

IMPLEMENTASI GOLANG CLEAN ARCHITECTURE PADA PERANCANGAN BACKEND POINT OF SALES WEBSITE

Resty Annisa^{1*}, Rachmat Agung Ananda², Wahyu Eko Sulistiono³

^{1,2,3}Teknik Informatika Universitas Lampung; Jl. Prof. Dr. Sumantri Brojonegoro No. 1 Bandar Lampung, 35145, Indon esia

Received: 10 Maret 2024
Accepted: 25 Maret 2024
Published: 2 April 2024

Keywords:

Clean Architecture;
Golang; Point of Sales
Website.

Correspondent Email:

resty.annisa@eng.unila.ac.id

Abstrak. Penelitian ini bertujuan untuk mengimplementasikan konsep Clean Architecture menggunakan Bahasa Pemrograman Golang dalam perancangan backend untuk situs web Point of Sales (POS). Implementasi ini berhasil dilakukan dengan membagi beberapa layer utama yaitu repository, usecase, dan handler fitur. Penelitian ini dilakukan menggunakan metode Agile Scrum, di mana konsep Clean Architecture diterapkan dalam tahapan sprint backlog untuk mengkodekan tugas-tugas yang harus diselesaikan. Pada implementasi Repository List Transaction, digunakan fungsi ListTransaction yang berperan dalam mengambil daftar transaksi berdasarkan kriteria. Pada implementasi Usecase, digunakan fungsi ListPaymentMethod untuk mengambil daftar metode pembayaran yang tersedia dalam konteks manajemen pesanan. Pada implementasi Handler Create Order, digunakan fungsi CreateOrder untuk mengelola proses pembuatan pesanan. Hasil penelitian menunjukkan bahwa penggunaan Clean Architecture dalam pengembangan backend POS memberikan struktur yang jelas dan memudahkan dalam pengelolaan kode serta pengembangan fitur lebih lanjut.

Abstract. This research aims to implement the concept of Clean Architecture using the Golang programming language in designing the backend for a Point of Sales (POS) website. The implementation was successfully carried out by dividing the main layers into repository, use case, and feature handler. The study was conducted using the Agile Scrum method, where the concept of Clean Architecture was applied during the sprint backlog stage to code the tasks that needed to be completed. In the implementation of the Repository List Transaction, the ListTransaction function was used to retrieve a list of transactions based on specific criteria. In the implementation of the Use Case, the ListPaymentMethod function was employed to fetch the list of available payment methods in the context of order management. In the implementation of the Handler Create Order, the CreateOrder function was utilized to manage the order creation process. The results of the study indicate that the use of Clean Architecture in POS backend development provides a clear structure and facilitates code management and further feature development.

1. PENDAHULUAN

Tantangan yang dihadapi perusahaan penjualan saat ini dalam bentuk kebutuhan akan pengelolaan penjualan yang lebih efisien, pemantauan inventaris yang akurat, dan analisis

data yang lebih mendalam untuk mendukung pengambilan keputusan bisnis yang lebih baik. Website dengan sistem Point of Sales (POS) dapat mengumpulkan data penjualan untuk analisis bisnis yang lebih mendalam. Ini akan

membantu dalam pengambilan keputusan yang lebih baik untuk mengoptimalkan operasional perusahaan. Salah satu pendekatan yang semakin populer dalam membangun sistem yang kokoh adalah penggunaan Clean Architecture.

Clean Architecture menekankan pemisahan yang jelas antara berbagai lapisan aplikasi, memungkinkan kode yang lebih mudah dipelihara, diuji, dan dikembangkan ulang. Dalam konteks aplikasi web, implementasi Clean Architecture pada backend menjadi krusial untuk memastikan ketahanan sistem, kemudahan pengembangan fitur baru, dan skalabilitas yang optimal.

