MongoDB

Ürün Kataloğu Uygulaması

Projesi

Amaç: Bu proje, C# programlama dilini kullanarak MongoDB veritabanı ile etkileşim kurma becerilerini geliştirmeyi amaçlamaktadır.Ürün bilgilerini MongoDB'de saklayabilecek, bu bilgilere erişebilecek, yeni ürünler ekleyebilecek, mevcut ürünleri düzenleyebilecek ve silebileceklerdir. Ayrıca, çeşitli filtreleme ve sıralama seçenekleriyle ürünleri listeleyebilecek ve arama yapabileceklerdir. Uygulama, olası hataları bir dosyaya kaydederek basit bir hata yönetimi mekanizması da sunacaktır.

Teknolojiler:

- C# Programlama Dili
- .NET Core veya .NET Framework
- MongoDB

Proje Detayları:

1. MongoDB Bağlantısı:

- Uygulama, MongoDB veritabanına başarılı bir şekilde bağlanabilmelidir.
- Bağlantı nesnesi (örneğin, `MongoClient` ve `IMongoDatabase`) uygulama boyunca güvenli ve verimli bir şekilde yönetilmelidir.

2. Modeller (Models):

- Aşağıdaki gibi temel modeller oluşturulmalıdır:
  - **`Urun` Modeli:**
    - `Id` (MongoDB tarafından otomatik oluşturulan `ObjectId`)
    - `Ad` (string, zorunlu)
    - `Aciklama` (string)
    - `Fiyat` (decimal, zorunlu, pozitif olmalı)
    - `StokAdedi` (int, zorunlu, sıfır veya pozitif olmalı)
    - `KategoriId` (`ObjectId`, `Kategori` modeline referans)
    - `EklenmeTarihi` (DateTime, otomatik olarak oluşturulmalı)
  - **`Kategori` Modeli:**
    - `Id` (MongoDB tarafından otomatik oluşturulan `ObjectId`)
    - `Ad` (string, zorunlu, benzersiz olmalı)

3. Servis Katmanı (Services):

- Ürünler ve kategoriler için ayrı servis sınıfları (`UrunService`, `KategoriService`) oluşturulmalıdır. Bu servis sınıfları aşağıdaki metotları içermelidir:

- **`UrunService` Metotları:**
  - `Ekle(Urun urun)`: Yeni bir ürünü veritabanına ekler.
  - `Guncelle(Urun urun)`: Mevcut bir ürünü günceller.
  - `Sil(string id)`: Belirtilen ID'ye sahip ürünü siler.
  - `Getir(string id)`: Belirtilen ID'ye sahip ürünü getirir.
  - `Listele(int sayfa = 1, int limit = 10)`: Tüm ürünleri sayfalandırarak listeler (varsayılan sayfa 1, sayfa başına 10 ürün). `Find`, `Limit`, `Skip` kullanılmalıdır. Son eklenenler ilk sırada gelmelidir (`SortByDescending`).
  - `Ara(string anahtarKelime)`: Ürün adında veya açıklamasında belirtilen anahtar kelimeyi içeren ürünleri listeler.
  - `Filtrele(decimal? minFiyat = null, decimal? maxFiyat = null, string kategoriId = null)`: Fiyat aralığına ve/veya kategoriye göre ürünleri filtreler (`FilterDefinitionBuilder` kullanılmalıdır).
- **`KategoriService` Metotları:**
  - `Ekle(Kategori kategori)`: Yeni bir kategori ekler.
  - `Guncelle(Kategori kategori)`: Mevcut bir kategoriyi günceller.
  - `Sil(string id)`: Belirtilen ID'ye sahip kategoriyi siler (silinmeye çalışılan kategoriye bağlı ürün varsa hata mesajı verilmelidir).
  - `Getir(string id)`: Belirtilen ID'ye sahip kategoriyi getirir.
  - `Listele()`: Tüm kategorileri listeler.

4. Hata Yönetimi:

- Uygulama içerisinde oluşabilecek hatalar (örneğin, veritabanı bağlantı hatası, geçersiz veri girişi, bulunamayan kayıt vb.) `try-catch` blokları ile yakalanmalıdır.
- Yakalanan hataların detayları (hata mesajı, zaman damgası, oluşan metot vb.) bir metin dosyasına (örneğin, `hata_log.txt`) yazılmalıdır.
- Log dosyasını okuyabilen ve belirli bir hata mesajını veya zaman aralığını arayabilen basit bir mekanizma geliştirmelidirler.

5. Kullanıcı Arayüzü (Konsol Uygulaması):

- Bu arayüz üzerinden aşağıdaki işlemler yapılabilmelidir:
  - Ürün Ekleme
  - Ürün Güncelleme (ID ile seçim yaparak)
  - Ürün Silme (ID ile seçim yaparak)
  - Ürün Listeleme (sayfalama ile)
  - Ürün Arama (anahtar kelimeye göre)
  - Ürün Filtreleme (fiyat aralığı ve/veya kategoriye göre)
  - Kategori Ekleme
  - Kategori Güncelleme (ID ile seçim yaparak)
  - Kategori Silme (ID ile seçim yaparak)
  - Kategori Listeleme
  - Hata Loglarını Görüntüleme
  - Hata Loglarında Arama (mesaj veya zaman aralığına göre)

Ek Gereksinimler ve İpuçları:

- **Veri Validasyonu:** Kullanıcıdan alınan verilerin (örneğin, fiyatın pozitif olması, zorunlu alanların doldurulması) servis katmanında kontrol edilmesi önemlidir. Geçersiz veri durumunda anlamlı hata mesajları verilmelidir.
- **ID Yönetimi:** MongoDB'nin `ObjectId` tipinin nasıl kullanıldığına dikkat edilmelidir.
- **Kod Kalitesi:** Kodlarını düzenli, okunabilir ve anlamlı değişken/metot adlarıyla yazmaları gerekmektedir.
- **Yorum Satırları:** Kodun önemli kısımlarına açıklayıcı yorum satırları eklemeleri istenmelidir.
- **Proje bitimi:** Proje tamamlandıktan sonra Github hesabın bir repo olarak anlaşılır ve açıklayıcı bir şekilde eklenmelidir.

Değerlendirme Kriterleri:

- MongoDB bağlantısının doğru bir şekilde kurulması.
- Model sınıflarının doğru bir şekilde tasarlanması.
- Servis katmanının işlevselliği ve doğru implementasyonu.
- Ekleme, düzenleme, silme ve listeleme işlemlerinin eksiksiz çalışması.
- Arama ve filtreleme özelliklerinin doğru implementasyonu.
- `Find`, `Limit`, `SortByDescending`, `Skip` kullanımının doğruluğu.
- Hata yönetimi mekanizmasının (loglama ve log okuma/arama) çalışması.
- Kodun genel kalitesi, okunabilirliği ve düzeni.
- Projenin belirtilen gereksinimleri ne kadar karşıladığı.