Rancang Bangun Aplikasi *Order* untuk *Cafe* dengan Penerapan Teknik *Caching* Menggunakan *Redis* di *Backend.*

Muhammad Rifqi[1], Hendrawaty [2*], Amirullah[3]

[1,2,3] *Jurusan Teknologi Informasi dan Komputer Politeknik Negeri Lhokseumawe*
Jln. B.Aceh Medan Km.280 Buketrata 24301 INDONESIA
[1]muhammadrifqi24062002@mail.com
[2]hendrawaty@pnl.ac.id
[3]amir@pnl.ac.id

Abstrak*— Dalam era industri makanan dan minuman yang terus berkembang, *cafe* menjadi pusat kegiatan sosial dan bisnis bagi banyak orang, didorong oleh perubahan gaya hidup urban yang semakin sibuk serta kebutuhan akan tempat yang nyaman untuk berkumpul, bekerja, atau bersantai. Agar dapat bersaing, *cafe* perlu menawarkan pelayanan yang efisien. Namun, tantangan utama yang dihadapi adalah lamanya waktu pelayanan dan manajemen pesanan yang kurang efisien, terutama pada jam sibuk. Pemesanan digital dapat membantu mengurangi antrian dan mempercepat layanan, tetapi memerlukan sistem penyimpanan data yang cepat dan andal. Teknologi *caching* seperti *Redis* dapat meningkatkan kinerja dan responsivitas aplikasi pemesanan kafe, mempermudah pengelolaan pesanan, serta memperbaiki pengalaman pelanggan. Penelitian ini bertujuan untuk merancang aplikasi pemesanan untuk *cafe* yang memudahkan proses pemesanan dengan menerapkan *Redis* sebagai mekanisme *caching* di *backend*. Hasil pengujian menunjukkan bahwa implementasi *Redis* berhasil meningkatkan kecepatan akses sebagian besar *endpoint API*. Pengujian tanpa *Redis* menunjukkan waktu respons yang fluktuatif dengan nilai puncak 142,86 ms, sementara pengujian dengan *Redis* menunjukkan peningkatan performa yang lebih konsisten dengan nilai puncak 166,67 ms. Secara keseluruhan, penerapan *Redis* mampu mempercepat waktu respons rata-rata dan mengurangi waktu antrian pada sebagian besar skenario pengujian, sehingga memperbaiki pengalaman pengguna secara keseluruhan.
Kata kunci—*Cafe*, Pemesanan Digital, *Redis*, Teknologi *Caching*, *Responsivitas*, *Backend*, *Database*.

Abstract*— *In the ever-evolving food and beverage industry, cafes have become central hubs for social and business activities, driven by increasingly busy urban lifestyles and the need for comfortable spaces to gather, work, or relax. To remain competitive, cafes must offer efficient service. However, the main challenges faced include prolonged service times and inefficient order management, especially during peak hours. Digital ordering can help reduce queues and speed up service, but it requires a fast and reliable data storage system. Caching technology like Redis can enhance the performance and responsiveness of cafe ordering applications, streamline order management, and improve customer experience. This study aims to design an ordering application for cafes that facilitates the ordering process by implementing Redis as a caching mechanism in the backend. The test results show that the implementation of Redis significantly improved the access speed of most API endpoints. Testing without Redis revealed fluctuating response times with a peak value of 142.86 ms, while testing with Redis demonstrated more consistent performance with a peak value of 166.67 ms. Overall, the implementation of Redis effectively reduced average response times and decreased queue times in most testing scenarios, thereby enhancing the overall user experience.
Keywords*— *Cafe, Digital Ordering, Redis, Caching Technology, Responsiveness, Backend, Database.

I. Pendahuluan

Dalam era industri makanan dan minuman yang berkembang pesat, *cafe* telah menjadi pusat kegiatan sosial dan bisnis bagi banyak individu. Fenomena ini didorong oleh pergeseran gaya hidup urban yang semakin sibuk dan kebutuhan akan tempat yang nyaman untuk berkumpul, bekerja, atau sekadar bersantai[1]. Dalam konteks ini, pelayanan yang efisien dan pengalaman pelanggan yang memuaskan menjadi kunci keberhasilan bagi pemilik *cafe* dalam menjaga daya saingnya di pasar yang kompetitif.

Meskipun *cafe* telah menjadi tempat populer, pengelolaannya masih menghadapi beberapa tantangan dalam hal pemesanan dan pelayanan pelanggan. Salah satu masalah utama yang sering muncul adalah lamanya waktu yang diperlukan untuk melayani pesanan, terutama pada jam-jam sibuk[2]. Selain itu, manajemen pesanan yang efisien dan akurat juga menjadi fokus utama dalam meningkatkan produktivitas dan mengurangi kesalahan dalam proses pengelolaan pesanan. Masalah lain yang sering dihadapi termasuk antrian panjang yang menurunkan pengalaman pelanggan, manajemen data yang tidak efisien dengan sistem pencatatan manual, serta kurangnya *respons server* dalam menampilkan menu dan memproses pemesanan[3].

Dalam upaya mengatasi tantangan-tantangan tersebut, penggunaan teknologi informasi telah menjadi solusi yang semakin populer di industri *cafe*[4]. Salah satu solusi teknologi yang banyak diterapkan adalah pengembangan aplikasi *order* untuk *cafe*. Aplikasi semacam ini memungkinkan pelanggan untuk memesan makanan dan minuman secara langsung melalui perangkat mereka sendiri, tanpa harus menunggu antrian atau berinteraksi langsung

dengan petugas *cafe*[3]. Selain itu, aplikasi ini juga memungkinkan pemilik cafe untuk mengelola pesanan dengan lebih efisien dan akurat melalui sistem manajemen pesanan terintegrasi.

