



Mindset Institute

Microservices API Document

Muhammet Enes B y k

+90 553 867 1937

contact@enesbuyuk.com

**Bu dok ma Mindset Institute'un Backend Developer pozisyonu i in verilen durum  alışması kapsamında hazırlanmıştır.*

İçindekiler

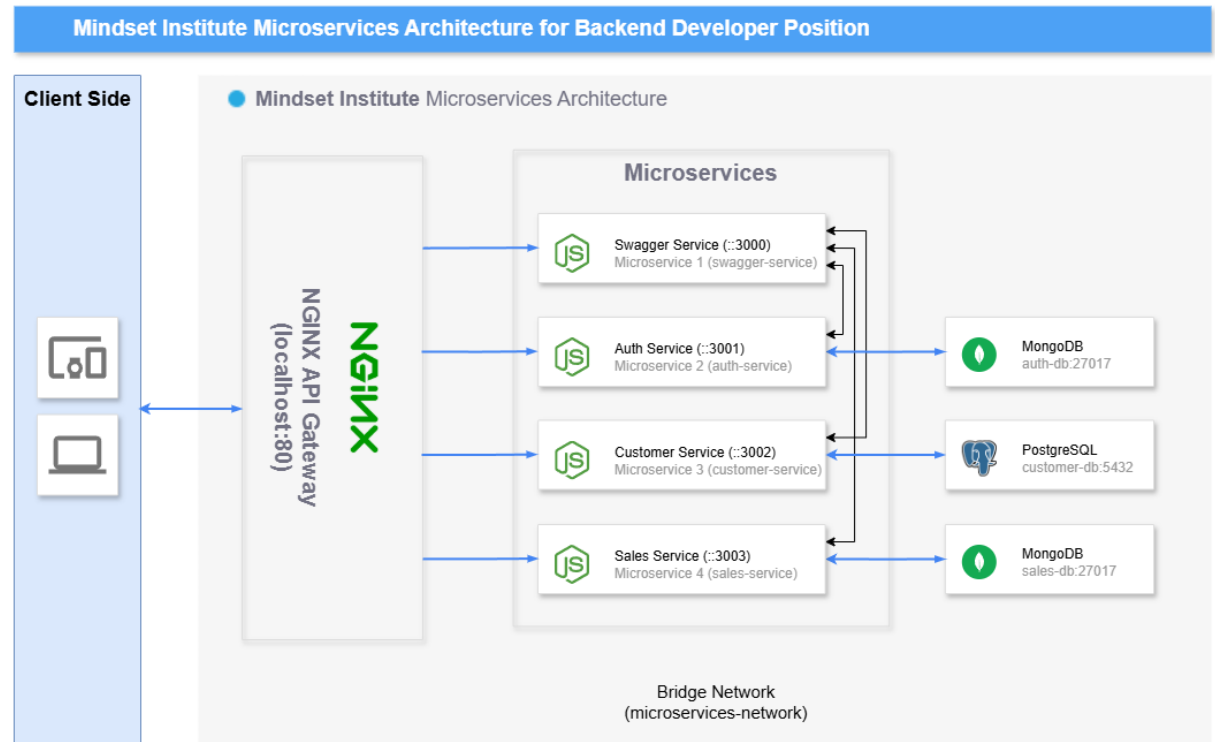
Mindset Institute Microservices API Document	1
Mikroservis Mimarisi	4
Mikroservis Mimarisi Diyagramı	4
Mimari Bileşenleri	5
1. API Gateway – Traefik	5
2. Mikroservisler	5
3. Veritabanları	5
4. Docker’ın Projedeki Yeri	6
5. Servisler Arası İletişim	6
Proje Kullanımı	6
Loglama Sistemi	7
Testler	7
Dokümantasyon	7
Swagger Dokümantasyonu	7
Github Repository Dokümanyastonu (README)	8
Videolu Dokümantasyon	Hata! Yer işareti tanımlanmamış.

Mikroservis Mimarisi

Bu projede mikroservis mimarisi benimsenmiş olup API Gateway olarak Traefik kullanılmıştır. Mikroservisler ise Node.js ile geliştirilmiş ve her biri bağımsız birer Docker container içerisinde çalışmaktadır. Veritabanları da yine bağımsız container'lar olarak yönetilmektedir.

