

PAMN 2022/23

TRABAJO DE CURSO

APLICACIÓN IOS



Contenido

1. Introducción.....	3
2. Desarrollo	4
Conceptualización.....	4
Definición	5
Diseño	6
Desarrollo	8
Temporización	19
Herramientas	20
3. Conclusiones	21
4. Bibliografía.....	22

1. Introducción

Para esta, la asignatura de Programación de Aplicaciones Móviles Nativas, se nos había asignado un trabajo con temática libre en el que plasmar lo aprendido en *Android Studio* durante el semestre. El proyecto que quisimos llevar a cabo en un inicio fue un juego *casual* para móviles, tomando inspiración en juegos como *Flappy Bird* o el famoso minijuego que ofrece el navegador *Google Chrome* cuando no se dispone de una conexión a internet.

Lamentablemente, descartamos dichas ideas por lo que implicaban a nivel de logística, donde no entraba en los planes hacer uso de los sistemas de almacenamiento de datos y autenticación que ofrece *FireBase*. Por tanto, decidimos hacer una aplicación de mensajería cotidiana como podría ser *WhatsApp*.

Una vez puestos de acuerdo en lo que queríamos hacer, procedimos con la creación del proyecto *Android*. Vista la pronta finalización de este, y tener en nuestro poder un ordenador MAC, el equipo acordó que sería buena idea realizar una versión de la aplicación para iOS como trabajo adicional.

He aquí, pues, el informe de dicho trabajo, en donde detallaremos las distintas etapas por las que ha pasado el desarrollo de la aplicación.

Comencemos...

2. Desarrollo

Conceptualización

En este apartado, como se mencionó en la introducción, habíamos tenido un par de ideas previas a la aplicación de mensajería por la cual nos decantamos al final.

El primer concepto que tuvimos de la aplicación que queríamos era un minijuego *casual* para móviles que se basase en la rejugabilidad y que consiguiese mantener una base estable de jugadores. A este modelo se le sumarían, en caso de ser una aplicación para comercializar, métodos de monetización que, en caso de cumplir los criterios anteriormente mencionados, podrían dar un resultado monetario interesante. Dado el fin didáctico de este programa, obviamos la obtención de beneficios y nos centramos en desarrollar un juego en base a nuestras preferencias y gustos.

Pese a nuestra decisión, el tiempo se encargó de hacernos cambiar de parecer, pues el juego que teníamos en mente no cumplía ciertos requisitos, los cuales eran métodos de almacenamiento de datos o métodos de autenticación para contar con una gestión de usuarios.

Tras meditar las alternativas, decidimos que un proyecto viable y que cumpliría los requisitos sería, como ya se ha dicho, una aplicación de mensajería.

Nuestra idea, por tanto, pasaría a ser una aplicación en la cual se pudiesen mantener conversaciones en línea, tal y como pueden hacer otras aplicaciones como *Whatsapp*, *Telegram*, *etc...*

Definición

Ya con un concepto en mente, pasamos a describir la aplicación que deseamos. Este proceso incluye la definición del usuario al que irá dirigida la aplicación y la funcionalidad de esta.

Nuestro objetivo es crear una aplicación que pueda llegar a un gran rango de usuarios, aunque con ciertas limitaciones en cuanto a la accesibilidad, puesto que solo se podrá hacer uso completo de la aplicación si el usuario es capaz de leer.

La aplicación en sí tendría una funcionalidad sencilla: se contaría con métodos de autenticación con los cuales podría iniciarse sesión o crear una cuenta. A continuación, se mostraría una pantalla en la cual se podría elegir al usuario con el que chatear (el número de usuarios disponibles dependerá de los que se encuentren registrados en la aplicación) y, al seleccionarlo, se podrían enviar mensajes y leer el historial de la conversación que se ha tenido con el usuario en cuestión.

