Bekommerce: Yapay Zekâ Chatbot Destekli Tam Entegre Bir E-Ticaret Platformu

Giriş

Bekommerce, yapay zekâ destekli chatbot entegrasyonu içeren, gerçek zamanlı ve çok katmanlı bir e-ticaret platformudur. Platform, kullanıcıların ürünleri keşfetmesine, filtrelemesine, alışveriş yapmasına, sipariş oluşturmasına ve doğal dil yoluyla ürün önerileri almasına imkân tanıyan modern bir çözüm sunmaktadır. Proje; frontend, backend ve chatbot bileşenlerinden oluşmakta olup her biri farklı teknolojiler kullanılarak geliştirilmiş, Docker konteynerleri aracılığıyla izole bir yapıda merkezi olarak orkestre edilmiştir.

Sistem, üç farklı kullanıcı rolünü desteklemektedir: Admin, Satıcı ve Alıcı. Giriş yapan kullanıcılar rollerine göre özelleştirilmiş hizmetlere erişebilirken, misafir kullanıcılar da ürünleri görüntüleyebilir ve kayıt/giriş işlemleri gerçekleştirebilir.

Genel Mimari

Platform üç temel servisten oluşmaktadır:

1. **Frontend:** React.js ve Vite ile geliştirilen istemci tarafı arayüz katmanı.
2. **Backend:** Java Spring Boot tabanlı, güvenli REST API'lerini ve veritabanı işlemlerini yöneten sunucu katmanı.
3. **Chatbot:** Python ve Flask kullanılarak geliştirilen, doğal dil işleme (NLP) ve görsel tanıma yetenekleriyle ürün önerileri sunan yapay zekâ destekli hizmet katmanı.

Bu bileşenler birbirinden bağımsız olarak geliştirilmiş ve Docker konteynerleri aracılığıyla aynı ağ üzerinde haberleşmeleri sağlanmıştır.

Frontend Katmanı

Kullanılan Teknolojiler

- **Dil & Framework:** JavaScript (ES6+), React 18
- **Yapılandırma Aracı:** Vite– hızlı başlangıç ve geliştirme sunucusu
- **UI Bileşen Kütüphanesi:** MUI (Material Ui) – modern ve erişilebilir bileşenler
- **Routing: React Router** DOM v6 – tek sayfa uygulamalar için istemci taraflı yönlendirme
- **HTTP İletişimi:** Axios – RESTful API'lere asenkron erişim
- **Durum Yönetimi:** LocalStorage + context tabanlı yönetim (ör. JWT saklama)

Mimarî Yapı ve Sayfa Organizasyonu

Uygulama, farklı kullanıcı rollerine (Misafir, Alıcı, Satıcı, Admin) özel sayfa yönlendirmeleriyle yapılandırılmıştır. Tüm bu rotalar AppRouter.jsx bileşeni içerisinde react-router-dom ile tanımlanmış olup, erişim katmanı yönetimi her role uygun olacak şekilde ayrıştırılmıştır.

- **Misafir (Guest):** Ana sayfa, ürün detayları, giriş ve kayıt gibi temel sayfalara erişebilir.
- **Alıcı (Buyer):** Siparişler, sepet, favoriler, profil, destek ve yorum modüllerini kapsayan sayfalara erişebilir.
- **Satıcı (Seller):** Ürün yönetimi, sipariş & kargo durumu, kazançlar, istatistikler, destek ve profil gibi işlemleri gerçekleştirebilir.
- **Yönetici (Admin):** Kullanıcı yönetimi, sistem ürünleri, siparişler ve analiz panellerine erişebilir.

