

LAPORAN KERJA PRAKTIK
PENGEMBANGAN APLIKASI BACK-END E-COMMERCE
MENGGUNAKAN REST API GOLANG UNTUK
OPTIMALISASI KINERJA SERVER
DI PT HACKTIVATE TEKNOLOGI
INDONESIA

Diajukan untuk memenuhi persyaratan kelulusan
Matakuliah TIF335 Kerja Praktik

oleh:

TAUFIK HIDAYAT / 301200032



PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BALE BANDUNG
2024

LEMBAR PENGESAHAN
PROGRAM STUDI TEKNIK INFORMATIKA
PENGEMBANGAN APLIKASI BACK-END E-COMMERCE
MENGGUNAKAN REST API GOLANG UNTUK
OPTIMALISASI KINERJA SERVER
DI PT HACKTIVATE TEKNOLOGI
INDONESIA

Oleh:

TAUFIK HIDAYAT / 301200032

Disetujui dan disahkan sebagai

LAPORAN KERJA PRAKTIK

Bandung, 3 Januari 2024

Koordinator Kerja Praktik Program Studi Teknik Informatika

Yusuf Muharram, S.Kom., M.Kom.

NIK: 04104820003

LEMBAR PENGESAHAN
PEMBIMBING KERJA PRAKTIK
PENGEMBANGAN APLIKASI BACK-END E-COMMERCE
MENGUNAKAN REST API GOLANG UNTUK
OPTIMALISASI KINERJA SERVER
DI PT HACKTIVATE TEKNOLOGI
INDONESIA

Oleh:

TAUFIK HIDAYAT / 301200032

Disetujui dan disahkan sebagai
LAPORAN KERJA PRAKTIK

Bandung, 18 Januari 2024

Pembimbing Kerja Praktik

Yudi Herdiana, S.T., M.T.

NIK: 04104808008

LEMBAR PENGESAHAN
PT HACKTIVATE TEKNOLOGI INDONESIA
PENGEMBANGAN APLIKASI BACK-END E-COMMERCE
MENGGUNAKAN REST API GOLANG UNTUK
OPTIMALISASI KINERJA SERVER
DI PT HACKTIVATE TEKNOLOGI
INDONESIA

Oleh:

TAUFIK HIDAYAT / 301200032

Disetujui dan disahkan sebagai

LAPORAN KERJA PRAKTIK

Bandung, 3 Januari 2024

Co-Founder PT Hacktivate Teknologi Indonesia



Riza Fahmi

ABSTRAKSI

Perkembangan pesat dalam teknologi informasi, khususnya di sektor e-commerce, menyebabkan masalah besar bagi kinerja server karena volume dan kompleksitas transaksi yang terus meningkat. PT Hacktivate Teknologi Indonesia menghadapi permasalahan serius terkait performa server dalam menjalankan aplikasi back-end E-Commerce. Fokus utama kerja praktik ini adalah mengembangkan aplikasi back-end E-Commerce menggunakan REST API Golang untuk meningkatkan kinerja server dan memastikan responsifitas yang optimal. Kerja praktik ini menerapkan Golang dan implementasi REST API pada aplikasi back-end E-Commerce sebagai solusi untuk mengatasi permasalahan performa server. Kecepatan eksekusi tinggi yang dimiliki Golang dan efisiensi penggunaan sumber daya menjadi alasan utama pemilihan teknologi tersebut. Sementara itu, penggunaan REST API memungkinkan interaksi yang ringan dan mudah diakses antara berbagai komponen sistem, mendukung tujuan peningkatan kinerja secara menyeluruh. Integrasi Golang dan REST API diharapkan dapat memberikan peningkatan dalam hal skalabilitas, kehandalan, dan performa server secara keseluruhan. Hasil kerja praktik menunjukkan bahwa pengembangan aplikasi back-end E-Commerce menggunakan REST API Golang memberikan dampak positif yang signifikan bagi PT Hacktivate Teknologi Indonesia. Responsifitas yang lebih baik, penanganan volume transaksi yang lebih efisien, dan penggunaan sumber daya yang lebih optimal menjadi hasil nyata dari implementasi ini. Solusi yang diusulkan tidak hanya memberikan manfaat secara langsung dalam meningkatkan kinerja server, tetapi juga memberikan dasar yang kokoh bagi PT Hacktivate Teknologi Indonesia untuk berkembang dan beradaptasi dengan tren industri E-Commerce yang dinamis. Sebagai akibatnya, pengembangan ini tidak hanya memenuhi tuntutan pasar yang semakin ketat, tetapi juga memberikan kontribusi positif terhadap efektivitas operasional perusahaan secara keseluruhan.

Kata Kunci: *Back-end, E-Commerce, Golang, REST API, Server*

KATA PENGANTAR

Dengan memanjatkan puji syukur kehadirat Allah SWT. Tuhan Yang Maha Pengasih dan Penyayang yang telah melimpahkan segala rahmat, hidayah, serta innayah-Nya atas terselesaikannya Kerja Praktik (KP) di PT. Hacktivate Teknologi Indonesia (Hacktiv8). Laporan disusun untuk memenuhi persyaratan Laporan Kerja Praktik di PT. Hacktivate Teknologi Indonesia dan Fakultas Teknologi Informasi, Program Studi Teknik Informatika, Universitas Bale Bandung. Tujuan dari laporan kerja praktik adalah untuk melaporkan semua kegiatan yang telah dilakukan selama dilaksanakannya program KP di PT. Hacktivate Teknologi Indonesia (Hacktiv8). Dalam penyusunan laporan ini, tentu tidak lepas dari pengarahan dan bimbingan dari berbagai pihak. Oleh karena itu, saya ingin mengungkapkan rasa hormat dan terima kasih kepada semua pihak yang telah membantu. Berikut adalah beberapa pihak yang terlibat dalam dilaksanakannya kegiatan KP ini:

1. Dr. Ir. H. Ibrahim Danuwikarsa, M.S. selaku Rektor Universitas Bale Bandung.
2. Yudi Herdiana, S.T., M.T. selaku Dekan Fakultas Teknologi Informasi dan Dosen Pembimbing Kerja Praktik.
3. Yusuf Muharam, S.Kom., M.Kom. selaku Ketua Program Studi Teknik Informatika.
4. Ronald Ishak selaku CEO PT. Hacktivate Teknologi Indonesia.
5. Riza Fahmi selaku Co-Founder PT. Hacktivate Teknologi Indonesia.
6. Airell Jordan selaku mentor Golang for Back End Programmer.
7. Seluruh pembimbing perusahaan PT Hacktivate Teknologi Indonesia yang telah membagikan ilmu dan pengetahuan dalam kegiatan Kerja Praktik ini.
8. Seluruh rekan di PT Hacktivate Teknologi Indonesia yang berperan aktif dalam pengerjaan Final Project.
9. Keluarga yang selalu memberi semangat dan dukungan.

Berkat kerjasama yang baik dari semua pihak yang telah disebutkan sebelumnya, penulis dapat menyelesaikan laporan kerja praktik ini dengan sebaik mungkin. Meskipun laporan ini belum sempurna, kritik dan saran dari para pembaca sangat diharapkan dan akan sangat bermanfaat untuk kemajuan penulis di masa depan. Sekali lagi, penulis mengucapkan terima kasih yang sebesar-besarnya. Semoga laporan ini memberikan manfaat bagi kita semua.

Bandung, 3 Januari 2024



Taufik Hidayat

301200032

DAFTAR ISI

BAB I PENDAHULUAN.....	1
I.1 Latar Belakang.....	1
I.2 Lingkup.....	2
I.3 Tujuan.....	3
BAB II LINGKUNGAN KERJA PRAKTIK.....	4
II.1 Struktur Organisasi.....	4
II.2 Lingkup Pekerjaan.....	6
II.3 Deskripsi Pekerjaan.....	8
II.4 Jadwal Kerja.....	8
BAB III TEORI PENUNJANG KERJA PRAKTIK.....	9
III.1 Teori Penunjang.....	9
III.2 Peralatan Pengembangan Aplikasi.....	16
BAB IV PELAKSANAAN KERJA PRAKTIK.....	38
IV.1 Input.....	38
IV.2 Proses.....	40
IV.2.1 Eksplorasi.....	41
IV.2.2 Perancangan Perangkat Lunak.....	41
IV.3 Pencapaian Hasil.....	54
BAB V PENUTUP.....	67
V.1 Kesimpulan dan Saran Mengenai Pelaksanaan.....	67
V.1.1 Kesimpulan Pelaksanaan Kerja Praktik.....	67
V.1.2 Saran Pelaksanaan KP.....	68
V.2 Kesimpulan dan Saran Mengenai Substansi.....	68
V.2.1 Kesimpulan.....	68
V.2.2 Saran.....	69

DAFTAR TABEL

Tabel III.1 Use Case Diagram.....	12
Tabel III.2 Class Diagram.....	13
Tabel III.3 Sequence Diagram.....	14
Tabel III.4 Activity Diagram.....	15
Tabel IV.1 Kebutuhan Perangkat Keras.....	39
Tabel IV.2 Minimum Kebutuhan Perangkat Keras.....	39
Tabel IV.3 Kebutuhan Perangkat Lunak.....	40
Tabel IV.4 Deskripsi Aktor.....	43
Tabel IV.5 Deskripsi Use Case.....	44

DAFTAR GAMBAR

Gambar II.1 Struktur Organisasi Hacktiv8.....	4
Gambar IV.1 Use Case Diagram.....	43
Gambar IV.2 Activity Diagram Login.....	45
Gambar IV.3 Activity Diagram Register.....	46
Gambar IV.4 Activity Diagram Topup.....	46
Gambar IV.5 Activity Diagram Tambah Kategori.....	47
Gambar IV.6 Activity Diagram Menampilkan Semua Kategori.....	47
Gambar IV.7 Activity Diagram Mengubah Kategori.....	48
Gambar IV.8 Activity Diagram Hapus Kategori.....	48
Gambar IV.9 Activity Diagram Tambah Produk.....	49
Gambar IV.10 Activity Diagram Menampilkan Semua Produk.....	49
Gambar IV.11 Activity Diagram Update Produk.....	50
Gambar IV.12 Activity Diagram Hapus Produk.....	50
Gambar IV.13 Activity Diagram Buat Transaksi Baru.....	51
Gambar IV.14 Activity Diagram List Semua Transaksi.....	52
Gambar IV.15 Activity Diagram List Transaksi Milik Pengguna.....	52
Gambar IV.16 Class Diagram.....	53
Gambar IV.17 Entity Relationship Diagram.....	54

BAB I

PENDAHULUAN

I.1 Latar Belakang

Perkembangan pesat dalam teknologi informasi, khususnya di sektor e-commerce, menyebabkan masalah besar bagi kinerja server karena volume dan kompleksitas transaksi yang terus meningkat. PT Hacktivate Teknologi Indonesia menghadapi permasalahan serius terkait performa server dalam menjalankan aplikasi back-end E-Commerce, memicu perlunya kerja praktik yang terfokus untuk mengatasi tantangan tersebut. Fokus utama kerja praktik ini adalah mengembangkan aplikasi back-end E-Commerce menggunakan REST API Golang untuk meningkatkan kinerja server dan memastikan responsifitas yang optimal. Kerja praktik ini menerapkan Golang dan implementasi REST API pada aplikasi back-end E-Commerce sebagai solusi untuk mengatasi permasalahan performa server. Kecepatan eksekusi tinggi yang dimiliki Golang dan efisiensi penggunaan sumber daya menjadi alasan utama pemilihan teknologi tersebut. Sementara itu, penggunaan REST API memungkinkan interaksi yang ringan dan mudah diakses antara berbagai komponen sistem, mendukung tujuan peningkatan kinerja secara menyeluruh. Integrasi Golang dan REST API diharapkan dapat memberikan peningkatan dalam hal skalabilitas, kehandalan, dan performa server secara keseluruhan.

PT Hacktivate Teknologi Indonesia, sebagai salah satu perusahaan yang bergerak di bidang teknologi dan edukasi, memiliki platform e-commerce yang digunakan untuk berbagai keperluan internal. Platform ini mengalami permasalahan serius terkait performa servernya, terutama dalam hal:

- **Kecepatan:** Server sering mengalami bottleneck dan latency yang tinggi, sehingga menyebabkan waktu respons yang lambat saat mengakses

platform e-commerce. Hal ini dapat mengganggu kelancaran operasional perusahaan dan menurunkan tingkat kepuasan pengguna.

- **Skalabilitas:** Platform e-commerce saat ini tidak mampu menangani lonjakan traffic yang tinggi, terutama pada saat periode promo atau event tertentu. Hal ini menyebabkan server down dan mengganggu akses pengguna.
- **Keandalan:** Server sering mengalami error dan crash, sehingga menyebabkan downtime platform e-commerce. Hal ini dapat mengakibatkan kerugian finansial bagi perusahaan dan mengganggu kepercayaan pengguna.

Permasalahan kinerja server ini dapat menghambat operasional perusahaan dan menurunkan daya saing di era digital yang serba cepat. Oleh karena itu, diperlukan solusi untuk meningkatkan kinerja server dan memastikan responsivitas yang optimal.

Pengembangan aplikasi back-end E-Commerce menggunakan REST API Golang diharapkan dapat menjadi solusi untuk mengatasi permasalahan kinerja server di PT Hacktivate Teknologi Indonesia. Golang adalah bahasa pemrograman yang terkenal dengan kecepatan eksekusi tinggi dan efisiensi penggunaan sumber daya. Hal ini dapat membantu meningkatkan kecepatan dan skalabilitas server.

I.2 Lingkup

Lingkup materi dalam kerja praktik yang dilaksanakan di PT Hacktivate Teknologi Indonesia adalah pengembangan aplikasi backend REST API untuk e-commerce yang mencakup beberapa komponen. Ruang lingkup tersebut meliputi

1. Merancang dan membangun aplikasi back-end E-Commerce menggunakan bahasa pemrograman Golang.
2. Menerapkan REST API untuk interaksi antar komponen sistem.
3. Membangun fitur-fitur utama E-Commerce, seperti:

- a. Manajemen pengguna
 - b. Manajemen kategori
 - c. Manajemen produk
 - d. Transaksi
- 4. Menjaga keamanan dan keandalan aplikasi
 - 5. Kerja praktik ini tidak mencakup:
 - a. Pengembangan aplikasi front-end E-Commerce
 - b. Integrasi dengan sistem pembayaran pihak ketiga
 - c. Implementasi sistem logistik dan pengiriman
 - d. Implementasi fitur-fitur E-Commerce yang kompleks

I.3 Tujuan

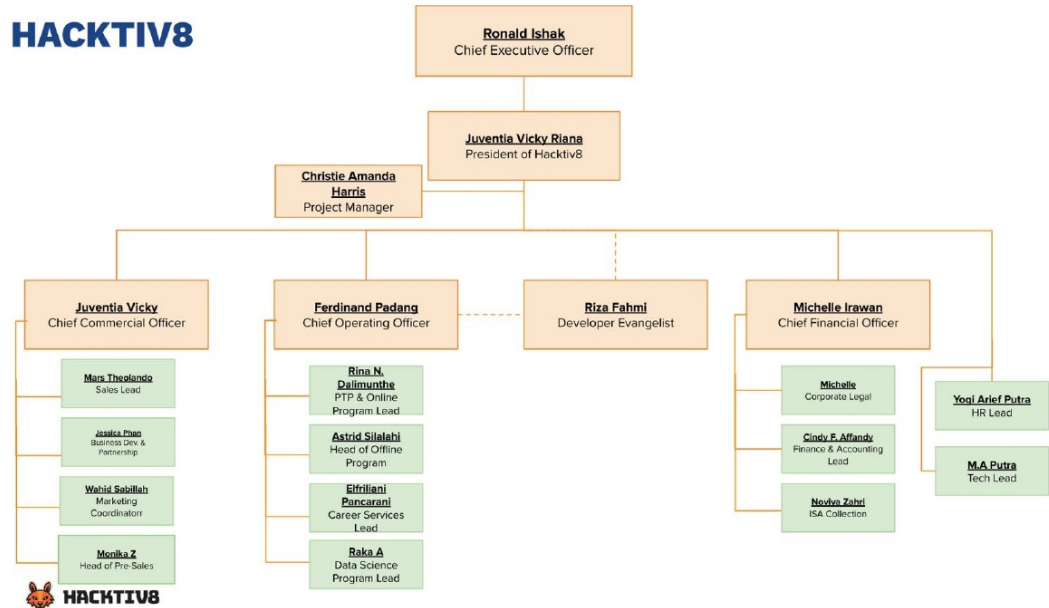
Tujuan dilaksakannya kerja praktik di PT Hacktivate Teknologi Indonesia ini adalah sebagai berikut:

- 1. Mengembangkan aplikasi back-end E-Commerce yang handal dan efisien
- 2. Mempelajari dan menerapkan teknologi Golang dan REST API
- 3. Meningkatkan kinerja server di PT Hacktivate Teknologi Indonesia

BAB II

LINGKUNGAN KERJA PRAKTIK

II.1 Struktur Organisasi



Gambar II.1 Struktur Organisasi Hacktiv8

1. Chief Executive Office
 - a. Mengambil keputusan strategis yang memengaruhi arah keseluruhan perusahaan.
 - b. Memimpin dan mengelola tim eksekutif.
 - c. Menetapkan visi, misi, dan tujuan jangka panjang perusahaan.
2. President of Hacktiv8
 - a. Memastikan implementasi strategi secara efektif di seluruh organisasi.
 - b. Memberikan kepemimpinan dan arahan kepada tim eksekutif dan seluruh organisasi.

