Jurnal Pendidikan dan Teknologi Indonesia (JPTI)
Vol. 5, No. 11, November 2025, Hal. 3505-3517
DOI: https://doi.org/10.52436/1.jpti.1183
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Pengembangan Backend Pada Sistem Pembayaran Virtual Account dengan REST API Menggunakan Metode Extreme Programming di UHAMKA

Irsana Ahmad[*1], Ade Davy Wiranata[2]

[1,2]Teknik Informatika, Universitas Muhammadiyah Prof. DR. HAMKA, Indonesia
Email: [1]irsanaahmad@uhamka.ac.id, [2]adedavy@uhamka.ac.id

Abstrak

Penelitian ini mengembangkan backend API untuk sistem pembayaran virtual account (VA) menggunakan pendekatan REST API dan metode Extreme Programming (XP). Bahasa pemrograman Go (Golang) dipilih karena kemampuannya dalam menangani permintaan secara efisien dan paralel. Sistem dirancang untuk mendukung proses pembayaran pendidikan di Universitas Muhammadiyah Prof. Dr. Hamka (UHAMKA). Proses pengembangan mengikuti prinsip XP yang menekankan iterasi pendek, kolaborasi intensif, dan pengujian berkelanjutan. Backend terdiri dari dua layanan utama, yaitu inquiry untuk pengecekan tagihan mahasiswa dan payment untuk pencatatan transaksi pembayaran dari bank. Integrasi dilakukan dengan database internal agar proses pencatatan tagihan dan transaksi berjalan otomatis dan realtime. Pengujian menunjukkan bahwa sistem mampu merespons permintaan inquiry dan payment dari bank secara cepat dan akurat. Penerapan sistem ini meningkatkan efisiensi pengelolaan pembayaran, meminimalkan kesalahan pencatatan, serta mendukung otomasi layanan administrasi akademik. Hasil penelitian ini menunjukkan bahwa kombinasi metode XP dan arsitektur REST API efektif dalam membangun layanan backend yang andal dan terintegrasi untuk kebutuhan pembayaran pendidikan.

Kata kunci: *Backend, Extreme Programming, Golang, REST API, Sistem Pembayaran, Virtual Account*

Backend Development in Virtual Account Payment Systems with REST API Using Extreme Programming Methods at UHAMKA

Abstract

This study develops a backend API for a virtual account (VA) payment system using the REST API approach and the Extreme Programming (XP) methodology. The backend is implemented using the Go (Golang) programming language due to its efficiency and ability to handle concurrent processes. The system is designed to support education payment services at Universitas Muhammadiyah Prof. Dr. Hamka (UHAMKA). The development process follows XP principles, emphasizing short iterations, intensive collaboration, and continuous testing. The backend consists of two main services: inquiry, for retrieving student billing information, and payment, for recording transactions from partner banks. Integration with the university's internal database enables automated and real-time processing of billing and transaction data. Testing results show that the system accurately and efficiently responds to inquiry and payment requests from banks. The implementation of this backend improves payment management efficiency, reduces the potential for transaction errors, and supports automation in academic administrative services. The results demonstrate that combining the XP methodology with a REST architecture is effective for developing a reliable and integrated backend service for education payment systems.

Keywords: *Backend, Extreme Programming, Golang, Payment System, REST API, Virtual Account*

1. PENDAHULUAN

Universitas Muhammadiyah Prof. Dr. Hamka (UHAMKA) merupakan perguruan tinggi swasta milik Persyarikatan Muhammadiyah yang berada di Jakarta. Seiring dengan kemajuan teknologi, sistem pembayaran di berbagai sektor terus mengalami perkembangan, termasuk dalam dunia pendidikan. Universitas Muhammadiyah Prof. Dr. HAMKA (UHAMKA), sebagai salah satu institusi pendidikan tinggi, telah mengadopsi sistem pembayaran digital melalui *virtual account* untuk memfasilitasi pembayaran uang kuliah oleh mahasiswa. Dan metode pembayaran melalui *virtual account* mempunyai kelebihan praktis, simple tanpa harus melakukan konfirmasi pembayaran, dan dapat dipakai kapanpun dan dimanapun [1].

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Hingga saat ini, UHAMKA telah menjalin kerja sama dengan dua bank, yaitu Bank Muamalat dan Bank Syariah Indonesia (BSI), dalam menyediakan layanan pembayaran melalui virtual account. Namun, sejalan dengan kebijakan dari Pimpinan Pusat (PP) Muhammadiyah, Muhammadiyah memutuskan untuk mengalihkan dananya dengan menyebarkan ke sejumlah bank syariah yang beroperasi di Indonesia. Keputusan pengalihan dana tersebut tertuang dalam memo Muhammadiyah Nomor 320/1.0/A/2024 [2]. Oleh karena itu, UHAMKA sebagai salah satu amal usaha Muhammadiyah turut menyesuaikan kebijakan tersebut dengan memperluas kerja sama layanan pembayaran melalui *virtual account* dengan bank syariah lain yang beroperasi di Indonesia. Langkah ini dilakukan untuk mendukung kebijakan pusat dan memberikan lebih banyak pilihan pembayaran kepada mahasiswa.

Sistem yang ada perlu dikembangkan untuk integrasi dengan bank baru tanpa mengganggu layanan pembayaran yang sudah berjalan. Solusinya adalah membangun layanan berbasis REST API menggunakan bahasa Go. REST API dipilih karena fleksibel, kompatibel lintas platform, dan mudah diskalakan [3][4]. Dengan Go, layanan ini diharapkan memiliki performa tinggi, efisien, dan mudah dipelihara [3].

REST API merupakan pendekatan arsitektur dalam pengembangan layanan web yang memungkinkan komunikasi data antara sistem internal dengan sistem eksternal (seperti API bank) melalui protokol HTTP. REST API memiliki kelebihan berupa kemudahan integrasi lintas platform, dukungan untuk berbagai format data (terutama JSON), serta kemudahan dalam pemeliharaan dan pengembangan berkelanjutan [5]. Sementara itu, Bahasa Go dipilih karena performanya yang tinggi, efisiensi dalam pengelolaan memori, serta kemampuannya dalam menangani *concurrent request* secara optimal [3], yang sangat dibutuhkan dalam sistem transaksi seperti layanan VA. Dengan menggunakan bahasa pemrograman Go, layanan API yang dikembangkan diharapkan memiliki performa yang optimal, kode yang efisien, dan mudah untuk dipelihara.

