

Sesión 03:
Xcode, Swift, SwiftUI e iOS: Listas y Navegación
Grupos de Trabajo
Líneas de Productos Software

Pablo Gómez Rivas

Departamento de Informática
Universidad de Almería

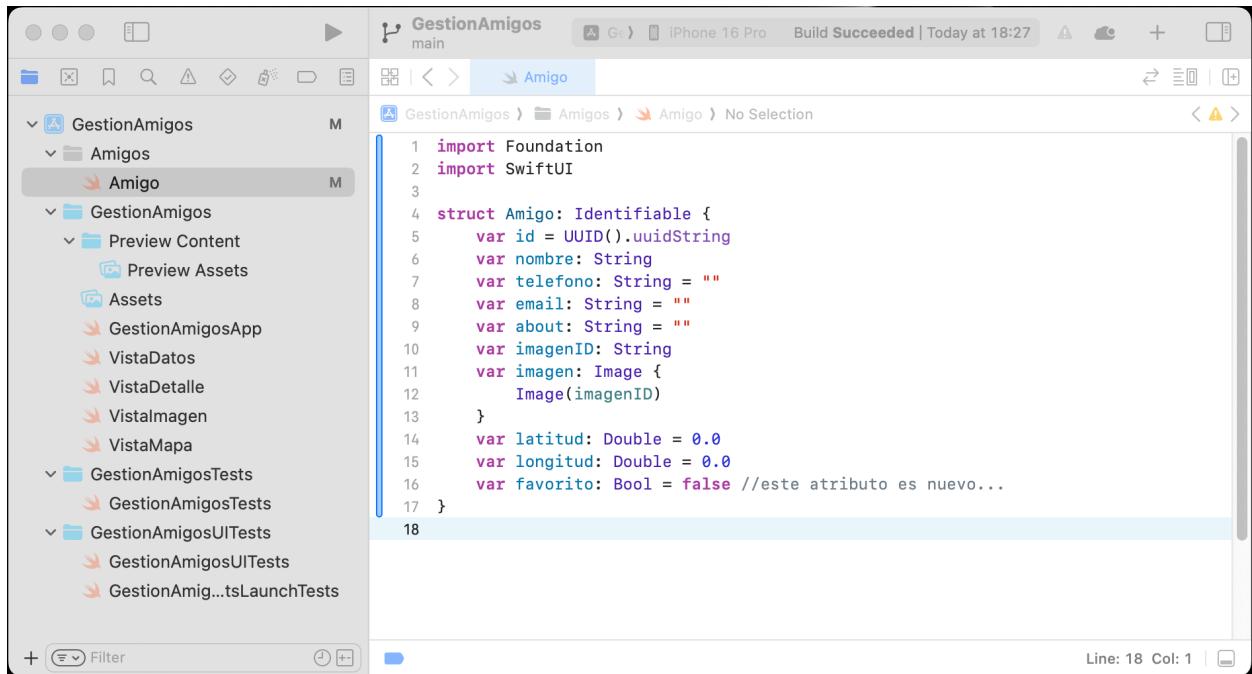
Almería, martes 1 de octubre de 2024

Índice de Contenidos

	Página
Creando el Modelo de Datos	3
Paso 1.....	3
Paso 2.....	3
Creando una lista de amigos.....	4
Paso 1.....	4
Paso 2 y 3.....	4
Paso 4.....	5
Paso 5.....	5
Paso 6.....	6
Permitiendo la modificación de datos (cambiando el estado de favorito).....	6
Pasos 1-5.....	6
Ejercicio 1.....	6
Permitiendo la modificación de datos (opinando sobre nuestro amigo).....	8
Paso 1.....	8
Ejercicio 2.....	9
Paso 2.....	9
Ejercicio 3.....	10
Ejercicio 4.....	10
Mejoras Futuras	12

Creando el Modelo de Datos

Paso 1



The screenshot shows the Xcode interface with the project 'GestionAmigos' open. The left sidebar shows the file structure. In the main editor, a new file named 'Amigo.swift' is being created. The code defines a struct 'Amigo' with various properties like id, nombre, telefono, email, about, imagenID, imagen, latitud, longitud, and favorito.

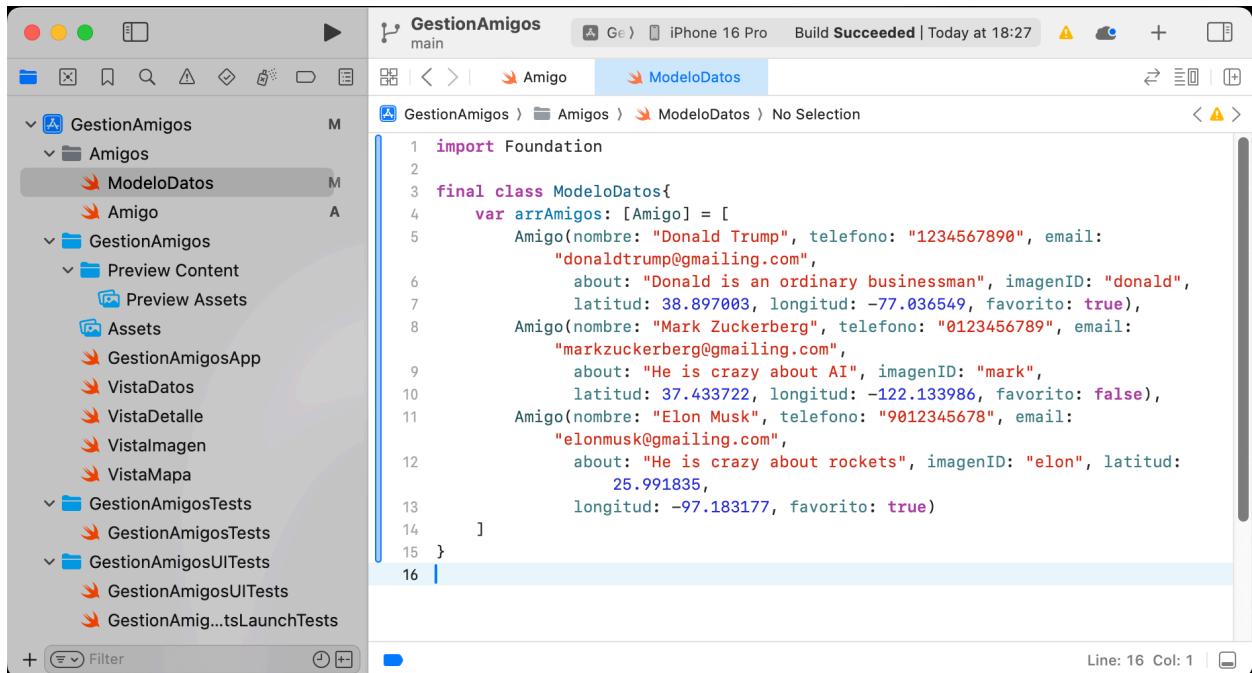
```

import Foundation
import SwiftUI

struct Amigo: Identifiable {
    var id = UUID().uuidString
    var nombre: String
    var telefono: String = ""
    var email: String = ""
    var about: String = ""
    var imagenID: String
    var imagen: Image {
        Image(imagenID)
    }
    var latitud: Double = 0.0
    var longitud: Double = 0.0
    var favorito: Bool = false //este atributo es nuevo...
}

