# E-COMMERCE PLATFORM

## Backend Developer — Proje Mimari Dökümanı

E-Ticaret Mikroservis Platformu

*Go · PostgreSQL · Elasticsearch · Redis · Kafka · Docker · Kubernetes*
*Nginx · Keycloak · OpenID Connect · OAuth2*

# 1. Projeye Genel Bakış

Bu proje, üretim ortamına hazır bir e-ticaret mikroservis platformudur. Kimlik doğrulama Keycloak üzerinden OAuth2 / OpenID Connect protokolleriyle yönetilir. Nginx tüm trafiğin giriş noktası olarak reverse proxy ve Keycloak token doğrulama görevlerini üstlenir. Servisler arası asenkron iletişim Kafka üzerinden sağlanır.

## 1.1 Teknoloji Stack

| Katman | Teknoloji | Kullanım Amacı |
|---|---|---|
| Backend Dil | Go 1.22+ | Tüm mikroservisler |
| HTTP Framework | Fiber v2 | REST API |
| Ana Veritabanı | PostgreSQL 16 | Transactional veriler |
| Arama Motoru | Elasticsearch 8 | Ürün araması, full-text, facets |
| Cache | Redis 7 | Cache, token blacklist |
| Message Broker | Apache Kafka | Servisler arası async iletişim |
| Reverse Proxy | Nginx | SSL termination, auth_request, routing |
| Identity Provider | Keycloak 24 | OAuth2, OpenID Connect, JWT, kullanıcı yönetimi |
| Containerization | Docker + Docker Compose | Geliştirme ortamı |
| Orchestration | Kubernetes | Prodüksiyon dağıtımı |
| Monitoring | Prometheus + Grafana | Metrik ve dashboard |
| Tracing | Jaeger (OpenTelemetry) | Distributed tracing |
| CI/CD | GitHub Actions + Terraform | Pipeline & infra as code |

# 2. Sistem Mimarisi

## 2.1 Genel Mimari

Tüm dış trafik Nginx üzerinden girer. Nginx her korumalı isteği önce Keycloak'a yönlendirerek token doğrular, başarılı olursa X-User-ID ve X-User-Role header'larını ekleyerek downstream servise iletir. Servisler arası senkron iletişim HTTP, asenkron iletişim Kafka üzerinden sağlanır.

### Katmanlı Akış

```
Client
  ↓
Nginx  (SSL termination · reverse proxy · auth_request)
  ↓ korumalı endpoint'ler için
Keycloak  (token introspection → 200 OK / 401)
  ↓ 200 OK ise
Downstream Service  (X-User-ID, X-User-Role header'larıyla)
```

## 2.2 Keycloak Sorumlulukları

- Kullanıcı kaydı ve girişi (OAuth2 Resource Owner Password / Authorization Code)
- JWT üretimi ve imzalama (RS256 — asimetrik anahtar)
- Access token (15 dakika) + Refresh token (7 gün) yönetimi
- Token introspection endpoint: Nginx auth_request buraya bakar
- Kullanıcı rolleri: customer, admin
- Realm ve Client konfigürasyonu

## 2.3 Nginx Sorumlulukları

- SSL termination
- Public endpoint'leri auth_request'ten muaf tutma
- Korumalı endpoint'lerde Keycloak'a auth_request yönlendirme
- Keycloak'tan dönen X-User-ID, X-User-Role header'larını downstream'e iletme
- Upstream servislere reverse proxy

## 2.4 Nginx Konfigürasyon Yapısı

```
# Public endpoint'ler — auth_request YOK
location /auth/                          { proxy_pass http://keycloak:8080; }
location /api/v1/search/products         { proxy_pass http://search-service:8086; }
location /api/v1/products                { proxy_pass http://product-service:8082; }

# Korumalı endpoint'ler — auth_request VAR
location /api/v1/orders {
  auth_request /internal/auth/validate;
  auth_request_set $user_id   $upstream_http_x_user_id;
  auth_request_set $user_role $upstream_http_x_user_role;
  proxy_set_header X-User-ID   $user_id;
```

```
  proxy_set_header X-User-Role $user_role;
  proxy_pass http://order-service:8083;
}

# Auth validate iç endpoint
location = /internal/auth/validate {
  internal;
  proxy_pass http://keycloak:8080/auth/realms/ecommerce/protocol/openid-connect/userinfo;
  proxy_set_header Authorization $http_authorization;
}
```

# 3. Mikroservis Mimarisi

## 3.1 Servis Listesi

| Servis | Port | Sorumluluk | Veritabanı |
|---|---|---|---|
| user-service | 8081 | Kullanıcı profili, adres yönetimi (auth Keycloak'ta) | PostgreSQL (users_db) |
| product-service | 8082 | Ürün CRUD, kategori, stok yönetimi, Outbox | PostgreSQL (products_db) |
| order-service | 8083 | Sipariş oluşturma, Saga koordinasyonu | PostgreSQL (orders_db) |
| payment-service | 8084 | Ödeme işleme, iade, idempotency | PostgreSQL (payments_db) |
| notification-service | 8085 | Kafka consumer, email/SMS gönderim | PostgreSQL (notifications_db) |
| search-service | 8086 | Elasticsearch wrapper, full-text arama | Elasticsearch |
| Nginx | 80/443 | Reverse proxy, auth_request, SSL | — |
| Keycloak | 8080 | Identity Provider, OAuth2, JWT | PostgreSQL (keycloak_db) |

## 3.2 user-service

Keycloak devreye girmesiyle user-service'in auth sorumluluğu değişmiştir. Register, login, logout ve token yönetimi tamamen Keycloak'tadır. user-service yalnızca uygulama spesifik kullanıcı verisini yönetir.