Dalam konteks pengembangan sistem POS, Menurut Wiji Sulistiani dan Wiwin Sulisty, penggunaan bahasa pemrograman Golang memiliki beberapa kelebihan yang signifikan diantaranya, berkinerja tinggi, efisiensi penggunaan memori, memiliki pustaka yang kuat yang digunakan dalam berbagai fungsi balikan data serta dapat mudah untuk dimodifikasi dalam penggunaan struktur folder nya dengan penggunaan package sebagai kuncinya. [13]

Penggunaan konsep clean architecture pada bahasa Golang diharapkan dapat menghasilkan sistem backend yang tangguh, mudah dipelihara, dan mampu memenuhi tuntutan bisnis yang semakin berkembang.

2. TINJAUAN PUSTAKA

2.1 Point of Sales (POS)

Sistem POS (point of sale) adalah sebuah sistem aplikasi yang diterapkan pada bisnis minimarket ataupun pertokoan untuk menangani pengolahan data transaksi pembelian (purchases), transaksi penjualan eceran (retails), transaksi retur pembelian (purchase returns), dan pelaporan transaksi (reporting) yang secara umum penting dibutuhkan dalam pengambilan keputusan strategis oleh para pebisnis swalayan, organisasi, atau perusahaan yang berskala kecil dan menengah. [11]

Setiap POS terdiri dari hardware berupa (Terminal/PC, Receipt Printer, Cash Drawer, Terminal pembayaran, Barcode Scanner) dan software berupa (Inventory Management, Pelaporan, Purchasing, Customer Management, Standar Keamanan Transaksi, Return Processing) dimana kedua komponen tersebut

digunakan untuk setiap proses transaksi. Point of sales dapat berupa sebuah checkout counters dalam sebuah toko atau tempat usaha dimana transaksi penjualan terjadi. [3]

2.2 Backend

Backend adalah tempat di mana proses pada suatu sistem informasi atau aplikasi berjalan, data dapat ditambahkan, diubah maupun dihapus. Backend biasanya mengurus segala jenis proses yang tidak berhubungan langsung dengan pengguna, seperti server dan basis data. Backend dibutuhkan dalam pengembangan sistem dan manajemen data pada sistem. [9]

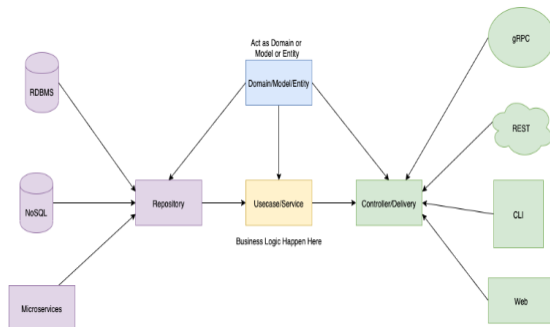
2.3 Golang

Bahasa pemrograman Go, juga dikenal sebagai Golang, adalah bahasa pemrograman sumber terbuka dengan gaya sintaksis yang mirip dengan bahasa C dan C++ yang diterbitkan oleh Google. Didesain oleh tiga ilmuwan komputer Google, Robert Griesemer, Ken Thompson, dan Rob Pike, bahasa ini awalnya dimaksudkan untuk memperbaiki kebocoran memori yang telah lama membatasi C++. Resmi dirilis ke publik pada tahun 2009, Go menawarkan beberapa fitur hebat: sederhana, sehingga mudah dipelajari; secara sintaksis mudah dan manajemen memori yang baik; perakitan cepat; kebutuhan (struktural dan prosedural) yang mendukung konkurensi; itu menampilkan pengetikan statis, yang menjadikan Go bahasa pemrograman yang aman; gratis dan open source dengan dokumentasi lengkap; dan mendukung kompilasi silang. [6]