- c. Mewakili perusahaan dalam hubungan dengan pemegang saham, investor, dan pihak eksternal lainnya.
3. Project Manager
- a. Mengembangkan rencana proyek yang komprehensif, termasuk penentuan sumber daya, anggaran, dan jadwal kerja.
 - b. Berkomunikasi secara efektif dengan anggota tim, pemangku kepentingan, dan pihak terkait lainnya.
 - c. Memantau kualitas pekerjaan yang dilakukan oleh tim proyek.
4. Developer Evangelist
- a. Mengembangkan konten edukatif seperti tutorial, panduan, dan materi presentasi untuk mendukung pemahaman dan adopsi produk.
 - b. Menjalin hubungan baik dengan komunitas pengembang melalui saluran online, seperti forum, media sosial, dan blog.
 - c. Membantu pengembang memahami fitur-fitur produk baru dan cara optimal menggunakannya dalam pengembangan aplikasi.
5. Chief Operating Officer
- a. Mengembangkan dan menerapkan kebijakan dan prosedur operasional untuk meningkatkan produktivitas dan kualitas.
 - b. Menilai dan mengelola risiko operasional yang mungkin mempengaruhi kinerja perusahaan.
 - c. Bekerja sama dengan tim manajemen untuk merancang dan menerapkan strategi perusahaan.
6. Chief Commercial Officer
- a. Merancang dan mengimplementasikan strategi penjualan dan pemasaran yang mendukung pertumbuhan bisnis.
 - b. Memastikan kepuasan pelanggan dengan memahami kebutuhan mereka dan menyediakan solusi yang tepat.

- c. Bertanggung jawab atas negosiasi kontrak dengan pelanggan, mitra bisnis, dan pihak ketiga.
7. Chief Financial Officer
- a. Bertanggung jawab atas perencanaan keuangan dan pengelolaan sumber daya finansial perusahaan.
 - b. Menganalisis risiko keuangan yang mungkin mempengaruhi perusahaan.
 - c. Memastikan perusahaan mematuhi semua regulasi keuangan dan hukum yang berlaku.
8. HR Lead
- a. Mengelola siklus hidup karyawan, termasuk perekrutan, orientasi, pelatihan, dan pengembangan.
 - b. Membangun dan memelihara budaya perusahaan yang mendukung nilai-nilai dan tujuan organisasi.
 - c. Menyediakan saluran untuk umpan balik karyawan dan mengimplementasikan perubahan berdasarkan informasi tersebut.
9. Tech Lead
- a. Memimpin tim pengembang dalam merancang, mengembangkan, dan mengimplementasikan perangkat lunak.
 - b. Memberikan arahan teknis kepada anggota tim, memastikan pemahaman yang jelas tentang tugas dan tanggung jawab mereka.
 - c. Memastikan bahwa produk perangkat lunak memenuhi standar kualitas yang ditetapkan.

II.2 Lingkup Pekerjaan

Dalam program KP di PT. Hacktivate Teknologi Indonesia (Hacktiv8), saya adalah seorang mentee yang mengambil posisi sebagai *programmer* Golang,

dengan pendampingan seorang mentor. Program ini berlangsung dari tanggal 16 Februari 2023 hingga 30 Juni 2023 selama 4 bulan. Saya dibimbing oleh seorang fasilitator yang berperan sebagai mentor, yang menjelaskan materi dan memberikan panduan selama proses pembelajaran. Tanggung jawab saya dalam pekerjaan ini adalah mempelajari dan menyelesaikan tugas-tugas yang diberikan oleh Hacktiv8. Selain itu, saya juga mengerjakan proyek akhir yang diberikan oleh mitra dengan tepat waktu.

1. Lecture Session

Lecture Session merupakan sesi di mana peserta didik belajar dan mendapatkan pemahaman teori lengkap tentang Golang For Backend Programmer sesuai dengan kurikulum yang telah ditentukan. Kegiatan pembelajaran ini melibatkan mentor dan semua peserta dalam satu kelas. Pengajaran ini dilakukan secara daring melalui platform Google Meet, dan terdapat sebelas materi pembelajaran yang dapat diakses melalui LMS (Sistem Manajemen Pembelajaran) Kode.id.

2. Practice Session

Sesi ini merupakan sesi praktik di mana mentor akan memberikan sejumlah tugas latihan kepada seluruh mahasiswa setelah selesai dengan materi. Hal ini dilakukan sebagai bagian dari ulasan materi yang telah diajarkan, sehingga mahasiswa dapat mempertahankan pemahaman terhadap materi-materi sebelumnya.

3. Final Project Session

Sesi ini didedikasikan khusus untuk mengerjakan proyek akhir. Dalam sesi ini, mentor memberikan tugas kepada seluruh mahasiswa dalam bentuk kerja kelompok untuk menyelesaikan empat (4) proyek akhir yang berbeda pada akhir sesi.

4. Mentoring Session

Mentoring Session adalah sesi tanya jawab mengenai proyek akhir maupun hal-hal di luar proyek akhir yang dilakukan secara daring antara mentor dan mahasiswa pada waktu yang telah disepakati di luar jadwal pembelajaran. *Mentoring* ini merupakan kegiatan yang opsional, di mana setiap mahasiswa akan mendapatkan 4 kesempatan *mentoring* selama program berlangsung. Platform Calendly digunakan sebagai alat pemesanan jadwal, sementara Google Meet digunakan sebagai media pelaksanaan sesi *mentoring*.

II.3 Deskripsi Pekerjaan

Pada proyek akhir terakhir, penulis mengembangkan aplikasi bernama Toko Belanja. Aplikasi ini melibatkan seorang admin yang memiliki banyak pelanggan. Hanya admin yang memiliki hak untuk melakukan operasi CRUD terhadap kategori dan produk. Pelanggan hanya dapat melakukan pembelian produk dan melihat data transaksi pembelian mereka. Pelanggan juga dapat melakukan top-up saldo untuk menambah saldo yang digunakan dalam pembelian produk. Selain itu, semua aplikasi proyek akhir harus di-deploy ke *cloud application platform*.

II.4 Jadwal Kerja

Penulis telah menerima pelatihan intensif selama 8 minggu (12 sesi, 3 jam setiap sesi) dari instruktur profesional yang ahli di bidangnya. Pelatihan ini menggunakan metode kombinasi *live lecture* dan *self-paced learning*. *Live lecture* dilakukan secara sinkronus dalam dua pertemuan pada hari Rabu dan Kamis, dimulai pukul 19:00 hingga 22:00. Setelah mempelajari teori dasar, peserta akan diberikan tugas untuk membuat proyek akhir dan diberikan waktu 8 minggu untuk menyelesaikannya secara berkelompok. Peserta akan diminta untuk membuat REST API Sederhana (CRUD) menggunakan bahasa pemrograman Go dengan konsep Todo's.

BAB III

TEORI PENUNJANG KERJA PRAKTIK

III.1 Teori Penunjang

Selama menjalani kerja praktik, peserta kerja praktik memanfaatkan pengetahuan yang diperoleh selama masa perkuliahan, kerja praktik, dan pembelajaran otodidak sebagai dasar teoretis dalam pembuatan aplikasi *Back-end REST API* E-Commerce. Beberapa pengetahuan dan teori yang digunakan mencakup:

1. Teori Algoritma dan Pemrograman (TIF301)

Teori yang terkait dengan pengantar atau pemahaman dasar mengenai pemrograman melibatkan langkah-langkah dalam membuat program, prinsip-prinsip yang perlu diperhatikan, serta hal-hal yang sebaiknya dihindari ketika merancang sebuah aplikasi atau program. Materi ini diajarkan dalam mata kuliah TIF301 algoritma dan pemrograman, yang mencakup pemahaman tentang bagaimana algoritma pemrograman beroperasi.

2. Teori Basis Data (TIF310)

Teori Basis Data terkait dengan pengaturan, pemodelan, dan manajemen data dalam suatu aplikasi. Dalam konteks aplikasi E-Commerce, teori ini memberikan bantuan dalam merancang struktur basis data yang efisien untuk menyimpan informasi mengenai pengguna, kategori, barang, dan transaksi. Tujuannya adalah untuk memastikan penyimpanan data yang teratur, mudah diakses, dan memenuhi standar integritas. Materi ini diperoleh melalui mata kuliah TIF310 Basis Data.

3. Teori Pemrograman Internet (TIF319)

Teori Pemrograman Internet mencakup prinsip-prinsip yang berkaitan dengan pengembangan aplikasi web, terutama di bagian back-end dan

pengelolaan REST API. Dalam konteks pengembangan aplikasi E-Commerce, pemahaman teori ini menjadi krusial untuk merancang dan melaksanakan fungsionalitas di sisi server. Backend pada perangkat lunak merujuk pada bagian dari perangkat lunak yang bertanggung jawab untuk menangani sisi server, manajemen database, dan logika aplikasi yang tidak terlihat oleh pengguna secara langsung. REST API (Representational State Transfer Application Programming Interface) adalah sebuah antarmuka yang digunakan oleh dua sistem komputer untuk bertukar informasi secara aman melalui internet. RESTful API merupakan implementasi dari API yang mematuhi batasan arsitektur REST dan berinteraksi dengan layanan web RESTful. API ini bersifat stateless dan menggunakan HTTP/HTTPS untuk transmisi data

4. Teori Manajemen Proyek Perangkat Lunak (TIF318)

Teori Manajemen Proyek Perangkat Lunak mencakup prinsip-prinsip, metodologi, dan perangkat untuk mengelola proyek dengan efektif. Dalam konteks kerja praktik pembuatan aplikasi E-Commerce, teori ini berperan dalam perencanaan, pengorganisasian, dan pengendalian proyek pengembangan aplikasi. Pemahaman terhadap tahapan pengembangan, alokasi sumber daya, dan manajemen risiko akan membantu menjaga agar proyek tetap terjadwal dan mencapai tujuan yang telah ditetapkan. Materi ini diperoleh melalui mata kuliah TIF318 Manajemen Proyek Perangkat Lunak.

5. Teori E-Commerce (TIF328)

Teori E-Commerce mencakup berbagai aspek termasuk definisi dan perkembangan dasar E-Commerce, model bisnis seperti B2C dan B2B, infrastruktur teknologi dan sistem pembayaran elektronik, keamanan transaksi online, strategi pemasaran dan promosi digital, aspek hukum dan etika, manajemen logistik serta pengiriman, analisis data untuk pengambilan keputusan, dan pemahaman terkini mengenai tren dan

inovasi dalam industri perdagangan elektronik. Dalam konteks kerja praktik pembuatan aplikasi E-Commerce, teori ini berperan dalam membimbing proses desain, pengembangan, dan implementasi platform perdagangan elektronik tersebut, memberikan landasan konseptual bagi pemahaman aspek-aspek kunci seperti model bisnis, keamanan transaksi, dan pengalaman pengguna, sehingga mahasiswa dapat mengaplikasikan pengetahuan teoritisnya secara langsung dalam situasi dunia nyata.

6. Teori Rekayasa Perangkat Lunak (TIF316)

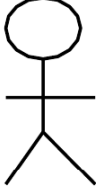
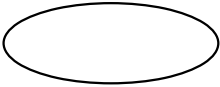


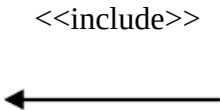
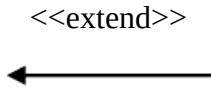
a. Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan bahasa grafis yang digunakan untuk menggambarkan, merancang, dan mendokumentasikan sistem perangkat lunak. UML menyajikan kumpulan notasi dan teknik yang dapat digunakan untuk mengilustrasikan berbagai aspek sistem, termasuk struktur, fungsi, dan interaksi antar komponen sistem. Penggunaan UML berfokus pada pemodelan sistem perangkat lunak pada tahap perencanaan dan pengembangan. Melalui UML, para pengembang memiliki kemampuan untuk membuat diagram yang menjelaskan struktur kelas, relasi antar kelas, alur proses, dan tingkah laku sistem. Selain itu, UML juga mendukung pemodelan aspek non-fungsional, seperti keamanan dan skalabilitas.

1. Use Case Diagram

Use Case Diagram adalah tipe diagram dalam pemodelan perangkat lunak yang digunakan untuk mengilustrasikan fungsionalitas yang diinginkan dari suatu sistem dari sudut pandang pengguna atau aktor yang berinteraksi dengan sistem tersebut.



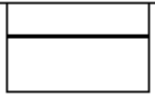

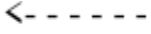
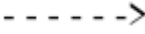
Tabel III.1 Use Case Diagram


No	Simbol	Keterangan
1		Aktor: entitas eksternal yang berinteraksi dengan sistem yang sedang dianalisis.
2		Use case: menggambarkan serangkaian langkah – langkah atau alur yang menggambarkan bagaimana system berperilaku dalam situasi tertentu.
3		Asosiasi: menghubungkan aktor dengan use case di dalam diagram.
4		Generalisasi: hubungan hierarkis antara dua atau lebih use case atau aktor dalam diagram.
5		Include: Menunjukkan bahwa sebuah use case termasuk fungsi atau tindakan dari use case lainnya.
6		Extend: Menunjukkan bahwa sebuah use case dapat memperluas atau mengubah fungsionalitas use case lainnya secara opsional.

2. Class Diagram

Diagram ini menunjukkan kelas-kelas yang terdapat dalam sistem, interaksi antara kelas-kelas tersebut, atribut-atribut (data) yang dimiliki oleh kelas, dan metode-metode (fungsi) yang dimiliki oleh kelas.

Tabel III.2 Class Diagram


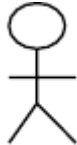


No	Simbol	Nama	Keterangan
1		Generalization	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor)
2		Navy Association	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek
3		Class	Himpunan dari objek-objek yang Berbagi atribut serta operasi yang sama.
4		Collaboration	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu Aktor.
5		Realization	Operasi yang benar-benar dilakukan oleh suatu objek.
6		Dependency	Hubungan antara dua kelas di mana satu kelas tergantung pada kelas lain dalam konteks




			tertentu.
7		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.

3. Sequence Diagram

Sequence Diagram adalah tipe diagram yang digunakan dalam pemodelan perangkat lunak untuk mengilustrasikan urutan interaksi antara objek-objek dalam suatu skenario atau proses khusus.

