**import** SwiftUI

**import** FirebaseFirestore

**struct** EmployeeView: View {

@EnvironmentObject **var** authManager: AuthManager

@State **private** **var** contractCount = 0

@State **private** **var** matchCount = 0

@State **private** **var** transferCount = 0

@State **private** **var** clientCount = 0

@State **private** **var** searchText = ""

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

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

@State **private** **var** showingLeadSheet = **false**

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

@State **private** **var** showingAddClubSheet = **false**

@State **private** **var** showingAddTransferSheet = **false**

@State **private** **var** showingAddFunktionärSheet = **false**

@State **private** **var** showingSearchSheet = **false**

@State **private** **var** showingContactsSheet = **false**

@State **private** **var** selectedTab = 0

@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** newClub = Club(name: "")

@State **private** **var** newTransfer = Transfer()

@State **private** **var** newFunktionär = Funktionär(

id: **nil**,

name: "",

vorname: "",

kontaktTelefon: **nil**,

kontaktEmail: **nil**,

adresse: **nil**,

clients: **nil**

)

// Definition der Tabs mit Titel, Symbol und zugehöriger Ansicht

**private** **let** tabs: [(title: String, icon: String, view: AnyView)] = [

("Home", "house", AnyView(HomeView())),

("Klienten", "person.2", AnyView(ClientListView())),

("Verträge", "doc.text", AnyView(ContractListView())),

("Vereine", "building.2", AnyView(ClubListView())),

("Transfers", "arrow.left.arrow.right", AnyView(TransferListView())),

("Spiele", "sportscourt", AnyView(MatchListView())),

("Sponsoren", "dollarsign.circle", AnyView(SponsorListView())),

("Funktionäre", "person.badge.key", AnyView(FunktionärListView())),

("Chat", "bubble.left", AnyView(ChatView()))

]

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

VStack(spacing: 0) {

// Scrollbare Tab-Leiste

ScrollView(.horizontal, showsIndicators: **false**) {

HStack(spacing: 20) {

ForEach(tabs.indices, id: \.**self**) { index **in**

Button(action: {

withAnimation(.easeInOut(duration: 0.3)) {

selectedTab = index

}

}) {

Label(tabs[index].title, systemImage: tabs[index].icon)

.foregroundColor(selectedTab == index ? .blue : .gray)

.padding(.vertical, 8)

.padding(.horizontal, 12)

.background(selectedTab == index ? Color.blue.opacity(0.1) : Color.clear)

.cornerRadius(8)

}

}

}

.padding(.horizontal)

}

.background(Color(.systemGray6))

// Hauptansicht basierend auf dem ausgewählten Tab mit Animation

ZStack {

ForEach(tabs.indices, id: \.**self**) { index **in**

tabs[index].view

.environmentObject(authManager)

.opacity(selectedTab == index ? 1 : 0)

.animation(.easeInOut(duration: 0.3), value: selectedTab)

}

}

.frame(maxWidth: .infinity, maxHeight: .infinity)

Spacer()

}

.toolbar {

ToolbarItem(placement: .navigationBarTrailing) {

Menu {

Button(action: { authManager.signOut() }) {

Label("Logout", systemImage: "rectangle.portrait.and.arrow.right")

}

} label: {

Image(systemName: "ellipsis.circle")

}

}

}

.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):

print("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**

}

)

}

.sheet(isPresented: $showingAddClubSheet) {

AddClubView(

club: $newClub,

onSave: { updatedClub **in**

FirestoreManager.shared.createClub(club: updatedClub) { result **in**

**switch** result {

**case** .success:

loadClients()

newClub = Club(name: "")

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

print("Fehler beim Speichern: \(error.localizedDescription)")

}

}

showingAddClubSheet = **false**

},

onCancel: {

newClub = Club(name: "")

showingAddClubSheet = **false**

}

)

}

.sheet(isPresented: $showingAddTransferSheet) {

Text("Transfer anlegen (Platzhalter)")

.onDisappear {

showingAddTransferSheet = **false**

}

}

.sheet(isPresented: $showingAddFunktionärSheet) {

AddFunktionärView(

funktionär: $newFunktionär,

onSave: { updatedFunktionär **in**

FirestoreManager.shared.createFunktionär(funktionär: updatedFunktionär) { result **in**

**switch** result {

**case** .success:

newFunktionär = Funktionär(

id: **nil**,

name: "",

vorname: "",

kontaktTelefon: **nil**,

kontaktEmail: **nil**,

adresse: **nil**,

clients: **nil**

)

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

print("Fehler beim Speichern: \(error.localizedDescription)")

}

}

showingAddFunktionärSheet = **false**

},

onCancel: {

newFunktionär = Funktionär(

id: **nil**,

name: "",

vorname: "",

kontaktTelefon: **nil**,

kontaktEmail: **nil**,

adresse: **nil**,

clients: **nil**

)

showingAddFunktionärSheet = **false**

}

)

