Jurnal Integrasi
Vol. 14 No. 2, Oktober 2022, 165-173
e-ISSN: 2548-9828

Article History
Received September 2022
Accepted October 2022

Penerapan Pendekatan Design-First dalam Pengembangan Sistem Manajemen Penagihan Transaksi Berbasis Web API

Metta Santiputri*, Dzulfikar*

* Politeknik Negeri Batam
Program Studi Teknik Informatika
Parkway Street, Batam Centre, Batam 29461, Indonesia
E-mail: metta@polibatam.ac.id.ac.id, kardzulfi1805@gmail.com

Abstrak

Transaksi merupakan kegiatan yang umum kita lakukan, namun untuk transaksi antar bisnis memerlukan dokumen transaksi dalam kegiatannya. Banyak pebisnis masih membuat dokumen tersebut secara manual sehingga menyebabkan banyak masalah sehingga Sistem Manajemen Penagihan dapat menyelesaikan masalah yang terjadi dikarenakan telah berbasis sistem. Dalam pengembangan, dokumentasi menjadi hal yang tidak dipentingkan padahal dokumentasi menggambarkan sistem dengan baik. Hal ini terjadi karena kebanyakan pengembang menggunakan penerapan *code-first* dimana pengembangan lebih penting dibandingkan sehingga menyebabkan masalah antara tim front-end dengan *back-end*. Oleh karena itu, diterapkanlah metode *design-first* dalam pengembangan sistem ini. Metode pengembangan yang dipakai adalah metode agile karena metode ini fleksibel dan cocok dalam menghadapi perubahan yang terjadi dalam bentuk apapun. Pengujian sistem menggunakan metode *blackbox* dengan teknik *equivalence partitions* yang bertujuan untuk ketidaksesuaian dan kesalahan fungsionalitas pada sistem. Setelah dilakukan pengujian, hasil yang didapatkan bahwa kebutuhan telah terpenuhi dan tidak ditemukan adanya kesalahan fungsionalitas sehingga sistem berjalan dengan baik dan telah sesuai.

Kata kunci: Penagihan Transaksi, Web API, Metode Agile, Pendekatan *Design-First*, *Blackbox*

Abstract

Transactions are a common activity that we do, but transactions between businesses require transaction documents in their activities. Many business people still make these documents manually therefore causing a lot of problems. In order to solve it, Invoicing Transaction Management System is able to solve problems that occur because it is system-based. In development, documentation is not important even though it describes the system well. This happens because most developers use code-first approach where development is more important than causing problems between the front-end and back-end teams. In order to fix it, the design-first approach is used in development. The development method used is agile method because this method is flexible and suitable for dealing with changes that occur in any form. System testing is using the blackbox method with equivalence partitions technique which aims to find discrepancies and functionality errors in the system. After testing, the results obtained are that the requirements have been met and there are no functionality errors found, therefore the systemruns well and is in accordance with.

Keywords: Invoice, Web API, Agile Method, Design-First Approach, Blackbox

1. Pendahuluan

Transaksi merupakan salah satu kegiatan yang umum dikehidupan kita. Biasanya alur transaksi yang kita lakukan yaitu membeli barang atau jasa, lalu membayar nya ketika selesai. Namun untuk pebisnis, alur yang terjadi bisa lebih sulit dari yang kita perkirakan karena terdapat penagihan. Biasanya untuk pebisnis, alur yang sering terjadi dimulai dari pelanggan yang ingin membeli, lalu pebisnis memberikan dokumen berupa quotation atau dokumen penawaran dari barang yang ingin dibeli. Apabila setuju, maka pelanggan akan memberikan dokumen berupa *purchase order* atau dokumen permintaan atas kebutuhan barang yang ingin dibeli. Setelah itu, pebisnis akan memberikan *sale invoice* atau dokumen faktur yang diberikan kepada pelanggan yang berisi informasi lengkap seperti

barang yang dibeli, kuantitas, serta harga yang diberikan [1].

Pada zaman sekarang, dokumen transaksi tersebut dibuat secara digital dengan menggunakan software teks seperti Microsoft Excel dan Microsoft Word. Namun, tanpa sistem yang dibuat khusus untuk menjalankan kegiatan ini, maka akan berkurang efektivitas baik dari segi pengguna maupun alur kerja. Pada penelitian yang dilakukan oleh Aenun Najib dan Fuaida Nabyla dengan judul "Sistem Informasi Penagihan (*Invoice*) Berbasis Desktop Menggunakan Metode *Extreme Programing*", disimpulkan bahwa sistem informasi yang masih manual menghasilkan banyak kekurangan, seperti memerlukan waktu yang cukup lama dalam memproses data, ketidakakuratan dari proses serta keterlambatan dalam memberikan informasi maupun laporan [2]. Penelitian tersebut menggambarkan bahwa kemajuan teknologi informasi sangat berdampak pada sektor bisnis baik dari segi kecepatan, ketepatan dan keakuratan. Kesalahan yang terjadi tanpa penggunaan sistem informasi yang sesuai, dapat merusak proses bisnis yang sedang berjalan. Sehingga, peranan sistem informasi sangat besar untuk memajukan dan memudahkan proses bisnis dengan harapan agar membuat konsumen nyaman dan menarik perhatian konsumen baru. Maka, penelitian tersebut membuat sistem dengan fitur yaitu mengelola data pelanggan serta membuat invoice untuk diberikan kepada pelanggan.

