

Modular Golang Backend Architecture for the Workit Commercial Platform Database Operations

Arsitektur Backend Golang Modular untuk Operasi Basis Data Platform Komersial Workit

Ricki Maulana Abdillah¹, Cindy Taurusta²
{221080200127@umsida.ac.id, cindytaurusta@umsida.ac.id}

^{1,2} Program Studi Informatika Universitas Muhammadiyah Sidoarjo

Abstract. *In the digital era, the development of e-commerce-based applications has become crucial for providing efficient and fast online shopping services. This research aims to explain the implementation of the Golang programming language in backend development for the Workit e-commerce website, an online tools sales platform developed by Jobhun. The backend is developed using Golang, which was chosen for its speed and capability in handling large-scale systems. The system is equipped with MySQL Workbench database for data storage and uses Docker Compose to facilitate containerization management. API testing is conducted periodically using Postman to ensure each endpoint functions according to requirements.*

Keywords – Workit, Golang, Backend, E-commerce, Docker Compose, MySQL, API Testing

Abstrak. *Dalam era digital, pengembangan aplikasi berbasis e-commerce menjadi sangat penting untuk menyediakan layanan belanja online yang efisien dan cepat. Penelitian ini bertujuan untuk menjelaskan penerapan bahasa pemrograman Golang dalam pengembangan backend untuk website Workit e-commerce, sebuah platform penjualan alat (tools) online yang dikembangkan oleh Jobhun. Backend dikembangkan menggunakan Golang, yang dipilih karena kecepatan dan kemampuannya dalam menangani sistem yang berskala besar. Sistem ini dilengkapi dengan basis data MySQL Workbench untuk penyimpanan data dan menggunakan Docker Compose untuk memfasilitasi pengelolaan kontainerisasi. Pengujian API dilakukan secara berkala menggunakan Postman untuk memastikan setiap endpoint berfungsi sesuai kebutuhan.*

Kata Kunci – Workit, Golang, Backend, E-commerce, Docker Compose, MySQL, API Testing

I. PENDAHULUAN

Website Workit e-commerce Jobhun merupakan platform yang menyediakan layanan penjualan alat-alat atau tools online, bertujuan untuk menghadirkan pengalaman belanja yang cepat, aman, dan handal bagi pengguna, baik untuk admin maupun buyer. Dalam sistem ini, terdapat dua peran utama, yaitu admin dan buyer. Buyer berfungsi sebagai pembeli yang melakukan proses transaksi untuk mendapatkan produk atau alat yang diinginkan, sementara admin bertugas mengelola produk yang tersedia, memantau aktivitas pengguna, serta memastikan keamanan dan kelancaran operasional sistem. Karena pentingnya keamanan dan performa yang tinggi, backend dari sistem ini dikembangkan menggunakan bahasa pemrograman Golang yang telah dikenal efisien dalam menangani proses dengan beban kerja tinggi, termasuk dalam lingkungan e-commerce yang membutuhkan kecepatan dan stabilitas.

Penerapan Golang pada backend Workit juga memungkinkan pengelolaan data secara konkuren,[1], [2] sehingga meningkatkan kemampuan sistem dalam melayani permintaan yang besar dari berbagai pengguna secara bersamaan. Database yang digunakan adalah MySQL Workbench, yang berfungsi sebagai basis penyimpanan data utama untuk informasi pengguna, produk, pembayaran, dan transaksi pesanan. Dengan menggunakan MySQL, sistem dapat memanipulasi data dengan cepat dan menjaga konsistensi informasi pengguna serta transaksi di dalamnya. Selain itu, proyek ini juga menggunakan Docker Compose untuk memudahkan proses pengembangan dan pengelolaan lingkungan aplikasi yang terisolasi, yang membantu dalam penerapan aplikasi di berbagai perangkat atau server tanpa mengkhawatirkan perbedaan pengaturan sistem.[3], [4]

Setiap API pada sistem backend Workit dibagi menjadi beberapa bagian penting, yaitu API untuk mengelola pengguna, produk, pembayaran, dan pemesanan. Masing-masing API ini dirancang dengan struktur modular, yang terdiri dari beberapa komponen, seperti handler, repository, usecase, dan route.go. Struktur modular ini memisahkan tanggung jawab dari setiap komponen, sehingga memudahkan pengelolaan dan pemeliharaan sistem dalam jangka panjang. Dalam modul ini juga terdapat service.go yang bertugas menghubungkan antar komponen API, memfasilitasi alur data, dan mendukung proses transaksi pada setiap fungsi dengan cara yang efisien. Untuk memastikan setiap API berfungsi dengan benar dan siap digunakan oleh pengguna, pengujian dilakukan secara berkala menggunakan

Postman. Pengujian ini mencakup berbagai skenario penggunaan, memastikan API berjalan sesuai kebutuhan dan memberikan hasil yang akurat serta dapat diandalkan. [5], [6]