Beberapa penelitian terdahulu yang relevan dengan penelitian ini dirangkum sebagai berikut: penelitian oleh Rahmad Adhi Sasono dkk. pada tahun 2025 yang berjudul "Optimasi Web Service REST Pada Backend Aplikasi Prospect Menggunakan Metode Extreme Programming" berhasil mengoptimalkan performa integrasi data dan kecepatan respons sistem melalui penerapan arsitektur REST dan metode pengembangan Extreme Programming (XP). Penggunaan REST API menggantikan RPC-API yang sebelumnya digunakan, menghasilkan layanan yang lebih efisien, fleksibel, serta mempercepat proses pengembangan backend aplikasi prospect [6]. Selanjutnnya Penelitian oleh Fadel Pamungkas dan Hari Setiaji pada tahun 2024 yang berjudul "Implementasi Clean Architecture pada Pembuatan API Menggunakan Golang", penelitian ini berhasil membangun REST API dengan struktur sistem yang lebih terorganisir dan mudah dipahami, serta menunjukkan hasil response time yang efisien, yaitu rata-rata 22–79 ms tergantung ukuran data [7]. Selanjutnya penelitian oleh Irfan Rizq Dzaky Muhammad, dan Irving V. Paputungan pada tahun 2024 yang berjudul "Pengembangan Backend Server Berbasis Arsitektur REST API pada Sistem Transfer Dompet Digital" Penelitian ini berhasil mengembangkan sistem server-side untuk transfer dana antar dompet digital menggunakan arsitektur REST API. Dengan memanfaatkan framework NestJS, database MySQL, serta metode pengembangan *Extreme Programming* (XP), penelitian ini menghasilkan API yang dapat digunakan oleh pengembang frontend dan *mobile*. Seluruh API diuji menggunakan *BlackBox Testing* dan dinyatakan berfungsi dengan baik, mencakup kebutuhan sistem seperti autentikasi, pengelolaan pengguna, transaksi, pengembalian dana, dan manajemen dompet digital [4]. Selanjutnya ada penelitian oleh Juan Angela Alma, dan Agus Prihanto pada tahun 2024 yang berjudul "Implementasi Backend System Untuk Integrasi Payment Gateway Pada Sistem Pembayaran Kost Menggunakan Express.js" Penelitian ini berhasil mengimplementasikan sistem backend menggunakan Express.js untuk mengintegrasikan *payment gateway* Midtrans dalam sistem pembayaran kost. *Backend system* dirancang untuk meningkatkan keamanan dengan menyimpan client key di sisi server, sehingga tidak terekspos ke frontend [8]. Selanjutnya ada penelitian oleh Nurfiqih, dan Intan kumalasari pada tahun 2023 yang berjudul "Implementasi Host To Host Bca Untuk Transaksi Virtual Account Bpr Niaga Menggunakan Restful Api" Penelitian ini menghasilkan aplikasi RESTful API menggunakan framework Spring Boot dengan bahasa pemrograman Java yang mampu menerima pesan dalam format JSON dan mengonversinya menjadi format ISO8583 untuk dikirim ke core banking BPR Niaga. Sistem ini memungkinkan nasabah BPR Niaga untuk melakukan penyetoran secara *online* melalui seluruh kanal yang dimiliki oleh Bank BCA. Proses pengujian menunjukkan bahwa fitur-fitur seperti *token access*, *inquiry*, dan *payment* berfungsi dengan baik. Aplikasi ini mendukung transaksi *virtual account* secara efisien dan memperluas akses nasabah terhadap layanan perbankan modern [9]. Selanjutnya ada penelitian oleh Dimas Kita Ladiba, Weda Adistianaya Dewa, dan Samsul Arifin pada tahun 2021 yang berjudul "Analisis dan Pengembangan API Siakad Menggunakan Arsitektur Restful Web Service pada Infrastruktur Microservice" Penelitian ini menghasilkan pengembangan API berbasis arsitektur Restful Web Service yang diterapkan pada infrastruktur microservice untuk mendukung Sistem Informasi Akademik (SIAKAD) di Kampus STIMATA. Hasil pengujian menunjukkan Sistem API berhasil mempermudah pengembangan aplikasi multiplatform dengan rata-rata indeks interpretasi sebesar 95,3% dan Pengujian *blackbox* memastikan bahwa sistem sudah dapat digunakan dengan baik meskipun terdapat beberapa kesalahan minor pada implementasi tertentu [10]. Selanjutnya ada Penelitian oleh Moh. Anshori Aris Widya dan Nurul Aini pada tahun 2021 yang berjudul "Design of a Student Payment System Based on Virtual Account (Case Study at SMK NU Al-

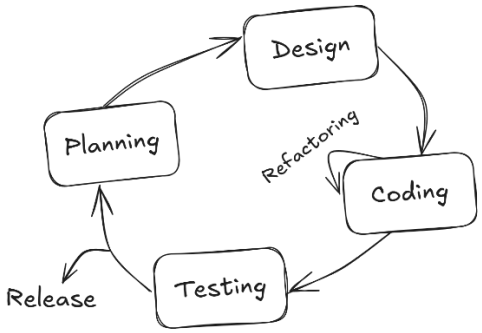
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Hidayah Ngimbang)" yang berhasil mengembangkan sistem pembayaran pendidikan berbasis virtual account (BRIVA) yang terintegrasi secara daring. Sistem ini terbukti lebih efisien dibandingkan metode konvensional karena mampu mempercepat proses pembayaran, memudahkan pelaporan keuangan, serta mengurangi antrean dan risiko kesalahan pencatatan [11]. Selanjutnya penelitian oleh Erry Julio dan Magdalena A. Ineke Pakereng pada tahun 2021 yang berjudul "Implementasi API Payment Gateway Menggunakan Arsitektur Microservice" berhasil membangun sistem payment gateway berbasis microservice yang dapat melakukan routing transaksi ke berbagai bank secara otomatis. Arsitektur ini terbukti meningkatkan fleksibilitas, skalabilitas, dan keamanan layanan melalui penggunaan JWT dan whitelist IP, serta memudahkan integrasi dengan bank baru melalui pemisahan layanan per bank [12]. Selanjutnya ada penelitian oleh Nina Wulandari, Argo Wibowo, dan Budi Susanto pada tahun 2021 yang berjudul "Penerapan RESTful API untuk Membangun Program Pembayaran Piutang Menggunakan Otentikasi OAuth 2.0", penelitian ini berhasil membangun sistem pembayaran piutang menggunakan RESTful API dan membuktikan bahwa metode otentikasi OAuth 2.0 memberikan performa yang lebih stabil dibandingkan Basic Auth, dilihat dari response time dan jumlah request per detik dalam uji beban [13]. Selanjutnya ada penelitian oleh Padeli, Eduard Hotman Purba, dan Bonari Simanjuntak pada tahun 2020 yang berjudul "Analisa Pembayaran Perkuliahan dengan Virtual Account pada Universitas Raharja" Penelitian ini menganalisis penggunaan sistem *Virtual Account* sebagai metode pembayaran perkuliahan di Universitas Raharja. Hasilnya menunjukkan bahwa penerapan *Virtual Account* dapat meningkatkan efektivitas dan efisiensi pembayaran, baik bagi mahasiswa maupun pihak kampus. Sistem ini memudahkan pengelolaan data pembayaran secara *real-time* dan mengurangi risiko kesalahan dibandingkan dengan sistem manual sebelumnya. Penelitian ini merekomendasikan kolaborasi dengan pihak bank untuk mengintegrasikan API *Virtual Account* ke dalam sistem kampus [1].

