



SoftITo 4. Dönem Backend Developer Eğitimi

FİZİK TEDAVİ TAKİP VE YÖNETİM
SİSTEMİ PROJE RAPORU

Esra Karaduman

İÇİNDEKİLER

1. GİRİŞ

- 1.1 Proje Amacı
- 1.2 Proje Kısıtları

2. MATERYEL VE YÖNETİM

- 2.1 Materyal (Kullanılan Teknolojiler)
- 2.2 Yönetim ve Yöntem (Mimari Yapı)

3. KURULUM

- 3.1 Sistem Gereksinimleri
- 3.2 Veritabanı Yapılandırması ve Migration
- 3.3 Servislerin (API) ve Arayüzün (MVC) Başlatılması
- 3.4 appsettings.json Yapılandırma Detayları

4. SONUÇ

- 4.1 Genel Değerlendirme ve Kazanımlar
- 4.2 Ekstra Entegrasyonlar ve Öne Çıkan Özellikler
- 4.3 Gelecek Çalışmalar ve Geliştirme Vizyonu

5. KAYNAKÇA

1. GİRİŞ

1.1 Proje Amacı

Modern sağlık kuruluşlarında ve özellikle fizik tedavi merkezlerinde operasyonel süreçlerin dijitalleştirilmesi, hastaların tedavi süreçlerinin yakından takip edilebilmesi ve idari işlerin minimum hata ile yönetilebilmesi kritik önem taşımaktadır. **Fizik Tedavi Takip ve Yönetim Sistemi (PhysicalTherapySystem)** projesinin temel amaçları şunlardır:

- **Hasta ve Tedavi Süreçlerinin Takibi:** Hastaların tanı, tedavi programı, atanan egzersizler, seans katılımları ve gelişim süreçlerini tek bir platformdan uçtan uca takip etmek.
- **Randevu Yönetimi:** Fizyoterapistlerin ve hastaların müsaitlik durumlarına göre randevu çakışmalarını önleyen dinamik bir randevu sistemi sunmak.
- **Hızlı ve Esnek Raporlama:** Faturalandırma, ödeme takibi ve tedavi gelişimlerine dair anlık verileri PDF ve Excel formatlarında dışa aktarabilmek.
- **Yapay Zeka Destekli Asistan (Chatbot):** Google Gemini 1.5 Flash modeli entegrasyonu ile hastalara ve terapistlere anlık tıbbi bilgilendirme ve yerel veritabanı analizine dayalı akıllı cevaplar sunmak.
- **Yüksek Performanslı Yazılım Mimarisi:** CQRS (Command Query Responsibility Segregation) prensiplerini benimseyerek, yazma (EF Core) ve okuma (Dapper) işlemlerini ayırıp sistemin yüksek yükler altında dahi performanslı ve kararlı çalışmasını sağlamak.

1.2 Proje Kısıtları

Projenin geliştirilmesinde, yayına alınmasında ve ölçeklenmesinde karşılaşılan temel teknik ve operasyonel kısıtlar şunlardır:

- **Yerel Veritabanı Bağımlılığı:** Projenin varsayılan olarak SQL Server LocalDB ((localdb)\MSSQLLocalDB) kullanması, canlı ortama (production) geçişte Azure SQL veya merkezi bir MSSQL Server konfigürasyonu gerektirmektedir.
- **Kimlik Doğrulama ve Güvenlik Sınırları:** JWT (JSON Web Token) tabanlı kimlik doğrulamanın çerezler (Cookies) üzerinden yönetilmesi, tarayıcı güvenlik ayarları (CORS, SameSite vb.) ve token ömrü yönetimi gibi entegrasyon hassasiyetlerini beraberinde getirir.
- **Yapay Zeka API Limitleri:** Gemini API entegrasyonunda internet bağlantısı bağımlılığı ve kota sınırları bulunmaktadır. Bu kısıtı aşmak adına, API anahtarı geçersiz olduğunda veya kota dolduğunda çalışmak üzere Dapper tabanlı akıllı yerel asistan mekanizması (Fallback) kurgulanmıştır.
- **Aşırı Veri Boyutları:** Çok sayıda yüksek çözünürlüklü egzersiz videosu veya geniş seans geçmişi saklanması durumunda veri tabanı boyutlarının şişmesini önlemek adına video saklama işlemleri doğrudan veritabanında değil, Youtube veya CDN linkleri üzerinden tutulacak şekilde kısıtlanmıştır.

