SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

RANCANG BANGUN *REST-API* PADA SISTEM INFORMASI PENJUALAN MENGGUNAKAN *EXPRESS JS* PADA TOKO KUE VARIANT CAKE

TUGAS AKHIR

MUTIA MUTHMAINNAH
0110220074

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI
DEPOK
AGUSTUS 2024

STT TERPADU
NURUL FIKRI

SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

RANCANG BANGUN *REST-API* PADA SISTEM INFORMASI PENJUALAN MENGGUNAKAN *EXPRESS JS* PADA TOKO KUE VARIANT CAKE

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TINGGI TEKNOLOGI TERPADU NURUL FIKRI

DEPOK

AGUSTUS 2024

Halaman Pernyataan Orisinalitas

Skripsi/Tugas Akhir ini adalah hasil karya penulis, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama : Mutia Muthmainnah

NIM : 0110220074

Depok, Senin, 22 Juli 2024

Tanda Tangan

Mutia Muthmainnah

STT - NF

Halaman Pengesahan

Skripsi/Tugas Akhir ini diajukan oleh :

Nama : Mutia Muthmainnah

NIM : 0110220074

Program Studi : Sistem Informasi

Judul Skripsi : Rancang Bangun *REST-API* pada Sistem Informasi Penjualan Menggunakan *Express JS* pada Toko Kue Variant Cake

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri

DEWAN PENGUJI

Pembimbing	Penguji
(Imam Haromain S.Si., M.Kom.)	(Salman El Farisi S.Kom., M.Kom.)

STT - NF

Ditetapkan di : Depok

Tanggal : 22 Juli 2024

Kata Pengantar

Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi/Tugas Akhir ini. Penulisan skripsi/Tugas Akhir ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana komputer Program Studi Teknik Informatika pada Sekolah Tinggi Teknologi Terpadu Nurul Fikri Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan skripsi/tugas akhir ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Allah SWT. atas rahmat dan keberkahan-Nya yang telah mengiringi setiap langkah perjalanan saya dalam menyelesaikan penelitian ini.
2. Seluruh anggota keluarga yang telah memberikan dorongan baik secara moril maupun materil, terutama Kedua Orang tua yang selalu memberikan ketulusan doanya dan telah memotivasi penulis sehingga dapat menyelesaikan perkuliahan ini.
3. Bapak Dr. Lukman Rosyidi, ST., MM., MT. selaku Dosen Pembimbing Akademik sekaligus Ketua Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
4. Ibu Tifanny Nabarian, S.Kom., M.T.I selaku Ketua Program Studi Teknik Informatika Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
5. Bapak Imam Haromain S.Si., M.Kom. selaku Dosen Pembimbing Tugas Akhir penulis dalam menyelesaikan penulisan ilmiah ini.
6. Para Dosen di lingkungan Sekolah Tinggi Teknologi Terpadu Nurul Fikri yang telah membimbing penulis dalam menuntut ilmu yang telah diberikan.
7. Para pembimbingi Studi Independen Hacktiv8, yang telah memberikan ilmu dan pembelajaran terkait *Node Js for Back End Web Developer* sehingga penulis dapat menyelesaikan Tugas Akhir.
8. UIUX Team yaitu Bambang Triatmaja, Fikri Pratama Al Fajri, Eka Amelia Putri, Meutia Farah Hidayah dan Pretty Nada Cahaya Irawan yang telah membersamai langkah penulis serta memberikan dukungan, Do'a, kritik dan saran kepada penulis dalam setiap proses penyusunan penelitian ini.

STT-NF

9. Sahabat penulis yaitu Selviana Ardiyani dan Dea Amelia Putri yang telah menjadi tempat berkeluh kesah, saling menyemangati, dan mendukung selama proses perkuliahan dan penyelesaian penelitian ini.
10. Kepada Mutia Muthmainnah, yang telah menepikan ego dan memilih kembali bangkit untuk bertanggung jawab dalam menyelesaikan apa yang telah dimulai. Terima kasih karena terus berusaha dan tidak menyerah, serta senantiasa menikmati setiap proses yang bisa dibilang tidak mudah.

Dalam penulisan ilmiah ini tentu saja masih banyak terdapat kekurangan-kekurangan yang mungkin disebabkan oleh keterbatasan kemampuan dan pengetahuan yang penulis miliki. Walaupun demikian, penulis telah berusaha menyelesaikan penulisan ilmiah ini sebaik mungkin. Oleh karena itu apabila terdapat kekurangan di dalam penulisan ilmiah ini, dengan rendah hati penulis menerima kritik dan saran dari pembaca.

Akhir kata, penulis berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 22 Juli 2024

Mutia Muthmainnah

STT - NF

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Sekolah Tinggi Teknologi Terpadu Nurul Fikri, saya yang bertanda tangan dibawah ini:

Nama : Mutia Muthmainnah

NIM : 0110220074

Program Studi : Teknik Informatika

Jenis karya : Skripsi / Tugas Akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada STT-NF **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty - Free Right*)** atas karya ilmiah saya yang berjudul :

RANCANG BANGUN *REST-API* PADA SISTEM INFORMASI PENJUALAN MENGGUNAKAN *EXPRESS JS* PADA TOKO KUE VARIANT CAKE

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini STT-NF berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*Database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : Senin, 22 Juli 2024

Yang Menyatakan

Mutia Muthmainnah

STT - NF

Abstrak

Nama : Mutia Muthmainnah

NIM : 0110220074

Program Studi : Teknik Informatika

Judul : Rancang Bangun *REST-API* pada Sistem Informasi Penjualan Menggunakan *Express JS* pada Toko Kue Variant Cake

Toko Kue Variant Cake, yang didirikan sejak tahun 2019, menjual berbagai macam makanan manis seperti *brownies*, bolu, dan *dessert box*. Penjualan dilakukan secara offline di rumah dan online melalui Instagram yang diarahkan ke *WhatsApp* untuk pemesanan. Proses ini dianggap kurang efektif karena pelanggan tidak dapat langsung membeli produk, dan transaksi memakan waktu cukup lama. Selain itu, pengelolaan data produk masih manual menggunakan buku, yang rentan terhadap kehilangan data dan dapat mengganggu pengelolaan stok serta keuangan. Untuk mengatasi masalah ini, diperlukan sistem yang dapat meningkatkan efisiensi proses jual beli. Penulis memilih untuk merancang bagian backend sistem dengan menggunakan metode pengembangan *waterfall programming* dan mengimplementasikan *Rest-API* dengan *framework Express.js* serta *database PostgreSQL*. Hasil pengujian menggunakan metode *black box testing* menunjukkan bahwa *Rest-API* Sistem Informasi Penjualan memenuhi seluruh persyaratan fungsional dengan persentase keberhasilan 100% dari 25 kasus uji. Keberhasilan ini menunjukkan bahwa *Rest-API* dirancang dan diimplementasikan dengan baik, memastikan semua fungsionalitas bekerja tanpa kesalahan. *Rest-API* ini diharapkan dapat dimanfaatkan pada berbagai platform, memudahkan pengembangan aplikasi berbasis *web* maupun *mobile*, serta memungkinkan integrasi yang mulus dengan aplikasi eksternal.

Kata kunci : *Rest-Api, Express JS*, Sistem, *Waterfall*

Abstract

Name : *Mutia Muthmainnah*

NIM : 0110220074

Study Program : *Informatics Engineering*

Title : *Design Of A Rest-Api Sales Information System Using Express Js On A Variant Cake Shop*

Variant Cake Shop, established in 2019, sells a variety of sweets such as brownies, sponge cakes, and dessert boxes. Sales are conducted offline at home and online through Instagram which is directed to WhatsApp for ordering. This process is considered ineffective because customer s cannot directly buy products , and transactions take a long time. In addition, product data management is still manual using books, which is prone to data loss and can interfere with stock and financial management. To overcome this problem, a system that can improve the efficiency of the buying and selling process is needed. The author chose to design the backend of the system by using the waterfall programming development method and implementing Rest-API with the Express.js framework and PostgreSQL database. The test results using the blackbox testing method show that the Sales Information System Rest-API fulfills All functional Requirement s with a Success percentage of 100% from 25 test cases. This Success shows that Rest-API is well designed and implemented, ensuring All functionality works without errors. This Rest-API is expected to be utilized on various platforms, facilitate the development of web-based and mobile applications, and allow seamless integration with external applications.

Keywords: Rest-API, Express JS, System, Waterfall

Daftar Isi

Halaman Pernyataan Orisinalitas iii
Halaman Pengesahan iv
Kata Pengantar v
Halaman Pernyataan Persetujuan Publikasi vii
Abstrak viii
Abstract ix
Daftar Isi x
Daftar Tabel xiii
Daftar Gambar xiv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan dan Manfaat Penelitian 3
1.4 Batasan Masalah 4
1.5 Sistematika Penulisan 4
BAB II KAJIAN LITERATUR 5
2.1. Jual Beli 5
2.2. Sistem Informasi 5
2.3. *Web Service* 6
2.4. *Rest-API* 6
2.5. *Postman* 7
2.6. *PostgreSQL* 7
2.7. *Json Web token* 7
2.8. *Express JS* 8

STT - NF

2.9. *Model View Controller* 8
2.10. *Waterfall* 9
2.11. *Unified Modeling Language (UML)* 9
2.12. *Black Box Testing* 10
2.13. Penelitian Terkait 10
BAB III METODOLOGI PENELITIAN 14
3.1. Tahapan Penelitian 14
3.2. Rancangan Penelitian 17
3.2.1 Jenis penelitian 17
3.2.2 Metode Analisis Data 17
3.2.3 Metode Pengumpulan Data 18
3.2.4 Metode Pengujian 18
3.2.5 Metode Implementasi dan Evaluasi 19
3.2.6 Lingkungan Pengembangan 19
BAB IV IMPLEMENTASI DAN EVALUASI 21
4.1. Analisis Kebutuhan 21
4.1.1. Analisis Sistem 21
4.1.2. *User Requirement* 21
4.1.3 *Use case Diagram* 23
4.2. Perancangan Sistem 24
4.2.1 *Entity Relationship Diagram* 24
4.2.2. Rancangan Arsitektur *Rest-API* 26
4.2.3. Rancangan Pengujian 42
4.3. Implementasi Fungsi Aplikasi 48
4.3.1. Fungsi *Register* 48
4.3.2. Fungsi *Login* 50

STT - NF

4.3.3. Fungsi *Categories* 54
4.3.4. Fungsi *Products* 57
4.3.5. Fungsi *Transaction History*.....61
4.4. Pengujian dan Evaluasi.....65
4.4.1 Black Box Testing.....65
BAB V KESIMPULAN DAN SARAN..... 74
5.1. Kesimpulan.....74
5.2. Saran 74
DAFTAR PUSTAKA 76

STT - NF

Daftar Tabel

Tabel 4. 1 Kategori Kebutuhan User.....21
Tabel 4. 2 Kategori User Aplikasi.....22
Tabel 4. 3 *List Endpoint*.....27
Tabel 4. 4 *Mapping Model*.....28
Tabel 4. 5 *Mapping Controller*.....28
Tabel 4. 6 Rancangan Pengujian *Black Box Testing*.....43
Tabel 4. 7 Pengujian *Black Box Testing*.....65

STT - NF

Daftar Gambar

Gambar 2. 1 Metode *Waterfall* 9
Gambar 3. 1 Tahapan Penelitian 14
Gambar 4. 1 *Use case Diagram - Role Admin* 23
Gambar 4. 2 *Use case Diagram - Role Customer* 23
Gambar 4. 3 *Entity Relationship Diagram* 24
Gambar 4. 4 Struktur Direktori 29
Gambar 4. 5 *Script Config.js* 30
Gambar 4. 6 *Script User.js* 31
Gambar 4. 7 *Script Categoty.js* 32
Gambar 4. 8 *Script Product.js* 33
Gambar 4. 9 *Script Transactionhistory.js* 34
Gambar 4. 10 *Script Index.js* 35
Gambar 4. 11 *UserController* 36
Gambar 4. 12 *CategoryController* 37
Gambar 4. 13 *TransactionRouter* 40
Gambar 4. 14 *Index.js* 41
Gambar 4. 15 *Register Admin* 48
Gambar 4. 16 *Register Admin Success* 49
Gambar 4. 17 *Database Table Users* 49
Gambar 4. 18 *Register Customer* 49
Gambar 4. 19 *Register Success* 50
Gambar 4. 20 *Data Database Tabel Users* 50
Gambar 4. 21 *Login* 51
Gambar 4. 22 *Json Web token* 51
Gambar 4. 23 *Update Users* 52
Gambar 4. 24 *Update Users* – Sukses 52
Gambar 4. 25 *Database Users - Before Update* 52
Gambar 4. 26 *Database Table User - After Updated* 53
Gambar 4. 27 *Top-Up Saldo* 53
Gambar 4. 28 *Top-Up Saldo* Sukses 53

STT - NF

Gambar 4. 29 *Get Users*.....54
Gambar 4. 30 *Create categories*.....54
Gambar 4. 31 *Create categories* Sukses.....55
Gambar 4. 32 *Update Categories*.....55
Gambar 4. 33 *Update Categories Succes*.....56
Gambar 4. 34 *Database Table Categories*.....56
Gambar 4. 35 *Get All Categories*.....57
Gambar 4. 36 *Create Products*.....58
Gambar 4. 37 *Create Products Success*.....58
Gambar 4. 38 *Update Products*.....59
Gambar 4. 39 *Update Products Success*.....59
Gambar 4. 40 *Database Table Products*.....59
Gambar 4. 41 *Update CategoryId*.....60
Gambar 4. 42 *Update Categor Success*.....60
Gambar 4. 43 *Get All Products*.....61
Gambar 4. 44 *Create Transactions*.....62
Gambar 4. 45 *Transaction success*.....62
Gambar 4. 46 *Get All Transactions*.....63
Gambar 4. 47 *Get Transaction History - User*.....63
Gambar 4. 48 *Database Transaction History*.....64

STT - NF

BAB I
PENDAHULUAN

1.1 Latar Belakang

Jual beli adalah Perjanjian yang dapat ditegakkan secara hukum antara penjual yang memberikan produk, dengan pembeli yang membayar barang yang dijual [1]. Jual beli merupakan salah satu aktivitas ekonomi yang paling mendasar dan penting dalam kehidupan manusia. Ini memungkinkan individu, perusahaan, dan entitas lain untuk memperoleh barang dan jasa yang mereka butuhkan atau inginkan, sementara pada saat yang sama memungkinkan produsen dan penjual untuk mendapatkan pendapatan dan keuntungan dari barang atau jasa yang mereka produksi atau tawarkan. Jual beli juga merupakan dasar dari ekonomi pasar, dimana harga dan alokasi sumber daya ditentukan oleh peminatan dan penawaran.

Toko Kue Variant Cake merupakan toko yang beralamat di Pesona kahuripan 4 Klapa nunggal Cileungsi dan didirikan pada tahun 2019. Toko Kue Variant Cake menjual berbagai jenis makanan manis seperti Brownies, Dessert Box, juga kue lainnya yang memiliki banyak peminat. Berdasarkan wawancara yang sudah dilakukan dengan pemilik Toko Kue Variant cake, pemilik toko menyatakan bahwa penjualan yang dilakukan saat ini melalui *online* dan *offline.* Penjualan *offline* dilakukan dengan cara membuka toko di rumah. Sedangkan penjualan *online* dilakukan dengan promosi menggunakan media social yaitu Instagram yang mana dari Instagram pelanggan akan diarahkan untuk memesan melalui *WhatsApp*. Hal ini dirasa kurang efektif karena pelanggan tidak bisa langsung membeli produk dan butuh waktu cukup lama untuk proses transaksi. Selain itu pengelolaan data produk pada toko ini juga masih manual menggunakan buku. Hal ini dikhawatirkan akan menimbulkan masalah seperti rentan terhadap kehilangan data yang dapat menyebabkan gangguan dalam pengelolaan stok dan keuangan. Seiring dengan perkembangan teknologi, jual beli *online* telah menjadi salah satu tren yang dominan dalam perdagangan. Ini memberikan kenyamanan bagi konsumen untuk berbelanja dan mempermudah penjual dalam mengatasi

STT-NF

permasalahan ketika berjualan secara langsung seperti pengelolaan atau manajemen produk yang masih manual menggunakan kertas.

Berdasarkan masalah yang ada, dibutuhkan perancangan sistem yang dapat merubah proses jual beli menjadi lebih efisien. Salah satu solusi yang dapat diterapkan adalah pengembangan sistem yang terdiri dari modul *front-end* dan *Backend.* Untuk mendukung penelitian ini penulis memilih untuk membangun bagian *Backend* sistem. Dengan memusatkan perhatian pada bagian *Backend*, diharapkan sistem yang dihasilkan dapat memberikan solusi yang tepat dan efisien untuk meningkatkan kinerja proses jual beli.

Backend yang bisa disebut juga dengan *server*-side pada dasarnya digunakan sebagai wadah untuk proses sebuah sistem atau aplikasi yang sedang berjalan, proses yang berjalan pada *Backend* meliputi *create*, *read*, *update*, dan *delete* [2]. Pemanfaatan teknologi *backend* seperti *Rest-API*, *PostgreSQL*, *Express JS* dan *Json Web token* memiliki peran yang penting dalam mengembangkan sektor jual beli yang modern dan efisien. Dalam proses jual beli *REST API* dapat digunakan untuk menyediakan akses data kepada pengguna. Penting juga untuk memiliki basis data yang dapat menangani jumlah data yang besar dengan cepat dan aman seperti *PostgreSql*. *Express.js* dapat digunakan untuk mengelola permintaan *HTTP* dari berbagai klien. Oleh karena itu Dibuatlah *Rest-API* untuk Sistem Informasi Penjualan menggunakan *Express JS* sebagai solusi permasalahan jual beli *offline* dimana Tujuan aplikasi dibuat yaitu untuk memberikan solusi dalam menemukan serta mengolah data produk penjualan dengan cara mengimplementasikan fungsi yang ada pada Sistem Informasi Penjualan sehingga memudahkan pengguna untuk bertransaksi.

1.2 Rumusan Masalah

1. Bagaimana merancang *REST-API* pada Sistem Informasi Penjualan agar berjalan dengan baik menggunakan *Express.js* pada Toko Kue Variant Cake?

2. Apakah rancangan *Rest-API* pada Sistem Informasi Penjualan bisa mempermudah dalam pengelolaan data produk penjualan dan proses jual beli?

1.3 Tujuan dan Manfaat Penelitian

Dari rumusan masalah, diharapkan penelitian ini dapat memberikan pengalaman yang lebih baik kepada pengguna untuk mempermudah proses jual beli. Adapun tujuan penelitian sebagai berikut:

1. Merancang *Rest-API* pada Sistem Informasi Penjualan menggunakan *Express JS* pada Toko Kue Variant Cake.
2. Rancangan *Rest-API* pada Sistem Informasi Penjualan dapat mempermudah dalam pengelolaan data produk penjualan dan proses jual beli.

Pada Penelitian Rancang Bangun Aplikasi *Rest-API* Sistem Informasi Penjualan Menggunakan *Express JS* memiliki manfaat yang dapat digunakan oleh pengguna yaitu:

1. *Rest-API* bersifat cepat dan ringan karena menggunakan standar HTTP yang dapat bekerja dengan berbagai format data salah satunya yaitu JSON.
2. API memungkinkan pengembang untuk menyesuaikan aplikasi sesuai dengan kebutuhan spesifik pengguna atau bisnis tanpa harus mengubah seluruh sistem.
3. *REST-API* memungkinkan sistem informasi penjualan Toko Kue Variant Cake dapat diakses dari berbagai platform, termasuk aplikasi web, aplikasi seluler Android dan iOS, serta perangkat IoT.
4. Dokumentasi dan analisis yang dihasilkan dalam penelitian ini dapat menjadi sumber daya berharga bagi akademisi dan praktisi yang ingin mempelajari dan menerapkan arsitektur *Rest-API* dalam proyek pengembangan *web* mereka.
5. Penelitian ini dapat memicu penelitian lebih lanjut dan inovasi di bidang teknologi *web* dan *e-commerce.*

6. Memperkaya literatur dan hasil penelitian di bidang pengembangan *Rest-API*

1.4 Batasan Masalah

Pada penelitian ini, terdapat Batasan masalah pada fungsi dan kegunaan aplikasi. Adapun batasan masalah penelitian sebagai berikut:

1. Tahapan *testing endpoint-endpoint* yang sudah dibuat dilakukan menggunakan aplikasi *postman*.
2. Hasil luaran aplikasi berupa *API* belum ada tampilan *Front-End.*
3. Pengujian sistem hanya dilakukan menggunakan *Black Box Testing.*

1.5 Sistematika Penulisan

Dalam menyusun penelitian ini, untuk memudahkan pemahaman mengenai tugas akhir secara umum, maka penelitian ini menggunakan sistematika penulisan sebagai berikut:

BAB 1 PENDAHULUAN, bab ini menjelaskan tentang Latar belakang penelitian, Rumusan Masalah, Tujuan dan Manfaat penelitian, serta Sistematika penulisan yang digunakan pada penelitian ini.