- GET/PUT /api/v1/users/me — Profil görüntüleme ve güncelleme
- GET /api/v1/users/me/addresses — Adres listesi
- POST /api/v1/users/me/addresses — Yeni adres
- DELETE /api/v1/users/me/addresses/:id — Adres silme
- X-User-ID header'ından Keycloak user_id alınır, profil bu ID ile eşleştirilir

## 3.3 product-service

Ürün kataloğunun CRUD operasyonlarını, kategori hiyerarşisini ve stok yönetimini sağlar. Ürün değişikliklerinde Outbox Pattern ile Kafka event'i yayınlanır, search-service bu event'i tüketerek Elasticsearch'i günceller.

- Outbox Pattern: DB transaction içinde hem ürün hem outbox_event yazılır
- Stok azaltma: Pessimistic lock ile race condition önleme
- Admin rolü kontrolü: X-User-Role header'ından doğrulanır

## 3.4 order-service

Saga pattern (Choreography) ile sipariş sürecini yönetir. Stok rezervasyonu, ödeme ve onay adımlarını Kafka event'leri üzerinden koordine eder. Hata durumunda compensating transaction'lar devreye girer.

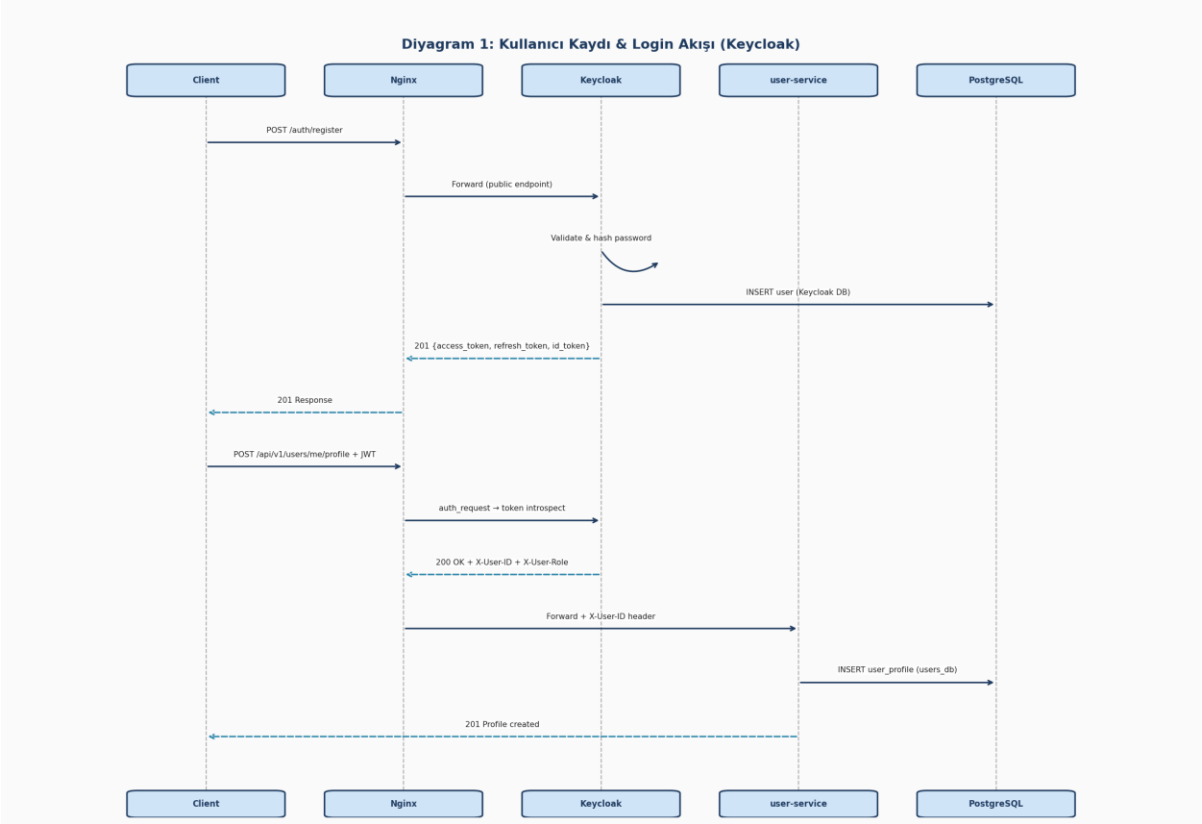
### 3.5 search-service

Elasticsearch üzerine ince bir wrapper'dır. Full-text arama, faceted filtreleme, autocomplete ve relevance scoring sağlar. Public endpoint olduğu için Nginx'te auth_request muafiyeti vardır.

# 4. Akış Diyagramları

Her diyagram hem görsel hem de Mermaid kodu olarak sunulmuştur. Mermaid kodunu mermaid.live adresinde interaktif olarak görüntüleyebilirsiniz.

