BAB II
LANDASAN TEORI

Pada bab ini membahas tentang teori, perangkat dan software penunjang yang digunakan pada Layanan *Internet of Things* (IoT) menggunakan *Virtual Privat Server* (VPS). Pembahasan mencakup *Virtual Privat Server* (VPS), Node Js, MongoDB, *RESTful API, Internet lamp*, *Microcontroller* ESP8266

2.1 Virtual Private Server (VPS)

VPS *(Virtual Private Server)* adalah sebuah metode membagi sumber daya pada sebuah *physical server* menjadi beberapa Virtual Server. VPS sering juga disebut sebagai teknologi *server side* yang memungkinkan sebuah mesin dengan kapasitas besar dibagi ke beberapa virtual mesin. Tiap virtual mesin ini melayani sistem operasi dan perangkat lunak secara mandiri dan dengan berbagia konfigurasinya[3].

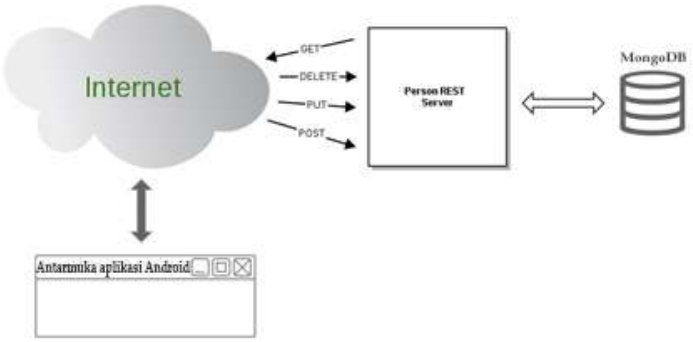
VPS *(Virtual Private Server)* merupakan bentuk virtual dari suatu server, yaitu dari sebuah server dibentuk atau dibagi menjadi beberapa virtual server, sehingga satu buah server dapat menampung beberapa VPS secara bersamaan. Secara fungsi tidak jauh berbeda dengan penggunaan server atau komputer pada umumnya, namun komputer atau server tersebut tidak secara *real* dapat di lihat, sehingga pengaturan untuk mematikan, menghidupkan, *install* sistem oprasi, dan lain-lain dapat dilakukan melalui *control panel* atau *console* yang diberikan oleh penyedia VPS tersebut. Pada gambar 2-1 menunjukan ilustrasi atau gambaran dari *Virtual Private Serve* (VPS) [3].



Gambar 2-1 Gambaran *Virtual Private Server* (VPS)

2.2 Node Js

Node.js adalah *platform* perangkat lunak pada sisi server dan aplikasi jaringan. Ditulis dengan bahasa JavaScript dan dijalankan pada Windows, Mac OS X, dan Linux. Node.js memiliki pustaka server HTTP sendiri sehingga memungkinkan untuk menjalankan server web tanpa menggunakan program server web seperti Apache[4]. Pada gambar 2-2 menunjukan cara kerja dari Node Js



Gambar 2-2 Cara Kerja Node Js

2.2.1 Kelebihan Node Js

Beberapa kelebihan dari server Node Js yaitu:

1. Dengan bahasa JavaScript, mempermudah memepelajari pemrograman sisi belakang *(back-end).* Jika memang sudah menguasai JavaScript, pemula dapat lebih cepat menguasainya.
2. Adanya pertukaran kode antara klien dan server, yaitu *server-side rendering* pada kerangka JavaScript.
3. Adanya fasilitas untuk membuat aplikasi waktu nyata *(realtime application).*
4. Open source, sehingga pengguna mengetahui bagaimana proses aplikasi berjalan, dapat mengubahnya, dan digunakan secara gratis[4].

2.3 Basis Data

Basis Data terdiri dari dua kata, yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau Gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pedagang), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang diwujudkan dalam bentuk angka, huruf, simbol, teks gambar, bunyi, atau kombinasinya. Maka dari itu Basis Data (*Database*) sendiri

dapat didefinisikan sebagai himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah[5].

2.4 MongoDB

MongoDB adalah salah satu produk basis data *noSQL OPEN SOURCE* yang menggunakan struktur data JSON untuk menyimpan datanya. Data pada MongoDB dikenal sebagai *document*. MongoDB menggunakan BSON sebagai format penyimpanan data untuk *document*. MongoDB sering dipakai untuk aplikasi berbasis *Cloud, Grid Computing*, atau *Big Data*. MongoDB telah dipakai oleh perusahaan besar seperti *SourceForge.net, CodeAcademy*, *FourSquare* dan *Craiglist*[6].

