**import** SwiftUI

**import** FirebaseFirestore

**import** FirebaseStorage

**struct** ClientView: View {

@EnvironmentObject **var** authManager: AuthManager

@Binding **var** client: Client

@State **private** **var** image: UIImage? = **nil**

@State **private** **var** showingImagePicker = **false**

@State **private** **var** showingEditSheet = **false**

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

@State **private** **var** isLoadingImage = **false**

@State **private** **var** navigateToContract = **false**

@State **private** **var** activities: [Activity] = []

@State **private** **var** club: Club?

@State **private** **var** isFavorite = **false**

@State **private** **var** imageCache: [String: UIImage] = [:] // Korrigierte Initialisierung

**var** previousClientAction: (() -> Void)?

**var** nextClientAction: (() -> Void)?

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

**let** formatter = DateFormatter()

formatter.dateStyle = .medium

**return** formatter

}()

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

NavigationStack {

ScrollView {

VStack(spacing: 15) {

VStack(spacing: 10) {

**if** isLoadingImage {

ProgressView("Lade Bild...")

.frame(width: 100, height: 100)

} **else** **if** **let** image = image ?? imageCache[client.profilbildURL ?? ""] {

Image(uiImage: image)

.resizable()

.scaledToFit()

.frame(width: 100, height: 100)

.clipShape(Circle())

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

} **else** {

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

.resizable()

.scaledToFit()

.frame(width: 100, height: 100)

.foregroundColor(.gray)

.clipShape(Circle())

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

}

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

Image(systemName: "arrow.up.circle")

.font(.system(size: 24))

.foregroundColor(.blue)

.opacity(0.6)

}

.sheet(isPresented: $showingImagePicker) {

ImagePicker(selectedImage: $image, isPresented: $showingImagePicker) { selectedImage **in**

uploadImage(selectedImage)

}

}

Text("\(client.vorname) \(client.name)\(calculateAge().map { ", \($0) Jahre" } ?? "")")

.font(.title2)

.bold()

.multilineTextAlignment(.center)

**if** **let** vereinID = client.vereinID {

HStack {

**if** **let** club = club, **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)

.onAppear { loadClubLogo() }

}

Text(vereinID)

.font(.headline)

}

}

}

.padding()

HStack(spacing: 20) {

VStack(alignment: .leading, spacing: 5) {

Text("Nationalitäten")

.font(.subheadline)

.foregroundColor(.gray)

**if** **let** nationalitaeten = client.nationalitaet, !nationalitaeten.isEmpty {

ForEach(nationalitaeten, id: \.**self**) { nat **in**

flagAndCode(for: nat)

}

} **else** {

Text("Keine")

.foregroundColor(.secondary)

}

}

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

VStack(alignment: .leading, spacing: 5) {

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

HStack(spacing: 4) {

Image(systemName: "calendar")

.foregroundColor(.gray)

Text(dateFormatter.string(from: geburtsdatum))

}

}

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

HStack(spacing: 4) {

Image(systemName: "tshirt")

.foregroundColor(.gray)

Text(liga)

}

}

}

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

}

.padding(.horizontal)

Form {

Section(header: Text("Wichtige Daten")) {

labeledField(label: "Vertragslaufzeit", value: client.vertragBis.map { dateFormatter.string(from: $0) })

labeledField(label: "Option", value: client.vertragsOptionen)

labeledField(label: "Gehalt", value: client.gehalt.map { String(format: "%.2f €", $0) })

Button("Zum Vertrag") {

**if** client.id != **nil** {

navigateToContract = **true**

} **else** {

errorMessage = "Kein Client-ID verfügbar"

}

}

.foregroundColor(.white)

.frame(maxWidth: .infinity)

.padding()

.background(Color.green)

.cornerRadius(8)

}

Section(header: Text("Kontaktinformationen")) {

**if** **let** phone = client.kontaktTelefon {

HStack {

labeledField(label: "Telefon", value: phone)

Spacer()

Button(action: { openURL("tel:\(phone)") }) {

Image(systemName: "phone.fill")

