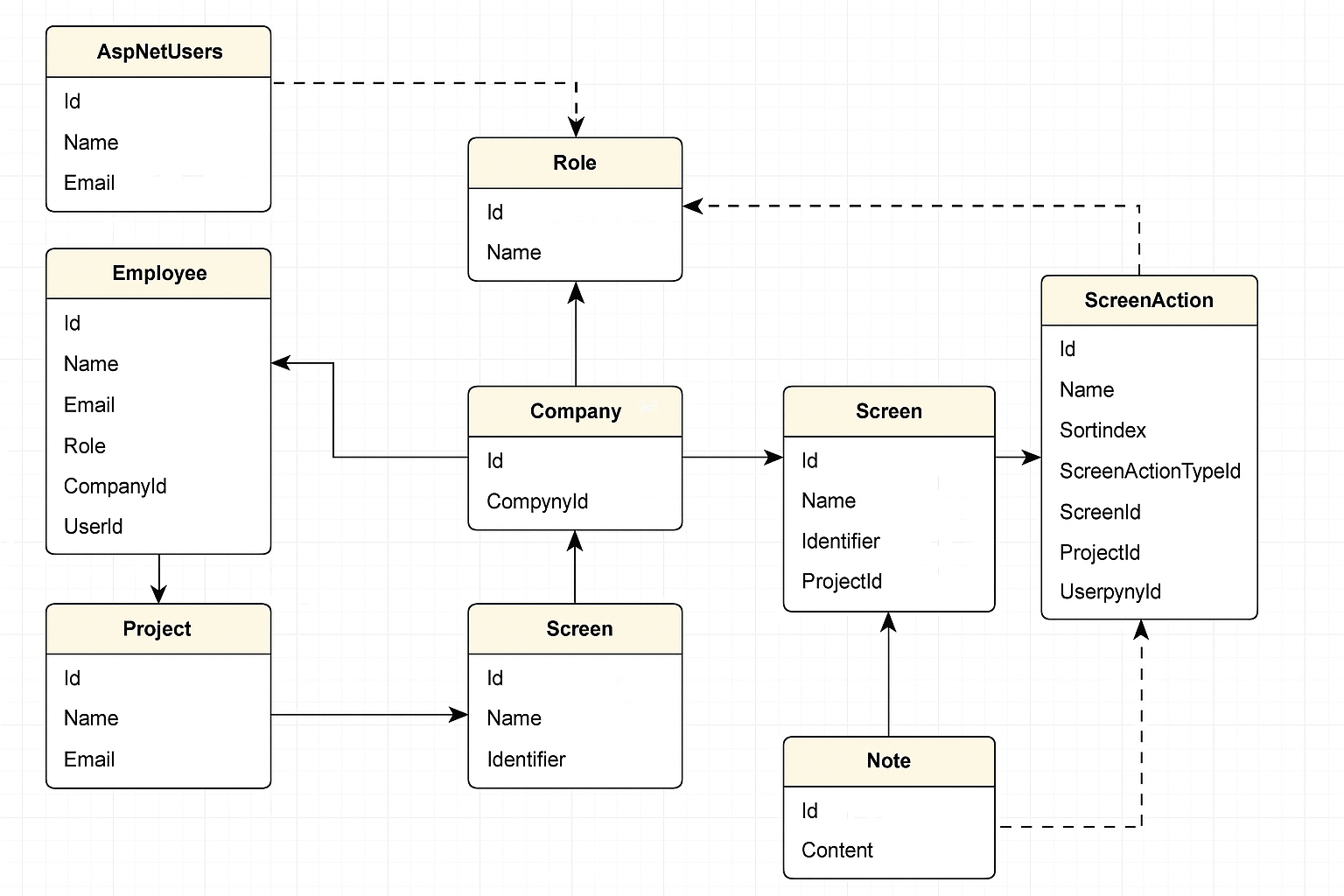
# 🧠 UserFlowAPI Technical Feature Overview

The **UserFlowAPI** is a modern, modular RESTful WebAPI for managing users, projects, actions and company contexts. It is based on current . NET technologies, providing a clean separation of data access, business logic, and representation.