II. METODE

Artikel ini disusun berdasarkan pengalaman langsung dalam mengembangkan backend untuk Website Workit e-commerce Jobhun, di mana seluruh data berasal dari proses pembuatan sistem backend yang dilakukan secara mandiri. Proses pengembangan ini mencakup tahap perencanaan, perancangan, hingga pengujian, dengan berfokus pada penggunaan bahasa pemrograman Golang dan database MySQL untuk mencapai performa optimal pada aplikasi e-commerce. Setiap tahapan pengembangan diikuti dengan peninjauan dan perbaikan secara iteratif guna memastikan bahwa setiap komponen backend berjalan sesuai dengan fungsinya. Langkah pertama dalam metode ini adalah merancang arsitektur modular yang memisahkan logika bisnis ke dalam beberapa API utama, termasuk pengelolaan pengguna, produk, pembayaran, dan pemesanan. Masing-masing API diorganisir dalam struktur module, yang terdiri dari handler, repository, usecase, dan route.go. Struktur ini diterapkan untuk mendukung skalabilitas dan mempermudah pemeliharaan serta pengembangan di masa depan. Selain itu, modul ini dilengkapi dengan service.go, yang berfungsi menghubungkan antar-komponen dalam modul, memastikan komunikasi yang efisien antara API dan database.[7]

Database yang digunakan adalah MySQL, yang diatur dengan skema relasional untuk mendukung pengelolaan data pengguna, produk, transaksi, dan pembayaran secara efektif. Seluruh rancangan skema database dirancang untuk memaksimalkan performa, dengan memperhatikan relasi antar tabel yang kompleks. MySQL Workbench digunakan untuk membantu visualisasi skema database, serta memudahkan pengelolaan dan pemantauan data selama proses pengembangan dan pengujian. Dalam pengembangan API, setiap endpoint diuji menggunakan Postman untuk memastikan fungsionalitas dan keamanannya. Tahap pengujian dilakukan pada setiap fase pengembangan untuk mendeteksi dan memperbaiki bug secepat mungkin, khususnya dalam skenario kritis seperti autentikasi pengguna, pemrosesan pembayaran, dan pengelolaan inventaris produk. Pada tahap pengujian ini, Bearer Token digunakan sebagai mekanisme autentikasi guna menjamin keamanan akses ke setiap API, terutama untuk transaksi yang melibatkan data sensitif pengguna dan pembelian.[8]

Untuk memastikan keseragaman dan portabilitas, Docker Compose digunakan dalam containerization, yang memungkinkan pengelolaan lingkungan pengembangan yang konsisten di berbagai sistem. Dengan menggunakan Docker, aplikasi backend dapat dijalankan dalam container yang terisolasi, yang meminimalkan risiko kesalahan konfigurasi antar perangkat dan mempermudah implementasi ke server produksi. Metode ini tidak hanya melibatkan penerapan teknologi terkini, tetapi juga memungkinkan pendekatan iteratif, di mana setiap modul dapat disesuaikan dan dikembangkan lebih lanjut. Dengan pendekatan berbasis pengalaman langsung ini, diharapkan artikel ini mampu memberikan panduan teknis yang mendalam serta solusi praktis untuk pengembangan aplikasi e-commerce yang andal dan responsif.

Perancangan sistem backend untuk website Workit e-commerce Jobhun ini menggunakan metode Waterfall, dengan menerapkan pendekatan perangkat lunak secara bertahap dan berurutan, mulai dari perencanaan, perancangan, implementasi, hingga pengujian. Dengan menggunakan metode Waterfall, proses pengembangan ini dapat dilakukan secara sistematis dan efisien, memastikan setiap komponen berjalan sesuai dengan fungsinya.

1. Analisis Kebutuhan

Tahap pertama adalah mengumpulkan dan menganalisis kebutuhan sistem backend untuk website Workit. Proses ini dilakukan melalui observasi dan wawancara dengan pihak terkait untuk memahami kebutuhan pengelolaan data pengguna, produk, transaksi, pembayaran, dan pemesanan. Kebutuhan utama sistem ini meliputi pengelolaan API untuk pengguna, produk, pembayaran, dan pemesanan, serta pengelolaan database yang efisien untuk mendukung aplikasi e-commerce yang andal.

2. Desain Sistem

Berdasarkan kebutuhan yang telah dianalisis, dilakukan perancangan sistem dengan menggunakan desain modular. Desain ini melibatkan pembuatan arsitektur API yang terdiri dari beberapa modul utama, termasuk pengelolaan pengguna, produk, pembayaran, dan pemesanan. Struktur sistem melibatkan pembuatan handler, repository, usecase, dan file route.go untuk setiap modul, yang bertujuan untuk mendukung skalabilitas dan mempermudah pemeliharaan sistem ke depan. Desain database menggunakan MySQL dengan skema relasional untuk mendukung pengelolaan data yang efisien, serta menggunakan MySQL Workbench untuk visualisasi dan pemantauan skema database.

