

**BUKU TUTORIAL PEMBUATAN DAN PENGGUNAAN APLIKASI  
DIGITAL INVOICE DAN SETTLEMENT PADA APLIKASI  
TELKOM PARTNER NETWORK  
(STUDI KASUS: PT. TELEKOMUNIKASI INDONESIA)**

**“IMPLEMENTASI PENAMBAHAN DIGITAL INVOICE DAN  
SETTLEMENT PADA APLIKASI TELKOM PARTNER NETWORK  
STUDI KASUS: PT.TELEKOMUNIKASI INDONESIA”**

Buku ini dibuat untuk memenuhi persyaratan kelulusan  
matakuliah Program Internship I



**Dibuat Oleh,**  
**1.16.4.039      Fransiscus Ivan Martongam Sinaga**

**PROGRAM DIPLOMA IV TEKNIK INFORMATIKA  
POLITEKNIK POS INDONESIA  
BANDUNG  
2020**

Penulisan laporan Internship I ini dan penulis menyadari bahwa penyusunan laporan ini masih belum sempurna. Ini mengingat keterbatasan pengetahuan, pengalaman serta kemampuan penulis. Penulis megharapkan kritik dan saran yang sifatnya membangun dari pembaca. Untuk itu penulis mengucapkan terima kasih kepada:

1. Allah S.W.T yang telah melimpahkan karunia-Nya sehingga kami dapat menyelesaikan tugas ini;
2. Kedua orang tua dan keluarga saya yang telah memberi motivasi kepada saya;
3. Hezekieli Gulo. selaku pembimbing eksternal yang telah membantu selama kegiatan intership.
4. Ibu Rd Nuraini Siti Fatonah,SS,M.Hum selaku dosen pembimbing internship 1.
5. Ibu Nisa Hanum Harani, S.T., M.T.I selaku Koordinator Internship I.
6. Bapak M. Yusril Helmi Setyawan, S.Kom., M.Kom. selaku Ketua Prodi D4 Teknik Informatika.
7. Pihak Telekomunikasi Indonesia yang telah banyak membantu dalam memberikan solusi dan saran terhadap penelitian yang saya buat;  
Akhir kata, penulis berharap Tuhan Yang Maha Esa berkenan membala segala kebaikan semua pihak yang telah membantu. Semoga buku ini membawa manfaat bagi pengembangan ilmu.

Bandung, 17 Januari 2020

Penulis

## **DAFTAR ISI**

### **DAFTAR ISI3**

#### **BAB I DASAR1**

##### **1.1 PENGENALAN WEB PROGRAMMING1**

1.1.1 Pengantar Web Programming1

1.1.2 HTML2

1.1.3 BASIS DATA11

1.1.4 SEJARAH16

1.1.5 BAHASA PEMROGRAMAN22

1.1.6 Macam Macam Framework60

**BAB II PENGENALAN DOCKER, REDISH, MICROSERVICES DAN REST..... 36**

##### **2.1 PENGENALAN ALGORITMA DATA AKSES LAYER74**

2.1.1 Pengertian DATA AKSES LAYER**Error! Bookmark not defined.**

2.1.2 Pengantar Algoritma DATA AKSES LAYER**Error! Bookmark not defined.**

**BAB III PENJELASAN TOOLS DAN BAHASA PEMROGRAMAN103**

##### **3.1 PENJELASAN TOOLS DAN BAHASA PEMOGRAMAN YANG DIGUNAKAN103**

3.1.1 Tools Yang Digunakan**Error! Bookmark not defined.**

3.1.2 Framework Express107

3.1.2 Library Nodejs107

## BAB IV INSTALASI TOOLS YANG DIGUNAKAN125

### 4.1 INSTALASI TOOLS YANG DIGUNAKAN125

#### 4.1.1 Tools Yang Digunakan**Error! Bookmark not defined.**

## BAB V PERANCANGAN DAN PEMBUATAN APLIKASI151

### 5.1 PERANCAGAN APLIKASI**Error! Bookmark not defined.**

#### 5.1.1 Analisis Yang Sedang Berjalan**Error! Bookmark not defined.**

#### 5.1.2 Analisis Registrasi Yang Akan Dibangun**Error! Bookmark not defined.**

#### 5.1.3 Flow Chart Prosedure Kunjungan Yang Akan Dibangun**Error! Bookmark not defined.**

#### 5.1.4 Use Case Diagram**Error! Bookmark not defined.**

#### 5.1.5 Class Diagram**Error! Bookmark not defined.**

#### 5.1.6 Component Diagram**Error! Bookmark not defined.**

#### 5.1.7 Component Diagram**Error! Bookmark not defined.**

### 5.2 PERANCAGAN DATABASE**Error! Bookmark not defined.**

#### 5.2.1 *Conceptual Data Model (CDM)***Error! Bookmark not defined.**

#### 5.2.2 *Physical Data Model (PDM)***Error! Bookmark not defined.**

### 5.3 TUTORIAL PEMBUATAN APLIKASI**Error! Bookmark not defined.**

#### 5.4.1 Mempersiapkan Tools**Error! Bookmark not defined.**

#### 5.4.2 Membuat database**Error! Bookmark not defined.**

#### 5.4.3 Konfigurasi Codeigniter**Error! Bookmark not defined.**

#### 5.4.4 Konfigurasi Template CSS Bootstrap**Error! Bookmark not defined.**

5.4.5 Membangun Program Bagian Penjungung*Error! Bookmark not defined.*

BAB VI PENERAPAN *BOYER MOORE**Error! Bookmark not defined.*

6.1 PENERAPAN ALGORITMA PADA APLIKASI*Error! Bookmark not defined.*

6.2.2 Alasan Menggunakan Algoritma *Boyer moore**Error! Bookmark not defined.*

Algortima Pencocokan Data WBP.*Error! Bookmark not defined.*

6.1.2 Algoritma Pencarian Data (WBP, REGISTRASAI,  
TANGGAL)*Error! Bookmark not defined.*

6.1.3 Hasil Pengujian*Error! Bookmark not defined.*

6.2 Kelebihan dan kekurangan Algoritma *Boyer moore**Error! Bookmark not defined.*

6.2.1 Kelebihan*Error! Bookmark not defined.*

6.2.2 Kekurangan*Error! Bookmark not defined.*

BAB VII HASIL ANTAR MUKA APLIKASI*Error! Bookmark not defined.*

7.1 Halaman Antarmuka Admin*Error! Bookmark not defined.*

7.4.1 Halaman Login*Error! Bookmark not defined.*

7.4.2 Utama Admin*Error! Bookmark not defined.*

7.4.3 Kelola Pengunjung*Error! Bookmark not defined.*

7.4.4 Halaman Kelola WBPE*Error! Bookmark not defined.*

7.4.5 Kelola Jadwal*Error! Bookmark not defined.*

7.4.6 Kelola Daftar Verifikasi*Error! Bookmark not defined.*

7.4.7 Cari NamaWBPE*Error! Bookmark not defined.*

7.4.8 Cari No Registrasi**Error! Bookmark not defined.**

7.4.2 Cetak**Error! Bookmark not defined.**

7.4.2 Pemberitahuan Email**Error! Bookmark not defined.**

6.4.2 Pesan**Error! Bookmark not defined.**

7.5 Halaman Antarmuka Pengunjung**Error! Bookmark not defined.**

7.5.1 Login**Error! Bookmark not defined.**

7.5.2 Pendaftaran**Error! Bookmark not defined.**

7.5.3 Prosedur Pendaftaran**Error! Bookmark not defined.**

7.5.4 Daftar Akun**Error! Bookmark not defined.**

7.5.5 Dashboard**Error! Bookmark not defined.**

7.5.5 Formulir Kunjungan**Error! Bookmark not defined.**

7.5.6 Cek Pesan**Error! Bookmark not defined.**

## **DAFTAR GAMBAR**



## **DAFTAR TABEL**

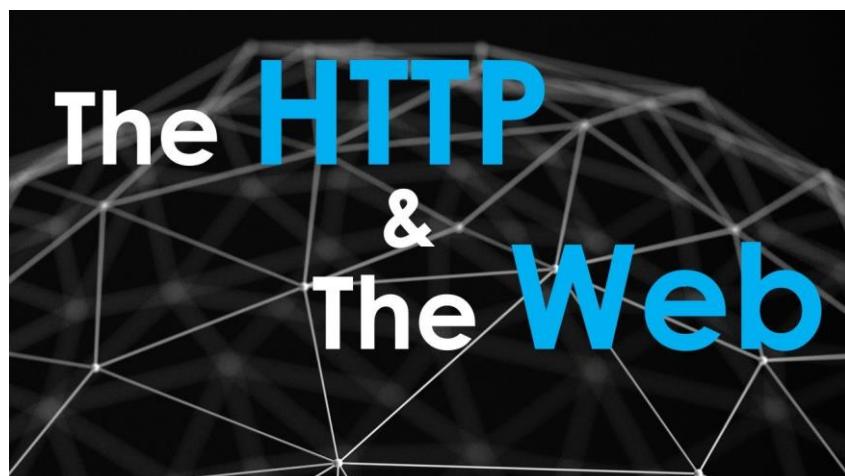
## BAB I DASAR

### 1.1 PENGENALAN WEB PROGRAMMING

Web programming, Pada buku ini penulis akan memaparkan materi tentang teknologi tersebut dengan penerapan teknologi pembuatan aplikasi berbasis Web Programming.

#### 1.1.1 Pengantar Web Programming

##### 1.1.1.1 HTTP & WEB



Web Programming atau biasa disebut Pemrograman Web, Web programming terdiri dari 2 kata yaitu web dan programming, programming dapat di artikan sebagai proses pembuatan suatu program. Sedangkan web dapat di artikan sebagai jaringan computer atau biasa disebut *website* yang terdiri dari situs jaringan internet yang menawarkan fitur dengan berbagai jenis seperti text, grafik, suara, Serta memelihara kode untuk membuat suatu pemrograman computer, kode ini ditulis dalam berbagai Bahasa pemrograman yang bertujuan untuk membuat suatu program yang dapat melakukan suatu perhitungan atau proses sesuai dengan keinginan pemrogram yang di akses melalui protocol HTTP.

Apa aitu HTTP? HTTP Merupakan protocol lapisan aplikasi (application layer) yang dikembangkan untuk membantu suatu proses transfer antar computer. Protokol ini berfungsi untuk melakukan transfer dokumen, file, gambar, dan video antar computer, Protokol HTTP menyediakan kumpulan suatu perintah didalam komunikasi antar suatu jaringan. Komunikasi tersebut

Fungsi HTTP yaitu mengatur Format dan bagaimana suatu data di transmisikan. HTTP juga berfungsi untuk mengatur bagaimana web server dan web browser saling terhubung dan memproses berbagai suatu perintah yang masuk. Fungsi lain dari HTTP ialah mengamankan data dari suatu pencurian dan hacker. Hal ini ditandai dengan munculnya HTTPS (Hypertext transfer Protocol Secure)..

### 1.1.2 HTML



HTML merupakan turunan atau pengembangan dari SGML (Standar Generalized Markup Language). HTML sendiri dikembangkan oleh Tim Berners-Lee sewaktu masih bekerja di CERN yang pertama kali dipopulerkan oleh browser Mosaic yang dikembangkan NCSA. Selama awal tahun 1990an, HTML semakin memiliki perkembangan yang sangat cepat. Akan tetapi pengembangan resmi HTML baru dikeluarkan oleh Internet Enggineering Task Force (IETF) yang dikeluarkan pada tahun 1995. HTML2 yang dikembangkan ini merupakan turunan dari HTML+ pada tahun 1993. HTML3 yang juga dirilis pada tahun 1995 mempunyai kemampuan yang jauh lebih bagus dari versi sebelumnya. Merupakan hasil dari usaha yang dikembangkan oleh World Wide Web Consortium's (W3C) yang kemudian menghasilkan HTML3 di tahun 1996 dan rilislah HTML4 pada akhir tahun tersebut yaitu 1997 dan 1998.

### **1.1.2.1 Perkembangan HTML**

1. HTML Versi 1.0 HTML Versi 1.0 merupakan pionir yang di dalamnya masih terdapat banyak sekali kelemahan hingga wajar jika tampilan yang dihasilkan sangat sederhana. Kemampuan yang dimiliki versi 1.0 ini antara lain heading, paragraf, hypertext, list, serta cetak tebal dan miring pada teks. Versi ini juga mendukung peletakan image pada dokumennya tanpa memperbolehkan teks di sekelilingnya (wrapping).
2. HTML Versi 2.0 Pada HTML Versi ini, penambahan kualitas HTML terletak pada kemampuannya untuk menampilkan suatu form pada dokumen. Dengan adanya form ini, kita dapat memasukkan nama, alamat, serta saran dan kritik. HTML versi 2.0 ini merupakan pionir dari adanya web interaktif.

### **3. HTML Versi 3.0**

Versi HTML 3.0 menambahkan beberapa fasilitas baru seperti FIGURE yang merupakan perkembangan dari IMAGE untuk meletakkan gambar dan tabel. Selain itu, HTML ini juga mendukung adanya rumus-rumus matematika dalam dokumennya. Versi ini yang disebut HTML+- tidak bertahan lama dan segera digantikan dengan versi 3.2.

### **4. HTML Versi 3.2**

HTML versi ini merupakan HTML yang sering digunakan. Di dalamnya terdapat suatu teknologi untuk meletakkan teks di sekeliling gambar, gambar sebagai latar belakang, tabel, frame, style sheet dan lain-lain. Selain itu pada HTML versi ini Kita bisa menggunakan script di luar HTML untuk mendukung kinerja HTML kita tersebut, seperti Javascript, VBScript dan lain-lain.

### **5. HTML Versi 4**

HTML ini memuat banyak sekali perubahan dan revisi dari pendahulunya yaitu HTML 3.2. Perubahan ini hampir terjadi di segala perintah HTML seperti tabel, image, link, text, meta, imagemaps, form dan lain-lain.

### **6. HTML Versi 4.01**

HTML versi 4.01 merupakan revisi dari HTML 4.0. Versi terbaru ini memperbaiki kesalahan-kesalahan kecil (minor errors) pada versi terdahulunya. HTML 4.01 ini juga menjadi standarisasi untuk elemen dan atribut dari script XHTML 1.0. 7.

### **7. HTML Versi 5.0**

Teknologi ini mulai diluncurkan pada tahun 2009, tetapi pada tanggal 4 Maret 2010 terdapat sebuah informasi bahwa W3C

(World Wide Web Consortium) dan IETF (Internet Engineering Task Force) yaitusebuah organisasi yang menangani HTML sejak versi 2.0 telah mengmbangkan versi HTML terbaru, yaitu versi 5.0. HTML 5 adalah sebuah prosedur pembuatan tampilan web baru yang merupakan penggabungan antara CSS, HTML itu sendiri dengan JavaScript.

Secara Umum Pemrograman Web dibagi menjadi 2 bagian yaitu :

### **1. Client Side Scripting (CSS)**

Client Side Scripting digunakan ketika browser (pengguna) klien memiliki semua kode dan halaman tersebut diubah berdasarkan informasi klien (pengguna). Browser Web mengeksekusi skrip sisi klien yang terletak di dalam komputer pengguna. Skrip sisi klien juga dikenal sebagai skrip tertanam (karena mereka sering disematkan dalam dokumen HTML atau XHTML).

Browser mendapatkan halaman yang dikirim oleh server & mengeksekusi skrip sisi klien. Skrip sisi klien tidak dapat digunakan untuk bergabung dengan database di server web. Skrip sisi klien tidak bisa mendapatkan sistem file yang terletak di server web.

Catatan dan pengaturan yang bersifat lokal di komputer klien (pengguna) dapat didekati menggunakan bahasa skrip sisi Klien. Secara umum diamati bahwa respons dari skrip sisi klien lebih cepat bila dibandingkan dengan bahasa skrip sisi server saat skrip disiapkan di komputer lokal.

## Contoh Script Paling Populer

1. Java Script

2. XML

3. CSS

## 2. Server Side Scripting (SSS)

Server Side Scripting adalah teknik yang digunakan dalam pengembangan web yang melibatkan penggunaan skrip pada server web yang menghasilkan respons yang disesuaikan untuk permintaan setiap pengguna (klien) ke situs web. Alternatifnya adalah untuk server web itu sendiri untuk memberikan halaman web statis. Skrip dapat ditulis dalam salah satu dari sejumlah bahasa skrip sisi server yang tersedia (lihat di bawah). Skrip sisi server dibedakan dari skrip sisi klien di mana skrip tertanam, seperti JavaScript, dijalankan sisi klien dalam browser web, tetapi kedua teknik ini sering digunakan bersama.

Server Side Scripting sering digunakan untuk menyediakan antarmuka yang disesuaikan untuk pengguna. Skrip ini dapat mengumpulkan karakteristik klien untuk digunakan dalam menyesuaikan respons berdasarkan karakteristik tersebut, persyaratan pengguna, hak akses, dll. Skrip sisi server juga memungkinkan pemilik situs web untuk menyembunyikan kode sumber yang menghasilkan antarmuka, sedangkan dengan sisi klien scripting, pengguna memiliki akses ke semua kode yang diterima oleh klien. Sisi buruk dari penggunaan skrip sisi server adalah bahwa klien harus membuat permintaan lebih lanjut melalui jaringan ke server untuk menunjukkan informasi baru kepada pengguna melalui browser web. Permintaan ini dapat memperlambat pengalaman bagi pengguna, menempatkan lebih

banyak beban di server, dan mencegah penggunaan aplikasi saat pengguna terputus dari server.

Ketika server menyajikan data dengan cara yang umum digunakan, misalnya sesuai dengan protokol HTTP atau FTP, pengguna dapat memilih sejumlah program klien (sebagian besar browser web modern dapat meminta dan menerima data menggunakan kedua protokol tersebut). Dalam hal aplikasi yang lebih khusus, pemrogram dapat menulis sendiri server, klien, dan protokol komunikasi mereka, yang hanya dapat digunakan satu sama lain.

Contoh Script Paling Populer:

- 1.PHP
- 2.ASP
- 3.JSP

### **1.1.2.2 Usur HTML**

#### **1 Tag**

Tag adalah simbol khusus (markup) berupa 2 karakter “<” dan “>” yang mengapit suatu teks sebagai nama tag. Contohnya tag <body> adalah tag dengan nama body.

#### **2 Atribut**

Atribut yaitu property yang mengatur bagaimana elemen dari suatu tag akan ditampilkan. Atribut ada yang memiliki nilai dan ada yang tidak memiliki nilai. Nilai suatu atribut ditulis di dalam tanda petik ganda (“...”), dipisahkan dengan symbol sama dengan (=) dari nama. Contohnya, <p align=”center”>.

### 3 ELEMENT

Element adalah bagian dari skrip HTML yang terdiri dari tag pembuka, isi element dan tag tutup. Jika element ditampilkan di browser maka hanya isi element saja yang akan tampil. Contohnya, <p> Politeknik Pos Indonesia </p> maka jika ditampilkan pada browser akan tampil hanya tulisa Politeknik Pos Indonesia.

#### 1.1.2.3 Struktur dasar HTML

Dalam penulisan HTML ada beberapa tag yang wajib dituliskan dengan struktur yang telah ditentukan. Tag tersebut adalah sebagai berikut :

```
1  <!DOCTYPE HTML>
2  <html>
3      <head>
4          <title>Fransiscus Ivan Martongam Sinaga</title>
5      </head>
6      <body>
7          <p>Halo dunia!</p>
8
9          <p><b>Tulisan tebal</b>, <i>tulisan miring</i>, <u>tulisan bergaris bawah</u></p>
10     </body>
11 </html>
```

- <!DOCTYPE HTML> adalah tag awal dari setiap dokumen HTML yang berfungsi untuk menginformasikan pada browser bahwa dokumen yang sedang dibuka adalah dokumen HTML. □
- <html>....</html> adalah tag yang menunjukkan pembuka dan penutup dokumen HTML.

- <head>....</head> adalah tag yang digunakan untuk menyimpan berbagai informasi tentang dokumen HTML.
- <title>....</title> adalah tag yang digunakan untuk membuat judul website yang nanti akan muncul di browser.
- <body>....</body?> Adalah tag yang menunjukkan bagian utama website. Semua yang akan ditampilkan pada halaman browser dituliskan di tag ini.

#### **1.1.2.4 HUBUNGAN PHP DAN HTML**



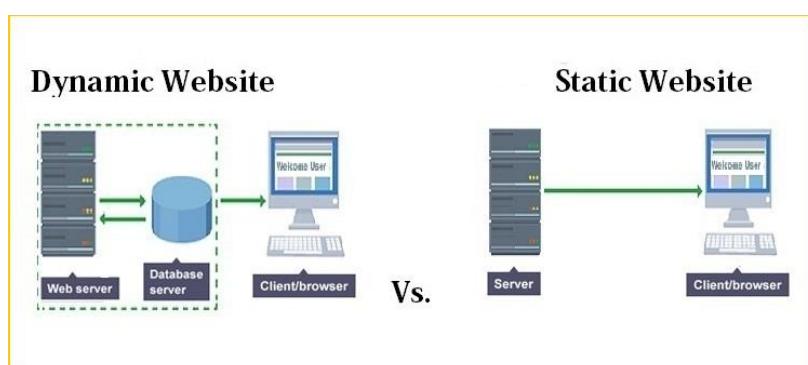
Halaman web biasanya disusun dari kode-kode html yang disimpan dalam sebuah file berekstensi .html. File html ini dikirimkan oleh server (atau file) ke browser, Kemudian browser menerjemahkan kode-kode tersebut sehingga menghasilkan suatu tampilan yang indah. Lain halnya dengan program php, program ini harus diterjemahkan oleh web-server sehingga menghasilkan kode html yang dikirim ke browser agar dapat ditampilkan. Program ini dapat berdiri sendiri ataupun disisipkan di antara kode-kode html sehingga dapat langsung ditampilkan bersama dengan kode-kode html tersebut. Program php dapat ditambahkan dengan mengapit program tersebut diantara tanda. Tanda-tanda tersebut biasanya disebut tanda untuk escaping (kabur) dari kode html. File html yang telah dibubuhinya program php harus diganti ekstensi-nya menjadi .php3 atau php. Php merupakan bahasa pemrograman web yang bersifat server-side

HTML=embedded scripting, di mana script-nya menyatu dengan HTML dan berada si server. Artinya adalah sintaks dan perintah-perintah yang kita berikan akan sepenuhnya dijalankan di server tetapi disertakan HTML biasa. PHP dikenal sebagai bahasa scripting yang menyatu dengan tag HTML, dieksekusi di server dan digunakan untuk membuat halaman web yang dinamis seperti ASP (Active Server Pages) dan JSP (Java Server Pages).

#### 1.1.2.5 Maksud Permrograman Web Dan Jenisnya

Web adalah fasilitas dari Hypertext yang memiliki fungsi untuk menampilkan data berupa text, gambar, suara, animasi dan data multimedia dan jika ingin dapat menguasai web maka diperlukan mengenal Bahasa pemrograman Web yaitu HTML dan PHP. HTML termasuk kedalam kategori Script Client Side sedangkan PHP termasuk Kedalam Script Server Side yang akan dimasukan perintah-perintah didalam suatu pemrograman web.

Web mengalami perkembangan yang sangat pesat mulai dari situs web E-commerce sampai dengan non profit situs. Dan dapat dikategorikan menjadi 2 yaitu:



## **1. Web Statis**

Web statis ialah web yang berisi tentang informasi informasi yang memiliki sifat statis (tetap) atau pengguna tidak dapat berinteraksi dengan website tersebut, web statis dapat dilihat dari tampilan website tersebut jika suatu web hanya berhubungan dengan halaman web lain yang berisi informasi tetap maka web tersebut termasuk kedalam kategori web statis, pada web statis pengguna hanya dapat melihat isi web tersebut dan jika di klik hanya akan berpindah pada halaman lainnya. Dalam web statis interaksi pengguna sangatlah terbatas

## **2. Web Dinamis**

Web dinamis adalah web yang dapat menampilkan informasi serta dapat membuat pengguna berinteraksi seperti dengan form input, button sehingga dapat mengolah informasi yang ditampilkan pada web tersebut, web dinamis bersifat tidak kaku dan terlihat lebih enak dipandang.

### **1.1.3 BASIS DATA**

#### **1. Pengertian Basis Data**

Basis Data terdiri dari kata basis dan data. Basis dapat diartikan sebagai markas atau gudang. Sedangkan data adalah catatan atas kumpulan fakta dunia nyata yang mewakili objek seperti manusia, barang, hewan, konsep, peristiwa dan sebagainya yang diwujudkan

dalam bentuk huruf, angka, simbol, gambar, teks, bunyi atau kombinasinya.

Sebagai suatu kesatuan maka pengertian basis data atau biasa disebut *database* adalah sebagai berikut:

Pengertian Basis Data atau *Database*:

1. Himpunan kelompok data yang saling terhubung dan diorganisasi sedemikian rupa supaya kelak dapat dimanfaatkan kembali secara cepat dan mudah.
2. Kumpulan data dalam bentuk file/tabel/arsip yang saling berhubungan dan tersimpan dalam media penyimpanan elektronis, untuk kemudahan dalam pengaturan, pemilahan, pengelompokan dan pengorganisasian data sesuai tujuan.

Dengan basis data seseorang dapat menyimpan sebuah informasi, seperti data mahasiswa, kepegawaian atau produk ke dalam media penyimpanan elektronis seperti cakram magnetis (*disk*) melalui perangkat komputer, Untuk kemudian data tersebut dapat kita gunakan sesuai keperluan. Database mempunyai 8 operasi dasar diantaranya adalah *Create database, Drop database, create table, Drop table, Insert, Read, Update dan Delete*.

## **2. Pengertian Sistem Basis Data**

Sistem basis data adalah sistem yang terdiri atas kumpulan tabel data yang saling berhubungan dan kumpulan program yang

memungkinkan beberapa pemakai atau program lain untuk mengakses dan memanipulasi tabel tabel data tersebut.

### **3. Komponen Sistem Basis Data**

#### a. Perangkat Keras

Perangkat keras atau hardware yang umumnya terdapat dalam sistem basis data adalah komputer, hard disk, memori sekunder offline (removable disk, fd), perangkat komunikasi jaringan.

#### b. Sistem Operasi

Sistem operasi adalah program yang dirancang untuk mengaktifkan sistem komputer dan mengendalikan seluruh sumber daya yang ada di dalamnya termasuk operasi- operasi dasar komputer. seperti Windows, Unix dan Linux.

#### c. Basis Data

Komponen adalah sekumpulan data yang terorganisir dengan baik sehingga data tersebut mudah disimpan, diakses, dan juga dapat dimanipulasi. Sistem basis data dapat terdiri dari beberapa basis data yang memiliki data masing- masing.

#### d. Database Management System (DBMS)

DBMS atau database management system adalah program aplikasi khusus yang dirancang untuk membuat dan juga mengelola database yang tersedia. Sistem ini berisi koleksi data dan set program yang digunakan untuk mengakses database tersebut.

DBMS adalah software yang berperan dalam mengelola, menyimpan, dan mengambil data kembali. Adapun mekanisme yang digunakan sebagai pelengkap adalah pengaman data, konsistensi data dan pengguna data bersama.

**Contoh dari DBMS** adalah *Microsoft Access, MySql, Oracle database, Sybase, Borland-Interbasi, PostgreSQL dll.*

e. Pemakai atau User

User adalah salah satu komponen database yang berinteraksi secara langsung dengan database. Ada beberapa tipe user, diantaranya, programmer aplikasi, User mahir (casual user), user umum (end user) dan user khusus (specialized user)

f. Aplikasi atau Perangkat Lain

Aplikasi ini tergantung kebutuhan, pemakai basis data bisa dibuatkan program khusus untuk melakukan pengisian, pengubahan atau pengambilan data yang mudah dalam pemakaiannya. Program tersebut ada yang tersedia langsung dalam DBMS atau dibuat menggunakan aplikasi lain seperti misalnya Visual Basic.

#### **4. Bahasa Basis Data (Database Language)**

Bahasa database merupakan bahasa data yang dapat ditempelkan kedalam bahasa pemrograman yang lain, sebut saja Java, Pascal, Fortran dst. Bahasa dimana instruksi data base menempel disebut inang. Beberapa komponen Bahasa data base menurut fungsinya dibagi menjadi:

a. *Data Definition Language*

*Data definition language* adalah sekumpulan definisi yang disimpan di dalam data dictionary.

b. *Data Manipulation Language*

*Data Manipulation Language* berisi akumulasi dari operasi manipulasi basis data yang dilakukan. Ini biasa disebut dengan

bahasa query sebab biasanya digunakan untuk meminta informasi yang ada dari basis data tersebut.

## 5. Fungsi dan Tujuan Basis Data

Fungsi basis data cukup banyak dan cakupannya pun luas dalam mendukung keberadaan lembaga atau organisasi, diantaranya adalah:

a. Ketersediaan/ *Availability*

Fungsi basis data yang pertama adalah untuk menyediakan data-data penting saat sedang diperlukan. Ya, ini adalah fungsi penting dari basis data yang meskipun tidak terletak dalam satu lokasi, dan tersimpan dalam bentuk disk, akan tetapi dengan cara penyimpanan yang sistematik, informasi tersebut mudah untuk didapatkan.

b. Mudah dan Cepat/ *Speed*

Selanjutnya, fungsi dari basis data ini adalah agar Anda sebagai pengguna bisa dengan mudah mengaksesnya saat sedang membutuhkan. Tidak perlu tunggu nanti, apalagi harus mengalokasikan waktu tertentu untuk memanggilnya.

c. Kelengkapan/ *Completeness*

Basis data harus menyimpan data yang lengkap, yang bisa melayani keperluan penggunanya secara keseluruhan. Meski kata lengkap yang dipakai disini sifatnya relatif, namun setidaknya data tersebut membantu memudahkan untuk menambah koleksi data, dan menjamin mudahnya pengguna untuk memodifikasi struktur data yang ada, sebut saja field-field data yang tersedia.

d. *Accuracy dan Security*

Fungsi data base selanjutnya adalah untuk accuracy atau keakuratan. Jadi, agar kesalahan dapat ditekan semaksimal mungkin, Anda bisa lakukan pengorganisasian file-file database dengan baik untuk menghindari kesalahan pada proses data entry dan juga dalam proses penyimpanan atau datastore.

Selain itu, fungsi database adalah untuk security atau keamanan. Ada fasilitas pengaman data yang disediakan oleh sistem basis data yang baik sehingga data tidak bisa dimodifikasi, diakses, diubah maupun dihapus oleh yang tidak mendapatkan hak untuk melakukannya.

e. *Storage Efficiency*

Pengorganisasian data dilakukan dengan baik dengan tujuan untuk menghindari duplikasi data yang berpengaruh pada bertambahnya ruang penyimpanan dari basis data tersebut. pengkodean dan juga relasi data bermanfaat untuk menghemat space penyimpanan dalam basis data.

#### **1.1.4 SEJARAH**

Cabang ilmu Pemrograman cukup luas, dan erat kaitannya dengan disiplin ilmu yang lainnya. Hal ini bisa dilihat dari berbagai aplikasi yang merupakan hasil kombinasi dari berbagai ilmu.

#### **1.1.4.1 Sejarah MongoDB**



MongoDB merupakan database open source berbasis dokumen (Document-Oriented Database) yang awalnya dibuat dengan bahasa C++. MongoDB sendiri sudah dikembangkan oleh 10gen sejak Oktober 2007, namun baru dipublikasikan pada Februari 2009. Selain karena performanya 4 kali lebih cepat dibandingkan MySQL serta mudah diaplikasikan, karena telah tergabung juga sebagai modul PHP. Dalam konsep MongoDB tidak ada yang namanya tabel, kolom ataupun baris yang ada hanyalah collection (ibaratnya tabel), document (ibaratnya record). Data modelnya sendiri disebut BSON dengan struktur mirip dengan JSON. Strukturnya cukup mudah dibaca, contohnya seperti ini.

```
{  
  "nama" : "Yani Fitrianti",  
  "kontak" : {  
    "alamat" : "Jl. Imogiri Barat",  
    "kota" : "Yogyakarta",  
    "kodepos" : "55187",  
    "telp" : "62839288",  
  }  
}
```

#### **1.1.4.2 Kelebihan MongoDB**

1. Performa yang ditawarkan MongoDB lebih cepat dibandingkan MySQL ini disebabkan oleh memcached dan format dokumennya yang berbentuk seperti JSON
2. Replikasi, adalah fitur yang sangat bermanfaat untuk backup data secara realtime. MongoDB sangat cocok digunakan untuk portal berita ataupun blog, namun belum cocok untuk digunakan pada sistem informasi yang berkaitan dengan keuangan karena MongoDB tidak mendukung transaction SQL
3. Auto-sharding, merupakan fitur untuk memecah database yang besar menjadi beberapa bagian demi optimalisasi performa database. Penggunaannya sendiri sangat berguna ketika memiliki website dengan database yang jutaan baris, sharding akan membantu memecahnya menjadi beberapa bagian
4. MongoDB juga sudah mendukung C, C++, C#, Erlang, Haskell, Java, JavaScript, .NET(C# F#, PowerShell), Lips, Perl, PHP, Python, Ruby dan Scala

5. Cross-platform, sehingga dapat digunakan di Windows, Linux, OS X dan Solaris

#### **1.1.4.3 Kekurangan MongoDB**

1. MongoDB harus diinstall di sebuah server.
2. MongoDB belum support di banyak hosting.
3. Tidak cocok untuk aplikasi proses transaksi.

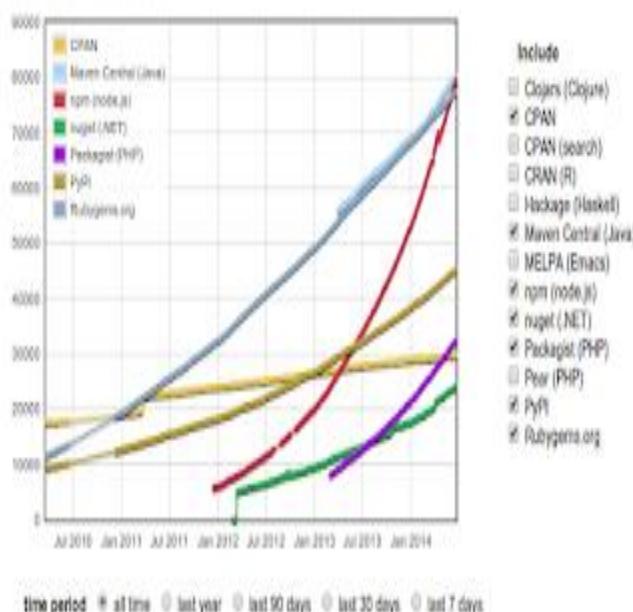
#### **1.1.4.4 Sejarah NODE.JS**



NodeJS itu platform perangkat lunak pada *server side* namun dalam perkembangannya client side juga dapat berjalan di berbagai macam OS (Operating System) Windows, Mac OS, dan Linux tanpa adanya perubahan pada kode program itu sendiri. Dan di tulis dengan bahasa *javascript*. Teknologi pada sebagian besar *webserver* seperti Apache HTTP Server modelnya menggunakan thread. jadi untuk setiap *request* yang di terima maka akan di buat satu *thread* untuk melayaninya. *Thread* itu sendiri memakai sumber daya dari sistem, *Thread* bisa juga berjalan pada satu waktu secara bersamaan. Di aplikasi *realtime thread* harus dibuat hidup untuk mendukung koneksi yang telah dibuat oleh seorang *client*. dan mengakibatkan jumlah pada *thread* yang dapat hidup tadi terbatas tergantung sumber daya sistem yang tersedia. Intinya apabila terjadi jumlah *request*

yang banyak melebihi kemampuan *server*, Maka *request* tersebut akan di tahan dulu sampai ada *thread* yang tersedia kembali karena ada *blocking* yang terjadi. untung nya NodeJs itu sendiri menggunakan teknik *non-blocking* untuk mempercepat process. NodeJS mempunyai pustaka *server* HTTP sendiri, Jadi tanpa atau adanya *webserver* seperti Apache, Lighttpd, dan lain-lain. NodeJs itu dapat berjalan. Sebagai Platform yang tergolong masih baru, Tanpa disadari NodeJS mampu menarik hati para developer yang dimana npm nya menjadi salah satu package manager terbesar. Makanya kenapa banyak perusahaan menggunakan NodeJS sebagai basis sistemnya.

## Module Counts



#### **1.1.4.5 Sejarah CSS**



Pada tanggal 17 Agustus 1996, World Wide Web Consortium (W3C) menjadikan CSS sebagai bahasa pemrograman standart dalam pembuatan dokumen web. Tujuannya adalah mengurangi pembuatan tag-tag baru oleh Netscape dan Internet Explorer, karena kedua browser pada saat itu bersaing mengembangkan TAG sendiri untuk mengatur tampilan web.

CSS level 1 mendukung pengaturan tampilan dalam hal:

1. Font (jenis, ketebalan)
2. Warna, teks, backgrounf dan elemen lainnya
3. Text attributes, misalnya spasi antar baris, kata dan huruf
4. Posisi teks, gambar, table dan elemen lainnya
5. Marjin, border dan padding.

Selanjutnya di tahun 1998, W3c menyempurnakan CSS awal dengan menciptakan standar CSS2 (CSS level 2) yang menjadi standar hingga saat

ini. Pada CSS level 2 ini, di masukkan semua atribut dari CSS1, serta diperluas dengan penekanan pada Internasional accessibility and capability khususnya media-specific CSS. Bahkan pada tahun 2000, tidak lama setelah CSS2 diimplementasikan. CSS2 ini sampai sekarang masih terus dikembangkan, spesifikasinya dibagi pada beberapa topik atau modul.

## 1.1.5 BAHASA PEMROGRAMAN

### 1.1.5.1 Pemrograman Web



Seperti yang sudah di sebutkan bahwa ada 2 kategori didalam pemrograman web, yaitu pemrograman Server Side dan Client side. Dalam pemrograman server side terdapat perintah program yang dijalankan pada server web lalu hasil akan dikirimkan dalam bentuk HTML biasa. Adapun

pada Client Side program dijalankan pada browser web dari sever kemudian akan dijalankan oleh browser yang bersangkutan

### **A. Mengenal Script Client Side**

Program web yang tergolong dalam Client Side Seperti Java Script, VB , HTML dll. Hasil dari parshing script pemrograman Client Side yang berupa Html dari web server dapat dilihat dengan menu view > source code dapat terlihat bahwa script program yang ditulis ditampilkan pada halaman source code.

### **B. Java Script**

JavaScript diperkenalkan pertama kali oleh Netscape di tahun 1995.pada mulanya Bahasa ini memiliki nama Livescript dan memiliki fungsi sebagai Bahasa sederhana untuk browser NetScope Navigator2. Pada masa pertama kali rilis Bahasa ini memiliki sangat banyak kritikan karena kurang aman, tidak ada pesan kesalahan dari setiap script program yang ditampilkan. Kemudian dengan kerja sama dengan Netscape dengan SUN (pengembang Bahasa pemrograman) maka Netscape merubah nama menjadi javascript pada tanggal 4 desember 1995.

Java Script adalah Bahasa pemrograman yang sederhana karena Bahasa yang digunakan tidak dapat membuat aplikasi, java script dapat dengan mudah kita temukan dalam suatu program yang sudah interaktif . program java script dituliskan pada file HTML. Dengan kata lain tidak perlu menuliskan program java script pada file terpisah. Bahasa ini adalah Bahasa pemrograman untuk memberikan kemampuan tambahan

terhadap Bahasa HTML. Dengan mengizinkan menjalankan perintah dari sisi Klien. Yang artinya disisi browser bukan disisi server web javascript bergantung kepada browser memanggil pada halaman web yang berisi script java script tidak memerlukan penerjemah khusus untuk menjalankan script tersebut.

### **1.1.5.2 Keistimewaan MySql**

Sebagai database server yang memiliki konsep database modern, MySQL memiliki banyak sekali keistimewaan. Berikut ini beberapa keistimewaan yang dimiliki oleh :

#### **a. Portability**

MySQL dapat berjalan stabil pada berbagai OS seperti Windows,

Linux, Unix, Mac OS, Solaris, Unix, Amiga, HP-UX, Symbian.

#### **b. Open Source "limited"**

Dahulu MySQL didistribusikan secara open source (gratis), dibawah lisensi GPL sehingga kita dapat menggunakannya secara cuma-cuma tanpa dipungut biaya. Namun, saat ini karena MySQL telah dibeli oleh SUN, maka kita tidak dapat lagi menikmati fitur-fitur baru yang ada di

MySQL, karena SUN akan membatasi fitur-fitur baru ini hanya untuk user yang membeli lisensinya. Sehingga MySQL tidak lagi sebuah opensource yang benar-benar gratis lagi. MySQL sekarang hanya menyediakan fitur-fitur "dasar" saja yang saat ini sudah menggunakan versi 5.1. Untuk mendownloadnya silahkan download di sini dan dicari versi MySQL dengan OS kita.

#### **c. Multiuser**

MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami konflik. Hal ini memungkinkan sebuah database server MySQL dapat diakses klien secara bersamaan.

#### **d. Performance Tuning**

MySQL memiliki kecepatan yang menakjubkan dalam menangani query sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu

#### **e. Column Types**

MySQL memiliki tipe kolom yang sangat kompleks, seperti signed/unsigned integer, float, double, char, varchar, text, blob, date, time, datetime, timestamp, year, set serta enum.

#### **f. Command dan Functions**

MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah SELECT dan WHERE dalam query.

#### **g. Security**

MySQL memiliki beberapa lapisan sekuritas seperti level subnetmask, nama host, dan izin akses user dengan sistem perizinan yang mendetail serta password terenkripsi.

#### **h. Scalability dan Limits**

MySQL mampu menangani database dalam skala besar dengan jumlah records lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris. Selain itu, batas index yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

### **i. Connectivity**

MySQL dapat melakukan koneksi dengan klien menggunakan

TCP/IP, Unix soket (Unix), atau Named Pipes (NT).

### **j. Localisation**

MySQL dapat mendekripsi pesan kesalahan (error code) pada klien

dengan menggunakan lebih dari dua puluh bahasa

### **k. Interface**

MySQL memiliki interface terhadap berbagai aplikasi dan bahasa

pemrograman dengan menggunakan fungsi API (Application

Programming Interface).

### **l. Clients dan Tools**

MySQL dilengkapi dengan berbagai tool yang dapat digunakan untuk

administrasi database, dan pada setiap tool yang ada disertakan

petunjuk online.

### **m. Struktur Tabel**

MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani

ALTER TABLE dibandingkan database lainnya.

#### **1.1.5.3 PHP**

PHP Merupakan singkatan dari "PHP: Hypertext Preprocessor", yang merupakan sebuah bahasa scripting (kode untuk membangun suatu program) yang dikombinasikan pada HyperText Markup Language (HTML). Tujuan utama penggunaan bahasa ini adalah untuk memungkinkan perancang web menulis halaman web dinamik dengan cepat

#### **1. Memiliki Community yang besar**

Programmer Web mana yang tidak mengetahui PHP, semua web programmer paling tidak pasti pernah mencoba PHP. Banyak sekali website yang menggunakan PHP sebagai bahasa pemrograman untuk membuat aplikasi web atau website nya. Facebook, Yahoo, Wikipedia, WordPress adalah contoh website terkenal yang menggunakan PHP. Forum untuk membahas dan juga saling bertukar pikiran dalam pemrograman PHP juga telah banyak muncul di berbagai situs. Kebanyakan kuliah di bidang IT mengajarkan PHP sebagai bahasa pemrograman awal untuk mahasiswanya yang berkuliah di jurusan website development ( baca juga: Daftar Mata Kuliah Teknik Informatika)

## **2. Mudah Dipelajari**

PHP mudah di install dan dikonfigurasi. membuatnya menjadi bahasa pemrograman tingkat entry level yang mudah dipelajari bagi seseorang yang baru memulai belajar pengembangan web. Tutorial untuk memulai belajar pemrograman PHP dapat diperoleh dengan mudah secara online, di toko buku, ataupun di lembaga bimbingan kursus pengembangan website.

## **3. Pengembangan Cepat**

Membuat Aplikasi menggunakan PHP jauh lebih cepat daripada mengembangkan aplikasi web menggunakan bahasa pemrograman lain. banyak sekali tools, boiler yang tersedia secara open source untuk bahasa pemrograman PHP. hal ini mempercepat proses dari start sampai dengan finish sebuah projek pembuatan aplikasi web.