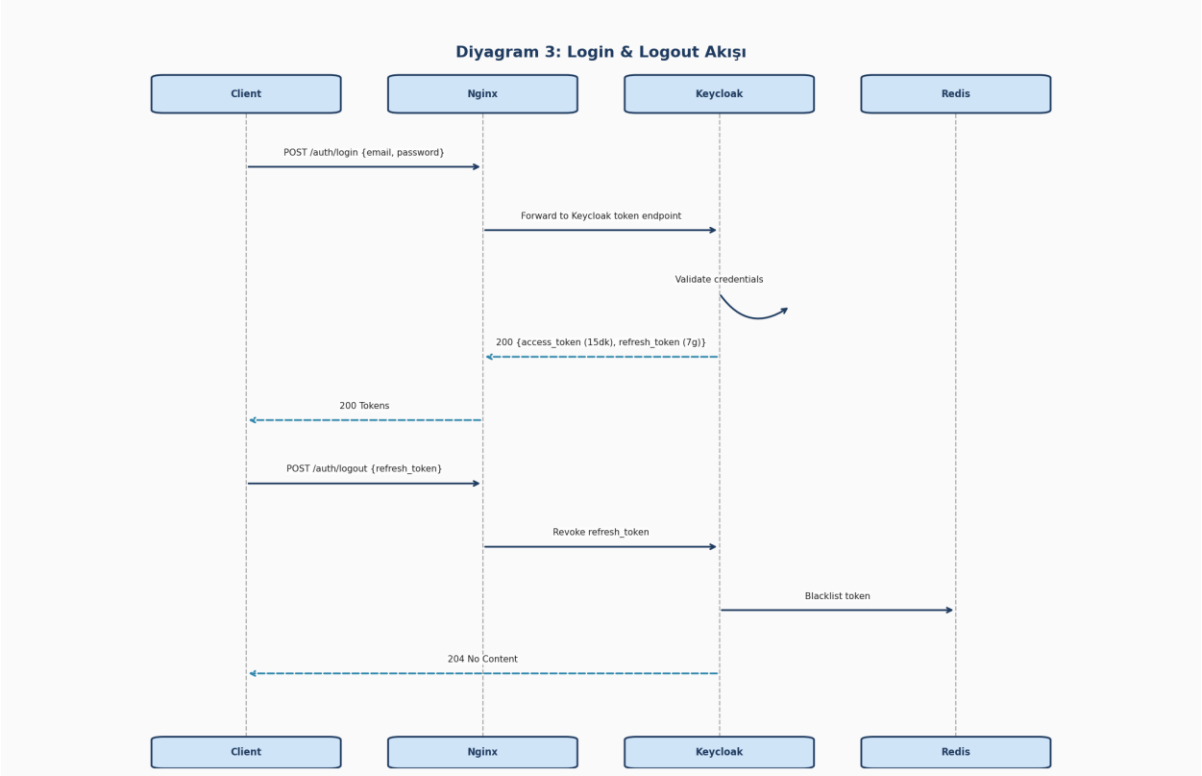
## 4.1 Kullanıcı Kaydı & Profil Oluşturma Akışı




**Mermaid Kodu:**

```
sequenceDiagram
  participant C as Client
  participant N as Nginx
  participant K as Keycloak
  participant US as user-service
  participant DB as PostgreSQL
  C->>N: POST /auth/register
  N->>K: Forward (public endpoint)
  K->>K: Validate input, hash password
  K->>DB: INSERT user (Keycloak DB)
  K-->>N: 201 {access_token, refresh_token, id_token}
  N-->>C: 201 Response
  Note over C,K: JWT içinde sub (user_id) ve roles bulunur
  C->>N: POST /api/v1/users/me/profile (ilk profil)
  N->>K: auth_request → token introspect
  K-->>N: 200 OK + X-User-ID + X-User-Role
  N->>US: Forward + X-User-ID header
  US->>DB: INSERT user_profile (users_db)
  US-->>C: 201 Profile created
```

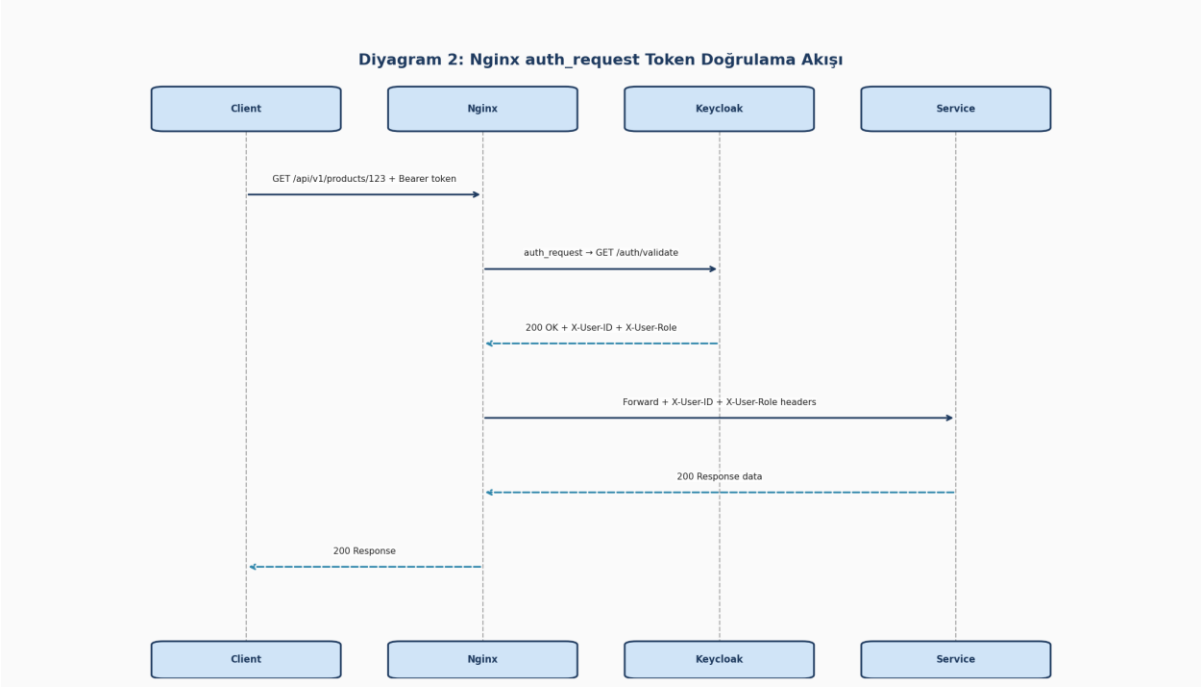
## 4.2 Login & Logout Akışı




**Mermaid Kodu:**

```
sequenceDiagram
  participant C as Client
  participant N as Nginx
  participant K as Keycloak
  participant R as Redis
  C->>N: POST /auth/login {email, password}
  N->>K: Forward to Keycloak token endpoint
  K->>K: Validate credentials
  K-->>N: 200 {access_token (15dk), refresh_token (7g), id_token}
  N-->>C: 200 Tokens
  Note over C,K: access_token = signed JWT (RS256)
  C->>N: POST /auth/logout {refresh_token}
  N->>K: Revoke refresh_token
  K->>R: Blacklist token
  K-->>C: 204 No Content
```

