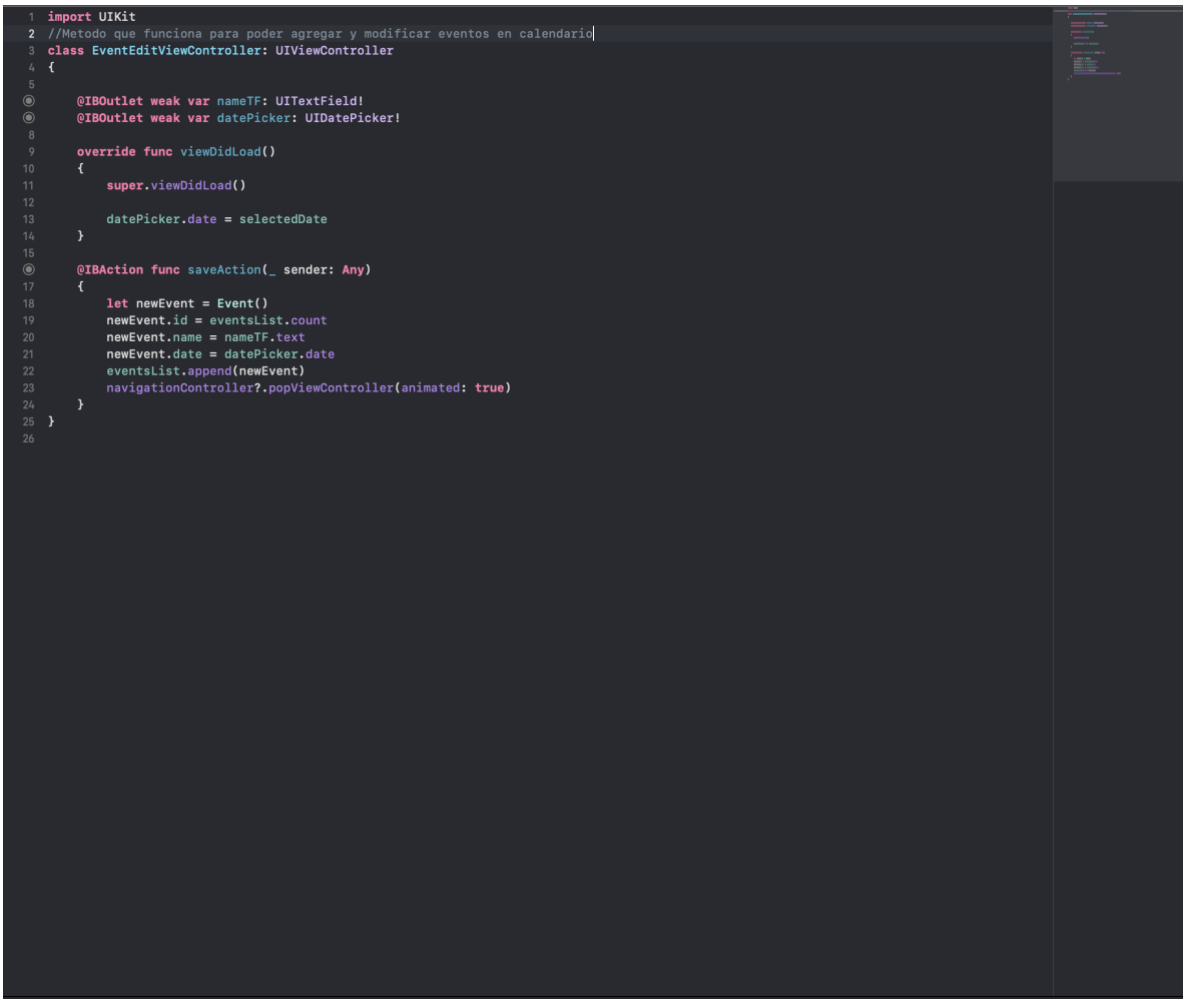
```swift
class DailyViewController: UIViewController, UITableViewDelegate, UITableViewDataSource
    func setEvents(_ cell: DailyCell, _ events: [Event])
        case 3:
            setEvent1(cell, events[0])
            setEvent2(cell, events[1])
            setEvent3(cell, events[2])

        case let count where count > 3:
            setEvent1(cell, events[0])
            setEvent2(cell, events[1])
            setMoreEvents(cell, events.count - 2)
        default:
            break
        }
    }


    func setMoreEvents(_ cell: DailyCell, _ count: Int)
    {
        cell.event3.isHidden = false
        cell.event3.text = String(count) + " More Events"
    }

    func setEvent1(_ cell: DailyCell, _ event: Event)
    {
        cell.event1.isHidden = false
        cell.event1.text = event.name
    }

    func setEvent2(_ cell: DailyCell, _ event: Event)
    {
        cell.event2.isHidden = false
        cell.event2.text = event.name
    }

    func setEvent3(_ cell: DailyCell, _ event: Event)
    {
        cell.event3.isHidden = false
        cell.event3.text = event.name
    }

    func hideAll(_ cell: DailyCell)
    {
        cell.event1.isHidden = true
        cell.event2.isHidden = true
        cell.event3.isHidden = true
    }


    func formatHour(hour: Int) -> String
    {
        return String(format: "%02d:%02d", hour, 0)
    }

    @IBAction func nextDayAction(_ sender: Any)
    {
        selectedDate = CalendarHelper().addDays(date: selectedDate, days: 1)
        setDayView()
    }
```

