IMPLEMENTASI REST API PADA CRINO.ID

Disusun Oleh:

N a m a : Febby Kurniawan
NIM : 18523273

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

IMPLEMENTASI REST PADA CRINO.ID

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Febby Kurniawan

NIM : 18523273

Yogyakarta, 15 Oktober 2022

Pembimbing,

17/10/2022

Irving Vitra Paputungan, S.T., M.SC., PH.D.

HALAMAN PENGESAHAN DOSEN PENGUJI

IMPLEMENTASI REST PADA CRINO.ID

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 03 November 2022

Tim Penguji

Irving Vitra Paputungan, S.T., M.Sc., Ph.D. _____

Anggota 1

Nur Wijayaning Rahayu, S.Kom., M.Cs. _____

Anggota 2

Affan Mahtarami, S.Kom., M.T. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Febby Kurniawan

NIM : 18523273

Tugas akhir dengan judul:

IMPLEMENTASI REST PADA CRINO.ID

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 17 Oktober 2022

Febby Kurniawan

HALAMAN PERSEMBAHAN

Laporan Tugas Akhir ini penulis persembahkan untuk orang-orang yang sudah meluangkan waktu untuk membantu dalam memberikan doa, mendukung, menyemangati, memberikan arahan, masukan dan motivasi, serta mendengarkan keluh kesah selama proses penulisan sehingga penulis mampu dan berhasil menyelesaikan Laporan Tugas Akhir ini.

HALAMAN MOTO

"Love the life you live. Live the life you love."

Bob Marley

"Dilarang Melarang."

Febby Kurniawan.

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Bismillahirrahmanirrahim, tidak lupa dengan mengucap Alhamdulillah hirobbil 'alamin, segala puji dan syukur kepada Allah SWT. berkat rahmat, karunia dan hidayah-Nya, sehingga penulis mampu menyelesaikan Laporan Tugas Akhir Magang ini dengan semaksimal mungkin. Adapun tujuan dibuatnya Laporan Tugas Akhir Magang ini adalah sebagai bukti pelaksanaan magang serta untuk memenuhi kewajiban penulis dalam memperoleh gelar sarjana pada Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

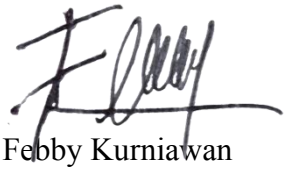
Kemudian, pada kesempatan ini tidak lupa juga penulis mengucapkan banyak terimakasih kepada semua pihak yang telah membantu meluangkan waktu dalam memberikan doa, mendukung, menyemangati, memberikan arahan, masukan dan motivasi, serta mendengarkan keluh kesah mulai dari proses pelaksanaan magang di PT Bhinneka Mentari Dimensi hingga penyusunan Laporan Tugas Akhir ini, antara lain:

1. Allah SWT., kedua orang tua dan keluarga.
2. Bapak Irving Vitra Paputungan, S.T., M.SC., PH.D. selaku Dosen Pembimbing Akademik sekaligus Dosen Pembimbing.
3. Segenap Bapak dan Ibu Dosen Prodi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.
4. Bapak Hendrik Tio selaku CEO PT Bhinneka Mentari Dimensi.
5. Mas Moch. Sudharmono, selaku mentor dari mitra PT Bhinneka Mentari Dimensi.
6. Semua rekan magang dan tim Brittlestar.
7. Rekan satu kontrakan Omah Biner, Kost Oren dan semua rekan sejawat.

Akhir kata, penulis berharap semoga bermanfaat bagi Terima kasih banyak, semoga Allah SWT. senantiasa memberikan kenikmatan sehat dan kesejahteraan bagi kita semua.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 17 Oktober 2022



Febby Kurniawan

SARI

Crino.id adalah salah satu *platform multi-channel* yang membantu untuk mengatur produk, stok, harga, promosi, dan menyediakan laporan penjualan dari berbagai *marketplace* di Indonesia. Sistem crino.id dikembangkan oleh PT Bhinneka Mentari Dimensi untuk mempermudah penggunanya dalam memanajemen transaksi penjualan produk dari beberapa *marketplace* popular di Indonesia yang terintegrasi dalam satu *platform*. Namun, sistem crino.id yang ada saat itu masih merupakan sistem yang berdiri sendiri dan terpisah dengan toko online bhinneka.com. Selain itu, sistem crino.id yang ada saat itu masih merupakan sistem dengan arsitektur monolith. Di sisi lain, semakin berkembanganya perusahaan, maka proses bisnis akan terus berubah sesuai dengan kebutuhan perusahaan. Hal tersebut tentu akan berpengaruh dalam pengembangan sistem yang sudah dimiliki perusahaan. Demi memenuhi salah satu dari sekian banyak kebutuhan strategi bisnisnya, PT Bhinneka Mentari Dimensi memiliki keinginan untuk mengembangkan sistem crino.id. Pengembangan sistem dilakukan dengan harapan crino.id dapat disatukan ke dalam *dashboard seller center* bhinneka.com yang sudah mengadopsi teknologi microservice. Oleh karena itu, dibutuhkan pengembangan untuk mengubah sistem crino.id yang masih monolith dengan cara membuat atau memecah beberapa proses bisnis pada sistem yang ada menjadi web *service* menggunakan arsitektur REST. Laporan ini membahas bagaimana mengimplementasikan REST pada crino.id sesuai dengan kebutuhan pengembangan. Akan tetapi, implementasi pengembangan hanya sebatas pada penambahan modul *Customer Management*. Pengembangan tersebut telah berhasil diimplementasikan menggunakan salah satu kerangka kerja dari bahasa pemrograman Python yaitu Django dan memanfaatkan *toolkit* yang bernama Django REST *Framework*. Kemudian hasil dari pengujiannya, setiap *endpoint* dipastikan sudah berhasil digunakan dan fungsi dari setiap *method* telah berjalan dengan baik serta sesuai dengan rancangan.

Kata kunci: Web Service, REST, Django, Django REST Framework.

GLOSARIUM

Bandwitdh	besar ukuran kapasitas dalam transfer data dalam satuan waktu tertentu.
Clone	proses salin hal serupa.
Endpoint	ujung titik akhir dari tautan.
Environment	lingkup alat atau kebutuhan teknologi yang digunakan.
Syntax	aturan penulisan kode program.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Crino.id adalah salah satu *platform multi-channel* yang membantu untuk mengatur produk, stok, harga, promosi, dan menyediakan laporan penjualan dari berbagai *marketplace* di Indonesia. Melalui crino.id, pengguna yang berjualan di beberapa *marketplace* atau lebih dari satu *marketplace* dapat mengelola transaksi penjualan produknya cukup hanya dalam satu *dashboard* sistem. Sistem crino.id dikembangkan oleh PT Bhinneka Mentari Dimensi untuk mempermudah penggunanya dalam memanajemen transaksi penjualan produk dari beberapa *marketplace* popular di Indonesia yang terintegrasi dalam satu *platform*(Bhinneka, 2022). Di sisi lain, karena semakin berkembanganya perusahaan, maka proses bisnis akan terus berubah sesuai dengan kebutuhan. Hal itu tentu membuat PT Bhinneka Mentari Dimensi memiliki keinginan untuk terus mengembangkan sistem yang ada. Salah satunya pada sistem crino.id yang masih berdiri sendiri dan terpisah dengan toko *online* bhinneka.com atau dapat dikatakan sistem dengan arsitektur monolith.

Sistem arsitektur monolith adalah sistem yang dalam pengembangannya semua fungsionalitas dan komponennya menjadi satu kesatuan(Christian & Bisma, 2021). Kemudian sistem dengan arsitektur monolith memiliki kekurangan yaitu keterbatasan ukuran dan kompleksitas dalam pengembangannya, karena semua proses fungsionalitas serta komponennya berada dalam satu kesatuan(Ngurah & Kusuma, 2022). Berbeda halnya dengan sistem yang mengadopsi arsitektur microservice. Sistem dengan arsitektur microservice merupakan sistem yang dalam pengembangannya fungsionalitas dan komponennya dibuat atau dipecah secara spesifik dalam bentuk web *service* yang lebih kecil(Uminingsih & Handayani, 2020). Web *service* adalah sistem perangkat lunak yang mendukung interaksi antar mesin melalui jaringan untuk menyediakan layanan yang berisi informasi atau data. Informasi atau data tersebut biasanya dikirim melalui HTTP (*Hypertext Transfer Protocol*) sehingga bisa di akses oleh sistem lain yang berbeda *platform*, sistem operasi, maupun bahasa pemrograman (Somya & Nathanael, 2019). Dalam pengembangannya web *service* terbagi menjadi 2 jenis dan dapat diimplementasikan dengan menggunakan REST (*Respresentational State Transfer*) dan SOAP (*Simple Object Access Protocol*) (Afrizal & Fitriyani, 2021). Walaupun kedua

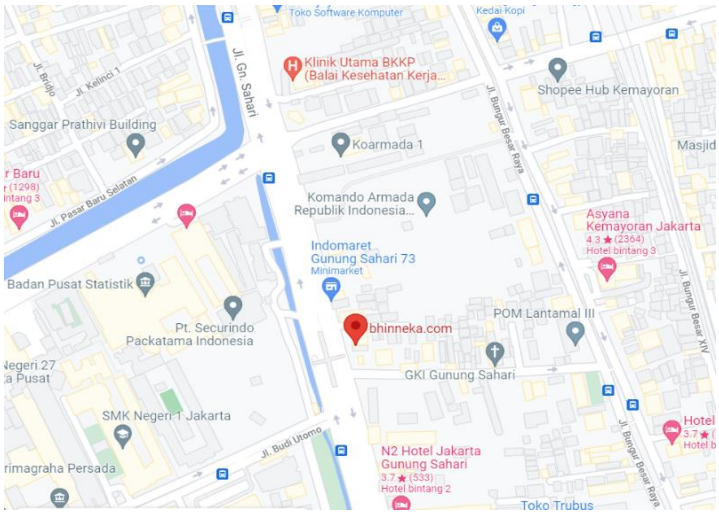
jenisnya memiliki kesamaaan dalam hal dapat melakukan komunikasi pertukaran data antar mesin, namun tetap memiliki beberapa hal yang berbeda (Kusumaningrum et al., 2019).

Pada penulisan laporan tugas akhir ini, pengembangan web *service* crino.id dibuat dengan menggunakan arsitektur REST. REST merupakan salah satu gaya arsitektur yang menerapkan konsep perpindahan antar state, dan setiap state mewakili satu *resource* yang ada pada server (Sy & Indo Intan, 2019). Setiap *resource* direpresentasikan dalam format JSON atau XML, diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau *global* ID(Pamuji, 2020), dan dengan memanfaatkan beberapa metode HTTP antara lain: *GET*, *POST*, *PUT*, serta *DELETE*(Somya & Nathanael, 2019). Berdasarkan beberapa penelitian terdahulu, menurut Putra & Putera, (2019), web *service* menggunakan arsitektur REST dipilih karena dirasa memiliki keunggulan peforma kecepatan melakukan *request* dan *response* atau melakukan proses pertukaran data dibandingkan arsitektur lainnya. Selain itu, REST juga memiliki format *response* yang lebih fleksibel seperti tipe data JSON, XML, atau format teks lainnya, tidak seperti SOAP yang hanya dengan XML-nya saja (Kusumaningrum et al., 2019). Adapun, menurut Adhiwibowo et al., (2019), web *service* REST lebih unggul dari SOAP karena dalam prosesnya REST tidak memerlukan *bandwidth* yang banyak sehingga aksesnya lebih cepat. Kemudian menurut Ranga & Soni, (2019), SOAP paling cocok untuk proyek kerja dengan jangka waktu yang panjang, lebih berfokus pada keamanan serta kenyamanan seperti sistem perbankan, finansial, dan layanan telekomunikasi. Namun REST adalah tetap yang terbaik dan cocok untuk proyek yang lebih fokus pada kesederhanaan kinerja dan waktu pengembangan yang singkat.

Oleh karena itu, apabila melihat dari permasalahan dan beberapa penelitian yang telah diuraikan, terdapat gagasan untuk mengubah sistem crino.id yang masih monolith dengan cara membuat atau memecah beberapa proses bisnis pada sistem yang ada menjadi web *service*. Dengan harapan agar menjadikan crino.id lebih optimal dalam sisi pengembangan dan perbaikan kedepannya serta memudahkan pekerjaan kolaborasi tim pengembang antar sistem yang berbeda dalam hal komunikasi dan pertukaran data. Adapun batasan yang dilakukan dalam penulisan laporan tugas akhir ini adalah pengembangan web *service* sistem crino.id hanya pada penambahan modul *Customer Management*. Kemudian pengembangan dibuat dengan bahasa pemrograman kategori tingkat tinggi Python (Lorentius et al., 2022), dan menggunakan kerja web dari Python yaitu Django (Afrizal & Fitriyani, 2021), serta *tookit* khusus bernama Django REST *Framework* yang dapat digunakan untuk membangun web *service* berbasis REST dengan Django (Arévalo et al., 2020).

1.1.1 Gambaran Umum PT Bhinneka Mentari Dimensi

PT Bhinneka Mentari Dimensi merupakan perusahaan yang bergerak di industri *e-commerce* yang memiliki toko *online* Bhinneka.com. Bhinneka adalah pelopor *e-commerce* di Indonesia dan mengkhususkan diri pada produk komputer dan IT, barang komunikasi, dan elektronik konsumen (3C). Dimulai pada tahun 1993 yang dipimpin oleh seorang Presiden Direktur yaitu Bapak Hendrik Tio, kemudian pada tahun 1999 meluncurkan toko *online* pertamanya Bhinneka.com. Bhinneka mempunyai visi "Menjadi pemimpin *platform* B2B *e-commerce* yang berfokus kepada pelanggan" dan mempunyai berbagai misi diantaranya: "Mengembangkan ekosistem bisnis yang inovatif agar pembeli dan penjual dapat bertransaksi", "Menyediakan barang/jasa yang lengkap dan terpercaya dengan didukung oleh proses yang berkesinambungan", "Mencapai keunggulan operasional untuk menjaga profitabilitas", serta "Membangun lingkungan kerja yang luar biasa". Dalam perjalanan bisnisnya, Bhinneka memiliki rekam jejak yang terbukti dalam melayani *Business to Customer* (B2C), *Business to Business* (B2B), dan *Business to Government* (B2G). PT Bhinneka Mentari Dimensi memiliki kantor utama di Jl. Gn. Sahari No.73C, RT.9/RW.7, Gn. Sahari Sel., Kec. Kemayoran, Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta 10610. Lokasi lebih jelas dapat dilihat pada Gambar 1.1.

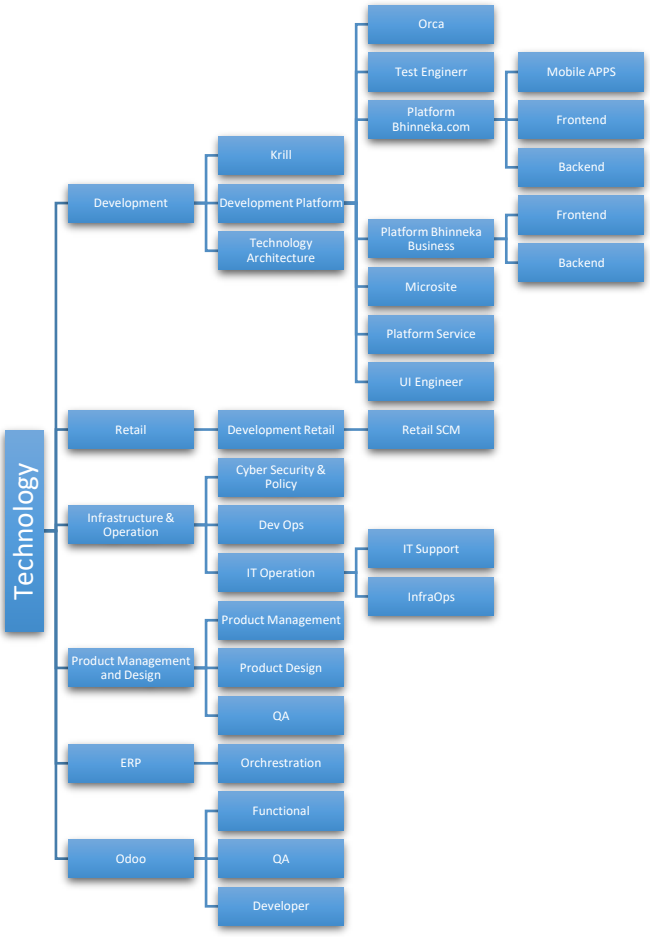


Gambar 1.1 Titik Lokasi PT Bhinneka Mentari Dimensi dari Google *Maps*.

Kemudian seperti pada umumnya perusahaan, PT Bhinneka Mentari Dimensi memiliki struktur organisasi yang terbagi menjadi 11 divisi besar, namun masing-masing memiliki struktur turunan yang banyak dan bervariasi, oleh karena itu pada penulisan laporan tugas akhir ini hanya dapat dituliskan bagian besar dari divisi yang ada diantaranya yaitu:

a. *Top Management*
b. *Strategic Business Unit & Digital Printing Solutions*
c. *Finance & Accounting*
d. *Technology*
e. *Strategic Management Office*
f. *Platform*
g. *Corporate Business*
h. *Operational*
i. *New Strategic Business Unit*
j. *Corporate Affair*
k. *Merchandising*

Akan tetapi, dikarenakan penulisan laporan tugas akhir ini ada kaitannya dengan peran atau posisi turunan dari divisi *Technology*, maka secara khusus berikut merupakan gambaran struktur dari divisi *Technology* yang dapat dilihat pada Gambar 1.2.



Gambar 1.2 Struktur Organisasi dari Divisi *Technology* PT Bhinneka Mentari Dimensi

1.1.2 Sejarah Institusi

Sejak tahun 1993, PT Bhinneka Mentari Dimensi mengawali bisnis nya sebagai pemegang merek dan distributor untuk produk printer format lebar dan PC. Perjalanan mereka tidak sepenuhnya mulus, sebab mereka juga harus melewati situasi genting karena dilanda badai krisis moneter di tahun 1998. Namun karena situasi tersebut, Nicholas Tio dan Hendrik Tio justru melihat peluang yang terinspirasi dari tren belanja *online* dan perkembangan internet yang luar biasa di Amerika Serikat. Hingga pada tahun 1999, mereka meluncurkan terobosan situs Bhinneka.Com sebagai pionir *e-commerce* pertama di Indonesia. Bhinneka.Com sebagai Toko IT *online* pertama di Indonesia. Walaupun hanya dengan sisa jumlah personel yang hanya tinggal 24 orang dari 129.