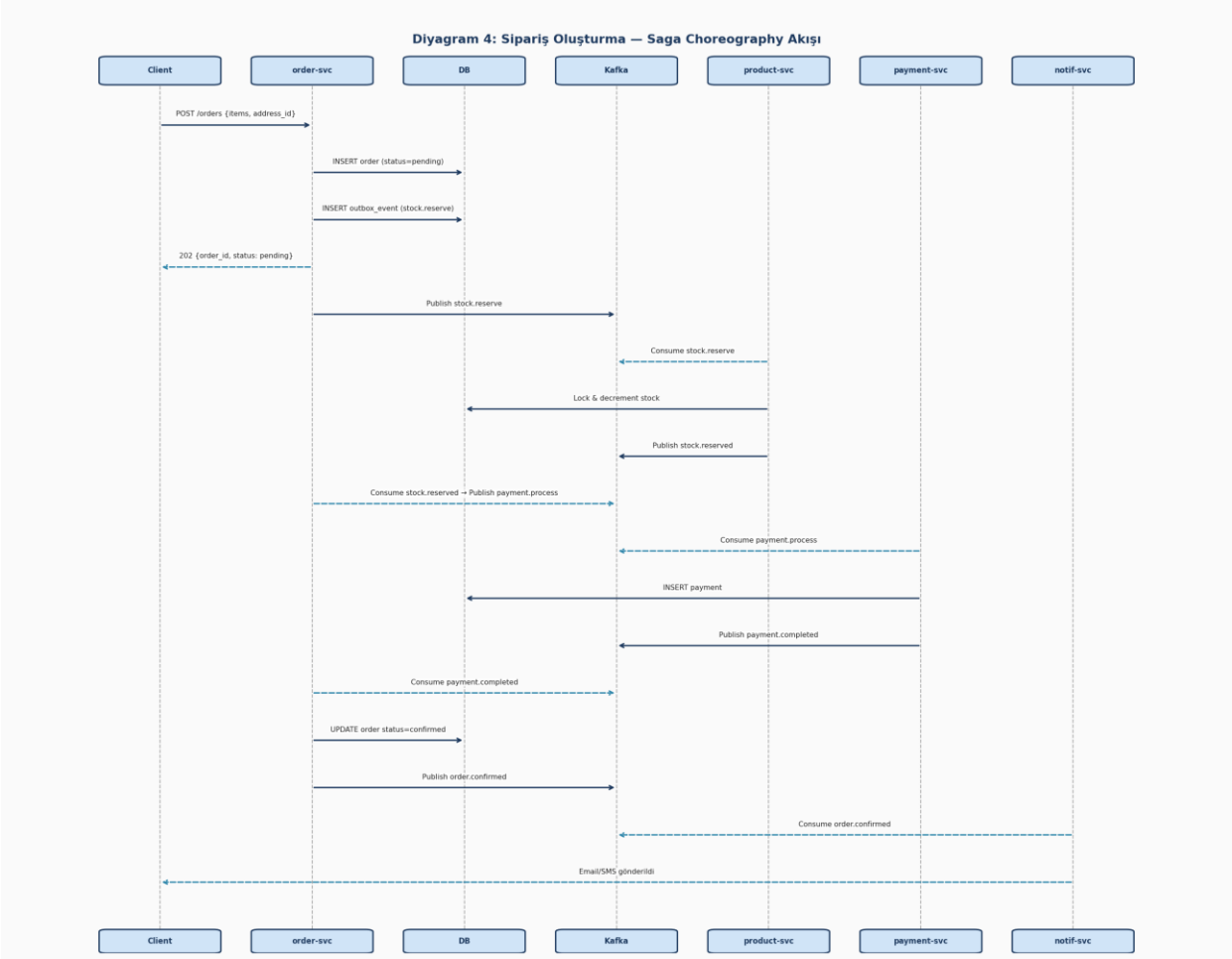
## 4.3 Nginx auth_request Token Doğrulama Akışı




**Mermaid Kodu:**

```
sequenceDiagram
  participant C as Client
  participant N as Nginx
  participant K as Keycloak
  participant S as Downstream Service
  C->>N: GET /api/v1/products/123 + Bearer token
  N->>K: auth_request → GET /auth/validate + token
  alt Token Geçerli
    K-->>N: 200 OK + X-User-ID + X-User-Role
    N->>S: Forward + X-User-ID + X-User-Role headers
    S-->>N: 200 Response
    N-->>C: 200 Response
  else Token Geçersiz / Süresi Dolmuş
    K-->>N: 401 Unauthorized
    N-->>C: 401 Unauthorized
  end
```

