TUGAS AKHIR - IS184853

PENGEMBANGAN BACKEND UNTUK PELANGGAN PADA MODUL AJAK TEMAN DI SISTEM PEMASARAN UMKM

BACKEND DEVELOPMENT FOR CUSTOMERS ON MSME'S REFERRAL MARKETING SYSTEM

M. IHSAN FARABI F
05211640000097

Dosen Pembimbing
Rully Agus Hendrawan, S.Kom, M.Eng

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR - IS184853

PENGEMBANGAN BACKEND UNTUK PELANGGAN PADA MODUL AJAK TEMAN DI SISTEM PEMASARAN UMKM

M. IHSAN FARABI F
05211640000097

Dosen Pembimbing
Rully Agus Hendrawan, S.Kom, M.Eng

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman Ini Sengaja Dikosongkan)

UNDERGRADUATE THESIS - IS184853

BACKEND DEVELOPMENT FOR CUSTOMERS ON MSME'S REFERRAL MARKETING SYSTEM

M. IHSAN FARABI F
05211640000097

Supervisor
Rully Agus Hendrawan, S.Kom, M.Eng

INFORMATION SYSTEM DEPARTMENT
Information Technology and Communication Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2020

(Halaman Ini Sengaja Dikosongkan)

LEMBAR PENGESAHAN

PENGEMBANGAN BACKEND UNTUK PELANGGAN PADA MODUL AJAK TEMAN DI SISTEM PEMASARAN UMKM

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer (S.Kom)
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas (ELECTICS)
Institut Teknologi Sepuluh Nopember

Oleh

M. Ihsan Farabi F
05211640000097

Surabaya, 18 Agustus 2020

Kepala Departemen Sistem Informasi

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS DEPARTEMEN SISTEM INFORMASI

Dr. Mudjahidin, ST., MT.
NIP. 197010102003121001

(Halaman Ini Sengaja Dikosongkan)

LEMBAR PERSETUJUAN

PENGEMBANGAN BACKEND UNTUK PELANGGAN PADA MODUL AJAK TEMAN DI SISTEM PEMASARAN UMKM

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

M. IHSAN FARABI F
NRP. 05211640000097

Disetujui Tim Penguji : 18 Juni 2020
Periode Wisuda : September 2020

Rully Agus Hendrawan, S.Kom, M.Eng **(Pembimbing I)**

Erma Suryani, S.T., M.T., Ph.D **(Penguji I)**

Andre Parvian Aristio, S.Kom., M.Sc **(Penguji II)**

(Halaman Ini Sengaja Dikosongkan)

PENGEMBANGAN BACKEND UNTUK PELANGGAN PADA MODUL AJAK TEMAN DI SISTEM PEMASARAN UMKM

Nama Mahasiswa	**: M. Ihsan Farabi F**
NRP	**: 05211640000097**
Departemen	**: Sistem Informasi FTIK-ITS**
Pembimbing I	**: Rully Agus Hendrawan, S.Kom, M.Eng**

ABSTRAK

UMKM dalam menjalankan usahanya membutuhkan cara untuk mempromosikan produknya dengan cepat dan dalam jangkauan yang luas dengan memanfaatkan sumber daya secara efisien. UMKM selama ini masih menggunakan cara manual untuk mencatat pelanggan baru dan pelanggan loyal. Strategi pemasaran UMKM yang masih menggunakan referral manual seperti melakukan sosialisasi kepada kerabat menyebabkan jangkauan penyebaran informasi tersebar pada lingkup yang kecil. Terdapat peluang untuk mengotomasi pekerjaan ini secara efisien menggunakan sistem.

Arsitektur yang sedang populer di cloud saat ini adalah dengan memisahkan frontend dan backend. Pada tugas akhir ini fokus pada pengembangan backend dimana layanan bisa dikonsumsi oleh frontend. Dengan beragam variasi frontend (web, mobile), proses bisnis dikembangkan terpisah pada backend dibuat supaya pengembangan kode lebih produktif. Limitasi dari tugas akhir ini adalah pengembangan aplikasi hanya berfokus pada backend yang akan menghasilkan web service dan pengembangan aplikasi hanya sampai pada tahap pengujian sistem.

Pengembangan backend menggunakan SDLC Waterfall yang dimulai dari analisis kebutuhan, perancangan, implementasi dan pengujian. Bahasa pemrograman yang dipakai adalah C# dengan menggunakan ASP.NET Core sebagai kerangka kerja pengembangan aplikasi dan Swagger sebagai tools untuk mendokumentasikan web API.

Luaran yang dihasilkan dari tugas akhir ini adalah aplikasi backend dan dokumentasinya. Aplikasi backend yang dikembangkan dipecah menjadi beberapa servis. Aplikasi backend dikembangkan menggunakan arsitektur REST yang menghasilkan Web API yang dapat dikonsumsi oleh berbagai macam frontend. Pengujian yang dilakukan adalah pengujian fungsional, pengujian performa dan pengujian keamanan. Semua endpoint yang dibuat lolos uji fungsional, performa, dan keamanan. Penelitian ini hanya fokus pada modul ajak teman sehingga dibutuhkan pengembangan modul lainnya dalam penelitian selanjutnya agar aplikasi mempunyai fitur yang lengkap.

Tugas akhir ini dapat digunakan sebagai acuan untuk mengembangkan perangkat lunak dengan arsitektur microservice. Aplikasi backend siap digunakan oleh aplikasi frontend yang dikembangkan oleh Kemal (2020), Agung (2020), dan Akram (2020) agar menjadi aplikasi pemasaran yang dapat dimanfaatkan oleh UMKM. Aplikasi backend telah memenuhi kebutuhan fungsional berdasarkan rancangan antar muka yang dirancang untuk melakukan otomasi terhadap pencatatan pelanggan baru dan pelanggan loyal UMKM. Oleh karena itu dengan digunakannya aplikasi frontend yang didukug oleh aplikasi backend pada tugas akhir ini, UMKM akan dapat melakukan pemasaran dan pencatatan pelanggan secara efisien.

Kata Kunci: UMKM, Referral Marketing, Microservice Architecture, Back-end.

BACKEND DEVELOPMENT FOR CUSTOMERS ON MSME'S REFERRAL MARKETING SYSTEM

Name	**: M. Ihsan Farabi F**
NRP	**: 05211640000097**
Department	**: Information System FTIK-ITS**
Supervisor	**: Rully Agus Hendrawan, S.Kom, M.Eng**

ABSTRACT

MSME which are selling their product using technology such as social media and e-commerce have proven to have better returns in terms of revenue, business opportunities, innovation and competitiveness compared to MSME which still selling conventionally. Referral marketing is a marketing strategy that attracts customers by giving gifts to buyers and relatives who refer offers. Referral marketing has advantages in terms of credibility because the recommendations given by close friends or family can be more trusted than the words in paid advertising.

MSME still used manual methods to record new customers and loyal customers. MSME still used manual referrals such as socialization to their family that are limited to the surrounding environment as their marketing strategy. This causes the spread of information about the product and brand is very limited. There is an opportunity to automate this work efficiently using the system.

Architecture which is popular in the cloud currently is to separate the frontend and backend. In this final project, the focus is on developing a backend where services can be consumed by the frontend. With a variety of frontend (web, mobile), business processes developed at the backend are made separately so that code development is more productive.

Backend development using Waterfall SDLC starts from the requirement analysis, design, implementation, and testing. The

programming language used is C # by using ASP.NET Core as an application development framework and Swagger as a tool to document web APIs.

The output generated from this thesis is the backend application and its documentation. The tests conducted are functional testing, performance testing and security testing.

Keywords: MSME, Referral Marketing, Microservice Architecture, Back-end.

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: M. Ihsan Farabi F
NRP	: 05211640000097
Tempat/Tanggal lahir	: Madiun/24 Desember 1998
Fakultas/Departemen	: FTEIC/Sistem Informasi
Nomor Telp/Hp/email	: 089523509361/ihsan.faraabi@gmail.com

Dengan ini menyatakan dengan sesungguhnya bahwa penelitian/makalah/tugas akhir saya yang berjudul

PENGEMBANGAN BACKEND UNTUK PELANGGAN PADA MODUL AJAK TEMAN DI SISTEM PEMASARAN UMKM

Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain.

Apabila dikemudian hari ditemukan seluruh atau sebagian penelitian/makalah/tugas akhir tersebut terdapat indikasi plagiarisme, maka saya bersedia menerima sanksi sesuai peraturan dan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sesungguhnya dan untuk dipergunakan sebagaimana mestinya.

Surabaya, 13 Agustus 2020

METERAI TEMPEL
6B828AHF280517869
6000
ENAM RIBU RUPIAH

M. IHSAN FARABI F
NRP.05211640000097

KATA PENGANTAR

Dengan mengucapkan rasa syukur kepada Tuhan Yang Maha Pengasih dan Maha Penyayang atas izin-Nya penulis dapat menyelesaikan buku yang sederhana ini dengan judul Pengembangan Backend untuk Pelanggan pada Modul Ajak Teman di Sistem Pemasaran UMKM. Dalam penyelesaian Tugas Akhir ini, penulis diiringi oleh pihak-pihak yang selalu memberi dukungan, saran, dan doa sehingga penelitian berlangsung dengan lancar. Secara khusus penulis mengucapkan terima kasih dari lubuk hati terdalam kepada:

1. Tuhan, yang selalu menemani dan membimbing penulis dalam segala aspek kehidupan.
2. Bapak Rully Agus Hendrawan, S.Kom, M.Eng selaku dosen pembimbing yang telah mencurahkan segenap tenaga, waktu dan pikiran dalam penelitian ini, serta memberikan motivasi yang membangun.
3. Ibu Erma Suryani, S.T., M.T., Ph.D. dan Bapak Andre Parvian Aristio, S.Kom, M.Sc. selaku dosen penguji yang telah memberikan kritik dan saran yang membuat kualitas penelitian ini lebih baik lagi.
4. Segenap dosen dan karyawan Departemen Sistem Informasi.
5. Orang tua penulis, yang tiada hentinya mendoakan dan memberikan dukungan kepada penulis.
6. Dan lainnya
7. Pihak lainnya yang berkontribusi dalam tugas akhir yang belum dapat penulis sebutkan satu per satu.

Penyusunan tugas akhir ini masih jauh dari kata sempurna, untuk itu penulis menerima segala kritik dan saran yang membangun sebagai upaya menjadi lebih baik lagi ke depannya. Semoga buku tugas akhir ini dapat memberikan manfaat untuk pembaca.

Surabaya, Juni 2020

M. IHSAN FARABI F

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

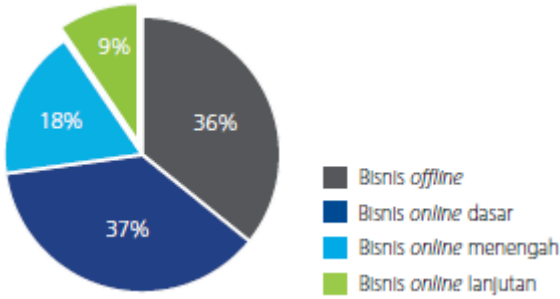
Bab ini akan menjelaskan tentang pendahuluan pengerjaan tugas akhir yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, dan manfaat yang akan diperoleh dari penelitian tugas akhir ini.

3.1 Latar Belakang

Usaha Mikro Kecil Menengah (UMKM) memegang peranan yang sangat penting bagi perekonomian di Indonesia. Hal ini dapat dibuktikan ketika Indonesia sedang mengalami krisis ekonomi pada tahun 1997/1998. Ketika perusahaan-perusahaan besar banyak yang gagal bertahan karena kehilangan kepercayaan pasar dan masyarakat, UMKM-lah sebagai sektor ekonomi dengan skala lokal, memakai sumber daya lokal dan proses produksi yang sederhana yang hasil produksinya dijual secara lokal yang mampu bertahan pada masa krisis karena tidak bergantung pada pasar modal besar. Menurut data dari Kementerian Koperasi dan UKM pada tahun 2015 menunjukkan bahwa jumlah perusahaan besar hanya 0,01 persen dari jumlah total unit usaha yang ada, sedangkan sisanya adalah UMKM dengan rincian 98,77 persen usaha mikro, 1,13 persen usaha kecil dan 0,09 persen usaha menegah. UMKM dapat menyerap tenaga kerja dalam jumlah besar dan merata karena UMKM merupakan sektor yang mendominasi usaha masyarakat dan juga tersebar di pelosok Indonesia. Dengan demikian UMKM dapat meningkatkan pendapatan masyarakat dan mengurangi angka kemiskinan [1].

Meskipun mendominasi dari jumlah unit usaha serta mempunyai berbagai macam ide dan produk yang ditawarkan, UMKM mempunyai kelemahan-kelemahan terutama dalam kemampuan pemasaran. Jika UMKM masih menerapkan strategi pemasaran manual tanpa mengikuti perkembangan teknologi dan informasi, dapat dipastikan bahwa daya saing UMKM akan tertinggal jauh dari para pelaku bisnis lainnya. Kecepatan dalam menjaring konsumen membutuhkan bantuan teknologi informasi yang sederhana misalnya dengan media

sosial *Facebook*, *Twitter*, *Instagram*, *WhatsApp*, *Telegram*. Media sosial mempunyai fitur yang dapat digunakan sebagai media promosi dan sosialisasi seperti fitur unggah foto, unggah video, menulis status di *wall*, dll [2].



Gambar 3.1 Distribusi bisnis berdasarkan tingkat keterlibatan secara digital

UMKM yang telah memanfaatkan teknologi seperti berjualan menggunakan media sosial dan *e-commerce* terbukti memperoleh keuntungan yang lebih baik dalam hal pendapatan, peluang bisnis, inovasi dan daya saing jika dibandingkan dengan UMKM yang masih berjualan secara konvensional. Meskipun teknologi informasi sudah cukup berkembang di Indonesia, faktanya sepertiga UMKM di Indonesia masih *offline* atau masih berjualan secara konvensional (36%) dan sepertiga lainnya hanya memiliki kemampuan berjualan daring tingkat dasar. Sedangkan yang mempunyai kemampuan menengah dan lanjutan masing-masing baru 18% dan 9% [3].

Pemasaran viral adalah cara pemasaran yang memanfaatkan media sosial yang ada dengan menarik calon konsumen untuk menyebarkan informasi produk kepada teman-teman dan kerabatnya. Penekanan kata viral adalah untuk menggambarkan bagaimana pesan-pesan yang disampaikan melalui internet menyebar dengan cepat seperti virus, namun dalam konotasi positif. Pemasaran viral dianalogikan dengan pemasaran dari mulut ke mulut atau *word-of-mouth* atau penyebarannya dilakukan dari satu pihak kepada pihak lainnya. Dalam pemasaran viral diharapkan terjadi efek berantai karena dari

satu orang menyebarkan kepada puluhan bahkan ribuan pengguna internet. Pemasaran viral mempunyai kelebihan dalam hal kredibilitas karena rekomendasi yang diberikan oleh teman dekat atau keluarganya dapat lebih dipercaya daripada kata-kata pada iklan berbayar. Walaupun begitu, sampai sekarang masih sangat sulit untuk mengukur seberapa berpengaruh rekomendasi dari seseorang ke temannya untuk membeli produk yang direkomendasikan [4].

Berdasarkan permasalahan tersebut, UMKM membutuhkan cara agar produknya dapat dipromosikan secara cepat dan luas tanpa memerlukan banyak sumber daya yang dibutuhkan. Maka salah satu caranya adalah dengan membuat *platform* untuk pemasaran viral bagi UMKM. *Platform* ini akan dikembangkan menggunakan kerangka kerja ASP.NET Core yang hanya berfokus pada sisi *backend* saja. Untuk sisi *frontend* dan integrasi akan dilakukan pada penelitian selanjutnya.