Selanjutnya pada tahun 2001, Bhinneka mulai membuka beberapa jaringan toko *offline*. Dibuka dan tersebar di beberapa kota. Fungsinya tidak hanya melakukan penjualan, tapi sekaligus juga menjadi *drop off point* untuk servis produk. Bhinneka menjadi perusahaan yang membuka gerai/toko fisik *omni channel* pertama di Indonesia. Hingga mereka memiliki banyak toko fisik yang hampir merata di Indonesia.

Kemudian, terjadi catatan penting lagi di tahun 2011. Bhinneka kembali menorehkan catatan prestasi nya. Mereka berhasil meluncurkan Bhinneka Bisnis, *platform* B2B pertama di Indonesia. Berjalan sampai tahun 2015, Bhinneka juga membantu Pemerintah membangun *e-Katalog*, menjadi toko *online* pertama sebagai penyedia *e-Katalog* LKPP. Melalui *e-Katalog Business to Government* (B2G), yang didukung oleh tim Bhinneka yang berada di 33 provinsi di Indonesia.

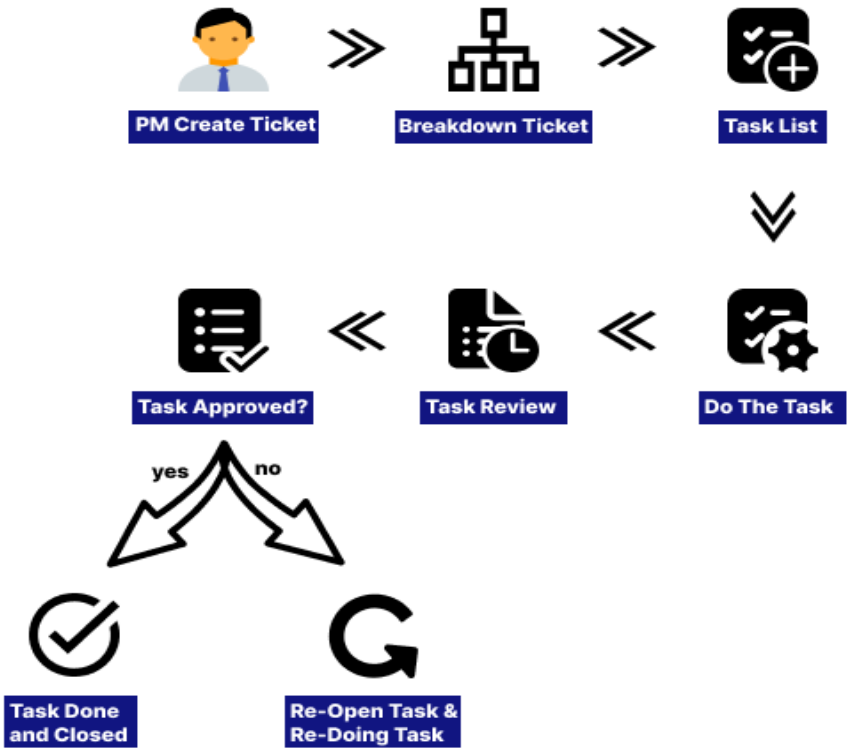
Hingga rentang tahun 2018-2020, Bhinneka masih memiliki beberapa rangkaian sejarah penting. Pada tahun 2018, Bhinneka menghadirkan *Business Solutions*, membangun solusi bisnis bagi klien korporasi dengan sistem terintegrasi IoT, dan infrastruktur solusi lainnya. Selanjutnya di tahun 2019 mereka melakukan perluasan kategori produk *non-IT*, menghadirkan MRO digital produk. Sampai tahun 2020, Bhinneka terus memperkokoh layanan B2B nya, bertransformasi menjadi *business Super-Ecosystem*, yakni sebuah ekosistem yang menghadirkan layanan sektor jasa pertama dan terlengkap bagi kebutuhan bisnis di Indonesia, dari UMKM sampai *Enterprises*.

Berikut ini merupakan gambaran tonggak pencapaian dari perjalanan selama 28 tahun PT Bhinneka Mentari Dimensi dari awal berdiri dapat dilihat pada Gambar 1.3.

Gambar 1.3 Tonggak Pencapaian atau Catatan Perjalanan PT Bhinneka Mentari Dimensi.

1.1.3 Alur Kerja

Pada saat melaksanakan kegiatan magang sebagai Software Developer Intern, berikut merupakan gambaran alur kerja yang dapat dilihat pada Gambar 1.4.



Gambar 1.4 Alur Kerja Sebagai *Software Developer Intern*.

1.1.4 Lingkungan Kerja

Selama menjalani proses penjaluran magang sebagai *software developer* yang berada di dalam tim proyek brittlestar, peran tersebut berada di dalam divisi *Technology* dengan sub divisi *Development*. Brittlestar merupakan sebuah tim yang dibentuk untuk mengembangkan sebuah proyek bernama Crino.id. Kemudian, di dalam tim proyek brittlestar peran *software developer* di pecah lagi dan dibagi menjadi 3 peran yaitu: *Lead*, *Back End*, dan *Front End*. Pembagian peran tersebut dilakukan oleh supervisor atau mentor tim proyek brittlestar yang bernama mas Moch. Sudharmono, beliau merupakan pimpinan dari divisi *Platform Service* yang ditugaskan untuk menaungi dan bertanggung jawab atas pekerjaan tim proyek brittlestar. Namun, pada penulisan laporan tugas akhir ini rata-rata peran dan kontribusi yang dilakukan dalam pengembangan proyek adalah sebagai *Back End*.

1.1.5 Penempatan Kerja

Pada program magang ini, awalnya ditempatkan di tim yang bernama brittlestar. Tim brittlestar sendiri merupakan tim yang beranggotakan 8 orang pemagang terdiri dari 5 peran, antara lain *Product Manager*, *Product Designer*, *Test Engineer*, *Quality Assurance*, dan *Software Developer*. Kemudian, dikarenakan magang dilakukan pada saat kondisi pandemi Covid-19, semua karyawan termasuk pemagang dipekerjakan secara *remote*. Kerja remot atau *remote working* adalah pekerjaan yang dilakukan tanpa perlu adanya interaksi langsung datang ke kantor. Selain itu, sebenarnya Bhinneka sendiri sudah cukup lama menerapkan pekerjaan jarak jauh atau *remote working* tersebut, kecuali peran pekerjaan yang berada pada divisi *Operational*.

1.1.6 Jadwal Harian

Bhinneka memiliki peraturan waktu kerja di hari senin-jumat dimulai dari pukul 09.00-12.00, kemudian pukul 12.00-13.00 istirahat, dan dilanjutkan bekerja kembali pukul 13.00-18.00, namun karena pekerjaan dilakukan secara *remote* atau jarak jauh, maka karyawan di Bhinneka juga bisa menyesuaikan aturan yang ada dengan prinsip 8 jam bekerja + 1 jam istirahat. Kemudian, sebagai seorang *software developer* memiliki jadwal rutin melakukan *daily stand-up meeting* pada pukul 10.45-11.00. *Daily stand-up meeting* ini bertujuan untuk menyampaikan progres pekerjaan antar teman tim sesuai peran masing-masing, seperti contoh apa yang sudah dikerjakan kemarin, hari ini, serta kendala atau hambatan apa yang didapatkan.

1.2 Ruang Lingkup Magang

Pelaksanaan magang di PT Bhinneka Mentari Dimentasi telah berlangsung dan dilaksanakan selama enam bulan, dalam kurun waktu yang dimulai dari 24 Agustus 2021 sampai dengan 28 Februari 2022. PT Bhinneka Mentari Dimensi merupakan perusahaan yang bergerak di industri *e-commerce* dan memiliki toko *online* Bhinneka.com. Selama menjalani proses magang dengan peran *software developer* di tim proyek brittlestar, adapun daftar aktivitas secara keseluruhan yang dilakukan sebagai berikut:

a. Mengikuti *on-boarding* dan *sharing session* dengan para pegawai senior.
b. Mengikuti beberapa pelatihan kemampuan untuk persiapan kebutuhan magang.
c. Mengikuti proses pengembangan dan pengerjaan *task* pada proyek yang diberikan.
d. Mengikuti agenda rapat untuk laporan progres harian dan mingguan dengan tim proyek.

1.3 Tujuan

Tujuan dari pelaksanaan magang ini adalah untuk mengembangkan sebuah *platform* yang dikembangkan oleh PT Bhinneka Mentari Dimensi bernama crino.id yang masih monolith dengan cara membuat atau memecah beberapa proses bisnis pada sistem yang ada menjadi web *service* menggunakan arsitektur REST.

1.4 Manfaat

Manfaat yang dihasilkan dari proses pengembangan sistem Crino.id dengan cara membuat atau memecah beberapa proses bisnis pada sistem yang ada menjadi web *service* menggunakan arsitektur REST adalah sebagai berikut:

a. Memudahkan komunikasi dan pertukaran data antar sistem yang berbeda.
b. Memudahkan pengembang dalam membuat fitur yang serupa.
c. Mempercepat dan menghemat waktu pengembangan perangkat lunak.
d. Memudahkan tim pengembang dalam mengembangkan modul pada sistem yang bersangkutan.

1.5 Sistematika Penulisan

Sistematika penulisan laporan ini bertujuan untuk memudahkan pembaca dalam memahami isi laporan secara menyerluruh. Berikut merupakan susunan sistematika penulisan yang ada dalam laporan akhir ini yaitu:

a. BAB I. PENDAHULUAN

Bab ini membahas tentang latar belakang pengembangan web *service Customer Management* pada Crino.id yang menggunakan arsitektur REST, ruang lingkup magang, tujuan dan manfaat tentang pengembangan perangkat lunak tersebut.

b. BAB II. KAJIAN PUSTAKA

Bab ini membahas tentang beberapa teori dan penelitian serupa yang terkait dengan impementasi REST API.

c. BAB III. PELAKSANAAN MAGANG

Bab ini membahas tentang pelaksanaan magang yang meliputi gambaran umum profil institusi, alur pelaksanaan magang, lingkukan kerja, penempatan dan jadwal harian, serta tahapan manajemen proyek dan tahapan pengembangan proyek yang telah dilakukan selama menjalani proses pelaksanaan magang.

d. BAB IV. REFLEKSI PELAKSANAAN MAGANG

Bab ini membahas tentang refleksi dari pelaksanaan magang pada PT Bhinneka Mentari Dimensi baik secara teknis atau non teknis yang meliputi hambatan, manfaat serta tantangan yang didapatkan.

e. BAB V. KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari pelaksanaan kegiatan magang yang telah dilakukan dan saran bagi peneliti yang hendak melakukan penelitian serupa.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

2.1 Web Service

Web *service* adalah sistem perangkat lunak yang digunakan untuk mendukung interaksi dan interoperabilitas antar mesin melalui jaringan. Web *service* biasanya dimanfaatkan sebagai fasilitas untuk menyediakan layanan yang berisi informasi atau data. Informasi atau data yang berada dalam web *service* umumnya disimpan dengan format JSON atau XML yang dikirim melalui HTTP (*Hypertext Transfer Protocol*) sehingga bisa di akses oleh sistem lain yang berbeda *platform*, sistem operasi, maupun bahasa pemrograman (Somya & Nathanael, 2019).

Kemudian menurut Afrizal & Fitriyani (2021) jenis-jenis web *service* terbagi menjadi 2 yaitu:

a. *Respresentational State Transfer* (REST)
b. *Simple Object Access Protocol* (SOAP)

2.2 REST

REST (*Representational State Transfer*) adalah salah satu gaya arsitektur yang menerapkan konsep perpindahan antar *state*, setiap state mewakili satu *resource* yang ada pada server(Sy & Indo Intan, 2019). Setiap *resource* diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau *global* ID(Pamuji, 2020). Biasanya *resource* tersebut direpresentasikan dalam format JSON atau XML. Dalam penerapannya REST dibangun menggunakan protokol HTTP (*Hypertext Transfer Protocol*) dan memanfaatkan beberapa metode yang ada di dalam HTTP antara lain: *GET*, *POST*, *PUT*, dan *DELETE* (Somya & Nathanael, 2019). REST bekerja dengan cara bernavigasi melalui tautan HTTP yang tersedia untuk melakukan aktivitas, sama seperti mengganti *state* dari halaman web, REST seakan-akan melakukan perpindahan *state* satu sama lain(Pamuji, 2020). Sementara REST API atau RESTful API merupakan bentuk implementasi dari web *service* yang menggunakan prinsip dasar REST (Putra & Putera, 2019).

2.3 Python

Python adalah salah satu bahasa pemrograman yang termasuk kedalam kategori tingkat tinggi (*high-level programing language*) atau dapat diartikan bahasa pemrograman yang level kode nya mendekati bahasa manusia, bersifat terbuka *multi-platform* yang dapat digunakan

dalam berbagai jenis tujuan dan berbagai macam sistem operasi (Windows, Mac, Linux) (Lorentius et al., 2022).

2.4 Django

Django adalah salah satu kerangka kerja web bersifat terbuka yang dirancang khusus untuk pembuatan web menggunakan bahasa pemrograman Python. Seperti pada umumnya, *framework* dapat membantu mempermudah *developer* dalam menyelesaikan pekerjaannya. Kerangka kerja web Django memiliki pola arsitektur MVT(Model-View-Template). Pola tersebut sebenarnya mirip dengan pola arsitektur MVC(Model-*View*-*Controller*), namun pada MVC perbedaannya yang mengontrol antara model dan *view* adalah *controller*, sedangkan pada MVT yang mengontrol antara model dan *view* adalah *template*. *Template* pada Django berisi *file* HTML yang didalamnya terdapat tambahan *syntax* DTL(Django-*Template*-*Language*) (Afrizal & Fitriyani, 2021).

2.5 Django REST Framework

Django REST *Framework* adalah sebuah *tookit* khusus milik kerangka kerja web Django yang dapat digunakan untuk membangun layanan web berbasis REST. Django REST *Framework* menyediakan banyak fitur yang memudahkan pekerjaan *developer* khususnya dalam pembuatan REST API. Seperti membantu menambahkan serializer objek yang mendefinisikan atau memetakan data yang kompleks, opsi autentifikasi tambahan, dan tampilan browser API untuk melihat *response* API yang sudah dibuat(Arévalo et al., 2020).

2.6 JSON

JSON (*Javascript Object Notation*) adalah salah satu format data atau teks yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan atau dibuat oleh computer (Saputra & Fathoni Aji, 2018). JSON dibuat berdasarkan bagian dari bahasa pemrograman Javascript(Firdaus et al., 2019). JSON merupakan format teks yang tidak bergantung dengan bahasa pemrograman apapun dan familiar dengan beberapa bahasa pemrograman yang umum digunakan oleh programmer seperti C, C++, C#, Java, Javascript, Perl, Python, dll. Oleh karena itu, hal tersebut dapat disimpulkan bahwa JSON dapat dikatakan sebagai bahasa format data yang ideal untuk melakukan pertukaran data(Ridha et al., n.d.).

2.7 Postman

Postman adalah sebuah *platform* yang dapat melakukan kolaborasi untuk pengembangan API atau dapat dikatakan sebagai GUI API *Caller* (Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti et al., 2017). Beberapa hal yang dapat dilakukan Postman antara lain: dapat bertindak sebagai *client* yang mengakses REST secara langsung, dapat digunakan untuk pengujian yang terotomatisasi, dapat melakukan simulasi *endpoint* secara langsung, dapat mengetahui peforma dan waktu *response* dari API, serta menyediakan fitur berbagi dalam *workspace* untuk kebutuhan membangun dan menggunakan API secara *real-time* (Galindra Wardhana et al., 2020).

2.8 Tinjauan Pustaka

Dalam penulisan laporan tugas akhir ini, terdapat beberapa sumber referensi penelitian terdahulu. Sumber referensi dengan kasus serupa dipilih untuk memenuhi kebutuhan sebagai landasan penulisan dan pengembangan sistem yang dapat dilihat pada Tabel 2.1.

Tabel 2.1 Perbandingan Arsitektur Web *Service* pada Penelitian Serupa.

No.	Keterangan	Hasil Penelitian
1.	**Judul**: *API Features Individualizing of Web Services*: REST *and* SOAP **Penulis**: (Ranga & Soni, 2019) **Arsitektur web service**: SOAP	Berdasarkan hasil penelitian tersebut didapat kesimpulan: - Waktu *response* REST lebih kecil dibanding SOAP yaitu berkisar sekitar 4ms sampai 7ms. - Sedangkan dengan peningkatan jumlah API, SOAP membutuhkan perkiraan penggunaan memori 1MB hingga 2MB lebih banyak daripada REST, hal tersebut dikarenakan SOAP memiliki muatan yang lebih berat daripada REST. - Kemudian dengan interaksi yang sedikit, REST dapat dengan mudah diperbarui tanpa perlu mengetahui dokumentasi API yang membuat *client* lebih mudah. - Selain itu SOAP lebih dapat diandalkan karena memiliki logika *successful/retry* yang digunakan ketika terjadi kesalahan, sedangkan REST tidak memiliki logika seperti itu. - SOAP memiliki fitur WS-*Security*, sementara REST terbatas ke https tetapi dapat ditambahkan metode tambahan demi meningkatkan keamanannya. REST lebih mudah dipamahami dan dikembangkan, ringan, dan memiliki format apa saja tidak seperti SOAP yang terbatas pada XML. - SOAP paling cocok untuk proyek kerja dengan jangka waktu yang panjang, lebih berfokus pada keamanan serta kenyamanan seperti sistem perbankan, finansial, dan layanan telekomunikasi. Namun REST adalah tetap yang terbaik dan cocok untuk proyek yang lebih fokus pada kesederhanaan kinerja dan waktu pengembangan yang singkat.

