

Sesion05:
Xcode, Swift, SwiftUI e iOS: Persistencia de datos
CoreData
Grupos de Trabajo
Líneas de Productos Software

Pablo Gómez Rivas

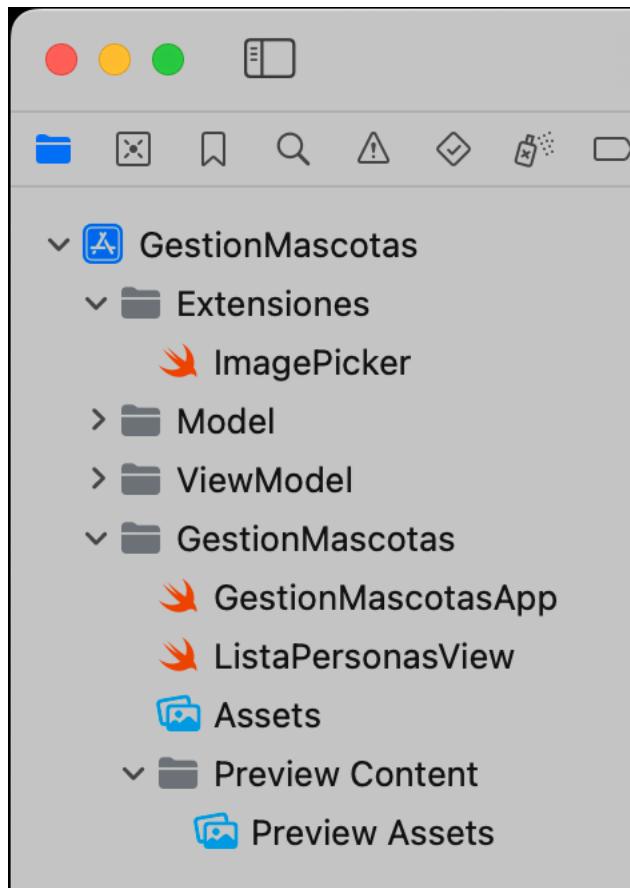
Departamento de Informática
Universidad de Almería