## **4. Ringkas**

Bagi Programmer web yang pernah mencoba bahasa ASP maupun java pasti mengetahui betul satu kelebihan ini. Mulai dari proses install yang tidak perlu setting berlebihan, konfigurasi dengan database yang mudah. hingga proses pengembangan yang tidak memerlukan waktu kompilasi. membuat PHP terasa sangat ringkas dan praktis berbeda dengan bahasa pemrograman lain yang membutuhkan proses kompilasi untuk dapat melihat website yang telah diselesaikan pembuatannya. Bahkan, bahasa pemrograman php dapat digunakan didalam dokumen html, hmm ringkas bukan. (baca juga: Dasar Dasar HTML)

## **5. Maintenance mudah**

Sekali web yang menggunakan PHP berjalan, programmer dapat dengan mudah melakukan update dari software PHP dengan mudah jika memang

diperlukan. karena sifat PHP yang merupakan interpreter. Aplikasi web yang dibuat dengan menggunakan PHP dapat dengan mudah diupgrade versi PHP tanpa harus melakukan kompilasi ulang source code. berbeda sekali dengan bahasa pemrograman lain yang membutuhkan kompilasi ulang jika melakukan upgrade versi dari bahasa pemrograman. PHP juga dapat berjalan pada berbagai macam web server seperti apache, nginx, dan IIS. ( baca juga : Pengertian Web Server Menurut Para Ahli )

## **6. Open Source**

PHP merupakan sebuah projek Open source dengan license yang dikeluarkan oleh PHP group yaitu PHP license V3.01. Inti dari license ini adalah setiap pengguna program PHP bebas menggunakan PHP secara gratis tanpa harus memberikan royalty apapun ke PHP group namun tetap wajib mencantumkan licensi atas PHP yang dimiliki PHP Group. Dengan kata lain selama pemakai program PHP tidak mengakui produk PHP adalah buatannya maka perjual belian program yang menggunakan PHP diperbolehkan tanpa harus membayar licensi apapun.

## **7. Perkembangan Pesa**

Karena sifat PHP yang open source, banyak sekali bermunculan projek projek open source besar yang menggunakan PHP seperti Prestashop, WordPress, Drupal, dan lain lain. Hal ini menjadi keunggulan yang sangat besar bagi orang yang menguasai pemrograman PHP. Dengan sangat luasnya perkembangan PHP, maka kesempatan untuk bisnis ataupun kerja pada bidang pemrograman PHP sangatlah luas

#### **1.1.5.4 Definisi CSS**

CSS adalah singkatan dari Cascading Style-Sheet, sebuah pengembangan atas kode HTML yang sudah ada sebelumnya. Dengan CSS, kita bisa menentukan sebuah struktur dasar halaman web secara lebih mudah dan cepat, serta irit size. CSS juga membantu kita untuk menyeragamkan seluruh halaman website dengan tampilan yang konsisten. Misalnya, kita mau seluruh font yang ada di website kita adalah font "Tahoma", maka dengan bantuan CSS kita bisa bikin proses itu menjadi otomatis tanpa harus mengganti-ganti font secara manual di setiap halaman.

Dahulu kala, sebelum CSS menjadi standar untuk mendesain halaman web seperti sekarang, halaman web di desain menggunakan <table>. jadi dibuat dulu desainnya, dalam format .psd atau jpeg, lalu di slice atau di potong potong menjadi bagian – bagian terpisah. setelah itu dibuat table dengan ukuran yang sesuai, lalu desain tadi di “tempel” pada table sebagai image yang melekat pada tabel, atau sebagai background. Kelemahan dari cara seperti ini adalah, halaman web menjadi berat karena kolom – kolom penyusun table <tr> dan <td> diberi tambahan atribut image source. Side effect dari hal ini adalah halaman web menjadi padat dan tidak SEO friendly.

Pada perkembangannya CSS sudah masuk level 3 untuk sekarang, dimana dimulai CSS level 1 atau yang sering disebut CSS saja, kemudian level 2 yang merupakan Penyempurnaan dari CSS level sebelumnya, yaitu CSS level 1. CSS merupakan alternatif bahasa pemrograman web masa yang akan datang, dimana mempunyai banyak keuntungan, diantaranya :

- a. Ukuran file lebih kecil
- b. Load file lebih cepat
- c. Dapat berkolaborasi dengan Javascript
- d. Pasangan setia XHTML
- e. Menghemat pekerjaan tentunya, dimana hanya membuat 1 halaman CSS.
- f. Mudah mengganti-ganti tampilan dengan hanya merubah file CSS nya saja.

#### **1.1.5.5 Definisi Bahasa Pemrograman C**

Bahasa Pemrograman C adalah sebuah bahasa pemrograman komputer yang bisa digunakan untuk membuat berbagai aplikasi (general-purpose programming language), mulai dari sistem operasi (seperti Windows atau Linux), antivirus, software pengolah gambar (image processing), hingga compiler untuk bahasa pemrograman, dimana C banyak digunakan untuk membuat bahasa pemrograman lain yang salah satunya adalah PHP.

Meskipun termasuk general-purpose programming language, yakni bahasa pemrograman yang bisa membuat berbagai aplikasi, bahasa pemrograman C paling cocok merancang aplikasi yang berhubungan langsung dengan Sistem Operasi dan hardware. Ini tidak terlepas dari tujuan awal bahasa C dikembangkan.

Bahasa pemrograman C dibuat pertama kali oleh Dennis M. Ritchie pada tahun 1972. Saat itu Ritchie bekerja di Bell Labs, sebuah pusat penelitian yang berlokasi di Murray Hill, New Jersey, Amerika Serikat. Ritchie membuat bahasa pemrograman C untuk mengembangkan sistem operasi UNIX. Sebelumnya, sistem operasi UNIX dibuat menggunakan bahasa assembly (assembly language). Akan tetapi bahasa assembly sendiri sangat rumit dan susah untuk dikembangkan.

Dengan tujuan mengganti bahasa assembly, peneliti di Bell Labs membuat bahasa pemrograman B. Namun bahasa pemrograman B juga memiliki beberapa kekurangan, yang akhirnya di lengkapi oleh bahasa pemrograman C. Dengan bahasa C inilah sistem operasi UNIX ditulis ulang. Pada gilirannya, UNIX menjadi dasar dari banyak sistem operasi modern saat ini, termasuk Linux, Mac OS (iOS), hingga sistem operasi Android.

#### **1.1.5.5.1 Fitur dan Keunggulan Bahasa Pemrograman C**

Berikut beberapa fitur serta keunggulan bahasa pemrograman C jika dibandingkan dengan bahasa pemrograman lain:

##### **1. C Sebagai Bahasa pemrograman procedural.**

Konsep pemrograman prosedural adalah sebuah metode pemrograman yang setiap baris perintah diproses secara berurutan dari baris paling atas hingga baris paling bawah. Selain itu bisa terdapat fungsi tambahan (*function*) yang digunakan untuk menyelesaikan berbagai tugas. Bahasa pemrograman C termasuk ke dalam kelompok ini.

Selain konsep prosedural, terdapat juga konsep pemrograman object (*object-oriented programming*). Di dalam bahasa pemrograman object, setiap tugas akan dijalankan menggunakan *class* dan *object*. Contoh bahasa pemrograman object adalah JAVA.

Bagi pemula, sangat disarankan untuk mempelajari bahasa pemrograman prosedural terlebih dahulu baru kemudian masuk ke dalam bahasa pemrograman object. Ini juga menjadi alasan untuk belajar bahasa C sebelum masuk ke bahasa pemrograman object seperti JAVA.

Beberapa bahasa pemrograman ada yang mendukung konsep prosedural dan object sekaligus, contohnya bahasa pemrograman C++, Python dan PHP.

## **2. Bahasa C sangat efisien.**

Aplikasi yang dibuat menggunakan bahasa C bisa dieksekusi dengan sangat cepat serta berukuran kecil. Ini karena C bisa langsung berkomunikasi dengan hardware, sebuah fitur yang jarang tersedia di bahasa pemrograman modern seperti JAVA, PHP, maupun Python. Akan tetapi, hal ini juga memiliki kelemahan. Bahasa C relatif sederhana dan tidak memiliki fitur-fitur modern seperti garbage collection dan dynamic typing.

## **3. C adalah portable language.**

Maksudnya, bahasa pemrograman C bisa di-compile ulang supaya berjalan di berbagai sistem operasi tanpa perlu mengubah kode-kode yang ada. Aplikasi yang dibuat di Windows dengan bahasa C, bisa dipindahkan ke Linux dengan sedikit atau tanpa modifikasi.

## **4. C merupakan induk dari Bahasa pemrograman modern.**

Bahasa pemrograman C banyak menginspirasi bahasa pemrograman lain, seperti C++, C#, Objective C, PHP, JAVA, JavaScript dan masih banyak lagi. Dengan mempelajari bahasa C, anda akan familiar dan lebih mudah saat berpindah ke bahasa pemrograman lain yang merupakan turunan dari bahasa C.

### **1.1.5.6 Definisi Bahasa Pemrograman Java**

Java adalah bahasa pemrograman tingkat tinggi yang berorientasi objek dan program java tersusun dari bagian yang disebut kelas. Kelas terdiri atas metode-metode yang melakukan pekerjaan dan mengembalikan informasi setelah melakukan tugasnya. Para pemrogram Java banyak mengambil keuntungan dari kumpulan kelas di pustaka kelas Java, yang disebut dengan *Java Application Programming Interface* (API). Kelas-kelas ini diorganisasikan menjadi sekelompok yang disebut paket (package). Java API telah menyediakan fungsionalitas yang memadai untuk menciptakan applet dan aplikasi canggih. Jadi ada dua hal yang harus dipelajari dalam Java, yaitu mempelajari bahasa Java dan bagaimana mempergunakan kelas pada Java API. Kelas merupakan satu-satunya cara menyatakan bagian eksekusi program, tidak ada cara lain. Pada Java program javac untuk mengkompilasi file kode sumber Java menjadi kelas-kelas bytecode. File kode sumber mempunyai ekstensi \*.java. Kompilator javac menghasilkan file bytecode kelas dengan ekstensi \*.class. Interpreter merupakan modul utama sistem Java yang digunakan aplikasi Java dan menjalankan program bytecode Java.

Beberapa keunggulan java yaitu java merupakan bahasa yang sederhana. Java dirancang agar mudah dipelajari dan digunakan secara efektif. Java tidak menyediakan fitur-fitur rumit bahasa pemrograman tingkat tinggi, serta banyak pekerjaan pemrograman yang mulanya harus dilakukan manual, sekarang digantikan dikerjakan Java secara otomatis seperti dealokasi memori. Bagi pemrogram yang sudah mengenal bahasa C++ akan cepat belajar susunan bahasa Java namun harus waspada karena mungkin Java mengambil arah (semantiks) yang berbeda dibanding C++.

Java merupakan bahasa berorientasi objek (OOP) yaitu cara ampuh dalam pengorganisasian dan pengembangan perangkat lunak. Pada OOP, program komputer sebagai kelompok objek yang saling berinteraksi. Deskripsi ringkas OOP adalah mengorganisasikan program sebagai kumpulan komponen, disebut objek. Objek-objek ini ada secara independen, mempunyai aturan-aturan berkomunikasi dengan objek lain dan untuk memerintahkan objek lain guna meminta informasi tertentu atau meminta objek lain mengerjakan sesuatu. Kelas bertindak sebagai modul sekaligus tipe. Sebagai tipe maka pada saat jalan, program menciptakan objek-objek yang merupakan instan-instan kelas. Kelas dapat mewarisi kelas lain. Java tidak mengijinkan pewarisan jamak namun menyelesaikan kebutuhan pewarisan jamak dengan fasilitas antarmuka yang lebih elegan.

Seluruh objek diprogram harus dideklarasikan lebih dulu sebelum digunakan. Ini merupakan keunggulan Java yaitu Statically Typed. Pemaksaan ini memungkinkan kompilator Java menentukan dan melaporkan terjadinya pertentangan (ketidakkompatibelan) tipe yang merupakan barikade awal untuk mencegah kesalahan yang tidak perlu (seperti mengurangkan variabel bertipe integer dengan variabel bertipe string). Pencegahan sedini mungkin diharapkan menghasilkan program yang bersih. Kebaikan lain fitur ini adalah kode program lebih dapat dioptimasi untuk menghasilkan program berkinerja tinggi.

Java menggunakan model pengamanan tiga lapis (three-layer security model) untuk melindungi sistem dari untrusted Java code. Pertama, bytecode verifier membaca bytecode sebelum dijalankan dan menjamin bytecode memenuhi aturan-aturan dasar bahasa Java. Kedua, class loader menangani pemuatan kelas Java ke runtime interpreter. Ketiga, manajer keamanan menangani keamanan tingkat aplikasi dengan mengendalikan apakah

program berhak mengakses sumber daya seperti sistem file, port jaringan, proses eksternal dan sistem window.

Platform independence adalah kemampuan program bekerja di sistem operasi yang berbeda. Bahasa Java merupakan bahasa yang secara sempurna tidak bergantung platform. Tipe variabel Java mempunyai ukuran sama di semua platform sehingga variabel bertipe integer berukuran sama tidak peduli dimana program java dikompilasi. Begitu telah tercipta file .class dengan menggunakan kompilator Java di platform manapun, maka file .class tersebut dapat dijalankan di platform manapun. Jadi “dimanapun dibuat, dimanapun dapat dijalankan”. Slogan ini biasa diringkas sebagai Write Once, Run Anywhere (WORA).

Java termasuk bahasa Multithreading. Thread adalah untuk menyatakan program komputer melakukan lebih dari satu tugas di satu waktu yang sama. Java menyediakan kelas untuk menulis program multithreaded, program mempunyai lebih dari satu thread eksekusi pada saat yang sama sehingga memungkinkan program menangani beberapa tugas secara konkuren. Program Java melakukan garbage collection yang berarti program tidak perlu menghapus sendiri objek-objek yang tidak digunakan lagi. Fasilitas ini mengurangi beban pengelolaan memori oleh pemrogram dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat di bahasa yang memungkinkan alokasi dinamis.

Java mempunyai mekanisme exception-handling yang ampuh. Exception-handling menyediakan cara untuk memisahkan antara bagian penanganan kesalahan dengan bagian kode normal sehingga menuntun ke struktur kode program yang lebih bersih dan menjadikan aplikasi lebih tegar. Ketika kesalahan yang serius ditemukan, program Java menciptakan exception. Exception dapat ditangkap dan dikelola program tanpa resiko

membuat sistem menjadi turun. Program Java mendukung native method yaitu fungsi ditulis di bahasa lain, biasanya C/C++. Dukungan native method memungkinkan pemrogram menulis fungsi yang dapat dieksekusi lebih cepat dibanding fungsi ekivalen di java. Native method secara dinamis akan di-link ke program java, yaitu diasosiasikan dengan program saat berjalan.

Selain itu keuntungan menggunakan bahasa pemrograman Java antara lain. Memori pada Java secara otomatis dilengkapi garbage collector yang berfungsi mendealokasi memori yang tidak diperlukan. Tidak ada lagi upaya yang dilakukan pemrogram untuk melakukan dispose(). Kita tidak lagi dibebani urusan korupsi memori. Java menerapkan array sebenarnya, menghilangkan keperluan aritmatika pointer yang berbahaya dan mudah menjadi salah. Menghilangkan pewarisan jamak (multiple inheritance) diganti fasilitas antarmuka. Dan mudah dijalankan diberbagai platform.

Grafical User Interface (GUI) adalah salah satu kemampuan Java dalam mendukung dan manajemen antarmuka berbasis grafis. Tampilan grafis yang akan ditampilkan terhubung dengan program serta tempat penyimpanan data. Elemen dasar di Java untuk penciptaan tampilan berbasis grafis adalah dua paket yaitu AWT dan Swing. Abstract Windowing Toolkit (AWT), atau disebut juga “Another Windowing Toolkit”, adalah pustaka windowing bertujuan umum dan multiplatform serta menyediakan sejumlah kelas untuk membuat GUI di Java. Dengan AWT, dapat membuat window, menggambar, manipulasi gambar, dan komponen seperti Button, Scrollbar, Checkbox, TextField, dan menu pull-down.

Penggunaan komponen AWT ditandai dengan adanya instruksi :  
import java.awt.\*; Swing merupakan perbaikan kelemahan di AWT. Banyak kelas swing menyediakan komponen alternatif terhadap AWT. Contohnya kelas JButton swing menyediakan fungsionalitas lebih banyak dibanding

kelas Button. Selain itu komponen swing umumnya diawali dengan huruf “J”, misalnya JButton, JTextField, JFrame, JLabel, JTextArea, JPanel, dan sebagainya. Teknologi swing menggunakan dan memperluas gagasan-gagasan AWT. Sementara, penggunaan komponen Swing ditandai dengan adanya instruksi : import javax.swing.\*;

Beberapa perbedaan AWT dan Swing, AWT merupakan komponen heavyweight (kelas berat) sedangkan Swing lightweight (kelas ringan). Swing memiliki lebih banyak komponen. Fasilitas Swing Look and Feel : Metal, Windows, Motif. Komponen Swing berdasar model-view, yaitu suatu cara pengembangan komponen dengan pemisahan penyimpanan dan penanganan data dari representasi visual data.

Bahasa pemrograman Java merupakan salah satu bahasa pemrograman yang umum digunakan untuk mengembangkan aplikasi basis data yang dibuat menggunakan MySQL.

#### **1.1.5.7 Definisi Bahasa Pemrograman Python**

Python adalah bahasa pemrograman yang populer. Bahasa pemrograman ini dibuat oleh Guido van Rossum dan dikenalkan sejak tahun 1991. Sebelum memulai untuk belajar Python dasar, akan lebih baik untuk memahami dulu apa itu Python dan bagaimana cara kerjanya. Python termasuk bahasa pemrograman yang mudah untuk dipelajari. Sampai saat ini bahasa pemrograman Python hampir dipakai di segala bidang seperti game, sistem berbasis web, dan bahkan dapat membuat mesin pencari sendiri. Jadi secara umum, bahasa pemrograman ini dipakai dalam pengembangan website, pengembangan software, matematika, dan system scripting.

#### **1.1.5.7.1 Hal-hal yang dapat dilakukan dengan python.**

Sebelum belajar Phyton lebih jauh, Anda harus mengetahui apa saja yang bisa dilakukan dengan bahasa pemrograman ini.

Berikut ini beberapa hal yang dapat Anda lakukan menggunakan Python:

1. Python dapat menjadi salah satu bahasa pemrograman untuk membangun server ketika Anda membuat website.
2. Ketika Anda membutuhkan proses pembuatan prototipe atau pengembangan perangkat lunak siap produksi, Python dapat Anda andalkan.
3. Python dapat digunakan untuk membuat workflow di dalam pengembangan perangkat lunak.
4. Python dimanfaatkan untuk membaca dan memodifikasi sebuah file di dalam pembangunan sistem database.
5. Python memungkinkan Anda untuk menangani big data dan menjalankan pemrosesan matematika yang komplek.

#### **1.1.5.7.2 Manfaat belajar python**

Meskipun ada banyak sekali bahasa pemrograman di luar sana, akan tetapi saya merekomendasikan Anda untuk belajar Python. Tentu juga ada beberapa kelebihan yang perlu Anda ketahui. Jika dibandingkan dengan bahasa pemrograman lain, berikut kelebihan bahasa pemrograman Python:

1. Python memiliki sintaksis yang sederhana dan lebih mirip dengan Bahasa Inggris.
2. Python dapat berjalan di berbagai macam sistem operasi.
3. Python berjalan di dalam sistem interpreter, artinya bahasa baris kode bahasa pemrograman ini akan segera dieksekusi setelah ditulis.

4. Python dapat diperlakukan dengan cara prosedural, cara berorientasi objek atau cara fungsional.
5. Python memiliki sintaks yang memungkinkan pengembang untuk menulis program dengan ringkas daripada bahasa pemrograman lain.

### **1.1.5.7.3 Komponen Python**

Sesudah Anda memastikan Python sudah terinstall dengan baik di perangkat. Langkah selanjutnya adalah melakukan percobaan beberapa eksekusi program Python. Namun sebelum itu akan lebih baik jika mengetahui terlebih dahulu apa saja komponen yang terdapat di dalam Python.

#### **1. Sintaks**

Python sintaks dapat dieksekusi langsung dengan mengetikkannya di Command Line. Selain itu, Anda dapat membuat file Python di dalam server menggunakan ekstensi .py dan menjalankannya menggunakan Command Line.

```
>>> print("Hello, World!")Hello, World!
```

#### **2. Komentar**

Sama seperti bahasa pemrograman lainnya, Python juga memiliki kode untuk menjadikan baris program menjadi komentar. Anda dapat menggunakan tanda pagar '#' untuk menjadikan baris kode di Python menjadi komentar.

```
# Ini adalah baris komentar di Python.print("Hello, World!")
```

### 3. Python Identitations

Berbeda dengan bahasa pemrograman lainnya, jika Anda menulis dalam bahasa Python, indentasi –penempatan kalimat atau baris kode– sangat diperhatikan. Python menggunakan indentasi untuk mengindikasikan baris kode.

```
if 9 > 2:    print("Sembilan lebih besar daripada dua!")
```

Namun ketika baris kode dituliskan menjadi satu kolom atau dalam tab yang sama, maka program akan menjadi error. Di bawah ini adalah contoh penulisan yang menghasilkan error.

```
if 9 > 2:print("Sembilan lebih besar daripada dua!")
```

### 4. Variable

Python juga memiliki Variabel, tidak berbeda dengan bahasa pemrograman lainnya. Variabel ini digunakan untuk proses penyimpanan dan bekerja dengan berbagai tipe data.

Python sendiri punya standar pendeklarasian variabel. Variabel di Python dapat berupa nama singkat (seperti x dan y tadi) atau nama yang lebih mendeskripsikan seperti umur, nama, alamat, dan lain sebagainya. Aturan penamaan variabel di Python seperti:

- Variabel tidak bisa diawali dengan angka,
- Variabel harus diawali dengan huruf, atau karakter garis bawah (underscore),
- Variabel hanya bisa mengandung karakter alfa-numerik dan karakter garis bawah,
- Variabel di *Python case-sensitive*

Namun berbeda dengan bahasa pemrograman lainnya, Python tidak memerlukan inisiasi variabel untuk mendeklarasikan variabel. Ini berarti sebuah variabel terbuat ketika pertama kali Anda menambahkan nilai ke dalamnya.

Contohnya ketika Anda ingin membuat variabel ‘x’ dan ‘y’, Anda tinggal memasukkan nilainya langsung seperti di bawah ini:

```
x = 10y = "Budi"  
print(x)print(y)
```

Perintah di atas akan mengisi variabel ‘x’ dengan nilai ‘5’ dan ‘y’ dengan nilai ‘Budi’. Jadi proses penyusunan baris kode lebih ringkas.

Kemudahan lainnya, Anda tidak perlu mendefinisikan tipe variabel. Python secara otomatis akan memberikan tipe variabel sesuai dengan nilai yang diberikan pada variabel tersebut. Misalnya pada contoh di bawah ini:

```
x = 10 # x bertipe integery = "Budi" # y bertipe string
```

## 5. Booleans

Setelah mempelajari variabel bekerja, di bagian ini Anda akan belajar tentang Booleans. Jika Variabel dapat menyimpan bilangan dengan satu tipe data, booleans juga digunakan untuk menyimpan sebuah tipe data, tapi tipe data yang berbeda.

Tipe data di Booleans hanya ‘benar’ atau ‘salah’. Jadi ini mirip dengan saklar lampu, hanya memiliki dua nilai. Anda dapat menggunakan booleans seperti contoh di bawah ini:

```
a = Trueb = False
```

## 6. Number

Ketika Anda belajar Python number, ada tiga tipe numerik variabel di Python, yaitu int, float, dan complex. Anda mungkin tidak akan pernah menuliskan tipe variabel di setiap pendeklarasiannya, karena (seperti yang sudah dijelaskan di atas) Python sudah menginisiasi tipe variabel ketika Anda menambahkan nilai ke dalamnya.

Int, float, dan complex mempunyai range yang berbeda. Int atau bilangan integer adalah bilangan bulat positif atau negatif, tanpa desimal, dengan panjang tak terbatas. Float atau ‘angka floating point’ adalah angka, positif atau negatif, yang mengandung satu atau lebih desimal. Sedangkan complex adalah bilangan kompleks yang ditulis dengan “j” sebagai bagian dari imajiner.

Sebagai contoh, di bawah ini adalah tiga tipe numerik variabel yang berbeda:

```
x = 1 # inty = 2.8 # floatz = 1j # complex
```

Jadi Python akan mengenali dan membedakan tipe setiap variabel pada saat Anda mengisinya dengan sebuah nilai. Sedangkan untuk mengetahui tipe setiap variabel Anda dapat menggunakan fungsi `type()`:

```
print(type(x))print(type(y))print(type(z))
```

## 7. String

Ketika ingin belajar Python string, Anda hanya perlu menambahkan tanda kutip tunggal atau tanda kutip ganda di antara nilai variabel yang ingin ditambahkan. Misalnya saja ketika Anda ingin menambahkan string “Budi” ke dalam variabel x maka yang

perlu Anda lakukan adalah mendeklarasikannya seperti di bawah ini:

```
x = "Budi"# Atau x = 'Budi'  
# "Budi" sama artinya dengan 'Budi'.
```

Sama seperti bahasa pemrograman lainnya, string dalam Python adalah array byte yang mewakili karakter unicode. Namun, Python tidak memiliki tipe data karakter sehingga satu karakter hanyalah string dengan panjang 1. Ketika Anda ingin mengakses satu karakter di dalam string, yang perlu Anda gunakan adalah menggunakan tanda kurung kotak.

Di bawah ini adalah contoh mengambil karakter kedua pada sebuah string.

```
a = "Hello, World!"  
print(a[1])
```

## 8. Operator

Bagian terakhir dari pengenalan komponen Python adalah Operator. Selama melakukan proses coding Anda pasti akan membutuhkan operator untuk membuat sebuah alur logika, penghitungan angka, atau yang lainnya.

Operator ini bekerja untuk melakukan operasi pada variabel dan nilai. Dalam bahasa pemrograman Python, terdapat beberapa grup dari operator, seperti operator aritmatika, penugasan (assignment), pembanding (comparison), logika (logical), identitas (identity), keanggotaan (membership), dan bitwise.

Di antara operator lainnya, operator aritmatika sering digunakan. Operator aritmatika ini mengandung beberapa operator. Di bawah ini adalah daftar operator aritmatika secara lengkapnya:

Operator	Nama	Contoh
+	Penambahan	$x + y$
-	Pengurangan	$x - y$
*	Perkalian	$x * y$
/	Pembagian	$x / y$
%	Modulus	$x \% y$
**	Exponensian (Pangkat)	$x ** y$
//	Floor division	$x // y$

Selain itu, beberapa proses pengembangan juga sering membutuhkan operator yang dapat menetapkan suatu nilai ke dalam variabel. Berikut ini adalah daftar operator assignment yang dapat Anda gunakan di dalam Python:

Operator	Contoh	Persamaan
=	$x = 5$	$x = 5$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
*=	$x *= 3$	$x = x * 3$
/=	$x /= 3$	$x = x / 3$
%=	$x %= 3$	$x = x \% 3$
//=	$x //= 3$	$x = x // 3$
**=	$x **= 3$	$x = x ** 3$
&=	$x &= 3$	$x = x \& 3$
=	$x  = 3$	$x = x   3$
^=	$x ^= 3$	$x = x ^ 3$
>>=	$x >>= 3$	$x = x >> 3$
<<=	$x <<= 3$	$x = x << 3$

Sedangkan ketika Anda ingin membandingkan antara satu atau beberapa variabel, biasanya membutuhkan operator perbandingan

di antaranya. Operator ini terdiri dari enam jenis. Berikut ini adalah operator perbandingan yang digunakan untuk membandingkan dua nilai:

Operator	Nama	Contoh
<code>==</code>	Sama dengan	<code>x == y</code>
<code>!=</code>	Tidak sama dengan	<code>x != y</code>
<code>&gt;</code>	Lebih besar dari	<code>x &gt; y</code>
<code>&lt;</code>	Kurang dari	<code>x &lt; y</code>
<code>&gt;=</code>	Lebih besar atau sama dengan	<code>x &gt;= y</code>
<code>&lt;=</code>	Lebih kecil atau sama dengan	<code>x &lt;= y</code>

## 9. Python if else

Ketika Anda membutuhkan perbandingan antara kondisi satu dengan yang lain, Python dapat Anda gunakan untuk mendukung kondisi logis dari matematika. Aturan logika ini biasanya digunakan untuk memberikan syarat sebelum sebuah baris program diambil.

Sesuai dengan tabel perbandingan yang sudah dibahas pada bagian sebelumnya, ada enam kondisi logis yang dapat digunakan di Python; sama dengan (`a == b`), tidak sama dengan (`a != b`), kurang dari (`a < b`), kurang dari atau sama dengan (`a <= b`), lebih besar dari (`a > b`), lebih besar atau sama dengan (`a >= b`).

Kondisi ini dapat digunakan dengan beberapa modifikasi, lebih sering digunakan untuk “pernyataan If” dan perulangan. Ketika digunakan untuk “pernyataan If”, contohnya seperti di bawah ini:

```
a = 8b = 10if a > a:    print("b lebih besar dari a")
```

Selain penggunaan “If”, Anda dapat menggunakan “Elif dan Else”. Elif berarti “jika kondisi sebelumnya tidak benar, maka coba kondisi ini. Sedangkan Else menyatakan “lakukan perintah berikut jika semua kondisi tidak sesuai”.

Di bawah ini adalah contoh penggunaan If, Elif, dan Else dalam satu logika.

```
a = 8b = 10if a > a:    print("b lebih besar dari a")elif a == b:    print("a sama dengan b")Else:    print( lebih besar  
dari b")
```

### 1.1.5.8 Definisi Bahasa Pemrograman Ruby

Bahasa Ruby merupakan sebuah bahasa pemrograman yang dinamis, reflektif, general-purposed, dan berbasis objek. Bahasa ini dirancang dan dikembangkan oleh Yukihiro “Matz” Matsumoto di Jepang. Sebelum menentukan namanya, Yukihiro memiliki dua nama untuk digunakan bahasa ini yaitu Rubi dan Coral. Akan tetapi, Yukihiro lebih memilih menggunakan Ruby karena sama dengan batu kelahiran dari salah satu koleganya.

Meskipun mudah dipelajari, bahasa ini memiliki fungsi yang sangat luar biasa diantaranya yaitu dapat digunakan untuk membangun Desktop GUI (Graphic User Interfaces), membuat aplikasi web, atau mengembangkan web itu sendiri. Telah banyak platform yang dibangun menggunakan bahasa ini mulai dari Hulu, Groupon, Airbnb, dan bahkan Twitter. Hal ini tentunya disebabkan karena kelebihan bahasa pemrograman Ruby tersebut.

Seperti yang dijelaskan sebelumnya, salah satu kelebihan bahasa pemrograman Ruby adalah mudah untuk dipelajari. Bahkan, dalam infografis bahasa pemrograman ini, bahasa pemrograman Ruby dinyatakan sebagai

bahasa pemrograman awal terbaik ketika kamu pertama kali belajar pemrograman.

Oleh Yukihiro, bahasa pemrograman ini dikembangkan agar dapat digunakan secara natural dan mudah untuk dimengerti oleh penggunanya. Selain itu, telah terdapat berbagai macam situs yang telah membahas Ruby, yang dapat digunakan sebagai referensi saat mempelajari bahasa ini. Selain mudah untuk dipelajari, kelebihan bahasa pemrograman Ruby lainnya juga memiliki kesamaan dengan kelebihan yang dimiliki bahasa Phyton. Bahasa pemrograman Ruby juga menawarkan berbagai macam library perlengkapan yang menakjubkan serta fungsionalitas yang sangat luas.

Dengan kelebihan ini, para developer dapat menggunakan bahasa pemrograman Ruby dengan sangat mudah dalam membangun berbagai macam hal. Para developer tidak harus membuat segala sesuatunya dari awal, mereka dapat menggunakan beberapa library yang sudah ada sebelumnya dalam membangun sebuah kode.

### **1.1.5.9 Definisi Bahasa Pemrograman COBOL**

COBOL (Common Business Oriented Language) adalah suatu bahasa computer awam (High Level language) yang berorientasi langsung kepada permasalahan bisnis. Cobol diciptakan pada tahun 1959. pengembangan bahasa cobol selanjutnya dilakukan oleh suatu group yang disebut CODASYL, Singkatan dari Conference On Data System Langguage.

Bahasa Cobol pertama – kali diperkenalkan secara formal pada bulan Januari tahun 1960, versi dari bahasa COBOL ini disebut dengan COBOL - 60, dan diperbarui pada tahun 1965 untuk mengatasi hal ini, pada tahun 1968 dan 1974, bahasa cobol dikembangkan dan disempurnakan lebih lanjut dan distandarisikan dengan nama Ansi Cobol (American National Standards Institute).

Program bahasa COBOL Merupakan Program terstruktur, yaitu program yang strukturnya jelas, mudah dibaca, dan mudah dipelajari, dan baik untuk didokumentasikan , Struktur Utama dari suatu program Cobol terdiri dari 4 divisi yaitu:

- IDENTIFICATION DIVISION
- ENVIRONMENT DIVISION
- DATA DIVISION
- PROCEDURE DIVISION

Kalau diinginkan Informal mengenai Identitas program (nama Program, Si pembuat, tanggal dibuat, tanggal dikompilasi, dan lainnya) dapat dilihat pada *identification division* Informasi mengenai Keadaan Komputer dan alat – alat lain yang dipergunakan, dapat dilihat pada *environment division*, Informasi mengenai bentuk, jenis dari data apa saja yang dipergunakan dalam program bersangkutan dapat terlihat pada *data division*, *procedure division* memuat prosedur pemrosesan data yang datanya tampak pada data division untuk dihasilkan outputnya.

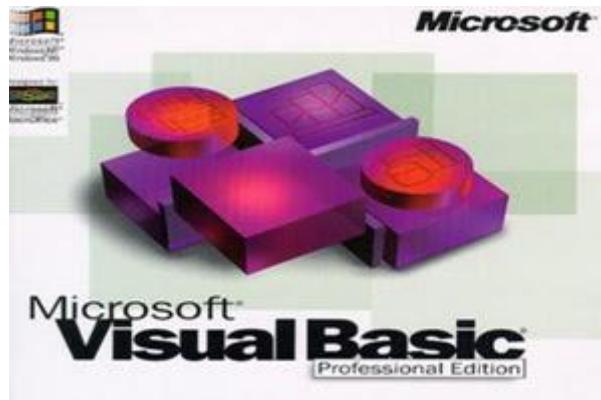
#### **1.1.5.9.1 Yang dapat dikerjakan COBOL**

- COBOL dibuat untuk operasi yang mencakup langkah dasar Pengolahan
- data, yaitu membaca data menghasilkan output informasi didalam program COBOL, dua bagian yang utama adalah data division dan Procedure division
- Dari apa yang dapat dikerjakan oleh cobol, konsep COBOL Orientasinya pada permasalahan yang berhubungan dengan pengolahan data.

### **1.1.5.9.2 Keuntungan Bahasa pemrograman cobol**

- Program COBOL dibuat dalam instruksi bahasa inggris, sehingga lebih mudah dipelajari dan dibuat .
- program COBOL sesuai untuk pengolahan data, yang banyak diterapkan pada permasalahan bisnis.
- program COBOL sifatnya standard, sehingga dapat dipergunakan pada computer – computer yang berbeda, tanpa banyak perbedaan.
- struktur program COBOL jelas, sehingga dapat dimengerti oleh orang seperti akuntan, auditor atau manajer – manajer yang hanya mempunyai pengetahuan pengolahan data yang sedikit.
- COBOL menyediakan fasilitas listing program , bila mana perlu dapat diperiksa oleh orang lain selain programmernya.
- mudah didokumentasikan dan dikembangkan bilamana perlu.

### **1.1.5.10 Definisi Bahasa Pemrograman Microsoft visual basic**



Microsoft Visual Basic adalah suatu paket Bahasa Pemrograman tingkat tinggi yang berbasis Under Windows dengan Orientasi Objek Programming (OOP). Maksudnya, program dapat aktif bila ada respon dari pemakai berupa event atau kejadian tertentu.

Setiap Bahasa Pemrograman yang berbasis objek mempunyai IDE, yaitu suatu tampilan Visual tempat bekerja atau membuat program dalam menyelesaikan suatu proyek.

Visual Basic dikembangkan oleh Microsoft sejak tahun 1991, merupakan pengembangan dari pendahulunya yaitu Bahasa Pemrograman Basic (Beginner of All Purpose Symbol) sebagai alat bantu untuk membuat berbagai macam program komputer, khususnya menggunakan sistem operasi Windows.

#### **1.1.5.10.1 Komponen *Microsoft Visual Basic***

Ketika program Visual Basic telah terbuka, maka akan ditemui banyak komponen yang terdapat di dalamnya, yaitu sebagai berikut :

##### **1. Control Menu**

Menu yang digunakan untuk memanipulasi jendela Bahasa Pemrograman, dari menu ini kita bisa mengubah ukuran-ukuran, memudahkan menutup jendela Bahasa Pemrograman.

##### **2. Toolbar**

Tombol-tombol yang mewakili suatu perintah tertentu.

##### **3. Toolbox**

Komponen-komponen yang akan digunakan dalam perancangan program, terdiri dari :

- a. Label, digunakan untuk menampilkan teks yang tidak dapat diedit oleh user.
- b. Text Box, digunakan untuk memasukan dan mengedit teks.
- c. Option button, merupakan bagian dari Optiongroup, digunakan untuk menampilkan beberapa pilhan dimana user hanya dapat memilih satu.

- d. Image, digunakan untuk menampilkan bitmap, Ikon,file JPG atau GIF.
- e. Chek box, digunakan untuk memilih satu pilhan atau lebih secara bersamaan yang disediakan pada program.
- f. ComboBox, digunakan untuk memilih satu pilhan yang disediakan pada program yang menampilkan seluruh pilhan yang tersedia saat objek tersebut diklik.
- g. Command Button, digunakan untuk membuat suatu aksi saat objek tersebut diklik.
- h. List box,digunakan untuk menampilkan dafatar item dimana user dapat memilih dirinya.
- i. Frame, digunakan sebagai wadah Control atau Toolbox yang lain.
- j. Timer, digunakan untuk mengeksekusi event timer dalam interval waktu tertentu.
- k. Data, digunakan untuk menghubungkan dengan sebuah database dan menampilkan informasinya pada form.

#### **4. Form Windows**

Daerah kerja utama pada saat membuat suatu program aplikasi akan otomatis tersedia.

#### **5. Properties Windows**

Jendela yang digunakan untuk melihat semua informasi objek yang terdapat pada aplikasi Bahasa Pemrograman Berbasis Objek. Properties adalah sifat dari sebuah objek.

#### **6. Project Explorer**

Jendela yang digunakan untuk melihat sebuah informasi tentang tampilan form atau tampilan jendela kode.

## **7. Code Windows**

Jendela tempat menulis perintah yang akan dilaksanakan jika suatu objek dijalankan, jendela berbasis kode-kode yang merupakan instruksi Bahasa Pemrograman Visual Basic.

### **1.1.5.11 Definisi Bahasa Pemrograman Node.JS**

Node.js adalah sebuah runtime environment opensource dan cross platform yang biasa digunakan untuk mengembangkan aplikasi web, walaupun node.js bukan merupakan framework javascript namun banyak modul dasarnya yang ditulis menggunakan javascript. Runtime environment yang dimiliki oleh node.js menterjemahkan javascript menggunakan google V8 javascript engine [6]. Node.js memiliki ciri khas yang sangat unik yaitu kemudahan dimana web server dan kode program nya tidak terpisah seperti pada bahasa pemrograman web lainnya. Pada node.js pengembang dapat mengaktifkan web server memodifikasinya dan menampilkan content dalam satu file tanpa harus mengubah banyak konfigurasi. Pada arsitektur MEAN node.js akan bertindak sebagai web server dan runtime environment yang menjalankan Express.js serta mengakses Mongo Db dan menampilkan ke html dengan Angular Js.

#### **1.1.5.11.1 Sejarah Node.JS**

Siapa sebenarnya pencipta Node.js pertama kali? Node.js pertama kali dibuat pada tahun 2009 oleh Ryan Dahl. Dengan diciptakannya, maka JavaScript bukan saja bisa digunakan di sisi client, tetapi juga bisa dipergunakan di sisi server sehingga berada satu kelas dengan ASP, PHP serta Ruby.

Perlu diketahui bahwa, Ryan Dahl merupakan seorang pengembang ternama dari Joyent, yaitu sebuah perusahaan yang mengembangkan perangkat lunak berbasis Cloud. Beliau sangat tertarik dengan penggunaan

pada bahasa pemrograman yang menggunakan single thread khususnya pada sisi server. Beliau memilih JavaScript untuk Node dimana sebelumnya sudah beliau coba di LUA, C, dan Haskell.

#### **1.1.5.11.2 Perbedaan JavaScript dan Node.JS**

Node.js sudah banyak dipakai oleh perusahaan-perusahaan besar di dunia. Adakah perbedaan antara JavaScript dan Node.js? Secara umum, bahasa\_ pemrograman yang digunakan pada JavaScript dan Node.js sama. Hanya saja, Node.js memiliki kumpulan API yang berbeda antara API yang satu dengan API yang lainnya. Dengan menggunakan Node.js, maka browser anda akan memiliki beberapa API DOM.WEB dimana semuanya terbuka sehingga memudahkan berinteraksi dengan UI dan mengakses hardware dengan batasan yang ada. Karena Node.js dilengkapi dengan banyak API, maka hal ini sangat cocok untuk mengembangkan banyak *back end* yang bisa didukung oleh permintaan https, proses anak, aus dan lain sebagainya. Sementara, browser memiliki berbagai dukungan dasar dimana masih bisa dibatasi demi alasan keamanan.

#### **1.1.5.11.3 Kelebihan Node.JS**

Node.js memiliki banyak keunggulan sehingga saat ini sudah banyak digunakan secara luas. Keunggulannya di antara lain adalah sebagai berikut:

##### **1. Mudah Di pelajari**

Pada tahun 2016 diadakan *user survey*. Dari hasil survei tersebut diketahui bahwa Node.js sangat populer digunakan oleh mereka yang mengembangkan *Front end*. Menurut para pengembang tersebut, mereka bisa dengan mudah memahami dan mengerti dengan bahasa

universal. Oleh karena itu, mereka bisa dengan mudah menerapkan Node.js Pada *back end*. Dalam laporannya, disebutkan bahwa programmer javaScript Junior pun bisa belajar dan bekerja dengan cepat.

## **2. Keberadaan Fullstack JS yang sangat memudahkan**

Sebelumnya, javaScript hanya digunakan pada sisi client. Dengan adanya Node.js fullstack, maka memudahkan para programmer untuk menulis pada *back and front end*. Komunitas yang sangat aktif

## **3. Komunitas yang sangat aktif**

Saat ini, banyak programmer yang tertarik dengan Node.js. Ada banyak komunitas-komunitas di luar sana yang akan memudahkan anda untuk mendapatkan solusi dari masalah-masalah yang mungkin timbul ketika anda sedang mengerjakan sebuah projek berbasis Node.js. Node.js adalah perangkat lunak yang bisa mengangani permintaan secara serentak. Seperti yang kita tahu bahwa, Node.js memiliki sistem terbaik yang disebut dengan IO non-blocking dimana dengan sistem ini, anda bisa memproses beberapa permintaan dalam waktu yang bersamaan. Dengan penanganan permintaan serentak seperti ini, maka Node.js memberi solusi lebih baik dari python maupun ruby sekalipun. Pada node.js permintaan akan diantrikan secara cepat sehingga penggunaan ram juga lebih sedikit.

### **1.1.5.11.3 Kekurangan Node.JS**

Meskipun telah kita ketahui kelebihan dari Node.js diatas, namun terdapat juga beberapa kekurangan yang harus anda tahu. Beberapa kekurangan dari Node.js di antaranya adalah sebagai berikut;

## **1. API Node.js tidak stabil**

Salah satu kekurangan utama ialah adanya API yang tidak stabil. Hal ini menjadi perhatian utama para programmer. API Node.js seringkali berubah dimana perubahan tidak sesuai bisa memberi masalah tersendiri bagi para programmer. Seringkali programmer harus membuat beberapa perubahan vital pada basis kode supaya bisa *compatible* dengan sistem terbaru yang ada.

## **2. Memerlukan waktu pengembangan yang lebih lama**

Banyak programmer yang mengeluh bahwa dengan menggunakan Node.js mereka memerlukan waktu lebih lama dalam mengembangkannya. Hal ini karena disebabkan banyak hal. Sebagai contoh, ketika anda menggunakan Ruby On Rain anda akan menemukan banyak petunjuk dalam hal melakukan sesuai. Sedangkan pada Node.js programmer harus menulis sesuatunya dari awal. Hal ini yang membuat sesuatunya lebih lama. Dalam hal ini, banyak programmer yang mensiasatinya dengan bekerjasama bersama programmer lain untuk membuat dan mengembangkan kode secara lebih cepat dan efektif.

