# PERANCANGAN APLIKASI MENGGUNAKAN METODE REPOSITORY DESAIN PATTERN STUDI KASUS PIUTANG DAGANG

Mahesa Adi Kusuma[1], Yulhendri[2*], Budi Tjahjono[3], Nizirwan Anwar[4]
[1,2,3,4]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Esa Unggul, Indonesia
*e-mail : [1]mahesaadikuzuma@gmail.com, [2*]yulhendri@esaunggul.ac.id,*
*[3]budi.tjahjono@esaunggul.ac.id, [4]nizirwan.anwar@esaunggul.ac.id*

***Sistem pengelolaan piutang dagang menjadi penting dalam menjaga arus kas keuangan perusahaan terutama dalam transaksi penjualan secara kredit. untuk merancang dan membangun sistem pengelolaan piutang berbasis website yang mendukung pencatatan, monitoring, dan pembayaran piutang. Metode Pengembangan sistem menggunakan salah satu pendekatan agile scrum yang memungkinkan berlangsung dalam proses pengembangan sistem lebih interaktif dan adaptif terhadap perubahan-perubahan kebutuhan pengguna. Sistem ini dibangun dengan pola Repository pattern dalam pengembangan menggunakan laravel dengan diintegrasikan dengan Midtrans sebagai payment gateway dalam proses pembayaran piutang, serta lebih tepat notifikasi otomatis jika piutang mendekati jatuh tempo masing-masing pengguna, pengelolaan laporan dalam piutang dan transaksi piutang. Pengujian performa menggunakan Apache JMeter membandingkan sistem dengan dan tanpa Repository Pattern. Hasil pengujian menunjukkan bahwa penerapan Repository Pattern menghasilkan rata-rata waktu respons 1696,333 ms, lebih cepat dibandingkan pendekatan tanpa Repository Pattern yang memiliki rata-rata 2081,667 ms.***

***Kata Kunci*— Piutang Dagang. Repository Pattern, Apache JMeter, Laravel, Agile Scrum**

## I. PENDAHULUAN

Dalam ranah bisnis, Sistem informasi diharapkan mampu memberikan dukungan aktif dalam hal pengelolaan informasi dan fungsi pengawasan, sebagaimana yang diterapkan dalam akuntansi atau pelaporan data akuntansi. Kondisi ini menuntut perusahaan untuk memiliki sistem informasi akuntansi yang sesuai agar proses pelaporan informasi akuntansi dapat berjalan lebih efisien. Salah satunya bentuk laporan akuntansi tersebut adalah laporan piutang usaha atau laporan penjualan secara kredit. Piutang usaha sendiri merupakan aktivitas yang muncul ketika perusahaan menjual barang atau jasa kepada pelanggan dan pelanggan tidak langsung melakukan pembayaran, Bisa dikatakan piutang merupakan salah satu aktivitas yang sangat krusial dalam suatu Perusahaan [1].

Dalam piutang, terdapat dua jenis yaitu piutang dagang dan piutang non-dagang ketika suatu barang atau jasa dijual berdasarkan kesepakatan bahwa pembayaran akan dilakukan di kemudian hari, maka pembayaran yang akan diterima dari pelanggan tersebut dicatat dalam jurnal sebagai piutang [2].

Piutang merupakan bentuk tagihan yang muncul dari transaksi penjualan kredit antara perusahaan dan pihak lain. Umumnya, piutang dijadwalkan untuk dilunasi oleh pelanggan dalam jangka waktu 30 hingga 60 hari. Jika dalam periode tersebut pelanggan belum melunasi kewajibannya, maka tagihan tersebut tetap tercatat sebagai piutang dan belum dapat dikonversi menjadi kas. Kondisi ini dapat menghambat kelancaran arus kas perusahaan dan berdampak pada penurunan efektivitas operasional Salah satu konsekuensi dari piutang adalah munculnya piutang tak tertagih, yaitu tagihan yang tidak dapat ditagih kembali akibat ketidakmampuan atau kelalaian pelanggan dalam memenuhi kewajibannya. Piutang tak tertagih merupakan risiko signifikan bagi perusahaan yang memberikan penjualan secara kredit, karena dapat mengancam stabilitas dan keberlangsungan usaha, meskipun sebelumnya telah dilakukan analisis kelayakan pelanggan secara menyeluruh. Untuk meminimalkan dampak tersebut, perusahaan biasanya menyusun daftar umur piutang sebagai alat bantu dalam memantau dan mengklasifikasikan piutang yang masih beredar berdasarkan jangka waktunya. Selain itu, perusahaan juga melakukan [3].

Oleh karena itu, dalam transaksi piutang dagang yang berpotensi menimbulkan piutang tak tertagih atau keterlambatan pembayaran akibat kelalaian pengguna dalam memperhatikan jatuh tempo, diperlukan sistem notifikasi otomatis. Dengan sistem tersebut, pengguna akan secara langsung mendapatkan pengingat berdasarkan tenggat waktu piutang yang telah ditentukan, sehingga mereka dapat segera melakukan pelunasan sebelum jatuh tempo, Selain permasalahan dalam pengingat piutang, Pada sistem sebelumnya pembayaran pelanggan atau customer masih dilakukan manual dengan menginputkan pembayaran yang sesuai dengan customer ingin bayarkan.

Pada sistem lama, bukti pembayaran harus dikirim manual oleh customer. Penelitian ini menawarkan opsi pembayaran digital melalui Midtrans agar lebih praktis.

Design Pattern adalah solusi umum yang dapat digunakan kembali untuk masalah yang sering terjadi dalam suatu konteks tertentu di software desain. Pattern ini mewakili best practice yang digunakan oleh software developer untuk memecahkan suatu masalah tertentu, seperti meningkatkan keterbacaan kode, fleksibilitas, skalabilitas dan kemudahan dalam maintenance. Sehingga developer dapat menerapkan strategi dan struktur yang terbukti untuk memecahkan permasalahan dengan lebih efisien [4].

Sebanyak tujuh dari sepuluh developer menghadapi kendala saat mengembangkan atau membangun aplikasi. Masalah tersebut dapat diatasi melalui penerapan pola perancangan (Design Pattern). Pemilihan design pattern bergantung pada jenis masalah yang ingin diselesaikan atau dicegah. Salah satu pola yang sering digunakan adalah MVC (Model-View-Controller), yang memiliki beberapa keunggulan, antara lain (1) logika bisnis terpisah dari model, (2) perubahan pada satu bagian tidak berdampak pada bagian lain yang tidak mengalami perubahan, dan (3) proses pengembangan menjadi lebih efisien, Namun kelemahan dari pola ini adalah beban proses yang terlalu besar pada controller, sehingga menyulitkan pengelolaan dan meningkatkan kompleksitas aplikasi seiring waktu [5].

Berdasarkan hal tersebut, dalam penelitian ini arsitektur MVC tetap digunakan sebagai dasar pengembangan aplikasi, namun ditingkatkan dengan menerapkan Repository Pattern untuk mengatasi permasalahan utama dari controller yang terlalu kompleks. Repository Pattern bertindak sebagai lapisan tambahan antara controller dan model yang bertujuan untuk memisahkan logika bisnis dan logika akses data, sehingga setiap komponen memiliki tanggung jawab yang lebih spesifik.

