BAB II
TEORI PENUNJANG

2.1 Employee Seeker

Employee Seeker atau dalam bahasa Indonesia yaitu pencarian pekerja adalah proses penentuan/pemilihan karyawan, pegawai, atau pekerja yang dilakukan oleh pencari kerja yang memenuhi syarat dan sesuai dengan ketentuan yang ditentukan pencari kerja [5]. Kita dapat melihat di era saat ini banyak para startup-startup yang mencoba mengembangkan bisnisnya, biasanya salah satu kendala dalam pembuatan usaha adalah karyawan yang nantinya bakal bekerja, tetapi juga ada beberapa pengusaha ataupun pebisnis yang membutuhkan karyawan ketika dibutuhkan saja atau biasa disebut paruh waktu, karena ada beberapa momen yang membuat mereka kekurangan tenaga kerja, tetapi itu hanya di waktu-waktu tertentu, ada beberapa pekerjaan yang dapat dilakukan secara paruh waktu.

2.1.1 Bisnis Katering

Pada bisnis ini tidak sedikit yang membutuhkan pekerja paruh waktu, terlebih lagi pesanan makanan yang datang tidak dapat di perkirakan jumlahnya. Bisnis ini lebih membutuhkan keahlian memasak, karena pesanan makanan yang banyak dan harus memasak dalam porsi yang besar, itu dapat membutuhkan karyawan yang banyak juga untuk dapat selesai sesuai target.

2.1.2 Pekerja Rumah Tangga (PRT)

Setiap keluarga memiliki kesibukan masing-masing, dan biasanya setiap tahunnya seluruh keluarga melakukan mudik ke kampung halaman. Kendalanya adalah tidak adanya orang yang mengurus pekerja dirumah, mulai dari mencuci, nyapu, memasak, dan lain-lain. Disinilah dibutuhkannya pekerja rumah tangga yang hanya bekerja selama pemilik rumah tidak ada ditempat atau bisa dibilang bekerja selama paruh waktu.

2.2 Employee

Employee bila diartikan kedalam bahasa Indonesia menurut kamus besar bahasa Indonesia adalah pekerja, karyawan, atau pegawai. Tenaga kerja, pekerja, karyawan sebagai penggerak organisasi dalam mewujudkan eksistensinya atau potensi yang merupakan asset dan berfungsi seabgai modal non material dalam organisasi bisnis yang dapat diwujudkan menjadi potensi nyata secara fisik dan non fisik dalam mewujudkan eksistensi organisasi [5].

Karyawan adalah setiap orang yang bekerja dengan menjual tenaganya (fisik dan pikiran) kepada suatu perusahaan dan memperoleh balas jasa yang sesuai dengan perjanjian (Hasibuan, 2009). Sedangkan definisi pegawai menurut (Mardiasmo,2010) adalah orang pribadi yang bekerja pada pemberi kerja baik sebagai pegawai tetap atau pegawai pegawai tidak tetap/ tenaga kerja lepas berdasarkan perjanjian atau kesepakatan kerja baik secara tertulis maupun tidak tertulis, untuk melaksanakan suatu pekerjaan dalam jabatan atau kegiatan tertentu dengan memperoleh imbalan yang dibayarkan berdasarkan periode tertentu.

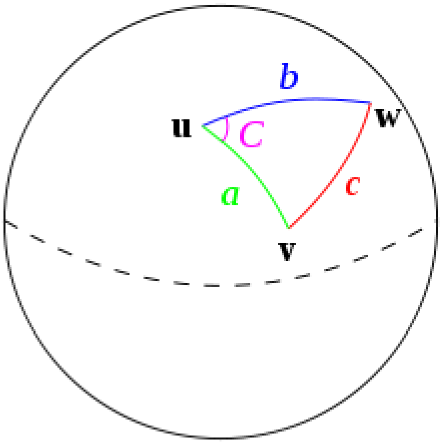
Berdasarkan uraian diatas dapat disimpulkan bahwa pegawai adalah aset organisasi yang paling berharga. Pengetahuan dan keahlian mereka mempengaruhi kualitas barang dan jasa yang diberikan ke para pelanggan. Pegawai terbagi menjadi dua macam, pegawai tetap dan pegawai tidak tetap/tenaga kerja lepas atau yang biasa disebut pekerja paruh waktu *part time*, pekerja paruh waktu adalah orang yang mencari pekerjaan tambahan untuk memenuhi kebutuhan pribadi, pekerja paruh waktu biasanya adalah orang orang yang sedang menempuh pendidikan yang memenuhi syarat usia, seperti mahasiswa pelajar.