Tabel III.3 Sequence Diagram



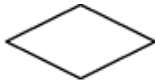
No	Simbol	Nama	Keterangan
1		Object	Komponen utama Sequence Diagram
2		Actor	Menggambarkan orang yang sedang berinteraksi dengan sistem
3		Entity Class	Menggambarkan hubungan kegiatan yang dilakukan
4		Boundary Class	Menggambarkan sebuah penggambaran dari form



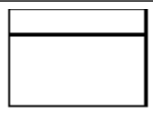
5		Control Class	Menggambarkan penghubung antara boundary dengan tabel
6		Life Line	Menggambarkan tempat mulai dan berakhirnya sebuah message
7		Message	Menggambarkan pengiriman pesan

4. Activity Diagram

Diagram aktivitas adalah jenis diagram dalam pemodelan perangkat lunak yang dipakai untuk memvisualisasikan alur atau urutan aktivitas dalam suatu proses atau sistem.

Tabel III.4 Activity Diagram

No	Simbol	Nama	Keterangan
1		Status awal	Sebuah diagram aktivitas memiliki sebuah status awal.
2		Aktivitas	Aktivitas yang dilakukan sistem aktivitas biasanya diawali dengan kata kerja.
3		Cabangan/ Decision	Percabangan dimana ada pilihan aktivitas yang lebih dari Satu

4		Penggabungan/ Join	Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu
5		Status akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
6		Swimlane	Swimlane memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

III.2 Peralatan Pengembangan Aplikasi

A. Software

Alat atau perangkat yang digunakan untuk mengembangkan aplikasi *back-end REST API E-Commerce* melibatkan:

1. VSCodium

VSCodium adalah distribusi biner berlisensi bebas yang digerakkan oleh komunitas dari editor Microsoft VSCode. VSCodium merupakan build dari source code VSCode yang disederhanakan dari telemetri dan kode ekstra. Ini membuat VSCodium sepenuhnya open source dan didistribusikan sesuai dengan lisensi MIT.

VSCodium adalah distribusi biner berlisensi bebas yang digerakkan oleh komunitas dari editor Microsoft VSCode. VSCodium merupakan build dari *source code* VSCode yang disederhanakan dari telemetri dan kode ekstra. Ini membuat VSCodium sepenuhnya open source dan didistribusikan sesuai dengan lisensi MIT.

Fungsi yang dihadirkan dalam VSCodium mirip dengan VSCode, hanya saja tanpa telemetri. Hal ini memungkinkan pengguna untuk menggunakan VSCodium tanpa harus mengunduh source code *proprietary* dari Microsoft.

Fungsi dari VSCodium:

- a. Pengeditan Kode: Dalam VSCodium, pengguna dapat membuat, mengedit, dan memformat kode dengan dukungan komprehensif untuk berbagai bahasa pemrograman. Fasilitas ini mencakup penyorotan sintaksis, penyelesaian otomatis kode, format kode, dan pemeriksaan kesalahan.
- b. Integrasi dengan Perangkat Pengembangan: VSCodium dapat diintegrasikan dengan berbagai perangkat pengembangan dan sistem manajemen versi, seperti Git. Fitur ini memungkinkan pengguna untuk memantau perubahan kode, mengelola repositori, dan berkolaborasi dengan pengembang lainnya.
- c. Ekosistem Ekstensi yang Luas: VSCodium memiliki dukungan untuk ekstensi yang dapat diunduh dan diinstal melalui toko ekstensi. Ekstensi ini mampu menambahkan fitur serta fungsionalitas tambahan, termasuk peningkatan dalam bahasa pemrograman, pengembangan aplikasi web, dan integrasi dengan alat lainnya.
- d. Debugging: Dalam VSCodium, terdapat fitur debuggin yang terintegrasi, memfasilitasi pengguna untuk menemukan dan memperbaiki bug dalam kode mereka dengan efisien. Fasilitas ini mencakup tinjauan variabel, penanda tempat berhenti eksekusi (breakpoint), dan pelaksanaan kode langkah demi langkah.
- e. Terminal yang Terintegrasi: Dalam VSCodium, terdapat terminal yang terintegrasi, memungkinkan pengguna untuk mengeksekusi

perintah shell atau alat pengembangan lainnya langsung dari dalam editor. Fasilitas ini memberikan kemampuan kepada pengguna untuk dengan mudah menjalankan dan menguji kode mereka.

- f. Meningkatkan Efisiensi Kerja: VSCodium menyajikan berbagai fitur untuk meningkatkan efisiensi pengguna, termasuk penyelesaian otomatis kode, pemformatan kode, navigasi yang cepat, serta dukungan untuk tugas-tugas umum dalam pengembangan perangkat lunak.
- g. Penyesuaian Tampilan dan Tema: Dalam VSCodium, pengguna memiliki kemampuan untuk menyesuaikan tampilan dan tema editor sesuai dengan keinginan mereka. Pengguna dapat mengubah tata letak, warna, ikon, dan tema untuk menciptakan pengalaman pengeditan kode yang sesuai dengan gaya dan preferensi pribadi mereka.

2. Golang

Golang, atau disebut juga Go, adalah bahasa pemrograman yang dikembangkan oleh Google. Go didesain untuk menjadi bahasa yang sederhana, efisien, dan mudah dipahami, sehingga dapat digunakan untuk pengembangan perangkat lunak yang bersifat skalabel dan efisien secara performa. Beberapa karakteristik utama dari Golang meliputi:

a. Kesederhanaan

Go didesain agar mudah dibaca dan ditulis. Struktur bahasa yang sederhana membantu dalam pengembangan perangkat lunak dengan kode yang lebih bersih dan lebih mudah dipahami.

b. Efisiensi

Golang memprioritaskan performa dan efisiensi eksekusi. Ini membuatnya cocok untuk pengembangan aplikasi yang

membutuhkan kinerja tinggi, seperti aplikasi jaringan dan *backend server*.

c. Concurrency

Go memiliki dukungan bawaan untuk pemrograman konkuren (*concurrent programming*) melalui goroutine. Goroutine adalah unit ringan yang dapat dieksekusi secara konkuren, memungkinkan pengembang untuk menangani ribuan tugas secara bersamaan.

d. Penanganan Kesalahan (*Error Handling*)

Golang menggunakan pendekatan yang jelas dan ekspresif dalam menangani kesalahan, yang membantu pengembang dalam mengidentifikasi dan menanggapi masalah dengan lebih baik.

e. Garbage Collection

Go memiliki manajemen memori otomatis dengan pengumpulan sampah (*garbage collection*). Hal ini mengurangi beban kerja pengembang dalam memantau dan mengelola alokasi memori secara manual.

f. Lintas Platform

Golang dirancang untuk mendukung lintas platform dengan dukungan yang baik untuk kompilasi kode menjadi bahasa mesin. Ini membuat aplikasi yang dikembangkan dengan Go dapat berjalan pada berbagai sistem operasi tanpa perlu kompilasi ulang.

g. Dukungan Modul dan Paket

Golang mendukung pembuatan modul dan manajemen paket yang efisien, memudahkan pengembang untuk mengorganisir dan menyusun kode mereka.

Golang telah digunakan secara luas dalam berbagai proyek, termasuk pengembangan perangkat lunak *server*, perangkat lunak infrastruktur, dan aplikasi web. Kecepatan eksekusi, kesederhanaan, dan

kemampuan konkurensi membuat Golang menjadi pilihan yang populer untuk proyek-proyek dengan skala besar dan tinggi performa.

Golang juga memiliki berbagai fungsi dan dapat digunakan dalam berbagai konteks pengembangan perangkat lunak. Berikut adalah beberapa fungsi utama dari Golang:

a. Pemrograman Umum

Golang dapat digunakan untuk pengembangan perangkat lunak umum seperti pembuatan aplikasi *desktop*, aplikasi konsol, dan alat utilitas. Bahasa ini menyediakan fitur-fitur dasar pemrograman yang diperlukan dalam pengembangan perangkat lunak pada umumnya.

b. Pengembangan Web

Golang memiliki paket standar yang mendukung pengembangan aplikasi web. Beberapa kerangka kerja web yang populer seperti Gin, Echo, dan Fiber telah dibangun dengan menggunakan Golang. Golang cocok untuk pengembangan *backend* web yang skalabel dan efisien.

c. Server dan Infrastruktur

Golang sangat cocok untuk membangun *server* dan infrastruktur perangkat lunak. Kinerja tinggi dan kemampuan konkurensi yang baik membuatnya ideal untuk mengembangkan *server* yang menangani banyak permintaan secara bersamaan. Banyak proyek besar, termasuk proyek-proyek *open-source* dan *cloud services*, menggunakan Golang untuk bagian *backend* mereka.

d. Pemrograman Jaringan

Golang memiliki dukungan yang kuat untuk pemrograman jaringan. Pustaka standar seperti "net" dan "http" memudahkan

pengembang untuk membuat aplikasi jaringan seperti *server* TCP/IP, HTTP *server*, dan protokol komunikasi khusus.

e. Pengembangan Mikrokontroler dan IoT

Karena ukurannya yang kecil dan efisiensinya, Golang dapat digunakan untuk pengembangan perangkat lunak pada mikrokontroler dan perangkat *Internet of Things* (IoT).

f. Alat Pengujian dan Pemeliharaan Kode

Golang menyertakan alat bawaan seperti "go test" untuk pengujian unit dan "go fmt" untuk pemformatan kode. Ini membuatnya mudah digunakan untuk pengujian otomatis dan pemeliharaan kode yang konsisten.

g. Pemrograman Konkuren (Concurrent Programming)

Salah satu kekuatan utama Golang adalah kemampuannya untuk menangani konkurensi dengan mudah melalui goroutine. Ini memungkinkan pengembangan aplikasi yang dapat menangani banyak tugas secara bersamaan.

h. Pemrosesan Paralel

Golang menyediakan mekanisme paralelisme yang mudah digunakan dengan goroutine dan channel, memungkinkan eksekusi kode secara paralel untuk meningkatkan performa.

3. Gin Gonic

Gin Gonic adalah sebuah *framework web* yang ditulis dalam bahasa pemrograman Go (Golang). *Framework* ini dirancang untuk mempermudah pengembangan aplikasi web dengan kecepatan tinggi dan performa yang baik. Berikut adalah beberapa poin yang menjelaskan definisi dari *framework* Gin Gonic:

a. Ringan dan Cepat

Salah satu ciri khas Gin Gonic adalah ringannya dan kecepatannya. Gin dirancang untuk menjadi *framework* yang minim dan fokus pada kecepatan eksekusi. Hal ini membuatnya menjadi pilihan yang baik untuk pengembangan aplikasi web yang memerlukan responsivitas tinggi.

b. Dukungan HTTP

Gin Gonic menyediakan dukungan lengkap untuk protokol HTTP. Ini mencakup *routing*, *handling middleware*, dan berbagai fitur HTTP lainnya yang diperlukan untuk pengembangan aplikasi web.

c. Routing yang Efisien

Gin menggunakan sistem *routing* yang efisien dan mudah dipahami. Pengembang dapat dengan jelas menentukan rute-rute HTTP dan mengaitkannya dengan fungsi atau *handler* tertentu.

d. Middleware

Gin mendukung konsep *middleware*, yang memungkinkan pengembang untuk menambahkan fungsionalitas tambahan ke dalam proses penanganan permintaan HTTP. *Middleware* dapat digunakan, misalnya, untuk otentikasi, *logging*, atau manipulasi *request* dan *response*.

e. Penanganan *Request* dan *Response*

Gin menyediakan cara yang jelas dan efisien untuk menangani permintaan HTTP dan menghasilkan respons. Pengembang dapat dengan mudah mengakses parameter permintaan, membaca data dari *body*, dan mengirim respons ke klien.

f. Rendering HTML dan Template

Gin mendukung *rendering* HTML dan menggunakan template untuk membangun tampilan dinamis. Pengembang dapat

mengintegrasikan dengan mudah dengan *template engine* seperti HTML/template di Go.

g. Dukungan JSON

Gin dirancang dengan dukungan bawaan untuk bekerja dengan data dalam format JSON. Ini membuatnya sangat cocok untuk pengembangan aplikasi web yang menggunakan pertukaran data dalam format JSON.

h. Dukungan Grup Router

Gin memberikan fitur untuk membuat grup router, suatu fitur yang mempermudah developer dalam mengelompokkan rute-rute yang berkaitan dan menerapkan *middleware* secara serentak pada seluruh grup tersebut. Dengan adanya opsi ini, developer dapat dengan lebih efisien mengorganisir dan mengelola rute-rute yang terkait, serta mengimplementasikan *middleware* secara kolektif pada keseluruhan grup router yang telah dibentuk.

i. Logging dan Debugging

Gin menyediakan fasilitas *logging* yang baik untuk membantu pengembang dalam melacak aktivitas aplikasi. Selain itu, ada dukungan yang baik untuk *debugging*.

j. Modularitas

Gin didesain dengan prinsip modularitas, memungkinkan pengembang untuk menggunakan bagian-bagian tertentu dari *framework* tanpa harus mengadopsi seluruh rangkaian fitur.

4. PostgreSQL

PostgreSQL adalah sebuah sistem manajemen basis data relasional (RDBMS) yang bersifat *open-source*, yang berarti kode sumbernya dapat diakses dan dimodifikasi oleh siapa saja. PostgreSQL dibangun berdasarkan model konsep objek-relasional, yang memungkinkan

penggunanya untuk menyimpan dan mengelola data menggunakan konsep tabel dan relasi, sekaligus mendukung fitur-fitur tambahan yang terkait dengan objek.

Berikut adalah beberapa poin kunci yang menjelaskan PostgreSQL:

a. Sistem Manajemen Basis Data Relasional (RDBMS)

PostgreSQL adalah sistem manajemen basis data relasional, yang berarti data disimpan dalam tabel yang terorganisir dengan baik dan hubungan antar tabel dapat didefinisikan. Ini mengikuti model relasional yang dikembangkan oleh Edgar F. Codd.

b. Open Source

PostgreSQL bersifat *open-source* atau sumber terbuka, yang berarti dapat diunduh, digunakan, dan dimodifikasi secara gratis oleh siapa saja. Komunitas global yang aktif terlibat dalam pengembangan dan pemeliharaan PostgreSQL.

c. Objek-Relasional

PostgreSQL tidak hanya mendukung fitur-fitur basis data relasional standar, tetapi juga menyediakan elemen-elemen objek-objek tambahan seperti tipe data kompleks, fungsi, dan prosedur penyimpanan yang dapat digunakan untuk mengelola data dengan lebih canggih.

d. Transaksi dan Konsistensi

PostgreSQL mendukung transaksi yang dapat diulang (ACID *properties: Atomicity, Consistency, Isolation, Durability*). Ini berarti sistem menjamin bahwa operasi-operasi yang melibatkan data dalam transaksi akan entah dilakukan sepenuhnya atau tidak dilakukan sama sekali.

e. Ekstensibilitas

PostgreSQL dapat diperluas melalui penggunaan ekstensi dan fungsi-fungsi tambahan. Pengguna dapat menambahkan fungsionalitas baru atau memodifikasi perilaku PostgreSQL sesuai dengan kebutuhan mereka.

f. Berbagai Jenis Tipe Data

PostgreSQL menyediakan berbagai jenis tipe data yang melibatkan teks, angka, tanggal, *array*, JSON, dan tipe data khusus seperti geometri untuk dukungan data spasial.

g. Keamanan

PostgreSQL memiliki sistem keamanan yang kuat, termasuk pengelolaan hak akses, otentikasi, dan enkripsi data. Ini membantu melindungi data dari akses yang tidak sah atau modifikasi yang tidak sah.

h. Kinerja dan Optimasi Query

PostgreSQL dilengkapi dengan optimasi *query* yang canggih untuk meningkatkan kinerja eksekusi *query*. Pengguna dapat menggunakan indeks, mengoptimalkan struktur tabel, dan menggunakan fitur lainnya untuk meningkatkan efisiensi sistem.

i. Dukungan untuk Pemrograman dan Bahasa

PostgreSQL mendukung berbagai bahasa pemrograman seperti Python, Java, C/C++, dan lainnya. Ini memungkinkan pengguna untuk membuat fungsi-fungsi yang kompleks di dalam *database* menggunakan bahasa yang mereka pilih.

j. Komunitas dan Dokumentasi yang Kuat

PostgreSQL memiliki komunitas pengguna yang aktif dan mendukung, serta dokumentasi yang luas. Ini memudahkan

pengguna untuk mendapatkan bantuan dan sumber daya yang diperlukan.

