



softITo BİTİRME PROJESİ RAPORU

GASTRORESERVE

Akıllı Restoran Rezervasyon & Yönetim Sistemi

Hibrit EF Core & Dapper Mimarisi ile ASP.NET Core Web API ve MVC Uygulaması

Baha Lor

Temmuz, 2026

softITo Yazılım Bilişim Akademisi, İstanbul

İÇİNDEKİLER

1. GİRİŞ

- 1.1. Projenin Amacı
- 1.2. Proje Kapsamı ve Kısıtları

2. MATERYAL VE YÖNTEM

- 2.1. Kullanılan Teknolojiler
- 2.2. Mimari Yaklaşım: N-Katmanlı (N-Tier) Mimari
- 2.3. Hibrit ORM Yaklaşımı: Entity Framework Core & Dapper

3. SİSTEM TASARIMI

- 3.1. Veri Modeli ve Varlıklar (Entities)
- 3.2. Kimlik Doğrulama ve Yetkilendirme (JWT)

4. UYGULAMA GELİŞTİRME SÜRECİ

- 4.1. API Katmanları: EF Core API & Dapper API
- 4.2. Web Arayüzü (ASP.NET Core MVC)
- 4.3. Şubeye Özel Salon Yerleşim Planı
- 4.4. Akıllı Ödeme Maskeleme
- 4.5. Dapper ile Performans Raporlama ve Canlı Log Akışı

5. TEST VE DOĞRULAMA

- 5.1. Swagger Üzerinden API Testi
- 5.2. Uygulama Arayüzü Üzerinden Test

6. SONUÇ

7. KAYNAKÇA

1. GİRİŞ

1.1. Projenin Amacı

GastroReserve, çok şubeli restoran işletmelerinin masa rezervasyonu, sipariş ve ödeme süreçlerini, şube bazlı performans analitiğini ve sistem loglarını tek bir çatı altında yönetebilmesi amacıyla geliştirilmiş uçtan uca bir web uygulamasıdır. Proje, softITo Yazılım Bilişim Akademisi Bitirme Projesi kapsamında; katmanlı (N-Tier) mimari, çoklu ORM kullanımı ve token tabanlı kimlik doğrulama gibi kurumsal düzeyde .NET geliştirme pratiklerini uygulamalı olarak göstermek amacıyla tasarlanmış ve kodlanmıştır.

Projenin temel motivasyonu, klasik tek-ORM'li CRUD uygulamalarının ötesine geçerek; günlük operasyonel işlemlerde geliştirici verimliliğini önceleyen Entity Framework Core ile, yoğun okuma/raporlama senaryolarında performansı önceleyen Dapper'ı aynı sistem içinde bir arada kullanan hibrit bir mimari ortaya koymaktır.

1.2. Proje Kapsamı ve Kısıtları

Sistem; şube, masa, ürün, rezervasyon, sipariş, ödeme, çalışan, kullanıcı ve değerlendirme (review) süreçlerini kapsayacak şekilde tasarlanmıştır. Kimlik doğrulama JWT (JSON Web Token) ile sağlanmakta, rol tabanlı yetkilendirme (Admin / Employee / Customer) ile hassas uç noktalar korunmaktadır.

- Kapsam dahilinde: şube ve masa yönetimi, rezervasyon akışı, sipariş/ödeme kaydı, Dapper tabanlı yönetim paneli raporlaması, Excel/PDF dışı aktarım.
- Kapsam dışında: online ödeme altyapısına gerçek bir sanal pos entegrasyonu (kart bilgileri yalnızca arayüzde biçimlendirilip maskelenmektedir, gerçek bir ödeme sağlayıcısına gönderilmemektedir).
- Proje eğitim amaçlı geliştirilmiş olup yerel bir MS SQL Server örneği üzerinde çalışacak şekilde yapılandırılmıştır.