Ana Özellikler

- **Dinamik Ürün Listeleme:**
    - Ürünler; kategori, marka, fiyat aralığı ve stok durumu gibi filtreleme kriterlerine göre listelenebilir.
    - Bu filtreleme özelliği, kullanıcı deneyimini artırmak için client-side filtreleme mantığı ve backend Specification API ile desteklenmiştir.
- **JWT Tabanlı Oturum Yönetimi:**
    - Kullanıcı kimlik doğrulama işlemleri login/register akışı üzerinden gerçekleştirilir.
    - JWT token'ı localStorage içinde saklanır ve her istekte Authorization başlığı ile gönderilir (bkz. authHeader() fonksiyonu).
    - Oturum kapatma işlemleri token'ın temizlenmesi ile sağlanır.
- **Sipariş Modülü:**
    - Kullanıcılar sipariş oluşturabilir, geçmiş siparişlerini görüntüleyebilir ve detay sayfaları üzerinden ürün/hizmet deneyimlerine erişebilir.
    - Sipariş verileri backend API üzerinden dinamik olarak çekilir.
- **Kullanıcı Yönetimi (Admin):**
    - Yönetici kullanıcılar, sistemdeki tüm kullanıcıları listeleyebilir, güncelleyebilir veya silebilir.
    - Kullanıcı verileri güvenli API çağrıları ile alınır (getAllUsers, updateUser, deleteUser).

- **Responsive Tasarım:**
    - Material UI ve modern CSS yaklaşımları (grid, flexbox, media queries) ile geliştirilen kullanıcı arayüzü, mobil cihazlar ve masaüstü ekranlar için optimize edilmiştir.
    - Tüm sayfalar farklı cihaz çözünürlüklerine göre uyarlanmıştır.

HTTP API Servisi (services/api.js)

Uygulama, tüm frontend-backend iletişimini Axios kütüphanesi ile sağlar. api.js modülü üzerinden dışa aktarılan fonksiyonlar şunlardır:

- **Kimlik Doğrulama İşlemleri:**
    - loginUser(email, password): Giriş yapma
    - registerUser(name, email, password, role): Yeni kullanıcı kaydı
    - logoutUser(): JWT temizleyerek oturum sonlandırma
- **Kullanıcı İşlemleri:**
    - getUserById(id): Belirli kullanıcıyı getirir
    - getAllUsers(): Tüm kullanıcı listesini çeker (Yönetici rolü için)
    - updateUser(id, updatedUser): Kullanıcı bilgilerini günceller
    - deleteUser(id): Kullanıcı silme işlemi
- **Ürün İşlemleri:**
    - getAllProducts(): Admin paneli için sistemdeki tüm ürünleri listeleme (JWT doğrulamalı)

Her bir API fonksiyonu, Authorization: Bearer <token> başlığını otomatik olarak gönderir ve bu sayede yalnızca yetkili erişime izin verilir. Bu yaklaşım, sistemin güvenlik, erişim kontrolü ve performans açısından bütüncül olarak çalışmasına olanak sağlar.

Backend Katmanı

Kullanılan Teknolojiler

- Dil: Java 23
- Framework: Spring Boot
- Veritabanı: PostgreSQL
- ORM: Hibernate (Spring Data JPA)
- API Belgelendirme: Springdoc / Swagger UI

- Güvenlik: Spring Security + JWT
- CORS: Özelleştirilmiş CORS konfigürasyonu ile client-backend güvenli iletişim

Temel Bileşenler

1. **Entity Sınıfları**
   Backend uygulamasında kullanılan Entity sınıfları, veritabanı ile olan etkileşimi yöneten ana bileşenlerdir. Uygulamada yer alan başlıca varlık sınıfları:
   - Product: Ürünlerin detaylarını (isim, fiyat, stok durumu, açıklama, vb.) içerir.
   - User: Kullanıcı bilgilerini (isim, e-posta, şifre, rol) depolar. Her kullanıcının Admin, Seller (satıcı) veya Buyer (alıcı) rolüne sahip olmasını sağlar.
   - Order: Siparişlerin detaylarını, satın alınan ürünleri ve siparişin durumunu takip eder.
   - OrderItem: Bir siparişteki her bir ürünün detaylarını (fiyat, miktar) tutar.
   - ProductReview: Ürünlere yapılan yorumları ve verilen puanları içerir.
   - Favorites: Kullanıcıların favori ürünlerini listeler.
   - Cart: Kullanıcıların alışveriş sepetlerini içerir.
2. **Kimlik Doğrulama**
   JWT (JSON Web Token) tabanlı kimlik doğrulama, backend ile frontend arasında güvenli iletişim sağlayan önemli bir bileşendir. Kullanıcı, kimlik doğrulaması için e-posta ve şifresini sağlayarak JWT token alır. Bu token, kullanıcı her bir API çağrısında doğrulama amacıyla gönderilir.