3.2 Rumusan Masalah

UMKM dalam menjalankan usahanya membutuhkan cara untuk mempromosikan produknya dengan cepat dan dalam jangkauan yang luas dengan memanfaatkan sumber daya secara efisien. UMKM selama ini masih menggunakan cara manual untuk mencatat pelanggan baru dan pelanggan loyal. Strategi pemasaran UMKM yang masih menggunakan referral manual seperti melakukan sosialisasi kepada kerabat menyebabkan jangkauan penyebaran informasi tersebar pada lingkup yang kecil. Terdapat peluang untuk mengotomasi pekerjaan ini secara efisien menggunakan sistem.

Arsitektur yang sedang populer di cloud saat ini adalah dengan memisahkan frontend dan backend. Aplikasi yang dikembangkan menggunakan arsitektur ini dapat berjalan pada berbagai sistem operasi seperti Windows, Mac OS, Android, dan iOS. Dengan tersedianya aplikasi pada berbagai macam platform, user dapat menggunakan aplikasi pada pilihan perangkat yang lebih luas. Oleh karena itu user yang menggunakan aplikasi menjadi lebih banyak.

Pada tugas akhir ini fokus pada pengembangan backend dimana layanan bisa dikonsumsi oleh frontend. Proses bisnis dikembangkan terpisah pada backend dibuat supaya pengembangan kode lebih produktif. Selain itu, memisahkan backend juga membuka peluang untuk memanfaatkan tools HeadlessCRM yang mempunyai fitur manajemen client sehingga aplikasi yang dikembangkan hanya fokus pada proses bisnis utama.

3.3 Batasan Permasalahan

Sesuai dengan deskripsi permasalahan yang telah dijelaskan diatas, adapun batasan permasalahan dari penyelesaian tugas akhir ini adalah sebagai berikut:

1. Pengembangan aplikasi hanya berfokus pada *backend* yang akan menghasilkan *web service*.
2. Penelitian tugas akhir ini hanya sampai pada tahap pengujian sistem.

3.4 Tujuan

Tujuan utama dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Merancang model data dan spesifikasi API endpoint berdasarkan pendekatan centralized user view dari antar muka tugas akhir Kemal (2020), Agung (2020), dan Akram (2020).
2. Mengembangkan *web service* menggunakan kerangka kerja ASP.NET Core untuk digunakan oleh penelitian tugas akhir selanjutnya.
3. Melakukan pengujian fungsional, performa, dan keamanan aplikasi menggunakan teknik *black box testing*.
4. Membuat dokumentasi *domain model* untuk memudahkan perawatan serta dapat menjadi referensi pengembang berikutnya.

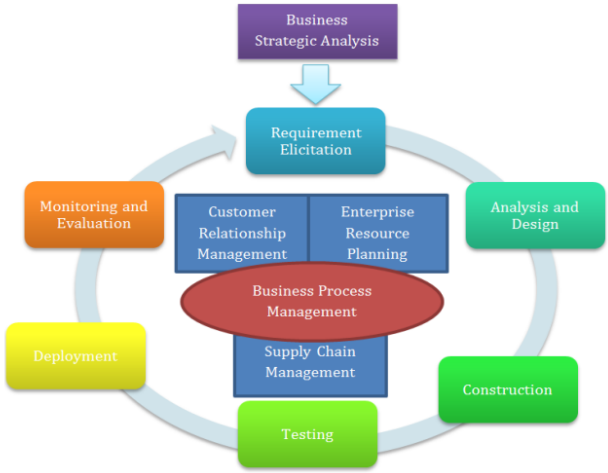
3.5 Manfaat

Manfaat yang diharapkan dari Tugas Akhir ini adalah:

1. Sebagai acuan untuk pengembangan *backend* aplikasi *marketing platform* untuk UMKM menggunakan kerangka kerja ASP.NET Core.
2. Sebagai acuan untuk meningkatkan keuntungan dalam hal pendapatan, peluang bisnis, inovasi dan daya saing bagi UMKM.

3.6 Relevansi

Laboratorium Sistem Enterprise (SE) Jurusan Sistem Informasi ITS memliki empat topik utama yaitu *customer relationship management* (CRM), *enterprise resource planning* (ERP), *supply chain management* (SCM) dan *business process management* (BPM) seperti yang terdapat pada Gambar 1.2. Dalam tugas akhir yang dikerjakan oleh penulis mengambil *customer relationship management* (CRM) sebagai topik utama. Mata kuliah yang berkaitan dengan CRM adalah Manajemen Hubungan Pelanggan.



Gambar 3.2 Kerangka kerja riset laboraturium sistem enterprise

BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka terdiri dari landasan-landasan yang akan digunakan dalam penelitian tugas akhir ini, mencakup penelitian-penelitian sebelumnya, kajian pustaka, dan metode yang digunakan selama pengerjaan.

4.1 Penelitian Sebelumnya

Terdapat beberapa penelitian yang memiliki topik yang hampir serupa dengan penelitian ini, diantaranya akan dijelaskan pada Tabel 4.2-**Error! Reference source not found.**.

Tabel 4.1 Literatur 1

Judul	Viral Marketing: Motivations to Forward Online Content
Nama, Tahun	Jason Y.C. Ho, Melanie Dempsey
Gambaran umum penelitian	Penelitian ini menjelaskan tentang salah satu faktor kritis yang mempengaruhi viral marketing. Penelitian ini mengidentifikasi empat motivasi yang membuat seseorang akan melakukan menyebarkan *online content*, yaitu (1) kebutuhan untuk menjadi bagian dari sebuah kelompok, (2) kebutuhan untuk menjadi sebuah individu, (3) kebutuhan untuk peduli kepada orang lain, (4) kebutuhan untuk mengembangkan diri sendiri. Metode yang dilakukan adalah survey kepada orang dewasa yang masih muda, yang digunakan untuk mengetahui apakah sesorang akan menyebarkan *online content* dilihat dari empat motivasi yang disebutkan sebelumnya. Kesimpulan penelitian ini adalah seseorang dengan karakter cenderung individual dan/atau cenderung peduli terhadap orang lain yang lebih banyak menyebarkan *online content* [5].
Keterkaitan penelitian	Tugas akhir ini dapat memanfaatkan kesimpulan yang diperoleh dari penelitian ini yaitu sesorang yang mempunyai karakter cenderung individual dan/atau cenderung peduli terhadap orang lain yang lebih banyak menyebarkan *online content*. Sehingga pengembangan aplikasi akan lebih fokus kea rah content yang sesuai pada karakter sesorang yang lebih individual dan/atau peduli terhadap orang lain.

Tabel 4.2 Literatur 2

Judul	Word of Mouth: Understanding and Managing Referral Marketing
Nama, Tahun	Francis A. Buttle
Gambaran umum penelitian	Penelitian ini membahas tentang penelitian-penelitian yang telah ada sebelumnya tentang *word of mouth* (WOM). Peneliti membuat sebuah model kontingensi dan mengidentifikasi *gaps* yang ada pada ilmu pengetahuan kita saat ini. Hasil akhir dari penelitian ini adalah sebuah model WOM. WOM mempunyai dua *environment*, yaitu *intrapersonal* dan *extrapersonal*. *Intrapersonal environment* mempunyai sebuah variable yaitu variable ekspektasi yang mempengaruhi keputusan yang diambil sesorang dari dalam dirinya. Sedangkan *extrapersonal environment* mempunyai empat variable utama yaitu *culture, social networks, incentive, business climate*. Penelitian ini juga menyimpulkan bahwa WOM lebih baik diterapkan untuk menarik pelanggan yang ingin membeli jasa, bukan barang. Hal ini karena jasa tidak terlihat dan calon konsumen membutuhkan testimoni dari orang-orang yang dipercayainya [6].
Keterkaitan penelitian	Penelitian ini memberikan gambaran tentang apa itu *word of mouth* dan hal-hal apa saja yang dapat mempengaruhi keberhasilan penggunaan metode *word of mouth* ini. Dengan mengetahui faktor-faktor yang mempengaruhi keberhasilan *word of mouth*, tugas akhir dapat menggunakan penelitian ini sebagai acuan untuk mendesain program referral apa saja yang bisa digunakan.

Tabel 4.3 Literatur 3

Judul	Black Box and White Box Testing Techniques – A Literature Review
Nama, Tahun	Srinivas Nidhra, Jagruthi Dondeti, 2012
Gambaran umum penelitian	Penelitian ini melakukan studi literatur terhadap semua teknik testing yang berhubungan dengan *Black box* dan *White box testing* secara bersama-sama. Penelitian ini menyediakan *test case* dan *test data* untuk *Black box* dan *White box testing*. Penelitian ini bertujuan untuk

	menjabarkan secara jelas tentang perbedaan teknik testing dengan menggunakan *case situation* dan kelebihan-kelebihan yang ditawarkan oleh masing-masing Teknik testing [7].
Keterkaitan penelitian	Pada tahap terakhir pada pembuatan aplikasi pada tugas akhir ini akan dilakukan black box testing untuk menguji kode yang telah dibuat. Keterkaitan dengan penelitian ini adalah penulis dapat mengetahui dimana posisi black box testing ini dan apa saja kelebihannya pada situasi tertentu. Selain itu penulis juga dapat mengetahui jenis testing yang lain yang mungkin dapat digunakan pada penelitian selanjutnya.

Tabel 4.4 Literatur 4

Judul	Rancang Bangun Back-End "SIAP": Sistem Informasi Aspirasi Dan Pengaduan Masyarakat Berbasis Web Dengan Menggunakan Metode Microservice Springboot
Nama, Tahun	Dedi Puji Jayanto, 2017
Gambaran umum penelitian	Penelitian ini membuat aplikasi SIAP (Sistem Informasi Aspirasi dan Pengaduan Masyarakat). Tujuan dari aplikasi ini adalah membuat aplikasi yang bisa dimanfaatkan oleh semua kabupaten/kota di Indonesia. Aplikasi yang dikembangkan memungkinkan pemerintahan kabupaten/ kota se-Indonesia untuk bisa dengan mudah mengimplementasikan layanan Aspirasi dan Pengaduan Masyarakat berbasis web serta dengan biaya investasi yang rendah dan memanfaatkan sumber daya manusia yang ada karena karena infrastruktur aplikasi, keamanan, penanganan error, update program dilakukan oleh pihak tim peneliti selaku penyedia layanan e-government as a Service SIAP. Metode yang digunakan di Tugas Akhir ini adalah menggunakan arsitektur Microservices Springboot. Penggunaan microservice adalah dengan membagi-bagi fungsionalitas aplikasi menjadi banyak bagian atau banyak layanan yang kecil (micro) yang saling berhubungan satu sama lain sehingga menjadi satu kesatuan bisnis proses aplikasi [8].
Keterkaitan penelitian	Keterkaitan penelitian ini dengan tugas akhir adalah menggunakan arsitektur yang sama yaitu arsitektur *microservice* untuk mengembangkan aplikasi. Penelitian ini dapat menjadi acuan untuk mengembangkan aplikasi dan

	untuk menyusun buku tugas akhir karena sudah terdokumentasi dengan lengkap.

4.2 Referral Marketing

Referral marketing adalah salah satu teknik pemasaran yang sering digunakan modern ini. Teknik pemasaran ini menggunakan komunikasi yang dilakukan antara penerima dan komunikator yang oleh penerima dianggap sebagai komunikasi non-komersial terhadap merek, produk atau layanan. Penerima informasi dan komunikator diibaratkan dengan dua orang yang mempunyai hubungan dekat yang sedang berkomunikasi dalam kehidupan nyata. Komunikator memberikan informasi pengalamannya terhadap produk atau jasa yang mereka gunakan. Penerima informasi akan lebih mendengarkan sesuatu yang disampaikan oleh orang terdekatnya daripada iklan berbayar [6].

Program referral marketing umumnya memberikan hadiah kepada pelanggan yang membeli produknya dan orang yang mereferensikan produknya kepada pelanggan tersebut. Hal ini dapat meningkatkan keinginan seseorang untuk menyebarkan informasi terhadap produk-produk yang ditawarkan oleh perusahaan. Orang yang mereferensikan produk akan mendapat hadiah setelah seseorang membeli produk dengan menggunakan kode referral. Selain memberikan hadiah untuk kedua pihak, terdapat penawaran yang hanya memberikan hadiah kepada salah satu pihak atau tidak memberikan hadiah untuk kedua pihak. Penawaran ini biasanya ditujukan untuk sosialisasi produk agar informasi produk dan merek tersebar secara luas dan cepat. Biasanya penawaran ini hanya meminta calon pelanggan untuk membuka halaman *landing page* saja untuk mengenalkan produk dan merek.

4.3 Microservices

Microservices adalah suatu arsitektur yang sering dibicarakan dalam pengembangan aplikasi modern ini. Sebelum mengetahui microservices, kita harus mengetahui salah satu arsitektur yang paling banyak digunakan saat ini yaitu arsitektur monolitik.

Kata monolitik digunakan untuk mendeskripsikan sebuah aplikasi yang terdiri dari satu bagian. Pengembang aplikasi tradisional Sebagian besar menggunakan arsitektur monolitik untuk mengembangkan aplikasi. Arsitektur monolitik dirancang untuk berjalan hanya pada satu objek komputasi. Proses pada arsitektur monolitik dapat didistribusikan pada banyak CPU, tetapi semuanya menggunakan sistem operasi dan perangkat keras yang sama. Jika sistem melebihi kapasitas, maka sistem memerlukan duplikasi. Hal ini berdampak pada kurangnya fleksibilitas dalam mengembangkan aplikasi. Misalnya, jumlah user yang dapat ditangani oleh suatu sistem melebihi batas maka arsitektur monolitik tidak bisa melakukan *horizontal scaling* tetapi hanya bisa melakukan *vertical scaling*.

Konsep dari arsitektur microservices adalah penguraian aplikasi menjadi bagian-bagian kecil yang spesifik. Masing-masing *service* independen terhadap service lainnya. Hal tersebut berlaku pada keseluruhan siklus hidup aplikasi, mulai dari pengembangan, penyebaran dan perawatan aplikasi. Arsitektur microservices menggunakan protokol komunikasi yang ringan dan tidak ketat. Setiap *service* tidak berkomunikasi secara langsung dengan *service* lainnya, melainkan menggunakan *interface* yang sudah terdefinisi. Jika salah satu komponen dari sistem mengalami kegagalan, maka kegagalan tersebut tidak akan mempengaruhi komponen lainnya. Kegagalan dapat diisolasi dan diselesaikan secara terpisah sementara sistem yang lain masih dapat bekerja [9].

4.4 ASP.NET Core

ASP.NET adalah sebuah kerangka kerja untuk mengembangkan aplikasi web pada *platform* .NET. ASP.NET Core adalah versi *open source* dari ASP.NET yang dapat berjalan pada macOS, Linux, dan Windows yang pertama kali diluncurkan pada tahun 2016. ASP.NET Core merupakan kerangka kerja hasil perancangan ulang dari kerangka kerja ASP.NET yang hanya dapat dijalankan pada perangkat windows. Salah satu alasan pemilihan kerangka kerja ASP.NET Core adalah karena performa yang ditawarkan lebih cepat dari kerangka kerja untuk

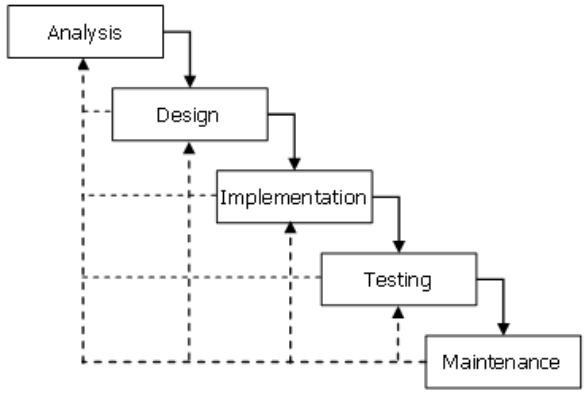
pengembangan web populer lainnya seperti Java Servlet dan Node.js. Bahasa C# digunakan untuk mengembangkan aplikasi pada penelitian ini dengan menggunakan Visual Studio 2019 sebagai *text editor* maupun *Integrated Development Environtment* (IDE) [10].