2.4 Clean Architecture

Clean Architecture adalah pendekatan dalam rekayasa perangkat lunak yang diusulkan oleh Robert C. Martin. Ini adalah pendekatan berbasis komponen dalam Rekayasa Perangkat Lunak (CBSE) yang membantu memisahkan aspek-aspek yang terkait dengan fungsionalitas yang bergantung pada platform dan yang tidak bergantung pada platform. Robert C. Martin mengatakan bahwa menggunakan arsitektur berlapis adalah titik awal yang baik, tetapi ketika perangkat lunak semakin besar dan kompleks, masalahnya adalah adanya tiga kontainer besar kode yang tidak memisahkan kode dengan benar. Selain itu, Clean Architecture adalah pendekatan yang berpusat pada domain, yang memungkinkan untuk menjelaskan semua elemen dari domain tersebut. Salah satu fitur penting dari Clean

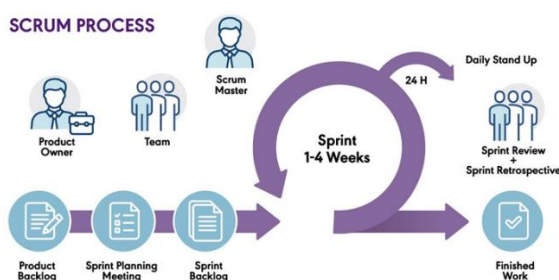
Architecture adalah kemampuannya untuk mengganti antarmuka pengguna (UI) dan sumber data tanpa masalah. Ini memungkinkan pengujian aturan bisnis tanpa bergantung pada UI, database, layanan, atau dependensi eksternal lainnya. Sebagai hasilnya, ini menghasilkan kode sumber dengan logika domain yang dikelilingi oleh komponen infrastruktur. [4]



Gambar 1. Diagram Struktur Clean Architecture
(Tumorang, 2017)[12]

3. METODE PENELITIAN

Penelitian ini dilakukan menggunakan metode Agile Scrum. Agile methodology adalah pendekatan pengembangan perangkat lunak yang fokus pada kolaborasi tim, adaptasi terhadap perubahan, dan pengiriman produk bernilai bagi pelanggan. Scrum, sebagai kerangka kerja dalam metodologi agile, memungkinkan adaptasi terhadap perubahan melalui Sprint Planning dan Sprint Review. Scrum mendorong kolaborasi tim melalui pertemuan harian (Daily Scrum) dan menerapkan prinsip transparansi dan inspeksi melalui Sprint Review dan Sprint Retrospective (Resmi Rachmawati et al., 2023). Adapun tahapan yang dilakukan dalam penelitian dapat dilihat pada gambar berikut. [10]



Gambar 2. Tahapan Penelitian
(Sumber: PM-Partner Group, 2021) [5]

3.1 Product Backlog

Dalam metodologi Agile Scrum, tahap pertama ialah Product Backlog. Di sini, kebutuhan sistem muncul dari hasil identifikasi user story. Ini adalah daftar yang berisi semua pekerjaan yang dibutuhkan untuk mengembangkan produk, seperti fitur baru, perbaikan, dan tugas lainnya. Semua entri diurutkan berdasarkan nilai bisnis dan prioritasnya. Menurut (Adi, 2015), menyatakan bahwa Product Backlog berperan sebagai sumber utama kebutuhan dan tujuan bagi tim pengembangan.[1]

3.2 Sprint Planning Meeting

Menurut (Pratama et al., 2022) menyatakan bahwa, Setelah Product Backlog terdefinisi, tim memasuki tahap Sprint Planning Meeting yang bertujuan menyusun rencana dan mengatur pekerjaan untuk periode Sprint tersebut. Item dari Product Backlog yang terpilih dipecah menjadi tugas-tugas terperinci, memberikan pemahaman mendalam kepada tim tentang pekerjaan yang harus dilakukan. Tim juga menggunakan diagram use case dan activity diagram serta entity relationship diagram untuk representasi visual yang jelas tentang pelaksanaan pekerjaan. Kombinasi perencanaan Sprint dan visualisasi diagram membantu tim dalam menjalankan pekerjaan dengan lebih terstruktur dan efisien. [8]

