BAB II
TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori yang akan digunakan dalam menyusun laporan tugas akhir dan pembangunan sistem Marketing Tools Menggunakan API Twitter. Berikut ini adalah teori yang akan dibahas dalam landasan ini.

2.1.1 *Marketing*

Di dalam sebuah bisnis,marketing atau pemasaran merupakan kegiatan yang secara langsung menentukan jumlah penjualan. Menurut Kotler (2006) mengatakan bahwa “pemasaran merupakan proses sosial dan manajerial dimana individu atau kelompok bertujuan untuk memenuhi kebutuhan dan keinginannya melalui kreativitas, penawaran dan pertukaran nilai produk dengan yang lain” [18].

2.1.2 *Marketing Tools*

marketing tools adalah sistem, teknik, strategi, sumber daya, teknologi dan bahan yang digunakan oleh perusahaan atau ahli pemasaran untuk menciptakan dan mengimplementasikan kampanye pemasaran yang sukses mempromosikan produk dan layanan mereka [20].

2.1.3 Twitter

Twitter adalah layanan bagi teman, keluarga, dan teman sekerja untuk berkomunikasi dan tetap terhubung melalui pertukaran pesan yang cepat dan sering. Pengguna memposting Tweet, yang dapat berisi foto, video, tautan dan teks [19].

2.1.4 Twitter API

Twitter API adalah beberapa *endpoints* programatik yang bisa digunakan untuk memahami dan membangun percakapan di twitter. API ini mengizinkan pengguna untuk menemukan, terhubung, atau membuat beberapa sumber daya[17]. sumber daya tersebut meliputi:

1. Tuitan
2. Pengguna
3. Pesan Langsung

4. Trend
5. Media
6. Tempat

2.1.5 API (*Application Programming Interface*)

Application Programing Interface (API) akan mengekspos layanan atau sumber daya yang didukung oleh aplikasi perangkat lunak melalui sumber daya yang telah ditentukan seperti *method,* objek, atau URI (stlos, Faulring, Yang, & Myers, 2009). Dengan memanfaatkan sumber data, aplikasi lain akan bisa mengakses data atau layanan tanpa harus mengimplementasikan objek atau method. API adalah sesuatu yang penting bagi arsitektur perangkat lunak modern, karena memberikan abstraksi tingkat tinggi yang memfasilitasi atau memudahkan tugas pemrograman, Mendukung desain terdistribusi dan perangkat lunak *modular* dan kode yang *reusable (*Robillard, 2009*)* [24].

2.1.6 *Text Blast*

Text Blast adalah layanan pengiriman pesan yang mengizinkan pengguna untuk mengirimkan pesan secara bersamaan dengan jumlah yang banyak dengan memanfaatkan sistem pesan terautomasi. Cara ini adalah salah satu cara efektif organisasi bisnis berhubungan, meningkatkan hubungan, dan meningkatkan *brand awareness* konsumen.

2.1.7 Web Server

Web Server merupakan sebuah perangkat lunak dalam server yang berfungsi menerima permintaan (request) berupa halaman web melalui HTTP atau HTTPS dari klien yang dikenal dengan browser web dan mengirimkan kembali (response) hasilnya dalam bentuk halaman-halaman web yang umumnya berbentuk dokumen HTML [8].

2.1.8 *Unified Modeling Language*

UML dalam sebuah bahasa untuk menentukan visualisasi, konstruksi, dan mendokumentasikan artifacts dari sistem software, untuk memodelkan bisnis, dan sistem non-software lainnya. UML merupakan sistem arsitektur yang bekerja dalam OOAD dengan satu bahasa yang konsisten untuk menentukan, visualisasi, konstruksi dan mendokumentasikan artifact yang terdapat dalam sistem. Artifact

adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software. Artifact dapat berupa` model, deskripsi atau software [10].