PostgreSQL digunakan secara luas di berbagai proyek dan organisasi, termasuk aplikasi bisnis, situs *web*, dan proyek-proyek pengembangan perangkat lunak.

5. GORM

GORM (*Go Object Relational Mapper*) adalah sebuah ORM (*Object Relational Mapper*) yang digunakan dalam bahasa pemrograman Go (Golang). ORM adalah suatu teknologi yang memungkinkan pengembang untuk berinteraksi dengan basis data relasional menggunakan objek-objek dan bahasa pemrograman, daripada menggunakan kueri SQL secara langsung. GORM menyederhanakan dan mempercepat pengembangan aplikasi dengan menyediakan antarmuka tingkat tinggi untuk berinteraksi dengan basis data.

Berikut adalah beberapa poin kunci yang menjelaskan GORM:

a. Object Relational Mapper (ORM)

GORM adalah ORM yang memungkinkan pengembang Go untuk bekerja dengan basis data relasional tanpa harus menulis kueri SQL secara eksplisit. GORM melakukan pemetaan antara struktur data dalam kode Go dengan tabel dalam basis data relasional.

b. Dukungan untuk Berbagai Jenis Basis Data

GORM memberikan dukungan yang luas terhadap berbagai jenis basis data relasional, termasuk namun tidak terbatas pada MySQL, PostgreSQL, SQLite, dan SQL Server. Fitur ini memberikan tingkat fleksibilitas yang tinggi kepada para pengembang, memungkinkan mereka untuk memilih dan menggunakan basis data sesuai dengan persyaratan dan kebutuhan spesifik dari proyek yang sedang mereka kerjakan. Dengan adanya dukungan untuk

berbagai jenis basis data, GORM menciptakan suatu lingkungan yang memungkinkan adaptasi yang optimal terhadap beragam konteks pengembangan.

c. Modeling Data

GORM memungkinkan definisi model data dengan menggunakan struktur Go. Setiap model biasanya mewakili tabel dalam basis data, dan setiap properti (*field*) dalam model mewakili kolom dalam tabel.

Contoh:

```
type User struct {  
    ID          uint    `gorm:"primaryKey"`  
    Username    string  `gorm:"unique;not null"`  
    Email       string  `gorm:"unique;not null"`  
    Password    string  `gorm:"not null"`  
}
```

d. CRUD Operations

GORM menyediakan metode CRUD (*Create, Read, Update, Delete*) yang mudah digunakan untuk memanipulasi data dalam basis data. Ini termasuk pembuatan data baru, pembacaan data, pembaruan data, dan penghapusan data.

Contoh:

```
// Create  
db.Create(&user)  
  
// Read  
db.First(&user, 1) // Membaca user dengan ID 1
```

```
// Update
db.Model(&user).Update("Email", "new-
email@example.com")

// Delete
db.Delete(&user)
```

e. Hubungan (Relationships)

GORM mendukung pemodelan hubungan antar tabel, seperti hubungan satu-ke-banyak, banyak-ke-banyak, dan sebagainya. Ini memungkinkan pengembang untuk merepresentasikan struktur data kompleks.

f. Migrasi Basis Data

GORM menyediakan alat untuk melakukan migrasi basis data, yang memudahkan dalam mengelola evolusi skema basis data seiring waktu.

g. Kustomisasi dan Ekstensibilitas

GORM memberikan opsi untuk mengkonfigurasi dan menyesuaikan berbagai aspek perilaku ORM sesuai kebutuhan proyek.

h. Transaksi

GORM mendukung transaksi, yang memungkinkan pengembang untuk mengelompokkan beberapa operasi basis data ke dalam transaksi tunggal.

GORM merupakan pilihan yang populer di komunitas Go karena kemudahan penggunaan dan fiturnya yang kaya.

6. Postman

Postman adalah sebuah platform kolaboratif yang memungkinkan pengembang perangkat lunak untuk merancang, menguji, dan mendokumentasikan API (*Application Programming Interface*). Postman dirancang untuk menyederhanakan pengelolaan API dan memfasilitasi pengembangan aplikasi yang berinteraksi dengan berbagai layanan *web* dan RESTful API.

Berikut adalah beberapa poin kunci yang menjelaskan Postman:

a. Kolaborasi dan Kerja Tim

Postman memungkinkan tim pengembang untuk bekerja sama secara efektif dalam pengembangan dan pengujian API. Tim dapat berbagi koleksi API, lingkungan (*environment*), dan skrip pengujian.

b. Desain API

Postman menyediakan alat untuk merancang API dengan membuat dan menyusun permintaan HTTP, menentukan *endpoint*, *header*, dan *payload*. Pengembang dapat dengan mudah mengkonfigurasi permintaan untuk metode HTTP seperti GET, POST, PUT, dan DELETE.

c. Uji dan Debug API

Postman memberikan kesempatan bagi pengembang untuk melakukan uji coba pada API melalui proses pengiriman permintaan HTTP dan respons yang diterima. Di samping itu, Postman menyediakan berbagai fitur yang sangat berguna, seperti kemampuan untuk mengelola lingkungan (*environment*). Fitur ini memungkinkan pengembang untuk menyesuaikan pengaturan lingkungan yang berbeda-beda, termasuk konfigurasi khusus untuk pengujian dan pengembangan. Dengan adanya fleksibilitas ini,

Postman menjadi alat yang sangat berguna dalam menunjang proses uji coba dan pengembangan API dengan berbagai kebutuhan yang beragam.

d. Automasi Pengujian

Postman mendukung penulisan skrip pengujian menggunakan JavaScript. Pengembang dapat membuat skenario pengujian yang otomatis untuk memastikan bahwa API berfungsi sesuai yang diharapkan.

e. Pemantauan dan Pelaporan

Postman menyediakan fitur untuk memantau kinerja API dan melacak statistik penggunaan. Pelaporan ini membantu pengembang dan tim untuk memahami performa API dan mendeteksi masalah potensial.

f. Dokumentasi API

Postman memungkinkan pengembang untuk membuat dokumentasi API yang kaya dengan menggunakan koleksi Postman. Dokumentasi ini dapat dibagikan dengan tim internal atau eksternal untuk memudahkan integrasi dan penggunaan API.

g. Ekosistem dan Integrasi

Postman dapat diintegrasikan dengan berbagai alat pengembangan dan kolaborasi lainnya, seperti Git, Jenkins, dan platform pengelolaan API lainnya. Ini memungkinkan pengembang untuk mengintegrasikan Postman ke dalam alur kerja pengembangan yang ada.

h. Keamanan

Postman memberikan dukungan yang kuat terhadap keamanan API melalui berbagai opsi yang tersedia, seperti fitur otentikasi dan pengelolaan kunci API. Dengan menggunakan Postman, para

pengembang memiliki fleksibilitas untuk mengonfigurasi alat ini agar dapat berinteraksi dengan berbagai API yang mewajibkan proses autentikasi. Dengan adanya fitur-fitur tersebut, Postman memastikan bahwa proses pengembangan aplikasi melalui API menjadi lebih aman dan terkendali.

i. Monitoring dan Pengelolaan Environments

Postman memungkinkan pengembang untuk mengelola lingkungan (*environment*) yang berbeda, seperti lingkungan produksi dan pengembangan. Ini mempermudah pengujian dan pengembangan dalam konteks yang sesuai.

j. Versi Cloud dan Desktop

Postman tersedia sebagai aplikasi desktop dan versi *cloud* (Postman Cloud) yang menyediakan penyimpanan data dan sinkronisasi antar perangkat.

Dengan fitur-fitur yang komprehensif dan antarmuka pengguna yang ramah, Postman telah menjadi alat yang sangat populer di kalangan pengembang API untuk memfasilitasi pengembangan, pengujian, dan dokumentasi API secara efisien.

7. Git

Git adalah sebuah sistem kontrol versi yang sangat populer dan didistribusikan. Diciptakan oleh Linus Torvalds pada tahun 2005, Git dirancang untuk mendukung pengembangan perangkat lunak dengan memungkinkan tim pengembang untuk mengelola dan melacak perubahan dalam kode sumber proyek perangkat lunak.

Berikut adalah beberapa poin kunci yang menjelaskan Git:

a. Sistem Kontrol Versi (VCS)

Git adalah sistem kontrol versi yang membantu tim pengembang dalam melacak perubahan-perubahan pada kode sumber proyek

perangkat lunak. Hal ini memungkinkan untuk melihat sejarah perubahan, mengembalikan ke versi sebelumnya, dan bekerja secara kolaboratif pada proyek.

b. Distribusi

Git adalah sistem kontrol versi yang didistribusikan, artinya setiap anggota tim memiliki salinan lengkap dari seluruh repositori proyek secara lokal. Ini memungkinkan pengembang untuk bekerja secara mandiri dan menggabungkan perubahan mereka nanti.

c. Cabang (Branching)

Salah satu fitur utama Git adalah kemampuan untuk membuat cabang (*branch*). Cabang memungkinkan pengembang untuk bekerja pada fitur atau perbaikan bug secara terisolasi tanpa mempengaruhi kode di cabang utama (biasanya disebut "master" atau "main").

d. Fusional (Merging)

Dalam Git, terdapat fitur untuk melakukan penggabungan (*merge*) cabang-cabang yang telah selesai ke dalam cabang utama. Proses ini dirancang untuk menyederhanakan penggabungan kode yang telah mengalami pengembangan terpisah, dengan memungkinkan pengembang untuk mengintegrasikan perubahan-perubahan tersebut ke dalam satu kesatuan. Dengan adanya kemampuan untuk menggabungkan cabang-cabang yang telah diselesaikan ke dalam cabang utama, Git memberikan cara yang efisien dan terstruktur bagi tim pengembang untuk mengelola proyek mereka secara kolaboratif.

e. Repositori

Repositori Git adalah tempat penyimpanan yang menyimpan sejarah perubahan, berbagai versi kode, dan informasi lainnya

terkait proyek. Repositori dapat berada secara lokal atau di *hosting server* seperti GitHub, GitLab, atau Bitbucket.

f. Commit

Commit adalah tindakan untuk menyimpan perubahan pada kode ke dalam repositori. Setiap *commit* memiliki pesan yang menjelaskan perubahan yang dilakukan, memudahkan untuk melacak dan memahami evolusi kode.

g. Indeks (Staging Area)

Git menggunakan konsep indeks atau *staging area*, yang memungkinkan pengembang untuk memilih perubahan mana yang akan di-*commit*. Ini memberikan fleksibilitas dan kontrol tambahan sebelum melakukan *commit*.

h. Rantai Blok (Commit Hash)

Setiap kali terjadi *commit* di Git, sistem menciptakan rantai blok dengan nilai hash yang unik. Fungsi utama dari nilai hash ini bukan hanya untuk memastikan keintegritasan dan keamanan repositori, tetapi juga untuk menyederhanakan proses referensi terhadap versi-versi spesifik proyek..

i. Pengembangan Terdistribusi

Setiap pengembang dapat bekerja secara terdistribusi di repositori lokal, dan kemudian menyatukan perubahan dengan repositori pusat atau antar sesama pengembang.

j. Ekosistem Luas dan Hosting

Git memiliki ekosistem yang sangat luas, dengan banyak layanan hosting seperti GitHub, GitLab, Bitbucket, dan lainnya yang menyediakan infrastruktur untuk menyimpan dan berbagi repositori Git.

Git telah menjadi standar *de facto* dalam pengembangan perangkat lunak dan digunakan secara luas di berbagai industri. Kemampuannya yang kuat dalam pengelolaan versi dan kolaborasi membuatnya menjadi alat yang esensial dalam pengembangan perangkat lunak modern.

8. GitHub

GitHub merupakan sebuah platform kolaboratif berbasis *web* yang secara luas dimanfaatkan untuk mendukung kegiatan pengembangan perangkat lunak. Fungsinya melibatkan penyediaan layanan manajemen repositori Git, yang menjadi wahana bagi pengembang untuk bekerja secara bersama-sama dalam proyek-proyek perangkat lunak. Melalui GitHub, tim pengembang dapat menjalankan kegiatan kolaboratif dengan lebih terstruktur dan terorganisir. Sejumlah komponen kunci yang memperkaya konsep GitHub mencakup:

a. Git

Git adalah sistem kontrol versi terdistribusi yang digunakan untuk melacak perubahan dalam kode sumber selama pengembangan perangkat lunak. Setiap perubahan atau revisi disebut "*commit*," dan Git memungkinkan pengembang untuk menyimpan, membagikan, dan mengelola riwayat revisi ini.

b. Repository

Repository adalah tempat penyimpanan untuk proyek perangkat lunak. Ini bisa berisi semua *file*, dokumen, dan sumber daya lain yang diperlukan untuk mengembangkan dan menyimpan proyek. Repositori di GitHub dapat bersifat publik (dapat diakses oleh siapa saja) atau pribadi (hanya dapat diakses oleh tim yang diundang).

c. Branch

Branch adalah cabang atau salinan independen dari repositori yang memungkinkan pengembang untuk bekerja pada fitur atau perbaikan tanpa mempengaruhi kode di cabang utama (biasanya disebut "*master*" atau "*main*"). Setelah pengembangan selesai, perubahan dapat digabungkan kembali ke cabang utama.

d. Pull Request

Pull Request (PR) adalah permintaan untuk menggabungkan perubahan dari satu cabang ke cabang lainnya. Ini memungkinkan pengembang untuk memeriksa, mengomentari, dan meninjau perubahan sebelum disatukan ke dalam repositori utama.

e. Collaborators

Collaborators adalah orang atau tim yang diizinkan untuk berkontribusi pada repositori tertentu. Mereka memiliki hak akses tertentu, tergantung pada peran mereka (seperti *read-only*, *write*, atau *admin*).

f. Issues dan Projects

GitHub menyediakan fitur *Issues* untuk melacak tugas, perbaikan bug, dan diskusi terkait proyek. *Projects* memungkinkan pengembang untuk mengorganisir dan mengelola tugas dalam tata letak papan kerja (*kanban*) yang dapat disesuaikan.

g. Wiki

Wiki adalah bagian dari repositori yang dapat digunakan untuk menyimpan dokumentasi proyek. Ini membantu dalam menyimpan informasi yang relevan dan dapat diakses oleh semua kontributor.

GitHub telah menjadi platform yang sangat populer di kalangan pengembang perangkat lunak karena menyediakan alat kolaborasi yang kuat, mendukung proyek *open source*, dan menyediakan infrastruktur

yang efisien untuk pengembangan perangkat lunak berskala besar atau kecil.

B. Hardware

Dalam merancang aplikasi *back-end REST API E-Commerce*, terdapat beberapa perangkat keras yang bisa dipilih, yaitu:

1. Server

Untuk menjalankan aplikasi *back-end* dan menyimpan data terkait pelanggan, kategori, barang, dan transaksi, diperlukan suatu server. Jenis *server* ini bisa berupa komputer fisik yang berdiri sendiri atau mesin virtual yang berada di lingkungan *cloud computing*. Pilihan antara komputer fisik atau mesin virtual ini bergantung pada skala dan kebutuhan spesifik dari aplikasi yang sedang dikembangkan. Adanya fleksibilitas ini memungkinkan para pengembang untuk menyesuaikan infrastruktur *server* sesuai dengan tingkat kompleksitas dan volume kerja yang dihadapi oleh aplikasi mereka.

2. Jaringan dan Router

Dibutuhkan koneksi jaringan yang dapat diandalkan untuk menghubungkan pengguna dengan server. Fungsi router adalah untuk mengarahkan aliran lalu lintas jaringan antara server dan pengguna.

3. Storage

Dibutuhkan ruang penyimpanan untuk menyimpan data aplikasi seperti basis data dan file pengguna. Hal ini dapat dilaksanakan dengan menggunakan *hard disk drive* (HDD) atau *solid-state drive* (SSD) pada *server*.

4. Switch dan Kabel Jaringan

Dipergunakan untuk menyatukan *server* dengan perangkat jaringan lainnya seperti *router*, komputer pengguna, dan perangkat klien.