Duplikasi kode akses data, tanpa repository pattern kode akses data sering kali terduplikasi di beberapa tempat di aplikasi, Hal ini menyebabkan penanganan data yang tidak konsisten yang menyebabkan bug dan kesulitan dalam debugging, jika ada perlu mengubah kueri data atau handle data validasi jadi harus mengubahnya di beberapa tempat, yang meningkatkan kemungkinan kesalahan dan bug, dan seiring berkembangnya aplikasi pengelolaan dan perubahan kode akses data di beberapa tempat menjadi lebih sulit [4].

Repository Pattern sering digunakan bersama dengan pattern lain dan arsitektur layer lainnya, terutama Domain-Driven Design (DDD) dan arsitektur Layered. Dalam konteks ini pola ini berfungsi sebagai perantara domain model dan akses data layer [4]. Dalam Penerapan repository pattern adalah sebuah desain arsitektur dalam pengembangan software [6]. Merupakan salah satu pendekatan dalam pengembangan perangkat lunak yang dapat membantu mengelola data dengan lebih struktur, fleksibel, dan reusable dalam pengelolaan code. Dengan memisahkan logika bisnis, sehingga dalam metode ini memungkinkan aplikasi lebih mudah di kembangkan dan di pelihara

Penelitian ini bertujuan untuk merancang sistem piutang dagang berbasis website dengan menggunakan Arsitektur MVC dan Repository Design Pattern. dan akan melakukan perbandingan dari sisi performance dan struktur kode ketika menggunakan MVC, MVC dan Repository, Yang akan diuji menggunakan Apache JMeter untuk memvisualisasikan user yang mengakses aplikasi

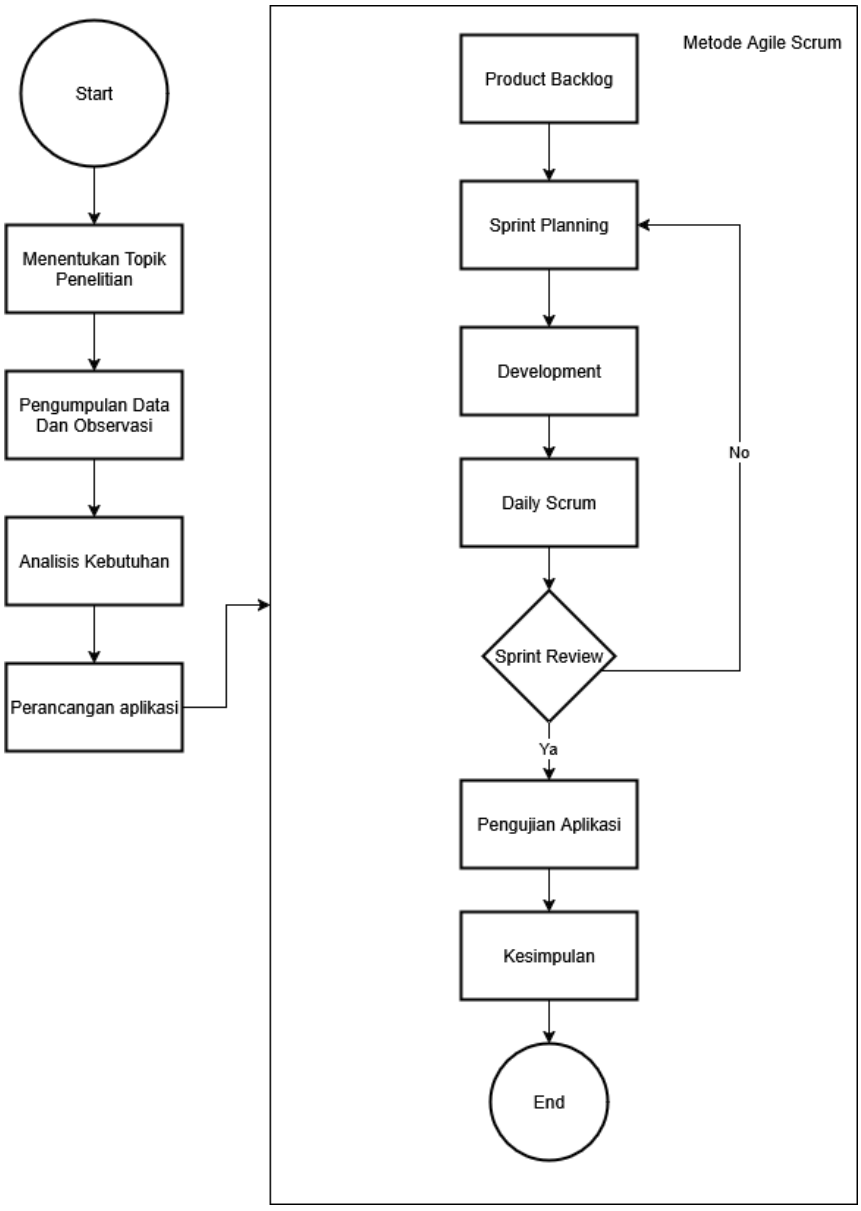
## II. METODE PENELITIAN

Penelitian ini menerapkan pendekatan Repository Pattern dengan pendekatan agile scrum yang berfokus pada pengelolaan sistem piutang dagang berbasis website Pada tahap ini, peneliti akan melakukan pengumpulan data dan observasi terkait sistem yang berjalan di PT Tayoh Sarana Sukses, khususnya sistem yang berkaitan dengan pengelolaan piutang dagang. Sistem yang akan dikembangkan dalam penelitian ini dirancang berbasis website, selain itu observasi ini bertujuan memahami alur proses bisnis tentang sistem piutang dagang yang sudah berjalan

Metode Pengembangan perangkat lunak yang digunakan dalam penelitian ini adalah scrum, salah satu framework dari agile. Scrum ini dipilih karena mendukung proses pengembangan perangkat lunak secara interaktif dan incremental, serta dalam pengembangan ini memungkinkan fleksibilitas terhadap perubahan-perubahan yang terjadi ketika kebutuhan di tengah pengembangan perangkat lunak

Scrum berjalan dalam siklus sprint yaitu interaksi waktu yang dimana pengembangan sistem difokuskan pada penyelesaian bagian-bagian tertentu pada produk. Dalam penelitian ini setiap sprint yang berlangsung akan ditentukan sesuai dengan waktunya. Sehingga setiap sprint planning menentukan fitur yang akan dikembangkan pada produk tersebut, dan dilanjutkan dengan Daily Scrum untuk memonitoring progress harian, sehingga setiap sprint akan mereview hasil yang telah dibuat apakah sesuai dengan fitur yang akan digunakan nantinya dan dapat meminimalisir bug pada aplikasi. Sprint Retrospective untuk mengevaluasi proses pengembangan dan menentukan perbaikan di sprint selanjutnya

Pada Tahap ini peneliti mulai Menyusun rancangan aplikasi dari sistem yang akan dibangun, yang berdasarkan kebutuhan sistem sebelumnya. Tahapan Penelitian ditunjukan pada Gambar 1




Gambar 1 Tahapan Penelitian

*A. Tahapan Penelitian*

1. Menentukan Topik Penelitian

Langkah pertama dalam penelitian ini adalah menentukan topik yang ingin diangkat, Topik yang ingin dalam penelitian ini adalah Perancangan Aplikasi Menggunakan Metode Repository Design Pattern Studi kasus Piutang Dagang.