4.5 Swagger

Swagger merupakan perangkat lunak *open-source* yang dapat membantu pengembang aplikasi merancang, membangun dan mengkonsumsi RESTful *web services*. Swagger mempunyai beberapa *tools* yang terbagi menjadi beberapa fungsi yaitu pengembangan, interaksi dengan API, dan dokumentasi. Pada penelitian ini, swagger digunakan untuk melakukan dokumentasi otomatis dan testing pada API yang telah dibuat [11].

4.6 Model Rekayasa Perangkat Lunak Waterfall

Proses membangun perangkat lunak dan sistem informasi selalu menggunakan metodologi pengembangan yang berbeda. Sebuah metodologi dalam pengembangan perangkat lunak mengacu pada kerangka kerja yang digunakan untuk merancang, mengelola dan mengkontrol semua proses dalam pengembangan perangkat lunak. Salah satu metodologi pengembangan perangkat lunak yang popular adalah waterfall SDLC [12].



Gambar 4.1 Model waterfall SDLC

Waterfall SDLC model adalah sebuah proses pengembangan perangkat lunak yang dilakukan secara berurutan. Model waterfall mempunyai beberapa fase yang harus diselesaikan satu demi satu dan dapat pindah ke fase selanjutnya setelah fase sebelumnya sepenuhnya dilakukan. Maka dari itu, model waterfall bersifat rekursif yang berarti setiap fase dapat diulang-ulang hingga sempurna. Gambar 2.1 menunjukkan setiap fase pada model waterfall. Pada dasarnya, model waterfall terdiri dari lima fase dimulai dari analisis, desain, implementasi, pengujian dan perawatan. Pada pengembangan aplikasi marketing platform ini akan digunakan metode waterfall.

4.7 Black Box Testing

Pengujian perangkat lunak adalah salah satu teknik yang paling sering digunakan untuk melakukan verifikasi dan validasi terhadap kualitas sebuah perangkat lunak. Pengujian perangkat lunak adalah serangkaian prosedur dalam mengeksekusi program atau sistem untuk menemukan kesalahan. Pengujian perangkat lunak tradisional dapat dibagi menjadi *black box* dan *white box* testing.

Black box testing juga biasa disebut dengan *functional testing*, yaitu pengujian fungsional yang mendesain *test case* berdasarkan informasi dari spesifikasi. Dalam menjalankan *black box testing*, seorang penguji seharusnya tidak mempunyai

akses terhadap kode internal. *Black box* testing tidak menekankan kepada mekanisme internal dari sebuah sistem, melainkan hanya fokus pada *output* yang dihasilkan sebagai sebuah respons terhadap *input* yang dimasukkan serta kondisi pada saat eksekusi [7].

4.8 HTTP Method

Hypertext Transfer Protocol (HTTP) adalah protokol tingkat aplikasi untuk sistem informasi hypermedia yang terdistribusi dan kolaboratif. HTTP mempunyai beberapa *method* yang digunakan untuk mengambil sumber daya yang diinginkan. *Method* GET mengambil informasi apapun dalam bentuk entitas yang didefinisikan oleh URI yang diberikan. *Method* POST digunakan untuk mengirim data ke server. *Method* PUT digunakan untuk mengirim data ke server. Perbedaan dengan *method* POST adalah *method* PUT menggantikan sumber daya yang ada di server. *Method* DELETE digunakan untuk menghapus sumber daya yang ada pada server dengan menggunakan URI [13]. Tugas akhir ini menggunakan *method* GET, POST, PUT dan DELETE.

4.9 UML

Unified Modeling Language (UML) adalah sebuah bahasa yang digunakan untuk melakukan spesifikasi, visualisasi, konstruksi dan dokumentasi artefak dari sistem perangkat lunak. Dalam pengembangan perangkat lunak sebaiknya pengembang melihat dari berbagai macam sudut pandang untuk mendapatkan gambaran secara menyeluruh. Oleh karena itu UML menyediakan 11 jenis diagram untuk menggambarkan beberapa sudut pandang dari sistem perangkat lunak. Berikut adalah daftar 11 diagram tersebut [14].

1. Class Diagram
2. Internal Structure
3. Collaboration Diagram
4. Component Diagram
5. Use Case Diagram
6. State Machine Diagram

7. Activity Diagram
8. Sequence Diagram
9. Communication Diagram
10. Deployment Diagram
11. Package Diagram

Penggunaan diagram tidak harus semuanya dipakai dalam pembuatan model. Penggunaan diagram menyesuaikan kebutuhan dalam pengembangan. Pembuatan model dengan UML membutuhkan *tools* untuk membantu pembuatan model yang realistis. *Tools* menyediakan cara yang interaktif untuk melihat dan merubah model.

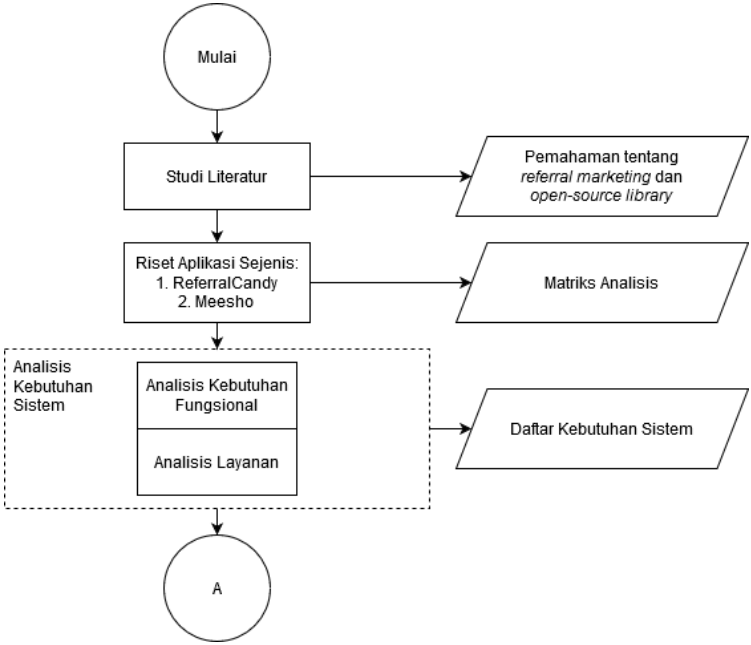
(Halaman ini sengaja dikosongkan)

BAB III
METODOLOGI

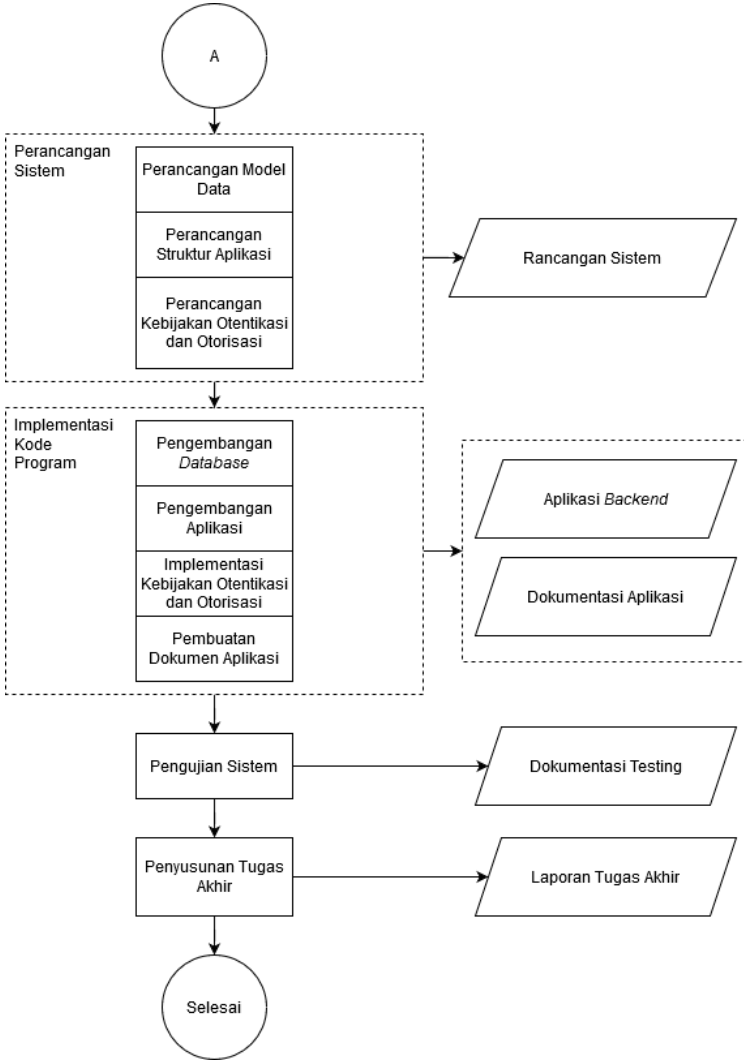
Pada bagian ini dijelaskan metodologi yang akan digunakan sebagai panduan untuk menyelesaikan penelitian tugas akhir ini.

5.1 Diagram Metodologi

Pada sub bab ini akan dijelaskan mengenai tahapan yang dilakukan dalam penelitian sesuai **Error! Reference source not found.Error! Reference source not found.** dan **Error! Reference source not found.**.



Gambar 5.1 Diagram metodologi 1



Gambar 5.2 Diagram metodologi 2

5.2 Uraian Metodologi

Metodologi penelitian dimulai dengan melakukan studi literatur yang terkait dengan Tugas Akhir, kemudian dilanjutkan dengan pengembangan aplikasi menggunakan metode SDLC waterfall dimulai dari analisis kebutuhan sistem, perancangan sistem, implementasi kode program, dan pengujian sistem.

5.2.1 Studi Literatur

Pada tahap ini akan dilakukan pengumpulan informasi yang berkaitan dengan topik pengembangan web. Beberapa sumber didapat dari buku, paten, dokumentasi *open-source library* dan penelitian sebelumnya yang sejenis. Tujuan dari tahap ini adalah untuk memperkuat pemahaman penulis dalam mengerjakan penelitian ini. Luaran dari penelitian ini adalah pemahaman penulis tentang pengembangan *backend* dan *open-source library* yang akan dipakai dalam penelitian ini.

5.2.2 Riset Aplikasi Sejenis

Pada tahap ini dilakukan komparasi terhadap aplikasi sejenis yang juga menggunakan *referral marketing*. Komparasi dilakukan untuk mengetahui fitur apa saja yang ada dan dapat diadopsi. Aplikasi yang menjadi referensi dalam pengembangan adalah ReferralCandy dan Meesho. Luaran dari aplikasi ini adalah matriks analisis.

5.2.3 Analisis Kebutuhan Sistem

Pada tahap ini akan dilakukan penggalian kebutuhan aplikasi. Penggalian kebutuhan dilakukan untuk mengetahui fitur apa saja yang dapat dipakai oleh pengguna aplikasi. Luaran dari tahap ini adalah daftar *functional requirement* dan *nonfunctional requirement*.

5.2.4 Perancangan Sistem

Pada tahap ini akan dilakukan perancangan sistem berdasarkan kebutuhan yang telah dianalisis sebelumnya. Perancangan sistem melibatkan perancangan data model, perancangan struktur aplikasi dan perancangan kebijakan otentikasi dan

otorisasi. Perancangan data model akan dilakukan *reverse engineering* pada data model aplikasi ReferralCandy dan Meesho. Dari hasil *reverse engineering* tersebut akan dimodifikasi sesuai dengan kebutuhan aplikasi. Perancangan sistem akan dirancang menggunakan UML.

Perancangan kebutuhan end point menggunakan arsitektur REST sebagai acuan dalam merancang *endpoint*. Hal pertama yang dilakukan adalah mendefinisikan semua objek model yang akan ditampilkan sebagai *resources*. Dari semua entitas yang telah dijabarkan sebelumnya, terdapat beberapa entitas yang digunakan sebagai *resource* yaitu Contents, Fans, Image, Merchants, Offers, Sales, Users. Setiap *resources* mempunyai *property* id sebagai tanda pengenal yang unik.

Setiap *resources* dihubungkan dengan satu URI. Desain URI pada penelitian ini menggunakan konsep hierarki agar lebih mudah dipahami. Terdapat dua pilihan representasi dari URI yaitu XML dan JSON, pada penelitian ini menggunakan JSON sebagai representasi dari URI yang buat. Setiap URI memiliki operasi yang diperbolehkan untuk digunakan. Karena REST menggunakan *method* HTTP sebagai protocol *interface*, maka *method* yang dipakai adalah sebagai berikut: GET, POST, PUT, DELETE.

5.2.5 Implementasi Kode Program

Pada tahap ini dilakukan implementasi kode program berdasarkan hasil rancangan yang telah dibuat. Implementasi kode program menggunakan kerangka kerja ASP.NET Core. Luaran dari tahap ini adalah web API yang didokumentasikan secara otomatis menggunakan swagger.

5.2.6 Pengujian Sistem

Pengujian dilakukan menggunakan metode *black box testing*. Metode ini tidak menekankan mekanisme internal dari aplikasi yang dibuat, melainkan menguji apakah output yang dihasilkan sesuai dengan apa yang diharapkan. Pengujian aplikasi juga memastikan apakah aplikasi yang dibuat sudah terbebas dari kesalahan program.

3.2.7 Penyusunan Tugas Akhir

Setelah proses pengembangan dan pengujian aplikasi dilakukan, pada tahap ini dilakukan dokumentasi terhadap semua yang telah dilakukan secara terperinci dalam bentuk buku.

BAB IV
ANALISA KEBUTUHAN DAN PERANCANGAN

Pada bab ini diuraikan proses analisis kebutuhan aplikasi dan perancangan yang diperlukan untuk mengembangkan aplikasi. Bab ini meliputi analisa kebutuhan aktor, analisa kebutuhan fungsional, perancangan model data, perancangan arsitektur aplikasi dan perancangan kebijakan dan keamanan aplikasi.

6.1 Analisa Kebutuhan Aktor

Analisa kebutuhan aktor dilakukan untuk mengetahui siapa saja aktor yang menggunakan aplikasi. Tabel 4.1 menunjukkan deskripsi dari aktor.

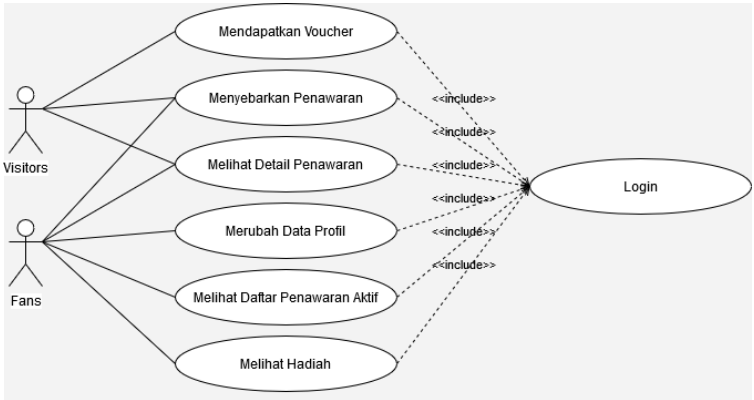
Tabel 6.1 Kebutuhan aktor

No	Aktor	Deskripsi
1	Fans	Merupakan orang yang telah terdaftar pada sistem dan memiki peran untuk membagikan penawaran melalui media daring milik mereka untuk memperoleh keuntungan tertentu
2	Visitors	Merupakan orang yang pertama kali menggunakan sistem untuk mendapatkan keuntungan tertentu melalui penawaran yang ia lihat pada media daring

6.2 Analisa Kebutuhan Fungsional

6.2.1 Analisa Kebutuhan Use Case

Kebutuhan use case dianalisa berdasarkan hasil Analisa kebutuhan aktor. Use case yang dibuat mendeskripsikan interaksi antara aktor dengan sistem. Gambar 4.1 menunjukkan diagram use case dari aplikasi.