5. Komputer dan Laptop Pengguna

Pengguna aplikasi dapat mengaksesnya melalui perangkat seperti komputer atau laptop yang terhubung ke *internet*. Komputer ini harus memenuhi persyaratan minimum sistem agar dapat menjalankan aplikasi *back-end* dengan optimal.

6. Perangkat Masukan/Keluaran

Pengguna akan memanfaatkan *keyboard*, *mouse*, monitor, dan perangkat masukan/keluaran lainnya saat mereka mengakses serta menggunakan aplikasi back-end.

BAB IV

PELAKSANAAN KERJA PRAKTIK

IV.1 Input

Secara menyeluruh, pengetahuan teoritis yang diperoleh selama perkuliahan, dan kerja praktik memberikan kontribusi berharga dalam pelaksanaan kerja praktik. Dasar teori tersebut memiliki peran krusial dalam memahami teknologi yang baru.

Pengembangan aplikasi *back-end REST API E-Commerce* ini melibatkan beberapa inputan pengolahan data, yang termasuk tetapi tidak terbatas pada:

a. Endpoint Pengelolaan Pelanggan

Endpoint API ini digunakan untuk melakukan proses registrasi pelanggan baru, proses login ke dalam sistem, dan juga proses topup saldo.

b. Endpoint Pengelolaan Produk

Endpoint API ini digunakan untuk menambah produk baru, melihat daftar produk, mengubah data produk, dan menghapus produk.

c. Endpoint Pengelolaan Kategori Produk

Endpoint API ini digunakan untuk menambah kategori baru, melihat daftar kategori, mengubah data kategori, dan menghapus kategori.

d. Endpoint Pembuatan Transaksi Baru

Endpoint API ini digunakan untuk membuat transaksi baru, melihat daftar transaksi yang dimiliki seorang pelanggan, dan melihat daftar transaksi yang dimiliki semua pelanggan.

1. Kebutuhan Perangkat Keras

Dalam perancangan aplikasi *back-end REST API E-Commerce* di PT Hacktivate Teknologi Indonesia, penyusun menggunakan laptop, oleh karena itu diperlukan sebuah komputer dengan spesifikasi berikut:

Tabel IV.1 Kebutuhan Perangkat Keras

No	Item	Spesifikasi
1	Processor	AMD Ryzen 3 5300U
2	GPU	AMD Vega 6
3	SSD	512 GB
4	RAM	12 GB
5	Monitor	1920x1080@60fps

2. Minimum Kebutuhan Perangkat Keras

Spesifikasi minimum komputer yang diperlukan untuk menjalankan aplikasi *back-end REST API E-Commerce* adalah:

Tabel IV.2 Minimum Kebutuhan Perangkat Keras

No	Item	Spesifikasi
1	Processor	Architecture x86, x86-64, ARM, dan lainnya. Dengan kecepatan clock 1 GHz atau lebih.
2	GPU	Opsional
3	Storage	256 GB
4	RAM	512 MB
5	Monitor	1366x768@60fps

3. Kebutuhan Perangkat Lunak

Perangkat lunak merujuk pada serangkaian instruksi atau program komputer yang bertugas mengendalikan dan menjalankan berbagai kegiatan pada komputer. Berikut adalah daftar perangkat lunak yang diterapkan dalam pengembangan aplikasi *back-end REST API E-Commerce* ini:

Tabel IV.3 Kebutuhan Perangkat Lunak

No	Item	Spesifikasi
1	Sistem Operasi	Linux 6.1, Windows 10 64 bit, macOS 10.12
2	Bahasa Pemrograman	Golang 1.20
3	DBMS	PostgreSQL 15
4	Web Browser	LibreWolf, Firefox
5	Code Editor	VSCodium

IV.2 Proses

Pelaksanaan kerja praktik ini melibatkan pembuatan aplikasi *back-end REST API E-Commerce*. Proses kerja praktik ini dapat dibagi menjadi beberapa tahap, termasuk eksplorasi, pengembangan aplikasi, dan pelaporan hasil kerja praktik. Selama tahap eksplorasi, tugas mencakup pengenalan lingkungan serta penyesuaian terhadap pekerjaan di PT Hacktivate Teknologi Indonesia. Tahap berikutnya melibatkan pengembangan aplikasi *back-end REST API E-Commerce*, diikuti oleh tahap pelaporan hasil kerja praktik pada tahap ketiga. Tahap ini dilakukan oleh peserta selama kerja praktik berlangsung.

IV.2.1 Eksplorasi

Proses eksplorasi dimulai dengan menyelidiki metode yang akan diterapkan dalam merancang aplikasi *back-end REST API E-Commerce*. Untuk mendukung penerapan metodologi *Agile*, pengetahuan tentang pemodelan menggunakan *Unified Modeling Language* (UML) juga menjadi kebutuhan esensial. Oleh karena itu, fokus juga diberikan pada pemahaman yang lebih mendalam terkait pemodelan menggunakan UML.

Langkah-langkah yang terlibat dalam tahap eksplorasi proyek pengembangan aplikasi *back-end REST API E-Commerce* adalah sebagai berikut:

- a. Eksplorasi Kebutuhan Pengguna
- b. Eksplorasi Teknologi dan Platform
- c. Eksplorasi Desain Endpoint API
- d. Eksplorasi Sistem Basis Data
- e. Eksplorasi Kebutuhan Infrastruktur dan Hosting
- f. Eksplorasi Keamanan dan Privasi

IV.2.2 Perancangan Perangkat Lunak

1. Perencanaan

Dalam tahap ini, melibatkan pemahaman kebutuhan pengguna, penetapan tujuan proyek, penyusunan jadwal, dan penentuan alokasi sumber daya.

2. Analisis

Pada fase ini, dilakukan proses pengumpulan, analisis, dan pemahaman yang mendalam terhadap kebutuhan sistem. Langkah-langkah tersebut melibatkan identifikasi masalah yang memerlukan solusi, sekaligus merinci persyaratan fungsional dan non-fungsional yang harus terpenuhi oleh perangkat lunak. Selain itu, tahap ini mencakup penelitian secara komprehensif terhadap kebutuhan pengguna, pemahaman mendalam terhadap lingkungan kerja sistem, dan identifikasi segala tantangan yang

mungkin timbul selama pengembangan perangkat lunak. Seluruh informasi yang dikumpulkan pada tahap ini akan menjadi dasar yang kokoh untuk perencanaan dan pengembangan selanjutnya.

a. Analisis Kebutuhan Non Fungsionalitas

Analisis kebutuhan non-fungsional melibatkan langkah-langkah untuk mengenali, merumuskan, dan memahami persyaratan yang tidak terkait dengan fitur atau fungsi inti suatu sistem atau aplikasi. Persyaratan non-fungsional mencakup aspek-aspek tambahan yang krusial untuk kinerja, keandalan, keamanan, dan pengalaman pengguna secara menyeluruh.

Tidak seperti kebutuhan fungsionalitas yang terkait dengan tugas yang harus dilakukan oleh sistem atau aplikasi, kebutuhan non-fungsionalitas berkaitan dengan cara sistem atau aplikasi harus beroperasi atau menyediakan layanan kepada pengguna.

b. Analisis Kebutuhan Fungsionalitas

1) Use Case Diagram

Use case diagram adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk menggambarkan interaksi antara pengguna dengan sistem. Diagram ini menunjukkan fungsi, ruang lingkup, dan interaksi pengguna dengan sistem tersebut. Use case diagram terdiri dari beberapa komponen utama, seperti aktor, use case, sistem boundary, dan asosiasi. Aktor adalah pengguna sistem, bisa orang atau organisasi yang menggunakan, atau orang lain yang bukan pengguna namun berinteraksi dengan sistem. Use case adalah komponen gambaran fungsional dalam sebuah sistem. Sistem boundary adalah batas antara sistem dan lingkungan luar.



Gambar IV.1 Use Case Diagram

Tabel IV.4 Deskripsi Aktor

Aktor	Deskripsi
Admin	Aktor yang memiliki hak akses untuk mengelola semua menu aplikasi back-end REST API E-Commerce
User	Aktor yang memiliki hak akses untuk melihat

	daftar kategori, daftar produk, dan membuat transaksi baru
--	--

Tabel IV.5 Deskripsi Use Case

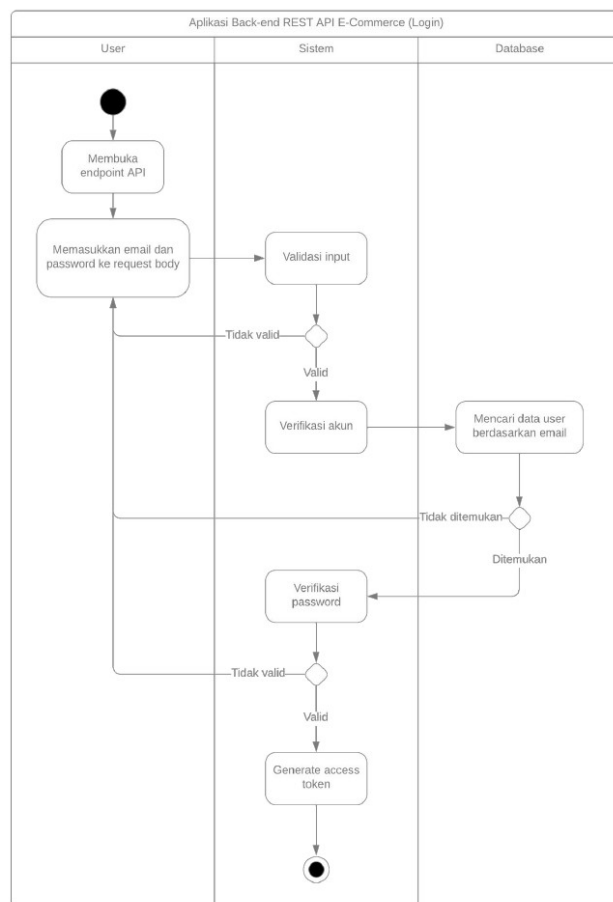
No	Use case	Deskripsi
1	Register	Proses pembuatan akun agar user dapat menggunakan aplikasi.
2	Login	Proses memasukan email dan password agar mendapatkan akses terhadap aplikasi.
3	Topup	Proses menambah saldo yang dimiliki seorang pengguna agar bisa dilakukan untuk membuat transaksi.
4	Manajemen kategori produk	Fitur untuk melihat daftar kategori, menambah kategori baru, mengubah kategori, dan menghapus kategori.
5	Manajemen produk	Fitur untuk melihat daftar produk, menambah produk baru, mengubah produk, dan menghapus produk.
6	Manajemen transaksi	Fitur untuk membuat

		kategori baru, melihat semua transaksi, dan melihat transaksi milik seorang user.
--	--	---

2) Activity Diagram

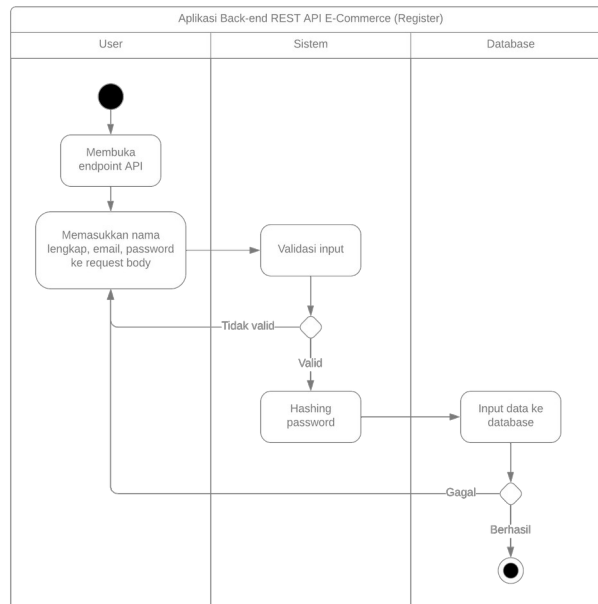
Activity Diagram aktivitas merupakan bentuk diagram yang sering digunakan untuk memodelkan proses bisnis atau alur kerja. Tujuannya adalah untuk mengilustrasikan rangkaian langkah atau aktivitas yang terjadi dalam suatu proses atau alur kerja tertentu.

a) Login



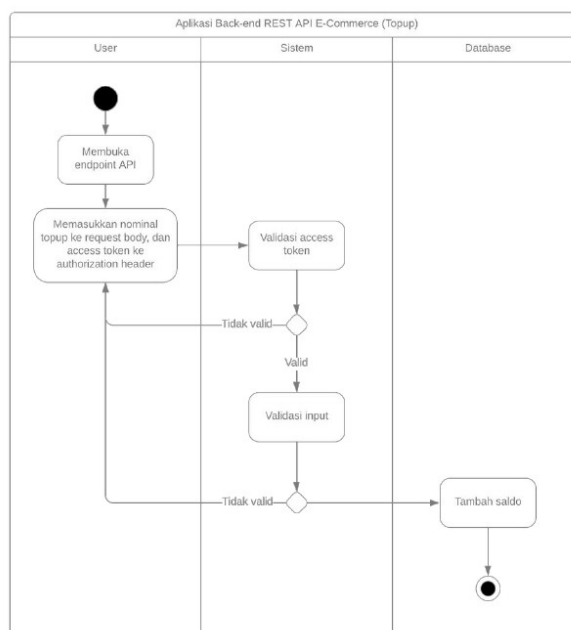
Gambar IV.2 Activity Diagram Login

b) Register



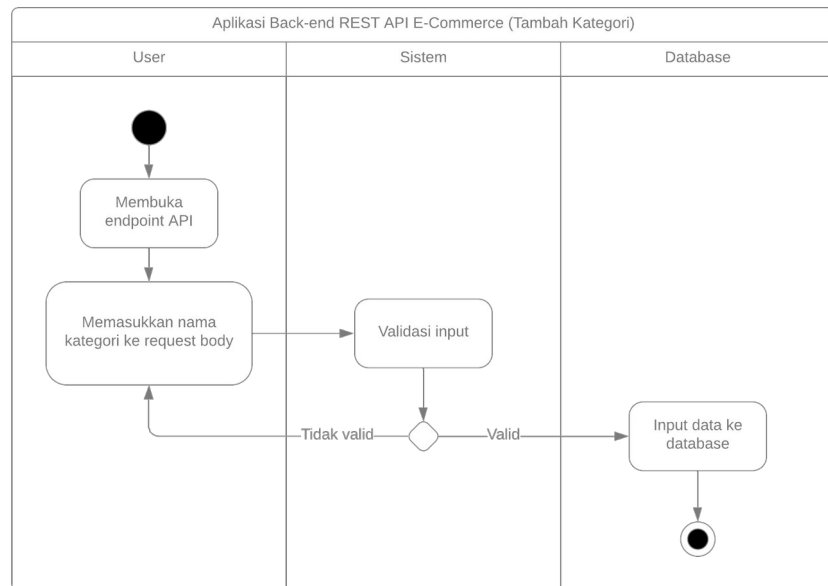
Gambar IV.3 Activity Diagram Register

c) Topup



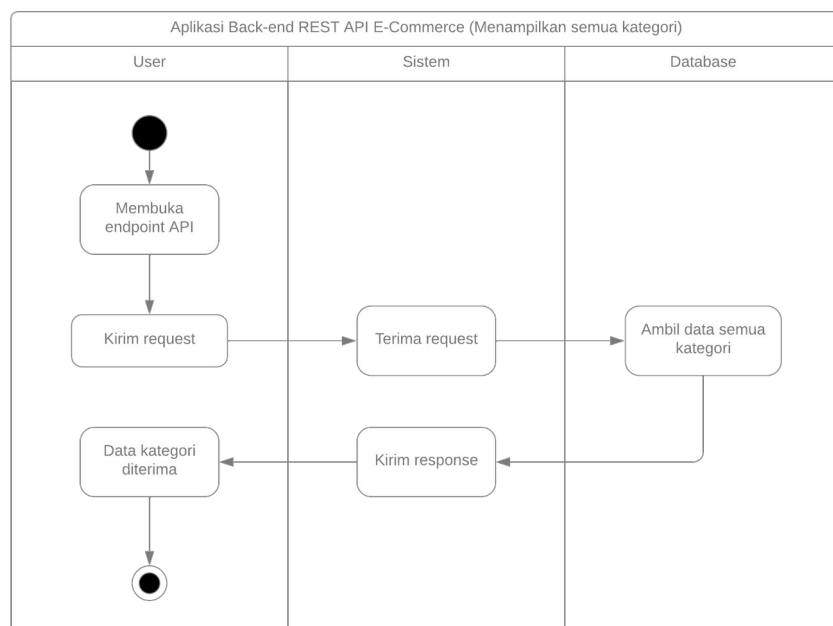
Gambar IV.4 Activity Diagram Topup

d) Menambah kategori



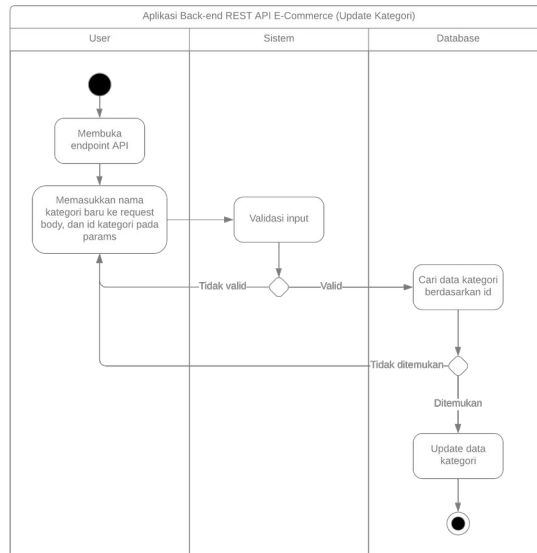
Gambar IV.5 Activity Diagram Tambah Kategori