CalendarHelper

```swift
import Foundation
import UIKit

class CalendarHelper
{
    let calendar = Calendar.current

    func plusMonth(date: Date) -> Date
    {
        return calendar.date(byAdding: .month, value: 1, to: date)!
    }

    func minusMonth(date: Date) -> Date
    {
        return calendar.date(byAdding: .month, value: -1, to: date)!
    }

    func monthString(date: Date) -> String
    {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "LLLL"
        return dateFormatter.string(from: date)
    }

    func monthDayString(date: Date) -> String
    {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "LLLL dd"
        return dateFormatter.string(from: date)
    }

    func yearString(date: Date) -> String
    {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "yyyy"
        return dateFormatter.string(from: date)
    }

    func timeString(date: Date) -> String
    {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "HH:mm"
        return dateFormatter.string(from: date)
    }

    func daysInMonth(date: Date) -> Int
    {
        let range = calendar.range(of: .day, in: .month, for: date)!
        return range.count
    }

    func dayOfMonth(date: Date) -> Int
    {
        let components = calendar.dateComponents([.day], from: date)
        return components.day!
    }

    func hourFromDate(date: Date) -> Int
```

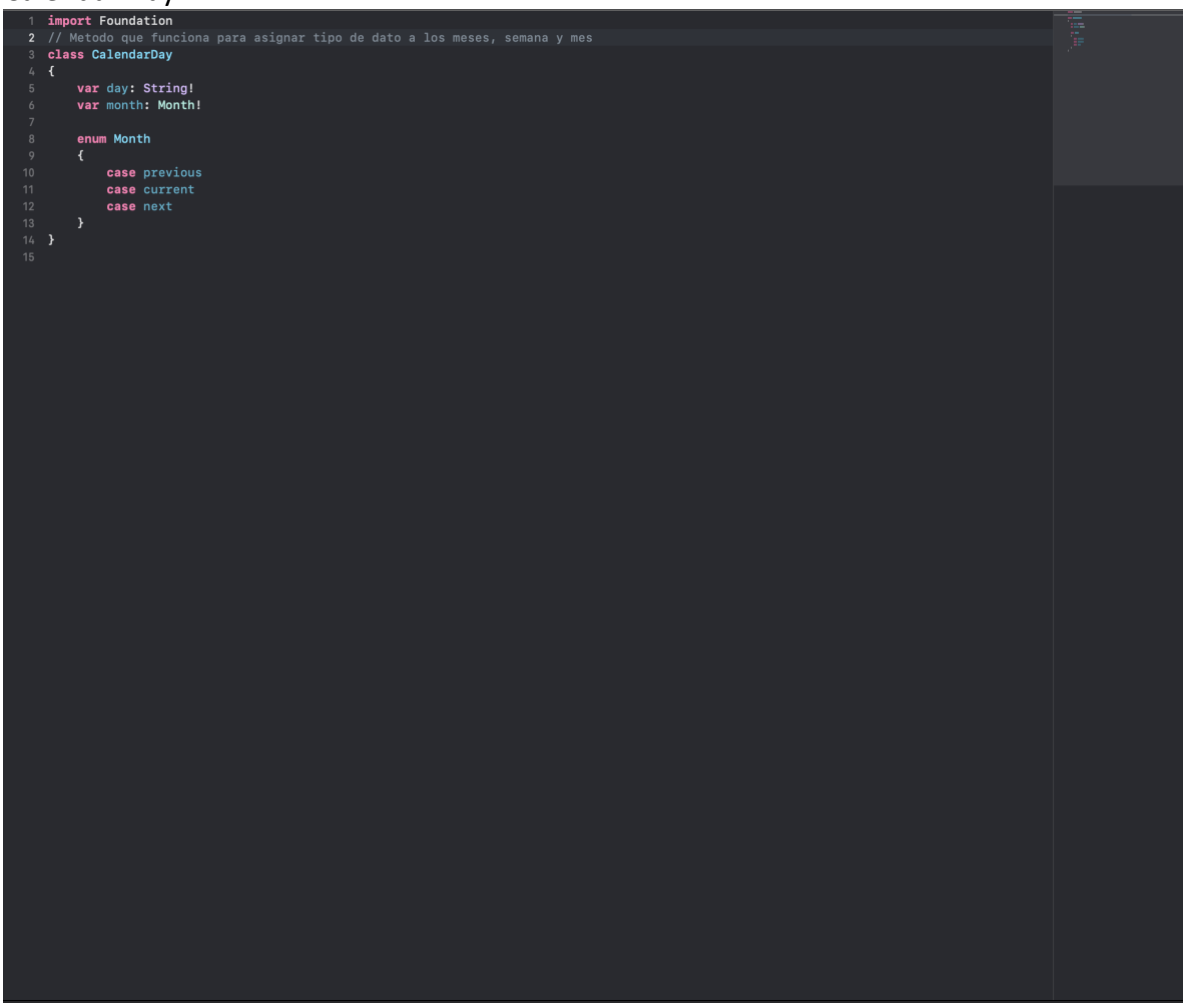
```swift
class CalendarHelper
