

**TUGAS PEMROGRAMAN WEB LANJUTAN (ARSITEKTUR
MONOLITH TO MICROSERVICE)**



Oleh:

Bagas Andreanto

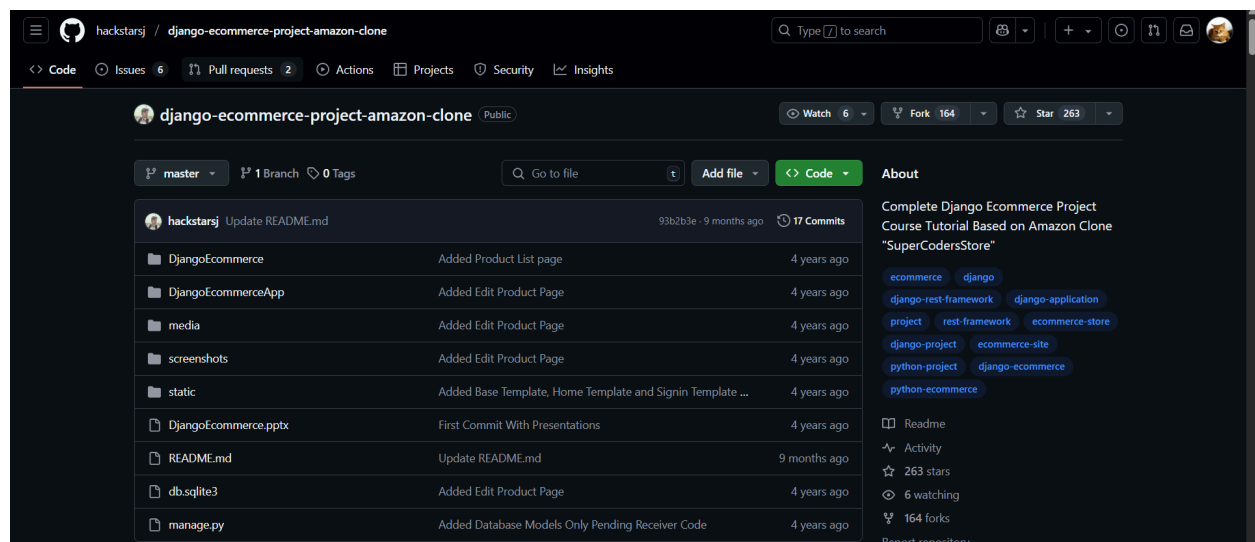
122140017

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
2025**

Penjelasan Repo

Link Sumber Repo Aplikasi Monolit: [Django E-commerce Project Amazon Clone](#)

Sumber aplikasi monolith dari proyek ini berasal dari Repo Github dengan username hackstarsj. Proyek ini bertujuan untuk merubah arsitektur aplikasi e-commerce yang awalnya menggunakan pendekatan monolitik menjadi microservice. Proyek ini didasarkan pada aplikasi Django yang meniru Amazon, yang mengelola berbagai aspek seperti manajemen pengguna, katalog produk, sistem pembayaran, dan manajemen pesanan. Langkah pertama adalah melakukan analisis terhadap struktur monolitik yang ada, kemudian melakukan pemisahan komponen menjadi layanan-layanan yang lebih modular.



Gambar 1. Tampilan repo sumber yang diambil

Struktur utama

django-ecommerce-project-amazon-clone-master/

- | — DjangoEcommerce/ # Folder utama Django (main project)
- | — DjangoEcommerceApp/ # Aplikasi utama
- | — media/ # Media storage
- | — screenshots/ # Folder berisi tangkapan layar
- | — static/ # Folder untuk static files
- | — db.sqlite3 # Database SQLite

— manage.py	# File utama untuk menjalankan Django
— DjangoEcommerce/settings.py	# Konfigurasi proyek
— DjangoEcommerce/urls.py	# Routing utama
— README.md	# Dokumentasi proyek
— DjangoEcommerce.pptx	# File presentasi proyek