Sebelum dibuatnya *invoice*, perlu adanya penawaran (*quotation*) kepada pelanggan yang di mana, quotation merupakan rincian mengenai produk yang diinginkan oleh pelanggan. *Quotation* merupakan salah satu dokumen penting dalam proses bisnis namun sayangnya banyak perusahaan masih belum menggunakan sistem informasi. Berdasarkan penelitian yang dilakukan oleh Krisna Adi Wicaksono dengan judul "Aplikasi Pembuatan Quotation Berbasis Web Dengan Menggunakan Angularjs 2 dan Firebase di CV Aditex Bangun Cipta", banyak perusahaan masih menggunakan Microsoft Word dalam pembuatan sebuah *quotation*. Hal ini menimbulkan masalah di mana belum terdapat format yang jelas dalam penulisannya, sehingga setiap kali pelanggan ingin dibuatkan *quotation*, format yang diberikan sering berubah. Selain itu, *quotation* sangat berhubungan dengan produk beserta stoknya, jika hanya satu orang yang mengetahui banyak stok produk beserta harganya, maka data yang diberikan bisa tidak valid. Sehingga penelitian tersebut membuat sistem dengan fitur manajemen produk dan pembuatan quotation [3].

Dalam transaksi, tidak jauh dari kebutuhan produk yang di mana produk merupakan barang yang akan dijual kepada pelanggan. Produk juga sebagai barang yang dicantumkan ke dalam *quotation* dan *invoice*. Terkadang, perusahaan memesan produk dari pihak penyuplai yang di mana saat pemesanan, diperlukan adanya purchase order kepada pihak penyuplai sebagai bentuk komunikasi yang terekam dan dapat dijadikan sebagai acuan pemesanan serta bukti apabila terdapat kesalahan pemesanan maupun kesalahan pemenuhan pesanan [4]. Berdasarkan penelitian yang dilakukan oleh Dina Tauhida dan Arinal Muna dengan judul "Rancang Bangun Aplikasi Purchase Order Pada Unit Purchasing PT. XYZ", perusahaan yang sering memesan produk dari penyuplai sangat membutuhkan purchase order sebagai bukti pemesanan. Namun, perusahaan masih menggunakan pencatatan manual dalam melakukan pencatatan transaksi pembelian penyuplai. Pencatatan transaksi tersebut masih dibuat menggunakan aplikasi Microsoft Excel serta file–file yang dihasilkan disimpan secara terpisah. Sehingga, proses pencatatan yang manual tersebut akan memakan waktu yang lebih lama jika dibandingkan dengan proses pencatatan yang sudah menggunakan sistem informasi. Pencatatan yang belum terintegrasi mengakibatkan dapat terjadi kemungkinan kesalahan pencatatan data yang tidak sesuai sehingga harus, mengulangi pembuatan *purchase order*-nya. Maka dari itu, hasil penelitian tersebut membuat aplikasi *purchasing* dengan fitur mengelola *purchase order*, mengelola data penyuplai dan mengelola data produk [5].

Penelitian–penelitian sebelumnya yang telah disampaikan, semua berfokus pada penyelesaian salah satu bagian dalam proses transaksi bisnis. Sedangkan, penelitian ini ditujukan untuk menggabungkan semua bagian proses bisnis agar terintegrasi dikarenakan, sistem yang terintegrasi dapat memberikan efek yang baik kepada pengusaha. Penelitian yang dilakukan oleh Raelida Turnip, dkk dengan judul "Analisis dan Perancangan Sistem Informasi Pembelian, Penjualan dan Persediaan pada UD. Parjuma Sonari", serta penelitian yang dilakukan oleh Triyugo Winarko dengan judul "Perancangan Sistem Informasi Pembelian, Penjualan Dan Persediaan Pada Pt. Indo Global", menjelaskan bahwa proses bisnis yang terintegrasi ke dalam satu sistem informasi dapat membantu kegiatan bisnis di perusahaan seperti, proses bisnis menjadi lebih efisien, data sudah terintegrasi antar tiap bagian, serta terjamin keakuratan data yang dikelola sehingga, perusahaan dapat lebih mengalokasikan sumber daya mereka untuk perkembangan perusahaan [6] [7].

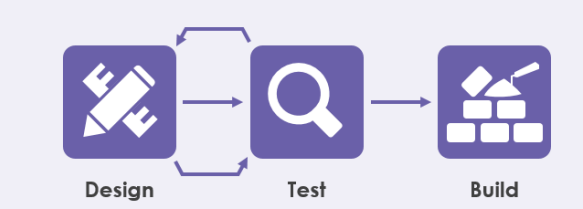
Saat mengembangkan sistem secara berkelompok, khususnya pengembangan sistem berbasis web API, maka akan ada tim untuk mengerjakan bagian *front-end* dan tim untuk mengerjakan bagian back-end. Tim *front-end* mengembangkan sistem dari segi tampilan serta memunculkan data yang diberikan back-end, sedangkan tim *back-end* mengembangkan sistem dari segi fitur serta memberikan data untuk ditampilkan *front-end* [8]. Salah satu hal yang penting dalam mengembangkan sistem secara berkelompok adalah kerja sama serta komunikasi antar anggota tim. Tim

front-end harus menunggu dari tim *back-end* agar dapat bekerja serta harus tahu apa saja data yang harus didapatkan serta ditampilkan oleh tim back-end, sehingga untuk mengurangi waktu maka sangat diperlukannya dokumentasi [9][10]. Dokumentasi merupakan hal yang sangat dibutuhkan karena menggambarkan suatu sistem dengan bagiannya, sayangnya hal ini sangat tidak dipentingkan dikarenakan kebanyakan pengembangan sistem menggunakan pendekatan *code-first* yang di mana, sangat berpengaruh dalam komunikasi dan kolaborasi antar tim. Penelitian yang dilakukan oleh Oona Hämäläinen, menjelaskan bahwa pendekatan *design-first* merupakan salah satu opsi terbaru dalam pengembangan sistem berbasis web API dikarenakan terdapat keuntungan yang didapatkan seperti reusable atau mudah digunakan kembali di aplikasi lain, dokumentasi yang dihasilkan mudah dibaca serta konsisten, menambah pengalaman pengembang dalam membuat sistem, dan yang terpenting yaitu berkurangnya waktu pengembangan dikarenakan tidak harus menunggu tim lain siap [11]. Oleh karena itu, pengembangan sistem pada penelitian ini akan dilakukan dengan menerapkan pendekatan *design-first*.

