2025

Ai Finance Projesi

STAJ BAŞVURUSU TEST ÇALIŞMASI

E-MARKETİNG MİKROSERVİS ÖDEVİ – KULLANICI VE ÜRÜN YÖNETİMİ

İçindekiler

Görev Açıklaması:

Bu projede, kullanıcı yönetimi, authentication/authorization, ürün yönetimi, alışveriş sepeti işlemleri ve satın alma süreci gibi temel modülleri iki ayrı mikroservise dağıtarak, gerçek hayat senaryolarını uygulamalı olarak geliştirme hedeflenmektedir.

- **user.service**: Kullanıcı kimlik doğrulama ve yetkilendirme işlemlerini yönetecek.
- **product.service**: Ürün listesi, satın alma, sepet işlemleri gibi e-ticaret fonksiyonlarını sağlayacak.

User Service:

Yapması Gereklenler:

1. Kullanıcı yönetimi
   - Kullanıcı ekleme, pasife alma, şifre değiştirme
2. Kullanıcı adres bilgileri
   - Adres ekleme,değiştirme,silme,listeleme(CRUD)
   - Adres Ev,İş olarak kategorize etme
3. Kullanıcı iletişim bilgilieri
   - İletişim Ekleme,Değiştirme,Silme (CRUD)
   - İletişim Ev,İş olarak kategorize etme
4. Yönetici kullanıcı yönetim paneli
   - Kullanıcı pasife alma, röl değiştirme,
   - Şifre Sıfırlama.
5. Authentication
   - Login işlemleri
   - Token üretimi, Token ve Süre kontrolü
6. Authorization
   - Yetkilerin database'den çekilmesi memory'de yada cache'de tutulması. (Cache'de tutulması isteğe bağlıdır zorunlu değildir.)
   - Kullanıcı bilgilerinin istek sırasında katmanlar arası taşınması.

Endpointler

Authentication Endpoint(Path: / auth)

Method	Path	Açıklama	Role	Permission
*******	*******	Giriş yap	-	*******
*******	*******	Oturum kapat	User,Admin	*******
*******	/checkLogin	Token geçerli mi kontrolü	User,Admin	*******

Authorization Endpoint(Path: / authz)

Method	Path	Açıklama	Role	Permission
*******	*******	Kullanıcının tüm izinlerini getir	User,Admin	*******
*******	*******	Kullanıcı bu role sahip mi	User,Admin	*******

*******	*******	Kullanıcının belirli permission var mı? (ops.)	User,Admin	*******

User Endpoint(Path: / user)

Method	Path	Açıklama	Role	Permission
*******	*******	Tüm kullanıcıları getir	Admin	*******
*******	*******	Kullanıcı detayları	Admin	*******
*******	*******	Yeni kullanıcı oluştur	Admin	*******
*******	*******	Kullanıcı güncelle	Admin	*******
*******	*******	Kullanıcı sil (soft delete)	Admin	*******
*******	*******	Şifre değiştir (eski→yeni)	User	*******
*******	*******	Şifre sıfırla	Admin	*******
*******	*******	Giriş yapan kullanıcı bilgisi	User,Admin	*******
*******	*******	Başka kullanıcıyı pasifleştir	Admin	*******
*******	*******	Kendi hesabını pasifleştir	User	*******

Adres ve İletişim Endpoint(Path: /address, /contact)

Method	Path	Açıklama	Role	Permission
*******	*******	Bilgileri listele	User	*******
*******	*******	Yeni adres ekle	User	*******
*******	*******	Adres güncelle	User	*******
*******	*******	Adres sil	User	*******

Açıklamalar:

Dil: Go veya Python olacaktır.
Db: PostgreSql olmalıdır.
Teknojiler:

- Pyhton: Flask, Fastapi
- Go: Tercihe bağlıdır.

Authentication: Ekrana ilk giriş yapıldığında sistem checkLogin ile kullanıcı login olmuş mu expire mi kontrol edecektir. Eğer expire olmuşsa tekrar login işlemi isteyecektir. Login işleminden sonra dönen token her requestte Authorization headerina yazılacaktır. Yazım şekli standartlara uygun olmalıdır. Gelen her istekte bu token kontrol edilmeilidir.

Authorization: Token ie kullanıcı ilişkilendirilmeli. Kullanıcı bilgileri ve rolleri istek boyunca katmanlar arası ulaşılabilmelidir. Rol tipleri user ve admin olacaktır. Dilerseniz rollere permissionlar bağlayıp yetkileri permission üzerinden kontrol edebilirsiniz. Böyle bir durumda permissionlarda istek boyunca ulaşabilir olması gerekmektedir.

Şifre Değiştirme: Kullanıcı şifresini kendi değiştirebilecektir. Eğer unuttuysa yönetici şifreyi sıfırlayacak ve ilk login olduğunda sıfırlama şifresi ile giriş yapsa dahi kullanıcıdan yeni bir şifre tanımlanması istenecektir.

Repository katmanı: Orm tabanlı olmalıdır.

Adres Bigileri: İş veya Ev olarak sınıflandırılabilecektir.