2. Pengumpulan Data dan Observasi

Pada tahap ini, peneliti akan melakukan pengumpulan data dan observasi terkait sistem yang berjalan di PT Tayoh Sarana Sukses, khususnya sistem yang berkaitan dengan pengelolaan piutang dagang. Sistem yang akan dikembangkan dalam penelitian ini dirancang berbasis website

3. Analisis Kebutuhan

Dari hasil analisis kebutuhan, peneliti memperoleh peta lengkap tentang apa saja yang harus dikerjakan dalam pengembangan sistem. Semua kebutuhan ini akan dimasukkan ke dalam product backlog untuk kemudian direncanakan pengerjaannya di setiap sprint menggunakan metode Agile Scrum.

*B. Perancangan Aplikasi*

1. Product Backlog

Pada tahap ini peneliti akan Menyusun product Backlog sebagai daftar prioritas untuk kebutuhan sistem seperti fitur dan tugas yang akan dikembangkan dalam pengembangan aplikasi

2. Sprint Planning

Pada Tahap ini merupakan tahap penting dalam metode agile scrum yang Dimana peneliti akan menentukan pekerjaan atau fitur apa saja di product Backlog yang akan dikerjakan pada setiap sprint

3. Development

Peneliti akan melakukan development aplikasi yang sudah direncanakan di setiap sprint planning. Dalam penelitian ini, development mencakup pengembangan aplikasi piutang dagang berbasis website menggunakan framework Laravel . Arsitektur yang dipakai adalah kombinasi Model View-Controller (MVC) dengan tambahan Repository Design Pattern

4. Daily Scrum

Melakukan proses development berdasarkan setiap sprint yang sudah dikerjakan oleh peneliti

5. Sprint Review

Pada tahap ini akan melakukan review atau proses kerja dalam setiap sprint apakah sudah sesuai dengan kebutuhan sistem serta mengatur kompleksitas kode dengan menggunakan Repository Pattern ini

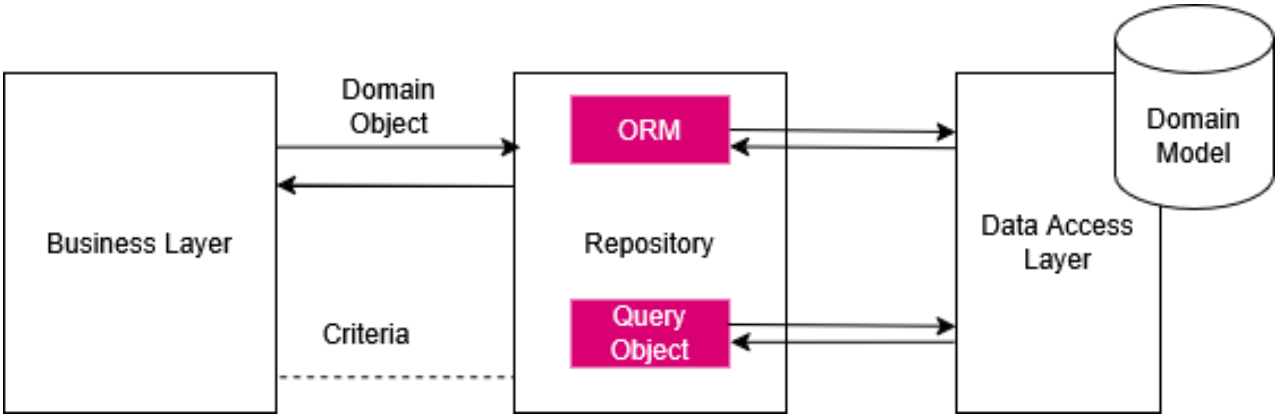
6. Pengujian Aplikasi

Pada tahap ini pengujian aplikasi, setelah sprint telah sesuai yang diharapkan dan sesuai dengan kebutuhan sistem, maka akan melakukan proses pengujian aplikasi menggunakan Apache JMeter

7. Kesimpulan

Pada tahap ini Kesimpulan dalam perancangan aplikasi yang sesuai yang diharapkan dan dapat melakukan perbandingan aplikasi dan proses penggunaan repository pattern dan tanpa repository pattern itu sendiri, dan melakukan perbandingan keduanya dengan response time Ketika menggunakan Apache JMeter

Repository Pattern adalah pola desain (*design pattern*) yang memisahkan logika akses data atau logic bisnis dalam aplikasi, Yang sebelumnya sering kali query ditulis langsung di controller. Dengan Repository Pattern dibuat lapisan tambahan antara controller dan model. Penerapan Repository Pattern bertujuan untuk mengatasi permasalahan struktur kode yang kurang rapi akibat terlalu banyak query yang ditampung oleh Controller.



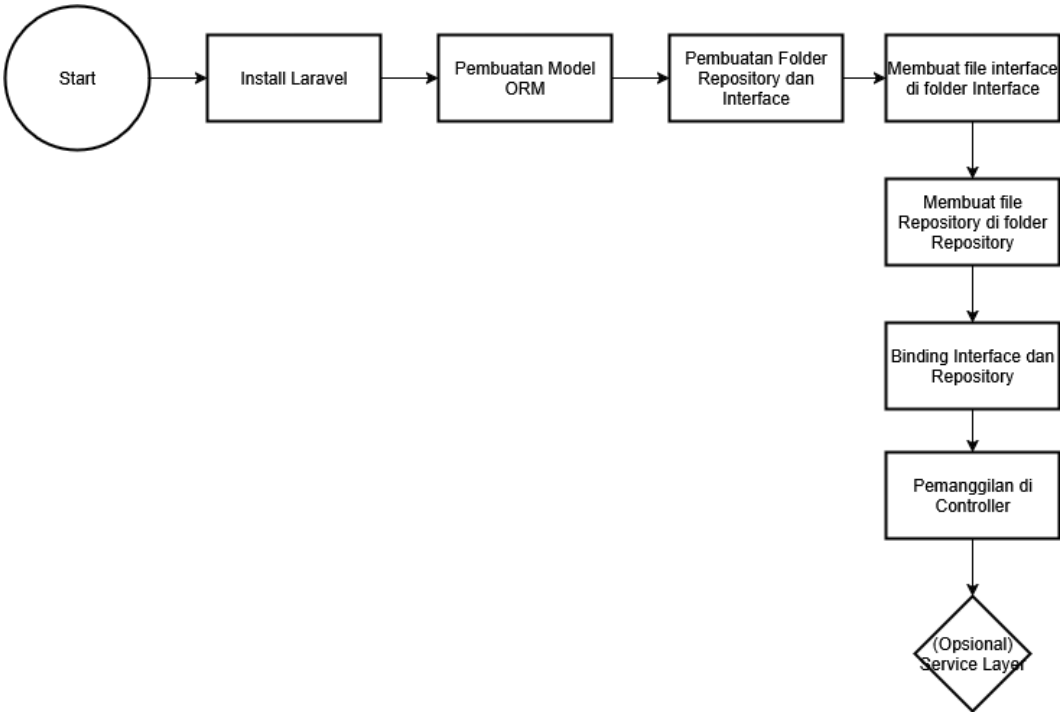

Gambar 2 Repository Desain Pattern

Pada Gambar 2 merupakan tahap perancangan, menggunakan pola desain Repository Pattern yang diterapkan dalam tiga lapisan utama: Business Layer, Repository, dan Data Access Layer