BAB II KAJIAN LITERATUR, bab ini menjelaskan tentang definisi dan teori yang berkaitan dengan penelitian, menganalisis hasil penelitian yang berkaitan, dan menjabarkan penjelasan mengenai penelitian lain yang telah dilakukan oleh para ahli sebelum penelitian ini dibuat.

BAB III METODOLOGI PENELITIAN, bab ini menjelaskan tentang bagaimana alur penelitian, rancangan penelitian, metode penelitian, serta metode pengujian yang digunakan.

BAB IV IMPLEMENTASI DAN EVALUASI, bab ini menjelaskan tentang Rancang Bangun *Rest-API* Toko Belanja menggunakan *Express JS* yang berupa hasil dari perancangan sistem *Rest-API* yang dilakukan penulis serta membahas hasil dari pengujian dan evaluasi dari *Rest-API.*

BAB V KESIMPULAN DAN SARAN, Bab ini berisi tentang kesimpulan yang menjawab tujuan dari penelitian dibuat serta saran dari penulis untuk penelitian selanjutnya.

STT - NF

BAB II
KAJIAN LITERATUR

2.1. Jual Beli

Jual beli merupakan salah satu aktivitas ekonomi yang paling mendasar dan penting dalam kehidupan manusia. Ini memungkinkan individu, perusahaan, dan entitas lain untuk memperoleh barang dan jasa yang mereka butuhkan atau inginkan, sementara pada saat yang sama memungkinkan produsen dan penjual untuk mendapatkan pendapatan dan keuntungan dari barang atau jasa yang mereka produksi atau tawarkan. Jual beli juga merupakan dasar dari ekonomi pasar, dimana harga dan alokasi sumber daya ditentukan oleh peminatan dan penawaran.

Pada umumnya proses jual beli lebih banyak dilakukan secara langsung, misalnya di warung, toko belanja, pasar tradisional, dan pasar modern. Namun Seiring dengan perkembangan teknologi jual beli *online* telah menjadi salah satu tren yang dominan dalam perdagangan. Jual beli online mengacu pada pembelian dan penjualan barang atau jasa dengan menggunakan metode elektronik, khususnya internet atau online. Hal ini dapat membuat belanja lebih nyaman bagi pelanggan sekaligus memudahkan penjual untuk mengatasi masalah saat berjualan secara langsung [3].

2.2. Sistem Informasi

Sistem adalah sekumpulan komponen yang saling berhubungan satu sama lain dan suatu kegiatan yang bertujuan untuk mencapai suatu tujuan tertentu. Informasi merupakan data mentah yang telah diproses sehingga menghasilkan sesuatu yang berguna dalam mengambil sebuah keputusan [4] Sistem informasi merupakan jenis sistem komunikasi yang dapat mewakili data serta dapat diproses sebagai jenis memori sosial. Sistem informasi juga dapat dilihat sebagai bahasa semi formal yang dapat membantu penggunanya dalam mengambil keputusan dan tindakan [5].

Tujuan dibuatnya sistem informasi yaitu untuk meningkatkan produktivitas dan efisiensi melalui integrasi proses juga menggunakan teknologi informasi yang tepat. Sistem informasi berperan dalam mengumpulkan data yang relevan dengan aktivitas dan operasi yang yang dilakukan secara terstruktur dan terorganisir agar mudah diakses di masa mendatang. Sistem informasi juga berfungsi untuk mengeluarkan informasi dalam berbagai macam bentuk seperti analisis, laporan, atau representasi visual dari data yang dikumpulkan dan disimpan.

2.3. *Web Service*

Web service merupakan aplikasi yang berisi kumpulan *Database* dan program perangkat lunak yang diakses secara *remote* melalui perantara tertentu, biasanya dalam bentuk *URL Web*, yang dapat mengatasi masalah interoperabilitas dan integrasi sistem yang berbeda berupa *URL web*. Perbedaan terletak pada interaksi yang diberikan *web service* itu sendiri, karena *URL web* hanya berisi kumpulan informasi, perintah juga konfigurasi kode program [6].

Web service dapat di definisikan sebagai kumpulan fungsionalitas yang dapat diakses melalui *internet protocol (IP)* standar. *Web service* banyak digunakan dalam aplikasi sistem terdistribusi yang heterogen, mulai dari sistem operasi hingga model objek. *Web service* adalah entitas yang dapat diprogram dan menyediakan berbagai jenis fungsionalitas, seperti informasi, aplikasi logis, dll. melalui berbagai media seperti *Hypertext Transfer protocol (http)* dan *XML* [7].

2.4. *Rest-API*

API adalah singkatan dari *Application Programming Interface* dan merupakan kumpulan kode yang memungkinkan dua atau lebih program untuk berkomunikasi. *API* adalah penghubung/perantara antara berbagai aplikasi, di dalam atau lintas *platform*. Sedangkan *REST* atau *Representational State Transfer* yaitu gaya arsitektur yang menyediakan standar antar komputer dan memfasilitasi komunikasi antar sistem

menggunakan *HTTP/HTTPS* untuk transmisi data. Salah satu protokol *API* yang sering digunakan yaitu *REST-API*, dimana *REST-API* ini merupakan Arsitektur atau metodologi komunikasi *API* yang bersifat *stateless* dan menggunakan *HTTP/HTTPS* dalam transmisi data [8].

2.5. *Postman*

Postman adalah sebuah *platform* untuk membangun dan menggunakan *API*. *Postman* mempermudah setiap langkah dalam siklus hidup *API* dan memperlancar kolaborasi sehingga dapat memberikan kemudahan dan mempercepat proses pembuatan *API* [9].

Menurut Antares (2020) *postman* merupakan aplikasi rest client untuk melakukan pengujian *REST API* , Aplikasi ini sering digunakan oleh pengembang pembuat *API* sebagai *tools* untuk menguji *API* yang telah dibuat. *Postman* dapat digunakan untuk mengumpulkan *API* yang dapat didokumentasikan secara komprehensif untuk projek tertentu. Jika dokumentasi *API* menggunakan *Postman* sudah lengkap, maka pengembangan projek akan lebih mudah karena setiap pengembang dapat memiliki referensi yang jelas untuk penggunaan setiap *API* [10].

2.6. *PostgreSQL*

PostgreSQL adalah sistem basis data rasional objek *open source* yang sangat andal dan telah dikembangkan lebih dari 35 tahun. Sistem ini terkenal karena kehandalannya , mempunyai fitur lengkap dan kinerja tinggi [11].

PostgreSQL memiliki banyak fitur yang dapat membantu pengembang untuk membuat aplikasi, administrator untuk melindungi integritas data dan menciptakan lingkungan yang toleran terhadap kesalahan. *PostgreSQL* dapat membantu dalam mengelola data berapapun ukurannya. Selain gratis juga open source, *postgreSQL* juga sangat dapat dikembangkan.

2.7. *Json Web token*

JWT atau *Json Web token* adalah token dalam bentuk *string* yang panjang dan sangat acak, *Json Web token* memungkinkan sistem untuk

mengautentikasi dan bertukar informasi. Secara umum, *login tid*ak seperti aplikasi *website* biasa di mana kita menggunakan *session* untuk mengingat siapa yang sedang *login* . *JWT* atau token ini seperti *password.* Jadi ketika pengguna berhasil *login* , *server* memberi token. pengguna kemudian akan menyimpan token di penyimpanan lokal atau *cookie browser.* jika pengguna ingin mengakses halaman tertentu, maka pengguna harus menyertakan token tersebut [12]. *JWT* terdiri dari tiga bagian yaitu *header, payload,* dan signature yang dipisahkan oleh titik.

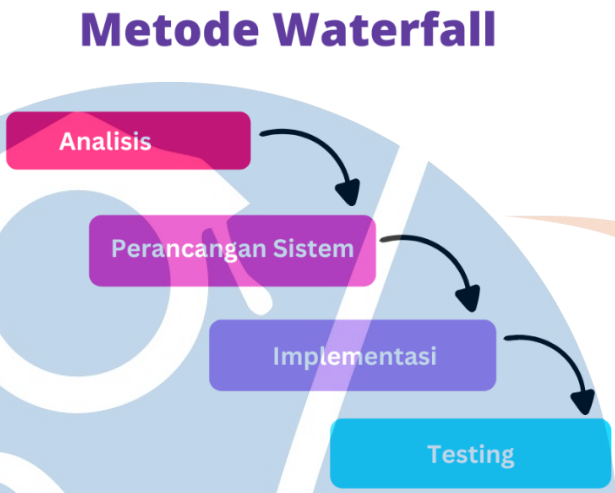
2.8. *Express JS*

Framework merupakan kerangka kerja yang dapat digunakan untuk pengembangan situs web. *Framework* dibuat untuk membantu pengembang *web* menulis baris kode. Penulisan kode akan jauh lebih efisien dengan menggunakan *framework* [13]. *Express JS* merupakan salah satu *framework* turunan dari bahasa pemrograman *JavaScript. Express JS* dirancang fleksibel dan minimalis untuk pengembangan aplikasi *Backend.* Secara fungsionalitas *Express JS* didukung oleh *middleware. Middleware* adalah fungsi *asynchronous* yang dapat mengubah hasil permintaan dan *respons* di *server* [14].

2.9. *Model View Controller*

MVC atau *Model View Controller* merupakan pola arsitektur ketika membuat suatu aplikasi yang dilakukan dengan cara memisahkan kode menjadi tiga komponen utama yaitu *model*, *view*, dan *Controller*. *Model* adalah bagian dari aplikasi yang bertanggung jawab untuk mengelola data, menyiapkan, mengatur serta mengorganisasikan data yang berada di *Database*. *View* adalah bagian aplikasi yang bertanggung jawab untuk menampilkan informasi atau data ke pengguna dalam bentuk *GUI (Graphical user interface)* sehingga pengguna dapat berinteraksi dengan aplikasi. *Controller* adalah bagian aplikasi yang bertanggung jawab untuk menghubungkan dan mengatur *model* dan *view* agar dapat saling terhubung [15].

2.10. *Waterfall*



Gambar 2. 1 Metode *Waterfall*

Menurut A. Adel dan B. Abdullah (2015) Metode *Waterfall* adalah salah satu *SDLC* atau *System Development Life Cycle* yang sudah banyak digunakan untuk pengembangan perangkat lunak. Langkah-langkah eksekusi menggunakan metode ini dilakukan dengan cara berurutan. Tahapan-tahapan pada model *Waterfall* ini meliputi Analisis kebutuhan, desain, pengkodean, pengujian dan pemeliharaan [16].

2.11. *Unified Modeling Language (UML)*

UML atau *Unified Modeling Language* merupakan suatu pemodelan visual yang digunakan sebagai sarana untuk merancang sistem berorientasi objek. *UML* terkadang disebut sebagai bahasa standar untuk memvisualisasikan, mendesain, dan mendokumentasikan sistem, serta bahasa standar untuk menulis spesifikasi perangkat lunak. Dengan menggunakan *UML* diharapkan pengembang dapat lebih mudah dalam pengembangan perangkat lunak serta memenuhi semua kebutuhan

STT - NF

pengguna secara lengkap, efektif, dan tepat. Terdapat beberapa pemodelan *UML* seperti *Use case Diagram, class diagram,* dan *Activity Diagram* [17] berikut penjelasannya:

1. *Use case Diagram*

   Use case atau diagram *Use case* adalah pemodelan untuk mengGambar kan interaksi antara sistem dan aktor pada sebuah sistem informasi yang akan dibuat.
2. *Class Diagram*

   Class Diagram merupakan pemodelan untuk mengGambar kan kelas-kelas yang akan dibuat untuk membangun sebuah sistem.
3. *Activity Diagram*

   Activity Diagram merupakan pemodelan yang mengGambar kan aktifitas atau proses yang terjadi pada sebuah sistem.

2.12. *Black Box Testing*

Testing merupakan sebuah aktivitas uji coba yang bertujuan untuk dapat mengidentifikasi, menguji dan mengevaluasi dari sisi kualitas, fungsi, serta kinerja suatu sistem atau aplikasi, apakah ditemukan *bug* atau tidak? Apakah aplikasi sudah berfungsi sesuai yang direncanakan atau tidak? Jika sudah sesuai, maka *testing* berhasil. Namun jika tidak, perlu adanya analisis terkait pengujian yang bersangkutan.

Behavioral Testing atau yang biasa disebut dengan *Black Box Testing* merupakan pengujian yang berfokus pada masukan dan keluaran program yang bertujuan untuk melihat apakah sistem berjalan dengan baik atau tidak. Salah satu keuntungan dari melakukan *testing* menggunakan *Black Box testing* yaitu Penguji tidak harus memiliki kemampuan suatu bahasa pemrograman ketika ingin melakukan *testing* [18].

2.13. Penelitian Terkait

Penelitian terkait membahas perihal beberapa penelitian terdahulu yang memiliki kaitan yang sama dengan penelitian penulis.

Tabel 2. 1 Penelitian Terkait

No.	Nama dan Tahun	Judul	Topik	Subjek	Hasil
1	Siti Sauda, M Barokah 2022	Penerapan *Node JS* dan *PostgreSQL* sebagai *Backend* Pada Aplikasi *Ecommerce* Localla	*Backend Aplikasi Ecommerce Localla*	Masyarakat di Indonesia yang menggunakan aplikasi *ecommerce* Localla	*Rest-API Backend* aplikasi *Ecommerce* Localla
2	Anang Bagus Prakoso, Wayan Santiyasa 2022	Rancang Bangun *Web Service API* Aplikasi Jual Beli Barang Bekas Menggunakan Metode *REST*	*Aplikasi jual beli barang bekas menggunakan metode REST*	Penjual dan Pembeli	*Web Service API* Aplikasi Jual Beli Barang Bekas
3	Eli Nurhayati, Agussalim 2023	Rancang Bangun *Backend API* pada Aplikasi *Mobile* AyamHub Menggunakan *Framework Node JS Express*	*API Aplikasi Mobile AyamHub menggunakan Framework Node JS Express*	Masyarakat umum di Indonesia yang menggunakan Aplikasi *Mobile* AyamHub	*Backend API* Aplikasi *Mobile* AyamHub
4	Nasrul, Achmad Izhar 2023	Pengembangan *Rest-API* Dengan Menggunakan *Express JS* Untuk Mencari Mentor Pribadi	*REST API untuk aplikasi pencarian mentor*	Pelajar dan Mahasiswa	*Rest-API* untuk Aplikasi pencarian mentor
5	Muhammad Fiqri Nugroho, Aji Primajaya,	*Rest-API* Aplikasi Manajemen Toko Menggunakan	*Rest-API Aplikasi Manajemen Toko*	Pengelola Toko Cantika Paint	*Rest-API* Aplikasi Manajemen Toko

	Mohamad Jajuli 2023	*Node JS* Pada Cantika Paint			

1. Penelitian yang berjudul "Penerapan *Node JS* dan *PostgreSQL* sebagai *Backend* Pada Aplikasi *Ecommerce* Localla" yang ditulis oleh Siti sauda dan M Barokah pada tahun 2020. Telah berhasil membuat *Backend* aplikasi *ecommerce Localla* menggunakan *Node JS* dan *PostgreSQL* untuk digunakan oleh masyarakat yang menggunakan aplikasi Localla. Tujuan dari penelitian ini dibuat yaitu agar pencarian informasi produk dan transaksi dapat lebih cepat dan akurat sehingga diharapkan dapat meningkatkan daya dorong masyarakat agar lebih menggunakan produk dalam negeri.
2. Penelitian yang berjudul "Rancang Bangun *Web Service API* Aplikasi Jual Beli Barang Bekas Menggunakan Metode *REST*" ditulis oleh Anang Bagus Prakoso dan Wayan Santiyasa pada tahun 2022. Telah berhasil membuat *Web Service API* Aplikasi Jual Beli Barang Bekas menggunakan *Javascript* dengan memanfaatkan *framework Node JS.* Tujuan dibuatnya penelitian ini yaitu Untuk mendukung proses transaksi data yang dapat digunakan oleh berbagai perangkat.
3. Penelitian yang berjudul "Rancang Bangun *Backend API* pada Aplikasi Mobile AyamHub Menggunakan *Framework Node JS Express*" ditulis oleh Eli Nurhayati dan Agussalim pada tahun 2023. Telah berhasil membuat *Backend API* Aplikasi Mobile AyamHub menggunakan arsitektur *Rest* dan *platform Node JS.* Tujuan dibuatnya penelitian ini yaitu untuk menciptakan *Backend* yang efektif dan berfungsi dengan baik untuk aplikasi mobile AyamHub.
4. Penelitian yang berjudul "Pengembangan *REST API* Dengan Menggunakan *Express JS* Untuk Mencari Mentor Pribadi" dibuat oleh Nasrul dan Achmad Izhar pada tahun 2023. Telah berhasil membuat *Rest-API* untuk Aplikasi pencarian mentor menggunakan *Express JS* dan Metode *Scrum.* Tujuan dibuatnya penelitian ini yaitu untuk mengimplementasikan fungsi pencarian mentor pada aplikasi *private coding* dengan baik sehingga memudahkan pengguna yang ingin mencari mentor pemrograman.

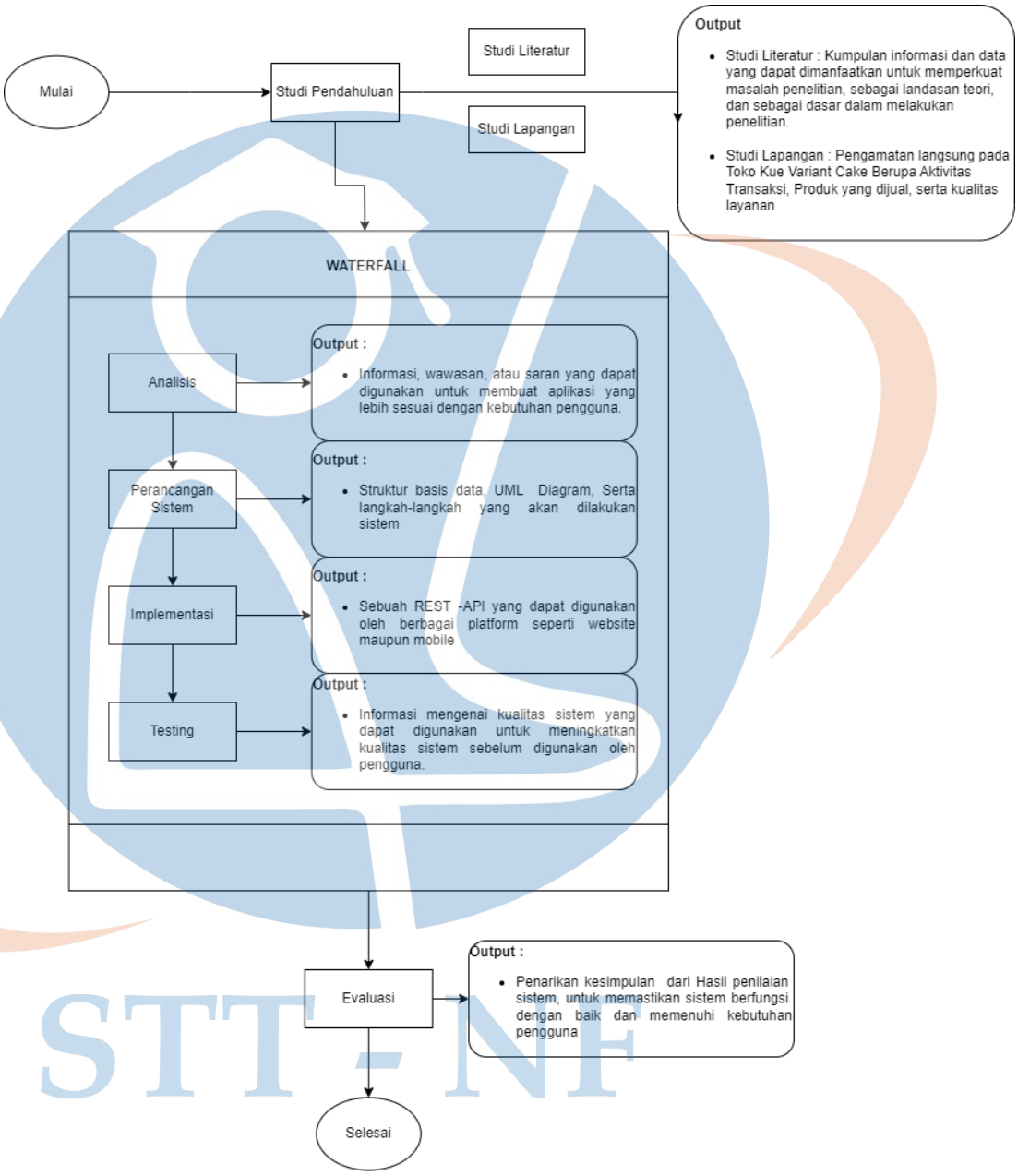
STT - NF

5. Penelitian yang berjudul "*Rest-API* Aplikasi Manajemen Toko Menggunakan *Node JS* Pada Cantika Paint" dibuat oleh Muhammad Fiqri Nugroho, Aji Primajaya, dan Mohamad Jajuli pada tahun 2023. Telah berhasil membuat *Rest-API* Aplikasi Manajemen Toko Menggunakan *Express JS* dan dengan metode *Waterfall* . tujuan penelitian ini dibuat yaitu untuk memudahkan pengembangan aplikasi Manajemen Toko agar dapat membantu toko Cantika Paint dalam memanajemen toko yang lebih baik.