2.	**Judul**: Analisis Perbandingan Metode Soap Dan Rest Yang Digunakan Pada *Framework* Flask Untuk Membangun Web *Service*. **Penulis**: (Putra & Putera, 2019) **Arsitektur web service**: REST dan SOAP	Berdasarkan hasil penelitian tersebut didapat kesimpulan: - Web *service* yang dibangun mengunakan kerangka kerja web Flask dengan menggunakan SOAP dan REST API dari grafik tersebut, REST memiliki performance yang lebih bagus dibandingkan dengan SOAP untuk pengatesan *request* dan *respon* untuk web *service*. - Dengan menggunakan socket JSON dan XML lebih baik menggunakan socket XML yang mana untuk mendapatkan beberapa metode tidak diketahui oleh *client* yang mengakses web *browser*.
3.	**Judul**: Rest And Soap *Comparison On* Web *Service Technology For* Android *Based* Data *Services*. **Penulis**: (Kusumaningrum et al., 2019) **Arsitektur web service**: REST dan SOAP	Berdasarkan hasil penelitian tersebut didapat kesimpulan: - Pengujian 30 sampel data dengan jumlah data bervariasi, diuji pada *device* smartphone yang berbeda, jaringan internet berbeda, dan ukuran *file* berbeda (REST API lebih besar ukuran *file*-nya dan SOAP lebih kecil ukuran *file*-nya), dihasilkan rata-rata waktu sinkronisasi data REST API lebih cepat dibandingkan SOAP. - Hasil rata-rata keseluruhan sinkronisasi data dengan format ukuran data yang sama pada REST API menghasilkan waktu 3.4 detik dan SOAP menghasilkan waktu 3.9 detik, sedangkan dengan format ukuran data yang berbeda pada REST API menghasilkan waktu 4.7 detik dan SOAP menghasilkan waktu 5.3 detik.
4.	**Judul**: Pengembangan PDPT Berbasis *Framework* Dengan Teknologi Web *Service* SOAP dan REST di Universitas Semarang. **Penulis**: (Adhiwibowo et al., 2019) **Arsitektur web service**: REST dan SOAP	Berdasarkan hasil penelitian tersebut didapat kesimpulan: - Web *service* REST lebih unggul dari SOAP karena dalam prosesnya REST tidak memerlukan *bandwidth* yang banyak sehingga aksesnya lebih cepat. - Web *service* REST API dapat menampilkan data maksimal 16,1092 *record* untuk 1 *field* data, sedangkan untuk 15 *field* didapatkan 25,577 *record*. - Sedangkan SOAP hanya dapat menampilkan data maksimal 97,010 *record* untuk 1 *field* data, sedangkan 15 *field* didapatkan data sebanyak 15,116 *record*.

Berdasarkan sumber referensi penelitian terdahulu yang ada pada tabel 1, bisa disimpulkan bahwa penggunaan web *service* API dapat berguna banyak untuk kebutuhan pengembangan sistem perangkat lunak. Baik penggunaan web *service* API yang menggunakan gaya arsitektur REST maupun SOAP (Putra & Putera, 2019). Pada penerapannya, web *service* API dengan arsitektur REST lebih mudah diimplementasi dibandingkan arsitektur SOAP, karena REST tidak memiliki spesifikasi khusus dalam penulisan, berbeda dengan SOAP yang memiliki aturan penulisan yang ketat mengikuti spesifikasi XML (Kusumaningrum et al., 2019). Selain itu, REST juga memiliki format *response* yang lebih fleksibel karena memiliki pilihan seperti: JSON, XML, atau format teks lainnya, sehingga memudahkan penerima *response*, tidak seperti SOAP dengan XML-nya saja, sehingga sedikit sulit membaca dan memahaminya karena tidak memiliki pilihan format *response* lainnya(Putra & Putera, 2019). Hal tersebut dapat berpengaruh juga dalam penggunaan *bandwidth*, karena jika dalam jumlah

request yang banyak, format *response* dengan bahasa markup seperti XML relatif lebih boros *bandwidth* (Kusumaningrum et al., 2019). Kemudian, berdasarkan dua penelitian sebelumnya pada tabel 1, terdapat hasil penelitian yang menguji *request* dan *response* web *service* API menggunakan arsitektur REST yang dibangun pada kerangka kerja web FLASK memiliki performa yang lebih bagus dibandingkan SOAP (Putra & Putera, 2019). Selain itu terdapat penelitian serupa yang menguji sinkronisasi 30 sampel data dengan jumlah yang bervariasi pada beberapa *device* smartphone berbeda, dihasilkan rata-rata waktu sinkronisasi data web *service* menggunakan arsitektur REST lebih cepat dibandingkan SOAP(Kusumaningrum et al., 2019). Kedua arsitektur tersebut sebenarnya sama-sama membantu interaksi antar sistem perangkat lunak agar saling terhubung atau terintegrasi satu sama lain, sehingga dapat melakukan transaksi pertukaran data melalui web *service*. Namun, berdasarkan perbandingan dari penelitian terdahulu, sepertinya REST lebih unggul dan cocok untuk diterapkan dalam pengembangan ini.

BAB III
PELAKSANAAN MAGANG

3.1 Manajemen Proyek

Dalam melakukan pengembangan dan implementasi REST API *Customer Management* pada Crino.id, terdapat beberapa tahapan manajemen proyek. Tujuan dari diterapkannya tahapan manajemen proyek ini adalah supaya pengembangan proyek yang dilakukan menjadi terarah dan terorganisir. Kemudian berikut ini tiap-tiap tahapan manajemen proyek akan dijelaskan sebagai berikut.

3.1.1 Pendefinisian Proyek

Crino.id merupakan *platform multi-channel* yang memudahkan untuk mengelola produk, stok, harga, dan promosi di semua toko *online* dari satu *dashboard* yang dikembangkan oleh PT Bhinneka Mentari Dimensi. Namun sistem tersebut merupakan sistem yang terpisah dengan Bhinneka.com dan memiliki tautan tersendiri yaitu Crino.Id. Oleh karena itu, dibutuhkan pengembangan untuk mengubah sistem crino.id yang masih monolith dan terpisah dengan cara membuat atau memecah beberapa proses bisnis pada sistem tersebut menjadi sistem dengan teknologi web *service* menggunakan arsitektur REST. Pengembangan tersebut dibutuhkan karena adanya gagasan untuk menyatukan sistem crino.id dengan sistem bhinneka.com yang sudah mengadopsi arsitektur microservice.

3.1.2 Inisialisasi Proyek

Pada tahap inisialisasi terdapat beberapa aktivitas yang dilakukan sebelum *kick-off project* dimulai. Tahapan ini dilakukan untuk mengetahui mengenai kebutuhan pengembangan. Berikut ini merupakan beberapa aktivitas yang dilakukan pada tahap inisialisasi.

Meet With Mentor

Tahapan ini dilakukan pertemuan antara pemagang dengan beberapa supervisor atau mentor setiap pemagang, tujuannya untuk membahas gambaran spesifikasi teknologi apa saja yang kemungkinan akan dimanfaatkan pada pengembangan. Selain itu, pada pertemuan ini juga membahas mengenai beberapa gambaran deskripsi pekerjaan dari setiap peran di dalam tim untuk mengembangkan sistem crino.id seperti yang dapat dilihat pada Tabel 3.1.

Tabel 3.1 Deskripsi Pekerjaan dari Setiap Peran.

Peran	Gambaran Deskripsi Pekerjaan
Product Manager	Pada proyek ini, *Product Manager* sebagai seorang pemimpin proyek yang bertanggung jawab atas pengelolaan produk secara keseluruhan, seperti berkoordinasi dengan *internal* dan *external* tim, membuat tiket tugas yang berisi *user story*, serta mengelola keseluruhan sumber daya proyek untuk mencapai tujuan proyek.
Product Designer	Seorang *Product Designer* dalam proyek ini berkoordinasi dengan *Product Manager* dan bertanggung jawab atas desain tampilan produk yang ingin diimplementasikan pada proyek.
Software Developer	Pada proyek ini, totalnya terdapat 3 orang *Software Developer*, kemudian peran tersebut dipecah dan dibagi lagi menjadi lebih spesifik oleh supervisor atau mentor antara lain adalah: - *Front End*, sebagai seorang yang bertanggung jawab atas eksekusi dan implementasi desain tampilan yang telah dibuat *Product Manager* dalam bentuk kode program sesuai dengan tiket yang tugas nya telah di spesifikasikan oleh *Lead Developer*. - *Back End*, sebagai seorang yang bertanggung jawab atas eksekusi dan implementasi alur logika dari sisi server yang berkaitan dengan data, *service* API, dll. sesuai dengan tiket yang tugas nya telah di spesifikasikan oleh *Lead Developer*. - *Lead Developer*, sebagai seorang pemimpin *Software Developer* yang bertanggung jawab atas tugas pekerjaan *Front End* dan *Back End*, serta aktif melakukan koordinasi dan komunikasi dengan peran lainnya yang ada di tim.
Quality Assurance	Sebagai seorang yang bertanggung jawab atas pengujian produk yang dikembangkan secara keseluruhan.
Test Engineer	Sebagai seorang yang melakukan pengujian otomatis dengan menggunakan kode program.

Meet Crinoid Project Existing

Tahapan ini dilakukan pertemuan antara *software developer intern* dengan pengembang perangkat lunak senior satu-satunya yang mengembangkan proyek crinoid. Pada tahapan ini dilakukan tanya jawab terkait spesifikasi teknologi apa saja yang digunakan pada proyek crino.id. Kemudian, dari hasil pertemuan ini dapat diketahui spesifikasi teknologi pengembangan ini sudah ditentukan oleh perusahaan, seperti yang dapat dilihat pada Tabel 3.2.

Tabel 3.2 Spesifikasi Teknologi Pada Sistem Crino.id.

Aspek	Spesifikasi
Bahasa Pemrograman	Python
Framework	Django
Basis	Web *Service*
Database	PostgreSQL
Tools	*Project Management*: Jira *Prototype Design*: Figma *Version Control System*: GitHub

3.1.3 Perencanaan Proyek

Setelah dilakukan tahapan inisialisasi proyek, tahapan selanjutnya yang dilakukan adalah perencanaan proyek. Tahap ini aktivitas yang dilakukan mulai dari menentukan metode pengembangan yang akan digunakan dalam proses pengembangan. Dalam proses pengembangan sistem crino.id milik bhinneka ini direncanakan akan mengadopsi pendekatan dari metodologi pengembangan agile menggunakan metode scrum. Pada scrum terdapat beberapa peran, antara lain: *product owner* sebagai pemilik aplikasi, scrum *master* sebagai yang bertanggung jawab memimpin setiap tahap scrum dan memastikan setiap sprint berjalan lancar, dan *developer team* sebagai orang yang membangun sistem. Kemudian di dalam scrum terdapat scrum events antara lain: Sprint Planning, Daily Scrum, Sprint Review, dan Sprint retrospective. Selain itu, scrum juga memiliki artefak scrum terdiri product backlog, sprint backlog, dan increment. Product backlog ini biasanya dimanfaatkan untuk pembuatan daftar fitur yang akan dikembangkan serta menentukan prioritasnya. Sementara sprint backlog merupakan daftar fitur hasil seleksi dari product backlog yang akan diselesaikan pada sprint yang sedang berlangsung. Sedangkan increment merupakan hasil rilis produk secara bertahap dari tiap sprint yang telah dilakukan.

Oleh karena itu, pada tahap ini didapatkan prioritas fitur yang akan dikerjakan, berapa lama durasi pelaksanaan pengembangannya, dan target-target pengerjaan dari tiap jangka waktu yang telah ditentukan dari product dan sprint backlog. Selain itu, pada tahap ini juga telah ditentukan rotasi perubahan peran setiap 2x satuan waktu sprint yang ada di dalam metode scrum. Oleh karena itu, pada tahapan ini didapatkan hasil gambaran *timeline* perencanaan awal seperti yang dapat dilihat pada Tabel 3.3.

Tabel 3.3 Gamabaran *Timeline* Perencanaan

Nama Aktivitas	**Tujuan**	**Tanggal**	**Peran**
Kick-Off Project	Start developing project.	10/09/2021	
Sprint 1	Setup & learn code flow of the project.	13/09/2021 – 24/09/2021	*Back End*
Sprint 2	100% *Customer Management*.	27/09/2021 – 22/11/2021	*Back End*
Sprint 3	100% *Product Management*.	25/11/2021 – 19/12/2021	*Lead Developer*
Sprint 4	100% *Order Management*.	22/11/2021 – 17/12/2022	*Lead Developer*
Sprint 5	100% *User Management.*	20/12/2022 – 14/01/2022	*Front End*
Sprint 6	Preparing For Project Presentation & Closing.	14/01/2022 – 28/01/2022	*Front End*

3.1.4 Pelaksanaan Proyek

Tahap selanjutnya setelah melewati tahap perencanaan proyek, kemudian lanjut ke tahap pelaksanaan proyek. Tahapan ini mulai dilakukan pengembangan sistem crino.id dengan menggunakan acuan metode scrum seperti yang telah direncanakan pada tahap sebelumnya. Berikut merupakan beberapa aktivitas dari scrum events yang akan diuraikan sebagai berikut:

a. Sprint Planning

Sprint planning merupakan bagian atau tahapan awal dari scrum events. Sprint planning dilakukan untuk membahas item atau fitur mana yang akan dikerjakan serta berapa lama durasi sprint tertentu. Pada tahapan ini product owner memberitahu dan mendiskusikan urutan prioritas dari product backlog yang paling penting sesuai product goal kepada seluruh anggota, termasuk scrum master dan developer team. Tujuannya product backlog awal yang telah dibuat dapat dikembangkan menjadi sprint backlog untuk developer team dalam setiap sprint

b. Daily Scrum/Daily Stand Up Meeting

Daily Scrum atau dapat dikatakan juga Daily Stand Up Meeting(DSM) merupakan scrum events yang dilakukan rutin setiap hari secara singkat dengan durasi selama 15-30 menit untuk mempertemukan developer team dengan scrum master. Tahapan ini bertujuan untuk memantau perkembangan proyek selama proses sprint berjalan. Dalam scrum events ini penulis yang termasuk kedalam developer team menyampaikan progres yang sudah dikerjakan kemarin, progres yang akan dikerjakan hari ini, dan kendala apa yang dihadapi.

c. Sprint Review

Sprint review merupakan bagian atau tahapan dari scrum events. Sprint review penting diadakan karena bertujuan untuk melihat hasil pekerjaan dari sprint yang telah dilakukan. Dalam acara ini scrum master akan mengintruksikan developer team untuk mendemonstrasikan hasil pekerjaannya selama satu sprint. Berdasarkan tinjauan hasil pekerjaan tersebut, scrum master dan stakeholder akan menentukan apa saja yang akan dilakukan selanjutnya serta item product backlog dapat disesuaikan kembali.

d. Sprint retrospective

Sprint restrospective merupakan bagian atau tahapan dari scrum events yang dilakukan untuk mengevaluasi tim. Acara ini diadakan dengan tujuan dan harapan dapat meningkatkan kualitas dan efektivitas yang akan dilakukan pada sprint selanjutnya. Evalusi yang diberikan pada acara ini biasanya dengan cara saling memberi masukan antar scrum team.

Berdasarkan scrum events yang telah diuraikan sebelumnya, pada tahapan ini semua anggota tim bekerja sesuai dengan peran dan deskripsi pekerjaannya masing-masing. Kemudian dalam tahap ini apabila dilihat dari timeline perencanaan yang sudah dijelaskan sebelumnnya, setelah sprint 2 *Customer Management* dari yang awalnya menjadi *Back End* seharusnya mendapatkan rotasi peran sebagai *Lead Developer*. Namun pada saat rotasi ke *Lead Developer*, teman *Software Developer* yang seharusnya menjadi *Back End* memiliki kendala dan membuat anggota *Software Developer* berkurang 1 orang, dari yang awalnya 3 orang hanya tersisa menjadi 2 orang. Hal itu membuat fokus pekerjaan yang seharusnya menjadi *Lead Developer* terbagi dan terpecah karena harus merangkap peran sebagai *Back End* kembali. Selain itu pada saat sprint 3 *Product Management* baru berjalan 3 minggu, terdapat kesalahpahaman dari pihak atasan bhinneka terkait perancangan pengembangan, walaupun perancangan sudah dipresentasikan oleh *Product Manager* jauh diawal sebelum *kick-off* proyek dimulai. Hal tersebut mengakibatkan pengembangan jadi terhambat dan membuat timeline tidak sesuai dengan perencanaan awal sehingga proyek diberhentikan dan ditutup dalam waktu dekat. Oleh karena itu, aktivitas kegiatan selama magang pada pelaksanaan proyek juga menjadi tidak sesuai ekspetasi, berikut merupakan gambaran timeline dengan aktivitas yang dilakukan beserta peran yang didapat pada pelaksanaannya selama menjalani magang seperti yang dapat dilihat pada Tabel 3.4.

Tabel 3.4 Gambaran Timeline Pada Pelaksanaan

Nama Aktivitas	**Kegiatan Yang Dilakukan**	**Tanggal**	**Peran**
Kick-Off Project Brittlestar	Start developing project.	10/09/2021	
Sprint 1	Setup & learn code flow of the project.	13/09/2021 – 24/09/2021	*Back End*
Sprint 2	100% *Customer Management*.	27/09/2021 – 29/10/2021	*Back End*
Sprint 3	Membantu *Product Manager* menganalisis kebutuhan pengembangan *Product Management*, mendetailkan tugas dalam tiket *Product Management* untuk *Back End* & *Front End*, serta membantu merangkap peran sebagai *Back End* kembali.	03/11/2021 – 26/11/2021	*Lead Developer* dan *Back End*
Menunggu keputusan pengembangan proyek	Sesuai dengan arahan *Product Manager* dan mentor sembari menunggu hasil keputusan pengembangan proyek, tetap lanjut ke Sprint 5 yaitu mengerjakan *Product* dan *Order Management*.	29/11/2021 – 17/12/2022	*Lead Developer* dan *Back End*
Mendapat hasil keputusan pengembangan proyek	Berdasarkan hasil kordinasi antara *Product Manager* dan mentor dengan pihak atasan bhinneka, pengembangan proyek crino.id sementara diberhentikan dan ditutup karena dirasa memiliki kekurangan pada perancangan proyek dan apabila dilihat serta ditinjau dari waktu durasi magang yang tersisa dinilai proyek ini sulit untuk dikerjakan oleh peserta magang.	20/12/2022 – 22/12/2022	*Lead Developer* dan *Back End*
Perubahan pergantian Proyek	Sampai pada tahap aktivitas ini, sebenernya proyek yang dikerjakan masih sama yaitu proyek crino.id. Namun, dari yang awalnya mengubah sistem monolith crino.id menjadi web *service* API, akhirnya hanya melakukan pengembangan memperbaiki bug pada sistem crino.id.	23/12/2022 – 14/01/2022	*Software Developer*

Tetapi, terlepas dari timeline pelaksanaan yang telah diuraikan sebelumnya, sesuai dengan tujuan dan batasan penulisan laporan tugas akhir ini, pelaksanaan pengembangan crino.id yang dilakukan dengan menambahkan web *service* API pada modul *Customer Management* menggunakan arsitektur REST, memiliki 3 tahapan utama yang dilakukan antara lain:

a. Perancangan REST API

Tahap perancangan merupakan salah satu tahap yang penting untuk dijadikan dasar dalam pengembangan yang dilakukan. Tahap perancangan dapat dijadikan dasar pedoman sebelum melakukan tahapan implementasi. Pada tahapan ini yang dilakukan antara lain membuat rancangan tambahan tabel basis data untuk kebutuhan mengimplentasikan fitur *Customer Management* pada sistem crino.id. Kemudian dilakukan perancangan diagram aktivitas dari pengembangan sistem. Selain itu, perancangan *endpoint* juga dibuat untuk mempermudah pengembang dalam tahap implementasi.

b. Implementasi REST API

Tahap implementasi ini merupakan tahapan kedua setelah tahap perancangan. Pada tahap ini dijelaskan bagaimana mengimplementasi rancangan yang telah dibuat pada tahap sebelumnya ke dalam salah satu kerangka kerja web bahasa pemrograman Python yaitu Django, dan menggunakan *toolkit* yang bernama Django REST *Framework* untuk mengembangkan web *service* API dengan arsitektur REST didalam Django.

c. Pengujian REST API

Tahap pengujian merupakan tahap ketiga atau tahap akhir. Pada tahap ini web *service* API dilihat dan diuji untuk memastikan bahwa fungsi dari setiap method telah berjalan dengan baik dan sesuai dengan rancangan. Pada tahap pengujian ini dilakukan dengan menggunakan metode *white-box* dengan cara memanfaatkan perangkat lunak pengujian API yang bernama postman.