2.1.8.1 Diagram UML

UML memiliki berbagai macam diagram untuk memvisualisasi rencana aplikasi perangkat lunak. Namun, dalam penelitian ini peneliti hanya akan menggunakan tiga macam diagram saja untuk memvisualisasikan model perangat lunak yang akan dibagung. Ketiga macam diagram itu adalah sebagai berikut:

1. *Use Case Diagram*

   Use case adalah model diagram yang menggambarkan interaksi antara aktor dengan sistem yang dirancang. Fungsi dari use case adalah menggambarkan urutan proses bisnis secara lebih jelas dan transparan agar mencegah terjadinya kesalahan pada sistem. Selain dari itu, fungsi lain dari *use case* membantu pengembang menentukan kebutuhan yang sesuai dengan perangkat lunak dan pengguna. Pembuatan *use case* diagram bertujuan untuk memberikan gambaran proses-proses yang terjadi pada sistem [11].

2. *Activity Diagram*

   Activity diagram adalah diagram yang digunakan untuk memodelkan runtutan proses yang terjadi pada sebuah sistem. Runtutan ini bisa berbentuk alur bisnis, runtutan menu dan lain sebagainya yang terdapat pada sistem.

3. *Sequence Diagram*

   Sequence Diagram atau diagram alur adalah model UML yang digunakan untuk memodelkan interaksi antar objek-objek dalam sebuah sistem secara detail. Tujuan dari diagram ini adalah untuk mengetahui urutan kejadian yang dapat menghasilkan output yang diinginkan.

2.1.9 Bahasa Pemrograman Go

Go atau biasa disebut golang adalah bahasa pemrograman yang dikembangkan di Google Oleh Robert Griesemer, Rob Pike dan Ken Thompson pada tahun 2007 dan bisa digunakan oleh khalayak umum pada 2009. Bahasa Go ditulis menggunakan bahasa C dan C++, oleh karena itu sintaksnya mirip dengan C [12].

2.1.10 Mongo

Mongodb merupakan basis data *open source* yang dibuat menggunakan bahasa C++. Mongodb diciptakan sejak Oktober 2007 oleh 10gen dan bisa digunakan oleh khalayak luas pada Februari 2009. Sampai pada penelitian ini ditulis mongodb sudah berada pada versi 5.0.7 dan tersedia beberapa sistem operasi seperti windows, ubuntu, Redhat, Debian dan berbagai macam sistem operasi lainya.

Basis dari mongodb adalah dokumen (Document Oriented Database) dan termasuk sistem basis data yang menganut paham NoSQL. NoSQL adalah singkatan dari Not Only SQL, ini berarti mongo adalah sebuah basis data yang tidak harus menggunakan perintah SQL untuk melakukan proses manipulasi data [7].

Mongodb tidak menggunakan istilah tabel, baris dan kolom. Format penulisan pada mongodb menggunakan BSON(*Binary JSON*) format ini persis dengan format JSON namun perbedaanya BSON mendukung tipe data yang sangat luas misalnya tipe data untuk *date,* menyimpan *script* javascript, regex (*regular expression*) dan lain sebagainya.

adapun komponen-komponen dasar dari mongodb adalah sebagai berikut:

1. Database

   Adalah sebuah penampung untuk *collection.* Setiap database memiliki beberapa file pada sistem. Sebuah server mongodb biasanya memiliki beberapa database.

2. *Collection*

   Collection merupakan tempat dimana *document* simpan. *Collection* jika bandingkan dengan basis data berbasis relasi maka *collection* ini sama dengan tabel.

3. *Documents*

   Adalah satuan unit terkecil yang ditampung didalam *collection*. *Documents* terdiri dari data-data yang tersimpan.

2.1.11 HTML

HTML adalah singkatan dari *HyperText Markup Language* dan html merupakan bahasa scripting[9]. HTML dibangun pada tahun 1991 oleh ilmuwan bernama Tim Berners-Lee, tujuan dibangunya HTML adalah untuk memudahkan ilmuwan berbagi dokumen antar ilmuwan lainya. Pada tahun 1992 HTML mulai bisa digunakan oleh khalayak umum hal ini tentunya didukung dengan peran dari *World Wide Web* keduanya saling bersinergi demi tersebarnya informasi yang dapat diakses oleh banyak orang.

adapun bagian-bagian terpenting dalam HTML, yaitu :

1. Tag

   Tag adalah tanda awalan dan akhiran dari sebuah elemen HTML. Tag berlawanan dengan kurung siku (<x>) dalam kurung siku diisi oleh nama tag, sedangkan untuk tanda akhir elemen menggunakan kurung siku namun didalamnya ada simbol *black slash* (/) diikuti dengan nama tag, jadi simbol penutupnya adalah </x>.
2. Element

   Element adalah sebuah komponen yang menyusun keseluruhan dokumen HTML, element terdiri dari simbol pembuka tag, konten tag dan penutup tag misalkan <title> judul </title>.
3. Attribute

   Attribute HTML merupakan informasi tambahan pada suatu tag atau elemen HTML. Tag Bisa memiliki beberapa atribut sekaligus, namun ada beberapa tag yang tidak mempunyai attribute.

