

RestAPI proje fikirleri



net core ile bir restapi örneği yapacağım, daha önce ürün, not yönetimi, iş yönetimi gibi denemeler yaptım bunların dışında bir fikir sub

Güzel 👍 Artık temel CRUD mantığını ürün/not/task gibi klasik örneklerle pekiştirmişsin. Şimdi seni hem teknik olarak biraz zorlayacak hem de *gerçek bir iş problemine benzeyen* birkaç özgün **.NET Core REST API** proje fikri önereyim. Her biri seni farklı alanlarda geliştirir:



1. Eğitim / Sertifika Yönetim Sistemi API'si

Senin eğitimci kimliğine de uygun.

- **Entity'ler:** Öğitmen, Öğrenci, Kurs, Katılım, Sertifika
- **Özellikler:**
 - Kurs kayıt ve iptal işlemleri
 - Kurs sonunda otomatik sertifika üretimi (örneğin PDF veya QR Code ile)
 - JWT + Role bazlı erişim (Öğitmen / Öğrenci rolleri)
 - Swagger ile token yönetimi



Bonus: Sertifikayı dosya olarak üretip bir "/certificates" endpoint'iyle indirt.



2. Randevu Yönetim Sistemi (Sağlık veya Kuaför için)

- **Entity'ler:** Kullanıcı, Hizmet, Randevu, ÇalışmaSaatleri
- **Özellikler:**
 - Çakışan randevuları kontrol et
 - JWT + rol bazlı (Admin, Personel, Müşteri)
 - Randevular için takvim mantığı (örneğin sadece 09:00–18:00 arası izin ver)
 - Email/SMS entegrasyonu (örneğin MailKit veya Twilio ile)



Bonus: Randevulara "durum" ekle (Onaylandı / Beklemede / İptal).



3. AI Destekli Öneri API'si

Bir örnek: "Makale Öneri API'si"

- **Entity'ler:** Kullanıcı, Makale, Okuma geçmişi, Öneri
- **Özellikler:**
 - Basit öneri algoritması (örneğin aynı kategoriye ait makaleleri önermek)

- Kullanıcı bazlı geçmiş kaydı
- Caching (MemoryCache veya Redis)
- Logging (Serilog + Elastic Stack)

→ *Bonus:* ChatGPT API gibi bir LLM entegre ederek “ilgine göre öner” özelliği ekle.

4. Kargo / Teslimat Takip API’si

- **Entity’ler:** Gönderi, KargoFirması, Kullanıcı, KonumLog
- **Özellikler:**
 - Kargonun konumunu güncelleyen endpoint
 - Durum geçişleri (Hazırlanıyor → Yolda → Teslim Edildi)
 - SignalR ile gerçek zamanlı konum takibi
 - Swagger UI + JWT auth

→ *Bonus:* Teslimat süresi istatistikleri için rapor endpoint’i.

5. Yorum ve Değerlendirme API’si

- **Entity’ler:** Kullanıcı, Ürün (veya Servis), Yorum, Puan
- **Özellikler:**
 - Ortalama puan hesaplama
 - JWT kimlik doğrulama
 - Sadece satın alan kullanıcı yorum yapabilsin
 - “Beğen/Beğenme” sistemi

→ *Bonus:* “Toksik yorum” analizi (örneğin HuggingFace API ile).

6. Masraf ve Fatura Takip API’si

- **Entity’ler:** Kullanıcı, Masraf, Kategori, FaturaDosyası
- **Özellikler:**
 - Dosya yükleme (IFormFile)
 - Tarih aralığına göre filtreleme
 - Toplam harcama raporu
 - Role-based auth (User, Admin, Accountant)

→ *Bonus:* Excel export endpoint (OpenXML kullanarak).

7. Güvenli Not Kasası API’si

- **Entity'ler:** Kullanıcı, Not, Etiket
- **Özellikler:**
 - Notlar AES ile şifreli saklanır
 - JWT + Refresh token sistemi
 - Rate limiting
 - Audit log (hangi kullanıcı hangi notu okudu vs.)

➔ **Bonus:** İki aşamalı doğrulama (2FA) eklersen, mükemmel bir "security demo API" olur.

İstersen bu projelerden birini seçelim, ben de sana:

- **Entity modellerini,**
- **DTO'ları,**
- **Controller yapısını,**
- **JWT + Swagger yapılandırmasını**
tam örnek olarak adım adım çıkarayım.

Hangisini istersin? (örneğin "Randevu Yönetim Sistemi" veya "Sertifika Yönetimi" gibi)



2\ ****Randevu Yönetim Sistemi (Sağlık veya Kuaför için)****
bunu seçiyorum, iş mantığını detaylandır

Harika seçim 🌟

"Randevu Yönetim Sistemi REST API" hem **gerçek hayatta işe yarar**, hem de .NET Core'da **middleware, validation, JWT, rol bazlı erişim, repository pattern, service katmanı, exception handling** gibi tüm temel kavramları göstermek için ideal bir projedir.

Aşağıda sana **tam iş mantığını** (business logic) detaylıca adım adım anlatıyorum:



1. Genel Amaç

Bu API, kullanıcıların (müşteriler) **randevu almasını**, çalışanların (personel) **randevuları yönetmesini**, yöneticilerin (admin) ise **sistemi kontrol etmesini** sağlar.

Kapsam:

- Sağlık merkezi, kuaför, diyetisyen, fizyoterapi, tamir servisi gibi ortamlarda kullanılabilir.
- JWT + Rol bazlı yetkilendirme içerir.