Dengan membangun backend yang berbasis REST API, UHAMKA dapat mengembangkan sistem pembayaran VA yang dapat diintegrasikan dengan berbagai bank baru secara bertahap, tanpa harus mengubah struktur utama sistem. Hal ini akan memberikan fleksibilitas dalam pengelolaan layanan pembayaran dan mendukung skalabilitas sistem ke depannya.

Penelitian ini bertujuan untuk menganalisis bagaimana penerapan arsitektur REST API dapat mendukung sistem pembayaran digital di UHAMKA. Selain itu, penelitian ini diharapkan menjadi dokumentasi teknis dalam pembuatan REST API untuk layanan virtual account, yang dapat dijadikan acuan dalam pengembangan sistem serupa. Hasil penelitian ini diharapkan memberikan kontribusi nyata terhadap pengembangan sistem pembayaran di UHAMKA secara berkelanjutan, sekaligus memperkuat ekosistem teknologi informasi yang ada di lingkungan kampus.

2. METODE PENELITIAN



Gambar 1. Alur Metode Extreme Programming (XP)

Extreme Programming (XP) adalah salah satu metode *Agile* yang paling banyak digunakan sejak pertama kali diperkenalkan oleh Kent Beck pada tahun 1996 hingga saat ini. XP bertujuan menciptakan sebuah model proses yang sederhana dan ringan, sehingga memungkinkan tim pengembang beradaptasi dengan cepat terhadap perubahan kebutuhan. Dibandingkan dengan kerangka kerja *Agile* lainnya, XP dikenal sebagai metodologi yang paling terfokus dan spesifik dalam menerapkan praktik rekayasa perangkat lunak yang baik dan disiplin [14].

Extreme Programming (XP) adalah salah satu metode dalam SDLC yang menekankan pada iterasi cepat, umpan balik langsung, dan kolaborasi tim yang erat. XP cocok digunakan dalam pengembangan aplikasi modern, termasuk pengembangan API karena fleksibilitasnya terhadap perubahan kebutuhan [15]. Perancangan sistem API dilakukan dengan menggunakan pendekatan Extreme Programming (XP), Pendekatan ini sangat sesuai untuk pengembangan backend karena memungkinkan penyesuaian secara dinamis terhadap perubahan kebutuhan sistem.

Gambar 1 menjelaskan siklus proses metode Extreme Programming yang diterapkan: dimulai dari Planning,

p-ISSN: 2775-4227
e-ISSN: 2775-4219

dilanjutkan Design, Coding, Testing, dan diakhiri dengan Release. Setiap iterasi bersifat berulang dengan Refactoring sebagai bagian dari perbaikan berkelanjutan. Tahapan XP yang diimplementasikan dalam penelitian ini ditunjukkan pada Gambar 1 dan dijelaskan sebagai berikut:

1) Perencanaan (*Planning*) merupakan Fase awal dalam metode *Extreme Programming (XP)* adalah tahap perencanaan, di mana terjadi kolaborasi erat antara pengembang dan pelanggan. Pada tahap ini, kedua belah pihak bekerja sama untuk merumuskan kebutuhan sistem, menetapkan fitur-fitur utama, menentukan fungsionalitas yang harus disediakan, serta memperkirakan waktu penyelesaian proyek. Pengembang mulai dengan mengumpulkan kebutuhan sistem dalam bentuk cerita pengguna (*user stories*), yang menggambarkan fitur dari sudut pandang pengguna [14]. Tahapan perencanaan dilakukan dengan merancang kebutuhan API yang akan digunakan. Backend developer menyusun daftar endpoint yang diperlukan serta fungsionalitas yang harus didukung oleh API.
2) Perancangan (*Design*) merupakan Kegiatan pemodelan sistem dilakukan berdasarkan hasil analisis kebutuhan, yang kemudian divisualisasikan menggunakan *Unified Modeling Language (UML)*. UML ini mencakup berbagai jenis diagram, di antaranya *Use Case Diagram* untuk menggambarkan interaksi antara pengguna dan sistem, serta *Activity Diagram* yang menunjukkan alur aktivitas atau proses bisnis dalam sistem. Sementara itu, untuk pemodelan basis data digunakan *Entity Relationship Diagram (ERD)* [16].
3) Pengkodean (*Coding*) merupakan Tahap implementasi merupakan proses penerapan dari hasil desain model sistem ke dalam bentuk kode program yang nyata. Hasil dari proses ini adalah *prototipe* yang dapat dijalankan dan dievaluasi lebih lanjut [14]. pengembangan API ini menggunakan framework Echo dengan bahasa pemograman Go Language (Golang).
4) Pengujian (*Testing*) merupakan Fase terakhir dalam metode Extreme Programming (XP) adalah pengujian, di mana setiap kode harus dilengkapi dengan *unit test* dan wajib lolos pengujian tanpa error sebelum diimplementasikan ke lingkungan pengguna. Jika ditemukan masalah, pengembang akan melakukan *refactoring* untuk memperbaiki kode tanpa mengubah fungsinya [16]. Pengujian dilakukan terhadap dua endpoint utama: /inquiry dan /payment menggunakan metode black box testing. Setiap endpoint diuji sebanyak 10 kali untuk masing-masing dari tiga skenario: input valid, input tidak valid, dan input kosong. Pengujian dilakukan dengan Postman dan hasilnya dicatat berdasarkan status HTTP respons serta isi body respons JSON. Suatu pengujian dianggap berhasil jika sistem memberikan respons yang sesuai ekspektasi (kode status 200/400/500) dan struktur respons JSON lengkap tanpa error.

Selain itu, Basis data menggunakan SQL Server. Kode editor menggunakan Visual Studio Code. Pengujian API dilakukan menggunakan Postman pada lingkungan server lokal yang dijalankan di laptop.

3. HASIL DAN PEMBAHASAN

3.1. Hasil

3.1.1. Perancangan (*Planning*)

Tahapan perencanaan dilakukan dengan merancang kebutuhan API yang akan digunakan. Berdasarkan tabel 1 kolom Fitur berisi identifikasi berdasarkan kebutuhan sistem yang berkaitan langsung dengan proses bisnis. Kebutuhan merujuk pada metode HTTP yang digunakan dalam pemanggilan API, sedangkan Deskripsi menjelaskan fungsi dari masing-masing fitur API tersebut.