Berdasarkan permasalahan yang dikemukakan sebelumnya, maka penelitian ini dilakukan dengan tujuan untuk mempermudah proses bisnis dengan menggunakan sistem informasi. Penelitian terkait sistem informasi yang telah dibahas sebelumnya memiliki latar belakang dan tujuan yang sama namun, penelitian yang dilakukan saat ini bertujuan untuk memperbarui sistem terdahulu. Di mana sebelumnya, terdapat fitur yang kurang dalam proses bisnis. Selain itu, sistem yang dibuat sekarang menggunakan metode pendekatan baru yang kurang diketahui banyak pengembang sistem sehingga, hasil yang didapatkan dari penelitian ini diharapkan dapat menjadi pertimbangan untuk menggunakan pendekatan *design-first* dalam pengembangan sistem sejenis.

2. Metode Penelitian

Penelitian ini akan mengembangkan sistem yang berbasis web API dengan melakukan penerapan pendekatan *design-first*.



Gambar 1: Ilustrasi Tahapan Pendekatan *Design-First*

Pendekatan design-first merupakan pendekatan baru yang digunakan untuk mengembangkan sistem berbasis web API. Pendekatan ini mengedepankan pada pembuatan dokumentasi (*contract*) sebagai hasil komunikasi antara pengembangan dengan stakeholder, serta menjadi panduan dalam pengembangan dan penggunaan API yang dibuat [12]. Terdapat tahapan dalam penggunaan pendekatan *design-first* yaitu:

1. Tahapan Perancangan (*Designing*)

Tahapan perancangan merupakan tahapan di mana dibuatnya dokumentasi sesuai dengan kebutuhan stakeholder. Kebutuhan dimulai dari apa yang diinginkan serta hasil yang didapatkan. Selain itu, dibutuhkan informasi berupa kemampuan serta batasan dari API yang akan dibuat sehingga menghasilkan dokumentasi sebagai bukti kesepakatan antara pengembang dengan *stakeholder*.

2. Tahapan Pengujian (*Testing*)

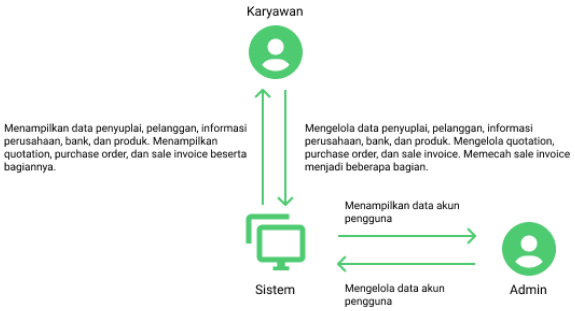
Setelah menghasilkan dokumentasi, maka selanjutnya dilakukan pengujian API palsu yang dihasilkan oleh dokumentasi dengan menggunakan data simulasi (*mock*) sehingga, tidak perlu menunggu API yang asli selesai dibuat. Hasil dari pengujian ini dapat menentukan apakah penggunaan API yang akan dibangun sesuai dengan kebutuhan *stakeholder*. Jika sudah sesuai, maka dapat lanjut ke tahap berikutnya dan jika belum, maka kembali ke tahap perancangan.

3. Tahapan Pembangunan (*Building*)

Jika API yang akan dibangun sudah sesuai dengan kebutuhan, maka tahap selanjutnya yaitu membangun API yang asli dengan mematuhi informasi yang didapatkan dari dokumentasi yang telah dibuat sebelumnya.

3. Hasil dan Pembahasan

Sistem Manajemen Penagihan Transaksi dibuat dengan tujuan untuk membantu proses bisnis perusahaan. Pengguna pada sistem ini berupa karyawan dan admin dimana karyawan dapat mengelola data seperti data bank, data penyuplai, data pelanggan, data produk, dan data tujuan pengiriman. Karyawan juga dapat mengelola dokumen transaksi seperti *purchase order* untuk membeli produk dari penyuplai, mengelola *quotation* untuk memberikan penawaran kepada pelanggan serta dapat mengelola *sale invoice* sebagai bukti transaksi penjualan kepada pelanggan. Apabila pelanggan ingin *sale invoice* dipisah ke beberapa bagian, maka karyawan dapat memecah *sale invoice* sesuai yang diinginkan pelanggan. Sedangkan admin dapat mengelola data akun pengguna serta dapat mengelola seluruh data yang karyawan dapat kelola. Gambaran umum sistem ini dapat dilihat pada Gambar 2.



Gambar 2: Gambaran Umum Sistem

Kebutuhan fungsional adalah merupakan kebutuhan-kebutuhan yang diperlukan untuk menyelesaikan suatu masalah oleh sistem yang akan dibuat. Kebutuhan fungsional yang dihasilkan untuk pengembangan sistem ini adalah:

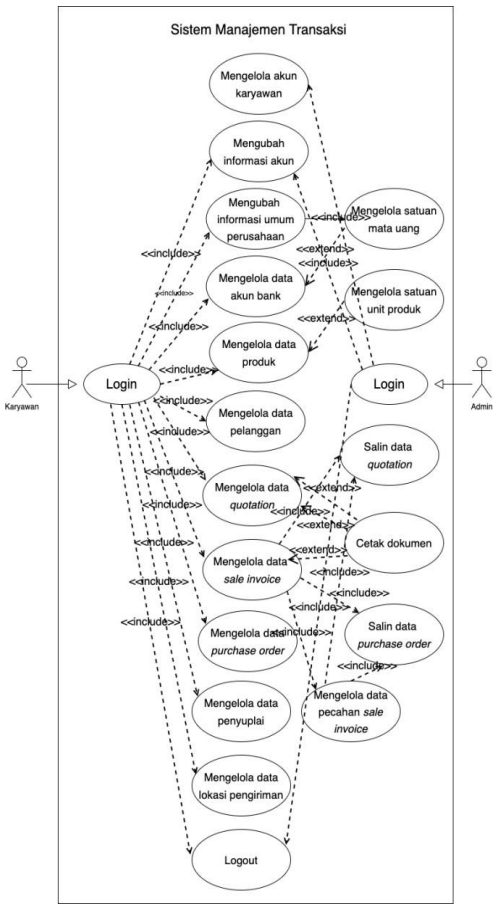
F001 : Karyawan dan admin dapat mengakses sistem dengan melakukan login terlebih dahulu.
F002 : Admin dapat mengelola data akun pengguna.
F003 : Pengguna dapat mengubah data akun pengguna tersebut.
F004 : Pengguna dapat mengubah informasi umum perusahaan.
F005 : Pengguna dapat mengelola data bank.
F006 : Pengguna dapat mengelola data satuan mata uang.
F007 : Pengguna dapat mengelola data pelanggan.
F008 : Pengguna dapat mengelola data penyuplai.
F009 : Pengguna dapat mengelola data produk.
F010 : Pengguna dapat mengelola data satuan unit produk.
F011 : Pengguna dapat mengelola data lokasi pengiriman.
F012 : Pengguna dapat mengelola data *quotation*.
F013 : Pengguna dapat mengelola data *purchase order*.
F014 : Pengguna dapat mengolah data *sale invoice*.
F015 : Pengguna dapat memecah *sale invoice* menjadi beberapa bagian.
F016 : Pengguna dapat mengelola bagian dari *sale invoice* yang telah dipecah.
F017 : Pengguna dapat menyalin data dari *purchase order* untuk digunakan sebagai data membuat *sale invoice* baru.
F018 : Pengguna dapat menyalin data dari *quotation* untuk digunakan sebagai data membuat *sale invoice* baru.
F019 : Pengguna dapat menghasilkan file berbentuk pdf dari *quotation* yang dipilih.
F020 : Pengguna dapat menghasilkan file berbentuk pdf dari *purchase order* yang dipilih.
F021 : Pengguna dapat menghasilkan file berbentuk pdf dari *sale invoice* yang dipilih.
F022 : Pengguna dapat menghasilkan file berbentuk pdf dari pecahan *sale invoice* yang dipilih.

Kebutuhan non fungsional adalah kebutuhan tambahan yang berisi batasan-batasan kapabilitas yang dapat dilakukan dari sistem yang akan dibuat. Kebutuhan non fungsional yang dihasilkan untuk pengembangan sistem ini antara lain:

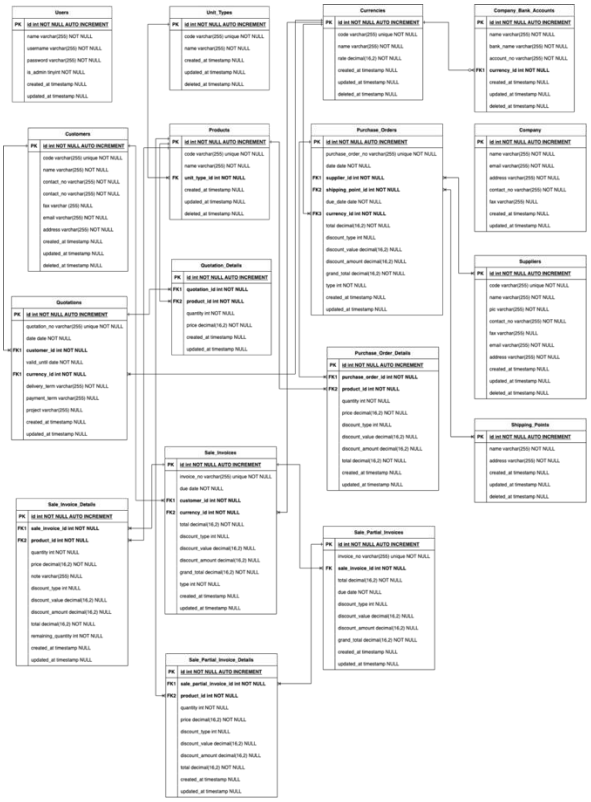
NF001 : Antarmuka pengguna yang mudah digunakan.
NF002 : Sistem dapat menyediakan Bahasa Indonesia.
NF003 : Sistem dapat diakses di lokasi manapun.

Use case diagram merupakan diagram yang menjelaskan suatu interaksi antara pengguna dengan sistem. Pengguna pada sistem ini yaitu karyawan dan admin di mana keduanya memiliki kemampuan untuk login sebelum mengakses sistem, mengubah informasi akun, mengubah informasi umum perusahaan, mengelola satuan mata uang, mengelola data akun bank, mengelola data produk beserta satuan unitnya, mengelola data pelanggan, mengelola data *quotation*, mengelola data *sale invoice*, mengelola data pecahan *sale invoice*, mengelola data penyuplai, mengelola data lokasi pengiriman, menyalin data dari *purchase order* atau *quotation* untuk digunakan di *sale invoice*, dan mencetak dokumen *quotation*, *purchase order*, *sale invoice* beserta pecahannya. Perbedaan dari pengguna karyawan dan admin yaitu pengguna admin dapat mengelola akun karyawan. Use case diagram yang dihasilkan untuk pengembangan sistem ini diperlihatkan pada Gambar 3.

Entity relationship diagram atau ERD merupakan diagram yang digunakan untuk menggambarkan suatu rancangan basis data pada suatu sistem. ERD berisi entitas atau objek yang berelasi serta atribut-atribut dari setiap entitas secara detil. ERD yang dihasilkan untuk pengembangan sistem ini diperlihatkan pada Gambar 4.



Gambar 3: Use Case Diagram



Gambar 4: ER Diagram

Pendekatan *design-first* dilakukan di awal bersamaan dengan tahap perancangan menuju tahap pengembangan pada metode *agile* karena pendekatan ini dilakukan sebelum melakukan pengembangan sistem serta menjadi panduan dalam proses pengembangannya. Kebutuhan yang didapatkan dari tahap perencanaan pada metode *agile* digunakan untuk merancang bentuk dokumentasi yang akan dibuat.