    func dayOfMonth(date: Date) -> Int
    {
        let components = calendar.dateComponents([.day], from: date)
        return components.day!
    }

    func hourFromDate(date: Date) -> Int
    {
        let components = calendar.dateComponents([.hour], from: date)
        return components.hour!
    }

    func firstOfMonth(date: Date) -> Date
    {
        let components = calendar.dateComponents([.year, .month], from: date)
        return calendar.date(from: components)!
    }

    func weekDay(date: Date) -> Int
    {
        let components = calendar.dateComponents([.weekday], from: date)
        return components.weekday! - 1
    }

    func weekDayAsString(date: Date) -> String
    {
        switch weekDay(date: date)
        {
        case 0:
            return "Sunday"
        case 1:
            return "Monday"
        case 2:
            return "Tuesday"
        case 3:
            return "Wednesday"
        case 4:
            return "Thursday"
        case 5:
            return "Friday"
        case 6:
            return "Saturday"
        default:
            return ""
        }
    }

    func addDays(date: Date, days: Int) -> Date
    {
        return calendar.date(byAdding: .day, value: days, to: date)!
    }

    func sundayForDate(date: Date) -> Date
    {
        var current = date
        let oneWeekAgo = addDays(date: current, days: -7)

        while(current > oneWeekAgo)
```