Tabel 1. Kebutuhan Fungsional API

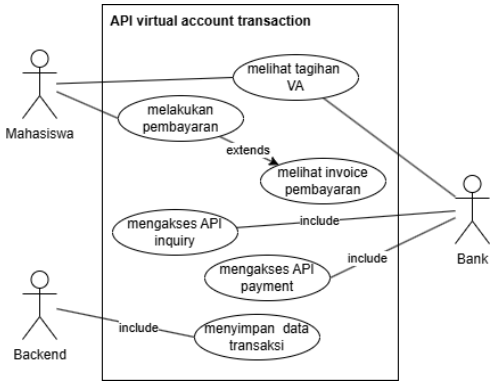
Fitur	Kebutuhan	Deskripsi
Inquiry	POST	API untuk proses pengecekan tagihan pembayaran
Payment	POST	API untuk proses pembayaran

3.1.2. Perancangan (Design)

a) Unified Modeling Language (UML)
UML (*Unified Modeling Language*) merupakan bahasa pemodelan standar yang digunakan untuk merancang, menggambarkan, dan mendokumentasikan sistem informasi berbasis objek dengan menggunakan simbol atau notasi yang telah ditetapkan [17].

p-ISSN: 2775-4227
e-ISSN: 2775-4219

- Use Case Diagram



Gambar 2. Use Case Diagram Virtual Account Transaction

Use Case Diagram adalah salah satu jenis diagram dalam UML yang menunjukkan interaksi antara aktor (pengguna atau sistem lain) dengan sistem, serta menjelaskan bentuk interaksi atau layanan yang tersedia bagi pengguna [18]. Gambar 2 menggambarkan proses interaksi antara mahasiswa dan sistem API backend virtual account dalam melakukan transaksi pembayaran pendidikan. Mahasiswa dapat melihat virtual account (VA) dan tagihan yang harus dibayar. Setelah itu, mahasiswa melakukan proses pembayaran, yang mencakup dua aktivitas utama, yaitu Mengakses API inquiry untuk memvalidasi tagihan dan Mengakses API payment untuk melakukan pembayaran Setelah pembayaran berhasil, sistem akan menyimpan data transaksi ke dalam basis data. Mahasiswa juga memiliki opsi untuk melihat invoice pembayaran, yang merupakan ekstensi dari proses pembayaran.

- Activity Diagram

Gambar 3. Activity Diagram VA Transaction

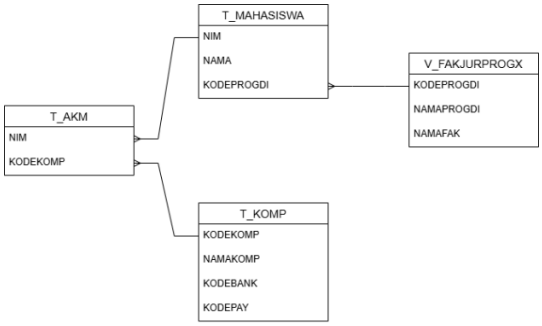
Activity Diagram digunakan untuk memodelkan alur kegiatan dalam perangkat lunak, di mana urutan proses digambarkan secara vertikal sesuai dengan jalannya aktivitas [18]. dari Gambar 3 dapat diketahui Dimana Mahasiswa adalah Pihak yang melakukan pembayaran tagihan kuliah melalui VA. Bank adalah Sistem perantara yang menerima permintaan pembayaran dari mahasiswa dan meneruskannya ke backend kampus. Backend adalah Sistem milik kampus yang menyimpan dan mengelola data tagihan dan pembayaran mahasiswa. Langkah pertama mahasiswa melihat tagihan dan nomor VA pada halaman akademik yang disediakan kampus, setelahnya Mahasiswa Melakukan Pembayaran, lalu Bank Melakukan Inquiry ke Backend, lalu Backend Melakukan Pengecekan Data, lalu Bank Menerima Response Inquiry, lalu Bank Melakukan API Payment ke Backend, lalu Backend Menyimpan Transaksi, lalu Backend Memberikan Response Payment, lalu Bank Menampilkan Hasil Payment, terakhir Mahasiswa Mendapatkan Bukti Pembayaran.

b) Entity Relationship Diagram (ERD)
Entity Relationship Diagram (ERD) merupakan sebuah diagram dengan notasi grafis yang digunakan dalam perancangan basis data untuk menggambarkan hubungan antar data. ERD berfungsi sebagai alat bantu dalam

p-ISSN: 2775-4227
e-ISSN: 2775-4219

proses perancangan database serta memberikan visualisasi mengenai cara kerja database yang akan dibangun [19].

- Inquiry



Gambar 4. ERD Inquiry

Gambar 4 menggambarkan proses inquiry pembayaran virtual account mahasiswa, sistem melakukan pengecekan terhadap data mahasiswa, program studi, dan tagihan yang dimiliki. Untuk mendukung proses tersebut. Tabel T_MAHASISWA untuk Menyimpan data dasar mahasiswa (NIM, Nama, Kode Program Studi), Berelasi dengan V_FAKJURPROGX (melalui KODEPROGDI) untuk mendapatkan detail program studi/fakultas mahasiswa dan Berelasi dengan T_AKM (melalui NIM) untuk menunjukkan tagihan/komponen pembayaran yang dimiliki mahasiswa. Selanjutnya V_FAKJURPROGX untuk Menyediakan informasi detail Program Studi dan Fakultas. Lalu tabel T_AKM untuk Mencatat komponen-komponen pembayaran yang menjadi tagihan untuk setiap mahasiswa (misal SPP semester ini), Berelasi dengan T_KOMP (melalui KODEKOMP) untuk menjelaskan jenis komponen tagihan tersebut (misal: BOP, SKS). Terakhir tabel T_KOMP untuk Menyimpan daftar dan detail dari setiap jenis komponen pembayaran (nama komponen, bank terkait, kode pembayaran).

- Payment

Gambar 5. ERD Payment

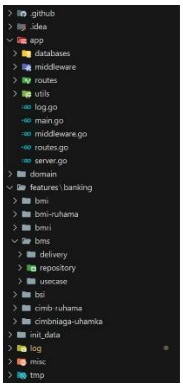
Setelah mahasiswa melakukan pembayaran melalui virtual account, sistem akan mencatat informasi transaksi pembayaran tersebut untuk keperluan rekonsiliasi dan verifikasi. Berdasarkan Gambar 5 tabel T_MAHASISWA Merepresentasikan mahasiswa yang melakukan pembayaran, Berelasi dengan tbl_payGateway (melalui NIM) untuk mencatat bahwa transaksi pembayaran ini dilakukan oleh/untuk mahasiswa tersebut. Lalu tabel T_CMBKOMP untuk Menyimpan kode bank yang terlibat dalam pembayaran, Berelasi dengan tbl_payGateway (melalui KODEBANK) untuk menunjukkan bank mana yang memproses transaksi. Terakhir tabel tbl_payGateway untuk Tabel utama untuk mencatat setiap transaksi pembayaran yang melalui gateway, termasuk detail seperti jumlah, tanggal, nomor VA, status, dan lain-lain.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

3.1.3. Pengkodean (Coding)

Proses implementasi API dilakukan sesuai hasil perancangan dengan menggunakan bahasa pemrograman Go. Definisi Go sebagai bahasa pemrograman mencakup kemampuan untuk mengembangkan perangkat lunak yang efisien dan cepat, dengan sintaks yang mudah dipahami [20]. Go menggunakan pendekatan konkurensi yang ringan untuk mengatasi masalah skalabilitas, serta menyediakan garbage-collector yang efisien untuk mengelola memori [3]. Pada penelitian ini juga menggunakan framework Echo. Framework merupakan kerangka kerja berisi kumpulan fungsi yang siap digunakan untuk tujuan tertentu dan framework Echo adalah kerangka kerja yang kuat dan serbaguna untuk membangun aplikasi yang skalabel dan berperforma tinggi menggunakan bahasa pemrograman Go [21]. Penulisan kode mengikuti prinsip clean code agar mudah dipelihara dan dikembangkan.