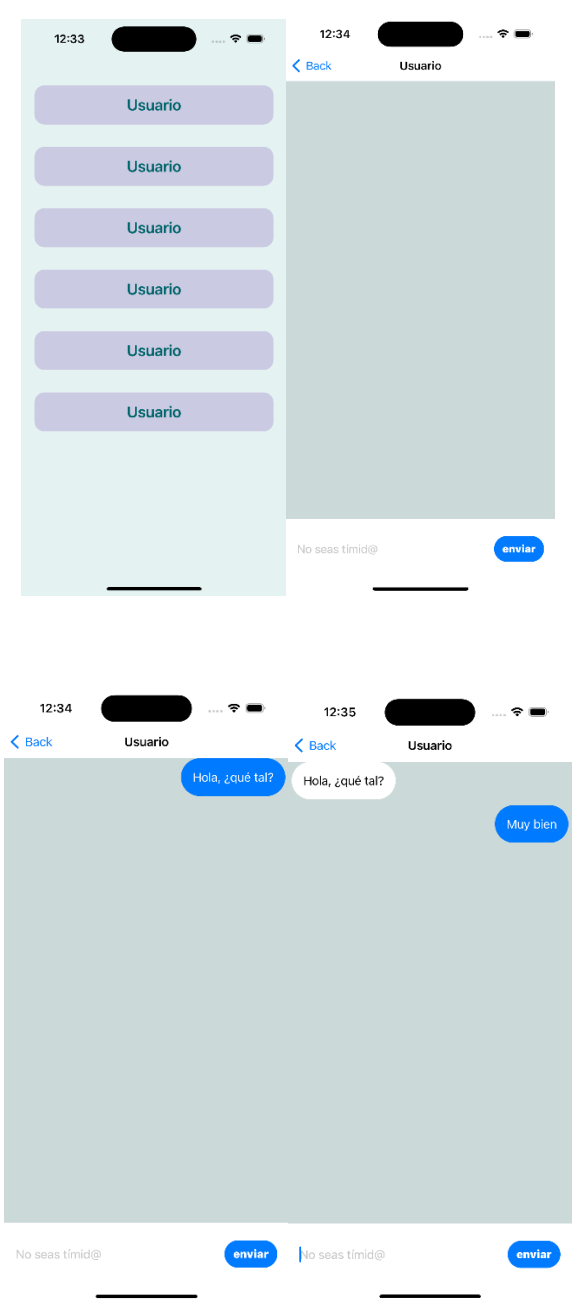
Diseño

El primer diseño que al equipo se le pasó por la cabeza para la versión de *iOS* de esta aplicación fue una interfaz minimalista y concisa, tal y como se realizó en la versión de *Android*.

Lo primero que se vería tras iniciar la aplicación sería una pantalla en la que poder *loggearse* con una cuenta ya registrada o crear la propia cuenta. *iOS* cuenta con una utilidad llamada *picker*, la cual nos permitirá tener el inicio de sesión y registro en una misma página de manera estética:

The image shows a mobile app interface with two side-by-side panels on a light blue background. Both panels have a status bar at the top showing the time 12:23, a black pill-shaped notch, and icons for signal strength, Wi-Fi, and battery. The left panel is titled 'Inicia Sesión' and contains a toggle with 'Inicar sesión' (active) and 'Registrarse'. Below are input fields for 'Email:' and 'Contraseña:', followed by a pink 'Iniciar sesión' button. The right panel is titled 'Regístrate' and contains a toggle with 'Inicar sesión' and 'Registrarse' (active). Below are input fields for 'Nombre:', 'Email:', and 'Contraseña:', followed by a pink 'Crear cuenta' button. Both panels have a black horizontal line at the bottom representing the home indicator.

Luego de haberse identificado en la aplicación, se mostraría un listado de los usuarios registrados donde, al pulsarlos, se accedería a un chat con el mismo. En dicho chat se mostrarían los mensajes previos, en caso de existir, y se podrán enviar mensajes nuevos.



Desarrollo

Tras plantear la funcionalidad de la aplicación y, en consecuencia, proponer los diseños, nos pusimos manos a la obra.

Lo primero que hicimos fue crear el proyecto en *Xcode* para poder comenzar a programar. Decidimos que la aplicación se haría con *SwiftUI*.

