

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



EDUMAN EĞİTİM YÖNETİM SİSTEMİ
TEKNİK ANALİZ VE MİMARİ BÜT RAPORU

Zafer Emre KILINÇ
Bilgisayar Mühendisliği

Proje Kaynak Kodları (GitHub):
GITHUB LİNKİ

Proje Video Dosyası:
Google Drive Bağlantısı

Danışman: Öğr. Gör. Enver BAĞCI

Özet

Bu rapor, modern eğitim kurumlarının yönetimsel ihtiyaçlarını dijitalleştirmek amacıyla geliştirilen **Edunex (Eduman)** platformunun teknik mimarisini, tasarım prensiplerini ve uygulama detaylarını kapsamlı bir şekilde sunmaktadır. Edunex; kurum, okul, sınıf ve kullanıcı hiyerarşisini esnek bir yapıda yönetebilen, ölçeklenebilir ve güvenli bir **Learning Management System** çözümüdür.

Sistemin backend altyapısı, **.NET 8** framework'ü kullanılarak **N-Layer Architecture** prensiplerine uygun olarak tasarlanmıştır. Veri tutarlılığı ve kodun sürdürülebilirliği için **Domain-Driven Design** yaklaşımlarından yararlanılmış; **Entity Framework Core** ve **PostgreSQL** ile güçlü bir veri yönetim katmanı oluşturulmuştur. Ayrıca, sistemdeki tüm veri değişiklikleri, geliştirilen **Audit Interceptor** mekanizması sayesinde otomatik olarak denetlenmekte ve tarihsel olarak izlenebilmektedir.

Frontend tarafında ise **React** ve **TypeScript** teknolojileri, **Vite** ile entegre edilerek yüksek performanslı bir kullanıcı deneyimi hedeflenmiştir. **Material UI** kütüphanesi ile kurumsal bir görsel dil oluşturulmuş, **React Query** ve **Axios** ikilisi ile asenkron veri yönetimi ve **caching** süreçleri optimize edilmiştir.

Güvenlik katmanında **ASP.NET Identity** altyapısı **JWT** tabanlı **authentication** ile birleştirilerek, **RBAC** ve **Multi-tenancy** sağlanmıştır. Raporun ilerleyen bölümlerinde, bu bileşenlerin birbiriyle olan etkileşimi, **request lifecycle** ve lisanslama algoritmaları teknik detaylarıyla analiz edilmektedir.

Anahtar Kelimeler: .NET 8, React, TypeScript, N-Layer Architecture, Multi-tenancy, JWT, PostgreSQL, LMS.

1. Giriş

1.1 Projenin Amacı ve Önemi

Eğitim kurumlarının dijital dönüşüm süreci, verinin merkezi bir noktadan yönetilmesini, güvenliğini ve ölçeklenebilirliğini zorunlu kılmaktadır. Edunex projesi; klasik eğitim yönetim sistemlerinin ötesine geçerek, **multi-layered** ve **multi-tenant** özellikleri birleştiren modern bir altyapı sunmayı amaçlamaktadır.

Projenin önemi; kurumsal hiyerarşiyi dijital ortamda birebir modellemesi, lisanslama süreçlerini dinamik metriklerle kontrol etmesi ve tüm bunları yaparken yüksek performanslı bir kullanıcı deneyimi sağlamasıdır. Yazılım, sadece bir yönetim aracı değil, aynı zamanda veriye dayalı stratejik kararlar alınmasını sağlayan bir karar destek altyapısı sunar.

1.2 Kapsam ve Temel Fonksiyonlar

Edunex projesi, bir eğitim kurumunun ihtiyaç duyabileceği tüm idari ve akademik süreçleri kapsayacak şekilde modüler bir yapıda tasarlanmıştır. Sistemin temel fonksiyonları şu şekilde sınıflandırılmaktadır:

- **Tenant Management:** Kurum oluşturma, adresleme ve tip tanımlama işlemleri.
- **Academic Hierarchy:** Okulların ve sınıfların kurum bazlı ilişkilendirilmesi.
- **License Control:** Kullanıcı limitleri, tarih aralıkları ve lisans tiplerinin yönetimi.
- **IAM:** JWT tabanlı güvenli giriş ve **Role-Based** yetkilendirme.
- **Analytical Reporting:** Dashboard üzerinden aylık kullanıcı grafikleri ve özet metriklerin takibi.