2.3 Haversine

Metode *Haversine* digunakan untuk menghitung jarak antara titik di permukaan bumi menggunakan garis lintang (*longitude*) dan garis bujur (*lattitude*) sebagai *variabel input*. *Haversine formula* adalah persamaan penting pada *navigasi*, memberikan jarak lingkaran besar antara dua titik pada permukaan bola (bumi) berdasarkan bujur dan lintang. [6]

Pada penelitian yang dilakukan Ingole dan Nichat melakukan penelitian untuk mencari *rute* jalur terpendek menggunakan *Haversine Formula*. Penelitian ini berhasil menerapkan *algoritama haversine formula* dengan dengan cara *user* menginputkan lokasi awal dan lokasi akhir pada *Java Processing Application*. Dari hasil penelitian yang di lakukan peneliti ingin mengembangkan supaya metode dapat di implementasikan dengan memanfaatkan GPS dan *web service* [7].

Haversine Formula adalah persamaan yang digunakan dalam *navigasi*, yang memberikan jarak lingkaran besar antara dua titik pada permukaan bola (bumi) berdasarkan bujur dan lintang [8]. Dengan mengasumsikan bahwa bumi berbentuk bulat sempurna dengan jari-jari R 6.367, 45 km, dan lokasi dari 2 titik di koordinat bola (lintang dan bujur) masing-masing adalah lon1, lat1, dan lon2, lat2, maka rumus *Haversine* dapat ditulis dengan persamaan sebagai berikut :



Gambar 3-1 Algoritma Haversine

Dalam unit bola, sebuah “segitiga” pada permukaan bola didefinisikan sebagai lingkaran-lingkaran besar yang menghubungkan tiga poin u, v, dan w pada bola. Jika panjang dari ketiga sisi adalah (dari u ke v), b (dari u ke w), dan c (dari v ke w), dan sudut sudut yang berlawanan c adalah C, maka hukum *haversines* menjadi:

Haversin (c) = haversine (a - b) + sin(a) sin(b) haversine (C).

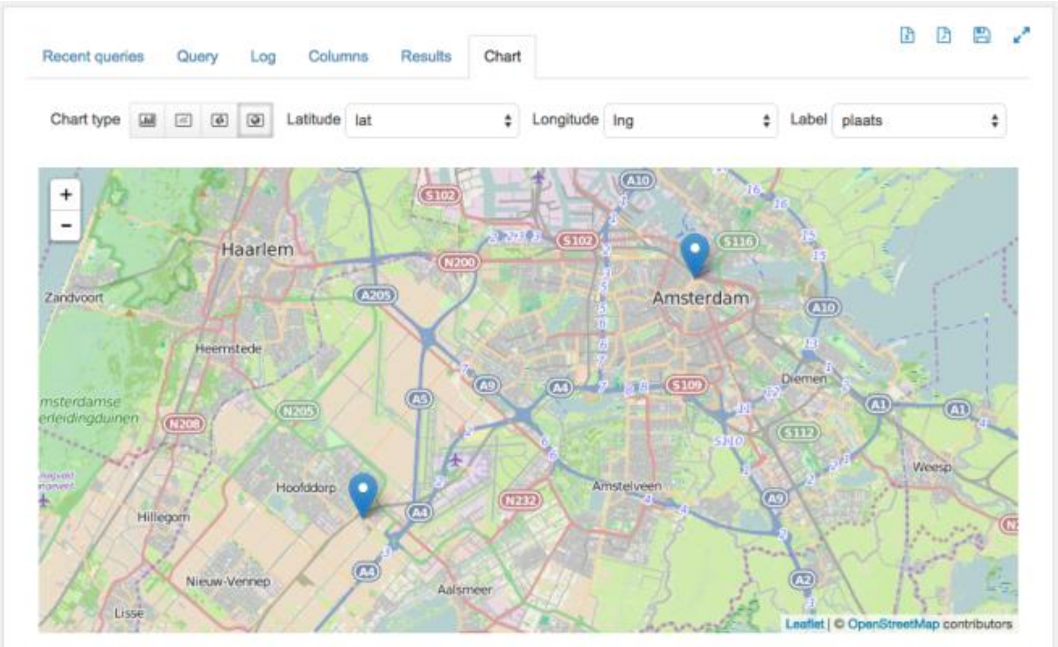
Sama halnya testing yang dilakukan oleh Alexander, Bij (2019). Dia menggunakan *query* di bawah ini untuk menemukan jarak dari Hoofddorp Station ke Amsterdam *Central Station*

```
select
    2 * asin(
      sqrt(
        cos(radians(52.2909264998)) *
        cos(radians(52.3773759354)) *
        pow(sin(radians((4.700868765513 - 4.896747677825)/2)), 2)
            +
        pow(radians(sin((52.2909264998 - 52.3773759354)/2)), 2)
      )
    ) * 6371 distance_km;

-- result: 16.415929129056497
```