.foregroundColor(.blue)

}

}

}

**if** **let** email = client.kontaktEmail {

HStack {

labeledField(label: "E-Mail", value: email)

Spacer()

Button(action: { openURL("mailto:\(email)") }) {

Image(systemName: "envelope.fill")

.foregroundColor(.blue)

}

}

}

labeledField(label: "Adresse", value: client.adresse)

}

Section(header: Text("Spieldaten")) {

labeledField(label: "Positionen", value: client.positionFeld?.joined(separator: ", "))

labeledField(label: "Starker Fuß", value: client.starkerFuss)

labeledField(label: "Grösse", value: client.groesse.map { "\($0) cm" })

labeledField(label: "Schuhmarke", value: client.schuhmarke)

labeledField(label: "Nationalmannschaft", value: client.nationalmannschaft)

}

Section(header: Text("Letzte Aktivitäten")) {

**if** activities.isEmpty {

Text("Keine Aktivitäten vorhanden.")

.foregroundColor(.gray)

} **else** {

ForEach(activities.prefix(3)) { activity **in**

NavigationLink(destination: ActivityDetailView(activity: activity)) {

VStack(alignment: .leading) {

Text(activity.description)

.lineLimit(1)

Text(activity.timestamp, style: .date)

.font(.subheadline)

.foregroundColor(.gray)

}

}

}

}

}

}

}

}

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

.navigationBarTitleDisplayMode(.inline)

.toolbar {

ToolbarItem(placement: .navigationBarLeading) {

HStack {

**if** **let** previous = previousClientAction {

Button(action: previous) {

Image(systemName: "chevron.left")

.foregroundColor(.blue)

}

}

**if** **let** next = nextClientAction {

Button(action: next) {

Image(systemName: "chevron.right")

.foregroundColor(.blue)

}

}

}

}

ToolbarItem(placement: .navigationBarTrailing) {

HStack {

Text(genderSymbol)

.font(.system(size: 14))

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

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

Image(systemName: "pencil")

.foregroundColor(.blue)

}

Button(action: { toggleFavorite() }) {

Image(systemName: isFavorite ? "star.fill" : "star")

.foregroundColor(isFavorite ? .yellow : .gray)

}

}

}

}

.sheet(isPresented: $showingEditSheet) {

EditClientView(client: $client, onSave: { updatedClient **in**

client = updatedClient

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

**switch** result {

**case** .success:

print("Client erfolgreich aktualisiert")

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

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

}

}

showingEditSheet = **false**

}, onCancel: {

showingEditSheet = **false**

})

}

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

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

errorMessage = ""

})

}

.onAppear {

loadImage()

loadActivities()

loadClub()

loadFavoriteStatus()

}

.navigationDestination(isPresented: $navigateToContract) {

**if** **let** clientID = client.id {

ContractView(clientID: clientID)

} **else** {

Text("Fehler: Kein Client-ID")

}

}

}

}

**struct** EditClientView: View {

@Binding **var** client: Client

**var** onSave: (Client) -> Void

**var** onCancel: () -> Void

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

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

@State **private** **var** vereinID: String? = **nil**

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

@State **private** **var** vertragBis: Date = Date()

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

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

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

@State **private** **var** nationalitaet: [String] = []

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

@State **private** **var** positionFeld: [String] = []

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

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

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

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

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

@State **private** **var** geburtsdatum: Date = Date()

@State **private** **var** clubOptions: [String] = []

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

NavigationView {

Form {

Section(header: Text("Klientendaten")) {

TextField("Vorname", text: $vorname)

TextField("Name", text: $name)

DatePicker("Geburtsdatum", selection: $geburtsdatum, displayedComponents: .date)

Picker("Verein", selection: $vereinID) {

Text("Kein Verein").tag(String?.none)

ForEach(clubOptions, id: \.**self**) { club **in**

Text(club).tag(String?.some(club))

}

}

.pickerStyle(.menu)

TextField("Liga", text: $liga)