## 4.4 Sipariş Oluşturma — Saga Choreography Akışı

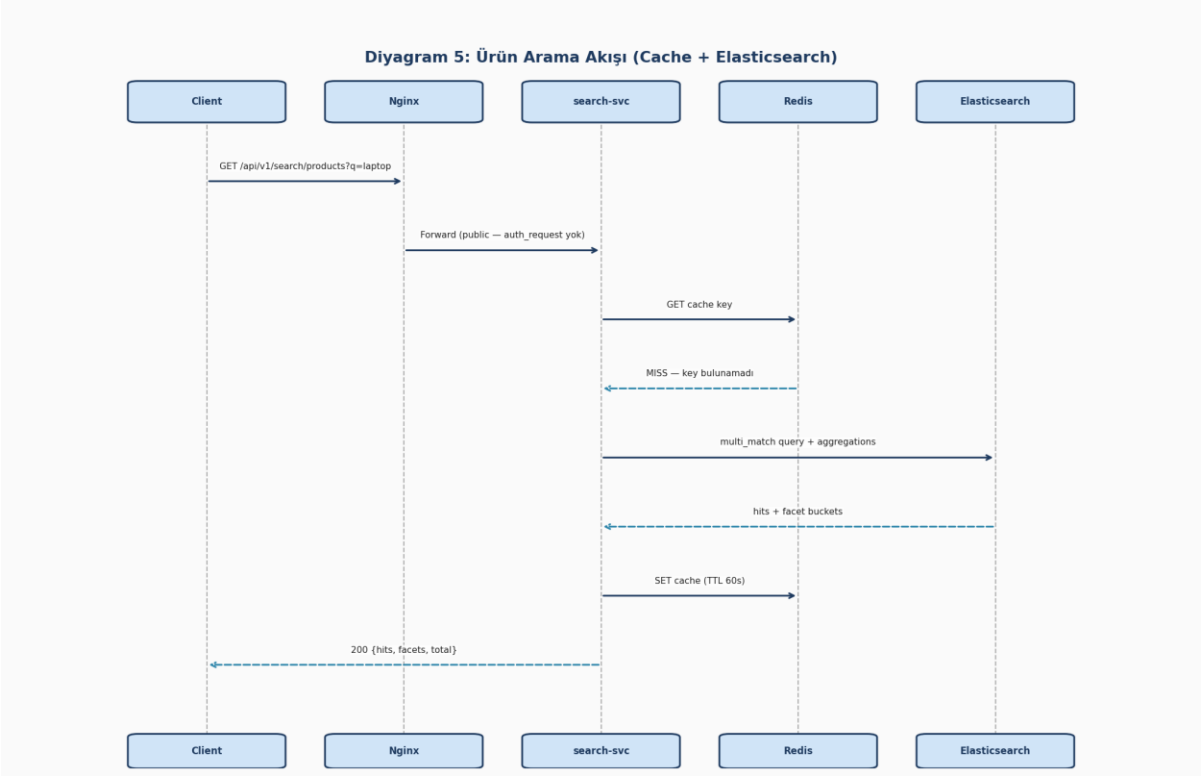



**Mermaid Kodu:**

```
sequenceDiagram
  participant C as Client
  participant OS as order-service
  participant DB as PostgreSQL
  participant KF as Kafka
  participant PS as product-service
  participant PAY as payment-service
  participant NS as notification-service
  C->>OS: POST /orders {items, address_id}
  OS->>DB: INSERT order (status=pending)
  OS->>DB: INSERT outbox_event (stock.reserve)
  OS-->>C: 202 {order_id, status: pending}
  OS->>KF: Publish stock.reserve
  PS->>KF: Consume stock.reserve
  PS->>DB: Lock & decrement stock
  PS->>KF: Publish stock.reserved
  OS->>KF: Consume stock.reserved → Publish payment.process
  PAY->>KF: Consume payment.process
  PAY->>DB: INSERT payment
  PAY->>KF: Publish payment.completed
  OS->>KF: Consume payment.completed
  OS->>DB: UPDATE order status=confirmed
```

```
  OS->>KF: Publish order.confirmed
  NS->>KF: Consume order.confirmed
  NS-->>C: Email/SMS gönderildi
```

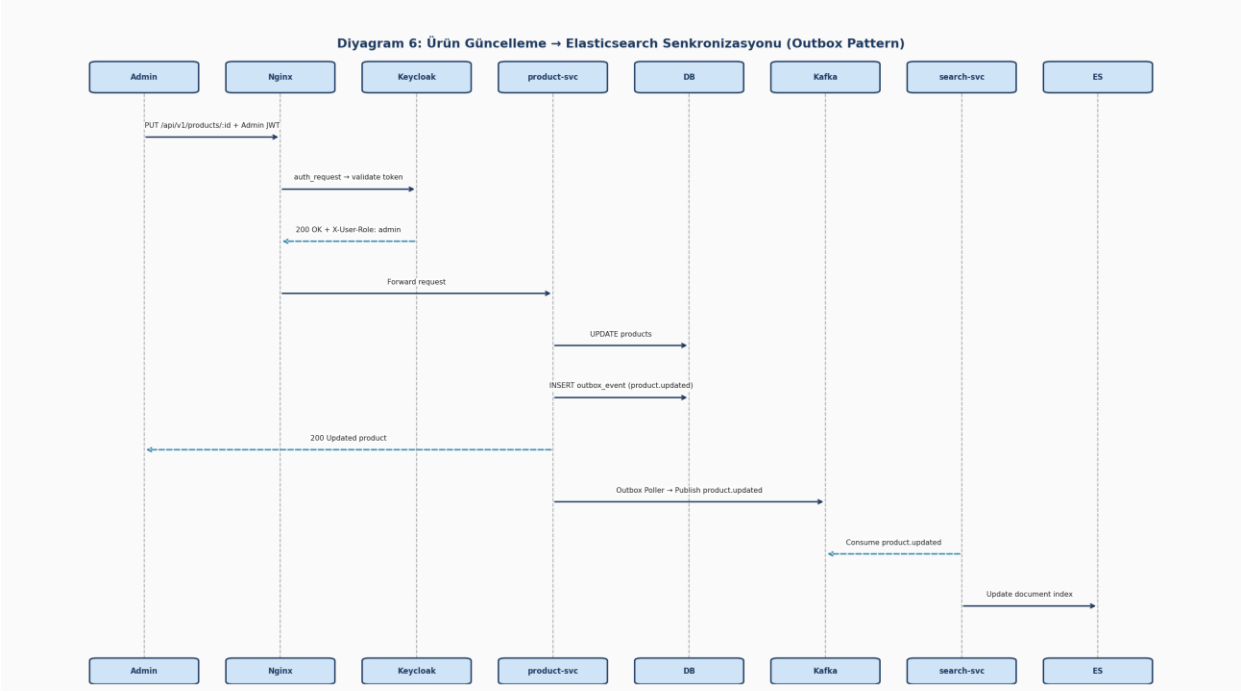
## 4.5 Ürün Arama Akışı (Cache + Elasticsearch)