```

Paso 2



The screenshot shows the Xcode interface with the project 'GestionAmigos' open. The left sidebar shows the file structure. In the main editor, a new file named 'ModeloDatos.swift' is being created. The code defines a final class 'ModeloDatos' with a variable 'arrAmigos' containing three Amigo instances: Donald Trump, Mark Zuckerberg, and Elon Musk.

```

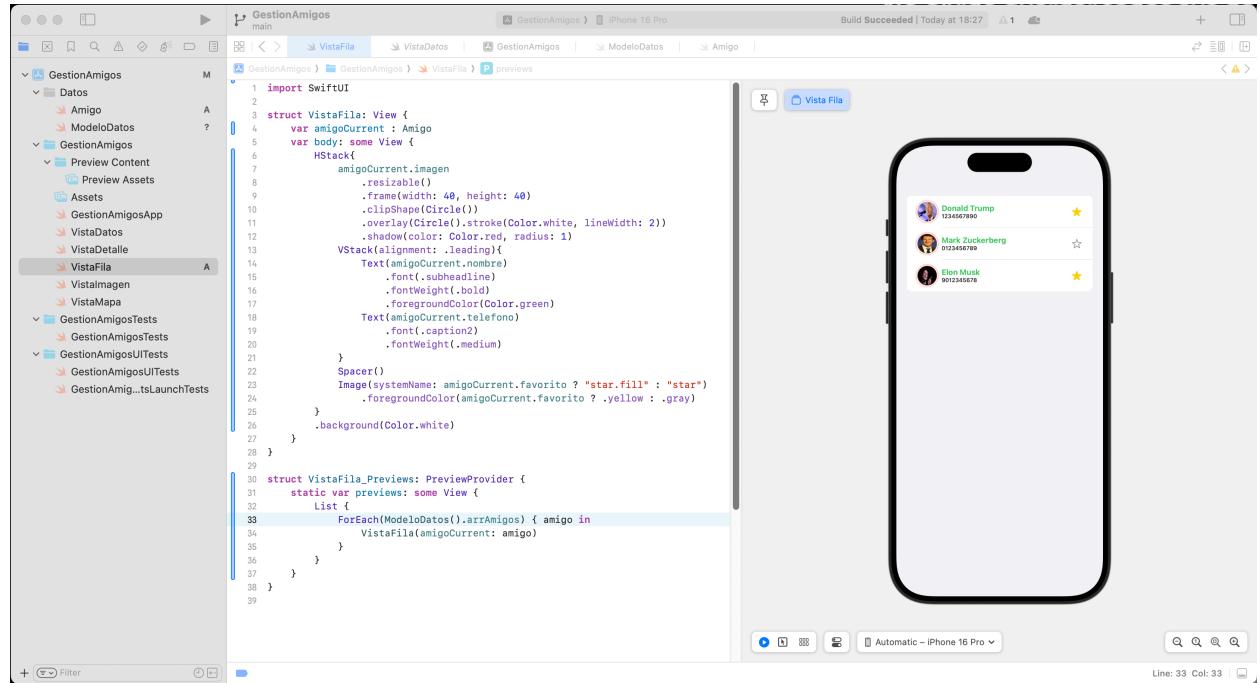
import Foundation

final class ModeloDatos{
    var arrAmigos: [Amigo] = [
        Amigo(nombre: "Donald Trump", telefono: "1234567890", email: "donaldtrump@gmailing.com",
               about: "Donald is an ordinary businessman", imagenID: "donald",
               latitud: 38.897003, longitud: -77.036549, favorito: true),
        Amigo(nombre: "Mark Zuckerberg", telefono: "0123456789", email: "markzuckerberg@gmailing.com",
               about: "He is crazy about AI", imagenID: "mark",
               latitud: 37.433722, longitud: -122.133986, favorito: false),
        Amigo(nombre: "Elon Musk", telefono: "9012345678", email: "elonmusk@gmailing.com",
               about: "He is crazy about rockets", imagenID: "elon", latitud: 25.991835,
               longitud: -97.183177, favorito: true)
    ]
}