STT - NF

BAB III
METODOLOGI PENELITIAN

3.1. Tahapan Penelitian



Gambar 3. 1 Tahapan Penelitian

STT - NF

Diagram alir yang tersedia pada Gambar 3.1 menggambarkan serangkaian tahapan yang akan dilaksanakan dalam proses penelitian Tugas Akhir. Diagram ini memberikan Gambar an visual mengenai urutan langkah-langkah yang akan diambil dalam penelitian. Berikut penjelasan dari tahapan-tahapan tersebut :

1. **Studi Pendahuluan** : terdiri dari dua bagian utama: Studi Literatur dan Studi Lapangan. Studi Literatur menghasilkan kumpulan informasi dan data yang digunakan untuk memperkuat masalah penelitian, menyediakan landasan teori, dan menjadi dasar dalam melakukan penelitian lebih lanjut. Studi Lapangan melibatkan pengamatan langsung di Toko Kue Variant Cake, mencakup aktivitas transaksi, produk yang dijual, serta kualitas layanan yang diberikan. Gabungan dari kedua *output* ini memberikan pemahaman menyeluruh dan mendalam mengenai kondisi dan kebutuhan toko, yang akan menjadi dasar dalam pengembangan sistem *Rest-API* yang sesuai.Analisis, tahap analisis dilakukan dengan cara Wawancara untuk mencari informasi dan mengidentifikasi permasalahan yang terjadi pada Toko Kue Variant cake. Hasil dari wawancara juga dapat dianalisis untuk men*get*ahui kebutuhan sistem guna mempermudah pembuatan *Rest-API.*
2. **Analisis** : Informasi, wawasan, atau saran yang dikumpulkan dari berbagai sumber dapat digunakan untuk mengembangkan aplikasi yang lebih sesuai dengan kebutuhan pengguna. Informasi ini mencakup data tentang preferensi dan perilaku pengguna, umpan balik pengguna sebelumnya, tren pasar, dan analisis kompetitor. Wawasan ini membantu dalam memahami apa yang diinginkan dan dibutuhkan oleh pengguna, memungkinkan pengembang untuk menyesuaikan fitur. Selain itu, saran dari ahli atau pengguna beta dapat memberikan ide-ide inovatif dan solusi praktis untuk meningkatkan kualitas dan kinerja aplikasi. Dengan memanfaatkan semua informasi ini, aplikasi yang dihasilkan akan lebih relevan, dan memenuhi harapan pengguna.

STT - NF

3. **Perancangan Sistem** : Struktur basis data menentukan bagaimana data disimpan, diorganisir, dan dihubungkan dalam sistem. Diagram *UML* (*Unified Modeling Language*) digunakan untuk mengGambar kan desain sistem, termasuk kelas, objek, dan interaksi antara mereka, memberikan visualisasi yang jelas tentang arsitektur dan alur kerja sistem. Langkah-langkah yang akan dilakukan oleh sistem merujuk pada proses atau alur kerja yang diikuti oleh sistem untuk mencapai tujuan tertentu, seperti menerima *input* dari pengguna, memproses data, dan memberikan *output* yang diinginkan. Pada penelitian ini peneliti hanya menggunakan UML yaitu *Use case* diagram.
4. **Implementasi** : tahap implementasi yaitu Pemrograman sistem dengan mengimplementasikan seluruh kebutuhan yang telah dirancang pada tahap perancangan sistem yang sudah dibuat sebelumnya. Hasil yang diharapkan adalah sebuah *REST-API* yang sesuai dengan spesifikasi yang telah ditetapkan dalam perancangan dan dapat digunakan oleh berbagai platform seperti website maupun mobile.
5. ***Testing*** : tahap *testing* dilakukan untuk menguji sistem yang sudah dibuat menggunakan metode *Black Box testing* untuk memastikan fungsionalitas dari setiap *endpoint Rest-API* sudah berjalan dengan baik. Tahap ini dilakukan guna mengidentifikasi potensi kegagalan atau kekurangan serta memastikan bahwa sistem siap untuk digunakan.
6. **Evaluasi** : Tahap evaluasi dilakukan untuk mengevaluasi hasil-hasil dari pengujian dan menarik kesimpulan dari penilaian proses sebelumnya. Hasil yang diharapkan pada tahap ini yaitu pemahaman yang mendalam tentang kinerja sistem, identifikasi area yang memerlukan perbaikan, serta rekomendasi untuk meningkatkan kualitas dan efisiensi sistem sebelum diperkenalkan kepada pengguna.

STT - NF

3.2. Rancangan Penelitian

3.2.1 Jenis penelitian

Jenis penelitian yang digunakan dalam penelitian ini yaitu jenis penelitian *Research and Development* atau bisa disebut dengan *RnD*. Menurut Sugiono (2016) Penelitian *RnD* adalah pendekatan penelitian yang melibatkan pengembangan produk dan kemudian mengujinya untuk men*get*ahui apakah produk itu layang digunakan. Hal ini didukung oleh pendapat dari Sudaryono (2016) yaitu penelitian *Rnd* adalah metode yang digunakan untuk menghasilkan suatu produk dan menguji produk tersebut untuk dilihat kegunaanya [19]. Jenis penelitian dipilih berdasarkan arah tujuan penelitian yaitu untuk menemukan dan mengembangkan solusi dan menyelesaikan atau meningkatkan suatu hal dari suatu kondisi atau permasalahan dengan suatu cara tertentu.

3.2.2 Metode Analisis Data

Metode analisis data pada penelitian ini menggunakan metode kualitatif yang merupakan suatu pendekatan penelitian yang menggunakan data berupa kalimat tertulis atau lisan, peristiwa-peristiwa, pengetahuan, atau objek studi. Metode ini dipilih berdasarkan hasil pengumpulan data yang didapat melalui wawancara, dan observasi, yang mana hasilnya berupa informasi mengenai pengelolaan toko, kebutuhan toko, serta pengalaman pemilik toko. Informasi ini kemudian diimplementasikan ke dalam sebuah rancangan *Rest-API,* yang selanjutnya diuji dan dievaluasi menggunakan metode pengujian *Blackbox testing* sehingga keberhasilan fungsionalitas *API* tersebut dapat dipastikan. Dengan demikian, penelitian ini tidak hanya menghasilkan pemahaman mendalam mengenai dinamika pengelolaan

toko, tetapi juga memberikan solusi praktis yang teruji dan dapat diimplementasikan secara efektif.

3.2.3 Metode Pengumpulan Data

Metode pengumpulan data yang digunakan dalam penelitian Rancang Bangun *REST-API* Toko Belanja Menggunakan *Express JS* ini ada tiga yaitu :

1. Studi Literatur

   Studi literatur dilakukan dengan tujuan menambah pen*get*ahuan tentang topik penelitian yang dilakukan guna menentukan teori-teori dan metode yang yang tepat dan dapat digunakan dalam proses penelitian melalui jurnal, Buku, artikel, dll.
2. Wawancara

   Wawancara dilakukan di lokasi Toko kue Variant Cake dan melibatkan langsung pemilik Toko. Pada wawancara penelitian ini penulis melakukan pendekatan untuk mendapatkan informasi mengenai pengelolaan toko, kebutuhan toko, serta pengalaman pemilik toko.
3. Observasi

   Dalam penelitian ini observasi dilakukan bertujuan untuk mendapatkan data dan fakta mengenai toko seperti produk apa saja yang dijual, proses pengelolaan data produk, kualitas layanan, serta aktivitas transaksi yang biasa dilakukan di toko

3.2.4 Metode Pengujian

Metode pengujian yang digunakan dalam penelitian ini adalah *Black Box Testing*. Pendekatan ini difokuskan pada evaluasi fungsionalitas sistem untuk memastikan apakah sistem beroperasi sesuai dengan harapan atau tidak. Proses pengujian dilakukan dengan memasukkan sampel data ke dalam fitur-fitur yang tersedia dalam sistem, menggunakan metode *Black Box Testing*. Metode ini dipilih untuk menguji keberfungsian sistem tanpa perlu men*get*ahui detail implementasinya.

STT - NF

Hasil dari pengujian ini akan diuraikan secara lebih rinci dalam rancangan pengujian yang akan disusun. Data hasil pengujian akan dideskripsikan dalam bentuk tabel untuk memberikan Gambar an yang lebih jelas terhadap hasil-hasil yang diperoleh. Ini akan membantu dalam mengevaluasi kinerja sistem dan mengidentifikasi area-area yang memerlukan perbaikan atau penyesuaian lebih lanjut.

3.2.5 Metode Implementasi dan Evaluasi

Berdasarkan data dan informasi yang telah diperoleh, penulis hanya merancang *use case* diagram untuk membantu dalam melihat dan mengGambar kan interaksi antara aktor dengan sistem. *Use case* diagram ini akan diimplementasikan ke dalam proses perancangan sebuah sistem. Sistem yang dimaksud adalah *REST-API* Sistem Informasi Penjualan menggunakan *framework Express JS* dengan *Database PostgreSQL*, serta mengimplementasikan autentikasi menggunakan *Json Web token*. Sistem ini bertujuan untuk mempermudah pengguna dalam manajemen produk dan transaksi, sehingga proses jual beli menjadi lebih efisien.

Evaluasi dilakukan untuk mengecek hasil-hasil penilaian dari pengujian *Black Box testing* yang dilakukan sebelumnya pada semua fitur yang ada pada sistem yang nantinya akan diberi persentase keberhasilan sesuai dengan hasil pengujian apakah sistem berjalan dengan baik atau tidak.

3.2.6 Lingkungan Pengembangan

1. Tempat dan Lokasi

   Penelitian ini dilakukan di Cibinong Bogor, namun Objek penelitian berada di Toko Kue Variant Cake yang berlokasi di Pesona kahuripan 4 Klapanunggal Cileungsi.

2. Spesifikasi Alat
   a. Perangkat

STT - NF

Menggunakan perangkat keras Laptop Asus vivoBook 14s X415DAP, CPU AMD Ryzen 3 3250U with Radeon Graphics 2.6GHz, RAM 8.00 GB dan Windows 10 Home single Language.

b. *Tools* Pemrograman

Penulis menggunakan aplikasi *Visual Studio Code* untuk membuat kode pemrograman, sedangkan untuk *Database* penulis menggunakan *PostgreSQL*

c. Bahasa Pemrograman dan *Framework*

Bahasa Pemrograman yang digunakan dalam pembuatan sistem yaitu *Javascript* dan *Framework* yang digunakan yaitu *Express JS.*

d. Alat Desain

Alat desain yang digunakan untuk membuat diagram yaitu *Software Online* yang bernama *draw.io*

e. Alat Kolaborasi

Adapun alat kolaborasi yang digunakan pada penelitian ini yaitu menggunakan *WhatsApp* dan *Github* untuk mempermudah pengembangan sistem.

STT - NF

BAB IV
IMPLEMENTASI DAN EVALUASI

Pada bab ini akan memaparkan implementasi terkait rancang bangun *Rest-API* pada Sistem Informasi Penjualan menggunakan *Express JS* pada Toko Kue Variant Cake dan membahas hasil pengujian serta evaluasi *Rest-API* pada Sistem Informasi Penjualan.

4.1. Analisis Kebutuhan

4.1.1. Analisis Sistem

Analisis sistem adalah langkah awal dalam proses desain perangkat lunak yang bertujuan untuk mengevaluasi kebutuhan sistem yang akan dikembangkan. Pada tahap ini, dilakukan evaluasi terhadap kebutuhan pengguna aplikasi melalui analisis kebutuhan pengguna dan pembuatan diagram *Use Case.*

4.1.2. *User Requirement*

Analisis sistem dilakukan bertujuan untuk meng*eta*hui kebutuhan *user*. Adapun hasil analisis sebagai berikut :

1. Kategori Kebutuhan *User*

Tabel 4. 1 Kategori Kebutuhan *User*

Deskripsi	***User***
Register dan *Login*	Admin, *Customer*
Melihat Data *User*	Admin
Melihat data *Category*	Admin
Mengelola data *Category*	Admin
Melihat data Produk	Admin, *Customer*
Mengelola data Produk	Admin
Melakukan Transaksi	Admin, *Customer*

STT - NF

Melihat data Transaksi	Admin, *Customer*
Melakukan *Top-Up* Saldo	Admin, *Customer*

Tabel diatas menyajikan deskripsi fungsionalitas sistem berdasarkan peran *user*. Berbagai fitur dan hak akses disajikan untuk dua jenis *user* aplikasi yaitu Admin dan *customer* . Berdasarkan tabel diatas dapat disimpulkan bahwa Admin memiliki wewenang untuk melakukan *Create, Read, Update, Delete*, terhadap data produk beserta kategorinya. Sedangkan *Customer* hanya bisa melakukan proses pembelian produk dan melihat data transaksi pembeliannya.

2. Kategrori *User* Aplikasi

Dari Gambar an kategori kebutuhan pengguna di atas, dapat disimpulkan bahwa dalam pengembangan *REST API* untuk sistem informasi penjualan, terdapat dua aktor penting yang berperan, antara lain:

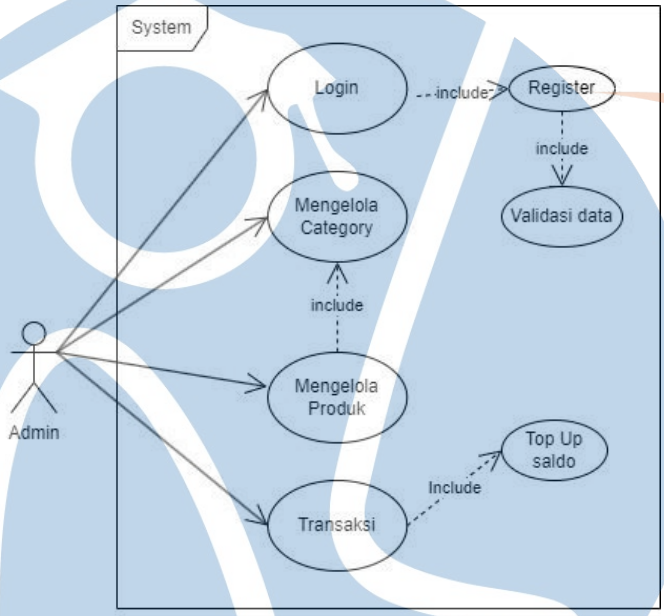
Tabel 4. 2 Kategori *User* Aplikasi

User	**Deskripsi**
Admin	*User* ini bertanggung jawab dalam pengelolaan keseluruhan aplikasi. Admin mempunyai hak akses penuh terhadap seluruh fitur pada aplikasi seperti mengelola data *User*, mengelola data produk beserta kategori nya, dan melihat data transaksi
Customer	*Costumer* dapat melihat dan membeli produk yang tersedia. *Customer* juga dapat melakukan transaksi dan melihat data transaksi pembeliannya. Selain itu *customer* dapat melakukan *Top-Up* untuk menambahkan saldonya

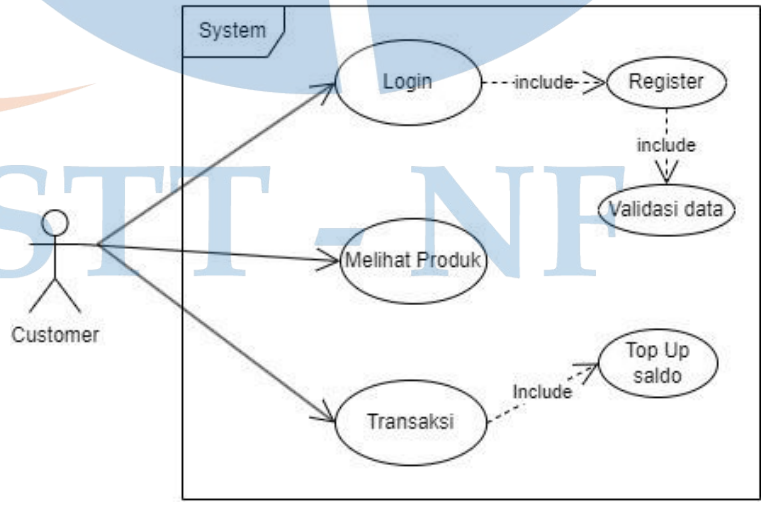
STT - NF

4.1.3 *Use case Diagram*

Bagian ini akan menjelaskan *Use case* Diagram yang bertujuan untuk mengGambar kan berbagai aktivitas yang dapat dilakukan oleh *end user* pada Sistem Informasi Penjualan.



Gambar 4. 1 *Use case Diagram - Role Admin*



Gambar 4. 2 *Use case Diagram - Role Customer*

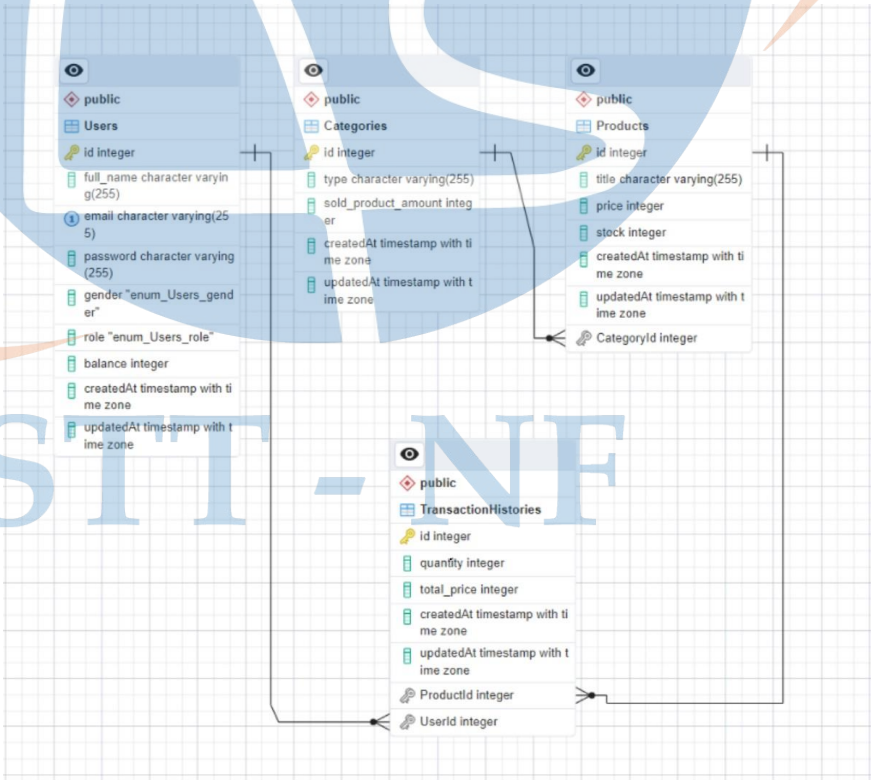
STT - NF

Dari Gambar diatas dapat dilihat bahwa terdapat dua kategori *user* yaitu Admin dan *Customer* . Admin dapat melihat semua *user*, mengelola produk beserta kategorinya, melakukan *Top-up* saldo, melakukan transaksi dan melihat riwayat transaksinya. Sedangkan *Customer* hanya dapat melihat produk, *Top-up* saldo, melakukan transaksi dan melihat riwayat transaksinya saja.