2.4.1 Kelebihan MongoDB

Ada beberapa kelebihan dari MongoDB yaitu:

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 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. *Cross-platform*, sehingga dapat digunakan di Windows, Linux, OS X dan Solaris.
5. Proses CRUD *(Create, Read, Update, Delete)* terasa sangat ringan[6].

2.5 RESTful API

REST atau *Representation State Transfer* merupakan bagian dari web service menjalankan oprasional CRUD dalam bentuk *Create* (membuat berkas atau *file* untuk document data), *Read* (membaca berka *file* atau dokumen data), *Update* (melakukan pembaharuan terhadap berkas *file* atau dokumen data), dan *Deleted* (melakukan penghapusan terhadap berkas atau dokumen data), memanfaatkan sumber daya yang telah terontentikasi melalui URL pada jaringan komputer(khusus internet). Sedangkan untuk implementasi proses CRUD yang dilakukan oleh REST, dapat menggunakan bahasa pemrograman yang dipilih. Misalnya dengan Java, PHP, ASP, dan lain-lain. REST menggunakan protokol HTTP kerana dihubungkan menggunakan jaringan komputer (internet)[7].

REST / RESTful merupakan implementasi dari API *(Application Programming Interface)* berfungsi memberikan layanan data mentah sehingga *developer* bertanggung jawab penuh untuk data-data tersebut yang akan digunakan, apakah akan ditampilkan ke browser web atau digunakan pada aplikasi mobile android atau akan digunakan untuk mengaktifkan device *smarthome* seperti Arduino, Raspberry Pi dan lain-lain. Untuk aplikasi REST ini, data disimpan di database melalui operasi *CRUD (Create, Read, Update, Delete)* yang dapat diakses melalui *request* API[8]. Gambar 2-3 menunjukan alur kerja RESTful API.



Gambar 2-3 Alur Kerja RESTful API

2.5.1 Komponen RESTful API

Ada 4 komponen yang terdapat dalam RESTful API diantaranya adalah:

1. Desain URL

   RESTful API diakses menggunakan protokol HTTP. Penamaan dan struktur URL yang konsisten akan menghasilkan API yang baik dan mudah untuk

dimengerti developer. URL API bisa disebut *endpoint* dalam pemanggilannya Contoh penamaan URL/endpoint yang baik adalah seperti berikut :

```
/users
/users/1234
/users/1234/photos
/users/1234/photos/abc
```

2. HTTP Verbs

   Setiap *request* yang dilakukan terdapat metode yang dipakai agar *server* mengerti apa yang sedang di *request client*, diantaranya yang umum dipakai adalah :

   a) *GET*

      GET adalah metode HTTP *Request* yang paling sederhana, metode ini digunakan untuk membaca atau mendapatkan data dari sumber.

      Contoh :

      GET /users : Mengembalikan daftar *user*

      GET /users/1234 : Mengembalikan data *user* dengan ID 1234

   b) *POST*

      POST adalah metode HTTP *Request* yang digunakan untuk membuat data baru dengan menyisipkan data dalam *body* saat *request* dilakukan.

      Contoh :

      POST /users : Membuat data *user* baru

   c) *PUT*

      PUT adalah metode HTTP *Request* yang biasanya digunakan untuk melakukan *update* data *resource*.

      Contoh :

      PUT /users/1234 : Memperbarui data *user* dengan ID 1234

   d) *DELETE*

      DELETE adalah metode HTTP *Request* yang digunakan untuk menghapus suatu data pada *resource*.

      Contoh :

      DELETE /users/1234 : Menghapus data *user* dengan ID 1234

   Selain HTTP Verbs diatas, masih ada metode *HEAD* dan *PATCH* dalam HTTP *Request*, tetapi jarang sekali digunakan.

3. HTTP Response Code

   HTTP *response* code adalah kode standarisasi dalam menginformasikan hasil request kepada client. Secara umum terdapat 3 kelompok yang biasa dijumpai pada RESTful API yaitu :

   a) 2XX : adalah response code yang menampilkan bahwa request berhasil.
   b) 4XX : adalah response code yang menampilkan bahwa request mengalami kesalahan pada sisi client.
   c) 5XX : adalah response code yang menampilkan bahwa request mengalami kesalahan pada sisi server.