Pada penelitian ini, setiap skenario *use case* merupakan satu alur dalam menjalankan tahapan *design-first* yaitu tahap perancangan dan pengujian. Setelah semua skenario *use case* telah dirancang dan diuji, maka hasilnya berupa dokumentasi lengkap yang digunakan untuk tahap pembangunan, dimana tahap ini bersamaan dengan tahap pengembangan pada metode *agile*.

Berikut contoh satu alur penerapan pendekatan *design-first* dengan menggunakan skenario "admin dan karyawan dapat mengakses sistem dengan melakukan login terlebih dahulu".

1. Tahap Perancangan (*Designing*)

Pada tahap ini, dilakukan analisis terhadap skenario yang diberikan. Dari skenario tersebut dapat ditentukan salah satu kebutuhan sistem bahwa:

a. Pengguna dari sistem ini adalah admin dan karyawan.
b. Pengguna harus melakukan login terdahulu sebelum mengakses sistem.

Setelah itu, dari kebutuhan sistem kita perlu merancang secara teknis apa yang harus dilakukan untuk mencapai kebutuhan skenario ini. Pada skenario ini, dari segi teknis ditentukan bahwa:

a. Login sistem menggunakan *username* dan *password*.
b. Dalam API, login yang digunakan berbeda dengan sistem non-API. Sehingga pada skenario ini, agar sistem dapat diakses setelah login perlu adanya sebuah token. Token merupakan sebuah kumpulan karakter yang sudah dienkripsi yang berisi informasi khusus. Oleh karena itu, setelah pengguna memasukkan username dan password maka sebuah token akan dibuat agar dapat autentikasi ke dalam sistem.

Setelah ditentukan kebutuhan secara teknis, selanjutnya yaitu membuat sebuah rute API sesuai dengan ketetapan yang sudah ditentukan.

2. Tahap Pengujian (*Testing*)

Pada tahap ini, dilakukan pengujian dari API yang sudah dibuat oleh pihak *front-end* sebagai pengguna dari API tersebut. Walaupun API yang sebenarnya belum dikembangkan secara nyata, namun dengan adanya dokumentasi pengembang dapat menggunakannya karena kita dapat membuat data palsu atau mock tetapi merepresentasikan bentuk data yang akan muncul ketika API dikembangkan.

Setelah diuji, perlu dievaluasi apakah terdapat kesalahan atau kekurangan dari API tersebut. Salah satu contohnya berdasarkan skenario sebelumnya yaitu ketika setelah login dan API mengirim respon berupa token, pihak *front-end* membutuhkan data lain yaitu data lengkap penggunanya seperti nama. Karena pada tahap pengujian terdapat sebuah kekurangan, maka kembali ke tahap sebelumnya yaitu perancangan untuk mengubah API agar merespon kembali berupa token dan nama pengguna.

Apabila sesuai dengan kebutuhan, maka selanjutnya yaitu kembali ke tahap perancangan untuk merancang skenario selanjutnya. Hal ini dilakukan sampai selesai semua skenario dirancang dan diuji, ketika sudah maka tahap selanjutnya yaitu tahap pembangunan.

3. Tahap Pembangunan (*Building*)

Pada tahap ini, dilakukan pengembangan API yang sebenarnya berdasarkan dari dokumentasi yang telah dibuat serta diwajibkan untuk mematuhi informasi yang telah ditentukan agar tidak terjadi kesalahan pengembangan dan penggunaan karena, hal tersebut merusak salah satu tujuan dari *design-first* yaitu kolaborasi yang baik serta konsisten. Tahap ini berjalan pada tahap pengembangan pada metode *agile*.

Penerapan pendekatan *design-first* menghasilkan sebuah dokumentasi rancangan API agar dapat digunakan oleh pengembang yang dimana dokumentasi ini dibuat berdasarkan informasi yang didapatkan dari tahap perancangan. Pada penelitian ini, dokumentasi dibuat menggunakan format dari OpenAPI dikarenakan format ini sudah sering

digunakan oleh para pengembang sistem yang berbasis web API. Hasil dari dokumentasi yang dibuat untuk sistem ini dapat dilihat pada Gambar 5.

Pada gambar tersebut, hasil dari dokumentasi OpenAPI terdiri dari 3 bagian dalam penggunaanya yaitu:

1. Informasi umum dokumentasi

Pada bagian pertama merupakan bagian informasi seputar dokumentasi seperti nama sistem, versi dokumentasi, informasi umum, dan kontak pengembang.

2. Konfigurasi umum dokumentasi

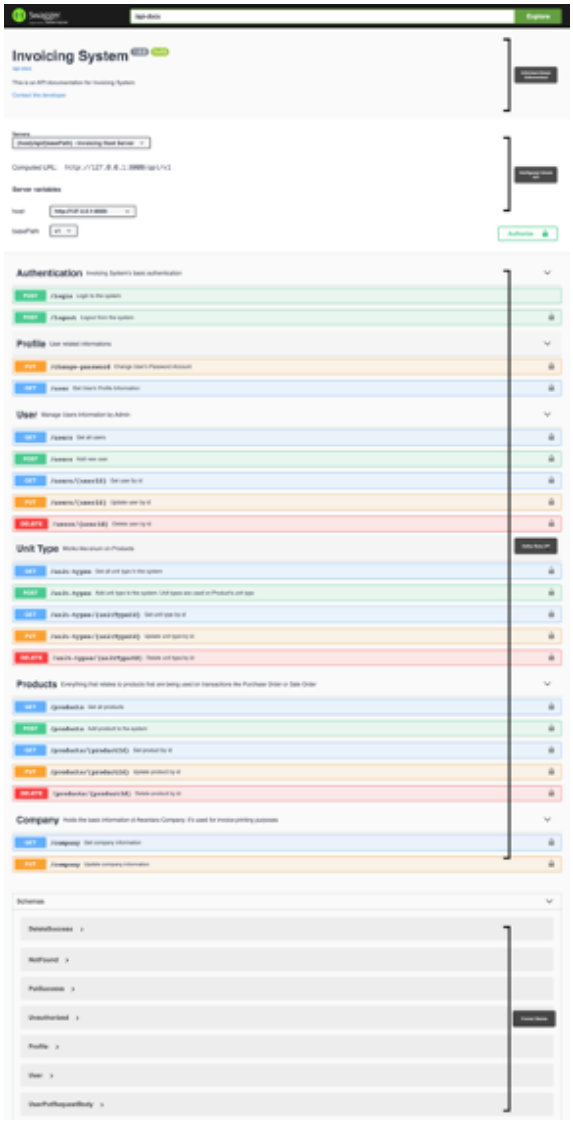
Pada bagian kedua berupa konfigurasi dari API dimana kita sebagai pembuat dokumentasi dapat memberikan opsi berupa letak alamat (*server*) API kita dapat diakses serta tempat untuk memasukkan token autentikasi untuk menggunakan API. Selain itu, OpenAPI memberikan kemudahan untuk mengubah parameter alamat (*server*) seperti memakai versi API terbaru atau yang lama sesuai dengan kebutuhan.