## **3. Untuk *Heavy-Computing* Node.js tidak direkomendasikan**

Semua software pasti punya kekurangan dan kelebihan. Meski memiliki banyak kelebihan tetap saja ada hal-hal yang tidak cocok untuk project tertentu. Nah, Node.js ini ternyata tidak cocok untuk *heavy computing*. Hal ini disebabkan karena tidak bisa menghandle multi thread yang berat. Meski bisa melayani aplikasi yang rumit, namun Node.js akan memerlukan waktu yang sangat lama. Pekerjaan berat seperti perhitungan berlebihan, memblokir permintaan yang masuk bisa secara signifikan menurunkan kinerjanya. Namun demikian, Node.js bisa bekerja

dengan baik pada aplikasi kompleks, namun demikian kinerjanya akan melemah bila harus menghadapi komputasi berat.

#### **4. Tool yang bisa dibilang kurang matang**

Inti dari Node.js bisa dibilang stabil. Namun demikian, ada banyak paket registrasi NPM yang memiliki kualitas buruk. Paket NPM ini merupakan manajer paket yang mengatur pemasangan dan melakukan pengelolaan program dari pihak ketiga di luar Node.js. Perlu diketahui bahwa sebagian besar dari *ecosystem* Node.js adalah *open source*. Karena merupakan *ecosystem open source*, tentunya akan banyak tool yang tidak terawasi dengan baik. Dalam hal ini, maka kualitasnya tidak standard karena memang tidak ada yang melakuk pengawasan. Contoh kasus lainnya, ada beberapa tools yang sebenarnya tidak lolos *standard* pengkodean tetapi tetap saja dipakai dan dipergunakan dalam *ecosystem*. Anda bisa meminta bantuan programmer yang berpengalaman untuk menemukan *tool* yang handal.

##### **1.1.5.11.3 Cara Kerja Node.JS**

Cara kerjanya berbeda dengan cara kerja bahasa pemrogramman server lainnya yang bersifat melakukan *blocking*. Sementara itu, Node.js memiliki cara kerja yang bersifat *Non-blocking*. Dalam hal bekerja, memiliki kesamaan dengan JavaScript, yaitu dalam hal menghadapi *event* dimana mereka bekerja secara basis event baru selanjutnya dilakukan pengalihan ke kode pemrograman selanjutnya.

Sebuah Node.js bisa memiliki algoritma seperti di bawah ini:

1. Halaman blog akan meminta request ke server
2. Data diambil dari data base
3. Data blog ditulis secara HTML
4. Selanjutnya respon akan dikirim ke klien.

Bahasa pemrograman blocking memiliki sifat multi-thread berpengaruh besar urutan dalam hal pemrosesan dan permintaan data. Ketika program berjalan dan ketika pengambilan data dalam waktu tertentu maka threat berikutnya sudah harus disiapkan agar proses pengambilan data bisa berjalan dengan lancar.

Hal ini berbeda dengan yang dilakukan dengan system yang menggunakan Node.js. Pada Node.js, threat hanya akan dibuat bila ada even yang membutuhkannya saja. Hal ini tentunya akan membuat proses pengolahan data bisa dilakukan dengan lebih efektif dan lebih cepat.

Software yang satu ini memungkinkan programmer melakukan pemasangan JavaScript bukan saja di bagian *client* browser tetapi juga pada bagian server. Ini adalah terobosan penting dalam dunia programming. Penggunaan Node.js semakin meluas. Banyak perusahaan-perusahaan terkemuka yang menggunakan Node.js untuk men-support aplikasi yang dikembangkannya. Node.js adalah solusi bagi beberapa perusahaan untuk memberikan inovasi terbaru di dunia programming.

## **1.1.6 Macam Macam Framework JavaScript**

### **1.1.6.1 React.Js**



React.Js merupakan salah satu putaka (library) javascript yang dikembangkan dan juga dirilis oleh Facebook pada tahun 2013, dan menjadi pendukung Facebook.

React menggunakan DOM virtual yang membantu mengintegrasikan dengan aplikasi apa pun. React juga mengandalkan JSX dalam menyusun komponen dan membantu mengembangkan halaman web yang lebih ramah SEO dibandingkan dengan perpustakaan atau kerangka kerja Javascript lainnya.

### **1.1.6.1.1 Kelebihan dari React.js development**

#### **1. DOM Binding**

React dapat mengikat pengikatan di beberapa area kode untuk menghubungkan elemen DOM ke fungsi.

#### **2. Perpustakaan (library), bukan kerangka kerja (framework)**

Bereaksi adalah perpustakaan dan bukan Kerangka kerja Javascript. Ia menyediakan metode pengantar untuk mendefinisikan komponen UI dan dapat dikombinasikan dengan perpustakaan lain seperti React-Redux dengan mudah.

#### **3. Data dan presentasi**

Menyediakan pemisahan lengkap antara lapisan data dan presentasi, dan juga digunakan untuk penyimpanan berlangsung dalam waktu yang sangat singkat.

#### **4. Unidirection Data Flow**

yaitu menggunakan data yang mengikat, yang membuat kode lebih stabil dan yang perlu dilakukan pengembang untuk mengubah suatu objek adalah mengubah posisi dan menerapkan pembaruan.

#### **5. Komponen yang dapat digunakan kembali**

Meningkatkan efisiensi desain dan memudahkan developer untuk menangani upgrade, dan tidak akan mempengaruhi komponen lain, sehingga dapat menghemat banyak waktu selama pengembangan.

### **1.1.6.1.2 Kekurangan dari React.js development**

#### **1. JSX as a Barrier**

meskipun dengan JSX bisa menggabungkan ekstensi sintak antara HTML dan Javascript dan terlindungi, tapi juga kekurangannya yaitu memiliki kesulitan tingkat tinggi.

- perubahan baru yang cepat dimana para developer harus menguasai perubahan itu.

### 1.1.6.2. Vue.JS



Vue adalah salah satu framework javascript terbaik yang termasuk progresif dan mudah beradaptasi dan cepat mengintegrasikan kode ke dalam aplikasi melalui tag <script>. Kemampuan framework ini menjadi alasan para developer menggunakannya. Vue dibuat oleh Evan You dan awal rilis pada tahun 2014, dan menjadi proyek *front-end* dari GitHub sehingga pertumbuhan yang cepat menjadikan Vue.js populer di tahun 2018.

#### 1.1.6.2.1 Kelebihan dari Vue.js development

- Kemampuan integrasi yang dimiliki vue.js sangat baik untuk membuat program singel-page, dan aplikasi web yang kompleks. Vue juga sudah terintegrasi dengan framework dan library lain seperti Laravel, Django, WordPress dan linnya.
- Serbaguna dan Skalabilitas, dimana vue.js bisa digunakan sebagai framework atau library.

- 3.** Para developer yang menggunakan vue mudah beradaptasi dengan framework lainnya.
- 4.** Mudah dibaca dengan fungsinya yang mudah diakses.
- 5.** Hasil produk memiliki ukuran yang sangat ringan sekitar 18 Kb sesudah di *zipping*.

#### **1.1.6.2.2 Kekurangan dari Vue.js development**

1. Sumber daya yang kurang karena framework ini relatif baru sehingga sumber daya yang tersedia masih kurang.
2. Dokumentasi menjadi sumber daya bagi para developer.
- 3.

#### **1.1.6.3 Angular.JS**



Angular adalah salah satu framework javascript terbaik yang dikembangkan dan juga dikelola oleh [Google](#) dan termasuk framework yang *open source*. Google menjadwalkan waktu rilis angular setiap 6 bulan dengan fitur-fitur baru, serta peningkatan kinerja dari angular.

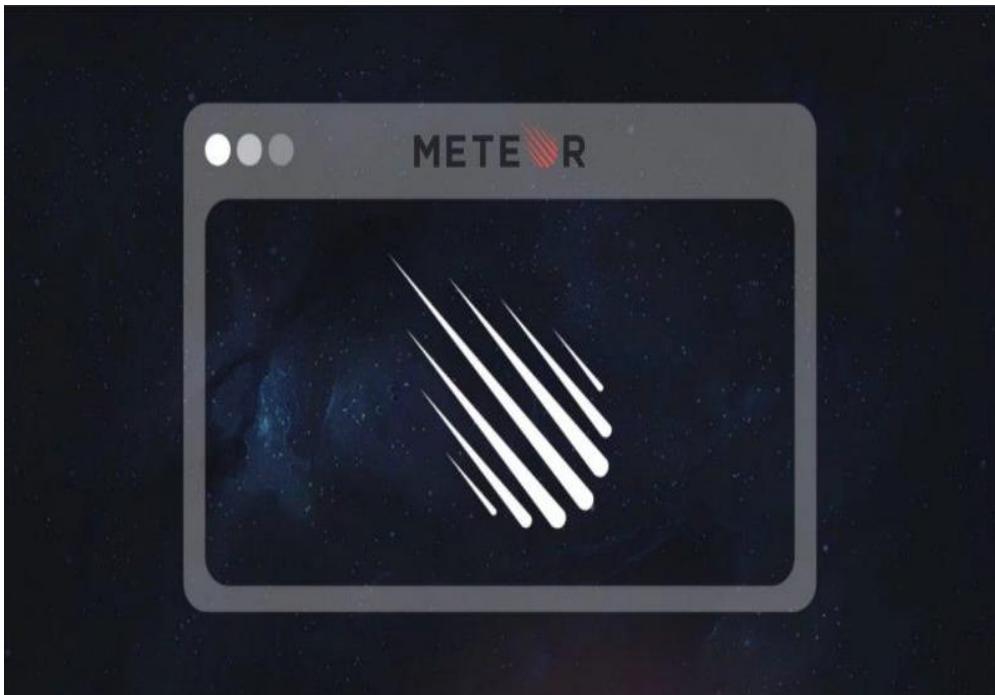
### **1.1.6.3.1 Kelebihan dari Angular.js**

1. Dokumentasi yang rinci, sehingga memudahkan para developer baru.
2. Progressive Web Application yang simpel, karena Angular sudah mengintegrasikan kemampuan dalam framework.
3. Optimal dalam Pengembangannya, yaitu mampu menghapus kode *untime* yang tidak dibutuhkan.
4. *Universal State Transfer API and DOM Support*, yang membantu membagikan kode antar versi palikasi server dan klien, serta embantu persepsi kinerj aplikasi meningkat.
5. Router hooks, perubahan router bisa dilacak dari pertama hingga elesai.
6. Penjilidan data dengan two-way yang memperkecil kesalaha, serta model view view model yang bekerja terpisah pada bagian aplikasi an set data yang serupa.

### **1.1.6.3.2 Kekurangan dari Angular.js**

1. Sekumpulan sintak yang rumitsama seperti Angular.Js versi pertama.
2. Terjadi masalah ketika memindahkan aplikasi Angular versi sebelumnya, terlebih lagi untuk aplikasi dengan sekala besar.

#### 1.1.6.4 Meteor.JS



Meteor adalah Framework javascript sebagai platform yang ditulis dengan Node.Js digunakan untuk membuat aplikasi web real-time, dan aplikasi mobile yang bersifat *open source*.

Meteor juga mencakup area yang signifikan seperti: *back-end development, management of database, business logic, dan rendering front-end*. Dimana dengan framework Meteor.js kita bisa membuat aplikasi prototipe yang cepat dan juga menghasilkan kode cross-platform (iOS, Android, Web).

Meteor.js juga sudah terintegrasikan dengan MongoDB dengan menggunakan *Distributed Data Protocol* dan menerbitkan pola untuk perubahan data secara otomatis ke klien tanpa mengharuskan pengembang menulis kode sinkronisasi.

#### **1.1.6.4.1 Fitur-fitur dari Meteor.js**

1. Web dan Mobile, yaitu platform untuk mengembangkan aplikasi pada Android, iOS, dan Web
2. Universal Apps, memiliki kode yang sama pada perangkat mobile (android dan iOS) dan web browser.
3. Packages, memiliki banyak paket dalam jumlah besar dan mudah untuk pemasangan dan penggunaannya.
4. Meteor Galaxy, sebuah layanan cloud untuk publikasi meteor app.

#### **1.1.6.4.2 Kelebihan dari Meteor.js**

1. Dengan Meteor Js pada developer hanya perlu javascript untuk server dan klien untuk sisi pengembangannya.
2. Mudah digunakan bagi pemula karena memiliki pengkodean yang sederhana dan ramah.
3. Meteor apps sebagai real-time dan reaktif yang default
4. memiliki paket-paket resmi atau pun komunitas.

### 1.1.6.5 Ember.JS



Ember.Js adalah *full-stack* framework javascript yang bersifat *open source*, banyak digunakan dan digemari oleh para developer salah satunya yaitu *single page application*. Fungsi lain dari Ember.Js yaitu desai front-end, menghilangkan boilerplate yang menyediakan arsitektur aplikasi standar, inti model pembangunan pada HTML dan CSS, Fitur route untuk mengelola sebuah URL.

Ember.Js dibangun pada bulan Desember 2011 yang sebelumnya bernama SproutSore 2.0. Ember.Js juga merupakan salah satu framework javascript terbaik yang dibuat oleh Yehuda Katz dan lainnya, dan sekarang Katz menjadi tim pengembang dari Ember.Js besama dengan Tom Dele dkk.

### **1.1.6.5.1 Kelebihan Ember.Js**

- 1.** Open source, semua orang bebas menggunakan.
- 2.** Framework yang fleksible, dengan konsep halaman web yang cepat, dimana mempercepat kinerja aplikasi tanpa harus merelog seluruh halaman.
- 3.** Perangkat lunak Ember.Js memiliki ukuran yang cukup kecil dibandingkan dengan yang lain.
- 4.** Otomatis menentukan route dan controller.
- 5.** Mempunyai dukungan yang luas pada jenis tampilannya.
- 6.** Memakai tamplate, yang membantu dalam memperbarui model secara otomatis.

### **1.1.6.5.2 Kekurangan Ember.**

- 1.** Kustomisasi yang terbatas, dan tidak menyediakan redux.
- 2.** Kurva pembelajaran ke atas, sehingga kerangka JS membutuhkan investasi waktu dan usaha yang lebih.

### **1.1.6.6 Mithril.JS**



Mithril adalah salah satu framework javascript terbaik yang menggunakan pola MVC klasik memiliki ukuran yang kecil sekitar 7 Kb dan juga cepat. Arsitekturnya mirip dengan Angular.Js, menggunakan DOM virtual seperti ReactJs, dan kebutuhan akan library seperti Jquery. Meskipun ukurannya kecil Mithril.Js memiliki API yang membuatnya menjadi ideal untuk widget javascript dan GUI dengan kinerja tinggi.

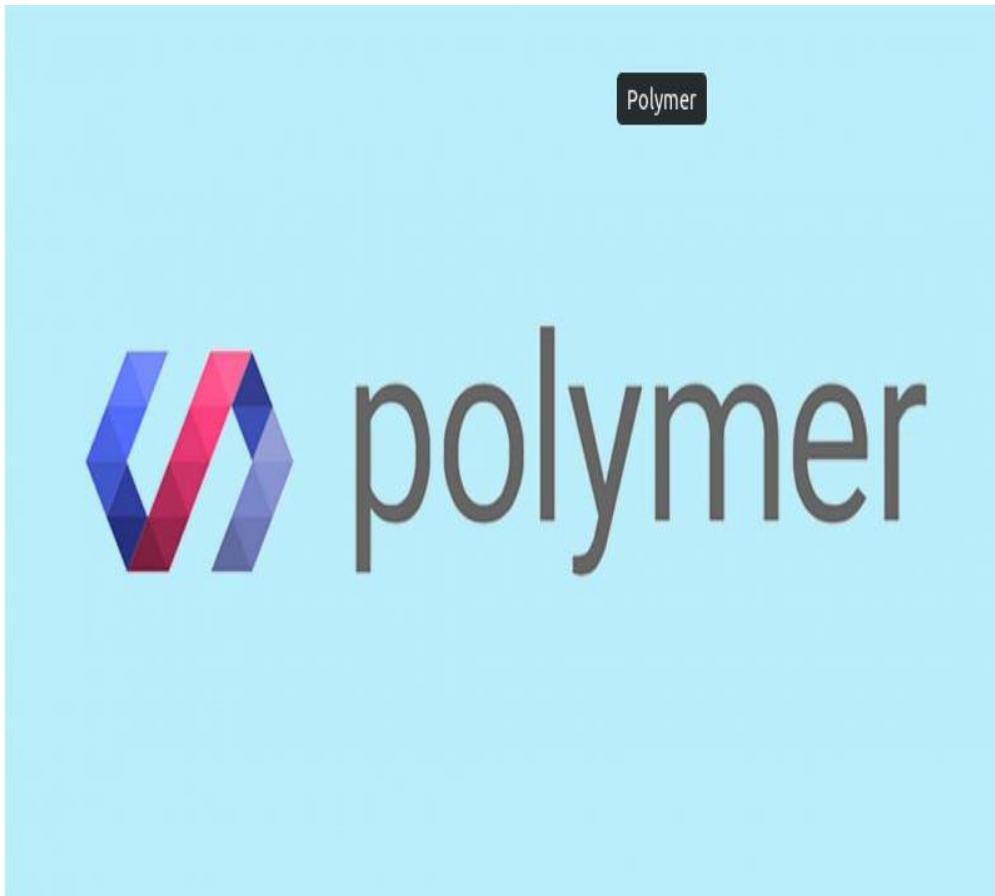
### **1.1.6.7 Backbone.JS**



Backbone adalah framework javascript yang populer dikalangan para developer karena mudah untuk dipahami dan dipelajari dan menjadi salah satu yang terbaik. Backbone.js dapat digunakan untuk membuat *single page application* dan lainnya.

Backbone.js juga berfungsi sebagai memiliki sisi server yang menggunakan API yang membantu dalam fungsional yang kompleks dengan sedikit kode.

### **1.1.6.8 Polymer**



Polymer adalah library javascript yang open-source yang dikembangkan oleh Google, fungsi dari polymer yaitu mampu membuat elemen-elemen pada halaman web tanpa tingkatan yang kompleks.

Polymer juga mendukung data binding dengan one-way atau two-way, serta bisa membuat aplikasi yang lebih luas.

### 1.1.6.9 Aurelia



Aurelia adalah framework javascript yang digunakan untuk mengimplementasikan antarmuka (GUI) apapun, sebagai framework dengan situs web yang jauh lebih kuat, memperluas HTML untuk berbagai keperluan, data binding, arsitektur yang modern, serta tujuan utama Aurelia yaitu untuk menginterpretasikan sisi server dan klien secara bersamaan.

## **BAB II PENGENALAN PROSES DAN ARSITEKTUR**

---

### **2.1 PENGENALAN DOCKER, REDISH, MICROSERVICES, dan REST**

#### **2.1.1 Pengertian DOCKER**

Docker adalah salah satu platform yang dibangun berdasarkan teknologi container. Docker merupakan sebuah project open source yang menyediakan platform terbuka untuk developer maupun sysadmin untuk dapat membangun, mengemas, dan menjalankan aplikasi dimanapun sebagai sebuah wadah (container) yang ringan. Dengan sangat populernya docker, sebagian orang sering menganggap docker adalah sebutan lain untuk container. Docker pertama kali dikembangkan oleh Solomon Hykes sebagai project internal di dotCloud bersama dengan beberapa koleganya seperti Andrea Luzzardi dan Francois-Xavier Bourlet. Perilisan platform ini secara open-source dilakukan pada mei 2013 silam. Docker terus berkembang hingga memiliki ribuan orang yang berkontribusi membuatnya menjadi lebih baik. Berbeda dengan virtualisasi yang mana aplikasi berjalan di atas hypervisor dan guest OS, docker dapat menjalankan aplikasi langsung tanpa kedua hal tadi. Docker juga dilengkapi dengan fitur sandbox yang menjamin pengembang dan sysadmin tidak terganggu. Sandbox pada istilah keamanan komputer adalah mekanisme pemisahan aplikasi atau program tanpa mengganggu host (isolasi). Bagi pengembang, sandbox Menjamin aplikasinya dapat berjalan tanpa ada gangguan atas perubahan lingkungan host. Sedangkan bagi sysadmin, menjamin host server yang dikelola tidak terganggu dan dapat melakukan update tanpa takut mengganggu aplikasi.

Dikutip dari situs resmi docker, pengembang dapat mengefektifkan waktu mereka dengan menghilangkan proses konfigurasi yang cocok dengan programnya. Selain itu, berkat fitur sandbox, pengembang leluasa untuk

berkreasi tanpa takut merusak programnya. Terakhir docker menjamin program yang kita buat, akan selamanya berjalan seperti seharusnya. Pemaketan aplikasi dan seluruh kebutuhannya, memastikan aplikasi berjalan lancar pada kondisi lingkungan apapun.

### 2.1.1.1 Cara kerja Docker

1. Cek docker versi

```
docker version
```

2. C

ek

apakah docker sudah berjalan dengan benar.

```
docker run alo
```

Docker akan mencari image image alo di mesin lokal, jika tidak ada, maka Docker akan cek dari sumber online yang defaultnya adalah docker hub.

3. Lihat image yang sudah di download

```
docker image rm [nama/kode image]
```

4. Menghapus image

```
docker container ls -all
```

5. Cek semua Container yang dijalankan di mesin

```
docker build
```

6. build image

Dalam proses build, docker mengikuti kumpulan perintah yang terdapat di file Dockerfile. Dokumentasinya sendiri bisa dilihat di:  
<https://docs.docker.com/engine/reference/builder/>

7. build sekaligus beri tag pada image

```
docker build -t [url registry]
```

## 8. upload image

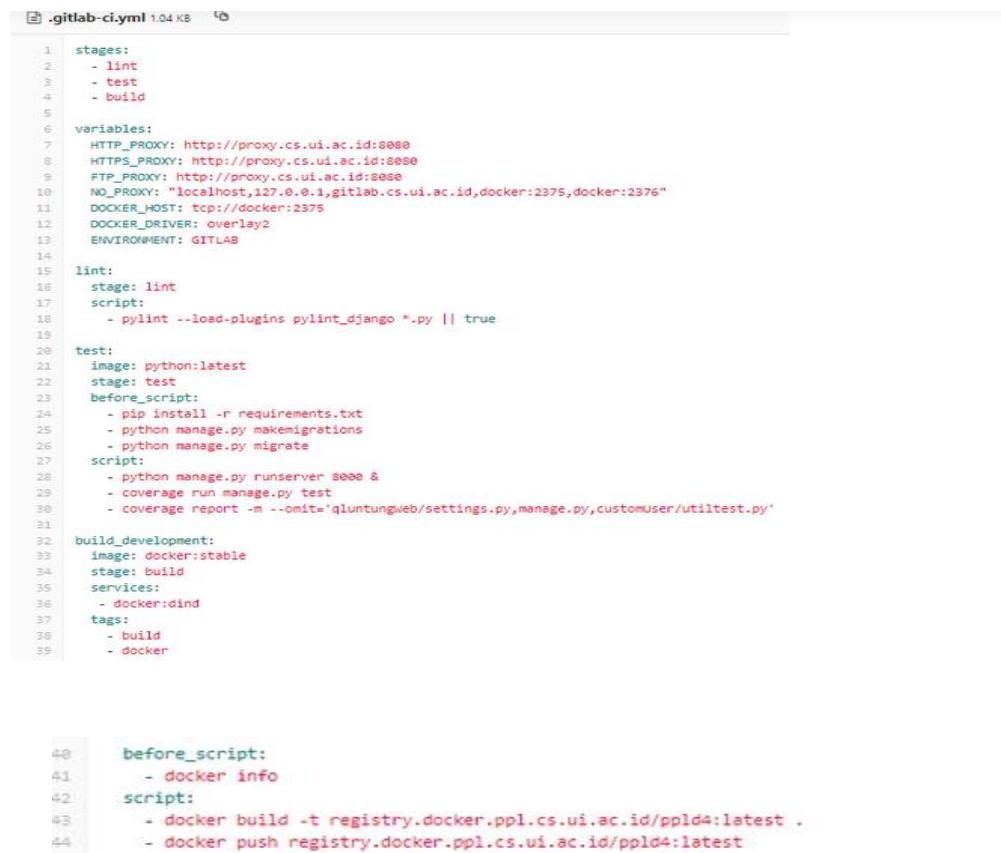


Pada proyek PPL ini, kami menggunakan portainer yang user-friendly secara tampilan. Pada portainer ini, kami bisa

Untuk proyek perangkat lunak ini, kami disediakan sebuah platform untuk melakukan *containerization* yaitu portainer yang mana tampilannya bukan cmd based dan user-friendly. di portainer bisa pull image dari suatu registry, lalu membuat suatu container serta menjalankan container tersebut sekaligus pause, stop, kill, restart, resume, dan remove container. Kita juga dapat melihat pengaturan dan atur sesuai keinginan kita.

Name	State	Quick actions	Stack	Image	Created	IP Address	Published Ports	Owner
backendQuntung	running	Start Stop Kill Restart Pause Resume Remove Add container		registry.docker.ppl.cs.ui.ac.id/pplq4/latest	2019-03-13 17:44:39	172.17.0.19	24402:8000	kemas.khaidar

Nah, setelah teman-teman developer tahu perintah-perintah apa saja yang digunakan untuk menjalankan docker, serta tahu *interface* dari portainer untuk docker, kita akan mempelajari bagaimana aplikasi kita yang ada pada *version control* untuk dapat terdeploy pada docker. Gambar dibawah akan menjelaskan secara keseluruhan mengenai bagaimana aplikasi kita terdeploy:



```
1 stages:
2   - lint
3   - test
4   - build
5
6 variables:
7   HTTP_PROXY: http://proxy.cs.ui.ac.id:8080
8   HTTPS_PROXY: https://proxy.cs.ui.ac.id:8080
9   FTP_PROXY: http://proxy.cs.ui.ac.id:8080
10 NO_PROXY: "localhost,127.0.0.1,gitlab.cs.ui.ac.id,docker:2375,docker:2376"
11 DOCKER_HOST: tcp://docker:2375
12 DOCKER_DRIVER: overlay2
13 ENVIRONMENT: GITLAB
14
15 lint:
16   stage: lint
17   script:
18     - pylint --load-plugins pylint_django *.py || true
19
20 test:
21   image: python:latest
22   stage: test
23   before_script:
24     - pip install -r requirements.txt
25     - python manage.py makemigrations
26     - python manage.py migrate
27   script:
28     - python manage.py runserver 8000 &
29     - coverage run manage.py test
30     - coverage report -m --omit='qluntungweb/settings.py,manage.py,customUser/utiltest.py'
31
32 build_development:
33   image: docker:stable
34   stage: build
35   services:
36     - docker:dind
37   tags:
38     - build
39     - docker
40
41   before_script:
42     - docker info
43   script:
44     - docker build -t registry.docker.ppl.cs.ui.ac.id/ppld4:latest .
45     - docker push registry.docker.ppl.cs.ui.ac.id/ppld4:latest
```

Gambar diatas merupakan CI dari *version control* (Gitlab). Konten pada gambar diatas dapat dibagi menjadi 5 bagian:

### 1. Stages

```
stages:
  - lint
  - test
  - build

Stages
```

Stages merupakan step-step yang akan dijalankan oleh gitlab

### 2. Variables

```
variables:
  HTTP_PROXY: http://proxy.cs.ui.ac.id:8080
  HTTPS_PROXY: https://proxy.cs.ui.ac.id:8080
  FTP_PROXY: http://proxy.cs.ui.ac.id:8080
  NO_PROXY: "localhost,127.0.0.1,gitlab.cs.ui.ac.id,docker:2375,docker:2376"
  DOCKER_HOST: tcp://docker:2375
  DOCKER_DRIVER: overlay2
  ENVIRONMENT: GITLAB

Variables
```

Variables merupakan variable-variable yang berpengaruh untuk gitlab

### 3. Lint

```
lint:
  stage: lint
  script:
    - pylint --load-plugins pylint_django *.py || true

lint
```

Lint merupakan proses implementasi *Clean Code*

## 4. Test

```
test:  
  image: python:latest  
  stage: test  
  before_script:  
    - pip install -r requirements.txt  
    - python manage.py makemigrations  
    - python manage.py migrate  
  script:  
    - python manage.py runserver 8000 &  
    - coverage run manage.py test  
    - coverage report -m --omit='qluntungweb/settings.py,manage.py,customUser/utiltest.py'
```

Test

*Test merupakan test implementation yang akan dijalankan oleh gitlab*

## 5. Build Development

```
build_development:  
  image: docker:stable  
  stage: build  
  services:  
    - docker:dind  
  tags:  
    - build  
    - docker  
  before_script:  
    - docker info  
  script:  
    - docker build -t registry.docker.ppl.cs.ui.ac.id/ppld4:latest .  
    - docker push registry.docker.ppl.cs.ui.ac.id/ppld4:latest
```

Build Development

Build development merupakan yang menjadi penghubung antara gitlab dan docker. Pada bagian ini terdapat before script dan script.

*Pada bagian Before Script, dapat dilihat bahwa terdapat perintah “docker info”. Docker info merupakan perintah untuk melakukan cek apakah image sebelumnya sudah ter-pull atau belum.*

Pada bagian Script terdapat 2 perintah, yaitu

1. docker build -t registry.docker.ppl.cs.ui.ac.id/ppld4:latest

perintah build -t merupakan perintah untuk membuat image dan memberi tag kepada registry.docker.ppl.cs.ui.ac.id/ppld4:latest

2. docker push registry.docker.ppl.cs.ui.ac.id/ppld4:latest

docker push kurang lebih fungsinya sama seperti git push. ketika kita melakukan perintah docker push, maka image yang sebelumnya telah kita build dan beri tag, akan kita push ke registry.docker.ppl.cs.ui.ac.id/ppld4. Dengan melakukan push, maka admin docker dapat melakukan pull dan melakukan update dan dapat deploy aplikasi dari kita.

## 2.1.2 Pengertian REDISH

Redis adalah salah satu *database* dari dunia NoSQL yang berbasis *key-value store*. Sistemnya yang *in-memory* membuat pengambilan data dari Redis menjadi lebih cepat, namun dapat juga *persistent* bila ingin menyimpan data kita ke *disk*. Redis memiliki sejumlah *query* yang pastinya mudah digunakan untuk menyimpan mulai dari data sederhana hingga data kompleks. Selain itu dokumentasinya yang lengkap membuat kamu dapat menguasai Redis tanpa harus banyak *googling*.

Kali ini kita akan melakukan kencan kilat bersama Redis, yang siapa tahu dapat bermanfaat untuk aplikasi *web* yang kamu kembangkan.

### 2.1.2.1 Instalasi

```
$ wget http://download.redis.io/releases/redis-2.8.19.tar.gz  
$ tar xzf redis-2.8.19.tar.gz  
$ cd redis-2.8.19  
$ make
```

Instalasi Redis sendiri cukup mudah, kamu dapat mengunduh paket instalasi Redis dari <https://redis.io/download>. Kemudian tinggal ekstrak dan set *path* ke direktori Redis di .bashrc atau *environment variable*. Kamu dapat melihatnya pada langkah - langkah berikut:

Untuk menyalakan Redis, kamu dapat menjalankan perintah **redis-server** di terminal. Sehingga dapat melihat *output* seperti berikut:

```
$ redis-cli  
127.0.0.1:6379> PING  
PONG  
127.0.0.1:6379>
```

Untuk melakukan operasi terhadap Redis melalui terminal, kamu dapat menggunakan **redis-cli**. Sehingga dapat dilihat *output* seperti berikut:

### **2.1.2.2 Melakukan Operasi terhadap Key**

Di Redis, kamu dapat menyimpan langsung bagaimana suatu *key* menyimpan informasi tertentu. Misal kamu dapat menyimpan informasi seperti berikut:

```
127.0.0.1:6379> set user:12345 ridwanbejo
OK
127.0.0.1:6379> get user:12345
"ridwanbejo"
```

Kamu juga dapat mengatur berapa lama key tersebut dapat hidup di dalam Redis. Kamu dapat menggunakan SETEX untuk menambahkan ekspirasi suatu key, dan memeriksanya dengan menggunakan TTL:

```
127.0.0.1:6379> SETEX cart:123 5 "{\"nama\":\"shampoo jwitsal\", \"amount\":\"10\"}"
OK
127.0.0.1:6379> GET cart:123
"[{"nama":"shampoo jwitsal", "amount":"10"}]"
127.0.0.1:6379> GET cart:123
"[{"nama":"shampoo jwitsal", "amount":"10"}]"
127.0.0.1:6379> TTL cart:123
(integer) 1
127.0.0.1:6379> TTL cart:123
(integer) -2
127.0.0.1:6379>
```

Yang unik lainnya adalah, kamu dapat melakukan operasi *increment* dan *decrement* terhadap suatu *key*:

```
127.0.0.1:6379> SET visitor:home 1
OK
127.0.0.1:6379> INCR visitor:home
(integer) 2
127.0.0.1:6379> INCR visitor:home
(integer) 3
127.0.0.1:6379> INCR visitor:home
(integer) 4
127.0.0.1:6379> INCR visitor:home
(integer) 5
127.0.0.1:6379> INCR visitor:home
(integer) 6
127.0.0.1:6379> INCR visitor:home
(integer) 7
127.0.0.1:6379> INCR visitor:home
(integer) 8
127.0.0.1:6379> GET visitor:home
"8"
127.0.0.1:6379> DECR visitor:home
(integer) 7
127.0.0.1:6379> DECR visitor:home
(integer) 6
127.0.0.1:6379> DECR visitor:home
(integer) 5
```

### 2.1.2.3 Melakukan Operasi terhadap Hash

*Hash* adalah suatu tipe data di Redis yang dapat menyimpan banyak *key* di dalam suatu *key*. Cocok bila kamu ingin menyimpan suatu data *user* yang memiliki beberapa atribut. Berikut contoh operasi *hash* menggunakan perintah HSET, HGET, dan HGETALL:

```
127.0.0.1:6379> HSET user:123 name "arslan"
(integer) 1
127.0.0.1:6379> HSET user:123 email "arslan@gmail.com"
(integer) 1
127.0.0.1:6379> HSET user:123 dob "1990-09-09"
(integer) 1
127.0.0.1:6379> HGET user:123
(error) ERR wrong number of arguments for 'hget' command
127.0.0.1:6379> HGETALL user:123
1) "name"
2) "arslan"
3) "email"
4) "arslan@gmail.com"
5) "dob"
6) "1990-09-09"
127.0.0.1:6379> HGET user:123 email
"arslan@gmail.com"
127.0.0.1:6379> HGET user:123 name
"arslan"
127.0.0.1:6379> HGET user:123 dob
"1990-09-09"
127.0.0.1:6379>
```

Selain itu kita dapat mengambil *key*-nya saja atau *value*-nya saja:

```
127.0.0.1:6379> HKEYS user:123
1) "name"
2) "email"
3) "dob"
127.0.0.1:6379> HVALS user:123
1) "arslan"
2) "arslan@gmail.com"
3) "1990-09-09"
127.0.0.1:6379>
```

Bila ingin mengetahui jumlah *key* yang ada di dalam suatu *hash* kamu dapat memeriksanya dengan HEXIST dan HSTRLEN:

```
127.0.0.1:6379> HDEL user:123 dob
(integer) 1
127.0.0.1:6379> HEXISTS user:123 dob
(integer) 0
127.0.0.1:6379>
```

Kamu juga dapat *menghapus* key yang tidak diperlukan dari suatu *hash* dengan menggunakan HDEL

#### 2.1.2.4 Melakukan Operasi terhadap List

Sekarang kita akan bermain dengan List yang merupakan sebuah tipe data yang menyimpan sejumlah data di dalam suatu *key*. Kamu dapat memulainya

```
127.0.0.1:6379> LPUSH tim_bola "Persib Bandung FC"
(integer) 1
127.0.0.1:6379> LPUSH tim_bola "Persipura Jayapura FC"
(integer) 2
127.0.0.1:6379> LPUSH tim_bola "Arema Cronus FC"
(integer) 3
127.0.0.1:6379> LPUSH tim_bola "Madura United FC"
(integer) 4
127.0.0.1:6379> LRANGE tim_bola 0 -1
1) "Madura United FC"
2) "Arema Cronus FC"
3) "Persipura Jayapura FC"
4) "Persib Bandung FC"
127.0.0.1:6379> LPUSH tim_bola "Mitra Kukar FC"
(integer) 5
127.0.0.1:6379> LPUSH tim_bola "Semen Padang FC"
(integer) 6
127.0.0.1:6379> RPUSH tim_bola "Persija Jakarta FC"
(integer) 7
127.0.0.1:6379> RPUSH tim_bola "PSM Makassar FC"
(integer) 8
127.0.0.1:6379> LRANGE tim_bola 0 -1
1) "Semen Padang FC"
2) "Mitra Kukar FC"
```

dengan menggunakan LRANGE, RPUSH dan LPUSH:

Dapat kita lihat bahwa LPUSH akan mengisi *list* dari sebelah "kiri", sedangkan RPUSH akan mengisi *list* dari sebelah kanan. Untuk menampilkannya, kamu dapat menggunakan LRANGE dengan melewatkkan indeks awal yang akan ditampilkan sampai indeks ke berapa. Sekarang mari kita coba periksa panjang *list* yang telah kita buat dan coba akses beberapa *value* yang ada di dalam *list*:

```
127.0.0.1:6379> LLEN tim_bola
(integer) 8
127.0.0.1:6379> LINDEX tim_bola 1
"Mitra Kukar FC"
127.0.0.1:6379> LINDEX tim_bola 2
"Madura United FC"
127.0.0.1:6379> LINDEX tim_bola 4
"Persipura Jayapura FC"
127.0.0.1:6379> LINDEX tim_bola 5
"Persib Bandung FC"
127.0.0.1:6379> LINDEX tim_bola 10
(nil)
127.0.0.1:6379>
```

Untuk mengganti suatu nilai pada suatu indeks di dalam *list*, kamu dapat menggunakan LSET:

```
127.0.0.1:6379> LSET tim_bola 1 "Mitra Kutai Kartanegara FC"
OK
127.0.0.1:6379> LRANGE tim_bola 0 -1
1) "Semen Padang FC"
2) "Mitra Kutai Kartanegara FC"
3) "Madura United FC"
4) "Arema Cronus FC"
5) "Persipura Jayapura FC"
6) "Persib Bandung FC"
7) "Persija Jakarta FC"
8) "PSM Makassar FC"
127.0.0.1:6379>
```

```
127.0.0.1:6379> LPOP tim_bola
"Semen Padang FC"
127.0.0.1:6379> LRANGE tim_bola 0 -1
1) "Mitra Kutai Kartanegara FC"
2) "Madura United FC"
3) "Arema Cronus FC"
4) "Persipura Jayapura FC"
5) "Persib Bandung FC"
6) "Persija Jakarta FC"
7) "PSM Makassar FC"
127.0.0.1:6379> RPOP tim_bola
"PSM Makassar FC"
127.0.0.1:6379> LRANGE tim_bola 0 -1
1) "Mitra Kutai Kartanegara FC"
2) "Madura United FC"
3) "Arema Cronus FC"
4) "Persipura Jayapura FC"
5) "Persib Bandung FC"
6) "Persija Jakarta FC"
127.0.0.1:6379> LREM tim_bola 3 "Arema Cronus FC"
(integer) 1
127.0.0.1:6379> LRANGE tim_bola 0 -1
1) "Mitra Kutai Kartanegara FC"
2) "Madura United FC"
```

Untuk menghapus suatu nilai di dalam *list*, kamu dapat menggunakan LPOP, RPOP, atau LREM:

### 2.1.2.5 Melakukan Operasi terhadap Set

```
127.0.0.1:6379> SADD myset Smith
(integer) 1
127.0.0.1:6379> SADD myset Paul
(integer) 1
127.0.0.1:6379> SADD myset George
(integer) 1
127.0.0.1:6379> SADD myset George
(integer) 0
127.0.0.1:6379> SADD myset George
(integer) 0
127.0.0.1:6379> SMEMBERS myset
1) "George"
2) "Paul"
3) "Smith"
```

Yang terakhir, kita akan bermain dengan Set. Dimana Set adalah sebuah tipe data yang menyimpan sejumlah data di dalam suatu *key* namun tidak memiliki urutan dan hanya dapat menyimpan satu nilai unik. Sekarang mari kita coba membuat sebuah *set* di dalam Redis dengan menggunakan perintah SADD:

Kamu dapat memeriksa suatu nilai apakah anggota suatu *set* atau bukan dengan perintah SISMEMBER:

```
127.0.0.1:6379> SISMEMBER myset George
```

```
(integer) 1
```

```
127.0.0.1:6379> SISMEMBER myset Alan
```

```
(integer) 0
```

```
127.0.0.1:6379>
```

*Untuk menghapus suatu nilai pada set dapat menggunakan SREM:*

```
127.0.0.1:6379> SREM myset Paul
```

```
(integer) 1
```

```
127.0.0.1:6379> SMEMBERS myset
```

```
1) "George"
```

```
2) "Smith"
```

Kamu juga dapat melakukan operasi himpunan seperti SUNION, SINTER, dan SDIFF:

```
127.0.0.1:6379> SMEMBERS myset
1) "George"
2) "Smith"
3) "Christoper"
127.0.0.1:6379> SMEMBERS employee
1) "George"
2) "Alan"
3) "Timothy"
4) "Michael"
5) "Paul"
6) "Smith"
127.0.0.1:6379> SDIFF myset employee
1) "Christoper"
127.0.0.1:6379> SINTER myset employee
1) "George"
2) "Smith"
127.0.0.1:6379> SUNION myset employee
1) "George"
2) "Alan"
3) "Timothy"
4) "Michael"
5) "Paul"
6) "Smith"
7) "Christoper"
```

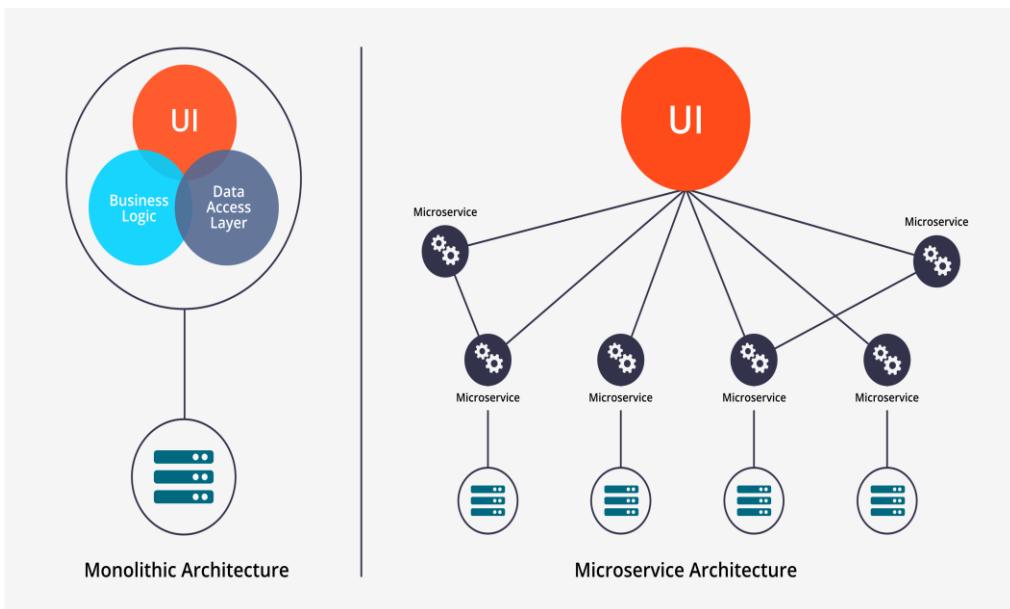
### 2.1.3. Pengertian Microservices

Microservice adalah sebuah teknologi baru yang sedang ramai diperbincangkan dalam merancang sebuah perangkat lunak sebagai layanan yang dapat digunakan secara independen [8]. Pada dasarnya Microservice membuat sebuah aplikasi yang besar dengan cara memecah menjadi service-service yang kecil lalu menggabungkannya agar dapat berjalan secara bersamaan.

Sejarah munculnya microservice waktu itu ketika para software arsitek sedang mengadakan pertemuan dan memiliki pandangan yang sama dalam membuat sebuah perangkat lunak. Pada saat tiga hari sebelum berakhirnya pertemuan tersebut, salah satu dari peserta berteriak dan berkata bagaimana cara membangun sistem yang ada diganti ketika dalam perbaikan? Dan akhirnya muncul istilah micro pada aplikasi tersebut menurut James Lewis[2].