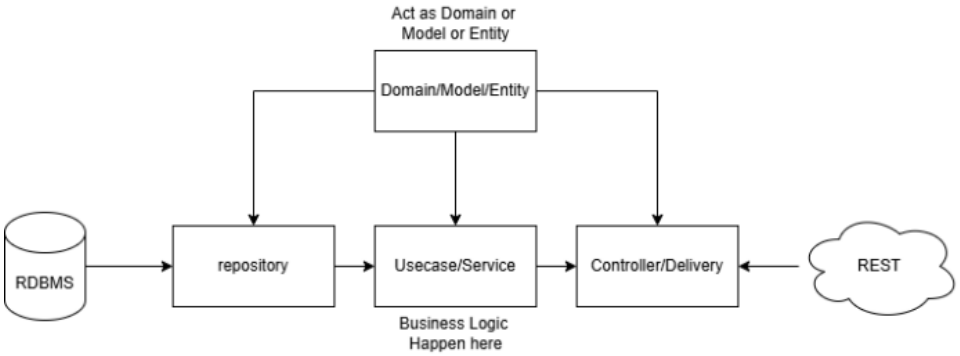
a) Struktur Folder



Gambar 6. Struktur Folder

Pada awal pengembangan, proyek Go diinisialisasi menggunakan perintah *go mod init*, yang akan membuat file *go.mod* sebagai penanda modul utama aplikasi. Selanjutnya, menyusun struktur folder yang rapi dan terorganisir berdasarkan pendekatan *clean architecture*, untuk memisahkan tanggung jawab antar komponen seperti yang digambarkan pada Gambar 6. Folder utama seperti *app/* dibuat untuk menyimpan file konfigurasi dan pengaturan inti aplikasi. Di dalamnya terdapat file *main.go*, yang berperan sebagai *entry point*. Di dalam *main.go*, aplikasi akan memanggil fungsi inisialisasi yang berada di file *server.go*. Lalu framework Echo diinstal dengan *go get github.com/labstack/echo/v4.*

b) Clean Architecture



Gambar 7. Architecture Component Diagram

Diagram pada Gambar 7 menggambarkan arsitektur pengembangan backend sistem virtual account yang mengadopsi pola Clean Architecture. *Clean Architecture* adalah pendekatan arsitektur perangkat lunak yang menekankan pemisahan tanggung jawab dalam beberapa lapisan, sehingga sistem menjadi lebih mudah dipahami, diuji, dan dikembangkan. Pada proyek ini memiliki 4 layer utama yaitu *Models Layer, Repository Layer, Use Case Layer,* dan *Delivery Layer* [7].

1) Models Layer
Berisi objek domain atau business model inti dari aplikasi. Pada proyek saya berisi struct go contohnya seperti tabel 2.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Tabel 2. Kebutuhan Fungsional API

Kode Program

```
type RequestBodyBank struct {
    REFFNUMBER  string `json:"reffnumber"`
    TIMESTAMP   string `json:"timestamp"`
    CHANNELID   string `json:"channelID"`
    COMPANYCODE string `json:"companyCode"`
    CHANNELED   string `json:"channeled"`
    AMOUNT      string `json:"amount"`
    NUMBERVA    string `json:"numberVa"`
    NIM         string `json:"nim"`
    TYPEINQ     string `json:"typeInq"`
    PROCODE     string `json:"procode"`
    REKDEBT     string `json:"rekdebt"`
    TERMINALID  string `json:"terminalID"`
    TID         string `json:"tid"`
    TRXDATETIME string `json:"trxDateTime"`
    USERNAME    string `json:"username"`
    PASSWORD    string `json:"Password"`
    KEY         string `json:"key"`
    SOF         string `json:"sof"`
}
```

2) Repository Layer

Repository berfungsi sebagai lapisan penghubung antara sistem dengan database, baik relational (RDBMS) maupun NoSQL. Komponen ini menangani semua operasi CRUD, dan dipanggil oleh usecase untuk mengambil atau menyimpan data, contohnya seperti tabel 3.

Tabel 3. Contoh Kode Program Repository

Kode Program

```
func (m mssqlTransactionRepo) PaymentBank(ctx context.Context, args domain.RequestBodyBank) (domain.DataStoreBank, error) {
    var transaction domain.DataStoreBank
    sqlExec := domain.ExecPaymentBank()
    rows, err := m.DB.QueryContext(ctx, sqlExec,
        args.TIMESTAMP, args.CHANNELID, args.REFFNUMBER, args.COMPANYCODE, args.CHANNELED, args.NIM,
        args.AMOUNT, args.TYPEINQ, args.PROCODE, args.TERMINALID)
    if err != nil {
        return transaction, err
    }

    defer func() {
        errRow := rows.Close()
        if errRow != nil {
            return
        }
    }()

    if rows.Next() {
        err := rows.Scan(&transaction.STATUS)
        if err != nil {
            return transaction, err
        }
    }

    return transaction, nil
}
```

3) Use Case Layer

Usecase berisi logika bisnis spesifik apa yang boleh dan tidak boleh dilakukan. Usecase akan memanggil fungsi repository untuk menyimpan/membaca data, dan melakukan proses seperti validasi atau transformasi data, contohnya seperti tabel 4.

Tabel 4. Contoh Kode Program Use Case

Kode Program

```
func (s transactionUseCase) PaymentBank(ctx context.Context, args domain.RequestBodyBank) (domain.DataStoreBank, error) {
    ctx, cancel := context.WithTimeout(ctx, s.contextTimeout)
    defer cancel()

    transaction, err := s.transactionRepo.PaymentBank(ctx, args)
    if err != nil {
        return domain.DataStoreBank{}, err
    }

    // Perform operation is a function that'll be executed after the transaction is fetched.
    if err := utils.PerformOperation(ctx); err != nil {
        return domain.DataStoreBank{}, err
    }

    return transaction, nil
}
```

p-ISSN: 2775-4227
e-ISSN: 2775-4219

4) Delivery Layer
Delivery merupakan lapisan paling luar yang menangani interaksi dengan client. Dalam proyek ini, komunikasi dilakukan melalui REST API menggunakan framework Echo. Delivery menerima permintaan HTTP, memvalidasi input, memanggil usecase yang relevan, lalu mengembalikan response ke client, contohnya seperti tabel 5.