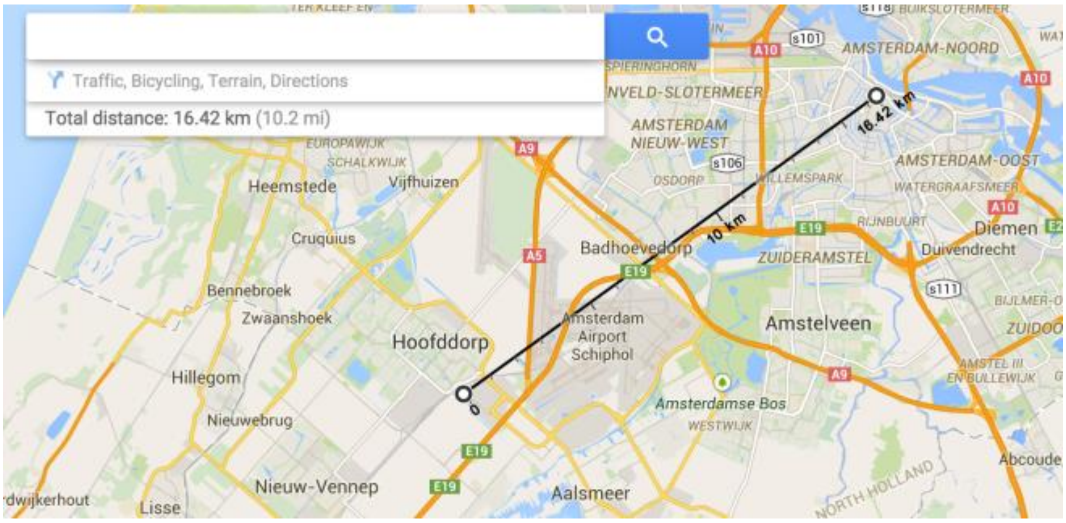
Hoofddorp Station: 52.2909264998, 4.700868765513

Amsterdam Station: 52.3773759354, 4.896747677825



Gambar 3-2 Peta Hoofddorp Station ke Amsterdam Central Station

Dan untuk memferifikasi hasil jarak yang ditemukannya, Alexander Bij membuktikannya menggunakan *google-maps-calculate-distance* [9].



Gambar 3-3 Jarak Hoofddorp Station ke Amsterdam Central Station

Haversine: 16.415929129056497

Google Distance: 16,42

Bukan hanya dalam bentuk *query*, ada banyak bahasa pemograman yang bisa digunakan untuk memanfaatkan formula ini, salah satunya bahasa pemograman C#:

```
public
static
class
Haversine
{
        public static double calculate(double lat1, double lon1, double lat2, double lon2) {
            var R = 6372.8; // In kilometers
            var dLat = toRadians(lat2 - lat1);
            var dLon = toRadians(lon2 - lon1);
            lat1 = toRadians(lat1);
            lat2 = toRadians(lat2);

            var a = Math.Sin(dLat / 2) * Math.Sin(dLat / 2) + Math.Sin(dLon / 2) *
    Math.Sin(dLon / 2) * Math.Cos(lat1) * Math.Cos(lat2);
            var c = 2 * Math.Asin(Math.Sqrt(a));
            return R * 2 * Math.Asin(Math.Sqrt(a));
        }
```

```
    public static double toRadians(double angle) {
      return Math.PI * angle / 180.0;
    }
}

void Main() {
    Console.WriteLine(String.Format("The distance between coordinates {0},{1} and {2},{3} is: {4}",
36.12, -86.67, 33.94, -118.40, Haversine.calculate(36.12, -86.67, 33.94, -118.40)));
}

// Returns: The distance between coordinates 36.12,-86.67 and 33.94,-118.4 is: 2887.25995060711
```