Dalam konteks pengembangan aplikasi order untuk *cafe*, penggunaan teknik caching telah menjadi fokus utama dalam upaya untuk meningkatkan kinerja dan responsivitas aplikasi. Teknik *caching* memungkinkan aplikasi untuk menyimpan data yang sering diakses dalam memori *cache*, sehingga mengurangi waktu yang diperlukan untuk mengambil data dari sumber aslinya[4]. Salah satu teknologi *caching* yang banyak digunakan dalam pengembangan aplikasi adalah *Redis*, sebuah sistem basis data berkinerja tinggi yang dirancang khusus untuk *caching* dan penyimpanan data yang cepat dan efisien[5].

Beberapa penelitian sebelumnya menunjukkan bahwa teknologi *caching* menggunakan *Redis* mampu meningkatkan kinerja aplikasi secara signifikan. Misalnya, Meriani et al. (2022) mengidentifikasi efektivitas *Redis* dalam meningkatkan kinerja aplikasi MaBaUS[6]. Zulfa et al. (2020) menerapkan strategi *caching* dengan *Redis* untuk mempercepat akses data relasional[5]. Penelitian oleh Iskandar et al. (2022) menunjukkan peningkatan efisiensi dan kinerja sistem E-Office dengan penggunaan Redis[7].

Penelitian ini berfokus pada pengembangan aplikasi *order cafe* dengan menerapkan teknik *caching* menggunakan *Redis* di *backend*. Dengan memanfaatkan teknik ini, diharapkan aplikasi order untuk *cafe* dapat memberikan pengalaman pengguna yang lebih baik dengan meningkatkan kecepatan akses data dan responsivitas aplikasi. Selain itu, penggunaan teknologi *caching* juga diharapkan dapat membantu pemilik *cafe* dalam mengelola pesanan dengan lebih efisien dan akurat, sehingga meningkatkan produktivitas dan mengurangi kesalahan dalam proses pengelolaan pesanan[4]. Dengan demikian, penerapan teknik *caching* menggunakan *Redis* menjadi langkah yang relevan dan strategis dalam memanfaatkan peluang yang ada di pasar cafe yang kompetitif saat ini.

II. Metodologi Penelitian

Beberapa langkah yang dilakukan dalam pengembangan sistem ini, termasuk teknik pengumpulan data, analisis kebutuhan, dan metode yang digunakan, dijelaskan dalam sebagian berikut.

A. Data dan Pengumpulan Data

Metode pengumpulan data yang digunakan dalam penelitian ini meliputi observasi dan wawancara. Observasi dilakukan untuk mempelajari sistem yang sedang berjalan, sedangkan wawancara dilakukan dengan pihak terkait untuk mendapatkan informasi lebih mendalam mengenai kebutuhan sistem.

B. Analisis Kebutuhan Sistem

Analisis sistem dilakukan untuk mengidentifikasi kebutuhan fungsional dan non-fungsional. Kebutuhan fungsional mencakup langkah-langkah yang diperlukan untuk mendukung kelancaran proses dalam sistem, seperti login, registrasi, pengelolaan menu, dan pemesanan. Kebutuhan non-fungsional mencakup spesifikasi perangkat keras dan perangkat lunak yang digunakan.

1) Kebutuhan Fungsional Admin
Login, *logout*, serta menambah, melihat, menghapus, dan mengubah data pemilik *cafe*.

2) Kebutuhan Fungsional *Owner*
Login, *logout*, dan mendaftar pada sistem. Selain itu, dapat menambah, melihat, menghapus, dan mengubah data stan dapur, data kasir, data pelayan, dan data menu, serta melihat laporan penjualan.

3) Kebutuhan Fungsional Stan
login, logout, melakukan *update profile* stan*,* menginput data menu yang ada pada stan*,* dan melihat data *order* dan informasi pesanan.

4) Kebutuhan Fungsional Dapur
Login, *logout*, serta melihat dan mengubah data pesanan.

5) Kebutuhan Fungsional Kasir
Login, *logout*, serta melihat dan mengubah status pembayaran pesanan.

6) Kebutuhan Fungsional Pelayan
Login, *logout*, serta menambah, melihat, menghapus, dan mengubah data pesanan.

7) Kebutuhan Fungsional Customer
Dapat mendaftar pada sistem, login, dan logout. Selain itu, bisa menambah, melihat, menghapus, dan mengubah data pesanan, melihat data menu, serta melihat data *cafe*.

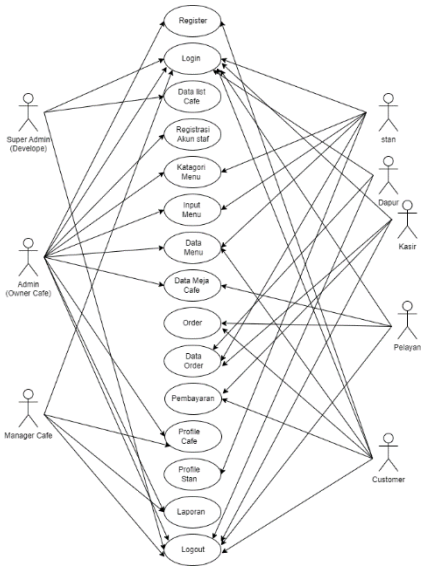
8) Kebutuhan non-fungsional Perangkat Lunak (*Software)*
Windows 11, Google Chrome, Visual Studio Code, Laragon (MySQL, PHP), Laravel, Vue.js, Node.js, dan Redis.