Tabel 5. Contoh Kode Program Delivery

Kode Program

```
if inquiry.NIM == "" {
        utils.InsertLog(file, d.Log, "Bill ID not found. Refno: "+body.REFFNUMBER, "error", c)
        uhamka := domain.ResponsePaymentEducation{
            BILLDETAILS: domain.BillDetailsBank{
                BILLNAME:       "",
                BILLAMOUNT:     "",
                BILLAMOUNTPAY:  "",
                BILLAMOUNTBAYAR: "",
                BILLID:         "",
                TAHUNID:        "",
                BILLNAMEID:     "",
            },
            REFFNUMBER:       body.REFFNUMBER,
            TIMESTAMP:        time.Now().In(utils.CurrentTimeLoc()).Format("2006-01-02 15:04:05"),
            ERRORCODE:        bank.RC_BILL_ID_NOT_FOUND,
            RESPONSECODE:     bank.RC_BILL_ID_NOT_FOUND,
            RESPONSEDESC:     bank.DESC_BILL_ID_NOT_FOUND + " Bill ID not found",
            STATUSDESCRIPTION: "",
            NIM:              body.NIM,
            PAYMENTAMOUNT:    body.AMOUNT,
            USERLOGIN:        "",
            TRANSACTIONID:    "",
            PASSWORD:         body.PASSWORD,
            SEQNO:            "",
            CHANNELID:        body.CHANNELID,
        }
        return c.JSON(http.StatusOK, uhamka)
    }
```

c) Run Program

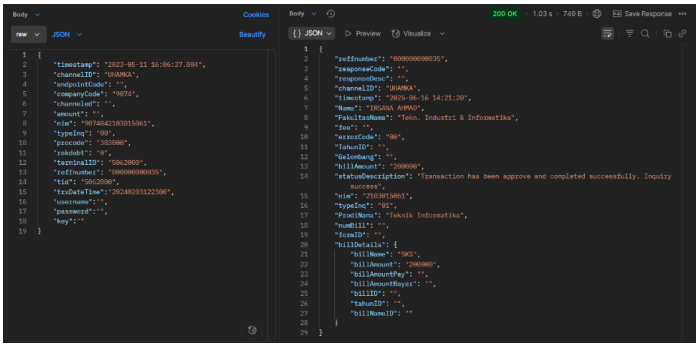
Gambar 8. menjalankan Program Golang dengan Framework Echo

Jika proses coding sudah selesai dan tidak ditemukan error maka untuk menjalankan program dengan menggunakan perintah *go run main.go,* jika program berjalan tanpa kendala makan akan tampil seperti Gambar 8.

3.1.4. Pengujian (Testing)

a) Pengujian Dengan Postman
Postman adalah alat untuk pengujian API yang memungkinkan pengembang untuk menguji endpoint, mengotentikasi permintaan, dan memverifikasi respons. Postman mendukung berbagai format data, seperti JSON dan XML, serta memfasilitasi pengujian API secara menyeluruh [3].

- Inquiry

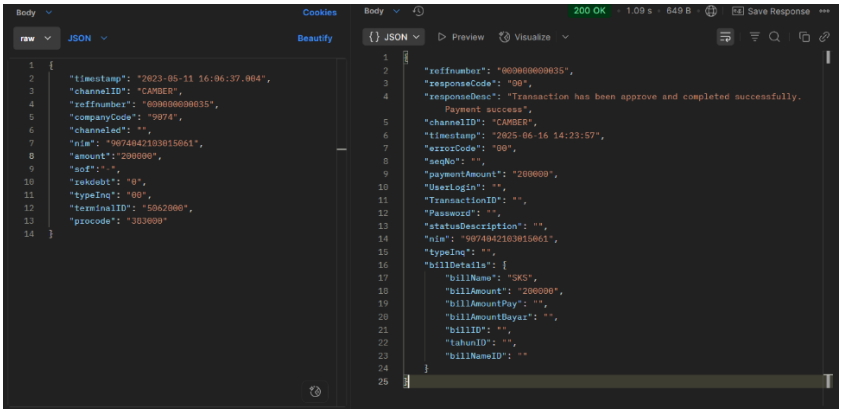


Gambar 9. Hasil hit endpoint inquiry pada postman

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Berdasarkan Gambar 9 hasil pengujian API bekerja sesuai harapan. Permintaan yang dikirim berhasil mengembalikan data mahasiswa sesuai dengan NIM yang diminta, lengkap dengan informasi fakultas, prodi, dan nominal tagihan. Status transaksi mengindikasikan bahwa inquiry berhasil dan sistem backend dapat membaca data mahasiswa dari database.

- Payment



Gambar 10. Hasil hit endpoint payment pada postman

Berdasarkan Gambar 10 transaksi payment berhasil dilakukan untuk mahasiswa dengan NIM: 9074042103015061. Nilai pembayaran 200000 diterima sistem dan diproses tanpa kesalahan (errorCode: "00"). Response menunjukkan bahwa API backend dapat menerima dan memproses pembayaran sesuai dengan standar virtual account.

b) Pengujian Dengan Black Box

Black box testing, yang juga dikenal sebagai Behavioral Testing, merupakan metode pengujian perangkat lunak yang berfokus pada pengamatan hasil dari input dan output tanpa memeriksa struktur internal kode program. Pengujian ini biasanya dilakukan pada tahap akhir pengembangan untuk memastikan bahwa perangkat lunak berfungsi sebagaimana mestinya [22]. Pengujian dilakukan terhadap dua endpoint utama dalam sistem backend Virtual Account UHAMKA, yaitu: inquiry dan payment.

Berikut merupakan hasil dari pengujian black box pada endpoint yang bisa dilihat pada tabel 6.

Tabel 6. Pengujian Endpoint Menggunakan Blackbox

No	Skenario	Input (Field Penting)	Output yang Diharapkan	Status
Inquiry endpoint: /transaction/nama-bank /dev/education/inquiry				
1	Inquiry dengan data lengkap dan NIM valid	nim: 9074042103015061	200, return detail tagihan VA	LULUS
2	Inquiry dengan NIM tidak terdaftar	nim: 9999999999999999	"statusDescription": "Bill ID not found.",	LULUS
3	Inquiry dengan NIM kosong	nim: ""	"statusDescription": "Format error. Payloads validation failed",	LULUS
4	Inquiry dengan channelID Kosong	"channelID": ""	"statusDescription": "Format error. Payloads validation failed",	LULUS
5	Inquiry dengan companyCode Kosong	"companyCode": ""	"statusDescription": "Format error. Payloads validation failed",	LULUS
6	Inquiry dengan typeInq Kosong	"typeInq": ""	"statusDescription": "Format error. Payloads validation failed",	LULUS
7	Inquiry dengan procode Kosong	"procode": ""	"statusDescription": "Format error. Payloads validation failed",	LULUS
8	Inquiry dengan terminalID Kosong	"terminalID": ""	"statusDescription": "Format error. Payloads validation failed",	LULUS