4.2. Perancangan Sistem

4.2.1 *Entity Relationship Diagram*

Entity Relationship Diagram atau *ERD* merupakan suatu diagram yang menghubungkan antara data satu dengan yang lain. *ERD* berfungsi sebagai alat bantu pembuatan *Database* untuk memberikan Gambar an tentang data dan alur kerja *Database* yang akan digunakan [20]. Berikut ini merupakan ERD dari *Database* yang digunakan pada Sistem Informasi Penjualan :



Gambar 4. 3 *Entity Relationship Diagram*

STT - NF

Dari Gambar diatas, dapat dilihat bahwa terdapat sebuah struktur *Database* dari Sistem Informasi Penjualan yang berjumlah 4 tabel, berikut penjelasan untuk setiap tabel :

a. Tabel *Users*
    - Tabel ini menyimpan informasi terkait pengguna yang telah terdaftar dalam sistem.
    - Kolom *"id"* merupakan *primary key* atau kunci utama pada tabel ini.
    - Kolom lainnya mencakup informasi seperti nama lengkap, email, password, *gender*, *role (admin, customer)*, dan *balance* (saldo).

b. Tabel *Categories*
    - Tabel ini menyimpan informasi terkait kategori Produk yang dijual
    - Kolom *"id"* merupakan *primary key* atau kunci utama pada tabel ini.
    - Hanya pengguna dengan *role* admin yang dapat mengakses tabel ini.
    - Kolom "*sold_product_amount*" default nya adalah 0, nanti akan terisi jika ada transaksi produk.

c. Tabel *Products*
    - Tabel ini menyimpan informasi terkait produk yang dijual.
    - Kolom *"id"* merupakan *primary key* atau kunci utama pada tabel ini.
    - Kolom lainnya mencakup informasi seperti nama barang, harga, *stock* , dan *Categories Id*.
    - Tabel "*Products* " mempunyai relasi *many-to-one* dengan tabel "*Categories* " dimana setiap kategori memiliki banyak produk dengan kategori yang sama.

STT - NF

- Pengguna dengan *role customer* hanya dapat melihat data *products* . Sedangkan pengguna dengan *role* admin dapat melihat, menambahkan, mengubah dan menghapus data *Products* .

d. Tabel *Transaction Histories*

- Tabel ini menyimpan informasi terkait transaksi yang dilakukan oleh *user*.
- Kolom *"id"* merupakan *primary key* atau kunci utama pada tabel ini.
- Kolom lainnya mencakup informasi seperti *quantity*, total harga, id *product* dan juga ide *user*.
- Kolom *"ProductId"* merupakan kunci asing yang mengacu pada kolom *"id"* di tabel "*Products* ".
- Kolom "*UserId*" merupakan kunci asing yang mengacu pada kolom *"id"* di tabel "*Users* ".

4.2.2. Rancangan Arsitektur *Rest-API*

Setelah melakukan analisis kebutuhan sistem dan mengidentifikasi kebutuhan pengguna berdasarkan user requirement dan use case diagram, data tersebut akan diimplementasikan dalam rancangan *Rest-API* menggunakan *framework Express.js* dengan bantuan *ORM Sequelize* dan *database PostgreSQL serta JWT (JSON Web Token). Express.js* akan digunakan untuk membuat rute *API* dan mengatur *middleware*, *Sequelize* akan memudahkan interaksi dengan database melalui model dan skema objek, *PostgreSQL* akan menyimpan dan mengelola data aplikasi, sementara *JWT* akan digunakan untuk otentikasi dan otorisasi pengguna secara aman.

Pada tahap analisis kebutuhan, telah digambarkan *usecase* diagram yang mencerminkan interaksi antara pengguna dan sistem. Dari *usecase diagram* tersebut, langkah selanjutnya adalah memetakan menjadi endpoint yang akan dibuat. Berikut list endpointnya :

Tabel 4. 3 *List Endpoint*

Method	***Endpoint***	**Deskripsi**
GET	*'/users'*	Mengambil daftar semua pengguna.
POST	*'/users/register'*	Mendaftarkan pengguna baru.
POST	*'/users/login'*	Melakukan proses login pengguna.
PUT	*'/users'*	Memperbarui informasi pengguna.
DELETE	*"/users/:id'*	Menghapus pengguna berdasarkan *id*.
PATCH	*'/users/topup'*	Melakukan proses *top up* saldo pengguna.
GET	*'/categories'*	Membuat kategori baru.
POST	*'/categories'*	Mengambil daftar semua kategori.
PATCH	*'/categories/:categoryId'*	Mengedit kategori berdasarkan *id* kategori.
DELETE	*'/categories/:categoryId'*	Menghapus kategori berdasarkan *id* kategori.
GET	*'/products'*	Mengambil daftar semua produk.
GET	*'/products/:productId'*	Mengambil detail produk berdasarkan *id* produk.
POST	*'/products'*	Membuat produk baru.
PUT	*'/products/:productId''*	Mengedit produk berdasarkan *id* produk.
PATCH	*'/products/:productId''*	Mengedit kategori produk berdasarkan *id* produk.
DELETE	*'/products/:productId''*	Menghapus produk berdasarkan *id* produk.
POST	*'/transactions'*	Membuat transaksi baru.

GET	*'/transactions/user'*	Mengambil daftar transaksi pengguna.
GET	*'/transactions/admin'*	Mengambil daftar semua transaksi (untuk admin).
GET	*'/transactions/:transactionId'*	Mengambil detail transaksi berdasarkan *id*.

Setelah membuat list endpoint, selanjutnya endpoint tersebut dipetakan ke dalam *Controller* dan Model pada express.js. berikut ini merupakan *mapping* dari Model dan *Controller* yang akan dimuat dalam sebuah tabel

Tabel 4. 4 Mapping Model

Model	Property
User	*`full_name`, `email`, `password`, `gender`, `role`, `balance`*
Category	*`type`, `sold_product_amount`*
Product	*`title`, `price`, `stock`, `CategoryId`*
TransactionHistory	*`UserId`, `ProductId`, `quantity`, `total_price`, `transaction_date`*

Tabel 4. 5 *Mapping Controller*

Controller	***Method***
UserController	*getAllUser*
	register
	login
	updateUser
	deleteUser
	topup
CategoryController	*createCategory*
	getAllCategories
	editCategory
	deleteCategory
ProductController	*createProduct*
	getAllProducts

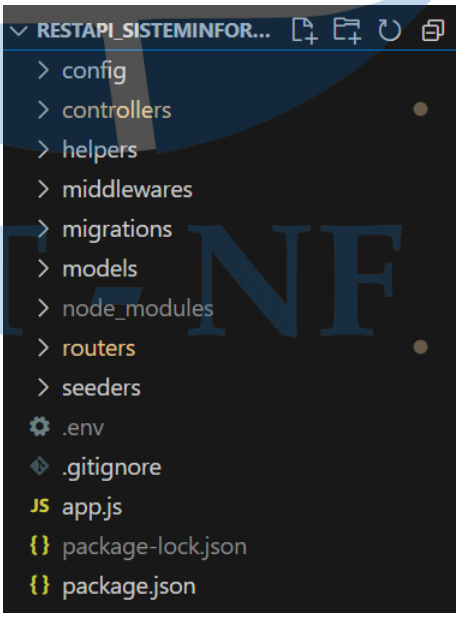
STT - NF

	getProductsbyId
	editProduct
	editCategoryId
	deleteProduct
TransactionController	*createTransaction*
	getAllTransaction
	getAllTransactionAdmin
	getTransactionbyId

Setiap endpoint dalam aplikasi terhubung dengan router yang menentukan cara mengelola permintaan HTTP. Router ini mengarahkan permintaan ke fungsi-fungsi yang ada dalam controller terkait. Controller menggunakan model untuk mengakses dan mengubah data dalam database, seperti menambah, mengambil, memperbarui, atau menghapus entri data sesuai kebutuhan aplikasi. Dengan cara ini, aplikasi dapat mengelola operasi dasar data dengan lebih terstruktur dan efisien.

1. **Struktur Direktori**

   Berikut ini merupakan struktur direktori yang terdapat pada *Rest-Api* Sistem Informasi Penjualan :

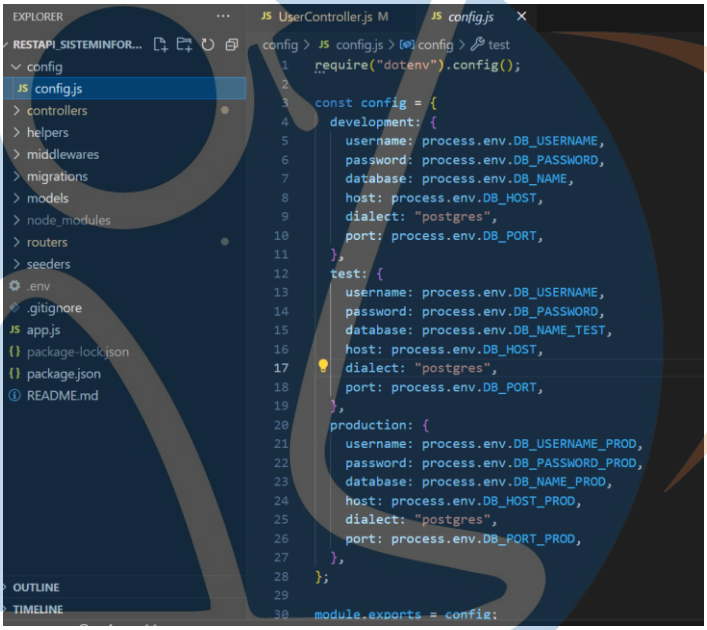


Gambar 4. 4 Struktur Direktori

2. **Konfigurasi *Project***

a. *'config/config.js'*

Berisi pengaturan konfigurasi seperti koneksi ke database (misalnya, nama *database*, pengguna, kata sandi, dan *host*), pengaturan *server*, dan pengaturan lain yang diperlukan untuk lingkungan aplikasi (*development, testing, production*). Penulisan kode ditulis di dalam file yang diberi nama *config.js* yang terletak di dalam direktori *config*



Gambar 4. 5 *Script Config.js*

b. *File '.env'*

File ini berisi variabel lingkungan seperti kredensial database, kunci rahasia JWT, dan pengaturan lainnya.

3. ***Models***

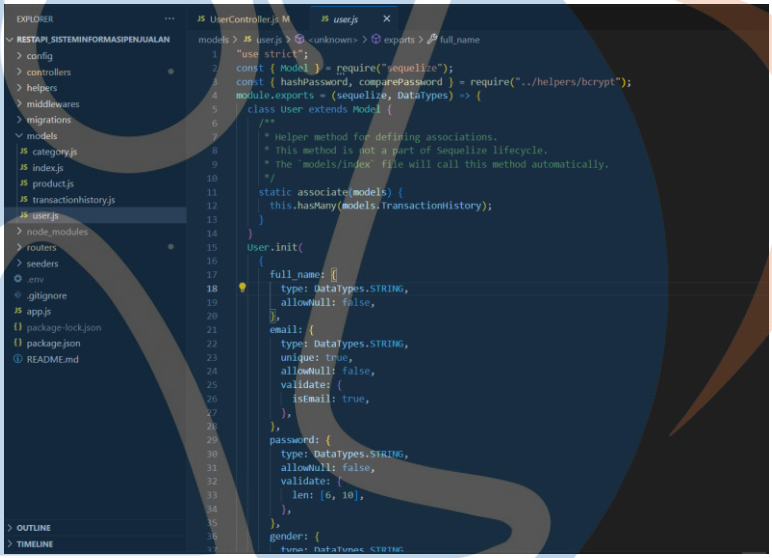
Models digunakan untuk berinteraksi dengan database. Model-model dalam direktori ini didefinisikan menggunakan *Sequelize*, sebuah *ORM (Object-Relational Mapping)* untuk *Node.js* yang memudahkan interaksi dengan *database*. Setiap model merepresentasikan tabel dalam database dan mendefinisikan atribut serta tipe data yang terkait. Selain itu, model-model ini juga dapat mendefinisikan asosiasi atau relasi antar tabel.

STT - NF

Penulisan kode Model pada rancangan *Rest-Api* Sistem Informasi Penjualan ini ditulis kedalam beberapa file yang disimpan dalam direktori yang bernama *'models'* diantaranya :

a. *User.js*

File `user.js` berisi definisi skema data dan model untuk pengguna dalam sistem. Model ini mencakup kolom-kolom yang diperlukan untuk menyimpan informasi tentang pengguna, seperti nama pengguna, *email*, *password*, *gender*, saldo, dan peran pengguna (*admin* atau *customer*).



Gambar 4. 6 *Script User.js*

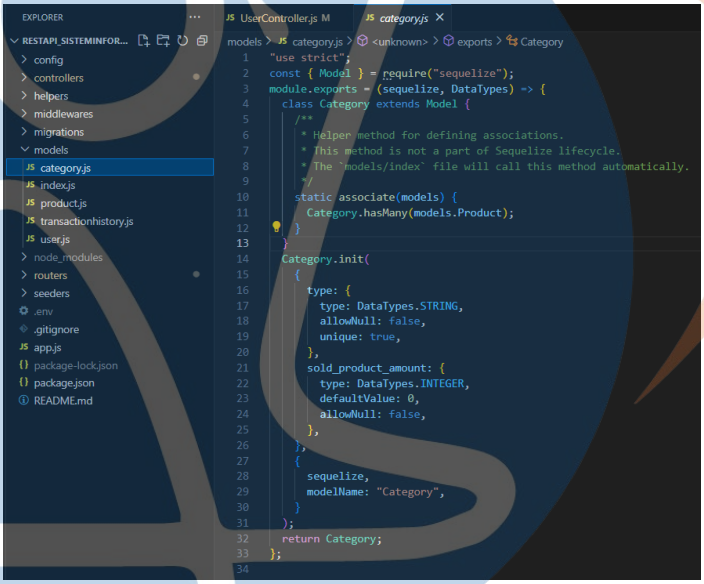
Kode di atas mendefinisikan model `*User*` menggunakan Sequelize dalam mode ketat, dan mengimpor fungsi `hashPassword` dan `*comparePassword*` dari `*bcrypt*` untuk meningkatkan keamanan password pengguna. Model `*User*` memiliki kolom `*full_name*`, `*email*`, `*password*`, `*gender*`, `*role*`, dan `*balance*`, dengan berbagai aturan validasi seperti `*allowNull*`, `*unique*`, `*validate*`, dan `*defaultValue*`. Sebelum menyimpan pengguna baru ke database, password di-*hash* menggunakan *hook* `*beforeCreate*`. Model ini juga memiliki asosiasi `*hasMany*` dengan model `*TransactionHistory*`, yang berarti satu pengguna dapat memiliki banyak riwayat transaksi. Selain itu, ada metode `*validatePassword*`

STT - NF

untuk membandingkan *password* yang diberikan dengan *password* yang sudah di-*hash*.

b. *Category,js*

File `category.js` berfungsi untuk menggambarkan skema data dan model untuk kategori produk dalam sistem. Model ini akan berisi definisi kolom yang diperlukan untuk menyimpan informasi tentang kategori produk, seperti nama kategori dan banyak produk yang telah terjual. Dengan adanya model ini, sistem dapat mengorganisir produk berdasarkan kategori yang telah ditentukan.



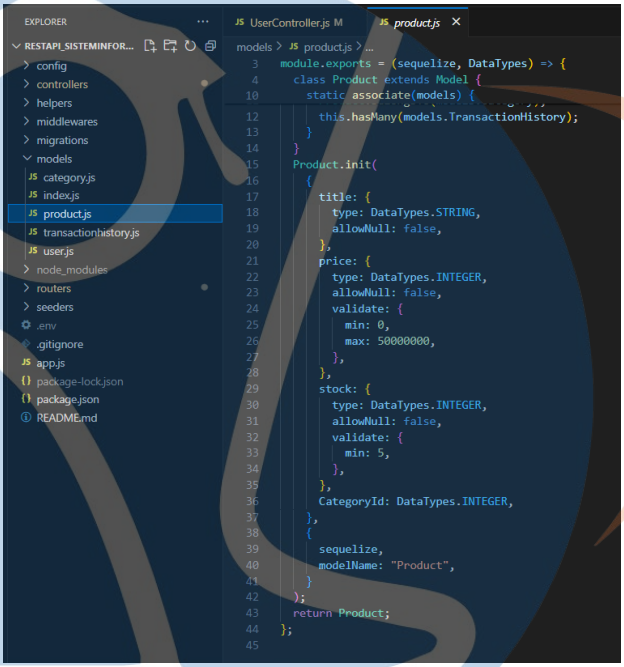
Gambar 4. 7 Script Categoty.js

Kode diatas mendefinisikan model `*Category*` menggunakan *Sequelize*. *Class* `*Category*` adalah turunan dari `*Model*` dan memiliki dua kolom: `*type*`, yang merupakan string unik dan tidak boleh kosong, serta `*sold_product_amount*`, yang merupakan angka dengan nilai default 0 dan tidak boleh kosong. Model ini memiliki asosiasi `*hasMany*` dengan model `*Product*`, yang berarti satu kategori dapat memiliki banyak produk. Model `*Category*` diinisialisasi dengan menggunakan *instance Sequelize* dan diberi nama `*Category*`, kemudian diekspor untuk digunakan di bagian lain dari aplikasi.

STT-NF

c. *Product.js*

File `product.js` berisi definisi skema data dan model untuk produk dalam sistem. Model ini mencakup kolom-kolom yang diperlukan untuk menyimpan informasi tentang produk, seperti nama produk, harga, stok, dan referensi ke kategori produk. Model produk ini memungkinkan sistem untuk mengelola data produk dengan efektif dan efisien.



Gambar 4. 8 *Script Product.js*

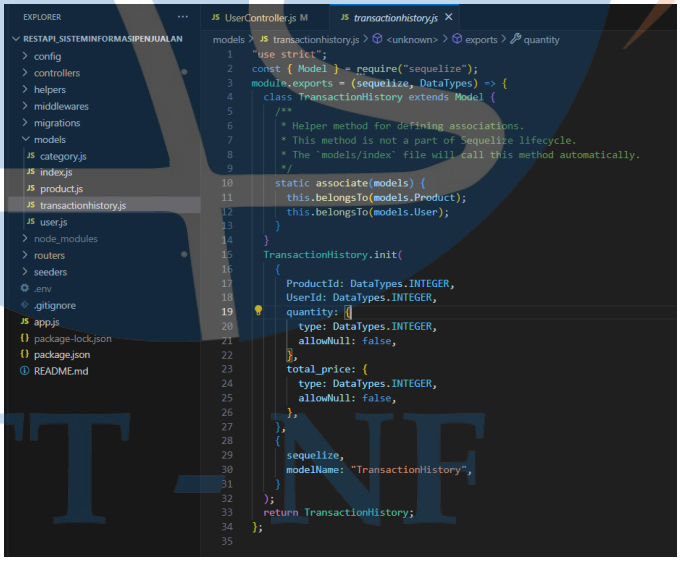
Potongan kode di atas mendefinisikan model `*Product*` menggunakan *Sequelize*. Model ini dirancang untuk merepresentasikan produk dalam aplikasi. Dalam definisinya, model `*Product*` memiliki beberapa kolom yang mewakili atribut produk, antara lain `*title*` untuk judul produk (tipe data string yang tidak boleh kosong), `*price*` untuk harga produk (tipe data integer dengan validasi minimal 0 dan maksimal 50.000.000), `*stock*` untuk stok produk (tipe data integer yang minimal harus 5), dan `*CategoryId*` untuk mengaitkan produk dengan kategori tertentu (tipe data *integer*). `*Product.belongsTo(models.Category)*` menunjukkan bahwa setiap produk dimiliki oleh satu kategori yang didefinisikan

STT-NF

dalam model `*Category*`, dan `*this.hasMany(models.TransactionHistory)*` menunjukkan bahwa setiap produk dapat memiliki banyak riwayat transaksi yang tercatat dalam model `*TransactionHistory*`. Model `*Product*` kemudian diekspor untuk digunakan dalam bagian lain dari aplikasi yang menggunakan *Sequelize*.

d. *transactionhistory.js*

File `transactionhistory.js` menggambarkan skema data dan model untuk riwayat transaksi dalam sistem. Model ini mencakup kolom-kolom yang diperlukan untuk melacak detail transaksi yang terjadi, seperti ID pengguna, ID produk, jumlah produk yang dibeli, total harga, dan tanggal transaksi. Dengan adanya model ini, sistem dapat mencatat dan menampilkan riwayat transaksi yang dilakukan oleh pengguna.