```

Creando una lista de amigos

Paso 1



The screenshot shows the Xcode interface with the code editor open. The file is named `VistaFila.swift`. The code defines a `VistaFila` struct which contains a `HStack` with a `Image` and a `Text` element. It also includes a `Spacer` and a `Image` representing a star. A `PreviewProvider` is defined to show a list of friends from the `ModeloDatos` array.

```

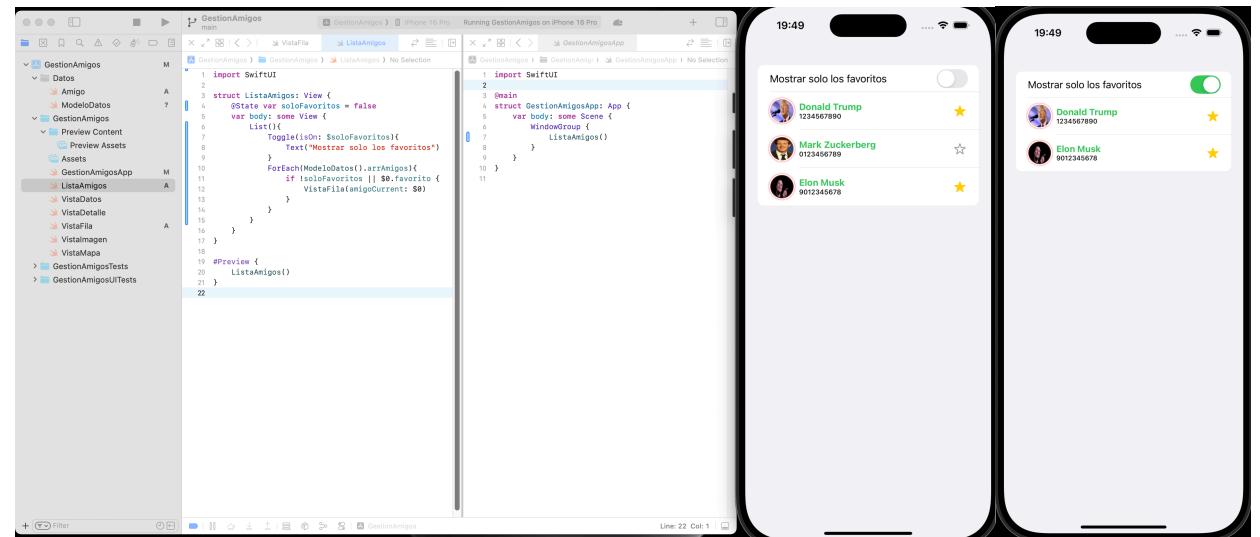
import SwiftUI

struct VistaFila: View {
    var amigoCurrent: Amigo
    var body: some View {
        HStack{
            amigoCurrent.imagen
                .resizable()
                .frame(width: 40, height: 40)
                .clipShape(Circle())
                .overlay(Circle().stroke(Color.white, lineWidth: 2))
                .shadow(color: Color.red, radius: 1)
            VStack(alignment: .leading){
                Text(amigoCurrent.nombre)
                    .font(.subheadline)
                    .fontWeight(.bold)
                    .foregroundColor(Color.green)
                Text(amigoCurrent.telefono)
                    .font(.caption2)
                    .fontWeight(.medium)
            }
            Spacer()
            Image(systemName: amigoCurrent.favorito ? "star.fill" : "star")
                .foregroundColor(amigoCurrent.favorito ? .yellow : .gray)
        }
        .background(Color.white)
    }
}

struct VistaFila_Previews: PreviewProvider {
    static var previews: some View {
        List {
           ForEach(ModeloDatos().arrAmigos) { amigo in
                VistaFila(amigoCurrent: amigo)
            }
        }
    }
}

```

Paso 2 y 3



The screenshot shows the Xcode interface with the code editor open. The file is named `VistaFila.swift`. The code defines a `ListadoAmigos` struct which contains a `Text` and a `ForEach` loop. The `ForEach` loop iterates over the `ModeloDatos` array and calls `VistaFila` with the current friend.

```

import SwiftUI

struct ListadoAmigos: View {
    @State var soloFavoritos = false
    var body: some View {
        List {
            Text("Mostrar solo los favoritos")
            Toggle(isOn: $soloFavoritos)
            ForEach(ModeloDatos().arrAmigos) {
                if !soloFavoritos || $0.favorito {
                    VistaFila(amigoCurrent: $0)
                }
            }
        }
    }
}

#Preview {
    ListadoAmigos()
}

```

Paso 4



The screenshot shows the Xcode interface with the following details:

- Top Bar:** Shows the project name "GestionAmigos" and the file "main".
- Navigation Bar:** Shows the navigation path: GestionAmigos > GestionAmigos > ListaAmigos > No Selection.
- Code Editor:** Displays the Swift code for the "ListaAmigos" view. The code uses SwiftUI to create a list of friends, filtering them by favorite status if the "soloFavoritos" state is true. It includes a toggle button to switch between showing all friends and only favorites. The code also uses NavigationView and NavigationLink to handle friend details.

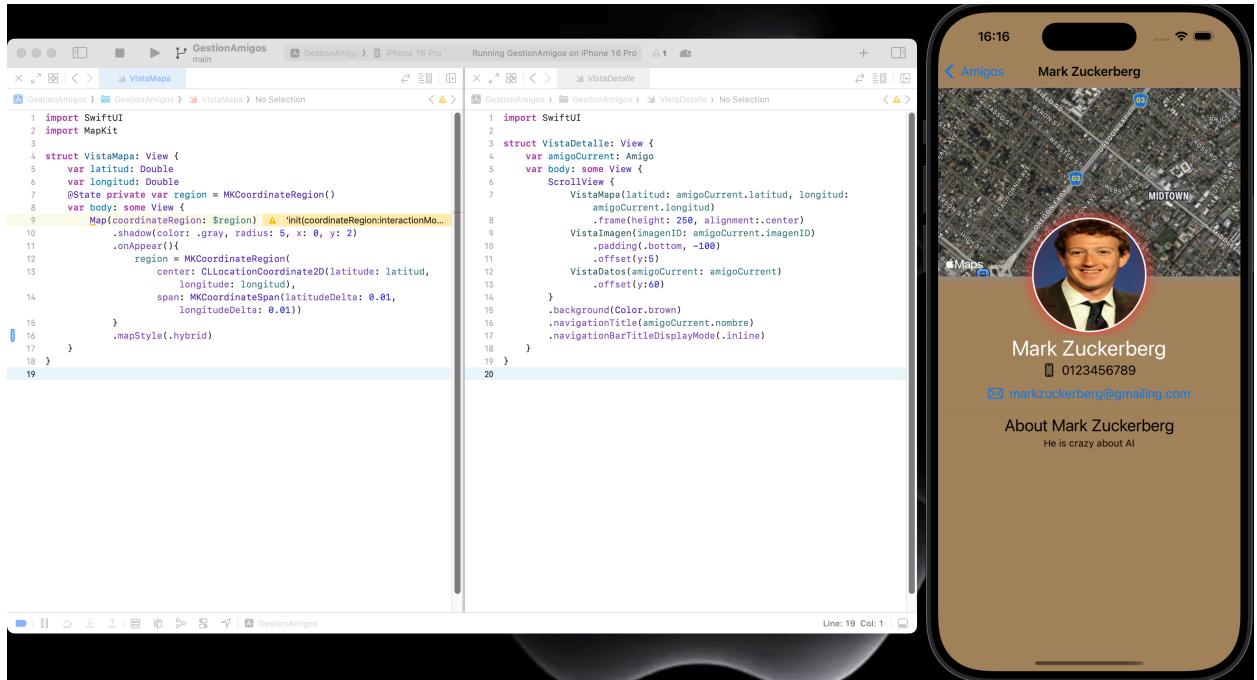
```
1 import SwiftUI
2
3 struct ListaAmigos: View {
4     @State var soloFavoritos = false
5     var body: some View {
6         NavigationView{
7             List(){
8                 Toggle(isOn: $soloFavoritos){
9                     Text("Mostrar solo los favoritos")
10                }
11                ForEach(ModeloDatos().arrAmigos){amigo in
12                    if !soloFavoritos || amigo.favorito {
13                        NavigationLink(destination: VistaDetalle()){
14                            VistaFila(amigoCurrent: amigo)
15                        }
16                    }
17                }
18            }.navigationTitle("Amigos")
19        }
20    }
21 }
22
23 #Preview {
24     ListaAmigos()
25 }
```