4. Format Response

   Setiap *request* yang dilakukan *client* akan menerima data *response* dari *server*, *response* tersebut biasanya berupa data XML ataupun JSON. Setelah mendapatkan data *response* tersebut barulah *client* bisa menggunakannya dengan cara memparsing data tersebut dan diolah sesuai kebutuhan[8].

2.6 Android

Android adalah sebuah sistem oprasi untuk perangkat *mobile* berbasis linux yang mencakup sistem oprasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka[9].

Android merupakan generasi baru *platform mobile, platform* yang memberikan pengembang untuk melakukan pengembangan sesuai yang diharapkannya. Sisitem oprasi yang mendasari Android dilisensikan di bawah GNU, *General Public Lisensi* Versi 2 (GPLv2), yang sering dikenal dengan "*copyleft*" lisensi di mana setiap perbaikan pihak ketiga harus terus jatuh dibawah terms[9].

2.6.1 Arsitektur Android

Secara garis besar Arsitektur Android dapat dilihat pada gambar 2-4 dan dapat dijelaskan sebagai berikut:

1. *Appilcations* dan *Widgets*

   Applications dan *Widgets* ini adalah *layer* di mana *user* berhubungan dengan aplikasi saja, di mana biasanya *user download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut.

2. *Application Frameworks*

   Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberik kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif.

3. *Libraries*

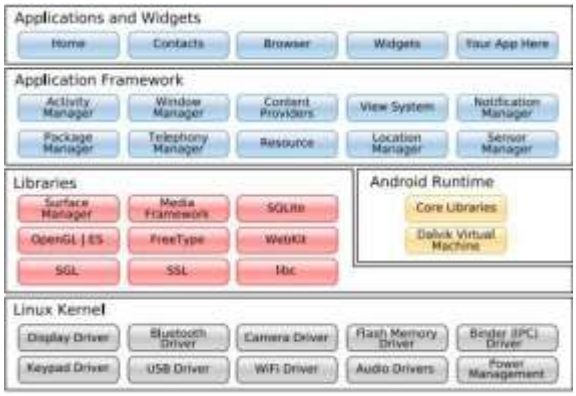
   Libraries ini adalah *layer* di mana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya.

4. Android *Run Time*

   Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux.

5. Linux *Karnel*

   Linux *Karnel* adalah *layer* di mana inti dari *oprating* sistem dari Android itu berada[9].



Gambar 2-4 Arsitektur Android

2.7 *Internet lamp*

Internet lamp yaitu lampu yang dapat terhubung ke jaringan internet dan dapat dikontrol dari mana saja menggunakan *Smartphone*. *Internet lamp* adalah produk yang dihasilkan dari perancagnan ini yang dibangun dari beberapa komponen yaitu *Microcontroller* ESP8266, Relay dan Lampu, juga sebagai alat uji sistem layanan IoT yang dibangun.

2.8 *Microcontroller* ESP8266

Microcontroller ESP8266 adalah sebuah *Microcontroller* pengembangan berbasis modul microcontroller ESP8266. *Microcontroller* ESP8266 dibuat sebagai solusi dari mahalnya sebuah sistem wireless berbasis *Microcontroller*

lainnya. Dengan menggunakan *Microcontroller* ESP8266 biaya yang dikeluarkan untuk membangun sistem wifi berbasis *Microcontroller* relatif murah [10].

Perbedaan pada Microcontroller ini yaitu kemampuannya untuk menyedikan fasilitas konektifitas wifi dengan mudah serta memory yang digunakan sangat besar yaitu 4 MB[10]. Gambar 2-5 menunjukan *Microcontroller* ESP8266.



Gambar 2-5 *Microcontroller* ESP8266

2.8.1 *Microcontroller* Chipset pada *Microcontroller* Wemos

Pada Microcontroller wemos memiliki 2 buah chipset yang digunakan sebagai otak kerja *platform* tersebut. Beberapa chipset pada *Microcontroller* ini adalah :

1. Chipset ESP8266.

   ESP8266 adalah sebuah chip *microcontroller* yang memiliki fitur wifi. Diproduksi oleh produsen Cina yang berbasis di Shanghai.

2. Chipset CH340.

   CH340 adalah sebuah Chipset yang mengubah USB menjadi *serial interface*. CH340 digunakan untuk berhubungan dengan bus USB secara langsung[10].