Almería, martes 29 de octubre de 2024

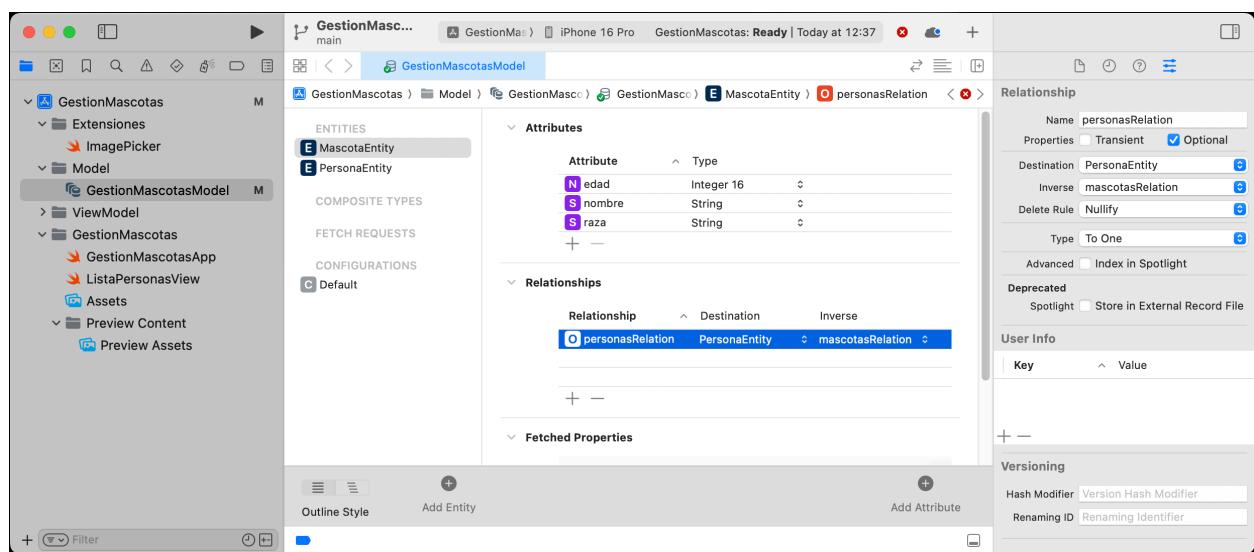
Índice de Contenidos

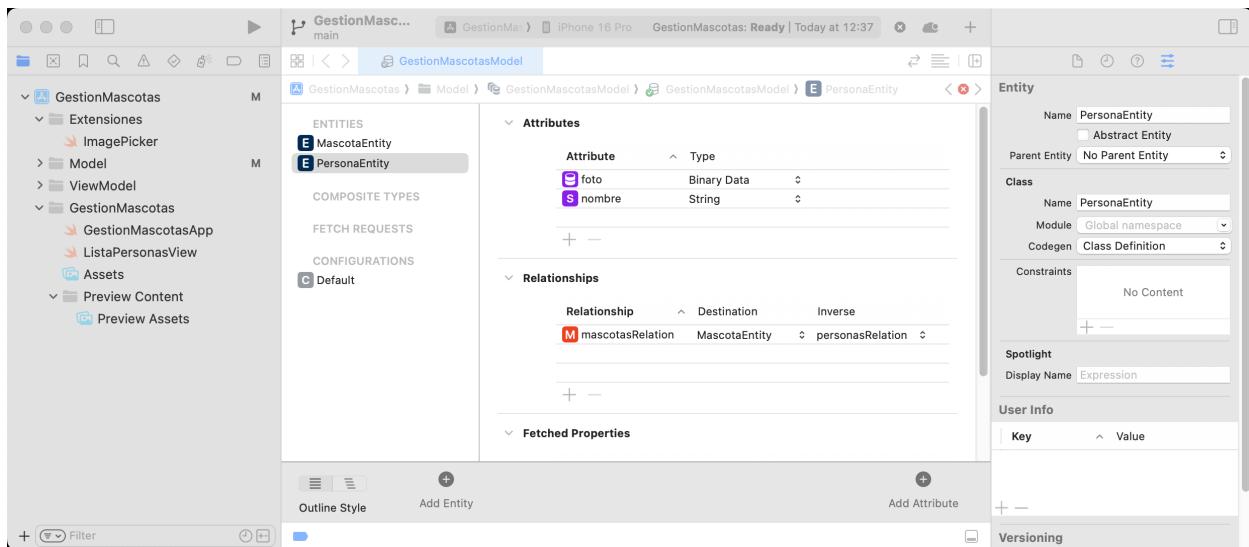
	Página
2. Creando el proyecto	3
3. Creando el modelo de datos	3
4. Creando el Gestor de Core Data.....	4
5. Creando el Modelo de Vista.....	5
6. Construyendo ListaPersonasView	6
Paso 1. Estructura principal de la vista	6
Paso 2. HeaderView	7
Paso 3. AddPersonaView.....	8
Paso 4. Añadiendo mascotas.....	9
Paso 5. Añadiendo mascotas (2 ^a parte)	10
Ejercicio 1	11