3.3 Sprint Backlog

Menurut (Agarina et al., 2022) sprint backlog merupakan salah satu elemen penting dalam kerang kerja pengembangan perangkat lunak scrum yang berisikan daftar tugas atau pekerjaan yang harus diselesaikan oleh tim pengembangan selama satu sprint tertentu.[2]

Sprint backlog dibuat oleh Product Owner dan tim pengembangan bekerja sama dalam mengidentifikasi dan memprioritaskan item-item pekerjaan yang akan dimasukkan ke dalam Sprint backlog. Sprint backlog membantu tim pengembangan fokus pada tujuan Sprint dan menghindari gangguan selama periode tersebut. Ini juga memungkinkan tim untuk membuat rencana yang lebih rinci tentang bagaimana mereka akan mencapai tujuan mereka dalam Sprint tersebut.

3.4 Daily Standup

Menurut (Noviandi & Rumana, 2022) daily Scrum adalah pertemuan harian yang singkat, biasanya berlangsung 15 menit atau kurang. Ini diadakan setiap hari kerja selama Sprint dengan tujuan meningkatkan

komunikasi, koordinasi, dan kolaborasi dalam tim pengembangan serta membantu tim untuk tetap terfokus pada pencapaian tujuan Sprint dan mengidentifikasi hambatan apa pun yang mungkin menghambat kemajuan pengembangan daily standup dihadiri oleh Semua anggota tim pengembangan dan scrum master yang terlibat dalam Sprint. Berikut merupakan kegiatan daily standup yang dihadiri setiap harinya dapat dilihat dibawah ini. Pada dasarnya, Daily Stand-Up adalah pertemuan harian singkat yang merupakan komponen utama dalam siklus Sprint di dalam kerangka kerja Scrum. Namun, ada dua elemen penting yang menjadi bagian dari siklus Sprint yang lebih besar, yaitu Sprint Review dan Sprint Retrospective. [7]

4. HASIL DAN PEMBAHASAN

Coding dilakukan menggunakan bahasa pemrograman golang dengan framework fiber. Source code ditampilkan pada beberapa bagian penting menggunakan konsep clean architecture dengan pembagian beberapa layer yaitu repository, usecase, dan handler fitur yang diimplementasikan adalah sebagai berikut.

4.1 Implementasi Repository

Transaction

```
func ListTransaction(transactions "[Jentiy transaction, page int, perPage int, searchFilter string, statusFilter string, outletId int, dateFilterFormatted string] (error, []response.TransactionList, int) {
    db := database.DB
    var responseofTransactions []response.TransactionList
    var total int = 0

    statusId, _ := strconv.Atoi(statusFilter)

    db.Raw(qGetListTransaction, page, perPage, outletId, searchFilter, statusId, dateFilterFormatted).
        Scan(&responseofTransactions)

    log.Println("transaction : ", responseofTransactions)

    var responseAllTransactions []response.TransactionList
    for _, respon := range responseofTransactions {
        if respon.TransactionId != 0 {
            db.Raw(qGetTransactionDetail, respon.TransactionId).Scan(&respon.ProductOrder)

            for i, resDetail := range respon.ProductOrder {
                db.Raw(qGetTransactionDetailExp, resDetail.ProductSKU, resDetail.TransactionDetailId).
                    Scan(&respon.ProductOrder[i].TransactionDetailExp)
            }
        }

        db.Raw("SELECT contact_code, member_no, member_name, phone_no, membership_source FROM transaction_member_details
            WHERE transaction_refer = ?", respon.TransactionId).Scan(&respon.MemberDetail)

        if respon.PaymentMethodRefer == 1 {
            respon.Pengiriman = "Antar ke lokasi customer"
        } else {
            respon.Pengiriman = "Ambil di outlet"
        }

        responseAllTransactions = append(responseAllTransactions, respon)
    }

    //sum total for pagination
    total = responseofTransactions[0].Total

    return nil, responseAllTransactions, total
}
```

