BAB III

METODOLOGI PENELITIAN

3.1 Gambaran Umum Objek Penelitian

Pada penelitian ini, perusahaan yang dijadikan objek penelitian adalah PT XYZ. Berdasarkan hal tersebut maka penjelasan mengenai informasi perusahaan akan dibagi menjadi beberapa bagian:

3.1.1 Profil PT XYZ

PT XYZ merupakan salah satu perusahaan yang bergerak di industri perbankan. Bank ini didirikan pada 21 Februari 1957. Saat ini, PT XYZ merupakan salah satu bank swasta dengan jumlah nasabah terbanyak di Indonesia.

Terhitung pada tahun 2019, PT XYZ memiliki 9 anak perusahaan, 1 kantor pusat, 139 kantor cabang utama, 873 kantor cabang pembantu, 244 kantor kas, 2 kantor perwakilan luar negeri, 17.928 jumlah jaringan ATM, dan 24.789 karyawan tetap dan tidak tetap.

3.1.2 Visi Misi PT XYZ

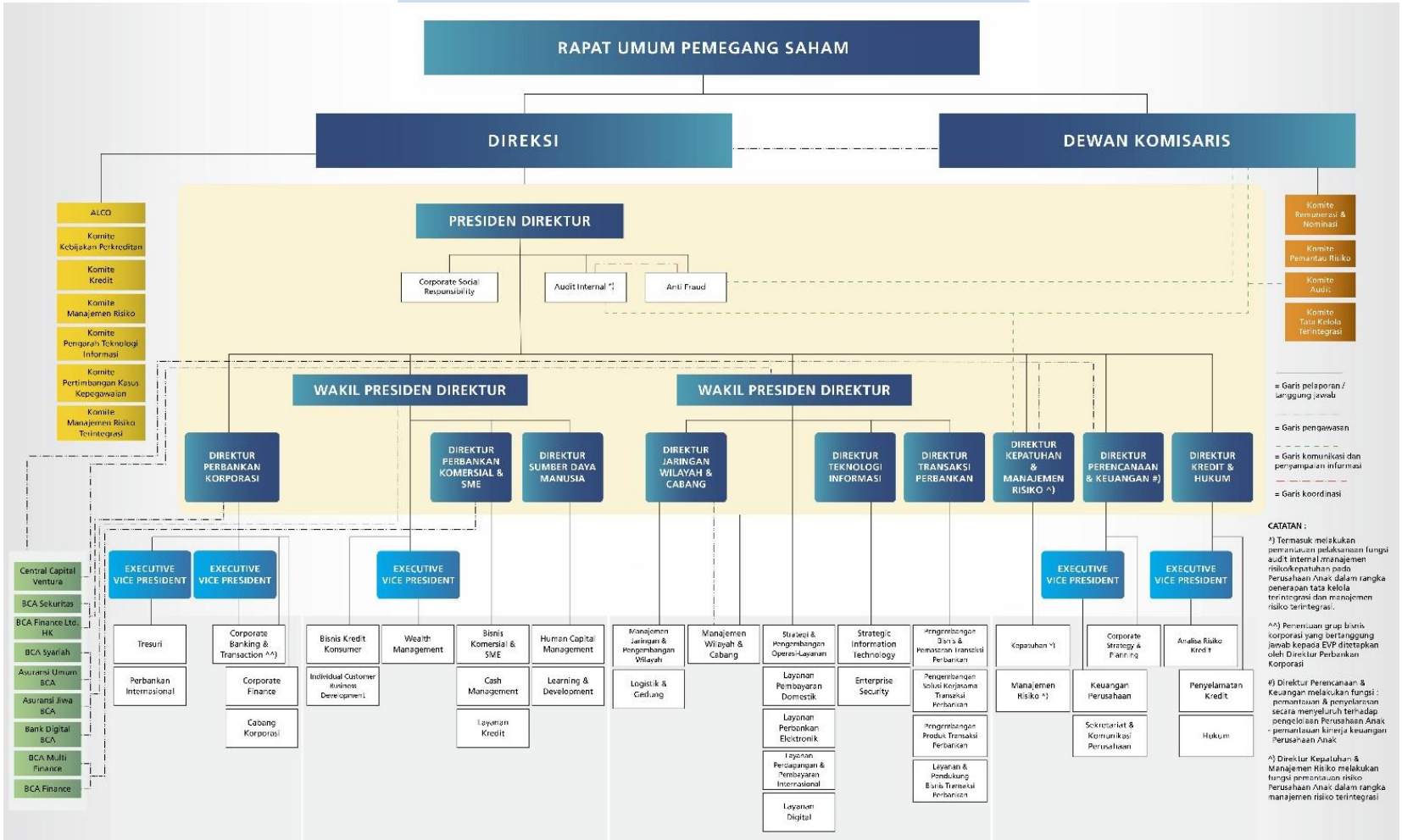
Visi PT XYZ yaitu "Bank pilihan utama andalan masyarakat yang berperan sebagai pilar penting perekonomian Indonesia."