JWT doğrulaması, Spring Security ile entegre edilmiştir ve sadece yetkilendirilmiş kullanıcıların API'ye erişmesi sağlanır. JWT token, her API isteğinde "Authorization" başlığı altında taşınır.

   - JWT token oluşturma ve doğrulama: Kullanıcı doğru kimlik bilgilerini sağladığında token oluşturulur ve kullanıcının her istekle birlikte bu token'ı göndermesi beklenir. Token doğrulandıktan sonra kullanıcı, istenilen verilere erişebilir.

3. **Yetkilendirme**
   Uygulama, kullanıcıların rollerine göre yetkilendirme yapmaktadır. Kullanıcılar, BUYER, SELLER veya ADMIN gibi rollerle sisteme giriş yapabilirler. Bu roller, kullanıcının uygulamadaki hangi kaynaklara erişebileceğini belirler.
   - Admin: Tüm verilere erişim sağlar ve kullanıcı yönetimi gibi işlemleri yapabilir.
   - Seller: Kendi ürünlerini ekleyebilir, güncelleyebilir, satış yapabilir.
   - Buyer: Ürünleri görüntüleyebilir, favorilerine ekleyebilir ve sipariş verebilir.
4. **Gelişmiş Arama**
   Specification API kullanılarak esnek ve güçlü bir arama özelliği sunulmaktadır. Bu özellik,

kullanıcılara ürünleri farklı kriterlere göre filtreleme imkanı tanır. Ürünler, kategori, fiyat aralığı, stok durumu ve diğer özelliklere göre sıralanabilir.

- Specification sınıfları, ürünlerin kategori, fiyat, stok durumu gibi özelliklere göre sorgulanmasına olanak tanır.
- Dinamik arama istekleri, Spring Data JPA kullanılarak veritabanına yönlendirilir ve esnek filtreleme yapılabilir.

5. **CORS (Cross-Origin Resource Sharing)**
   CORS özelleştirilmiş yapılandırması, frontend ve backend arasındaki güvenli iletişimi sağlar. Özellikle geliştirme sürecinde frontend ve backend farklı portlarda çalıştığında, CORS politikası devreye girer.

Backend, frontend'in taleplerine güvenli bir şekilde cevap verebilmesi için CORS yapılandırmasını özelleştirir. Bu sayede sadece belirli frontend adreslerinden gelen talepler kabul edilir. Uygulama, bu yapılandırmayla http://localhost:5173 (React frontend adresi) gibi belirli kaynaklardan gelen istekleri kabul eder.

```
@Configuration
public class CorsConfig {
  @Bean
  public WebMvcConfigurer corsConfigurer() {
    return new WebMvcConfigurer() {
      @Override
      public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
            .allowedOrigins("http://localhost:5173")  // React URL'sini ekle
            .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS")
            .allowedHeaders("*")
            .allowCredentials(true);
      }
    };
  }
}
```

Özet

Backend katmanı, Spring Boot kullanılarak geliştirilmiş ve PostgreSQL veritabanı ile entegre edilmiştir. Uygulamanın güvenliği, JWT tabanlı kimlik doğrulama ve Spring Security ile sağlanmaktadır. Kullanıcıların veritabanı işlemleri ise Hibernate (Spring Data JPA) ile yapılmaktadır. API belgelendirmesi için Swagger UI ve Springdoc kullanılarak, API uç noktaları hakkında otomatik dökümantasyon oluşturulmaktadır. Ayrıca, CORS özelleştirilmiş yapılandırması ile frontend ve backend arasında güvenli iletişim sağlanmaktadır.