2. MATERYEL VE YÖNETİM

2.1 Materyal (Kullanılan Teknolojiler)

Projenin mimari yapısını oluşturmak ve gereksinimleri karşılamak amacıyla endüstri standartlarında kabul görmüş aşağıdaki kütüphaneler ve teknolojiler kullanılmıştır:

- **Framework & Runtime:** .NET 10.0 SDK & C# 14.
- **Kullanıcı Arayüzü (Frontend):** ASP.NET Core MVC (Model-View-Controller), HTML5, CSS3 ve Vanilla Javascript.
- **ORM & Veri Erişim Katmanı (Yazma):** Entity Framework Core (SQL Server sağlayıcısı ile ilişkisel verilerin yönetimi, Code-First yaklaşımı).
- **Mikro ORM (Okuma & Raporlama):** Dapper (SQL sorgularını doğrudan çalıştırarak yüksek performanslı okuma işlemleri).
- **Raporlama Araçları:**
 - **QuestPDF:** Dinamik PDF fatura tasarımı ve oluşturulması (Community License ile).
 - **ClosedXML:** Tablo verilerinin Excel formatında formatlanarak indirilmesi.
 - **Yapay Zeka Entegrasyonu:** Google Gemini 1.5 Flash API (REST istemcisi üzerinden).
 - **Loglama Altyapısı:** Serilog (Konsol ve günlük bazlı dosya loglama (Logs/ef-api-log-.txt ve Logs/mvc-client-log-.txt)).
 - **QR Kod Oluşturucu:** QRCode (Faturaların doğrulanabilmesi için QR kod üretimi).
 - **Kimlik Doğrulama:** ASP.NET Core Identity ve JWT Bearer Yetkilendirmesi.

2.2 Yönetim ve Yöntem (Mimari Yapı)

Projede veri tutarlılığı ile sistem performansını optimize etmek amacıyla hibrit bir yaklaşım benimsenmiştir:

- **CQRS Ayrımı:**
- **Komutlar (Commands):** Veri ekleme, silme ve güncelleme işlemleri PhysicalTherapySystem.EF.Api (Port: 7001) üzerinden, Entity Framework Core ORM ve Unit of Work deseni kullanılarak gerçekleştirilir.
- **Sorgular (Queries):** Raporlama, arama ve listeleme gibi yüksek performans gerektiren okuma işlemleri PhysicalTherapySystem.Dapper.Api (Port: 7101) üzerinden, optimize edilmiş Dapper SQL sorguları ile yürütülür.
- **Katmanlı Mimari:** Model tanımlamaları (PhysicalTherapySystem.Models), Veri erişim katmanı (PhysicalTherapySystem.Data), API Katmanları ve son kullanıcı arayüzü (PhysicalTherapySystem.Mvc) birbirinden ayrılmıştır.
- **Proje Yönetimi:** Geliştirme sürecinde çevik (Agile) prensipler uygulanmış, veri tabanı şemasındaki değişiklikler EF Migrations aracılığıyla versiyonlanarak yönetilmiştir.
-

3. KURULUM

Projenin yerel geliştirme veya sunucu ortamında sorunsuz şekilde çalıştırılması için aşağıdaki adımların sırasıyla uygulanması gerekmektedir:

3.1 Sistem Gereksinimleri

- **İşletim Sistemi:** Windows 10/11, macOS veya Linux (Veritabanı için SQL Server/Docker).
- **SDK:** .NET 10.0 SDK veya üzeri.
- **Veritabanı:** LocalDB (Windows için Visual Studio ile birlikte kurulur) veya MS SQL Server / Docker MSSQL Image.
- **Editör:** Visual Studio 2022 (v17.10+ önerilir) veya Visual Studio Code.