Misi PT XYZ yaitu:

1. Membangun institusi yang unggul di bidang penyelesaian pembayaran dan solusi keuangan bagi nasabah bisnis dan perseorangan.
2. Memahami beragam kebutuhan nasabah dan memberikan layanan finansial yang tepat demi tercapainya kepuasan optimal bagi nasabah.
3. Meningkatkan nilai francais dan nilai stakeholder PT XYZ.

3.1.3 Struktur Organisasi PT XYZ

Berikut merupakan struktur organisasi yang dimiliki PT XYZ:



Gambar 3.1 Struktur Organisasi PT XYZ
Sumber: PT XYZ

Sebagai sebuah korporasi yang besar, PT XYZ memiliki banyak divisi seperti digambarkan pada gambar 3.1. Rapat Umum Pemegang Saham merupakan tingkat pengambilan keputusan tertinggi yang terdapat pada PT XYZ yang setelahnya diikuti oleh jajaran Direksi dan Dewan Komisaris.

Divisi Strategic Information Technology merupakan divisi yang bertugas untuk mengembangkan dan memberikan solusi kebutuhan teknologi serta memastikan seluruh layanan teknologi perbankan PT XYZ berjalan dengan baik. Divisi ini membawahi beberapa sub-divisi antara lain Group Application Management, Group IT Infrastructure & Operation, Group Data Management & IT Management Office, Group IT Architecture & Service Quality, dan Group Digital Innovation Solution.

3.2 Metode Penelitian

3.2.1 Metode Pengembangan Sistem

SDLC (*Systems Development Life Cycle*), merupakan proses membuat dan memodifikasi sistem serta model dan metode yang digunakan untuk mengembangkan sistem tersebut [38]. Konsep SDLC mendasari model pengembangan perangkat lunak lainnya. Model pengembangan perangkat lunak yang populer dan banyak digunakan antara lain model *waterfall* dan model *agile*.

Tabel 3.1 Perbandingan Model Waterfall dan Model Agile

Aspek pembanding	***Waterfall* Model**	***Agile* Model**
Kemudahan dalam modifikasi	Sulit dimodifikasi	Mudah dimodifikasi
Pendekatan pengembangan	Prediktif	Adaptif
Orientasi pengembangan	Berorientasi kepada proses	Berorientasi kepada *customer*
Skala cakupan proyek	Proyek besar	Proyek kecil atau sedang
Skala perencanaan	Jangka panjang	Jangka pendek
Gaya manajemen	Komando dan kendali	Kepemimpinan dan kolaborasi
Pembelajaran dalam proyek	Belajar berkelanjutan sembari berkembang	Pengembangan lebih penting daripada pembelajaran
Dokumentasi	Tinggi	Rendah

Aspek pembanding	*Waterfall* Model	*Agile* Model
Tipe organisasi	Besar	Kecil hingga menengah