9) Kebutuhan non-fungsional Perangkat Keras (*Hardware)*
Processor 12th Gen Intel(R) Core(TM) i3-1220P 1.50 GHz

C. Rancangan Sistem

Rancangan sistem meliputi use case diagram, activity diagram, dan sequence diagram. Rancangan ini bertujuan untuk memvisualisasikan dan merancang struktur statis dari sistem, serta memperjelas bagaimana komponen-komponen dalam sistem berinteraksi.

1) Use Case Diagram: Pada Gambar 1 menjelaskan tentang peran aktor dari system dimana terdapat delapan aktor yaitu Super Admin (Developer), Admin (Pemilik Cafe), Manajer Cafe, Staf (Stan), Dapur (Staf Dapur), Kasir, Pelayan, Pelanggan. Untuk penjelasan yang lebih rinci mengenai fitur-fitur di dalam sistem yang dilihat pada Tabel 1.

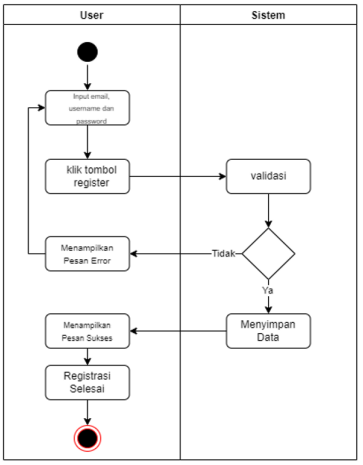


Gambar 1. Use Case Diagram

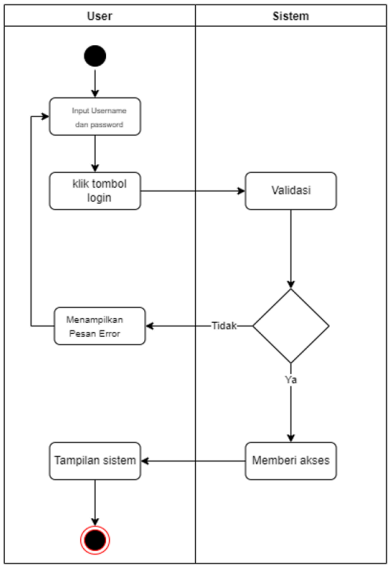
TABEL I
DEFINISI USE CASE DIAGRAM

No	Use Case	Deskripsi
1	Register	Memungkinkan pengguna baru untuk membuat akun.
2	Login	Memungkinkan pengguna untuk mengakses sistem dengan kredensial mereka.
3	Data List Cafe	Mengelola daftar cafe dalam sistem.
4	Registrasi Akun Staf	Memungkinkan pendaftaran anggota staf.
5	Kategori Menu	Mengelola berbagai kategori menu.
6	Input Menu	Memungkinkan penambahan item menu baru.
7	Data Menu	Mengelola item menu yang ada.
8	Data Meja Cafe	Mengelola penugasan meja dan ketersediaannya.
9	Order	Mengelola pesanan pelanggan.
10	Data Order	Mengelola data pesanan.
11	Pembayaran	Mengelola proses pembayaran untuk pesanan.
12	Profile Cafe	Mengelola profil cafe.
13	Profile Stan	Mengelola profil stan
14	Laporan	Menghasilkan laporan tentang berbagai metrik dan aktivitas.
15	Logout	Memungkinkan pengguna untuk keluar dari sistem.

2) *Activity Diagram:* Pada gambar 2 merupakan alur dari *activity diagram register*, yang dimana proses dimulai dengan *user* mengisi *form* data yang disediakan kemudian *user* mengklik tombol *register*. Sistem akan memvalidasi apakah inputan benar atau tidak, jika tidak benar maka sistem akan menampilkan pesan *error* ke *user* dan jika benar maka sistem akan menyimpan data ke dalam *database* dan menampilkan pesan sukses dan registrasi selesai.

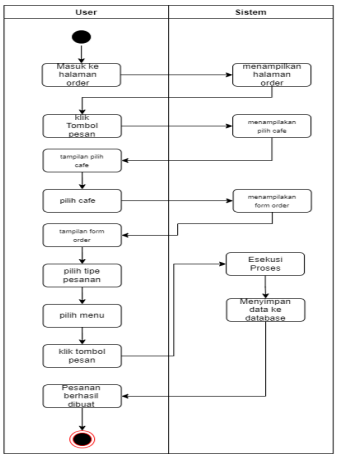


Gambar 2. *Activity* Diagram *Register*



Gambar 3. Activity Diagram *Login*

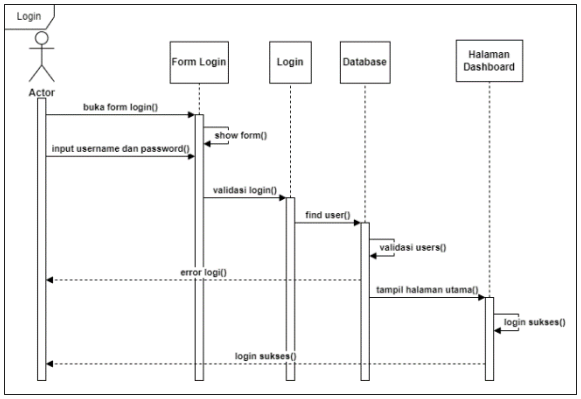
Pada gambar 3 merupakan alur dari *activity* diagram *login* yang dimana *user* memasukkan informasi *login* dan menekan tombol *login* maka sistem melakukan validasi apakah data yang dimasukkan benar atau tidak, jika tidak benar maka sistem akan menampilkan pesan *error* dan *user* memasukkan kembali inputan yang benar dan jika inputan *user* benar maka sistem akan memberikan akses dan akan masuk kedalam sistem.