Paso 5



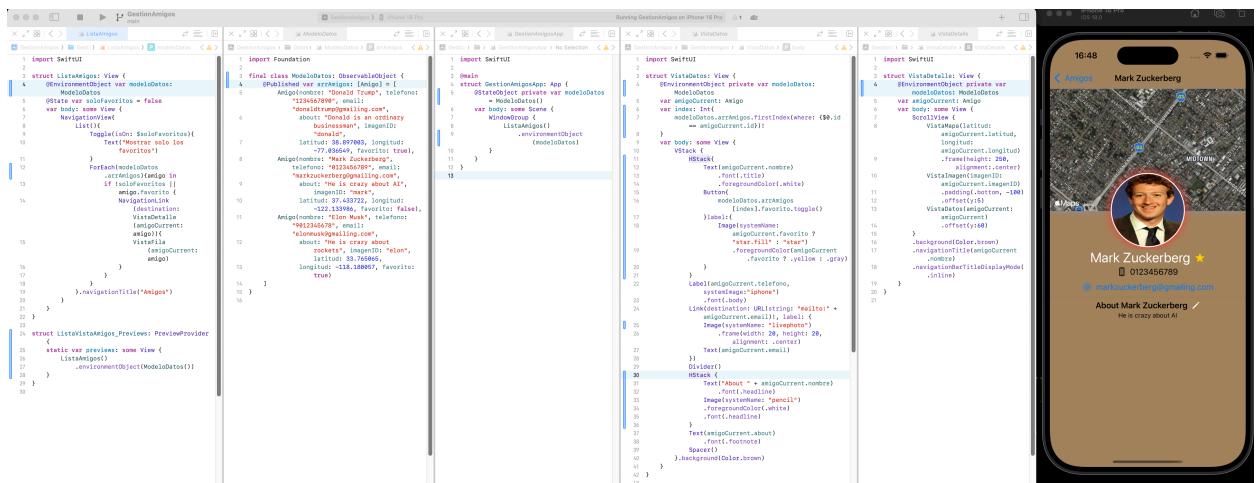
The screenshot shows a developer's environment with five Xcode tabs open, each displaying code related to a 'VistaMisAmigos' view controller. The tabs are titled 'VistaMisAmigos.swift', 'VistaMisAmigos.h', 'VistaMisAmigos.m', 'VistaMisAmigos+UI.swift', and 'VistaMisAmigos+UI.h'. The code includes imports for 'UIKit', 'Foundation', and 'CoreLocation'. It defines a class 'VistaMisAmigos' with methods for handling favorites and displaying a list of amigos. The 'VistaMisAmigos+UI' files contain Swift code for setting up UI components like labels, buttons, and tables. An iPhone X simulator is visible on the right, showing a map of Los Angeles with a circular callout for 'Elon Musk' containing his contact information.

Paso 6



Permitiendo la modificación de datos (cambiando el estado de favorito)

Pasos 1-5



Ejercicio 1

Análisis crítico

Ventajas de eliminar amigoCurrent y usar un índice:

- Simplicidad: Evitar tener una variable adicional puede simplificar el código en cuanto a la claridad de su propósito.
- Facilidad para encontrar el elemento en un array: Si trabajamos con arrays, acceder al amigo mediante su índice es una operación de tiempo constante, mientras que el uso de variables extra puede agregar complejidad innecesaria.
- Consistencia: Si se trabaja directamente con el índice del array, el estado de edición es más explícito y no se corre el riesgo de que amigoCurrent quede desincronizado con el array si este cambia.

Desventajas de eliminar amigoCurrent:

- Mayor acoplamiento a la estructura: Si más adelante se cambia el almacenamiento de un array a un diccionario, acceder mediante índices ya no será posible. En ese caso, sería necesario un identificador único para acceder a los elementos.
- Contexto compartido: amigoCurrent podría estar funcionando como una referencia a la entidad en edición. Si eliminamos esa referencia directa y usamos índices, tendríamos que ajustar otros lugares donde amigoCurrent se esté usando como tal.