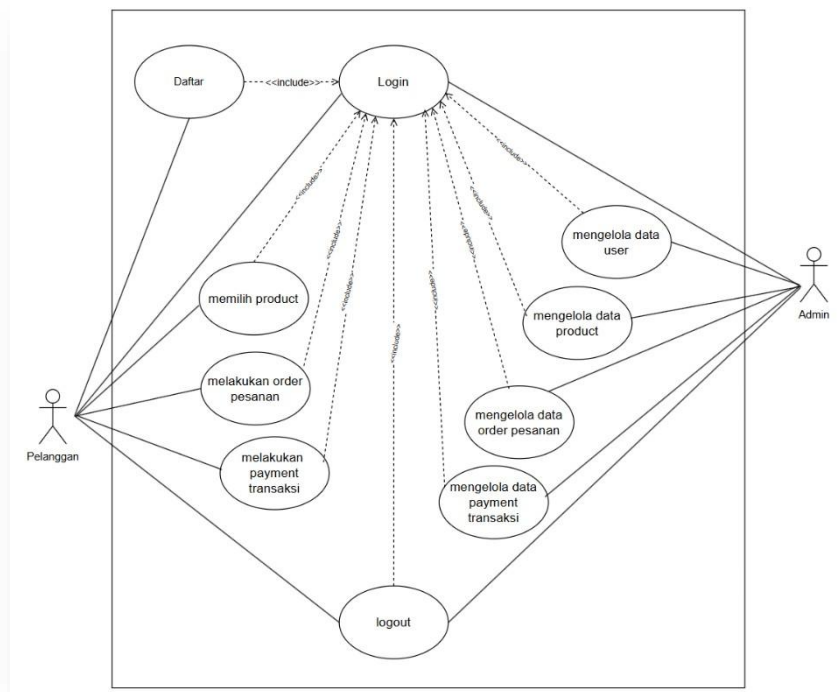
3. Implementasi

Sistem dikembangkan sesuai dengan desain yang telah dirancang sebelumnya. Backend website **Workit** menggunakan bahasa pemrograman Golang untuk menciptakan API yang efisien dan cepat. Pengelolaan database menggunakan MySQL, dengan setiap API diorganisir dalam modul terpisah dan diuji

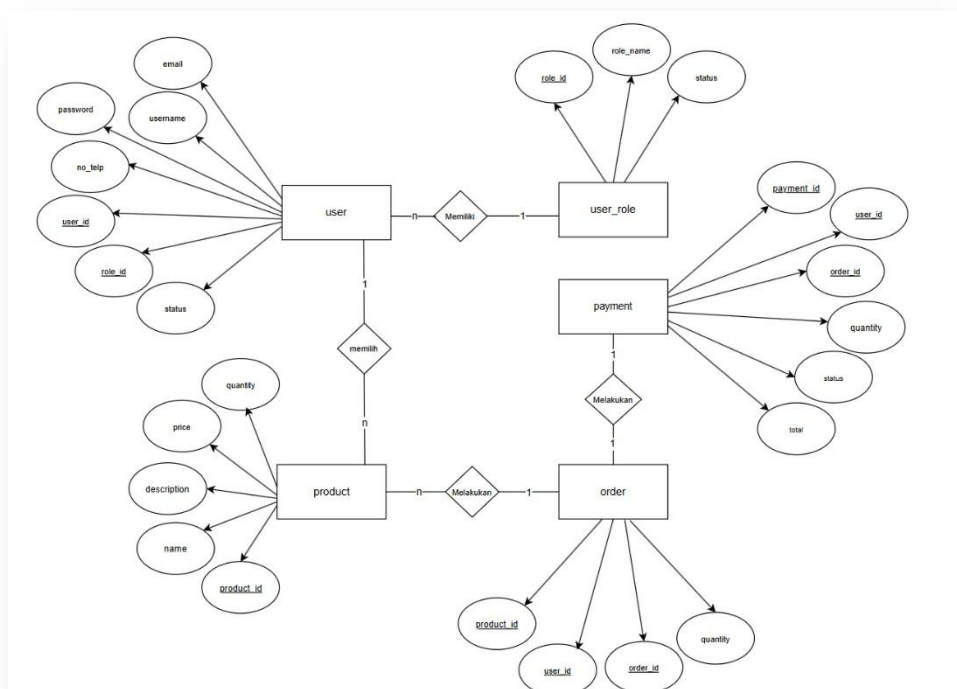
menggunakan Postman untuk memastikan fungsionalitas dan keamanannya. Seluruh API yang dikembangkan mencakup pengelolaan pengguna, transaksi, pembayaran, serta pemesanan produk. Proses pengembangan ini dilakukan secara iteratif untuk memastikan kualitas dan integrasi antar modul yang baik.

