Kullanıcı Yönetimli Envanter Yönetim Sistemi - Proje Dokümantasyonu

1. Giriş

Bu proje, kullanıcı yönetimi ve envanter kontrolünü içeren bir RESTful API sistemidir. Proje, C#, .NET Core, Entity Framework, MSSQL, MongoDB ve Redis kullanılarak geliştirilmiştir. API, kimlik doğrulama, rol tabanlı yetkilendirme ve ürün verilerinin önbelleğe alınmasını sağlar.

Brute force saldırılarını önlemek için giriş uç noktasına 1 dakikada 5 adet istek atılabilecek şekilde ayarlanmıştır.

2. Kurulum Talimatları

Gerekli Araçlar:

- .NET SDK (6.0 veya daha yeni)
- MSSQL
- MongoDB
- Redis(Memurai)
- Visual Studio 2022

Kurulum Adımları:

A. Gerekli Yazılımların Kurulumu:

1. **Projeyi GitHub'dan indirin**:
   https://github.com/mehmetakifyilmaz53/PROFEN-WEB-API
2. **MSSQL LocalDB:**
   Visual Studio ile birlikte gelir. Alternatif olarak, SQL Server Express yükleyebilirsiniz.
   Sıfırdan veritabanı oluşturmak istemiyorsanız, kullanmış olduğum veri tabanı scriptini proje dosyası içine ekliyorum. **Bu scripti kullanırsanız admin hesap eklenir. Username: testadmin şifre:123 olarak kullanabilirsiniz.**

3. **MongoDB:**
   İndirin ve yükleyin: MongoDB Resmi Sitesi
   Compass versiyonu kullanılmıştır.
4. **Redis:**
   Redis önbellemesi için Memurai kullandım indirme linki: Memurai Redis Alternatifi veya Docker üzerinden Redis çalıştırabilirsiniz.

B. Veritabanı Yapılandırması:

1. **LocalDB Ayarları:**
   Local veri tabanı oluşturmak için Visual studio terminaline bağlanarak Migration oluşturarak veritabanını güncelleyin.

   Migration Oluşturma:
   `dotnet ef migrations add InitialCreate`

   Veritabanını Güncelleme:
   `dotnet ef database update`

2. **MongoDB Yapılandırması:**
   `appsettings.json` dosyasındaki kod sayesinde bir ürün kaydı atıldığında table otomatik oluşacaktır.

```
"MongoDB": {
  "ConnectionString": "mongodb://localhost:27017",
  "DatabaseName": "ProWebApiDb"
}
```

3. **Redis Yapılandırması:**
   - `appsettings.json` dosyasını Redis bağlantısını local olması için kontrol edin:

```
"Redis": {
  "ConnectionString": "localhost:6379"
}
```

3. Mimari Genel Bakış

- **Katmanlar:**
  - WebAPI (Controller katmanı)
  - Servis Katmanı (Business Logic)
  - Veri Katmanı (Repository & Entity Framework)
  - Veritabanı Yönetimi (PostgreSQL & MongoDB)
- **Tasarım Desenleri:**
  - Dependency Injection

- Repository Pattern

4. API Referansı

Kullanıcı Yönetimi

- **POST** `/api/users/register` - Yeni kullanıcı kaydı.

Request Body:

```
{
  "UserName": "admin",
  "PasswordHash": "P@ssw0rd123",
  "Email": "admin@example.com",
  "Role": 0
}
```

Response:

```
{
  "Success": true,
  "Data": {
    "Id": 1,
    "UserName": "admin",
    "Email": "admin@example.com",
    "Role": 0
  },
  "Message": "Kullanıcı başarıyla kaydedildi."
}
```

- **POST** `/api/users/login` - Kullanıcı girişi ve JWT alma.

Request Body:

```
{
  "UserName": "admin",
  "Password": "P@ssw0rd123"
}
```

Response:

```
{
  "Success": true,
  "Data": {
    "JWTToken": 1
  },
  "Message": "Giriş başarılı."
}
```

- **GET** `/api/users/{id}` - Kullanıcı bilgilerini getirme (Admin).

Response:

```
{
  "Success": true,
  "Data": {
    "Id": 1,
    "UserName": "admin",
    "Email": "admin@example.com",
    "Role": 0
  },
  "Message": "Kullanıcı başarıyla getirildi."
}
```

- **GET** `/api/users` - Tüm kullanıcıları listeleme (Admin).

Response:


```
{
  "Success": true,
  "Data": [
    {
      "Id": 1,
      "UserName": "admin",
      "Email": "admin@example.com",
      "Role": 0
    },
    {
      "Id": 2,
      "UserName": "manager",
      "Email": "manager@example.com",
      "Role": 1
    }
  ],
  "Message": "Kullanıcılar başarıyla listelendi."
}
```


- **PUT** `/api/users/{id}` - Kullanıcı bilgilerini güncelleme (Admin).

Request Body:


```
{
  "UserName": "updatedUser",
  "PasswordHash": "NewPassword123",
  "Email": "updated@example.com",
  "Role": "Manager"
}
```


Response:


```
{
  "Success": true,
  "Message": "Kullanıcı başarıyla güncellendi.",
  "Data": true
}
```


- **DELETE** `/api/users/{id}` - Kullanıcıyı silme (Admin).

Response:


```
{
  "Success": true,
  "Message": "Kullanıcı başarıyla silindi.",
  "Data": true
}
```

Ürün Yönetimi

- **POST** `/api/products` - Yeni ürün ekleme (Admin, Manager).