DatePicker("Vertragslaufzeit", selection: $vertragBis, displayedComponents: .date)

TextField("Vertragsoptionen", text: $vertragsOptionen)

TextField("Gehalt (€)", text: $gehalt)

.keyboardType(.decimalPad)

TextField("Grösse (cm)", text: $groesse)

.keyboardType(.numberPad)

TextField("Nationalitäten (durch Komma getrennt)", text: Binding(

get: { nationalitaet.joined(separator: ", ") },

set: { nationalitaet = $0.split(separator: ",").map { $0.trimmingCharacters(in: .whitespaces) } }

))

TextField("Nationalmannschaft", text: $nationalmannschaft)

TextField("Positionen (durch Komma getrennt)", text: Binding(

get: { positionFeld.joined(separator: ", ") },

set: { positionFeld = $0.split(separator: ",").map { $0.trimmingCharacters(in: .whitespaces) } }

))

TextField("Schuhmarke", text: $schuhmarke)

Picker("Starker Fuß", selection: $starkerFuss) {

Text("Nicht angegeben").tag("")

Text("rechts").tag("rechts")

Text("links").tag("links")

Text("beide").tag("beide")

}

.pickerStyle(.menu)

}

Section(header: Text("Kontaktinformationen")) {

TextField("Telefon", text: $kontaktTelefon)

TextField("E-Mail", text: $kontaktEmail)

TextField("Adresse", text: $adresse)

}

}

.navigationTitle("Profil bearbeiten")

.toolbar {

ToolbarItem(placement: .cancellationAction) {

Button("Abbrechen") { onCancel() }

}

ToolbarItem(placement: .confirmationAction) {

Button("Speichern") {

updateClient()

onSave(client)

}

}

}

.onAppear {

loadClientData()

loadClubOptions()

}

}

}

**private** **func** loadClientData() {

vorname = client.vorname

name = client.name

vereinID = client.vereinID

liga = client.liga ?? ""

vertragBis = client.vertragBis ?? Date()

vertragsOptionen = client.vertragsOptionen ?? ""

gehalt = client.gehalt != **nil** ? String(client.gehalt!) : ""

groesse = client.groesse != **nil** ? String(client.groesse!) : ""

nationalitaet = client.nationalitaet ?? []

nationalmannschaft = client.nationalmannschaft ?? ""

positionFeld = client.positionFeld ?? []

schuhmarke = client.schuhmarke ?? ""

starkerFuss = client.starkerFuss ?? ""

kontaktTelefon = client.kontaktTelefon ?? ""

kontaktEmail = client.kontaktEmail ?? ""

adresse = client.adresse ?? ""

geburtsdatum = client.geburtsdatum ?? Date()

}

**private** **func** loadClubOptions() {

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

**switch** result {

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

DispatchQueue.main.async {

**self**.clubOptions = clubs.filter { club **in**

**if** client.geschlecht == "männlich" {

**return** club.abteilung == "Männer"

} **else** **if** client.geschlecht == "weiblich" {

**return** club.abteilung == "Frauen"

}

**return** **false**

}.map { $0.name }

**if** **let** currentVereinID = client.vereinID, !clubOptions.contains(currentVereinID) {

**self**.clubOptions.insert(currentVereinID, at: 0)

}

**self**.vereinID = client.vereinID

}

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

DispatchQueue.main.async {

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

}

}

}

}

**private** **func** updateClient() {

client.vorname = vorname.isEmpty ? client.vorname : vorname

client.name = name.isEmpty ? client.name : name

client.vereinID = vereinID

client.liga = liga.isEmpty ? **nil** : liga

client.vertragBis = vertragBis

client.vertragsOptionen = vertragsOptionen.isEmpty ? **nil** : vertragsOptionen

client.gehalt = Double(gehalt) ?? **nil**

client.groesse = Int(groesse) ?? **nil**

client.nationalitaet = nationalitaet.isEmpty ? **nil** : nationalitaet

client.nationalmannschaft = nationalmannschaft.isEmpty ? **nil** : nationalmannschaft