2. MATERYAL VE YÖNTEM

2.1. Kullanılan Teknolojiler

Projede aşağıdaki teknoloji ve kütüphaneler kullanılmıştır:

- **.NET 10.0** — tüm katmanların hedef çatısı
- **ASP.NET Core Web API** — GastroReserve.EfApi, GastroReserve.DapperApi ve raporlama uçlarını barındıran hibrit API
- **ASP.NET Core MVC** — GastroReserve.Web, son kullanıcı ve yönetim arayüzü
- **Entity Framework Core 10 (SQL Server)** — Code-First model, migration'lar ve temel CRUD işlemleri
- **Dapper 2.1.79** — ham SQL sorguları ile yüksek performanslı raporlama ve dashboard metrikleri
- **Microsoft SQL Server** — ilişkisel veri tabanı
- **JWT Bearer Authentication** — token tabanlı kimlik doğrulama ve rol bazlı yetkilendirme
- **AutoMapper 16.2.0** — Entity ↔ DTO dönüşümleri

- **Swashbuckle / Swagger 6.6.2** — API dokümantasyonu ve JWT destekli canlı test arayüzü
- **EPPlus & iText7** — Excel ve PDF formatında rapor dışı aktarımı
- **FluentValidation.AspNetCore** — istek (request) doğrulama kuralları
- **Lucide Icons, Bootstrap, jQuery** — arayüz bileşenleri ve HSL tabanlı altın (gold) temalı özel CSS

2.2. Mimari Yaklaşım: N-Katmanlı (N-Tier) Mimari

Sistem, sorumlulukların net biçimde ayrıldığı katmanlı bir mimariyle organize edilmiştir. Her katman yalnızca bir alt katmana bağımlıdır; bu sayede veri erişim teknolojisi (EF Core / Dapper) değiştirilse dahi iş kuralları ve sunum katmanı etkilenmez.

GastroReserve.Core IUnitOfWork)	→ Entity sınıfları ve arayüzler (IRepository,
GastroReserve.DataAccess Repository/UnitOfWork,	→ AppDbContext (EF Core), DapperContext,
DbInitializer	DapperRepository/DapperUnitOfWork, Migrations,
GastroReserve.Business AutoMapper profili,	→ DTO'lar, servis arayüzleri/implementasyonları,
	SecurityHelper (parola hash'leme)
GastroReserve.EfApi	→ EF Core tabanlı Web API (JWT + Swagger)
GastroReserve.DapperApi	→ Dapper tabanlı Web API
GastroReserve.API uçları)	→ Hibrit API + DapperReportController (Admin raporlama
GastroReserve.Web ExportController)	→ ASP.NET Core MVC arayüzü (Razor Views,

2.3. Hibrit ORM Yaklaşımı: Entity Framework Core & Dapper

Temel CRUD işlemleri (şube, masa, ürün, rezervasyon, sipariş oluşturma/güncelleme) Entity Framework Core'un Generic Repository ve Unit of Work desenleri üzerinden yürütülmektedir. Buna karşılık, yönetim panelindeki özet metrikler, şube bazlı istatistikler ve son rezervasyon listeleri gibi salt-okunur, JOIN ağırlıklı sorgular Dapper ile yazılmış ham SQL komutlarıyla çalıştırılmaktadır. Bu ayırım, EF Core'un sağladığı geliştirme hızından ödün vermeden, raporlama tarafında sorgu planı üzerinde tam kontrol ve düşük gecikme elde edilmesini sağlamıştır.

3. SİSTEM TASARIMI

3.1. Veri Modeli ve Varlıklar (Entities)

Veri modeli 14 ana varlık üzerine kurulmuştur: Role, User, Employee, Branch, Table, Category, Product, Reservation, Order, OrderDetail, Modifier, Payment, Review ve Favorite. İlişkiler AppDbContext içerisinde Fluent API ile açıkça tanımlanmıştır.

- **User - Role:** çokla bir (N:1) ilişki; her kullanıcının bir rolü (Admin/Employee/Customer) vardır.
- **Employee - User:** bire bir (1:1) ilişki; personel kaydı bir kullanıcı hesabını genişletir.
- **Branch - Table - Reservation:** bir şubenin birden çok masası, bir masanın birden çok rezervasyonu olabilir.
- **Order - OrderDetail - Modifier:** bir sipariş birden çok kalemden, her kalem birden çok ek malzemeden (modifier) oluşabilir.
- **Order - Payment:** bir siparişe birden fazla ödeme kaydı (kısmi ödeme senaryoları için) düşebilir.

Silme davranışları (DeleteBehavior) ilişki tipine göre özenle seçilmiştir: örneğin bir şube silindiğinde masaları Cascade ile silinirken, bir müşteri silindiğinde geçmiş rezervasyon kayıtlarının bütünlüğünü korumak için Restrict davranışı tercih edilmiştir.