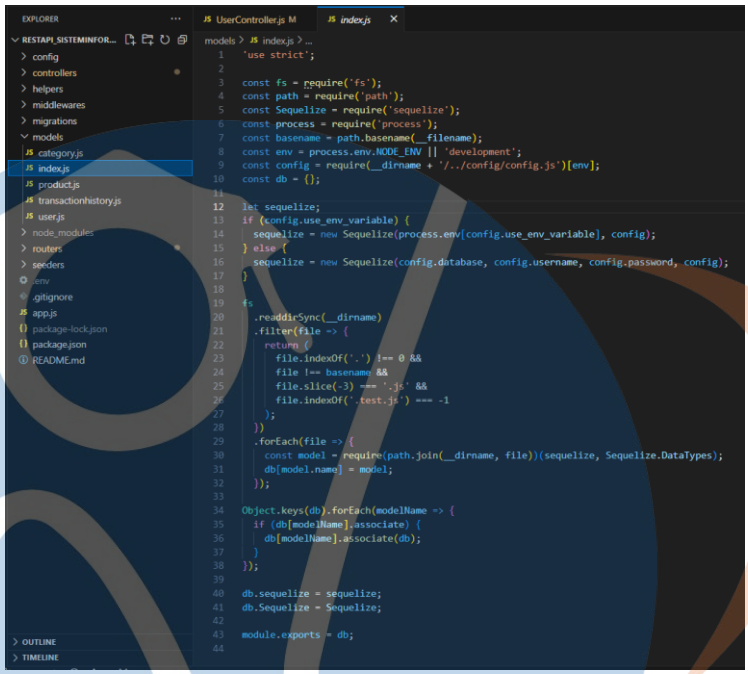


Gambar 4. 9 Script Transactionhistory.js

e. *index.js*

File `index.js` bertindak sebagai *file* penghubung atau *entry point* untuk semua model lainnya dalam direktori model. File ini bertanggung jawab untuk menginisialisasi *Sequelize* dan mengimpor

semua model yang ada di direktori. Setelah itu, file ini juga mengatur asosiasi antar model jika ada. Dengan kata lain, file ini menyatukan semua model dan memastikan mereka dapat diakses secara konsisten di seluruh aplikasi.



Gambar 4. 10 Script Index.js

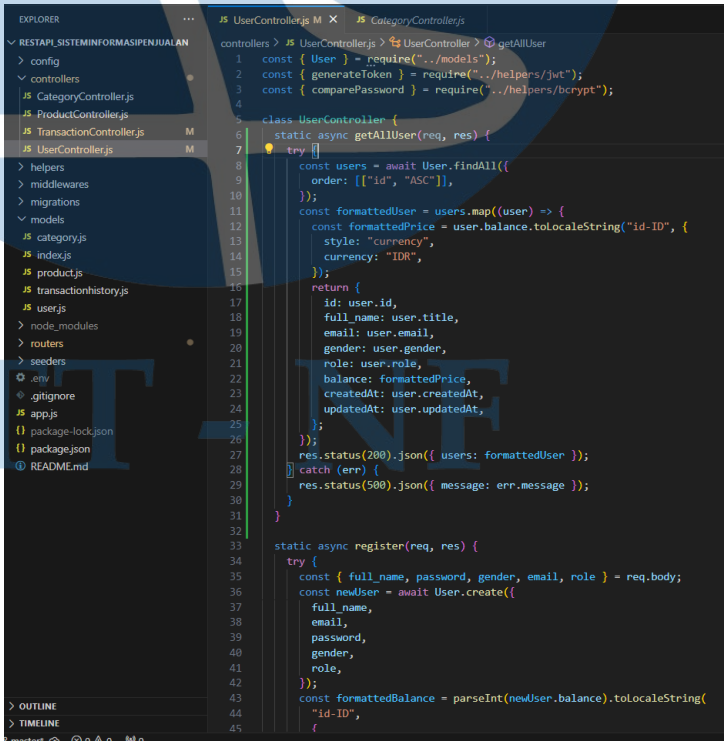
4. *Controller*

Dalam proyek REST API, direktori “controllers” berperan penting dalam menangani permintaan HTTP yang masuk dari klien dan mengelola respons yang diberikan. Setiap file di direktori ini didedikasikan untuk entitas tertentu dalam sistem. Terdapat beberapa penulisan kode *Controller* pada Rest-API Sistem Informasi Penjualan ini ditulis kedalam beberapa file yang disimpan dalam direktori yang bernama *‘Controller*’ diantaranya :

a. *UserController.js*

Potongan kode di bawah mendefinisikan kelas ‘*UserController’* yang menangani berbagai operasi terkait pengguna dalam aplikasi. Diawali dengan mengimpor model ‘*User’*, serta fungsi *‘generateToken’* dan ‘*comparePassword*’ dari modul terkait,

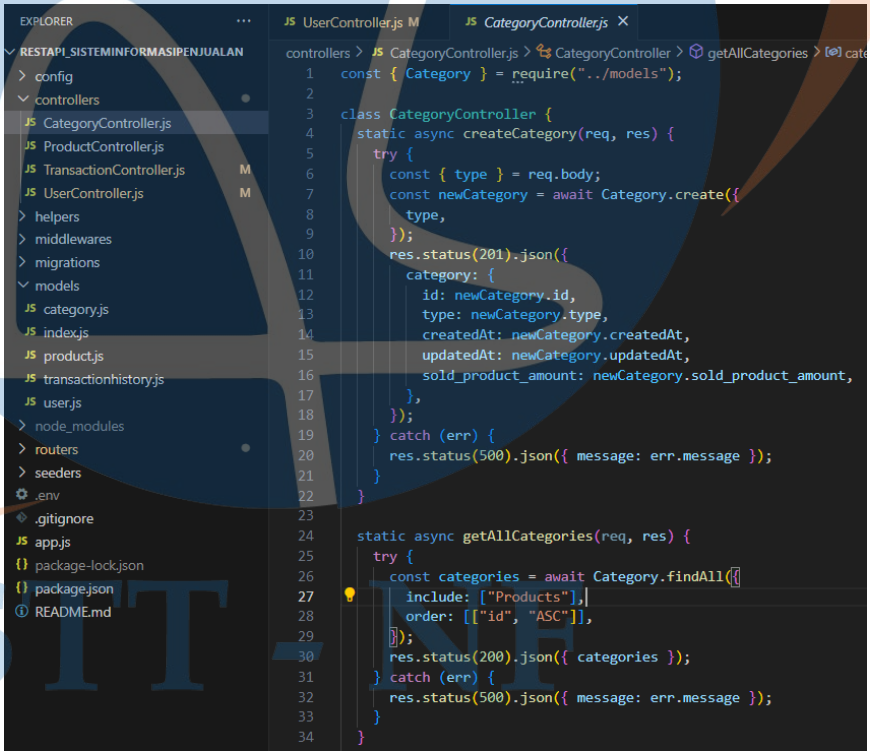
kelas ini memiliki beberapa metode penting. '*getAllUser*' mengambil semua pengguna dari *database*, mengurutkannya berdasarkan '*ide*', memformat saldo dalam mata uang IDR, dan mengirimkannya sebagai *respons JSON*. '*register*' membuat pengguna baru berdasarkan data yang diterima dari '*req.body*', memformat saldo pengguna baru, dan mengembalikan informasi pengguna tersebut dalam format *JSON*. '*login*' memvalidasi pengguna berdasarkan email dan password, menghasilkan *token JWT* jika *valid*, dan mengembalikannya sebagai respons. '*updateUser*' memungkinkan pengguna memperbarui nama lengkap dan email mereka, menyimpan perubahan, dan mengembalikan data terbaru. '*deleteUser*' menghapus akun pengguna yang saat ini terautentikasi jika ditemukan dalam *database*. Terakhir, *method* '*topup*' memungkinkan pengguna menambah saldo mereka, memperbarui saldo dalam *database*, dan mengembalikan saldo yang telah diformat. Seluruh kelas ini diekspor sehingga dapat digunakan di bagian lain dari aplikasi.



Gambar 4. 11 *UserController*

b. *CategoryController.js*

Kelas *`CategoryController`* menangani operasi *CRUD* untuk kategori dalam aplikasi, menggunakan model *`Category`* dari *`../models`*. *method `createCategory`* membuat kategori baru dari data *`req.body`* dan mengembalikan informasi kategori tersebut dalam format *JSON*. *method `getAllCategories`* mengambil semua kategori beserta produk terkait dan mengembalikannya sebagai *JSON*. *method `editCategory`* memperbarui kategori berdasarkan *id* dari *`req.params`* dan data dari *`req.body`*, lalu mengembalikan informasi kategori yang diperbarui. *method `deleteCategory`* menghapus kategori berdasarkan *'id'* dari *`req.params`*, jika ditemukan, dan mengembalikan pesan sukses.



Gambar 4. 12 CategoryController

c. *ProductController*

Product controller bertanggung jawab untuk mengelola seluruh operasi terkait produk dalam aplikasi, termasuk pembuatan, pengambilan, pembaruan, dan penghapusan produk dari basis

data. Fungsi-fungsi utama *controller* ini meliputi menerima permintaan untuk membuat produk baru, mengelola permintaan untuk mengambil daftar produk atau detail produk spesifik berdasarkan *id*, serta memproses pembaruan informasi produk dan penghapusan produk dari database. Selain itu, product controller juga dapat mengatur operasi terkait kategori produk, seperti mengubah kategori produk tertentu. Dengan demikian, controller ini memastikan bahwa data produk dikelola dengan efisien dan sesuai dengan kebutuhan aplikasi.

d. *TransactionController*

TransactionController bertanggung jawab atas manajemen operasional terkait transaksi dalam aplikasi. Fungsi utamanya meliputi pengambilan daftar transaksi pengguna, pembuatan transaksi baru dengan validasi data, dan pengambilan detail transaksi berdasarkan *id*. *Controller* ini juga mendukung fungsi khusus seperti pengambilan semua transaksi untuk administrator. Dengan demikian, *TransactionController* memfasilitasi interaksi yang aman dan efisien antara rute aplikasi dan basis data untuk pengelolaan transaksi yang konsisten dan dapat dipercaya.

5. *Routers*

Route (rute) dalam pengembangan *web* merujuk pada alamat atau endpoint yang terhubung dengan aplikasi atau *API*. Mereka menentukan bagaimana aplikasi harus menanggapi permintaan *HTTP* tertentu dari klien, seperti menambah, membaca, memperbarui, atau menghapus data. Setiap route memisahkan fungsionalitas tertentu dalam aplikasi, memungkinkan pengelolaan logika bisnis dengan jelas dan terstruktur. Dalam kerangka kerja seperti. *Express js* untuk *Node.js, route* didefinisikan dengan metode seperti *.get(), .post(), .put(), .delete(),* mengaitkan setiap jenis permintaan dengan fungsi penanganan yang

sesuai, membantu dalam mengelola struktur dan operasi aplikasi secara efisien Penulisan kode *router* pada *Rest-API* Sistem Informasi Penjualan ini ditulis kedalam beberapa file yang disimpan dalam direktori yang bernama *'Routers' :*

a. *userRouter.js*

   File ini berisi definisi Router terkait dengan pengguna aplikasi. Contoh rute yang bisa ada di sini adalah untuk pendaftaran pengguna baru, proses masuk (*login*), atau untuk mengelola profil pengguna seperti mengubah informasi pengguna atau menghapus akun.

b. *categoryRouter.js*

   File ini berfokus pada definisi rute-rute yang terkait dengan entitas kategori dalam sistem. Contohnya, rute untuk mengambil daftar kategori yang ada atau untuk membuat kategori baru.

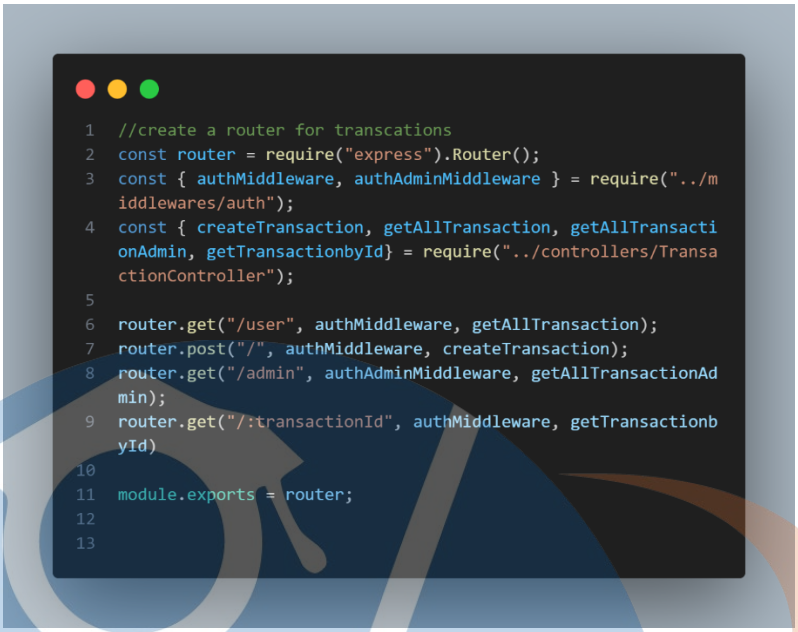
c. *ProductRouter.js*

   File ini mengandung definisi rute-rute yang berkaitan dengan produk. Ini bisa mencakup operasi seperti mengambil daftar produk, menambahkan produk baru ke dalam sistem, atau menghapus produk yang ada.

d. *transactionRouter.js*

   Berfokus pada definisi rute-rute terkait dengan transaksi dalam aplikasi. Ini termasuk mengambil riwayat transaksi yang telah terjadi atau memungkinkan pembuatan transaksi baru dalam sistem.

STT - NF



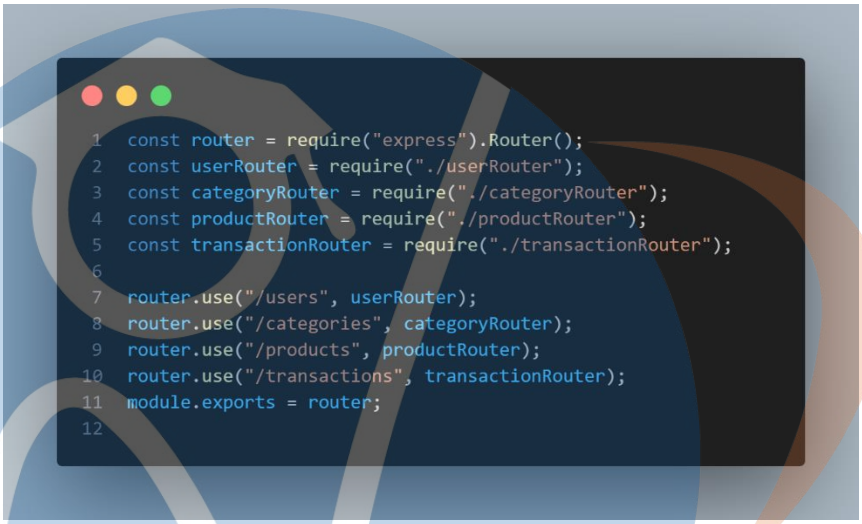
```
//create a router for transcations
const router = require("express").Router();
const { authMiddleware, authAdminMiddleware } = require("../middlewares/auth");
const { createTransaction, getAllTransaction, getAllTransactionAdmin, getTransactionbyId} = require("../controllers/TransactionController");

router.get("/user", authMiddleware, getAllTransaction);
router.post("/", authMiddleware, createTransaction);
router.get("/admin", authAdminMiddleware, getAllTransactionAdmin);
router.get("/:transactionId", authMiddleware, getTransactionbyId)

module.exports = router;
```

Gambar 4. 13 *TransactionRouter*

Kode di atas mendefinisikan sebuah router untuk transaksi dalam aplikasi *Express.js*. Pertama, *router Express* dibuat untuk mendefinisikan rute-rute *HTTP*. *Middleware* *`authMiddleware`* dan *`authAdminMiddleware`* di*import* dari *`../middlewares/auth`* untuk memastikan pengguna telah diautentikasi dan memiliki hak akses yang sesuai sebelum mengakses rute tertentu. Fungsi-fungsi pengontrol seperti *`createTransaction`*, *`getAllTransaction`*, *`getAllTransactionAdmin`*, dan *`getTransactionbyId` diimpor dari `../controllers/TransactionController`* untuk menangani logika bisnis terkait transaksi. Rute *`GET` ke `/user`* digunakan untuk mendapatkan semua transaksi milik pengguna yang telah diautentikasi, sedangkan rute *`POST` ke `/`* digunakan untuk membuat transaksi baru. Rute *`GET` ke* `/admin` digunakan untuk mendapatkan semua transaksi dan hanya dapat diakses oleh admin. Rute *`GET`* ke *`/:transactionId`* digunakan untuk mendapatkan transaksi berdasarkan *id*-nya, dan semua rute ini menggunakan middleware untuk memastikan otorisasi yang tepat. Terakhir, *router* diekspor untuk digunakan di bagian lain dari aplikasi.

e. *indeks.js*

Merupakan titik masuk utama untuk semua *rute* yang ada dalam aplikasi. Di sini, file ini menggabungkan semua router yang ada, menetapkan rute utama (seperti *rute* beranda), dan mengatur bagaimana aplikasi menanggapi permintaan *HTTP* yang masuk.



```
const router = require("express").Router();
const userRouter = require("./userRouter");
const categoryRouter = require("./categoryRouter");
const productRouter = require("./productRouter");
const transactionRouter = require("./transactionRouter");

router.use("/users", userRouter);
router.use("/categories", categoryRouter);
router.use("/products", productRouter);
router.use("/transactions", transactionRouter);
module.exports = router;
```

Gambar 4. 14 Index.js

Kode di atas mendefinisikan router utama untuk aplikasi Express.js dan menghubungkan berbagai sub-router yang menangani rute-rute spesifik terkait pengguna, kategori, produk, dan transaksi. Pertama, router utama diinisialisasi dengan *`require("express").Router()`*. Kemudian, sub-router untuk pengguna (`userRouter`), kategori *(`categoryRouter`)*, produk *(`productRouter`)*, dan transaksi *(`transactionRouter`)* diimpor dari file masing-masing yang ada di direktori `./`. Setelah itu, router utama mengatur rute-rute dengan menggunakan *`router.use()`*, yang menghubungkan masing-masing sub-router ke jalur tertentu: *`/users`* untuk rute pengguna, *`/categories`* untuk rute kategori, `/products` untuk rute produk, dan *`/transactions`* untuk rute transaksi. Terakhir, router utama diekspor dengan *`module.exports`* agar dapat digunakan di bagian lain dari aplikasi.

4.2.3. Rancangan Pengujian

Berdasarkan pembahasan sebelumnya, penelitian ini menggunakan satu desain uji untuk memperoleh hasil evaluasi yang sesuai dengan tujuan penelitian. Penelitian ini menyatakan bahwa *REST API* sistem informasi penjualan layak digunakan oleh masyarakat dan hasilnya sesuai dengan harapan penulis. Desain pengujian yang digunakan dalam pengembangan *REST API* sistem informasi penjualan adalah metode *Black Box testing*. Hasil pengujian ini diharapkan dapat menjadi pertimbangan bagi masyarakat dalam menggunakan aplikasi tersebut.

Rancangan pengujian dalam penelitian ini adalah *Black Box testing*, yang digunakan untuk menyesuaikan desain yang telah dibuat oleh penulis. Pengujian ini fokus pada tampilan antarmuka dan fungsionalitas aplikasi yang dapat dilihat dan diakses oleh pengguna. Kasus pengujian disajikan dalam bentuk tabel berikut.