Código propuesto

Supongamos que cada amigo tiene un identificador único (id), podríamos modificar el código de la siguiente manera:

Modelo de datos:

```
struct Amigo: Identifiable {
    var id: UUID
    var nombre: String
    var telefono: String
    var email: String
    // otros atributos...
}
```

Buscar amigo para editar por su id:

```
class AmigosModel: ObservableObject {
    @Published var amigos: [Amigo] = []
    func getAmigo(byId id: UUID) -> Amigo? {
        return amigos.first { $0.id == id }
    }
    func updateAmigo(_ updatedAmigo: Amigo) {
        if let index = amigos.firstIndex(where: { $0.id == updatedAmigo.id }) {
            amigos[index] = updatedAmigo
        }
    }
}
```

Vista de edición:

```
struct EditarAmigoView: View {
    @EnvironmentObject var amigosModel: AmigosModel
    var amigoId: UUID
    @State private var amigo: Amigo
    init(amigoId: UUID, amigosModel: AmigosModel) {
        self.amigoId = amigoId
        _amigo = State(initialValue: amigosModel.getAmigo(byId: amigoId)!)
```

```

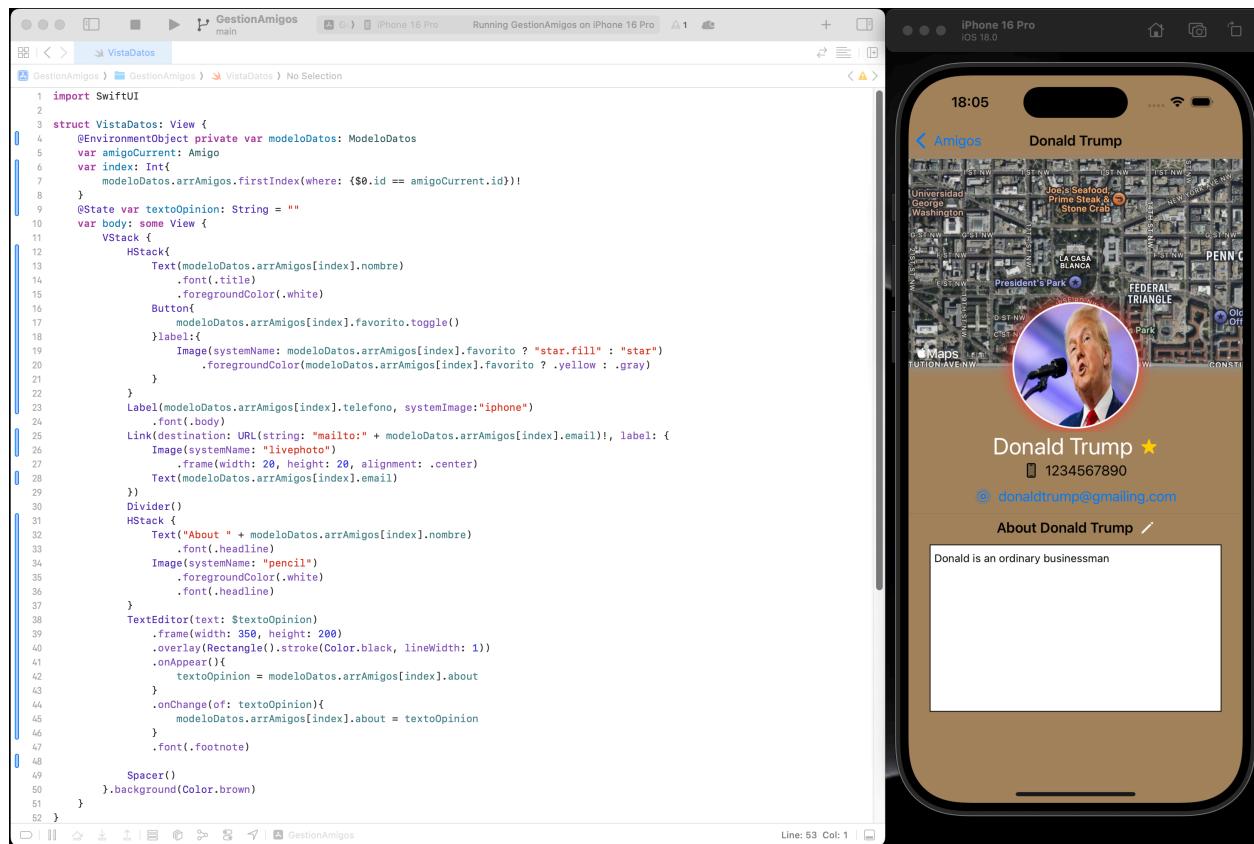
    }

var body: some View {
    // Contenido de la vista de edición
    // por ejemplo, para modificar el nombre:
    TextField("Nombre", text: $amigo.nombre)
    Button("Guardar") {
        amigosModel.updateAmigo(amigo)
    }
}
}

```

Permitiendo la modificación de datos (opinando sobre nuestro amigo)

Paso 1



Ejercicio 2

The screenshot shows the Xcode interface with the Swift code for the `VistaDatos` view. The code is as follows:

```

1 import SwiftUI
2
3 struct VistaDatos: View {
4     @EnvironmentObject private var modeloDatos: ModeloDatos
5     var amigoCurrent: Amigo
6     var index: Int
7     modeloDatos.arrAmigos.firstIndex(where: {$0.id == amigoCurrent.id})!
8 }
9 @State var textoOpinion: String = ""
10 var body: some View {
11     VStack {
12         HStack {
13             Text(modeloDatos.arrAmigos[index].nombre)
14                 .font(.title)
15                 .foregroundColor(.white)
16             Button {
17                 modeloDatos.arrAmigos[index].favorito.toggle()
18             }
19             label: {
20                 Image(systemName: modeloDatos.arrAmigos[index].favorito ? "star.fill" : "star")
21                     .foregroundColor(modeloDatos.arrAmigos[index].favorito ? .yellow : .gray)
22             }
23         }
24         Label(modeloDatos.arrAmigos[index].telefono, systemImage:"iphone")
25         Link(destination: URL(string: "mailto:" + modeloDatos.arrAmigos[index].email)!, label: {
26             Image(systemName: "livephoto")
27                 .frame(width: 20, height: 20, alignment: .center)
28             Text(modeloDatos.arrAmigos[index].email)
29         })
30         Divider()
31         HStack {
32             Text("About " + modeloDatos.arrAmigos[index].nombre)
33                 .font(.headline)
34             Image(systemName: "pencil")
35                 .foregroundColor(.white)
36                 .font(.headline)
37         }
38         TextEditor(text: $textoOpinion)
39             .frame(width: 350, height: 200)
40             .overlay(Rectangle().stroke(Color.black, lineWidth: 1))
41             .onAppear(){
42                 textoOpinion = modeloDatos.arrAmigos[index].about
43             }
44             .onChange(of: textoOpinion){
45                 modeloDatos.arrAmigos[index].about = textoOpinion
46             }
47             .font(.footnote)
48             .scrollContentBackground(.hidden)
49         Spacer()
50     }.background(Color.brown)
51 }
52