Hasil keluarannya : 2887.26

2.4 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah Bahasa untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan *artifacts* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, artifact tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak. UML dibuat oleh salah satu tokoh dari *Grady Booch*, *James Rumbaugh* dan *Ivar Jacobson* . [10]

2.1.3 Bagian – bagian dari UML

Bagian-bagian utama dari UML, antara lain :

1. *View*

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. Ada beberapa jenis *view* dalam UML antara lain sebagai berikut:

a. *Use Case View*

Ialah bentuk fungsionalitas dari sistem yang diinginkan oleh *user* (dalam hal ini aktor). *View* ini digambarkan dalam *user case* diagram dan seringkali juga dibuat dalam *class diagram*. *View* digunakan dalam pelanggan, perancang, pengembang dan penguji sistem.

b. *Logical View*

Mendeskripsikan bagaimana f*ungsionalitas* dari sistem, struktur statis (*class*, *object* dan *ralationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* digunakan untuk perancang dan pengembang.

c. *Component View*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen ini ialah tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrasi lainnya. *View* digunakan untuk pengembang.

d. *Deployment View*

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat dan bagaimana hubungannya dengan lainnya. *View* digunakan untuk pengembang, pengintegrasi dan penguji.

2. *Diagram*

Diagram ialah presentasi grafis dari sekumpulan elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu. Adapun jenis diagram antara lain:

a. *Use Case Diagram*

Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* ialah deskripsi fungsi yang disediakan oleh sistem dalam bentuk *teks* sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagrams*. *Use case* digambarkan hanya dilihat dari luar aktor dan bukan fungsi yang ada di dalam sistem.

b. *Class Diagram*

Menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara, yaitu: *associated, dependent, specialed* atau *package*.

c. *Statechart Diagram*

Menggambarkan semua *state* yang dimiliki oleh suatu *object* dari suatu class dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik.

d. *Sequence Diagram*

Menggambarkan semua *state* yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik

e. *Collaboration Diagram*

Menggambarkan kolaborasi dinamis seperti *sequnce diagram*. Dalam menunjukkan pertukaran pesan, *collaboration diagram* menggambarkan *object* dan hubungannya.

f. *Activity Diagram*

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.

g. *Component Diagram*

Menggambarkan alokasi semua kelas dan *object* kedalam komponen-komponen dalam desain fisik sistem *software*. Diagram ini memperlihatka pengaturan dan ketergantungan antara komponen-komponen *software* seperti *source code, binary code* dan komponen yang tereksekusi.

h. Deployment Diagram

Menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak sistem, menunjukkan hubungan komputer dengan perangkat satu sama lain dan jenis hubungannya.

2.5 Sistem Informasi

Sistem adalah sekelompok unsur yang erat hubungannya satu dengan yang lain yang berfungsi bersama-sama untuk mencapai tujuan tertentu. Pendekatan sistem yang lebih menekankan pada prosedur : “Sistem adalah suatu jaringan kerja prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu” [11].

Informasi adalah salah satu sarana untuk memperkenalkan suatu perusahaan atau organisasi, sangat erat hubungannya dengan perkembangan organisasi yang masih dalam tahap perkembangan. Dengan tidak adanya informasi, maka suatu organisasi tidak akan pernah dapat cepat berkembang seperti apa yang diinginkan.

Sistem informasi dapat berupa gabungan dari beberapa elemen teknologi berbasis komputer yang saling berinteraksi dan bekerja sama berdasarkan suatu prosedur kerja yang telah ditetapkan, dimana memproses dan mengolah data menjadi suatu bentuk informasi yang dapat digunakan dalam mendukung keputusan [12].