Fitur Utama

1. Manajemen Pengguna (Login, registrasi, profil pengguna)
2. Katalog Produk (Manajemen produk, kategori, pencarian)
3. Sistem Pembayaran (Proses checkout dan transaksi)
4. Manajemen Pesanan (Riwayat pesanan dan status pesanan)
5. Admin Panel (Manajemen data melalui Django Admin)

BAB 2: Perencanaan Migrasi (Planning a Migration)

1. Identifikasi Komponen yang Akan Dipisah

Pada tahap pertama perencanaan migrasi, dilakukan identifikasi komponen yang akan dipisah dari aplikasi monolitik yang ada. Proses ini dimulai dengan menganalisis fungsionalitas aplikasi yang ada dan memetakan komponen-komponen tersebut menjadi layanan-layanan baru.

Layanan Baru	File/Folder Dalam Proyek
Layanan Autentikasi (Login, Register, User)	users/models.py, users/views.py, users/urls.py
Layanan Katalog Produk (Produk, Kategori)	products/models.py, products/views.py, products/urls.py
Layanan Transaksi & Pesanan (Checkout, Order, Payment)	orders/models.py, orders/views.py, orders/urls.py

2. Rencana Pemisahan API

Memisahkan setiap layanan menjadi API yang terpisah. Ini akan memungkinkan komunikasi antar-layanan melalui protokol HTTP/HTTPS. Untuk membangun API, saya akan menggunakan Django REST Framework (DRF) karena framework ini sudah terintegrasi dengan baik dengan Django dan mudah digunakan. Pemishan API ini digunakan untuk dapat memudahkan integrasi dengan sistem lain di masa depan dan memungkinkan pengembangan layanan secara independen.

Berikut adalah endpoint yang akan saya buat untuk masing-masing layanan:

- /api/auth/ → Login, register, profile
- /api/products/ → List produk, pencarian
- /api/orders/ → Checkout, histori transaksi.

3. Menentukan Teknologi yang Akan Digunakan

Layanan	Framework	Database
Autentikasi	Django + DRF	PostgreSQL
Katalog Produk	FastAPI / Django	MongoDB
Transaksi & Pesanan	Django + Celery	PostgreSQL

4. Evaluasi Risiko & Rencana Backup

Saya menyadari bahwa migrasi ini memiliki beberapa risiko, seperti downtime dan ketidakkonsistenan data. Oleh karena itu, saya merencanakan beberapa strategi untuk meminimalkan risiko:

Downtime Minim: Saya akan menggunakan pendekatan strangler pattern, yaitu memigrasi aplikasi secara bertahap. Ini akan meminimalkan downtime dan memungkinkan pengguna untuk tetap menggunakan aplikasi selama proses migrasi.

Data Consistency: Saya akan membuat data migration script untuk memindahkan data pengguna dan produk dari SQLite ke PostgreSQL dan MongoDB. Ini memastikan bahwa data tetap konsisten selama migrasi.

Keamanan API: Saya akan mengimplementasikan JWT Authentication untuk setiap layanan agar komunikasi antar-layanan aman dan terautentikasi.

Untuk mengoptimalkan migrasi dari monolith ke microservice menggunakan Docker untuk menyiapkan lingkungan pengembangan yang konsisten dan mudah di-deploy. Docker memastikan bahwa setiap layanan dapat berjalan di lingkungan yang terisolasi dan konsisten, mengurangi masalah yang disebabkan oleh perbedaan lingkungan pengembangan dan produksi.

BAB 3: Splitting The Monolith

Setelah melakukan ekstraksi dari aplikasi monolitik, saya mengatur direktori proyek menjadi beberapa layanan microservices yang terpisah. Struktur ini dirancang untuk memastikan setiap layanan memiliki tanggung jawab yang jelas dan dapat dikembangkan secara independen. Berikut adalah penjelasan tentang struktur direktori yang telah saya buat:

1. Layanan Autentikasi (auth_service/)

Layanan ini bertanggung jawab untuk mengelola autentikasi pengguna, termasuk proses login, registrasi, dan manajemen profil. Saya memindahkan file-file yang terkait dengan autentikasi dari direktori DjangoEcommerceApp/ ke dalam auth_service/

