**import** SwiftUI

**import** FirebaseFirestore

**struct** ClientListView: View {

@EnvironmentObject **var** authManager: AuthManager

@State **private** **var** clients: [Client] = []

@State **private** **var** errorMessage: String = ""

@State **private** **var** showingAddClientSheet = **false**

@State **private** **var** newClient = Client(

id: **nil**,

typ: "Spieler",

name: "",

vorname: "",

geschlecht: "männlich",

vereinID: **nil**,

nationalitaet: [],

geburtsdatum: **nil**,

liga: **nil**,

profilbildURL: **nil**

)

@State **private** **var** clubs: [Club] = [] // Für Vereinslogos

@State **private** **var** imageCache: [String: UIImage] = [:] // Cache für Vereinslogos

**var** body: **some** View {

NavigationStack {

List {

ForEach(clients, id: \.**self**) { client **in**

NavigationLink(destination: ClientView(client: client)) {

HStack(spacing: 10) {

// Vereinslogo

**if** **let** vereinID = client.vereinID, **let** club = clubs.first(where: { $0.name == vereinID }), **let** logoURL = club.logoURL, **let** url = URL(string: logoURL), **let** cachedImage = imageCache[logoURL] {

Image(uiImage: cachedImage)

.resizable()

.scaledToFit()

.frame(width: 20, height: 20)

.clipShape(Circle())

} **else** {

Image(systemName: "building.fill")

.resizable()

.scaledToFit()

.frame(width: 20, height: 20)

.foregroundColor(.gray)

.clipShape(Circle())

.onAppear {

loadClubLogo(for: client.vereinID) // Korrigiert: client.vereinID

}

}

// Profilbild und Name

**if** **let** profilbildURL = client.profilbildURL, **let** url = URL(string: profilbildURL), **let** cachedImage = imageCache[profilbildURL] {

Image(uiImage: cachedImage)

.resizable()

.scaledToFill()

.frame(width: 40, height: 40)

.clipShape(Circle())

.overlay(Circle().stroke(Color.gray, lineWidth: 1))

} **else** {

Image(systemName: "person.circle.fill")

.resizable()

.scaledToFill()

.frame(width: 40, height: 40)

.foregroundColor(.gray)

.clipShape(Circle())

.overlay(Circle().stroke(Color.gray, lineWidth: 1))

.onAppear {

loadImage(for: client)

}

}

VStack(alignment: .leading, spacing: 2) {

Text("\(client.vorname) \(client.name)")

.font(.headline)

.scaleEffect(0.7)

**if** **let** geburtsdatum = client.geburtsdatum {

Text(dateFormatter.string(from: geburtsdatum))

.font(.caption)

}

**if** **let** liga = client.liga {

Text(liga)

.font(.caption)

}

}

.frame(maxWidth: .infinity, alignment: .leading)

Text(client.typ == "Spieler" ? "♂" : "♀")

.font(.system(size: 14))

.foregroundColor(client.typ == "Spieler" ? .blue : .pink)

}

.padding(.vertical, 5)

}

.swipeActions {

Button(role: .destructive) {

deleteClient(client)

} label: {

Label("Löschen", systemImage: "trash")

}

}

}

}

.navigationTitle("Klienten verwalten")

.toolbar {

ToolbarItem(placement: .navigationBarTrailing) {

Button(action: { showingAddClientSheet = **true** }) {

Image(systemName: "plus")

}

}

}

.sheet(isPresented: $showingAddClientSheet) {

AddClientView(

client: $newClient,

isEditing: **false**,

onSave: { updatedClient **in**

FirestoreManager.shared.createClient(client: updatedClient) { result **in**

**switch** result {

**case** .success:

loadClients()

newClient = Client(

id: **nil**,

typ: "Spieler",

name: "",

vorname: "",

geschlecht: "männlich",

vereinID: **nil**,

nationalitaet: [],

geburtsdatum: **nil**,

liga: **nil**,

profilbildURL: **nil**

)

**case** .failure(**let** error):

errorMessage = "Fehler beim Speichern: \(error.localizedDescription)"

}

}

showingAddClientSheet = **false**

},

onCancel: {

newClient = Client(

id: **nil**,

typ: "Spieler",

name: "",

vorname: "",

geschlecht: "männlich",

vereinID: **nil**,

nationalitaet: [],

geburtsdatum: **nil**,

liga: **nil**,

profilbildURL: **nil**

)

showingAddClientSheet = **false**

}

)

}

.alert(isPresented: .constant(!errorMessage.isEmpty)) {

Alert(title: Text("Fehler"), message: Text(errorMessage), dismissButton: .default(Text("OK")) {

errorMessage = ""

})

}

.onAppear {

loadClients()

loadClubs()

}

}

}

**private** **let** dateFormatter: DateFormatter = {

**let** formatter = DateFormatter()

formatter.dateStyle = .medium

**return** formatter

}()

**private** **func** loadClients() {

FirestoreManager.shared.getClients { result **in**

**switch** result {

**case** .success(**let** loadedClients):

clients = loadedClients

**for** client **in** loadedClients {

**if** **let** profilbildURL = client.profilbildURL {

loadImage(for: client)

}

}

**case** .failure(**let** error):

errorMessage = "Fehler beim Laden der Klienten: \(error.localizedDescription)"

}

}

}

**private** **func** loadClubs() {

FirestoreManager.shared.getClubs { result **in**

**switch** result {

**case** .success(**let** loadedClubs):

clubs = loadedClubs

**for** club **in** loadedClubs {

**if** **let** logoURL = club.logoURL {

loadClubLogo(for: club.name)

}

}

**case** .failure(**let** error):

print("Fehler beim Laden der Vereine: \(error.localizedDescription)")

}

}

}

**private** **func** deleteClient(\_ client: Client) {

**guard** **let** id = client.id **else** { **return** }

FirestoreManager.shared.deleteClient(clientID: id) { result **in**

**switch** result {

**case** .success:

loadClients()

**case** .failure(**let** error):

errorMessage = "Fehler beim Löschen des Klienten: \(error.localizedDescription)"

}

}

}

**private** **func** loadImage(for client: Client) {

**if** **let** profilbildURL = client.profilbildURL, **let** url = URL(string: profilbildURL) {

URLSession.shared.dataTask(with: url) { data, response, error **in**

**if** **let** data = data, **let** image = UIImage(data: data) {

DispatchQueue.main.async {

**self**.imageCache[profilbildURL] = image

}

}

}.resume()

}

}

**private** **func** loadClubLogo(for vereinID: String?) {

**guard** **let** vereinID = vereinID, **let** club = clubs.first(where: { $0.name == vereinID }), **let** logoURL = club.logoURL, **let** url = URL(string: logoURL) **else** { **return** }

URLSession.shared.dataTask(with: url) { data, response, error **in**

**if** **let** data = data, **let** image = UIImage(data: data) {

DispatchQueue.main.async {

**self**.imageCache[logoURL] = image

}

}

}.resume()

}

}

#Preview {

ClientListView()

.environmentObject(AuthManager())

}