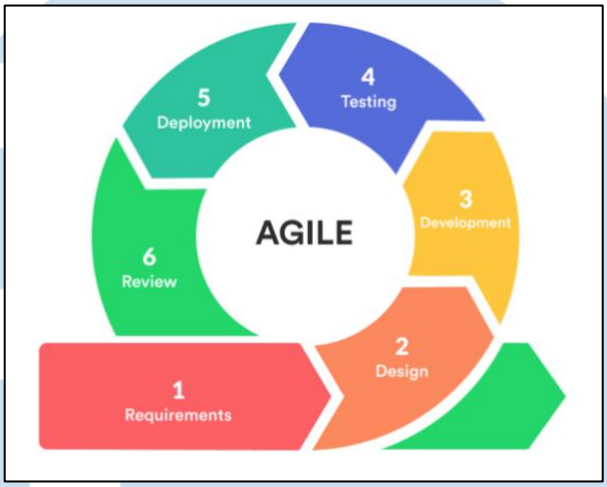
Sumber: [39]

Berdasarkan perbandingan yang telah dirinci pada tabel 3.1 maka penelitian ini akan menggunakan model SDLC *agile* karena pengembangan perangkat lunak dengan metode *agile* lebih adaptif terhadap perubahan. Penelitian ini memungkinkan untuk terjadinya perubahan dalam proses pengerjaannya, sehingga diperlukan model SDLC yang adaptif terhadap perubahan. Hasil akhir dari penelitian ini digunakan untuk menjawab kebutuhan anak perusahaan terkait penyediaan data nasabah PT XYZ. Oleh sebab itu, penelitian ini berorientasi kepada *customer* (anak perusahaan PT XYZ). Di lain sisi, tahap perencanaan dalam model *agile* tidak membutuhkan waktu yang panjang dibandingkan dengan model *waterfall* yang memerlukan waktu yang lebih panjang.

Model SDLC Agile memiliki beberapa tahapan seperti pada gambar 3.2. Berikut merupakan tahapan dari model SDLC *agile* yang digunakan untuk pengembangan perangkat lunak dalam penelitian ini:

1. *Requirement*, dalam tahap ini pengembang harus mengetahui seluruh informasi mengenai kebutuhan *software* yang diinginkan oleh pengguna.
2. *Design*, dalam tahap ini spesifikasi *requirement* yang sudah disepakati kemudian dipelajari dan berdasarkan *requirement* tersebut dibuatkan desain sistem yang akan dikembangkan.
3. *Development*, dalam tahap ini pengembang mulai mengembangkan kode menggunakan bahasa pemrograman dan *tools* yang sudah ditentukan sebelumnya.
4. *Testing*, dalam tahap ini dilakukan pengujian terhadap hasil dari proses pengembangan sebelumnya.
5. *Implementation*, pada tahap ini sistem yang dikembangkan kemudian diimplementasi sehingga dapat digunakan oleh pengguna.

6. *Review*, pada tahap ini sistem yang sudah di-implementasi kemudian dievaluasi kembali apakah keseluruhan sistem sudah berjalan dengan baik.



Gambar 3.2 Model SDLC Agile
Sumber: [40]

Model SDLC *agile* memiliki beberapa metode pendekatan yang berbeda-beda [40]. Beberapa metode tersebut antara lain *Test-Driven Development* (TDD), *Feature Driven Development* (FDD), *Extreme Programming* (XP), metode *Scrum*, *Dynamic System Development Model* (DSDM), dan metode *Crystal*. Metode-metode tersebut memiliki kelebihan dan kekurangannya masing-masing, sehingga penerapan metode dapat disesuaikan dengan budaya organisasi atau proyek yang dikembangkan. Seluruh metode *agile* tersebut menerapkan proses iterasi dan proses *incremental* dalam pengembangan perangkat lunak.

Penelitian ini menggunakan pendekatan metode *scrum* karena pengerjaan proyek dibagi menjadi bagian-bagian kecil sehingga memudahkan pengembang untuk berfokus kepada fitur-fitur yang akan dikembangkan. Metode ini berfokus untuk memberikan nilai terbaik dalam waktu yang singkat, sehingga penelitian ini dapat memberikan hasil yang maksimal dalam waktu yang telah ditentukan.