4. Pengujian

Setelah implementasi, dilakukan pengujian sistem secara menyeluruh untuk memastikan bahwa seluruh fungsi berjalan sesuai dengan kebutuhan. Pengujian dilakukan untuk setiap endpoint API, dengan fokus pada skenario penting seperti autentikasi pengguna, pemrosesan pembayaran, dan pengelolaan inventaris produk. Dalam tahap ini, Bearer Token digunakan untuk autentikasi guna menjamin keamanan akses API yang melibatkan data sensitif pengguna dan transaksi. Selain itu, pengujian dilakukan pada setiap fase pengembangan untuk mendeteksi dan memperbaiki bug secepat mungkin.



Gambar 1. Usecase Diagram



Gambar 2. Entity Relationship Diagram

III. HASIL DAN PEMBAHASAN

Pengembangan backend untuk Website Workit e-commerce Jobhun dilakukan dengan pendekatan berbasis modul menggunakan bahasa pemrograman Golang. Golang dipilih karena kemampuannya dalam menangani beban kerja tinggi dan efisiensi dalam mengelola operasi secara konkuren, sehingga sesuai untuk aplikasi e-commerce yang membutuhkan performa optimal. Beberapa langkah utama dalam metode pengembangan ini mencakup:

A. Struktur Modular Backend

Hasil implementasi menunjukkan bahwa arsitektur modular berhasil memisahkan fungsi utama dari setiap API, seperti user, product, payment, dan order, sehingga mempermudah pengelolaan logika bisnis. Dengan memisahkan handler, repository, usecase, dan route.go pada masing-masing modul, setiap fitur dapat dikembangkan secara independen tanpa mempengaruhi bagian lain dari aplikasi. Hal ini juga memudahkan tim dalam melakukan pembaruan atau perbaikan pada setiap modul sesuai dengan kebutuhan tanpa mengganggu stabilitas sistem secara keseluruhan.

B. Kinerja dan Efisiensi Golang

Penggunaan Golang sebagai bahasa pemrograman backend terbukti memberikan performa tinggi, khususnya dalam menangani permintaan pengguna secara paralel. Bahasa ini dirancang untuk memaksimalkan efisiensi, terutama pada aplikasi yang membutuhkan banyak permintaan dalam waktu singkat. Dengan dukungan pengelolaan proses secara konkuren, Golang berhasil mempertahankan kestabilan sistem meskipun terjadi lonjakan permintaan secara tiba-tiba. Penggunaan struktur goroutine pada Golang juga mempercepat eksekusi kode, yang penting untuk mengoptimalkan waktu respons API dalam menangani transaksi pembelian.

C. Manajemen Data dengan MySQL

Database MySQL yang digunakan terbukti efektif dalam mengelola data pengguna, produk, transaksi, dan pembayaran. Desain skema database yang relasional mampu mengorganisir data dengan baik serta mempercepat pemrosesan query untuk kebutuhan transaksi e-commerce. Efisiensi pada MySQL didukung oleh struktur tabel yang optimal, yang membantu mempercepat waktu akses data dan meminimalkan konflik pada proses pembacaan dan penulisan data. Penggunaan MySQL Workbench juga mempermudah visualisasi dan pemantauan struktur database selama proses pengembangan.

D. Keamanan API dengan Bearer Token

Autentikasi Bearer Token diterapkan untuk menjaga keamanan akses setiap endpoint API. Hasil pengujian menunjukkan bahwa sistem autentikasi ini efektif dalam membatasi akses, terutama untuk endpoint yang memproses data sensitif, seperti informasi pembayaran dan riwayat transaksi. Dengan adanya Bearer Token, API hanya bisa diakses oleh pengguna yang telah terverifikasi, sehingga meminimalkan risiko akses tidak sah. Penerapan token ini memberikan perlindungan tambahan yang dibutuhkan untuk keamanan data pengguna, serta meningkatkan kepercayaan buyer dalam bertransaksi.

E. Penggunaan Docker Compose untuk Containerization

Docker Compose mempermudah pengelolaan lingkungan pengembangan dengan menyediakan container yang terisolasi untuk setiap komponen backend. Dengan containerization, aplikasi dapat dijalankan dalam lingkungan yang seragam, yang membuat proses pengujian dan deploy menjadi lebih efisien dan cepat. Hasil implementasi menunjukkan bahwa Docker Compose meminimalkan masalah kompatibilitas antar perangkat, sehingga aplikasi backend dapat dengan mudah diimplementasikan pada server yang berbeda tanpa memerlukan konfigurasi ulang.