3.2 Projenin İndirilmesi ve Dizin Düzeni

1. Projenin yer aldığı GitHub reposunu yerel bilgisayarınıza klonlayın:

```
git clone https://github.com/KULLANICI_ADI/PhysicalTherapySystem.git
```

2. Klonlanan proje dizinine geçiş yapın:

```
cd PhysicalTherapySystem
```

3.3 Veritabanı Yapılandırması ve Migration

Proje, veritabanını otomatik olarak oluşturma özelliğine sahiptir (`db.Database.EnsureCreated()`). Ancak şemanın güncellenmesi ve migration geçmişinin yönetilmesi için şu komutlar çalıştırılmalıdır:

1. Bir terminal penceresi açarak projenin kök (root) dizininde olduğunuzdan emin olun.
2. NuGet paketlerini geri yükleyin (Restore):

```
dotnet restore
```
3. Entity Framework Core CLI araçlarının sisteminizde kurulu olduğundan emin olun (Yoksa yüklemek için: `dotnet tool install --global dotnet-ef`).
4. Veritabanını en son migration seviyesine güncellemek ve başlangıç verilerini yüklemek için şu komutu çalıştırın:

```
dotnet ef database update --project PhysicalTherapySystem.Data --startup-project  
PhysicalTherapySystem.EF.Api
```

Not: Bu komut sonucunda SQL Server üzerinde 'PhysicalTherapyDb' adında bir veritabanı otomatik olarak oluşturulacak, tablolar kurulacak ve sistem için gerekli başlangıç verileri (Seed Data) yüklenecektir.

3.4 Servislerin (API) ve Arayüzün (MVC) Başlatılması

Projenin çalışması için 3 farklı uygulamanın aynı anda ayakta olması gerekmektedir.

Yöntem A: Visual Studio ile Çalıştırma (Önerilen)

1. PhysicalTherapySystem.slnx veya PhysicalTherapySystem.sln çözüm dosyasını Visual Studio ile açın.
2. Çözüm (Solution) üzerine sağ tıklayın ve "**Configure Startup Projects...**" (**Başlangıç Projelerini Yapılandır...**) seçeneğini seçin.

3. "Multiple Startup Projects" (Birden Çok Başlangıç Projesi) seçeneğini aktif hale getirin.

4. Aşağıdaki projeleri sırasıyla "Start" (Başlat) durumuna getirin:

- PhysicalTherapySystem.EF.Api (Yazma API'si)
- PhysicalTherapySystem.Dapper.Api (Okuma & Raporlama API'si)
- PhysicalTherapySystem.Mvc (Kullanıcı Arayüzü)

5. F5 veya Start tuşuna basarak tüm projeleri aynı anda başlatın.

Yöntem B: .NET CLI (Komut Satırı) ile Çalıştırma

Projenin kök dizinindeyken 3 farklı terminal sekmesi/penceresi açın ve sırasıyla şu komutları çalıştırın:

1. Terminal 1: EF API Servisi (Yazma)

```
cd PhysicalTherapySystem.EF.Api
dotnet run --urls=https://localhost:7001
```

2. Terminal 2: Dapper API Servisi (Okuma)

```
cd PhysicalTherapySystem.Dapper.Api
dotnet run --urls=https://localhost:7101
```

3. Terminal 3: MVC Web Uygulaması (Arayüz)

```
cd PhysicalTherapySystem.Mvc
dotnet run --urls=https://localhost:7201
```

3.4 appsettings.json Yapılandırma Detayları

Projenin doğru çalışabilmesi için appsettings.json dosyalarındaki ayarların doğrulanması gerekir.

- **EF API (PhysicalTherapySystem.EF.Api/appsettings.json):**

```
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=(localdb)\\MSSQLLocalDB;Database=PhysicalTherapyDb;Trusted_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True;"
  },
  "Jwt": {
    "Key": "PhysicalTherapySystemSecretSuperKey2026!",
    "Issuer": "PhysicalTherapyEF",
    "Audience": "PhysicalTherapyClient"
  }
}
```

- **Dapper API (PhysicalTherapySystem.Dapper.Api/appsettings.json):**

```
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=(localdb)\\MSSQLLocalDB;Database=PhysicalTherapyDb;Trusted_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True;"
  },
  "AiSettings": {
```

```
"ApiKey": "YOUR_GEMINI_API_KEY_HERE"  
}  
}
```

(Not: "YOUR_GEMINI_API_KEY_HERE" alanına geçerli bir Google Gemini API anahtarı girildiğinde yapay zeka asistanı aktif olur. Boş bırakıldığında ise Dapper tabanlı yerel asistan çalışmaya devam eder.)