1. Business Layer mengimplementasikan logika bisnis dan berkomunikasi dengan Repository untuk mengelola objek domain dan kriteria

2.Repository berfungsi sebagai penghubung antara Business Layer dan Data Access Layer, menyediakan antarmuka untuk operasi CRUD pada objek domain dan memfilter data berdasarkan kriteria dari Business Layer

3.Data Access Layer (DAL) berinteraksi langsung dengan database, menangani koneksi, eksekusi query, dan transaksi. Data dari DAL dikirim ke Repository untuk diolah sebelum dikembalikan ke Business Layer




Gambar 3 Proses Repository Pattern Laravel

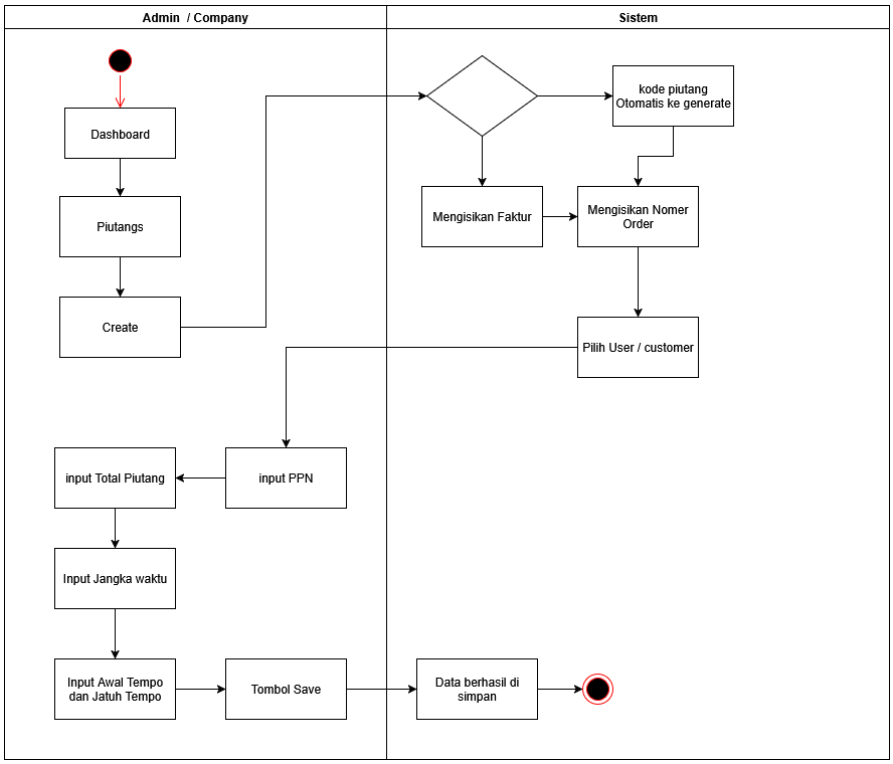
Pada Gambar 3 merupakan alur dari proses penggunaan Repository Pattern menggunakan framework Laravel

1. Install laravel dengan melakukan command atau menggunakan software dari laravel yaitu Laravel Herd konfigurasi ketika install laravel Install composer versi terbaru yaitu versi 2 dan PHP minimal versi 8.2 untuk bisa install laravel versi 12
2. Menggunakan Laravel dengan fitur Eloquent ORM bawaan Laravel, peneliti membangun model yang merepresentasikan masing-masing tabel pada database. Pada tahap ini, relasi antar tabel juga ditentukan, Eloquent ORM membantu mempermudah pemetaan struktur tabel ke dalam bentuk objek yang dapat diakses melalui kode program
3. Dalam membuat folder repository dan interface itu bisa dibuat dimana saja, tapi penulis membuat folder Repository dan interface di App\Repository dan interface ada di App\Repository\Interface
4. Membuat file Interface di setiap model yang ada, jadi Interface ini merupakan kontrak abstrak yang mendefinisikan fungsi-fungsi dasar yang harus dimiliki oleh setiap repository
5. Implementasi class repository. Di tahap ini, peneliti membuat class repository yang merealisasikan kontrak yang telah ditetapkan pada interface sebelumnya. Seluruh logika akses data termasuk Create, Read, update, dan delete di satu Repository pada gambar berikut. Hal ini dilakukan untuk meminimal keberadaan query kode di dalam satu Controller sehingga dapat tetap fokus pada alur kerja (Workflow
6. Pada Tahap ini interface dan Repository didaftarkan (Binding) di laravel service provider bagian file AppServiceProvider atau membuat provider baru untuk menampung Repository Pattern. Pada tahap ini bisa bersifatnya opsional jika melakukan dependency injection menggunakan interface maka harus melakukan binding di AppServiceProvider. Tapi jika melakukan dependency injection langsung menggunakan Repository maka tidak perlu
7. Setelah membuat Repository dan Interface maka selanjutnya melakukan pemanggilan di controller, untuk pada gambar berikut merupakan contoh penggunaan dependency Injection dengan livewire. Tapi ketika tidak pakai livewire hanya laravel biasa maka menggunakan __construct di controller untuk melakukan dependency Injection
8. Membuat folder terpisah dari repository dan interface, masih di struktur App\Service. Layer service ini bersifat optional dan digunakan untuk mengelola logika bisnis atau query kode yang melibatkan lebih dari satu repository. Selain itu layer service digunakan untuk mengelola service dari pihak ketika, dalam penelitian ini akan menggunakan Midtrans sebagai payment gateway pihak ketiga, Untuk mengelola WebHook API untuk Midtrans