Chatbot Servisi

Kullanılan Teknolojiler

- **Programlama Dili**: Python 3.13
- **Web Framework**: Flask
- **Veritabanı Bağlantısı**: psycopg2 (PostgreSQL entegrasyonu için)
- **Doğal Dil İşleme (NLP) Modeli**: paraphrase-multilingual-MiniLM-L12-v2 (SentenceTransformers)
- **Görsel Tanıma Modeli**: clip-vit-base-patch32 (OpenAI CLIP)
- **Benzerlik Hesaplama**: Kosinüs benzerliği (scikit-learn)

Sistem Tasarımı ve İşleyiş Detayları

Chatbot servisi, kullanıcı deneyimini en üst seviyeye çıkarmak amacıyla metin ve görsel verileri çok modlu biçimde işleyebilecek şekilde yapılandırılmıştır. Uygulamanın bu yönü, yalnızca kategorik eşleşmeye değil, aynı zamanda bağlamsal ve semantik benzerliğe dayalı ürün önerileri sunabilmesini mümkün kılmıştır.

Çok Modlu Girdi İşleme

- **Metin Girdisi (Text Input):**
  Kullanıcının metinsel talepleri, dil bağımsız ve çok dilli desteğe sahip SentenceTransformer modeli kullanılarak anlam vektörlerine dönüştürülmektedir. Bu vektörler, önceden tanımlı ürün kategorilerine ait anahtar kelimelerle karşılaştırılarak en uygun kategori tespit edilir. Bu yapı sayesinde sistem, karmaşık veya doğal dilde ifade edilmiş kullanıcı isteklerini doğru şekilde anlamlandırabilmektedir.
- **Görsel Girdisi (Image Input):**
  Yapılan önemli iyileştirmeler neticesinde, sistem artık görsel içerikten yalnızca kategori değil, aynı zamanda **ürünle ilişkili temel özellikleri** de tespit edebilmektedir. CLIP modelinin çapraz-modal karşılaştırma yetenekleri sayesinde görseldeki nesneler, konseptler ve bağlam algılanarak sistemin öneri kararı bu verilerle zenginleştirilmektedir. Örneğin bir çanta görseli verildiğinde, sadece “Moda” kategorisi değil, aynı zamanda bu

çantanın günlük, spor veya resmi kullanıma uygun olup olmadığı da vektör temelli içerik analiziyle anlaşılmaktadır.

Ürün Öneri Süreci

Bu versiyonda recommend_products() fonksiyonu yeniden yapılandırılarak yalnızca kategoriyi değil, aynı zamanda **görsel ve açıklama içeriğine** dayalı ürün özelliklerini de dikkate alacak biçimde genişletilmiştir. Bu sayede önerilen ürünler:

- Belirlenen kategoriyle yüksek örtüşme gösterir,
- Ürün açıklamaları ve kullanıcı yorumları bağlamında analiz edilir,
- Görsel ve içerik temelli benzerlik düzeyine göre filtrelenir,
- Ortalama kullanıcı puanı (rating) ve yorum sayısına göre sıralanır,
- En uygun **3 ürün**, kullanıcıya açıklamalı ve zengin görsel destekle sunulur.

Bu çok katmanlı öneri stratejisi, klasik anahtar kelime eşlemesi yapan sistemlerin ötesine geçerek; **daha kişiselleştirilmiş, daha bağlamsal ve daha isabetli öneriler** üretmektedir.