2. Creando el proyecto

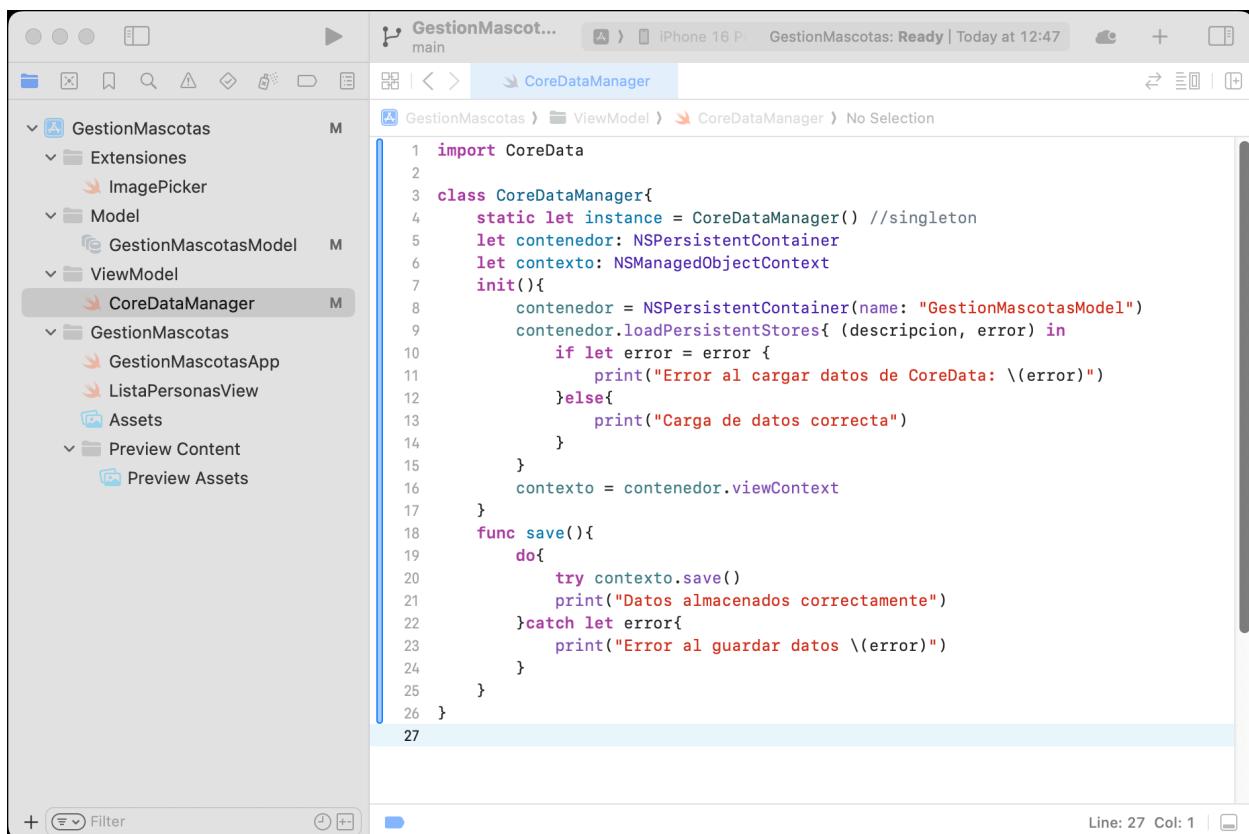


3. Creando el modelo de datos





4. Creando el Gestor de Core Data



5. Creando el Modelo de Vista

The screenshot shows three code editors side-by-side in Xcode:

- ViewModel.swift**: Contains Swift code for a View Model class. It imports Foundation, CoreData, and SwiftUI. It defines a ViewModel class with methods for loading data from Core Data and saving it back.
- GestionMascotasApp.swift**: Contains Swift code for the main application structure. It imports SwiftUI and defines a struct GestionMascotasApp with an @StateObject property for the ViewModel and a @MainActor var vm: ViewModel.
- CoreDataManager.swift**: Contains Swift code for a Core Data manager. It defines a static class CoreDataManager with a singleton instance. It handles loading persistent stores and saving data to the context.

```

// ViewModel.swift
import Foundation
import CoreData
import SwiftUI

class ViewModel: ObservableObject {
    @Published var personasArray: [PersonasEntity] = []
    @Published var mascotasArray: [MascotaEntity] = []

    init() {
        cargarDatos()
    }

    func cargarDatos() {
        personasArray.removeAll()
        mascotasArray.removeAll()

        let fetchPersonas = NSFetchRequest<PersonaEntity>(entityName: "PersonasEntity")
        let fetchMascotas = NSFetchRequest<MascotaEntity>(entityName: "MascotaEntity")

        do {
            self.personasArray = try gestorCoreData.contexto.fetch(fetchPersonas)
                .sorted(by: {$0.nombre! < $1.nombre!})
            self.mascotasArray = try gestorCoreData.contexto.fetch(fetchMascotas)
                .sorted(by: {$0.nombre! < $1.nombre!})
        } catch let error {
            print("Error al cargar los datos: \(error)")
        }
    }

    func guardarDatos() {
        gestorCoreData.save()
        cargarDatos()
    }

    func addPersonas(nombre: String, foto: UIImage) {
        let nuevaPersona = PersonaEntity(context: gestorCoreData.contexto)
        nuevaPersona.nombre = nombre
        nuevaPersona.foto = foto.pngData()
        guardarDatos()
    }

    func deletePersonas(indexSet: IndexSet) {
        for index in indexSet {
            gestorCoreData.contexto.delete(personasArray[index])
        }
        guardarDatos()
    }

    func addMascota(persona: PersonasEntity, nombre: String, edad: Int16, raza: String) {
        let nuevaMascota = MascotaEntity(context: gestorCoreData.contexto)
        nuevaMascota.nombre = nombre
        nuevaMascota.edad = edad
        nuevaMascota.raza = raza
        nuevaMascota.personasRelation = persona
        guardarDatos()
    }

    func deleteMascota(mascota: MascotaEntity) {
        gestorCoreData.contexto.delete(mascota)
        guardarDatos()
    }
}

// GestionMascotasApp.swift
import SwiftUI

@main
struct GestionMascotasApp: App {
    @StateObject private var vm: ViewModel = ViewModel()

    var body: some Scene {
        WindowGroup {
            ListaPersonasView()
                .environmentObject(vm)
        }
    }
}

// CoreDataManager.swift
import CoreData

class CoreDataManager {
    static let instance = CoreDataManager() //singleton
    let contenedor: NSPersistentContainer
    let contexto: NSManagedObjectContext

    init() {
        contenedor = NSPersistentContainer(name: "GestionMascotasModel")
        contenedor.loadPersistentStores { (descripcion, error) in
            if let error = error {
                print("Error al cargar datos de CoreData: \(error)")
            } else {
                print("Carga de datos correcta")
            }
        }
        contexto = contenedor.viewContext
    }

    func save() {
        do {
            try contexto.save()
            print("Datos almacenados correctamente")
        } catch let error {
            print("Error al guardar datos \(error)")
        }
    }
}