3.2. Kimlik Doğrulama ve Yetkilendirme (JWT)

Kullanıcı kaydı ve girişi AuthController üzerinden yürütülür. Giriş başarılı olduğunda kullanıcı bilgileri ve rolü içeren imzalı bir JWT üretilir; bu token sonraki isteklerde Authorization: Bearer {token} başlığı ile taşınır. Program.cs içerisinde JwtBearer şeması varsayılan doğrulama/karşılama (authenticate/challenge) şeması olarak tanımlanmış; Issuer, Audience ve imza anahtarı appsettings üzerinden okunmaktadır. Rol bazlı yetkilendirme sayesinde örneğin şube oluşturma/güncelleme veya Dapper raporlama uçları yalnızca Admin rolüne sahip kullanıcılara açıktır.

```
builder.Services.AddAuthentication(options => {
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(options => {
    options.TokenValidationParameters = new TokenValidationParameters {
        ValidateIssuer = true, ValidateAudience = true,
        ValidateLifetime = true, ValidateIssuerSigningKey = true,
        ValidIssuer = jwtIssuer, ValidAudience = jwtAudience,
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey))
    };
});
```

4. UYGULAMA GELİŞTİRME SÜRECİ

4.1. API Katmanları: EF Core API & Dapper API

GastroReserve.EfApi projesinde AuthController, ReservationController (şube/masa/rezervasyon uçları), ProductController ve OrderController yer almaktadır. Her controller ilgili servis arayüzü (IReservationService, IProductService, IOrderService, IUserService) üzerinden Business katmanına, oradan da UnitOfWork aracılığıyla DataAccess katmanına bağlanır. Aynı uç nokta sözleşmesi GastroReserve.DapperApi projesinde Dapper tabanlı repository'ler kullanılarak yeniden uygulanmış; böylece iki farklı veri erişim stratejisinin aynı iş kuralları üzerinde nasıl davrandığı karşılaştırmalı olarak gözlemlenebilmiştir.

Uygulama başlangıcında (Program.cs) veritabanı DbInitializer.SeedAsync ile otomatik olarak doldurulur: 3 şube (Gastro Merkez Şube – Kadıköy, Gastro Cadde Şube – Bağdat Caddesi, Gastro Marina VIP – Pendik Marina), 4 ürün kategorisi (Başlangıçlar, Ana Yemekler, İçecekler, Tatlılar) ve şubeye özel kapasitelerde masalar seed edilir.

4.2. Web Arayüzü (ASP.NET Core MVC)

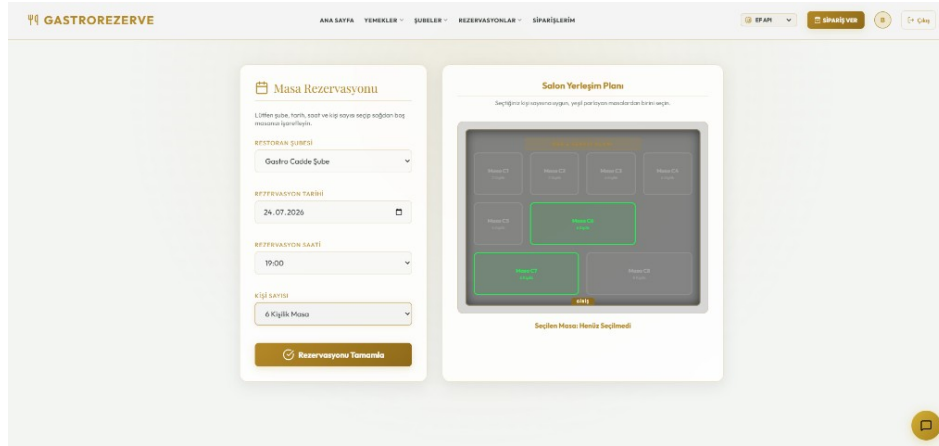
GastroReserve.Web projesi, müşteri ve yönetici deneyimini Razor View'lar üzerinden sunar. Ana sayfa; şube seçimi, menü vitrin görselleri ve rezervasyon çağrısını (call-to-action) tek ekranda bir araya getirecek şekilde, Lucide ikon seti ve özel bir altın (gold) tonlu HSL renk paletiyle tasarlanmıştır.