2.1.12 CSS

CSS adalah singkatan dari *Cascading Style Sheets.* CSS adalah sekumpulan perintah yang mengendalikan penampilan dari halaman web [9]. Dengan menggunakan CSS pengembang dapat melakukan *layouting,* pengaturan ukuran, pengaturan posisi dari text dan gambar dan lain sebagainya.

CSS dikembangkan oleh *World Wide Consortium* atau W3C pada tahun 1996. dengan alasan kekurangan HTML yang tidak memiliki tag yang berfungsi sebagai alat untuk format halaman juga untuk mengurangi tag-tag baru yang dibuat oleh Netscape dan Internet Explorer dikarenakan kedua browser ini saling bersaing untuk mengembangkan tag sendiri dalam pengaturan tampilan web.

2.1.13 Javascript

Javascript adalah bahasa pemrograman interpreted, yang dibangun diatas ECMAScript standar. Javascript pada awalnya dikembangkan oleh Brendan Eich dari *netscape* dibawah nama mocha, awalnya bahasa pemrograman ini bernama livescript hingga akhirnya berganti nama menjadi javascript, Perubahan nama ini berasal dari keputusan *netscape* versi 2.0b3 pada tahun 1995 yang menambahkan dukungan terhadap bahasa java. Javascript umumnya digunakan sebagai bahasa

pemograman *client side* yang diimplementasikan sebagai bagian dari web browser agar pengembang dapat meningkatkan kedinamisan antarmuka pada halaman web [9].

2.1.14 *Clean Architecture*

Clean architecture adalah sebuah konsep arsitektur perangkat lunak yang diciptakan oleh Robert C. Martin yang bisa digunakan pada perangkat lunak berbasis web, aplikasi mobile, dan lain sebagainya. Arsitektur ini merupakan gabungan dari beberapa arsitektur atau *desain pattern* demi memberikan solusi arsitektur perangkat lunak yang *readable, maintainable, testable,* dan *reusable* [4]*.* adapun beberapa arsitektur yang digunakan dalam arsitektur ini adalah sebagai berikut:

1. Hexagonal Architecture
2. Onion Architecture
3. Screaming Architecture
4. DCI (*Data, Context, Interaction*)

meskipun semua arsitektur memiliki perbedaan dalam praktik, istilah dan detailnya tetapi semua arsitektur ini memiliki tujuan yang sama yaitu *separation of concern* (pemisahan tanggung jawab) artinya kode akan dipisahkan sesuai dengan tugasnya masing-masing [6]. Semua arsitektur yang ada dalam *clean architecture* mencapai pemisahan tanggung jawab dengan cara yang sama yaitu dipisahkan kedalam beberapa *layers* kode. *clean architecture* akan menghasilkan perangkat lunak dengan karakter sebagai berikut:

1. *independent of framework*

   Dengan menggunakan clean architecture perangkat lunak tidak akan terpengaruh dan bergantung pada framework hal ini menjadikan framework hanya sebatas infrastruktur pada perangkat lunak bukan inti dari perangkat lunak.
2. *Testable*

   Alur atau logika bisnis dapat diuji tanpa harus ada antarmuka, basis data, atau elemen luar.
3. *Independent Of UI*

   Antarmuka dapat diubah-ubah tanpa berpengaruh terhadap keseluruhan sistem.
4. *Independent Of Database*

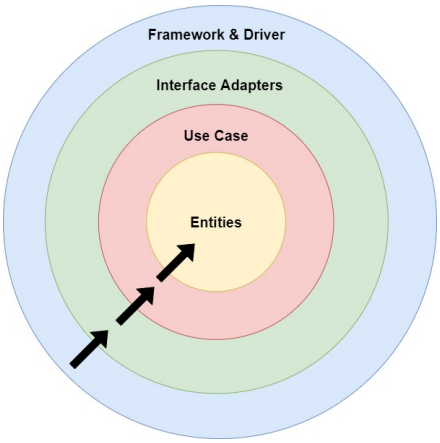
Pada arsitektur ini memungkinkan perubahan basis data tanpa berpengaruh terhadap keseluruhan sistem.