*C. Workflow Diagram*

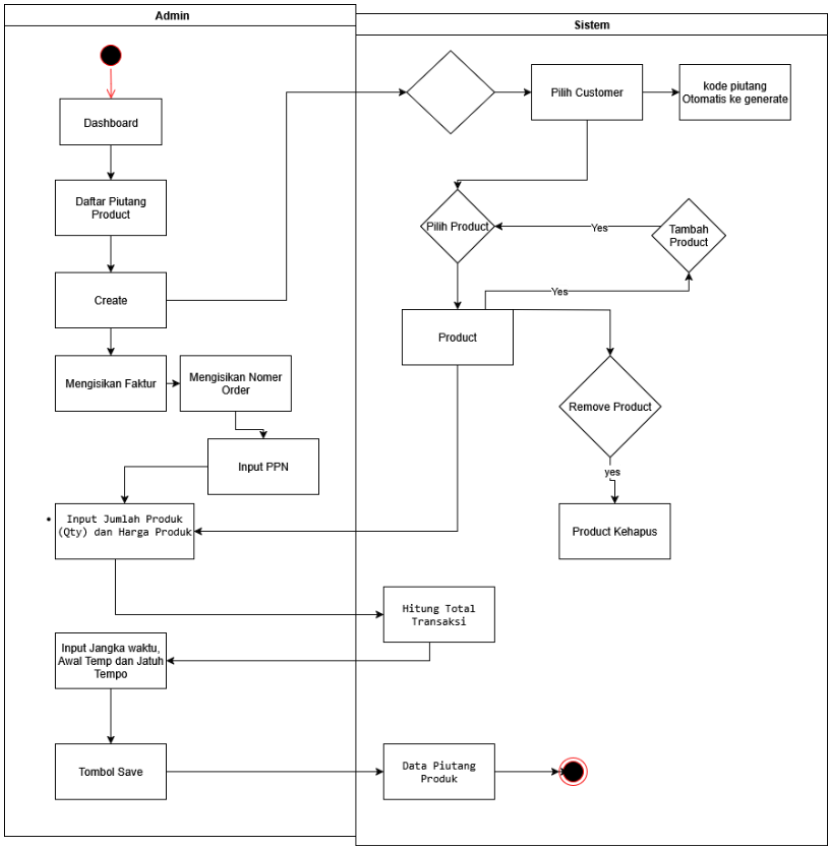

Gambar 4 Use Case Diagram

Pada Gambar 4 use case diagram yang menjelaskan alur proses sistem pengelolaan piutang dalam sistem, yang dimulai dari user login hingga proses pelaporan dan riwayat pembayaran piutang




Gambar 5 Activity Diagram Piutang

Gambar 5 merupakan fitur piutangs melalui halaman dashboard. Pada tahap ini pengguna dapat melihat daftar piutang yang sudah ada serta membuat data piutang baru berdasarkan customer pilih pengguna akan memasukan informasi terkait piutang seperti, Total Piutang, PPN, Jangka Waktu, Nomor Order, Nomor Faktur, memilih customer, Tanggal Transaksi, dan Tanggal Jatuh Tempo. Sistem akan melakukan secara otomatis ketika melakukan create piutang yang akan menghasilkan kode piutang




Gambar 6 Activity diagram Piutang Product

Gambar 6 menggambarkan Alur proses piutang products dimulai dari daftar piutang products. Pada halaman ini pengguna dapat melihat daftar piutang product yang telah disimpan oleh sistem, serta melakukan pembuatan data piutang baru berdasarkan pelanggan (customer) dan produk tertentu. Pengguna mengisikan data Nomor Faktur, Nomor Order, memilih customer yang sudah mempunyai kode product, Pengguna juga diminta juga mengisikan data PPN, Jangka Waktu, serta Tanggal Transaksi, dan Tanggal Jatuh Tempo, setelah semua informasi terisi, sistem akan menghitung total piutang berdasarkan product yang telah dimasukan, setelah itu piutang products akan disimpan dan menghasilkan kode piutang yang akan di generate oleh sistemPengujian Performa

Dalam penelitian ini akan melakukan pengujian performa sistem yang dilakukan menggunakan alat bantu Apache JMeter. peneliti dapat mensimulasikan aktivitas pengguna virtual yang melakukan permintaan (*request*) ke aplikasi. Sehingga dalam proses pengujian ini akan dilakukan ketika aplikasi menggunakan Repository dan tanpa Repository

Pada Tabel 1 merupakan skenario pengujian performa dilakukan untuk mengukur seberapa baiknya sistem dapat menangani pengguna secara bersamaan. Apache JMeter sebagai alat bantu untuk simulasikan terhadap berbagai jumlah pengguna yang mengakses sistem dalam jangka waktu singkat. Number of Threads (Users): Menentukan jumlah virtual user yang akan mengakses sistem secara bersamaan. Ramp-Up Period (Seconds): Waktu yang dibutuhkan untuk men-start semua thread. Nilai 1 detik berarti semua user akan memulai secara bersamaan dalam waktu yang sangat singkat. Loop Count: Jumlah pengulangan request per user. Dalam hal ini, setiap user hanya melakukan 1 kali request (1 loop).

Tabel 1
Skenario Testing Apache JMeter

| | | |
|---|---|---|
| Skenario 1 | Number of Threads (users) | 10 |
| | Ramp-up period (seconds) | 1 |
| | Loop count | 1 |
| Skenario 2 | Number of Threads (users) | 100 |
| | Ramp-up period (seconds) | 1 |
| | Loop count | 1 |
| Skenario 3 | Number of Threads (users) | 500 |
| | Ramp-up period (seconds) | 1 |
| | Loop count | 1 |

Tabel 2
EndPoint API

| API | URL | HTTP Method | Keterangan |
|---|---|---|---|
| **Login** | /api/auth/login | POST | User melakukan login |
| **Authentication** | /api/auth/user | GET | Menampilkan user yang sedang login |
| **User All** | /api/auth/users | GET | Menampilkan semua data users |
| **Categories** | /api/category/categories | GET | Menampilkan semua data Category |
| **Products** | /api/product/products | GET | Menampilkan semua data Products |
| **Roles** | /api/role/roles | GET | Menampilkan semua data Roles |
| **Permissions** | /api/permission/permissions | GET | Menampilkan semua data Permissions |
| **Transactions** | /api/transaction/transactions | GET | Menampilkan semua data Transactions |
| **Piutangs** | /api/piutang/piutangs | GET | Menampilkan semua data Piutangs |
| **Piutang Products** | /api/piutang-product/piutangs | GET | Menampilkan semua data Piutang Products |

Pada Tabel 2 menggambarkan beberapa endpoint API utama dalam sistem manajemen piutang, yang digunakan untuk proses otentikasi, manajemen data master, transaksi, dan piutang. Setiap endpoint menggunakan metode HTTP yang sesuai dengan fungsinya, baik untuk pengambilan data (GET) maupun pengiriman data.