3. Daftar rute API

Bagian ketiga merupakan daftar seluruh rute API yang dapat digunakan. Pada bagian ini, setiap rute dapat dikategorikan sesuai dengan fungsi masing-masing dan dapat diberi label untuk memberikan informasi kegunaan rute tesebut.

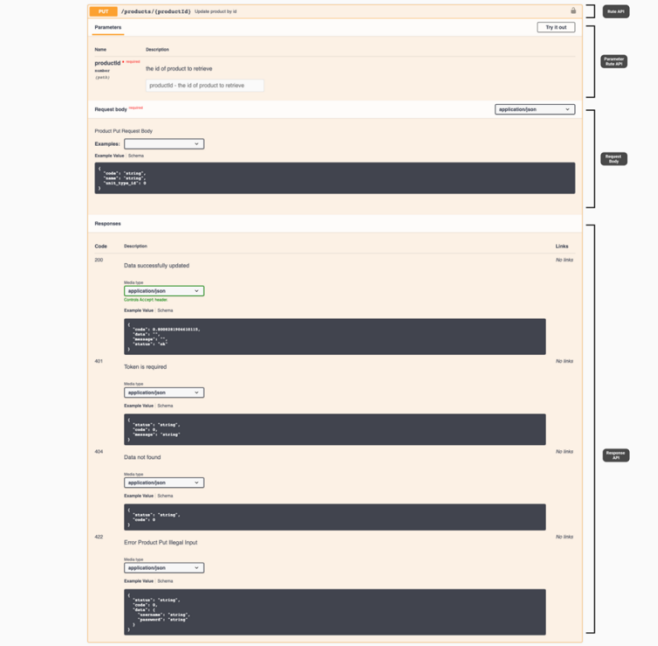
4. Format skema

Bagian terakhir berupa daftar skema yang fungsinya sebagai tempat untuk menyimpan skema-skema yang dapat digunakan oleh rute API dan bersifat *reusable*.



Gambar 5: Dokumentasi Sistem

Ketika pengguna membuka rute API nya, maka akan muncul detil informasi apa saja yang dibutuhkan dalam penggunaannya serta kemungkinan hasil yang diterima dari rute API tersebut. Contoh detail rute API dapat dilihat pada Gambar 6 yaitu rute API untuk mengubah data produk.



Gambar 6: Detail Rute API Ubah Produk

Pada Gambar 6, dapat dilihat isi dari rute API terdiri dari beberapa bagian dalam penggunaannya yaitu:

1. Rute API

Pada bagian pertama berupa rute lengkap API serta metode komunikasi yang digunakan (HTTP Method). HTTP Method merupakan komunikasi yang digunakan oleh *client* dalam melakukan permintaan kepada server. HTTP Method digunakan oleh sistem berbasis web API dengan menggunakan gaya arsitektur REST (Representational State Transfer). Pada contoh di atas, method yang digunakan berupa **PUT** dengan tujuan yaitu mengubah data secara keseluruhan. Selain **PUT**, terdapat beberapa HTTP Method lain seperti **GET** untuk mengambil data, **POST** untuk membuat data baru, dan **DELETE** untuk menghapus data [14].

2. Parameter Rute API

Pada bagian kedua berupa parameter atau data yang ditambahkan ke rute API. Pada contoh diatas, parameter yang harus dikirim yaitu *productId* (kode identifikasi dari produk) sehingga rute API yang akan dikirim berupa `{alamat server}/product/1` dimana 1 merupakan kode identifikasi produk.

3. Request Body

Pada bagian ketiga berupa *request body* (data yang dikirim) saat melakukan permintaan. Pada gambar di atas, format data yang dikirimkan dalam bentuk JSON (*JavaScript Object Notation*). JSON merupakan format teks untuk menyimpan dan mengirim data dengan bentuk dasarnya berupa objek yang berisi kumpulan nilai yang saling berpasangan [15]. Dari gambar di atas, format JSON yang dikirimkan berupa

```
{
  "code": "string",
  "name": "string", "unit_type_id": 0
}
```

Dimana **code** berupa kode unik dari produk, **name** berupa nama produk, dan **unit_type_id** berupa kode identifikasi dari satuan produk. Contoh dari data yang dikirimkan dalam melakukan permintaan API yaitu

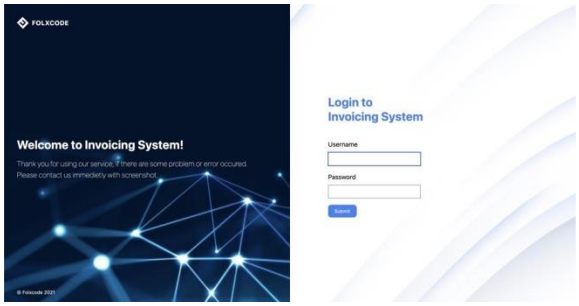
```
{
  "code": "PROD001",
  "name": "Produk Kecantikan Lipstik",
  "unit_type_id": 5
}
```