- Request Body:

```
{
  "ProductName": "Laptop",
  "Description": "16GB RAM, 512GB SSD",
  "Price": 1200.99,
  "Quantity": 10,
  "Category": "Elektronik"
}
```

- Response:

```
{
  "Success": true,
  "Data": {
    "Id": "609b67f5",
    "Name": "Laptop",
    "Description": "16GB RAM, 512GB SSD",
    "Price": 1200.99,
    "Quantity": 10,
    "Category": "Elektronik"
  },
  "Message": "Ürün başarıyla kaydedildi."
}
```

- **PUT** `/api/products/{id}` - Ürünü güncelleme (Admin, Manager).

PUT `/api/products/{id}`

- Request Body:

```
{
  "ProductName": "Tablet",
  "Description": "10 inç ekran, 128GB depolama",
  "Price": 500.75,
  "Quantity": 15,
  "Category": "Elektronik"
}
```

- Response:

```
{
  "Success": true,
  "Message": "Ürün güncellendi.",
  "Data": true
}
```

- **DELETE** `/api/products/{id}` - Ürünü silme (Admin, Manager).

Response:

```
{
  "Success": true,
  "Message": "Ürün silindi.",
  "Data": true
}
```

- **GET** `/api/products/{id}` - Ürün detaylarını getirme.

Response:

```
{
  "Success": true,
  "Data": {
    "Id": "609b67f5",
    "Name": "Laptop",
    "Description": "16GB RAM, 512GB SSD",
    "Price": 1200.99,
    "Quantity": 10,
    "Category": "Elektronik"
  },
  "Message": "Ürün bulundu."
}
```

- **GET** `/api/products` - Tüm ürünleri listeleme.

Response:

```
{
  "Success": true,
  "Data": [
    {
      "Id": "609b67f5",
      "Name": "Laptop",
      "Description": "16GB RAM, 512GB SSD",
      "Price": 1200.99,
      "Quantity": 10,
      "Category": "Elektronik"
    },
    {
      "Id": "609b68a7",
      "Name": "Tablet",
      "Description": "10 inç ekran, 128GB depolama",
      "Price": 500.75,
      "Quantity": 15,
      "Category": "Elektronik"
    }
  ],
  "Message": "Ürünler başarıyla listelendi."
}
```

Kategori Yönetimi

- **POST** `/api/categories` - Yeni kategori ekleme (Admin).

Request Body:

```
{
  "categoryName": "Elektronik"
}
```

Response:

```
{
  "data": {
    "id": 3,
    "category_name": "Elektronik"
  },
  "success": true,
  "message": "Kategori kaydedildi."
}
```

- **GET** `/api/categories` - Tüm kategorileri listeleme.

Response:


```
{
  "data": [
    {
      "id": 1,
      "category_name": "kahve"
    },
    {
      "id": 2,
      "category_name": "Elektronik"
    }
  ],
  "success": true,
  "message": "Kategoriler getirildi"
}
```


5. Veritabanı Yapılandırması

MSSQL(Entity Framework):

- **Tablolar:** Kullanıcılar, Kategoriler
- **Veritabanı Bağlantısı:** `appsettings.json`

MongoDB:

- **Koleksiyonlar:** Ürünler

6. Kimlik Doğrulama ve Yetkilendirme

- **JWT Doğrulama:** API'ye erişim için kimlik doğrulama zorunludur.
- **Rol Tabanlı Yetkilendirme:** Admin, Manager, Viewer rollerine özel yetkiler sağlanmıştır.

7. Önbellekleme ve Performans

- Redis, ürün verilerini sık erişim için önbelleğe alır ve 10 dakika boyunca bellekte tutar.
- Ürün ekleme ve güncelleme işlemleri önbelleği otomatik olarak temizler.

8. Güvenlik Özellikleri

- **Parola Hashleme:** Parolalar BCrypt ile hashlenir.
- **API Güvenliği:** HTTPS ve JWT ile güvenli API bağlantısı.

9. Test Senaryoları

- **Kullanıcı Testleri:** Kayıt, giriş ve yetkilendirme testleri.
- **Ürün Testleri:** CRUD işlemleri

10. Loglama ve Hata Yönetimi

- Önemli API'ler için veritabanına Log kayıtları atılır.
- Hatalar uygun HTTP durum kodları ile döndürülür.

11. Varsayımlar ve Kararlar

- **Varsayımlar:** Ürün fiyatları pozitif ve benzersiz kategori adları olacaktır.
- **Teknik Kararlar:** Entity Framework ve MongoDB'nin kombinasyonu, esneklik ve performans sağlamak için seçilmiştir.

12. API HTTP Kodları

API Hata Kodları Açıklamaları:

- **200 OK:** İstek başarıyla tamamlandı.
- **400 Bad Request:** Geçersiz veri gönderildi.
- **401 Unauthorized:** Kimlik doğrulama başarısız.
- **403 Forbidden:** İşlem için yeterli yetkiye sahip değil.
- **404 Not Found:** Kayıt bulunamadı.
- **429 Too Many Requests:** Çok fazla istek yapıldı.
- **500 Internal Server Error:** Sunucu tarafında bir hata oluştu.
- **503 Service Unavailable:** Hizmet geçici olarak kullanılamıyor.

Önemli Not Admin hesap oluşturmak için veri tabanında bu kodu çalıştırabilirsiniz.

USE [ProfenDB] GO

INSERT INTO [dbo].[Users] ([user_Name] ,[password_Hash] ,[email] ,[role])

VALUES
('testadmin','$2a$11$QYIZnITDwwyf7BDGpW2NcumcG0Kwanvp33.2krchY0FyqUvzWJ54m','stri1223ng@gmail.com',0)

GO

bu kodu çalıştırdığınızda kullanıcı adı 'testadmin' ve şifresi 123 olan bir admin hesap ekleyecektir.

Hazırlayan: Mehmet Akif Yılmaz

Tarih:15.12.2024