## III. HASIL DAN PEMBAHASAN

Aplikasi ini dirancang untuk sistem mengelola piutang dagang berbasis website. Proses perancangan sistem ini dilakukan menerapkan Model-View-Controller (MVC) sebagai arsitektur utama dan Repository Design Pattern sebagai mengelola akses data atau logika bisnis dalam proses pengembangan aplikasi, yang dimana proses pengembangan aplikasi banyaknya pengembang aplikasi dalam menghadapi kompleksitas kode atau duplikasi logika bisnis dalam aplikasi

Dalam proses Sistem Piutang ini juga menyertakan fitur notifikasi otomatis terhadap piutang yang mendekati jatuh tempo, serta mengintegrasikan metode pembayaran digital melalui midtrans, dan melakukan perjanjian dengan MoU sesuai dengan yang disepakati,

*A. Implementasi Repository Pattern*

```
public function paginateFilteredNotProducts(
        $search = null,
        $customerFilter = null,
        $status = null,
        $year = null,
        $month = null,
        $sortBy = 'newest',
        $perPage = 10
    ) {
        return $this->getFilteredQueryNotProducts(
            $search,
            $customerFilter,
            $status,
            $year,
            $month,
            $sortBy,
        )->paginate($perPage);
```

1. Sebuah antarmuka (interface) yang mendefinisikan kontrak untuk repository pada modul piutang dalam aplikasi pengelolaan piutang dagang. Interface PiutangInterface berisi deklarasi fungsi-fungsi yang wajib diimplementasikan oleh kelas PiutangRepositor

```
interface PiutangInterface
{
    public function getFilteredQueryNotProducts(
        ?string $search = null,
        ?string $customerFilter = null,
        ?string $status = null,
        ?int $year = null,
        ?int $month = null,
        string $sortBy = 'newest'
    ): Builder;

    public function paginateFilteredNotProducts(
        $search = null,
        $customerFilter = null,
        $status = null,
        $year = null,
        $month = null,
        $sortBy = 'newest',
        $perPage = 10
    );
    public function generateKodePiutang(): string;
    public function createPiutang(array $data): Piutang;
    public function update(Piutang $piutang, array $data): bool;

}
```

2. Dengan menerapkan interface, setiap fungsi pengelolaan piutang akan menyediakan fungsionalitas yang konsisten, mengurangi risiko duplikasi kode, dan memastikan bahwa proses pengembangan serta pemeliharaan aplikasi dapat dilakukan dengan lebih efisien dan terstruktur

```
public function register(): void
     {
         $this->app->singleton(CategoryInterface::class,
  CategoryRepository::class);
         $this->app->singleton(CustomerInterface::class,
  CustomerRepository::class);
         $this->app->singleton(PermissionInterface::class,
  PermissionRepository::class);
         $this->app->singleton(PiutangInterface::class,
  PiutangRepository::class);
         $this->app->singleton(ProductInterface::class,
```

```
  ProductRepository::class);
        $this->app->singleton(RoleInterface::class, RoleRepository::class);
        $this->app->singleton(SettingInterface::class,
  SettingRepository::class);
        $this->app->singleton(TransactionInterface::class,
  TransactionRepository::class);
        $this->app->singleton(UserInterface::class, UserRepository::class);
    }
```

3. Pada tahap selanjutnya mendaftarkan Interface dan Repository atau bisa disebut binding di laravel di AppServiceProvider atau membuat file baru dengan RepositoryProvider. Pada tahap ini bisa bersifatnya opsional jika melakukan dependency injection menggunakan interface maka harus melakukan binding di AppServiceProvider. Tapi jika melakukan dependency injection langsung menggunakan Repository maka tidak perlu

```
protected PiutangInterface $piutangRepo;
    public function boot(PiutangInterface $piutangRepo)
    {
        $this->piutangRepo = $piutangRepo;
    }
// cara pakai Repository
$piutangs = $this->piutangRepo->paginateFilteredNotProducts(
            $this->search,
            $this->customerFilter,
            $this->status,
            $this->years,
            $this->months,
            $this->sortBy,
            $this->perPage
        );
```

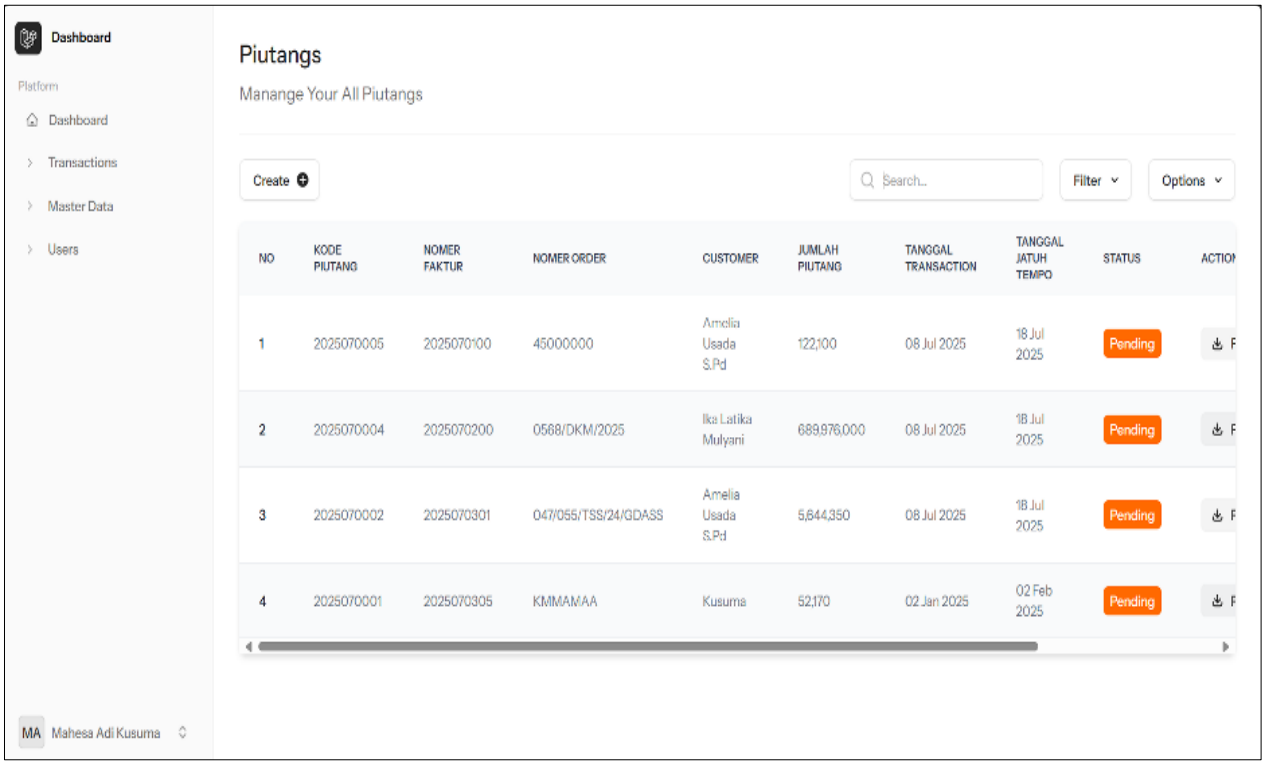
4. Proses pemanggilan repository kedalam controller dilakukan melalui Dependency Injection menggunakan livewire dengan memanfaatkan repository yang telah disediakan fungsi-fungsi untuk mengambil data piutang atau telah dibuat sebelumnya

```
class MidtransService
{
    protected $transactionRepository;
    protected $piutangRepository;
    public function __construct(TransactionRepository $transactionRepository,
PiutangRepository $piutangRepository)
    {
        $this->transactionRepository = $transactionRepository;
        $this->piutangRepository = $piutangRepository;
        $this->configureMidtrans();
    }
    private function configureMidtrans()
    {
        Config::$serverKey = config('midtrans.server_key');
        Config::$isProduction = config('midtrans.is_production');
        Config::$isSanitized = config('midtrans.is_sanitized');
        Config::$is3ds = config('midtrans.is_3ds');
    }
}
```

5. Membuat folder terpisah dari repository dan interface, masih di struktur App\Service. Layer service ini bersifat optional dan digunakan untuk mengelola logika bisnis atau query kode yang melibatkan lebih dari satu repository. Selain itu layer service digunakan untuk mengelola service dari pihak ketiga, dalam pengembangan aplikasi ini menggunakan Midtrans sebagai pihak ketiga dalam proses payment gateway, untuk dapat mengelola WebHook API untuk midtrans

*B. Implementasi Aplikasi*

Pada tahap ini akan mengimplementasikan aplikasi sesuai dengan metode scrum atau product backlog yang sudah di analisis dalam proses pengembangan aplikasi piutang dagang. Berikut gambaran aplikasi yang sudah dikembangkan.