Gambar 4. Activity Diagram Order

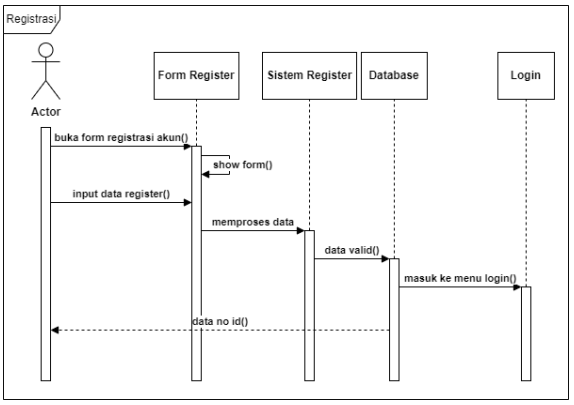
Pada gambar 4 merupakan alur dari activity diagram tambah order yang dimana proses dimulai dengan user masuk ke halaman order, kemudian mengklik tombol pesan yang mengarahkan ke tampilan pilih cafe. User memilih cafe dan diarahkan ke form order di mana mereka memilih tipe pesanan dan menu yang diinginkan. Setelah itu, user mengklik tombol pesan untuk menyelesaikan pesanan. Pada sisi Sistem, setelah user masuk ke halaman order, sistem menampilkan halaman order, form order, dan mengeksekusi proses penyimpanan data ke database. Akhirnya, sistem mengonfirmasi bahwa pesanan berhasil dibuat.

3) Sequence Diagram:



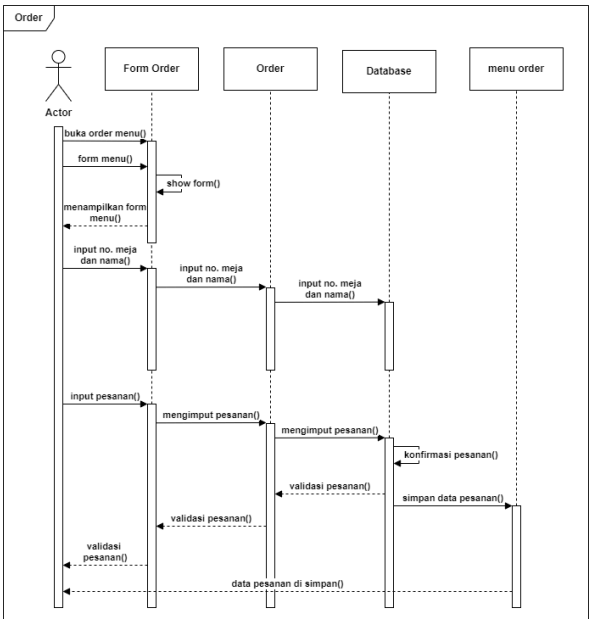
Gambar 5. Sequence Diagram Login

Pada gambar 5 merupakan perancangan *sequence diagram login* yang dimana menggambarkan alur proses *login* dalam aplikasi, dimulai dengan pengguna (Actor) yang membuka *form login*, yang kemudian ditampilkan oleh sistem. Setelah pengguna memasukkan *username* dan *password*, sistem melakukan validasi *login* melalui objek "*Login*", yang kemudian mengirimkan permintaan ke "*Database*" untuk mencari data pengguna. *Database* memvalidasi pengguna dan mengirimkan hasilnya kembali ke objek "*Login*". Jika validasi berhasil, sistem menampilkan halaman utama melalui "Halaman *Dashboard*" dan menginformasikan bahwa *login* berhasil kepada pengguna. Namun, jika terjadi kesalahan, sistem akan menampilkan pesan *error* kepada pengguna.



Gambar 6. Sequence Diagram Registrati

Pada gambar 6 merupakan perancangan *sequence diagram* registrasi yang menggambarkan proses registrasi dalam sebuah sistem. Proses dimulai dengan aktor (pengguna) yang membuka *form registrasi* melalui aksi "buka *form registrasi* akun()". Sistem merespons dengan menampilkan *form* melalui "*show* form()". Pengguna kemudian menginput data registrasi yang dikirim ke sistem untuk diproses melalui "memproses data". Sistem memeriksa validitas data ke *database*; jika data valid, sistem melanjutkan dengan validasi input menggunakan "*data valid()*". Setelah validasi berhasil, sistem mengarahkan pengguna ke menu *login* dengan aksi "masuk ke menu *login()*". Jika tidak ditemukan identifikasi (misalnya, data duplikat atau data yang kurang lengkap), sistem mengirimkan pesan "*data no id()*" untuk memberi tahu aktor bahwa proses registrasi gagal.