}

.sheet(isPresented: $showingSearchSheet) {

Text("Appübergreifende Suche (Platzhalter)")

.onDisappear {

showingSearchSheet = **false**

}

}

.sheet(isPresented: $showingLeadSheet) {

ClientContactView(authManager: authManager)

}

.sheet(isPresented: $showingContactsSheet) {

ContactsView()

.environmentObject(authManager)

.onDisappear {

showingContactsSheet = **false**

}

}

.overlay(

VStack {

Spacer()

HStack(spacing: 0) { // Entfernt ScrollView, verwendet festes HStack

Button(action: { withAnimation(.easeInOut(duration: 0.3)) { selectedTab = 0 } }) {

VStack {

Image(systemName: "house")

.font(.title2)

Text("Home")

.font(.caption)

.lineLimit(1)

}

.foregroundColor(selectedTab == 0 ? .blue : .gray)

.frame(maxWidth: .infinity) // Gleichmäßige Breite

}

Menu {

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

Label("Lead (neuer Lead)", systemImage: "envelope")

}

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

Label("Klient (neuen Klienten)", systemImage: "person.crop.circle.badge.plus")

}

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

Label("Verein (neuer Verein)", systemImage: "building.2.fill")

}

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

Label("Transfer (neuer Transfer)", systemImage: "arrow.left.arrow.right.circle")

}

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

Label("Funktionär (neuer Funktionär)", systemImage: "person.badge.key")

}

} label: {

VStack {

Image(systemName: "plus.circle")

.font(.title2)

Text("Create")

.font(.caption)

.lineLimit(1)

}

.foregroundColor(.blue)

.frame(maxWidth: .infinity) // Gleichmäßige Breite

}

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

VStack {

Image(systemName: "magnifyingglass")

.font(.title2)

Text("Suche")

.font(.caption)

.lineLimit(1)

}

.foregroundColor(.gray)

.frame(maxWidth: .infinity) // Gleichmäßige Breite

}

Button(action: { withAnimation(.easeInOut(duration: 0.3)) { selectedTab = tabs.count - 1 } }) {

VStack {

Image(systemName: "bubble.left")

.font(.title2)

Text("Chat")

.font(.caption)

.lineLimit(1)

}

.foregroundColor(selectedTab == tabs.count - 1 ? .blue : .gray)

.frame(maxWidth: .infinity) // Gleichmäßige Breite

}

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

VStack {

Image(systemName: "person.2.fill")

.font(.title2)

Text("Kontakte")

.font(.caption)

.lineLimit(1)

}

.foregroundColor(.gray)

.frame(maxWidth: .infinity) // Gleichmäßige Breite

}

}

.frame(height: 70)

.background(Color.black)

.foregroundColor(.white)

}

, alignment: .bottom

)

.onAppear {

loadCounts()

loadClients()

}

}

**private** **func** loadCounts() {

FirestoreManager.shared.getContracts { result **in**

**switch** result {

**case** .success(**let** contracts): contractCount = contracts.count

**case** .failure: contractCount = 0

}

}

FirestoreManager.shared.getMatches { result **in**

**switch** result {

**case** .success(**let** matches): matchCount = matches.count

**case** .failure: matchCount = 0

}

}

FirestoreManager.shared.getTransfers { result **in**

**switch** result {

**case** .success(**let** transfers): transferCount = transfers.count

**case** .failure: transferCount = 0

}

}

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

**switch** result {

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

clients = loadedClients

clientCount = loadedClients.count

applySearch()

**case** .failure:

clients = []

clientCount = 0

applySearch()

}

}

}

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

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

**switch** result {

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

clients = loadedClients

clientCount = loadedClients.count

applySearch()

**case** .failure:

clients = []

clientCount = 0

applySearch()

}

}

}

**private** **func** applySearch() {

**if** searchText.isEmpty {

filteredClients = clients

} **else** {

filteredClients = clients.filter {

$0.vorname.lowercased().contains(searchText.lowercased()) ||

$0.name.lowercased().contains(searchText.lowercased())

}

}

}

}

**struct** HomeView: View {

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

Text("Home View")

}

}

**struct** ChatView: View {

@EnvironmentObject **var** authManager: AuthManager

@State **private** **var** chats: [Chat] = []

@State **private** **var** selectedChat: Chat? = **nil**

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

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

NavigationStack {

**if** selectedChat == **nil** {

List {

ForEach(chats) { chat **in**

Button(action: {

selectedChat = chat

}) {

HStack {

VStack(alignment: .leading) {

Text(chatParticipants(chat))

.font(.headline)

**if** **let** lastMessage = chat.lastMessage {

Text(lastMessage)

.font(.subheadline)

.foregroundColor(.gray)

.lineLimit(1)

}

}

Spacer()

**if** **let** timestamp = chat.lastMessageTimestamp {

Text(timestamp, style: .time)

.font(.caption)

.foregroundColor(.gray)