client.positionFeld = positionFeld.isEmpty ? **nil** : positionFeld

client.schuhmarke = schuhmarke.isEmpty ? **nil** : schuhmarke

client.starkerFuss = starkerFuss.isEmpty ? **nil** : starkerFuss

client.kontaktTelefon = kontaktTelefon.isEmpty ? **nil** : kontaktTelefon

client.kontaktEmail = kontaktEmail.isEmpty ? **nil** : kontaktEmail

client.adresse = adresse.isEmpty ? **nil** : adresse

client.geburtsdatum = geburtsdatum

}

}

**private** **func** flagAndCode(for nationalitaet: String) -> **some** View {

**let** code = countryCode(for: nationalitaet) ?? "??"

**let** isCustomImage = (UIImage(named: code.lowercased()) != **nil**)

**return** HStack(spacing: 2) {

**if** isCustomImage {

Image(uiImage: UIImage(named: code.lowercased())!)

.resizable()

.scaledToFit()

.frame(width: 12, height: 8)

} **else** {

Image(systemName: "flag")

.resizable()

.scaledToFit()

.frame(width: 12, height: 8)

.foregroundColor(.gray)

}

Text(code)

.font(.caption2)

}

}

**private** **func** countryCode(for nationalitaet: String) -> String? {

**let** countryMap: [String: String] = [

"Deutschland": "DE",

"Österreich": "AT",

"Schweiz": "CH",

"Frankreich": "FR",

"Italien": "IT",

"Spanien": "ES",

"England": "GB",

"USA": "US"

]

**return** countryMap[nationalitaet]

}

**private** **func** labeledField(label: String, value: String?) -> **some** View {

HStack {

Text(label)

.font(.subheadline)

.foregroundColor(.gray)

Spacer()

Text(value ?? "")

.padding(.horizontal, 8)

.padding(.vertical, 4)

.background(value != **nil** ? Color(.systemGray6) : Color.clear)

.cornerRadius(4)

.foregroundColor(.black)

}

}

**private** **var** genderSymbol: String {

**switch** client.typ {

**case** "Spieler": **return** "♂"

**case** "Spielerin": **return** "♀"

**default**: **return** ""

}

}

**private** **func** calculateAge() -> Int? {

**guard** **let** birthDate = client.geburtsdatum **else** { **return** **nil** }

**let** calendar = Calendar.current

**let** now = Date()

**let** ageComponents = calendar.dateComponents([.year], from: birthDate, to: now)

**return** ageComponents.year

}

**private** **func** loadImage() {

**guard** **let** urlString = client.profilbildURL, **let** url = URL(string: urlString) **else** {

isLoadingImage = **false**

**return**

}

**if** **let** cachedImage = imageCache[urlString] {

image = cachedImage

isLoadingImage = **false**

} **else** {

isLoadingImage = **true**

URLSession.shared.dataTask(with: url) { data, \_, \_ **in**

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

DispatchQueue.main.async {

**self**.image = loadedImage

**self**.imageCache[urlString] = loadedImage

**self**.isLoadingImage = **false**

}

} **else** {

DispatchQueue.main.async { **self**.isLoadingImage = **false** }

}

}.resume()

}

}

**private** **func** uploadImage(\_ selectedImage: UIImage) {

isLoadingImage = **true**

**guard** **let** clientID = client.id **else** {

errorMessage = "Kein Client-ID verfügbar"

isLoadingImage = **false**

**return**

}

FirestoreManager.shared.uploadProfileImage(documentID: clientID, image: selectedImage, collection: "profile\_images") { result **in**

**switch** result {

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

client.profilbildURL = url

FirestoreManager.shared.updateClient(client: client) { result **in**

DispatchQueue.main.async {

isLoadingImage = **false**

**if** **case** .failure(**let** error) = result {

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

} **else** {

loadImage()

}

}

}

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

DispatchQueue.main.async {

isLoadingImage = **false**

errorMessage = "Fehler beim Hochladen des Bildes: \(error.localizedDescription)"