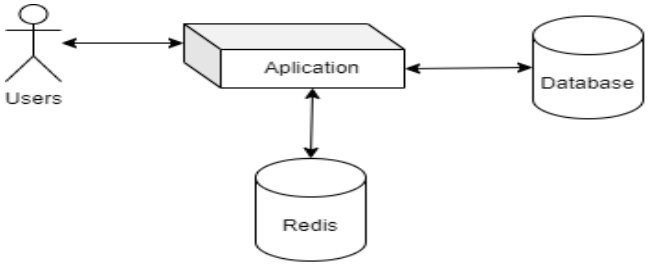


Gambar 7. Sequence diagram Order

Pada gambar 7 merupakan perancangan sequence diagram order yang menggambarkan proses pemesanan menu dalam sistem. Proses dimulai ketika pengguna (Actor) membuka form order menu, yang kemudian ditampilkan oleh sistem. Pengguna memasukkan nomor meja dan nama, yang diteruskan ke objek "Order" dan kemudian ke "Database" untuk disimpan. Selanjutnya, pengguna menginput pesanan, yang juga diteruskan ke "Order" dan "Database". Setelah itu, sistem memvalidasi pesanan dan mengonfirmasi bahwa data pesanan telah disimpan di "Database".

D. Implementasi Teknik Caching dengan Redis

Teknik caching menggunakan Redis diterapkan di backend aplikasi untuk meningkatkan kecepatan akses data dan responsivitas sistem. Alur kerja teknik caching dengan Redis meliputi penyimpanan data yang sering diakses dalam memori cache, sehingga mengurangi waktu yang diperlukan untuk mengambil data dari sumber aslinya.



Gambar 8. Arsitektur Teknik Caching menggunakan Redis

Pada gambar 8 merupakan alur kerja redis yang dimana ketika ada request data dari user ke application, request pertama akan di ambil dari database dan hasil dari database akan disimpan kedalam redis dan dikembalikan datanya ke user. Ketika ada request data kedua dengan data yang sama dari user maka akan menyecek apakah datanya ada di redis jika ada data tersebut maka akan langsung diambil dan dikembalikan ke user dan jika data yang di minta tidak ada pada redis maka akan melakukan proses seperti pertama kali data diminta.

1) *Penerapan Caching Data User:* Gambar 9 merupakan sebuah kode dengan *function userProfile* yang mengambil profil pengguna yang sedang *login*. Dimulai dengan inisialisasi variabel *$status, $code,* dan *$result*. Dalam blok *try*, pengguna yang sedang *login* diambil menggunakan *Auth::user(),* lalu data profil pengguna diambil dari *cache* atau dari database jika tidak ada dalam *cache*, menggunakan *Cache::remember* dengan *key cache* yang terdiri dari nama pengguna dan durasi 150 menit.

```
app > Services > UserService.php > PHP Intelephense > UserService > userLogin
14  class UserService
15  {
16      public function userProfile()
17      {
18          $status = false;
19          $code = 200;
20          $result = null;
21          try {
22              $message = "Get data Profile";
23              $auth = Auth::user();
24              $result = Cache::remember("profile_{$auth->name}_data", now()->addMinute(150), function () use
                ($auth){
25                  return  Models\User::with('user_level')->findOrFail($auth->id);
26              });
27              $status = true;
28          } catch (\Throwable $e) {
29              $code = $e->getCode();
30              $message = $e->getMessage();
31              $result = [
32                  'get_file' => $e->getFile(),
33                  'get_line' => $e->getLine()
34              ];
35          }
36
37          return [
38              'code' => $code,
39              'status' => $status,
40              'message' => $message,
41              'result' => $result
```

Gambar 9. Caching Data Profile Pengguna

Gambar 10 merupakan sebuah kode dengan *function userLogin* yang digunakan untuk menangani proses *login* pengguna, yang dimana kode menerima permintaan *login* dan memulai dengan inisialisasi variabel *$status*, *$code*, dan *$result*. Dalam blok *try*, jika *autentikasi (Auth::attempt)* dengan *email* dan *password* yang diberikan berhasil, pengguna yang sedang login diambil menggunakan *Auth::user()*. Data pengguna tersebut kemudian diambil dari *cache* atau dari *database* jika tidak ada dalam *cache*, menggunakan *Cache::remember* dengan *key cache* yang terdiri dari nama pengguna dan durasi 150 menit. Dalam *closure cache*, data pengguna beserta relasi *user_level* diambil dari *database*. Jika login berhasil, pesan sukses dan status operasi diatur, sementara jika autentikasi gagal, pesan *'Unauthorized'*.

```
14  class UserService
143
144     public function userLogin($request)
145     {
146         $status = false;
147         $code = 200;
148         $result = null;
149         try {
150             if (Auth::attempt(['email' => $request->email, 'password' => $request->password])) {
151                 $user = Auth::user();
152
153                 $result = Cache::remember("user_{$user->name}_data", now()->addMinute(150), function ()
                    use($user){
154                     $getUser =  Models\User::with('user_level')->findOrFail($user->id);
155                     $result['token'] = $this->createToken($user);
156                     $result['user'] = $getUser;
157
158                     return $result;
159                 });
160
161                 $message = 'Succesfully User Login';    "Succesfully": Unknown word.
162                 $status = true;
163             } else {
164                 $message = 'Unauthorized...';
165                 $code = 401;
166             }
```