Gambar 4.1 Implementasi *Repository List Transaction*

Fungsi ListTransaction berperan dalam mengambil daftar transaksi berdasarkan sejumlah kriteria tertentu dan menghasilkan respons yang sesuai dalam konteks aplikasi. Fungsi ini menerima beberapa parameter,

termasuk transactions yang merupakan slice dari transaksi, page dan perPage untuk mengatur halaman dan jumlah data per halaman, searchFilter untuk pencarian, statusFilter untuk filter status transaksi, outletId untuk mengidentifikasi outlet terkait, dan dateFilterFormatted untuk filter tanggal yang telah diformat. Pertama-tama, fungsi ini mempersiapkan variabel-variabel yang akan digunakan nanti, seperti responseofTransactions untuk menyimpan hasil query database dan total untuk menghitung jumlah total data yang sesuai dengan kriteria tertentu. Selanjutnya, fungsi ini menggunakan database query yang telah diformat (qGetListTransaction) untuk mengambil data transaksi dari basis data. Query ini mengambil data transaksi berdasarkan berbagai parameter seperti halaman, jumlah data per halaman, outlet, kata kunci pencarian, status, dan tanggal. Hasil dari query tersebut disimpan dalam responseofTransactions. Selanjutnya, fungsi ini melakukan iterasi melalui data transaksi yang ditemukan dan menambahkan informasi tambahan ke setiap transaksi. Dalam setiap iterasi, fungsi ini melakukan beberapa tugas seperti mengambil detail produk dari transaksi, termasuk informasi kedaluwarsa produk, jika ada. Selain itu, fungsi ini juga mengambil detail anggota terkait transaksi. Selanjutnya, terdapat logika untuk menentukan metode pengiriman yang akan ditampilkan dalam respons, berdasarkan metode pembayaran yang digunakan dalam transaksi. Akhirnya, semua data transaksi yang telah diolah dengan informasi tambahan disimpan dalam responseAllTransactions, dan total jumlah data yang sesuai dengan kriteria disimpan dalam variabel total. Sebagai hasilnya, fungsi ini mengembalikan nil untuk error (jika tidak ada error), daftar transaksi yang telah diolah dengan informasi tambahan, dan total jumlah data yang sesuai. Data ini kemudian akan digunakan dalam pembuatan respons HTTP untuk pengguna yang meminta daftar transaksi sesuai dengan kriteria tertentu.

4.2 Implementasi Usecase

```
func ListPaymentMethod() (int, string, string, []response.PaymentMethod) {
    var paymentMethods []entity.PaymentMethod
    var responseofPaymentMethods []response.PaymentMethod

    err, responseofPaymentMethods := repository.ListPaymentMethod(&paymentMethods)
    if err != nil {
        return 404, err.Error(), "", responseofPaymentMethods
    }

    return 0, "Payment Method Ditemukan", "", responseofPaymentMethods
}
```

Gambar 4.33 Implementasi Usecase List Payment Method

Fungsi ListPaymentMethod dalam kode diatas memiliki tujuan untuk mengambil daftar metode pembayaran yang tersedia dalam konteks manajemen pesanan. Fungsi ini bertugas memberikan informasi tentang metode pembayaran yang dapat digunakan dalam proses pemesanan kepada aplikasi atau pengguna. Fungsi ini tidak memerlukan parameter eksternal dan bekerja dengan menginisialisasi dua variabel, yaitu paymentMethods yang akan menampung daftar metode pembayaran dari database, dan responseofPaymentMethods yang akan menampung hasil akhir yang akan dikembalikan. Selanjutnya, fungsi memanggil repository dengan memanggil repository.ListPaymentMethod(&paymentMethods). Hasil dari panggilan ini dimasukkan ke dalam variabel responseofPaymentMethods. Proses ini bertujuan untuk mengambil daftar metode pembayaran yang tersedia dari database. Setelah itu, fungsi melakukan pemeriksaan apakah terdapat kesalahan dalam pengambilan data. Jika ada kesalahan, fungsi akan mengembalikan kode status HTTP yang sesuai (misalnya, 404) bersama dengan pesan kesalahan yang relevan. Namun, jika metode pembayaran ditemukan dalam hasil query dan tidak ada kesalahan yang terjadi, maka fungsi akan mengembalikan pesan yang menyatakan bahwa metode pembayaran ditemukan, bersama dengan daftar metode pembayaran yang tersedia.