3.1.5 Pemantauan dan Pengendalian Proyek

Pada tahap pemantauan dan pengendalian ini dilakukan untuk melihat perkembangan hasil pekerjaan dari waktu yang telah ditetapkan. Dikarenakan pekerjaan dilakukan secara remot, maka tahapan ini pun dilakukan dengan memanfaatkan beberapa *platform*. Berikut ini merupakan beberapa *platform* yang digunakan untuk pemantauan dan pengendalian proyek pengembangan dan implementasi REST API *Customer Management* pada crino.id, antara lain:

a. Slack

Slack merupakan sebuah *platform* komunikasi multi-guna yang menunjang produktivitas kolaborasi pekerjaan secara bekelompok. Pada pengembangan proyek crino.id ini, slack digunakan untuk melakukan komunikasi cepat melalui pertukaran pesan, *file*, dll. antar tim pengembang baik yang berstatus pegawai maupun pemagang.

b. Jira

Jira merupakan sebuah perangkat lunak yang digunakan untuk memanejemen atau organisasi kerja proyek yang melibatkan banyak orang. Pada pengembangan proyek crino.id ini, jira digunakan untuk memanajemen proyek selama proses pengembangan berjalan. Dengan menggunakan jira, semua anggota tim proyek dapat melihat perkembangan pekerjaan proyek melalui status tiket atau tugas yang terbaru.

c. GitHub

Github merupakan layanan untuk menyimpan repositori berbabis web yang digunakan untuk memanjemen kode. Semua kode program yang dikerjakan pengembang dapat dilihat oleh pengembang lain apakah sudah benar-benar dikerjakan atau belum.

3.1.6 Penutupan Proyek

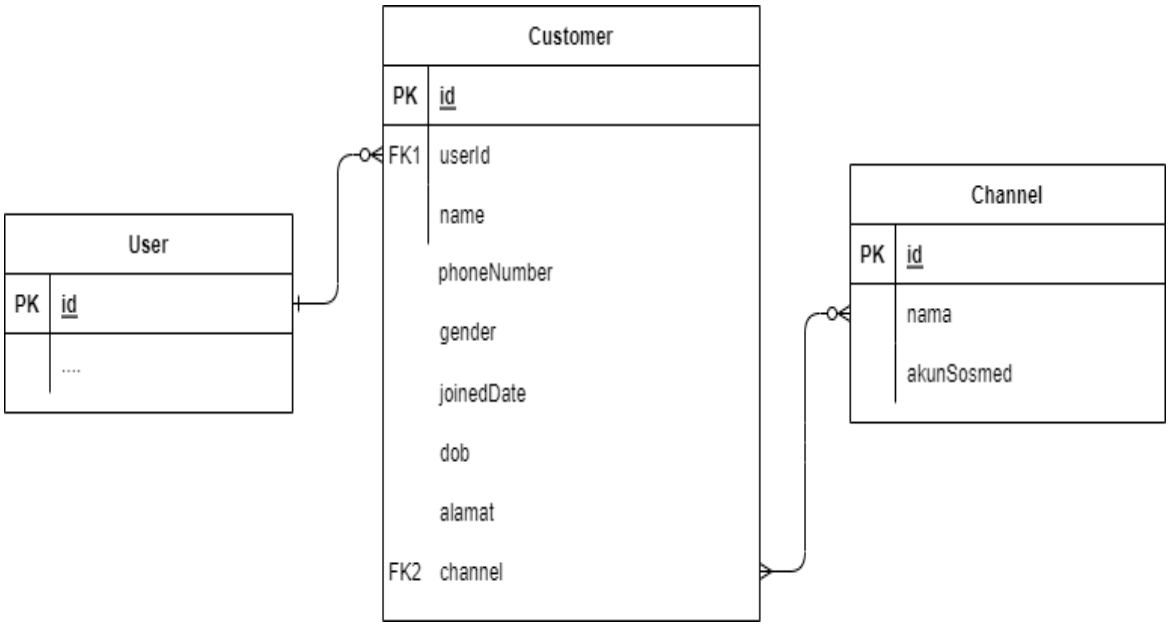
Tahap penutupan proyek ini sejatinya diakukan apabila proyek telah selesai dikerjakan. Namun, jika dilihat dari perencanaan secara keselurahan dan tidak hanya berfokus pada implementasi REST API *Customer Management* saja, sebenarnya proyek masih ada pada tahap pengembangan. Oleh karena itu, pengembangan proyek crino.id ditutup dalam waktu dekat. Masih banyak modul atau fitur crino.id selain *Customer Management* yang belum dikembangkan menjadi teknologi web *service*.

3.2 Pengembangan REST API Customer Management

Pengembangan dan implementasi REST API *Customer Management* pada crino.id, merupakan salah satu bentuk untuk memenuhi kebutuhan menyatukan sistem crino.id dan Bhinneka.com yang terpisah dapat melakukan komunikasi pertukaran data. Pada bab ini, akan dibahas secara jelas hasil dari setiap tahapan yang telah diuraikan pada bab sebelumnya.

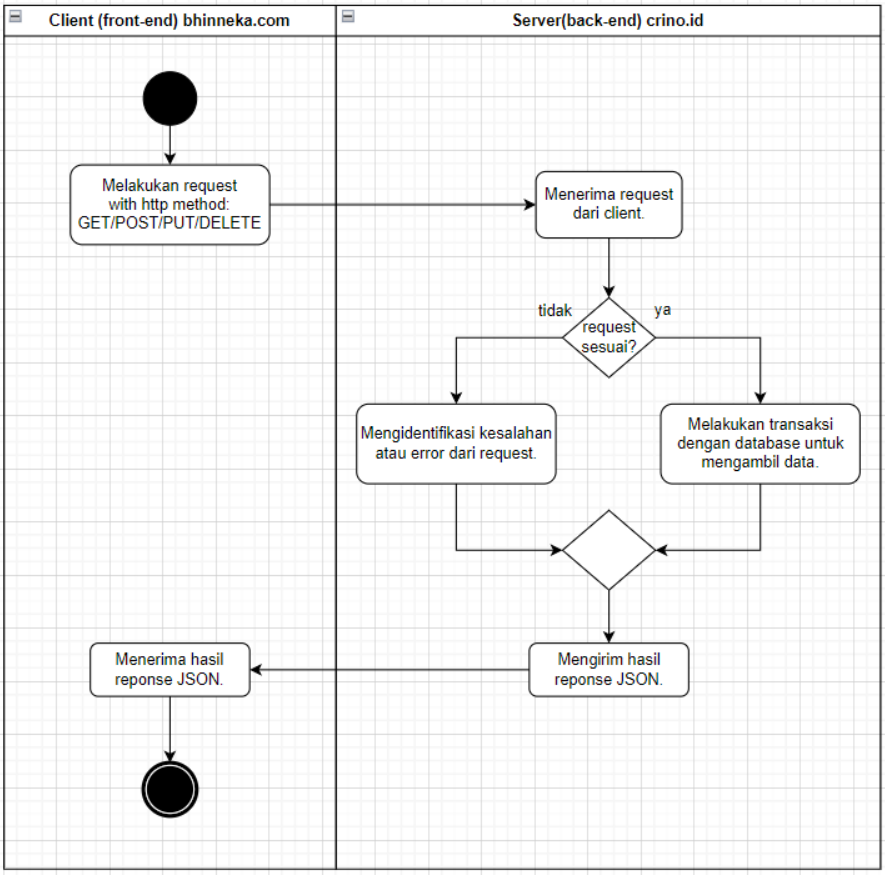
3.2.1 Perancangan REST API

Sebagaimana yang telah dijelaskan sebelumya, pada tahap ini dilakukan pembuatan rancangan tabel data tambahan, perancangan diagram aktivitas, serta *endpoint* yang akan digunakan pada tahap implementasi. Perancangan pertama, yaitu dimulai dengan melakukan analisis dari tabel crino.id yang ada untuk menambahkan tabel baru "*customer*" dan "*channel*". Setelah dilakukan analisis, maka didapatkan hasil rancangan seperti pada Gambar 3.1. Tabel data baru tersebut ditambahkan dengan memanfaatkan konsep basis data relasional, yang artinya memiliki hubungan relasi antar tabel. Oleh karena itu, dapat dilihat tabel data "*customer*" memiliki relasi dengan tabel "*user*" dan "*channel*". Tabel basis data tersebut berfungsi untuk memenuhi kebutuhan pembuatan REST API *Customer Management* pada crino.id, yang menyimpan dan mencatat data pelanggan dari setiap pengguna yang membeli produk nya dari *platform* atau *channel* lain selain toko *online*.



Gambar 3.1 Rancangan Tambahan Tabel Basis Data

Perancangan kedua yaitu membuat rancangan diagram aktivitas dari sistem yang akan dikembangkan. Diagram aktivitas ini merupakan gambaran aktivitas perilaku sistem crino.id pada *Customer Management* yang dapat dilihat seperti pada Gambar 3.2.



Gambar 3.2 Diagram Aktivitas Sistem

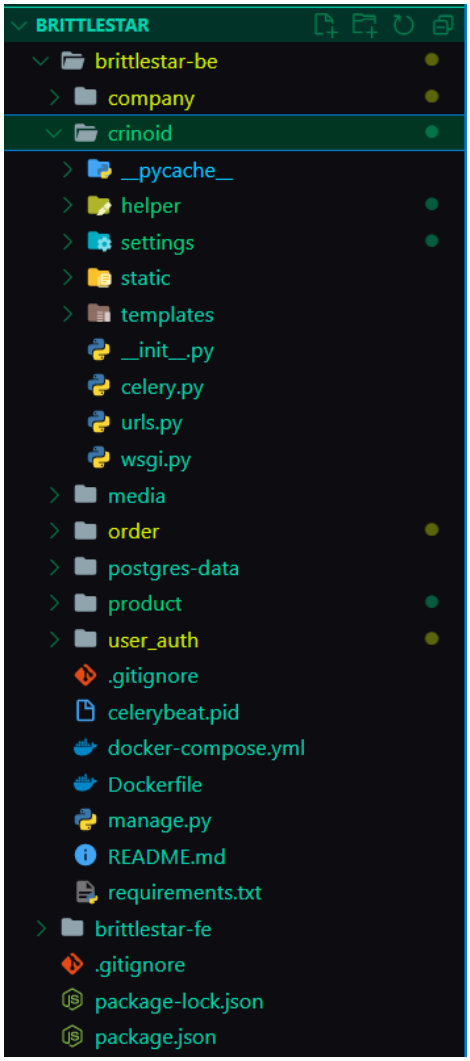
Selain itu, perancangan kedua yaitu terdapat rancangan *endpoint* yang akan diimplementasikan ke dalam web *service* API berbasis REST. Berdasarkan hasil diskusi peran *software developer* dengan Mentor, rancangan *endpoint* dibuat dengan ketentuan penamaan memiliki informasi seperti method, versi API, dan juga kata bentuk jamak agar mudah dimengerti oleh *client* seperti pada Tabel 3.5.

Tabel 3.5 Rancangan *Endpoint*.

Method HTTP	**Endpoint**	**Keterangan**
GET	/api/v1/customers	Get all customer
GET	/api/v1/customers/[id]	Get detail customer
POST	/api/v1/customers	Create customer
PUT	/api/v1/customers/[id]	Update customer
DELETE	/api/v1/customers/[id]	Delete customer
GET	/api/v1/customers?search=value	Search data customers by name
GET	/api/v1/customers?searchPhone=value	Search data customers by phone number

3.2.2 Implementasi REST API

Tahap selanjutnya merupakan tahap implementasi REST API. Impelementasi REST API modul *Customer Management* pada crino.id ini memiliki operasi dasar CRUD (*Create*, *Read*, *Update*, *Delete*). Pada tahap implementasi, semua kebutuhan konfigurasi *environment* proyek yang dibutuhkan telah disiapkan dan dibantu mentor, oleh karena itu pada tahapan ini hanya perlu menginstal dan *clone* proyek dari repositori yang bernama brittlestar, sehingga dapat dilihat struktur folder utama proyek di dalam *code editor* seperti yang dapat dilihat pada Gambar 3.3.



Gambar 3.3 Struktur Folder Proyek

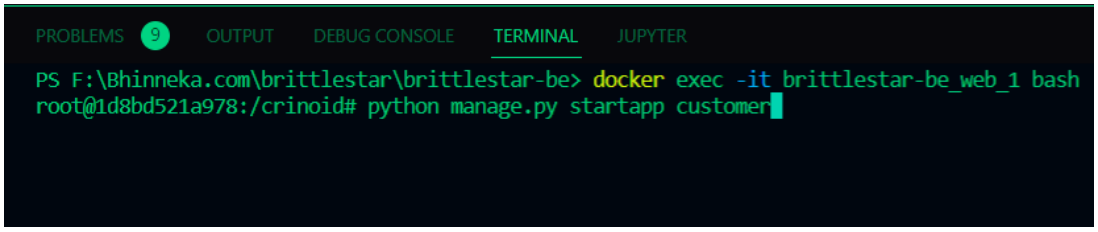
Jika dilihat dari Gambar 3.3 diatas, setelah berhasil melakukan *clone proyek* dari repositori yang disediakan kedalam laptop, maka perlu diketahui pemahaman terkait beberapa struktur folder dan *file* proyek yang telah menggunakan kerangka kerja django, berikut ini merupakan penjelasan dari beberapa folder dan *file* utama khususnya yang berada di folder brittlestar-be:

a. crinoid, merupakan folder basic *project* atau pengaturan dasar Django.
b. crinoid/settings, merupakan folder yang didalamnya berisi beberapa konfigurasi proyek.
c. urls.py, merupakan *file router* yang digunakan untuk menulis dan mendaftarkan *address* atau *endpoint* dari setiap app.
d. *company*, *media*, *order*, *product*, *user_auth*, merupakan foder dari beberapa app yang telah dikembangkan sebelumnya.

Kemudian dalam pengerjaannya, terdapat beberapa langkah yang dilakukan, dimulai dari membuat app *customer*, mendefinisikan model, membuat serializers, membuat *controller*, hingga membuat tautan *endpoint* nya. Berikut merupakan penjelasan dari beberapa langkah aktivitas yang telah disebutkan sebelumya.

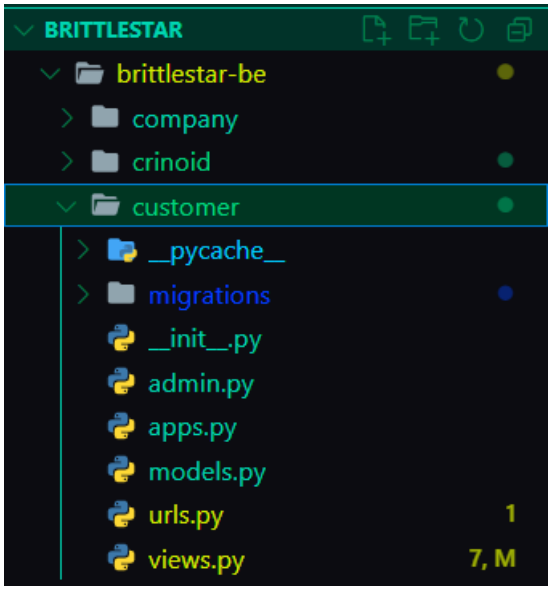
Membuat App Customer

Pada langkah ini, aktivitas yang dilakukan adalah membuat app baru untuk menampung atau menyimpan modul baru yang bernama *customer*. Aktivitas ini dimulai dari masuk kedalam *virtual environment* (venv) proyek terlebih dahulu, caranya dengan cara menuliskan perintah `docker exec -it brittlestar-be_web_1 bash` di terminal dan pastikan direktori terminal sudah di dalam brittlestar-be. Kemudian setelah berhasil masuk kedalam venv, tuliskan perintah dengan *syntax* `python manage.py startapp customer` untuk membuat app baru dengan nama *customer* seperti yang dapat dilihat pada Gambar 3.4.



Gambar 3.4 Perintah Untuk Membuat App *Customer*

Setelah proses instalasi app baru *customer* berhasil, maka folder baru dengan nama *customer* akan otomatis terbuat didalam folder proyek seperti yang dapat dilihat pada Gambar 3.5.



Gambar 3.5 Struktur Folder App *Customer*

Jika mengacu pada Gambar 3.5 diatas, dapat dilihat didalam folder app *customer* yang telah berhasil dibuat, terdapat beberapa *file* yang otomatis terbuat juga. Berikut merupakan penjelasan dari beberapa *file* yang akan digunakan untuk menuliskan beberapa blok kode yang terdapat didalam folder *customer*:

a. apps.py, merupakan *file* yang digunakan untuk mendefinisikan app *customer*.
b. models.py, merupakan *file* yang digunakan untuk mendefinisikan model tabel dan atribut data yang akan digunakan pada database.
c. urls.py, merupakan *file* yang digunakan untuk mendefinisikan tautan *endpoint* dari modul *customer*.
d. views.py, merupakan *file* yang digunakan untuk membuat fungsi yang akan digunakan pada *endpoint*.