* 🔐 **JWT authenticationSecure user login via JSON Web Tokens incl.** RefreshToken support and optional setup flow (e.g. set password after invitation).
* 🏢 **Multi-tenancy & role management**  
  Multi-tenancy through CompanyId, automatic filtering via ICurrentUserService. User roles (Admin, GlobalAdmin, Manager, User) via ASP.NET Core Identity.
* 🧼 **Soft Delete & Global FiltersAutomatic hiding of deleted entities by EF Core** Global QueryFilter (e.g. IsDeleted).
* 📦 **DTOs & MappingUse of DTO classes for all data operations with central mapping via a** DTOMapper helper class (Expression-Based Projection & Conversion).
* 🔄 **Modular & ExtensibleNew entities can be easily integrated through central** EntityTypeConfigurations, interfaces (ISoftDelete, IUserOwned, ICompanyOwned) and DTO mapping.
* 📊 **Admin Dashboard EndpointsAdmins get access to targeted statistics: number of users, number of projects, last user logins, etc.**
* 📝 **Seeder**  
  generates sample companies, users (incl. admins & roles), as well as associated projects, screens, actions, and notes. In addition, test user lists are generated as JSON.
* 🧪 **Swagger & Logging**  
  Integrated Swagger interface with JWT support. Logging via Serilog with output to file and console (UTF-8 with emoji support).

# 🧩 Entity Relationship Diagram



## 🌐 General

* Technology Stack: ASP.NET Core 8 (ready for .NET 9), Entity Framework Core, PostgreSQL, Identity, Serilog
* Architecture: RESTful Web API with multi-tenant support and finely granular role rights
* Client Compatibility: Optimized for WPF, MAUI, and others. NET Clients
* Logging: Serilog with daily log files (incl. emoji symbolism) + UTF-8 console
* Swagger: Fully integrated Swagger UI with JWT support
* Global Error Handling: Custom ExceptionsMiddleware for consistent JSON error returns

## 🔐 Authentication & Security

* ASP.NET Core Identity with user-defined user type (long IDs)
* JWT + Refresh Token Login-System
* Password Replacement (First Login Flow with NeedsPasswordSetup)
* Admin-controlled user registration (no self-registering except company admin)
* Role management with GlobalAdmin, Admin, Manager, User
* Central access control with ICurrentUserService & Claims

## 🧾 Company and user management

* CompanyController: CRUD for companies (GlobalAdmin only), anonymous registration with admin user
* UserController: Overview of users (role-specific visible), admin-controlled user creation, update & soft-delete

## 🗂️ Project Structure (Multi-Tenant)

* Project management: Projects belong to companies & users, visibility by role
* Screens: Projects with screens, identification by identifier
* ScreenActions: Actions on screens with coordinates & description, ActionType assignment

## 🧠 Advanced entities

* Notes: Notes on screens/projects (visible by company/user)
* Employees: Represent users with a company connection (incl. role, email, UserId, CompanyId)

## 🛠️ DTO Mapping & Abstraction

* Central DTOMapper class with ToDTO() / UpdateEntity() methods
* Expression<Func<>> for project-specific selection in LINQ

## 🧩 Soft Delete & Filter

* Global query filters: IsDeleted, UserId & CompanyId are automatically taken into account

## 📊 Dashboard & Analytics

* DashboardController: Number of users/projects, last user logins – role-specific visible

## 🧪 Test Data & Seeding

* Rollen: GlobalAdmin, Admin, Manager, User
* Initial data: Global Admin, 10 companies with admins/managers/users, projects, screens, notes, actions
* Faker: realistic dummy content for names, companies, etc.
* JSON-Listen: global-admins.json & company-users.json mit Namen, E-Mail, Rolle

## 🧰 Other Features & Technology

* Serilog-Konfiguration via appsettings.json (inkl. LogLevel Override)
* Emoji-Logs & UTF-8 Konsole
* Auto-Migration & Seeding im Development-Modus
* REST-compliant controllers with full logging

## 🧪 Integration & Tests

* Swagger: Documentation + JWT Login Support
* Client (MAUI/WPF): Use of JSON lists for test login

## 🔮 Planned expansions and roadmap

The current version of the UserFlow API already forms a solid and production-ready basis. The following features are planned for upcoming releases:

**1. 📡 Real-time synchronization with SignalR**

* Real-time updates for changes to projects, screens or notes
* Display of "User is editing..." in the client
* Use of groups per CompanyId or ProjectId

**2. 🔁 ChangeStreams / WebHooks (eventuell später PostgreSQL-Listen/Notify)**

* Responding to changes without polling
* Basis for audit trails, external monitoring or automated workflows

**3. 🛠️ Soft Delete Recovery**

* Recovery of deleted objects (projects, screens, notes, etc.)
* Separate Views for Active and Deleted Records
* Possibly with archiving function or TTL

**4. 🌐 Multilingual support (MultiKulti)**

* UI translations via JSON resx files
* Localization of error messages and database entries
* Culture-dependent validation

**5. 📊 Admin Dashboard Extension**

* KPIs: Number of active users, Last logins
* Audit trail for user actions
* Export functions (CSV, Excel)

**6. 🧩 Plugin support for external services**

* Integration of OCR, image annotations, workflows
* Upload screenshots for automated analysis

**7. 🧪 Testdaten & Simulation**

* Dev tools in the client to select a seed user (login as dropdown)
* Automatic project generator for UI demonstrations