p-ISSN: 2775-4227
e-ISSN: 2775-4219

9	Inquiry dengan reffnumber Kosong	"reffnumber": ""	"statusDescription": "Format error. Payloads validation failed",	LULUS
10	Inquiry dengan trxDateTime Kosong	"trxDateTime":""	"statusDescription": "Format error. Payloads validation failed",	LULUS
	Payment endpoint: /transaction/mega-syariah/dev/education/payment			
11	Payment dengan data valid	nim: 9074042103015061, amount: 200000	200, Return response detail paymaent VA	LULUS
12	Payment dengan amount lebih dari billAmount	amount: "300000"	"responseDesc": "Invalid full amount. Payment not equal total nominal. Payment must be <= Tagihan",	LULUS
13	Payment dengan NIM salah	nim: 9999999999999999	"responseDesc": "Bill ID not found. Bill ID not found",	LULUS
14	Payment dengan nim Kosong	"nim": ""	"responseDesc": "Format error. Payloads validation failed",	LULUS
15	Payment dengan channelID Kosong	"channelID": ""	"responseDesc": "Format error. Payloads validation failed",	LULUS
16	Payment dengan reffnumber Kosong	"reffnumber": ""	"responseDesc": "Format error. Payloads validation failed",	LULUS
17	Payment dengan companyCode Kosong	"companyCode": ""	"responseDesc": "Format error. Payloads validation failed",	LULUS
18	Payment dengan typeInq Kosong	"typeInq": ""	"responseDesc": "Format error. Payloads validation failed",	LULUS
19	Payment dengan terminalID Kosong	"terminalID": ""	"responseDesc": "Format error. Payloads validation failed",	LULUS
20	Payment dengan procode Kosong	"procode": ""	"responseDesc": "Format error. Payloads validation failed",	LULUS

3.2. Pembahasan

Gambar 11. Cara Kerja API

Berdasarkan Gambar 11, API (*Application Programming Interface*) memfasilitasi aplikasi yang berbeda untuk saling terhubung dan bekerja sama secara serentak [12]. REST (*Representational State Transfer*) merupakan arsitektur standar dalam pengembangan API untuk layanan web, yang memungkinkan sistem melakukan permintaan guna mengakses dan mengelola sumber daya melalui protokol HTTP. Gaya arsitektur REST API menetapkan seperangkat prinsip dan aturan dalam pembuatan layanan web. Format data seperti JSON atau XML dalam REST API dapat dideskripsikan lebih lanjut menggunakan modeling language tertentu untuk merepresentasikan struktur dan kontennya [13]. Sebagai backend developer, peran utama adalah membangun dan mengelola API agar dapat melayani permintaan data maupun fungsi dari sisi client secara efisien, aman, dan terstruktur. Proses komunikasi API dimulai dari client, Saat pengguna melakukan suatu aksi (Hit API), maka client akan mengirimkan HTTP request ke endpoint API yang telah disediakan. API yang dibuat oleh backend developer kemudian menerima permintaan tersebut dan meneruskannya ke server. Di sinilah logika bisnis dijalankan. Misalnya, API dapat mengambil data mahasiswa dari basis data, memvalidasi input, atau memproses suatu transaksi. Setelah proses di server selesai, hasilnya dikemas dalam bentuk response (umumnya dalam format JSON) yang kemudian dikirimkan kembali ke API. Selanjutnya, API meneruskan response ini ke client agar dapat ditampilkan kepada pengguna dalam bentuk antarmuka yang mudah dipahami [4].

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan dan mudah dibaca oleh manusia serta mudah diproses oleh mesin. JSON sering digunakan dalam komunikasi antara client dan server pada REST API karena formatnya yang sederhana dan terstruktur [13].

Sistem backend API yang dibangun memiliki sejumlah kelebihan sebagai berikut:

1) Struktur Modular: Penerapan Clean Architecture memisahkan logika bisnis, pengelolaan data, dan antarmuka sehingga memudahkan perawatan dan pengembangan lanjutan.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

2) Performa Ringan: Penggunaan bahasa Go (Golang) memungkinkan penanganan concurrent request dengan efisiensi tinggi.
3) Integrasi Real-time: Sistem mampu merespons permintaan inquiry dan payment dari bank secara otomatis dan real-time.
4) Pengujian Fleksibel: Penggunaan metode black box testing melalui Postman memungkinkan pengujian cepat terhadap berbagai skenario permintaan tanpa bergantung pada struktur internal kode.

Dibandingkan penelitian sebelumnya, terdapat beberapa perbedaan dan keunggulan misalnya dalam studi oleh Widya & Aini (2021) yang merancang sistem pembayaran Virtual Account tingkat SMK menggunakan metode R&D dan polling bank, namun belum mengimplementasikan API backend secara modular dan real-time [11]. Sementara itu, dalam studi oleh Sasono et al. (2025) yang menerapkan XP untuk optimasi REST web service, dengan hasil peningkatan performa, tetapi penerapannya belum spesifik pada sistem pembayaran pendidikan [6]. Oleh karena itu, penelitian ini melengkapi kekosongan tersebut dengan mengimplementasikan backend API VA menggunakan Golang dan XP, lengkap dengan pengujian black-box, Clean Architecture, dan integrasi real-time untuk sektor pendidikan tinggi. Meskipun sistem berjalan dengan baik dalam pengujian, terdapat beberapa keterbatasan seperti Belum Diuji untuk Beban Tinggi (Stress/Load Test), Pengujian hanya dilakukan secara lokal dalam jumlah request terbatas dan Belum Tersedia Mekanisme Fallback Saat integrasi dengan bank gagal (misalnya, server down atau respons timeout), sistem saat ini belum memiliki mekanisme fallback seperti retry otomatis, log antrian, atau notifikasi ke admin.

4. KESIMPULAN

Penelitian ini berhasil mengembangkan backend layanan pembayaran Virtual Account (VA) di Universitas Muhammadiyah Prof. Dr. HAMKA (UHAMKA) menggunakan pendekatan REST API dan metode Extreme Programming. Penerapan prinsip Clean Architecture menghasilkan sistem yang modular, mudah dirawat, dan siap untuk dikembangkan lebih lanjut. Pengujian menunjukkan bahwa sistem berjalan sesuai fungsi yang diharapkan dan mendukung proses pembayaran secara akurat dan efisien. Untuk pengembangan ke depan, disarankan sistem perlu dikembangkan lebih lanjut dengan mekanisme autentikasi yang lebih aman dan menambahkan fitur dashboard monitoring secara real-time.

DAFTAR PUSTAKA

[1] P. Padeli, E. H. Purba, and B. Simanjuntak, "Analisa Pembayaran Perkuliahan dengan Virtual Account pada Universitas Raharja," *Cyberpreneurship Innovative and Creative Exact and Social Science*, vol. 6, no. 1, pp. 59–70, 2020, doi: 10.33050/cices.v6i1.878.