Metode *scrum* menerapkan iterasi pada waktu singkat yang disebut *sprint*. Pada metode ini, sistem dikembangkan secara bertahap dan menghasilkan bagian-bagian proses yang berbeda yang disebut *backlog*. Tabel 3.2 merupakan tabel yang menggambarkan *Sprint Backlog* yang berisi daftar tugas yang

dilakukan, penilaian kebutuhan waktu, penilaian tingkat kesulitan, dan penentuan *sprint* dari masing-masing tugas.

Tabel 3.2 Tabel *Sprint Backlog*

Backlog Task	***Level of Effort (hour)***	***Degree of Difficulties***	***Sprint Cycle***
Requirement gathering dan diskusi gambaran besar proyek	5	*Moderate*	1
Diskusi paket data yang akan diberikan	3	*Low*	1
Diskusi teknis *flow* API secara keseluruhan	2	*Moderate*	1
Diskusi teknis API Data	2	*High*	1
Perancangan basis data dan *flow stored procedure*	4	*High*	1
Pembuatan *activity diagram* untuk setiap *service* API data	2	*Low*	2
Pengembangan tabel basis data	4	*Moderate*	2
Pengembangan *stored procedure*	4	*High*	2
Pengembangan *job* ETL untuk penyediaan data	5	*High*	2
Pengembangan *service* Checking API	3	*Moderate*	3
Pengembangan *service* Get Fields API	3	*Moderate*	3
Pengembangan *service* Inquiry API	5	*High*	3
Pengembangan *service* Master API	5	*High*	3
Black box testing	2	*Low*	4
Load testing API	2	*Low*	4
Deployment job ETL untuk penyediaan data	5	*High*	5
Deployment Stored procedure	2	*Moderate*	5
Deployment service API data	5	*Moderate*	5

3.2.2 Perbandingan Arsitektur *Web Service* REST dan SOAP

Tabel 3.3 Perbandingan Arsitektur *Web Service* REST dan SOAP

Aspek pembanding	**REST**	**SOAP**
Penggunaan *bandwidth* yang dibutuhkan	Tidak membutuhkan banyak *bandwidth* karena sebagian besar *service* REST hanya mengembalikan data dengan format JSON	Membutuhkan lebih banyak *bandwidth* dibandingkan REST karena *service* SOAP hanya dapat mengembalikan data dengan format XML yang memiliki ukuran *file* lebih besar.

Aspek pembanding	REST	SOAP
Format data	Dapat mengembalikan beragam tipe format data seperti *plain text*, JSON, XML, HTML, dsb.	Hanya dapat mengembalikan data dengan format XML
Transfer protocol	Hanya dapat dilakukan melalui protokol HTTP	Menggunakan SOAP *envelop* kemudian data dikirim dengan protokol HTTP, atau SMTP, atau FTP.
Media akses	Menggunakan URL (*Uniform Resource Locator*) untuk mengakses fungsi nya.	Menggunakan antarmuka layanan untuk mengakses fungsi nya.

Tabel 3.3 merupakan tabel perbandingan arsitektur *web service* REST dan SOAP. Penelitian ini menggunakan arsitektur REST dibandingkan SOAP karena penelitian ini mempertimbangkan kecepatan *response time* dari *web service* yang dibuat, sehingga segala aspek yang ditentukan harus mendukung performa kecepatan *web service* tersebut. Pada penelitian [34] membuktikan bahwa gaya arsitektural REST dapat memberikan performa kecepatan *web service* yang lebih baik dibandingkan dengan SOAP.

3.2.3 Perbandingan Format Data JSON dan XML

Tabel 3.4 Tabel Perbandingan Format Data JSON dan XML

Aspek pembanding	JSON	XML
Tipe data	JSON mendukung tipe data *string*, *number*, *array*, dan *Boolean*.	Dalam format XML, seluruh data harus bertipe *string*.
Komentar	JSON tidak mendukung penulisan komentar.	XML mendukung penulisan komentar.
Pengambilan nilai	Pengambilan nilai lebih mudah.	Pengambilan nilai lebih sulit dibanding JSON.
Keamanan	JSON kurang aman.	XML lebih aman dibanding JSON.
Kemudahan untuk dibaca	JSON lebih mudah dibaca dibanding XML.	XML lebih sulit dibaca dibanding JSON.