```

6. Construyendo ListaPersonasView

Paso 1. Estructura principal de la vista

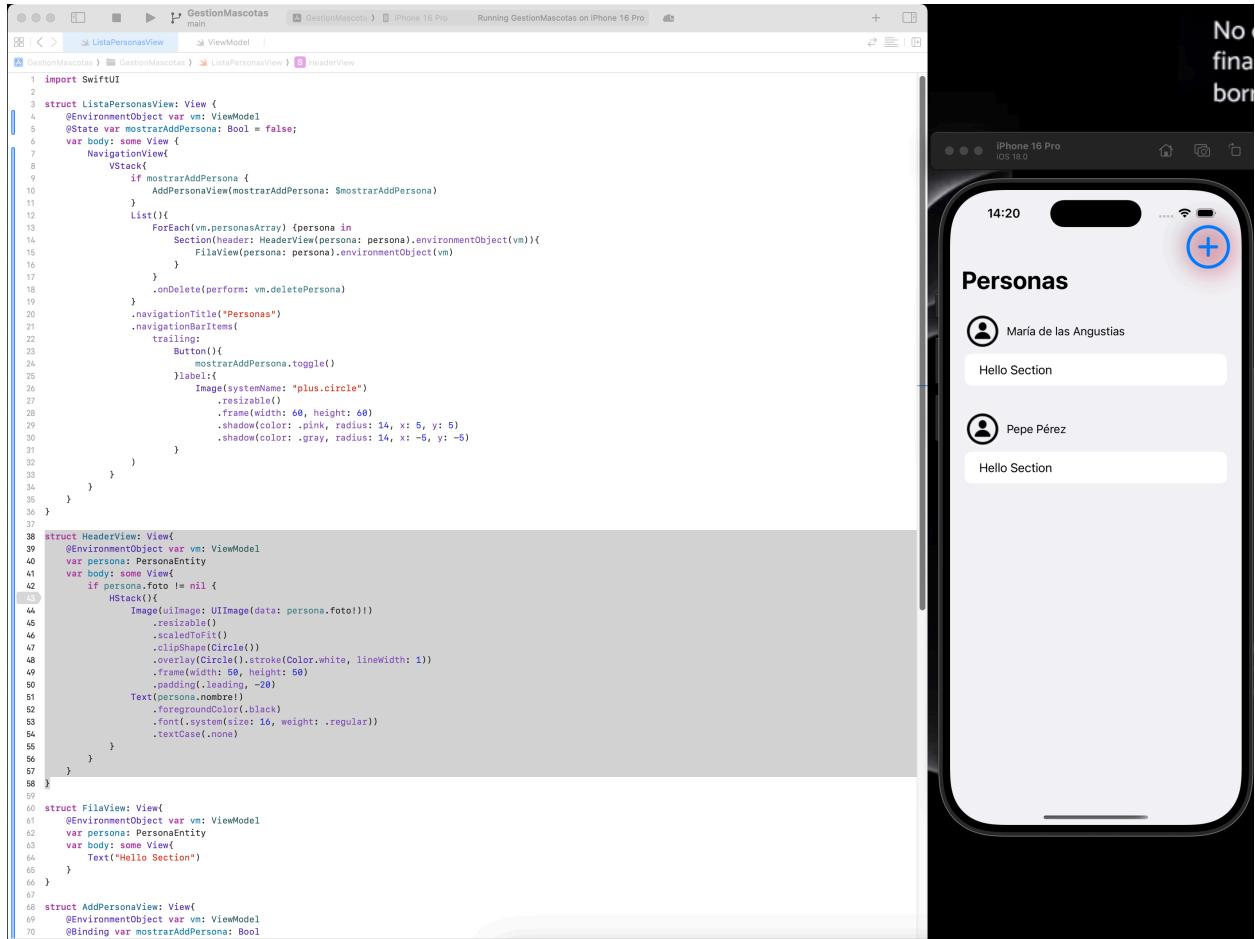
The image shows a split-screen view. On the left is the Xcode interface with the code editor open to `ViewModel.swift`. The code defines a `ViewModel` class that interacts with Core Data to manage persons and pets. On the right is a screenshot of an iPhone 16 Pro running iOS 18.0, displaying a list titled "Persons" with sections for "Hello Add Persona" and "Hello Section".

```

1 import Foundation
2 import CoreData
3 import SwiftUI
4
5 class ViewModel: ObservableObject{
6     let gestorCoreData = CoreDataManager.instance //singleton
7     @Published var personasArray: [PersonaEntity] = []
8     @Published var mascotasArray: [MascotaEntity] = []
9
10    init(){
11        addPersona(nombre: "Pepe Pérez", foto: UIImage(systemName: "plus")!)
12        addPersona(nombre: "María de las Angustias", foto: UIImage(systemName: "minus")!)
13        cargarDatos()
14    }
15    func cargarDatos(){
16        personasArray.removeAll()
17        mascotasArray.removeAll()
18        let fetchPersonas = NSFetchedResultsController<PersonaEntity>(entityName: "PersonaEntity")
19        let fetchMascotas = NSFetchedResultsController<MascotaEntity>(entityName: "MascotaEntity")
20        do{
21            self.personasArray = try gestorCoreData.contexto.fetch(fetchPersonas).sorted({$0.nombre! < $1.nombre!})
22            self.mascotasArray = try gestorCoreData.contexto.fetch(fetchMascotas).sorted({$0.nombre! < $1.nombre!})
23        }catch let error{
24            print("Error al cargar los datos: \(error)")
25        }
26    }
27    func guardarDatos(){
28        gestorCoreData.save()
29        cargarDatos()
30    }
31
32    func addPersona(nombre: String, foto: UIImage){
33        let nuevaPersona = PersonaEntity(context: gestorCoreData.contexto)
34        nuevaPersona.nombre = nombre
35        nuevaPersona.foto = foto.pngData()
36        guardarDatos()
37    }
38
39    func deletePersona(indexSet: IndexSet){
40        for index in indexSet{
41            gestorCoreData.contexto.delete(personasArray[index])
42        }
43        guardarDatos()
44    }
45
46    func addMascota(persona: PersonaEntity, nombre: String, edad: Int16, raza: String){
47        let nuevaMascota = MascotaEntity(context: gestorCoreData.contexto)
48        nuevaMascota.nombre = nombre
49        nuevaMascota.edad = edad
50        nuevaMascota.raza = raza
51        nuevaMascota.personasRelation = persona
52        guardarDatos()
53    }
54
55    func deleteMascota(mascota: MascotaEntity){
56        gestorCoreData.contexto.delete(mascota)
57        guardarDatos()
58    }
59}
60