```

The iPhone 16 Pro preview shows a profile page for Donald Trump. It includes a map of Washington D.C., a circular profile picture of Donald Trump speaking, his name "Donald Trump" with a star icon, his phone number "1234567890", his email "donaldtrump@gmail.com", and a section titled "About Donald Trump" containing the placeholder text "Donald is an ordinary businessman".

Paso 2

The screenshot shows the Xcode interface with the updated Swift code for the `VistaDatos` view. The code is as follows:

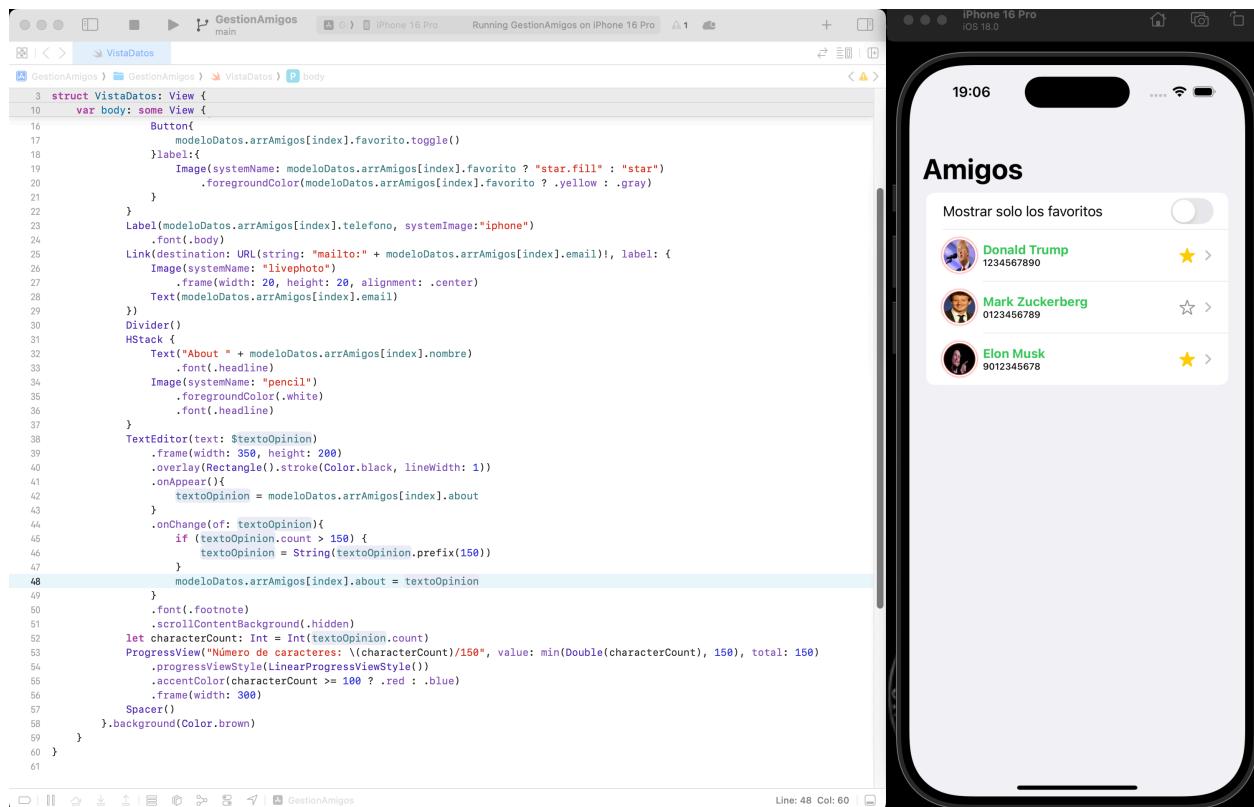
```

3 struct VistaDatos: View {
4     var body: some View {
5         Button {
6             modeloDatos.arrAmigos[index].favorito.toggle()
7         }
8         label: {
9             Image(systemName: modeloDatos.arrAmigos[index].favorito ? "star.fill" : "star")
10            .foregroundColor(modeloDatos.arrAmigos[index].favorito ? .yellow : .gray)
11        }
12     }
13     Label(modeloDatos.arrAmigos[index].telefono, systemImage:"iphone")
14     Link(destination: URL(string: "mailto:" + modeloDatos.arrAmigos[index].email)!, label: {
15         Image(systemName: "livephoto")
16             .frame(width: 20, height: 20, alignment: .center)
17         Text(modeloDatos.arrAmigos[index].email)
18     })
19     Divider()
20     HStack {
21         Text("About " + modeloDatos.arrAmigos[index].nombre)
22             .font(.headline)
23             Image(systemName: "pencil")
24                 .foregroundColor(.white)
25                 .font(.headline)
26         }
27         TextEditor(text: $textoOpinion)
28             .frame(width: 350, height: 200)
29             .overlay(Rectangle().stroke(Color.black, lineWidth: 1))
30             .onAppear(){
31                 textoOpinion = modeloDatos.arrAmigos[index].about
32             }
33             .onChange(of: textoOpinion){
34                 modeloDatos.arrAmigos[index].about = textoOpinion
35             }
36             .font(.footnote)
37             .scrollContentBackground(.hidden)
38     let characterCount: Int = Int(textoOpinion.count)
39     ProgressView("Número de caracteres: \(characterCount)/150", value: min(Double(characterCount), 150), total: 150)
40         .progressViewStyle(.linearProgressViewStyle())
41         .accentColor(characterCount >= 100 ? .red : .blue)
42         .frame(width: 300)
43     Spacer()
44 }
45 }.background(Color.brown)
46