Gambar 10. Caching Data Pengguna Saat Login

2) *Penerapan Caching Data Cafe:* Gambar 11 merupakan kode *function getCafe* yang dimana mengambil data *cafe* berdasarkan parameter *$id* dan opsional *$ket*, dengan prioritas dari *cache* sebelum mengakses database. Pesan default diatur menjadi "Get data Cafe" dan status operasi dianggap berhasil. Nama cache dibentuk dari kombinasi $id dan $ket, lalu jika $ket bernilai *'user_id'*, data cafe diambil dari cache dengan masa berlaku 150 menit menggunakan *Cache::tags(['get_cafe_data'])->remember()*. Jika data

tidak ditemukan di *cache*, *query* dilakukan ke *database* untuk mencari *cafe* dengan *user_id* yang sesuai. Jika tetap tidak ditemukan, kode *respons* diubah menjadi 404 dan pesan menjadi *'Data Not Found'*.

```
class CafeService
    public function getCafe($id, $ket = "")
    {
        $status = false;
        $code = 200;
        $result = null;
        try {
            $message = "Get data Cafe";
            $status = true;
            $cacheName = "cafe_{$id}_{$ket}";

            switch ($ket) {
                case 'user_id':
                    $result = Cache::tags(['get_cafe_data'])->remember($cacheName, now()->addMinute(150),
                    function () use($id){
                        return  Models\Cafe::where('user_id', $id)->first();
                    });
                    if (!$result) {
                        $code = 404;
                        $message = 'Data Not Found';
                        $result = null;
                    }
                    break;
```

Gambar 11. Cache Data Informasi Cafe

```
class CafeService
    public function getListCafe()
    {
        $status = false;
        $code = 200;
        $result = null;
        try {
            $result = Cache::remember("list_cafe", now()->addMinute(150), function (){
                return  Models\Cafe::get();
            });
            $message = 'Get List Data Cafe';
            $status = true;
        } catch (\Throwable $e) {
            $code = $e->getCode();
            $message = $e->getMessage();
            $result = [
                'get_file' => $e->getFile(),
                'get_line' => $e->getLine()
            ];
        }

        return [
            'code' => $code,
            'status' => $status,
            'message' => $message,
            'result' => $result
        ];
    }
```

Gambar 12. Cache Daftar Semua Cafe

Pada gambar 12 merupakan kode *function getListCafe* yang bertujuan untuk mengambil daftar semua *cafe*. Fungsi ini pertama-tama menginisialisasi variabel *$status, $code*, dan *$result* untuk menangani status operasi masing-masing. Dalam blok *try*, fungsi mencoba untuk mengambil data dari cache dengan nama "*list_cafe*" menggunakan *Cache::remember()* dengan masa berlaku 150 menit. Jika data tidak tersedia di *cache*, fungsi akan mengambil semua data cafe dari *database* menggunakan *Eloquent ORM* dari laravel dengan *Models\Cafe::get()* dan menyimpannya dalam *cache*.

3) *Penerapan Caching Data Order:* Pada gambar 13 merupakan kode *function getDataByID* yang bertujuan untuk mengambil data tertentu berdasarkan *ID* dan *parameter* opsional $*ket*. Fungsi ini menginisialisasi variabel *$status, $code,* dan *$result* untuk menangani status operasi, dan hasil operasi. Dalam blok *try*, pesan diatur menjadi "*Get data Order*" dan status operasi diatur menjadi *true*. Dua nama cache dihasilkan, yaitu *$cacheOrderName* dan *$cacheCafeName*, berdasarkan kombinasi *$id* dan *$ket*. Pada *$ket* bernilai *'cafe_id'*, fungsi akan mencoba mengambil data dari cache dengan nama *$cacheCafeName* menggunakan *Cache::tags(['get_list_order'])->remember()*, dengan masa berlaku cache 150 menit.

```
class OrderService
    public function getDataByID($id, $ket = "")
    {
        $status = false;
        $code = 200;
        $result = null;
        try {
            $message = "Get data Order";
            $status = true;
            $cacheOrderName = "order_{$id}_{$ket}";
            $cacheCafeName = "cafe_management_{$id}";
            switch ($ket) {
                case 'cafe_id':
                    $cafeManagement =  Cache::tags(['get_list_order'])->remember($cacheCafeName, now()
                    ->addMinute(150), function () use($id){
                        return Models\CafeManagement::select('id', 'user_id', 'cafe_id')->where
                        (['user_id' => $id])->first();
                    });
```

Gambar 13. *Cache Data Order*

4) *Penerapan Caching Data Product:* Gambar 14 merupakan kode *function getDataByID* yang bertujuan untuk mengambil data berdasarkan *ID* dan *parameter* opsional *$ket*. Fungsi ini menginisialisasi variabel *$status, $code*, dan *$result* untuk menangani status operasi dan hasil operasi. Dalam blok *try*, pesan diatur menjadi "*Get data Product*" dan status operasi diatur menjadi *true*. Dua nama *cache* dihasilkan, yaitu *$cacheProductName* dan *$cacheCafeName*, berdasarkan kombinasi *$id* dan *$ket*. Pada kasus *$ket* bernilai *'cafe_id'*, fungsi akan mencoba mengambil data dari *cache* dengan nama *$cacheCafeName* menggunakan *Cache::tags(['get_list_product'])->remember()*, dengan masa berlaku *cache* 150 menit.

```
    public function getDataByID($id, $ket = "")
    {
        $status = false;
        $code = 200;
        $result = null;
        try {
            $message = "Get data Product";
            $status = true;
            $cacheProductName = "order_{$id}_{$ket}";
            $cacheCafeName = "cafe_management_product_{$id}";
            switch ($ket) {
                case 'cafe_id':
                    $cafeManagement = Cache::tags(['get_list_product'])->remember($cacheCafeName, now()
                    ->addMinute(150), function () use($id){
                        return  Models\CafeManagement::select('id', 'user_id', 'cafe_id')->where
                        (['user_id' => $id, 'status' => true])->first();
                    });
```