1.3 Hedef Kitle ve Kullanım Senaryoları

Sistem, eğitim ekosistemindeki farklı paydaşların ihtiyaçlarına yanıt verecek şekilde kurgulanmıştır:

- **System Administrators:** Tüm kurumların lisanslarını yöneten ve sistemi yapılandıran üst düzey kullanıcılar.
- **Tenant Administrators:** Kendi kurumuna bağlı okulları ve sınıfları yöneten paydaşlar.
- **Educators and Analysts:** Sistem üzerinden raporlamaları takip eden ve operasyonel süreçleri yöneten kullanıcılar.

Example Scenario: Bir kurum yöneticisi sisteme dahil olduktan sonra, sahip olduğu lisans limitleri dahilinde yeni bir okul tanımlayabilir, bu okula bağlı sınıflar oluşturabilir ve kullanıcı atamalarını gerçekleştirebilir. Sistem, her işlemde lisans limitlerini otomatik olarak denetler.

1.4 Kullanılan Teknolojik Yığın (Tech Stack)

Edunex, modern yazılım geliştirme standartlarına uygun olarak aşağıdaki teknoloji yığını ile geliştirilmiştir:

Tablo 1.1: Edunex Tech Stack

Layer	Technology	Reasoning
Backend	.NET 8 (C#)	High performance, strong type safety and enterprise library support.
Frontend	React + TS	Component-based structure and minimizing runtime errors.
Database	PostgreSQL	Open source, relational data management and scalability.
ORM	EF Core	Object-oriented database management via Code First.
Bundler	Vite	Fast HMR and optimized frontend build management.
Auth	JWT & Identity	Distributed and secure authentication mechanism.

2. Sistem Mimarisi ve Tasarım Prensipleri

2.1 N-Layer Architecture Yaklaşımı

Edunex platformu, endüstri standardı olan **N-Layer Architecture** yapısını benimsemiştir. Bu yaklaşımın temel amacı, **UI**, **Business Logic** ve **Data Access** süreçlerini birbirinden tamamen soyutlayarak **maintainability** sağlamaktır.

Sistemde kullanılan katmanlar ve sorumlulukları şu şekildedir:

- **Eduman.API (Presentation Layer):** Dış dünyaya açılan kapıdır. HTTP isteklerini karşılar ve sadece **DTO** yapıları üzerinden veri alışverişi yapar.
- **Eduman.Application (Application Layer):** **Use-cases** yönetiminin yapıldığı katmandır. Servisler, **DTO**'lar ve iş kuralları burada tanımlanır.
- **Eduman.Domain (Domain Layer):** Sistemin merkezidir. Teknolojik bağımlılıklardan arındırılmıştır; sadece **entities** ve temel iş kuralları yer alır.
- **Eduman.Infrastructure (Infrastructure Layer):** **Persistence** yapılandırması, repository implementasyonları ve migration süreçleri bu katmandadır.

2.2 Domain-Driven Design Esintileri

Sistem tasarımı tam bir **DDD** mimarisi olmasa da, **Rich Domain Model** ve **Separation of Concerns** ilkelerinden esinlenmiştir.

- **Entities:** **Institution**, **School** ve **Class** gibi nesneler sadece **Data Holder** değil, kendi içindeki kuralları doğrulayan yapılardır.
- **Value Objects:** Belirli bir kimliği olmayan ancak bir kavramı temsil eden değer nesneleri kullanılmıştır.
- **Aggregates:** Birbirine sıkı bağlı nesneler bir **Consistency Boundary** içinde yönetilir.

2.3 Shared Kernel ve Kod Yeniden Kullanılabilirliği

Projenin ölçeklenebilirliğini artırmak adına, tüm alt projelerin ortak kullandığı **Edu-nex.SharedKernel** kütüphanesi geliştirilmiştir. Bu kütüphane sayesinde **DRY** prensibi uygulanmıştır.