Gambar 6.1 Diagram use case

6.2.1.1 Analisa Kebutuhan Use Case Fans

Pada tahap ini akan dilakukan identifikasi tabel kebutuhan use case dari aktor fans.

Tabel 6.2 Kebutuhan use case fans

UC ID	Nama	Deskripsi
UC-F1	Menyebarkan Penawaran	Fans menyebarkan penawaran yang telah dibuat oleh merchants. Penawaran bisa disebar melalui media sosial atau hanya menyalin tautan
UC-F2	Melihat Detail Penawaran	Fans dapat melihat detail penawaran yang dibuat oleh merchant
UC-F3	Merubah Data Profil	Fans dapat merubah data profil. Merubah kode referral,

		informasi bank dan informasi alamat pengiriman termasuk dalam use case ini
UC-F4	Melihat Penawaran Aktif	Fans dapat melihat daftar penawaran aktif yang dibuat oleh merchant
UC-F5	Melihat Hadiah	Fans dapat melihat hadiah yang didapatkan dari hasil penyebaran penawaran yang dilakukan

6.2.1.2 Analisa Kebutuhan Use Case Visitors

Pada tahap ini akan dilakukan identifikasi tabel kebutuhan use case dari aktor visitors.

Tabel 6.3 Kebutuhan use case visitors

UC ID	Nama	Deskripsi
UC-V1	Mendapatkan Voucher	Visitors dapat mendapatkan voucher untuk ditukarkan pada merchant saat melakukan pembelian
UC-V2	Menyebarkan Penawaran	Visitors dapat menyebarkan penawaran menggunakan referral miliknya sendiri

UC-V3	Melihat Detail Penawaran	Visitors dapat melihat detail penawaran yang dibagikan oleh fans

6.2.2 Analisa Kebutuhan Fungsional Fans

Pada tahap ini akan dilakukan identifikasi tabel fungsional dari aktor fans. Kebutuhan fungsional diidentifikasi berdasarkan hasil analisa kebutuhan use case. Berikut adalah tabel kebutuhan fungsional dari aktor fans.

Tabel 6.4 Kebutuhan fungsional fans

FR ID	UC ID	Nama
FR-F1	UC-F1	Sistem memungkinkan fans menyalin *link* penawaran
FR-F2	UC-F1	Sistem memungkinkan fans menyebarkan penawaran di Facebook
FR-F3	UC-F1	Sistem memungkinkan fans menyebarkan penawaran di Whatsapp
FR-F4	UC-F1	Sistem memungkinkan fans menyebarkan penawaran di Twitter
FR-F5	UC-F1	Sistem memungkinkan fans menyebarkan penawaran di Telegram
FR-F6	UC-F5	Sistem memungkinkan fans melihat semua hadiah yang didapat
FR-F7	UC-F5	Sistem memungkinkan fans melihat detail hadiah
FR-F8	-	Sistem memungkinkan fans login menggunakan nomor hp

FR-F9	UC-F3	Sistem memungkinkan fans merubah data profil
FR-F10	UC-F3	Sistem memungkinkan fans merubah kode referral
FR-F11	UC-F3	Sistem memungkinkan fans merubah informasi rekening
FR-F12	UC-F3	Sistem memungkinkan fans merubah informasi alamat pengiriman
FR-F13	UC-F4	Sistem memungkinkan fans melihat daftar penawaran aktif
FR-F14	UC-F4	Sistem memungkinkan fans melihat detail penawaran
FR-F15	UC-F5	Sistem memungkinkan fans melihat daftar hadiah
FR-F16	UC-F5	Sistem memungkinkan fans melihat detail hadiah

6.2.3 Analisa Kebutuhan Fungsional Visitors

Pada tahap ini akan dilakukan identifikasi tabel fungsional dari aktor visitors. Kebutuhan fungsional diidentifikasi berdasarkan hasil analisa kebutuhan use case. Berikut adalah tabel kebutuhan fungsional dari aktor visitor.

Tabel 6.5 Kebutuhan fungsional visitors

FR ID	UC ID	Nama
FR-V1	UC-V3	Sistem memungkinkan visitors melihat detail penawaran
FR-V2	UC-V1	Sistem memungkinkan visitors mendapatkan voucher
FR-V3	UC-V2	Sistem memungkinkan visitors membagikan penawaran

FR-V4	-	Sistem memungkinkan visitors login menggunakan nomor hp

6.2.4 Analisa Kebutuhan End Point

Berikut adalah tabel kebutuhan *endpoint* dari aplikasi.

Tabel 6.6 Kebutuhan end point

No	Layanan Web	HTTP Method
1	Login	POST
2	Logout	GET
3	Melihat semua penawaran aktif	GET
4	Mencari penawaran	GET
5	Melihat penawaran berdasarkan kategori	GET
6	Melihat penawaran terbaru	GET
7	Melihat detail penawaran	GET
8	Melihat daftar hadiah	GET
9	Melihat detail hadiah	GET
10	Merubah informasi profil	PUT
11	Merubah kode referral	PUT
12	Merubah informasi rekening	PUT
13	Membuat user baru	POST
14	Melakukan *random* kode voucher	GET
15	Membagikan penawaran	GET

6.3 Perancangan Model Data

Perancangan model data dimulai dari perancangan user view untuk melihat perspektif dari masing-masing role. Perancangan entitas dilakukan berdasarkan hasil perancangan user view. Setiap entitas kemudian dirancang hubungannya. Luaran dari perancangan model data adalah ERD.

6.3.1 Perancangan User View

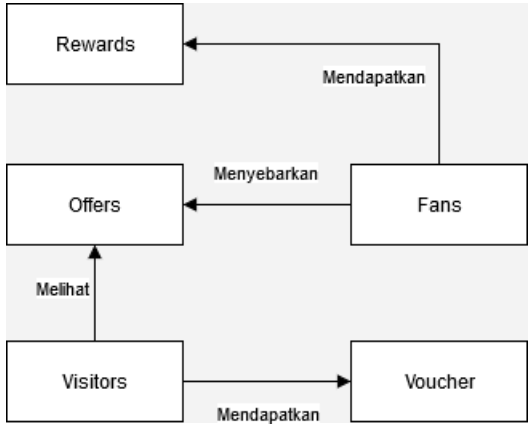
Tahap pertama dalam merancang model data adalah membuat user view aplikasi. Pembuatan user view dilakukan agar tidak ada user yang terlupa saat merancang model data. User view berisi data apa saja yang dilihat oleh user serta apa saja kegiatan yang dilakukan oleh user saat menggunakan aplikasi. Pendekatan *centralized* digunakan karena terdapat user yang mempunyai kebutuhan yang sama dengan user lainnya.

Tabel 6.7 Community view

Community View	Fans	Visitors
Detail penawaran	v	v
Halaman login	v	v
Menyebarkan penawaran	v	v
Detail hadiah	v	
Merubah informasi profil	v	
Merubah informasi rekening	v	
Merubah informasi pengiriman	v	
Merubah kode referral	v	
Mendapatkan voucher		v

6.3.2 Perancangan Model Data Konseptual

Model data konseptual dirancang berdasarkan hasil perancangan view model yang dilakukan sebelumnya. User view yang dimiliki oleh beberapa aktor kemudian diambil atributnya untuk disatukan menjadi satu entitas. Terdapat beberapa entitas yang teridentifikasi yaitu User, Offers, Reward, Voucher. Deskripsi entitas yang lebih lengkap dapat dilihat pada Lampiran B. Gambar 4.2 menunjukkan diagram rancangan model konseptual.



Gambar 6.2 ER Diagram model data konseptual

6.3.2.1 Identifikasi Entitas

Terdapat 5 entitas utama yang diidentifikasi dari user view yaitu entitas Reward, Offers, Fans, Visitors, dan Voucher. Pada Tabel 4.8 dijelaskan deskripsi setiap entitas berserta kejadian yang dimilikinya.

Tabel 6.8 Deskripsi entitas model data konseptual

Nama Entitas	Deskripsi	Alias	Kejadian
Rewards	Istilah umum yang mendeskripsikan semua hadiah	Hadiah	Setiap hadiah dimiliki oleh banyak fans

	yang dimiliki oleh fans		
Offers	Istilah umum yang mendeskripsikan semua penawaran yang bisa dibagikan uleh fans dan visitors	Penawaran	Setiap penawaran dibagikan oleh banyak fans dan pengunjung
Visitors	Istilah umum yang mendeskripsikan semua user yang mengambil voucher yang dibagikan oleh fans	Pengunjung	Setiap pengunjung memiliki banyak voucher
Voucher	Istilah umum yang mendeskripsikan voucher yang diperoleh visitors	Voucher	Setiap voucher dimiliki oleh satu pengunjung
Fans	Istilah umum yang mendeskripsikan user yang menyebarkan penawaran	Fans	Setiap fans menyebarkan banyak penawaran dan mendapatkan banyak hadiah.

6.3.2.2 Identifikasi Atribut

Setiap entitas mempunyai atribut yang digunakan untuk menyimpan data yang terjadi pada proses transaksi. Deskripsi lengkap atribut dapat dilihat pada Lampiran C.

6.3.2.3 Menentukan Domain Atribut

Pada tahap ini diidentifikasi domain atribut yang dimiliki oleh masing masing atribut pada entitas. Domain atribut adalah sebuah wadah dari value yang dimiliki oleh satu atau banyak atribut. Terdapat beberapa domain atribut yang didefinisikan seperti:

1. Panjang karakter yang diperbolehkan
2. Kemungkinan nilai yang diperbolehkan

6.3.2.4 Menentukan Atribut Kunci

Pada tahap ini diidentifikasi atribut kunci yang mengidentifikasi setiap kejadian pada suatuu entitas. Tabel 4.10 menunjukkan semua atribut kunci yang ada pada setiap entitas.

Tabel 6.9 Tabel atribut kunci

Nama Entitas	**Atribut Kunci**
Fans	id, phone
Visitors	id, phone
Rewards	id
Voucher	id, coupon_code
Offers	id

6.3.3 Perancangan Model Data Logis

Model data logis dirancang berdasarkan hasil perancangan data model konseptual. Dalam proses perancangan data model logis, terdapat beberapa hal yang dilakukan seperti merancang hubungan antara entitas, mengidentifikasi atribut yang dimiliki oleh masing-masing entitas dan melakukan normalisasi entitas.

Deskripsi atribut dan hubungan antar entitas yang lebih lengkap dapat dilihat pada Lampiran C. Gambar 4.3 menunjukkan diagram model data logis.

6.3.3.1 Menentukan Relasi

Setiap entitas mempunyai hubungan dengan entitas lainnya. Tabel 4.9 menunjukkan hubungan entitas dengan menentukan multiplisitas dan hubungannya.

Tabel 6.10 Hubungan antar entitas

Nama Entitas	**Multiplisitas**	**Hubungan**	**Multiplisitas**	**Nama Entitas**
Fans	Satu	Memiliki	Banyak	Rewards
	Satu	Menyebarkan	Banyak	Offers
Visitors	Satu	Memiliki	Banyak	Voucher
Offers	Satu	Memiliki	Banyak	Voucher
Voucher	Satu	Memiliki	Satu	Reward

6.3.3.2 Normalisasi Hubungan Entitas

Pada pembuatan data model konseptual, telah diidentifikasi semua entitas dari *user view*. Pada tahap ini dilakukan validasi terhadap semua entitas yang telah diidentifikasi menggunakan aturan normalisasi. Tujuannya adalah untuk memastikan bahwa entitas dan hubungannya dengan entitas lain memiliki jumlah minimum tetapi memenuhi kebutuhan sistem. Selain itu, hubungan antar entitas juga harus meminimalkan data yang redundan.

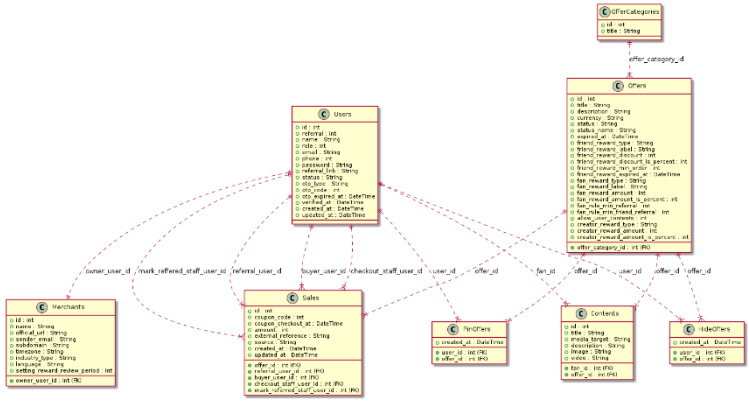
Tabel 6.11 Normalisasi entitas

Entitas Lama	**Entitas Baru**	**Normal Form**
Fans	Users	3NF

Visitors		
Offers	Offers	3NF
Voucher	Sales	3NF
-	OffersCategories	3NF
-	Contents	3NF

6.3.3.3 Merancang Data Model Global

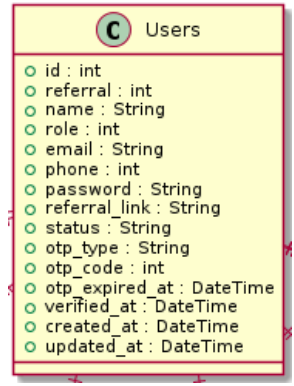
Pada tahap ini dirancang data model global yang dibuat setelah semua entitas telah mempunyai 3NF normal form. Gambar 4.3 menunjukkan diagram entitas data model logis.



Gambar 6.3 ER Diagram model data logis

1. Users

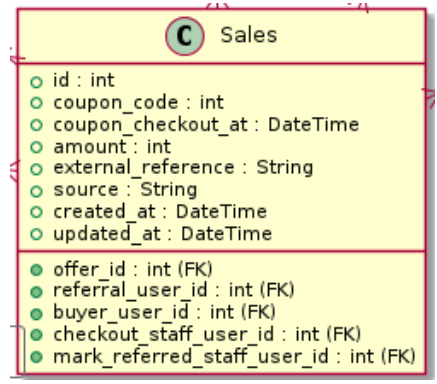
Tabel user digunakan untuk menyimpan data pengguna seperti nama, role, email, phone, password, referral, status, otp_type, otp_code, otp_expired_at, verified_at, created_at dan updated_at. Tabel users juga digunakan untuk melakukan update data profil oleh users. Gambar 4.4 menunjukkan tabel users.



Gambar 6.4 Tabel Users

2. Sales

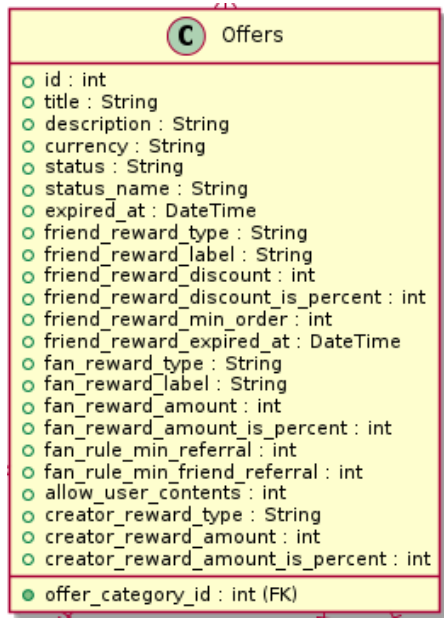
Tabel sales digunakan untuk menyimpan data transaksi yang dilakukan oleh fans dan visitors. Tabel sales mulai digunakan saat visitors melakukan klaim voucher. Fitur lain yang menggunakan tabel sales adalah saat fans melihat data hadiah yang dimilikinya. Selain itu, fitur lain yang dikembangkan diluar penelitian ini juga menggunakan tabel sales seperti fitur saat merchants melakukan validasi transaksi yang dilakukan oleh visitors. Gambar 4.5 menunjukkan tabel sales.