4. SONUÇ

4.1 Genel Değerlendirme ve Kazanımlar

Geliştirilen Fizik Tedavi Takip ve Yönetim Sistemi, klinik yönetim süreçlerini tek bir çatı altında birleştirerek operasyonel verimliliği en üst seviyeye çıkarmıştır. Sistem sayesinde:

- Randevu çakışmaları sıfıra indirilmiştir.
- Hastaların seans katılımları ve egzersiz programlarındaki ilerlemeler grafiksel ve nicel verilerle anlık olarak izlenebilir hale gelmiştir.
- Fatura oluşturma ve ödeme takibi süreçleri otomatikleştirilmiş, fatura doğrulama işlemleri için QR Kod entegrasyonu başarıyla sağlanmıştır.
- CQRS benzeri mimari tasarım sayesinde veri yazma ve veri okuma süreçleri birbirinin performansını etkilemeyecek şekilde optimize edilmiş; Dapper kullanımı ile listeleme ekranlarındaki bekleme süreleri milisaniyeler seviyesine çekilmiştir.

4.2 Ekstra Entegrasyonlar ve Öne Çıkan Özellikler

Projede standart yönetim sistemlerinin ötesinde şu fark yaratan teknik ve fonksiyonel özellikler uygulanmıştır:

- **Hibrit Yapay Zeka (Gemini & Dapper Fallback) Chatbot:** Uygulamaya entegre edilen akıllı asistan, kullanıcıların sorularına anında yanıt verebilmektedir. Eğer internet bağlantısı yoksa veya API anahtarı girilmemişse, sistem hata vermek yerine arka planda Dapper ile veritabanını analiz ederek dinamik Türkçe yanıtlar ("Klinikte şu an X aktif hasta, Y randevu bulunmaktadır" gibi) üretmektedir.
- **Dinamik Belge Üretimi:** QuestPDF ile bellek dostu (Memory Friendly) ve hızlı PDF fatura çıktısı üretilmekte ve bu faturalar üzerine benzersiz QR Kodlar basılarak fatura doğrulama mekanizması sunulmaktadır.
- **Detaylı Loglama (Serilog):** Uygulamadaki hataların, yetkisiz erişim denemelerinin ve veri işlemlerinin takibi Serilog ile yapılandırılmış; hata tespit süreleri minimuma düşürülmüştür.
- **Otomatik Veri Besleme (Data Seeding):** Projenin ilk açılışında gerçekçi hasta, egzersiz, terapist ve randevu verilerinin otomatik olarak veri tabanına yazılması sağlanmış, böylece sistemin doğrudan test edilebilir bir yapıda kurulması kolaylaştırılmıştır.

4.3 Gelecek Çalışmalar ve Geliştirme Vizyonu

Sistemin daha da geliştirilmesi adına gelecekte şu çalışmaların eklenmesi planlanmaktadır:

- **Bilgisayarlı Görü (Computer Vision) Destekli Egzersiz Analizi:** Hastaların evde egzersiz yaparken telefon veya bilgisayar kamerası aracılığıyla eklemlerinin doğru açıda olup olmadığını tespit eden yapay zeka tabanlı bir modül entegrasyonu.
- **Mobil Uygulama:** Hastaların kendilerine atanan egzersiz programlarını görebileceği, seans hatırlatmalarını bildirim (push notification) olarak alabileceği bir Flutter veya React Native mobil istemcisi.
- **Ödeme Geçidi Entegrasyonu:** Faturaların doğrudan sistem üzerinden kredi kartıyla (iyzico, Stripe vb.) ödenebilmesini sağlayan sanal POS entegrasyonu.

5. KAYNAKCA

1. **Microsoft Learn .NET Belgeleri:** ASP.NET Core MVC ve Web API geliştirme standartları. (Erişim: learn.microsoft.com)
2. **Entity Framework Core Resmi Dokümantasyonu:** İlişkisel veritabanı haritalama ve veri yazma teknikleri. (Erişim: learn.microsoft.com/ef/core)
3. **Dapper Tutorial & GitHub Sayfası:** Mikro-ORM kullanımı, hızlı veri okuma ve mapping süreçleri. (Erişim: dapper-tutorial.net)
4. **QuestPDF Resmi Dokümantasyonu:** C# ile PDF tasarımı ve akış yerleşimi kuralları. (Erişim: questpdf.com)
5. **Google AI Edge Developer Guide:** Gemini API entegrasyonu ve REST istek protokolleri. (Erişim: ai.google.dev)
6. **ClosedXML Wiki:** C# ve OpenXML tabanlı Excel rapor üretimi. (Erişim: github.com/ClosedXML/ClosedXML)
7. **Serilog Documentation:** ASP.NET Core üzerinde yapılandırılmış log yönetimi. (Erişim: serilog.net)