Para poder conseguir el resultado deseado, diferenciamos 3 tipos de funcionalidades para las clases mediante una variante del patrón *MVC* (Modelo, Vista, Controlador):

MODEL:

Son las clases que usaremos para definir los objetos que se usarán en el programa. Estos objetos son:

- *ChatMessage*: define la estructura de un mensaje del chat.
- *ChatUser*: define la estructura de un usuario de la aplicación.


```

1  //
2  // ChatMessage.swift
3  // ChatApplicationIos
4  //
5  // Created by Álvaro Antonio Suárez Quintana on 19/11/22.
6  //
7
8  import Foundation
9
10 struct ChatMessage : Identifiable{
11     let message, from, to, documentID : String
12
13     var id : String { documentID }
14
15     init(documentID: String, data: [String : Any]){
16         self.documentID = documentID
17         self.from = data["from"] as? String ?? ""
18         self.message = data["message"] as? String ?? ""
19         self.to = data["to"] as? String ?? ""
20     }
21 }

```

```

1  //
2  // ChatUser.swift
3  // ChatApplicationIos
4  //
5  // Created by Álvaro Antonio Suárez Quintana on 19/11/22.
6  //
7
8  import Foundation
9
10 struct ChatUser: Identifiable {
11
12     var id: String { uid }
13     let email, name, uid : String
14
15     init (data: [String: Any]){
16         self.email = data["email"] as? String ?? ""
17         self.name = data["name"] as? String ?? ""
18         self.uid = data["uid"] as? String ?? ""
19     }
20 }

```

Ambas clases cuentan con un id, el cual será asignado por *FireBase*. Este id será muy útil para, a posteriori, poder facilitar la iteración con los arrays compuestos por estos objetos, además de ser una manera de diferenciar objetos entre ellos.

Cabe destacar el uso de *TimeStamps* en *ChatMessage* que, pese a no aparecer en la declaración del objeto, al guardarse un mensaje en *FireStore* se almacenará el momento en el que se guardó, teniendo así una manera de ordenar los mensajes al cargarlos desde la base de datos ya que, por defecto, se ordenan por id.

UTIL:

Estas clases son usadas para facilitar el uso de paquetes o definir variables de entorno que se usarán repetidas veces a lo largo del programa. Contamos con:

- *FirebaseConstants*: esta clase es utilizada para definir las variables de entorno y, de esta manera, poder evitar errores debido a un fallo de escritura a la hora de leer o escribir en la base de datos.
- *FirebaseManager*: esta clase tiene la función de proporcionar un objeto tipo *firestore* y tipo *auth* para facilitar el uso de los respectivos servicios de *FireBase*.

```

1  //
2  //  FirebaseConstants.swift
3  //  ChatApplicationIos
4  //
5  //  Created by Álvaro Antonio Suárez Quintana on 19/11/22.
6  //
7
8  import Foundation
9
10 struct FirebaseConstants {
11     static let from = "from"
12     static let to = "to"
13     static let message = "message"
14     static let chats = "chats"
15     static let users = "users"
16     static let name = "name"
17     static let uid = "uid"
18     static let email = "email"
19     static let timestamp = "timestamp"
20 }

```

```

1  //
2  //  FirebaseManager.swift
3  //  ChatApplicationIos
4  //
5  //  Created by Álvaro Antonio Suárez Quintana on 19/11/22.
6  //
7
8  import Foundation
9  import Firebase
10
11 class FirebaseManager : NSObject {
12     let auth: Auth
13     let firestore: Firestore
14
15     static let shared = FirebaseManager()
16
17     override init(){
18         FirebaseApp.configure()
19         self.auth = Auth.auth()
20         self.firestore = Firestore.firestore()
21         super.init()
22     }
23 }

```

VIEW:

Estas clases son usadas para presentar la interfaz gráfica que verá el usuario al ejecutar el programa. Las clases son:

- *LoginView*: esta vista es la que presenta la pantalla de inicio, en donde se iniciará sesión o se registrará el usuario. Dentro de esta vista hay funciones internas que permiten saber si el usuario se ha identificado o creado una cuenta correctamente o si ha ocurrido algún fallo. En caso de identificarse correctamente, se pasará a la siguiente vista, *ContactsView*.
- *ContactsView*: esta vista es la que cargará todos los usuarios que se encuentren en la base de datos y los renderizará para poder comenzar un chat con ellos. Aparecerán botones con los nombres de los usuarios y, en caso de pulsar un botón, se abrirá el chat con la persona correspondiente.
- *ChatView*: es el chat entre dos usuarios, el que está usando la aplicación y al que haya elegido. Dentro de esta vista hay funciones que permiten cargar y guardar mensajes que se hayan recibido o enviado. Los mensajes se cargarán en tiempo real, como es de esperar de una aplicación de chat.

```

1 import SwiftUI
2 import Firebase
3
4 struct LoginView: View {
5
6     @State var login : Bool = true
7     @State var statusMessage : String = ""
8     @State var email : String = ""
9     @State var password : String = ""
10    @State var name : String = ""
11    @State var toContacts : Bool = false
12
13    var body: some View {
14        NavigationView {
15            Color(red: 0.9, green: 0.95, blue: 0.95).ignoresSafeArea()
16                .overlay{
17
18                    ScrollView {
19                        VStack{
20
21                            NavigationLink(
22                                destination: ContactsView()
23                                .navigationBarTitle("")
24                                .navigationBarHidden(true),
25                                isActive: $toContacts
26                            ) {
27                                EmptyView()
28                            }
29
30                            Picker(selection: $login, label: Text("Picker")){
31                                Text("Iniciar sesión")
32                                    .tag(true)
33                                Text("Registrarse")
34                                    .tag(false)
35                            }.pickerStyle(.segmented)
36                                .padding()
37
38                            if login {
39
40                                TextField("Email: ", text: $email).textInputAutocapitalization(.never)
41                                    .padding()
42                                    .background(.white)
43                                    .padding(.bottom, 15)
44                                    .padding(.top, 15)
45                                SecureField("Contraseña: ", text: $password).textInputAutocapitalization(.never)
46                                    .padding()
47                                    .background(.white)
48
49                                Button(
50                                    handleButton()
51                                )label: {
52                                    HStack{
53                                        Spacer()
54                                        Text("Iniciar sesión")
55                                        Spacer()
56                                    }.padding(.vertical, 10)
57                                    .background(Color(red: 1, green: 0.5, blue: 1))
58                                    .cornerRadius(100)
59                                    .foregroundColor(.white).padding(.horizontal, 100)
60                                }.padding(.top, 25)
61
62                                Text(statusMessage)
63                            }
64
65                            else {
66
67                                TextField("Nombre: ", text: $name).textInputAutocapitalization(.never)
68                                    .padding()
69                                    .background(.white)
70                                    .padding(.top, 15)
71
72                                TextField("Email: ", text: $email).textInputAutocapitalization(.never)
73                                    .padding()
74                                    .background(.white)
75                                    .padding(.bottom, 15)
76                                    .padding(.top, 15)
77
78                                SecureField("Contraseña: ", text: $password).textInputAutocapitalization(.never)
79                                    .padding()
80                                    .background(.white)
81
82                                Button(
83                                    handleButton()
84                                )label: {
85                                    VStack{
86                                        Spacer()
87                                        Text("Crear cuenta")
88                                        Spacer()
89                                    }.padding(.vertical, 10)
90                                    .background(color(red: 1, green: 0.5, blue: 1)).foregroundColor(.white)
91                                    .cornerRadius(100)
92                                    .padding(.horizontal, 100)
93                                }.padding(.top, 25)
94
95                                Text(statusMessage)
96                            }
97
98                        }.padding()
99
100                    }.padding()
101
102                }.navigationBarTitle(login ? "Inicio Sesión" : "Registrarse")
103            }
104        }
105
106    private func handleButton() {
107        if login {
108            loginAccount()
109        }else{
110            createAccount()
111        }
112    }
113
114    private func loginAccount () {
115        FirebaseManager.shared.auth.signIn(withEmail: email, password: password){
116            result, err in
117            if let err = err {
118                statusMessage = "\{(err)"
119                return
120            }
121            statusMessage = "Se ha iniciado sesión"
122            toContacts.toggle()
123        }
124    }
125
126    private func createAccount () {
127        FirebaseManager.shared.auth.createUser(withEmail: email, password: password){
128            result, err in
129            if let err = err {
130                statusMessage = "\{(err)"
131                return
132            }
133            statusMessage = "Usuario creado satisfactoriamente"
134            storeUser()
135            toContacts.toggle()
136        }
137    }
138
139    private func storeUser(){
140        guard let uid = FirebaseManager.shared.auth.currentUser?.uid else { return }
141        FirebaseManager.shared.firestore.collection(FirebaseConstants.users).document(uid).setData([FirebaseConstants.name:name, FirebaseConstants.email:email, FirebaseConstants.uid:uid])
142    }
143
144 }