}

}

}

}

**private** **func** loadActivities() {

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

FirestoreManager.shared.getActivities(forClientID: clientID) { result **in**

**switch** result {

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

DispatchQueue.main.async {

**self**.activities = loadedActivities

}

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

DispatchQueue.main.async {

**self**.errorMessage = "Fehler beim Laden der Aktivitäten: \(error.localizedDescription)"

}

}

}

}

**private** **func** loadClub() {

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

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

**switch** result {

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

**if** **let** matchingClub = clubs.first(where: { $0.name == vereinID }) {

DispatchQueue.main.async {

**self**.club = matchingClub

}

}

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

DispatchQueue.main.async {

**self**.errorMessage = "Fehler beim Laden des Vereins: \(error.localizedDescription)"

}

}

}

}

**private** **func** loadClubLogo() {

**guard** **let** vereinID = client.vereinID, **let** club = club, **let** logoURL = club.logoURL, **let** url = URL(string: logoURL) **else** { **return** }

**if** imageCache[logoURL] == **nil** {

URLSession.shared.dataTask(with: url) { data, \_, \_ **in**

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

DispatchQueue.main.async {

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

}

}

}.resume()

}

}

**private** **func** toggleFavorite() {

**guard** **let** userID = authManager.userID, **let** clientID = client.id **else** { **return** }

isFavorite.toggle()

**var** favorites = (**try**? JSONDecoder().decode(Set<String>.**self**, from: UserDefaults.standard.data(forKey: "favorites\_\(userID)") ?? Data())) ?? Set<String>()

**if** isFavorite {

favorites.insert(clientID)

} **else** {

favorites.remove(clientID)

}

**if** **let** data = **try**? JSONEncoder().encode(favorites) {

UserDefaults.standard.set(data, forKey: "favorites\_\(userID)")

}

}

**private** **func** loadFavoriteStatus() {

**guard** **let** userID = authManager.userID, **let** clientID = client.id **else** { **return** }

**if** **let** data = UserDefaults.standard.data(forKey: "favorites\_\(userID)"),

**let** favorites = **try**? JSONDecoder().decode(Set<String>.**self**, from: data) {

isFavorite = favorites.contains(clientID)

}

}

**private** **func** openURL(\_ urlString: String) {

**if** **let** url = URL(string: urlString), UIApplication.shared.canOpenURL(url) {

UIApplication.shared.open(url)

}

}

**struct** ImagePicker: UIViewControllerRepresentable {

@Binding **var** selectedImage: UIImage?

@Binding **var** isPresented: Bool

**var** onImageSelected: (UIImage) -> Void

**func** makeUIViewController(context: Context) -> UIImagePickerController {

**let** picker = UIImagePickerController()

picker.delegate = context.coordinator

picker.sourceType = .photoLibrary

**return** picker

}

**func** updateUIViewController(\_ uiViewController: UIImagePickerController, context: Context) {}

**func** makeCoordinator() -> Coordinator {

Coordinator(**self**)

}

**class** Coordinator: NSObject, UINavigationControllerDelegate, UIImagePickerControllerDelegate {

**let** parent: ImagePicker

**init**(\_ parent: ImagePicker) {

**self**.parent = parent

}

**func** imagePickerController(\_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey: **Any**]) {

**if** **let** uiImage = info[.originalImage] **as**? UIImage {

parent.selectedImage = uiImage

parent.onImageSelected(uiImage)

}

parent.isPresented = **false**

}

**func** imagePickerControllerDidCancel(\_ picker: UIImagePickerController) {

parent.isPresented = **false**

}

}

}

}

#Preview {

ClientView(client: .constant(Client(

id: "1",

typ: "Spieler",

name: "Müller",

vorname: "Thomas",

geschlecht: "männlich",

vereinID: "Bayern Munich",

nationalitaet: ["Deutschland"],

geburtsdatum: Date().addingTimeInterval(-25 \* 365 \* 24 \* 60 \* 60),

liga: "1. Bundesliga"

)))

.environmentObject(AuthManager())

}