}

}

.padding(.vertical, 5)

}

}

}

.navigationTitle("Chats")

.toolbar {

ToolbarItem(placement: .navigationBarTrailing) {

Button(action: {

startNewChat()

}) {

Image(systemName: "plus")

}

}

}

} **else** {

ChatDetailView(chat: selectedChat!, onBack: {

selectedChat = **nil**

})

}

}

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

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

errorMessage = ""

})

}

.onAppear {

loadChats()

}

}

**private** **func** chatParticipants(\_ chat: Chat) -> String {

**guard** **let** userID = authManager.userID **else** { **return** "Unbekannt" }

**let** otherParticipants = chat.participantIDs.filter { $0 != userID }

**return** otherParticipants.joined(separator: ", ")

}

**private** **func** loadChats() {

**guard** **let** userID = authManager.userID **else** {

errorMessage = "Nicht angemeldet"

**return**

}

FirestoreManager.shared.getChats(forUserID: userID) { result **in**

**switch** result {

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

chats = loadedChats

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

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

}

}

}

**private** **func** startNewChat() {

**guard** **let** userID = authManager.userID, **let** userEmail = authManager.userEmail **else** { **return** }

**let** newChat = Chat(participantIDs: [userID, "test\_user"], lastMessage: **nil**, lastMessageTimestamp: **nil**)

FirestoreManager.shared.createOrUpdateChat(chat: newChat) { result **in**

**switch** result {

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

chats.append(Chat(id: chatID, participantIDs: newChat.participantIDs, lastMessage: **nil**, lastMessageTimestamp: **nil**))

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

errorMessage = "Fehler beim Erstellen des Chats: \(error.localizedDescription)"

}

}

}

}

**struct** ChatDetailView: View {

@EnvironmentObject **var** authManager: AuthManager

**let** chat: Chat

**let** onBack: () -> Void

@State **private** **var** messages: [Message] = []

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

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

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

VStack {

ScrollView {

VStack(alignment: .leading, spacing: 10) {

ForEach(messages) { message **in**

HStack {

**if** message.senderID == authManager.userID {

Spacer()

Text(message.content)

.padding()

.background(Color.blue.opacity(0.2))

.cornerRadius(8)

.foregroundColor(.black)

} **else** {

Text(message.content)

.padding()

.background(Color.gray.opacity(0.2))

.cornerRadius(8)

.foregroundColor(.black)

Spacer()

}

}

.padding(.horizontal)

}

}

}

HStack {

TextField("Nachricht eingeben...", text: $newMessage)

.textFieldStyle(RoundedBorderTextFieldStyle())

.submitLabel(.send)

.onSubmit {

sendMessage()

}

Button(action: sendMessage) {

Image(systemName: "paperplane.fill")

.foregroundColor(.blue)

}

}

.padding()

}

.navigationTitle(chatParticipants(chat))

.navigationBarBackButtonHidden(**true**)

.toolbar {

ToolbarItem(placement: .navigationBarLeading) {

Button("Zurück") { onBack() }

}

}

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

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

errorMessage = ""

})

}

.onAppear {

loadMessages()

}

}

**private** **func** chatParticipants(\_ chat: Chat) -> String {

**guard** **let** userID = authManager.userID **else** { **return** "Unbekannt" }

**let** otherParticipants = chat.participantIDs.filter { $0 != userID }

**return** otherParticipants.joined(separator: ", ")

}

**private** **func** loadMessages() {

**guard** **let** chatID = chat.id **else** {

errorMessage = "Keine Chat-ID verfügbar"

**return**

}

FirestoreManager.shared.getMessages(forChatID: chatID) { result **in**

**switch** result {

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

messages = loadedMessages

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

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

}

}

}

**private** **func** sendMessage() {

**guard** **let** userID = authManager.userID, **let** userEmail = authManager.userEmail, **let** chatID = chat.id **else** {

errorMessage = "Nicht angemeldet oder keine Chat-ID"

**return**

}

**guard** !newMessage.trimmingCharacters(in: .whitespaces).isEmpty **else** { **return** }

**let** message = Message(senderID: userID, senderEmail: userEmail, content: newMessage, timestamp: Date())

FirestoreManager.shared.sendMessage(chatID: chatID, message: message) { result **in**

**switch** result {

**case** .success:

newMessage = ""

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

errorMessage = "Fehler beim Senden der Nachricht: \(error.localizedDescription)"

}

}

}

}

**struct** StatCard: View {

**let** title: String

**let** count: Int

**let** icon: String

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

VStack {

Image(systemName: icon)

.font(.title2)

.foregroundColor(.blue)

Text(title)

.font(.caption)

.foregroundColor(.gray)

Text("\(count)")

.font(.title)

.fontWeight(.bold)

}

.frame(width: 80, height: 80)

.padding()

.background(Color(.systemGray6))

.cornerRadius(10)

}

}

#Preview {

EmployeeView()

.environmentObject(AuthManager())

}