STT - NF

Tabel 4. 6 Rancangan Pengujian *Black Box Testing*

No	Pengujian	Http *Method* /End Point	Hasil yang diharapkan	Kesimpulan
1	*Register* dengan *request body* yang sesuai	***POST,*** *http://localhost:*3000*/Users/register*	Akan menghasilkan *response Code* : “201“ dan menampilkan akun telah terbuat	Sesuai atau Tidak Sesuai
2	*Register* dengan *request body* yang sesuai	***POST,*** *http://localhost:*3000*/Users/register*	Akan menghasilkan *response Code* : “201“ dengan *message* “*validation isEmail on email failed*”	Sesuai atau Tidak Sesuai
3	*Login* dengan akun yang benar	***POST,*** *http://localhost:*3000*/Users/login*	Akan menghasilkan *response Code* : “200“ dan menghasilkan token *login*	Sesuai atau Tidak Sesuai
4	*Login* dengan *request body* yang salah	***POST,*** *http://localhost:*3000*/Users/login*	Akan menghasilkan *response Code* : “400“ dan *message* “*invalid email/password*”	Sesuai atau Tidak Sesuai

STT - NF

5	Melihat data *Users* dengan akun admin	***GET,*** *http://localhost:*3000*/Users/*	Akan menghasilkan *response Code* : “200“ dan menampilkan data *Users*	Sesuai atau Tidak Sesuai
6	Melihat data *Users* dengan token yang salah	***GET,*** *http://localhost:*3000*/Users/*	Akan menghasilkan *response Code* : “401“ dan *message “unauthorized*”	Sesuai atau Tidak Sesuai
7	Memperbarui data *Users*	***PUT,*** *http://localhost:*3000*/Users/*	Akan menghasilkan *response Code* : “200“ dan menampilkan data yang telah di *update*	Sesuai atau Tidak Sesuai
8	Menghapus data *Users*	***DELETE,*** *http://localhost:*3000*/Users/*	Akan menghasilkan *response Code : “*200*”* dengan *message "Your account has been Success fully deleted"*	Sesuai atau Tidak Sesuai
9	Melakukan *Top-up* saldo dengan akun admin	***PATCH,*** *http://localhost:*3000*/Users/topup*	Akan menghasilkan *response Code : “*200*”* dengan *message "Your balance has been Success fully updated"*	Sesuai atau Tidak Sesuai

STT - NF

10	Melakukan *Top-up* saldo dengan akun *customer*	***PATCH,*** *http://localhost:*3000/*Users/topup*	Akan menghasilkan *response Code* : "200*"* dengan *message "Your balance has been Success fully updated"*	Sesuai atau Tidak Sesuai
11	Membuat data *Categories* dengan akun admin	***POST,*** *http://localhost:*3000/*Categories /*	Akan menghasilkan *response Code* "201" dan menampilkan data yang baru dibuat	Sesuai atau Tidak Sesuai
12	Membuat data *Categories product* dengan token yang salah	***POST,*** *http://localhost:*3000/*Categories /*	Akan menghasilkan *response Code* : "401" dan *message* "unauthorized"	Sesuai atau Tidak Sesuai
13	Memperbarui data *Categories* sesuai *request body*	***PATCH****,* *http://localhost:*3000/*Categories /* *:Categories* Id	Akan menghasilkan *response Code* "200" dan menampilkan data yang telah di *update*	Sesuai atau Tidak Sesuai
14	Melihat data *Categories* dengan akun admin	***GET,*** *http://localhost:*3000/*Categories /*	Akan menghasilkan *response Code* : "200" dan menampilkan seluruh data *Categories*	Sesuai atau Tidak Sesuai
15	Melihat data *Categories* dengan token yang salah	***GET,*** *http://localhost:*3000/*Categories /*	Akan menghasilkan *response Code* : "401" dan *message* "*unauthorized*"	Sesuai atau Tidak Sesuai

STT-NF

16	Membuat data *Products* dengan akun admin	***POST,*** *http://localhost:*3000*/products /*	Akan menghasilkan *response Code :* “201” dan menampilkan data yang baru dibuat	Sesuai atau Tidak Sesuai
17	Membuat data *Products* dengan token yang salah	***POST,*** *http://localhost:*3000*/products /*	Akan menghasilkan *response Code* : “401“ dan *message* “unauthorized”	Sesuai atau Tidak Sesuai
18	Memperbarui data *Products* sesuai *request body*	***PUT***, *http://localhost:*3000*/products /:produtc*Id	Akan menghasilkan *response Code* : “200” dan menampilkan data yang telah di *update*	Sesuai atau Tidak Sesuai
19	Memperbarui data *Products* dengan id yang salah	***PUT***, *http://localhost:*3000*/products /:produtc*Id	Akan menghasilkan *response* Code “404” dengan *message “Product Not Found”*	Sesuai atau Tidak Sesuai
20	Memperbarui data *Categories* Id *Products*	***PATCH***, *http://localhost:*3000*/products /:produtc*Id	Akan menghasilkan *response Code* : “200” dan menampilkan data yang telah di *update*	Sesuai atau Tidak Sesuai

STT - NF

21	Melihat data *Products* dengan akun admin	***GET,*** *http://localhost:*3000*/products /*	Akan menghasilkan *response Code* : “200“ dan menampilkan seluruh data *Products*	Sesuai atau Tidak Sesuai
22	Melihat data *Products* dengan akun *Customer*	***GET,*** *http://localhost:*3000*/products /*	Akan menghasilkan *response Code* : “200“ dan menampilkan seluruh data *Products*	Sesuai atau Tidak Sesuai
23	Melakukan transaksi dengan akun admin	***POST,*** *http://localhost:*3000*/transactions /*	Akan menghasilkan *response Code* : “201” dengan *message “You Have Success fully purchase the product*” dan menampilkan *Transaction Bill*	Sesuai atau Tidak Sesuai
24	Melakukan transaksi dengan akun *Customer*	***POST,*** *http://localhost:*3000*/transactions /*	Akan menghasilkan *response Code* “201” dengan *message “You Have Success fully purchase the product”* dan menampilkan *Transaction Bill* serta mengurangi *stock* produk yang ada pada *table Products*	Sesuai atau Tidak Sesuai
25	Melakukan transaksi jika produk habis	***POST,*** *http://localhost:*3000*/transactions /*	Akan menghasilkan *response Code* “400” dengan *message “Product is out of stock”*	Sesuai atau Tidak Sesuai

STT - NF

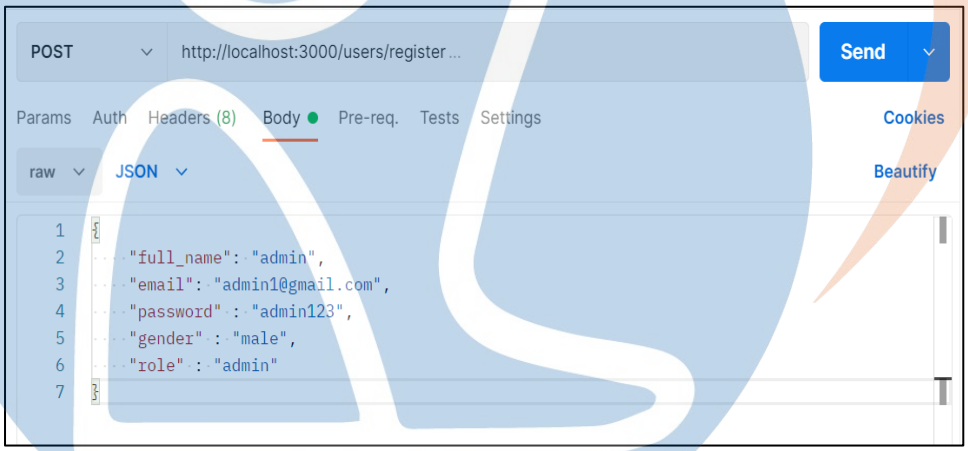
4.3. Implementasi Fungsi Aplikasi

Dalam penulisan implementasi fungsi pada aplikasi, penulis menjelaskan alur yang ada pada *Rest-API* Sistem Informasi Penjualan. Alur ini dirancang agar dapat mempermudah pengguna. Adapun beberapa fitur yang terdapat pada *Rest-API* Sistem Informasi Penjualan ini yaitu :

4.3.1. Fungsi *Register*

a. *Register* Admin

Dalam penerapan fungsi *Rest-API* adapun tahapan *register* admin yang dibuat untuk *register* akun admin. Berikut adalah tampilan *register* admin yang dibuat:

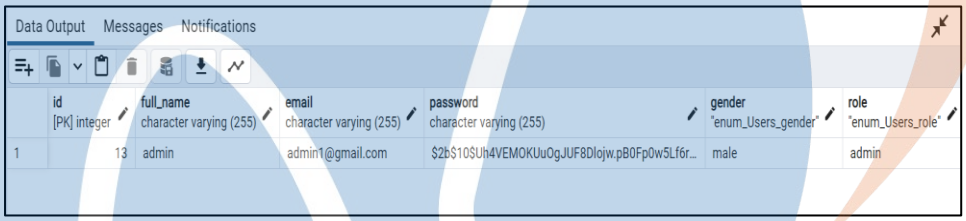


Gambar 4. 15 *Register Admin*

Pada Gambar diatas, untuk mendaftar dengan *role* admin, *user* harus me*request* ke *http://localhost:3000/Users/register* dengan *method POST*. Lalu dalam *body request* nya harus berisi data diri seperti *{"full_name", "email", "password", "gender", "role"}*. Lalu jika berhasil akan menghasilkan *response* :

Gambar 4. 16 *Register Admin Success*

Maka di dalam *Database table Users* akan terisi data yang telah didaftarkan tadi seperti ini :



Gambar 4. 17 *Database Table Users*

b. Register Customer

Berikut ini adalah tampilan *register Customer* yang akan dibuat :

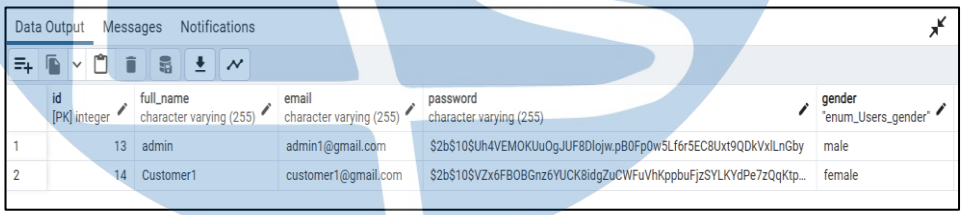
Gambar 4. 18 *Register Customer*

STT - NF

Sama halnya dengan *role* admin, untuk mendaftar dengan *role customer user* harus me*request* ke *http://localhost:3000/Users/register* dengan *method POST*. Lalu dalam *body request* nya harus berisi data diri seperti *{"full_name", "email", "password", "gender", "role"}*. Lalu jika berhasil akan menghasilkan *response* :

Gambar 4. 19 *Register Success*

Lalu di dalam *Database table Users* akan terisi data yang telah didaftarkan tadi seperti ini :



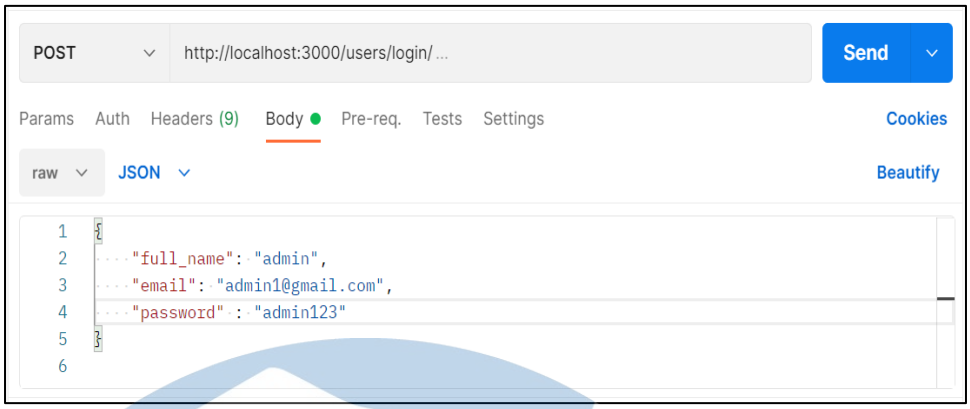
Gambar 4. 20 *Data Database Tabel Users*

4.3.2. Fungsi *Login*

1. *Login* Admin

Berikut ini adalah tampilan *Login* dengan *role* Admin yang akan dibuat :

STT - NF



Gambar 4. 21 *Login*

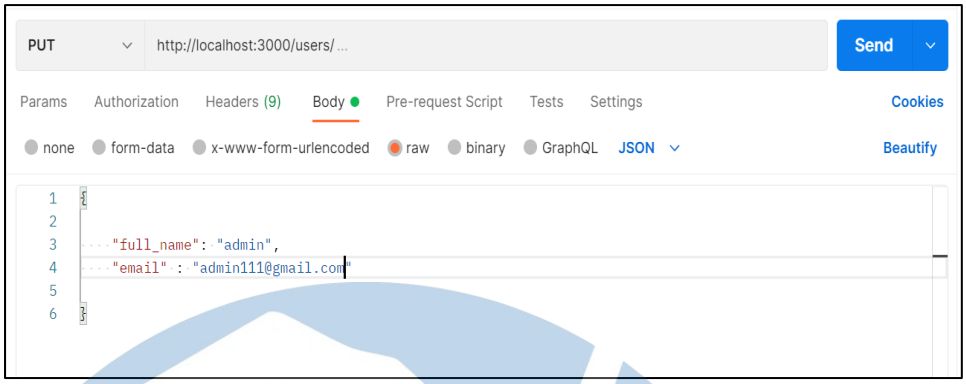
Pada Gambar diatas, untuk dapat *Login* , *user* harus me*request* ke *http://localhost:*3000*/Users/login* dengan *method POST*. lalu dalam *body request* nya harus berisi data *{"full_name", "email", "password"}* yang sudah didaftarkan pada *register* tadi. Lalu jika berhasil maka akan menghasilkan *response* seperti ini :

Gambar 4. 22 *Json Web token*

Pada Gambar tersebut, setelah *login user* akan mendapatkan *response* berupa *Json Web token* yang berbentuk string random panjang. *Token* ini akan dimasukkan ke dalam *header* setiap kali *user* ingin me*request API* tertentu yang bertujuan agar sistem dapat mengenali bahwa yang sedang melakukan *request* tersebut adalah *user* dan *token* tersebut.

STT NF

2. *Update Users*

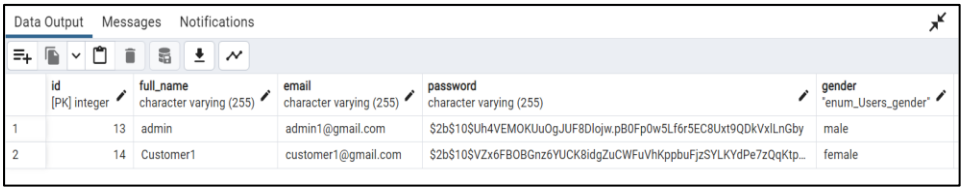


Gambar 4. 23 *Update Users*

Pada Gambar diatas, untuk memperbarui data *user*, *user* harus me*request* ke *http://localhost:3000/Users/* dengan *method POST*. lalu di dalam *body request*nya berisi informasi yang ingin diperbarui seperti *{"full_name", "email"}*. Jika berhasil maka akan menghasilkan *response* seperti ini :

Gambar 4. 24 *Update Users – Sukses*

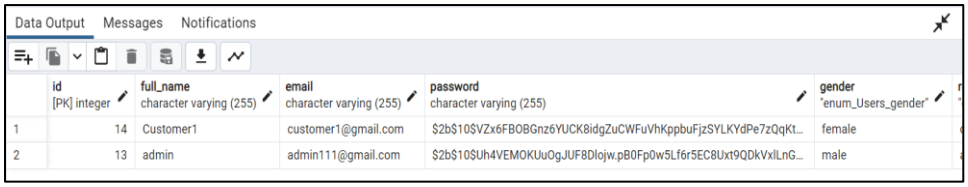
Dari yang sebelumnya seperti ini :



Gambar 4. 25 *Database Users - Before Update*

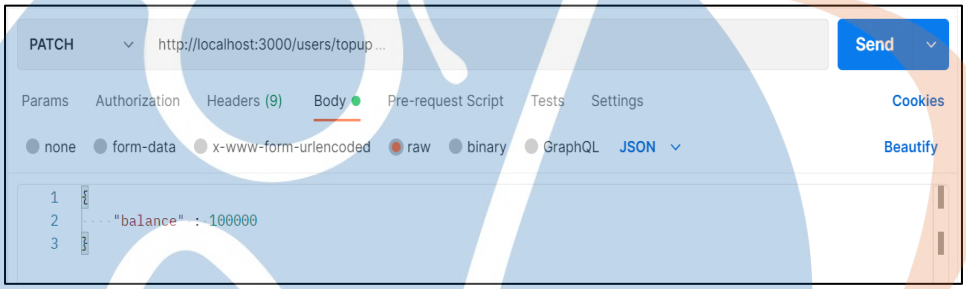
STT - NF

Menjadi seperti ini :



Gambar 4. 26 *Database Table User - After Updated*

3. *Top-Up* Saldo



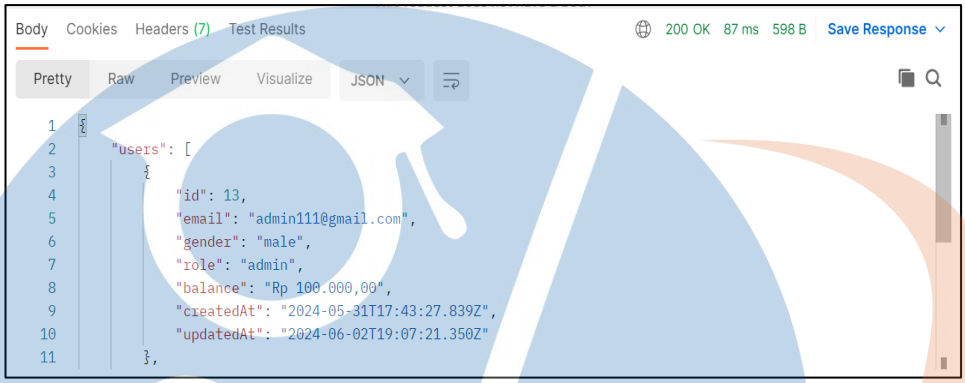
Gambar 4. 27 *Top-Up Saldo*

Pada Gambar 4.16 *Top-Up* Saldo diatas, baik pengguna dengan *role* admin maupun *customer* dapat melakukan *Top-Up* saldo. *User* dapat melakukan proses *Top-Up* saldo dengan cara me*request* ke *http://localhost:3000/Users/topup* dengan *method POST*. Lalu masukkan jumlah saldo yang ingin dimasukkan *{"balance"}*. Jika berhasil maka saldo akan bertambah dan akan menampilkan *response* seperti ini :

Gambar 4. 28 *Top-Up Saldo Sukses*

4. *Get Users*

Hanya pengguna dengan *role* admin yang dapat melihat data *Users* yang telah terdaftar. Untuk melihatnya admin dapat *merequest* ke alamat *http://localhost:3000/Users/* dengan *method GET*. Jika berhasil maka akan menampilkan *response* seperti dibawah ini :

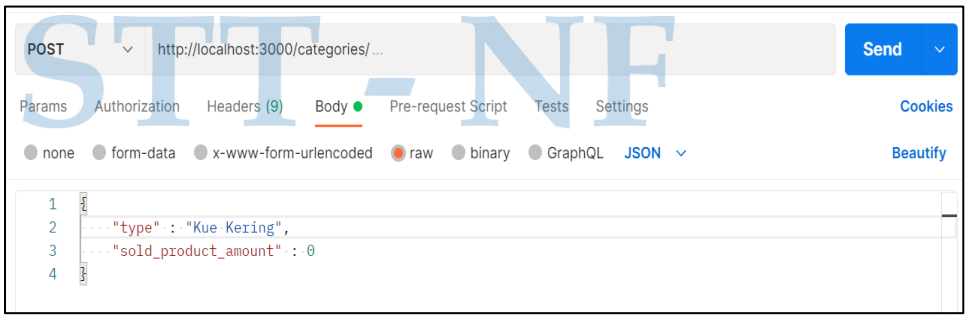


Gambar 4. 29 *Get Users*

4.3.3. Fungsi *Categories*

Table ini berfungsi untuk menyimpan informasi terkait kategori produk yang dijual. Hanya pengguna dengan *role* Admin yang dapat mengakses *table* ini. Berikut adalah beberapa fungsi *Categories* yang ada pada *Rest-API* Sistem informasi penjualan :

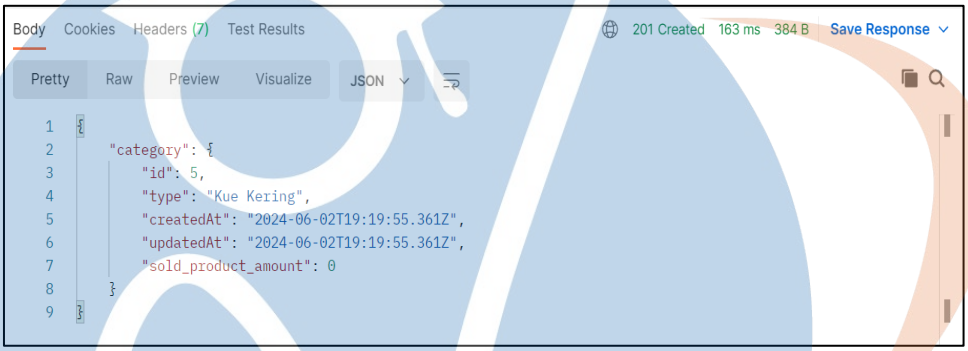
1. Membuat/*Create categories*



Gambar 4. 30 *Create categories*

STT-NF

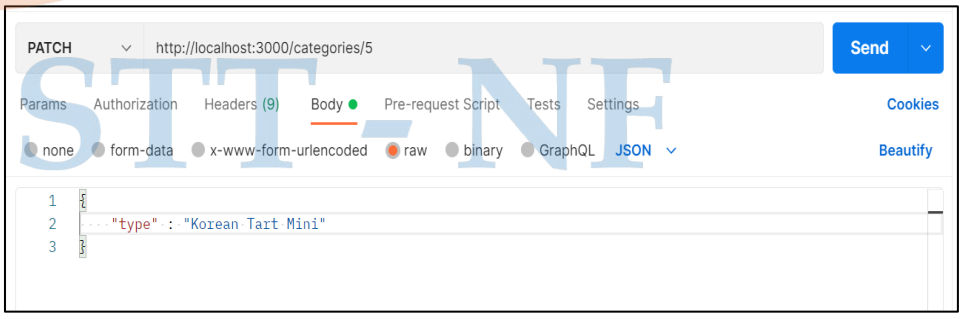
Pada Gambar diatas, untuk membuat *Categories user* harus me*request* ke *http://localhost:3000/Categories /* dengan *method POST*. lalu di dalam *body request*nya harus berisi *{"type", "sold_product_amount "}*. Pada bagian *sold_product_amount* awalnya akan di *set* menjadi 0. Nantinya *sold_product_amount* akan bertambah ketika ada pembelian *product* oleh *user*. Jika berhasil *create categories* maka akan menampilkan *response* seperti ini :



Gambar 4. 31 *Create categories Sukses*

2. Memperbarui/*Update Categories*

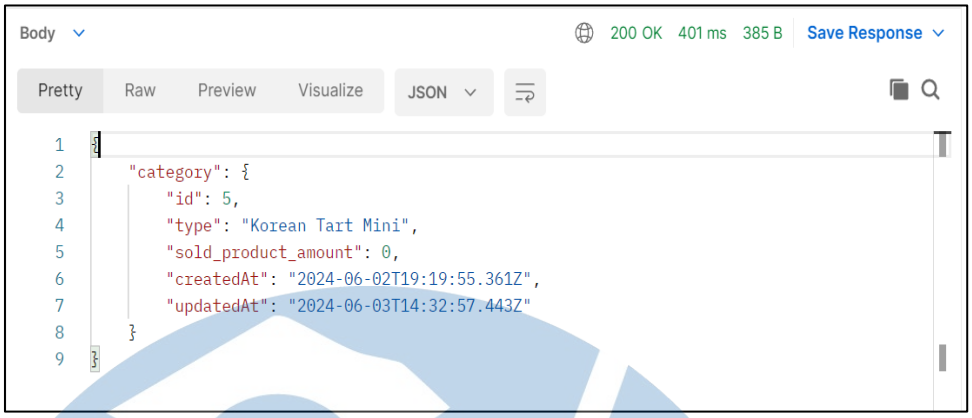
Untuk mengubah informasi pada *table Categories* , *user* harus me*request* ke http://localhost:3000/*Categories* /:*Categories* Id dengan *method PATCH*. Lalu dalam body *request*nya berisi informasi yang akan diperbarui seperti *{"type"}*.