```

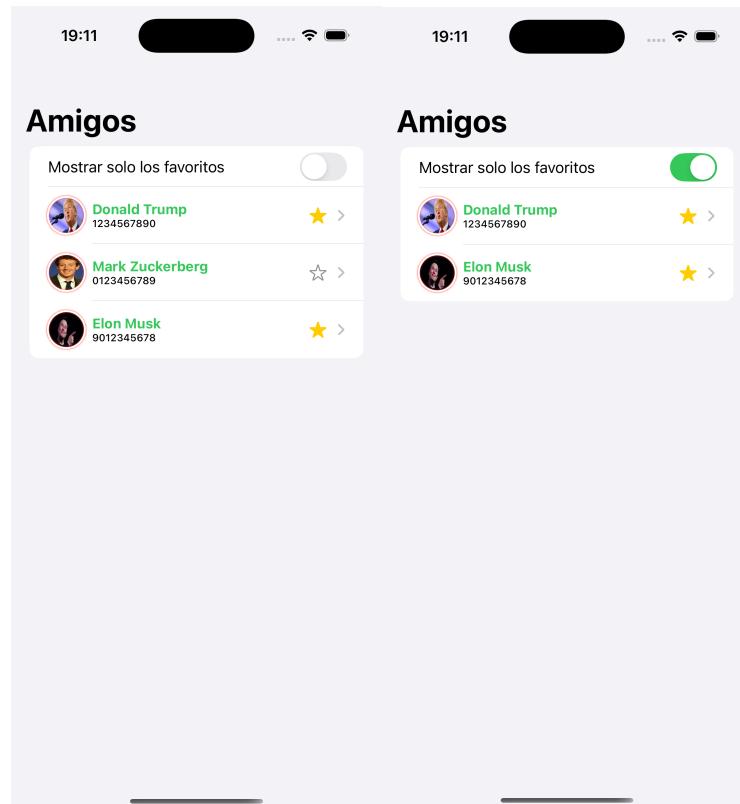
The iPhone 16 Pro preview shows the same profile page for Donald Trump, including the progress bar indicating 100/150 characters in the text editor.