Gambar2.1 adalah gambaran sederhana Microservicedimana setiap aplikasi berjalan sendiri dan berkomunikasi menggunakanREST API. Pada masing-masing aplikasi dapat menggunakan bahasa pemrograman yang berbeda dan menggunakan database yang berbeda sesuai dengan kebutuhan yang akan dibuat oleh developer.

*Microservice* adalah kumpulan proses independen dan kecil yang berkomunikasi antara satu dengan lainnya untuk membentuk aplikasi kompleks yang agnostik terhadap bahasa *API* apa pun. Servis-servis ini terdiri dari blok-blok kecil, terpisah, dan fokus pada tugas-tugas ringan untuk memfasilitasi metode modular dalam pembangunan sistem. Mircoservice sendiri merupakan pengembangan lanjutan dari *Service-oriented architecture (SOA)* karena *Microservice* merupakan sistem yang terdiri dari komponen komponen berupa *services* yang *modular, autonomous* yang memiliki tujuan masing masing namun ter-orkestrasi melalui protokol *light-weight* dengan satu sama lain untuk mencapai satu tujuan tertentu terutama di dalam pengembangan *software* [12]. Pada penelitian ini menggunakan arsitektur *microservice* sebagai potongan-potongan kecil dari bagian *service* yang difokuskan terhadap satu modul. .



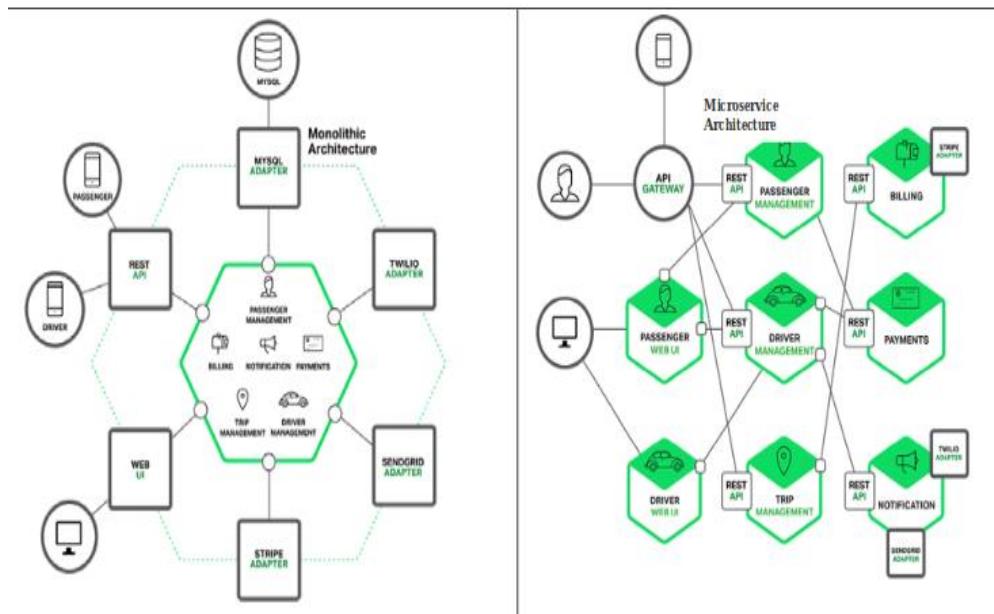
yang mengambil setiap message dari *message broker* diperlukan *library* seperti *Java Message Services(JMS)*. *Message* yang masuk satu persatu tidak dapat langsung digunakan untuk membuat pengembangan proses *digital invoice* dan *satllement* di korespondensi sistem maka diperlukan *database* untuk menyimpan *message* sementara menunggu *message* telah diterima semuanya oleh *middleware*[26]. Selain itu, terdapat *message COMPLETED* yang menandakan bahwa semua *message* telah terkirim dan *middleware* dapat men-trigger untuk menjalankan proses integrasi *digital invoice* dan *satllement* di *microservices*. Sebagaimana kebutuhan dari *microservices* sebagai jembatan, dimana perlu menyimpan sementara. Untuk menyimpannya diperlukan sebuah layanan untuk menyimpan *message-message* tersebut. Selain itu diperlukan layanan untuk menjalankan proses integrasi yaitu melengkapi kebutuhan input korespondensi sistem. Dalam proses integrasi dilakukan beberapa kegiatan diantaranya, menyusun urutan baris *message*, menambahkan baris yang berisikan informasi versi dokumen

yang dibutuhkan dan menambahkan baris yang berisikan nama file dari setiap dokumen pendukung. *Microservices* juga mengirimkan file ke lokasi dimana korespondensi sistem dapat membaca dokumen pendukung melalui *Secure File Transfer Protocol(SFTP)*. Layanan provisi dokumen memiliki table untuk menyimpan versi. Layanan ini dipublikasi sebagai sebuah *REST API* menggunakan format *JSON* untuk permintaan dan responnya. Dengan begitu lebih mudah bagi *microservices* untuk mengkonsumsi informasinya karena dapat dipetakan sebagai sebuah objek dibahasa pemrograman seperti *Java*. Selanjutnya terdapat layanan untuk menghubungkan proses integrasi ke *Document Management System(DMS)*. Layanan ini juga dipublikasi sebagai *REST API* menggunakan format *JSON*. Setiap dokumen pendukung memiliki folder dan metadata yang digunakan untuk mem-filter informasi yang dicari. Layanan yang dibuat perlu didefinisikan secara umum agar pengguna layanan dapat leluasa dalam mencari dokumen yang dibutuhkan dengan kombinasi berupa nama folder dan pencarian berdasarkan metadata yang dibutuhkan[25].

Terakhir ada *microservices* untuk menghubungkan ke korespondensi sistem. Dipublikasi menggunakan *REST API*, sehingga dapat dipanggil menggunakan protokol http ketika dibutuhkan. Layanan ini menkonversi setiap panggilan dari client ke format yang dipahami oleh korespondensi sistem. Format tersebut akan dikonsumsi dan didistribusikan ke variabel-variabel yang sudah didefinisikan dikorespondensi sistem. Komunikasi dari *microservices* ke korespondensi sistem menggunakan *protocol message broker*, yang bersifat *asynchronous* [13].

### 2.1.3.1 Keunggulan Microservices

Microservices memiliki sejumlah keungulan dan untuk mengilustrasikan keungulan dari arsitektur microservice ini dengan membadingkanya dengan Arsitektur Monolitik. Digambarkan sebagai berikut.



Gambar 3.3 Perbandingan Arsitektur Monolitik dan Arsitektur Microservice

Pada Pengilustrasian mencontohkan tentang pembagunan aplikasi baru yakni pemesanan taxi. Pada penggambaran Arsitektur Monolitik pengembang membuat satu layanan lalu hanya perlu menyalin satu aplikasi yang telah dibuat lalu dikemas/packets ke server. Dalam waktu-kewaktu aplikasi ini menjadi besar sehingga menjadi suatu aplikasi yang kompleks dan sangat kompleks untuk pengembang tunggal untuk memahami aplikasi tersebut. Akibatnya untuk memperbaiki bug dan menerapkan fitur baru menjadi benar-benar sulit dan memakan waktu jika baris kode sulit dipahami. Sehingga untuk menagulangi hal tersebut dibuatlah Arsitektu Microservice yang mana setiap area fungsional dijalankan oleh MICROSERVICE sendiri.

Selain itu aplikasi web dibagi menjadi seperangkat aplikasi web sederhana (seperti satu untuk penumpang dan satu untuk driver contoh memanggil taxi). Hal ini membuat lebih mudah menerapkan layanan yang berbeda bagi pengguna tertentu, perangkat, atau penggunaan khusus.

Dari Penjabaran penggambaran Aplikasi Microservice diatas dapat diterjemahkan keungulan Microservice

1. Menangani masalah kompleksitas
  2. Arsitektur ini memungkinkan setiap layanan untuk dikembangkan independent oleh tim yang difokuskan pada layanan tersebut.
  3. Memungkinkan layanan untuk digunakan secara mandiri.
  4. Baris kode ditulis lebih sedikit dan lebih banyak membuat flexibilitas untuk membuat perubahan pada aplikasi yang dikelola.
  5. Setiap layanan dapat dipantau, diperbarui secara independent untuk mencocokan tautan
  6. Setiap layanan dapat dikembangkan dan diperbarui secara mandiri.
  7. Menghilangkan komitmen jangka panjang untuk setumpuk teknologi.

(Richardson, Chris, 2015).

## 2.1.4 Pengertian REST

*REST* adalah filosofi desain yang mendorong kita untuk menggunakan protokol dan fitur yang sudah ada pada *Web* untuk memetakan permintaan terhadap sumber daya pada berbagai macam representasi dan manipulasi data di Internet (Scribner, 2009). *REST* adalah gaya arsitektural yang memiliki aturan seperti antar muka yang seragam, sehingga jika aturan tersebut diterapkan pada *web service* akan dapat memaksimalkan kinerja *web service* terutama pada performa, skalabilitas, dan kemudahan untuk dimodifikasi[15].

Gagasan utama dari *REST* sebagai komponen dari aplikasi yang perlu digunakan atau dialamatkan [16]. *REST* WS cara yang lebih ringan dan sederhana, dan berfokus pada sumber daya [17]. *REST* dapat dijelaskan dalam lima batasan, diantaranya:

1. *Resource Identification*: *Identification Web* bergantung pada *Uniform Resource Identifier (URI)* untuk mengidentifikasi sumber daya, sehingga link kesumber daya dapat dibentuk menggunakan skema. sehingga *link* ke sumber daya dapat dibentuk menggunakan skema identifikasi yang mudah untuk dikenali [16].
2. *Connectedness*: artinya klien dari *RESTful Service* seharusnya mengikuti link untuk menemukan sumber daya agar dapat berinteraksi dengan *Service* [6].
3. *Uniform Interface*: artinya sumber daya harus tersedia melalui antarmuka yang seragam dengan semantik yang mendefinisikan interaksi, seperti *Hypertext Transfer Protocol (HTTP)* [16]. *HTTP* mencakup metode *POST*, *POST GET*, *PUT* dan *DELETE* [18].

4. *Self-Describing Messages*: resource yang ada, RESTful menggunakan lebih dari satu format data (XML, ( JSON, RDF, dll) dibandingkan dengan *SOAP* (XML), namun hal ini tergantung *developer*.

*Stateless Interactions*: mengharuskan setiap *request* dari klien lengkap, dalam arti bahwa semua informasi untuk melayani *request* ke *server* harus berisi setiap informasi yang dibutuhkan agar request dapat dipahami [16], , dan tidak ada ketergantungan dengan *state* atau penanda dari *client*.

Terdapat dua bagian pesan yang digunakan untuk membangun komunikasi dengan *server* yaitu pesan *Header* dan pesan *Body*. *HTTP Header*, *Header* yang umum meliputi header request diilustrasikan pada gambar 1, 1 header response pada gambar 2, dan terdapat bidang *entitas-header* [19]. Setiap request sumberdaya dari masing-masing masing client dapat dikendalikan dengan memanfaatkan *HTTP Header* [20]. Setiap kolom *Header* terdiri dari nama diikuti dengan titik dua (":") atau *white space* dan konten *field*. Nama *field* bersifat *case-sensitive*. *Header* berisikan semua informasi yang diperlukan untuk mengumpulkan gumpulkan metode *request* [21] dan *respon* Sedangkan *HTTP body* mencakup pesan *HTTP* yang digunakan untuk memuat *entitas body* melalui protkol *HTTP* yang berhubungan dengan *request* tuangkan pada gambar 3 dan *respon* pada gambar 4 [19]. Saat *client* melakukan request, *HTTP body* bisanya berisikan informasi setiap parameter yang untuk *request*. Sedangkan *respon* berisikan informasi yang didapatkan dari hasil request. berbagai klien yang ditulis dalam bahasa pemrograman yang berbeda [22]. . Dengan demikian REST dapat mengoperasikan operasi *CRUD* (*create*, *read*, *update* dan *delete*) yang [23] dapat dilakukan dengan memanfaatkan metode *HTTP* antara lain *POST*,*POST GET*, *PUT* dan *DELETE*. Format *application/x-www-form-urlencoded* yang digunakan oleh masing-masing masing metode

*HTTP* diantaranya

- 1) *GET* dan *DELETE* berbeda dengan *POST* dan *PUT* adalah berbeda, hal ini dikarenakan cara parsing parsin data yang berbeda. Parsing data pada metode *GET* dan *DELETE* dimuai melalui *URL*, sedangkan *POST* dan *PUT* melakukan *parsing* data melalui *payload* *HTTP* dengan memanfaatkan media type ‘*application/x-www-form-URL URLEncoded*’. Kode status *HTTP* respon *server* terhadap aksi yang dilakukan oleh *client*.
- 2) *Kode status 201*: request telah terpenuhi dan menghasilkan sumber daya yang baru Create.

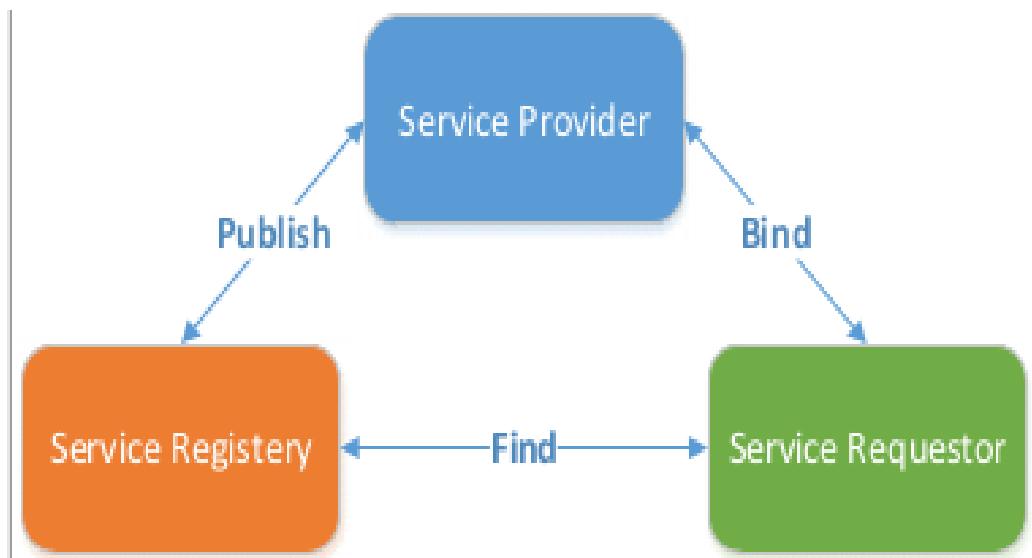
*Kode status 200*: respon standar untuk request *HTTP* dari client yang dinyatakan sukses oleh *server*. Respon sebenarnya akan tergantung te pada metoderequest yang digunakan.

### **1.1.5 Web Services**

Web Service adalah layanan yang tersedia di Internet. Web Service menggunakan format standar XML untuk pengiriman pesannya. Web Services juga tidak terikat kepada bahasa pemrograman atau sistem operasi tertentu (Ethan Cerami, 2002). Web Services adalah antar muka yang mendeskripsikan koleksi yang dapat diakses dalam jaringan menggunakan format standar XML untuk pertukaran pesan. Web Services mengerjakan tugas yang spesifik. Web Services dideskripsikan menggunakan format standar notasi XML yang disebut services description (Gottschalk, 2002).

### **2.1.6 Arsitektur Web Services**

Arsitektur web service secara umum dapat dilihat pada gambar 2.1 dibawah ini;



Gambar 2.1 Arsitektur web service

(sumber: Brittenham, 2002 )

Pada gambar diatas, ada tiga komponen utama dari web service yaitu:

1. Service provider: Penyedia web service yang berfungsi menyediakan kumpulan web services yang dapat diakses oleh pengguna.
2. Service requestor: Adalah aplikasi yang bertindak sebagai pengguna yang melakukan permintaan layanan (berupa web services) ke service provider.
3. Service registry: Adalah tempat dimana service provider mempublikasikan layanannya. Pada arsitektur Web service, Service registry bersifat opsional.

## **BAB III PENJELASAN TOOLS DAN BAHASA PEMROGRAMAN**

### **3.1 PENJELASAN TOOLS DAN BAHASA PEMOGRAMAN YANG DIGUNAKAN**

#### **3.1.1 Visual Studio Code**

Aplikasi Visual Studio Code Text merupakan aplikasi editor yang digunakan untuk kode dan teks yang dapat berjalan diberbagai platform operating system dengan menggunakan teknologi Phyton API. Aplikasi ini diciptakan karena terinspirasi dari aplikasiVim, Aplikasi ini sangatlah fleksibel dan powerfull. Aplikasi ini dapat dikembangkan dengan menggunakan Visual Studio Code-packages.

Visual Studio Code Text bukanlah aplikasi opensource dan juga aplikasi yang dapat digunakan dan didapatkan secara gratis, namun beberapa fitur pengembangan fungsionalitas (packages) dari aplikasi ini merupakan hasil dari temuan dan memperoleh dukungan penuh dari komunitas serta memiliki lisensi aplikasi gratis. Aplikasi ini mendukung berbagai bahasa pemrograman dan mampu menyajikan fitur syntax highlight hampir di semua bahasa pemrogramman yang didukung ataupun dikembangkan oleh komunitas.

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of the project structure under 'TPN-BILLING-API'. The 'components/billings' folder is expanded, showing files like 'billing.js', 'billingAPI.js', 'billingController.js', etc. The main editor area displays the content of 'billingAPI.js'. The status bar at the bottom shows the file path 'D:\TPN-BILLING-API\components\billings\billingAPI.js' and the line number '30'.

```
const express = require('express');
const router = express.Router();
const BillingController = require('../billingController');
const validation = require('../middlewares/validation');
const validationSchema = require('../billingValidation');
const auth = require('../middlewares/auth');

router.get('/', function(req,res,next){
    res.locals.function_id = "billings-index"
    next();
},
auth,
BillingController.index);

router.post('/', function(req,res,next){
    res.locals.jsonSchema = validationSchema.createSchema
    res.locals.function_id = "billings-store"
    next();
},
auth,
validation,
BillingController.store);

router.get('/exportexcel', function(req,res,next){
    res.locals.function_id = "billings-export-excel"
    next();
},
auth,
BillingController.exportToExcel);
```

Gambar 3. 1 Tampilan Visual Studio Code

Berikut beberapa fitur yang diunggulkan dari aplikasi Visual Studio Code Text:

- Goto Anything

Fitur yang sangat membantu dalam membuka file ataupun menjelajahi isi dari file hanya dengan beberapa keystrokes.

- Multiple Selections

Fitur ini memungkinkan user untuk mengubah secara interaktif banyak baris sekaligus, mengubah nama variabel dengan mudah, dan memanipulasi file lebih cepat dari sebelumnya.

- **Command Pallete**

Dengan hanya beberapa keystrokes, user dapat dengan cepat mencari fungsi yang diinginkan, tanpa harus menavigasi melalui menu.

- **Distraction Free Mode**

• Bila user memerlukan fokus penuh pada aplikasi ini, fitur ini dapat membantu user dengan memberikan tampilan layar penuh.

- **Split Editing**

• Dapatkan hasil yang maksimal dari monitor layar lebar dengan dukungan editing perpecahan. Melakukan editing di sisi file dengan sisi, atau mengedit dua lokasi di satu file. Anda dapat mengedit dengan banyak baris dan kolom yang user inginkan.

- **Instant Project Switch**

Mengambil seluruh file yang dimasukkan kedalam project pada aplikasi ini terhubung dengan fitur Goto Anything untuk menjelajahi semua file yang ada ataupun untuk beralih ke file dalam project lainnya dengan cepat.

- **Plugin API**

Aplikasi ini memiliki plugin API berbasis Phyton sehingga membuat aplikasi ini sangat tangguh.

- **Customize Anything**

Aplikasi Visual Studio Code mempunyai dan memberikan user fleksibilitas dalam hal pengaturan fungsional dalam aplkasi ini.

- **Cross Platform**

Aplikasi ini dapat berjalan hampir disemua operating system modern seperti Windows, OS X, dan Linux based operating system.

### **3.1.1.2 MongoDB Compass**

MongoDB merupakan salah satu *database* NoSQL dengan basis dokumen yang sangat populer saat ini, MongoDB didirikan oleh tiga serangkai yang terdiri dari Kevin Ryan, Eliot Horowitz, dan Dwight Merriman. Ketiganya tergabung di MongoDB Inc. dan berperan pada jabatannya masing — masing. Kevin menjabat *board member*, Dwight Merriman menjabat sebagai *chairman*, dan Eliot menjabat sebagai CTO di MongoDB Inc.

MongoDB menawarkan fitur *high performance, high availability*, dan *automatic scaling*, MongoDB menggunakan Javascript untuk melakukan operasi seperti CRUD, agregasi, *indexing*, dan operasi *database* lainnya, karena MongoDB menggunakan javascript maka dalam penyimpanan datanya MongoDB tidak menggunakan table, tetapi MongoDB menyimpan datanya dalam suatu dokumen yang strukturnya seperti JSON. Selanjutnya saya akan memberikan contoh penggunaan MongoDB, langkah pertama yang harus kita lakukan adalah menginstall MongoDB, ada beberapa cara yang dapat digunakan dalam menginstall MongoDB, di sini saya install MongoDB dengan menggunakan via brew

#### **1. Kelebihan**

1. Performa yang ditawarkan MongoDB lebih cepat disebabkan oleh memcached dan format dokumennya yang berbentuk seperti JSON
2. Kita tidak perlu membuat struktur tabel, karena MongoDB akan otomatis membuatkan struktur tabelnya pada saat proses insert (fleksibel skema)

## **2. Kekurangan**

1. Belum banyak hosting yang support
2. fleksibelitas dalam query (sebagai contoh tidak adanya JOIN)

### **3.1.2 Framework Express**

Express adalah salah satu framework terbaik Node. Memiliki dukungan besar dan sekelompok fitur berguna. Ada banyak artikel besar di luar sana, yang mencakup semua dasar-dasar. Namun, kali ini saya ingin untuk menggali sedikit lebih dalam dan berbagi alur kerja untuk membuat situs web lengkap. Secara umum, artikel ini adalah tidak hanya untuk Express, tapi untuk menggunakannya dalam kombinasi dengan beberapa alat besar lainnya yang tersedia untuk pengembang Node.

Saya berasumsi bahwa Anda sudah familiar dengan Nodejs, telah terinstall pada sistem Anda dan bahwa Anda mungkin telah membangun beberapa aplikasi dengan itu. Di jantung Express adalah connect. Ini adalah kerangka middleware, yang datang dengan banyak hal-hal yang berguna. Jika Anda bertanya-tanya apa itu middleware, berikut adalah sebuah contoh cepat:

```

var connect = require('connect'),
    http = require('http');

var app = connect()
    .use(function(req, res, next) {
        console.log("That's my first middleware");
        next();
    })
    .use(function(req, res, next) {
        console.log("That's my second middleware");
        next();
    })
    .use(function(req, res, next) {
        console.log("end");
        res.end("hello world");
    });

http.createServer(app).listen(3000);

```

Gambar 3. 2 Konfigurasi Express

Middleware pada dasarnya adalah fungsi yang menerima request dan response objek dan fungsi next. Setiap middleware dapat memutuskan untuk merespons dengan menggunakan objek response atau meneruskan aliran ke fungsi berikutnya dengan memanggil callback next. Dalam contoh di atas, jika Anda menghapus panggilan metode next() di middleware kedua, string hello world tidak akan pernah dikirim ke browser. Secara umum, begitulah cara kerja Express. Ada beberapa middlewares standar, yang tentu saja, menghemat banyak waktu. Seperti misalnya, Body parser yang mem-parsing permintaan body dan mendukung aplikasi json, application/x-www-form-urlencoded dan multipart/form-data. Atau Cookie parser, yang mem-parsing cookie header dan saran populates req.cookies dengan objek mengetik

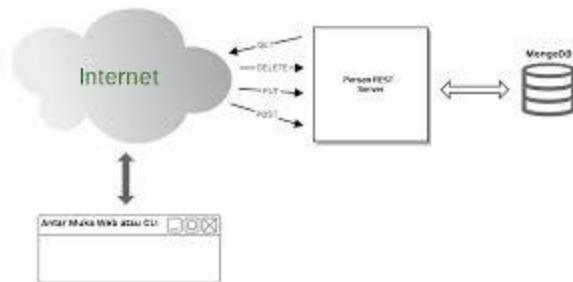
```

app.get('/hello.txt', function(req, res){
    var body = 'Hello World';
    res.setHeader('Content-Type', 'text/plain');
    res.setHeader('Content-Length', body.length);
    res.end(body);
});

```

dengan nama kuki. Check benar-benar membungkus Connect dan menambahkan beberapa fungsi baru di sekitarnya. Seperti misalnya, routing logic, yang membuat prosesnya lebih lancar. Berikut adalah contoh menangani permintaan GET

### 3.1.2.1 Cara Kerja



Gambar 3. 3 Alur kerja Express

- SERVER

---

- Instalasi

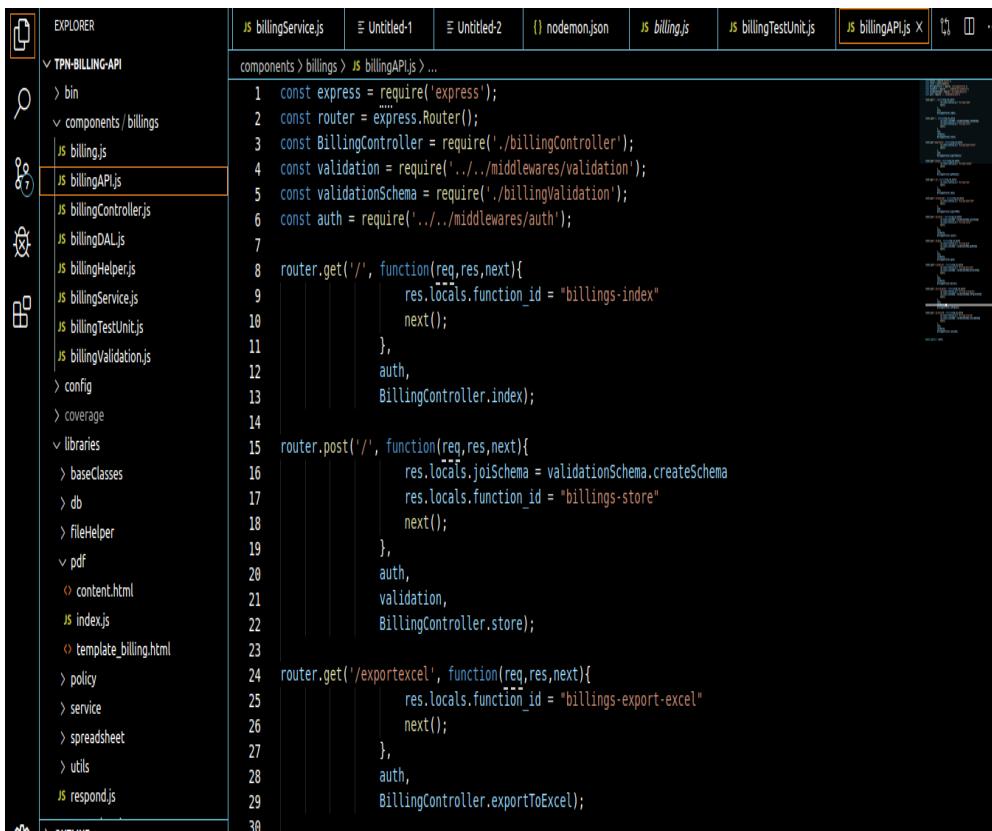
- Code
  - Pengujian

Contoh File untuk Schema mongoose, API, Controller, DAL, Service, Unit Test, nodemon, app.js, packages.json, dan package-lock.json:

## 1. Schema Mongoose/Billing

Gambar 3. 4 Alur kerja Express js

## 2. API/Billing



The screenshot shows a code editor interface with the following details:

- EXPLORER:** On the left, the project structure is shown under the folder `TPN-BILLING-API`. The `components/billings` folder contains files: `billing.js`, `billingAPI.js` (which is currently selected), `billingController.js`, `billingDAL.js`, `billingHelper.js`, `billingService.js`, and `billingTestUnit.js`. Other folders like `bin`, `config`, `coverage`, `libraries`, and `utils` are also listed.
- CODE EDITOR:** The main area displays the `billingAPI.js` file content. The code uses Express.js to define routes for billings. It includes imports for express, Router, BillingController, validation, validationSchema, auth, and BillingValidation. It defines two main routes: a get route for '/' and a post route for '/exportexcel'. Both routes call the `BillingController` with specific function IDs and authentication checks.

```
const express = require('express');
const router = express.Router();
const BillingController = require('../billingController');
const validation = require('../middlewares/validation');
const validationSchema = require('../validation');
const auth = require('../middlewares/auth');

router.get('/', function(req,res,next){
    res.locals.function_id = "billings-index"
    next();
},
auth,
BillingController.index);

router.post('/', function(req,res,next){
    res.locals.joiSchema = validationSchema.createSchema
    res.locals.function_id = "billings-store"
    next();
},
auth,
validation,
BillingController.store);

router.get('/exportexcel', function(req,res,next){
    res.locals.function_id = "billings-export-excel"
    next();
},
auth,
BillingController.exportToExcel);
```

Gambar 3. 5 API

### 3. Controller/Billing

The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows the project structure under "TPN-BILLING-API". The "components/billings" folder contains files: billing.js, billingAPI.js, and billingController.js (which is selected). Other files in the folder include billingDAL.js, billingHelper.js, billingService.js, billingTestUnit.js, and billingValidation.js.
- EDITOR:** Displays the content of the selected file, `billingController.js`. The code is as follows:

```
1 const billingService = require('./billingService');//call class invoice services
2 const respond = require("../libraries/respond") //call libraries respond
3 const spreadsheet = require('../libraries/spreadsheet');
4 const PdfMaker = require('../libraries/pdf');
5 const utils = require('../libraries/utils');
6 const config = require('config');

7
8
9
10 /**
11 * get all item
12 *@param {Object} req express request object
13 *@param {Object} res express response object
14 *@api public
15 */
16
17 exports.index = async(req, res) => {
18     //filter by app_id
19     if(req.headers['eg-consumer-id'] != config.get('tpn.app_id')){
20         req.query['filter-created.app_id'] = req.headers['eg-consumer-id'];
21     }
22     const getResult = await billingService.get(req.query);
23     let meta = {};
24     //for datatables
25     meta.total = getResult.total;
26     meta.filtered = getResult.filtered;
27
28     return respond.resSuccessData(res,undefined getResult.data,meta);
29 }
30
31 /**
```

Gambar 3. 6 controller

#### 4. DAL/Billing

The screenshot shows a code editor interface with a sidebar on the left displaying the project structure of 'TPN-BILLING-API'. The current file being viewed is 'components/billings/billingDAL.js'. The code in the editor is as follows:

```
components > billings > JS billingDAL.js > ⓘ getData > ⓘ exports.getData > ⓘ result > ⓘ $project > Ⓢ 'purchaseorder.final_price'  
38 exports.getData = async(query) => { //untuk get data perpage dan menampilkan result hasil pencarian  
39   query['sort-po_date'] = (query['sort-po_date'])?query['sort-po_date']:-1;  
40  
41   let result = await Billing.aggregate([  
42     {  
43       $match:{  
44         "$or":utils.getQueryOr(query,'likeor','like'),  
45         ...utils.getQuery(query,'filterin','in'),  
46         ...utils.getQuery(query,'filter-'),  
47         ...utils.getQuery(query,'filternum','number'),  
48         ...utils.getQuery(query,'exists','exists'),  
49         ...utils.getQuery(query,'filterobjid','objid'),  
50         ...utils.getQuery(query,'bool','bool')  
51         ,...utils.getQuery(query,'like','like')  
52         ,...utils.betweenDate(utils.getQuery(query,'between'))  
53       }  
54     },  
55     {  
56       $project:{  
57         billing_number: 1,  
58         'purchaseorder.product.name': 1,  
59         'purchaseorder.to.name': 1,  
60         'billing_date': 1,  
61         due_date: 1,  
62         'purchaseorder.final_price': 1 ,  
63         status_billing: 1 ,  
64         status_paid: 1 ,  
65         status_delivered: 1,  
66         _id: 0,  
67       }  
68     }  
69   ]);  
70   return result;  
71 };
```

Gambar 3. 7 DAL

## 5. Service/Billing

The screenshot shows a code editor interface with two panes. The left pane displays the project structure of 'TPN-BILLING-API' with various files and folders like bin, components/billings, config, coverage, libraries, db, fileHelper, pdf, and utils. The right pane shows the content of the selected file, 'components/billings/billingService.js'. The code is as follows:

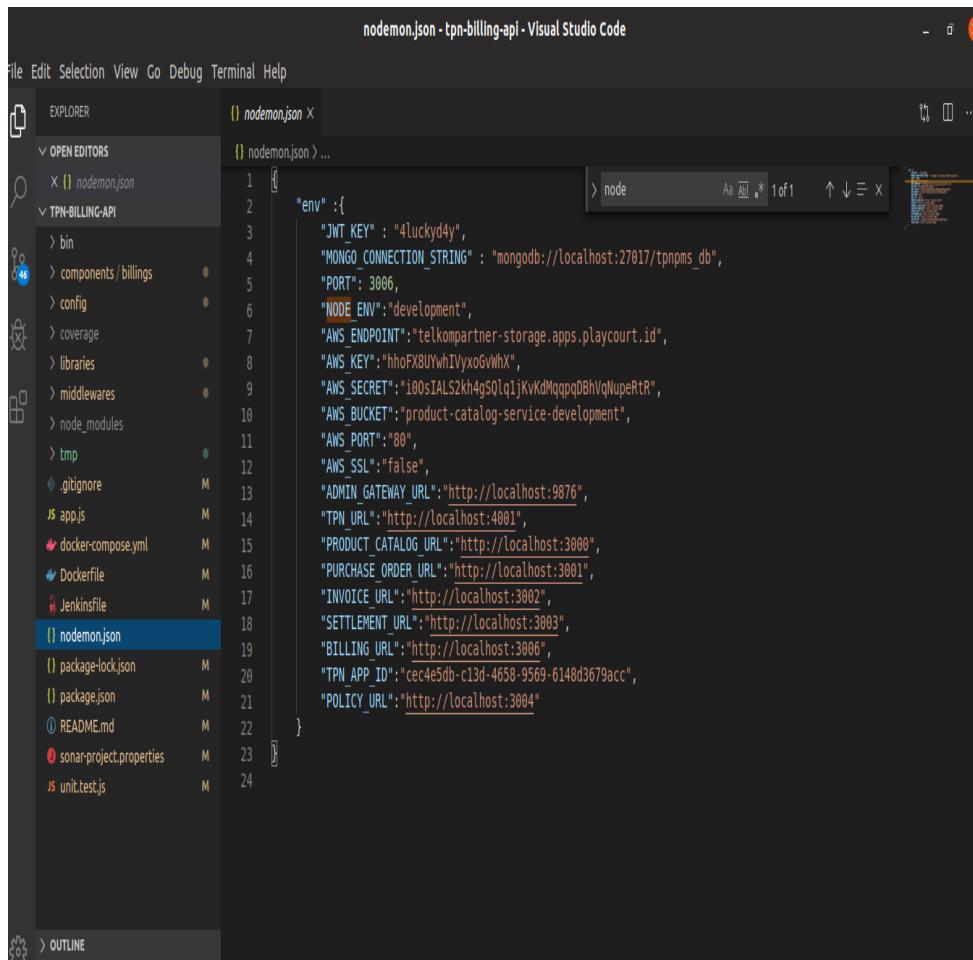
```
components > billings > JS billingService.js > getExportData
1 const billingDAL = require('./billingDAL'); //call class invoice services
2 const service = require('../libraries/service');
3 const utils = require('../libraries/utils');
4 const moment = require('moment');
5 const billingHelper = require('./billingHelper');
6 const fileHelper = require('../libraries/fileHelper');
7
8
9 /**
10 * Get all item
11 * @param {Object} req express request object
12 * @param {Object} res express response object
13 * @api public
14 */
15 const get = async (query) => { //get data from query billingDAL
16     let getResult = {};
17
18     getResult.data = await billingDAL.get(query);
19     for(let i=0; i< getResult.data.length; i++){
20         getResult.data[i].billing_date = moment(getResult.data[i].billing_date).format('YYYY-MM-DD');
21         getResult.data[i].due_date = moment(getResult.data[i].due_date).format('YYYY-MM-DD');
22     }
23     getResult.total = await billingDAL.getTotal();
24     getResult.filtered = await billingDAL.getTotalFiltered(query);
25
26     return getResult;
27 }
28 }
```

Gambar 3. 8 Services

## 6. Unit Testing/Billing

### Gambar 3.9 Unit Testing

## 7. Nodemon

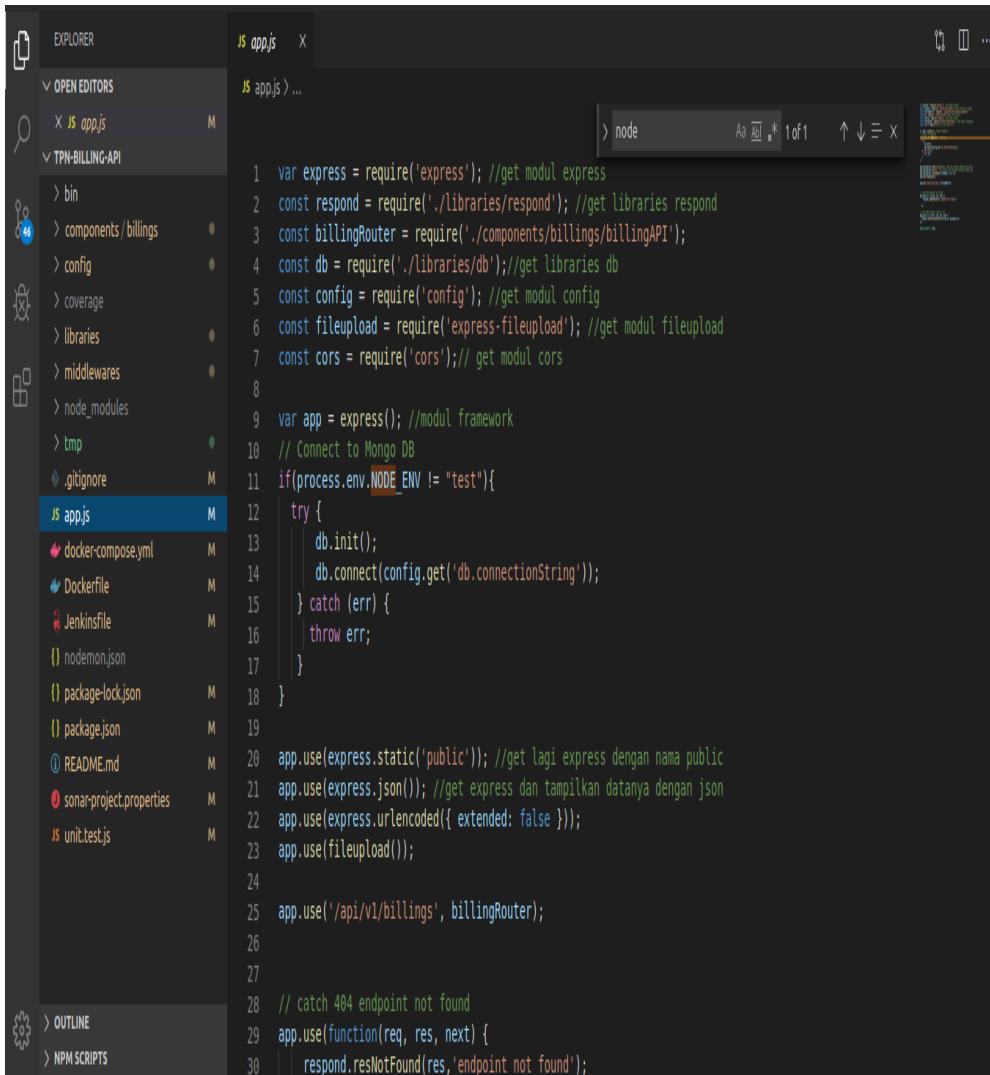


The screenshot shows the Visual Studio Code interface with the title bar "nodemon.json - tpn-billing-api - Visual Studio Code". The left sidebar shows a tree view of the project structure under "OPEN EDITORS" and "TPN-BILLING-API". The main editor area displays the content of the "nodemon.json" file:

```
1  {
2    "env": {
3      "JWT_KEY": "4luckyd4y",
4      "MONGO_CONNECTION_STRING": "mongodb://localhost:27017/tpnpms_db",
5      "PORT": 3006,
6      "NODE_ENV": "development",
7      "AWS_ENDPOINT": "telkompartner-storage.apps.playcourt.id",
8      "AWS_KEY": "hofFX8UYwhIVyxogWhX",
9      "AWS_SECRET": "100siALS2kh4g5Qlq1jKvKdMqqpqDBhVqNupeRtR",
10     "AWS_BUCKET": "product-catalog-service-development",
11     "AWS_PORT": "80",
12     "AWS_SSL": "false",
13     "ADMIN_GATEWAY_URL": "http://localhost:9876",
14     "TPN_URL": "http://localhost:4001",
15     "PRODUCT_CATALOG_URL": "http://localhost:3000",
16     "PURCHASE_ORDER_URL": "http://localhost:3001",
17     "INVOICE_URL": "http://localhost:3002",
18     "SETTLEMENT_URL": "http://localhost:3003",
19     "BILLING_URL": "http://localhost:3006",
20     "TPN_APP_ID": "cec4e5db-c13d-4658-9569-6148d3679acc",
21     "POLICY_URL": "http://localhost:3004"
22   }
23 }
24 }
```

Gambar 3. 10 Nodemon

## 8. App.js



```
var express = require('express'); //get modul express
const respond = require('./libraries/respond'); //get libraries respond
const billingRouter = require('./components/billings/billingAPI');
const db = require('./libraries/db');//get libraries db
const config = require('config'); //get modul config
const fileupload = require('express-fileupload'); //get modul fileupload
const cors = require('cors');// get modul cors

var app = express(); //modul framework
// Connect to Mongo DB
if(process.env.NODE_ENV != "test"){
    try {
        db.init();
        db.connect(config.get('db.connectionString'));
    } catch (err) {
        throw err;
    }
}

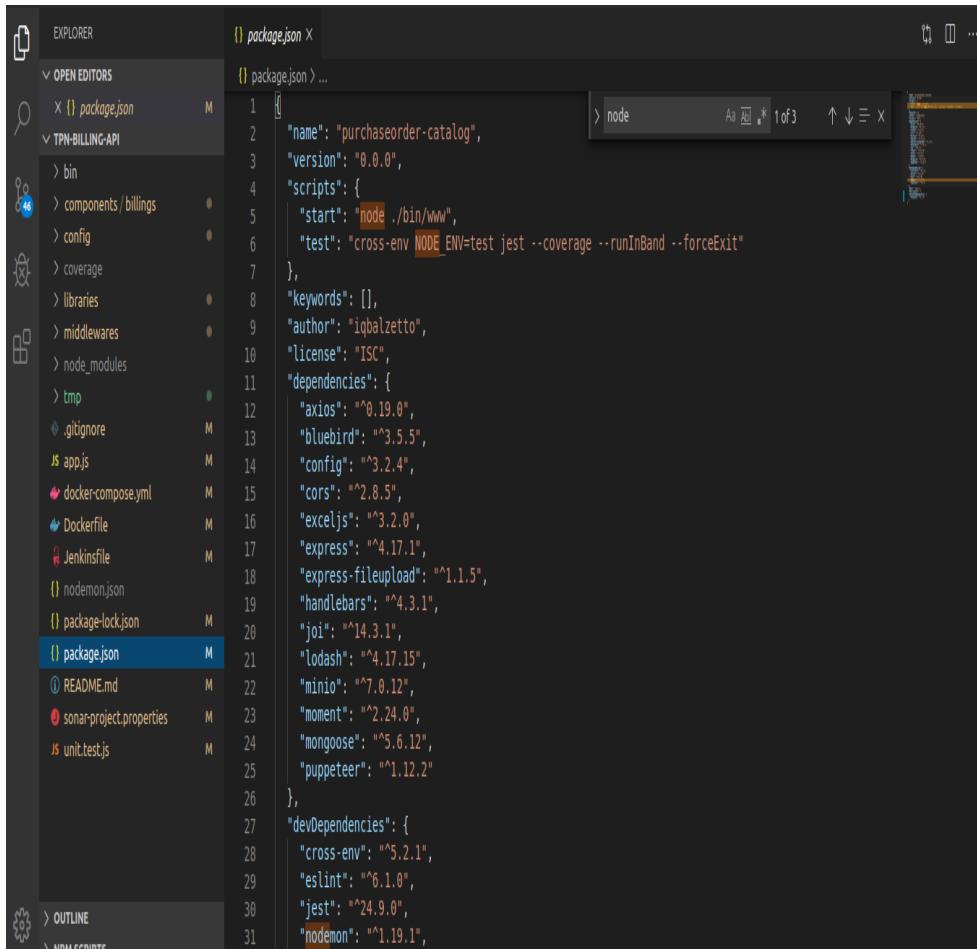
app.use(express.static('public')) //get lagi express dengan nama public
app.use(express.json()); //get express dan tampilkan datanya dengan json
app.use(express.urlencoded({ extended: false }));
app.use(fileupload());

app.use('/api/v1/billings', billingRouter);

// catch 404 endpoint not found
app.use(function(req, res, next) {
    respond.NotFound(res,'endpoint not found');
});
```