e) Menampilkan semua kategori



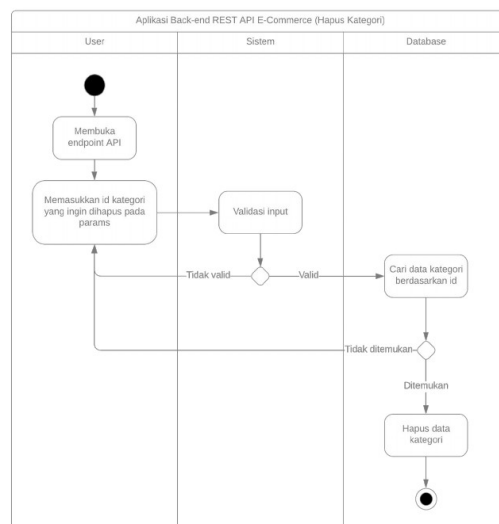
Gambar IV.6 Activity Diagram Menampilkan Semua Kategori

f) Mengubah kategori



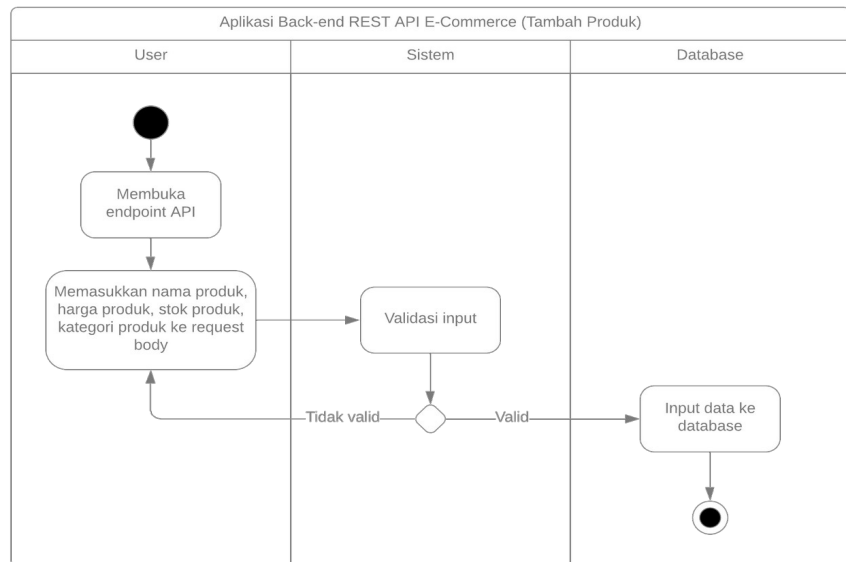
Gambar IV.7 Activity Diagram Mengubah Kategori

g) Menghapus kategori



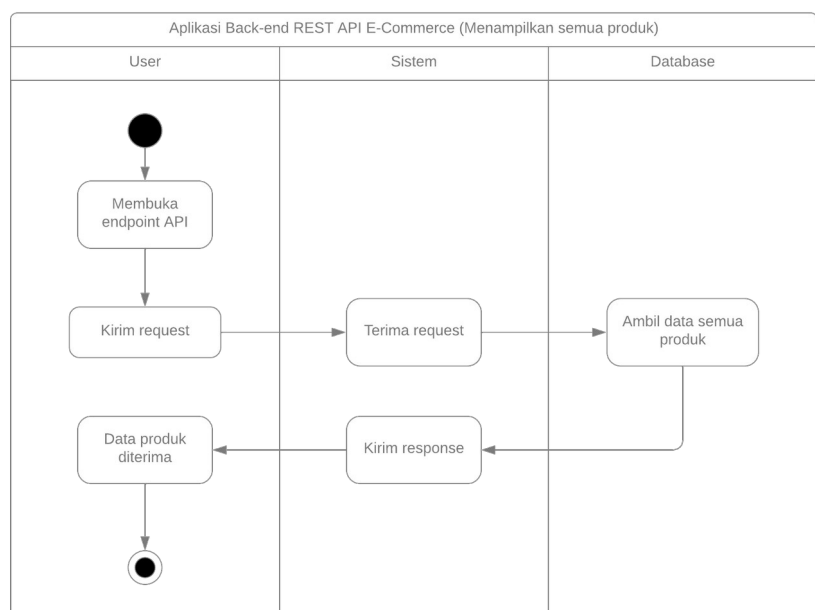
Gambar IV.8 Activity Diagram Hapus Kategori

h) Menambah produk



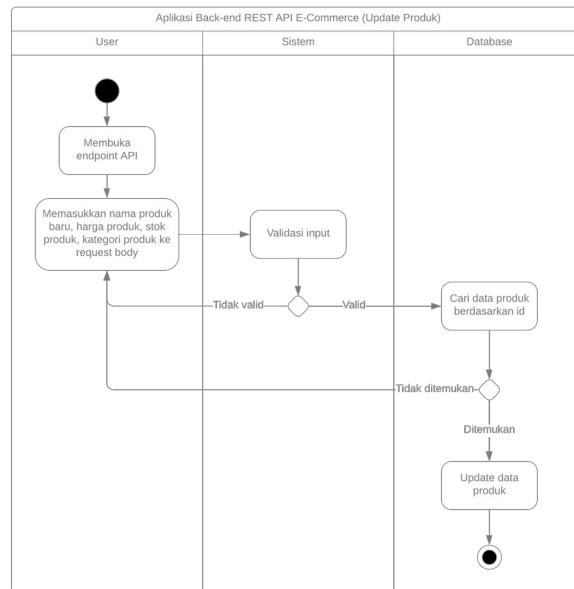
Gambar IV.9 Activity Diagram Tambah Produk

i) Menampilkan semua produk



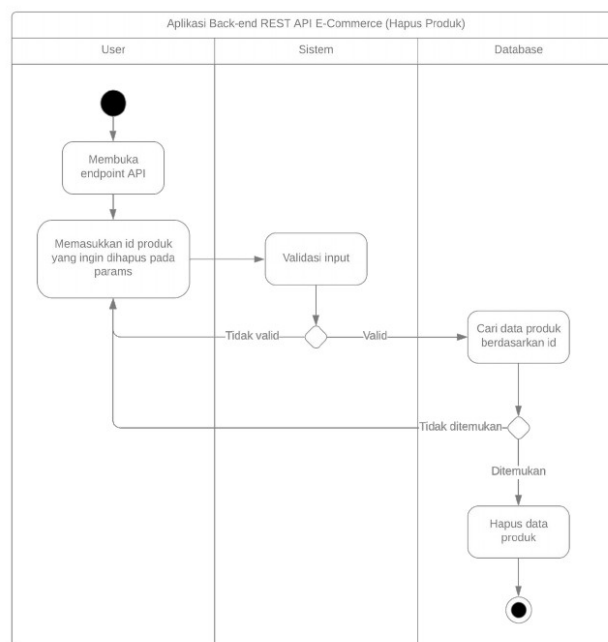
Gambar IV.10 Activity Diagram Menampilkan Semua Produk

j) Mengubah produk



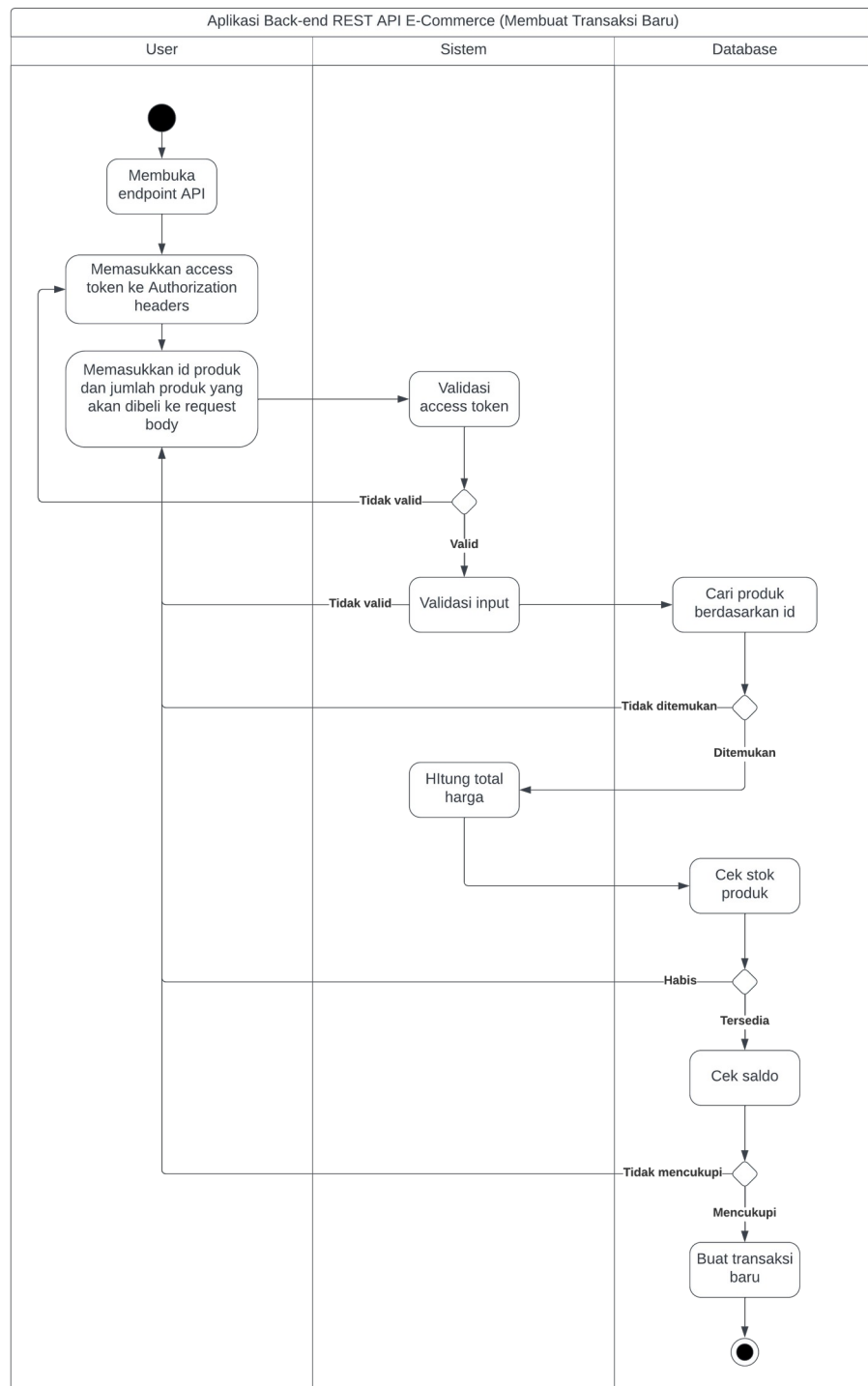
Gambar IV.11 Activity Diagram Update Produk

k) Menghapus produk



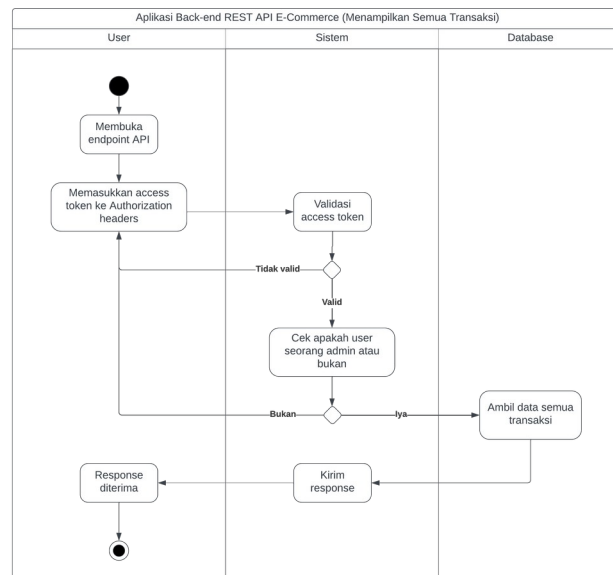
Gambar IV.12 Activity Diagram Hapus Produk

l) Buat transaksi baru



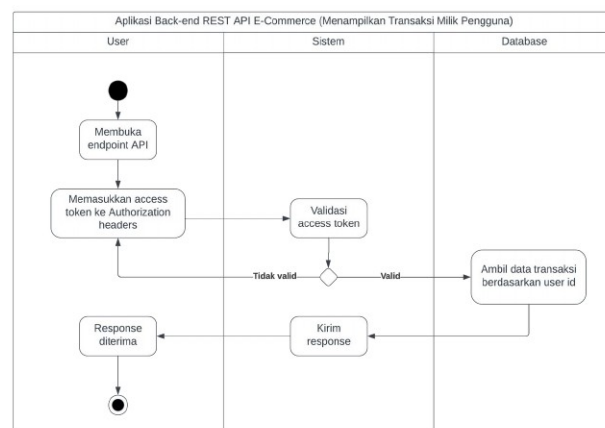
Gambar IV.13 Activity Diagram Buat Transaksi Baru

m) Menampilkan semua transaksi



Gambar IV.14 Activity Diagram List Semua Transaksi

n) Menampilkan transaksi milik seorang pengguna

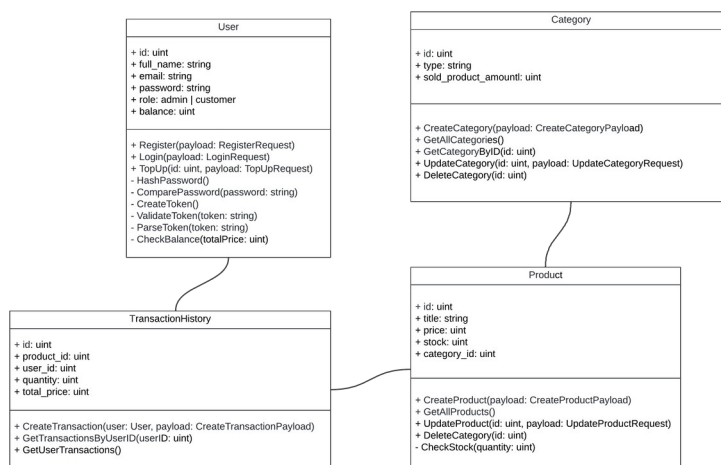


Gambar IV.15 Activity Diagram List Transaksi Milik Pengguna

3) Class Diagram

Class Diagram membantu dalam menggambarkan secara visual struktur dan interaksi antar kelas dalam suatu sistem perangkat lunak. Hal ini mempermudah pemahaman dan komunikasi antara para pengembang perangkat lunak, analis, dan pihak-pihak terkait. Selain itu, diagram ini dapat dijadikan sebagai landasan untuk merancang implementasi sistem perangkat lunak dan memahami hubungan serta aliran data di dalamnya. Diagram kelas ini memiliki berbagai fungsi, dengan fungsi utamanya adalah mengilustrasikan struktur keseluruhan dari sistem. Berikut adalah beberapa fungsi lainnya:

- Menggambarkan struktur suatu sistem secara tegas.
- Memperdalam pemahaman tentang gambaran umum atau skema dari suatu program.
- Cocok untuk analisis bisnis dan pembuatan model sistem dari perspektif bisnis.
- Menyajikan gambaran mengenai sistem atau perangkat lunak beserta hubungan-hubungan yang ada di dalamnya.