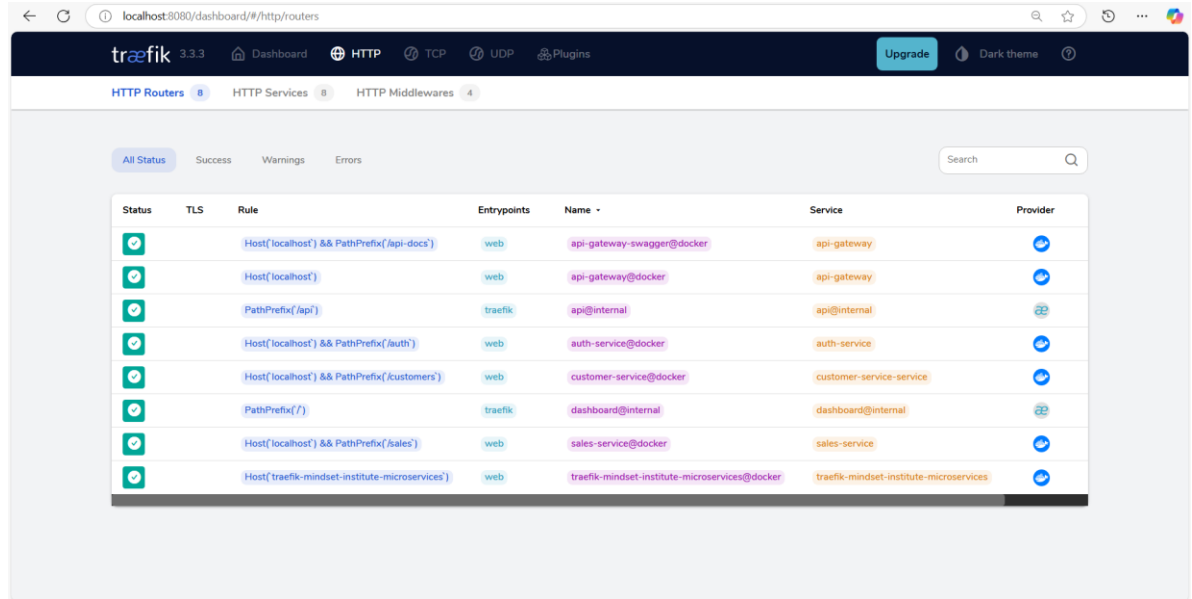
Mikroservis Mimarisi Diyagramı

Projenin daha kolay anlaşılabilmesi ve genel yapısının daha net görülebilmesi için aşağıdaki diyagram hazırlanmıştır. Bu diyagram projenin temel bileşenlerini ve bunların birbirleriyle olan ilişkilerini görselleştirerek kavramayı kolaylaştırmayı amaçlamaktadır.



Mimari Bileşenleri

1. API Gateway – Nginx



The screenshot shows the Traefik dashboard at localhost:8080. The top navigation bar includes 'Dashboard', 'HTTP', 'TCP', 'UDP', and 'Plugins'. Below the navigation bar, there are tabs for 'HTTP Routers' (8), 'HTTP Services' (8), and 'HTTP Middlewares' (4). The main content area displays a table of HTTP routers with columns: Status, TLS, Rule, Entrypoints, Name, Service, and Provider. All routers have a status of 'Success' and a provider of 'docker'.

Status	TLS	Rule	Entrypoints	Name	Service	Provider
Success		Host('localhost') && PathPrefix('/api-docs')	web	api-gateway-swagger@docker	api-gateway	docker
Success		Host('localhost')	web	api-gateway@docker	api-gateway	docker
Success		PathPrefix('/api')	traefik	api@internal	api@internal	traefik
Success		Host('localhost') && PathPrefix('/auth')	web	auth-service@docker	auth-service	docker
Success		Host('localhost') && PathPrefix('/customers')	web	customer-service@docker	customer-service-service	docker
Success		PathPrefix('/')	traefik	dashboard@internal	dashboard@internal	traefik
Success		Host('localhost') && PathPrefix('/sales')	web	sales-service@docker	sales-service	docker
Success		Host('traefik-mindset-institute-microservices')	web	traefik-mindset-institute-microservices@docker	traefik-mindset-institute-microservices	docker

API Gateway olarak **Nginx** tercih edilmiştir. **Nginx**; mikroservislerin trafik yönetimini sağlamak, yük dengeleme, SSL yönetimi ve servis keşfi gibi işlemleri yönetmek için güçlü bir çözümdür. Gateway gelen tüm istekleri uygun mikroservise yönlendirmekten sorumludur. JWT gereken mikroservislerde önce Auth Servisine giderek JWT validate işlemi gerçekleşir.

2. Mikroservisler

Projede üç farklı mikroservis bulunmaktadır:

a) Authentication Servisi:

- Kullanıcı kimlik doğrulama ve yetkilendirme işlemlerini yönetir.
- JSON Web Token (JWT) kullanarak güvenli erişim sağlar.
- MongoDB veritabanını kullanır.

b) Customer Servisi:

- Müşteri bilgilerini yönetir.
- Müşterilere ait detaylı bilgileri tutar ve günceller.
- PostgreSQL veritabanını kullanır.

c) Sales Servisi:

- Satış işlemlerini yönetir.
- Ürün satışı, sipariş yönetimi gibi fonksiyonlara sahiptir.
- MongoDB veritabanını kullanır.