İletişim Bigileri: İş veya Ev olarak sınıflandırılabilecektir.

Kullanıcı,Adres ve İletişim Ekranları: Bu ekranların tasarımı size ait olacaktır. Bütün panellerde isteğe bağlı ek bilgiler eklenebilir. Ön yüzde herhangi bir teknoloji kullanabilirsiniz. Kullanıcı adres ve iletişim bilgilerini kaydetme ayrı ayrı veya tek bir seferde DTO objesi ile yapılabilir. Bu tamamen sizin tasarımınıza bağlıdır. Kullanıcı login olduktan sonra kullanıcının adı soyadını ve istenilen diğer bilgiler endpoint üzerinden alınacaktır. Bu kullanıcı bilgi paneli ekranın sağ üst köşesinde olacaktır. Ek bilgiler eklenebilir.

Hata yönetimi: Herhangi bir istekte token expire olması durumunda kullanıcı otomatik olarak login ekranına yönlendirilecektir. Diğer hatalar notification olarak fırlatılacaktır.

Validasyon işlemleri: Ekranlarda girilen değerler bir validasyondan geçecektir. İstenirse bu validasyon backend'te yapılabilir. Yada ikisinde birden hem ui'da hem backend'te olabilir. Bu sizin isteğinize bağlıdır.

Product Service:

Yapması gerekenler:

1. **Ürün Yönetimi**
   - Ürün ekleme, güncelleme, silme, listeleme
   - Ürün kategorisi desteği (örn: elektronik, giyim, vs.)
   - Ürün stok ve fiyat bilgileri
2. **Alışveriş Sepeti İşlemleri**
   - Sepete ürün ekleme / çıkarma
   - Sepeti görüntüleme
   - Sepeti temizleme
3. **Satın Alma (Order)**
   - Sepetteki ürünleri satın alma işlemi
   - Sipariş geçmişini görüntüleme
   - Stoktan düşme işlemi
4. **Yönetim Paneli (Admin için)**
   - Ürünleri toplu güncelleme
   - Satış raporları (opsiyonel)
   - Ürün görünürlük değiştirme (aktif/pasif)
5. **Authentication ve Authorization (user.service üzerinden)**
   - Gelen JWT token kontrol edilecek
   - Kullanıcı kimliği token → user_id, role, permissions
   - Yetki kontrolü bir endpoint kontrolü veya liste dönen bir endpoint üzerinden yapılacak
   - (Opsiyonel) Yetkiler Redis cache üzerinden saklanabilir

Açıklamalar:

Dil: NET Core 6+ veya Go / Python (proje tercihlerine göre)
Db: PostgreSql olmalıdır.
Cache: Redis (İsteğe bağlı)
Teknojiler:

- Pyhton: Flask, Fastapi
- Go: Tercihe bağlıdır.

Authorization: Token authorization header'indan çekilir. Session bilgileri user.service üzerinden checkLogin ile kontrol edilir.

Endpointler

Product Endpoint(Path: /product)

Method	Path	Açıklama	Role	Permission
****	******	Tüm ürünleri listele	User,Admin	*******
****	*****	Ürün detay	User,Admin	*******
****	*****	Yeni ürün ekle	Admin	*******
****	****	Toplu ürün ekle	Admin	*******
****	*******	Ürün güncelle	Admin	*******
****	****	Ürün sil	Admin	*******

Card Endpoint(Path: /card)

Method	Path	Açıklama	Role	Permission
****	*****	Kullanıcının sepetini getir	User	*******
****	****	Sepetteki ürün detayını getir	User	*******
****	****	Sepete ürün ekle	User	*******
****	*****	Sepette ürün adet güncelle	User	*******
***	****	Sepetten ürün çıkar	User	*******
****	****	Sepeti temizle	User	*******

Order Endpoint(Path: /order)

Method	Path	Description	Role	Permission
*****	****	Sepeti satın al	User	*******
****	****	Sipariş geçmişi	User	*******

Repository Katmanı (ORM Tabanlı)

- Entity: Product, Cart, Order
- ORM: Entity Framework Core (C#), GORM (Go), SQLAlchemy (Python)
- Repository Interface ve Concrete klasör yapısı önerilir.

Kullanıcı Bilgileri

- Her request'te user_id token'dan çıkarılıp, siparişler veya sepet işlemlerine bağlanacak.
- Admin role'ü olan kullanıcılar ürün işlemlerini gerçekleştirebilir.