```

```

1  import SwiftUI
2  import Firebase
3
4  class ContactsViewModel : ObservableObject {
5      @Published var users = [ChatUser]()
6      @State var errorMessage = ""
7      @State var currentUser : String = ""
8
9      init(){
10         fetchAllUsers()
11     }
12
13     private func fetchAllUsers(){
14         FirebaseManager.shared.firestore.collection(FirebaseConstants.users)
15             .getDocuments { snapshotDocument, error in
16                 if let error = error{
17                     self.errorMessage = error.localizedDescription
18                     return
19                 }
20                 snapshotDocument?.documents.forEach({ snapshot in
21                     let toAdd = ChatUser(data: snapshot.data())
22                     self.users.append(toAdd)
23                 })
24             }
25     }
26 }
27
28 struct ContactsView: View {
29     @State var logoutOptions : Bool = false
30     @ObservedObject var vm = ContactsViewModel()
31     @State var toChat : Bool = false
32     @State var desiredUser : ChatUser = ChatUser(data: [FirebaseConstants.name:"", FirebaseConstants.email:"", FirebaseConstants.uid:""])
33
34     var body: some View {
35         NavigationView {Color(red: 0.9, green: 0.95, blue: 0.95)
36             .ignoresSafeArea()
37             .overlay(
38
39                 ScrollView {
40
41                     VStack{
42
43                         NavigationLink("", isActive: $toChat){
44                             ChatView(chatUser: desiredUser)
45                             }.labelsHidden().hidden()
46
47                         ForEach(vm.users) { user in
48
49                             if (user.uid != FirebaseManager.shared.auth.currentUser?.uid) {
50                                 Button {
51                                     accessChat(clickedUser: user)
52                                 } label: {
53                                     HStack{
54                                         Spacer()
55                                         Text(user.name).font(.system(size: 23))
56                                         .fontWeight(.semibold)
57                                         .foregroundColor(Color(red: 0/255, green: 101/255, blue: 106/255))
58                                         Spacer()
59                                     }.padding(.vertical, 15)
60                                     .background(Color(red: 202/255, green: 202/255, blue: 226/255))
61                                     .cornerRadius(15)
62                                     .foregroundColor(.white)
63                                     .padding(.horizontal, 20)
64
65                                 }.padding(.top, 25)
66                             }
67                         }
68                     }
69                 }.frame(width: UIScreen.main.bounds.size.width)
70             }
71         }
72     }
73
74     private func accessChat(clickedUser: ChatUser){
75         desiredUser = clickedUser
76         toChat.toggle()
77     }
78
79 }
80
81 }
82
83 }
84 }

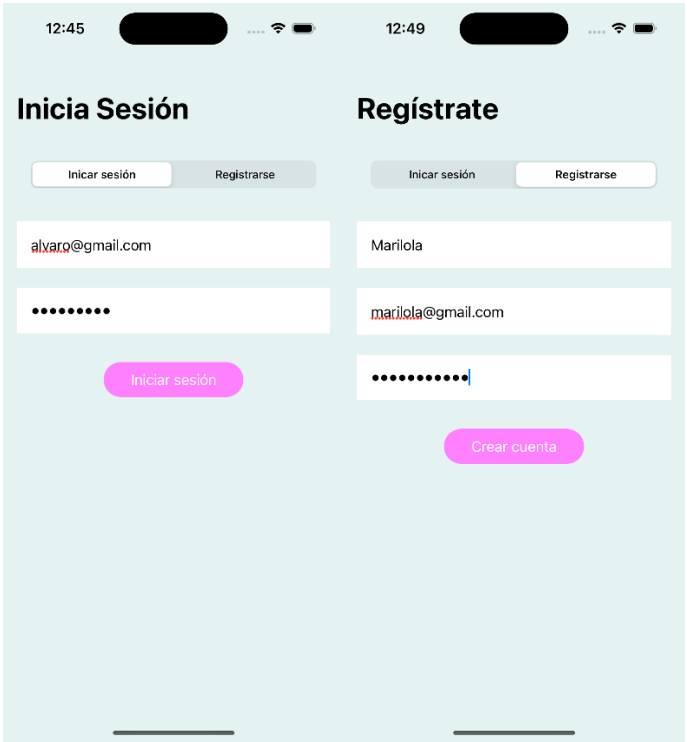
```

```

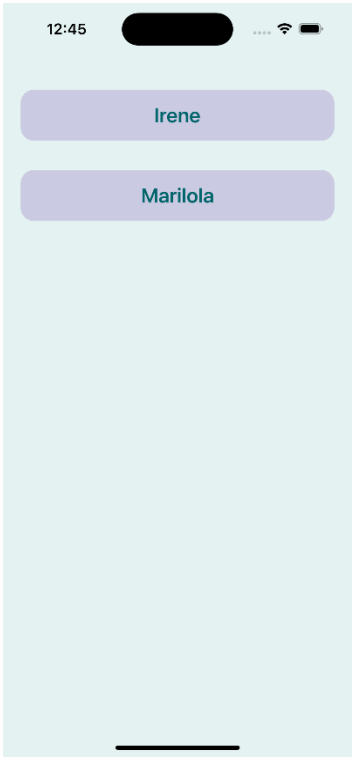
1 import SwiftUI
2 import Firebase
3
4 class ChatModelView : ObservableObject {
5     @State var errorMessage : String = ""
6     let chatUser : ChatUser?
7     @Published var chatLog = [ChatMessage]()
8
9     init(chatUser: ChatUser){
10         self.chatUser = chatUser
11
12         fetchMessages()
13     }
14
15     func handleSend(messageToSend : String){
16         guard let fromId = FirebaseManager.shared.auth.currentUser?.uid else { return }
17         guard let toId = chatUser?.uid else{ return }
18
19         let documentFrom = FirebaseManager.shared.firestore.collection(FirebaseConstants.chats).document(fromId).collection(toId).document()
20
21         let documentTo = FirebaseManager.shared.firestore.collection(FirebaseConstants.chats).document(toId).collection(fromId).document()