Şekil 4.2.1. GastroReserve Ana Sayfa

4.3. Şubeye Özel Salon Yerleşim Planı

Uygulamanın öne çıkan özelliklerinden biri, her şube için ayrı kurgulanmış interaktif masa yerleşim krokisidir. Merkez, Cadde ve Marina VIP şubeleri, kendi masa sayısı ve kapasite dağılımına göre (2, 4, 6 ve 8 kişilik masalar) ayrı bir kroki üzerinde görselleştirilir. Kullanıcı rezervasyon için kişi sayısını girdiğinde, o kişi sayısına uygun ve müsait masalar krokide yeşil renkle vurgulanarak öne çıkarılır; bu da masa seçim sürecini sezgisel hale getirir.



Şekil 4.3.1. İnteraktif Salon Yerleşim Planı

4.4. Akıllı Ödeme Maskeleye

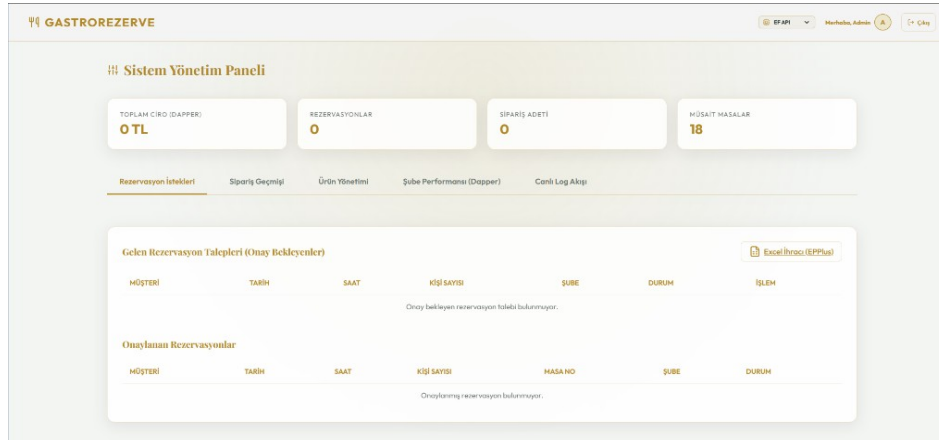
Sipariş kapama ekranında yer alan ödeme formunda; kredi kartı numarası 4'erli gruplar halinde otomatik boşluklanarak biçimlendirilir, son kullanma tarihi AA/YY kalıbına göre yönlendirilir ve CVC alanı yalnızca 3 haneli sayısal girişe izin verecek şekilde regex tabanlı kısıtlarla sınırlandırılır. Bu katman tamamen istemci tarafında (client-side) çalışır; girilen kart bilgileri harici bir ödeme sağlayıcısına iletilmez, yalnızca kullanıcı deneyimini gerçekçi kılmak amacıyla biçimlendirilir.

4.5. Dapper ile Performans Raporlama ve Canlı Log Akışı

Yönetim paneli, DapperReportController üzerinden sunulan ve yalnızca Admin rolüne açık üç uç noktayla beslenir: dashboard-metrics (toplam rezervasyon, sipariş, gelir, ürün ve müsait masa sayısını tek sorguda özetler), branch-stats (şube bazlı rezervasyon/sipariş/gelir kırılımını JOIN ve GROUP BY ile hesaplar) ve recent-reservations (son 10 rezervasyonu müşteri, masa ve şube bilgisiyle birlikte listeler). Bu uçların tamamı IDbConnection üzerinden çalıştırılan ham SQL sorgularıdır ve admin panelinde

gerçek zamanlı, monospace bir konsol kutusunda çalıştırılan sorgu ve sistem loglarıyla birlikte görüntülenir.

```
[HttpGet("dashboard-metrics")]
public async Task<IActionResult> GetDashboardMetrics()
{
    using var db = _dapperContext.CreateConnection();
    var totalReservations = await db.ExecuteScalarAsync<int>(
        "SELECT COUNT(*) FROM Reservations");
    var totalRevenue = await db.ExecuteScalarAsync<decimal>(
        "SELECT COALESCE(SUM(TotalAmount),0) FROM Orders WHERE Status='Closed'");
    // ... totalOrders, totalProducts, activeTables
    return Ok(new { totalReservations, totalRevenue, /* ... */ });
}
```



