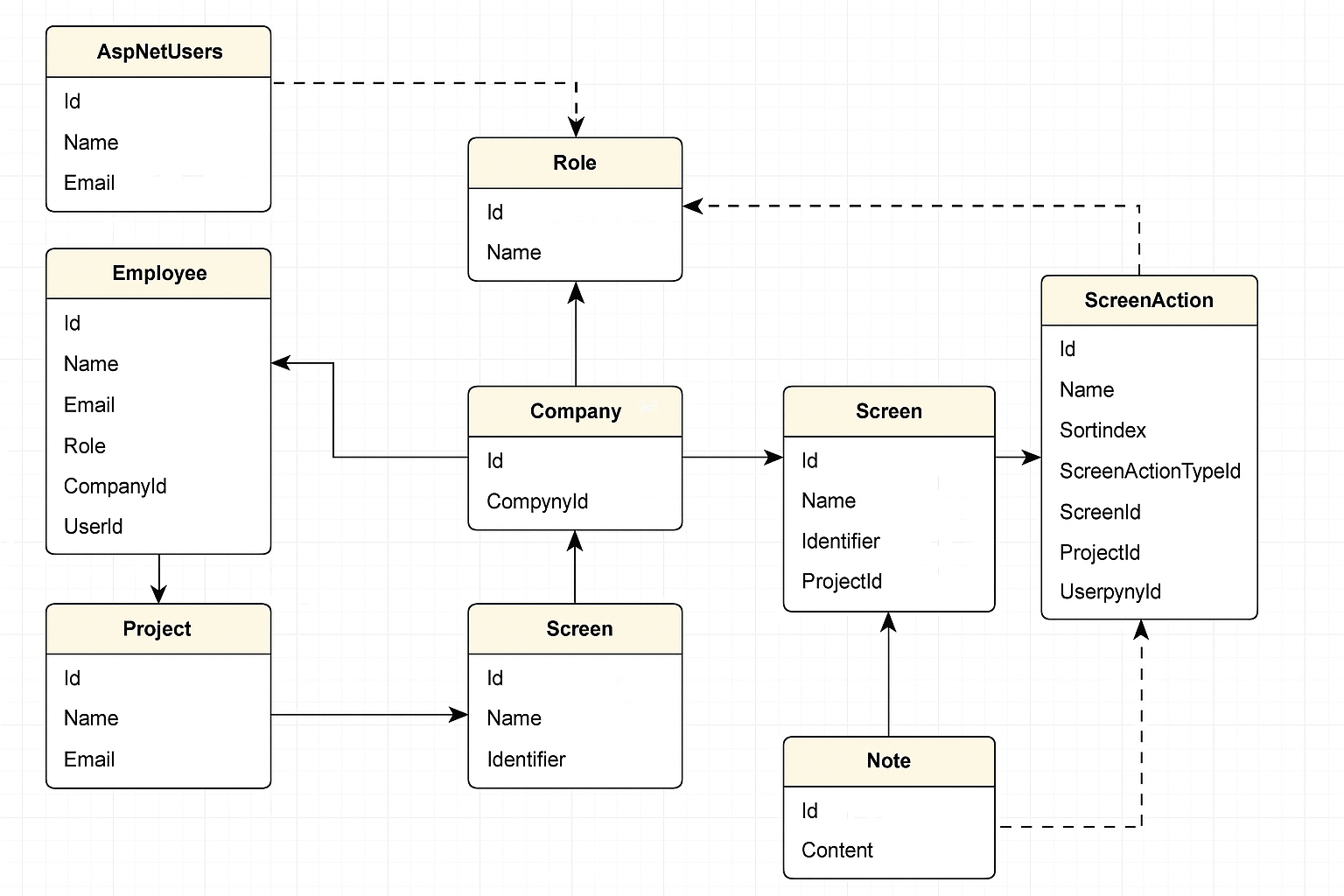
# 🧠 UserFlowAPI – Technische Funktionsübersicht

Die **UserFlowAPI** ist eine moderne, modulare RESTful WebAPI zur Verwaltung von Benutzern, Projekten, Aktionen und Firmenkontexten. Sie basiert auf aktuellen .NET-Technologien und bietet eine saubere Trennung von Datenzugriff, Geschäftslogik und Darstellung.