Aktivitas selanjutnya adalah melakukan konfigurasi untuk mendaftarkan app *customer* yang telah dibuat kedalam *file* base.py, yang berada didalam folder "crinoid/settings/". Cara pendaftaran app ini dimulai dengan menuliskan fungsi pada *file* apps.py, yang berada didalam folder "*customer*" seperti yang dapat dilihat pada Gambar 3.6.

```
from django.apps import AppConfig


class CustomerConfig(AppConfig):
    name = 'customer'
```

Gambar 3.6 Blok Kode *File* apps.py Pada App Customer

Kemudian pada *file* base.py, tuliskan blok kode program yang memanggil fungsi dari *file* apps.py seperti yang dapat dilihat pada Gambar 3.7.

```
INSTALLED_APPS = [
    'customer.apps.CustomerConfig',
]
```

Gambar 3.7 Blok Kode *File* base.py Pada Folder crinoid\settings

Mendefinisikan Model

Pada aktivitas ini, yang lakukan adalah mengimplementasikan hasil rancangan tabel data baru yang telah dibuat pada tahap perancangan. Aktivitas ini dimulai dengan memanfaatkan *file* models.py yang ada di dalam folder atau app *customer*, kemudian dilkakukan penulisan beberapa blok kode seperti yang dapat dilihat pada Gambar 3.8.

```
from django.db import models
from django.contrib.auth import get_user_model
User = get_user_model()

class Channel(models.Model):
    SOSMED = {
        ('OF', 'Offline Store'),
        ('WA', 'Whatsapp'),
        ('IG', 'Instagram'),
        ('FB', 'Facebook'),
        ('TW', 'Twitter'),
        ('LL', 'Lainnya'),
    }
    name = models.CharField(choices=SOSMED, max_length = 30)
    akunSosmed = models.CharField(max_length=30, null=True, blank=True)

    def __str__(self):
        return self.name

class Customer(models.Model):
    GENDER = {
        ('L', 'Laki-laki'),
        ('P', 'Perempuan'),
    }
    userId = models.ForeignKey(User, on_delete=models.CASCADE, blank=True,
null=True)
    name = models.CharField(max_length=25)
    phoneNumber = models.CharField(max_length=20)
    gender = models.CharField(choices=GENDER, max_length=1, null=True,
blank=True)
    joinedDate = models.DateTimeField(auto_now_add=True)
    dob = models.DateField(null=True, blank=True)
    alamat = models.CharField(max_length=200, null=True, blank=True)
    channel = models.ManyToManyField(Channel, related_name='channel')

    def __str__(self):
        return self.name
```

Gambar 3.8 Blok Kode *File* models.py Pada App Customer

Jika mengacu pada Gambar 3.8 tersebut, dapat dilihat model *customer* yang memiliki hubungan dengan model *user* dan *channel* telah ditulis ke dalam kode lengkap dengan atribut serta tipe datanya. Setelah itu, langkah selanjutnya adalah melakukan migrasi model yang sudah dibuat dengan cara menuliskan perintah dengan *syntax* `python manage.py makemigrations customer` dan `python manage.py migrate customer`.

Membuat Serializers

Pada langkah ini, aktivitas yang dilakukan adalah membuat *file* serializers untuk mengonversi *instance* model menjadi JSON sehingga *frontend* dapat bekerja dengan data yang diterima dengan mudah. Aktivitas ini dimulai dari membuat *file* baru yang bernama serializers.py pada folder atau app *customer*. Kemudian dilakukan penulisan beberapa blok kode seperti yang dapat dilihat pada Gambar 3.9.

```
from django.db.models import fields
from .models import Customer, Channel
from rest_framework import serializers

class ChannelSerializer(serializers.ModelSerializer):
    class Meta:
        model = Channel
        fields = ['name', 'akunSosmed']

class CustomerSerializer(serializers.ModelSerializer):
    channel = ChannelSerializer(many=True)

    class Meta:
        model = Customer
        fields = ('userId','id', 'name', 'phoneNumber', 'gender',
'joinedDate', 'dob', 'alamat', 'channel')

    def validate(self, data):
        channel = data['channel']
        listChannel = []
        for chn in channel:
            listChannel.append(chn['name'])
        if len(listChannel) == len(set(listChannel)):
            return data
        else:
            raise serializers.ValidationError("Channel tidak boleh lebih
dari satu")

    def create(self, validated_data):
        channel = validated_data.pop('channel', None)
        customer = Customer.objects.create(**validated_data)
        if channel:
            for chn in channel:
                channel_instance = Channel.objects.create(**chn)
                customer.channel.add(channel_instance)

        return customer
```

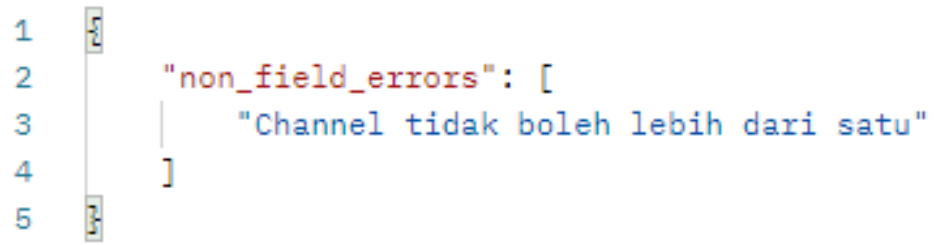
```
    def update(self, instance, validated_data):
        channel_new = validated_data.pop('channel')

        channel_old = instance.channel
        instance.name = validated_data.get('name', instance.name)
        instance.phoneNumber = validated_data.get('phoneNumber',
instance.phoneNumber)
        instance.gender = validated_data.get('gender', instance.gender)
        instance.joinedDate = validated_data.get('joinedDate',
instance.joinedDate)
        instance.dob = validated_data.get('dob', instance.dob)
        instance.alamat = validated_data.get('alamat', instance.alamat)
        instance.save()
        instance.channel.clear()
        for chn in channel_new:
            channel_instance = Channel.objects.create(**chn)
            instance.channel.add(channel_instance)
        return instance
```

Gambar 3.9 Blok Kode *File* serializers.py Pada App *Customer*

Jika mengacu pada Gambar 3.9 tersebut, dapat dilihat terdapat class `class ChannelSerializer` dan `class CustomerSerializer`. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut:

- Blok kode `class ChannelSerializer` dan `class CustomerSerializer`, berfungsi untuk menampung `class Meta`.
- Blok kode `class Meta`, berfungsi untuk mendefinisikan model dan *fields* yang akan digunakan dan ditampilkan dengan *response* format JSON.
- Blok kode `def validate`, `def create`, dan `def update` pada `CustomerSerializer`, berfungsi untuk memegang kendali validasi ketika membuat atau mengubah data *customer* berserta data *channel*, maka *channel* yang ditambahkan tidak boleh sama atau tidak boleh lebih dari satu, seperti yang dapat dilihat pada Gambar 3.10.



Gambar 3.10 Hasil Validasi Data *Channel*.

Membuat Controller

Pada langkah ini, aktivitas yang dilakukan adalah membuat beberapa fungsi yang akan digunakan pada setiap modul *customer*. Fungsi yang dibuat pada *file view* ini nantinya akan digunakan dan dipanggil juga untuk membuat tautan *endpoint* sesuai dengan rancangan. Aktivitas ini dimulai dengan memanfaatkan *file* views.py yang ada di dalam folder atau app *customer*, kemudian dilakukan penulisan beberapa blok kode seperti yang dapat dilihat pada Gambar 3.11.

```
from rest_framework.response import Response
from rest_framework import status
from customer.models import Customer
from customer.serializers import CustomerSerializer
from rest_framework.views import APIView
from django.http import Http404
import math

# Create your views here.
class CustomerList(APIView):

    def get(self, request, format=None):
        user = request.user
        customers_count = Customer.objects.filter(userId=user).count()
        customer_page = int(request.GET.get('page')) if
request.GET.get('page') is not None and request.GET.get('page') != '' else
1
        customer_limit = int(request.GET.get('limit')) if
request.GET.get('limit') is not None and request.GET.get('limit') != ''
else 10
        customer_search = request.GET.get('search') if
request.GET.get('search') is not None and request.GET.get('search') != ''
else ""
        phoneNumber_search = request.GET.get('searchPhone') if
request.GET.get('searchPhone') is not None and
request.GET.get('searchPhone') != '' else ""
        if phoneNumber_search:
            customers = Customer.objects.filter(userId=user,
phoneNumber=phoneNumber_search).order_by('-id')[((customer_page-
1)*customer_limit):(customer_limit + ((customer_page-1)*customer_limit))]
        else:
            customers = Customer.objects.filter(userId=user,
name__icontains=customer_search).order_by('-id')[((customer_page-
1)*customer_limit):(customer_limit + ((customer_page-1)*customer_limit))]

        meta = {
            'totalData': customers_count,
            'limit': customer_limit,
            'totalPages': math.ceil(customers_count/customer_limit),
            'page': customer_page,
        }
        serializer = CustomerSerializer(customers, many=True)

        if customers.exists:
            return Response({"success": True, "code": status.HTTP_200_OK,
"message": "Berhasil mendapatkan daftar data customers", "meta": meta,
"data": serializer.data})
```

```
    def post(self, request, format=None):
        serializer = CustomerSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save(userId=self.request.user)
            return Response({"success": True, "code":
status.HTTP_201_CREATED, "message": "Simpan data berhasil", "data":
serializer.data})
        return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)

class CustomerDetail(APIView):

    def get_object(self, pk):
        try:
            return Customer.objects.get(pk=pk)
        except Customer.DoesNotExist:
            raise Http404

    def get(self, request, pk, format=None):
        try:
            user = request.user
            customers = Customer.objects.get(userId=user, id=pk)
            if customers:
                serializer = CustomerSerializer(customers)
                return Response({"success": True, "code":
status.HTTP_200_OK, "message": "Berhasil mendapatkan detail data
customers", "data": serializer.data})
        except Customer.DoesNotExist:
            return Response({"success": True, "code":
status.HTTP_400_BAD_REQUEST, "message": "Gagal mendapatkan detail data
customers"})

    def put(self, request, pk, format=None):
        try:
            user = request.user
            customers = Customer.objects.get(userId=user, id=pk)
            serializer = CustomerSerializer(instance=customers,
data=request.data)
            if serializer.is_valid():
                serializer.save()
                return Response({"success": True, "code":
status.HTTP_200_OK, "message": "Ubah detail data customer berhasil",
"data": serializer.data})
            return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)
        except Customer.DoesNotExist:
            return Response({"success": True, "code":
status.HTTP_400_BAD_REQUEST, "message": "Gagal mengubah detail data
customer"})

    def delete(self, request, pk, format=None):
        try:
            user = request.user
            customers = Customer.objects.get(userId=user, id=pk)
            customers.delete()

            return Response({"success": True, "code": status.HTTP_200_OK,
"message": "Hapus data customer berhasil"})
        except Customer.DoesNotExist:
```

```
            return Response({"success": True, "code":
status.HTTP_204_NO_CONTENT, "message": "Hapus data customer gagal"})
```

Gambar 3.11 Blok Kode *File* views.py Pada App *Customer*

Jika mengacu pada Gambar 3.11 tersebut, dapat dilihat `class CustomerList` dan `class CustomerDetail`. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut:

- Blok kode `class CustomerList`, berfungsi untuk memegang kendali dari setiap fungsi yang berkaitan dengan data *customer* secara keseluruhan. Contohnya seperti fungsi untuk melihat semua data dan menambah data *customer* yang sudah pasti tidak memerlukan parameter detail seperti "id" dari data.
- Blok kode `def get` pada `class CustomerList` berfungsi memegang kendali seperti mendapatkan semua data *customer* lengkap dengan informasi total keseluruhan data, limit data, halaman limit, dan total halaman limit yang didefinisikan didalam "meta" seperti yang dapat dilihat pada Gambar 3.12.

```
5    "meta": {
6        "totalData": 3,
7        "limit": 10,
8        "totalPages": 1,
9        "page": 1
10   },
```

Gambar 3.12 Hasil Meta Data

- Blok kode `def post` pada `class CustomerList` berfungsi memegang kendali untuk menambahkan data *customer* baru sesuai dengan format serializers.
- Blok kode `class CustomerDetail`, berfungsi untuk memegang kendali dari setiap fungsi yang berkaitan dengan data *customer* secara detail. Contohnya seperti fungsi untuk melihat, mengubah, atau menghapus data tertentu sehingga memerlukan parameter detail seperti "id" dari data.

Membuat Endpoint

Pada aktivitas ini, yang dilakukan adalah mengimplementasikan hasil rancangan *endpoint* yang telah dibuat pada tahap perancangan. Aktivitas ini dimulai dengan memanfaatkan *file* urls.py yang ada di dalam folder atau app *customer*, kemudian dilakukan penulisan beberapa blok kode seperti yang dapat dilihat pada Gambar 3.13.

```
from django.urls import path
from . import views

urlpatterns = [
    path('api/v1/customers/', views.CustomerList.as_view()),
#getcustomerList
    path('api/v1/customers/<int:pk>', views.CustomerDetail.as_view()),
#getcustomerDetail
]
```