```

Paso 2. HeaderView



The screenshot shows the Xcode interface with the HeaderView code in the main editor and a preview of the iPhone 16 Pro application in the storyboard editor.

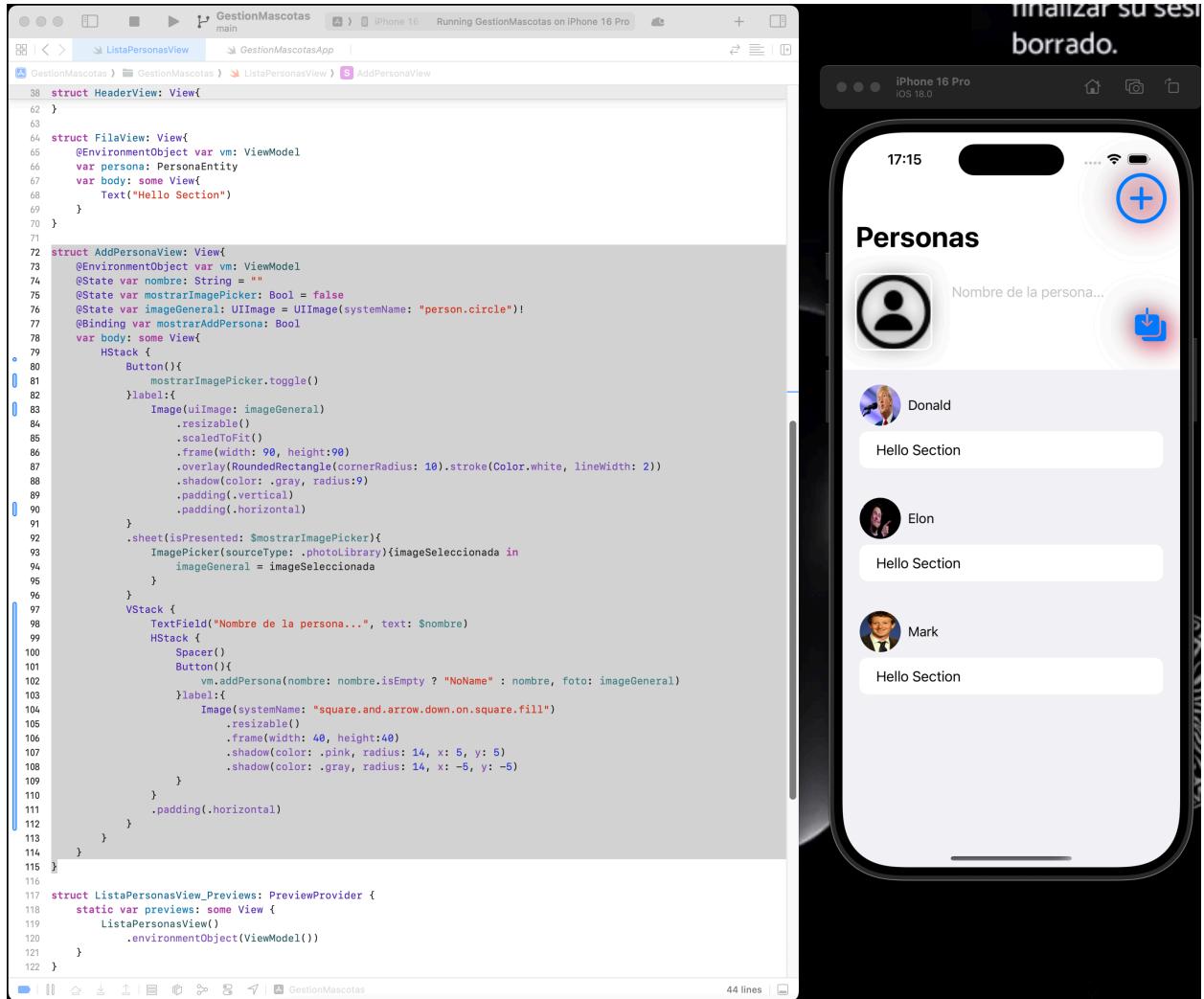
```

1 import SwiftUI
2
3 struct ListaPersonasView: View {
4     @EnvironmentObject var vm: ViewModel
5     @State var mostrarAddPersona: Bool = false;
6     var body: some View {
7         NavigationView{
8             VStack{
9                 if mostrarAddPersona {
10                     AddPersonaView(mostrarAddPersona: $mostrarAddPersona)
11                 }
12             List(){
13                 ForEach(vm.personasArray) {persona in
14                     Section(header: HeaderView(persona: persona).environmentObject(vm)){
15                         FilaView(persona: persona).environmentObject(vm)
16                     }
17                 }
18             .onDelete(perform: vm.deletePersona)
19             }
20             .navigationTitle("Personas")
21             .navigationBarItems{
22                 trailing:
23                     Button(){
24                         $mostrarAddPersona.toggle()
25                     }
26                     .label{
27                         Image(systemName: "plus.circle")
28                         .resizable()
29                         .frame(width: 60, height: 60)
30                         .shadow(color: .pink, radius: 14, x: 5, y: 5)
31                         .shadow(color: .gray, radius: 14, x: -5, y: -5)
32                     }
33                 }
34             }
35         }
36     }
37 }
38 struct HeaderView: View{
39     @EnvironmentObject var vm: ViewModel
40     var persona: PersonaEntity
41     var body: some View{
42         if persona.foto != nil {
43             HStack(){
44                 Image(uiImage: UIImage(data: persona.foto!))
45                     .resizable()
46                     .scaledToFit()
47                     .clipShape(Circle())
48                     .overlay(Circle().stroke(Color.white, lineWidth: 1))
49                     .frame(width: 50, height: 50)
50                     .shadow(radius: 20)
51             Text(persona.nombre)
52                 .foregroundColor(.black)
53                 .font(.system(size: 16, weight: .regular))
54                 .textCase(.none)
55             }
56         }
57     }
58 }
59 struct FilaView: View{
60     @EnvironmentObject var vm: ViewModel
61     var persona: PersonaEntity
62     var body: some View{
63         Text("Hello Section")
64     }
65 }
66 struct AddPersonaView: View{
67     @EnvironmentObject var vm: ViewModel
68     @Binding var mostrarAddPersona: Bool
69 }
70