4. Response API

Pada bagian keempat berupa kemungkinan respons dari permintaan yang kita kirimkan. Pada gambar di atas, respons yang dikirim dari API berupa kode status dan data yang dikirim balik dalam format JSON. Kode status merupakan kode tiga dikit sebagai bentuk jawaban permintaan yang dikirim, kode tersebut memiliki tujuan masing-masing sebagai pesan apakah permintaan kita berhasil atau bermasalah [16]. Pada gambar di atas, kode dari kemungkinan respons dari permintaan yang kita kirimkan yaitu:

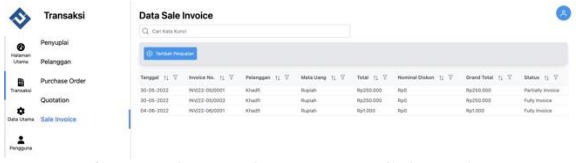
- Kode 200: Menandakan berhasil memproses permintaan
- Kode 401: Menandakan masalah karena tidak memiliki kewenangan dalam mengakses rute API.
- Kode 404: Menandakan masalah dimana data yang kita kirim dalam konteks ini berupa productId tidak ditemukan di server.
- Kode 422: Menandakan masalah karena data yang dikirim tidak sesuai, biasa dikarenakan bentuk JSON yang kita kirim tidak valid.

Beberapa tampilan aplikasi diperlihatkan pada Gambar 7 sampai dengan Gambar 9 berikut. Gambar 7 memperlihatkan tampilan halaman login. Halaman login merupakan halaman utama ketika pertama kali membuka sistem. Pada halaman ini, pengguna memasukkan data pengguna mereka berupa username dan password untuk mengakses sistemnya. Setelah berhasil melakukan login, maka tampil halaman utama seperti diperlihatkan pada Gambar 8. Pada halaman ini, pengguna disajikan informasi awal berupa profil dari pengguna itu sendiri dan pengguna dapat mengubah informasi data akun mereka. Selanjutnya dari halaman ini pengguna bisa melakukan berbagai fungsi melalui menu di sebelah kiri. Salah satunya adalah pengelolaan *sale invoice* melalui halaman sale invoice seperti diperlihatkan pada Gambar 9. Halaman ini menampikan data *sale invoice* beserta pecahan yang tersedia. Pada halaman ini, pengguna dapat mengelola data *sale invoice* perusahaan beserta pecahannya.



Gambar 7: Tampilan Halaman Login

Gambar 8: Tampilan Halaman Utama



Gambar 9: Tampilan Halaman *Sale Invoice*

Pengujian sistem merupakan sebuah proses untuk mengevaluasi apakah sebuah sistem memenuhi kebutuhan yang telah ditentukan atau tidak, tujuan dari pengujian sistem ini untuk mengetahui aplikasi yang dibangun dapat berjalan baik dan sesuai dengan kebutuhan fungsionalitasnya [17]. Metode pengujian yang digunakan pada aplikasi ini berupa metode *blackbox*, yaitu sebuah metode pengujian yang berfokus dalam menguji fungsionalitasnya di level teratas yaitu melalui tampilan untuk menguji input output apakah sudah sesuai dan tanpa ada masalah.

Teknik yang digunakan dalam pengujian menggunakan metode *blackbox* yaitu teknik *equivalence partitioning*, yaitu teknik pengujian berdasarkan data masukan berbeda sesuai (valid) atau tidak sesuai (invalid) pada fungsionalitas untuk menentukan hasil yang sesuai dengan kebutuhan. Hasil pengujian fungsionalitas dapat dilihat pada Tabel 1.

TABEL I

HASIL PENGUJIAN FUNGSIONALITAS

No	Kebutuhan Fungsionalitas	Kesimpulan Pengujian
1	Karyawan dan admin dapat mengakses sistem dengan melakukan login terlebih dahulu.	Valid, sesuai rancangan
2	Admin dapat mengelola data akun pengguna.	Valid, sesuai rancangan
3	Pengguna dapat mengubah data akun pengguna tersebut.	Valid, sesuai rancangan
4	Pengguna dapat mengubah informasi umum perusahaan.	Valid, sesuai rancangan
5	Pengguna dapat mengelola data bank.	Valid, sesuai rancangan
6	Pengguna dapat mengelola data satuan mata uang.	Valid, sesuai rancangan
7	Pengguna dapat mengelola data pelanggan.	Valid, sesuai rancangan
8	Pengguna dapat mengelola data penyuplai.	Valid, sesuai rancangan
9	Pengguna dapat mengelola data produk.	Valid, sesuai rancangan
10	Pengguna dapat mengelola data satuan unit produk.	Valid, sesuai rancangan
11	Pengguna dapat mengelola data lokasi pengiriman.	Valid, sesuai rancangan
12	Pengguna dapat mengelola data *quotation*.	Valid, sesuai rancangan
13	Pengguna dapat mengelola data *purchase order*.	Valid, sesuai rancangan
14	Pengguna dapat mengolah data *sale invoice*.	Valid, sesuai rancangan
15	Pengguna dapat memecah *sale invoice* menjadi beberapa bagian.	Valid, sesuai rancangan
16	Pengguna dapat mengelola bagian dari *sale invoice* yang telah dipecah.	Valid, sesuai rancangan
17	Pengguna dapat menyalin data dari *purchase order* untuk digunakan sebagai data membuat *sale invoice* baru.	Valid, sesuai rancangan
18	Pengguna dapat menyalin data dari *quotation* untuk digunakan sebagai data membuat *sale invoice* baru.	Valid, sesuai rancangan
19	Pengguna dapat menghasilkan file berbentuk pdf dari *quotation* yang dipilih.	Valid, sesuai rancangan
20	Pengguna dapat menghasilkan file berbentuk pdf dari *purchase order* yang dipilih.	Valid, sesuai rancangan
21	Pengguna dapat menghasilkan file berbentuk pdf dari *sale invoice* yang dipilih.	Valid, sesuai rancangan
22	Pengguna dapat menghasilkan file berbentuk pdf dari pecahan *sale invoice* yang dipilih.	Valid, sesuai rancangan

Berdasarkan hasil pengujian dapat dilihat bahwa seluruh kebutuhan fungsional yang telah dirancang sebelumnya telah berjalan dengan baik, sehingga disimpulkan bahwa sistem telah memenuhi kebutuhan.

