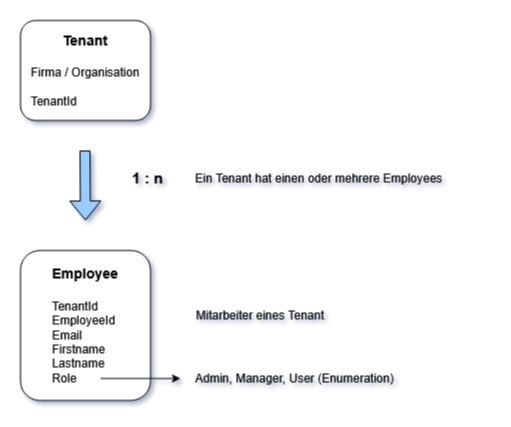
**UserFlowAPI Systemübersicht  
  
Multi-Tenant-, Employee- und Rollenmodell**

****

**1. Zielsetzung des Systems**

Das System trennt **Firmen (Tenants)** und deren **Mitarbeiter (Employees)** logisch und technisch sauber. Jede Firma verwaltet ihre eigenen Projekte, Screens, Actions und Notizen – unabhängig von anderen Firmen. Durch die Einführung von **rollenbasierten Rechten** kann pro Mitarbeiter gesteuert werden, **welche Daten** er sieht und **welche Aktionen** er ausführen darf.

**2. Grundlegende Architektur**

| **Entität** | **Beschreibung** |
| --- | --- |
| **Tenant** | Eine Firma oder Organisation, die sich registriert und eigene Mitarbeiter und Projekte verwalten kann. |
| **Employee** | Ein Benutzerkonto eines Mitarbeiters innerhalb eines Tenants. Jeder Employee ist eindeutig einem Tenant zugeordnet. |
| **Role** | Eine Rolle definiert die Berechtigungen eines Employees innerhalb seines Tenants, z.B. Admin, Manager, User. |

**3. Login- und Authentifizierungsmodell**

* **Tenant-Login**: Der Tenant-Administrator kann sich anmelden, um seine Firma zu verwalten (klassisches Firmenkonto).
* **Employee-Login** (Neu!): Jeder Employee erhält ein eigenes Login, basierend auf ASP.NET Core Identity.
* **JWT-Token**: Nach erfolgreichem Login erhalten sowohl Tenants als auch Employees ein sicheres JWT (JSON Web Token) für die API-Nutzung.

**4. Sicherheit und Multi-Tenancy**

* **Datenisolation**:  
  Alle Datenbankabfragen sind mit einem **globalen Query-Filter** abgesichert:  
  → Mitarbeiter oder Admins sehen **nur** die eigenen Daten (eigenes TenantId).
* **Soft Delete**:  
  Daten werden nicht sofort gelöscht, sondern mit IsDeleted = true markiert und automatisch ausgeblendet.
* **Rollenbasierte Autorisierung** (geplant / Aufbau läuft):  
  → Zugriffsrechte werden abhängig von der Employee-Rolle dynamisch geprüft (Admin, Manager, Mitarbeiter).

**5. Typische Nutzungsbeispiele**

| **Aktion** | **Wer?** | **Beschreibung** |
| --- | --- | --- |
| Projekte verwalten | Tenant-Admin | Erstellen, Bearbeiten und Löschen von Projekten. |
| Screens erstellen | Manager, Admin | Mitarbeiter mit entsprechenden Rollen können Screens zu Projekten hinzufügen. |
| Notizen schreiben | Alle Rollen | Jeder Mitarbeiter kann Notizen zu Screens anlegen. |
| Mitarbeitereinstellungen ändern | Admin | Nur Administratoren dürfen Mitarbeiter erstellen oder löschen. |

**6. Vorteile des Konzepts**

| **Vorteil** | **Beschreibung** |
| --- | --- |
| **Hohe Sicherheit** | Daten strikt nach Tenant und Rolle getrennt. |
| **Flexibilität** | Neue Rollen und Rechte jederzeit einfach erweiterbar. |
| **Skalierbarkeit** | Multi-Tenant-ready: mehrere Firmen arbeiten parallel und unabhängig. |
| **Wartbarkeit** | Saubere Struktur: klare Trennung von Authentifizierung, Autorisierung und Business-Logik. |
| **Erweiterbarkeit** | Einfach um zusätzliche Funktionen wie Self-Service-Portale, Rollenmanagement oder komplexe Berechtigungen erweiterbar. |

**7. Technische Besonderheiten**

* **Optimierte Controller**Bei Abfragen / Queries erfolgt die Konvertierung von Entitäten in DTOs (die der Client erhält) durch Projektion per Expressions   
  (DTO-Mapping direkt durch EntityFramework Core)
* **Automatisches Setzen von CreatedAt / UpdatedAt Feldern** beim Speichern von Daten.
* **Seed-Daten** für schnelle Initialisierung von Test-Tenants und -Mitarbeitern.
* **Schlanke API**: Optimiert für einfache Integration in Web- und Mobile-Clients (z.B. MAUI-App).

**📋 Zusammenfassung**

Das neue System ermöglicht es,

* Firmen (Tenants) und deren Mitarbeiter (Employees) **sicher getrennt** zu verwalten,
* **fein abgestufte Berechtigungen** über Rollen zu verteilen,
* und **viele Firmen parallel** in einer einzigen Instanz der WebAPI betreiben zu können.

Damit ist **UserFlowAPI** optimal vorbereitet für den professionellen Einsatz in echten Multi-Tenant-Umgebungen.