Gambar 4. 32 *Update Categories*

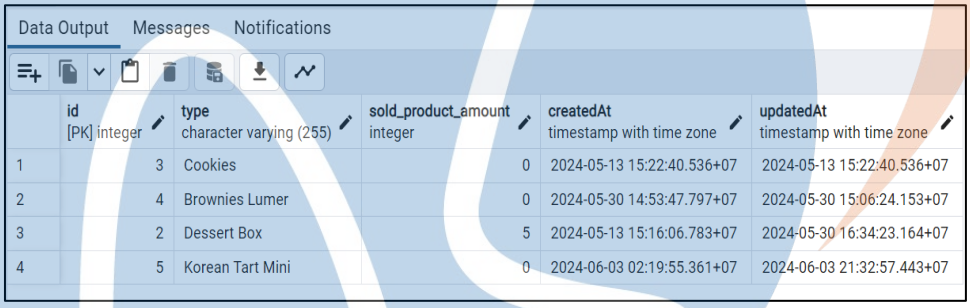
Jika data berhasil di*update* maka akan mendapatkan *response* seperti ini :

STT - NF



Gambar 4. 33 *Update Categories Succes*

Setelah nya, pada *table Categories* di *Database* akan terisi seperti ini :



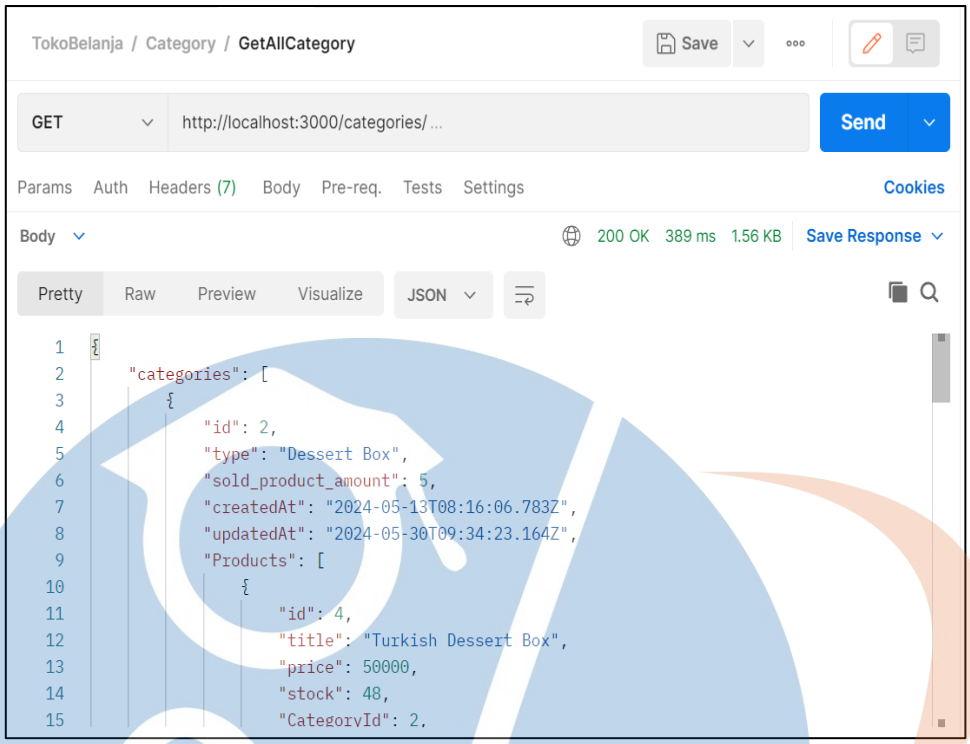
	id [PK] integer	type character varying (255)	sold_product_amount integer	createdAt timestamp with time zone	updatedAt timestamp with time zone
1	3	Cookies	0	2024-05-13 15:22:40.536+07	2024-05-13 15:22:40.536+07
2	4	Brownies Lumer	0	2024-05-30 14:53:47.797+07	2024-05-30 15:06:24.153+07
3	2	Dessert Box	5	2024-05-13 15:16:06.783+07	2024-05-30 16:34:23.164+07
4	5	Korean Tart Mini	0	2024-06-03 02:19:55.361+07	2024-06-03 21:32:57.443+07

Gambar 4. 34 *Database Table Categories*

Pada Gambar 4.20 dan 4.22 dapat dilihat bahwa *field "type"* yang sebelumnya berisi "Kue Kering" menjadi "*Korean Tart Mini*".

3. *Get All Categories*

Pada Gambar dibawah ini, untuk melihat seluruh data *Categories user* dapat melakukan suatu *request* ke http://localhost:3000/*Categories* dengan *method GET* seperti Gambar dibawah ini :



Gambar 4. 35 *Get All Categories*

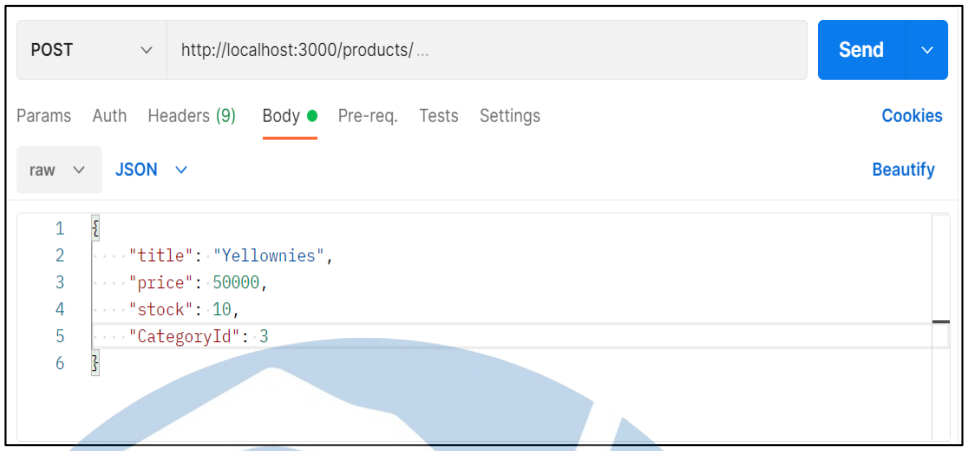
4.3.4. Fungsi *Products*

Table ini berfungsi untuk menyimpan informasi terkait Produk yang dijual. Hanya pengguna dengan *role* Admin yang dapat melakukan *Create*, Read, *Update* dan *Delete* pada *table* ini. sedangkan pengguna dengan *Role Customer* hanya dapat melihat daftar *Product* saja. Berikut adalah beberapa fungsi *Products* yang ada pada *Rest-API* Sistem informasi penjualan :

1. Membuat/*Create Products*

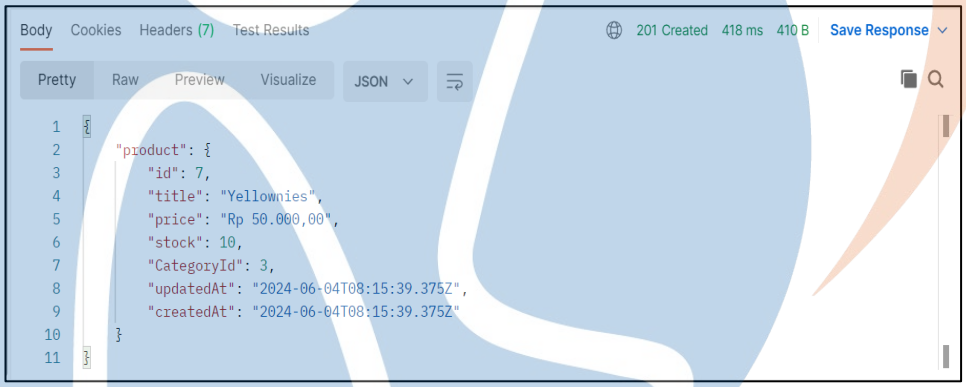
   Pada Gambar dibawah ini, untuk membuat *Categories user* harus me*request* ke *http://localhost:*3000/*products*/ dengan *method POST*. lalu di dalam *body request*nya harus berisi *{"title", "price", "stock", "Categories* Id*"}*.

STT - NF



Gambar 4. 36 *Create Products*

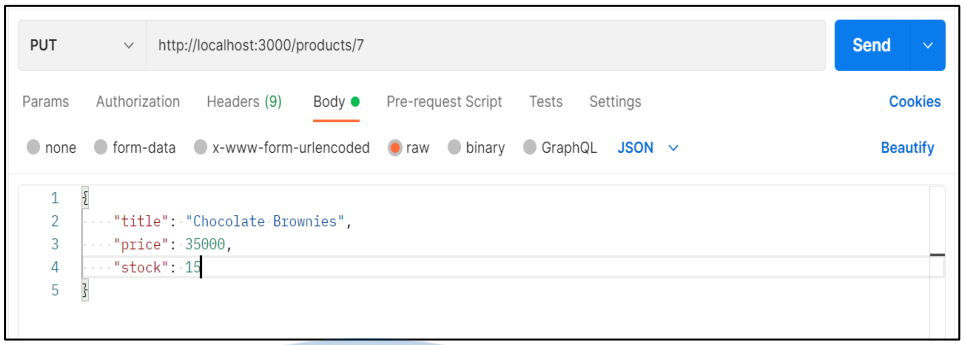
Jika berhasil *create products* maka akan menampilkan *response* seperti ini:



Gambar 4. 37 *Create Products Success*

2. Mengubah/*Update Products*

Untuk mengubah informasi pada *table product*, *user* harus me*request* ke *http://localhost:3000/products /:products Id* dengan *method PUT*. Lalu dalam *body request*nya berisi informasi yang akan diperbarui seperti *{"title", "price", "stock"}* seperti pada Gambar dibawah ini :

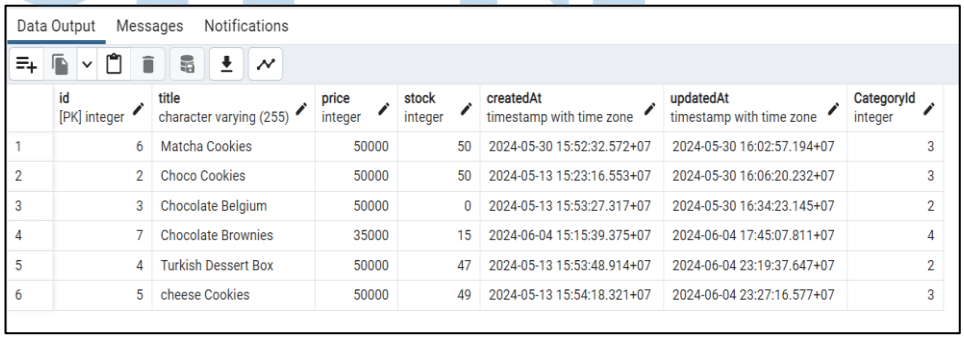


Gambar 4. 38 *Update Products*

Jika berhasil berubah, maka akan menghasilkan *response* seperti ini :

Gambar 4. 39 *Update Products Success*

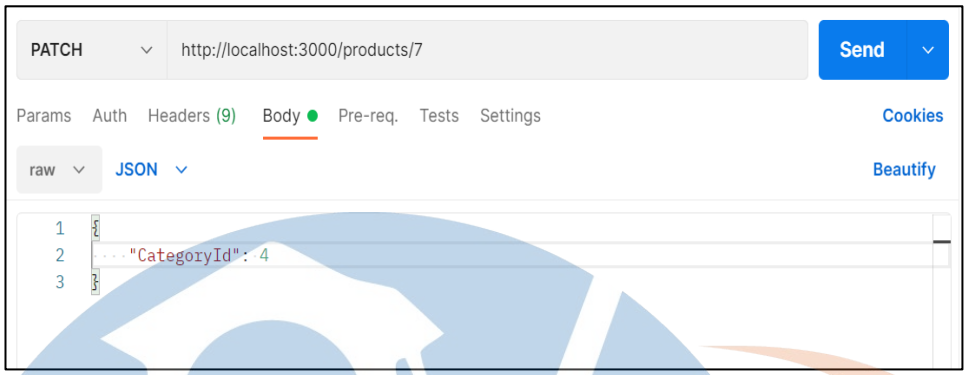
Pada Gambar 4.27 dan 4.28 dapat dilihat bahwa *field "title"* yang sebelumnya adalah "*Yellownies*" berubah menjadi *"Brownies Chocolate"*, *field "price"* yang awalnya "Rp 50.000,00" berubah menjadi "Rp 35.000,00", *field* "stock" yang awalnya "10" berubah menjadi "15". Berikut ini adalah tampilan *Database* pada *table Products* setelah diubah:



	id [PK] integer	title character varying (255)	price integer	stock integer	createdAt timestamp with time zone	updatedAt timestamp with time zone	CategoryId integer
1	6	Matcha Cookies	50000	50	2024-05-30 15:52:32.572+07	2024-05-30 16:02:57.194+07	3
2	2	Choco Cookies	50000	50	2024-05-13 15:23:16.553+07	2024-05-30 16:06:20.232+07	3
3	3	Chocolate Belgium	50000	0	2024-05-13 15:53:27.317+07	2024-05-30 16:34:23.145+07	2
4	7	Chocolate Brownies	35000	15	2024-06-04 15:15:39.375+07	2024-06-04 17:45:07.811+07	4
5	4	Turkish Dessert Box	50000	47	2024-05-13 15:53:48.914+07	2024-06-04 23:19:37.647+07	2
6	5	cheese Cookies	50000	49	2024-05-13 15:54:18.321+07	2024-06-04 23:27:16.577+07	3

Gambar 4. 40 *Database Table Products*

3. Mengubah/*Update CategoryId Products*



Gambar 4. 41 *Update CategoryId*

Jika ingin mengubah *Category*Id *table product*, *user* harus me*request* ke http://localhost:3000/*products /:products Id* dengan *method PATCH*. Lalu dalam *body request*nya berisi *Category*Id yang akan diperbarui *{"Category*Id*"}*. Jika id berhasil diubah maka akan me*response* seperti pada Gambar dibawah ini :

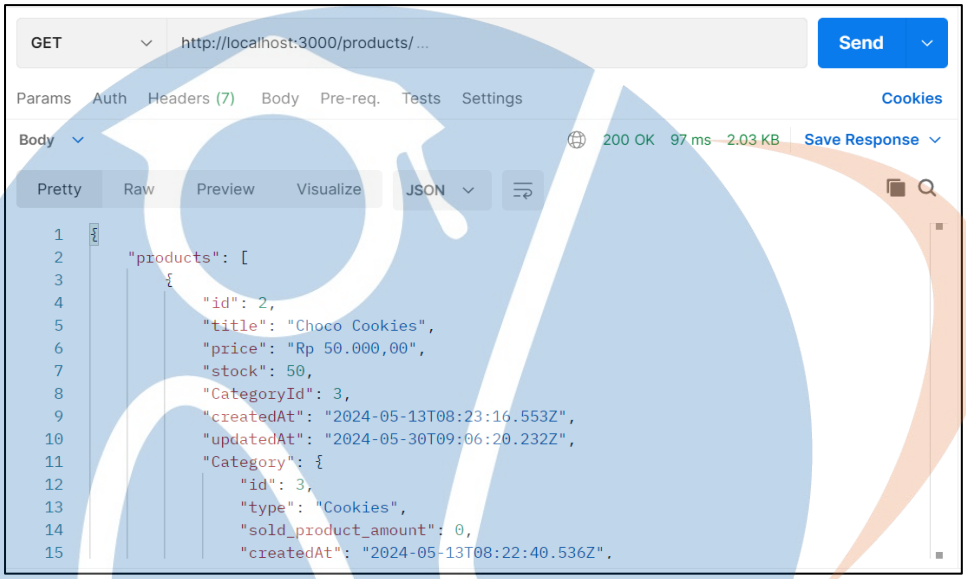
Gambar 4. 42 *Update Categor Success*

Pada Gambar 4.29 dan 4.30 dapat dilihat bahwa *field "CategoryId"* yang sebelumnya berisi "3" telah berubah menjadi "4".

STT - NF

4. *Get All Products*

User dengan *role* admin maupun *customer* dapat melihat seluruh data *Products* . Gambar dibawah ini menjelaskan bahwa untuk melihat seluruh data *Products* *user* dapat melakukan suatu *request* ke *http://localhost:3000/products /* dengan *method* *GET* seperti Gambar dibawah ini



Gambar 4. 43 *Get All Products*

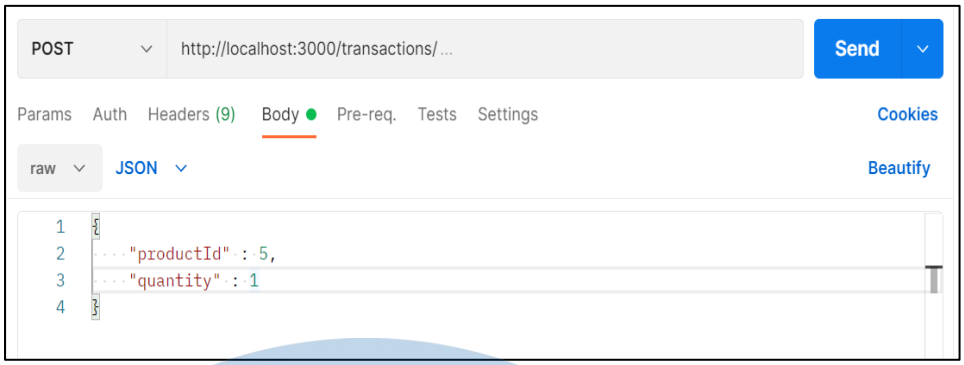
4.3.5. Fungsi *Transaction History*

Table ini berfungsi untuk menyimpan informasi terkait Transaksi *User*. Pengguna dengan *Role Customer* dapat melakukan transaksi dan melihat riwayat transaksinya sendiri. Sedangkan admin dapat melihat seluruh riwayat transaksi *user*. Berikut adalah beberapa Fungsi *Transaction History* yang ada pada *Rest-API* Sistem informasi penjualan :

1. Melakukan/*Create* Transaksi

Pada Gambar dibawah ini, untuk melakukan Transaksi *user* harus me*request* ke *http://localhost:3000/transactions /* dengan *method* *POST*. lalu di dalam *body requestnya* harus berisi *{"product*Id*", "quantity"}*.