**Mermaid Kodu:**

```
sequenceDiagram
  participant C as Client
  participant N as Nginx
  participant SS as search-service
  participant R as Redis
  participant ES as Elasticsearch
  C->>N: GET /api/v1/search/products?q=laptop
  N->>SS: Forward (public — auth_request yok)
  SS->>R: GET cache key
  alt Cache HIT
    R-->>SS: Cached result
  else Cache MISS
    SS->>ES: multi_match query + aggregations
    ES-->>SS: hits + facet buckets
    SS->>R: SET cache (TTL 60s)
  end
  SS-->>C: 200 {hits, facets, total}
```

## 4.6 Ürün Güncelleme → Elasticsearch Senkronizasyonu (Outbox Pattern)




**Mermaid Kodu:**

```
sequenceDiagram
  participant A as Admin Client
  participant N as Nginx
  participant K as Keycloak
  participant PS as product-service
  participant DB as PostgreSQL
  participant KF as Kafka
  participant SS as search-service
  participant ES as Elasticsearch
  A->>N: PUT /api/v1/products/:id + Admin JWT
  N->>K: auth_request → validate token
  K-->>N: 200 OK + X-User-Role: admin
  N->>PS: Forward request
  PS->>DB: UPDATE products
  PS->>DB: INSERT outbox_event (product.updated)
  PS-->>A: 200 Updated product
  Note over PS,DB: Outbox Poller çalışır (100ms interval)
  PS->>KF: Publish product.updated
  SS->>KF: Consume product.updated
  SS->>ES: Update document index
```

# 5. Endpoint Kataloğu & Authentication Haritası

Renk kodlaması: Yeşil = Public, Mavi = Keycloak JWT gerekli, Turuncu = Admin JWT gerekli, Mor = Internal (servisler arası)

## 5.1 Auth Endpoints (Keycloak — Nginx üzerinden)

| Method | Path | Auth | Açıklama |
|---|---|---|---|
| POST | /auth/realms/ecommerce/protocol/openid-connect/token | Public | Login — access_token + refresh_token döner |
| POST | /auth/realms/ecommerce/protocol/openid-connect/registrations | Public | Kullanıcı kaydı |
| POST | /auth/realms/ecommerce/protocol/openid-connect/logout | Public | Logout — refresh_token revoke |
| POST | /auth/realms/ecommerce/protocol/openid-connect/token (grant_type=refresh_token) | Public | Token yenileme |

## 5.2 user-service Endpoints

| Method | Path | Auth | Açıklama |
|---|---|---|---|
| GET | /api/v1/users/me | Keycloak JWT | Profil görüntüleme |
| PUT | /api/v1/users/me | Keycloak JWT | Profil güncelleme |
| GET | /api/v1/users/me/addresses | Keycloak JWT | Adres listesi |
| POST | /api/v1/users/me/addresses | Keycloak JWT | Yeni adres ekle |
| PUT | /api/v1/users/me/addresses/:id | Keycloak JWT | Adres güncelle |
| DELETE | /api/v1/users/me/addresses/:id | Keycloak JWT | Adres sil |
| GET | /api/v1/admin/users | Admin JWT | Tüm kullanıcılar (admin) |

## 5.3 product-service Endpoints

| Method | Path | Auth | Açıklama |
|---|---|---|---|
| GET | /api/v1/products | Public | Ürün listesi (paginated) |
| GET | /api/v1/products/:id | Public | Ürün detayı |

| Method | Path | Auth | Açıklama |
|---|---|---|---|
| POST | /api/v1/products | Admin JWT | Yeni ürün oluştur |
| PUT | /api/v1/products/:id | Admin JWT | Ürün güncelle |
| DELETE | /api/v1/products/:id | Admin JWT | Ürün sil (soft delete) |
| GET | /api/v1/categories | Public | Kategori ağacı |
| POST | /api/v1/categories | Admin JWT | Yeni kategori |
| PATCH | /api/v1/products/:id/stock | Admin JWT | Stok güncelle |

## 5.4 order-service Endpoints

| Method | Path | Auth | Açıklama |
|---|---|---|---|
| POST | /api/v1/orders | Keycloak JWT | Sipariş oluştur (Saga başlatır) |
| GET | /api/v1/orders | Keycloak JWT | Kullanıcının siparişleri |
| GET | /api/v1/orders/:id | Keycloak JWT | Sipariş detayı |
| POST | /api/v1/orders/:id/cancel | Keycloak JWT | Sipariş iptal |
| GET | /api/v1/admin/orders | Admin JWT | Tüm siparişler (admin) |
| PATCH | /api/v1/admin/orders/:id/status | Admin JWT | Sipariş durumu güncelle |

## 5.5 payment-service Endpoints

| Method | Path | Auth | Açıklama |
|---|---|---|---|
| POST | /api/v1/payments | Internal | Ödeme oluştur (order-service çağırır) |
| GET | /api/v1/payments/:id | Keycloak JWT | Ödeme detayı |
| POST | /api/v1/payments/:id/refund | Admin JWT | İade işlemi |
| GET | /api/v1/admin/payments | Admin JWT | Tüm ödemeler |