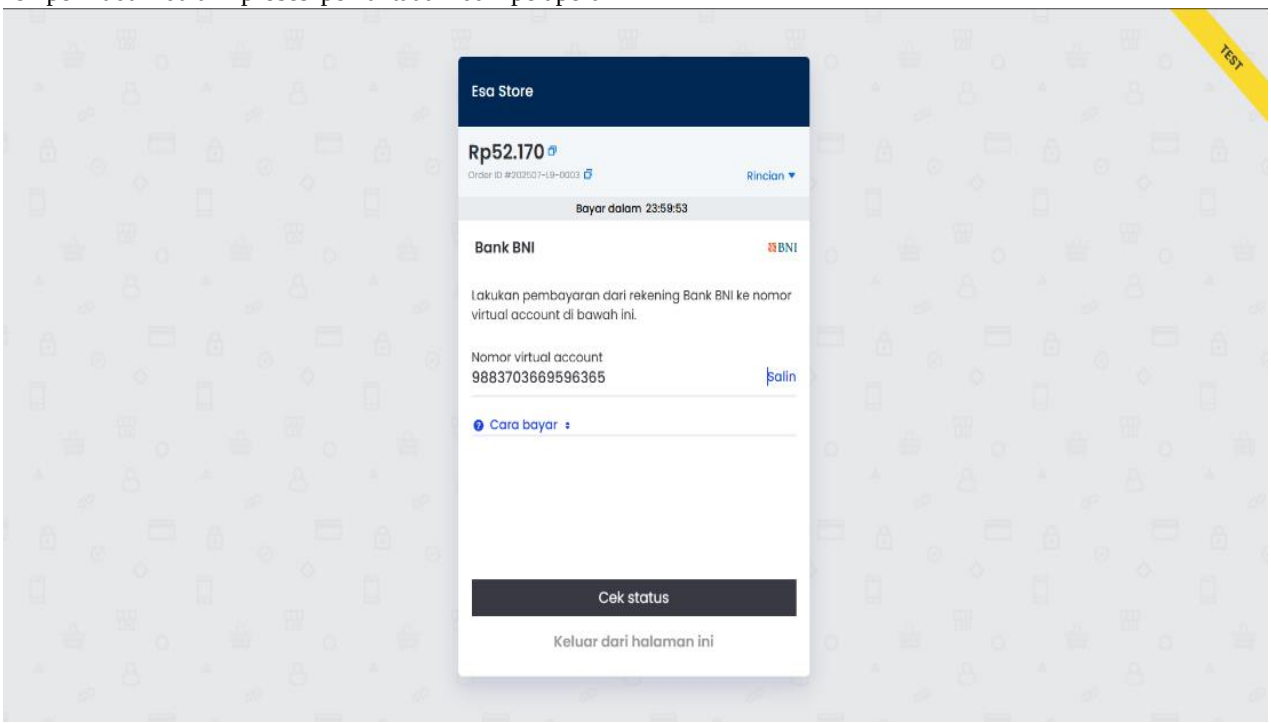



Gambar 7 Halaman Piutang

1. Halaman Piutang

Menampilkan halaman piutangs yaitu halaman yang dikelola oleh user admin atau company dalam melihat dan mengelola seluruh data piutang dari customer yang sudah tercatat di dalam sistem. Halaman ini dapat mempermudah dalam proses pemantauan dan pelaporan kewajiban customer terhadap piutang mereka masing-masing, terdapat fitur filter data dan export data dalam menyusun laporan dan menganalisa piutangs.

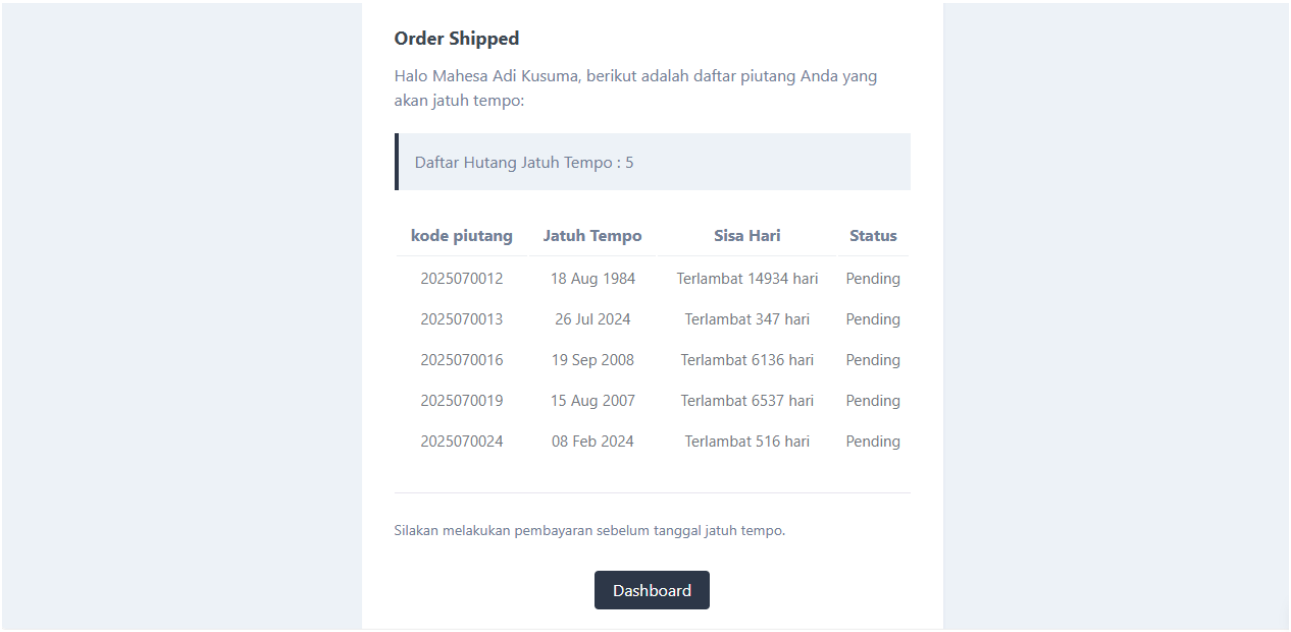
2. Payment Gateway Midtrans




Gambar 8 Payment Gateway Midtrans

Merupakan pembayaran piutang secara langsung atau lunas, dengan pembayaran langsung dengan payment gateway Midtrans payment. merupakan pembayaran payment gateway midtrans dan tidak perlu konfirmasi melalui admin untuk mengubah status piutang menjadi success karena, di aplikasi sistem piutang dagang ini sudah dikelola langsung menggunakan Application Programming Interface (API) WebHook Midtrans yang mengubah status piutang.