F. Pengujian API dengan Postman

Setiap endpoint API diuji menggunakan Postman untuk memastikan bahwa sistem dapat beroperasi dengan stabil dalam berbagai skenario, termasuk autentikasi pengguna, transaksi produk, dan pemrosesan pembayaran. Pengujian ini mengidentifikasi bug yang potensial pada fase awal pengembangan, sehingga dapat diperbaiki sebelum diterapkan pada server produksi. Hasil pengujian menunjukkan bahwa sistem dapat menangani permintaan API dengan cepat dan konsisten. Pengujian ini juga memperkuat validasi keamanan dan ketepatan respons, sehingga memberikan pengalaman pengguna yang optimal bagi buyer dan admin.[9]

G. Penggunaan Bearer Token untuk Otentikasi

Bearer Token diterapkan dalam sistem backend Website Workit e-commerce Jobhun sebagai metode otentikasi untuk melindungi akses ke endpoint yang sensitif. Pengujian terhadap implementasi Bearer Token menunjukkan bahwa sistem berhasil membatasi akses hanya kepada pengguna yang memiliki token yang valid. Setiap kali pengguna melakukan login, mereka menerima Bearer Token yang perlu disertakan dalam header setiap permintaan API untuk otentikasi. Jika token valid, server memproses permintaan tersebut, tetapi jika token tidak valid atau kedaluwarsa, server akan menolak permintaan dan memberikan respons kesalahan yang sesuai. Hal ini memastikan bahwa hanya pengguna yang sah yang dapat mengakses fitur-fitur seperti transaksi pembelian atau melihat riwayat pemesanan. Penerapan Bearer Token juga memberikan tingkat keamanan yang lebih tinggi dibandingkan dengan penggunaan cookie atau sesi tradisional karena token ini hanya disimpan di header permintaan dan tidak rentan terhadap serangan Cross-Site Request Forgery (CSRF). Implementasi ini terbukti efektif dalam melindungi data pengguna dan menjaga integritas sistem dari potensi ancaman keamanan.

Cari Expert
Jadi Expert
Layanan

Andika Maria
Dokter
Pengantar
Masa

Workit by Jobhun

Sedang mencari tempat atau vendor untuk membuat corporate gift yang unik demi meningkatkan citra perusahaan dan membangun hubungan yang langgeng dengan klien, mitra bisnis, atau karyawan Anda? Temukan solusinya di Workit by Jobhun, di sini kamu dapat memesan dan kustomisasi corporate gift sesuai kebutuhan. Workit by Jobhun merupakan layanan dari Jobhun yang menyediakan corporate gift untuk klien, karyawan, hingga berbagai acara perusahaan.

Mengapa harus memesan di Workit by Jobhun?

Kreativitas tanpa batas
Workit by Jobhun siap untuk bekerjasama dengan Anda dalam menyediakan layanan corporate gift yang sesuai dengan identitas perusahaan. Mulai dari desain yang kreatif hingga personalisasi yang detail, kami memastikan setiap produk mencerminkan keunggulan bisnis Anda.

Memiliki jenis produk yang beragam menyesuaikan budget
Mulai dari merchandise promosi hingga corporate gift karyawan yang fungsional, kami memiliki beragam jenis produk yang siap diorder, dan dapat disesuaikan dengan kebutuhan dan budget perusahaan.

Kualitas bintang lima
Kami menggunakan bahan material yang terbaik dan memperhatikan setiap detail dalam setiap produk. Dengan standar dan kualitas tinggi, corporate gift dari kami bukan hanya sekedar hadiah, tetapi juga investasi jangka panjang untuk meningkatkan citra dan reputasi perusahaan Anda.

Pelayanan yang luar biasa
Kami tidak hanya menyediakan produk berkualitas, tetapi juga memberikan pelayanan yang unggul. Kami berkomitmen untuk memastikan setiap tahap, dari konsep hingga pengiriman, berjalan dengan lancar dan memenuhi harapan Anda dan perusahaan.

Apa saja produk corporate gift yang dapat dipesan?

Notebook

Totebag

Kaos

Lanyard

Kaos

Dan masih banyak lagi!

Gimana cara pemesanannya?
Tertarik bekerjasama dengan kami? Konsultasikan bersama marketing kami melalui [WhatsApp!](#)

Dapatkan produk-produk penunjang produktivitas kerjamu di sini!

Jobhun Productive Book
Rp50.000,-
[Detail produk](#)

Working Kit by Jobhun
Rp180.000,-
[Detail produk](#)

Jobhun Sticker
Rp15.000,-
[Detail produk](#)

Jobhun Totebag
Rp50.000,-
[Detail produk](#)