Gambar 14. *Cache Data Product*

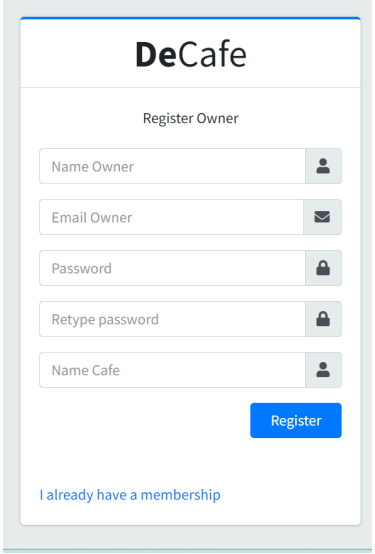
III. Hasil dan Pembahasan

A. *Hasil User Interface*

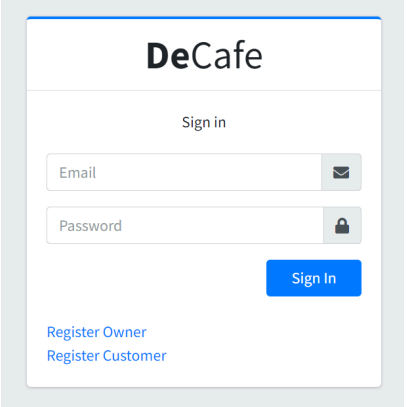
1) *Tampilan Halaman Register:* Gambar 15 merupakan halaman registrasi untuk owner cafe yang dimana pada halaman ini, owner diminta untuk mengisi form inputan yang berupa Name Owner, Email Owner, Password, Retype password, dan Name Cafe. Setelah data dimasukkan maka owner dapat menekan tombol Register berwarna biru untuk mengirimkan data registrasi. Selain itu, terdapat tautan bertuliskan I already have a membership yang berfungsi untuk berpindah ke halaman login jika sudah memiliki akun.

2) *Tampilan Halaman Login:* Gambar 16 merupakan halaman login yang dimana pengguna diminta untuk memasukkan alamat *email* dan kata sandi pada form inputan. Setelah mengisi informasi yang diperlukan, pengguna dapat menekan tombol *Sign In* berwarna biru untuk masuk ke dalam aplikasi. Selain itu, terdapat dua

tautan di bagian bawah halaman yaitu *Register Owner* untuk mendaftarkan pemilik *cafe* baru dan Register *Customer* untuk mendaftarkan pelanggan baru.



Gambar 15. Tampilan Halaman *Register*

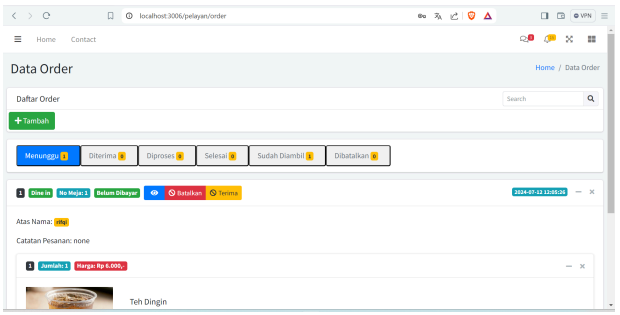


Gambar 16. Tampilan Halaman *Login*

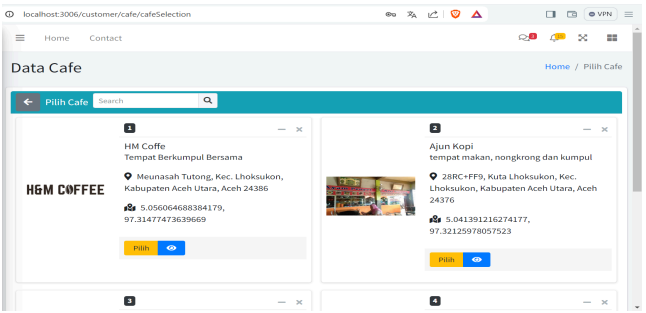
3) *Tampilan Halaman Order:* Gambar 17 merupakan halaman order yang dimana pada bagian atas halaman terdapat beberapa tombol status pesanan, seperti menunggu, diterima, diproses, selesai, sudah diambil, dan dibatalkan. terdapat tombol tambah untuk menambah pesanan baru.

4) *Tampilan Halaman Pilih Cafe:* Pada gambar 18 merupakan halaman pilih cafe yang dimana pada halaman ini menampilkan daftar cafe yang telah terdaftar dan pengguna dapat menekan tombol pilih untuk masuk ke dalam cafe dan melakukan pemesanan menu yang pada cafe tersebut.

5) *Hasil Data Cache Pada Redis:* Pada gambar 19 merupakan hasil dari cache data yang sudah tersimpan ke dalam penyimpanan redis, data yang disimpan adalah data dari code yang telah diterapkan cache.