Gambar 3.13 Blok Kode *File* urls.py Pada App *Customer*

Kemudian setelah selesai menuliskan beberapa blok kode pada *file* urls.py yang terdapat pada folder atau app *customer*, langkah selanjutnya yaitu mendaftarkan atau mendefinisikan *file* tersebut di dalam "crinoid\urls.py" seperti yang dapat dilihat pada Gambar 3.14.

```
from django.urls import path
from django.conf.urls import include
from django.contrib import admin
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [

    path('', include('customer.urls')),

] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT) +
static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

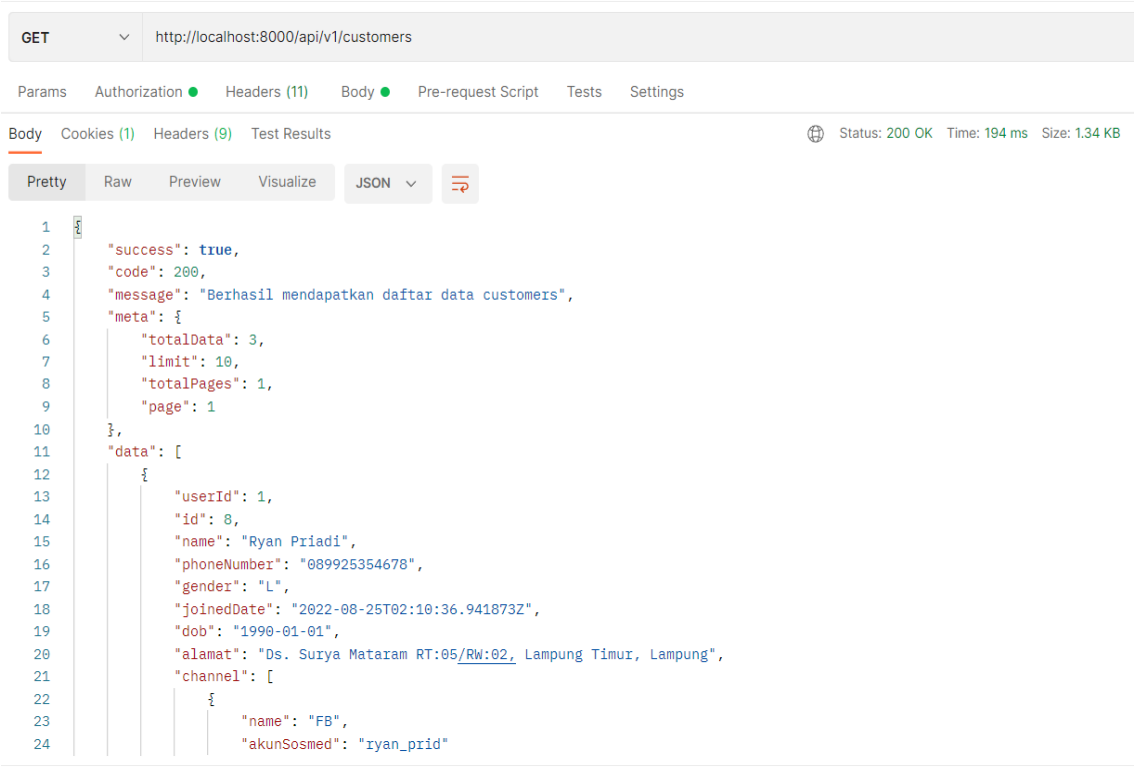
Gambar 3.14 Blok Kode *File* urls.py Pada Folder Crinoid

3.2.3 Pengujian REST API

Setelah diimplementasi, kemudian dilakukan dan didapatkan hasil pengujian. Pengujian dilakukan menggunakan metode *white-box* dengan cara memanfaatkan aplikasi *testing* api yang bernama postman. Dengan menggunakan postman, sehingga dapat dilihat kelayakan dan keseuaian dari setiap unit kode REST API yang telah dibuat. Berikut ini beberapa hasil pengujian dari tiap-tiap fungsi dan metode dari tautan *endpoint* yang telah dibuat antara lain:

a. Melihat Semua Data *Customer*

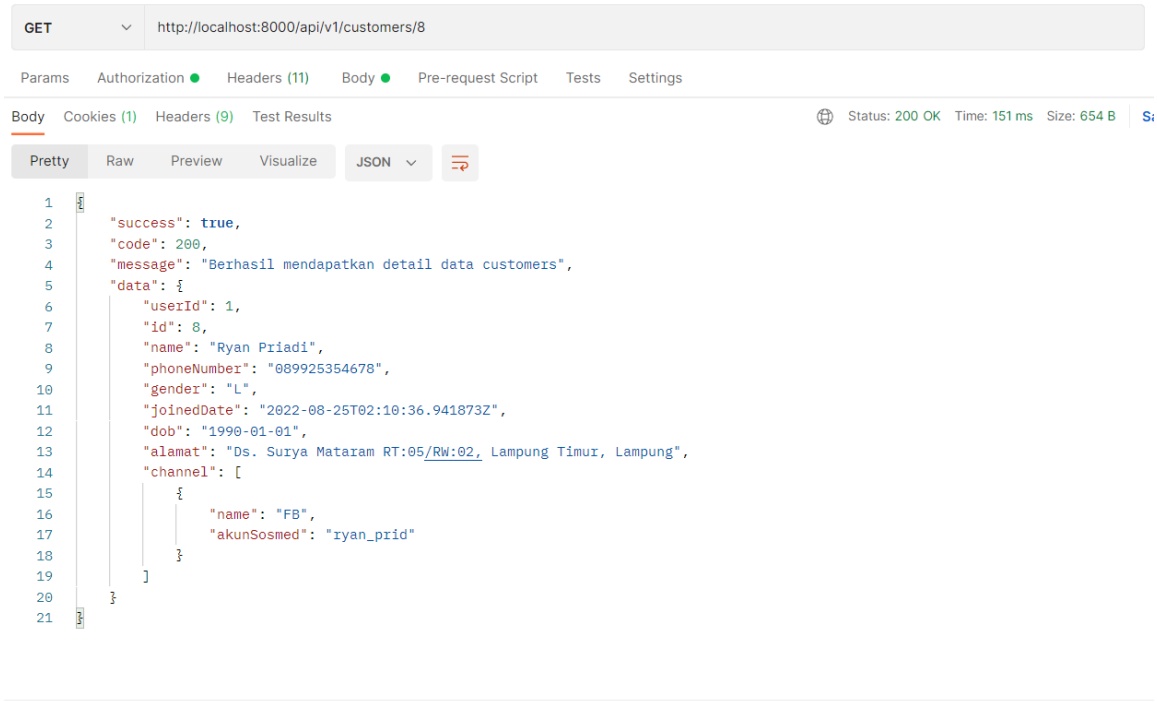
Pada pengujian pertama, dapat dilihat *endpoint* api/v1/customers yang digunakan untuk melihat semua data *customer* telah berhasil mengirim *response* melalui protokol HTTP seperti pada Gambar 3.15. Melalui postman yang seolah-olah menjadi REST *client* mengirimkan *request* melalui tautan *endpoint* http://localhost:8000/api/v1/customers dan *method GET*, kemudian REST server akan mengirimkan response berupa *resource* dalam format data JSON. Response tersebut dapat dilihat berhasil tidaknya melalui status *code* http 200 yang menandakan memiiki arti *success*.



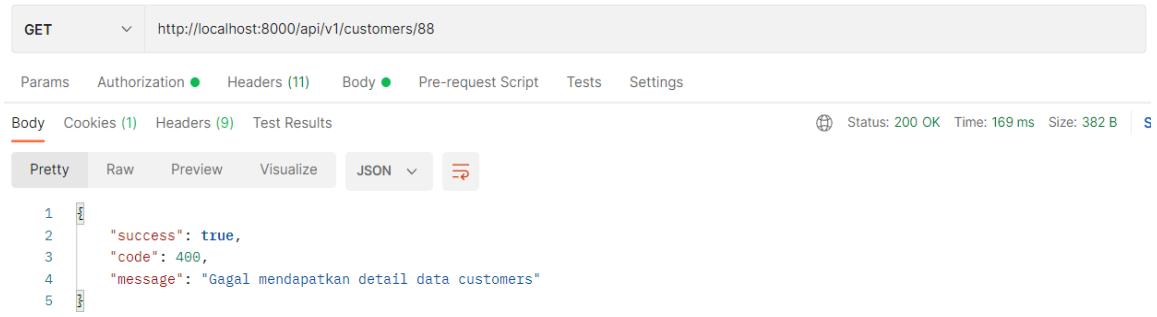
Gambar 3.15 *Response* Melihat Semua Data *Customer* Berhasil

b. Melihat Detail Data *Customer*

Pengujian kedua yaitu pengujian untuk melihat detail data *customer* berdasarkan id, dapat dilihat endpoint api/v1/customers/id yang digunakan untuk melihat semua data *customer* telah berhasil mengirim *response* melalui protokol HTTP seperti pada Gambar 3.16. Melalui postman yang seolah-olah menjadi rest *client* mengirimkan *request* melalui tautan *endpoint* http://localhost:8000/api/v1/customer/8 dan masih menggunakan *method GET*, kemudian rest server akan mengirimkan response berupa *resource* yang memiliki ID 8 dalam format data JSON. Jika *request* berhasil maka *response* akan mengirimkan status *code* http 200 yang menandakan memiiki arti *success*, tetapi jika *request* gagal maka *response* akan mengirimkan status *code* http 400 yang menandakan kesalahan dalam melakukan *request* seperti pada Gambar 3.17.



Gambar 3.16 *Response* Melihat Detail Data *Customer* Berhasil.

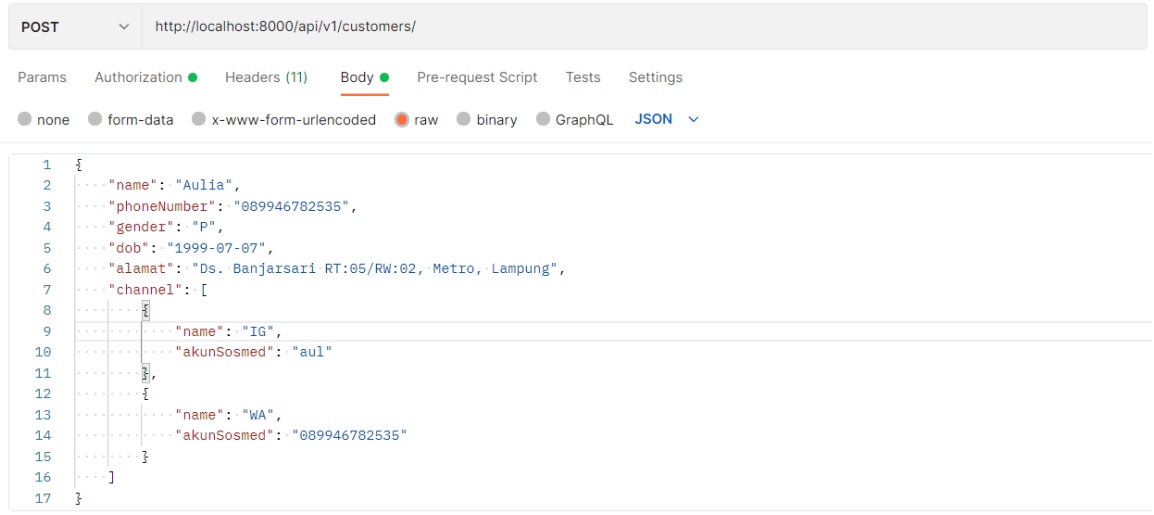


Gambar 3.17 *Response* Melihat Detail Data *Customer* Gagal.

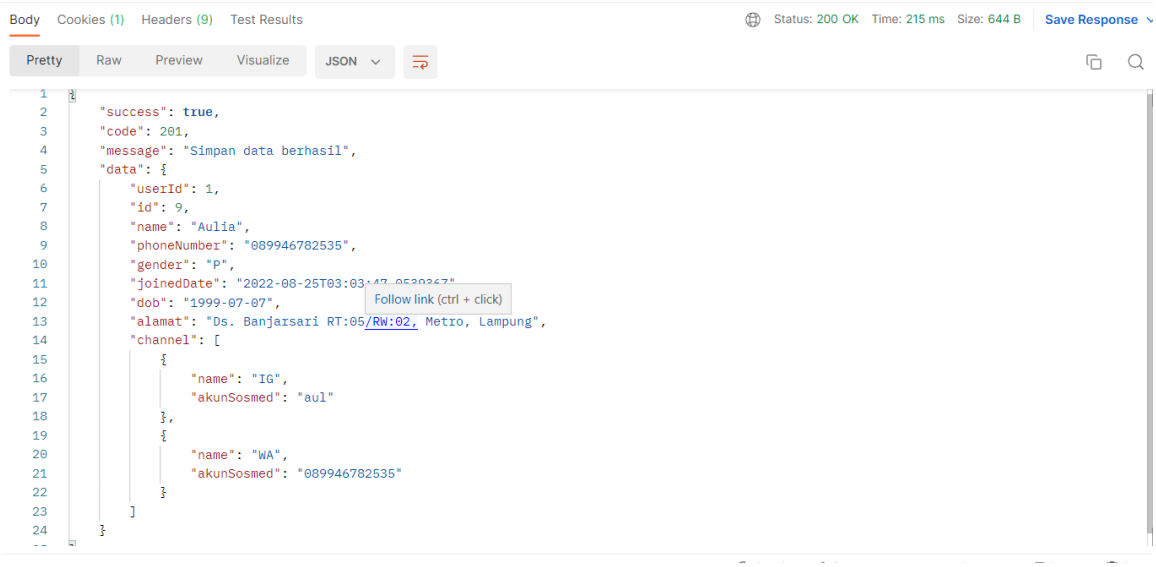
c. Membuat Data *Customer*

Pengujian ketiga yaitu merupakan pengujian untuk membuat data *customer* baru. Pengujian ini dilakukan dengan menggunakan *endpoint* api/v1/customers, walaupun *endpoint* yang digunakan sama dengan yang terdapat pada point a, namun terdapat perbedaan perlakuan pada saat melakukan *request*. Dapat dilihat seperti pada Gambar 3.18, pada pengujian kali ini membutuhkan ‘Body’ yang berisi ‘raw’ parameter atribut beserta *value* datanya dalam format JSON. Setelah itu, melalui postman yang seolah-olah menjadi rest *client* mengirimkan *request* dengan ‘Body’ tersebut melalui tautan *endpoint* http://localhost:8000/api/v1/customers dan menggunakan method POST, kemudian rest server akan mengirimkan response berupa *resource* yang baru ditambahkan dalam format

data JSON beserta dengan status *code* http 200 yang menandakan memiliki arti *success* seperti pada Gambar 3.19.



Gambar 3.18 Body Request Untuk Membuat Data *Customer* Baru

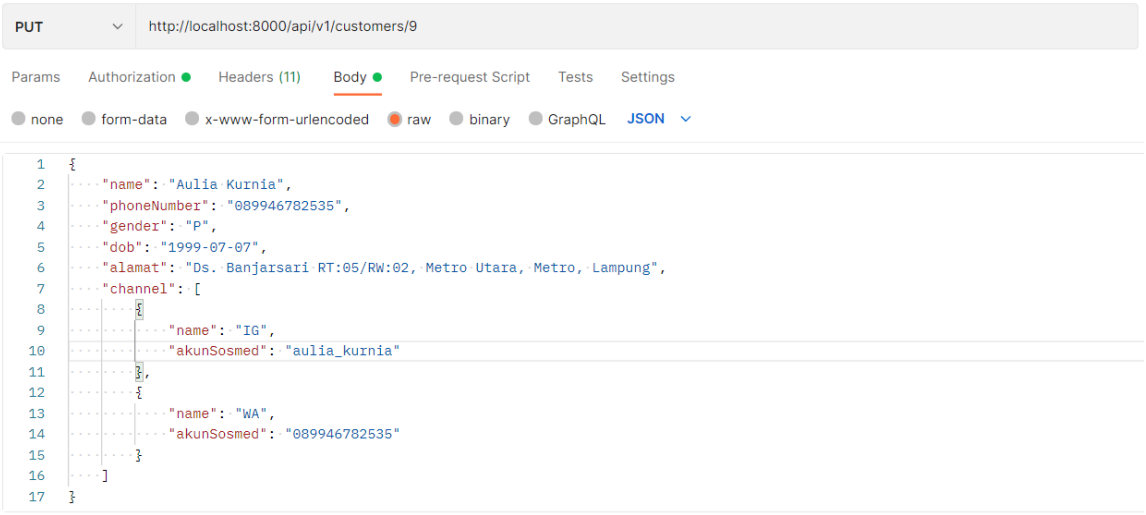


Gambar 3.19 *Response* Membuat Data *Customer* Baru Berhasil

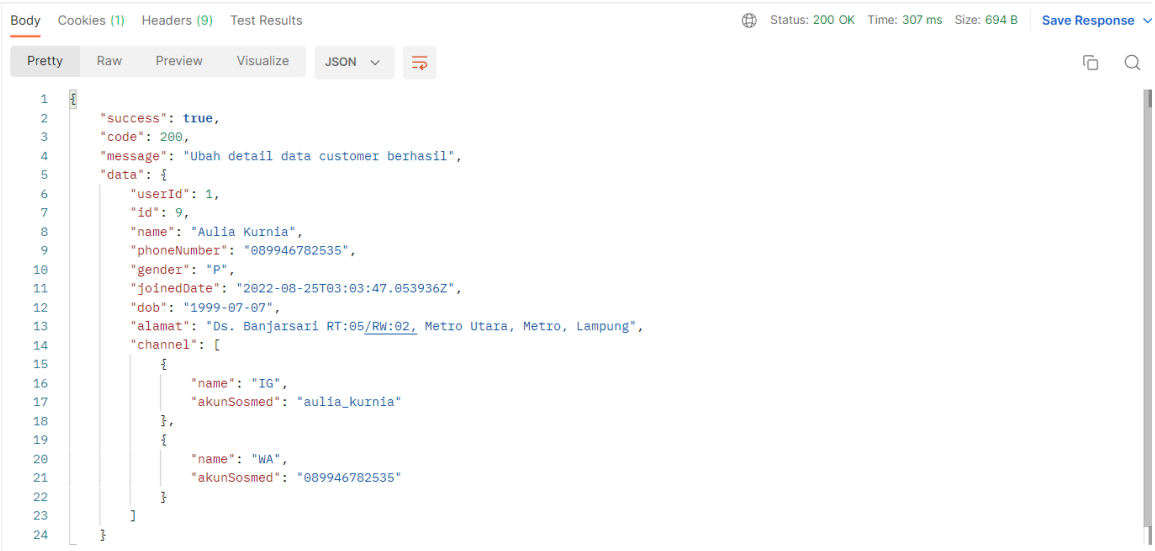
d. Mengubah Data *Customer*

Pengujian keempat yaitu merupakan pengujian untuk mengubah data *customer*. Pengujian ini dilakukan dengan menggunakan *endpoint* api/v1/customers/id, walaupun *endpoint* yang digunakan sama dengan yang terdapat pada point b, namun terdapat perbedaan perlakuan pada saat melakukan *request*. Dapat dilihat pada Gambar 3.20, pengujian kali ini membutuhkan ‘Body’ yang berisi ‘raw’ parameter atribut beserta *value* datanya yang akan diubah dalam format JSON. Setelah itu, melalui postman yang seolah-olah menjadi rest *client* mengirimkan *request* dengan ‘Body’ tersebut melalui tautan *endpoint*

http://localhost:8000/api/v1/customer/9 dan menggunakan *method PUT*, kemudian REST server akan mengirimkan *response* berupa *resource* dengan ID 9 telah diubah dalam format data JSON beserta dengan status *code* http 200 yang menandakan memiliki arti *success* seperti yang dapat dilihat pada Gambar 3.21.



Gambar 3.20 *Body Request* Untuk Mengubah Data *Customer*

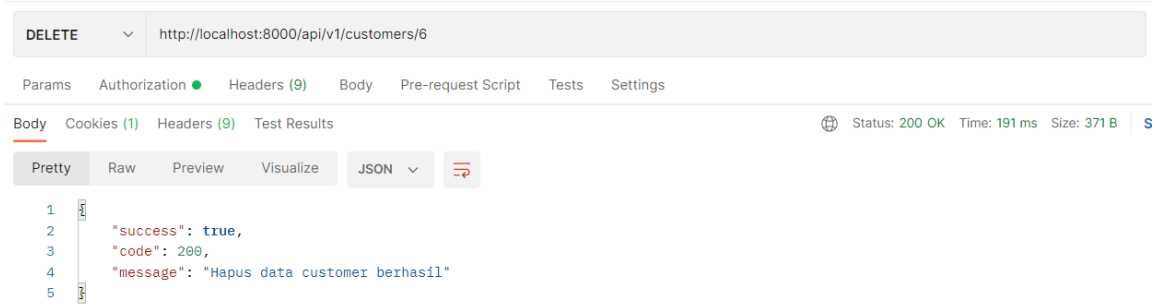


Gambar 3.21 *Response* Mengubah Data *Customer* Berhasil

e. Menghapus Data *Customer*

Pengujian kelima yaitu pengujian untuk menghapus data *customer* berdasarkan id, dapat dilihat *endpoint* api/v1/customers/id yang digunakan untuk menghapus data *customer* telah berhasil mengirim *response* melalui protokol HTTP. Melalui postman yang seolah-olah menjadi REST *client* mengirimkan *request* melalui tautan *endpoint* http://localhost:8000/api/v1/customer/6 dan menggunakan *method DELETE*, kemudian

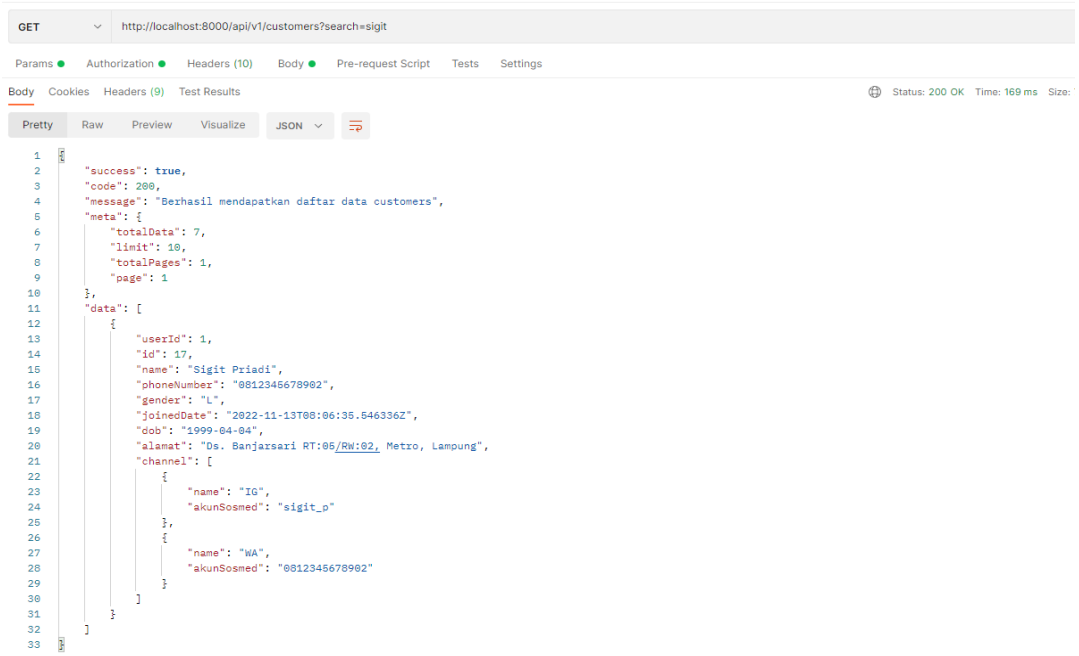
REST server akan mengirimkan *response* berupa status *code* http 200 yang menandakan memiiki arti *success* seperti yang dapat dilihat pada Gambar 3.22.



Gambar 3.22 *Response* Menghapus Data *Customer* Berhasil

f. Mencari Data Customer Berdasarkan Nama

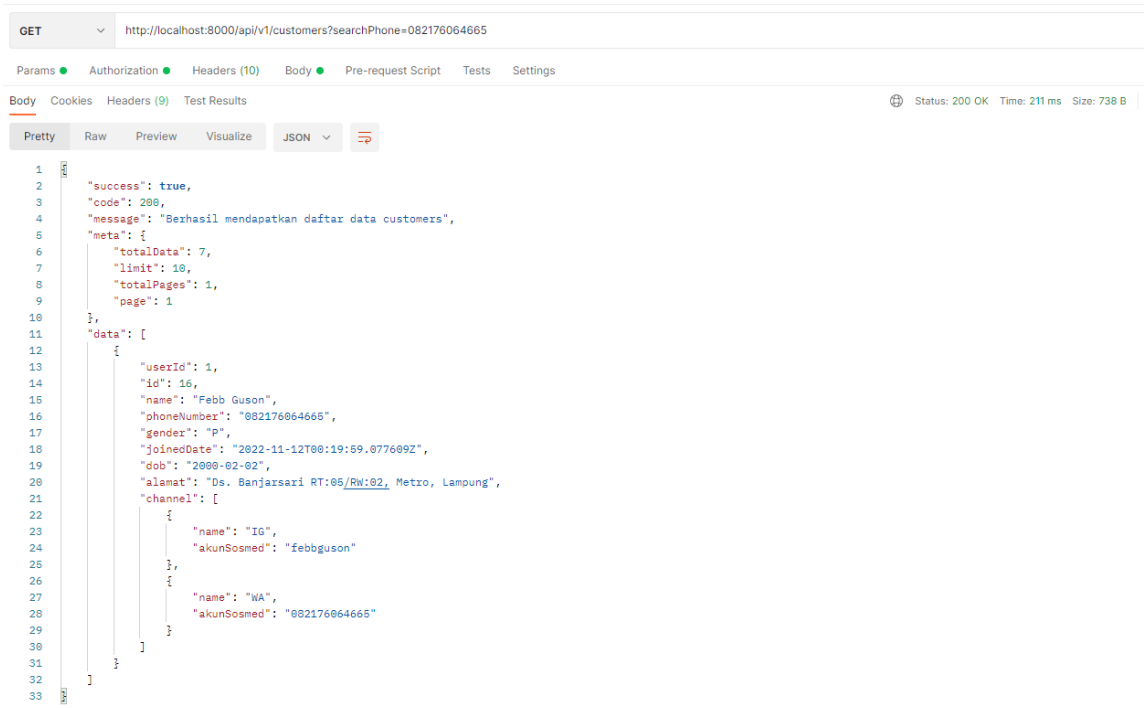
Pengujian keenam yaitu pengujian untuk mencari data customer berdasarkan nama customer. Pengujian memanfaatkan endpoint api/v1/customers?key=value yang digunakan untuk mencari data telah berhasil mengirim response melalui protokol HTTP. Dengan memanfaatkan postman yang seolah-olah menjadi REST client mengirimkan request melalui tautan endpoint http://localhost:8000/api/v1/customer?search=febb dan menggunakan method GET, kemudian rest server akan mengirimkan response berupa *resource* data customer yang memiliki value “febb” pada key “search” seperti yang dapat dilihat pada Gambar 3.23.



Gambar 3.23 *Response* Mencari Data *Customer* Berdasarkan Nama

g. Mencari Data Customer Berdasarkan Nomor Telepon

Pengujian ketujuh yaitu pengujian untuk mencari data customer berdasarkan nomor telepon. Pengujian memanfaatkan endpoint api/v1/customers?key=value yang digunakan untuk mencari data telah berhasil mengirim response melalui protokol HTTP. Walaupun *endpoint* yang digunakan sekilas sama dengan yang terdapat pada point f, namun terdapat perbedaan nama key yang digunakan. Pada pengujian point f memiliki nama key "search, sedangkan pada point g memiliki nama key "searchPhone". Dengan memanfaatkan postman yang seolah-olah menjadi REST client mengirimkan request melalui tautan endpoint http://localhost:8000/api/v1/customer?searchPhone=082176064665 dan menggunakan method GET, kemudian rest server akan mengirimkan response berupa resource data customer yang memiliki value "082176064665" pada key "searchPhone" seperti yang dapat dilihat pada Gambar 3.24.



Gambar 3.24 *Response* Mencari Data *Customer* Berdasarkan Nomor Telepon

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Selama kurang lebih dalam kurun waktu 6 bulan melaksanakan proses magang di PT Bhinneka Mentari Dimensi, terdapat teori yang berkaitan dengan implementasi yang terjadi di lapangan. Teori tersebut merupakan penerapan web *service* API yang menggunakan arsitektur REST. Perenapan konsep REST ini digunakan dalam pengembangan proyek crino.id untuk penambahan modul *Customer Management*. Pada pengembangan proyek tersebut, penerapan REST dibangun menggunakan salah satu kerangka kerja dari bahasa pemrograman Python yaitu Django, dan menggunakan *toolkit* milik Django bernama Django REST *Framework*. REST menggunakan protokol HTTP (*Hypertext Transfer Protocol*) dan memanfaatkan beberapa metode yang ada di dalam HTTP antara lain: *GET*, *POST*, *PUT*, dan *DELETE*. Berdasarkan metode tersebut, maka dibuatlah fungsi atau *controller* yang kemudian dapat diakses melalui tautan *endpoint* sehingga memiliki keluaran berupa data dengan format JSON.

4.2 Pembelajaran Magang Teknis dan Non Teknis

4.2.1 Manfaat Magang