## 5.6 search-service Endpoints

| Method | Path | Auth | Açıklama |
|---|---|---|---|
| GET | /api/v1/search/products | Public | Full-text arama, facets, filtreleme |
| GET | /api/v1/search/products/autocomplete | Public | Autocomplete önerileri (min 2 karakter) |
| GET | /api/v1/search/products/similar/:id | Public | Benzer ürünler |

# 6. Kafka Topikleri & Event Akışı

## 6.1 Topik Listesi

| Topik | Producer | Consumer(s) | Açıklama |
|---|---|---|---|
| product.created | product-service | search-service | Yeni ürün ES'e indekslenir |
| product.updated | product-service | search-service | Güncelleme ES'e yansıtılır |
| product.deleted | product-service | search-service | Ürün ES'den silinir |
| stock.reserve | order-service | product-service | Stok rezervasyonu isteği |
| stock.reserved | product-service | order-service | Stok rezervasyonu başarılı |
| stock.reserve.failed | product-service | order-service | Stok yetersiz — saga rollback |
| stock.release | order-service | product-service | İptal sonrası stok serbest bırak |
| payment.process | order-service | payment-service | Ödeme isteği |
| payment.completed | payment-service | order-service, notification-service | Ödeme başarılı |
| payment.failed | payment-service | order-service | Ödeme başarısız — saga rollback |
| order.confirmed | order-service | notification-service | Sipariş onaylandı |
| order.cancelled | order-service | product-service, notification-service | İptal edildi |

## 6.2 Saga Hata Senaryoları — Compensating Transactions

| Hata Noktası | Event | Compensating Action |
|---|---|---|
| Stok yetersiz | stock.reserve.failed | order-service → sipariş iptal, kullanıcıya bildirim |
| Ödeme başarısız | payment.failed | order-service → stock.release event, sipariş iptal |
| Saga timeout | — (scheduled job) | order-service → compensate all steps, status=failed |

# 7. Veritabanı Şemaları

## 7.1 users_db (PostgreSQL)

| Tablo | Kolon | Tip | Açıklama |
|---|---|---|---|
| user_profiles | id | UUID PK | Birincil anahtar |
| user_profiles | keycloak_id | UUID UNIQUE | Keycloak sub (user_id) |
| user_profiles | full_name | VARCHAR(255) | Ad Soyad |
| user_profiles | phone | VARCHAR(20) | Telefon |
| user_profiles | created_at | TIMESTAMPTZ | Kayıt tarihi |
| addresses | id | UUID PK | Adres ID |
| addresses | user_id | UUID FK | user_profiles referansı |
| addresses | title | VARCHAR(100) | Ev, İş vb. |
| addresses | street | TEXT | Cadde/sokak |
| addresses | city | VARCHAR(100) | Şehir |
| addresses | zip | VARCHAR(20) | Posta kodu |
| addresses | is_default | BOOLEAN | Varsayılan adres |

## 7.2 products_db (PostgreSQL)

| Tablo | Kolon | Tip | Açıklama |
|---|---|---|---|
| categories | id | UUID PK | Kategori ID |
| categories | parent_id | UUID FK NULL | Üst kategori (self-ref) |
| categories | name | VARCHAR(255) | Kategori adı |
| categories | slug | VARCHAR(255) UNIQUE | URL slug |
| products | id | UUID PK | Ürün ID |
| products | category_id | UUID FK | Kategori referansı |
| products | name | VARCHAR(500) | Ürün adı |
| products | description | TEXT | Açıklama |
| products | price | DECIMAL(12,2) | Fiyat |
| products | stock | INTEGER | Stok adedi |
| products | status | ENUM(active,passive,deleted) | Durum |
| products | created_at | TIMESTAMPTZ | Oluşturma tarihi |
| product_images | id | UUID PK | Resim ID |
| product_images | product_id | UUID FK | Ürün referansı |

| Tablo | Kolon | Tip | Açıklama |
|---|---|---|---|
| product_images | url | TEXT | Resim URL |
| product_images | sort_order | INTEGER | Sıralama |
| outbox_events | id | UUID PK | Event ID |
| outbox_events | aggregate_id | UUID | İlgili kayıt ID |
| outbox_events | event_type | VARCHAR(100) | Event tipi |
| outbox_events | payload | JSONB | Event verisi |
| outbox_events | published | BOOLEAN DEFAULT false | Yayınlandı mı |
| outbox_events | created_at | TIMESTAMPTZ | Oluşturma zamanı |

## 7.3 orders_db (PostgreSQL)

| Tablo | Kolon | Tip | Açıklama |
|---|---|---|---|
| orders | id | UUID PK | Sipariş ID |
| orders | user_keycloak_id | UUID | Keycloak user ID |
| orders | status | ENUM(pending,confirmed,cancelled,shipped,delivered) | Durum |
| orders | total_amount | DECIMAL(12,2) | Toplam tutar |
| orders | address_snapshot | JSONB | Sipariş anındaki adres |
| orders | created_at | TIMESTAMPTZ | Sipariş zamanı |
| order_items | id | UUID PK | Kalem ID |
| order_items | order_id | UUID FK | Sipariş referansı |
| order_items | product_id | UUID | Ürün ID (snapshot) |
| order_items | product_name | VARCHAR(500) | Ürün adı snapshot |
| order_items | unit_price | DECIMAL(12,2) | Birim fiyat snapshot |
| order_items | quantity | INTEGER | Adet |
| saga_states | saga_id | UUID PK | Saga ID |
| saga_states | order_id | UUID FK | Sipariş referansı |
| saga_states | current_step | VARCHAR(100) | Aktif adım |
| saga_states | status | ENUM(running,completed,failed) | Durum |

| Tablo | Kolon | Tip | Açıklama |
|---|---|---|---|
| saga_states | context | JSONB | Ara durum verisi |
| saga_states | updated_at | TIMESTAMPTZ | Son güncelleme |