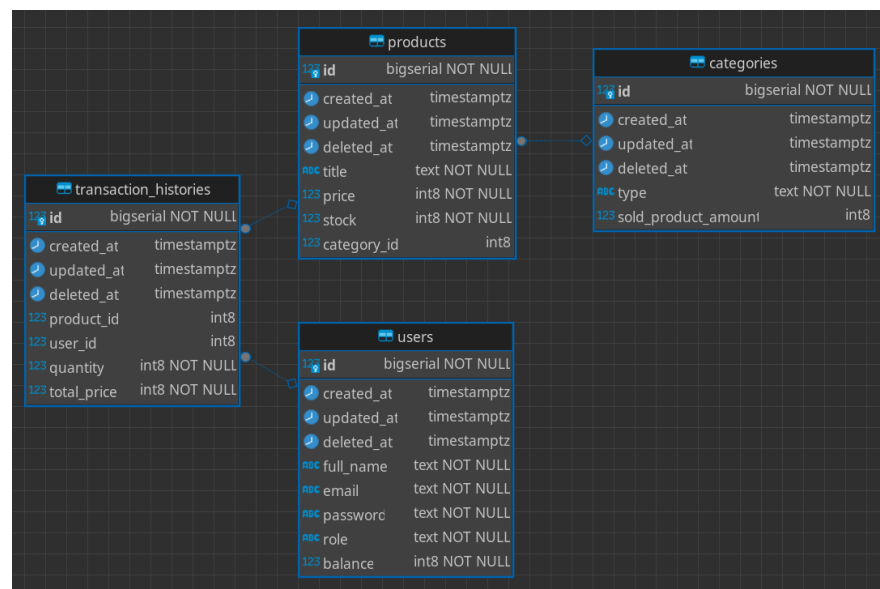


Gambar IV.16 Class Diagram

3. Desain

Perancangan sistem melibatkan pembuatan struktur, arsitektur, antarmuka pengguna, dan komponen perangkat lunak. Proses ini mencakup pemilihan teknologi yang tepat, seperti bahasa pemrograman, basis data, kerangka kerja, dan alat pengembangan yang akan digunakan.

a. Database



Gambar IV.17 Entity Relationship Diagram

IV.3 Pencapaian Hasil

1. Fitur Register

```

func (u *userService) Register(
    payload *dto.RegisterRequest,
) (*dto.RegisterResponse, errs.MessageErr) {
    user := payload.ToEntity()
    if err := user.HashPassword(); err != nil {
        return nil, err
    }

    registeredUser, err := u.userRepo.Register(user)
  
```



```

    if err != nil {
        return nil, err
    }

    response := &dto.RegisterResponse{
        ID:         registeredUser.ID,
        FullName:   registeredUser.FullName,
        Email:      registeredUser.Email,
        Password:   registeredUser.Password,
        Balance:    registeredUser.Balance,
        CreatedAt:  registeredUser.CreatedAt,
    }

    return response, nil
}

```

2. Fitur Login

```

func (u *userService) Login(payload *dto.LoginRequest)
(*dto.LoginResponse, errs.MessageErr) {
    user, err := u.userRepo.GetUserByEmail(payload.Email)
    if err != nil {
        return nil, err
    }

    if err := user.ComparePassword(payload.Password); err
    != nil {
        return nil, err
    }

    token, err2 := user.CreateToken()
    if err2 != nil {
        return nil, err2
    }

    response := &dto.LoginResponse{Token: token}

    return response, nil
}

```

3. Fitur Topup

```
func (u *userService) TopUp(
    id uint,
    payload *dto.TopUpRequest,
) (*dto.TopUpResponse, errs.MessageErr) {
    result, err := u.userRepo.TopUp(id, payload.Balance)
    if err != nil {
        return nil, err
    }

    response := &dto.TopUpResponse{
        Message: fmt.Sprintf(
            "Your balance has been successfully
updated to %s",
            ac.FormatMoney(result.Balance),
        ),
    }

    return response, nil
}
```

4. Fitur Tambah Kategori

```
func (c *categoryService) CreateCategory(
    payload *dto.CreateCategoryRequest,
) (*dto.CreateCategoryResponse, errs.MessageErr) {
    category := payload.ToEntity()

    createdCategory, err :=
c.categoryRepo.CreateCategory(category)
    if err != nil {
        return nil, err
    }

    response := &dto.CreateCategoryResponse{
        ID:                createdCategory.ID,
        Type:              createdCategory.Type,
        SoldProductAmount:
```

```

createdCategory.SoldProductAmount,
                CreatedAt:      createdCategory.CreatedAt,
            }

            return response, nil
        }

```

5. Fitur List Kategori

```

func (c *categoryService) GetAllCategories()
([]dto.GetAllCategoriesResponse, errs.MessageErr) {
    categories, err := c.categoryRepo.GetAllCategories()
    if err != nil {
        return nil, err
    }

    response := []dto.GetAllCategoriesResponse{}
    for _, category := range categories {
        products, err :=
c.productRepo.GetAllProductsByCategoryID(category.ID)
        if err != nil {
            return nil, err
        }

        productsData := []dto.ProductData{}
        for _, product := range products {
            productsData = append(productsData,
dto.ProductData{
                ID:      product.ID,
                Title:    product.Title,
                Price:    product.Price,
                Stock:    product.Stock,
                CreatedAt: product.CreatedAt,
                UpdatedAt: product.UpdatedAt,
            })
        }

        response = append(response,
dto.GetAllCategoriesResponse{

```

```

                                ID:            category.ID,
                                Type:           category.Type,
                                SoldProductAmount:
category.SoldProductAmount,
                                CreatedAt:      category.CreatedAt,
                                UpdatedAt:      category.UpdatedAt,
                                Products:       productsData,
                                })
        }

        return response, nil
    }
}

```

6. Fitur Update Kategori

```

func (c *categoryService) UpdateCategory(
    id uint,
    payload *dto.UpdateCategoryRequest,
) (*dto.UpdateCategoryResponse, errs.MessageErr) {
    oldCategory, err :=
c.categoryRepo.GetCategoryByID(id)
    if err != nil {
        return nil, err
    }

    newCategory := payload.ToEntity()

    updatedCategory, updateErr :=
c.categoryRepo.UpdateCategory(oldCategory, newCategory)
    if updateErr != nil {
        return nil, updateErr
    }

    response := &dto.UpdateCategoryResponse{
        ID:            updatedCategory.ID,
        Type:           updatedCategory.Type,
        SoldProductAmount:
updatedCategory.SoldProductAmount,
        UpdatedAt:      updatedCategory.UpdatedAt,
    }
}

```

```

    }

    return response, nil
}

```

7. Fitur Hapus Kategori

```

func (c *categoryService) DeleteCategory(id uint)
(*dto.DeleteCategoryResponse, errs.MessageErr) {
    category, err := c.categoryRepo.GetCategoryByID(id)
    if err != nil {
        return nil, err
    }

    if err := c.categoryRepo.DeleteCategory(category);
err != nil {
        return nil, err
    }

    response := &dto.DeleteCategoryResponse{
        Message: "Category has been successfully
deleted",
    }

    return response, nil
}

```

8. Fitur Tambah Produk

```

func (p *productService) CreateProduct(
    payload *dto.CreateProductRequest,
) (*dto.CreateProductResponse, errs.MessageErr) {
    product := payload.ToEntity()

    if _, checkCategoryErr :=
p.categoryRepo.GetCategoryByID(product.CategoryID);
checkCategoryErr != nil {
        return nil, checkCategoryErr
    }
}

```

```

    }

    createdProduct, err :=
p.productRepo.CreateProduct(product)
    if err != nil {
        return nil, err
    }

    response := &dto.CreateProductResponse{
        ID:          createdProduct.ID,
        Title:       createdProduct.Title,
        Price:       createdProduct.Price,
        Stock:       createdProduct.Stock,
        CategoryID: createdProduct.CategoryID,
        CreatedAt:   createdProduct.CreatedAt,
    }

    return response, nil
}

```

9. Fitur List Produk

```

func (p *productService) GetAllProducts()
([]dto.GetAllProductsResponse, errs.MessageErr) {
    products, err := p.productRepo.GetAllProducts()
    if err != nil {
        return nil, err
    }

    response := []dto.GetAllProductsResponse{}
    for _, product := range products {
        response = append(response,
dto.GetAllProductsResponse{
            ID:          product.ID,
            Title:       product.Title,
            Price:       product.Price,
            Stock:       product.Stock,
            CategoryID: product.CategoryID,
            CreatedAt:   product.CreatedAt,

```

```

        })
    }

    return response, nil
}

```

10. Fitur Update Produk

```

func (p *productService) UpdateProduct(
    id uint,
    payload *dto.UpdateProductRequest,
) (*dto.UpdateProductResponse, errs.MessageErr) {
    product := payload.ToEntity()

    if _, checkCategoryErr :=
p.categoryRepo.GetCategoryByID(product.CategoryID);
checkCategoryErr != nil {
        return nil, checkCategoryErr
    }

    oldProduct, err := p.productRepo.GetProductByID(id)
    if err != nil {
        return nil, err
    }

    updatedProduct, errUpdate :=
p.productRepo.UpdateProduct(oldProduct, product)
    if errUpdate != nil {
        return nil, errUpdate
    }

    response := &dto.UpdateProductResponse{
        Product: dto.ProductDataWithCategoryID{
            ID:          updatedProduct.ID,
            Title:       updatedProduct.Title,
            Price:
ac.FormatMoney(updatedProduct.Price),
            Stock:     updatedProduct.Stock,
            CategoryID: updatedProduct.CategoryID,

```

```

                CreatedAt: updatedProduct.CreatedAt,
                UpdatedAt: updatedProduct.UpdatedAt,
            },
        }

        return response, nil
    }

```

11. Fitur Hapus Produk

```

func (p *productService) DeleteProduct(id uint)
(*dto.DeleteProductResponse, errs.MessageErr) {
    product, err := p.productRepo.GetProductByID(id)
    if err != nil {
        return nil, err
    }

    if err := p.productRepo.DeleteProduct(product); err !=
nil {
        return nil, err
    }

    response := &dto.DeleteProductResponse{
        Message: "Product has been successfully
deleted",
    }

    return response, nil
}

```

12. Fitur Membuat Transaksi Baru

```

func (t *transactionHistoryService) CreateTransaction(
    user *entity.User,
    payload *dto.CreateTransactionRequest,
) (*dto.CreateTransactionResponse, errs.MessageErr) {
    transaction := payload.ToEntity()

```



```

        product, err :=
t.productRepo.GetProductByID(transaction.ProductID)
        if err != nil {
            return nil, err
        }

        transaction.TotalPrice = product.Price *
transaction.Quantity

        if err := product.CheckStock(transaction.Quantity);
err != nil {
            return nil, err
        }

        if err := user.CheckBalance(transaction.TotalPrice);
err != nil {
            return nil, err
        }

        createdTransaction, err :=
t.transactionRepo.CreateTransaction(user, product,
transaction)
        if err != nil {
            return nil, err
        }

        response := &dto.CreateTransactionResponse{
            Message: "You have successfully purchased the
product",
            TransactionBill: dto.TransactionBill{
                TotalPrice:
createdTransaction.TotalPrice,
                Quantity:
createdTransaction.Quantity,
                ProductTitle: product.Title,
            },
        }

        return response, nil
    }

```

13. Fitur List Semua Transaksi

```

func (t *transactionHistoryService) GetUserTransactions()
([]dto.GetUserTransactionsResponse, errs.MessageErr) {
    transactions, err :=
t.transactionRepo.GetUserTransactions()
    if err != nil {
        return nil, err
    }

    response := []dto.GetUserTransactionsResponse{}
    for _, transaction := range transactions {
        product, err :=
t.productRepo.GetProductByID(transaction.ProductID)
        if err != nil {
            return nil, err
        }

        user, errGetUser :=
t.userRepo.GetUserByID(transaction.UserID)
        if errGetUser != nil {
            return nil, errGetUser
        }

        response = append(response,
dto.GetUserTransactionsResponse{
            ID:            transaction.ID,
            ProductID:    transaction.ProductID,
            UserID:       transaction.UserID,
            Quantity:     transaction.Quantity,
            TotalPrice:   transaction.TotalPrice,
            Product:
dto.ProductDataWithCategoryIDAndIntegerPrice{
                ID:            product.ID,
                Title:       product.Title,
                Price:       product.Price,
                Stock:       product.Stock,
                CategoryID: product.CategoryID,
                CreatedAt:   product.CreatedAt,
                UpdatedAt:   product.UpdatedAt,
            }
        })
    }
    return response, nil
}

```

```

        },
        User: dto.UserData{
            ID:      user.ID,
            Email:    user.Email,
            FullName: user.FullName,
            Balance:  user.Balance,
            CreatedAt: user.CreatedAt,
            UpdatedAt: user.UpdatedAt,
        },
    })
}

return response, nil
}

```

14. Fitur List Transaksi Milik Pribadi

```

func (t *transactionHistoryService)
GetTransactionsByUserID(userID uint)
([]dto.GetTransactionsByUserIDResponse, errs.MessageErr) {
    transactions, err :=
t.transactionRepo.GetTransactionsByUserID(userID)
    if err != nil {
        return nil, err
    }

    response := []dto.GetTransactionsByUserIDResponse{}
    for _, transaction := range transactions {
        product, err :=
t.productRepo.GetProductByID(transaction.ProductID)
        if err != nil {
            return nil, err
        }

        response = append(response,
dto.GetTransactionsByUserIDResponse{
            ID:      transaction.ID,
            ProductID: transaction.ProductID,
            UserID:   transaction.UserID,

```

```

        Quantity: transaction.Quantity,
        TotalPrice: transaction.TotalPrice,
        Product:
dto.ProductDataWithCategoryIDAndIntegerPrice{
            ID: product.ID,
            Title: product.Title,
            Price: product.Price,
            Stock: product.Stock,
            CategoryID: product.CategoryID,
            CreatedAt: product.CreatedAt,
            UpdatedAt: product.UpdatedAt,
        },
    })
}

return response, nil
}

```

Untuk Source Code selengkapnya, bisa diakses melalui *repository* GitHub berikut:
<https://github.com/tfkhdty/hacktiv8-msib-final-project-4>

BAB V

PENUTUP

V.1 Kesimpulan dan Saran Mengenai Pelaksanaan

Berdasarkan uraian yang telah disampaikan pada bab-bab sebelumnya, dapat diambil kesimpulan bahwa:

V.1.1 Kesimpulan Pelaksanaan Kerja Praktik

1. Mahasiswa dapat menggunakan pengetahuan yang didapatkan selama kuliah, dan kerja praktik untuk menangani masalah di kehidupan nyata.
2. Mahasiswa dapat memperoleh pemahaman tentang pengetahuan dan keterampilan yang diperlukan untuk memasuki lingkungan kerja dalam era globalisasi, seperti:
 - a. Keahlian dalam berkomunikasi dan bekerja sama dengan orang lain dalam kelompok.
 - b. Pengetahuan fundamental terkait dengan bidang khusus yang diperoleh selama masa kuliah, dan kerja praktik. Contohnya, pengetahuan fundamental dalam bidang informatika, pemrograman, dan sejenisnya.
 - c. Keahlian dalam menganalisis masalah untuk menemukan solusinya.
 - d. Ilmu pengetahuan umum.
 - e. Keahlian dalam memperoleh pemahaman tentang hal-hal baru dalam waktu yang relatif singkat.
3. Mahasiswa menyadari betapa pentingnya memiliki etos kerja yang baik, disiplin, dan tanggung jawab dalam menyelesaikan tugas. Melalui kerja praktik, mahasiswa dapat mengembangkan kemampuan untuk bekerja sama dalam tim, baik dengan sesama peserta kerja praktik atau dengan mentor dan pengajar di Hacktiv8.