3. Veritabanları

Her mikroservisin kendine ait bağımsız bir veritabanı bulunmaktadır:

- **Auth Servisi:** MongoDB
- **Sales Servisi:** MongoDB

- **Customer Servisi:** PostgreSQL

Bu veritabanları da **Docker container** olarak çalışmaktadır ve her servis yalnızca kendi veritabanına erişebilmektedir.

4. Docker'ın Projedeki Yeri

Proje tamamen Docker container tabanlı çalışmaktadır. Her bileşen için ayrı bir Docker container oluşturulmuştur:

- **Traefik (API Gateway)**
- **Auth Servisi (Node.js + MongoDB)**
- **Sales Servisi (Node.js + MongoDB)**
- **Customer Servisi (Node.js + PostgreSQL)**
- **MongoDB (Auth için ayrı, Sales için ayrı olmak üzere iki instance)**
- **PostgreSQL (Customer için)**

Tüm bileşenler **Docker Compose** ile yönetilmektedir ve servisler arasındaki iletişim izole bir Docker ağı (network) üzerinden sağlanmaktadır.

5. Servisler Arası İletişim

Servisler birbirleriyle iletişim kurarken doğrudan IP adresleri yerine **service discovery** özelliğinden yararlanır. Traefik gelen istekleri yönlendirerek servislere erişimi sağlar.

Örnek İletişim Akışı (Customer için)

1. Kullanıcı API Gateway'e giriş yapar ve kimlik doğrulama talebinde bulunur.
2. API Gateway isteği **Auth Servisine** yönlendirir.
3. Kimlik doğrulama başarılı olursa kullanıcı bir JWT token alır.
4. Müşteri bilgilerini almak için **Customer Servisine** sorgu gönderir.
5. Customer Servisi PostgreSQL veritabanından müşteri bilgilerini çekerek yanıt döner.

Proje Kullanımı

Projede mikroservis mimarisinde de belirtildiği üzere Docker Container'ları üzerinde deploy işlemi yapılmıştır. Her mikroservis, API Gateway ve veritabanı için ayrı ayrı Docker Container'ı bulunmakla beraber bunlar gerektiği şekilde güvenli iletişim kurmaktadır.

Projeyi kullanmaya başlamak için önce Github üzerinden projeyi kendi lokalimize indirmeliyiz. Git servisi bilgisayarınızda kurulu olmak şartıyla aşağıdaki komutla beraber projeyi kendi lokalize indirebilirsiniz.

```
git clone https://github.com/enesbuyuk/mindset-institute-microservices.git
```

İndirdikten sonra dilediğiniz konfigürasyonlara göre portları, alan adlarını düzenleyebilirsiniz. Bunu yönetmek için proje dizinine girerim

```
cd ./mindset-institute-microservices
```

Bu dizinde `.env` adlı dosyayı göreceksiniz. Bu dosyada tüm konfigürasyonlar açıkça belirtilmiş olup bu dosya üzerinden tüm konfigürasyonları dilediğinize göre ayarlayabilirsiniz.

Projede ana dizininde bir adet `docker-compose.yml` dosyası bulunmaktadır. Bu dosya sayesinde diğer mikroservisleri, veritabanları ve API Gateway’i ayağa kaldırabiliyoruz. Projeyi Github’tan kendi bilgisayarımıza çektikten sonra aşağıdaki komutla Docker Compose dosyası üzerinden tüm Docker Containerlarımızı ayağa kaldırabiliriz.

```
docker compose up -d
```

Loglama Sistemi

Testler

Proje birim testlerinde her mikroservis için ayrı ayrı testler oluşturulmuştur. Genel olarak controller ve models için oluşturulmuştur. Bu testler oluşturulurken Node.js’te “Jest” ve “Supertest” kütüphaneleri kullanılmıştır. Bu kütüphanelerin amacı;