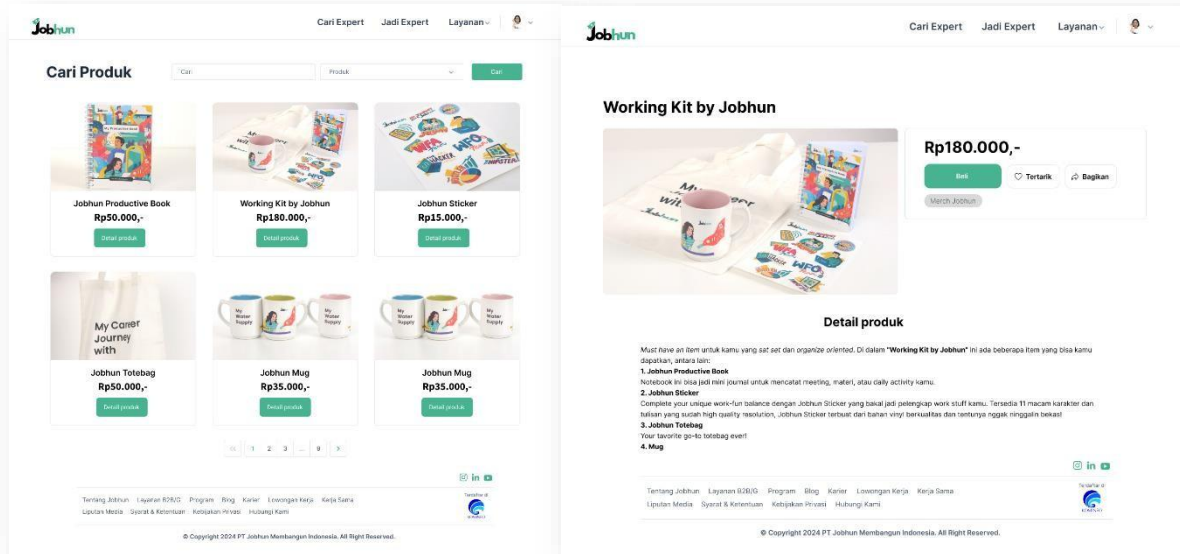
Jobhun Mug
Rp35.000,-
[Detail produk](#)

[Cari produk lainnya →](#)

Tentang Jobhun
Layanan B2B/G
Program
Blog
Karier
Lowongan Kerja
Kerja Sama
Upitan Media
Syarat & Ketentuan
Keijakan Privasi
Hubungi Kami

© Copyright 2024 PT Jobhun Membangun Indonesia. All Right Reserved.

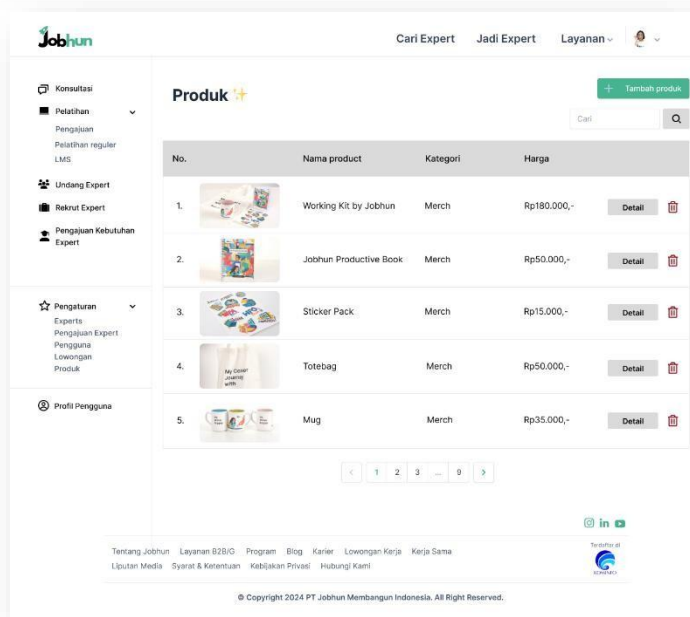
Gambar 3. Homepage Workit Jobhun



Gambar 4. Product Interface Workit Jobhun



Gambar 5. Product API Workit Jobhun



Gambar 6. ADMIN Product Interface Workit Jobhun

```
route.go X
service > modules > user > route.go > ...
1 package user
2
3 import (
4     "github.com/gin-gonic/gin"
5     "github.com/ricnah/workit-be/config"
6     "github.com/ricnah/workit-be/service/middlewares"
7     "github.com/ricnah/workit-be/service/modules/user/handler"
8     "github.com/ricnah/workit-be/service/modules/user/usecase"
9     "github.com/ricnah/workit-be/types/constants"
10 )
11
12 func InitRoutes(v1 *gin.RouterGroup, userCase usecase.IUsecase, cfg *config.Config) {
13     userHandler := handler.UserCreateHandler(userCase)
14
15     moduleRoute := v1.Group("/user")
16
17     moduleRoute.POST("/login", userHandler.AuthLogin)
18     moduleRoute.POST("/create", userHandler.UserCreate)
19
20     adminRoute := moduleRoute.Use(roleCheck(userCase, constants.ROLES_ADMIN))
21     {
22         adminRoute.PUT("/edit", userHandler.UserUpdate)
23         adminRoute.GET("/detail/:id", userHandler.UserGetByID)
24         adminRoute.POST("/search", userHandler.UserSearch)
25     }
26
27     authRoute := moduleRoute.Use(roleCheck(userCase, constants.ROLES_ADMIN, constants.ROLES_BUYER))
28     {
29         authRoute.POST("/get-by-email", userHandler.UserGetByEmail)
30         authRoute.POST("/list-user", userHandler.UserGetAllUser)
31         moduleRoute.GET("/session", userHandler.AuthSession)
32     }
33 }
34
35 func roleCheck(userCase usecase.IUsecase, roles ...constants.Roles) gin.HandlerFunc {
36     return middlewares.CheckRole(userCase, roles)
37 }
```