* 🔐 **JWT-Authentifizierung**  
  Sichere Benutzeranmeldung via JSON Web Tokens inkl. RefreshToken-Support und optionalem Setup-Flow (z. B. Passwort nach Einladung setzen).
* 🏢 **Multi-Tenancy & Rollenverwaltung**  
  Mandantenfähigkeit durch CompanyId, automatische Filterung per ICurrentUserService. Benutzerrollen (Admin, GlobalAdmin, Manager, User) via ASP.NET Core Identity.
* 🧼 **Soft Delete & globale Filter**  
  Automatische Ausblendung gelöschter Entitäten durch globale EF Core QueryFilter (z. B. IsDeleted).
* 📦 **DTOs & Mapping**  
  Verwendung von DTO-Klassen für alle Datenoperationen mit zentralem Mapping über eine DTOMapper-Hilfsklasse (Ausdrucksbasierte Projektion & Konvertierung).
* 🔄 **Modular & Erweiterbar**  
  Neue Entitäten lassen sich durch zentrale EntityTypeConfigurations, Interfaces (ISoftDelete, IUserOwned, ICompanyOwned) und DTO-Mapping leicht integrieren.
* 📊 **Admin-Dashboard-Endpoints**  
  Admins erhalten Zugriff auf gezielte Statistiken: Benutzeranzahl, Projektanzahl, letzte Benutzeranmeldungen etc.
* 📝 **Seeding & Testdaten**  
  Seeder generiert Beispiel-Firmen, Benutzer (inkl. Admins & Rollen), sowie zugehörige Projekte, Screens, Aktionen und Notizen. Zusätzlich werden Test-Benutzerlisten als JSON erzeugt.
* 🧪 **Swagger & Logging**  
  Integrierte Swagger-Oberfläche mit JWT-Unterstützung. Logging via Serilog mit Ausgabe in Datei und Konsole (UTF-8 mit Emoji-Unterstützung).
* **📡 Echtzeit-Synchronisation mit SignalR**
* Realtime-Updates bei Änderungen von Projekten, Screens oder Notizen
* Anzeige von "User is editing…" im Client
* Nutzung von Gruppen pro CompanyId oder ProjectId
* Helper Klassen für Live Updates der Collection. (Erspart Reload)
* **🔁 ChangeStreams / WebHooks (eventuell später PostgreSQL-Listen/Notify)**
* Reaktion auf Änderungen ohne Polling
* Grundlage für Audit-Trails, externes Monitoring oder automatisierte Workflows
* **🛠️ Soft Delete Recovery**
* Wiederherstellung gelöschter Objekte (Projekte, Screens, Notes etc.)
* Getrennte Views für aktive und gelöschte Datensätze
* Eventuell mit Archivierungsfunktion oder TTL

# 🧩 Entity-Relationship-Diagramm



## 🌐 Allgemein

* Technologie-Stack: ASP.NET Core 8 (bereit für .NET 9), Entity Framework Core, PostgreSQL, Identity, Serilog
* Architektur: RESTful Web API mit Multi-Tenant-Support und fein granulierten Rollenrechten
* Client-Kompatibilität: Optimiert für WPF, MAUI und andere .NET-Clients
* Logging: Serilog mit täglichen Log-Dateien (inkl. Emoji-Symbolik) + UTF-8 Konsole
* Swagger: Vollständig integriertes Swagger UI mit JWT-Unterstützung
* Global Error Handling: Eigene ExceptionsMiddleware für konsistente JSON-Fehlerrückgaben