# 8. Klasör & Proje Yapısı

```
ecommerce-platform/
├── services/
│   ├── user-service/
│   │   ├── cmd/main.go
│   │   ├── internal/
│   │   │   ├── handler/      # Fiber HTTP handlers
│   │   │   ├── service/      # Business logic
│   │   │   ├── repository/   # PostgreSQL layer
│   │   │   ├── domain/       # Entities, DTOs
│   │   │   └── config/
│   │   ├── migrations/
│   │   ├── Dockerfile
│   │   └── go.mod
│   ├── product-service/      # + outbox/ klasörü
│   ├── order-service/        # + saga/ klasörü
│   ├── payment-service/
│   ├── notification-service/ # + consumer/ klasörü
│   └── search-service/
├── shared/
│   └── pkg/
│       ├── logger/           # Zap wrapper
│       ├── tracing/          # OpenTelemetry
│       ├── kafka/            # Producer/Consumer wrappers
│       └── errors/           # Custom error types
├── infra/
│   ├── nginx/
│   │   ├── nginx.conf        # Ana konfigürasyon
│   │   └── conf.d/           # Servis bazlı upstream tanımları
│   ├── keycloak/
│   │   └── realm-export.json # Realm konfigürasyonu
│   ├── docker-compose.yml
│   ├── terraform/
│   └── k8s/
├── .github/
│   └── workflows/
│       ├── ci.yml
│       └── cd.yml
└── docs/
    └── openapi/              # Swagger YAML files
```

# 9. Docker Compose Servisleri

| Servis | Image | Port | Notlar |
|---|---|---|---|
| nginx | nginx:alpine | 80, 443 | nginx.conf mount edilir |
| keycloak | quay.io/keycloak/keycloak:24 | 8080 | Start-dev veya prod mode |
| keycloak-db | postgres:16-alpine | 5433 | Keycloak'a özel DB |
| user-db | postgres:16-alpine | 5434 | users_db |
| product-db | postgres:16-alpine | 5435 | products_db |
| order-db | postgres:16-alpine | 5436 | orders_db |
| payment-db | postgres:16-alpine | 5437 | payments_db |
| notification-db | postgres:16-alpine | 5438 | notifications_db |
| redis | redis:7-alpine | 6379 | Cache + blacklist |
| kafka | confluentinc/cp-kafka:7 | 9092 | Zookeeper ile birlikte |
| zookeeper | confluentinc/cp-zookeeper:7 | 2181 | Kafka koordinatörü |
| elasticsearch | elasticsearch:8.x | 9200 | Single-node dev |
| kafdrop | obsidiandynamics/kafdrop | 9000 | Kafka UI (dev) |
| kibana | kibana:8.x | 5601 | ES UI (dev) |

# 10. Önerilen Geliştirme Sırası

| Hafta | Görev | Tamamlanma Kriteri |
|---|---|---|
| 1 | Repo yapısı + docker-compose | Tüm altyapı ayakta, Keycloak admin paneli açılıyor |
| 1-2 | Keycloak realm & client konfigürasyonu | Register/login Postman'dan çalışıyor, JWT üretiliyor |
| 2-3 | Nginx + auth_request entegrasyonu | Korumalı endpoint'e token olmadan 401 dönüyor |
| 3-4 | user-service (profil + adres) | Register sonrası profil oluşturma uçtan uca çalışıyor |
| 5-6 | product-service (CRUD + Outbox) | Ürün oluşturunca Kafka'ya event düşüyor |
| 7 | search-service (ES entegrasyonu) | Ürün araması ES'ten dönüyor, cache çalışıyor |
| 8-9 | order-service (temel akış) | Sipariş oluşturuluyor, status takip ediliyor |
| 10-11 | Saga implementasyonu | Tam akış: stok→ödeme→onay→bildirim |
| 12 | payment-service | Mock ödeme, idempotency key, iade |
| 13 | notification-service | Kafka consumer, email mock |
| 14 | Observability | Prometheus metrikler, Jaeger tracing |
| 15-16 | CI/CD + K8s manifests | GitHub Actions pipeline yeşil, K8s'e deploy |

## 10.1 Önerilen Go Kütüphaneleri

| Kütüphane | Kullanım |
|---|---|
| github.com/gofiber/fiber/v2 | HTTP framework |
| github.com/gofiber/contrib/otelfiber | Fiber için OpenTelemetry middleware |
| github.com/ansrivas/fiberprometheus/v2 | Fiber için Prometheus metrics |
| github.com/jackc/pgx/v5 | PostgreSQL driver |
| github.com/redis/go-redis/v9 | Redis client |
| github.com/segmentio/kafka-go | Kafka producer/consumer |
| github.com/olivere/elastic/v7 | Elasticsearch client |
| github.com/Nerzal/gocloak/v13 | Keycloak Go client (token introspection) |
| go.uber.org/zap | Structured logging |
| go.opentelemetry.io/otel | Distributed tracing |
| github.com/golang-migrate/migrate/v4 | DB migration |
| github.com/stretchr/testify | Test assertion |
| github.com/testcontainers/testcontainers-go | Integration test containers |

# 11. Test Stratejisi

## 11.1 Unit Test

Her servisin service katmanı, repository interface mock'lanarak test edilir. testify/mock kullanılır. Keycloak bağımlılıkları mock'lanır. Hedef: %80+ coverage.

## 11.2 Integration Test

testcontainers-go ile PostgreSQL, Redis ve Kafka containerları ayağa kaldırılır. Repository katmanı gerçek DB'ye karşı test edilir. Keycloak için test realm kullanılır.

## 11.3 E2E Test

docker-compose ile tüm servisler ayağa kaldırılır. Register → login → ürün ara → sipariş oluştur → saga tamamlan → bildirim gönder akışı uçtan uca test edilir.

---