Gambar 7. User API Workit Jobhun

```
workit-be-ricki
route.go X repository.go
service > modules > order > route.go > ...
1 package order
2
3 import (
4     "github.com/gin-gonic/gin"
5     "github.com/ricnah/workit-be/service/modules/order/handler"
6 )
7
8 func InitRoutes(v1 *gin.RouterGroup, orderHandler *handler.OrderHandler) {
9     orderRoute := v1.Group("/orders")
10
11     {
12         orderRoute.POST("/create", orderHandler.CreateOrder)
13         orderRoute.GET("/list", orderHandler.GetOrders)
14         orderRoute.GET("/:id", orderHandler.GetOrderByID)
15         orderRoute.PUT("/update/:id", orderHandler.UpdateOrder)
16         orderRoute.DELETE("/delete/:id", orderHandler.DeleteOrder)
17     }
18 }
```

Gambar 8. Order API Workit Jobhun

```
workit-be-ricki
route.go X
service > modules > payment > route.go > ...
1 package payment
2
3 import (
4     "github.com/gin-gonic/gin"
5     "github.com/ricnah/workit-be/service/middlewares"
6     "github.com/ricnah/workit-be/service/modules/payment/handler"
7     "github.com/ricnah/workit-be/service/modules/user/usecase"
8     "github.com/ricnah/workit-be/types/constants"
9 )
10
11 func InitRoutes(v1 *gin.RouterGroup, userCase usecase.IUsecase, paymentHandler *handler.PaymentHandler) {
12     paymentRoute := v1.Group("/payments")
13     paymentRoute.Use(middlewares.CheckPaymentGateway())
14
15     {
16         paymentRoute.POST("/create", paymentHandler.CreatePayment) // Client membuat pembayaran saat membeli produk
17         paymentRoute.GET("/", paymentHandler.GetPayments) // Mendapatkan daftar pembayaran client
18         paymentRoute.GET("/:id", paymentHandler.GetPaymentByID) // Mendapatkan detail pembayaran client
19         paymentRoute.POST("/check-expired", paymentHandler.CheckExpiredPayments) // Memeriksa pembayaran yang kadaluarsa
20     }
21
22     // Middleware khusus untuk admin
23     adminRoute := paymentRoute.Use(roleCheck(userCase, constants.ROLES_ADMIN))
24     {
25         adminRoute.PUT("/update/:id", paymentHandler.UpdatePayment)
26         adminRoute.DELETE("/delete/:id", paymentHandler.DeletePayment)
27     }
28 }
29
30 func roleCheck(userCase usecase.IUsecase, roles ...constants.Roles) gin.HandlerFunc {
31     return middlewares.CheckRole(userCase, roles)
32 }
33
34 }
```

Gambar 8. Payment API Workit Jobhun

IV. SIMPULAN

Penelitian ini menunjukkan bahwa penerapan bahasa pemrograman Golang dalam pengembangan backend untuk Website Workit e-commerce Jobhun memberikan banyak keuntungan, terutama dalam hal performa dan efisiensi. Golang mampu menangani permintaan tinggi secara konkuren, memastikan API memberikan respons cepat dan optimal. Penggunaan MySQL sebagai basis data utama terbukti efektif dalam mengelola data pengguna, produk, transaksi, dan pembayaran dengan efisien. Desain skema relasional yang tepat memungkinkan akses data yang cepat dan meminimalkan konflik saat membaca atau menulis data. Keamanan sistem juga diperkuat dengan implementasi Bearer Token untuk autentikasi, yang membatasi akses API hanya pada pengguna yang terverifikasi, sehingga melindungi transaksi sensitif dan data pengguna.

Docker Compose memfasilitasi containerization, memungkinkan pengelolaan aplikasi dalam lingkungan pengembangan terisolasi yang mengurangi masalah kompatibilitas antar perangkat dan mempermudah deployment aplikasi pada server yang berbeda. Pengujian API menggunakan Postman memastikan setiap endpoint berfungsi stabil dan sesuai kebutuhan, memperkuat validasi keamanan dan kinerja API dalam berbagai skenario. Secara keseluruhan, implementasi backend dengan Golang, MySQL, Bearer Token, Docker Compose, dan Postman membuktikan bahwa sistem ini memberikan performa yang stabil, efisien, dan aman, serta dapat menjadi model yang baik bagi pengembangan aplikasi e-commerce yang mengutamakan kecepatan, keamanan, dan skalabilitas.