Ejercicio 3

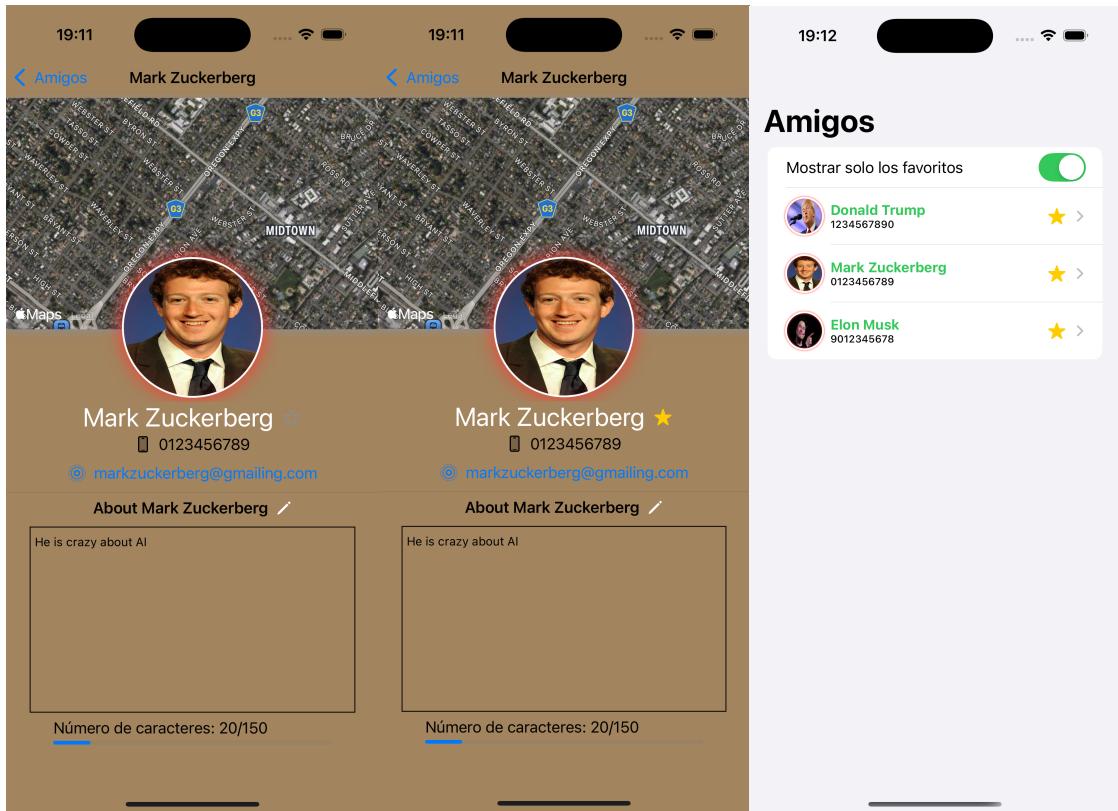


Ejercicio 4

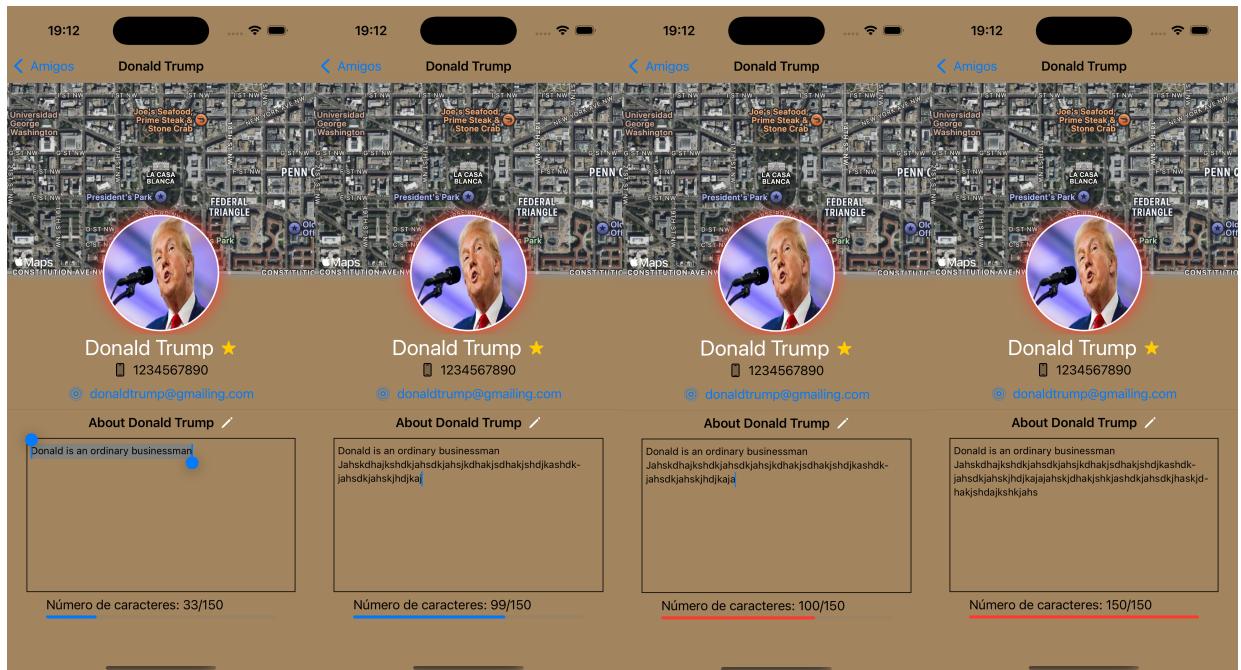
Paso 1: Comprobación del filtro de favoritos



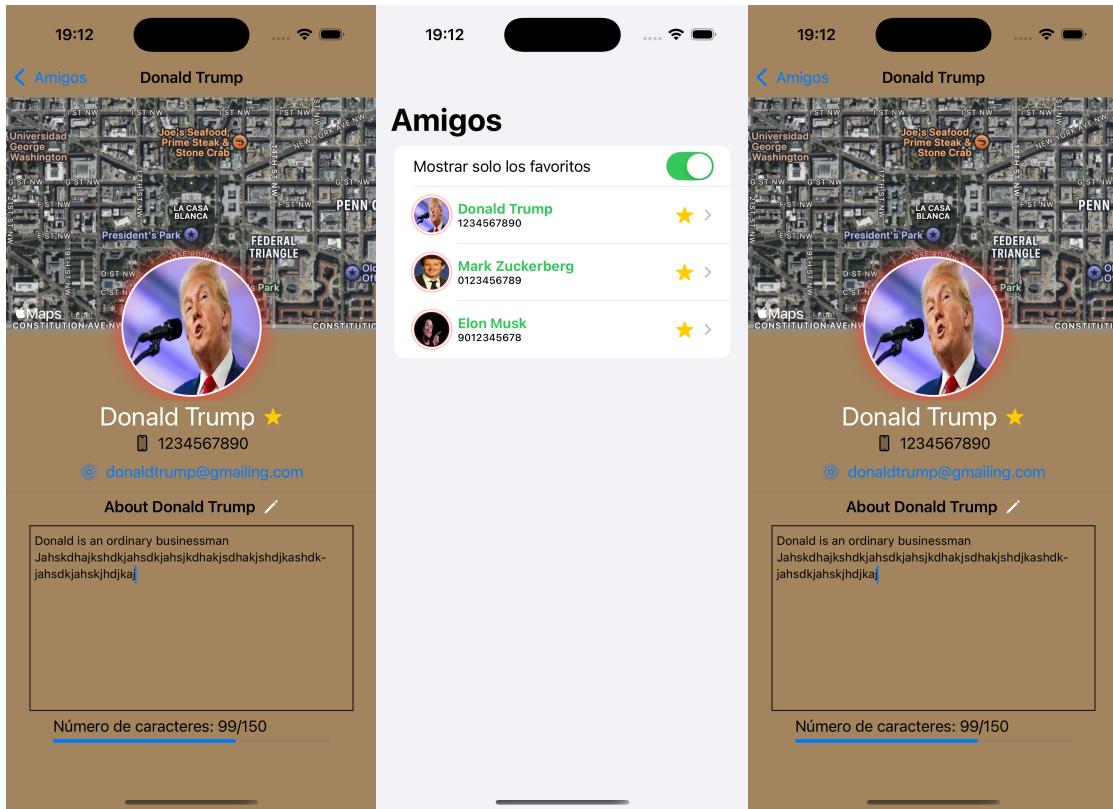
Paso 2: Comprobación del botón de favoritos



Paso 3: Comprobación de colores y límite de caracteres de la barra de progreso



Paso 4: Comprobación de guardado automático de la descripción



Mejoras Futuras

Más amigos en la lista

Para agregar más amigos, solo hay que incrementar el array arrAmigos dentro del modelo de datos. Pueden añadirse tantos amigos como se desee.

Eliminar un amigo de la lista

Para permitir la eliminación de amigos, se podría implementar una función que permita al usuario deslizar sobre una celda para eliminarla.

Ordenar la lista

Para ordenar la lista de amigos, se puede agregar un botón o menú que permita ordenar por diferentes criterios, como nombre o si son favoritos:

```
var amigosOrdenados: [Amigo] {
    modeloDatos.arrAmigos.sorted {
        if soloFavoritos {
            return $0.favorito && !$1.favorito
        } else {
            return $0.nombre < $1.nombre
        }
    }
}
```

Persistencia de datos

Opción 1: Base de datos local

Core Data: Es una opción muy robusta, diseñada para gestionar datos persistentes en dispositivos locales. Permite manejar relaciones entre entidades, consultas, y persistencia sin necesidad de conexión a Internet.

Opción 2: Servidor remoto (Firebase o un servidor propio con API REST)

Para que los datos sean accesibles desde múltiples dispositivos y estén sincronizados en tiempo real, se puede optar por una solución basada en la nube.