Selama 6 bulan menjalani proses penjaluran magang melalui program magang bersertifikat kampus merdeka, banyak sekali manfaat yang didapatkan. Pasalnya, magang merupakan pengalaman yang sangat berharga, apalagi bisa mendapat kesempatan magang di salah satu perusahaan *e-commerce* pertama di Indonesia yaitu bhinneka. Manfaat pertama adalah dapat merasakan pengalaman kerja dan menerapkan ilmu dari beberapa mata kuliah di kampus langsung dalam dunia pekerjaan secara profesional. Ilmu tersebut antara lain: mulai dari ilmu berfikir secara komputasional dari mata kuliah Logika Pemrograman yang dapat menjadi dasar mendukung ilmu *programming* web dari mata kuliah Pengembangan Aplikasi Berbasis Web (PABW) dan Pengembangan Sistem Informasi (PSI), kemudian ilmu manajemen proyek dari mata kuliah Pemikiran Desain dan mata kuliah Manajemen Pengembangan Teknologi Informasi. Manfaat kedua adalah dapat menambah wawasan pengetahuan dan relasi, belajar meningkatan kemampuan manajemen waktu yang didapat dari mata kuliah Manajemen Diri, serta dapat meningkatkan komunikasi baik secara lisan maupun tulisan yang didapatkan

langsung dari mata kuliah Bahasa Indonesia Komunikasi Ilmiah dan mengikuti kegiatan organisasi yang ada di kampus.

4.2.2 Kendala, Hambatan, dan Tantangan Magang

Selama 6 bulan menjalani proses penjaluran magang, selain manfaat tentu terdapat juga kendala, hambatan, dan tantangan yang dialami. Kendala yang dialami adalah sedikit kesulitan menyesuaikan diri dengan beberapa alat atau teknologi yang biasa digunakan perusahaan belum pernah diajarkan atau dipraktikan secara langsung selama masa perkuliahan. Contohnya antara lain seperti pemanfaatan bahasa pemrograman Python dalam membangun perangkat lunak berbasis web service REST API dan menggunakan kerangka kerja web Django. Kemudian, pemanfaatan alat pengujian API bernama postman, pemanfaatan alat manajemen proyek bernanama Jira, hingga penggunaan alat komunikasi bernama Slack.

Kemudian, terdapat hambatan yang dialami selama menjalani proses magang. Hambatan pertama yaitu diakibatkan karena pekerjaan yang dilakukan secara *remote* atau pekerjaan yang dilakukan dari jarak jauh. Pekerjaan jarak jauh sebenarnya sudah bukan menjadi hal yang aneh seharusnya, apalagi jika berposisi sebagai orang IT. Tetapi, terkadang diskusi secara daring masih sering menimbulkan potensi yang akan membuat terjadinya kesalahan komunikasi. Kesalahan komunikasi tersebut tentu sangat beragam, diantaranya dapat timbul karena kesalahan seseorang individu itu sendiri, atau dikarenakan ketidakstabilan dan kurang merata nya fasilitas internet di seluruh daerah Indonesia. Selain itu, terkadang komunikasi yang dilakukan hanya melalui *chat* dirasa kurang efektif atau efisien. Hambatan kedua yaitu diakibatkan karena adanya kesalahpahaman terkait perencanaan atau perancangan pengembangan yang akan dilakukan antara pihak atasan perusahaan dengan peserta magang khususnya di tim brittlestar yang mengembangkan crino.id. Hambatan ketiga yaitu dikarenakan berkurangnya jumlah peran Software Developer dari yang awalnya 3 orang menjadi 2 orang pada saat sprint 3 berlangsung. Kejadian itu bertepatan dengan waktu rotasi peran yang telah ditentukan diawal pada saat pergantian peran dari *Back End* ke *Lead Developer*. Hal tersebut membuat pengembangan proyek yang dilakukan menjadi terhambat dan tidak sesuai waktu pengembangan yang telah dirancang pada awalnya hingga akhirnya proyek diberhentikan sementara dan ditutup.

Berikutnya, berdasarkan beberapa kendala dan hambatan yang telah dijelaskan, maka dapat disimpulkan tantangan utama yang dirasakan adalah pada manajemen waktu. Pasalnya, selama menjalani proses magang, sebagai profesional tetap harus dapat melakukan pekerjaan

dan kewajiban lain yang berjalan secara bersamaan. Meskipun terkadang rasa kewalahan muncul apalagi jika tugas yang dikerjakan memiliki kompleksitas tinggi ditambah batas waktu pengerjaan yang harus berkejaran dengan durasi magang. Kemudian, adanya sistem rotasi peran setiap 2 kali waktu sprint yang diterapkan sebenarnya dirasa memiliki kelebihan tersendiri. Kelebihan itu antara lain dapat menambah referensi atau wawasan terkait peran tersebut, dan dapat merasakan pengalaman yang berbeda sehingga dengan harapan dapat menemukan minat peran di jenjang karir pekerjaan kedepannya. Namun pada realitanya, sistem rotasi tersebut membuat beban pekerjaan dan tanggung jawab semakin terasa berat. Hal tersebut dikarenakan pada setiap rotasi peran sebagai seorang *Software Developer*, harus mempelajari perbedaan alur pekerjaan pecahan peran antara *Back End*, *Lead Developer*, atau *Front End*. Adapun terkait perubahan rencana pengembangan proyek karena adanya beberapa kesalahan teknis maupun non-teknis itu sudah menjadi hal biasa di dalam industri pekerjaan. Pengalaman tersebut terasa selama menjalani proses magang di PT Bhinneka Mentari Dimensi.

Oleh karena itu, sebagai seorang IT atau apapun bidang profesi nya, diharapkan tetap harus terbiasa untuk beradaptasi akan suatu perubahan. Kemudian, hikmah atau pelajaran yang dapat diambil selama menjalani magang adalah apapun situasi dan kondisinya, perihal komunikasi harus tetap yang utama. Pastikan selalu melakukan komunikasi secara intens, detail dan berkala ke seluruh elemen tim atau stakeholder proyek mulai dari perencanaan, pengerjaan, hingga akhirnya pengembangan diselesaikan. Komunikasi tersebut dapat meliputi beberapa hal mulai dari teknis dan non-teknis pengembangan proyek secara detail, sehingga nantinya tidak ada lagi kesalahpahaman yang membuat proyek diberhentikan sementara atau ditutup.

4.2.3 Kontribusi Magang

Selama menjalani proses magang, sedikit banyak tentu harus melakukan beberapa kontribusi. Kontribusi yang telah diberikan terjadi dan terbagi di berbagai pengembangan proyek. Pada proyek pertama dengan tim brittlestar, sebagai seorang *software developer intern* sudah mendapatkan 2 kali rotasi perubahan peran. Dalam proyek brittlestar, pertama kali mendapatkan pecahan peran menjadi seorang *Back End*. Pada saat mendapatkan peran *Back End*, selain hanya mengerjakan tugas yang ada di sprint 1 dan 2, kontribusi yang dilakukan juga termasuk ikut membantu *Product Manager* melakukan perancangan penambahan tabel dan atribut baru pada database pada sistem yang ada. Kemudian, memberikan masukkan kepada PD terhadap hasil rancangan antarmuka dan membantu teman *Software Developer* lainnya apabila memiliki kendala dalam pengerjaan tugas.

Kemudian saat mendapatkan rotasi peran menjadi *Lead Developer*, kontribusi yang telah dilakukan adalah antara lain adalah membantu *Product Manager* untuk merancang *sprint Product Management* dan *Order Management* dan membantu *Product Manager* menganalisis API yang akan digunakan pada sistem crino.id dari beberapa *marketplace*. Selain itu, ikut turut serta membantu *Quality Assurance* dalam melakukan pengujian web service API, hal tersebut dikarenakan *Quality Assurance* kurang memahami dalam penggunaan alat pengujian API bernama postman. Adapun kontribusi lain saat menjadi *Lead Developer* adalah sembari merangkap peran menjadi seorang *Back End* kembali. Hal itu dikarenakan teman yang seharusnya berperan sebagai *Back End* pada rotasi tersebut, memiliki kendala yang membuat beban pekerjaan dan tanggung jawab pada sprint 3 terasa semakin lebih berat. Walaupun pada akhirnya, proyek brittlestar yang akan mengembangkan crino.id ini diberhentikan dan ditutup sementara dalam waktu dekat.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Selama berkontribusi dalam pengembangan proyek crino.id dan khusus untuk penambahan modul *Customer Management*, maka dapat disimpulkan bahwa penambahan modul berhasil mencapai tujuan. Pada pengembangan tersebut telah berhasil dilakukan pengerjaan proyek sesuai dengan spesifikasi teknologi dan kebutuhan perancangan awalnya. Pengembangan dilakukan menggunakan salah satu kerangka kerja dari bahasa pemrograman Python yaitu Django dan memanfaatkan *toolkit* yang bernama Django REST *Framework*, sehingga modul *Customer Management* berbasis web *service* API dengan arsitekteur REST telah berhasil diimplementasikan. Hasil pengujian modul *customer* dari setiap *endpoint* dipastikan sudah berhasil dan fungsi dari setiap *method* telah berjalan dengan baik serta sesuai dengan rancangan. Kemudian hasil penerapannya, web *service* API *Customer Management* crino.id yang telah dibuat juga berhasil digunakan pada *dashboard seller center bhinneka.com* seperti yang dapat dilihat pada Lampiran A. Namun, apabila dilihat dari gagasan PT Bhinneka Mentari Dimensi untuk mengubah atau memecah semua proses bisnis pada sistem crino.id secara keseluruhan menjadi web *service*, maka dapat disimpulkan masih jauh dari kata berhasil. Hal itu dikarenakan satu dan lain hal yang membuat proyek brittlestar di tunda pengembangannya, sehingga masih banyak modul lain dari sistem yang ada belum sempat dikerjakan.

Oleh karena itu, dari kejadian tersebut dapat disimpulkan bahwa didalam dunia kerja yang sesungguhnya, perancangan awal pengembangan perangkat lunak yang sudah dirancang sangat matang dapat sewaktu-waktu berubah, baik dikarenakan hal teknis maupun non teknis. Kemudian dapat dipetik pelajaran bahwa tidak semua hal yang telah direncanakan dapat diwujudkan dengan mudah, tetapi harus tetap siap terbiasa untuk beradaptasi sesuai dengan situasi dan kondisi yang ada selama waktu pengembangan. Selain itu, selama kegiatan magang di PT Bhinneka Mentari Dimensi dirasa membuat pengalaman semakin bertambah. Dari pengalaman tersebut juga dapat disimpulkan bahwa peran sebagai *Back End* dirasa cocok untuk kedepannya. Namun seiring berjalannya waktu, kemampuan yang telah dimiliki saat ini perlu diasah, serta keinginan dan cita-cita tersebut harus diimbangi oleh usaha dan doa supaya terwujud di kemudian hari.