STT-NF



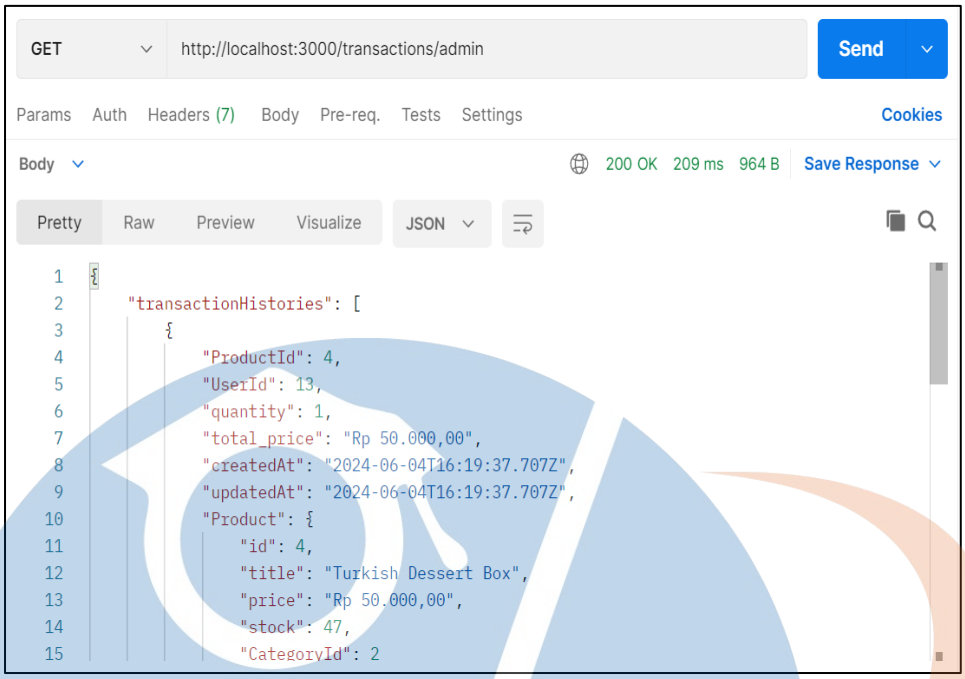
Gambar 4. 44 *Create Transactions*

Jika berhasil maka akan menampilkan *response Transaction Bill* seperti ini:

Gambar 4. 45 *Transaction success*

2. Melihat data Transaksi

Pada Gambar dibawah ini, hanya *user* dengan *role* admin yang dapat melihat seluruh data riwayat transaksi. Gambar dibawah ini menjelaskan bahwa untuk melihat seluruh data riwayat transaksi *user* dapat melakukan suatu *request* ke *http://localhost:*3000/*transactions /admin* dengan *method GET* seperti Gambar dibawah ini :



Gambar 4. 46 *Get All Transactions*

Pada Gambar dibawah ini, *user* dapat melihat data riwayat transaksi yang sudah dilakukan. Gambar dibawah ini menjelaskan bahwa untuk melihat data riwayat transaksi *user* dapat melakukan suatu *request* ke *http://localhost:3000/transactions /user* dengan *method GET seperti* Gambar *dibawah ini*

Gambar 4. 47 *Get Transaction History - User*

STT - NF

Berikut ini adalah tampilan dari *Database Table Transaction History*

Data Output | Messages | Notifications

	id [PK] integer	quantity integer	total_price integer	createdAt timestamp with time zone	updatedAt timestamp with time zone	ProductId integer	UserId integer
1	5	1	50000	2024-06-04 23:19:37.707+07	2024-06-04 23:19:37.707+07	4	13
2	6	1	50000	2024-06-04 23:27:16.617+07	2024-06-04 23:27:16.617+07	5	15

Gambar 4. 48 *Database Transaction History*

STT - NF

4.4. Pengujian dan Evaluasi

4.4.1 Black Box Testing

Black Box adalah metode pengujian yang dilakukan bertujuan untuk memastikan sistem berjalan dengan baik dan menghasilkan *output* sesuai rancangan atau tidak. Berikut ini adalah hasil pengujian *Black Box testing* yang telah dilakukan.

Tabel 4. 7 Pengujian *Black Box Testing*

No	Pengujian	Http *Method* /End Point	Input	Hasil yang diharapkan	Kesimpulan	Skor
1	*Register* dengan *request body* yang sesuai	***POST,*** *http://localhost:*3000*/Users/register*	{ "full_name": "admin", "email": "admin1@gmail.com", "password" : "admin123", "gender" : "male", "role" : "admin" }	Akan menghasilkan *response Code* : "201" dan menampilkan akun telah terbuat	Sesuai	100%

STT - NF

2	*Register* dengan *request body* yang tidak sesuai	***POST,***
		
		*http://localhost:*3000*/Users/register*
			<pre>{
			"full_name": "admin",
			"email": "",
			"password" : "admin123",
			"gender" : "male",
			"role" : "admin"
			}</pre>
				Akan menghasilkan *response Code* : "400" dengan *message* "*validation isEmail on email failed*"	Sesuai	100%

3	*Login* dengan akun yang benar	***POST,***
		
		*http://localhost:*3000*/Users/login*
			<pre>{
			"full_name": "admin",
			"email": "admin1@gmail.com",
			"password" : "admin123"
			}</pre>
				Akan menghasilkan *response Code* : "200" dan menghasilkan token *login*	Sesuai	100%

4	*Login* dengan *request body* yang salah	***POST,***
		
		*http://localhost:*3000*/Users/login*
			<pre>{
			"full_name": "admin lain",
			"email": "adminLain@gmail.com",
			"password" : "admin123"
			}</pre>
				Akan menghasilkan *response Code* : "400" dan *message* "*invalid email/password*"	Sesuai	100%

5	Melihat data *Users* dengan akun admin	***GET,*** *http://localhost:*3000*/Users/*	Tidak butuh body request	Akan menghasilkan *response Code* : "200" dan menampilkan data *Users*	Sesuai	100%

STT - NF

6	Melihat data *Users* dengan token yang salah	***GET,*** *http://localhost:*3000*/Users/*	Tidak butuh body request	Akan menghasilkan *response Code* : "401" dan *message "unauthorized"*	Sesuai	100%

7	Memperbarui data *Users* dengan akun admin	***PUT,***
		
		*http://localhost:*3000*/Users/*
			{
			"id" : 2,
			"full_name": "admin"
			}
				Akan menghasilkan *response Code* : "200" dan menampilkan data yang telah di *update*	Sesuai	100%

8	Menghapus data *Users*	***DELETE,*** *http://localhost:*3000*/Users/*:id	Tidak butuh body request	Akan menghasilkan *response Code : "*200*"* dengan *message "Your account has been Success fully deleted"*	Sesuai	100%

9	Melakukan *Top-up* saldo dengan akun admin	***PATCH,***
		
		*http://localhost:*3000*/Users/top up*
			{
			"balance" : 100000
			}
				Akan menghasilkan *response Code : "*200*"* dengan *message "Your balance has been Success fully updated"*	Sesuai	100%

STT - NF

10	Melakukan *Top-up* saldo dengan akun *customer*	***PATCH,***
		
		*http://localhost:*3000/*Users/top up*
			{
			"balance" : 100000
			}
				Akan menghasilkan *response Code : "*200*"* dengan *message "Your balance has been Success fully updated"*	Sesuai	100%

11	Membuat data *Categories* dengan akun admin	***POST,***
		
		*http://localhost:*3000/*Categories /*
			{
			"type" : "Brownies",
			"sold_product_amount" : 0
			}
				Akan menghasilkan *response Code* "201" dan menampilkan data yang baru dibuat	Sesuai	100%

12	Membuat data *Categories product* dengan token yang salah	***POST,***
		
		*http://localhost:*3000/*Categories /*
			{
			"type" : "Brownies",
			"sold_product_amount" : 0
			}
				Akan menghasilkan *response Code* : "401" dan *message* "unauthorized"	Sesuai	100%

13	Memperbarui data *Categories* sesuai *request body*	***PATCH,***
		
		*http://localhost:*3000/*Categories /*
		
		:Categories Id
			{
			"type" : "Brownies Lumer"
			}
				Akan menghasilkan *response Code* "200" dan menampilkan data yang telah di *update*	Sesuai	100%

STT - NF

14	Melihat data *Categories* dengan akun admin	***GET,*** *http://localhost:*3000*/Categories /*	Tidak butuh body request	Akan menghasilkan *response Code* : "200" dan menampilkan seluruh data *Categories*	Sesuai	100%
15	Melihat data *Categories* dengan token yang salah	***GET,*** *http://localhost:*3000*/Categories /*	Tidak butuh body request	Akan menghasilkan *response Code* : "401" dan *message* "*unauthorized*"	Sesuai	100%

16	Membuat data *Products* dengan akun admin	***POST,***
		
		*http://localhost:*3000*/products /*
			{
			"title": "cheese Cookies",
			"price": 50000,
			"stock": 50,
			"CategoryId": 3
			}
				Akan menghasilkan *response Code :* "201" dan menampilkan data yang baru dibuat	Sesuai	100%

17	Membuat data *Products* dengan token yang salah	***POST,***
		
		*http://localhost:*3000*/products /*
			{
			"title": "cheese Cookies",
			"price": 50000,
			"stock": 50,
			"CategoryId": 3
			}
				Akan menghasilkan *response Code* : "401" dan *message* "unauthorized"	Sesuai	100%

STT - NF

18	Memperbarui data *Products* sesuai *request body*	***PUT***, *http://localhost:*3000/*products* / *:produtc*Id	{ "title": "Matcha Cookies", "price": 50000, "stock": 50 }	Akan menghasilkan *response Code* : "200" dan menampilkan data yang telah di *update*	Sesuai	100%
19	Memperbarui data *Products* dengan id yang salah	***PUT***, *http://localhost:*3000/*products* / *:produtc*Id	{ "title": "Matcha Cookies", "price": 50000, "stock": 50 }	Akan menghasilkan *response* Code "404" dengan *message "Product Not Found"*	Sesuai	100%
20	Memperbarui data *Categories* Id *Products*	***PATCH***, *http://localhost:*3000/*products* / *:produtc*Id	{ "CategoryId": 3 }	Akan menghasilkan *response Code* : "200" dan menampilkan data yang telah di *update*	Sesuai	100%
21	Melihat data *Products* dengan akun admin	***GET,*** *http://localhost:*3000/*products* /	Tidak butuh body request	Akan menghasilkan *response Code* : "200" dan menampilkan seluruh data *Products*	Sesuai	100%
22	Melihat data *Products* dengan	***GET,*** *http://localhost:*3000/*products* /	Tidak butuh body request	Akan menghasilkan *response Code* : "200" dan menampilkan seluruh data *Products*	Sesuai	100%

STT - NF

	akun *Customer*					

23	Melakukan transaksi dengan akun admin	***POST,***
		
		*http://localhost:*3000*/transactions /*
			{
			"productId" : 3,
			"quantity" : 1
			}
				Akan menghasilkan *response Code* "201" dengan *message "You Have Success fully purchase the product"* dan menampilkan *Transaction Bill* serta mengurangi *stock* produk yang ada pada *table Products*	Sesuai	100%

24	Melakukan transaksi dengan akun *Customer*	***POST,***
		
		*http://localhost:*3000*/transactions /*
			{
			"productId" : 4,
			"quantity" : 1
			}
				Akan menghasilkan *response Code* "201" dengan *message "You Have Success fully purchase the product"* dan menampilkan *Transaction Bill* serta mengurangi *stock* produk yang ada pada *table Products*	Sesuai	100%

STT - NF

25	Melakukan transaksi jika produk habis	***POST,***
		
		*http://localhost:*3000*/transactions /*
			{
			"productId" : 4,
			"quantity" : 1
			}
				Akan menghasilkan *response Code* “400” dengan *message “Product is out of stock”*	Sesuai	100%

STT - NF

Berdasarkan tabel pengujian, telah dilakukan 25 uji fungsional terhadap *Rest-API* pada Sistem Informasi Penjualan dengan rumus sebagai berikut:

$$\text{Persentase Keberhasilan} = \left(\frac{\text{Jumlah Total Kasus Uji}}{\text{Jumlah Kasus Uji yang Lulus}}\right) \times 100\%$$

$$\text{Persentase Keberhasilan} = \left(\frac{25}{25}\right) \times 100\% = 100\%$$

Hasil pengujian menunjukkan bahwa *Rest-API* Sistem Informasi Penjualan telah memenuhi seluruh persyaratan fungsional yang diuji dengan persentase keberhasilan mencapai 100%. Hal ini menegaskan bahwa perangkat lunak berfungsi sesuai dengan yang diharapkan dalam semua skenario yang diuji. Keberhasilan pengujian ini merupakan indikator positif yang menunjukkan bahwa penulis berhasil merancang dan mengimplementasikan *Rest-API* dengan baik, memastikan bahwa semua fungsionalitas yang diharapkan bekerja tanpa adanya kesalahan. Dengan demikian, *Rest-API* Sistem Informasi Penjualan ini diharapkan dapat dimanfaatkan pada berbagai platform, memberikan kemudahan pengembangan aplikasi berbasis website maupun mobile, dan memungkinkan integrasi yang mulus dengan berbagai platform dan aplikasi eksternal. Dikarenakan *Rest-API* Sistem Informasi Penjualan belum memiliki tampilan *User Interface* maka peneliti memilih metode pengujian *Black Box testing* karena metode ini lebih tepat untuk mengevaluasi fungsionalitas Rest-API tanpa terpaku pada detail implementasi internal seperti *UI UX*.

STT - NF

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil pengujian Rancang Bangun *Rest-API* Pada Sistem Informasi Penjualan Menggunakan *Express JS* Pada Toko Kue Variant Cake maka didapatkan kesimpulan sebagai berikut :

a. Perancangan *Rest-API* pada Sistem Informasi Penjualan melibatkan tahapan pengumpulan data melalui studi literatur, wawancara, dan observasi. Desain sistem menggunakan pemodelan *Use case* Diagram dan metode *Waterfall* , serta implementasi program menggunakan Bahasa pemrograman *Javascript Framework Express JS* dengan *Database PostgreSQL*. Evaluasi sistem mencakup metode pengujian *Black Box testing* guna menguji fungsionalitas sistem apakah berjalan dengan baik atau tidak.
b. Implementasi fungsi pada *Rest-API* Sistem Informasi Penjualan telah berfungsi dengan baik sehingga memungkinkan pengguna untuk mengelola data produk secara *real-time* dengan mudah dan efisien. Hal tersebut sesuai dengan hasil pengujian yang dilakukan untuk fungsi *Rest-API* Sistem Informasi Penjualan yang memperoleh persentase hasil keberhasilan mencapai 100%. Ini menunjukkan bahwa sistem dapat berjalan dengan baik dan dapat mempermudah dalam proses jual beli.

STT - NF

5.2. Saran

Adapun Rancang Bangun *Rest-API* pada Sistem Informasi Penjualan menggunakan *Express JS* perlu pengembangan lebih lanjut karena masih memiliki beberapa kekurangan. Berikut adalah saran dan masukan terkait pengembangan *Rest-API* pada Sistem Informasi Penjualan menggunakan *Express JS* untuk penelitian mendatang, diantaranya :

1. Fungsi keamanan belum sepenuhnya baik oleh karena itu Perlu adanya peningkatan keamanan sistem dengan tidak hanya mengandalkan keamaan bawaan *framework*.
2. Sampai saat ini, pengembangan sistem hanya mencakup *Rest-API*, dan belum mencakup tampilan antarmuka pengguna atau *frontend* dari aplikasi. Pengembangan lebih lanjut diperlukan untuk menyertakan aspek visual dan interaktif dari aplikasi, sehingga pengguna dapat berinteraksi dengan sistem secara lebih efektif.
3. Diharapkan *Rest-API* Sistem Informasi Penjualan bisa dikembangkan lebih lanjut tidak hanya menjadi sistem pengelolaan produk pada Sistem Informasi Penjualan tetapi bisa dimanfaatkan untuk Sistem Pengelolaan yang lain.

STT - NF

DAFTAR PUSTAKA

[1] A. Haris, S. S2, H. Ekonomi, S. Uin, S. Gunung, and D. Bandung, "Pelaksanaan Jual Beli Dengan Menggunakan Akad As-Salam Ditinjau Dari Prinsip Tabadul Al-Manafi," vol. 15, no. 1, 2019.

[2] I. Kurniawan and F. Rozi, "*REST API* Menggunakan Node JS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android," vol. 1, no. 4, pp. 127–132, 2020.

[3] A. Mega, P. Pratiwi, and D. K. Belitang, "Pengaruh Jual Beli Online *E-commerce* Shopee Terhadap Minat Beli Saat Pandemi Covid-19 Pada Masyarakat Milenial," *Journal of Management and Creative Business (JMCBUS)*, vol. 1, no. 2, 2023.

[4] R. Sangga Rasefta and S. Esabella, "Sistem Informasi Akademik Smk Negeri 3 Sumbawa Besar Berbasis Web," *Jurnal JINTEKS*, vol. 2, no. 1, p. 50, 2020.

[5] K. Amira, "Pengertian Sistem Informasi: Tujuan dan Komponennya - Gramedia." Accessed: Apr. 18, 2024. [Online]. Available: https://www.gramedia.com/literasi/sistem-informasi/

[6] Dicoding Intern, "Apa itu *Web* Service? Beserta Pengertian dan Contohnya - Dicoding Blog." Accessed: Apr. 03, 2024. [Online]. Available: https://www.dicoding.com/blog/apa-itu-web-service/

[7] A. B. Prakoso and W. Santiyasa, "Rancang Bangun *Web* Service API Aplikasi Jual Beli Barang Bekas Menggunakan Metode REST," *JNATIA*, vol. 1, no. 1, 2022.

[8] I. Kurniawan and F. Rozi, "*REST API* Menggunakan Node JS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android," vol. 1, no. 4, pp. 127–132, 2020, [Online]. Available: http://jurnal-itsi.org

[9] Postman, "Postman API Platform | Sign Up for Free." Accessed: May 20, 2024. [Online]. Available: https://www.postman.com/

[10] K. Javista, "Implementasi Keamanan Restfull API Dengan Firebase Authentication Pada Aplikasi Presensi," 2021.

STT-NF

[11] PostgreSQL, “PostgreSQL: The world’s most advanced open source *Database*.” Accessed: May 20, 2024. [Online]. Available: https://www.postgresql.org/

[12] F. Otra, “*Json Web token*.” Accessed: Apr. 18, 2024. [Online]. Available: http://lea.si.fti.unand.ac.id/2022/03/json-web-token/

[13] R. Setiawan, “Apa itu Framework? Developer Wajib Tahu - Dicoding Blog.” Accessed: Apr. 18, 2024. [Online]. Available: https://www.dicoding.com/blog/apa-itu-framework/

[14] C. Henny, “(9) Apa itu *Express JS*? Pengertian, Manfaat, Kelebihan, Cara Kerja | LinkedIn.” Accessed: Apr. 18, 2024. [Online]. Available: https://www.linkedin.com/pulse/apa-itu-express-js-pengertian-manfaat-kelebihan-cara-kerja-henny/

[15] R. Setiawan, “Apa Itu MVC? Pahami Konsepnya dengan Baik - Dicoding Blog.” Accessed: Apr. 18, 2024. [Online]. Available: https://www.dicoding.com/blog/apa-itu-mvc-pahami-konsepnya/

[16] A. B. Prakoso and W. Santiyasa, “Rancang Bangun *Web* Service API Aplikasi Jual Beli Barang Bekas Menggunakan Metode REST,” *JNATIA*, vol. 1, no. 1, 2022.

[17] Dicoding Intern, “Apa itu *UML*? Beserta Pengertian dan Contohnya - Dicoding Blog.” Accessed: Apr. 18, 2024. [Online]. Available: https://www.dicoding.com/blog/apa-itu-*UML*/

[18] T. Snadhika Jaya, P. Studi Manajemen Informatika, J. Ekonomi dan Bisnis, and P. Negeri Lampung JlnSoekarno, “Pengujian Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung),” *Jurnal Informatika: Jurnal Pengembangan IT (JPIT)*, vol. 03, no. 02, 2018.

[19] N. Syavira, “Pengembangan Media Pembelajaran Berbasis Powerpoint Interaktif Materi Sistem Pencernaan Manusia Untuk Siswa Kelas V Sd,” *OPTIKA: Jurnal Pendidikan Fisika*, vol. 5, no. 1, 2021.

[20] K. ’ Afiifah, Z. Fira Azzahra, A. D. Anggoro, D. Redaksi, R. Akhir, and D. Online, “Analisis Teknik Entity-*Relationship* Diagram dalam Perancangan

Database: Sebuah Literature Review," *JURNAL INTECH*, vol. 3, no. 1, pp. 8–11, 2021.

STT - NF

STT - NF