EventEdit



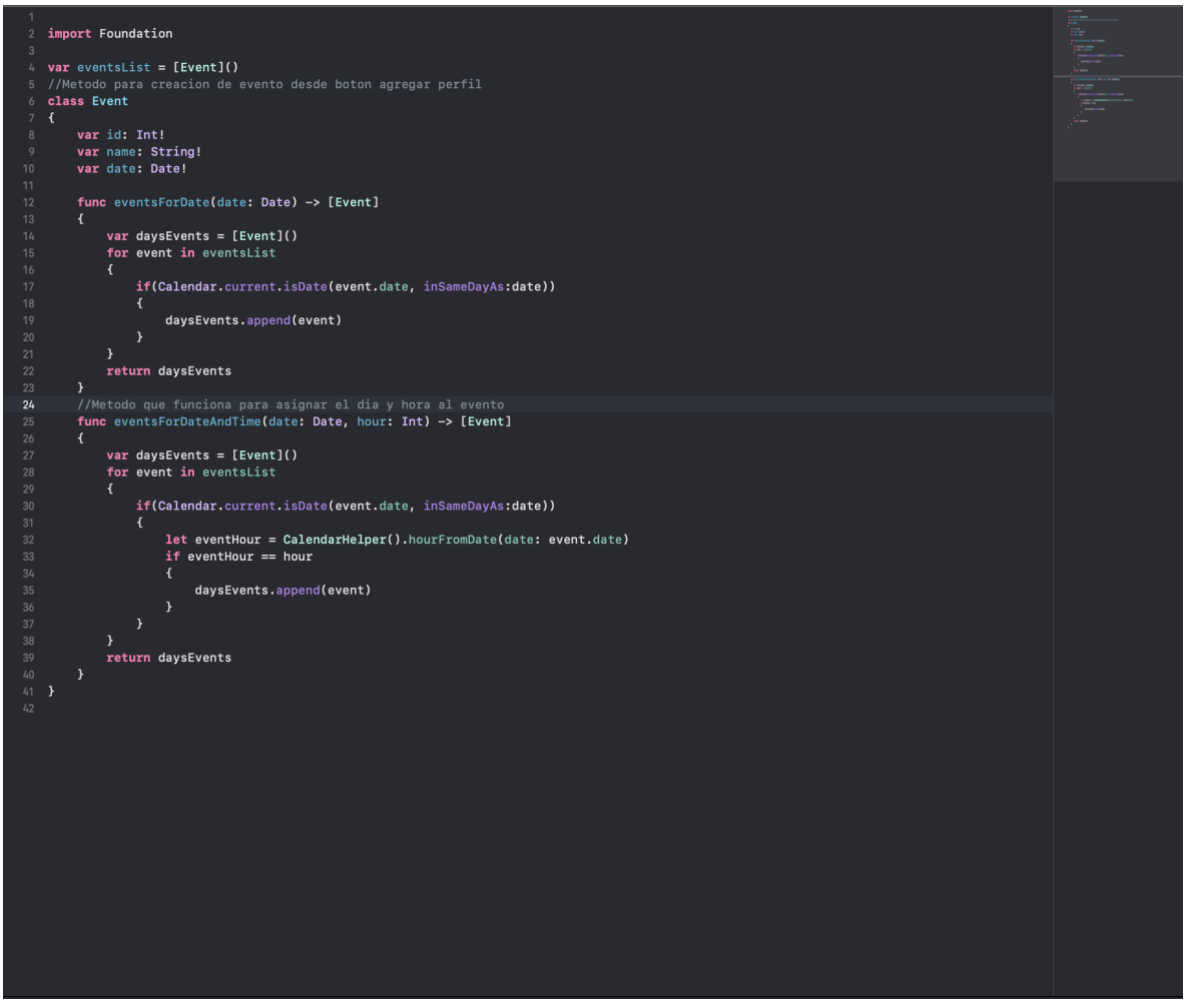
```swift
import UIKit
//Metodo que funciona para poder agregar y modificar eventos en calendario
class EventEditViewController: UIViewController
{

    @IBOutlet weak var nameTF: UITextField!
    @IBOutlet weak var datePicker: UIDatePicker!

    override func viewDidLoad()
    {
        super.viewDidLoad()

        datePicker.date = selectedDate
    }

    @IBAction func saveAction(_ sender: Any)
    {
        let newEvent = Event()
        newEvent.id = eventsList.count
        newEvent.name = nameTF.text
        newEvent.date = datePicker.date
        eventsList.append(newEvent)
        navigationController?.popViewController(animated: true)
    }
}

```

Calendar Day



```swift
import Foundation
// Metodo que funciona para asignar tipo de dato a los meses, semana y mes
class CalendarDay
{
    var day: String!
    var month: Month!

    enum Month
    {
        case previous
        case current
        case next
    }
}
```

Event



```swift

import Foundation

var eventsList = [Event]()
//Metodo para creacion de evento desde boton agregar perfil
class Event
{
    var id: Int!
    var name: String!
    var date: Date!

    func eventsForDate(date: Date) -> [Event]
    {
        var daysEvents = [Event]()
        for event in eventsList
        {
            if(Calendar.current.isDate(event.date, inSameDayAs:date))
            {
                daysEvents.append(event)
            }
        }
        return daysEvents
    }
    //Metodo que funciona para asignar el dia y hora al evento
    func eventsForDateAndTime(date: Date, hour: Int) -> [Event]
    {
        var daysEvents = [Event]()
        for event in eventsList
        {
            if(Calendar.current.isDate(event.date, inSameDayAs:date))
            {
                let eventHour = CalendarHelper().hourFromDate(date: event.date)
                if eventHour == hour
                {
                    daysEvents.append(event)
                }
            }
        }
        return daysEvents
    }
}

```