5.2 Saran

Penelitian pada laporan akhir ini pastinya masih memiliki banyak sekali kekurangan. Adapun berikut ini beberapa saran pada laporan akhir adalah sebagai berikut.

a. Saran Pengembangan, semoga dengan adanya penelitian laporan akhir ini, semoga dapat dijadikan acuan atau referensi untuk mengembangakan web *service* API menggunakan *framework* dan bahasa pemrograman serta arsitektur lainnya.
b. Saran PT Bhinneka Mentari Dimensi, semoga dapat ditingkatkan lagi untuk kesiapan proyek yang akan diberikan kepada pemagang, sehingga tidak terjadi lagi kesalahan komunikasi atau kesalahan kesepakatan untuk pengembangan proyek.
c. Saran Prodi Informatika UII, semoga dapat memberikan kesempatan yang lebih banyak lagi bagi mahasiswa/i untuk melakukan kegiatan magang, sehingga dapat menambah pengalaman dan wawasan yang lebih luas di lapangan.

DAFTAR PUSTAKA

Adhiwibowo, W., Setiaji, G. G., & Kumkamdhani, T. J. (2019). Pengembangan PDPT Berbasis Framework Dengan Teknologi Web Service SOAP dan REST di Universitas Semarang. *Jurnal Informatika Upgris*, *5*(2). http://journal.upgris.ac.id/index.php/JIU/article/view/4209

Afrizal, R., & Fitriyani. (2021). *Perancangan Web Service Berbasis REST API Untuk Aplikasi Penerimaan Peserta Didik Baru | eProsiding Teknik Informatika (PROTEKTIF)*. https://eprosiding.ars.ac.id/index.php/pti/article/view/201

Arévalo, J. G., Viecco, L., & Arévalo, L. (2020). Methodology to define an integration process between frameworks SCRUM, Django REST framework y Vue.js, implemented for software development, from quality management approach and agility. *IOP Conference Series: Materials Science and Engineering*, *844*(1), 012022. https://doi.org/10.1088/1757-899X/844/1/012022

Bhinneka. (2022). *Crinoid: Aplikasi Manajemen Marketplace Indonesia*. https://www.bhinneka.com/promo/crinoid

Christian, Y., & Bisma, R. (2021). Studi Perbandingan Performa Aplikasi Web Monolitik dan Microservice Berbasis Apache Kafka. *Journal of Informatics and Computer Science (JINACS)*, *3*(01), 79–88. https://doi.org/10.26740/JINACS.V3N01.P79-88

Firdaus, A., Widodo, S., Sutrisman, A., Gading, S., Nasution, F., Mardiana, R., Komputer, J. T., Negeri, P., & Palembang, S. (2019). Rancang Bangun Sistem Informasi Perpustakaan Menggunakan Web Service Pada Jurusan Teknik Komputer POLSRI. *Jurnal Informanika*, *5*(2). www.kursuswebsite.org

Galindra Wardhana, W., Arwani, I., & Rahayudi, B. (2020). *Implementasi Teknologi Restful Web Service Dalam Pengembangan Sistem Informasi Perekaman Prestasi Mahasiswa Berbasis Website (Studi Kasus: Fakultas Teknologi Pertanian Universitas Brawijaya)* (Vol. 4, Issue 2). http://j-ptiik.ub.ac.id

Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti, S., Pramusinto, W., Waluyo, S., Informatika, T., Teknologi Informasi, F., & Budi Luhur, U. (2017). Terakreditasi SINTA Peringkat 2 Pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order. *Masa Berlaku Mulai*, *1*(3), 106–112.

Kusumaningrum, A., Sajati, H., & Anarianto, D. (2019). *Rest and Soap Comparison on Web Service Technology for Android Based Data Services | Kusumaningrum | Conference SENATIK STT Adisutjipto Yogyakarta*. https://senatik.itda.ac.id/index.php/senatik/article/view/355

Lorentius, C., Gunadi, K., & Tjondrowiguno, A. (2022). *Pengenalan Aksara Jawa dengan Menggunakan Metode Convolutional Neural Network | Lorentius | Jurnal Infra*. https://publication.petra.ac.id/index.php/teknik-informatika/article/view/8075

Ngurah, G., & Kusuma, A. (2022). Aplikasi Pencatatan Inventori Berbasis Website Dengan Skema Autentikasi Dan Otorisasi Stateless Sederhana. *Journal of Innovation Research and Knowledge*, *1*(9), 1111–1120. https://www.bajangjournal.com/index.php/JIRK/article/view/1456

Pamuji, A. (2020). *Rancang Bangun Web Service Menggunakan Representational State Transfer Untuk Pengolahan Data Barang - UTY Open Access*. http://eprints.uty.ac.id/6287/

Putra, M. G. L., & Putera, M. I. A. (2019). Analisis Perbandingan Metode Soap Dan Rest Yang Digunakan Pada Framework Flask Untuk Membangun Web Service. *SCAN - Jurnal Teknologi Informasi Dan Komunikasi*, *14*(2), 1–7. https://doi.org/10.33005/SCAN.V14I2.1480

Ranga, V., & Soni, A. (2019). API Features Individualizing of Web Services: REST and SOAP SDN-based Attack Detection and Mitigation Frameworks View project API Features Individualizing of Web Services: REST and SOAP View project. *Article in International Journal of Innovative Technology and Exploring Engineering*, *8*, 2278–3075. https://doi.org/10.35940/ijitee.I1107.0789S19

Ridha, A. A., Ajie, H., & Duskarnaen, M. F. (n.d.). *Pengembangan Web Service Sistem Pembayaran Multibank Universitas Negeri Jakarta*. www.bankmandiri.co.id,

Saputra, D., & Fathoni Aji, R. (2018). Analisis Perbandingan Performa Web Service Rest Menggunakan Framework Laravel, Django Dan Ruby On Rails Untuk Akses Data Dengan Aplikasi Mobile (Studi Kasus: Portal E-Kampus STT Indonesia Tanjungpinang). *Bangkit Indonesia*, *2*.

Somya, R., & Nathanael, T. M. E. (2019). Pengembangan Sistem Informasi Pelatihan Berbasis Web Menggunakan Teknologi Web Service Dan Framework Laravel. *Techno Nusa Mandiri: Journal of Computing and Information Technology*, *16*(1), 51–58. https://doi.org/10.33480/TECHNO.V16I1.164

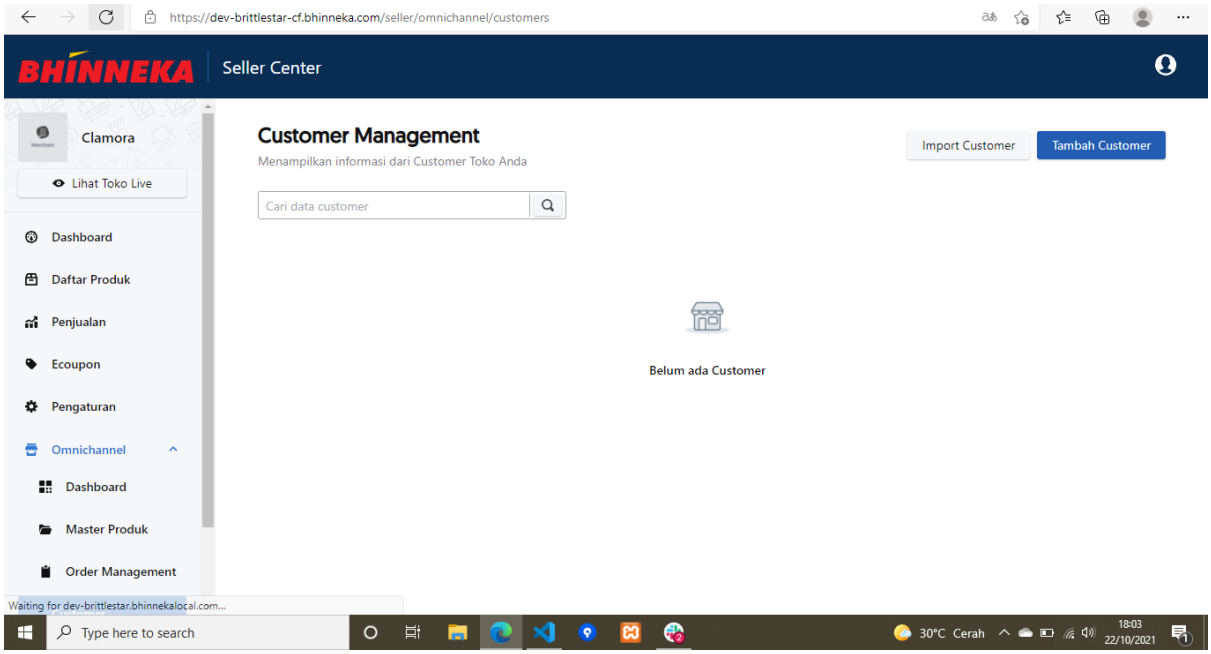
Sy, H., & Indo Intan. (2019). *Implementasi Restful Api Portal Akademik Stmik Dipanegara Berbasis Android | SISITI : Seminar Ilmiah Sistem Informasi dan Teknologi Informasi*. https://ejurnal.dipanegara.ac.id/index.php/sisiti/article/view/63-70

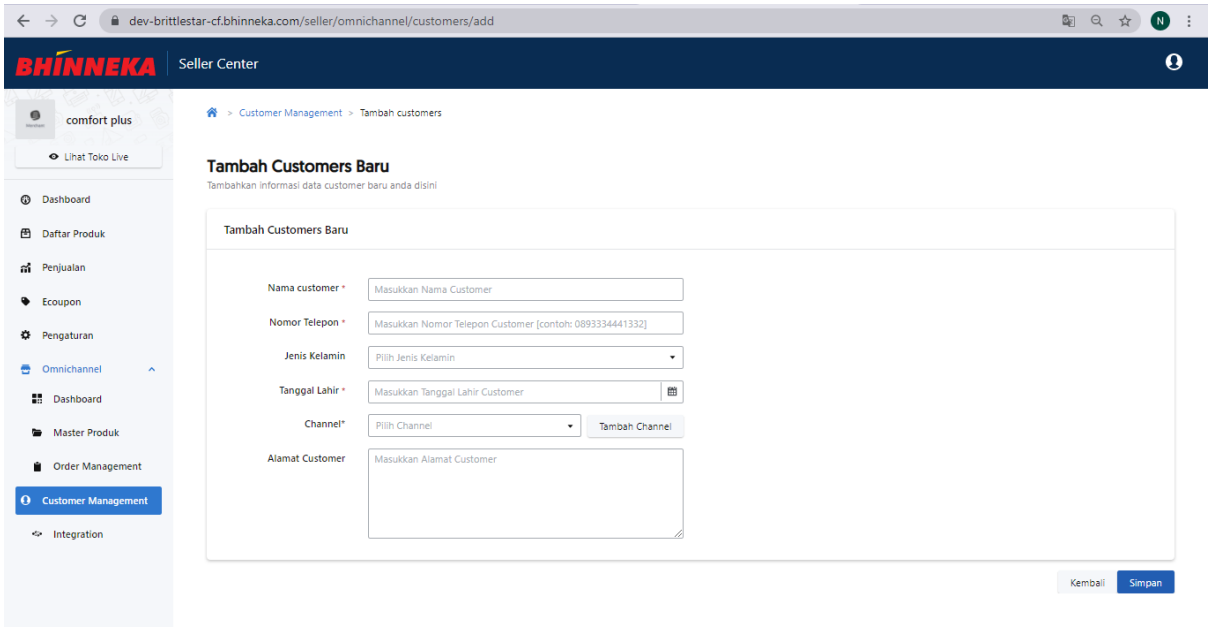
Uminingsih, U., & Handayani, S. D. (2020). Pengorganisasian Kerja Sistem Parkir Menggunakan Arsitektur Microservice. *Jurnal Teknologi*, *13*(1), 27–35. https://doi.org/10.3415/JURTEK.V13I1.2891

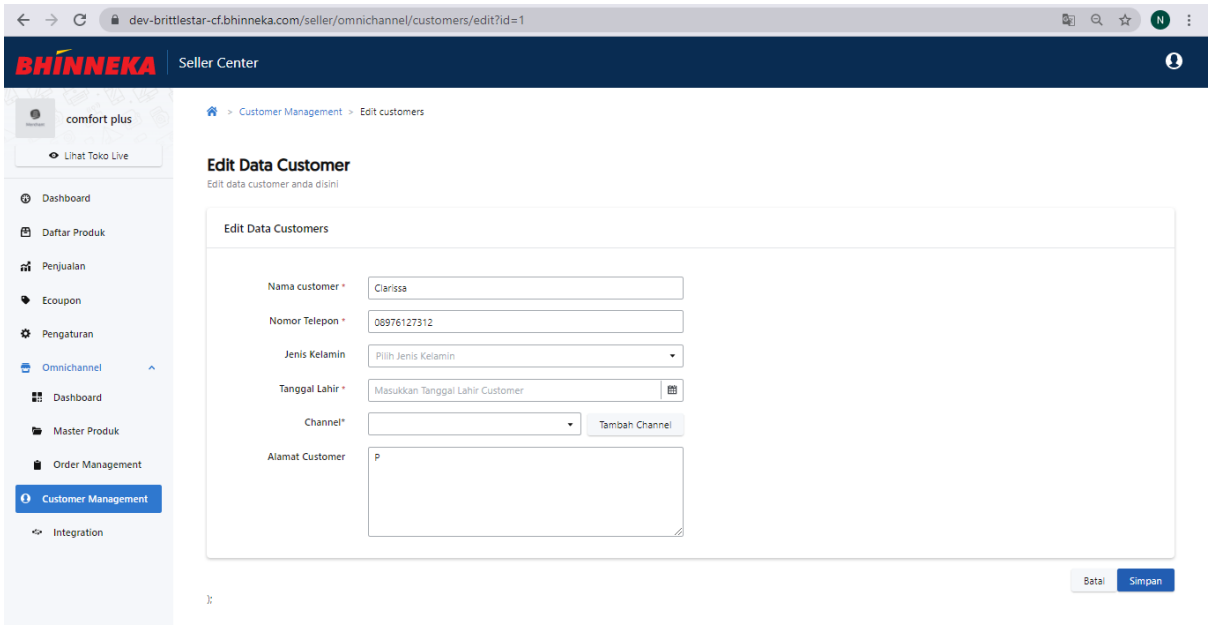
LAMPIRAN

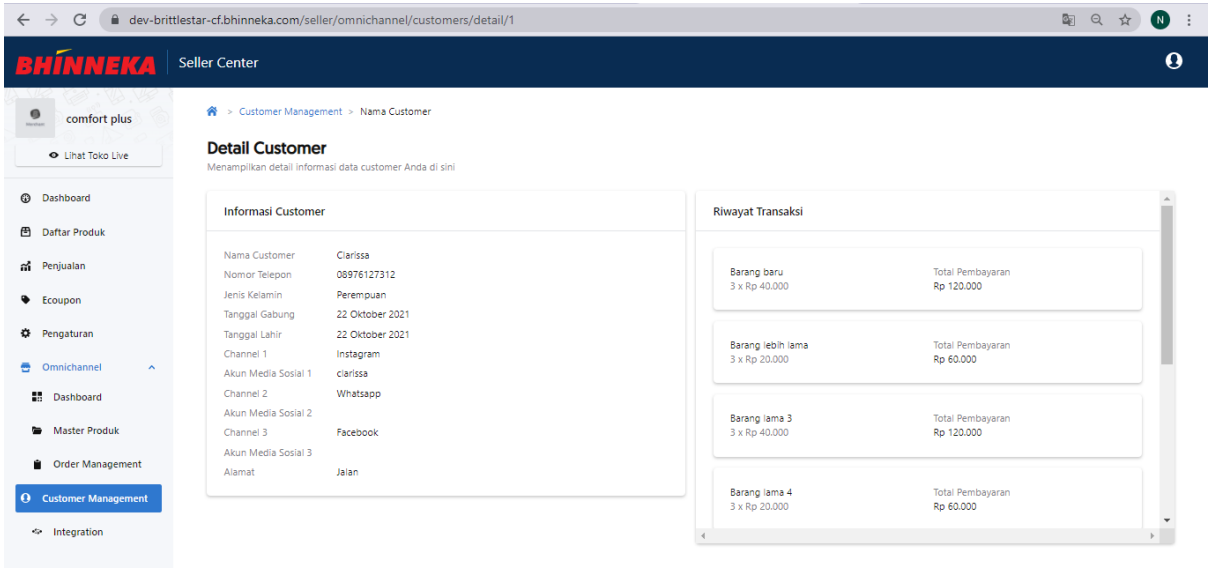
Lampiran A. Hasil penerapan web service API Customer Management crino.id

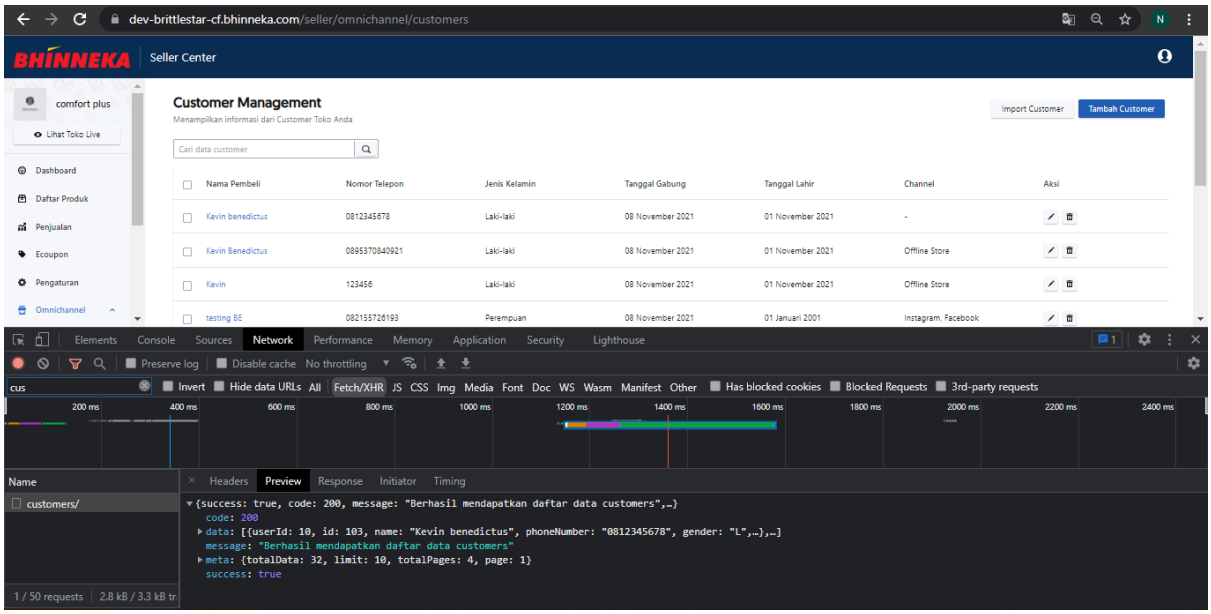
Berikut merupakan hasil penerapan web service API *Customer Management* crino.id pada *dashboard seller center* bhinneka.com.






dev-brittlestar-cf.bhinneka.com/seller/omnichannel/customers/edit?id=1
BHINNEKA
Seller Center
comfort plus
Lihat Toko Live
Dashboard
Daftar Produk
Penjualan
Ecoupon
Pengaturan
Omnichannel
Dashboard
Master Produk
Order Management
Customer Management
Integration
Customer Management > Edit customers
Edit Data Customer
Edit data customer anda disini
Edit Data Customers
Nama customer *
Clarissa
Nomor Telepon *
08976127312
Jenis Kelamin
Pilih Jenis Kelamin
Tanggal Lahir *
Masukkan Tanggal Lahir Customer
Channel*
Tambah Channel
Alamat Customer
P
Batal
Simpan


dev-brittlestar-cf.bhinneka.com/seller/omnichannel/customers/detail/1
BHINNEKA
Seller Center
comfort plus
Lihat Toko Live
Dashboard
Daftar Produk
Penjualan
Ecoupon
Pengaturan
Omnichannel
Dashboard
Master Produk
Order Management
Customer Management
Integration
Customer Management > Nama Customer
Detail Customer
Menampilkan detail informasi data customer Anda di sini
Informasi Customer
Nama Customer Clarissa
Nomor Telepon 08976127312
Jenis Kelamin Perempuan
Tanggal Gabung 22 Oktober 2021
Tanggal Lahir 22 Oktober 2021
Channel 1 Instagram
Akun Media Sosial 1 clarissa
Channel 2 Whatsapp
Akun Media Sosial 2
Channel 3 Facebook
Akun Media Sosial 3
Alamat Jalan
Riwayat Transaksi
Barang baru
3 x Rp 40.000
Total Pembayaran
Rp 120.000
Barang lebih lama
3 x Rp 20.000
Total Pembayaran
Rp 60.000
Barang lama 3
3 x Rp 40.000
Total Pembayaran
Rp 120.000
Barang lama 4
3 x Rp 20.000
Total Pembayaran
Rp 60.000


dev-brittlestar-cf.bhinneka.com/seller/omnichannel/customers
BHINNEKA
Seller Center
Customer Management
Menampilkan informasi dari Customer Toko Anda
Import Customer
Tambah Customer
Cari data customer
Nama Pembeli
Nomor Telepon
Jenis Kelamin
Tanggal Gabung
Tanggal Lahir
Channel
Aksi
Kevin benedictus 0812345678 Laki-laki 08 November 2021 01 November 2021 -
Kevin Benedictus 0895370840921 Laki-laki 08 November 2021 01 November 2021 Offline Store
Kevin 123456 Laki-laki 08 November 2021 01 November 2021 Offline Store
testing BE 082155726193 Perempuan 08 November 2021 01 Januari 2001 Instagram, Facebook
Elements Console Sources Network Performance Memory Application Security Lighthouse
Preserve log Disable cache No throttling
cus
Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies Blocked Requests 3rd-party requests
200 ms 400 ms 600 ms 800 ms 1000 ms 1200 ms 1400 ms 1600 ms 1800 ms 2000 ms 2200 ms 2400 ms
Name
customers/
Headers Preview Response Initiator Timing
{success: true, code: 200, message: "Berhasil mendapatkan daftar data customers",…}
code: 200
data: [{userId: 10, id: 103, name: "Kevin benedictus", phoneNumber: "0812345678", gender: "L",…},…]
message: "Berhasil mendapatkan daftar data customers"
meta: {totalData: 32, limit: 10, totalPages: 4, page: 1}
success: true
1 / 50 requests 2.8 kB / 3.3 kB tr