2. Layanan Katalog Produk (product_service/)

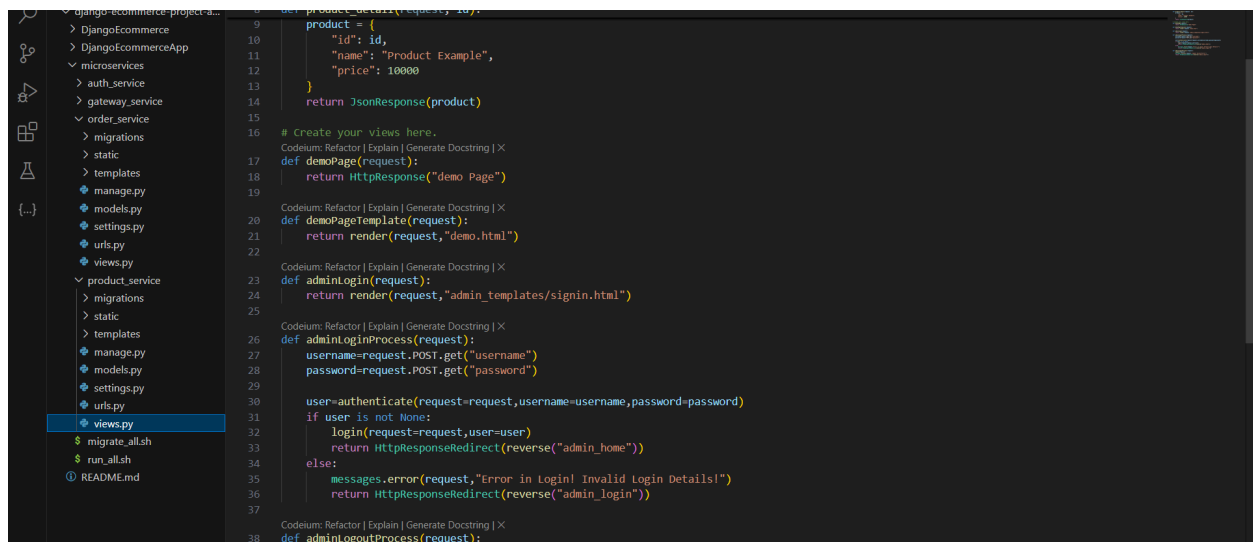
Layanan ini mengelola katalog produk, termasuk menampilkan daftar produk, kategori, dan fitur pencarian. Saya memindahkan file-file terkait produk dari DjangoEcommerceApp/ ke dalam product_service/. Struktur direktori ini mirip dengan auth_service/, dengan komponen-komponen seperti models.py, views.py, dan urls.py yang disesuaikan untuk kebutuhan katalog produk.

3. Layanan Pemesanan (order_service/)

Layanan ini bertanggung jawab untuk mengelola proses pemesanan, termasuk checkout, pembayaran, dan riwayat transaksi. Saya memindahkan file-file terkait pemesanan dari DjangoEcommerceApp/ ke dalam order_service/. Struktur direktori ini juga mengikuti pola yang sama, dengan penyesuaian untuk logika bisnis dan model yang terkait dengan pemesanan.

4. API Gateway (gateway_service/)

API Gateway berfungsi sebagai pintu masuk utama untuk semua permintaan dari klien ke microservices. Layanan ini akan mengarahkan permintaan ke layanan yang sesuai (auth_service, product_service, atau order_service) dan mengelola autentikasi, logging, serta load balancing. Saya menyiapkan direktori gateway_service/ dengan struktur yang serupa, tetapi fokusnya adalah pada pengelolaan routing dan integrasi antar-layanan.



