**import** FirebaseAuth

**import** FirebaseFirestore

**import** Foundation

**enum** UserRole: String, Codable {

**case** mitarbeiter = "Mitarbeiter"

**case** klient = "Klient"

**case** gast = "Gast"

}

**class** AuthManager: ObservableObject {

@Published **var** isLoggedIn = **false**

@Published **var** userRole: UserRole?

@Published **var** userEmail: String?

@Published **var** userID: String?

**private** **var** authHandle: AuthStateDidChangeListenerHandle?

**init**() {

// Listener für Authentifizierungsstatus

authHandle = Auth.auth().addStateDidChangeListener { [**weak** **self**] auth, user **in**

**guard** **let** self = **self** **else** { **return** }

**if** **let** user = user {

self.isLoggedIn = **true**

self.userEmail = user.email

self.userID = user.uid

self.loadUserRole(user.uid)

} **else** {

self.isLoggedIn = **false**

self.userRole = **nil**

self.userEmail = **nil**

self.userID = **nil**

// Auto-Login wird hier NICHT mehr aufgerufen

}

}

}

**func** loadUserRole(\_ uid: String) {

**let** db = Firestore.firestore()

db.collection("users").document(uid).getDocument { [**weak** **self**] snapshot, error **in**

**guard** **let** self = **self** **else** { **return** }

**if** **let** error = error {

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

DispatchQueue.main.async {

self.userRole = .gast // Standardrolle bei Fehler

}

**return**

}

**if** **let** data = snapshot?.data(), **let** roleString = data["rolle"] **as**? String, **let** role = UserRole(rawValue: roleString) {

DispatchQueue.main.async {

self.userRole = role

}

} **else** {

DispatchQueue.main.async {

self.userRole = .gast // Standardrolle

}

}

}

}

**func** login(email: String, password: String, completion: **@escaping** (Result<Void, Error>) -> Void) {

Auth.auth().signIn(withEmail: email, password: password) { [**weak** **self**] result, error **in**

**guard** **let** self = **self** **else** { **return** }

**if** **let** error = error {

completion(.failure(error))

} **else** **if** **let** user = result?.user {

self.userEmail = user.email

self.userID = user.uid

self.isLoggedIn = **true**

self.loadUserRole(user.uid)

completion(.success(()))

}

}

}

**func** signOut() {

**do** {

**try** Auth.auth().signOut()

DispatchQueue.main.async {

**self**.isLoggedIn = **false**

**self**.userRole = **nil**

**self**.userEmail = **nil**

**self**.userID = **nil**

}

} **catch** {

print("Logout fehlgeschlagen: \(error.localizedDescription)")

}

}

**func** register(email: String, password: String, role: UserRole, completion: **@escaping** (Result<Void, Error>) -> Void) {

Auth.auth().createUser(withEmail: email, password: password) { [**weak** **self**] result, error **in**

**guard** **let** self = **self** **else** { **return** }

**if** **let** error = error {

completion(.failure(error))

} **else** **if** **let** user = result?.user {

**let** db = Firestore.firestore()

db.collection("users").document(user.uid).setData([

"email": email,

"rolle": role.rawValue

]) { error **in**

**if** **let** error = error {

completion(.failure(error))

} **else** {

self.userEmail = email

self.userID = user.uid

self.userRole = role

self.isLoggedIn = **true**

completion(.success(()))

}

}

}

}

}

// Automatischer Login als separate Funktion

**func** autoLogin() {

**let** defaultEmail = "eggimann@t-online.de"

**let** defaultPassword = "test123"

login(email: defaultEmail, password: defaultPassword) { result **in**

**switch** result {

**case** .success:

print("Automatischer Login erfolgreich für \(defaultEmail)")

DispatchQueue.main.asyncAfter(deadline: .now() + 1.0) {

**self**.loadUserRole(Auth.auth().currentUser?.uid ?? "")

}

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

print("Automatischer Login fehlgeschlagen: \(error.localizedDescription)")

}

}

}

**deinit** {

**if** **let** handle = authHandle {

Auth.auth().removeStateDidChangeListener(handle)

}

}

}