Gambar 6.5 Tabel Sales

3. Offers

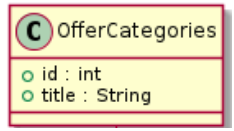
Tabel offers digunakan untuk menyimpan penawaran yang dibuat oleh merchants. Peraturan yang dimiliki oleh penawaran juga disimpan pada tabel ini. Tabel offers juga digunakan untuk menyajikan data kepada fans dan visitors Gambar 4.6 menunjukkan tabel offers.



Gambar 6.6 Tabel Offers

4. OfferCategories

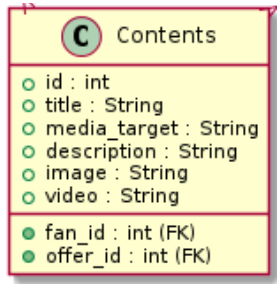
Tabel offerCategories digunakan untuk menyimpan data kategori penawaran yang dibuat oleh merchants. Tabel ini berisi id dan judul kategori. Gambar 4.7 menunjukkan tabel kategori.



Gambar 6.7 Tabel OfferCategories

5. Contents

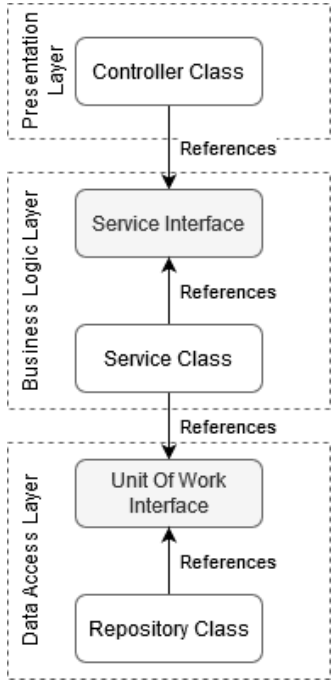
Tabel contents digunakan untuk menyimpan data konten yang dibuat oleh merchants. Tabel ini berisi gambar, judul, dan id dari konten. Setiap penawaran dapat memiliki lebih dari satu konten. Gambar 4.8 menunjukkan tabel contents.



Gambar 6.8 Tabel Contents

6.4 Perancangan Arsitektur Aplikasi

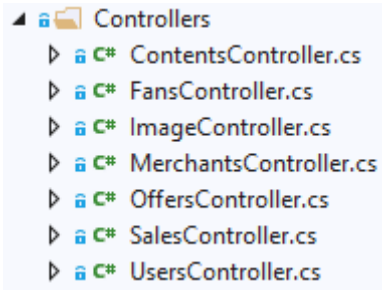
Aplikasi dibuat mengikuti arsitektur N-Layer yang terdiri dari tiga layer yaitu *presentation layer*, *business logic layer* dan *data access layer*. Setiap *layer* independen antara satu dengan lainnya. Gambar 4.9 menunjukkan diagram arsitektur yang digunakan.



Gambar 6.9 Diagram arsitektur aplikasi

6.4.1 Presentation Layer

Semua interaksi dengan *client* ada pada *layer* ini. *Client* menggunakan REST *endpoint* telah didefinisikan pada kelas *controller*. Pada *layer* ini hanya sampai validasi *request* yang diterima, sedangkan logika bisnis terdapat pada *layer* selanjutnya. *Presentation layer* berada pada folder *controller* seperti pada Gambar 4.10.



Gambar 6.10 Folder Controllers

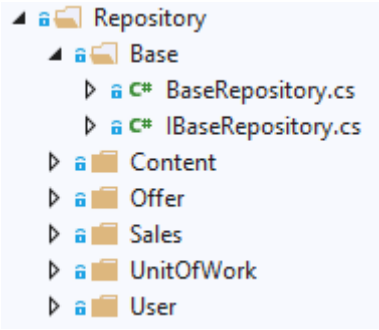
6.4.2 Business Logic Layer

Business Logic Layer berisi semua logika bisnis yang ada pada sistem. *Layer* ini menjembatani *presentation layer* dengan *Data Access Layer*. Kelas dibuat terpisah dengan *interface*-nya, hal ini membuat *interface* dapat dirancang secara terpisah. Implementasi kelas repository dengan *interface* menggunakan teknik *Dependency Injection* seperti Gambar 4.11.

Gambar 6.11 Folder Services

6.4.3 Data Access Layer

Data Access Layer digunakan untuk berkomunikasi dengan *database*. Kelas *BaseRepository* dibuat untuk generalisasi metode CRUD sederhana. Metode dengan *query* yang lebih kompleks menggunakan *repository* entitasnya masing-masing. Kelas UnitOfWork digunakan untuk menghubungkan *data access layer* dengan *business logic layer*. Gambar 4.12 menunjukkan *data access layer* yang berada pada folder *repository*.



Gambar 6.12 Folder Repository

BAB V
IMPLEMENTASI RANCANGAN APLIKASI

Pada bab implementasi dijelaskan tentang proses implementasi rancangan aplikasi yang telah dirancang sebelumnya. Implementasi dilakukan menggunakan bahasa pemrograman C# dan menggunakan kerangka kerja ASP.NET Core. Terdapat tiga pengembangan yang dilakukan yaitu pengembangan model data, pengembangan layanan aplikasi dan pengembangan dokumentasi layanan aplikasi

7.1 Lingkungan Implementasi

Terdapat banyak pilihan bahasa pemrograman untuk mengimplementasikan REST API. Penelitian ini menggunakan bahasa pemrograman C# dengan menggunakan kerangka kerja ASP.NET Core. Pemilihan Visual Studio 2019 sebagai IDE karena Visual Studio 2019 mempunyai kemudahan dalam hal debugging dan mempunyai banyak *tools* yang bisa membantu dalam proses implementasi. Tabel dibawah menunjukkan spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam tahap implementasi kode.

Tabel 7.1 Spesifikasi perangkat keras

Perangkat Keras		Spesifikasi
PC/Laptop	Prosesor	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz (8 CPUs), ~2.8GHz
	RAM	8192 MB

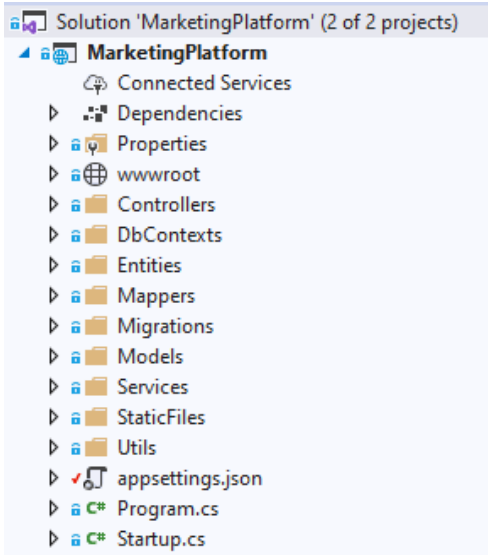
Tabel 7.2 Spesifikasi perangkat lunak

Perangkat Lunak / Tools	Spesifikasi
Sistem Operasi	Windows 10
Bahasa Pemrograman	C#

Web Server	IIS HTTP Server
Basis Data	PostgreSQL 12.2
Kerangka kerja Backend	ASP.NET Core
IDE	Visual Studio 2019

7.2 Inisiasi Folder Project

Pembuatan folder project menggunakan aplikasi Visual Studio 2019 yang juga digunakan sebagai IDE. Tahap selanjutnya adalah memilih *template* ASP.NET Core Web Application dengan bahasa C# dan SDK ASP.NET Core 3.1. File Project menggunakan nama MarketingPlatform. Gambar 5.1 menunjukkan tampilan folder project setelah dibuat.



Gambar 7.1 Folder project marketing platform

7.3 Implementasi Akses Data

7.3.1 Konfigurasi Penyedia Data Akses

Npgsql adalah penyedia data akses untuk *database* postgre pada EF Core. Package tambahan diperlukan untuk menjalankan postgresql sebagai provider *database* bagi EF Core. Konfigurasi dilakukan pada file .csproj seperti yang ditunjukkan pada Gambar 5.2.

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Npgsql.EntityFrameworkCore.PostgreSQL" Version="3.1.3" />
  </ItemGroup>
</Project>
```

Gambar 7.2 Konfigurasi *package npgsql data provider*

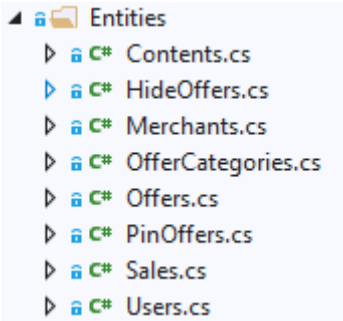
7.3.2 Implementasi Kelas Model

Kelas model yang dibuat dikenal sebagai kelas POCO (*Plain Old CLR Objects*). Kelas model tidak ada keterikatan dengan EF Core karena kelas model hanya mendefinisikan properti yang dimiliki oleh data. Kelas model mendefinisikan semua atribut yang ada pada entitas yang telah didefinisikan sebelumnya berserta tipe datanya. Pendefinisian relasi juga dilakukan pada kelas model. Penulisan kode pada kelas model dilakukan seperti yang ditunjukkan pada Gambar 5.3.

```
using System;

namespace MarketingPlatform.Entities
{
    public class Merchants
    {
        public int id { get; set; }
        public String name { get; set; }
        public String official_url { get; set; }
        public String sender_email { get; set; }
        public String subdomain { get; set; }
        public DateTime timezone { get; set; }
        public String industry_type { get; set; }
        public String language { get; set; }

        // Foreign Key
        public int owner_user_id { get; set; }

        // Reference Navigation Property
        public virtual Users Users { get; set; }
    }
}
```

Gambar 7.3 Kelas mode merchants

Semua kelas model dibuat dan disimpan pada folder Entities. Kelas model dibuat per masing-masing entitas yang telah dirancang sebelumnya. Gambar 5.4 menunjukkan folder *entities* yang berisi semua kelas model.

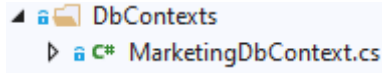


Gambar 7.4 Folder entities

7.3.3 Implementasi Kelas Database Context

Kelas *database context* merepresentasikan hubungan aplikasi dengan *database*. Kelas *database context* digunakan untuk

melakukan *query* dan penyimpanan data pada entitas yang telah didefinisikan pada kelas model. Nama kelas *database context* pada pengembangan aplikasi adalah *MarketingDbContext*. Kelas *MarketingDbContext* mengimplmentasikan *interface DbContext* yang telah disediakan oleh ASP.NET Core. Implementasi *interface DbContext* pada kelas *MarketingDbContext* menggunakan teknik *dependency injection* (DI). Kelas *MarketingDbContext* berada pada folder *DbContexts* yang terpisah dari folder *entities* Gambar 5.5 menunjukkan folder DbContexts.



Gambar 7.5 Folder DbContext

Kelas model yang telah dibuat kemudian didaftarkan pada kelas *MarketingDbContext* seperti potongan kode pada Gambar 5.6.

```
using MarketingPlatform.Entities;
using Microsoft.EntityFrameworkCore;

namespace MarketingPlatform.DbContexts
{
    public class MarketingDbContext : DbContext
    {
        public MarketingDbContext(DbContextOptions<MarketingDbContext> options) :
base(options)
        {

        }
        public DbSet<Merchants> Merchants { get; set; }
        public DbSet<Offers> Offers { get; set; }
        public DbSet<OfferCategories> OfferCategories { get; set; }
        public DbSet<PinOffers> PinOffers { get; set; }
        public DbSet<HideOffers> HideOffers { get; set; }
        public DbSet<Users> Users { get; set; }
        public DbSet<Contents> Contents { get; set; }
        public DbSet<Sales> Sales { get; set; }

    }

}
```

Gambar 7.6 Kelas MarketingDbContext

Kelas *MarketingDbContext* yang telah dibuat kemudian didaftarkan pada kelas *startup.cs*. Pendaftaran kelas *MarketingDbContext* menggunakan teknik *dependency injection* (DI). Pendaftaran kelas *MarketingDbContext* dilakukan seperti Gambar 5.7.

```
namespace MarketingPlatform
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }
        public IConfiguration Configuration { get; }
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<MarketingDbContext>(options =>
options.UseNpgsql(Configuration.GetConnectionString("DbConnection")));

        }

    }

}
```

Gambar 7.7 Pemasangan DbContext pada kelas Startup

Implementasi *DbContext* dilakukan pada *repository*. *Repository* menggunakan *DbContext* sebagai media untuk berinteraksi dengan *database*. Gambar 5.8 menunjukkan implementasi *DbContext* pada kelas *ContentRepository*.

```
using MarketingPlatform.DbContexts;
using System;
using System.Linq;

namespace MarketingPlatform.Services
{
    public class ContentRepository : IContentRepository
    {
        private readonly MarketingDbContext _context;

        public ContentRepository(MarketingDbContext context)
        {
            _context = context ?? throw new ArgumentNullException(nameof(context));
        }

    }

}
```

Gambar 7.8 Pemasangan MarketingDbContext pada ContentRepository

Implementasi DbContext dilakukan menggunakan teknik *dependency injection* (DI) dengan mendeklarasikan atribut _context yang mempunyai tipe data objek MarketingDbContext dan mempunyai akses modifier *private readonly*.

7.3.4 Konfigurasi Connection String Database

Kelas *database context* yang telah didaftarkan pada startup membutuhkan *connection string* untuk terhubung dengan *database*. *Connection string* berisi kumpulan informasi seperti nama server, nama *database*, *username* dan *password*. Penelitian ini menggunakan server localhost untuk pengembangan dengan nama *database* "dbmarketing" yang berjalan pada port 5433. Konfigurasi *connection string* dilakukan pada file *appsetting.json* seperti pada Gambar 5.9.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DbConnection":
"Server=localhost;Port=5433;Database=dbmarketing;Username=postgres;Password=123;"
  }
}
```

Gambar 7.9 Konfigurasi connection string pada appsetting.json

7.3.5 Pembuatan Migrasi Database

Sering terjadi perubahan terhadap model data pada saat fase pengembangan. Ketika perubahan model data dilakukan, aplikasi dan *database* menjadi tidak sinkron. EF Core menyediakan fitur migrasi agar aplikasi dan *database* tetap sinkron. Pembuatan migrasi dilakukan dengan cara menuliskan perintah *Add-Migration* pada *package manager console* (PMC) seperti pada Gambar 5.10.

```
PM> Add-Migration InitialCreate

PM> Update-Database
```

Gambar 7.10 Perintah Migrasi pada PMC

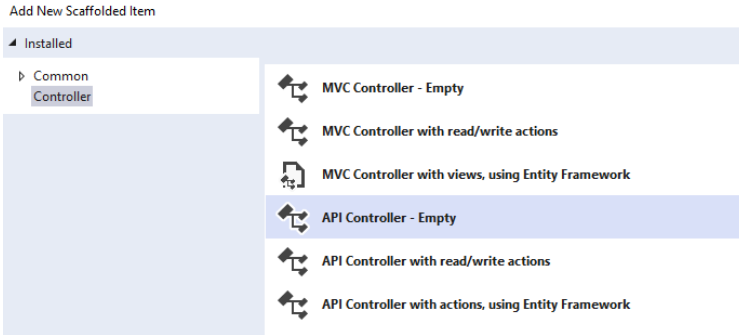
Perintah Update-Database dilakukan untuk membuat *database*. *Database* dibuat berdasarkan konfigurasi pada connection string. Perintah Update-Database akan membuat *database* berdasarkan migration terakhir yang dibuat.

7.4 Implementasi Web API Endpoint

Pada tahap ini dilakukan implementasi terhadap rancangan endpoint yang telah dibuat sebelumnya. Setiap endpoint dibuat pada kelas controller dengan menuliskan anotasi yang diperlukan sesuai dengan kebutuhannya. Anotasi yang ditulis akan diidentifikasi oleh kerangka kerja ASP.NET Core sebagai suatu endpoint. Contohnya, Gambar 5.12 menunjukkan penggunaan anotasi *[Route]* pada kelas controller.