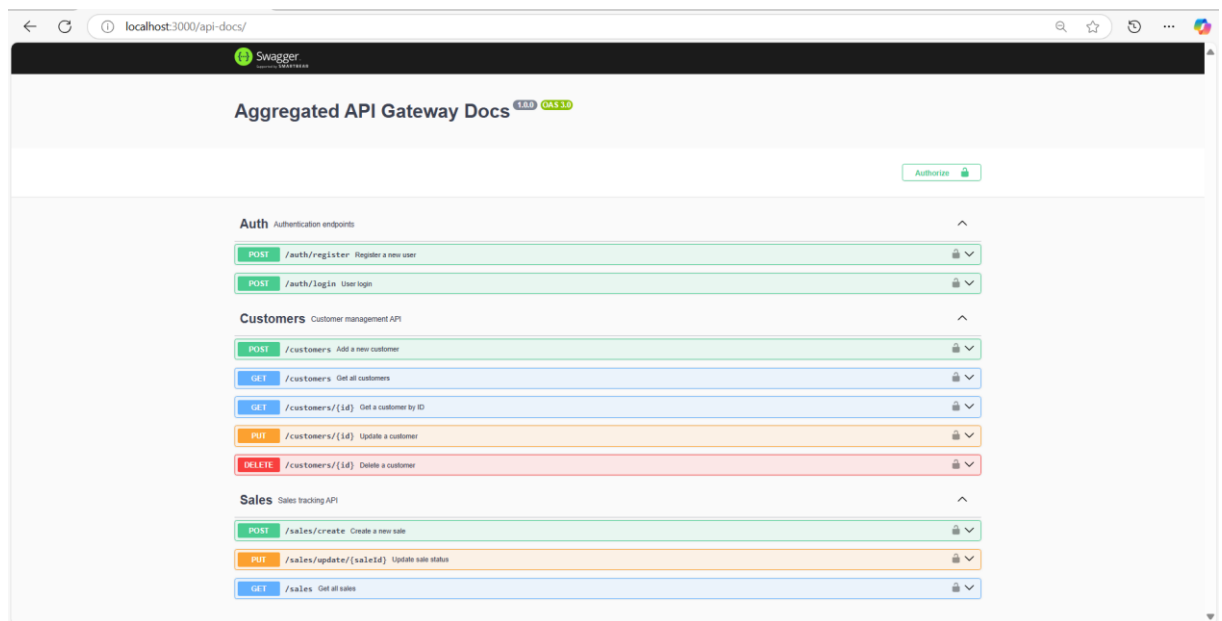
Jest: Testlerin yazılmasını ve çalıştırılmasını sağlamak için, testlerin geçip geçmediğini kontrol etmek için.

Supertest: API’nin doğru çalışıp çalışmadığını test etmek için HTTP istekleri gönderir.

Mikroservislerin “__test__” dizininde test dosyaları bulunmaktadır. “npm test” komutu ile çalıştırabilirsiniz.

Dokümantasyon

Swagger API Dokümantasyonu



Bu projede her mikroservis için ayrı ayrı **Swagger** dokümantasyonu hazırlanmıştır. Ve bunların hepsi API Gateway /api-docs adresinden aggregatö olarak sunulmuştur. Her mikroservisin /swagger.json dosyasından çekmektedir. Swagger sayesinde servislerin uç noktaları (endpoints), istek ve yanıt formatları, kullanılan HTTP metodları ve hata kodları gibi detaylar açık bir şekilde incelenebilir. Bu dokümantasyon hem geliştiricilerin hem de API tüketicilerinin sistemin nasıl çalıştığını anlamalarını kolaylaştırırken, test süreçlerini hızlandırarak geliştirme sürecini daha verimli hale getirir.

Swagger dokümantasyonu **localhost:3000/api-docs/** adresinden erişilebilen bir arayüz üzerinden sunulmaktadır. Bu projede **aggregated Swagger API** kullanılmış olup, tüm servislerin dokümantasyonu tek bir merkezi arayüzde birleştirilmiştir. Böylece, tüm uç noktalar tek bir yerden gözlemlenebilir ve test edilebilir.

Ayrıca Swagger arayüzüne **JWT kimlik doğrulama mekanizması** entegre edilmiştir. Bu sayede, yetkilendirme gerektiren uç noktalara erişmek için JWT token'ı Swagger üzerinden girerek test işlemleri gerçekleştirilebilir. Kullanıcıların, yetkilendirme gerektiren API çağrılarını kolayca doğrulayabilmesi, uygulamanın güvenliğini ve test edilebilirliğini artırmaktadır.

Bu yapı sayesinde Swagger sadece bir dokümantasyon aracı olmadan ziyade, API testlerini doğrudan gerçekleştirebileceğiniz interaktif bir ortam sunmaktadır.

Github Repository Dokümantasyonu (README)

Proje versiyon kontrol sistemi kapsamında GitHub'a eklenmiş olup tüm geliştirme süreçleri şeffaf bir şekilde takip edilebilmektedir. GitHub reposu üzerinden projeye ait kaynak kodlara, detaylı dokümantasyona ve README dosyasına erişebilirsiniz. README dosyasında, projenin lisans bilgileri, kullanım talimatları ve teknik detaylar ayrıntılı bir şekilde açıklanmıştır.

Herhangi bir katkıda bulunmak veya projeyi kendi sisteminize entegre etmek isterseniz GitHub reposunu inceleyerek gerekli bilgilere ulaşabilirsiniz.

Proje Github Reposuna <https://github.com/enesbuyuk/mindset-institute-microservices> adresinden ulaşabilirsiniz.