Temel Fonksiyonlar

- **predict_category_from_text(prompt: str):**
  Metin girişini anlamlandırarak en uygun ürün kategorisini tahmin eder.
  SentenceTransformer ile semantik analiz yapılır.
- **predict_category_from_image_url(image_url: str):**
  Görseldeki içeriği analiz ederek kategori ve ürün türü hakkında tahminlerde bulunur. CLIP modelinin görsel-metin eşleştirme kapasitesi kullanılır.
- **recommend_products(category: str):**
  Tahmin edilen kategoriye ve çıkarılan içerik özelliklerine göre veritabanından ürünleri çeker. Ürünlerin açıklama, puan ve görsel analizlerine göre sıralama yapılır.

REST API Mimarisi

Flask tabanlı API, /chat endpoint'i aracılığıyla POST isteklerini kabul etmektedir. JSON verisi içerisindeki "message" alanı üzerinden gelen giriş metin ya da görsel URL olarak ayrıştırılmakta ve uygun işleme yönlendirilmektedir. Sistem yanıtı, kullanıcıya doğal dilde bir geri bildirim ve önerilen ürünlerle birlikte sunulmaktadır.

Bu yapıyla birlikte, chatbot servisi yalnızca temel bir öneri sistemi değil, aynı zamanda **çok modlu yapay zeka tabanlı karar destek sistemi** işlevi görmektedir. Uygulamanın bu şekilde kapsamlı hale getirilmesi, kullanıcıya daha anlamlı ve değerli bir etkileşim sunmakla kalmamış, aynı zamanda teknik olarak güçlü ve ileri düzey bir çözüm ortaya koymuştur.

Veritabanı Katmanı

İlişkisel modelleme kullanılarak ürünler, kullanıcılar, siparişler ve incelemeler arasında tutarlı ilişkiler kurulmuştur. En ilk başta internetten bazı veri setleri birleştirilmiş ve veri tabanı e-ticaret sitesine dönüşebilmesi için ek sütunlar ve tablolar ile büyütülmüştür. PostgreSQL tercih edilerek ACID uyumlu, güvenilir ve açık kaynaklı bir veri tabanı çözümü sağlanmıştır. Db klasörüne

“

pg_dump -U postgres -d bekommerce -f "C:\Users\ebube\Desktop\Dersler\Yapay Zeka\proje-kod-ve-raporu-Ebu13\db\bekommerce.sql"

”

komutu ile bekommerce.sql dosyasını oluşturarak Devops süreci için uygun hazır bir veritabanı oluşturulmuştur.

Konteynerleşme ve Dağıtım

Kullanılan Teknolojiler

- Docker: Her bir bileşen için izole çalışma ortamı
- Docker Compose: Servislerin birlikte konfigüre edilmesi ve orkestrasyonu
- Bridge Network: Servisler arası güvenli haberleşme altyapısı

Servis Yapılandırması

Servis	Teknoloji	Port
Frontend	React + Vite	3000
Backend API	Spring Boot	8081
Chatbot API	Flask	5000
Veritabanı	PostgreSQL	5432

Tüm servisler için özelleştirilmiş Dockerfile tanımları yapılmış ve docker-compose.yml dosyası ile tek komutla ayağa kaldırılabilir hâle getirilmiştir.

docker-compose up --build

komutu ile ayağı kaldırabilir.

Veri Seti

Veri seti anlık olarak zaten veritabanından çekiyor. Veri tabanındaki veriler tutarlı olduğundan veri ön işleme adımları zaten veri tabanı oluşturulurken yapılmıştır. Ürünler, kullanıcılar, yorumlar ve kategoriler gerçekçi bir senaryo oluşturacak şekilde çeşitlendirilmiş, sistem testleri ve chatbot eğitim süreci bu veri seti üzerinden yürütülmüştür.

Kazanımlar ve Sonuç

Bekommerce projesi, full-stack yazılım geliştirme, mikroservis mimarisi, konteynerleşme ve yapay zekâ entegrasyonu gibi güncel yazılım mühendisliği yaklaşımlarını bütüncül bir biçimde bir araya getirmiştir. Proje boyunca güvenlik, ölçeklenebilirlik, kod okunabilirliği ve sürdürülebilirlik gibi değerler ön planda tutulmuştur.