3. Notifikasi email piutang




Gambar 9 Notifikasi email piutang

Gambar 9 merupakan format notifikasi email piutang customer, notifikasi ini akan dikirimkan ke masing-masing customer yang mempunyai piutang dengan status piutang yang masih pending dan piutangnya jatuh tempo kurang dari 15 hari.

*C. Pengujian Aplikasi*

Pada tahap ini, sistem akan melakukan pengujian terhadap aplikasi yang telah dikembangkan dengan memanfaatkan Apache JMeter Pengujian ini dilakukan melalui API dari sistem aplikasi, dengan fokus pada pengujian performa ketika menggunakan Repository Pattern dan tanpa menggunakan Repository Pattern dalam proses pengembangan aplikasi.

Pengujian performa menggunakan Apache JMeter menunjukkan bahwa penerapan Repository Design Pattern pada framework Laravel memberikan performa yang lebih stabil. Repository Pattern menghasilkan waktu respons rata-rata sebesar 1696.333 ms, yang lebih cepat dibandingkan dengan pendekatan tanpa Repository Pattern yang memiliki waktu respons rata-rata sebesar 2081.667 ms.

Tabel 3 menampilkan hasil pengujian performa terhadap endpoint API dari sistem Piutang Non Repository Pattern menggunakan Apache JMeter. Pada skenario 10 dan 100 pengguna, sistem menunjukan stabilitas sistem dengan menunjukan error 0%, namun pada skenario 500 performa sistem mengalami penurunan dan error sampai dengan 86.28% dengan mencapai response time 23977 ms dan dengan rata-rata response time 3033 ms. Dan Rata-rata waktu respon dari ketiga skenario adalah 2081.667 ms.

Tabel 3
Pengujian Piutang Non Repository Pattern

| Piutang Non Repository Pattern | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Samples** | Average | Min | Max | Std.Dev | Error % | Throughput | Received KB/sec |
| **10** | 297 | 26 | 1256 | 270.3 | 0% | 27.0/sec | 271.3 |
| **100** | 2915 | 729 | 14393 | 2816.82 | 0% | 32.2/sec | 323.75 |
| **500** | 3033 | 2 | 23977 | 2892.69 | 86.28% | 110.5/sec | 228.57 |
| **Rata-Rata Average** | 2081.667 | | | | | | |

Tabel 4 menampilkan hasil pengujian performa terhadap endpoint API dari sistem Piutang Repository Pattern menggunakan Apache JMeter, Pada skenario 10 dan 100 pengguna, sistem menunjukan stabilitas sistem dengan menunjukan error 0%, namun pada skenario 500 performa sistem mengalami penurunan dan error sampai dengan 85.46% dengan mencapai response time 17339 ms dan dengan rata-rata response time 2618 ms. Dan Rata-rata waktu respon dari ketiga skenario Adalah.

Tabel 4
Pengujian Piutang Repository Pattern

| Piutang Repository Pattern | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Samples** | Average | Min | Max | Std.Dev | Error % | Throughput | Received KB/sec |
| **10** | 173 | 31 | 657 | 134.73 | 0% | 42.2/sec | 350.73 |
| **100** | 2298 | 620 | 10574 | 2106.72 | 0% | 40.8/sec | 339.08 |
| **500** | 2618 | 0 | 17339 | 2097.47 | 85.46% | 142.2/sec | 249.53 |
| **Rata-Rata Average** | | | | 1696.333 | | | |

## IV. KESIMPULAN DAN SARAN

Berdasarkan hasil perancangan dalam menerapkan sistem piutang berbasis website dengan menggunakan Laravel livewire serta menggunakan Repository Pattern dapat meningkatkan dalam proses pengembangan sistem yang lebih modular, terstruktur dan dapat mengurangi duplikasi kode, sehingga memudahkan dalam proses pengelolaan kode dan pemeliharaan aplikasi.

Berdasarkan hasil pengujian performa aplikasi menggunakan Apache JMeter, dapat disimpulkan bahwa penerapan Repository Design Pattern dalam pengembangan aplikasi berbasis Laravel memberikan dampak yang positif terhadap kestabilan sistem. Pengujian dilakukan dengan membandingkan dua pendekatan, yaitu dengan menggunakan Repository Pattern dan tanpa Repository Pattern, melalui API sistem aplikasi

## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih sebesar-besarnya kepada pihak-pihak yang mendukung proses dalam pembuatan penelitian ini dan penulis mengucapkan terimakasih kepada perusahaan terkait dalam proses penelitian serta dalam proses pengembangan aplikasi.

## DAFTAR PUSTAKA

[1] S. W. Wahyuning and A. Febriana, “Sistem Informasi Akuntansi Pengendalian Piutang Dagang Berbasis Web,” 2023. doi: 10.51903/dinamika.v3i1.327.

[2] Meyla Nur Vita Sari, “Pengaruh Periode Piutang Dagang Terhadap Tingkat Profitabilitas pada Perusahaan Jasa Perhotelan,” *Al-Iqtishod J. Ekon. Syariah*, vol. 4, no. 2, pp. 91–99, Dec. 2022, doi: 10.51339/iqtis.v4i2.717.

[3] S. Loviriani, L. Syafitri, and A. Munandar, “Analisis Faktor-Faktor Penyebab Piutang Tak Tertagih dan Upaya Penanggulangan serta Penyelesaian Piutang Tak Tertagih,” *J. Econ. Manag. Sci.*, pp. 206–209, 2023, doi: 10.37034/jems.v5i4.26.

[4] K. S. Saif, “Smart Report design using SET Hierarchies and Bex queries in SAP S/4 HANA,” *Int. Sci. J. Eng. Manag.*, vol. 04, no. 01, pp. 1–6, Aug. 2025, doi: 10.55041/isjem00104.

[5] B. R. P. Surya, A. P. Kharisma, and N. Yudistira, “Perbandingan Kinerja Pola Perancangan MVC, MVP, dan MVVM Pada Aplikasi Berbasis Android (Studi kasus _ Aplikasi Laporan Hasil Belajar Siswa SMA BSS),” 2020. [Online]. Available: http://j-ptiik.ub.ac.id

[6] F. F. Anhar and F. T. Anggraeny, “Implementasi Clean Architecture Mvvm Dan Repository Pattern Untuk Pengembangan Aplikasi Android Jual Beli Barang Bekas ‘Secondhand,’” Jun. 2022. doi: 10.33005/scan.v17i2.3317.