



**İSTANBUL TİCARET ODASI SOFTITO
BACKEND GELİŞTİRME EĞİTİMİ
BİTİRME PROJESİ RAPORU**

Hastane Randevu Sistemi

Faruk Furkan Özduran

Temmuz, 2026

Softito Yazılım Bilişim Akademisi, İstanbul

İÇİNDEKİLER

1. GİRİŞ
 - 1.1. Projenin Amacı
 - 1.2. Proje Kısıtları
2. MATERYAL VE YÖNTEM
 - 2.1. Backend ve ORM Stratejisi
 - 2.2. Frontend ve Entegrasyon
 - 2.3. Performans Optimizasyonu
 - 2.4. Güvenlik
 - 2.5.Raporlama Çıktıları
3. KURULUM
 - 3.1. Veritabanı Konfigürasyonu
 - 3.2. Migration İşlemleri
 - 3.3. Başlatma Ayarları
4. SONUÇ
5. KAYNAKÇA

1. GİRİŞ

Bu bölümde projenin amacı ve projede karşılaşılan kısıtlara yer verilmiştir.

1.1 Projenin Amacı

Softlto Hospital projesinin temel amacı, bir hastanenin klinik, doktor, hasta, randevu ve faturalandırma gibi tüm kritik yönetim süreçlerini uçtan uca dijitalleştiren, yüksek performanslı ve modern bir otomasyon sistemi geliştirmektir. Sistem; hasta kayıtlarının güvenli oluşturulmasını, hastaların kendi adlarına randevu alabilmesini, yöneticilerin ise hastane gelirlerini, klinik yoğunluklarını ve diğer istatistikleri gelişmiş dinamik dashboard ekranları üzerinden anlık olarak takip edebilmesini hedefler.

1.2. Proje Kısıtları

- **Gevşek Bağlı (Decoupled) Mimari Zorunluluğu:** Kullanıcı arayüzünü (UI) barındıran MVC katmanının veritabanı ile doğrudan temas kurmaması, tüm verilerin bağımsız bir Web API üzerinden HTTP çağrıları ile yönetilmesi kısıtı.
- **Frontend Teknolojisi:** React veya Angular gibi modern Javascript framework'leri kullanılmadan; geleneksel ASP.NET Core MVC (Razor), Vanilla JS ve jQuery AJAX kullanılarak dinamik bir SPA (Single Page Application) hissi yaratılması gerekliliği.
- **Ayrık Kimlik Yönetimi:** ASP.NET Core Identity sistemi ile sistemdeki iş (Domain) modellerinin (Hasta, Doktor) doğrudan birbirine bağlı olmaması durumu ve bu durumun veri bütünlüğü çerçevesinde (IdentityUserId eşleştirmesiyle) çözülmesi zorunluluğu.

2. MATERYAL VE YÖNTEM

Projenin geliştirilme sürecinde modern yazılım mühendisliği prensiplerinden olan Çok Katmanlı Mimari (N-Tier Architecture) uygulanmıştır. Proje; Core, Data, Business, API ve UI olmak üzere 5 ana katmana bölünerek kod tekrarının ve karmaşanın (Spaghetti Code) önüne geçilmiştir.

2.1 Backend ve ORM Stratejisi

NET Core altyapısında geliştirilen Web API katmanında sektörde nadir görülen "Dual-ORM" stratejisi benimsenmiştir. Klasik C.R.U.D. (Ekleme/Okuma/Güncelleme/Silme) işlemleri ve ilişkisel veriler için Entity Framework Core kullanılırken; yüksek okuma hızı ve performans gerektiren ham SQL sorguları için Dapper mikro-ORM'si sisteme entegre edilmiştir.

2.2 Frontend ve Entegrasyon

Kullanıcı arayüzü ASP.NET Core MVC mimarisinde tasarlanmıştır. API haberleşmeleri IHttpConnectionFactory üzerinden gerçekleştirilerek JSON formatında işlenmiştir. Arayüzde özel CSS kuralları, dinamik "Sticky Header" tasarımları ve AJAX tabanlı asenkron yüklemeler (Örn: Kliniğe göre doktor filtreleme) kullanılmıştır.

2.3 Performans Optimizasyonu

Yöneticilerin anasayfasında bulunan ve veritabanını yoran ağır istatistiksel hesaplamalar (Dashboard Raporları), IMemoryCache kullanılarak RAM üzerinde önbelleklenmiş ve sunucu maliyetleri minimize edilmiştir.

2.4 Güvenlik

Sistem güvenliği, ASP.NET Core Identity kullanılarak Rol tabanlı (Admin, Patient vb.) ve Token tabanlı (JWT - JSON Web Token) mimariyle sağlanmıştır.

2.5 Raporlama Çıktıları

Generic bir yapı ile kodlanan ReportService sınıfı üzerinden, sistemdeki tüm listelerin iTextSharp ve Excel kütüphaneleri kullanılarak tek tıkla PDF ve Excel formatında dışa aktarılması sağlanmıştır.

3. KURULUM

Projenin yerel veya sunucu ortamında çalıştırılabilmesi için aşağıdaki yapılandırma adımları izlenmelidir.

3.1 Veritabanı Konfigürasyonu

Softlto.Hospital.API projesi içerisinde bulunan appsettings.json dosyasındaki DefaultConnection bağlantı dizesi (Connection String), kullanılacak Microsoft SQL Server ayarlarına göre güncellenir.

3.2 Migration İşlemleri

Visual Studio üzerinden Package Manager Console açılarak "Default Project" olarak Softlto.Hospital.Data katmanı seçilir. Update-Database komutu çalıştırılarak Code-First yaklaşımıyla veritabanı tablolarının oluşturulması sağlanır.

3.3 Başlatma Ayarları

Proje mimarisi gereği API ve UI katmanlarının aynı anda ayağa kalkması gerekmektedir. Solution özelliklerinden "Multiple Startup Projects" seçeneği işaretlenerek API ve UI projelerinin başlatma eylemleri "Start" konumuna getirilir.

4. SONUÇ

Softlto Hospital Yönetim Sistemi, kurumsal (Enterprise) seviyede yazılım mimarisi standartlarına uygun, güvenli ve ölçeklenebilir bir altyapıyla başarıyla tamamlanmıştır.

Projede uygulanan "Katmanlı Mimari" ve "Ayrık (Decoupled) UI" yapısı sayesinde; sistem gelecekteki potansiyel eklentilere (örneğin Android ve iOS için mobil uygulama yazılması) sıfır backend eforuyla entegre edilebilecek durumdadır. Hastaların sistem üzerinden kendi profillerini tanımlayıp asenkron randevu alabilmeleri, yöneticilerin ise platform genelindeki finansal ve operasyonel verileri gelişmiş dashboardlar üzerinden takip edebilmeleri sağlanmıştır. Modern kullanıcı arayüzü ve sağlam backend altyapısı bir araya getirilerek uçtan uca eksiksiz bir sağlık otomasyonu elde edilmiştir.