Gambar 2. Tampilan direktori setelah di lakukan splitting

BAB 4: Dekomposisi Database (Decomposing the Database)

Dekomposisi Database untuk Arsitektur Microservices. Database yang sebelumnya terpusat dalam satu sistem monolitik dipisah sesuai dengan layanan masing-masing:

1. Analisis Database Monolitik

Semua tabel berada dalam satu database SQLite. Tabel utama yang ada adalah:

- a. users: Data pengguna
- b. products: Data produk
- c. orders: Data transaksi
- d. cart: Daftar belanja
- e. payment: Informasi pembayaran
- f. shipping: Informasi pengiriman

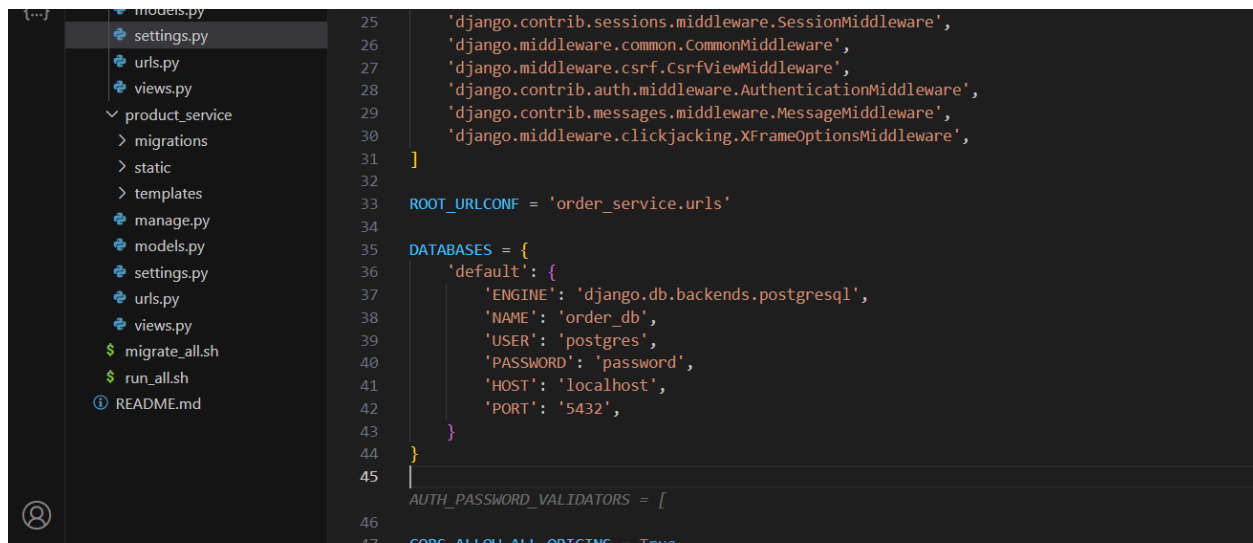
2. Dekomposisi Database ke Microservices

Setelah migrasi, masing-masing layanan memiliki database sendiri:

1. Auth Service: Menyimpan data pengguna
2. Product Service: Menyimpan data produk
3. Order Service: Menyimpan data pemesanan, pembayaran, dan pengiriman

3. Perubahan Skema Database

Contoh perubahan skema database untuk layanan Autentikasi:



```
25     'django.contrib.sessions.middleware.SessionMiddleware',
26     'django.middleware.common.CommonMiddleware',
27     'django.middleware.csrf.CsrfViewMiddleware',
28     'django.contrib.auth.middleware.AuthenticationMiddleware',
29     'django.contrib.messages.middleware.MessageMiddleware',
30     'django.middleware.clickjacking.XFrameOptionsMiddleware',
31 ]
32
33 ROOT_URLCONF = 'order_service.urls'
34
35 DATABASES = {
36     'default': {
37         'ENGINE': 'django.db.backends.postgresql',
38         'NAME': 'order_db',
39         'USER': 'postgres',
40         'PASSWORD': 'password',
41         'HOST': 'localhost',
42         'PORT': '5432',
43     }
44 }
45
46 AUTH_PASSWORD_VALIDATORS = [
47     'django.contrib.auth.password_validation.CommonPasswordValidator',
```

Gambar 3. Contoh dekomposing database untuk setiap file service.

Link Repo yang digunakan (Monolith):

<https://github.com/hackstarsj/django-ecommerce-project-amazon-clone>

Link Repo saya (Microservice):

https://github.com/bagas017/122140017_Bagas-Andreanto_MonolithToMicroservice