2. Rollerin Tanımı

Rol	Açıklama
Admin	Yeni kullanıcı ekler, tüm randevuları görüntüler/siler.
Personel	Kendi müşterilerinin randevularını yönetir.
Müşteri (User)	Randevu oluşturabilir, iptal edebilir, geçmiş randevularını görüntüleyebilir.

3. Ana Varlıklar (Entity'ler)

Kullanıcı (User)

- Id, Name, Email, PasswordHash, Role, CreatedAt
- JWT ile giriş yapar.
- Her kullanıcı bir rol taşır (User, Staff, Admin).

Hizmet (Service)

- Id, Name, Description, DurationMinutes, Price
- Örn: "Saç Kesimi", "Cilt Bakımı", "Diyet Konsültasyonu"
- Admin tarafından tanımlanır.

Randevu (Appointment)

- Id, UserId, StaffId, ServiceId, AppointmentDate, Status, CreatedAt
- Status = "Pending" | "Approved" | "Completed" | "Cancelled"
- Randevular arasında **çakışma kontrolü** yapılır:

```
csharp
if (db.Appointments.Any(a => a.StaffId == staffId &&
    a.AppointmentDate == requestedDate &&
    a.Status != "Cancelled"))
{
    throw new ConflictException("Bu saat dolu!");
}
```

Çalışma Saatleri (WorkingHours)

- Id, StaffId, DayOfWeek, StartTime, EndTime
- Örn: Pazartesi 09:00–17:00
- Randevu, sadece bu saat aralıklarında alınabilir.

Randevu Günlüğü (AppointmentLog)

- Randevuların durum deęişikliklerini kaydeder.
(AppointmentId, OldStatus, NewStatus, ChangedBy, ChangedAt)

⚙️ 4. İş Kuralları (Business Rules)

📅 Randevu Oluşturma

- Sadece **müşteri rolü** randevu oluşturabilir.
- Tarih **personelin çalışma saatine** denk gelmeli.
- Aynı tarih/saatte aynı personel başka randevu alamaz.
- Yeni randevu oluşturulduğunda **Status = "Pending"** olur.

✅ Randevu Onayı

- Sadece **personel** onaylayabilir.
- Onaylandığında **Status = "Approved"**
- Onaylanan randevuların aynı saat için çakışmaması garanti edilir.

❌ Randevu İptali

- **Müşteri** kendi randevusunu iptal edebilir.
- **Personel/Admin** de iptal edebilir (örneğin hastalık durumunda).
- **Status = "Cancelled"** olarak güncellenir.

📄 Randevu Geçmişi

- Her kullanıcı kendi geçmiş randevularını listeleyebilir.
- Admin tüm randevuları görebilir.
- Filtreleme örnekleri:
 - Tarihe göre (/api/appointments?from=2025-10-01&to=2025-10-31)
 - Statüye göre (/api/appointments?status=Approved)

🔔 Bildirim / Entegrasyon (Opsiyonel)

- Randevu oluşturulduğunda veya iptal edildiğinde e-posta bildirimi gönderilebilir (MailKit).
- İleri seviye olarak SignalR ile **gerçek zamanlı onay bildirimleri** yapılabilir.

🧱 5. Katmanlı Mimari Önerisi

pgsql

```

AppointmentSystem/
├── Controllers/
├── Services/
│   ├── AppointmentService.cs
│   ├── UserService.cs
│   └── EmailService.cs
├── Repositories/
│   ├── AppointmentRepository.cs
│   └── GenericRepository.cs
├── Models/
│   ├── Entities/
│   └── DTOs/
├── Middleware/
│   └── GlobalExceptionHandler.cs
├── Data/
│   └── AppDbContext.cs
├── Program.cs
└── appsettings.json

```

6. Güvenlik ve Yetkilendirme

- JWT tabanlı kimlik doğrulama (`Microsoft.AspNetCore.Authentication.JwtBearer`)
- `[Authorize(Roles = "User,Staff,Admin")]`
- Swagger UI üzerinde **Authorize** butonu ile token girme.
- **Role-based endpoint'ler:**

csharp

```

[Authorize(Roles = "Admin,Staff")]
[HttpPost("approve/{id}")]
public IActionResult ApproveAppointment(int id) { ... }

```

7. Örnek API Endpoint'leri

HTTP	Endpoint	Açıklama	Roller
POST	<code>/api/auth/register</code>	Yeni kullanıcı oluştur	Public
POST	<code>/api/auth/login</code>	Token al	Public
GET	<code>/api/services</code>	Hizmet listesi	Public
POST	<code>/api/appointments</code>	Randevu oluştur	User
PUT	<code>/api/appointments/{id}/approve</code>	Randevuyu onayla	Staff
PUT	<code>/api/appointments/{id}/cancel</code>	Randevuyu iptal et	User, Staff, Admin
GET	<code>/api/appointments/my</code>	Kullanıcının kendi randevuları	User
GET	<code>/api/appointments/all</code>	Tüm randevular (raporlama)	Admin

8. Ekstra (Opsiyonel) Özellikler

- **Caching:** Popüler hizmetleri `MemoryCache` ile tut.

- **Rate Limiting:** Kullanıcı başına dakika başına istek sınırı.
 - **Unit Testler:** xUnit + Moq ile servis testleri.
 - **Audit Trail:** Her işlem log tablosuna kaydedilsin.
-

İstersen bir sonraki adımda:

👉 **Entity** sınıfları + **DTO** yapısı + **AppDbContext** şemasını hazırlayayım
ve ardından Controller ile ilk CRUD işlemini (örneğin **AppointmentController**) birlikte yazalım.

Devam edelim mi bu sırayla?