7.4.1 Controller dan Routing

Pembuatan *controller* menggunakan *scaffolded item*, yaitu *code generation* yang merupakan fitur pada Visual Studio untuk project WEB API. Template kosong dipilih karena *controller* menggunakan pola yang berbeda dengan template standar Visual Studio.



Gambar 7.11 Pembuatan *controller* dengan teknik scaffolded item

Gambar 5.12 menunjukkan *controller* yang telah dibuat dengan *tools scaffolded item*.

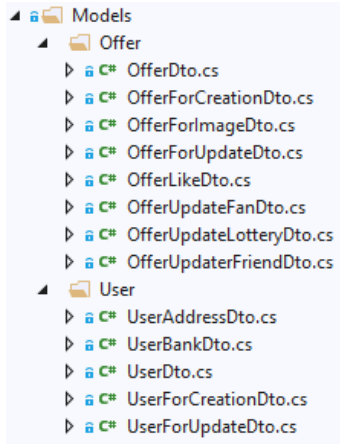
```
using Microsoft.AspNetCore.Mvc;

namespace MarketingPlatform.Controller
{
        [Route("api/merchants")]
        [ApiController]
        public class MerchantsController : ControllerBase
        {

        }

}
```

Gambar 7.12 Kelas Merchants *Controller* yang masih kosong

Controller pada *project* WEB API selalu mewarisi kelas ControllerBase karena *controller* pada *project* ini tidak memiliki *view*, tidak seperti *controller* pada ASP.NET MVC. Routing menggunakan *annotation* [Route("api/merchants")] untuk mengarahkan *request* HTTP yang masuk.

7.4.2 Mencegah Posting Berlebihan dengan DTO

Data Transfer Object (DTO) merupakan subset dari model yang telah dibuat sebelumnya. Penggunaan DTO dimaksudkan untuk mengurangi data yang dikirim agar tidak mempengaruhi performa sistem dan menyembunyikan data yang tidak seharusnya dilihat oleh *client* untuk alasan keamanan. Gambar 5.13 menunjukkan folder Models, dimana semua kelas DTO disimpan.



Gambar 7.13 Folder Models

Gambar 5.14 menunjukkan kelas UserForCreationDto. DTO ini digunakan saat membuat users baru. Kelas UserForCreationDto hanya mengambil beberapa properti dari entitas users.

```
using System;

namespace MarketingPlatform.Models
{
    public class UserForCreationDto
    {
        public String name { get; set; }
        public String email { get; set; }
        public long phone { get; set; }
    }
}
```

Gambar 7.14 Kelas DTO untuk membuat user baru

Semua kelas DTO yang telah dibuat kemudian didaftarkan pada *mapper* sesuai dengan kebutuhan. *Mappers* digunakan untuk konfigurasi "jalan" DTO menuju entitasnya atau sebaliknya. Gambar 5.15 menunjukkan kelas *UserMapper* yang berisi konfigurasi semua DTO users.

```
using AutoMapper;

namespace MarketingPlatform.Mappers
{
    public class UserMapper : Profile
    {
        public UserMapper()
        {
            CreateMap<Entities.Users, Models.UserDto>();
            CreateMap<Models.UserForCreationDto, Entities.Users>();
            CreateMap<Models.UserForUpdateDto, Entities.Users>();
            CreateMap<Models.UserBankDto, Entities.Users>();
            CreateMap<Models.UserAddressDto, Entities.Users>();
        }
    }
}
```

Gambar 7.15 Kelas UserMapper

Gambar 5.16 menunjukkan implementasi penggunaan *mapper* pada kelas *OfferController*. *Mapper* didefinisikan menggunakan teknik *dependency injection* yang diberi nama _mapper. Kode pada Gambar 5.16 menunjukkan *mapper* digunakan pada *method* GetOffer(id). Pada method tersebut, *mapper* digunakan pada saat menerima objek dari repository yang kemudian objek tersebut diubah sesuai dengan propoerti pada DTO. Objek tersebut kemudian dikirim kepada *client* menggunakan responsse kode HTTP 200.

```
namespace MarketingPlatform.Controller
{
    [Route("api/offer")]
    [ApiController]
    public class OffersController : ControllerBase
    {
        private readonly IOfferRepository _offerRepository;
        private readonly IMapper _mapper;

        public OffersController(IOfferRepository offerRepository,
IMapper mapper, IWebHostEnvironment environment)
        {
            _offerRepository = offerRepository ??
                throw new ArgumentNullException(nameof(_offerRepository));
            _mapper = mapper ??
                throw new ArgumentNullException(nameof(_mapper));
        }


        [HttpGet("{id}")]
        public ActionResult<Offers> GetOffer(long id)
        {
            var offer = _offerRepository.GetOffer(id);

            if (offer == null)
            {
                return NotFound();
            }

            return Ok(_mapper.Map<OfferDto>(offer));
        }
    }

}
```

Gambar 7.16 Implementasi mapper pada controller

7.4.3 Memetakan Objek dengan AutoMapper

DTO yang telah dibuat harus dipetakan pada entitas aslinya. Sebuah *library* bernama *AutoMapper* dibuat untuk membantu dan mempermudah memetakan objek DTO kepada entitasnya. Perintah pada Gambar 5.17 diperlukan untuk memasang *package* pada *project*.

```
PM> Install-Package AutoMapper.Extensions.Microsoft.DependencyInjection
```

Gambar 7.17 Perintah untuk memasang AutoMapper

Package yang telah dipasang kemudian didaftarkan pada kelas startup.cs seperti Langkah yang dilakukan pada saat pemasangan *DbContext*. Gambar 5.18 menunjukkan potongan kode pendaftaran *package AutoMapper* pada startup.cs.

```
services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());
```

Gambar 7.18 Konfigurasi AutoMapper pada kelas startup.cs

7.4.4 Implementasi Interface Arsitektur Aplikasi

Implementasi *interface* arsitektur dilakukan dengan cara membuat interface pada *business logic layer* dan *data access layer*.

7.4.4.1 Interface Business Logic Layer

Interface pada *layer* ini diimplementasikan pada kelas *service* dan juga *controller*. Gambar 5.19 menunjukkan salah satu *interface* yang bernama *IOfferService*. *Interface* ini akan diimplementasikan pada kelas *OfferService* dan menjadi referensi kelas *OffersController*.

```
namespace MarketingPlatform.Services.Offer
{
    public interface IOfferService
    {
        IEnumerable<Offers> GetAllOffer();

        Offers GetOfferId(int id);

        void PostOffer(Offers offer);

        object GetOfferStats(int id);
    }
}
```

Gambar 7.19 Interface IOfferService

7.4.4.2 Interface Data Access Layer

Interface pada *layer* ini diimplementasikan pada kelas *repository* dan juga *services*. *Interface* pada layer ini bernama *IUnitOfWork*. *Interface* ini akan diimplementasikan pada semua *repository* dan menjadi referensi semua *services*.

```
namespace DataAccess.Repository.UnitOfWork
{
    public interface IUnitOfWork
    {
        IOfferRepository offerRepository { get; }
        IUserRepository userRepository { get; }
        ISalesRepository salesRepository { get; }
        IContentRepository contentRepository { get; }
        IMerchantRepository merchantRepository { get; }
        void Commit();
        void Rollback();
    }
}
```

Gambar 7.20 Interface IUnitOfWork

7.4.5 Menangani Request yang Masuk

Implementasi *endpoint* dimulai dari pembuatan *method* pada *service repository*. *Method* yang telah dibuat kemudian diimplementasikan pada kelas yang berisi kumpulan logika bisnis. *Method* tersebut memanggil *interface data access layer* untuk mendapatkan sumber daya yang diinginkan. Pembuatan *endpoint* pada *controller* berdasarkan fitur yang dibuat. *Method* pada *controller* memanggil *service interface* untuk menggunakan logis bisnis yang sudah dibuat sebelumnya. *Controller* kemudian memberikan kode status HTTP kepada *client*

7.4.5.1 Menyajikan Semua Data

Endpoint ini digunakan untuk menyajikan semua data yang ada pada suatu entitas. Beberapa fitur yang menggunakan *endpoint* ini adalah fitur menyajikan semua penawaran, menyajikan semua hadiah dan menyajikan semua penawaran. Pembuatan *endpoint* ini menggunakan *method* HTTP GET pada *controller* dengan tidak memberikan parameter tambahan. Gambar 5.12 menunjukkan salah satu implementasi *endpoint* untuk fitur menyajikan semua penawaran. *Method* ini mempunyai tipe data *IEnumerable* yang merupakan objek berupa list. *Method* yang digunakan memanggil *method* pada kelas *service* yang

memanggil *method* pada kelas *repository* untuk mendapatkan sumber daya.

```
namespace MarketingPlatform.Controller
{
    [Route("api/offer")]
    [ApiController]
    public class OffersController : ControllerBase
    {
        private IOfferService offerService;
        private readonly IMapper mapper;


        public OffersController(IOfferService offerService, IMapper mapper)
        {
            this.offerService = offerService;
            this.mapper = mapper;
        }

        [HttpGet]
        [Route("")]
        public IEnumerable<Offers> GetAllOffer() => offerService.GetAllOffer();
```

Gambar 7.21 Endpoint GET api/offer pada OffersController

7.4.5.2 Menyajikan Data Berdasarkan Id

Endpoint ini digunakan untuk menyajikan data berdasarkan id yang dimasukkan. Beberapa fitur yang menggunakan *endpoint* ini adalah fitur menyajikan data *reward*, menyajikan data penawaran dan menyajikan data profil. Pembuatan *endpoint* ini menggunakan *method* HTTP GET pada *controller* dengan memberikan parameter tambahan sesuai dengan kebutuhan. Gambar 5.22 menunjukkan salah satu implementasi *endpoint* untuk fitur menyajikan satu penawaran. *Method* ini mempunyai parameter tambahan yaitu parameter *id*. *Method* ini mempunyai tipe data *ActionResult* yang mengembalikan nilai berupa objek entitas *offers*. *Method* yang digunakan memanggil *method* pada kelas *offersService* yang memanggil *method* pada kelas *repository* untuk mendapatkan sumber daya.

```
namespace MarketingPlatform.Controller
{
    [Route("api/offer")]
    [ApiController]
    public class OffersController : ControllerBase
    {
        private IOfferService offerService;
        private readonly IMapper mapper;


        public OffersController(IOfferService offerService, IMapper mapper)
        {
            this.offerService = offerService;
            this.mapper = mapper;
        }

        [HttpGet]
        [Route("{id}")]
        public ActionResult<Offers> GetOffer(int id) => offerService.GetOfferId(id);
```

Gambar 7.22 Endpoint GET api/offer/{id} pada OffersController

7.4.5.3 Menambah Data Baru

Endpoint ini digunakan untuk menambah data pada *database*. Beberapa fitur yang menggunakan *endpoint* ini adalah fitur pendaftaran user dan membuat penawaran. Pembuatan *endpoint* ini menggunakan *method* HTTP POST pada *controller* dengan memberikan parameter tambahan sesuai dengan kebutuhan. Gambar 5.23 menunjukkan salah satu implementasi *endpoint* untuk fitur membuat user. *Method* ini mempunyai parameter tambahan yaitu parameter *tipe* yang digunakan untuk menentukan *role* user yang dibuat. *Method* ini mempunyai tipe data *ActionResult* yang mengembalikan nilai berupa objek *UserDto*. Objek yang dikirim dipetakan menggunakan *mapper*. *Mapper* memetakan objek DTO yang dikirim *client* menjadi objek entitas user menggunakan *mapper*. *Method PostUser* pada *userService* dipanggil untuk menambahkan objek pada *database*. Jika berhasil, *controller* akan memberikan kode respons HTTP 201.

```
namespace MarketingPlatform.Controller
{
    [Route("api/user")]
    [ApiController]
    public class UsersController : ControllerBase
    {
        private IUserService userService;
        private readonly IMapper mapper;

        public UsersController(IUserService userService, IMapper mapper)
        {
            this.userService = userService;
            this.mapper = mapper;
        }

        [HttpPost]
        [Route("{tipe}")]
        public ActionResult<UserDto> PostUser([FromBody] UserCreateDto user, int tipe)
        {

            var userEntity = mapper.Map<Users>(user);
            userService.PostUser(userEntity, tipe);
            var userToReturn = mapper.Map<UserDto>(userEntity);

            return CreatedAtAction("GetAllUser", new { id = userToReturn.id }, userTo-
Return);
        }
```

Gambar 7.23 Endpoint POST api/user/{tipe} pada UsersController

7.4.5.4 Memperbarui Data

Endpoint ini digunakan untuk memperbarui data pada *database*. Beberapa fitur yang menggunakan *endpoint* ini adalah fitur *update* profil, mengganti kode referral, mengganti informasi rekening, mengganti informasi alamat pengiriman dan klaim voucher untuk fans baru. Pembuatan *endpoint* ini menggunakan *method* HTTP PUT pada *controller* dengan memberikan parameter tambahan sesuai dengan kebutuhan. Gambar 5.24 menunjukkan salah satu implementasi *endpoint* untuk fitur membuat user. *Method* ini mempunyai parameter tambahan yaitu parameter *{id}/update*. *Method* ini mempunyai tipe data *IActionResult* tidak mengembalikan nilai apapun. Objek yang dikirim dipetakan menggunakan *mapper*. *Mapper* memetakan objek DTO yang dikirim *client* menjadi objek entitas user menggunakan *mapper*. *Method UpdateUser* pada *userService* dipanggil untuk memperbarui objek pada *database*. Jika berhasil, *controller* akan memberikan kode respons HTTP 204.

```
namespace MarketingPlatform.Controller
{
    [Route("api/user")]
    [ApiController]
    public class UsersController : ControllerBase
    {
        private IUserService userService;
        private readonly IMapper mapper;

        public UsersController(IUserService userService, IMapper mapper)
        {
            this.userService = userService;
            this.mapper = mapper;
        }

        [HttpPut]
        [Route("{id}/update")]
        public IActionResult UpdateUserAll(int id, [FromBody] UserUpdateDto user)
        {
            var userEntity = userService.GetUserId(id);
            mapper.Map(user, userEntity);
            userService.UpdateUser(userEntity);
            return NoContent();
        }
```

Gambar 7.24 Endpoint PUT api/user/{id}/update pada UsersController

7.4.5.5 Menyebarkan Penawaran

Endpoint ini digunakan saat fans baru dan fans loyal untuk menyebarkan penawaran yang dibuat oleh *merchants*. Pembuatan *endpoint* ini menggunakan *method* HTTP GET pada *controller* dengan memberikan parameter *{id}/share/{status}*. *Method* ini mempunyai tipe data *IActionResult* yang mengembalikan nilai berupa JSON. *Method* yang digunakan memanggil *method* pada kelas *contentService* yang memanggil *method* pada kelas *repository* untuk mendapatkan sumber daya. Jika berhasil, *controller* akan memberikan kode respons HTTP 200.

```
namespace MarketingPlatform.Controller
{
    [Route("api/content")]
    [ApiController]
    public class ContentsController : ControllerBase
    {
        private IContentService contentService;
        private readonly IMapper mapper;


        public ContentsController(IContentService contentService, IMapper mapper)
        {
            this.contentService = contentService;
            this.mapper = mapper;
        }

        [HttpGet("{id}/share/{status}")]
        public IActionResult Share(int id, int user_id, string status)
        {
            var result = contentService.ShareContent(id, user_id, status);
            return Ok(result);
        }
```