4.3 Implementasi Handler Create Order

```
func CreateOrder(c *fiber.Ctx) error {
    name := fmt.Sprintf("%v", c.Locals("name"))
    outletId := int(c.Locals("outlet_id").(uint))
    order := new(request.Order)

    err := c.BodyParser(order)
    if err != nil {
        logger.CreateLogData(c, fiber.StatusBadRequest, "Periksa kembali input anda", order, err.Error())
        return helper.ResponseError(400, "Periksa kembali input anda!", "")(c)
    }

    errorCode, errorMessage, errorData, responseOfOrder := usecases.CreateOrder(order, outletId, name)
    if errorCode != 0 {
        logger.CreateLogData(c, errorCode, errorMessage, order, errorData)
        return helper.ResponseError(errorCode, errorMessage, errorData)(c)
    }

    logger.CreateLogData(c, fiber.StatusOK, "Order sukses dibuat", order, responseOfOrder)
    return helper.ResponseSuccess(200, "Order sukses dibuat", responseOfOrder)(c)
}
```

Gambar 4.39 Implementasi Handler Create Order

Fungsi CreateOrder dalam kode diatas bertujuan untuk membuat pesanan baru (order). Fungsi ini digunakan dalam framework Fiber untuk mengelola proses pembuatan pesanan. Pertama-tama, fungsi ini mengambil informasi penting dari konteks, seperti nama pengguna (name) dan ID outlet (outletId). Kemudian, fungsi ini membaca data pesanan (order) yang dikirim dalam bentuk permintaan (request) oleh pengguna atau aplikasi. Selanjutnya, fungsi melakukan validasi data pesanan dan memprosesnya dengan memanggil use case usecases.CreateOrder. Use case ini bertanggung jawab untuk membuat pesanan baru dengan menggunakan data yang diberikan dan melakukan berbagai validasi tambahan. Setelah pemrosesan selesai, fungsi CreateOrder memeriksa kode status yang dikembalikan oleh use case. Jika kode statusnya tidak nol, ini menunjukkan bahwa ada kesalahan dalam proses pembuatan pesanan, dan fungsi akan menciptakan log yang mencatat kesalahan tersebut. Kemudian, fungsi mengembalikan respons error dengan kode status HTTP yang sesuai, serta pesan kesalahan dan data kesalahan yang relevan kepada pengguna atau aplikasi. Namun, jika tidak ada kesalahan yang terdeteksi, fungsi akan menciptakan log yang mencatat keberhasilan pembuatan pesanan dan mengembalikan respons sukses dengan kode status HTTP 200. Respons ini berisi pesan yang menyatakan keberhasilan serta data pesanan yang telah dibuat kepada pengguna atau aplikasi.

5 KESIMPULAN

Berdasarkan hasil dari penelitian yang telah disampaikan, didapatkan kesimpulan sebagai berikut:

- a. Implementasi konsep clean architecture menggunakan Bahasa Golang untuk Perancangan Backend Point of Sales Website berhasil dilakukan dengan pembagian beberapa layer yaitu repository, usecase, dan handler fitur.
- b. Penelitian dilakukan menggunakan metode Agile Scrum. Pada tahapan sprint backlog diterapkan konsep clean architecture dalam pengkodean tugas-tugas yang harus diselesaikan.
- c. Pada implementasi Repository List Transaction digunakan fungsi ListTransaction. Pada implementasi Usecase digunakan fungsi ListPaymentMethod. Pada implementasi Handler Create Order digunakan fungsi CreateOrder.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada pihak-pihak terkait yang telah memberi dukungan terhadap penelitian ini.