Gambar 17. Tampilan Halaman *Order*

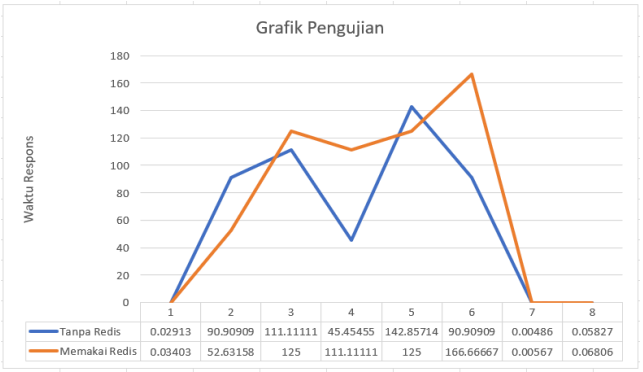


Gambar 18. Tampilan Halaman Pilih Cafe

Gambar 19. Hasil Data Cache Pada Redis

B. *Hasil Pengujian Caching Menggunakan Redis*

Pada 20 gambar merupakan grafik pengujian ini membandingkan waktu *respons* antara sistem tanpa *Redis* (garis biru) dan sistem yang menggunakan *Redis* (garis oranye) untuk 8 titik pengujian. Analisis menunjukkan bahwa sistem dengan *Redis* umumnya memberikan waktu *respons* yang lebih stabil dan konsisten, meskipun tidak selalu lebih cepat pada setiap titik. Perbedaan kinerja terlihat jelas pada beberapa titik kritis: di titik kedua, sistem dengan Redis menunjukkan peningkatan kinerja yang signifikan (52.63 ms vs 90.90 ms), sementara pada titik kelima, sistem tanpa Redis mengalami lonjakan waktu respons yang drastis (142.85 ms) dibandingkan dengan sistem Redis yang tetap stabil.



Gambar 20. Grafik Pengujian Caching Menggunakan Redis

Meskipun sistem dengan Redis mencapai waktu respons tertingginya pada titik keenam (166.67 ms), secara keseluruhan kinerjanya lebih dapat diprediksi. Sistem tanpa Redis menunjukkan fluktuasi yang lebih besar, yang bisa menjadi masalah dalam situasi beban tinggi. Berdasarkan grafik ini, penggunaan Redis sebagai mekanisme caching tampaknya memberikan manfaat dalam hal stabilitas dan konsistensi kinerja sistem, meskipun tidak selalu menghasilkan waktu respons yang lebih cepat pada setiap titik pengujian.

IV. KESIMPULAN

Implementasi *Redis* sebagai mekanisme *caching* di *backend* aplikasi terbukti efektif dalam meningkatkan performa secara signifikan. Dengan menyimpan data yang sering diakses dalam memori, *Redis* berhasil mengurangi beban pada *database* utama, menghasilkan waktu respons aplikasi yang lebih cepat. Hasil pengujian menunjukkan peningkatan kecepatan akses yang konsisten pada sebagian besar *endpoint API*. Meskipun pengujian tanpa Redis menampilkan waktu *respons* yang fluktuatif dengan nilai puncak 142,86 ms, implementasi *Redis* menghasilkan performa yang lebih stabil. Walaupun nilai puncak dengan *Redis* sedikit lebih tinggi pada 166,67 ms, konsistensi performa yang ditawarkan memberikan keuntungan signifikan dalam hal reliabilitas dan efisiensi aplikasi secara keseluruhan. Peningkatan ini menunjukkan bahwa penggunaan Redis sebagai solusi caching adalah langkah yang tepat untuk mengoptimalkan kinerja aplikasi *order cafe*.

REFERENSI

[1] E. Aryani, Y. Zanaria, and A. Kurniawan, "ANALISIS PERKEMBANGAN COFFEE SHOP SEBAGAI SALAH SATU PERANAN UMKM DI KOTA METRO (Study Kasus Pada Coffee Shop Janji Jiwa Dan Coffee Et Bien)."

[2] G. Sukarno and L. Nirawati, "KONTRIBUSI HUMAN CAPITAL DAN CUSTOMER CAPITAL DALAM MENGGAPAI KINERJA CAFÉ DAN RESTO DI SURABAYA."

[3] A. Jonathan, R. Saputra, C. Witta, P. Santoso, and T. Mustiadi, "Perancangan Aplikasi Android Untuk Pemesanan Makanan Di Cafe Fourture," 2023.

[4] A. Syaefulloh and F. Yusrizal, "Implementasi Dan Analisa Performa DataBase Cache Redis," 2019. [Online].Available:https://www.researchgate.net/publication/338195534

[5] M. I. Zulfa, A. Fadli, and A. W. Wardhana, "Application caching strategy based on in-memory using Redis server to accelerate relational data access," *Jurnal Teknologi dan Sistem Komputer*, vol. 8, no. 2, pp. 157–163, Apr. 2020, doi: 10.14710/jtsiskom.8.2.2020.157-163.

[6] A. P. Meriani and D. R. Asih, " Pengujian Distributed Cached Database Dengan Menggunakan Redis Pada Aplikasi MaBaUS," 2023. [Online]. Available: https://www.researchgate.net/publication/369007644

[7] R. J. Iskandar, "Hal 50 PENERAPAN REDIS DALAM SISTEM E-OFFICE DESA SUTERA."

[8] S. Mufti Prasetiyo, M. Ivan Prayogi Nugroho, R. Lima Putri, and O. Fauzi, "BULLET : Jurnal Multidisiplin Ilmu Pembahasan Mengenai Front-End Web Developer dalam Ruang Lingkup Web Development", [Online].Available:https://journal.mediapublikasi.id/index.php/bullet

[9] I. Ketut Aditya Herdinata Putra, D. Pramana, N. Luh Putri Srinadi, and S. STIKOM Bali Jl Raya Puputan, "Sistem Manajemen Arsip Menggunakan Framework Laravel dan Vue.Js (Studi Kasus : BPKAD Provinsi Bali)".

[10] L. Wulandari, I. Ramadhan, P. Magister Teknologi dan Rekayasa, and U. Gunadarma, "Infrastruktur High-Available Learning Management System Universitas Menggunakan Least-Connected Load Balancer," 2022.