## 🔐 Authentifizierung & Sicherheit

* ASP.NET Core Identity mit benutzerdefiniertem User-Typ (long IDs)
* JWT + Refresh Token Login-System
* Passwort-Erstsetzung (First Login Flow mit NeedsPasswordSetup)
* Admin-gesteuerte Benutzerregistrierung (kein Self-Registering außer Firmen-Admin)
* Rollenverwaltung mit GlobalAdmin, Admin, Manager, User
* Zentrale Zugriffskontrolle mit ICurrentUserService & Claims

## 🧾 Firmen- und Benutzerverwaltung

* CompanyController: CRUD für Firmen (nur GlobalAdmin), anonyme Registrierung mit Admin-User
* UserController: Übersicht über Benutzer (rollenspezifisch sichtbar), Admin-gesteuerte Benutzererstellung, Aktualisierung & Soft-Delete

## 🗂️ Projektstruktur (Multi-Tenant)

* Projektverwaltung: Projekte gehören zu Firmen & Benutzern, Sichtbarkeit nach Rolle
* Screens: Projekte mit Screens, Identifikation per Identifier
* ScreenActions: Aktionen auf Screens mit Koordinaten & Beschreibung, ActionType-Zuweisung

## 🧠 Erweiterte Entities

* Notes: Notizen an Screens/Projekten (sichtbar nach Firma/User)
* Employees: Repräsentieren Benutzer mit Firmenbezug (inkl. Rolle, Email, UserId, CompanyId)

## 🛠️ DTO-Mapping & Abstraktion

* Zentrale DTOMapper-Klasse mit ToDTO() / UpdateEntity() Methoden
* Expression<Func<>> für projektspezifische Auswahl in LINQ

## 🧩 Soft Delete & Filter

* Globale Query-Filter: IsDeleted, UserId & CompanyId werden automatisch berücksichtigt

## 📊 Dashboard & Analyse

* DashboardController: User-/Projektanzahl, letzte Benutzeranmeldungen – rollenspezifisch sichtbar

## 🧪 Testdaten & Seeding

* Rollen: GlobalAdmin, Admin, Manager, User
* Initialdaten: Global Admin, 10 Firmen mit Admins/Managern/Usern, Projekte, Screens, Notes, Actions
* Faker: realistische Dummy-Inhalte für Namen, Firmen etc.
* JSON-Listen: global-admins.json & company-users.json mit Namen, E-Mail, Rolle

## 🧰 Weitere Features & Technik

* Serilog-Konfiguration via appsettings.json (inkl. LogLevel Override)
* Emoji-Logs & UTF-8 Konsole
* Auto-Migration & Seeding im Development-Modus
* REST-konforme Controller mit vollständigem Logging

## 🧪 Integration & Tests

* Swagger: Dokumentation + JWT Login Support
* Client (MAUI/WPF): Nutzung der JSON-Listen zum Test-Login

## 🔮 Geplante Erweiterungen und Roadmap

Die aktuelle Version der UserFlow API bildet bereits eine solide und produktionsreife Basis. Folgende Features sind für kommende Releases vorgesehen:

**1. 🌐 Multilinguale Unterstützung (MultiKulti)**

* UI-Übersetzungen via JSON-Resx-Dateien
* Lokalisierung von Fehlermeldungen und Datenbankeinträgen
* Culture-abhängige Validierung

**2. 📊 Admin-Dashboard-Erweiterung**

* KPIs: Anzahl aktiver Benutzer, Letzte Logins
* Audit-Trail für Benutzeraktionen
* Exportfunktionen (CSV, Excel)

**3. 🧩 Plugin-Support für externe Dienste**

* Einbindung von OCR, Bildannotationen, Workflows
* Upload von Screenshots zur automatisierten Analyse

**4. 🧪 Testdaten & Simulation**

* Dev-Tools im Client zur Auswahl eines Seed-Users (Login als Dropdown)
* Automatischer Projektgenerator für UI-Demonstrationen