DAFTAR PUSTAKA

- [1] Adi, P. (2015). Scrum Method Implementation in a Software Development Project Management. *International Journal of Advanced Computer Science and Applications*, 6(9), 198–204. <https://doi.org/10.14569/ijacsa.2015.060927>
- [2] Agarina, M., Suryadi Karim, A., & Fauzi, Mr. (2022). Media Pembelajaran Interaktif Kelas Praktikum Berbasis Digital di IIB Darmajaya.
- [3] B. Loardy, B. Bunawan, and P. Hartono, "Aplikasi Point Of Sales Yang Terhubung Dengan Electronic Data Capture," no. Lc, no. Lc, pp. 3–6, 2010. *Ijccs*, 16, No.01(x), 1–5.
- [4] Daniel Sanchez, Alix E. Rojas, and Hector Florez, "Towards a Clean Architecture for Android Apps using Model Transformations," *IAENG Int. J. Comput. Sci.*, vol. 49, no. 1, 2022.
- [5] Group, P.-P. (2021). The Agile Journey: A Scrum overview. *AGILE AND SCALED AGILE*. <https://www.pm-partners.com.au/the-agile-journey-a-scrum-overview/>
- [6] Ni Kadek Dwi Sabrina, Dian Pramana, and Tubagus Mahendra Kusuma, "Implementation of Golang and ReactJS in the COVID-19 Vaccination Reservation System," *ADI J. Recent Innov.*, vol. 5, no. 1, pp. 1–12, 2023. <https://doi.org/10.34306/ajri.v5i1.877>
- [7] Noviandi, N., & Rumana, N. A. (2022). Implementasi Agile Method untuk Pengembangan Sistem Pembatasan Pengunjung Wisata Berbasis Mobile. *Journal of Information System Research (JOSH)*, 4(1), 65–72. <https://doi.org/10.47065/josh.v4i1.2077>
- [8] Pratama, S., Ibrahim, S., & Reybaharsyah, M. A. (2022). Jurnal Penggunaan Metode Scrum Dalam Membentuk Sistem Informasi Penyimpanan Gudang Berbasis Web. *Intech*, 3(1), 27–35. <https://doi.org/10.54895/intech.v3i1.1192>
- [9] R. Pangestika and R. T. Dirgahayu, "Pengembangan Back-end Sistem Informasi Pendataan Sekolah Desa Komunitas Pendar Foundation," *Automata*, vol. 1, no. 2, pp. 184–189, 2020.
- [10] Resmi Rachmawati, O. C., Deyana Kusuma Wardani, Wifda Muna Fatihia, Arna Fariza, & Hestiasari Rante. (2023). Implementing Agile Scrum Methodology in The Development of SICITRA Mobile Application. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 7(1), 41–50. <https://doi.org/10.29207/resti.v7i1.4688>
- [11] Sugihartono, K. I. Satoto, and E. D. Widiyanto, "Pembuatan Aplikasi Point of Sale Toko Cabang Perusahaan Torani Menggunakan Framework CodeIgniter," *J. Teknol. dan Sist. Komput.*, vol. 3, no. 4, p. 445, 2015, doi: 10.14710/jtsiskom.3.4.2015.445-455.
- [12] Tumorang, I. (2017). Trying Clean Architecture on Golang Independent, Testable, and Clean. *Easyread*. <https://medium.easyread.co/golang-clean-architecture-efd6d7c43047>
- [13] Wiji Sulistiani, & Wiwin Sulistyo. (2020). Implementasi Web Service dengan Metode REST Berbasis Golang pada Layanan Google Cloud Platform. 672015229.