Tabel 3.4 merupakan tabel perbandingan format data JSON dan XML. Penelitian ini menggunakan tipe format JSON sebagai format pertukaran data karena penelitian ini mempertimbangkan performa kecepatan *web service*, sehingga segala aspek yang ditentukan harus mendukung performa kecepatan *web service* tersebut. Pada penelitian [36] memberikan pembuktian bahwa format JSON dapat memberikan performa 30% hingga 40% lebih cepat dibandingkan XML.

3.2.4 Perbandingan Arsitektur *Software Monolithic* dan *Microservices*

Tabel 3.5 Tabel Perbandingan Arsitektur *Monolithic* dan *Microservices*

Aspek pembanding	Monolithic	Microservices
Struktur	Terdiri dari basis *code* tunggal dengan beberapa modul berdasarkan fungsi bisnis.	Terdiri dari layanan-layanan individu yang masing-masing layanannya memiliki satu fungsi.
Implementasi	Lebih mudah pada saat implementasi karena seluruh komponen sistem tergabung menjadi satu kesatuan.	Implementasi menjadi lebih kompleks karena masing-masing komponen sistem terpecah menjadi beberapa bagian.
Testing	Proses testing dapat dilakukan secara *end-to-end*.	Masing-masing komponen *microservice* dilakukan *testing* secara individu.
Reusability	Modul yang digunakan dalam sistem *monolithic* sulit untuk digunakan ulang dalam *software* yang berbeda.	Modul *microservice* dapat secara mudah digunakan kembali dalam pengembangan *software* yang berbeda.

Tabel 3.5 merupakan tabel perbandingan arsitektur *monolithic* dan *microservices*. Penelitian ini menggunakan konsep arsitektur sistem *microservices* karena dengan menggunakan arsitektur *microservices*, modul yang sudah dikembangkan dapat dengan mudah digunakan untuk keperluan pengembangan proyek yang berbeda. Selain itu, pada penelitian [37] memberikan pembuktian bahwa arsitektur *microservices* dapat memberikan performa yang lebih baik dibandingkan arsitektur *monolithic* ketika menangani *request* HTTP dalam jumlah yang besar.

3.3 Variabel penelitian

3.3.1 Variabel Independen

Variabel independen merupakan variabel yang tidak berhubungan dengan variabel lainnya. Variabel independen dalam penelitian ini antara lain:

1. Id anak perusahaan yang digunakan dalam proses autentikasi dan pemberian paket data.
2. Kategori paket data yang digunakan untuk menentukan *fields* data yang akan dilakukan *inquiry*.
3. Id nasabah yang akan diproses untuk pengambilan data.

3.3.2 Variabel Dependen

Variabel dependen merupakan variabel yang dipengaruhi oleh variabel independen. Variabel independen yang terdapat dalam penelitian ini yaitu *service* API yang dapat berfungsi secara optimal.

3.4 Teknik Pengumpulan Data

3.4.1 Studi Pustaka

Pengumpulan data dengan studi pustaka dilakukan dengan mencari jurnal penelitian yang menjadi acuan dalam melakukan penelitian terkait rancang bangun *web service* API menggunakan bahasa pemrograman Java dengan gaya arsitektural REST serta menggunakan *framework* Spring dan *query language* GraphQL.

3.4.2 Observasi

Pengumpulan data dalam penelitian ini juga dilakukan dengan observasi pada rapat yang diadakan setiap minggu. Rapat tersebut dihadiri oleh beberapa tim yaitu tim API Data, tim API Internal, tim API External, tim Pengembangan Operasi Layanan, dan tim Pengembang Aplikasi. Tujuan dilakukannya rapat mingguan adalah untuk saling menyampaikan perkembangan pengerjaan dan penetapan *sprint* dari masing-masing tim sehingga hasil akhir penelitian ini dapat memenuhi kebutuhan pertukaran data PT XYZ dengan anak perusahaan.

UNIVERSITAS MULTIMEDIA NUSANTARA