22
23         let messageData = [FirebaseConstants.message : messageToSend, FirebaseConstants.from:fromId, FirebaseConstants.to:toId, FirebaseConstants.timestamp:timestamp()] as [String : Any]
24
25         documentFrom.setData(messageData)
26         documentTo.setData(messageData)
27
28     }
29
30     private func fetchMessages() {
31         guard let fromId = FirebaseManager.shared.auth.currentUser?.uid else { return }
32         guard let toId = chatUser?.uid else{ return }
33
34         if toId == "" || toId == nil{
35             return
36         }
37
38         if fromId == "" || fromId == nil{
39             return
40         }
41
42         FirebaseManager.shared.firestore.collection(FirebaseConstants.chats)
43             .document(fromId)
44             .collection(toId)
45             .orderBy("timestamp")
46             .addSnapshotListener { querySnapshot, error in
47                 if let error = error {
48                     self.errorMessage = error.localizedDescription
49                     return
50                 }
51                 self.chatLog.removeAll()
52                 querySnapshot?.documents.forEach({ snapshot in
53                     let data = snapshot.data()
54                     let documentId = snapshot.documentID
55                     self.chatLog.append(.init(documentID: documentId, data: data))
56                 })
57             }
58     }
59 }
60
61 }
62
63 struct ChatView : View {
64     let chatUser : ChatUser?
65     @State var textToSend : String = ""
66     @ObservedObject var cm : ChatModelView
67
68     init(chatUser : ChatUser){
69         self.chatUser = chatUser
70         cm = ChatModelView(chatUser: chatUser)
71     }
72
73     var body: some View {
74         VStack{
75             ScrollView{
76
77                 ForEach (cm.chatLog){ chatMessage in
78                     if chatMessage.from == FirebaseManager.shared.auth.currentUser?.uid{
79                         HStack{
80                             Spacer()
81                             HStack{
82                                 Text(chatMessage.message)
83                                 .foregroundColor(.white)
84                             }.padding()
85                             .background(.blue)
86                             .cornerRadius(100)
87                             .padding(.horizontal, 5)
88                         }
89                     }
90                     else{
91                         HStack{
92                             HStack{
93                                 Text(chatMessage.message)
94                                 .foregroundColor(.black)
95                             }.padding()
96                             .background(.white)
97                             .cornerRadius(100)
98                             .padding(.horizontal, 5)
99                         }
100                         Spacer()
101                     }
102                 }
103             }
104
105             HStack{ Spacer() }
106
107             .background(Color(red: 0.8, green: 0.85, blue: 0.85))
108             .padding(.vertical, 0.5)
109
110             HStack{
111                 TextField("No seas tímido", text: $textToSend)
112                 Button{
113                     cm.handleSend(messageToSend: textToSend)
114                     textToSend = ""
115                 }label: {
116                     Text("enviar")
117                     .padding(.vertical, 10)
118                     .padding(.horizontal, 12)
119                     .background(.blue)
120                     .cornerRadius(100)
121                     .foregroundColor(.white)
122                     .fontWeight(.black)
123                     .font(.system(size: 15))
124                 }
125             }.padding()
126
127         }.navigationTitle(chatUser?.name ?? "")
128         .navigationBarTitleDisplayMode(.inline)
129     }
130 }
131
132 }