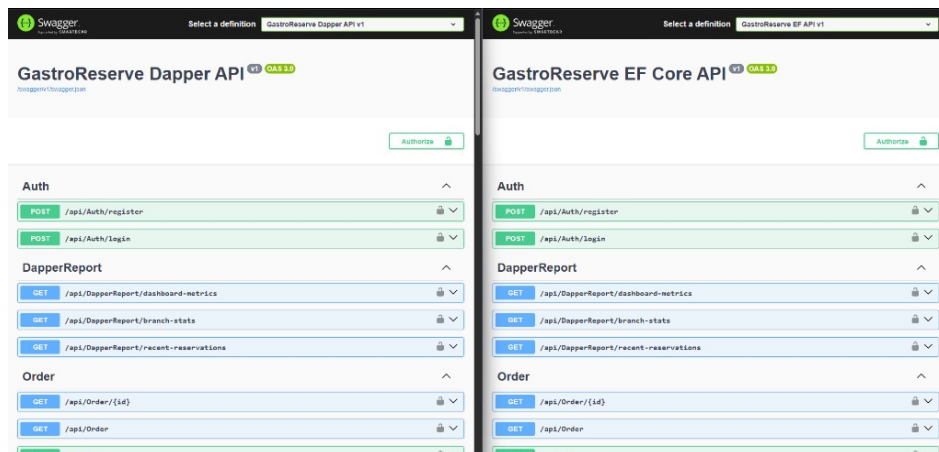
Şekil 4.5.1. Yönetim Paneli — Şube İstatistikleri ve Canlı Log Akışı

Ayrıca GastroReserve.Web içerisindeki ExportController, üretilen raporları EPPlus ile Excel (.xlsx) ve iText7 ile PDF formatında dışa aktarma imkânı sunmaktadır.

5. TEST VE DOĞRULAMA

5.1. Swagger Üzerinden API Testi

GastroReserve.EfApi ve GastroReserve.DapperApi projeleri, Swashbuckle ile üretilen Swagger arayüzü üzerinden test edilmiştir. Swagger yapılandırmasına JWT için özel bir güvenlik tanımı (Bearer scheme) eklenmiş; böylece /api/Auth/login ucundan alınan token doğrudan Swagger arayüzüne girilerek yetkilendirme gerektiren uçlar (şube oluşturma, Dapper raporlama vb.) canlı olarak denenebilmiştir.



Şekil 5.1.1. Swagger API Dokümantasyonu ve JWT Yetkilendirme Testi

5.2. Uygulama Arayüzü Üzerinden Test

API seviyesindeki testlerin ardından uçtan uca senaryolar (kullanıcı kaydı → giriş → şube/masa seçimi → rezervasyon oluşturma → sipariş ve ödeme → admin panelinde metriklerin güncellenmesi) tarayıcı üzerinden manuel olarak yürütülmüş; DbInitializer ile seed edilen veriler üzerinde tüm CRUD akışlarının ve raporlama uçlarının beklenen sonuçları döndürdüğü doğrulanmıştır.

6. SONUÇ

Bu çalışmada, çok şubeli bir restoran işletmesinin rezervasyon, sipariş ve raporlama ihtiyaçlarını karşılayan GastroReserve adlı bir web uygulaması uçtan uca geliştirilmiştir. Katmanlı mimari sayesinde iş kuralları veri erişim teknolojisinden bağımsız hale getirilmiş; Entity Framework Core ile Dapper'ın aynı sistem içinde farklı sorumluluklarda (CRUD vs. raporlama) bir arada kullanılabildiği hibrit bir yaklaşım pratik olarak uygulanmıştır.

JWT tabanlı kimlik doğrulama ve rol bazlı yetkilendirme ile hassas işlemler korunmuş; şubeye özel interaktif salon krokisi ve akıllı ödeme maskeleme gibi kullanıcı deneyimini önceleyen ek özellikler sisteme entegre edilmiştir. Sonuç olarak proje, yalnızca CRUD tabanlı bir uygulama değil; kimlik doğrulama, çoklu veri erişim stratejisi, raporlama ve dışa aktarım gibi kurumsal bir uygulamada beklenen bileşenleri bir arada barındıran kapsamlı bir bitirme projesi olarak tamamlanmıştır.

7. KAYNAKÇA

Microsoft Learn — ASP.NET Core dokümantasyonu, <https://learn.microsoft.com/aspnet/core>

Microsoft Learn — Entity Framework Core dokümantasyonu, <https://learn.microsoft.com/ef/core>

Dapper — resmi GitHub deposu ve dokümantasyonu, <https://github.com/DapperLib/Dapper>

Swashbuckle.AspNetCore — Swagger/OpenAPI entegrasyonu, <https://github.com/domaindrivendev/Swashbuckle.AspNetCore>

Proje kaynak kodu — <https://github.com/bahalar1/SoftITO>