Gambar 7.25 Endpoint GET api/content/{id}/share/{status} pada ContentsController

Method ShareContent menggunakan beberapa API media sosial seperti Facebook, Twitter, Telegram, WhatsApp. Mula-mula *method ShareContent* didefinisikan pada *ContentService*. *Method* ini mempunyai tipe data *object*. *Method* ini mempunyai tiga parameter. Parameter pertama yaitu parameter *id* yang merupakan penanda unik dari konten. Parameter kedua yaitu parameter *user_id* yang merupakan *id* dari *fans* yang menyebarkan penawaran. Parameter *user_id* merupakan parameter opsional yang berarti bisa tidak ada nilainya karena penawaran bisa saja dibagikan tanpa ada *fans* yang mereferensikan. Parameter ketiga yaitu parameter *status* yang digunakan untuk mengetahui media sosial apa yang akan digunakan untuk menyebarkan penawaran. *Method* ini mengembalikan nilai *referral_uri*. *Referral_uri* adalah sebuah *link* untuk menuju halaman penawaran untuk fans baru. Didalam *referral_uri* terdapat kode referral dari fans loyal agar fans loyal mendapat hadiah dari transaksi yang dilakukan oleh fans baru. Jika fans baru menyebarkan penawaran dengan cara menyalin link, maka nilai yang dikembalikan hanya tautan menuju halaman penawaran untuk fans baru. Sedangkan jika fans menyebarkan penawaran melalui media sosial, maka nilai

yang dikembalikan adalah masing-masing api media sosial sesuai dengan media sosial yang dipilih. Gambar 5.26 menunjukkan implementasi logis bisnis dari fitur membagikan penawaran.

```
namespace MarketingPlatform.Services.Content
{
    public class ContentService : IContentService
    {
        private IUnitOfWork _unitOfWork;

        public ContentService(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        public object ShareContent(int id, int? user_id, string status)
        {
            var redirect_uri = "";
            var content = _unitOfWork.contentRepository.GetContentId(id);
            var subdomain = _unitOfWork.merchantRepository.GetMerchantId(id).subdomain;
            if (user_id != null)
            {
                var referral = _unitOfWork.userRepository.GetUserId(id).referral;
                redirect_uri = subdomain + "/offer_id=" + content.offerId + "?referral="
+ referral;
            }
            else
            {
                redirect_uri = subdomain + "/offer_id=" + content.offerId;
            }
```

Gambar 7.26 Method ShareContent pada ContentService (1)

```
            if (status == "copy")
            {
                var result = new { redirect_uri = redirect_uri };
                return result;
            }
            else if (status == "fb")
            {
                var result = new { redirect_uri =
"https://www.facebook.com/v2.9/dialog/feed?app_id=165794590184026&link=" + redirect_uri +
"%3Ft%3Dfb&redirect_uri=http%3A%2F%2Fportal.referralcandy.com%2FVTK9S53%2Ffb_callback%3Ft
ype%3Dshare_on_facebook_click" };
                return result;
            }
            else if (status == "wa")
            {
                var result = new { redirect_uri =
"https://api.whatsapp.com/send?text=Promo%20Murah%2020%25%20hanya%20di%20marketing%20plat
form.%20https%3A%2F%2F" + redirect_uri + "%3Ft%3Dwa" };
                return result;
            }
            else if (status == "twitter")
            {
                var result = new { redirect_uri =
"http://twitter.com/share?text=Gunakan%20link%20referral%20ini%20untuk%20mendapat%20disco
unt%20yang%20menarik%20dari%20kami%21%21%21&url=https%3A%2F%2F" + redirect_uri +
"%3Ft%3Dtw" };
                return result;
            }
            else if (status == "telegram")
            {
                var result = new { redirect_uri =
"https://telegram.me/share/url?url=http%3A%2F%2F" + redirect_uri +
"%3Ft%3Dtg&text=Gunakan%20Link%20ini%20untuk%20mendapatkan%20promo20menarik%20saat%20memb
eli%20produk%20kami%20di%20Marketing%20Platform%20" };
                return result;
            }
            else
            {
                return 0;
            }

        }
    }
}
```

Gambar 7.27 Method ShareContent pada ContentService (2)

7.4.5.6 Mendapatkan Voucher

Endpoint ini digunakan saat visitors melakukan klaim voucher. Pembuatan *endpoint* ini menggunakan *method* HTTP POST pada *controller* dengan memberikan parameter *claim*. *Method* ini mempunyai tipe data *ActionResult* yang mengembalikan nilai berupa objek SalesDto. Objek yang dikirim dipetakan menggunakan *mapper*. *Mapper* memetakan objek DTO yang dikirim *client* menjadi objek entitas sales menggunakan *mapper*. *Method ClaimSales* pada *salesService* dipanggil untuk menambahkan objek pada *database*. Jika berhasil, *controller* akan memberikan kode respons HTTP 201

```
namespace MarketingPlatform.Controller
{
    [Route("api/sales")]
    [ApiController]
    public class SalesController : ControllerBase
    {
        private ISalesService salesService;
        private readonly IMapper mapper;

        public SalesController(ISalesService salesService, IMapper mapper)
        {
            this.salesService = salesService;
            this.mapper = mapper;
        }

        [HttpPost]
        [Route("claim")]
        public ActionResult<SalesDto> ClaimOffer([FromBody] SalesCreateDto sales)
        {

            var salesEntity = mapper.Map<Sales>(sales);
            salesService.ClaimSales(salesEntity);
            var salesToReturn = mapper.Map<SalesDto>(salesEntity);

            return CreatedAtAction("GetAllSales", new { id = salesToReturn.id },
salesToReturn);
        }
```

Gambar 7.28 Endpoint POST api/sales/{claim} pada SalesController

Method ClaimSales pada kelas *salesService* berisi logis bisnis yang melakukan pengacakan terhadap kode voucher untuk visitors. Kode yang telah diacak kemudian dimasukkan pada objek sales yang diterima dari *controller*. *Method* ini kemudian menggunakan *salesRepository* untuk memasukkan data pada *database*.

```
namespace MarketingPlatform.Services.Saleses
{
    public class SalesService : ISalesService
    {
        private IUnitOfWork _unitOfWork;

        public SalesService(IUnitOfWork unitOfWork)
        {
            _unitOfWork = unitOfWork;
        }

        public void ClaimSales(Sales sales)
        {
            Random ran = new Random();
            String b = "abcdefghijklmnopqrstuvwxyz0123456789";
            String sc = "1234567890";

            int length = 6;

            String random = "";

            for (int i = 0; i < length; i++)
            {
                int a = ran.Next(b.Length);
              //string.Lenght gets the size of string
                random = random + b.ElementAt(a);
            }
            for (int j = 0; j < 2; j++)
            {
                int sz = ran.Next(sc.Length);
                random = random + sc.ElementAt(sz);
            }

            sales.coupon_code = random;
            //sales.coupon_checkout_at = DateTime.Now;
            _unitOfWork.salesRepository.Insert(sales);
        }
```

Gambar 7.29 Method ClaimSales pada SalesService

7.5 Dokumentasi Web API Endpoint Menggunakan Swagger

Dokumentasi web API dilakukan menggunakan *tools* swagger karena swagger mendukung pembuatan dokumentasi Web API secara otomatis. Pembuatan dokumentasi Web API tidak perlu dilakukan secara manual karena swagger dapat melakukan dokumentasi Web API seacara otomatis dengan cara membaca anotasi yang ditulis pada controller.

Pemasangan swagger pada project menggunakan *library* swashbuckle. Gambar 5.30 menunjukkan perintah yang digunakan untuk memasang *library* swashbuckle.

```
PM> Install-Package Swashbuckle.AspNetCore -Version 5.0.0
```

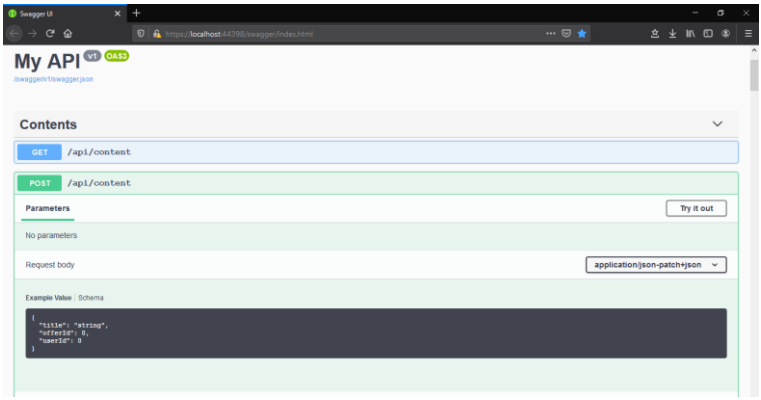
Gambar 7.30 Perintah untuk memasang library swashbuckle

Library swashbuckle yang telah dipasang kemudian dilakukan konfigurasi pada kelas startup. Gambar 5.31 menunjukkan konfigurasi swagger pada kelas startup.

```
services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo
    { Title = "My API", Version = "v1" });
});
```

Gambar 7.31 Konfigurasi swagger pada kelas startup

Semua endpoint yang dibuat pada controller otomatis dihasilkan oleh swagger. Gambar 5.32 menunjukkan tampilan SwaggerUI ketika mengakses endpoint `/swagger/v1/swagger.json`.



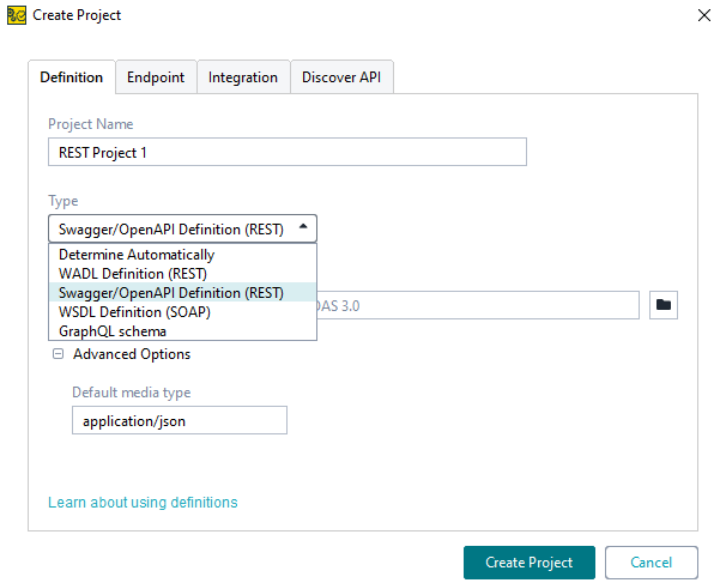
Gambar 7.32 Tampilan dokumentasi endpoint pada SwaggerUI

BAB VI
PENGUJIAN APLIKASI

Pada bab ini akan diuraikan hasil pengujian endpoint yang dilakukan dengan menggunakan teknik pengujian *black box*. Terdapat tiga jenis pengujian yaitu pengujian fungsional, pengujian performa dan pengujian keamanan. Pengujian endpoint menggunakan bantuan ReadyAPI sebagai alat bantu dalam melakukan ketiga jenis pengujian tersebut.

8.1 Memasukkan Dokumentasi Swagger untuk Pengujian

ReadyAPI menyediakan dukungan untuk definisi API menggunakan Swagger. Gambar 6.1 menunjukkan pembuatan project baru dengan memilih jenis project yaitu Swagger/OpenAPI Definition (REST). Pada *field* URL dimasukkan URL dari definisi Swagger pada aplikasi. Pengujian dengan menggunakan Swagger dapat mengefisienkan waktu untuk memasukkan endpoint satu persatu pada *tools* ReadyAPI.



Gambar 8.1 Menu membuat project baru pada ReadyAPI

8.2 Pengujian Fungsional

Endpoint yang digunakan didaftarkan pada ReadyAPI. Setiap endpoint diuji dengan menggunakan beberapa ketentuan. Ketentuan yang pertama adalah kode HTTP output yang dihasilkan sesuai dengan yang diharapkan. Kode respons HTTP GET yang diharapkan adalah 200. Kode respons HTTP POST yang diharapkan adalah 201, kecuali endpoint login yang memiliki kode respom HTTP 200. Kode respons HTTP PUT yang diharapkan adalah 204. Selain kode respons, waktu yang dibutuhkan untuk sebuat request maksimal adalah 200ms. Semua endpoint lolos uji fungsional dengan waktu respons dibawah 200ms. Tabel 6.1 menunjukkan hasil pengujian fungsional pada semua endpoint.

Tabel 8.1 Hasil pengujian fungsional

No	Nama	Http Method	Endpoint	Kode Respon	Hasil Tes
1	Register	POST	api/user/register	201	PASS
2	Login	POST	api/user/login	200	PASS
3	Claim Offers	POST	api/sales/claim	201	PASS
4	Get Users By Id	GET	api/user/{id}	200	PASS
5	Get Offers By Id	GET	api/offer/{id}	200	PASS
6	Get Contents By Id	GET	api/content/{id}	200	PASS
7	Get All Active Offers	GET	api/offer/active	200	PASS
8	Get User Referral	GET	api/user/{id}/referral	200	PASS
9	Get Reward Lottery	GET	api/sales/{id}/lottery	200	PASS
10	Get Voucher	GET	api/sales/{id}/voucher	200	PASS
11	Share Offers	GET	api/share/{id}/share/{status}	200	PASS
12	Like Offers	PUT	api/offer/{id}/like	204	PASS
13	Dislike Offers	PUT	api/offer/{id}/unlike	204	PASS
14	Update Users	PUT	api/user/{id}/update/	204	PASS
15	Update Users Bank	PUT	api/user/{id}/update/bank	204	PASS
16	Update Users Address	PUT	api/user/{id}/update/address	204	PASS
17	Update Users Referral	PUT	api/user/{id}/update/referral	204	PASS

8.3 Pengujian Keamanan

Pengujian kemanan dilakukan untuk memastikan endpoint aman dari serangan dan tidak menampilkan informasi yang sensitif. Terdapat beberapa pilihan pemindaian yang tersedia pada ReadyAPI. Pemindaian yang dilakukan pada tugas akhir ini adalah pemindaian *SQL injection* dan *weak authentication*. Pemindaian *SQL injection* dilakukan pada endpoint yang memiliki metode HTTP POST, sedangkan pemindaian *weak authentication* dilakukan pada endpoint yang memiliki metode HTTP PUT dan GET. Semua endpoint lolos uji kemanan. Tabel 6.2 Menunjukkan hasil pengujian keamanan pada semua endpoint.

Tabel 8.2 Hasil pengujian keamanan

No	Nama	Http Method	Endpoint	Scan Type	Hasil Tes
1	Register	POST	api/user/register	SQL Injection	PASS
2	Login	POST	api/user/login	SQL Injection	PASS
3	Claim Offers	POST	api/sales/claim	SQL Injection	PASS
4	Get Users By Id	GET	api/user/{id}	Weak Authentication	PASS
5	Get Offers By Id	GET	api/offer/{id}	Weak Authentication	PASS
6	Get Contents By Id	GET	api/content/{id}	Weak Authentication	PASS
7	Get All Active Offers	GET	api/offer/active	Weak Authentication	PASS
8	Get User Referral	GET	api/user/{id}/referral	Weak Authentication	PASS
9	Get Reward Lottery	GET	api/sales/{id}/lottery	Weak Authentication	PASS
10	Get Voucher	GET	api/sales/{id}/voucher	Weak Authentication	PASS
11	Share Offers	GET	api/share/{id}/share/{status}	Weak Authentication	PASS
12	Like Offers	PUT	api/offer/{id}/like	Weak Authentication	PASS
13	Dislike Offers	PUT	api/offer/{id}/unlike	Weak Authentication	PASS
14	Update Users	PUT	api/user/{id}/update/	Weak Authentication	PASS
15	Update Users Bank	PUT	api/user/{id}/update/bank	Weak Authentication	PASS
16	Update Users Address	PUT	api/user/{id}/update/address	Weak Authentication	PASS
17	Update Users Referral	PUT	api/user/{id}/update/referral	Weak Authentication	PASS

8.4 Pengujian Performa

Pengujian performa menggunakan waktu pemanasan selama 15 detik. Saat waktu pemanasan, sistem akan menerima request tetapi tidak mengumpulkan data. Waktu pemanasan berguna untuk menghilangkan data yang tidak reliabel saat awal server digunakan. Pengujian performa menggunakan waktu uji selama 30 detik. Pengujian performa menggunakan VUs (Virtual Users) sebanyak 10 dan 25 sebagai pembanding. VUs akan meminta request bersama-sama setiap detiknya selama 30 detik. Endpoint yang digunakan adalah endpoint login, claim voucher, dan share offers karena ketiga endpoint tersebut merupakan endpoint yang paling banyak digunakan oleh user. Kriteria lolos uji yang diberikan adalah waktu yang diperlukan untuk merespons satu request dibawah 100ms. Hasil yang diperoleh adalah semua endpoint lolos uji performa pada kriteria yang diberikan. Tabel 6.3 menunjukkan detail uji performa dari ketiga endpoint yang dipilih.