```

The application preview shows a list of contacts with a header section. The header section includes a profile picture placeholder, the contact's name, and a plus icon for adding new contacts. Below the header, there are two contact entries: "Maria de las Angustias" and "Pepe Pérez", each with a "Hello Section" message.

Paso 3. AddPersonaView



The image shows a split-screen view. On the left is a code editor in Xcode displaying the `AddPersonaView` file. The code is written in Swift and uses SwiftUI components like `Text`, `Image`, `TextField`, and `Button`. It includes logic for adding new persons and handling image selection. On the right is a screenshot of an iPhone 16 Pro running the app. The screen shows a list of three persons: Donald, Elon, and Mark. Each person has a circular profile picture, a name, and a "Hello Section" placeholder below it. A blue floating action button (+) is visible at the top right of the screen.

```

1 struct HeaderView: View {
2     @EnvironmentObject var vm: ViewModel
3     var persona: PersonaEntity
4     var body: some View {
5         Text("Hello Section")
6     }
7 }
8
9 struct AddPersonaView: View {
10     @EnvironmentObject var vm: ViewModel
11     @State var nombre: String = ""
12     @State var mostrarImagePicker: Bool = false
13     @State var imageGeneral: UIImage = UIImage(systemName: "person.circle")!
14     @Binding var mostrarAddPersonas: Bool
15     var body: some View {
16         HStack {
17             Button() {
18                 mostrarImagePicker.toggle()
19             }
20             Image(uiImage: imageGeneral)
21                 .resizable()
22                 .scaledToFit()
23                 .frame(width: 90, height:90)
24                 .overlay(RoundedRectangle(cornerRadius: 10).stroke(Color.white, lineWidth: 2))
25                 .shadow(color: .gray, radius:9)
26                 .padding(.vertical)
27                 .padding(.horizontal)
28         }
29         .sheet(isPresented: $mostrarImagePicker) {
30             ImagePicker(sourceType: .photolibrary) { imageSeleccionada in
31                 imageGeneral = imageSeleccionada
32             }
33         }
34         VStack {
35             TextField("Nombre de la persona...", text: $nombre)
36             HStack {
37                 Spacer()
38                 Button() {
39                     vm.addPersona(nombre: nombre.isEmpty ? "NoName" : nombre, foto: imageGeneral)
40                 }
41                 Image(systemName: "square.and.arrow.down.on.square.fill")
42                     .resizable()
43                     .frame(width: 40, height:40)
44                     .shadow(color: .pink, radius: 14, x: 5, y: 5)
45                     .shadow(color: .gray, radius: 14, x: -5, y: -5)
46             }
47             .padding(.horizontal)
48         }
49     }
50 }
51
52 struct ListaPersonasView_Previews: PreviewProvider {
53     static var previews: some View {
54         ListaPersonasView()
55             .environmentObject(ViewModel())
56     }
57 }
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560