Gambar 3. 11 App.js

## 9. Packages.json



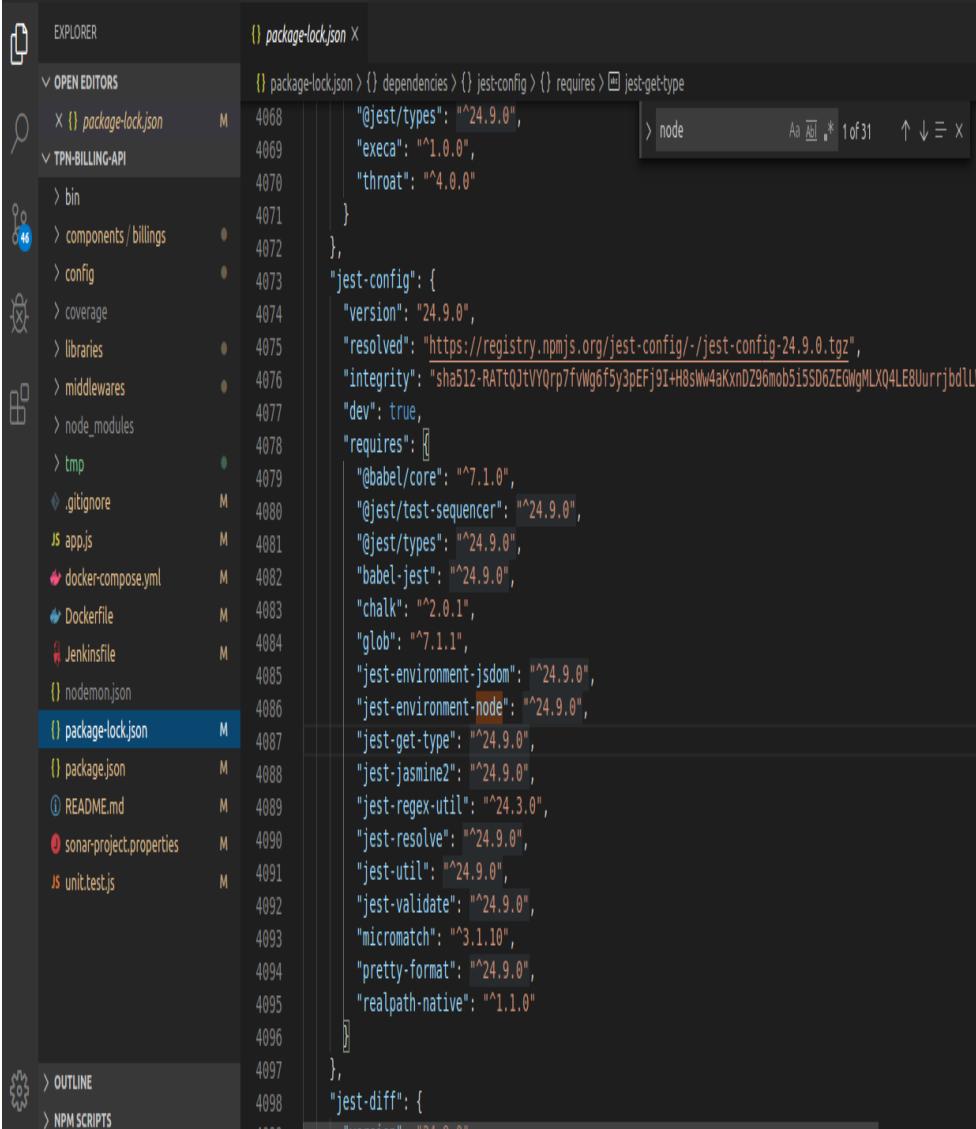
The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** pane on the left: Shows the project structure with files like `bin`, `components/billings`, `config`, `coverage`, `libraries`, `middlewares`, `node_modules`, `tmp`, `.gitignore`, `app.js`, `docker-compose.yml`, `Dockerfile`, `Jenkinsfile`, `nodemon.json`, `package-lock.json`, and `package.json`.
- EDITOR** pane on the right: Displays the `package.json` file content.
- Status Bar**: Shows the file name `package.json`, the current line `1 of 3`, and other status indicators.

```
1 {  
2   "name": "purchaseorder-catalog",  
3   "version": "0.0.0",  
4   "scripts": {  
5     "start": "node ./bin/www",  
6     "test": "cross-env NODE_ENV=test jest --coverage --runInBand --forceExit"  
7   },  
8   "keywords": [],  
9   "author": "iqbalzetto",  
10  "license": "ISC",  
11  "dependencies": {  
12    "axios": "^0.19.0",  
13    "bluebird": "3.5.5",  
14    "config": "^3.2.4",  
15    "cors": "^2.8.5",  
16    "exceljs": "3.2.0",  
17    "express": "4.17.1",  
18    "express-fileupload": "1.1.5",  
19    "handlebars": "4.3.1",  
20    "joi": "14.3.1",  
21    "lodash": "4.17.15",  
22    "minio": "7.0.12",  
23    "moment": "2.24.0",  
24    "mongoose": "5.6.12",  
25    "puppeteer": "1.12.2"  
26  },  
27  "devDependencies": {  
28    "cross-env": "5.2.1",  
29    "eslint": "6.1.0",  
30    "jest": "24.9.0",  
31    "nodemon": "1.19.1",  
32  }  
33}
```

Gambar 3. 12 Packages.json

## 10. Packages.lock.json



The screenshot shows the VS Code interface with the Explorer sidebar open. The current file being viewed is `package-lock.json`. The code editor displays the JSON content of the lock file, which lists various dependencies and their versions. The file includes entries for Jest, Babel, and other testing and development tools.

```
{ "dependencies": { "jest": "24.9.0", "jest-config": "24.9.0", "jest-environment-jsdom": "24.9.0", "jest-environment-node": "24.9.0", "jest-get-type": "24.9.0", "jest-jasmine2": "24.9.0", "jest-regex-util": "24.3.0", "jest-resolve": "24.9.0", "jest-util": "24.9.0", "jest-validate": "24.9.0", "micromatch": "3.1.10", "pretty-format": "24.9.0", "realpath-native": "1.1.0" }, "devDependencies": { "babel-core": "7.1.0", "jest-test-sequencer": "24.9.0", "@jest/types": "24.9.0", "babel-jest": "24.9.0", "chalk": "2.0.1", "glob": "7.1.1", "jest-environment-jsdom": "24.9.0", "jest-environment-node": "24.9.0", "jest-get-type": "24.9.0", "jest-jasmine2": "24.9.0", "jest-regex-util": "24.3.0", "jest-resolve": "24.9.0", "jest-util": "24.9.0", "jest-validate": "24.9.0", "micromatch": "3.1.10", "pretty-format": "24.9.0", "realpath-native": "1.1.0" }, "requires": [ "jest": "24.9.0", "jest-config": "24.9.0", "jest-environment-jsdom": "24.9.0", "jest-environment-node": "24.9.0", "jest-get-type": "24.9.0", "jest-jasmine2": "24.9.0", "jest-regex-util": "24.3.0", "jest-resolve": "24.9.0", "jest-util": "24.9.0", "jest-validate": "24.9.0", "jest-diff": { "version": "24.9.0" } ] }
```

Gambar 3. 13 Packages.lock.json

### **3.1.3 Postman**

Postman ini merupakan tool wajib bagi para developer yang berkutat pada pembuatan API, fungsi utama postman ini adalah sebagai GUI API Caller namun sekarang postman juga menyediakan fitur lain yaitu Sharing Collection API for Documentation (free), Testing API (free), Realtime Collaboration Team (paid), Monitoring API (paid), Integration (paid) detailnya silahkan dicek disini. Dulu awal pertama kali postman muncul sebagai add on dari Chrome namun sekarang sudah menjadi aplikasi native (y). Jika kalian sedang develop API sangat direkomendasikan untuk menggunakan Postman.

#### **3.1.3.1 Postman, software untuk pengujian API**

API yang kita buat atau yang disediakan oleh pihak ketiga terkadang mempunyai Http method yang bermacam-macam. Jika method tersebut berupa GET, kita bisa mengujinya langsung lewat web browser. Tapi bagaimana dengan method lainnya?

Selain masalah method, pengujian yang tidak bisa dilakukan di web browser adalah ketika API mengharuskan kita untuk menambahkan parameter di header. Solusi dari semua masalah tersebut adalah Postman.

Berikut adalah beberapa fitur yang bisa Anda gunakan :

## 1. Http Method yang lengkap

The screenshot shows the Postman application interface. On the left, there's a sidebar with tabs for 'History', 'Collections' (which is selected and highlighted in red), and 'APIs BETA'. Below these are several collection items: 'TPN API' (75 requests), 'FINANCE' (1 request), 'integrasi' (1 request), 'integrasi copy' (1 request), 'IQBAL GANTENG SEKALI' (13 requests), 'TPN Document Service' (7 requests), and 'TPN SMILE' (11 requests). A 'New Collection' button is also visible. The main workspace on the right displays a request configuration for 'Get All (v1/products/)'. The method dropdown shows 'GET' is selected. The URL field contains '({{gateway}})v1/products'. To the right of the URL are tabs for 'Headers (1)', 'Body', and 'Pre-request Script'. A large vertical list of HTTP methods is displayed on the far right, including: GET, POST, PUT, PATCH, DELETE, COPY, HEAD, OPTIONS, LINK, UNLINK, PURGE, LOCK, UNLOCK, PROPFIND, and VIEW. The 'POST' method is currently highlighted with a red border.

Tidak seperti di web browser biasa, dengan menggunakan Postman kita dapat menggunakan berbagai macam http method

## 2. Parameter dalam Header

The screenshot shows the Postman interface for a GET request to the endpoint `/v1/products/`. The request URL contains a placeholder `{(gateway)}`. The Headers tab is selected, showing one header entry:

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
Authorization	apiKey {{gateway.key}}/{{gateway.secret}}				
Key	Value	Description			

Below the table, there is a section labeled "Response".

Jika memerlukan parameter tambahan yang perlu dimasukkan ke dalam header, Postman telah menyediakan fiturnya. Masuk ke tab Headers lalu tambahkan key dan value.

### 3. File dalam parameter body

The screenshot shows the Postman interface with a POST request to `{gateway}/v1/products`. The Body tab is selected, showing a JSON payload:

```
1  {
2    "name": "Iqbal genteng sekali part 2",
3    "categories": [
4      "75"
5    ],
6    "tags": [
7      "Nasa"
8    ],
9    "founder": [
10      "Kiki Ginanjar"
11    ],
12    "description": "NASA.gov brings you the latest news, images and videos from America's space agency, pioneering the future in space exploration, scientific discovery",
13    "established_on": "2019-12-09T00:00:00Z",
14    "contact": [
15      {
16        "email": "kiki.jtk10@gmail.com",
17        "address": "Jl. Medan Jaya 1",
18        "phone": "08999359994",
19        "website": "http://nasa.gov"
20      }
21    ],
22    "features": [
23      {
24        "image": "http://telkonpartner-storage-dev.vsan-apps.playcourt.id/product-catalog-service-development/images.jpeg",
25        "title": "Nasa",
26        "description": "NASA.gov brings you the latest news, images and videos from America's space agency, pioneering the future in space exploration"
27      }
28    ]
29  }
```

Response

Fitur ini cukup membantu sekali ketika Anda ingin menguji api untuk registrasi misalnya, yang membutuhkan foto pengguna untuk disimpan di server.

### 4. History

Request yang telah kita kirimkan akan ditrack oleh Postman dan akan masuk ke dalam history. Dengan adanya fitur ini kita dapat memanggil kembali request yang pernah kita lakukan sebelumnya, jadi kita tidak perlu menuliskan ulang.

## **5. Share request atau collection**

Postman adalah kita bisa membagikan request yang kita buat kepada orang lain ataupun kesebuah tim. Request-request yang telah dibuat juga bisa dikumpulkan menjadi satu ke dalam sebuah collection agar lebih rapi.

## BAB IV INSTALASI TOOLS YANG DIGUNAKAN

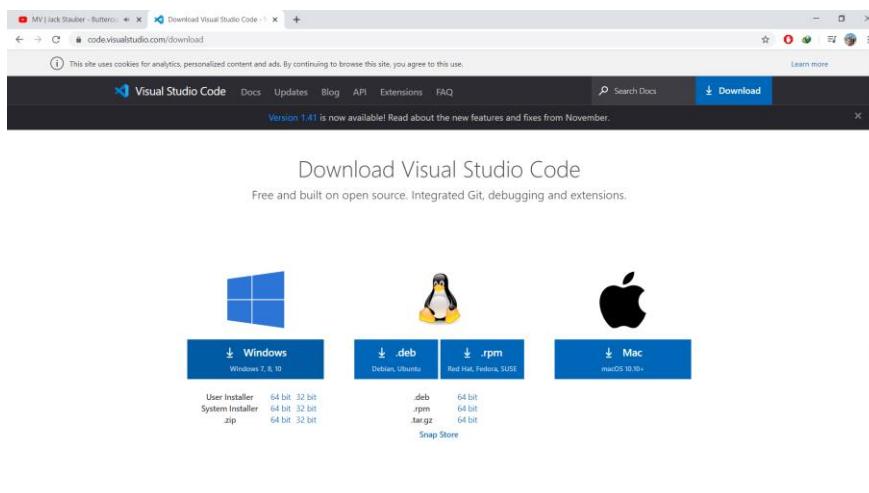
### 4.1 INSTALASI TOOLS YANG DIGUNAKAN

#### 4.1.1 Instalasi Visual Studio Code Windows

Berikut adalah langkah-langkah untuk melakukan instalasi Visual Studio Code :

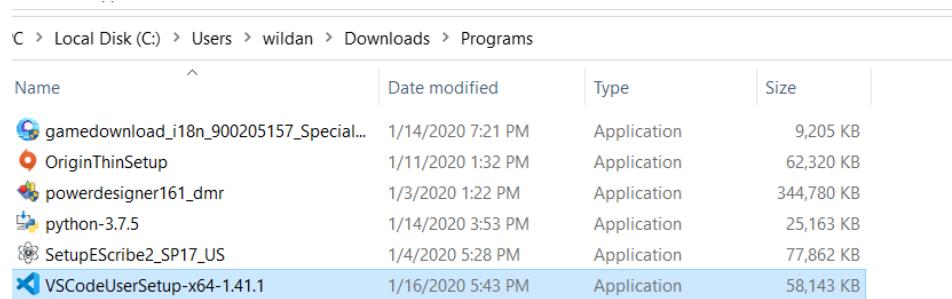
1. Tahapan yang pertama untuk melakukan instalasi maka download dahulu file .Exe pada link berikut :

<https://code.visualstudio.com/download>



Gambar 4. 1 Downlod file .exe

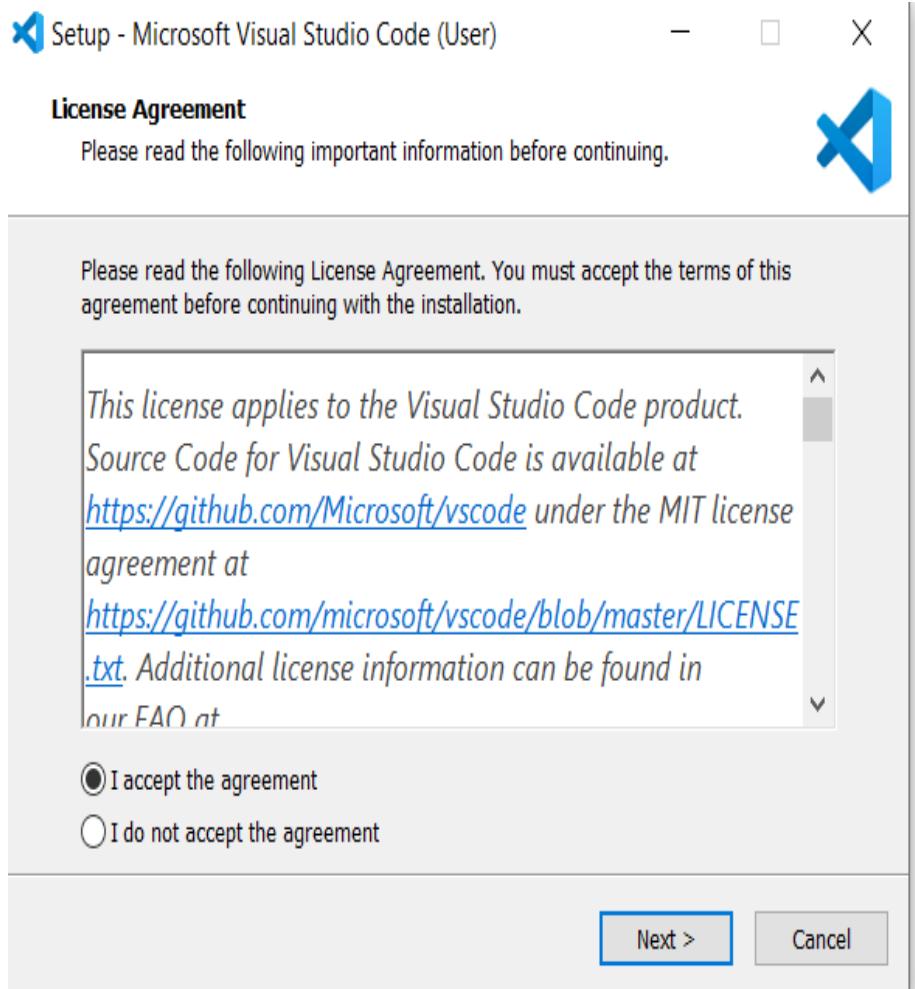
2. Tahapan yang kedua setelah selesai mendownload maka buka directory dimana file .exe tersimpan, kemudian Double klik file .exe hasil download tadi :



C > Local Disk (C:) > Users > wildan > Downloads > Programs				
Name	Date modified	Type	Size	
gamedownload_i18n_900205157_Special...	1/14/2020 7:21 PM	Application	9,205 KB	
OriginThinSetup	1/11/2020 1:32 PM	Application	62,320 KB	
powerdesigner161_dmr	1/3/2020 1:22 PM	Application	344,780 KB	
python-3.7.5	1/14/2020 3:53 PM	Application	25,163 KB	
SetupEScribe2_SP17_US	1/4/2020 5:28 PM	Application	77,862 KB	
VSCodeUserSetup-x64-1.41.1	1/16/2020 5:43 PM	Application	58,143 KB	

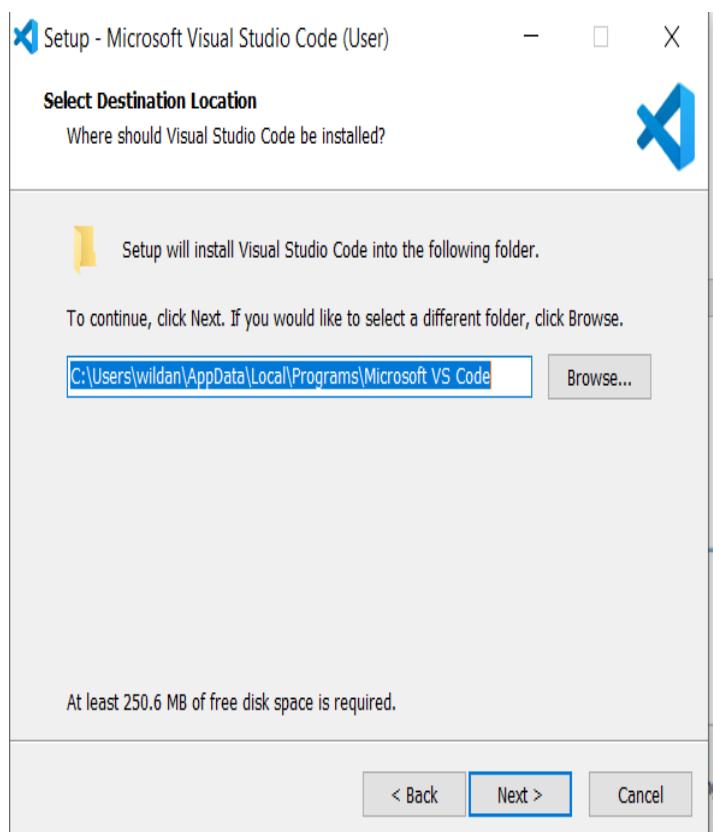
Gambar 4. 2 Double Klik File .exe

3. Tahapan ketiga jika sudah double klik file .exe maka akan muncul tampilan berikut kemudian klik NEXT:



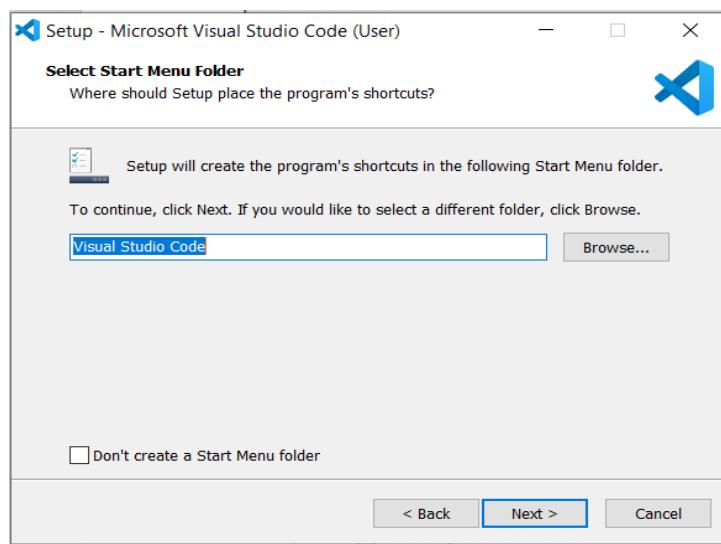
Gambar 4. 3 Proses Instalasi Visual Studio Code

4. Selanjutnya klik Next.



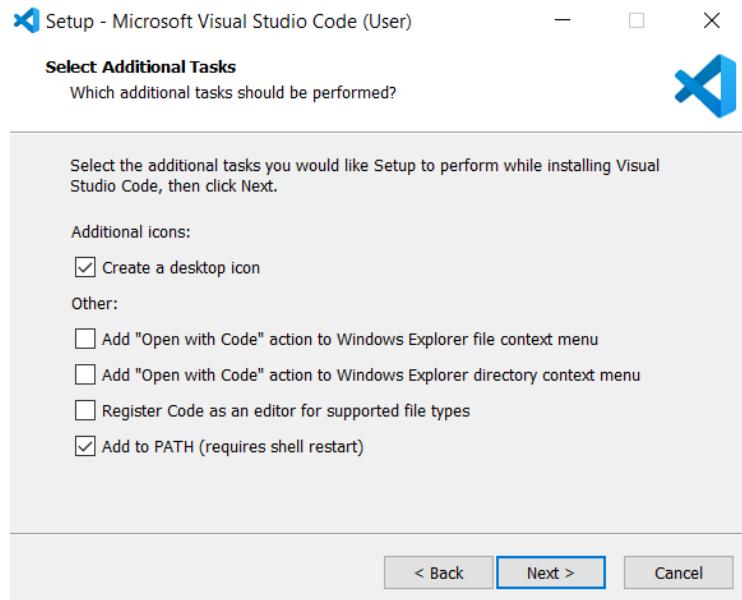
Gambar 4. 4 Proses Instalasi Visual Studio Code

5. Selanjutnya klik next



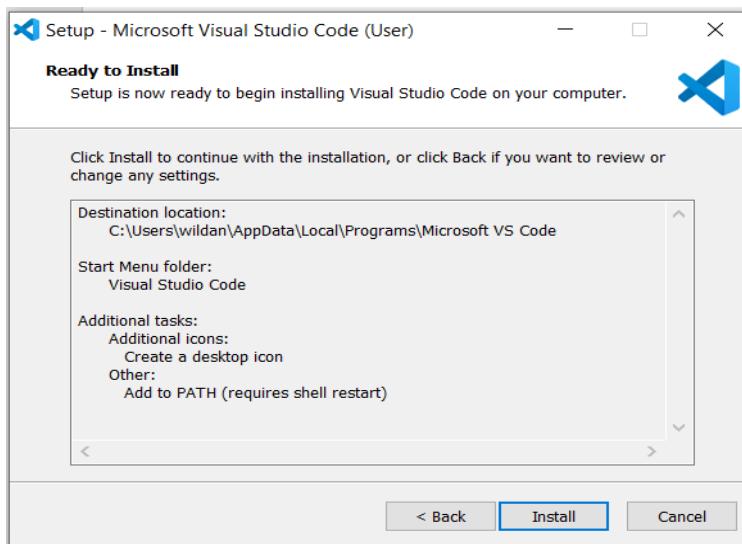
Gambar 4. 5 Proses Instalasi Visual Studio Code

6. Selanjutnya klik next



Gambar 4. 6 Proses Instalasi Visual Studio Code

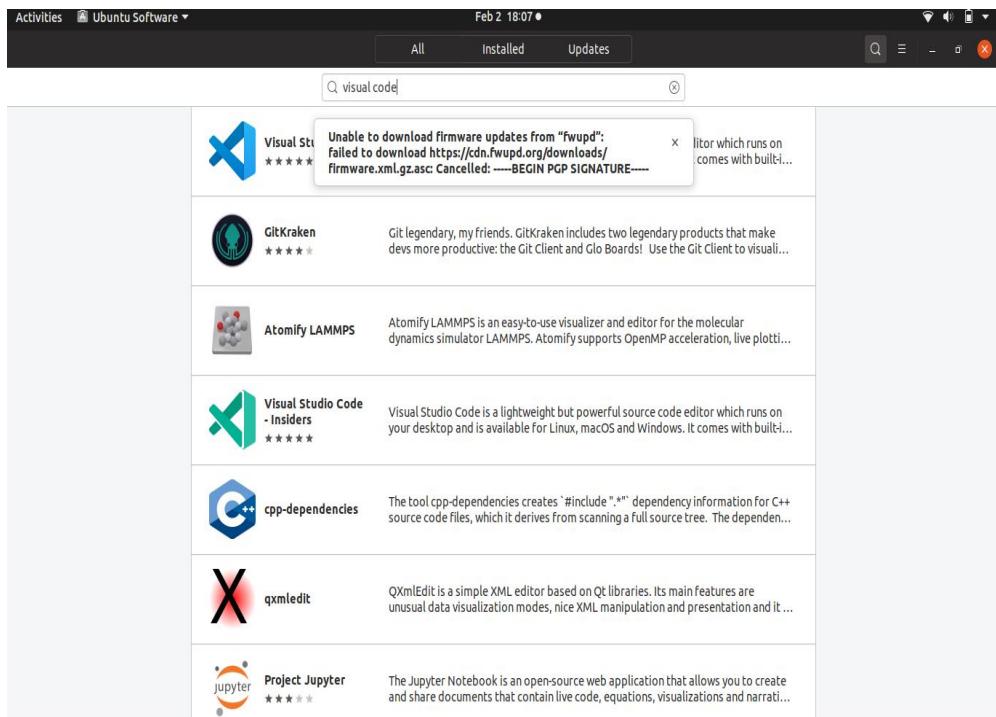
7. Tekan Install lalu Tunggu sampai proses instalasi selesai, Setelah proses ini selesai maka Visual Studio Code telah bisa digunakan.



Gambar 4. 7 Proses Instalasi Visual Studio Code

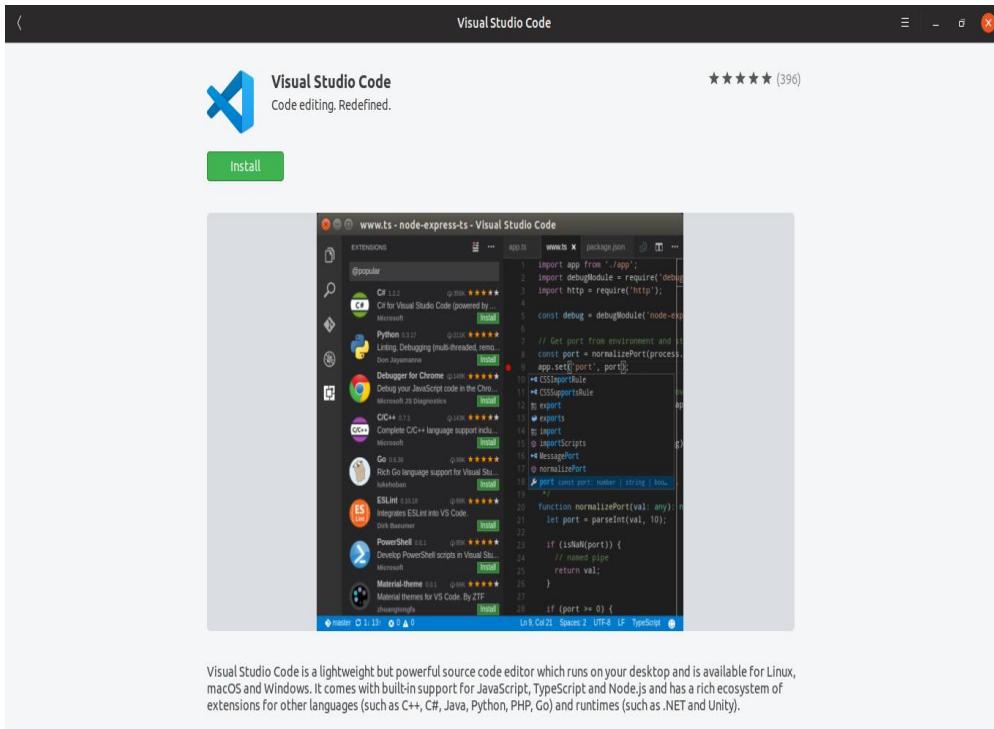
#### 4.1.1.1 Instalasi Visual Studio Code linux

1. Buka software ubuntu didesktop, lalu masukan software/aplikasi visual code studio, seperti pada gambar berikut;

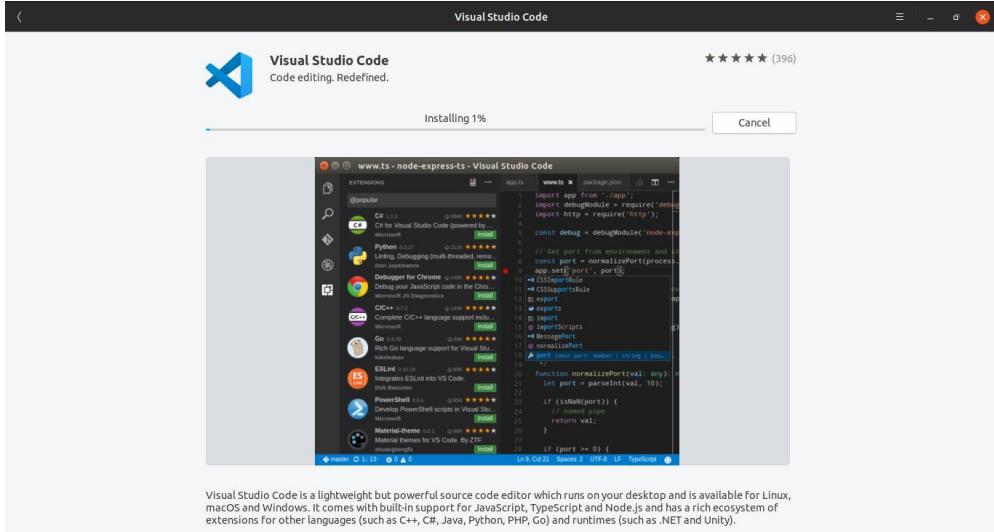


Gambar 4. 8 Proses Instalasi Visual Studio Code

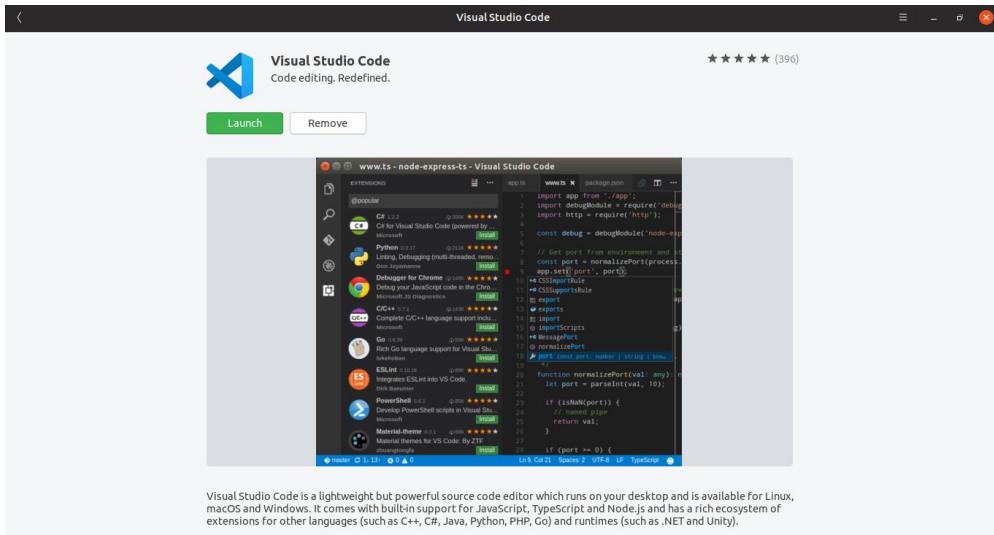
## 2. Lalu pilih visual code studio, seperti pada gambar berikut;



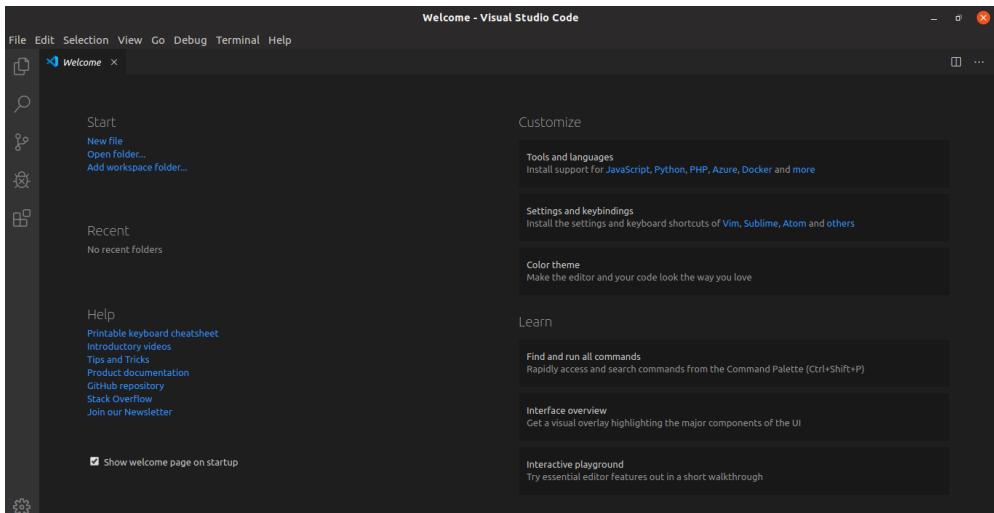
## 3. Lalu click Install dan tunggu hingga proses install selesai, seperti pada gambar berikut;



4. Proses install selesai, seperti pada gambar berikut;

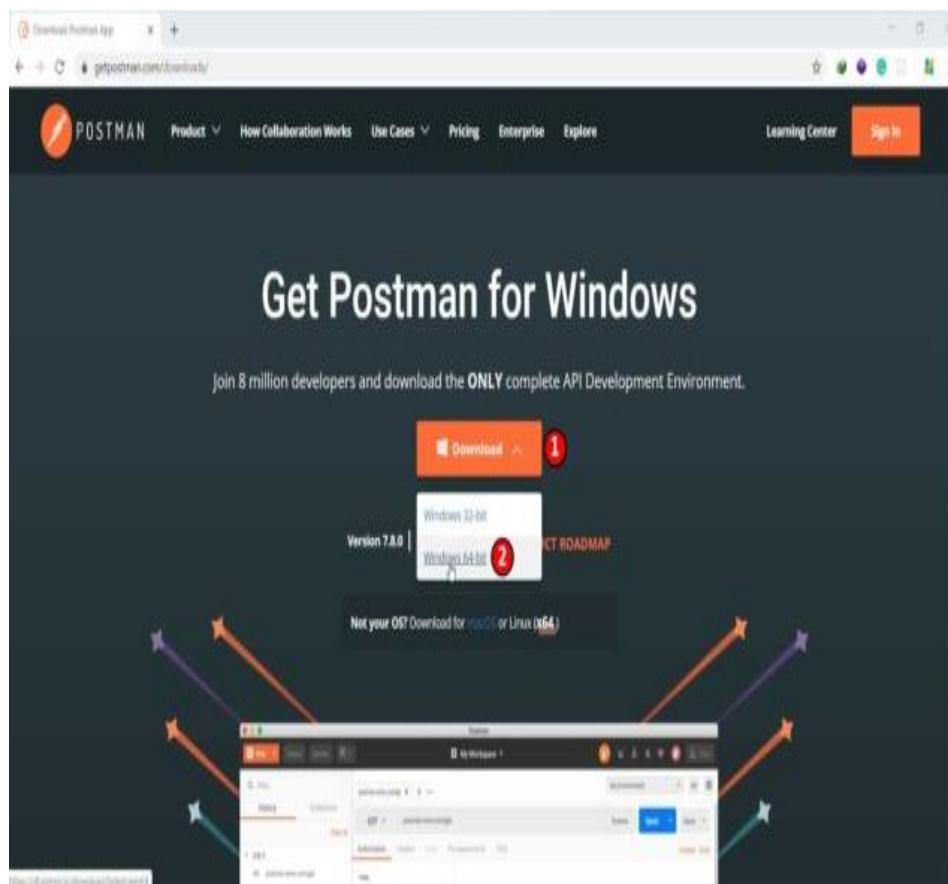


5. Click launch untuk membuka aplikasi visual code , seperti pada gambar berikut;

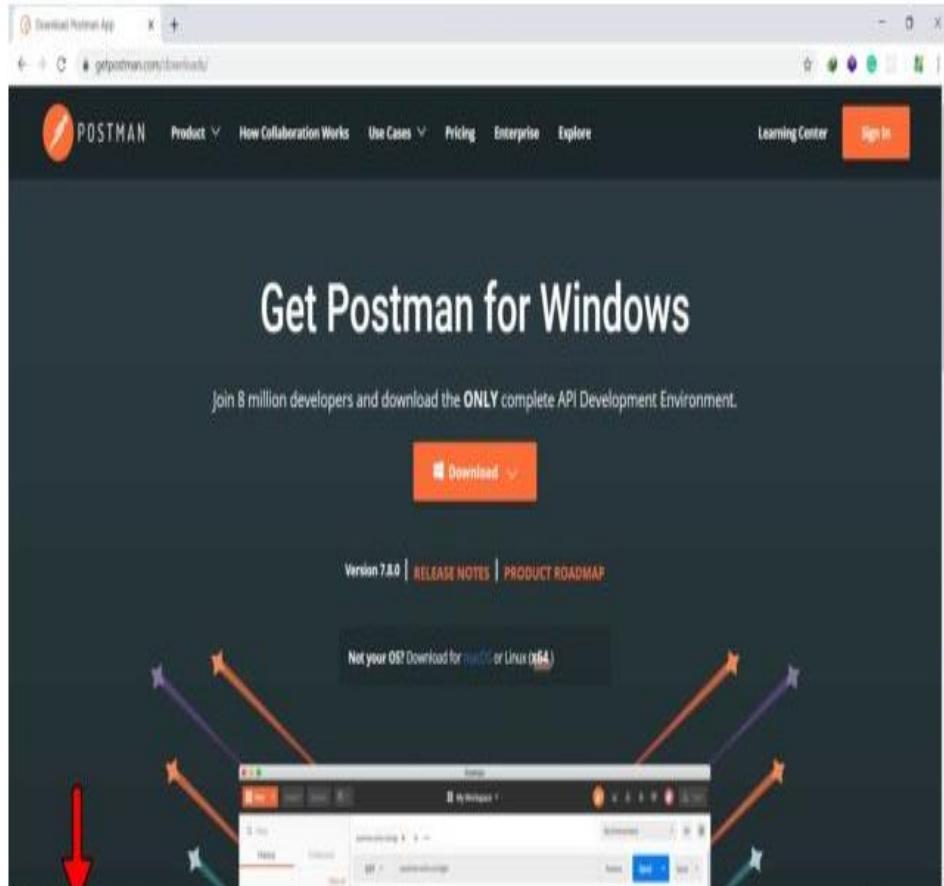


#### 4.1.1.2 Install Postman di Windows

1. Buka [Browser Google Chrome](#) Anda dan kunjungi alamat berikut: [getpostman.com/downloads](https://getpostman.com/downloads)
2. Setelah Anda berada di situs web Postman, langkah selanjutnya sentuh kursor Anda pada button Download (“1”), setelah itu pilih installer postman sesuai tipe dari Sistem Operasi Windows 10 Anda.

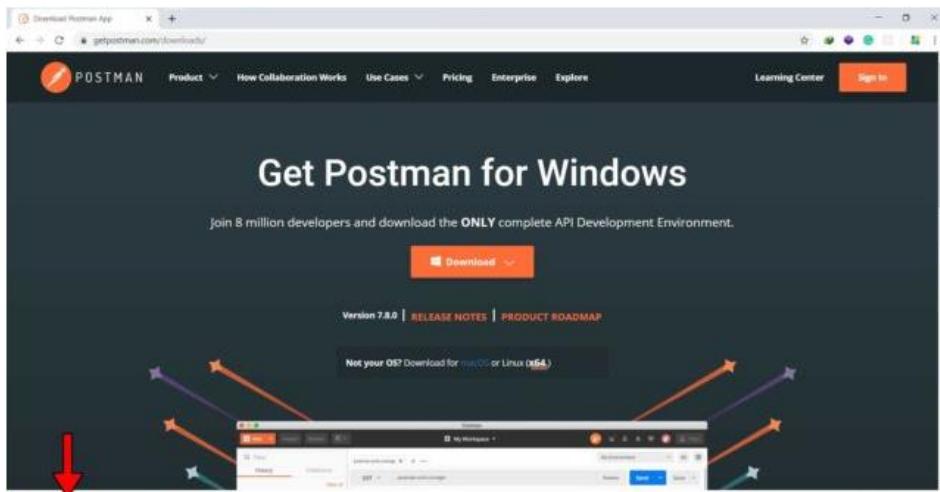


3. Setelah Anda mengklik “link download” sesuai tipe Sistem Operasi Anda, Anda akan mendapatkan proses download di bagian pojok kiri bawah browser Google Chrome Anda.

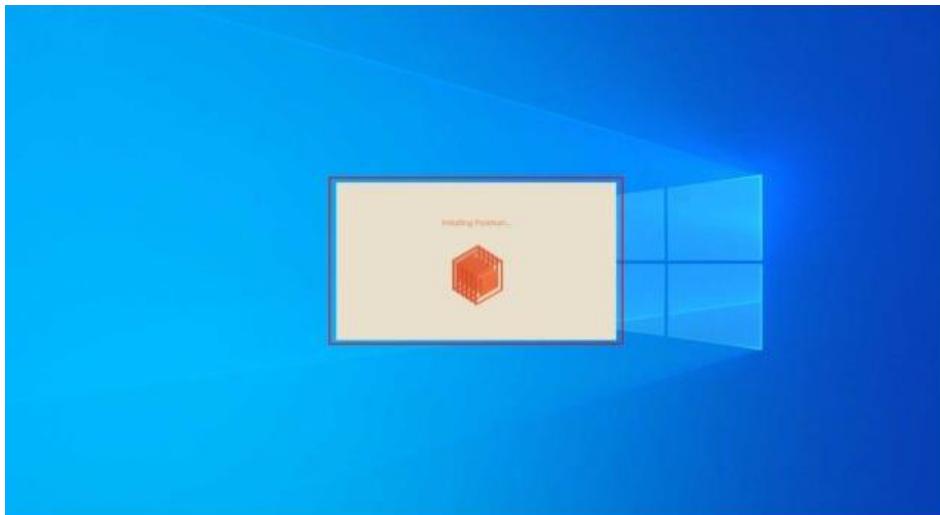


Untuk melanjutkan, tunggu sampai proses download selesai.