Tabel 8.3 Hasil pengujian performa

Nama	Http Method	Endpoint	Average Time Taken (ms)		Hasil Tes	
			10 VUs	25 VUs	10 VUs	25 VUs
Login	POST	api/user/login	4,3	7,36	PASS	PASS
Claim Voucher	POST	api/sales/claim	11,2	5,8	PASS	PASS
Share Offers	GET	api/share/{id}/share/{status}	5,4	4,96	PASS	PASS

BAB VII
KESIMPULAN DAN SARAN

Bab kesimpulan dan saran membahas mengenai kesimpulan proses penelitian yang telah dilakukan dan saran yang diusulkan baik untuk perusahaan maupun untuk penelitian serupa di masa mendatang.

9.1 Kesimpulan

Dari proses pengembangan aplikasi untuk pelanggan pada modul ajak teman, didapat kesimpulan-kesimpulan sebagai berikut:

1. Berdasarkan 9 user view, telah dirancang diagram data model logis berisikan 10 entitas. Entitas yang utama antara lain entitas Users, entitas Sales, dan entitas Offers.
2. Web service dikembangkan menggunakan kerangka kerja ASP.NET Core dengan arsitektur n-layer. Dalam kode perangkat lunak terdapat kelas controller sejumlah 6, kelas service sejumlah 5, dan kelas repository sejumlah 5. Web service ini berisikan 16 endpoint api.
3. Hasil pengujian web service adalah sebagai berikut:
   a. Pengujian fungsional: Semua endpoint lolos uji fungsional dengan kriteria lolos uji sesuai dengan metode HTTP yang dimiliki
   b. Pengujian Performa: Semua endpoint lolos uji keamanan dengan jenis pemindaian *SQL injection* dan *weak authentication*
   c. Pengujian Keamanan: Tiga sampel endpoint yang dipilih lolos uji performa dengan kriteria lolos uji yaitu waktu rata-rata yang dibutuhkan untuk menangani semua request perdetik dibawah 100ms
4. Dokumentasi web service dibangun menggunakan anotasi swagger pada controller. Dengan menggunakan swagger, API endpoint dapat dibaca dengan mudah oleh software pengujian.

5. Tugas akhir ini dapat digunakan sebagai acuan untuk mengembangkan perangkat lunak dengan arsitektur microservice.
6. Aplikasi backend siap digunakan oleh aplikasi frontend yang dikembangkan oleh Kemal (2020), Agung (2020), dan Akram (2020) agar menjadi aplikasi pemasaran yang dapat dimanfaatkan oleh UMKM.
7. Aplikasi backend telah memenuhi kebutuhan fungsional berdasarkan rancangan antar muka yang dirancang untuk melakukan otomasi terhadap pencatatan pelanggan baru dan pelanggan loyal UMKM. Oleh karena itu dengan digunakannya aplikasi frontend yang didukug oleh aplikasi backend pada tugas akhir ini, UMKM akan dapat melakukan pemasaran dan pencatatan pelanggan secara efisien.

9.2 Saran

Dalam pengerjaan tugas akhir, terdapat beberapa saran yang diharapkan dapat bermanfaat bagi UMKM maupun untuk pengembangan penelitian ke depan, yaitu:

1. Pengembangan pertama aplikasi masih sampai tahap pengujian black box. Diharapkan dalam pengembangan selanjutnya dilakukan pengujian integrasi dengan frontend yang telah dikembangkan secara terpisah.
2. Pengembangan ini masih menggunakan test case dasar. Oleh karena itu diharapkan untuk pengembangan selanjutnya menggunakan test case yang lebih kompleks untuk menjaga kualitas aplikasi.
3. Pengembangan ini hanya fokus pada modul ajak teman. Oleh karena itu diharapkan untuk pengembangan selanjutnya melanjutkan modul yang lain sehingga aplikasi mempunyai fitur yang lebih lengkap.
4. Otentikasi user pada pengembangan ini menggunakan sistem membership yang disediakan oleh ASP.NET Core karena web API hanya dikonsumsi oleh aplikasi internal.

Oleh karena itu diharapkan untuk pengembangan selanjutnya menggunakan IdentityServer4 untuk meningkatkan keamanan yang berguna ketika web API dapat dikonsumsi oleh semua user, seperti pada penggunaan HeadlessCRM.

DAFTAR PUSTAKA

[1] R. Maulidiana, “Analisa Pengaruh Knowledge Gap (K-Gap) terhadap Non-Financial Business Performance(NFPI) pada Dimensi Manajemen Sumber Daya Manusia (MSDM) serta Kualitas Produk dan Jasa di UMKM Sektor Tenun di Nagari Pandai Sikek,” 2016.

[2] Taufik dan Rahmat, “Masalah yang dihadapi Usaha Kecil Menengah di Indonesia,” *Jurnal Studi Ekonomi Syariah,* vol. II, no. 6, 2017.

[3] Deloitte, “UKM pemicu kemajuan Indonesia,” Deloitte, Jakarta, 2015.

[4] Leskovec, A. Jure, H. Lada A dan Bernardo A, “The dynamics of viral marketing,” *ACM Transactions on the Web (TWEB),* vol. 1, no. 1, p. 5, 2007.

[5] Jason dan Melaine, “Viral marketing: Motivations to forward online content,” *Journal of Business Research,* vol. 63, no. 9-10, pp. 1000-1006, 2010.

[6] Buttle dan Francis, “Word of mouth: Understanding and managing referral marketing,” *Journal of Strategic Marketing,* vol. 6, pp. 241-254, 1998.

[7] S. Nidhra dan J. Dondeti, “Black Box and White Box Testing Techniques - A Literature Review,” *International Journal of Embedded Systems and Applications (IJESA),* vol. 2, no. 2, 2012.

[8] D. P. Jayanto, “Rancang Bangun Back-End “SIAP”: Sistem Informasi Aspirasi Dan Pengaduan Masyarakat Berbasis Web Dengan Menggunakan Metode Microservice Springboot,” *Thesis,* 2017.

[9] Goetz, S. Benjamin, B. Daniel, H. Dennis, Christian Einberger, B. Peter dan Thomas, “Challenges of Production Microservices,” *Procedia CIRP,* vol. 67, pp. 167-172, 2018.

[10] Microsoft, “What is ASP.NET Core,” 2020. [Online]. Available: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core. [Diakses 5 28 2020].

[11] S. Software, “API Documentation & Design Tools for Teams,” SmartBear Software, 2020. [Online]. Available: https://swagger.io/. [Diakses 5 29 2020].

[12] Y. Bassil, “A Simulation Model for the Waterfall Software Development Life Cycle,” *International Journal of Engineering & Technology (iJET),* vol. 2, no. 5, 2012.

[13] W. W. W. Consortium, “HTTP/1.1: Method Definitions,” [Online]. Available: https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html. [Diakses 13 6 2020].

[14] France dan Robert, Advanced Praise for The Unified Modeling Language Reference Manual, Reading, MA, USA: Addison-Wesley, 2004.

LAMPIRAN A. DAFTAR ENDPOINT

No	Nama	HTTP Method	Endpoint	Deskripsi
1	Register	POST	api/user/register	Membuat user saat user login
2	Login	POST	api/user/login	Otentikasi user
2	Get Users By Id	GET	api/user/{id}	Menyajikan data untuk satu user
3	Update Users	PUT	api/user/{id}	Memperbarui data untuk satu user
4	Get Offers By Id	GET	api/offer/{id}	Menyajikan data untuk satu penawaran
5	Get Contents By Id	GET	api/content/{id}	Menyajikan data untuk satu konten
6	Like Offers	PUT	api/offer/{id}/like	Memperbarui data ketika user menyukai satu penawaran
7	Dislike Offers	PUT	api/offer/{id}/unlike	Memperbarui data ketika user tidak suka pada satu penawaran

8	Get All Active Offers	GET	api/offer/active	Menyajikan semua penawaran yang aktif
9	Update Users Bank	PUT	api/user/{id}/bank	Memperbarui data bank pada satu user
10	Update Users Address	PUT	api/user/{id}/address	Memperbarui data alamat pada satu user
11	Update Users Referral	PUT	api/user/{id}/update/referral	Memperbarui data kode referral pada satu user
12	Get User Referral	GET	api/user/{id}/referral	Menyajikan data kode referral pada satu user
13	Get Reward Lottery	GET	api/sales/{id}/lottery	Menyajikan data hasil undian hadiah
14	Get Voucher	GET	api/sales/{id}/voucher	Menyajikan data kode referral saat visitors melakukan klaim voucher

15	Claim Offers	POST	api/sales/claim	Membuat data penawaran saat visitors melakukan klaim voucher
16	Share Offers	GET	api/share/{id}/share/{status}	Menyajikan data saat user menyebarkan penawaran

LAMPIRAN B. DESKRIPSI ENTITAS

Entitas	Deskripsi
Users	Entitas yang digunakan untuk menyimpan data users yang mendaftar pada aplikasi. User dibedakan menjadi tiga *role* yaitu merchants, fans dan visitors
Merchants	Entitas yang digunakan untuk menyimpan data merchants. Pada penelitian ini terbatas pada satu merchant saja
Offers	Entitas yang digunakan untuk menyimpan data penawaran yang dibuat oleh merchant
Sales	Entitas yang digunakan untuk menyimpan semua transaksi. Entitas Sales digunakan saat visitors mendapatkan voucher, merchant melakukan validasi voucher, dan fans mendapatkan hadiah
Contents	Entitas yang digunakan untuk menyimpan data konten yang dibuat oleh merchant. Data pada entitas ini digunakan untuk melakukan share pada media sosial
PinOffers	Entitas yang digunakan untuk menyimpan data penawaran yang ditandai oleh fans
HideOffers	Entitas yang digunakan untuk menyimpan data penawaran yang diarsipkan oleh fans
Ofers Categories	Entitas yang digunakan untuk menyimpan jenis kategori penawaran.

LAMPIRAN C. ATRIBUT ENTITAS

Nama Entitas	Atribut	Deskripsi	Tipe Data
Users	id	Primary Key	long
	referral	Kode referral milik fans	varchar
	name	Nama user	varchar
	role	Terdapat 3 jenis role seperti yang dijabarkan pada Analisa kebutuhan aktor	int
	email	Email user	varchar
	phone	Nomor telepon user	varchar
	status	Status user yang menandakan bahwa user aktif atau tidak aktif	varchar
	otp_type	Jenis OTP	varchar
	otp_code	Kode OTP	int
	otp_expired_at	Masa berlaku kode OTP	datetime

	created_at	Waktu kapan user dibuat	datetime
	updated_at	Waktu kapan user merubah profil	datetime
Merchants	id	Primary Key	long
	name	Nama merchant	varchar
	official_url	Situs resmi merchant	varchar
	sender_email	Email pemilik merchant	varchar
	subdomain	Domain merchant pada marketing platform	varchar
	industry_type	Jenis industry merchant	varchar
	setting_reward_review_period	Pengaturan waktu untuk verifikasi voucher yang ditukarkan	int
	owner_user_id	Foreign Key tabel User untuk id pemilik merchant	long
Offers	id	Primary Key	long
	title	Judul penawaran	varchar

	description	Deskripsi penawaran	varchar
	status	Status penawaran, apakah aktif atau tidak aktif	varchar
	expired_at	Waktu keduluarsa penawaran	datetime
	friend_reward_type	Jenis benefit untuk visitor	varchar
	friend_reward_label	Label jenis benefit	varchar
	friend_reward_discount	Nominal benefit	int
	friend_reward_discount_is_percent	Benefit berbentuk prosentase atau tidak	bool
	friend_reward_min_order	Minimal pembelian agar mendapat diskon	int
	friend_reward_expired_at	Waktu keduluarsa penawaran	datetime
	fan_reward_type	Jenis benefit untuk fans	varchar
	fan_reward_label	Label jenis benefit	varchar
	fan_reward_amount	Nominal benefit	int

	fan_reward_amount_is_percent	Benefit berbentuk prosentase atau tidak	bool
	fan_rule_min_referral	Minimal referral yang dilakukan agar fans mendapatkan hadiah	int
	fan_rule_min_friend_referral	Minimal referral yang dilakukan temannya fans agar fans mendapatkan hadiah	int
	allow_user_contents	Apakah fans boleh membuat konten	bool
	creator_reward_type	Jenis hadiah yang diberikan untuk kreator konten	string
	creator_reward_amount	Nominal hadiah yang diberikan untuk kreator konten	int
	creator_reward_amount_is_percent	Benefit berbentuk prosentase atau tidak	bool
	offer_category_id	Foreign Key tabel OfferCategory	long
Sales	id	Primary Key	long

	coupon_code	Kode kupon yang didapatkan visitors	string
	coupon_checkout_at	Waktu voucher ditukarkan pada merchant	datetime
	amount	Nominal transaksi pembelian	long
	expired_at	Waktu keduluarsa penawaran	datetime
	created_at	Waktu dibuatnya penawaran	datetime
	updated_at	Waktu dirubahnya penawaran	datetime
	offer_id	Foreign Key tabel Offers	long
	referral_user_id	Foreign Key tabel Users untuk fans	long
	buyer_user_id	Foreign Key tabel Users untuk visitors	long
	checkout_staff_user_id	Foreign Key tabel Users untuk staff merchant yang melayani pembelian	long
	mark_referred_staff_user_id	Foreign Key tabel Users yang melakukan validasi penawaran	long
Contents	id	Primary Key	long

	title	Judul konten	varchar
	description	Deskripsi konten	varchar
	image	Gambar konten	varchar
	video	Video konten	varchar
	fan_id	Foreign Key tabel Users untuk fans	long
	offer_id	Foreign Key tabel Users untuk penawaran	long
PinOffers	created_at	Waktu dibuatnya penawaran	datetime
	user_id	Foreign Key tabel Users untuk fans	long
	offer_id	Foreign Key tabel Offers untuk penawaran	long
HideOffers	created_at	Waktu dibuatnya penawaran	datetime
	user_id	Foreign Key tabel Users untuk fans	long
	offer_id	Foreign Key tabel Offers untuk penawaran	long

OffersCategories	id	Primary Key	long
	title	Judul kategori penawaran	varchar

BIODATA PENULIS

Penulis lahir di Madiun, 24 Desember 1998. Penulis menempuh Pendidikan formal di SD Islam Lukman Al-Hakim Surabaya, SMPN 19 Surabaya, dan SMAN 2 Surabaya. Penulis melanjutkan studinya dalam mencari gelar sarjana di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Penulis mulai berminat pada bidang pengembangan aplikasi *web* semenjak mengikuti perkuliahan Pemrograman Berbasis Web. Penulis pernah menyelesaikan aplikasi permainan menggunakan teknologi web sebagai tugas besar dari mata kuliah Pemrograman Berbasis Web. Penulis juga pernah menyelesaikan proyek pembuatan sistem monitoring budaya perusahaan berbasis web untuk PT Transforma Dinamika Unggul. Pada bidang non-akademis, penulis telah mengikuti berbagai macam kegiatan kepanitiaan dan organisasi pada lingkungan kampus. Penulis pernah menjadi staff Internal Affair di HMSI. Penulis juga pernah menjabat sebagai ketua divisi minat bakat pada BEM FTIK untuk menyalurkan minat penulis pada bidang non akademis terutama pada bidang olah raga. Penulis dapat dihubungi melalui *e-mail*: ihsan.faraabi@gmail.com

(Halaman Ini Sengaja Dikosongkan)