```

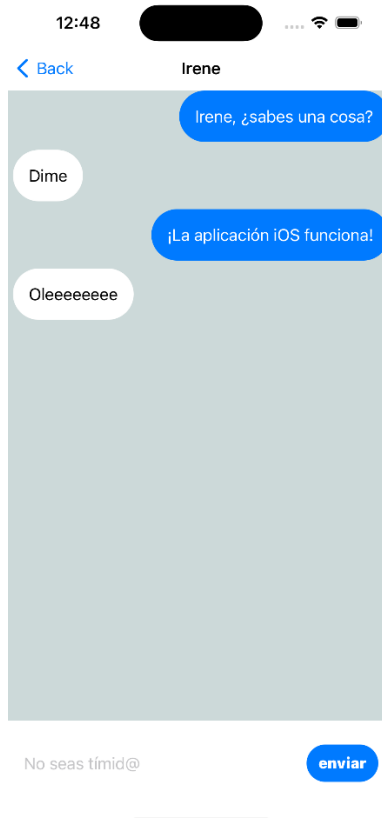
Una vez terminado el código, la aplicación resultante es la siguiente:
INICIAR SESIÓN / REGISTRARSE:



LISTA DE USUARIOS:






CHAT CON UN USUARIO:



FIREBASE:

En cuanto respecta a *Firebase*, se utilizan diferentes aspectos que ya se han mencionado con anterioridad.