```

Paso 4. Añadiendo mascotas

The image shows the Xcode interface with the 'ListaPersonasView' code in the main editor and the 'GestionMascotas' file in the sidebar. The code defines a struct 'ListaPersonasView' containing a 'HeaderView' and a 'FilaView'. The 'HeaderView' includes a button to add a new person. The 'FilaView' displays three items: Donald, Elon, and Mark, each with a placeholder 'Hello Section'.

```

1  // swiftlint:disable all
2  import SwiftUI
3
4  struct ListaPersonasView: View {
5      @EnvironmentObject var vm: ViewModel
6      var persona: PersonaEntity
7      var body: some View {
8          if persona.foto != nil {
9              HStack() {
10                  Button(action: {
11                      let randomNumber = Int.random(in: 1000...2000)
12                      vm.addMascota(persona: persona, nombre: "\u{persona.nombre!}\(\randomNumber)", edad: 3, raza: "Gato")
13                  }) {
14                      Image(uiImage: UIImage(data: persona.foto!)!)
15                          .resizable()
16                          .clipShape(Circle())
17                          .overlay(Circle().stroke(Color.white, lineWidth: 1))
18                          .frame(width: 50, height: 50)
19                          .padding(.leading, -20)
20                  }
21                  Text(persona.nombre!)
22                      .foregroundColor(.black)
23                      .font(.system(size: 16, weight: .regular))
24                      .textCase(.none)
25              }
26          }
27      }
28  }
29
30  struct FilaView: View {
31      @EnvironmentObject var vm: ViewModel
32      var persona: PersonaEntity
33      var body: some View {
34          Text("Hello Section")
35      }
36  }
37
38  struct AddPersonaView: View {
39      @EnvironmentObject var vm: ViewModel
40      @State var nombre: String = ""
41
42      var body: some View {
43          Text("Carga de datos correcta
44              Datos almacenados correctamente
45              Datos almacenados correctamente
46              Datos almacenados correctamente
47              Datos almacenados correctamente")
48      }
49  }
50
51  struct HeaderView: View {
52      @EnvironmentObject var vm: ViewModel
53      var persona: PersonaEntity
54      var body: some View {
55          if persona.foto != nil {
56              HStack() {
57                  Button(action: {
58                      let randomNumber = Int.random(in: 1000...2000)
59                      vm.addMascota(persona: persona, nombre: "\u{persona.nombre!}\(\randomNumber)", edad: 3, raza: "Gato")
60                  })
61
62              }
63          }
64      }
65  }
66
67  struct ListaMascotasView: View {
68      @EnvironmentObject var vm: ViewModel
69
70      var body: some View {
71          Text("Hello Section")
72      }
73  }
74
75  struct MainView: View {
76      @EnvironmentObject var vm: ViewModel
77
78      var body: some View {
79          NavigationView {
80              ListaPersonasView()
81          }
82      }
83  }
84
85  struct GestionMascotas: App {
86      @EnvironmentObject var vm: ViewModel
87
88      var body: some Scene {
89          WindowGroup {
90              MainView()
91          }
92      }
93  }

```

Paso 5. Añadiendo mascotas (2^a parte)

The image displays a comparison between the SwiftUI code editor and the final mobile application interface.