4. Setelah proses download selesai, untuk melanjutkan klik Intaller atau hasil download dari Postman tersebut.

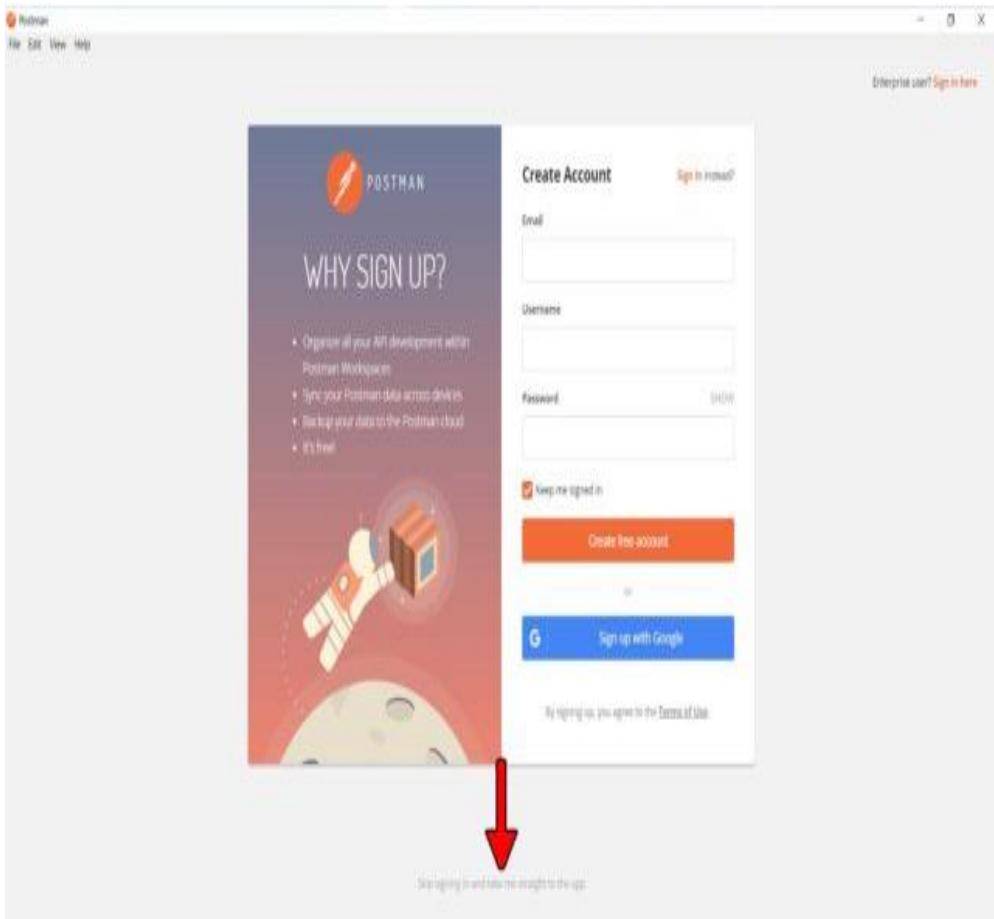


5. Setelah Anda mengklik “Installer” Postman tersebut, Anda akan dibawa ke tampilan penginstallan.

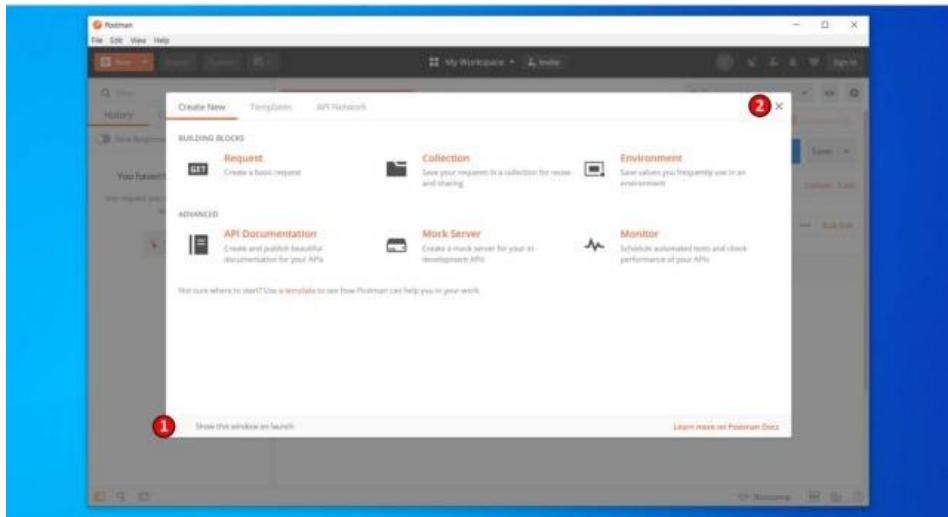


Untuk melanjutkan, tunggu sampai proses penginstallan selesai.

- 6. Setelah proses penginstallan selesai, Postman akan langsung terbuka.  
Untuk melanjutkan, pada bagian bawah silahkan Anda **klik Skip....****

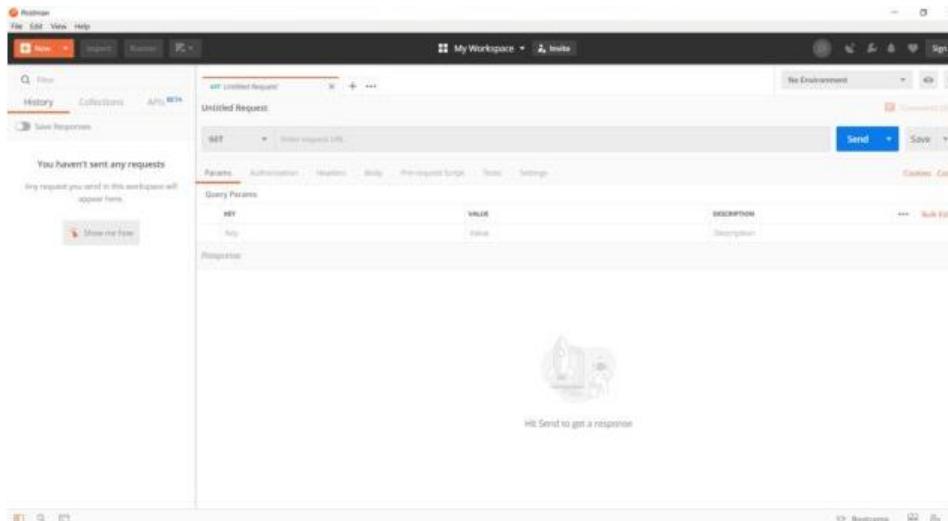


7. Setelah proses penginstallan selesai, Postman akan langsung terbuka.  
Untuk melanjutkan, pada bagian bawah silahkan Anda klik Skip....



Untuk melanjutkan, silahkan Anda hilangkan tanda ceklist (“1”),  
setelah itu klik tanda “x” (“2”).

- Setelah Anda mengklik tanda “x”, Anda akan dibawa ke tampilan dari Postman Anda dan siap untuk Anda gunakan.



- Selesai

Setelah Anda mengikuti langkah-langkah di atas, maka sekarang Anda telah berhasil menginstall Postman di Windows 10 Anda.

#### 4.1.1.3 Install Postman di Linux

```
sudo tar -xzf postman.tar.gz -C /opt
```

- Unpack file installer yang sebelumnya telah anda download (saya unpack ke folder opt), seperti pada gambar dibawah ini;

```
sudo ln -s /opt/Postman/Postman /usr/bin/postman
```

2. Kemudian lakukan symbolic link, ini berguna agar postman bisa di panggil via CLI, seperti pada gambar dibawa ini;
3. Lalu, buat shortcut untuk bisa di pin panel ubuntu atau muncul dalam pencarian aplikasi anda, seperti pada gambar berikut;

```
cat > ~/.local/share/applications/postman.desktop <<EOL
[Desktop Entry]
Encoding=UTF-8
Name=Postman
Exec=postman
Icon=/opt/Postman/resources/app/assets/icon.png
Terminal=false
Type=Application
Categories=Development;
EOL
```

Selesai, kini anda bisa memulai menggunakan postman.

4. “apt-get install postman”, Well, dengan command dibawah ini anda bisa install postman sekaligus update postman apabila terdapat notif update.

```
bash <(wget -O - https://raw.githubusercontent.com/CreatorB/Postman/master/apt-get-install-postma
n.sh)
```

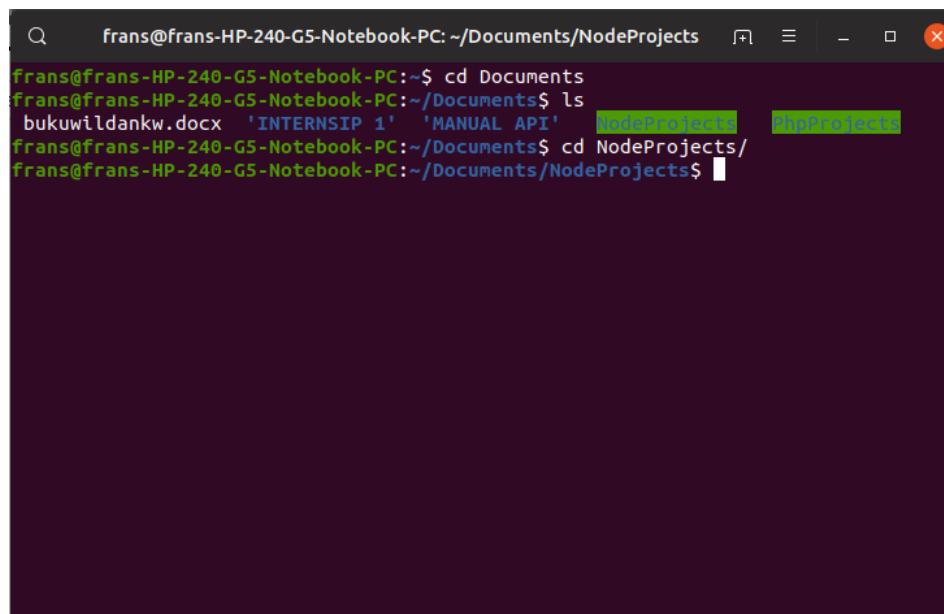
5. Berikut SS (screenshot) instalasi postman via apt-get-install-postman diatas.

```
2018-03-04 21:19:39 (158 MB/s) - written to stdout [867/867]
Downloading Postman ... ↙ Please wait ...
Installing to opt...
[sudo] password for anonymous: ↙ how long?
Creating symbolic link...
Creating .desktop file...
Installation completed, Lets Rock!
21:38:46 ➤ tools
```

#### 4.1.1.4 Install Framework Express, nodejs dan Npm

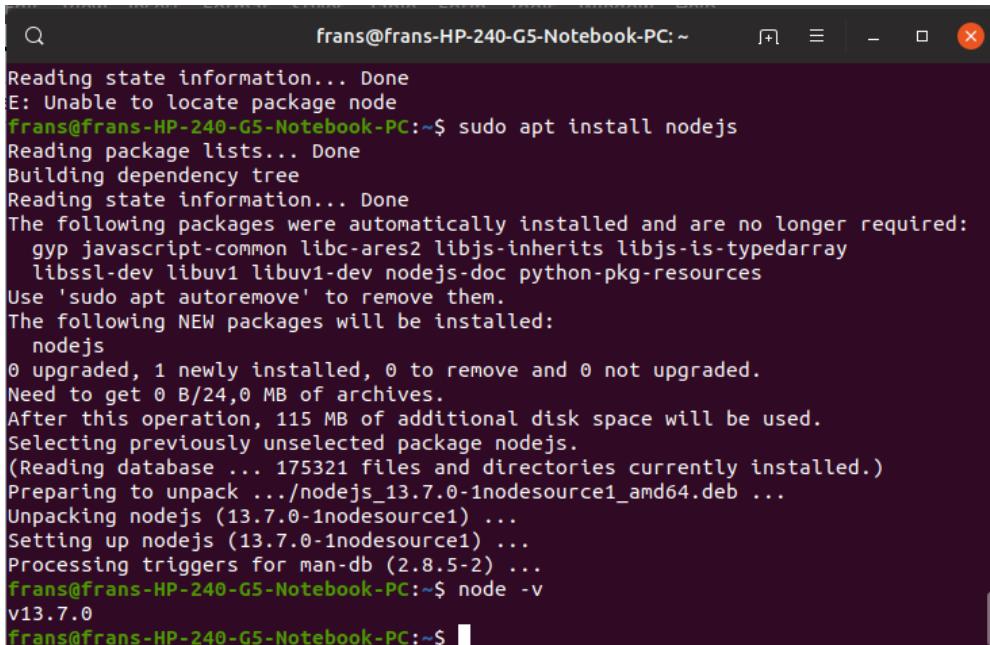
Baik. untuk memulai install express framework kalian harus menentukan direktori penyimpanan file yang nantinya akan di download secara otomatis. Berarti kalian harus terkoneksi dengan internet.

1. Buat folder baru pada direktori D:/
2. Berinama folder tersebut menjadi **NodeProjects**
3. Buka CMD pada Linux Ubuntu, dengan cara tekan keyboard Logowindows+R.
4. Masuk direktori D:/**NodeProjects** seperti gambar di bawah ini



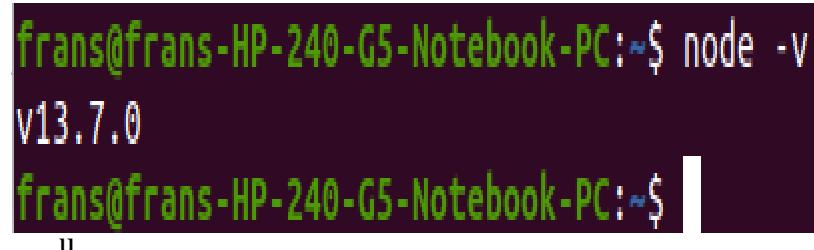
```
frans@frans-HP-240-G5-Notebook-PC:~/Documents/NodeProjects$ cd Documents
frans@frans-HP-240-G5-Notebook-PC:~/Documents$ ls
bukuwildankw.docx 'INTERNSIP 1' 'MANUAL API' NodeProjects PhpProjects
frans@frans-HP-240-G5-Notebook-PC:~/Documents$ cd NodeProjects/
frans@frans-HP-240-G5-Notebook-PC:~/Documents/NodeProjects$
```

5. Tuliskan perintah : *Sudo apt install nodejs*



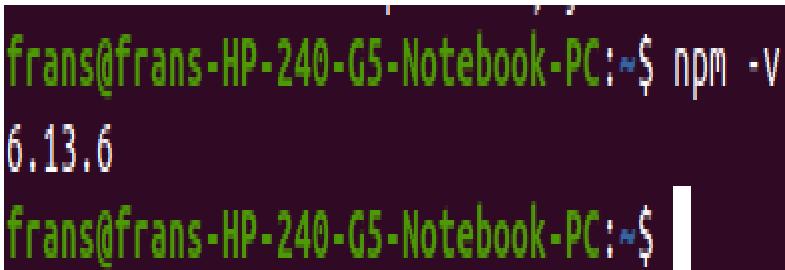
```
frans@frans-HP-240-G5-Notebook-PC:~$ sudo apt install nodejs
Reading state information... Done
E: Unable to locate package node
frans@frans-HP-240-G5-Notebook-PC:~$ sudo apt install nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gyp javascript-common libc-ares2 libjs-inherits libjs-is-typedarray
  libssl-dev libuv1 libuv1-dev nodejs-doc python-pkg-resources
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/24,0 MB of archives.
After this operation, 115 MB of additional disk space will be used.
Selecting previously unselected package nodejs.
(Reading database ... 175321 files and directories currently installed.)
Preparing to unpack .../nodejs_13.7.0-1nodesource1_amd64.deb ...
Unpacking nodejs (13.7.0-1nodesource1) ...
Setting up nodejs (13.7.0-1nodesource1) ...
Processing triggers for man-db (2.8.5-2) ...
frans@frans-HP-240-G5-Notebook-PC:~$ node -v
v13.7.0
frans@frans-HP-240-G5-Notebook-PC:~$
```

6. Lalu ketikan Node -v untuk melihat version nodejs yang digunakan



```
frans@frans-HP-240-G5-Notebook-PC:~$ node -v
v13.7.0
frans@frans-HP-240-G5-Notebook-PC:~$
```

ketikan Npm -v untuk melihat version packages npm yang digunakan, berikut code diterminal;



```
frans@frans-HP-240-G5-Notebook-PC:~$ npm -v
6.13.6
frans@frans-HP-240-G5-Notebook-PC:~$
```

8. Berikut adalah format dari framework express.js

```
    < components / billings      •
      JS billing.js              M
      JS billingAPI.js           M
      JS billingController.js    M
      JS billingDAL.js           M
      JS billingHelper.js         M
      JS billingService.js        M
      JS billingTestUnit.js       M
      JS billingValidation.js    M
    > config                      •
    > coverage
    > libraries                   •
    > middlewares                 •
    > node modules
```

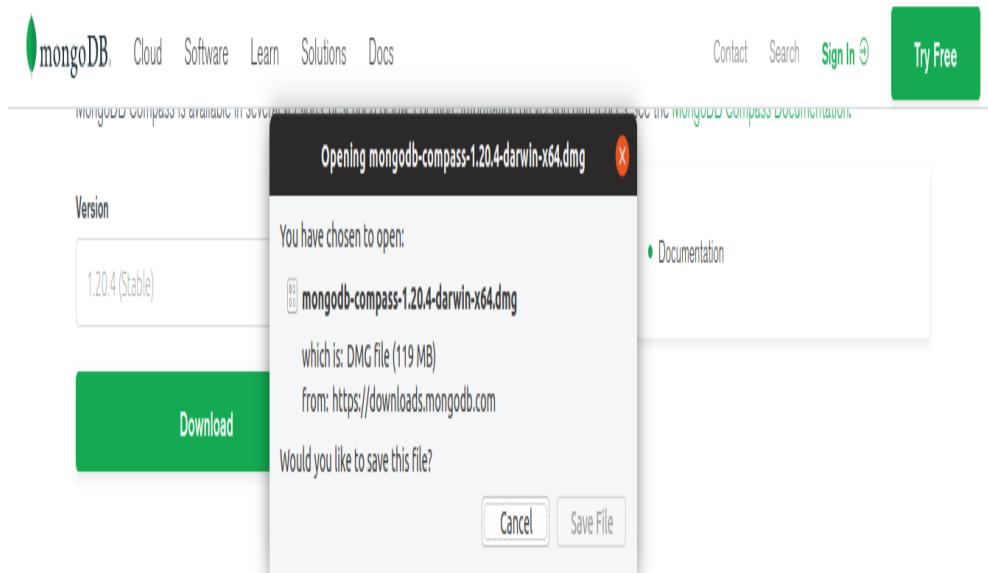
#### 4.1.1.5 Install Mongodb Compass

Langkah pertama yang dilakukan saat untuk download yaitu mengunjungi website <https://www.mongodb.com/download-center/compass>.

1. Kunjungi website saat ingin mendownload, gambar seperti berikut ini;

The screenshot shows a web browser window with the URL <https://www.mongodb.com/download-center/compass> in the address bar. The page header includes the MongoDB logo, navigation links for Cloud, Software, Learn, Solutions, and Docs, and a green 'Try Free' button. Below the header, there are tabs for Cloud, Server, and Tools, with 'Tools' being the active tab. The main content area is titled 'MongoDB Compass' and describes it as the GUI for MongoDB. It mentions that MongoDB Compass allows users to make smarter decisions about document structure, querying, indexing, document validation, and more. Commercial subscriptions include technical support for MongoDB Compass. A note states that MongoDB Compass is available in several versions, with more information provided in the MongoDB Compass Documentation. There are dropdown menus for 'Version' (set to 1.20.4 (Stable)) and 'Platforms' (set to OS X 64-bit (10.10+)). A large green 'Download' button is prominently displayed. At the bottom, there are links for 'Compass', 'Readonly Edition', 'Isolated Edition', and 'Community Edition'.

2. Click download untuk memulai, gambar seperti berikut ini;



#### Compass

The full version of MongoDB Compass, with all features and capabilities.

#### Readonly Edition

This version is limited strictly to read operations, with all write and delete capabilities removed.

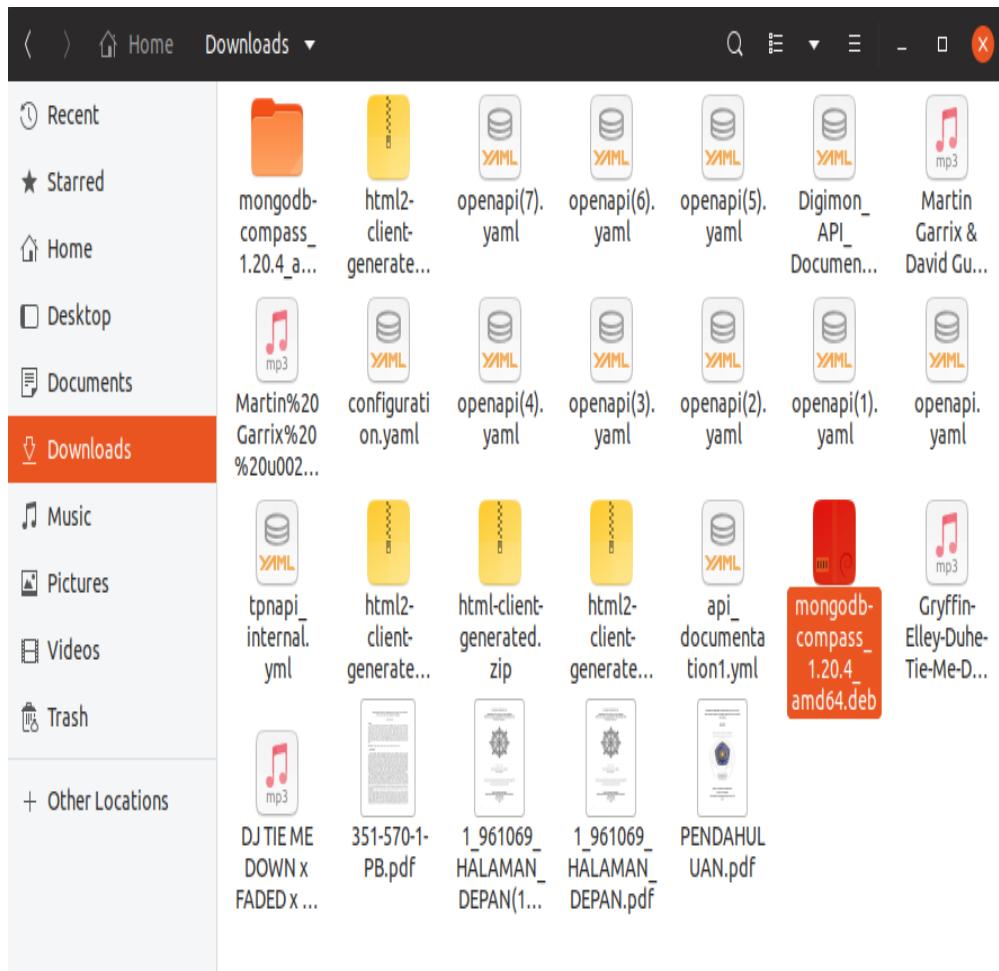
#### Isolated Edition

This version disables all network connections except the connection to the MongoDB instance.

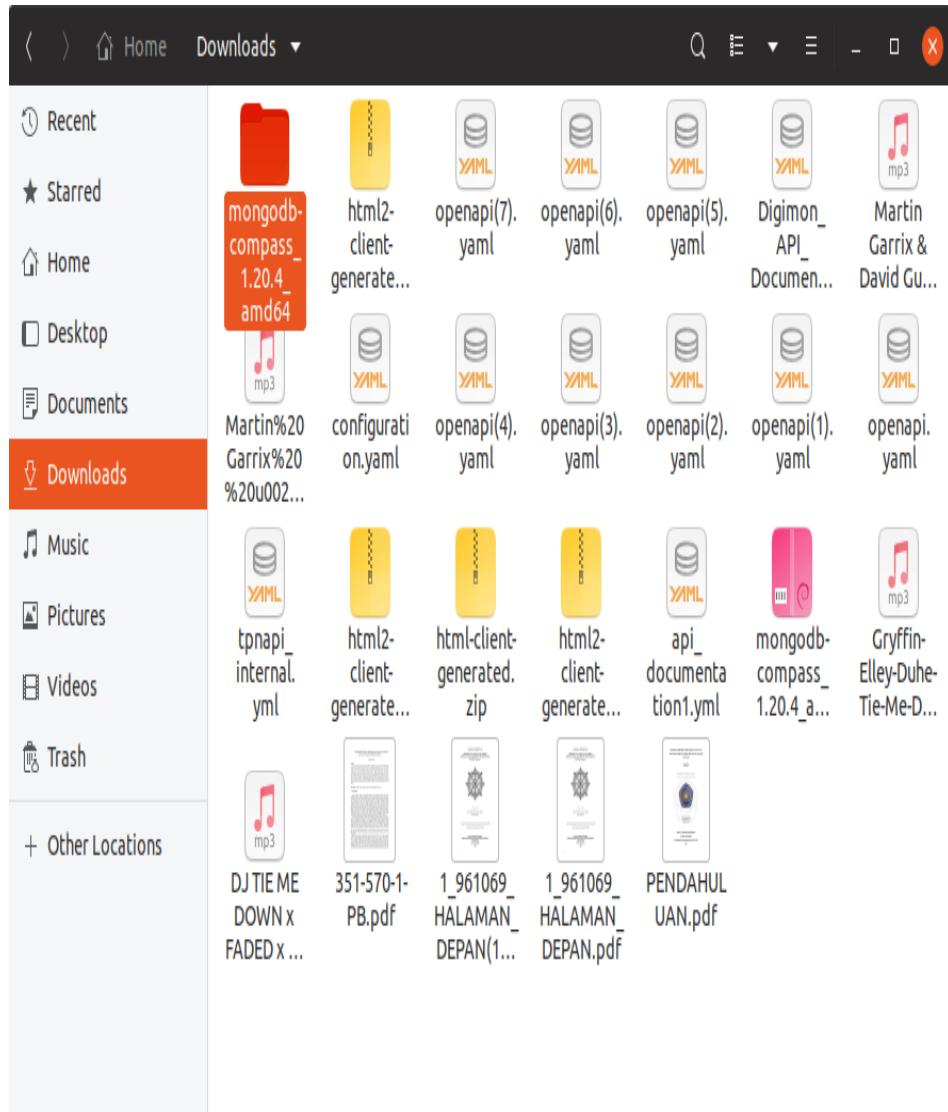
#### Community Edition

This version is distributed with Community Server downloads and includes a subset of the features of Compass.

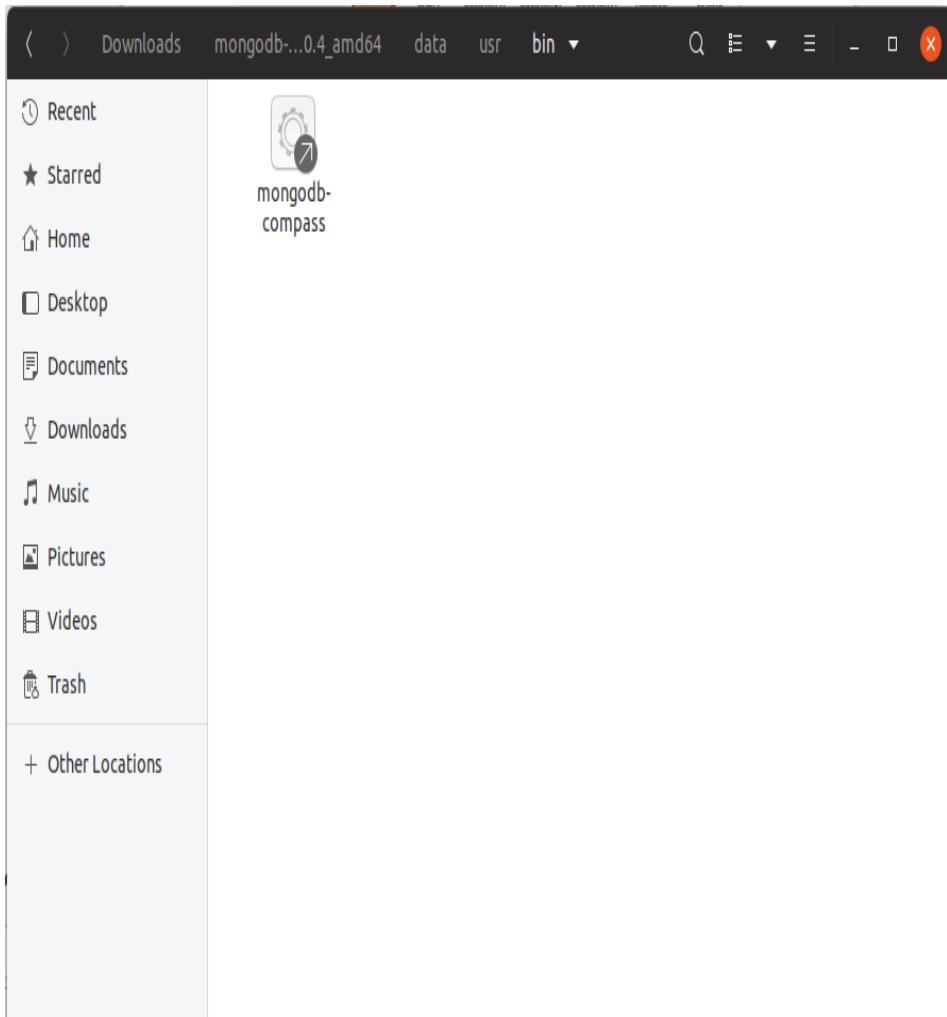
3. Kemudian buka directory download untuk memastikan bahwa file sudah ter-download.



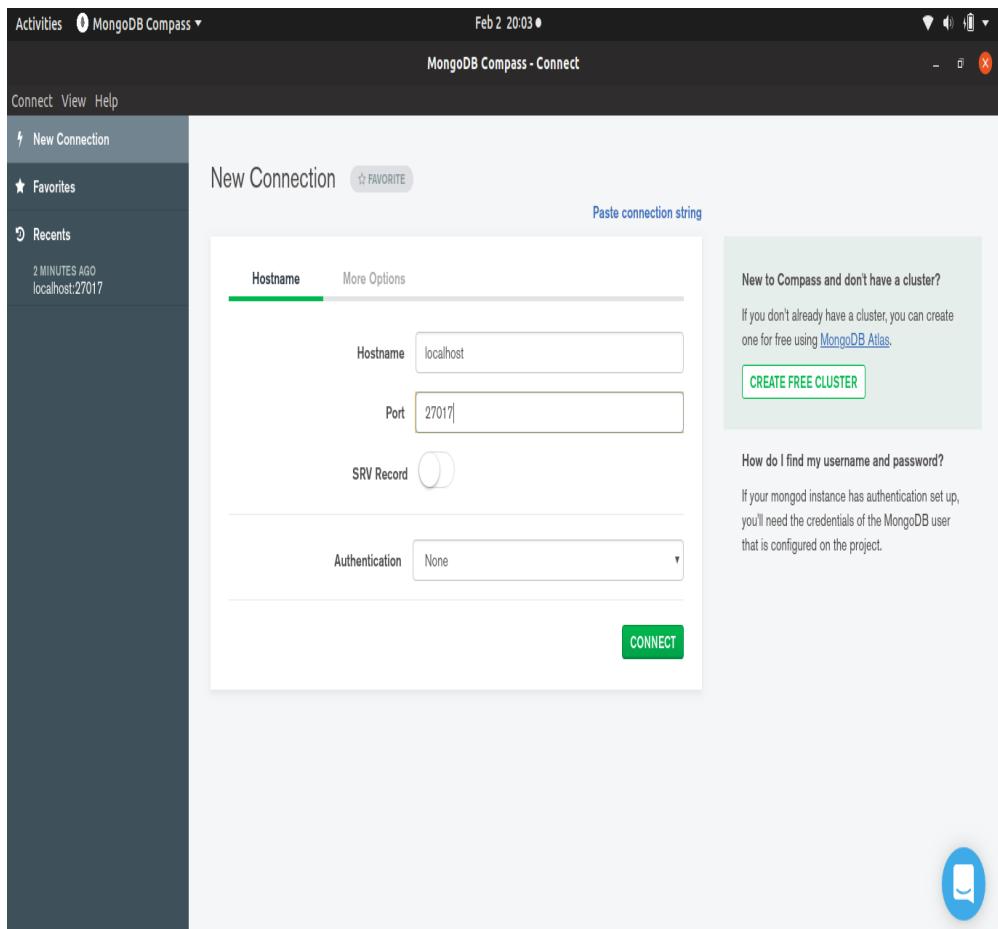
#### 4. Extract file .deb



5. Lalu masuk kedalam folder,data/usr/bin, seperti gambar berikut ini;



6. Click 2 kali pada file aplikasi mongodb compass, seperti pada gambar berikut ini



7. Lalu isi hostname: localhost dan port: 27017 lalu tekan connect. Maka akan langsung masuk kedalam database, seperti pada gambar berikut ini;

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'Local' selected, showing 'HOST: localhost:27017', 'CLUSTER: Standalone', and 'EDITION: MongoDB 3.6.8 Community'. Below that is a search bar and a list of databases: 'admin', 'config', and 'local'. The main area is titled 'MongoDB Compass - localhost:27017' and has tabs for 'Databases' and 'Performance'. Under 'Databases', there's a 'CREATE DATABASE' button. A table lists the databases with columns: Database Name, Storage Size, Collections, and Indexes. The 'admin' database has 16.4KB storage, 0 collections, and 1 index. The 'config' database has 16.4KB storage, 0 collections, and 2 indexes. The 'local' database has 36.9KB storage, 1 collection, and 1 index. Each database row has a trash icon in the last column.

Database Name	Storage Size	Collections	Indexes
admin	16.4KB	0	1
config	16.4KB	0	2
local	36.9KB	1	1

## BAB V TAHAPAN PEMBUATAN APLIKASI

### 5.1 BIN/WWW

#### a) File WWW

www merupakan file nodejs router configurasi yang digunakan untuk menjalankan aplikasi serta configurasi untuk database. Untuk penggunaan port di file sendiri harus berbeda disetiap fitur/services, berikut merupakan gambar bin/www;



#### b) Source Code

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('test:server');
var http = require('http');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
console.log('connected to port '+port);
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);
```

```
/**  
 * Listen on provided port, on all network interfaces.  
 */  
  
server.listen(port);  
server.on('error', onError);  
server.on('listening', onListening);  
  
/**  
 * Normalize a port into a number, string, or false.  
 */  
  
function normalizePort(val) {  
  var port = parseInt(val, 10);  
  
  if (isNaN(port)) {  
    // named pipe  
    return val;  
  }  
  
  if (port >= 0) {  
    // port number  
    return port;  
  }  
  
  return false;  
}  
  
/**  
 * Event listener for HTTP server "error" event.  
 */  
  
function onError(error) {  
  if (error.syscall !== 'listen') {  
    throw error;  
  }  
  
  var bind = typeof port === 'string'  
    ? 'Pipe ' + port  
    : 'Port ' + port;
```

```

// handle specific listen errors with friendly messages
switch (error.code) {
  case 'EACCES':
    console.error(bind + ' requires elevated privileges');
    process.exit(1);
    break;
  case 'EADDRINUSE':
    console.error(bind + ' is already in use');
    process.exit(1);
    break;
  default:
    throw error;
}
}

/***
 * Event listener for HTTP server "listening" event.
 */

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}

```

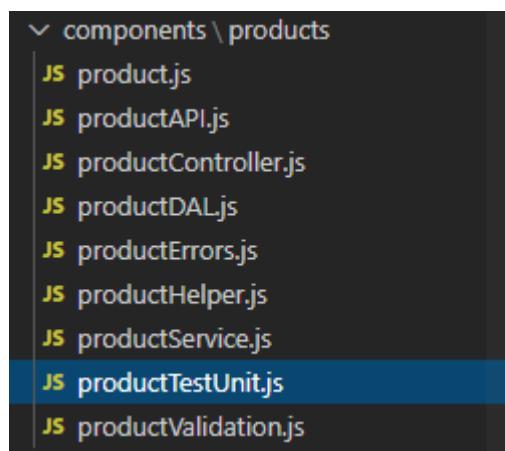
## 5.2 COMPONENTS

Component/ Merupakan file file yang didalamnya terdapat function mongoose yang ketika dijalankan akan membentuk suatu table unstructure yang mempunya format xml, function router ketika dijalankan sebagai function API, fungsi controller menjadi function yang paling penting karena menjadi karena menjadi wadah untuk memanggil function services/logic program, function services yaitu kumpulan function logic, function DAL sama hal nya seperti models yaitu memanipulasi data dalam database yang digunakan seperti mengambilkan data yang dinginikan serta memanipulasi

data yang diinginkan, function helper yang digunakan sebagai sebuah function tambahan misalkan generate product code, function untuk melakukan unit testing code, validation code sebagai function untuk memvalidasi schema mongoose.

### 1. File Components

Berikut adalah file yang terdapat didalam folder components products



### 2. Source Code components

#### a. Product.js

Berikut merupakan source code untuk schema mongoose product.

```
const mongoose = require('mongoose');

let mongooseSchema = new mongoose.Schema({
  name: {type:String,index:{collation:{locale:'en',strength:2}}},
  categories: [String],
  description: String,
  tags: [],
  image_logo: String,
  founder: [],
  established_on: Date,
  status_product:String, //waiting|approved|rejected
  npwp: String,
  config:{
    po_number_periode: String, //daily|monthly|yearly
  }
})
```

```
administration_fee: Number,  
unique_code_type:String, //discount|addition  
invoice_signer:String, //user_id tpn  
transaction_type:String, //cash basis atau accrual basis  
akun_pendapatan:String, //diberikan oleh snba  
akun_piutang: String, //diberikan oleh snba  
akun_bank: String, //akun tempat uang masuk di dds. diberikan oleh snb  
a  
    product_ownership: String //milik mitra atau milik telkom  
},  
contact:{  
    website: String,  
    phone: String,  
    email: String,  
    address: String,  
},  
is_ready_for_market: Boolean,  
companion:{  
    organization_company_id: String, //tpn organization id  
    division_id: String, // tpn division id  
    companion_id: String //tpn companion id  
},  
partnership:{  
    company_id: String,  
    partnership_id: String,  
    contract_id: String,  
    business_scheme: String,  
},  
bank:{  
    bank_account_owner: String,  
    bank_account_number: String,  
    bank_name: String,  
    bank_branch_office: String  
},  
features:[{  
    image:String,  
    title:String,  
    description:String  
}],  
benefit:[{  
    image:String,  
    title:String,  
    description:String  
}]]
```

```
        title:String,
        description:String
    }],
    gateway:mongoose.Schema.Types.Mixed,
    product_code:String,
    deleted:{
        at:Date,
        by:String,
        app_id:String,
        reference_id:String
    },
    created:{
        at:Date,
        by:String,
        app_id:String,
        reference_id:String
    },
    updated:{
        at:Date,
        by:String,
        app_id:String,
        reference_id:String
    },
    approved:{
        message:String,
        at:Date,
        by:String,
        app_id:String,
        reference_id:String
    },
    rejected:{
        message:String,
        at:Date,
        by:String,
        app_id:String,
        reference_id:String
    },
    meta:mongoose.Schema.Types.Mixed
});
```

```
exports.Product = mongoose.model('Product', mongooseSchema);
```

### b. ProductAPI.js

Berikut merupakan source code untuk router API product.

```
const express = require('express');
const router = express.Router();
const ProductController = require('./productController');
const validation = require('../middlewares/validation');
const auth = require('../middlewares/auth');
const validationSchema = require('./productValidation');

//GET PRODUCT DATA
router.get('/',function(req,res,next){
    res.locals.function_id = "products-index"
    next();
},
auth,
ProductController.index);

//GET COUNT
router.get('/count',ProductController.count);

//STORE PRODUCT DATA
router.post('/',function(req,res,next){
    res.locals.joiSchema = validationSchema.createSchema;
    res.locals.function_id = "products-store"
    next();
},
auth,
validation,
ProductController.store);

//GET ONE PRODUCT DATA BY ID
router.get('/:id', function(req,res,next){
    res.locals.function_id = "products-show"
    next();
},
auth,ProductController.show);
```

```
//UPDATE PRODUCT DATA
router.put('/:id',function(req,res,next){
    res.locals.joiSchema = validationSchema.updateSchema;
    res.locals.function_id = "products-update"
    next();
},
auth,
validation,
ProductController.update);

//DELETE PRODUCT DATA (IS_DELETED FLAG)
router.post('/:id/delete',function(req,res,next){
    res.locals.joiSchema = validationSchema.deleteSchema;
    res.locals.function_id = "products-delete"
    next();
},
auth,
validation,
ProductController.delete);

//CHANGE PRODUCT STATUS
router.post('/:id/approve', function(req,res,next){
    res.locals.joiSchema = validationSchema.approveSchema;
    res.locals.function_id = "products-approve"
    next();
},
auth,
validation,
ProductController.approve);

router.post('/:id/reject', function(req,res,next){
    res.locals.joiSchema = validationSchema.rejectSchema;
    res.locals.function_id = "products-reject"
    next();
},
auth,
validation,
ProductController.reject);

router.post('/:id/upload', function(req,res,next){
```

```

        res.locals.joiSchema = validationSchema.rejectSchema;
        res.locals.function_id = "products-upload"
        next();
    },
    // auth,
    validation,
    ProductController.upload);

router.post('/:id/uploadfeature/:id_feature', function(req,res,next){
    res.locals.joiSchema = validationSchema.rejectSchema;
    res.locals.function_id = "products-upload-feature"
    next();
},
// auth,
validation,
ProductController.uploadFeature);

router.post('/:id/uploadbenefit/:id_benefit', function(req,res,next){
    res.locals.joiSchema = validationSchema.rejectSchema;
    res.locals.function_id = "products-upload-benefit"
    next();
},
// auth,
validation,
ProductController.uploadBenefit);

module.exports = router;

```

### c. ProductController.js

Berikut merupakan source code untuk controller product.

```

const productService = require('./productService');
const respond = require('../libraries/respond');

/**
 * Get all item
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public

```

```

*/
exports.index = async(req, res) => {

    const getResult = await productService.get(req.query);
    let meta = { };
    //for datatables
    meta.total = getResult.total;
    meta.filtered = getResult.filtered;
    meta.currrpage = (req.query.currrpage)?parseInt(req.query.currrpage):0;
    meta.perpage = (req.query.perpage)?parseInt(req.query.perpage):10;
    meta.totalpage = Math.ceil(meta.filtered/meta.perpage);

    return respond.resSuccessData(res,undefined getResult.data,meta);
}

/***
* Get Count
* @param {Object} req express request object
* @param {Object} res express response object
* @api public
*/
exports.count = async(req, res) => {
    data = await productService.count(req.query);

    return respond.resSuccessData(res,undefined,data,{ });
}

/***
* store item to db
* @param {Object} req express request object
* @param {Object} res express response object
* @api public
*/
exports.store = async(req, res) => {
    //save consumer id
    try{
        if(!req.body.created){
            req.body.created = { };
        }
        req.body.created.app_id = req.headers['eg-consumer-id'];
        const item = await productService.save(req.body);
    }
}

```

```

        return respond.resCreated(res,undefined,item);
    }catch(e){
        return respond.resBadRequest(res,'Product Name Already Exists');
    }

}

/***
 * get item by id
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
*/
exports.show = async(req, res) => {
    const id = req.params.id;
    result = await productsService.findById(id);
    if(result){
        return respond.resSuccessData(res,undefined,result);
    }else{
        return respond.resNotFound(res,'Item not found');
    }
}
/***
 * update item by id
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
*/
exports.update = async(req, res) => {
    const id = req.params.id;
    try{
        //save consumer id
        if(!req.body.updated){
            req.body.updated = { };
        }
        req.body.updated.app_id = req.headers['eg-consumer-id'];
        const result = await productsService.update(id,req.body);
        return respond.resUpdated(res,undefined,result);
    } catch(e){
        return respond.resNotFound(res,e.message);
    }
}

```

```

}

/***
 * delete item by id
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
 */
exports.delete = async (req, res) => {
  const id = req.params.id;
  try{
    if(!req.body.deleted){
      req.body.deleted = { };
    }
    req.body.deleted.app_id = req.headers['eg-consumer-id'];
    const result = await productsService.deleteById(id,req.body);
    return respond.resSuccessData(res,'deleted',result);
  } catch(e){
    return respond.resNotFound(res,e.message);
  }
}

/***
 * delete item by id
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
 */
exports.approve = async (req, res) => {
  const id = req.params.id;
  try{
    if(!req.body.approved){
      req.body.approved = { };
    }
    req.body.approved.app_id = req.headers['eg-consumer-id'];
    const result = await productsService.approval(id,req.body,'approve');
    return respond.resSuccessData(res,undefined,result);
  } catch(e){
    return respond.resNotFound(res,e.message);
  }
}

```

```

/***
 * delete item by id
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
 */
exports.reject = async (req, res) => {
  const id = req.params.id;
  try{
    if(!req.body.rejected){
      req.body.rejected = { };
    }
    req.body.rejected.app_id = req.headers['eg-consumer-id'];
    const result = await productsService.approval(id,req.body,'reject');
    return respond.resSuccessData(res,undefined,result);
  } catch(e){
    return respond.resNotFound(res,e.message);
  }
}

/***
 * delete item by id
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
*/
exports.upload = async (req, res) => {
  const id = req.params.id;
  try{
    const result = await productsService.upload(id);
    return respond.resSuccessData(res,undefined,result);
  } catch(e){
    return respond.resNotFound(res,e.message);
  }
}

/***
 * delete item by id
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
*/

```

```

*/
exports.uploadFeature = async (req, res) => {
  const id = req.params.id;
  try{
    const result = await productsService.upload(id);
    return respond.resSuccessData(res,undefined,result);
  } catch(e){
    return respond.resNotFound(res,e.message);
  }
}

/**
 * delete item by id
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
 */
exports.uploadBenefit = async (req, res) => {
  const id = req.params.id;
  try{
    const result = await productsService.upload(id);
    return respond.resSuccessData(res,undefined,result);
  } catch(e){
    return respond.resNotFound(res,e.message);
  }
}

```

#### d. ProductDAL.js

Berikut adalah source code untuk Product DAL.