Shared Kernel içeriğinde yer alan temel bileşenler:

- **BaseEntity:** Tüm entity'ler için ortak olan **Id**, **CreatedAt** ve **UpdatedAt** alanlarını içerir.
- **Generic Repository Pattern:** CRUD işlemleri için standart bir arayüz sağlar (**IRepository<T>**).
- **Unit of Work:** İşlemlerin tek bir transaction içinde **atomic** bir şekilde gerçekleşmesini garanti eder.

2.4 Veritabanı Tasarımı ve PostgreSQL Entegrasyonu

Edunex, **RDBMS** olarak ****PostgreSQL**** kullanmaktadır. Tasarım, **Code-First** yaklaşımı ile yürütülmüş, veri modelleri **C#** sınıfları olarak tanımlanıp **EF Core** aracılığıyla şemaya dönüştürülmüştür.

- **Multi-Tenancy:** Her tabloda bulunan **TenantId** alanı sayesinde, farklı kurumların verileri aynı veritabanında mantıksal olarak izole edilmiştir.
- **Indexing:** Sorgu performansını artırmak için **foreign key** alanları üzerinde indeksleme yapılmıştır.
- **Audit Logging:** Veritabanı işlemleri, **Audit Interceptor** aracılığıyla otomatik olarak mühürlenir.

3. Backend Teknik Analizi

3.1 Domain Katmanı ve Entity Yapıları

Sistemin kalbi olan Domain katmanı, tüm iş nesnelerini ve kurallarını barındırır. `Eduman.Domain` projesi içerisinde yer alan **entities**, veritabanı şemasının temelini oluşturur. Tüm varlıklar `SharedKernel` içindeki `BaseEntity` sınıfından türetilerek standart bir **audit** yapısına kavuşturulmuştur.

Öne çıkan temel varlık yapıları:

- **Institution:** Sistemin en üst düğümüdür. Tenant yönetimi bu varlık üzerinden yapılır.
- **EdumanLicense:** Kurumların kullanım haklarını tanımlar. `UserLimit`, `StartDate`, `EndDate` gibi kritik alanları içerir.
- **School & Class:** Kuruma hiyerarşik olarak bağlı olan birimlerdir.

3.2 Application Katmanı: Servisler ve DTO Yönetimi

Application katmanı, Domain katmanı ile dış dünya arasındaki köprüdür. Bu katmanda veritabanı nesneleri doğrudan dışarıya açılmaz; bunun yerine **Data Transfer Object** yapıları kullanılır.

- **DTO Usage:** Hassas verilerin gizlenmesi ve **payload** optimizasyonu sağlanır.
- **Service Logic:** `InstitutionService` gibi sınıflar, `IRepository` ve `IUnitOfWork` arayüzlerini kullanarak **use-case** akışlarını yürütür.

3.3 Infrastructure Katmanı: Persistence ve EF Core

`Eduman.Infrastructure` projesi, sistemin **persistence** stratejisini belirler. **Entity Framework Core** kullanılarak geliştirilen bu katman, PostgreSQL veritabanı ile **OOP** dünyası arasındaki eşleşmeyi sağlar.

- **DbContext Configuration:** EdumanDb sınıfı üzerinden tüm tabloların ilişkileri ve kısıtları tanımlanır.
- **Audit Interceptor:** Kayıt ekleme veya güncelleme anında **audit fields** değerlerini otomatik olarak dolduran mekanizma burada devreye girer.

3.4 API Katmanı: RESTful Endpoint Tasarımı

Eduman.API projesi, istemci tarafına standartlaştırılmış servisler sunar. Tüm endpoint tasarımlarında **RESTful** prensipleri gözetilmiştir.

- **Controller Structure:** Her **aggregate** için ayrı bir controller tanımlanmıştır.
- **Error Handling:** Sistemde oluşan tüm hatalar **RFC 7807** standardında dönerek frontend tarafında anlamlı hata mesajları oluşturulmasına olanak tanır.

3.5 Authentication ve Authorization (Identity & JWT)

Güvenlik katmanı, **ASP.NET Core Identity** ve **JWT** entegrasyonu üzerine kuruludur.