5. *Independent Of Any External Agency*

   Secara singkat ini berarti aturan/alur/logika bisnis tidak harus mengetahui kode pihak ketiga.

2.1.14.1 Bagian-bagian Pada *Clean Architecture*

Clean architecture akan memisahkan kode ke dalam beberapa *layers* sesuai pada Gambar 2.2



Gambar 2.1 *Gambaran Clean Architecture*

adapun penjelasan dari *layers* arsitektur pada gambar diatas adalah sebagai berikut:

1. *Entities*

   Tugas dari entities adalah untuk mendefinisikan aturan bisnis yang akan dipakai dalam perangkat lunak, pendefinisian aturan bisnis bisa dilakukan dengan berbagai cara aturan bisnis bisa didefinisikan sebagai objek dengan method atau struktur data. Kode pada layer ini akan jarang berubah hal ini terjadi karena kode pada *layer* ini merupakan inti dari perangkat lunak walaupun ada perubahan kode pada *layer ini* maka perubahan akan terjadi pada layer terluarnya juga.

2. *Use Case*

   Use case atau sebagian lain menyebutnya dengan istilah *service* merupakan *layer* kode yang berfungsi untuk implementasi aturan bisnis yang sudah didefinisikan pada *layer entities.*

3. *Interface Adapters*

Fungsi utama dari *layer* ini adalah untuk menerima masukan dan menampilkan pengeluaran, pada layer ini juga terjadi beberapa proses misalnya konversi data agar dapat digunakan oleh layer berikutnya.

4. *Framework and Driver*

   Layer ini adalah layer terluar dari arsitektur ini, adapun kegunaan dari layer ini adalah untuk menyimpan kode infrastruktur seperti framework, database driver, library pihak ketiga dan lain sebagainya.

2.1.14.2 Aturan Dependensi

Seperti yang dijelaskan pada gambar 2.2 arah dependensi tiap-tiap layer akan mengarah ke dalam sehingga arah dependensi antar layer akan dimulai dari *framework & driver* ke *interface adapters,* lalu dari *interface adapter* ke *use case,* dan terakhir adalah dari *use case* ke *entities* [6].

2.1.15 Metode Pengujian

Pengujian perangkat lunak adalah proses untuk mencari kesalahan pada setiap *item* perangkat lunak, mencatat hasilnya, mengevaluasi setiap aspek pada setiap komponen dan mengevaluasi fasilitas-fasilitas dari perangkat lunak yang dikembangan[1]. Glen Myers menyatakan beberapa aturan yang dapat digunakan sebagai penjelasan tentang pengujian perangkat lunak yaitu:

1. Pengujian merupakan sebuah proses eksekusi program dengan tujuan utama untuk mencari kesalahan.
2. Sebuah kasus pengujian dikatakan baik jika memiliki kemungkinan penemuan kesalahan yang tinggi.
3. Pengujian yang berhasil adalah pengujian yang menemukan kesalahan pada program.

Pada penelitian ini terdapat dua macam metode pengujian yang akan digunakan yaitu *white box* dan *black box.* berikut ini masing-masing penjelasan dari kedua jenis pengujian:

2.1.15.1 White Box Testing

Metode pengujian *white box* adalah salah satu cara pengujian perangkat lunak dengan menganalisis kode program atau bisa juga diartikan metode pengujian

dengan cara menguji struktur internal sistem[1]. dalam metode ini ada beberapa teknik pengujian diantaranya:

1. *Statement Coverage*

   Statement Coverage merupakan teknik pengujian *white box* yang melibatkan eksekusi pernyataan secara menyeluruh setidaknya satu kali eksekusi dalam *source code*-nya.

2. *Branch Coverage*

   Teknik *branch coverage* adalah salah satu teknik dalam pengujian *white box* yang digunakan untuk memastikan bahwa setiap percabangan dalam sistem harus dieksekusi.