Para la autenticación, se utiliza el sistema que proporciona y que, tras hacer pruebas, ha quedado de la siguiente manera:

Buscar por dirección de correo electrónico, número de teléfono o UID de usuario					Agregar usuario		
Identificador	Proveedores	Fecha de creación	↓	Fecha de acceso	UID de usuario		
marilola@gmail.com		6 dic 2022		6 dic 2022	G4cs9QlPPEVo9HGsh8f95hxGbxr1		
irene@gmail.com		6 dic 2022		6 dic 2022	0xasVmks8TeJzJnc4RRb7DmVSd...		
alvaro@gmail.com		6 dic 2022		6 dic 2022	n0xAyz7FpQdxmLwMdU5Tats7ou...		
Filas por página:					50	1 - 3 of 3	

Al crear un usuario, además de registrarse en el apartado de autenticación, como se puede ver en la imagen anterior, también se crea una entrada del mismo en la base de datos:

🏠 > users > G4cs9QIPPEVo9HGsh8f95hxGbxr1 Más funciones en Google Cloud		
chatapplicationios	users	G4cs9QIPPEVo9HGsh8f95hxGbxr1
+ Iniciar colección	+ Agregar documento	+ Iniciar colección
chats	0xasVmks8TeJzJnc4RRb7DmVSdt1	+ Agregar campo
users >	G4cs9QIPPEVo9HGsh8f95hxGbxr1 >	email: "marilola@gmail.com"
	n0xAyz7FpQdxmLwMdU5Tats7ou83	name: "Marilola"
		uid: "G4cs9QIPPEVo9HGsh8f95hxGbxr1"

Por último, para guardar los mensajes, se crea una entrada ubicada en `chats -> IDusuario1 -> IDusuario2` y en `chats -> IDusuario2 -> IDusuario1` para que, de esta manera, cada usuario tenga un registro de la conversación:

🏠 > chats > n0xAyz7FpQdxmLwMdU5Tats7ou83 > 0xasVmks8TeJzJnc4RRb7DmVSdt1 > UMwAwqQvhLwLnqcxEth Más funciones en Google Cloud		
n0xAyz7FpQdxmLwMdU5Tats7ou83	0xasVmks8TeJzJnc4RRb7DmVSdt1	UMwAwqQvhLwLnqcxEth
+ Iniciar colección	+ Agregar documento	+ Iniciar colección
0xasVmks8TeJzJnc4RRb7DmVSdt1 >	IN0Q2wrfvjXUzKJyAYys	+ Agregar campo
	NP217R7XmvMdHAcZjn98	from: "n0xAyz7FpQdxmLwMdU5Tats7ou83"
	UMwAwqQvhLwLnqcxEth >	message: "¡La aplicación iOS funciona!"
	sy1mdYkfxzI80d4ESfEi	timestamp: 6 de diciembre de 2022, 12:47:56 UTC
		to: "0xasVmks8TeJzJnc4RRb7DmVSdt1"
+ Agregar campo		

Temporización

Para este proyecto han sido invertidas, aproximadamente, 16 horas totales, las cuales han sido producto de un fin de semana de trabajo intenso.

Herramientas

Las herramientas usadas para el desarrollo de la aplicación han sido:

- *Firebase*: herramienta gratuita que proporciona métodos de autenticación y almacenamiento.
- *Xcode*: entorno de desarrollo con el cual se ha podido llevar a cabo la producción del *software* en *SwiftUI*.

3. Conclusiones

Luego de finalizar la aplicación, nos hemos sentido muy satisfechos con el trabajo realizado, aprendiendo *Swift* en el proceso.

Hemos podido determinar también, aunque de manera subjetiva, la facilidad de utilizar *SwiftUI* frente a *Kotlin*. Pese a esto, también podemos nombrar cosas negativas de este lenguaje, siendo los puntos negativos de mayor peso el solo poder producir aplicaciones si se dispone de un dispositivo *Apple* y que, como consecuencia de ello, solo se pueden producir aplicaciones para dispositivos *Apple*.

Con esto concluye el informe de la aplicación para *iOS*. Gracias por leer y esperamos que la aplicación haya sido de su agrado.

4. Bibliografía

- i. [SwiftUI Firebase Building Real Time Chat with Firestore - YouTube](#)
- ii. [SwiftUI | Apple Developer Documentation](#)
- iii. [Hacking with Swift – learn to code iPhone and iPad apps with free Swift tutorials](#)