- **Authentication:** Kullanıcı doğrulaması sonrası sunucu tarafından imzalanmış bir JWT üretilir.
- **Authorization:** [Authorize] attribute'ları ve **role-based policies** ile kullanıcıların sadece yetkili oldukları **resources** erişimi sağlanır.

4. Frontend Teknik Analizi

4.1 React ve Vite ile Modern Web Geliştirme

Edunex frontend mimarisi, yüksek performans sunan **Vite** üzerine inşa edilmiş bir **React** uygulamasıdır. Vite tercih edilmesinin temel sebebi, **HMR** hızı ve optimize edilmiş **build** süreçleridir. React'ın **component-based** yapısı, kodun yeniden kullanılabilirliğini en üst düzeye çıkarmıştır.

4.2 TypeScript ile Tip Güvenliği ve Arayüzler

Projenin tamamında **TypeScript** kullanımı zorunlu kılınmıştır. Bu tercih, **runtime** hatalarının henüz geliştirme aşamasında tespit edilmesini sağlar. Özellikle API'den dönen karmaşık DTO yapıları için tanımlanan *Interface* ve *Type* tanımları, uygulama içinde tutarlı bir veri akışı sağlar.

4.3 Material UI ile Bileşen Tabanlı Tasarım

Arayüz tasarımında endüstri standardı olan **MUI** kütüphanesi kullanılmıştır. **Material Design** ilkelerini temel alan bu kütüphane sayesinde; hem estetik hem de **accessibility** standartlarına uygun bir deneyim sunulmuştur. **Responsive Layout** stratejisiyle, uygulamanın tüm cihazlarda sorunsuz çalışması garanti altına alınmıştır.

4.4 React Query ile Durum ve Cache Yönetimi

Uygulamanın sunucu tarafındaki verilerini yönetmek için **React Query** kullanılmıştır.

- **Caching:** Mükerrer API istekleri önlenerek network trafiği optimize edilmiştir.
- **Loading/Error States: Async** süreçlerdeki bekleme ve hata durumları merkezi bir yapıda yönetilir.
- **Stale-While-Revalidate:** Kullanıcıya eski veri gösterilirken arka planda **background refetch** yapılarak kesintisiz bir deneyim sağlanır.

4.5 API Client ve Interceptor Yapılandırması

Backend ile iletişim, **Axios** kütüphanesi üzerine kurulu özelleştirilmiş bir istemci üzerinden yürütülmektedir.

- **Request Interceptor:** Her istekte **localStorage** üzerindeki güncel JWT otomatik olarak header'a eklenir.
- **Response Interceptor:** Sunucudan dönen 401 veya 403 hataları merkezi olarak yakalanır ve kullanıcı **logout** yapılarak yönlendirilir.

5. Veri Akışı ve İşleme Döngüsü

5.1 Request Lifecycle

Edunex sisteminde bir istemci isteği, katı bir hiyerarşik rotayı takip eder. Döngü; frontend tarafında tetiklenir, Axios üzerinden API katmanına ulaşır. API katmanında **Middleware** yapıları tarafından karşılanan istek, geçerli bulunursa Application katmanındaki ilgili servise iletilir. Servis, Domain entity'lerini kullanarak **business logic** yürütür ve Infrastructure katmanı üzerinden **database operations** tamamlar. Yanıt, aynı rotayı DTO formatına dönüştürülerek tersine izler.

5.2 Kurum Kayıt ve Lisanslama Süreç Analizi

Sistemin en kritik iş akışlarından biri olan "Kurum Kayıt" süreci, lisanslama mekanizmasıyla sıkı sıkıya bağlıdır:

- **Validation:** Gelen veriler Application katmanında doğrulanır.
- **Tenant Assignment:** Yeni kuruma benzersiz bir `TenantId` atanır.
- **License Assignment:** Kurum türüne göre `EdumanLicense` kaydı oluşturulur.
- **Atomic Operation:** Tüm bilgiler **Unit of Work** sayesinde tek bir transaction içinde kaydedilir; hata durumunda **rollback** gerçekleştirilir.