```

const {Product} = require('./product');
const utils = require('../..../libraries/utils');

/**
 * Get all item
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
 */
exports.get = async(query) => {

```

```

let perpage = (query.perpage)?parseInt(query.perpage):10;
let currpage = (query.currrpage)?parseInt(query.currrpage):0;
query['sort-created.at'] = (query['sort-created.at'])?query['sort-created.at']:-1;

//merge array of object, the result will be merged in array1
let arrayOr = utils.getQueryOr(query,'likeor-','like');
let arrayOr2 = utils.getQueryOr(query,'filteror-');
Array.prototype.push.apply(arrayOr,arrayOr2);
//filter empty object
arrayOr = arrayOr.filter(value=> Object.keys(value).length >0);
//if empty, add
if(arrayOr.length == 0){
    arrayOr.push({ });
}

let result = await Product.aggregate([
    {
        $match:{ 
            "$or":arrayOr,
            ...utils.getQuery(query,'filterin-','in'),
            ...utils.getQuery(query,'filter-'),
            ...utils.getQuery(query,'exists-','exists'),
            ...utils.getQuery(query,'filternum-','number'),
            ...utils.getQuery(query,'filterobjid-','objid'),
            ...utils.getQuery(query,'bool-','bool')
            ,...utils.getQuery(query,'like-','like')
        }
        ,...utils.betweenDate(utils.getQuery(query,'between-'))
    }
])
.collation({locale: "en",strength:2 })
.sort(utils.getQuery(query,'sort-'))
.skip(currpage*perpage)
.limit(perpage);

return result;
}
/***
* Get count all purchase number
* @api public
*/

```

```

exports.getTotal = async() => {
  let result = await Product.countDocuments({ });
  return result;
}

/**
 * Get count all purchase number based on filter
 * @param {object} query query string
 * @api public
 */
exports.getTotalFiltered = async(query) => {
  //merge array of object, the result will be merged in array1
  let arrayOr = utils.getQueryOr(query,'likeor-','like');
  let arrayOr2 = utils.getQueryOr(query,'filteror-');
  Array.prototype.push.apply(arrayOr,arrayOr2);
  //filter empty object
  arrayOr = arrayOr.filter(value=> Object.keys(value).length >0);
  //if empty, add
  if(arrayOr.length == 0){
    arrayOr.push({ });
  }
  let result = await Product.aggregate([
    {
      $match:{
        "$or":arrayOr,
        ...utils.getQuery(query,'filterin-','in'),
        ...utils.getQuery(query,'filter-'),
        ...utils.getQuery(query,'exists-','exists'),
        ...utils.getQuery(query,'filternum-','number'),
        ...utils.getQuery(query,'filterobjid-','objid'),
        ...utils.getQuery(query,'bool-','bool')
        ....utils.getQuery(query,'like-','like')
        ....utils.betweenDate(utils.getQuery(query,'between-'))
      }
    },
    {
      $group: {_id:null, count:{$sum:1}}
    },
    {
      $project: {_id:0, count:1}
    }
  ])
}

```

```

        }
    })

    if(result[0]){
        return result[0].count;
    }else{
        return 0;
    }
}

/***
 * Get all item
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
*/
exports.getOne = async(query) => {
    let result = await Product.findOne({
        ...utils.getQuery(query,'filterin-','in'),
        ...utils.getQuery(query,'filter-'),
        ...utils.getQuery(query,'exists-','exists'),
        ...utils.getQuery(query,'filternum-','number'),
        ...utils.getQuery(query,'filterobjid-','objid')
        ...utils.getQuery(query,'bool-','bool')
        ,...utils.getQuery(query,'like-','like')
        ,...utils.betweenDate(utils.getQuery(query,'between-'))
    })
    .collation({locale: "en" })
    .sort(utils.getQuery(query,'sort-'));
    return result;
}

/***
 * Get all item
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
*/
exports.save = async(context) => {
    let item = new Product(context);
    item = await item.save();
}

```

```
        return item;
    }

    /**
     * Get all item
     * @param {Object} req express request object
     * @param {Object} res express response object
     * @api public
     */
    exports.findById = async(id) => {
        item = await Product.findOne({_id:id});
        return item;
    }

    /**
     * Get all item
     * @param {Object} req express request object
     * @param {Object} res express response object
     * @api public
     */
    exports.update = async(item) => {

        item = await item.save();

        return item;
    }
}

/**
 * Get all item
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
 */
exports.deleteById = async(id) => {
    item = await Product.findOneAndDelete({_id:id});
    return item;
}
```

e.ProductHELPER.js

Berikut adalah source code untuk Product HELPER.

```
const utils = require('../libraries/utils');
const productDAL = require('./productDAL');

const generateProductCode = async (name) =>{
    let result = "";
    let temp = "";

    if(name.length >1){
        //coba substring
        let res = await productDAL.getOne({
            'filter-product_code':name.substring(0,2).toUpperCase()
        })
        if(res){
            let found = false;
            let limit = 0;
            while(limit < 1000 && !found){
                temp = await utils.makeRandomString(2);
                res = await productDAL.getOne({
                    'filter-product_code':temp
                })
                if(res == null){
                    found = true;
                }
                limit = limit+1;
            }
            result = temp;
        }else{
            result = name.substring(0,2).toUpperCase();
        }
    }else{
        let found = false;
        let limit = 0;
        while(limit < 1000 && !found){
            temp = await utils.makeRandomString(2);
            res = await productDAL.getOne({
                'filter-product_code':temp
            })
            if(res == null){
                found = true;
            }
            limit = limit+1;
        }
    }
}
```

```

        }
        result = temp;
    }

    return result;
}

module.exports = {
    generateProductCode
}

```

#### f. ProductService.js

Berikut adalah source code untuk Product Services.

```

const productsDAL = require('./productDAL');
const gatewayHelper = require('../libraries/gateway');
const config = require('config');
const moment = require('moment');
const productHelper = require('./productHelper');

exports.get = async (query) => {

    let getResult = { };
    getResult.data = await productsDAL.get(query);
    getResult.total = await productsDAL.getTotal();
    getResult.filtered = await productsDAL.getTotalFiltered(query);

    return getResult;
}

exports.count = async (query) => {
    const data = await productsDAL.getTotalFiltered(query);

    return data;
}

/**
 * Get all item
 * @param {Object} req express request object

```

```

* @param {Object} res express response object
* @api public
*/
exports.save = async (context) => {
    //save to gateway
    const adminUrl = config.get('gateway.admin.url');
    //check if user already exists
    let gatewayUser = await gatewayHelper.getUser(adminUrl,'product-
catalog');
    if(!gatewayUser){
        gatewayUser = await gatewayHelper.createUser(adminUrl,'product-
catalog','product','catalog','iqbalzetto@gmail.com');
    }
    let appName = context.name.split(' ').join('-').toUpperCase();
    try{
        const gatewayApp = await gatewayHelper.createApp(adminUrl,'product-
catalog',appName);
        const gatewayCredential = await gatewayHelper.createCredential(admi
nUrl,gatewayApp.data.id,'key-auth',{ });
        context.gateway = {
            'user': gatewayUser.data,
            'app': gatewayApp.data,
            'credential':gatewayCredential.data
        }
    }catch(e){
        throw Error('e');
    }
    context.created.at = moment().utcOffset('7').format('YYYY-MM-
DD[T]HH:mm:ss.SSS[Z]');
    context.status_product = 'waiting';
    //set product code
    context.product_code = await productHelper.generateProductCode(context.name);
    //save item
    const item = await productsDAL.save(context);

    return item;
}

/**
* Get all item

```

```

* @param {Object} req express request object
* @param {Object} res express response object
* @api public
*/
exports.findById = async (id) => {
  const item = await productsDAL.findById(id);
  return item;
}

/**
* Get all item
* @param {Object} req express request object
* @param {Object} res express response object
* @api public
*/
exports.update = async (id,object) => {
  //get data and set new value
  let item = await productsDAL.findById(id);
  if(!item) throw new Error('Item with id:'+id+', not found!');

  for(let key in object){
    item[key] = object[key];
  }
  //check for partnership
  if(!object.partnership){
    delete item.partnership;
  }

  //reset approval
  delete item.approved;
  delete item.rejected;
  item.status_product = 'waiting';

  item.updated.at = moment().utcOffset('7').format('YYYY-MM-
DD[T]HH:mm:ss.SSS[Z]');
  const result = await productsDAL.update(item);
  return result;
}

/**
* Get all item

```

```

* @param {Object} req express request object
* @param {Object} res express response object
* @api public
*/
exports.deleteById = async (id,object) => {
    let item = await productsDAL.findById(id);
    if(!item) throw new Error('Item with id:'+id+', not found!');
    for(let key in object){
        item[key] = object[key];
    }
    item.deleted.at = moment().utcOffset('7').format('YYYY-MM-
DD[T]HH:mm:ss.SSS[Z]');
    const result = await productsDAL.update(item);
    return result;
}

/**
* Get all item
* @param {Object} req express request object
* @param {Object} res express response object
* @api public
*/
exports.approval = async (id,object,status) => {
    let item = await productsDAL.findById(id);
    if(!item) throw new Error('Item with id:'+id+', not found!');
    for(let key in object){
        item[key] = object[key];
    }
    if(status == 'approve'){
        item.approved.at = moment().utcOffset('7').format('YYYY-MM-
DD[T]HH:mm:ss.SSS[Z]');
        item.status_product = 'approved';
    }else if(status == 'reject'){
        item.rejected.at = moment().utcOffset('7').format('YYYY-MM-
DD[T]HH:mm:ss.SSS[Z]');
        item.status_product = 'rejected';
    }
    await productsDAL.update(item);
    return item;
}

```

```

/***
 * Get all item
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
 */
exports.upload = async (id,object,status) => {
    let item = await productsDAL.findById(id);
    if(!item) throw new Error('Item with id:' + id + ', not found!');
    for(let key in object){
        item[key] = object[key];
    }
    if(status == 'approve'){
        item.approved.at = moment().utcOffset('7').format('YYYY-MM-
DD[T]HH:mm:ss.SSS[Z]');
        item.status_product = 'approved';
    }else if(status == 'reject'){
        item.rejected.at = moment().utcOffset('7').format('YYYY-MM-
DD[T]HH:mm:ss.SSS[Z]');
        item.status_product = 'rejected';
    }
    await productsDAL.update(item);
    return item;
}

```

#### g. ProductUnitTesting.js

Berikut adalah source code untuk Product Unit Testing.

```

const productService = require('./productService');
const {Product} = require('./product');
const productDAL = require('./productDAL');
const _ = require('lodash');
const gatewayHelper = require('../..libraries/gateway');
const config = require('config');
const productHelper = require('./productHelper');
const utils = require('../..libraries/utils');
var baseSample;

describe('Test Product DAL', ()=>{

```

```
beforeEach(async () =>{  
  
    baseSample = {  
        "name": "produk",  
        "categories": ["5d54baeb07ceed1efa8ae399"],  
        "description": "Cloud storage service by amazon",  
        "tags": ["cloud", "storage"],  
        "image_logo": "https://minio.playcourt.co.id/product-catalog/",  
        "founder": ["iqbal"],  
        "established_on": "2019-01-01",  
        "status_product": "waiting",  
        "npwp": "123123",  
        "config": {  
            "po_number_periode": "daily",  
            "administration_fee": 123,  
            "unique_code_type": "discount",  
            "invoice_signer": "1"  
        },  
        "contact": {  
            "website": "https://s3.amazonaws.com",  
            "phone": "083829365965",  
            "email": "s3helpdesk@amazon.com",  
            "address": "USA"  
        },  
        "is_ready_for_market": false,  
        "companion": {  
            "organization_company_id": "1",  
            "division_id": "1",  
            "companion_id": "1"  
        },  
        "partnership": {  
            "company_id": "1",  
            "partnership_id": "1",  
            "contract_id": "1",  
            "business_scheme": "asdf"  
        },  
        "bank": {  
            "bank_account_owner": "iqbal",  
            "bank_account_number": "1123",  
            "bank_name": "iqbal sejahtera",  
            "bank_branch_office": "iqbal"  
        }  
    }  
})
```

```

        },
        "features": [],
        "benefit": [],
        "meta": []
    );
});

afterEach(async () =>{
    jest.restoreAllMocks();
});

describe('Function get()',()=>{
    it('should return all item',async()=>{
        //mock external function
        const sample1 = new Product(baseSample);
        const sample2 = new Product(_.set(baseSample,'name','produk2'));

        const limitMethodResult = jest.fn().mockResolvedValue([sample1,sample2]);
        const limitMethod = {limit: limitMethodResult};
        const skipMethodResult = jest.fn(()=>limitMethod);
        const skipMethod = {skip: skipMethodResult};
        const sortMethodResult = jest.fn(()=>skipMethod);
        const sortMethod = {sort: sortMethodResult};
        const collationMethodResult = jest.fn(()=>sortMethod);
        const collationMethod = {collation: collationMethodResult};
        Product.aggregate = jest.fn(()=>collationMethod);

        //function
        const query = {
            perpage: 2,
            currpage: 0
        }
        const result = await productDAL.get(query);
        //assert
        expect(result.length).toBe(2);
        expect(result.some(g=>g.name === 'produk')).toBeTruthy();
        expect(result.some(g=>g.name === 'produk2')).toBeTruthy();
    });
});

```

```
describe('Function getOne()',()=>{
  it('should return selected item',async()=>{
    //mock external function
    const sample1 = new Product(baseSample);
    const sortMethodResult = jest.fn().mockResolvedValue(sample1);
    const sortMethod = { sort: sortMethodResult};
    const collationMethodResult = jest.fn(()=>sortMethod);
    const collationMethod = { collation: collationMethodResult};
    Product.findOne = jest.fn(()=>collationMethod);

    const result = await productDAL.getOne();
    //assert
    expect(result.name === 'produk').toBeTruthy();
  });
});

describe('Function save()',()=>{
  it('should save the item',async()=>{
    //mock external function
    const sample1 = new Product(baseSample);
    jest.spyOn(Product.prototype,'save').mockResolvedValue(sample1);
    //function
    const result = await productDAL.save(baseSample);

    //assert
    expect(result.name === 'produk').toBeTruthy();
  });
});

describe('Function findById()',()=>{
  it('should get selected id ',async()=>{
    //mock external function
    const sample1 = new Product(baseSample);
    Product.findOne = jest.fn().mockResolvedValue(sample1);
    //function
    const id = "test";
    const result = await productDAL.findById(id);
    //assert
    expect(result.name === 'produk').toBeTruthy();
  });
});
```

```
});

describe('Function update()',()=>{
  it('should update selected id ',async()=>{
    //mock external function
    const sample1 = new Product(baseSample);
    sample1.description = 'edited';
    jest.spyOn(Product.prototype,'save').mockResolvedValue(sample1);
    //function
    const result = await productDAL.update(sample1);
    //assert
    expect(result.name === 'produk').toBeTruthy();
    expect(result.description === 'edited').toBeTruthy();
  });
});

describe('Function deleteById()',()=>{
  it('should delete selected id ',async()=>{
    //mock external function
    const sample1 = new Product(baseSample);
    Product.findOneAndDelete = jest.fn().mockResolvedValue(sample1);
    ;
    //function
    const id = "test";
    const result = await productDAL.deleteById(id);
    //assert
    expect(result.name === 'produk').toBeTruthy();
    expect(Product.findOneAndDelete).toHaveBeenCalled();
  });
});

describe('Function getTotal()',()=>{
  it('should return total count of all records',async()=>{
    //mock external function
    Product.countDocuments = jest.fn().mockResolvedValue(2);
    //function
    const result = await productDAL.getTotal();

    //assertz
    expect(result === 2).toBeTruthy();
  });
});
```

```

    });
});

describe('Function getTotalFiltered()',()=>{
  it('should return total filtered records ',async()=>{
    Product.aggregate = jest.fn().mockResolvedValue([
      'count': 1
    ]);
    //function
    const result = await productDAL.getTotalFiltered();
    //assert
    expect(result == 1).toBeTruthy();
  });
  it('should return 0 if empty',async()=>{
    Product.aggregate = jest.fn().mockResolvedValue([]);
    //function
    const result = await productDAL.getTotalFiltered();
    //assert
    expect(result == 0).toBeTruthy();
  });
});
});

describe('Test Product Services',()=>{

  beforeEach(async () =>{

    baseSample = {
      "name":"produk",
      "categories":["5d54baeb07ceed1efa8ae399"],
      "description":"Cloud storage service by amazon",
      "tags":["cloud","storage"],
      "image_logo":"https://minio.playcourt.co.id/product-catalog/",
      "founder":["iqbal"],
      "established_on":"2019-01-01",
      "status_product":"waiting",
      "npwp": "123123",
      "config":{
        "po_number_periode":"daily",
        "administration_fee":123,
        "unique_code_type": "discount",
      }
    }
  });
});

```

```

    "invoice_signer": "1"
},
"contact": {
    "website": "https://s3.amazonaws.com",
    "phone": "083829365965",
    "email": "s3helpdesk@amazon.com",
    "address": "USA"
},
"is_ready_for_market": false,
"companion": {
    "organization_company_id": "1",
    "division_id": "1",
    "companion_id": "1"
},
"partnership": {
    "company_id": "1",
    "partnership_id": "1",
    "contract_id": "1",
    "business_scheme": "asdf"
},
"bank": {
    "bank_account_owner": "iqbal",
    "bank_account_number": "1123",
    "bank_name": "iqbal sejahtera",
    "bank_branch_office": "iqbal"
},
"features": [],
"benefit": [],
"meta": []
};

});

afterEach(async () =>{
    jest.restoreAllMocks();
});

describe('Function get()',()=>{
    it('should return all item',async()=>{
        //mock external function
        const sample1 = new Product(baseSample);
        const sample2 = new Product(_.set(baseSample,'name','produk2'));
    });
});

```

```

let result = { };
productDAL.get = jest.fn().mockResolvedValue(
    [sample1,sample2]
);
productDAL.getTotal = jest.fn().mockResolvedValue(
    2
);
productDAL.getTotalFiltered = jest.fn().mockResolvedValue(
    1
);

const data = await productDAL.get();
const total = await productDAL.getTotal();
const filtered = await productDAL.getTotalFiltered();

result.data = data;
result.total = total;
result.filtered = filtered;
//assert
expect(result.data.length).toBe(2);
expect(result.data.some(g=>g.name === 'produk')).toBeTruthy();
expect(result.data.some(g=>g.name === 'produk2')).toBeTruthy();
});
});

describe('Function save()',()=>{
it('should save the item',async()=>{
//mock external function
baseSample.created = {
    by:'iqbal'
};
const sample1 = new Product(baseSample);
productDAL.save = jest.fn().mockResolvedValue(sample1);
const user = {
    data: {
        "isActive": true,
        "username": "product-catalog",
        "id": "e5db8237-6bf7-4855-a070-c3a9a80cb1a2",
        "firstname": "product",
        "lastname": "catalog",
        "email": "iqbalzetto@gmail.com",
    }
});
```

```
        "createdAt": "Sun Sep 22 2019 21:36:46 GMT+0700 (Western I  
ndonesia Time)",  
        "updatedAt": "Sun Sep 22 2019 21:36:46 GMT+0700 (Western  
Indonesia Time)"  
    }  
};  
const app = {  
    data: {  
        "isActive": true,  
        "id": "e161cf4a-5e63-4855-b54b-aece9fdd9460",  
        "userId": "e5db8237-6bf7-4855-a070-c3a9a80cb1a2",  
        "name": "TPN",  
        "createdAt": "Sun Sep 22 2019 21:36:54 GMT+0700 (Western I  
ndonesia Time)",  
        "updatedAt": "Sun Sep 22 2019 21:36:54 GMT+0700 (Western  
Indonesia Time)"  
    }  
};  
const credential = {  
    data: {  
        "consumerId": "e161cf4a-5e63-4855-b54b-aece9fdd9460",  
        "isActive": true,  
        "createdAt": "Sun Sep 22 2019 21:37:19 GMT+0700 (Western I  
ndonesia Time)",  
        "updatedAt": "Sun Sep 22 2019 21:37:19 GMT+0700 (Western  
Indonesia Time)",  
        "keyId": "0wsPae66F7Iy0K3T7HCSWZ",  
        "keySecret": "0qhmER9nfAanjDTzKT6Ndb",  
        "scopes": null,  
        "id": "0wsPae66F7Iy0K3T7HCSWZ",  
        "type": "key-auth"  
    }  
};  
config.get = jest.fn().mockResolvedValue('asdf');  
gatewayHelper.getUser = jest.fn().mockResolvedValue(user);  
gatewayHelper.createUser = jest.fn().mockResolvedValue(user);  
gatewayHelper.createApp = jest.fn().mockResolvedValue(app);  
gatewayHelper.createCredential = jest.fn().mockResolvedValue(cred  
ential);  
//productHelper.generateProductCode = jest.fn().mockResolvedValue  
('AA');
```

```
jest.spyOn(productHelper,'generateProductCode').mockResolvedValue('AA');
    //function
    const result = await productService.save(baseSample);
    //assert
    expect(result.name === 'produk').toBeTruthy();
});
});

describe('Function findById()',()=>{
it('should get selected id ',async()=>{
    //mock external function
    const sample1 = new Product(baseSample);
    productDAL.findById = jest.fn().mockResolvedValue(sample1);
    //function
    const id = "test";
    const result = await productService.findById(id);
    //assert
    expect(result.name === 'produk').toBeTruthy();
});
});

describe('Function update()',()=>{
it('should update selected id ',async()=>{
    //mock external function
    const sample1 = new Product(baseSample);
    sample1.description = 'edited';
    productDAL.update = jest.fn().mockResolvedValue(sample1);
    //function
    const result = await productService.update('test',baseSample);
    //assert
    expect(result.name === 'produk').toBeTruthy();
    expect(result.description === 'edited').toBeTruthy();
});
});

describe('Function deleteById()',()=>{
it('should delete selected id ',async()=>{
    //mock external function
    const sample1 = new Product(baseSample);
    productDAL.deleteById = jest.fn().mockResolvedValue(sample1);
});
```

```

//function
const id = "test";
const result = await productService.deleteById(id);
//assert
expect(result).toBeTruthy();
});

});

describe('Function count()',()=>{
  it('should return total item record',async()=>{
    //mock external function
    productDAL.getTotalFiltered = jest.fn().mockResolvedValue(
      1
    );
    const result = await productService.count();

    //assert
    expect(result == 1).toBeTruthy();

  });
});

describe('Function approval()',()=>{
  it('should update status to approved on approval',async()=>{
    //set data input
    const input = {
      "approved": {
        "by": "iqbal ganteng 2024",
        "reference_id": "iqbal ganteng 2025"
      }
    }
    //mock external function
    const sample1 = new Product(baseSample);
    productDAL.update = jest.fn().mockResolvedValue(sample1);

    const result = await productService.approval('test',input,'approve');

    //assert
    expect(result.name === 'produk').toBeTruthy();
  });
  it('should update status to rejected on reject',async()=>{

```

```

//set data input
const input = {
  "rejected": {
    "by": "iqbal ganteng 2024",
    "reference_id": "iqbal ganteng 2025"
  }
}
//mock external function
const sample1 = new Product(baseSample);
productDAL.update = jest.fn().mockResolvedValue(sample1);

const result = await productService.approval('test',input,'reject');

//assert
expect(result.name === 'produk').toBeTruthy();
});

});
});

describe('Test Product Helper',()=>{

beforeEach(async () =>{
  jest.restoreAllMocks();
  baseSample = {
    "name":"produk",
    "categories":["5d54baeb07ceed1efa8ae399"],
    "description":"Cloud storage service by amazon",
    "tags":["cloud","storage"],
    "image_logo":"https://minio.playcourt.co.id/product-catalog/",
    "founder":["iqbal"],
    "established_on":"2019-01-01",
    "status_product":"waiting",
    "npwp": "123123",
    "config": {
      "po_number_periode": "daily",
      "administration_fee": 123,
      "unique_code_type": "discount",
      "invoice_signer": "1"
    },
    "contact": {
      "website": "https://s3.amazonaws.com",
    }
  }
})
});
```

```

    "phone":"083829365965",
    "email":"s3helpdesk@amazon.com",
    "address":"USA"
},
"is_ready_for_market":false,
"companion":{
    "organization_company_id":"1",
    "division_id":"1",
    "companion_id":"1"
},
"partnership":{
    "company_id":"1",
    "partnership_id":"1",
    "contract_id":"1",
    "business_scheme":"asdf"
},
"bank":{
    "bank_account_owner":"iqbal",
    "bank_account_number":"1123",
    "bank_name":"iqbal sejahtera",
    "bank_branch_office": "iqbal"
},
"features":[],
"benefit":[],
"meta": []
};

});

afterEach(async () =>{
    jest.restoreAllMocks();
});

describe('Function generateProductCode()',()=>{
    it('should return random product code if product name length less than one char',async()=>{
        const name = 'i';
        jest.spyOn(utils,'makeRandomString').mockResolvedValue('AS');
        jest.spyOn(productDAL,'getOne').mockResolvedValue(null);
        //function
        const result = await productHelper.generateProductCode(name);
        //assert
    });
});

```

```
expect(result.length).toBe(2);
expect(result == 'AS').toBeTruthy();
});
it('should use first two character of name if not exist in db',async()=>{
  const name = 'iqbal';
  jest.spyOn(productDAL,'getOne').mockResolvedValue(null);
  //function
  const result = await productHelper.generateProductCode(name);
  //assert
  expect(result == 'IQ').toBeTruthy();
});
it('should use random product code if name already exists',async()=>{
  const name = 'iqbal';
  jest.spyOn(utils,'makeRandomString').mockResolvedValue('AS');
  jest.spyOn(productDAL,'getOne').mockResolvedValue('sudah ada');
  //function
  const result = await productHelper.generateProductCode(name);
  //assert

  expect(result == 'AS').toBeTruthy();
});
});
});
```

### 5.3 CONFIQ/

Merupakan file untuk setting envirotment, penamaan dafult, penamaan development, penamaan production, serta penamaan test



a. File Custom-environment-variables.json

Berikut adalah source code File Custom-environment-variables.

```
"db":{  
    "connectionString":"MONGO_CONNECTION_STRING"  
},  
"jwt":{  
    "key":"JWT_KEY"  
},  
"port":"PORT",  
"aws":{  
    "endpoint":"AWS_ENDPOINT",  
    "key":"AWS_KEY",  
    "secret":"AWS_SECRET",  
    "bucket":"AWS_BUCKET",  
    "port":"AWS_PORT",  
    "ssl":"AWS_SSL"  
},  
"gateway":{  
    "admin":{  
        "url":"ADMIN_GATEWAY_URL"  
    }  
},  
"policy":{  
    "url" : "POLICY_URL"  
}  
}
```

b. File Default.json

Berikut adalah source code File default.

```
{  
  "name": "TPN - PRODUCT CATALOG API SERVICE"  
}
```

c. File Development.json

Berikut adalah source code File development.

```
{  
  "name": "TPN - PRODUCT CATALOG API SERVICE - DEVELOPMENT"  
}
```

d. File Production.json

Berikut adalah source code File production.

```
{  
  "name": "TPN - PRODUCT CATALOG API SERVICE - PRODUCTION"  
}
```

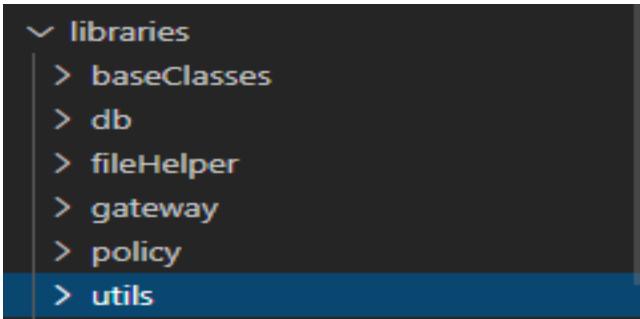
a. File Test.json

Berikut adalah source code File Test.

```
{  
  "name": "TPN - PRODUCT CATALOG API SERVICE - TEST"  
}
```

## 5.4 LIBRARIES/

Merupakan folder yang didalamnya terdapat library function yang akan digunakan, library sendiri bisa menggunakan library bawaan dari node atau juga bisa membuat library sendiri dengan kebutuhan yang diinginkan.



a. File db/test

Berikut adalah source code File db/test.

```
const db = require('./index');

exports.runTest = (server) =>{
    beforeAll(async () =>{

    });
    afterAll(async () =>{

    });

    describe('Test DB Library',()=>{
        describe('Function init',()=>{
            it('should be using bluebird promise',async()=>{
                await db.init();
                const connString = await server.getConnectionString();
                await db.connect(connString);

                expect(db.mongoose.Promise.version).toBeTruthy();
                db.mongoose.connection.close();
            });
        });
    });

    describe('Function connect',()=>{
        it('should be connected to mongodb',async()=>{
            await db.init();
            const connString = await server.getConnectionString();
            const conn = await db.connect(connString);
        });
    });
}
```

```

    expect(conn).toBeTruthy();
    db.mongoose.connection.close();
});

it('should log error ',async()=>{
    await db.init();
    await expect(db.connect('asdf')).toThrow(Error);
});
});

});
}

```

### b.File db/index

Berikut adalah source code library db/index default.

```

const mongoose = require('mongoose');
const Bluebird = require('bluebird');

module.exports = {
  mongoose,
  init: () => {
    mongoose.Promise = Bluebird;
  },
  connect: async database => {
    try {
      const conn = await mongoose.connect(
        database,
        { useNewUrlParser: true ,useCreateIndex: true }
      );
      console.log(`MongoDb Connected`);

      return conn;
    } catch (err) {
      //eslint-disable-next-line
      console.log('Error to connect on mongo', err);
      // throw err;
    }
  }
}

```

```

        }
    },
    disconnect: async () => await mongoose.disconnect(),
    close: async () => await mongoose.connection.close(),
};


```

### c. File Filehelper/fileAPI

Berikut adalah source code library File Filehelper/fileAPI.

```

const express = require('express');
const router = express.Router();
const respond = require('../respond');
const config = require('config');
const Minio = require('minio');
const fs = require('fs');
const dir = './tmp';
//const cors = require('../middlewares/cors');

router.post('/', async(req,res)=>{
    if (Object.keys(req.files).length == 0) {
        return respond.resBadRequest(res,'No file uploaded');
    }
    if (!fs.existsSync(dir)){
        fs.mkdirSync(dir);
    }
    //save file temporarily
    await req.files.file.mv('tmp/'+req.files.file.md5);
    //await req.files.file.mv('public/'+req.files.file.name);

    let aws_endpoint = config.get('aws.endpoint');
    aws_endpoint = aws_endpoint.replace('http://','');
    aws_endpoint = aws_endpoint.replace('https://','');

    const minioClient = new Minio.Client({
        endPoint: aws_endpoint,
        useSSL:(config.get('aws.ssl') == "true")?true:false,
        port:parseInt(config.get('aws.port')),
        //useSSL: config.get('aws.ssl'),
        //port:config.get('aws.port'),
    }


```

```

accessKey: config.get('aws.key'),
secretKey: config.get('aws.secret')
});
//minio upload
await minioClient.fPutObject(config.get('aws.bucket'),req.files.file.name,'tmp/'+req.files.file.md5);

return respond.resSuccessData(res,'File uploaded',{ "name":req.files.file.name});
});

// router.get('/:name', ()=>{
// });

module.exports = router;

```

#### d. File Filehelper/fileAPI

Berikut adalah source code library File Filehelper/index.

```

const Minio = require('minio');
const config = require('config');
const fs = require('fs');
const dir = './tmp';
const Path = require('path');
const Util = require('util');
const ReadFile = Util.promisify(fs.readFile);

/**
 * Get all item
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
 */
const uploadMinio = async (file, minioPath,filename)=>{
  if (!fs.existsSync(dir)){
    fs.mkdirSync(dir);
  }
  const fileStream = fs.createWriteStream(minioPath);
  fileStream.write(file.buffer);
  fileStream.end();
}

```

```

        }

    //save file temporarily
    await file.mv('tmp/'+file.md5);
    //await req.files.file.mv('public/'+req.files.file.name);

    let aws_endpoint = config.get('aws.endpoint');
    aws_endpoint = aws_endpoint.replace('http://','');
    aws_endpoint = aws_endpoint.replace('https://','');

    const minioClient = new Minio.Client({
        endPoint: aws_endpoint,
        useSSL:(config.get('aws.ssl') == "true")?true:false,
        port:parseInt(config.get('aws.port')),
        //useSSL: config.get('aws.ssl'),
        //port:config.get('aws.port'),
        accessKey: config.get('aws.key'),
        secretKey: config.get('aws.secret')
    });
    //minio upload
    await minioClient.fPutObject(config.get('aws.bucket'),minioPath+'/'+filename,'tmp/'+file.md5);

    let uploaded={
        extension: file.name.substr(file.name.lastIndexOf('.') +1),
        url: config.get('aws.public_url')+'/'+config.get('aws.bucket')+'/'+minioPath+'/'+filename
    }

    return uploaded;
}

/***
 * Get all item
 * @param {Object} req express request object
 * @param {Object} res express response object
 * @api public
 */
const upload = async (files)=>{
    if (Object.keys(files).length == 0) {
        throw new Error('No file uploaded!');
    }
}

```

```

        }
        if (!fs.existsSync(dir)){
            fs.mkdirSync(dir);
        }
        //save file temporarily
        let extension = files.file.name.substr(files.file.name.lastIndexOf('.') +1);
        let fullPath = 'tmp/'+files.file.md5+'.'+extension;
        await files.file.mv(fullPath);

        let uploaded={
            fullPath:fullPath,
            name: files.file.name
        }

        return uploaded;
    }

    /**
     * Get all item
     * @param {Object} req express request object
     * @param {Object} res express response object
     * @api public
     */
    const deleteFile = (path)=>{

        if (fs.existsSync(path)){
            fs.unlinkSync(path);
        }else{
            return false;
        }

        return true;
    }

    /**
     * Get all item
     * @param {Object} req express request object
     * @param {Object} res express response object
     * @api public
     */
    const readFile = async (path)=>{

```

```

try{
    const templatePath = Path.resolve(__dirname, path);
    const content = await ReadFile(templatePath);
    return content;
}catch(e){
    return false;
}
}

module.exports = {
    upload,
    deleteFile,
    readFile,
    uploadMinio
}

```

#### e. File Gateway/index

Berikut adalah source code library File Gateway/index.

```

const axios = require('axios');

/*
  USER
*/

exports.createUser = async (adminUrl,username, firstname, lastname, email, redirectUri) =>{
    try {
        const item = {
            "username": username,
            "firstname": firstname,
            "lastname": lastname,
            "email": email,
            "redirectUri": redirectUri
        };
        const url = adminUrl + '/users';
        const response = await axios.post(url, item);
    }
}
```

```
        return response;
    } catch (error) {
        console.error(error);
    }
}

exports.getUser = async (adminUrl,userId) =>{
    try {
        const url = adminUrl+'/users/'+userId;
        const response = await axios.get(url);
        return response;
    } catch (error) {
        //console.error(error);
    }
}

exports.listUser = async (adminUrl) =>{
    try {
        const url = adminUrl+'/users';
        const response = await axios.get(url);

    } catch (error) {
        console.error(error);
    }
}

exports.updateUser = async (adminUrl,appId,name,redirectUri) =>{
    try {
        const item = {
            "name": name,
            "redirectUri": redirectUri
        };
        const url = adminUrl+ "/" + "apps" + "/" + appId;
        const response = await axios.put(url,item);

    } catch (error) {
        console.error(error);
    }
}

exports.deleteUser = async (adminUrl,appId) =>{
```

```
try {
    const url = adminUrl+'/apps/'+appId;
    const response = await axios.delete(url);

} catch (error) {
    console.error(error);
}
}

/*
    APPS
*/
exports.createApp = async (adminUrl,userId, name,redirectUri) =>{
    try {
        const item = {
            "userId": userId,
            "name": name,
            "redirectUri": redirectUri
        };
        const url = adminUrl+'/apps';
        const response = await axios.post(url,item);
        return response;
    } catch (error) {
        console.error(error);
    }
}

exports.getApp = async (adminUrl,appId) =>{
    try {
        const url = adminUrl+'/apps/'+appId;
        const response = await axios.get(url,item);

    } catch (error) {
        console.error(error);
    }
}

exports.listApp = async (adminUrl,appId) =>{
    try {
        const url = adminUrl+'/apps';
    }
}
```

```
const response = await axios.get(url,item);

} catch (error) {
  console.error(error);
}
}

exports.updateApp = async (adminUrl,appId,name,redirectUri) =>{
try {
  const item = {
    "name": name,
    "redirectUri": redirectUri
  };
  const url = adminUrl+'/apps/'+appId;
  const response = await axios.put(url,item);

} catch (error) {
  console.error(error);
}
}

exports.deleteApp = async (adminUrl,appId) =>{
try {
  const url = adminUrl+'/apps/'+appId;
  const response = await axios.delete(url);

} catch (error) {
  console.error(error);
}
}

/*
 CREDENTIAL
 */

exports.createCredential = async (adminUrl,consumerId, type, credential) =>
{
try {
  const item = {
    "consumerId": consumerId,
    "type": type
  }
}
```

```
};

const url = adminUrl+'/credentials';
const response = await axios.post(url,item);
return response;
} catch (error) {
  console.error(error);
}
}

exports.getCredential = async (adminUrl,appId) =>{
try {
  const url = adminUrl+'/apps/'+appId;
  const response = await axios.get(url,item);

} catch (error) {
  console.error(error);
}
}

exports.listCredential = async (adminUrl,appId) =>{
try {
  const url = adminUrl+'/apps';
  const response = await axios.get(url,item);

} catch (error) {
  console.error(error);
}
}

exports.updateCredential = async (adminUrl,appId,name,redirectUri) =>{
try {
  const item = {
    "name": name,
    "redirectUri": redirectUri
  };
  const url = adminUrl+'/apps/'+appId;
  const response = await axios.put(url,item);

} catch (error) {
  console.error(error);
}
}
```

```
}
```

```
exports.deleteCredential = async (adminUrl,appId) =>{
    try {
        const url = adminUrl+'/apps/'+appId;
        const response = await axios.delete(url);

    } catch (error) {
        console.error(error);
    }
}
```

#### f. File Policy/index

Berikut adalah source code library File Policy/index.

```
const axios = require('axios');
const config = require('config');

exports.verify = async (app_id, function_id) => {
    try {
        const policyUrl = config.get('policy.url');
        const url = policyUrl+'/api/v1/authorization/policy';
        const response = await axios.get(url,{
            params:{
                "app_id": app_id,
                "function_id": function_id
            }
        });
        return response.status;
    } catch (error) {
        console.error(error);
    }
}
```

#### g. File Utils/index

Berikut adalah source code library File Utils /index.

```

const moment = require('moment');
const mongoose = require('mongoose');

const toObjId = async (string)=>{
    return mongoose.Types.ObjectId(string);
}

const getQuery = (query, keyword, type) =>{//untuk pencarian di dalam list
    res = { };
    for(let key in query){
        if(key.includes(keyword)){
            let newKey = key.replace(keyword,"");
            if(type==='like'){
                res[newKey] = {$regex: `.*${query[key]}.*`, $options:'i'};
            }else if(type==='bool'){
                if(query[key] ==='false'|| query[key] ==='0'){
                    res[newKey] = false;
                }else if(query[key] ==='true'|| query[key] ==='1'){
                    res[newKey] = true;
                }
            }else{
                res[newKey] = false;
            }
        }else if(type==='number'){
            res[newKey] = Number(query[key]);
        }else if(type==='in'){
            res[newKey] = {$in:query[key]};
        }else if(type==='objid'){
            res[newKey] = mongoose.Types.ObjectId(query[key]);
        }else if(type==='exists'){
            if(query[key] ==='false'|| query[key] ==='0'){
                res[newKey] = null;
            }else if(query[key] ==='true'|| query[key] ==='1'){
                res[newKey] = {$ne:null};
            }
        }else{
            res[newKey] = {$ne:null};
        }
    }
    else{
        res[newKey] = query[key];
    }
}

```

```

        }
    }
}

return res;
};

const getQueryOr = (query, keyword, type) =>{//untuk pencarian di dalam list
let res = [];
for(let key in query){
    let obj = {};
    if(key.includes(keyword)){
        let newKey = key.replace(keyword,"");
        if(type==='like'){
            obj[newKey] = {$regex: `.*${query[key]}.*`, $options:'i'};
        }
        else{
            obj[newKey] = query[key];
        }
    }
    if(Object.keys(obj).length > 0)
        res.push(obj);
}
if(res.length >0){
    return res;
}else{
    return [{}];
}
};

const betweenDate = (filter) =>{
res = {};

for(let key in filter){
    let string = filter[key];
    let dates = string.split("|");
    if(string.indexOf('|') == string.length-1){ //jika hanya startdate
        dates[1] = '2100-01-01';
    }else if (string.indexOf('|') == 0){ //jika hanya enddate
}
}
}

```

```

        dates[0] = '1900-01-01';
    }

    res[key] = {
        '$gte': new Date(moment(dates[0]).utc().format('YYYY-MM-
DD[T]HH:mm:ss.SSS[Z]')),
        '$lte': new Date(moment(dates[1]).utc().format('YYYY-MM-
DD[T]HH:mm:ss.SSS[Z]'))
    };
}

return res;
};

const getFirstDayOfPastMonths = async (months)=>{
    const res = await moment().utc().subtract(months, 'months').startOf('mont
h').format('YYYY-MM-DD[T]HH:mm:ss.SSS[Z]');
    return res
}

const getLastDayOfPastMonths = async (months)=>{
    const res = await moment().utc().subtract(months, 'months').endOf('month'
).format('YYYY-MM-DD[T]HH:mm:ss.SSS[Z]');
    return res;
}

const pad_with_zeroes = (number,length)=>{
    var my_string = " + number;
    while (my_string.length < length) {
        my_string = '0' + my_string;
    }

    return my_string;
}

const getMonthNumber = (monthName)=>{
    let months = { };
    if(monthName.length <4){
        months = {
            'Jan' : '01',
            'Feb' : '02',

```

```

'Mar' : '03',
'Apr' : '04',
'May' : '05',
'Jun' : '06',
'Jul' : '07',
'Aug' : '08',
'Sep' : '09',
'Oct' : '10',
'Nov' : '11',
'Dec' : '12'
}
}else{
months = {
'January' : '01',
'February' : '02',
'March' : '03',
'April' : '04',
'May' : '05',
'June' : '06',
'July' : '07',
'August' : '08',
'September' : '09',
'October' : '10',
'November' : '11',
'December' : '12'
}
}

return months[monthName];
}

const penyebut = (value) =>{
value = Math.floor(Math.abs(Number(value)));

const words = [
","", "satu", "dua", "tiga", "empat", "lima", "enam", "tujuh", "delapan", "sembilan", "sepuluh", "sebelas"
];
let temp = "";
if(value < 12){
temp = " "+words[value];
}

```



```
const convertToRupiah = (angka)=>{
    let rupiah = "";
    let angkarev = angka.toString().split("").reverse().join("");
    for(let i = 0; i < angkarev.length; i++) if(i%3 == 0) rupiah += angkarev.substring(i,3)+ '.';
    return 'Rp. '+rupiah.split(",rupiah.length-1).reverse().join(""));
}

var buildQuery = function (data) {

    // If the data is already a string, return it as-is
    if (typeof (data) === 'string') return data;

    // Create a query array to hold the key/value pairs
    var query = [];

    // Loop through the data object
    for (var key in data) {
        if (data.hasOwnProperty(key)) {

            // Encode each key and value, concatenate them into a string, and push them to the array
            query.push(encodeURIComponent(key) + '=' + encodeURIComponent(data[key]));
        }
    }

    // Join each item in the array with a `&` and return the resulting string
    return query.join('&');
};

const makeRandomString = (length)=>{
    var result      = "";
    var characters  = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    var charactersLength = characters.length;
    for ( var i = 0; i < length; i++ ) {
        result += characters.charAt(Math.floor(Math.random() * charactersLength));
    }
    return result;
}
```

```
}

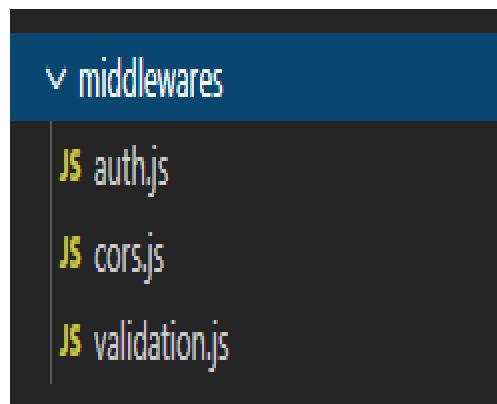
module.exports = {
    getQuery,
    getQueryOr,
    betweenDate,
    getFirstDayOfPastMonths,
    getLastDayOfPastMonths,
    pad_with_zeroes,
    getMonthNumber,
    terbilang,
    convertToRupiah,
    toObjId,
    buildQuery,
    makeRandomString
}

}
```

## 5.5 MIDDLEWARES/

*Middleware* pada dasarnya adalah ‘penengah’. Kalau dalam aplikasi *middleware* adalah sebuah aturan yang harus dilewati terlebih dahulu sebelum masuk kedalam sebuah sistem atau keluar dari sebuah sistem. Analoginya seperti ini, misalnya teman-teman ingin mendaftar di

sebuah pekerjaan yaitu menjadi seorang *programmer*, maka sebelum teman-teman masuk menjadi karyawan, harus melewati prosedur terlebih dahulu. Misalnya, teman-teman akan dicek apakah usianya sudah layak untuk bekerja, atau teman-teman akan di interview terlebih dahulu, atau mengerjakan test untuk mengetahui skillnya terlebih dahulu. *Nah*, semua tahapan-tahapan tersebut disebut dengan *middleware*. *Middleware* sendiri sebenarnya konsep umum yang banyak dipakai pada *library* maupun *framework*, adapun pembuatan middlewares didigital invoice dan settlement ini yaitu middlewares auth, middlewares cors, dan middlewares validation.



#### a. File middlewares/auth

Berikut adalah source code File middlewares/auth.