Code Editor (Left):

```

38 struct HeaderView: View{
39     var body: some View{
40         .overlay(Circle().stroke(Color.white, lineWidth: 1))
41         .frame(width: 50, height: 50)
42         .padding(.leading, -20)
43     }
44 }
45
46 struct FilaView: View{
47     @EnvironmentObject var vm: ViewModel
48     var persona: PersonaEntity
49     var body: some View{
50         if let mascotas = persona.mascotasRelation?.allObjects as? [MascotaEntity] {
51             VStack{
52                 Text("\(mascotas.count) mascotas")
53                ForEach(mascotas) {mascota in
54                     HStack{
55                         Image(mascota.raza!)
56                         .resizable()
57                         .frame(width: 20, height: 20)
58                         .scaledToFit()
59                         Text("\(mascota.nombre!) (\(mascota.edad) años)")
60                         Spacer()
61                         Image(systemName: "minus.circle")
62                         .font(.headline)
63                         .foregroundColor(.red)
64                         .onTapGesture {
65                             vm.deleteMascota(mascota: mascota)
66                         }
67                     }
68                 }
69             }
70         }
71     }
72 }
73
74 struct AddPersonaView: View{
75     @EnvironmentObject var vm: ViewModel
76     @State var nombre: String = ""
77 }
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94

```

Mobile Application (Right):

- Donald:** 2 mascotas
 - Donald - 1799 (3 años)
 - Donald - 1594 (3 años)
- Elon:** 3 mascotas
 - Elon - 1545 (3 años)
 - Elon - 1441 (3 años)
 - Elon - 1669 (3 años)
- Mark:** 1 mascotas
 - Mark - 1491 (3 años)

Ejercicio 1

```

// ListaPersonasView.swift
import SwiftUI
@EnvironmentObject var vm: ViewModel
var persona: PersonEntity
@State var mostrarAddMascota = false
var body: some View {
    if persona != nil {
        HStack {
            Button(action: {
                Image(uiImage: UIImage(data: persona.foto)!)
                    .resizable()
                    .clipShape(Circle())
                    .overlay(Circle().stroke(Color.white, lineWidth: 1))
                    .frame(width: 50, height: 50)
                    .padding(.leading, -20)
            })
            .sheet(isPresented: $mostrarAddMascota) {
                AddMascotaView(mostrarAddMascota: $mostrarAddMascota, persona: persona)
                    .environmentObject(vm)
            }
            Text(persona.nombre!)
                .foregroundColor(.black)
                .font(.system(size: 16, weight: .regular))
                .textCase(.none)
        }
    }
}
struct FilaView: View {
    @EnvironmentObject var vm: ViewModel
    var persona: PersonEntity
    var body: some View {
        if let mascotas = persona.mascotasRelation?.allObjects as? [MascotaEntity] {
            VStack {
                Text("\(mascotas.count) mascotas")
                ForEach(mascotas) { mascota in
                    HStack {
                        Image(mascota.raza!)
                            .resizable()
                            .frame(width: 20, height: 20)
                            .scaledToFit()
                        Text("\(mascota.nombre), (\(mascota.edad) años)")
                    }
                }
            }
        }
    }
}

// AddMascotaView.swift
import SwiftUI
@EnvironmentObject var vm: ViewModel
@State var nombre: String = ""
@State var edad: Double = 0
@State var tipoMascota: String = "Perro"
@Binding var mostrarAddMascota: Bool
var persona: PersonEntity // Persona a la que se le añadirá la mascota
var body: some View {
    VStack(spacing: 10) {
        TextField("Nombre de la mascota", text: $nombre)
            .font(.system(size: 18, weight: .bold))
            .textCase(.none)
            .textFieldStyle(PlainTextFieldStyle())
            .padding()
        // Control deslizante para la edad
        VStack {
            HStack {
                Text("Edad: \(Int(edad))") // Convertimos la edad a Int para mostrar
                .font(.system(size: 18, weight: .bold))
                .foregroundColor(.black)
                .textCase(.none)
                Slider(value: $edad, in: 0...20, step: 1) // Ajuste de rango y paso
            }
            .padding(.horizontal)
        }
        Picker("Tipo de Mascota", selection: $tipoMascota) {
            Text("Perro").tag("Perro")
            Text("Gato").tag("Gato")
        }
        .pickerStyle(SegmentedPickerStyle())
        .padding()
        // Botones de aceptar y cancelar
        HStack(spacing: 50) {
            // Botón cancelar
            Button(action: {
                mostrarAddMascota = false
            }) {
                Image(systemName: "hand.thumbsdown")
                    .resizable()
                    .frame(width: 40, height: 40)
                    .foregroundColor(.red)
            }
            // Botón aceptar
            Button(action: {
                // Añadir mascota al ViewModel
                vm.addMascota(persona: persona, nombre: nombre, edad: Int16(edad), raza: tipoMascota)
                mostrarAddMascota = false
            }) {
                Image(systemName: "hand.thumbsup")
                    .resizable()
                    .frame(width: 40, height: 40)
                    .foregroundColor(.blue)
            }
        }
        .padding()
        .background(RoundedRectangle(cornerRadius: 15).fill(Color(.systemGray6)).shadow(radius: 10))
        .padding()
    }
}

```

18:26

Nombre de la mascota

Edad: 0

PERRO GATO

👎 👍

18:26

Personas

Donald

2 mascotas

Miau (20 años)

Toby (5 años)

Elon

2 mascotas

Doge (4 años)

Misifu (0 años)

Mark

2 mascotas

Woof (3 años)

MEOW (7 años)