5.3 Global Exception Handling

Sistemde hata yönetimi, merkezi bir **Global Exception Handler** middleware'i üzerinden yürütülür. Bu yapı, oluşan tüm hataları yakalar ve **RFC 7807 (Problem Details)** standardına uygun bir JSON çıktısı üretir. Bu sayede frontend tarafı, hatanın tipini ve kaynağını standart bir formatta alır.

5.4 Audit Interceptor Mekanizması

Veri güvenliği ve **traceability** için geliştirilen **Audit Interceptor**, veritabanı seviyesinde çalışan bir kancadır:

- **Added State:** CreatedAt alanına sunucu saati işlenir.
- **Modified State:** UpdatedAt alanı güncellenir.
- **Identity Integration:** CreatedBy ve UpdatedBy alanlarına **HTTP Context** üzerinden alınan kullanıcı kimliği mühürlenir.

6. Lisans ve Tenant Yönetimi

6.1 Multi-Tenant Architecture

Edunex, **multi-tenant** mimari yapısını benimseyerek, tek bir **instance** üzerinden birden fazla kuruma hizmet verebilmektedir. Veri izolasyonu, **Logical Separation** yöntemiyle sağlanmıştır. Tablolardaki **TenantId** sütunu sayesinde veriler birbirinden izole edilmiştir.

Sistemdeki sorgular, **Global Query Filters** ile otomatik olarak filtrelendirir. Bu sayede bir tenant yöneticisi, başka bir tenant'ın verilerine asla erişemez.

6.2 License Types and Usage Metrics

Lisanslama modülü, kurumların kullanım haklarını dinamik olarak denetler. **EdumanLicense** üzerinden iki temel tür tanımlanmıştır:

- **Demo License:** Kısa süreli ve kısıtlı kapasiteye sahip paket.
- **Standard License:** Kurumun ihtiyacına göre belirlenen tam sürüm paket.

Lisans denetimi, **User Create** gibi işlemlerde tetiklenir ve **UserLimit** aşımı durumunda işlem engellenir.

6.3 RBAC ve Güvenlik Politikaları

Yetkilendirme mekanizması, **RBAC** prensiplerine dayanır. İzinler rollere, kullanıcılar ise rollere atanır. Sistemdeki temel roller:

- **SuperAdmin:** Tüm kurumları ve lisansları yönetme yetkisine sahip kullanıcı.
- **TenantAdmin:** Sadece kendi kurumuna bağlı birimleri yönetebilir.
- **Viewer:** Sadece raporlama ve dashboard ekranlarına erişebilen kısıtlı kullanıcı.

Erişim kontrolleri, backend tarafında **Authorize** nitelikleri ile, frontend tarafında ise **Route Guards** ile çift aşamalı olarak doğrulanır.

7. Sonuç ve Gelecek Çalışmalar

7.1 Projenin Kazanımları

Edunex projesi süresince elde edilen temel kazanımlar şunlardır:

- **Modularity:** N-Layer mimari sayesinde bağımsız geliştirilebilen bir yapı kurulmuştur.
- **Data Isolation:** Multi-tenant yapı ile kurumsal veri gizliliği korunmuştur.
- **UX:** React ve MUI entegrasyonu ile modern ve **responsive** bir arayüz sunulmuştur.
- **Automation:** Audit mekanizması ile tüm hareketlerin kaydı otomatikleştirilmiştir.

7.2 Scalability ve Maintainability

Sistem, gelecekteki büyüme potansiyelini karşılayacak şekilde tasarlanmıştır:

- **Horizontal Scaling:** Stateless yapı ve JWT kullanımı, **load balancer** desteğine olanak tanır.
- **Clean Code:** Shared Kernel ve merkezi hata yönetimi, bakım maliyetlerini minimize etmektedir.

7.3 Future Work ve EventBus Entegrasyonu

Edunex için planlanan gelecek geliştirmeler:

- **EDA:** Hazır bulunan `Edunex.EventBus` katmanı devreye alınarak **async** entegrasyonlar sağlanacaktır.
- **Mobile App:** Mevcut API kullanılarak **cross-platform** bir mobil uygulama geliştirilecektir.
- **AI Reporting:** Dashboard katmanına tahminleme yapan regresyon modelleri entegre edilecektir.