UCAPAN TERIMA KASIH

Terima kasih yang sebesar-besarnya kepada PT Jobhun Membangun Indonesia atas dukungan dan kepercayaan yang diberikan selama proses pengembangan Backend Website Workit ini. Terima kasih kepada Bapak Andreas Peter Chandra, S. Kom., sebagai pembimbing lapangan, yang telah memberikan arahan dan dukungan selama pelaksanaan penelitian disana. Pengalaman ini sangat berharga, memberikan wawasan yang mendalam tentang dunia kerja serta tantangan nyata dalam pengembangan sistem informasi di lingkungan industri.

Terima kasih kepada Universitas Muhammadiyah Sidoarjo (UMSIDA), terlebih itu khususnya kepada Ibu Cindy Taurusta, S.ST., M.T., selaku dosen pembimbing, yang telah memberikan bimbingan, arahan, dan dukungan yang sangat berarti sepanjang penyusunan tugas akhir ini. Semua kontribusi dan bantuan yang diberikan sangat berharga dan menjadi motivasi besar dalam menyelesaikan penelitian ini.

REFERENSI

- [1] U. Brawijaya, R. Alfarisi, B. Rahayudi, and D. Pramono, "Fakultas Ilmu Komputer Pengembangan Aplikasi Web E-Commerce dan Donasi (Studi Kasus : Green Welfare Indonesia)," 2017. [Online]. Available: <http://j-ptiik.ub.ac.id>
- [2] Y. M. Tripathi and M. P. Modi, "Real-Time Chat Application."
- [3] A. Himawan and A. Dwi Putra, "APLIKASI DAILY CALORIE BERBASIS WEB MENGGUNAKAN RESTFUL API GOLANG."
- [4] Y. Muharam and T. Hidayat, "PENGEMBANGAN APLIKASI BACK-END E-COMMERCE MENGGUNAKAN REST API GOLANG UNTUK OPTIMALISASI KINERJA SERVER."
- [5] M. Lodder, "Token Based Authentication and Authorization with Zero-Token Based Authentication and Authorization with Zero-Knowledge Proofs for Enhancing Web API Security and Privacy Knowledge Proofs for Enhancing Web API Security and Privacy." [Online]. Available: <https://scholar.dsu.edu/theses/425>
- [6] M. Avatara and R. Tan, "Implementasi Framework Gin dan gRPC pada Pengembangan Back-end Web," 2024.
- [7] A. A. Kroons and C. Dewi, "PENGEMBANGAN DASHBOARD TRIVY BERBASIS WEBSITE MENGGUNAKAN REACT JS DAN GOLANG," *Jurnal Indonesia: Manajemen Informatika dan Komunikasi*, vol. 4, no. 3, pp. 1037–1049, Sep. 2023, doi: 10.35870/jimik.v4i3.295.
- [8] D. Dykstra, M. Altunay, and J. Teheran, "Secure Command Line Solution for Token-based Authentication," *EPJ Web Conf*, vol. 251, p. 02036, 2021, doi: 10.1051/epjconf/202125102036.
- [9] R. Annisa, R. Agung Ananda, and W. E. Sulistiono, "IMPLEMENTASI GOLANG CLEAN ARCHITECTURE PADA PERANCANGAN BACKEND POINT OF SALES WEBSITE," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 2, pp. 2830–7062, doi: 10.23960/jitet.v12i2.4668.
- [10] T. Muszyński and G. Koziel, "A security analysis of authentication and authorization implemented in web applications based on the REST architecture," *Journal of Computer Sciences Institute*, 2020, doi: 10.35784/jcsi.1925
- [11] "Towards secure APIs: A survey on RESTful API vulnerability detection," *Computers, Materials & Continua*,

- [12] S.-W. Jang and S.-H. Lee, "Vulnerabilities and encryption applications of JWT-based authentication methods," *Journal of Information Systems Engineering and Management*, vol. 10, no. 8s, 2025, doi: 10.52783/jisem.v10i8s.1055.
- [13] H. Hendra, A. Awan, Waisen, W. Wilianto, and Y. Yudi, "Memperkuat autentikasi dan integritas data REST-API menggunakan token HMAC SHA-256," *Jurnal Minfo Polgan*, vol. 13, no. 2, pp. 2189–2197, 2024, doi: 10.33395/jmp.v13i2.14406.
- [14] A. W. Syahroni, N. P. Dewi, N. Ramadhani, Ubaidi, and B. Said, "Security testing of web-based application back end using black box testing method," *Journal PROCESSOR*, vol. 19, no. 2, 2024, doi: 10.33998/processor.2024.19.2.1752.
- [15] V. B. Sanjaya, S. Rahman, and H. Surasa, "Perancangan application programming interface pada website Saheb menggunakan arsitektur REST," *KHARISMA Tech*, vol. 20, no. 1, pp. 29–43, 2025, doi: 10.55645/kharismatech.v20i1.529.