Test Senaryoları

Authentication & Authorization Test Senaryoları

Senaryo No	Açıklama
A1	Geçerli kullanıcı bilgileriyle login → 200 + JWT token döner
A2	Geçersiz şifre ile login → 401 Unauthorized
A3	Login olmadan korumalı bir endpoint'e erişim → 401 Unauthorized
A4	Token süresi dolmuş kullanıcı → /CheckLogin → 401
A5	Logout sonrası token geçersiz olmalı → 401 dönmeli
A6	JWT token ile kullanıcı bilgilerini çeken servis testi → Kullanıcı bilgileri dönmeli
A7	Kullanıcı yetki ve röl kontrolleri → Yetki veya röl varsa 200

Kullanıcı Test Senaryoları

Senaryo No	Açıklama
U1	Admin kullanıcı yeni kullanıcı oluşturur → 201 Created
U2	Aynı kullanıcı adıyla tekrar kayıt → 409 Conflict
U3	Kullanıcı kendi şifresini doğru eski şifre ile değiştirir → 200 OK
U4	Kullanıcı şifre değiştirirken eski şifre yanlışsa → 400 Bad Request
U5	Admin başka bir kullanıcının şifresini sıfırlar → 200 OK
U6	Kullanıcı durumu güncellenir → 200 OK
U7	Kullanıcı kendi kullanıcısını iptal eder. → 200 OK
U8	Kullanıcı bilgisi doğru id ile çekilir→ 200 OK
U9	Kullanıcı olmayan bir ID ile sorgulanır → 404 Not Found

Adres ve İletişim Bilgileri Senaryoları

Senaryo No	Açıklama
C1	Kullanıcı yeni bir ev ve iş adresi ekler → 201
C2	Kullanıcı adresini günceller → 200
C3	Geçersiz adres ID ile silme → 404
C4	Kullanıcı iş ve ev telefonunu ekler → 201
C5	Kullanıcı iletişim bilgisini güncellerken format hatası varsa → 400

Role & Permission Test Senaryoları

Senaryo No	Açıklama
R1	Admin yeni rol tanımlar → 201
R2	Admin rol detaylarını günceller → 200
R3	Kullanıcı rolü admin tarafından değiştirilir → 200
R4	Kullanıcı olmayan bir role atanırsa → 404
R5	Admin role permission atar → 200

Ürün (Product) Servisi Test Senaryoları

Senaryo No	Açıklama
P1	User rolü ürün listesi çeker → 200 + ürün listesi
P2	Admin yeni ürün ekler → 201
P3	Admin aynı isimde ürün ekler → 409
P4	Admin ürün günceller → 200
P5	Admin olmayan kullanıcı ürün silmeye çalışır → 403
P6	Ürün detay bilgisi çeklilir → ürün detay döner
P7	Geçersiz ID → 404
P8	Toplu ürün ekleme isteği → 201 ve ürün listesi döner

Alışveriş Sepeti (Cart) Test Senaryoları

Senaryo No	Açıklama
S1	Kullanıcı sepete ürün ekler → 200 OK
S2	Sepete aynı ürünü tekrar ekler → miktar güncellenir
S3	Kullanıcı sepeti listeler → ürün bilgileri ile döner
S4	Sepetten ürün çıkarır → 200 OK
S5	Sepeti temizler → sepet boş döner
S6	Ürün ekleme sırasında stok yetersizse → 400 Bad Request
S7	Login olmayan kullanıcı sepet ekleme yaparsa → 401

Sipariş Test Senaryoları

Senaryo No	Açıklama
O1	Kullanıcı sepetteki ürünleri sipariş verir → 201 + sipariş ID
O2	Sepet boşsa sipariş işlemi → 400
O3	Sipariş sonrası stoktan düşülür → stok güncellenir
O4	Kullanıcı sipariş geçmişini görüntüler → 200 + liste
O5	Yetkisiz kullanıcı sipariş geçmişini görüntüler → 403

Validasyon Test Senaryoları (Genel)

Senaryo No	Açıklama
V1	Şifre en az 8 karakter değil → 400
V2	Email formatı hatalı → 400
V3	Adres tipi "iş/ev" dışında bir şeyse → 400
V4	Telefon numarası regex ile valid değilse → 400
V5	JSON eksik alanla gönderilirse → 422 Unprocessable Entity

Arayüz Tasarımı

- UI framework tercihi serbesttir (örnek: React, Vue, Angular)
- Login olduktan sonra kullanıcının adı, soyadı ve diğer bilgileri sağ üst köşede gösterilecektir
- Kullanıcı bilgileri (adres, iletişim) giriş ekranları olacaktır.
- Ürün listeleme ekranı yapılacaktır.
- Sepet otomasyonu ve ekranı yapılacaktır.
- Ödeme ekranı yapılacaktır.
- Admin paneli (Kullanıcı yönetimi) yapılacaktır.
- Ürün yönetim paneli yapılacaktır.
- Stok durum bilgisi ekranı yapılacaktır.
- Kullanıcı şifresini değiştirme ve logout işlemleri yine sağ üst köşede gösterilecektir.

Teslimat Beklentisi

- user.service ve product.service için ayrı repo veya proje klasörü
- API dokümantasyonu (Swagger/OpenAPI veya Postman Collection)
- Dockerfile ve docker-compose.yml dosyası
- Örnek testler dışında tüm endpointleri için pozitif ve negatif test senaryoları hazırlanmalıdır.
- Readme dosyasında:
    - Kurulum ve çalıştırma adımları
    - Kullanılan teknolojiler
    - API endpoint listesi
    - Varsayılan admin kullanıcı bilgileri
- Kodun temiz, okunabilir ve katmanlı olması (service, repository, model, controller ayrımı)