4. Kesimpulan

Hasil penelitian berisi penerapan dari *design-first*

yang telah dilakukan dengan dibuatnya dokumentasi di awal pada tahap perancangan sebelum pengembangan sistemnya. Dokumentasi yang dibuat berisi gambaran desain dan fungsionalitas sistem yang dapat digunakan sebagai panduan dalam pengembangan sistem. Berdasarkan hasil implementasi dan pengujian, dapat disimpulkan bahwa sistem yang telah dibuat telah berjalan sesuai kebutuhan melalui pengujian sistem menggunakan metode *blackbox* dengan menggunakan teknik *equivalence partitions*. Setiap fungsionalitas sistem yang dibutuhkan telah diuji dan tidak ditemukannya masalah dalam pengujiannya.

Oleh karena itu, penerapan pendekatan *design-first* telah dilakukan dengan menghasilkan sebuah dokumentasi yang berisi gambaran serta panduan pengembangan sistem dengan tujuan mengembangkan sebuah sistem manajemen penagihan transaksi yang berfungsi untuk mengelola data pelanggan, penyuplai, produk, bank, mata uang, purchase order, *quotation*, dan *sale invoice*.

Saran untuk penelitian selanjutnya adalah agar dapat dikembangkan metode untuk menggabungkan pendekatan *code-first* dengan *design-first* sehingga lebih fleksible dalam pengembangan sistem. Selain itu, penulis dapat menggunakan teknik pengujian lain seperti *whitebox* untuk menerapkan *unit-testing* agar melakukan pengujian secara *code-level* agar pengujian lebih akurat dan lebih terjaga kejaminan fungsionalitas nya.

Daftar Pustaka

[1] Nugraha, Daniel, 2019, *Quotation dan Invoice, Perbedaan dan Perannya Dalam Proses Purchasing*, https://www.paper.id/blog/tips-dan-nasihat-umkm/perbedaan-quotation-dan-invoice/, diakses tgl 30 Oktober 2021.

[2] Aenun Najib., dan Fuaida Nablya., 2020, *Sistem Informasi Penagihan (Invoice) Berbasis Dekstop Menggunakan Metode Extreme Programing*, Skripsi, Program Sarjana Sistem Informasi, Univ. Peradaban, Jawa Tengah.

[3] Adi Wicaksono, Krisna, 2018, *Aplikasi Pembuatan Quotation Berbasis Web Dengan Menggunakan Angularjs 2 Dan Firebase Di CV Aditex Bangun Cipta*, Skripsi, Program Sarjana Teknik Informatika, Univ. Islam Indonesia, DIY Yogyakarta.

[4] Fadila, A. I. & Oktivasari, P. (2015). *Analisis dan Perancangan Proses Purchase Order pada PT. Cybertrend Intrabuana. Multinetics*, Vol. 1(1), 57-62.

[5] Dina Tauhida., Arinal Muana., 2019, *Rancang Bangun Aplikasi Purchase Order Pada Unit Purchasing PT. XYZ*, Skripsi, Program Sarjana Teknik Industri, Univ. Muria Kudus, Jawa Tengah.

[6] Raelida Turnip., dkk., 2021, *Analisis dan Perancangan Sistem Informasi Pembelian, Penjualan dan Persediaan pada UD. Parjuma Sonari,* Skripsi, Program Sarjana Teknik Informasi, Univ Mikroskil, Sumatera Utara

[7] Winarko, Triyugo, 2020, *Perancangan Sistem Informasi Pembelian, Penjualan Dan Persediaan Pada PT. Indo Global,* Bandar Lampung, Skripsi, Program Teknologi Informasi, Univ Mitra Indonesia, Lampung

[8] Inixindo, 2018, *Front-End vs Back-End Developer. Apa Bedanya?,* https://inixindojogja.co.id/front-end-vs-back-end- developer-apa-bedanya/, diakses tgl 30 Oktober 2021.

[9] Shilpi, 2019, *Divide or Join: The Frontend Backend Dilemma*, https://opensenselabs.com/blog/articles/frontend-backend, diakses tgl 30 Oktober 2021.

[10] Paguirigan, Ohmel, 2017, *How do front and back end developers work together on the same project at work?*, https://www.quora.com/How-do-front-and-back-end-developers- work-together-on-the-same-project-at-wor , diakses tgl 30 Oktober 2021.

[11] Hämäläinen, Oona, 2019, *API-First Design with Modern Tools*, *Tesis*, Program Sarjana Teknologi Informasi Bisnis, Jyväskylä University of Applied Sciences, Finlandia.

[12] Levin, Guy, 2020, *API Development with Design-first Approach*, https://blog.restcase.com/api-development-with-design-first-approach/, diakses tgl 16 November 2021.

[13] Huzefril, 2021, *HTTP Method*, https://www.huzefril.com/posts/microservices/rest-http-method/, diakses tgl 30 Januari 2022.

[14] Nayoan, Aldwin, 2020, *JSON: Pengertian, Fungsi dan Cara Menggunakannya*, https://www.niagahoster.co.id/blog/json-adalah/, diakses tgl 30 Januari 2022.

[15] M. Haekal, Mirza, 2020, *Panduan Lengkap Daftar HTTP Status Code dan Cara Mengatasinya [Terbaru]*, https://www.niagahoster.co.id/blog/http-status-code/, diakses tgl 30 Januari 2022.

[16] Muhammad Abid Jamil, Muhammad Arif, dkk, 2016, Software Testing Techniques: A Literature Review, *Literature*, Program Fakultas Informasi dan Komunikasi Teknologi, International Islamic University Malaysia.