[2] Aditya P D and Rizal S N, "Kronologi PP Muhammadiyah Alihkan Dana dari BSI, Rencana sejak 2020," Kompas.com. Accessed: Oct. 17, 2024. [Online]. Available: https://www.kompas.com/tren/read/2024/06/06/180000165/kronologi-pp-muhammadiyah-alihkan-dana-dari-bsi-rencana-sejak-2020?page=all

[3] F. Febriansyah, R. M. Awangga, and R. Andarsyah, *MEMBANGUN RESTFUL API DENGAN GO*. Penerbit Buku Pedia, 2023. [Online]. Available: https://books.google.co.id/books?id=KYXOEAAAQBAJ

[4] I. R. D. Muhammad and I. V. Paputungan, "Pengembangan Backend Server Berbasis Arsitektur REST API pada Sistem Transfer Dompet Digital," *Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi*, vol. 3, no. 2, pp. 79–87, Jan. 2024, doi: 10.20885/snati.v3.i2.35.

[5] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions," *Applied Sciences (Switzerland)*, vol. 12, no. 9, p. 4369, May 2022, doi: 10.3390/app12094369.

[6] R. Adhi Sasono, S. Purnama Kristanto, L. Hakim, and D. Yusuf, "Optimasi Web Service REST Pada Backend Aplikasi Prospect Menggunakan Metode Extreme Programming," *Journal Zetroem*, vol. 7, no. 1, pp. 96–103, 2025, doi: 10.36526/ztr.v7i1.4202.

[7] F. Pamungkas and H. Setiaji, "IMPLEMENTASI CLEAN ARCHITECTURE PADA PEMBUATAN API MENGGUNAKAN GOLANG," *Jurnal INSTEK: Informatika Sains dan Teknologi*, vol. 9, no. 1, pp. 80–86, 2024, doi: 10.24252/instek.v9i1.46409.

[8] J. A. Alma and A. Prihanto, "Implementasi Backend System Untuk Integrasi Payment Gateway Pada Sistem Pembayaran Kost Menggunakan Express.js," *Journal of Informatics and Computer Science*, vol. 06, no. 01, pp. 167–178, 2024, doi: 10.26740/jinacs.v6n01.p167-178.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

[9] I. kumalasari, "IMPLEMENTASI HOST TO HOST BCA UNTUK TRANSAKSI VIRTUAL ACCOUNT BPR NIAGA MENGGUNAKAN RESTFUL API," *JORAPI : Journal of Research and Publication Innovation*, vol. 1, no. 3, pp. 866–870, 2023, [Online]. Available: https://jurnal.portalpublikasi.id/index.php/JORAPI/index

[10] D. K. Ladiba, W. A. Dewa, and S. Arifin, "Analisis dan Pengembangan API Siakad Menggunakan Arsitektur Restful Web Service pada Infrastruktur Microservice," *Prosiding Seminar SeNTIK*, vol. 5, no. 1, pp. 255–265, 2021.

[11] M. Anshori, A. Widya, N. Aini, and K. A. W. Hasbullah, "Design of a Student Payment System Based on Virtual Account (Case Study at SMK NU Al-Hidayah Ngimbang)," *NEWTON: Networking and Information Technology*, vol. 1, no. 1, pp. 35–40, 2021, doi: 10.32764/newton.v1i1.1835.

[12] E. Julio, M. A. I. Pakereng, and I. Artikel, "Implementasi API Payment Gateway Menggunakan Arsitektur Microservice," *JURNAL INFORMATIKA*, vol. 8, no. 2, pp. 123–130, 2021, doi: 10.31294/ji.v8i2.10590.

[13] N. Wulandari, A. Wibowo, and B. Susanto, "Penerapan RESTful API untuk Membangun Program Pembayaran Piutang Menggunakan Otentikasi OAuth 2.0," *Jurnal Terapan Teknologi Informasi*, vol. 5, no. 1, pp. 1–10, Apr. 2021, doi: 10.21460/jutei.2021.51.230.

[14] A. Pambudi and W. Apriandari, "An Extreme Programming Approach for Instructor Performance Evaluation System Development," *Journal of Informatics Information System Software Engineering and Applications (INISTA)*, vol. 5, no. 2, pp. 126–135, May 2023, doi: 10.20895/inista.v5i2.1050.

[15] S. Dhina Pohan and I. Firdaus, "IMPLEMENTATION OF EXTREME PROGRAMMING METHOD IN THE DEVELOPMENT OF PEKANBARU COMMUNITY TRAINING INFORMATION SYSTEM," *Jurnal Pendidikan Teknologi Informasi*, vol. 6, no. 1, pp. 20–33, 2022, doi: 10.22373/cj.v6i1.11851.

[16] A. F. Lestari, "Implementasi Extreme Programming Pada Perancangan Sistem Informasi Penjualan Buku Menggunakan Java," *Journal of Accounting Information System*, vol. 3, no. 1, pp. 6–12, 2023, doi: 10.31294/jais.v3i01.2010.

[17] B. Alturas, "Connection between UML use case diagrams and UML class diagrams: a matrix proposal," *International Journal of Computer Applications in Technology*, vol. 72, no. 3, pp. 161–168, 2023, doi: 10.1504/IJCAT.2023.133294.

[18] Y. Fatman, N. Khoirun Nafisah, and P. Bendoro Jembar Pambudi, "Implementasi Payment Gateway dengan Menggunakan Midtrans pada Website UMKM Geberco," *Jurnal KomtekInfo*, vol. 10, no. 2, pp. 64–72, Jun. 2023, doi: 10.35134/komtekinfo.v10i2.364.

[19] K. ' Afiifah, Z. Fira Azzahra, and A. D. Anggoro, "Analisis Teknik Entity-Relationship Diagram dalam Perancangan Database Sebuah Literature Review," *JURNAL INTECH*, vol. 3, no. 2, pp. 18–22, 2022, doi: 10.54895/intech.v3i2.1682.

[20] A. Putri Yulandi, "Analisis Performa Backend Framework: Studi Komparasi Framework Golang dan Node.js," *Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)*, vol. 8, no. 1, pp. 155–168, 2023, doi: 10.30645/jurasik.v8i1.551.

[21] Anggraeni D Mutia, Utomo S Fandy, and Marcos Hendra, "Integrasi Backend Golang-Echo pada Aplikasi Greenly sebagai Solusi Teknologi Pengelolaan Sampah Digital," *Jurnal Informatika: Jurnal pengembangan IT*, vol. 10, no. 2, pp. 527–536, 2025, doi: 10.30591/jpit.v10i2.8227.

[22] H. Nurfauziah and I. Jamaliyah, "PERBANDINGAN METODE TESTING ANTARA BLACKBOX DENGAN WHITEBOX PADA SEBUAH SISTEM INFORMASI," *Jurnal VISUALIKA*, vol. 8, no. 2, pp. 105–113, 2022.