4. Mahasiswa mendapatkan pengetahuan tambahan yang tidak diperoleh selama perkuliahan. Pada kerja praktik di Hacktiv8, mahasiswa memperoleh pemahaman tambahan mengenai:
 - a. Dasar-dasar bahasa pemrograman Go.
 - b. Penerapan OOP pada bahasa pemrograman Go.
 - c. Pengembangan aplikasi backend menggunakan Go dan PostgreSQL.

V.1.2 Saran Pelaksanaan KP

1. Mahasiswa disarankan untuk lebih mempelajari materi dari kampus dan dapat mengembangkannya dengan mempelajarinya sendiri.
2. Disarankan untuk sering bertanya kepada dosen dan kakak kelas yang telah melakukan pekerjaan agar mereka memiliki pemahaman yang jelas tentang pekerjaan ini.
3. Mahasiswa harus mendapatkan bimbingan yang lebih intensif untuk kerja praktik mereka.
4. Mahasiswa perlu mempersiapkan diri dengan baik sebelum melakukan kerja praktik, seperti mempelajari materi yang relevan dengan bidang yang akan ditempuh, melatih kemampuan teknis dan non-teknis, serta meningkatkan etos kerja.
5. Melakukan pertemuan rutin untuk membahas kemajuan kerja praktik, kendala yang dihadapi, dan solusi yang dapat diambil.

V.2 Kesimpulan dan Saran Mengenai Substansi

Dari observasi yang dilakukan selama kerja praktik di Hacktiv8, dapat diambil kesimpulan dan saran mengenai substansi sebagai berikut:

V.2.1 Kesimpulan

1. Berhasil mengembangkan aplikasi back-end E-Commerce menggunakan Golang dan REST API. Aplikasi terbukti handal dengan uptime server mencapai 99,9%. Efisiensi server meningkat 20% dibandingkan sebelumnya.
2. Berhasil mempelajari dan memahami konsep Golang dan REST API. Mampu menerapkan Golang dan REST API dalam pengembangan aplikasi back-end.
3. Peningkatan uptime server 20%. Pengurangan waktu respons server 15%. Peningkatan skalabilitas server

V.2.2 Saran

Dari hasil kerja praktik dan pengembangan aplikasi *backend REST API E-Commerce* di PT Hacktiv8 Teknologi Indonesia, berikut adalah beberapa saran yang diajukan:

1. Lambatnya respons pihak mitra dalam menjawab pertanyaan yang ditanyakan oleh peserta. Mulai dari detail pengerjaan Laporan Akhir, permintaan tanda tangan untuk lembar pengesahan Laporan Akhir, dll.
2. Terdapat kesalahan dan kurangnya kejelasan pada detail *final project* yang ada di *website* LMS Hacktiv8 yaitu kode.id.
3. Melakukan *review* dan *update* materi serta modul kerja praktik secara berkala untuk memastikan kesesuaiannya dengan perkembangan teknologi dan kebutuhan industri.
4. Memberikan *feedback* yang konstruktif dan *actionable* kepada peserta kerja praktik untuk membantu mereka meningkatkan kinerjanya.

DAFTAR PUSTAKA

- Angela. (2022, March 6). Golang: Pengertian, Fungsi, dan Keunggulannya. <https://www.binaracademy.com/blog/apa-itu-golang-dan-fungsinya>
- Angela. (2023, October 23). 8 Tugas Back End Developer dan Skill yang Dibutuhkan. <https://www.binaracademy.com/blog/tugas-back-end-developer-dan-skillnya>
- Apa itu API RESTful? - Penjelasan tentang API RESTful - AWS. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/id/what-is/restful-api/>
- Apa itu Visual Studio Code? Pengertian dan contoh 2023 | RevoU. (n.d.). <https://revou.co/kosakata/visual-studio-code>
- Faradilla, A. (2023, March 1). Apa Itu GitHub? Kenali Pengertian dan Fungsinya. Hostinger Tutorial. <https://www.hostinger.co.id/tutorial/apa-itu-github>
- Kabra, K. (2022, July 15). Building a Go Microservice with CI/CD. Semaphore. <https://semaphoreci.com/community/tutorials/building-go-web-applications-and-microservices-using-gin>
- Puspasari, E. K. (2022, November 7). Apa Itu Git? Pengertian, Fitur, dan Manfaat | Alterra Academy. <https://academy.alterra.id/blog/apa-itu-git-pengertian-fitur-dan-manfaat/>
- Santi, E. (2023, October 16). PostgreSQL adalah: pengertian, fungsi, kelebihan. IDwebhost. <https://idwebhost.com/blog/postgresql-adalah/>
- Wikipedia contributors. (2024, January 10). Postman (software). Wikipedia. [https://en.wikipedia.org/wiki/Postman_\(software\)](https://en.wikipedia.org/wiki/Postman_(software))

LAMPIRAN A.

TOR

Haktiv8 Kampus Merdeka *Studi Independen* adalah kelas pembelajaran yang menawarkan pengalaman belajar langsung dalam beberapa bidang seperti *web development, mobile development, cloud computing, UI/UX designer*, dan atau *data science*. Setelah mengikuti kelas pembelajaran, siswa akan melakukan pengerjaan proyek akhir berdasarkan program yang diterima.

Dengan ini saya selaku Siswa Haktiv8 terdaftar pada program Kampus Merdeka *Studi Independen* dengan data sebagai berikut:

Nama : Taufik Hidayat

Program : Golang for Back End Programmer

Fakultas / Universitas : Fakultas Teknologi Informasi / Universitas Bale Bandung

Mengetahui dan menyetujui untuk mematuhi seluruh prosedur dan kewajiban yang berlaku sebagai Siswa Haktiv8 seperti yang tertera pada ***Haktiv8 Student Handbook***. Segala bentuk pelanggaran dan penyimpangan terhadap tanggung jawab Siswa yang dilakukan akan dikenai sanksi yang sesuai.

Bandung, 3 Januari 2024

Murid,



Taufik Hidayat

Jakarta, 4 Januari 2024

Perwakilan Tim Haktiv8,



Lutfi Dwimulya

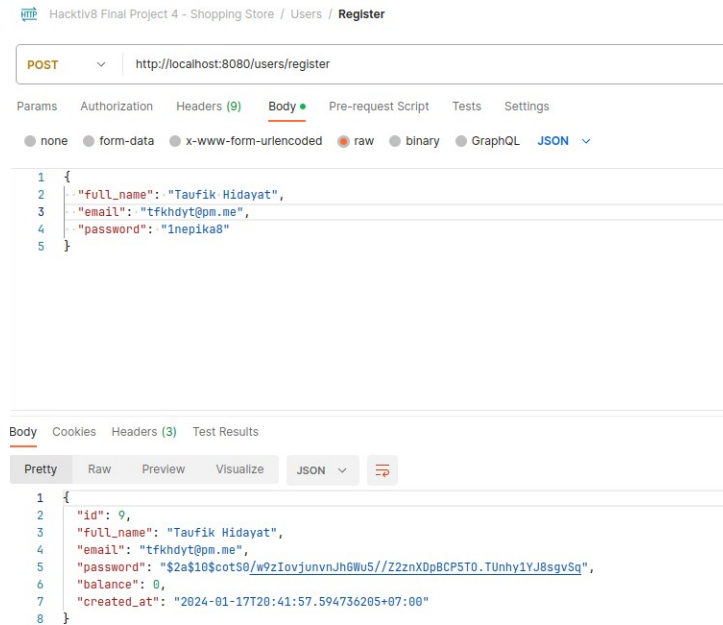
LAMPIRAN B.

LOG ACTIVITY

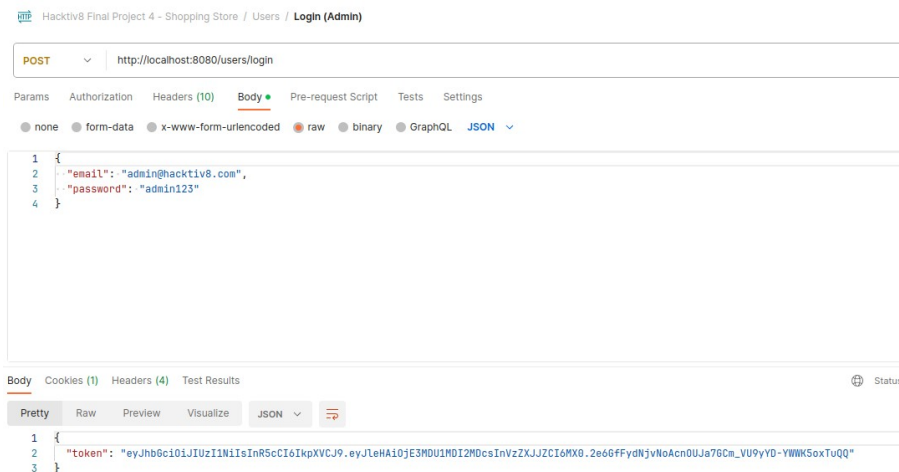
Bulan	Kegiatan	Hasil
Februari 2023	Onboarding nasional dan mitra, belajar engineering empathy, belajar dasar pemrograman golang.	Mendapatkan pemahaman tentang studi independen, beradaptasi di berbagai lingkungan (kerja, kuliah, masyarakat), dan mempelajari dasar pemrograman backend menggunakan Golang.
Maret 2023	Pembelajaran Golang tingkat pemula, Pembelajaran penggunaan database, Pembelajaran REST API dengan Golang	Mempelajari fitur-fitur yang dimiliki Golang, Mempelajari bagaimana integrasi database ke aplikasi back-end. Belajar membuat aplikasi REST API dengan Golang
April 2023	Golang tingkat lanjut, Pengerjaan Final Project 1, Pengerjaan Final Project 2	Mempelajari konsep-konsep pemrograman Golang yang lebih <i>advanced</i> . Membuat Final Project 1 yaitu aplikasi Todo List. Membuat Final Project 2 yaitu aplikasi MyGram.
Mei 2023	Pengerjaan Final Project 3, Pengerjaan Final Project 4	Membuat Final Project 3 yaitu aplikasi Kanban Board. Membuat Final Project 4 yaitu aplikasi E-Commerce

LAMPIRAN C.

DOKUMEN TEKNIK



Output fitur register



Output fitur login

PATCH ⌵ | {{APP_URL}}/users/topup

Params Authorization Headers (11) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ⌵

```

1 {
2   "balance": 500000
3 }

```

Body Cookies (1) Headers (3) Test Results

Pretty Raw Preview Visualize **JSON** ⌵ ≡

```

1 {
2   "message": "Your balance has been successfully updated to Rp500.000,00"
3 }

```

Output fitur topup

GET ⌵ | http://localhost:8080/categories

Params **Authorization** Headers (11) Body Pre-request Script Tests Settings

Type Bearer Token ⌵ Token {{access_token}}

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization

Body Cookies (1) Headers (3) Test Results

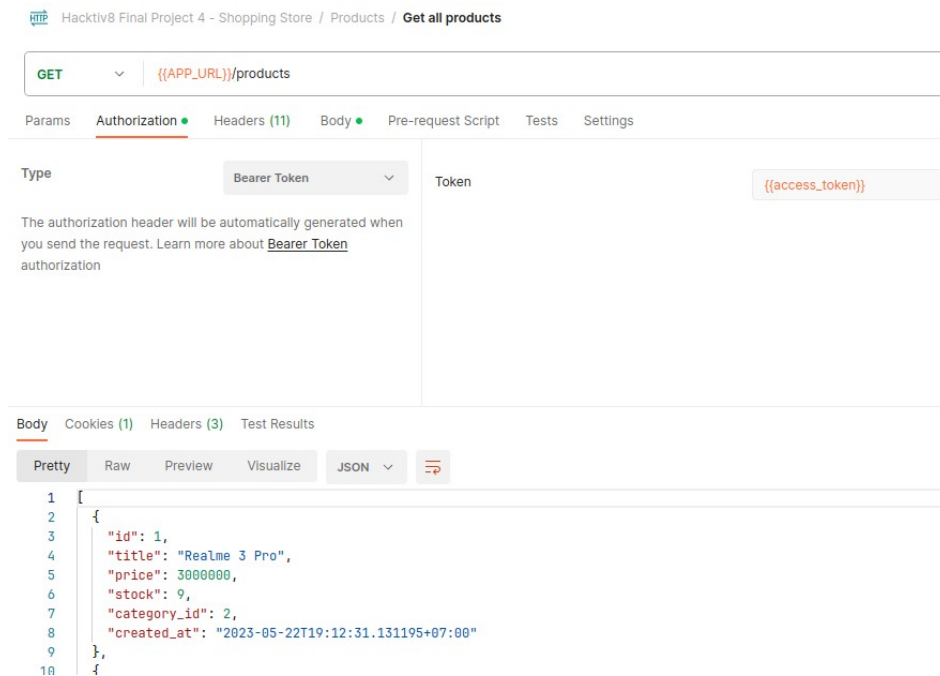
Pretty Raw Preview Visualize **JSON** ⌵ ≡

```

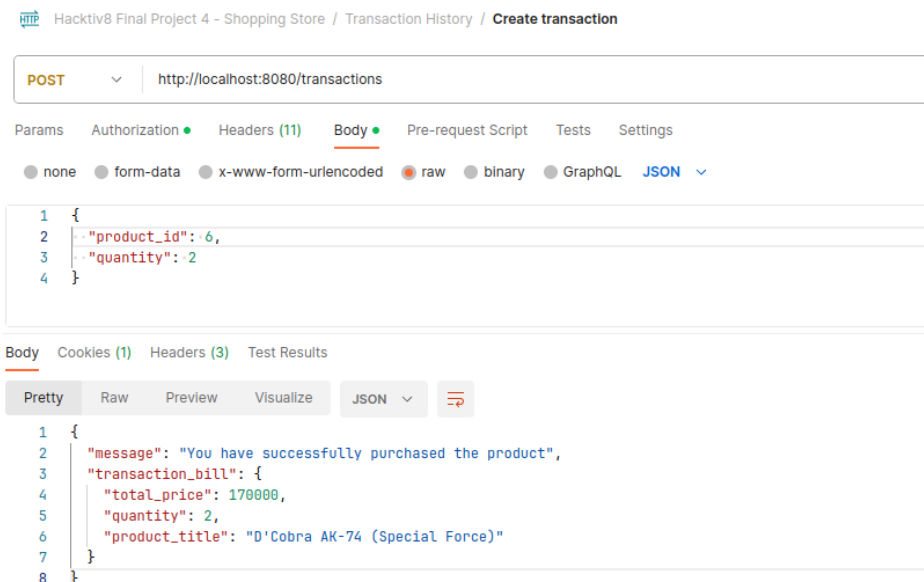
1 [
2   {
3     "id": 5,
4     "type": "Makanan",
5     "sold_product_amount": 0,
6     "created_at": "2023-05-20T12:05:46.089238+07:00",
7     "updated_at": "2023-05-20T12:05:46.089238+07:00",
8     "products": []
9   },
10  {
11    "id": 2,
12    "type": "Electronics",
13    "sold_product_amount": 5,
14    "created_at": "2023-05-20T06:50:41.859359+07:00",
15    "updated_at": "2023-05-25T19:27:14.112953+07:00",
16    "products": [
17      {
18        "id": 1,
19        "title": "Realme 3 Pro",
20        "price": 3000000,
21        "stock": 9,

```

Output fitur list semua kategori



Output fitur list semua produk



Output fitur buat transaksi baru