3. *Conditional Coverage*

   Conditional Coverage adalah teknik pengujian yang bertujuan untuk memeriksa hasil individu pada setiap kondisi logis tertentu.

4. *Multiple Condition Coverage*

   Multiple Condition Coverage adalah sebuah teknik yang dilakukan untuk menguji semua kombinasi yang mungkin terjadi dari berbagai kondisi yang ada setidaknya satu kali.

5. *Basis Path Testing*

   Teknik ini digunakan untuk mengukur kompleksitas kode program dan mendefinisikan alur yang dieksekusi.

6. *loop Testing*

   loop testing adalah teknik pengujian untuk memeriksa hasil akhir pengulangan atau *loop* serta memastikan apakah pengulangan sudah berjalan dengan semestinya.

2.1.15.2 Black Box Testing

Metode pengujian *black box* adalah pengujian yang dilakukan untuk memastikan fungsionalitas bekerja dengan baik[1]. Pada metode ini pengujian dilakukan sepenuhnya dengan menilai kebutuhan dan spesifikasi dari perangkat lunak tanpa harus menilai dari sisi internal pada sebuah sistem, pada pengujian *black box* penguji berusaha untuk menemukan beberapa kesalahan yang dikategorikan sebagai berikut:

1. Fungsi yang salah atau tidak berfungsi
2. Kesalahan pada antar muka

3. Kesalahan kinerja
4. Kesalahan struktur data
5. Inisialisasi dan kesalahan terminasi

pada *black box* testing ada beberapa metode yang dapat digunakan yaitu:

1. *Graph Based Testing*

   Teknik desain *test cases* yang menggambarkan logika dari kondisi terhadap aksi yang dilakukan atau bisa di artikan sebuah pengujian perangkat lunak yang menggunakan grafik yang menggambarkan sebab dari aksi yang dilakukan pada sistem.
2. *Equivalence Partitioning*

   Adalah metode pengujian *black box* yang membagi domain masukan dari suatu program ke dalam kelas-kelas data, dimana *test case* dapat diturunkan.
3. *Boundary Value Analysis*

   Teknik *boundary value analysis* merupakan teknik komplemen dari teknik *equivalence partitioning,* dimana teknik ini berguna untuk melakukan pengujian terhadap nilai sekitar dari pusat domain masukan.

2.2 Perangkat Lunak Pendukung

Untuk membangun sistem ini peneliti membutuhkan beberapa perangkat lunak pendukung, diantaranya:

2.2.1. Visual Studio Code

Visual studio code adalah *code editor* yang ringan namun kuat yang dapat berjalan pada sistem operasi Windows, macOS dan linux. *editor* ini datang dengan dukungan untuk bahasa javascript, typescript, dan Node.js juga memiliki ekosistem yang kaya akan *extension* untuk bahasa pemrograman lain (seperti C, C++, C#, Java, Python, PHP, Go) dan runtime (seperti .NET dan Unity) [13].

2.2.2. MongoDB Compass

Mongodb Compass adalah perangkat lunak GUI untuk melakukan proses querying, aggregating dan menganalisa data mongoDB dalam lingkungan visual [14]. Selain dari itu mongodb compass juga membantu peneliti untuk melakukan monitoring dan pengujian instalasi pada server nantinya.

2.2.3. PuTTY

PuTTY adalah sebuah client SSH dan telnet, dibangun oleh Simon Tatham untuk sistem operasi Windows. Putty adalah perangkat lunak *open source* yang tersedia juga dalam bentuk *source code* dan dikembangkan serta didukung oleh kelompok relawan [15].

2.2.4. Postman

Postman adalah sebuah *platform* API untuk pembangunan dan menggunakan API. Postman menyederhanakan setiap langkah lingkar kehidupan API dan mempermudah kolaborasi sehingga dapat membuat API yang lebih baik[15].

2.2.5. Google Chrome

Chrome adalah program *open source* untuk mengakses internet dan mengeksekusi aplikasi berbasi web. Google chrome bersumber dari proyek *chromium*. Google meluncurkan google chrome pada tahun 2008 dan melakukan beberapa update dalam satu tahun. Google chrome tersedia untuk sistem operasi Windows, macOs, Linux, Android, dan IOS [16].