```
const policy = require('../libraries/policy');
const respond = require('../libraries/respond');
```

```
module.exports = async(req,res,next)=>{

  const verify = await policy.verify(req.headers['eg-consumer-
id'],res.locals.function_id);
  if(!verify){
    return respond.resUnauthorized(res,"Unauthorized Access",undefined,u
ndefined);
  }

  next();
}
```

### b.File middlewares/cors

Berikut adalah source code File middlewares/cors.

```
module.exports = (req,res,next)=>{
  res.header("Access-Control-Allow-
Origin", "*"); // update to match the domain you will make the request from
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-
With, Content-Type, Accept");
  next();
}
```

### c.File middlewares/validation

Berikut adalah source code File middlewares/ validation.

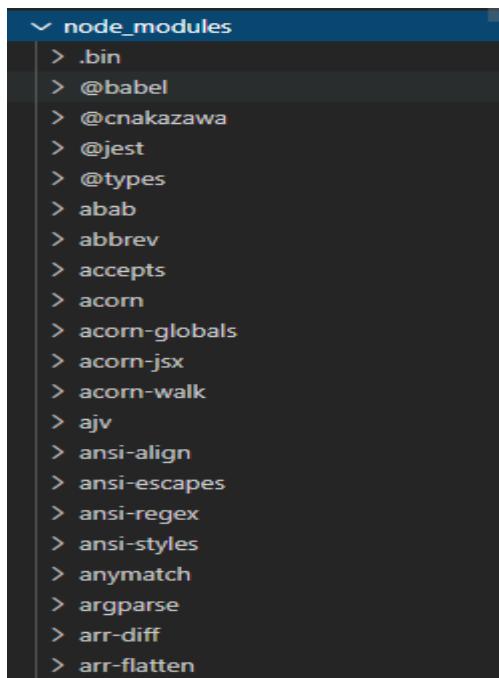
```
const Joi = require('joi');
const respond = require('../libraries/respond');

module.exports = (req,res,next)=>{
```

```
const { error } = Joi.validate(req.body,res.locals.joiSchema);
if(error) return respond.resBadRequest(res,error.details[0].message);
next();
}
```

## 5.6 NODE-MODULES

Modules bisa dikatakan sebuah library yang berisi fungsi fungsi yang sudah disiapkan dan sudah jadi. Sehingga kita tidak usah membuat fungsi dari 0. Misal kita ingin membuat tulisan pada console terminal berwarna dan kalian tidak ingin membuat fungsi ini dari awal karena terlalu memakan waktu. Nah, Kalian tinggal mencari modules untuk merubah warna tersebut, kalian bisa mencari modules / library pada situs [npmjs.com](https://www.npmjs.com). berikut adalah semua modules yang digunakan dalam aplikasi invoice dan settlement;



```
> array-flatten
> array-unique
> asn1
> assert-plus
> assign-symbols
> astral-regex
> async
> async-each
> async-limiter
> asynckit
> atob
> aws-sign2
> aws4
> axios
> babel-jest
> babel-plugin-istanbul
> babel-plugin-jest-hoist
> babel-preset-jest
> balanced-match
> base
> bcrypt-pbkdf
> binary-extensions
> block-stream2
```

```
> block-stream2
> bluebird
> body-parser
> boxen
> brace-expansion
> braces
> browser-process-hrtime
> browser-resolve
> bser
> bson
> buffer-from
> busboy
> bytes
> cache-base
> callsites
> camelcase
> capture-exit
> capture-stack-trace
> caseless
> chalk
> chardet
> chokidar
> ci-info
```

```
> ci-info
> class-utils
> cli-boxes
> cli-cursor
> cli-width
> cliui
> co
> code-point-at
> collection-visit
> color-convert
> color-name
> combined-stream
> commander
> component-emitter
> concat-map
> config
> configstore
> content-disposition
> content-type
> convert-source-map
> cookie
> cookie-signature
> cookiejar
```

```
> copy-descriptor
> core-util-is
> cors
> create-error-class
> cross-env
> cross-spawn
> crypto-random-string
> cssom
> cssstyle
> dashdash
> data-urls
> debug
> decamelize
> decode-uri-component
> deep-extend
> deep-is
> define-properties
> define-property
> delayed-stream
> depd
> destroy
> detect-newline
```

```
> duplexer3
> ecc-jsbn
> ee-first
> emoji-regex
> encodeurl
> end-of-stream
> error-ex
> es-abstract
> es-to-primitive
> es6-error
> escape-html
> escape-string-regexp
> escodegen
> eslint
> eslint-scope
> eslint-utils
> eslint-visitor-keys
> espree
> esprima
> esquery
> esrecurse
> estraverse
```

```
> fast-deep-equal
> fast-json-stable-stringify
> fast-levenshtein
> fb-watchman
> figures
> file-entry-cache
> fill-range
> finalhandler
> find-up
> flat-cache
> flattened
> follow-redirects
> for-in
> forever-agent
> form-data
> formidable
> forwarded
> fragment-cache
> fresh
> fs.realpath
```

```
> glob
> glob-parent
> global-dirs
> globals
> got
> graceful-fs
> growly
> handlebars
> har-schema
> har-validator
> has
> has-flag
> has-symbols
> has-value
> has-values
> hoek
> hosted-git-info
> html-encoding-sniffer
> http-errors
> http-signature
> iconv-lite
> ignore
```

## 5.7 APP.JS

app.js, file utama Express.js yang berisi penggunaan *package* utama dan konfigurasi utama, berikut source code untuk file APP.js;

```
var express = require('express');
const respond = require('./libraries/respond');
const productRouter = require('./components/products/productAPI');
const fileRouter = require('./libraries/fileHelper/fileAPI');
const db = require('./libraries/db');
const config = require('config');
const fileupload = require('express-fileupload');
const cors = require('cors');

var app = express();
// Connect to Mongo DB
if(process.env.NODE_ENV != "test"){
    try {
        db.init();
        db.connect(config.get('db.connectionString'));
    } catch (err) {
        throw err;
    }
}
//app.options('/api/product-catalog/v1/products',cors());
//app.use(cors({credentials: true, origin: '*'}));
app.use(express.static('public'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(fileupload());

app.get('/', function (req, res) {
    res.send(config.get('name'));
})
app.get('/api/', function (req, res) {
    res.send("It's work!");
})

app.use('/api/v1/products', productRouter);
app.use('/api/v1/files', fileRouter);

// catch 404 endpoint not found
```

```
app.use(function(req, res, next) {  
    respond.resNotFound(res,'endpoint not found');  
  
});  
  
// catch 500 internal server error  
app.use(function(err, req, res, next) {  
    respond.resInternalServerError(res,err.message,err);  
});  
  
module.exports = app;
```

## 5.8 DOCKER-COMPOSE

Docker adalah sebuah aplikasi yang bersifat open source yang berfungsi sebagai wadah/container untuk mengepak/memasukkan sebuah software secara lengkap beserta semua hal lainnya yang dibutuhkan oleh software tersebut dapat berfungsi. Pengaturan software beserta file/hal pendukung lainnya akan menjadi sebuah Image (istilah yang diberikan oleh docker). Kemudian sebuah instan dari Image tersebut kemudian disebut Container. Jika pembaca pernah belajar tentang OOP/Java – maka analogi hubungan Container dengan Image adalah seperti Object dengan Class-nya. Sehingga jika object adalah instance-of class maka container adalah instance-of image.berikut source code docker compose yang digunakan dalam pembuatan fitur dalam aplikasi digital invoice dan settlement.

```
version: "2"  
  
volumes:  
    mongo_data: {}  
  
services:  
    product-catalog-service:  
        # image: 'telkomindonesia/alpine:nodejs-8.9.3'  
        build: .  
        environment:
```

```
-  
MONGO_CONNECTION_STRING=mongodb://tpnpms:pass4user@mong  
odb:27017/tpnpms_db  
  - JWT_KEY=4luckyd4y  
  - PORT=8080  
  - NODE_ENV=production  
user: root  
networks:  
  - webnettest  
ports:  
  - "6001:8080"  
depends_on:  
  - mongodb  
# volumes:  
# - ./usr/src/app  
  
mongodb:  
image: 'telkomindonesia/alpine:mongodb-3.6'  
networks:  
  - webnettest  
environment:  
  - MONGODB_USER=tpnpms  
  - MONGODB_PASSWORD=pass4user  
  - MONGODB_ADMIN_PASSWORD=pass4admin  
  - MONGODB_DATABASE=tpnpms_db  
ports:  
  - "27017:27017"  
volumes:  
  - mongo_data:/data/db  
  
networks:  
  webnettest:
```

## 1. DOCKERFILE

Berikut adalah source code untuk docker file.

```
FROM telkomindonesia/alpine:nodejs-8.9.3

WORKDIR /usr/src/app
COPY package*.json .
RUN apk add --update --no-cache --virtual .build-dev build-
base python python-dev \
&& npm i -g npm \
&& npm i -g node-gyp \
&& npm i --build-from-source=bcrypt\
&& apk del .build-dev
COPY ..

EXPOSE 8080

CMD [ "npm", "start" ]
```

## 2. JENKINFILE

Berikut adalah source code untuk jenkin file.

```
pipeline {
    parameters {
        string(name: 'PRODUCTION_NAMESPACE', description: 'Product
ion Namespace', defaultValue: 'prod-telkompartner')
        string(name: 'STAGING_NAMESPACE', description: 'Staging Na
mespace', defaultValue: 'stage-telkompartner')
        string(name: 'DEVELOPMENT_NAMESPACE', description: 'Devel
opment Namespace', defaultValue: 'dev-telkompartner')

        string(name: 'VSAN_REGISTRY_URL', description: 'Openshift r
egistry URL', defaultValue: 'docker-registry-default.vsan-
apps.playcourt.id')
        string(name: 'STAGE_REGISTRY_URL', description: 'Openshift
registry URL', defaultValue: 'docker-registry-default.stage-
apps.playcourt.id')
```

```

        string(name: 'DOCKER_IMAGE_NAME',           description: 'Docker I
mage Name',           defaultValue: 'tpn-product-catalog-api')

        string(name: 'CHAT_ID',           description: 'chat id of telegram gr
oup',           defaultValue: '-383243277')
    }
    agent none
    options {
        // Skip default checkout behavior
        skipDefaultCheckout()
    }
    stages {
        stage('Checkout SCM') {
            agent { label "nodejs" }
            steps {
                checkout scm
                script {
                    echo "get COMMIT_ID"
                    sh 'echo -n $(git rev-parse --short HEAD) > ./commit-id'
                    commitId = readFile('./commit-id')
                }
                // stash this current workspace
                stash(name: 'ws', includes:'**',./commit-id')
            }
        }
        stage('Initialize') {
            parallel {
                stage("Agent: nodejs") {
                    agent { label "nodejs" }
                    steps {
                        cleanWs()
                        script{
                            def node = tool name: 'NodeJS-
8.9', type: 'jenkins.plugins.nodejs.tools.NodeJSInstallation'
                            env.PATH = "${node}/bin:${env.PATH}"
                        }
                    }
                }
                stage("Agent: Docker") {
                    agent { label "Docker" }
                    steps {

```

```

cleanWs()
script{
    // Setup variable for retagging purpose
    if ( env.BRANCH_NAME == 'master' ){
        projectName = "${params.PRODUCTION_NAMESPACE}"
    }
    registryURL = "${params.VSAN_REGISTRY_URL}"
    envStage = "Production"

} else if ( env.BRANCH_NAME == 'release' ){
    projectName = "${params.STAGING_NAMESPACE}"
    registryURL = "${params.STAGE_REGISTRY_URL}"
    envStage = "Staging"

} else if ( env.BRANCH_NAME == 'develop'){
    projectName = "${params.DEVELOPMENT_NAMESPACE}"
}
registryURL = "${params.VSAN_REGISTRY_URL}"
envStage = "Development"
}

// Defind for Final image name
imageNameFinal = "${registryURL}/${projectName}/${params.DOCKER_IMAGE_NAME}"
}

stage('Unit Test') {
    agent { label "nodejs" }
    steps {
        unstash 'ws'
        echo "Do Unit Test Here"
        script{
            sh "npm install"
            sh "npm run test"
        }
    }
}
stage('SonarQube Analysis') {
    when {

```



```

        withCredentials([string(credentialsId: 'VSAN_OC_REGI
STRY_TOKEN', variable: 'TOKEN')]) {
            sh "docker login ${registryURL} -u jenkins -
p ${TOKEN}"
        }
    } else if ( env.BRANCH_NAME == 'release' ) {
        withCredentials([string(credentialsId: 'STAGE_OC_REGI
STRY_TOKEN', variable: 'TOKEN')]) {
            sh "docker login ${registryURL} -u jenkins -
p ${TOKEN}"
        }
    } else if ( env.BRANCH_NAME == 'develop' ) {
        withCredentials([string(credentialsId: 'VSAN_OC_REGI
STRY_TOKEN', variable: 'TOKEN')]) {
            sh "docker login ${registryURL} -u jenkins -
p ${TOKEN}"
        }
    }
}
stage ('Re-tag Image'){
    steps{
        echo "Retaging Image"
        sh "docker tag ${params.DOCKER_IMAGE_NAME}:${BUILD_NUMBER}-${commitId} \
${imageNameFinal}:latest"
    }
}
stage ('Deploy'){
    steps{
        unstash 'ws'
        //vars envStage, ProjectName and imageNameFinal is from previous deployment stage (re-tagiin image)
        script{
            // this step will be hold until deployment confirmed
            if (env.BRANCH_NAME == 'master'){
                timeout(10) {
                    input message: 'Deploy to PRODUCTION?', ok: 'Deploy'
                }
            }
        }
    }
}

```



```

success{
    node("Docker"){
        script{
            withCredentials([string(credentialsId: 'telegram-
token', variable: 'TELEGRAM_TOKEN')]) {
                if (BRANCH_NAME == 'master' || BRANCH_NAME == 're
lease' || BRANCH_NAME == 'develop'){
                    textMessage = "ヽ(•▽•)/ Jenkins Job ---"
                    ${JOB_NAME}-${BUILD_NUMBER}-${commitId} is SUCCESS"
                    sh "curl -s -
X POST 'https://api.telegram.org/bot${TELEGRAM_TOKEN}/sendMessage?chat_id=${params.CHAT_ID}&text=${textMessage}"
                }
            }
        }
    }
}

```

## 5.9 NODEMON

Siapa pun yang pernah belajar menggunakan Node.js pasti sudah kenal dengan *command* `node app.js`. biasanya kita menggunakan itu untuk menjalankan *code* yang sudah kamu buat. Sayangnya, setiap ada perubahan dalam *code* kita harus mengulang *command* tersebut untuk melihat hasil perubahannya. Nodemon adalah 1 library dari node. Nodemon ini membantu kita dengan merestart *server* secara otomatis ketika ada perubahan di dalam *code* yang Kamu buat. Berikut adalah source code dari library nodemon.

```
{  
  "env":{  
    "JWT_KEY" : "4luckyd4y",  
    "MONGO_CONNECTION_STRING" : "mongodb://localhost/tpnpmms_db",  
    "PORT": 3000,  
    "NODE_ENV":"development",  
    "AWS_ENDPOINT":"telkompartner-storage.apps.playcourt.id",  
    "AWS_KEY": "hhoFX8UYwhIVyoGvWhX",  
    "AWS_SECRET": "i0OsIALS2kh4gSQLq1jKvKdMqqpqDBhVqNupeRtR",  
    "AWS_BUCKET": "product-catalog-service-development",  
    "AWS_PORT": "80",  
    "AWS_SSL": "false",  
    "ADMIN_GATEWAY_URL": "http://localhost:9876",  
    "TPN_URL": "http://localhost:4001",  
    "PRODUCT_CATALOG_URL": "http://localhost:3000",  
    "PURCHASE_ORDER_URL": "http://localhost:3001",  
    "INVOICE_URL": "http://localhost:3002",  
    "SETTLEMENT_URL": "http://localhost:3003",  
    "POLICY_URL": "http://localhost:3004"  
  }  
}
```

## 5.10 PACKAGE

File package.json adalah file yang berisi keterangan dari *project* Javascript, seperti: judul *project*, deskripsi, versi, *library* yang dibutuhkan, dan script untuk mengeksekusi *project*. File ini akan dibutuhkan oleh `npm` dan `yarn`. Secara sederhana, file **package.json** dapat diibaratkan seperti SOP yang akan diikuti oleh `npm` dan `yarn` dalam membangun (*build*) *project*. Berikut package yang digunakan dalam pembuatan fitur didalam aplikasi invoice dan settlement adalah;

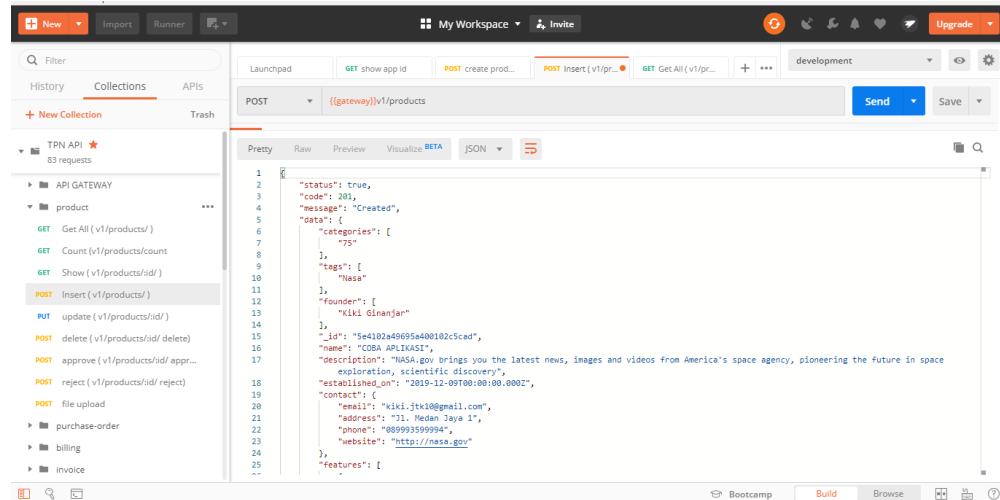
```
{  
  "name": "product-catalog",  
  "version": "0.0.0",  
  "description": "",  
  "scripts": {  
    "start": "node ./bin/www",  
    "test": "cross-env NODE_ENV=test jest --coverage --runInBand --forceExit"  
  },  
  "keywords": [],  
  "author": "iqbalzetto",  
  "license": "ISC",  
  "dependencies": {  
    "axios": "^0.19.0",  
    "bluebird": "^3.5.5",  
    "config": "^3.2.2",  
    "cors": "^2.8.5",  
    "express": "^4.17.1",  
    "express-fileupload": "^1.1.5",  
    "joi": "^14.3.1",  
    "lodash": "^4.17.15",  
    "minio": "^7.0.12",  
    "moment": "^2.24.0",  
    "mongoose": "^5.6.9"  
  },  
  "devDependencies": {  
    "cross-env": "^5.2.0",  
    "eslint": "^6.1.0",  
    "jest": "^24.8.0",  
    "nodemon": "^1.19.1",  
    "supertest": "^4.0.2"  
  }  
}
```

## BAB V TAHAPAN PENGGUNAAN APLIKASI

### 6.1 PENGGUNAAN DENGAN POSTMAN

#### 6.1.1 Create Product

Companion melakukan Create Product terlebih dahulu dengan memilih method post (Insert ( v1/products/ ) lalu tekan Send data akan terinput, berikut adalah gambar create product dalam postman;



The screenshot shows the Postman application interface. In the top navigation bar, there are buttons for 'New', 'Import', 'Runner', and 'Upgrade'. Below the navigation is a search bar and a workspace dropdown labeled 'My Workspace'. The main area shows a collection named 'TPN API' with 83 requests. Under the 'product' section, there are several methods listed: GET All (v1/products/), GET Count (v1/products/count), GET Show (v1/products/:id/), POST Insert (v1/products/), POST update (v1/products/:id/), POST delete (v1/products/:id/delete), POST approve (v1/products/:id/approve), POST reject (v1/products/:id/reject), POST file upload, and others like purchase-order, billing, and invoice. The 'POST Insert (v1/products/)' method is selected. The request details panel shows the method as 'POST' and the URL as '({{gateway}})v1/products'. The preview tab displays a JSON response with status: true, code: 201, message: 'Created', date: '2019-12-09T00:00:00.000Z', categories: [75], tags: [NASA], founder: [KLI Ginanjan], id: '6e410d4d6095d400102c5ced', name: 'COBA APIKASTI', description: 'NASA.gov brings you the latest news, images and videos from America's space agency, pioneering the future in space exploration, scientific discovery', establishedOn: '2019-12-09T00:00:00.000Z', contact: {email: 'KLI.Ivin10@gmail.com', address: 'Jl. Medan Jaya 1', phone: '08999359994', website: 'http://nasa.gov'}, and features: [ ]. The bottom right of the interface has buttons for 'Send', 'Save', and other tools.

#### 6.1.2 Approval Product

Companion menunggu product yang dibuat untuk di approval oleh admin berdasarkan id product. Admin akan memilih method post(approve ( v1/products/:id/ approve)), lalu send, ketika messages memberikan informasi success, product yang telah dibuat telah diapprove.

```

1
2   "status": true,
3   "code": 200,
4   "message": "Success",
5   "data": {
6     "config": {
7       "po_number_periode": "daily",
8       "edukasi_registration_fee": 5000,
9       "unique_code_type": "discount",
10      "invoice_signer": "1234567890",
11      "transaction_type": "cash",
12      "akun_pendapatan": "123123",
13      "akun_piutang": "123123",
14      "akun_bank": "123213123",
15      "product_ownership": "telkom"
16    },
17    "contact": {
18      "email": "kiki.jtk10@gmail.com",
19      "address": "Jl. Medan Jaya 1",
20      "phone": "089993599994",
21      "website": "http://nsa.gov"
22    },
23    "companion": {
24      "organization_company_id": "TELKOM_INDONESIA",
25      "division_id": "DIV-DOS",
26      "companion_id": "DOS-10a00000"
27    }
28  },
29  "status_product": "approved",
30  "product_code": "CO",
31  "__v": 0
32}
33

```

### 6.1.3 SETELAH APPROVE

Setelah product di approve oleh admin, maka status yang tadinya waiting akan otomatis berubah menjadi approved.

```

98
99   "isActive": true,
100  "createdAt": "Mon Feb 10 2020 14:13:40 GMT+0700 (WIB)",
101  "updatedAt": "Mon Feb 10 2020 14:13:40 GMT+0700 (WIB)",
102  "keyId": "7zIhLt0mbstg0WuW2spE9n",
103  "keySecret": "2W2vkd3EAYMlpHxXJA9ga",
104  "scopes": null,
105  "consumerId": "7d7a5466-874c-46d0-8dd0-1695ff223cad",
106  "id": "7zIhLt0mbstg0WuW2spE9n"
107 },
108   "status_product": "approved",
109   "product_code": "CO",
110   "__v": 0
111 }
112

```

### 6.1.4 PRODUCT REJECT

Ketika product direject oleh superadmin maka product akan memberikan notifikasi kepada companion bahwa data product yang dibuat salah, dan companion diberitahu untuk merubah data dari product yang dibuat sebelumnya. Super admin memilih method post (reject ( v1/products/:id/ reject)).seperti pada gambar berikut;

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** {{gateway}}/v1/products/5e40f3a39695a400102c5ca8/reject
- Body (JSON):**

```

1
2   "status": true,
3   "code": 200,
4   "message": "Success",
5   "data": {
6     "config": {
7       "to_number_periode": "yearly",
8       "administration_fee": 1000,
9       "unique_code_type": "addition",
10      "invoice_signer": "I",
11      "transaction_type": "postpaid",
12      "product_ownership": "telkom",
13      "skum_code": "333312345678",
14      "skun_pendapatan": "41711015",
}

```
- Status:** 200 OK
- Time:** 287ms
- Size:** 2.78 KB

## 6.1.5 PRODUCT SETELAH DIREJECT

Ketika product di reject oleh superadmin maka response yang diberikan system sebagai berikut;

The screenshot shows the Postman interface with the following details:

- Request Method:** GET
- URL:** {{gateway}}/v1/products/5e40f3a39695a400102c5ca8
- Body (JSON):**

```

82   },
83   },
84   "app": {
85     "isActive": true,
86     "id": "475a6418-cd9b-4d93-b07b-2d43c0f4c5e2",
87     "userId": "934c3ab7-04a5-4ea8-9d42-c8e141e0ff76",
88     "name": "EDC-TMONEY",
89     "createdAt": "Mon Feb 10 2020 13:09:39 GMT+0700 (WIB)",
90     "updatedAt": "Mon Feb 10 2020 13:09:39 GMT+0700 (WIB)"
91   },
92   "credential": {
93     "isActive": true,
94     "createdAt": "Mon Feb 10 2020 13:09:39 GMT+0700 (WIB)",
95     "updatedAt": "Mon Feb 10 2020 13:09:39 GMT+0700 (WIB)",
96     "KeyId": "3MRmq3wkkupqbYt5UPeQJ",
97     "keySecret": "21665gaeIJKpjwAESy3vyO",
98     "scopes": null,
99     "consumerId": "475a6418-cd9b-4d93-b07b-2d43c0f4c5e2",
100    "id": "3MRmq3wkkupqbYt5UPeQJ"
101  },
102  },
103  ],
104  "status_product": "rejected",
105  "product_code": "OK",
106  "_v": 0
107 }

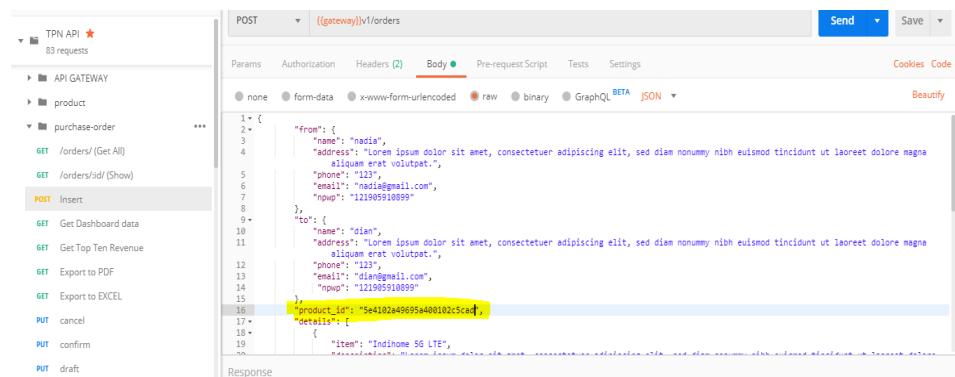
```
- Status:** 200 OK

## 6.1.6 CREATE PURCHASE ORDER DAN BILLING STATEMENT

Setelah product dibuat oleh companion dan super admin telah approve productnya tahap selanjutnya companion membuat purchase order dan billing atas product yang telah dibuat. Berikut adalah contoh pembuatan billing statement oleh companion;

### A. Create Purchase order

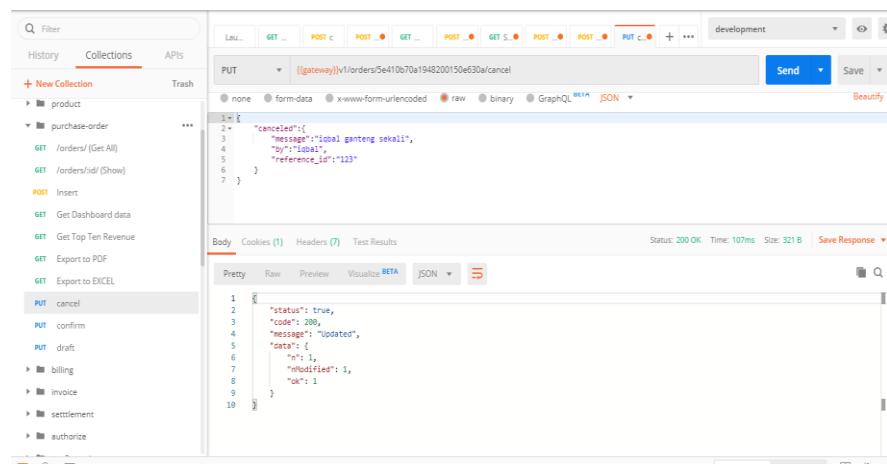
Membuat purchase order dengan memilih method post (INSERT), berikut adalah gambar create purchase order.



```
POST ((gateway))/v1/orders
{
  "from": {
    "name": "nadia",
    "address": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.",
    "phone": "123",
    "email": "nadi@gmail.com",
    "npwp": "121905910899"
  },
  "to": {
    "name": "dian",
    "address": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.",
    "phone": "123",
    "email": "dian@gmail.com",
    "npwp": "121905910899"
  },
  "product_id": "5e4102a49695a400102c5cd",
  "details": [
    {
      "item": "Indihome 5G LTE"
    }
  ]
}
```

### B. Cancel Purchase order

Cancel purchase order dengan memilih method put (CANCEL), berikut adalah gambar cancel purchase order.



```
PUT ((gateway))/v1/orders/5e410b070a1948200150e630a/cancel
{
  "canceler": {
    "message": "lital ganteng sekali",
    "by": "lital",
    "reference_id": "123"
  }
}
```

Body Cookies (1) Headers (7) Test Results Status: 200 OK Time: 107ms Size: 321 B Save Response ▾

```
Pretty Raw Preview Visualize Beta JSON
```

## C. Confirm Purchase order

Confirm purchase order dengan memilih method put (CONFIRM), berikut adalah gambar confirm purchase order.

The screenshot shows the Postman interface with the following details:

- Collection:** development
- Request Type:** PUT
- URL:** `({gateway})v1/orders/5e410b70a1948200150e630a/confirm`
- Body (JSON):**

```
1  {
2     "confirmed": {
3         "by": "ideal",
4         "reference_id": "123"
5     }
6 }
```
- Status:** 401 Unauthorized
- Time:** 267ms
- Size:** 289 B
- Save Response:** Available

## D. Draft Purchase order

Draft purchase order dengan memilih method put (CONFIRM), berikut adalah gambar Draft purchase order.

The screenshot shows the Postman interface with the following details:

- Collection:** development
- Request Type:** PUT
- URL:** `({gateway})v1/orders/5e410b70a1948200150e630a/draft`
- Headers (11):** Query Params
- Body (JSON):**

```
1  [
2      {
3          "status": false,
4          "code": 401,
5          "message": "Unauthorized Access"
6      }
7  ]
```
- Status:** 401 Unauthorized
- Time:** 465ms
- Size:** 289 B
- Save Response:** Available

## E. Create Billing Statement

Create billing statement dengan memilih method post (INSERT), Send lalu respond yang akan diberikan sebagai berikut;

The screenshot shows the Postman interface with a collection named 'billing'. A POST request is selected with the URL `(gateway)/v1/billings`. The Body tab contains the following JSON:

```
1 - {  
2   "po_id": "5e410ce4a1948200150e6311",  
3   "billing_date": "2019-12-18 10:00:00",  
4   "due_date": "2019-12-18 10:00:00",  
5   "is_overdue_invoice": false,  
6   "config_id": "1",  
7   "kode_waktu": "123",  
8   "affiliasi": "123",  
9   "created": {  
10    "by": "label",  
11    "reference_id": "123"  
12  },  
13  "bank": {  
14    "name": "Klik Garut",  
15    "account_number": "2194982104",  
16    "bank_name": "Bank Mandiri",  
17    "branch_office": "Jl. Kebayoran"  
18  }  
21 }
```

The response status is 201 Created, Time: 295ms, Size: 3.04 KB.

## F. Cancel Billing Statement

Cancel billing statement dengan memilih method put (CANCEL), Send lalu respond yang akan diberikan sebagai berikut;

The screenshot shows the Postman interface with a collection named 'billing'. A PUT request is selected with the URL `(gateway)/v1/billings/5e410ce4a1948200150e6311/cancel`. The Body tab contains the following JSON:

```
1 - {  
2   "canceled": {  
3     "message": "label garut",  
4     "by": "label",  
5     "reference_id": "123"  
6   }  
7 }
```

The response status is 404 Not Found, Time: 130ms.

## G. Paid Billing Statement

Invoice billing statement dengan memilih method put (PAID),

Send lalu respond yang akan diberikan sebagai berikut;

The screenshot shows the Postman interface with the following details:

- Collection:** development
- Request Type:** PUT
- URL:** `[[gateway]]v1/billings/5e410ce4a1948200150e6311/paid`
- Body (JSON):**

```
1 <!
2   "paid": {
3     "by": "label",
4     "reference_id": "123"
5   }
6 >
```
- Status:** 404 Not Found
- Time:** 280ms
- Size:** 310 B
- Save Response:** checked

## H. In-Progres Billing Statement

In-Progres billing statement dengan memilih method put (INPROGRES), Send lalu respond yang akan diberikan sebagai berikut;

The screenshot shows the Postman interface with the following details:

- Collection:** development
- Request Type:** PUT
- URL:** `[[gateway]]v1/billings/5e410ce4a1948200150e6311/in-progress`
- Body (JSON):**

```
1 <!
2   "returned": {
3     "by": "label",
4     "reference_id": "123"
5   }
6 >
```
- Status:** 404 Not Found
- Time:** 215ms
- Size:** 329 B
- Save Response:** checked

## I. Deliver Billing Statement

In-Progres billing statement dengan memilih method put (INPROGRES), Send lalu respond yang akan diberikan sebagai berikut;

The screenshot shows the Postman interface with the following details:

- Collection:** development
- Request Type:** POST
- URL:** `(@gateway)v1/billings/5e153f8684ee420d41a075fb/deliver`
- Body (JSON):**

```
{"by": "iqbal", "reference_id": "123"}
```
- Headers:** (2)
- Params:** none
- Authorization:** none
- Tests:** none
- Settings:** none

### 6.1.7 SETTLEMENT/PAYMENT

#### A. Create Payment

Membuat payment dengan memilih method post (INSERT), berikut adalah gambar create payment.

The screenshot shows the Postman interface with the following details:

- Collection:** development
- Request Type:** POST
- URL:** `(@gateway)v1/payments/`
- Body (JSON):**

```
1  {
2     "Invoice_Id": "5dd4bf4105e0de12eb923ca9",
3     "Branch": "branch1",
4     "account": {
5         "transaction_date": "2019-04-01",
6         "description": "asdf",
7         "ref_no": "123 asdf",
8         "debit": 123123,
9         "credit": 123123,
10        "balance": 123123123,
11        "branch_code": "1234"
12    },
13    "settlement_type": "direct",
14    "credit": 123123,
15    "by": "label",
16    "reference_id": "123"
17 }
```
- Headers:** (11)
- Params:** none
- Authorization:** none
- Tests:** none
- Settings:** none

At the bottom, it shows: Status: 400 Bad Request Time: 201ms Size: 298 B Save Response

## B. Cancel Purchase order

Cancel purchase order dengan memilih method put (CANCEL), berikut adalah gambar cancel purchase order.

The screenshot shows the Postman interface with the following details:

- URL:** `(gateway)v1/orders/5e410b70a1948200150e630a/cancel`
- Method:** PUT
- Body (JSON):**

```
1: {
2:   "cancelled": {
3:     "message": "label ganteng sekali",
4:     "by": "label",
5:     "reference_id": "123"
6:   }
7: }
```
- Response Body (JSON):**

```
1: {
2:   "status": true,
3:   "code": 200,
4:   "message": "Updated",
5:   "data": [
6:     {
7:       "id": 1,
8:       "modified": 1,
9:       "ok": 1
10:  }
11: }
```

## C. Confirm Purchase order

Confirm purchase order dengan memilih method put (CONFIRM), berikut adalah gambar confirm purchase order.

The screenshot shows the Postman interface with the following details:

- URL:** `(gateway)v1/orders/5e410b70a1948200150e630a/confirm`
- Method:** PUT
- Body (JSON):**

```
1: {
2:   "confirmed": {
3:     "by": "label",
4:     "reference_id": "123"
5:   }
6: }
```
- Response Status:** 401 Unauthorized

## D. Draft Purchase order

Draft purchase order dengan memilih method put (CONFIRM), berikut adalah gambar Draft purchase order.

The screenshot shows the Postman interface with the following details:

- Method:** PUT
- URL:** {{(gateway)}}/v1/orders/5e410b70a1948200150e630a/draft
- Body:** (JSON)
 

```

1 {
2   "status": false,
3   "code": 401,
4   "message": "Unauthorized Access"
5 }
```
- Status:** 401 Unauthorized
- Time:** 465ms
- Size:** 289 B

## E. Create Billing Statement

Create billing statement dengan memilih method post (INSERT), Send lalu respond yang akan diberikan sebagai berikut;

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** {{(gateway)}}/v1/billing
- Body:** (JSON)
 

```

1 {
2   "po_id": "5e410b70a1948200150e6311",
3   "billable": "2019-12-18 10:00:00",
4   "due_date": "2019-12-18 10:00:00",
5   "include_tax_invoice": false,
6   "config": {
7     "name": "Kiki Ginenjan",
8     "email": "kiki@kiki.com",
9     "address": "Jl. Kebayoran",
10    "city": "Jakarta Selatan",
11    "zip_code": "12345",
12    "reference_id": "123"
13  },
14  "bank": {
15    "owner": "Kiki Ginenjan",
16    "account_number": "21949382194",
17    "bank_name": "Bank Mandiri",
18    "branch_office": "Jl. Kebayoran"
19  }
20 }
```
- Status:** 201 Created
- Time:** 295ms
- Size:** 3.04 KB

## F. Cancel Billing Statement

Cancel billing statement dengan memilih method put (CANCEL), Send lalu respond yang akan diberikan sebagai berikut;

The screenshot shows the Postman interface with the following details:

- Collection:** development
- Request Type:** PUT
- URL:** {{gateway}}/v1/billings/5e410ce4a1948200150e6311/cancel
- Headers:** (11) - Authorization, Headers (11)
- Body:** (raw, JSON)  
A JSON object with the following content:

```
1 = [
2     "canceled": {
3         "message": "label garuteng",
4         "by": "label1",
5         "reference_id": "123"
6     }
7 ]
```
- Status:** 404 Not Found
- Time:** 130ms
- Size:** 329 B

## G. Paid Billing Statement

Invoice billing statement dengan memilih method put (PAID), Send lalu respond yang akan diberikan sebagai berikut;

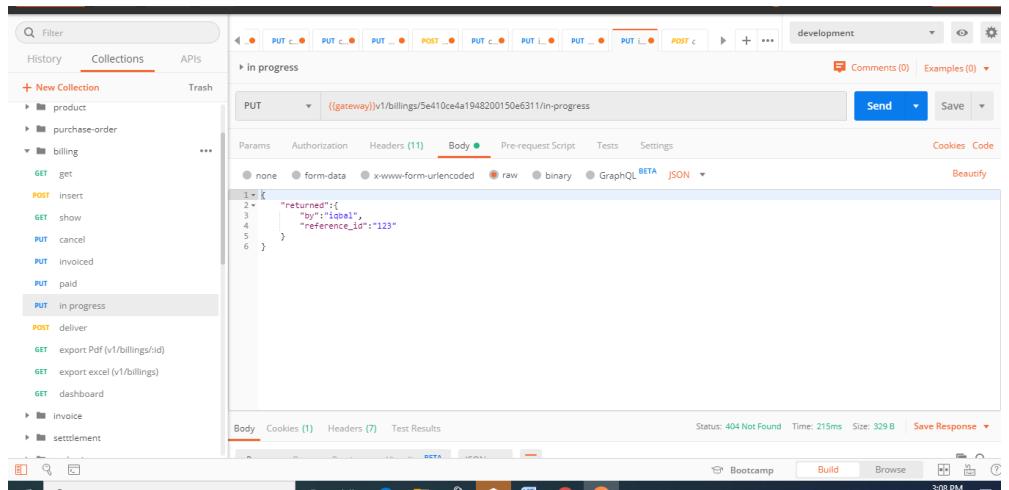
The screenshot shows the Postman interface with the following details:

- Collection:** development
- Request Type:** PUT
- URL:** {{gateway}}/v1/billings/5e410ce4a1948200150e6311/paid
- Headers:** (11) - Authorization, Headers (11)
- Body:** (raw, JSON)  
A JSON object with the following content:

```
1 = [
2     "paid": {
3         "by": "label",
4         "reference_id": "123"
5     }
6 ]
```
- Status:** 404 Not Found
- Time:** 280ms
- Size:** 310 B

## H. In-Progres Billing Statement

In-Progres billing statement dengan memilih method put (INPROGRES), Send lalu respond yang akan diberikan sebagai berikut;



## I. Deliver Billing Statement

In-Progres billing statement dengan memilih method put (INPROGRES), Send lalu respond yang akan diberikan sebagai berikut;

The screenshot shows the Postman application interface. On the left, there's a sidebar with tabs for History, Collections, APIs, and Trash. The Collections tab is selected, showing a list of collections: product, purchase-order, billing, get, insert, show, cancel, invoiced, paid, in progress, deliver, export Pdf, export excel, dashboard, invoice, and settlement. The 'deliver' collection is currently selected. The main workspace on the right displays a collection named 'deliver'. Below it, a specific POST request is shown with the URL `((gateway))v1/billings/{5e153f0684ee420d41a075fb}/deliver`. The 'Body' tab is active, showing the following form-data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> file	tppapi_internal.yml	
<input type="checkbox"/> file	Select Files	
<input checked="" type="checkbox"/> delivered	{"by": "qbali", "reference_id": "123"}	
Key	Value	Description

Below the body editor, there's a section labeled 'Response' which is currently empty. At the bottom of the interface, there are various buttons and icons for Bootstrap, Build, Run, and other tools.

## 6.1.8