Blog Yönetim Sistemi (BYS) — Sınav Dökümanı

1) Amaç ve Kapsam

Bu sınavda küçük ölçekli bir "Blog Yönetim Sistemi" geliştirmeniz beklenmektedir. Sistem, bir blog sitesinin temel içerik yönetimini sağlar.
Ön yüz (UI) yalnızca basit yönetim ekranlarıdır; değerlendirme odağı **Backend'in mimari kalitesi** ve **katmanlı tasarım** ve Front End Programming (designing degil) üzerinedir.

Beklenen Çıktılar

- .NET (API) tarafında **Onion Architecture** veya **Hexagonal Architecture (Ports & Adapters)** yaklaşımıyla katmanlı bir yapı
- Angular ile **yönetim paneli** tarzında basit CRUD ekranları
- Temiz proje yapısı, doğru sorumluluk ayrımı,validasyon ve hata yönetimi(Unit Test istenmemektedir)

---

2) İş Kuralları ve Modüller

2.1 İçerik Yönetimi

Sistem; içerik üretimi ve yönetimi için aşağıdaki kavramları destekler:

A) Yazılar (Posts)

- Sistem üzerinde yayımlanan temel içerik birimidir.
- Bir yazı:
    - Başlık (Title)
    - İçerik (Content)
    - Özet (Excerpt) (UI'da opsiyonel; Backend iş kuralı ile üretilebilir veya kullanıcı girebilir)
    - Yayın durumu (Draft / Published)
    - Oluşturulma tarihi ve güncellenme tarihi
    - Bir veya birden fazla kategori ile ilişkilendirilebilir
    - Bir veya birden fazla etiket ile ilişkilendirilebilir

İş Kuralları

- Title boş olamaz ve belirli bir uzunluk sınırı içinde olmalıdır.
- Content boş olamaz.
- Published durumda olan yazının Title + Content validasyonları zorunlu olarak sağlanmalıdır.
- Draft olarak kaydedilen yazı daha sonra Published yapılabilmelidir.

B) Kategoriler (Categories)

- Yazıları sınıflandırmak için kullanılır.
- Kategori:
    - Ad (Name)
    - Açıklama (Description) (opsiyonel)
- Kategori adı benzersiz olmalıdır (case-insensitive).

C) Etiketler (Tags)

- Yazıları daha esnek şekilde sınıflandırmak için kullanılır.
- Etiket:
    - Ad (Name)
- Etiket adı benzersiz olmalıdır (case-insensitive).

D) Yorumlar (Comments) — Opsiyonel Modül

Bu modül yapılırsa artı puan getirir, yapılmazsa puan kırmaz.

- Bir yazıya kullanıcı yorumu eklenebilir.
- Yorum:
    - Yazar adı (AuthorName)
    - İçerik (Text)
    - Oluşturulma tarihi
    - Onay durumu (Approved / Pending) (isteğe bağlı)
      İş Kuralları
- Yorum metni boş olamaz.
- Yönetim panelinden yorumlar listelenebilmelidir (silme/onaylama opsiyonel).

Not: Opsiyonel katman açacaksanız mimariyi bozmadan eklemeniz beklenir.(Misal Onion'da bagımlılık her zaman içe dogru olmalıdır)

---

3) API Gereksinimleri

3.1 HTTP Endpoints

Aşağıdaki operasyonlar sağlanmalıdır (temel CRUD):

- Posts:
    - Listeleme (filtreleme opsiyonel)
    - Detay (id ile)
    - Ekleme
    - Güncelleme
    - Silme (Hard delete veya Soft delete tercih edilebilir; seçiminizi dokümante edin)
- Categories:
    - Listeleme
    - Ekleme
    - Güncelleme
    - Silme
      Silme işleminde ilişkili postlar varsa davranışınızı belirleyin:
    - ya engelle (validation/hata)
    - ya ilişkiyi kaldır
    - ya soft delete
- Tags:
    - Listeleme
    - Ekleme
    - Silme (opsiyonel güncelleme)

Yanıt Standardı

- Başarılı yanıtlar anlamlı HTTP status code ile dönmeli.
- Validation hataları: 400 (BadRequest) ve hata detayları.
- Bulunamadı: 404
- Sunucu hatası: 500 + log (UI'a ham exception yansıtılmamalı)

3.2 DTO ve Domain Ayrımı

- API'dan içeri giren/giden modeller (DTO/ViewModel) ayrımını yapmanız beklenmektedir. Yapılar aynı gözükse bile sorumluluk olarak farklı oldukları için buna dikkat edilmelidir
- Domain katmanında gereksiz UI alanları bulunmamalıdır.
- Mapping (manual veya AutoMapper) yapılabilir.

---

4) Mimari Gereksinimler (Değerlendirme Odak Noktası)

4.1 Onion / Hexagonal Beklentisi

  - Entities, Value Objects, Domain rules
  - Domain, Infrastructure bağımlılığı almaz(Bu isimler sadece standartlara göre belirlenmektedir. Görev yapısı dogru oldugu sürece farklı isimlendirmeleri lütfen belirtin)

Temel Kural

- Dependency yönü içe doğru olmalı (UI/Infra → Application → Domain).
- "Business logic" Controller içine dağılmamalı.

4.2 Dependency Injection (DI)

- DI Container'da bağımlılıklar doğru şekilde register edilmelidir.
- Repository implementasyonları Infrastructure'da kalmalı.
- Application services test edilebilir şekilde tasarlanmalıdır.

4.3 Validation

- En az iki seviyede validation beklenir:
  1. API sınırında (DTO validation)

2. Domain/Application seviyesinde (iş kuralı validation)

Örn:

- “Category name unique” kontrolü yalnızca Angular’da değil, Backend’de kesin olmalı.
- “Publish edilen post content boş olamaz” gibi kurallar Backend’de garanti edilmeli.

---

5) UI Gereksinimleri (Angular)

UI değerlendirmesi temel düzeyde yapılır: amaç Backend’i doğru tüketmeniz.

5.1 Ekranlar

- Posts yönetimi:
  - Liste
  - Ekleme/Güncelleme formu
- Categories yönetimi:
  - Liste
  - Ekleme/Güncelleme
- Tags yönetimi:
  - Liste
  - Ekleme/Silme

5.2 UI Kuralları

- Reactive Forms veya Signals yaklaşımı kullanılabilir.
- En azından form alanlarında temel UI validation olmalı (required vb.)

---

6) Non-Functional Beklentiler

6.1 Kod Kalitesi

- Katmanlar arası geçişte “leak” olmamalı (EF Core entity’lerini Controller’dan döndürmek gibi).

- SRP, OCP, DIP gibi SOLID prensiplerini bozmayın.
- Gereksiz “God Service” oluşmasın; use case’ler net olsun.

---

7) Teslim Formatı

- Tek bir solution altında düzenli bir klasörleme
- README.md içinde:
    - Mimari seçiminiz (Onion mı Hexagonal mı)
    - Katmanların sorumlulukları
    - Kurduğunuz iş kuralları (kısa)
    - Çalıştırma adımları (Angular + API)