2.6 Firebase

Firebase adalah *BaaS (Backend as a Service)* yang saat ini dimiliki oleh Google. *Firebase* ini merupakan solusi yang ditawarkan oleh Google untuk mempermudah pekerjaan *Mobile Apps Developer*. Dengan adanya *Firebase*, *apps developer* bisa fokus mengembangkan aplikasi tanpa harus memberikan *effort* yang besar untuk urusan *backend*. Beberapa fitur yang dimiliki oleh *Firebase* adalah sebagai berikut:

1. *Firebase Analytics*.
2. *Firebase Cloud Messaging* dan *Notifications.*
3. *Firebase Authentication.*
4. *Firebase Remote Config.*
5. *Firebase Real Time Database*.
6. *Firebase Crash Reporting.*

Dua fitur yang menarik adalah *Firebase Remote Config* dan *Firebase Real Time Database.* Secara sederhananya, *Remote Config* adalah fitur yang memungkinkan *developer* mengganti / mengubah beberapa konfigurasi aplikasi *Android / iOS* tanpa harus memberikan *update* aplikasi *via Play Store / App Store*. Salah satu konfigurasi yang bisa dimanipulasi adalah seperti warna / tema aplikasi [13].

2.7 Google *Maps*

Google *Maps* adalah layanan aplikasi peta *online* yang disediakan oleh Google secara gratis. Layanan peta Google *Maps* secara resmi dapat diakses melalui situs *http://maps.google.com*. Pada situs tersebut dapat dilihat informasi *geografis* pada hampir semua permukaan di bumi kecuali daerah kutub utara dan selatan. Layanan ini dibuat sangat *interaktif*, karena di dalamnya peta dapat digeser sesuai keinginan pengguna, mengubah level *zoom*, serta mengubah tampilan jenis peta. Google *Maps* mempunyai banyak fasilitas yang dapat dipergunakan misalnya pencarian lokasi dengan memasukkan kata kunci, kata kunci yang dimaksud seperti nama tempat, kota, atau jalan, fasilitas lainnya yaitu perhitungan *rute* perjalanan dari satu tempat ke tempat lainnya. [14]

2.1.4 Cara Kerja Google *Maps*

Google *Maps* dibuat dengan menggunakan kombinasi dari gambar peta, database, serta *obyek-obyek interaktif* yang dibuat dengan bahasa pemrograman *HTML*, *Javascript* dan *AJAX*, serta beberapa bahasa pemrograman lainnya. Gambar-gambar yang muncul pada peta merupakan hasil komunikasi dengan database pada *web server* Google untuk menampilkan gabungan dari potonganpotongan gambar yang diminta. Keseluruhan citra yang ada diintegrasikan ke dalam *database* pada Google *Server*, yang nantinya akan dapat dipanggil sesuai kebutuhan permintaan. Bagian- bagian gambar map merupakan gabungan dari potongan gambar-gambar bertipe PNG yang disebut tile yang berukuran 256 x 256 *pixel.*

2.8 Google *Maps* API

API atau *Application Programming Interface* merupakan suatu dokumentasi yang terdiri dari *interface*, fungsi, kelas, struktur dan sebagainya untuk membangun sebuah perangkat lunak. Dengan adanya API ini, maka memudahkan *programmer* untuk "membongkar" suatu *software* untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. API dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya yang memungkinkan *programmer* menggunakan sistem *function*. Proses ini dikelola melalui operating *system*. Keunggulan dari API ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya dapat saling berhubungan dan berinteraksi. Bahasa pemrograman yang digunakan oleh Google *Maps* yang terdiri dari *HTML*, *Javascript* dan *AJAX* serta *XML*, memungkinkan untuk menampilkan peta Google *Maps* di *website* lain. Google juga menyediakan layanan Google *Maps* API yang memungkinkan para pengembang untuk mengintegrasikan Google *Maps* ke dalam *website* masingmasing dengan menambahkan data *point* sendiri. Dengan menggunakan Google *Maps* API, Google *Maps* dapat ditampilkan pada *web* site eksternal. Agar aplikasi Google *Maps* dapat muncul di *website* tertentu, diperlukan adanya API key.

API *key* merupakan kode unik yang digenerasikan oleh google untuk suatu *website* tertentu, agar server Google *Maps* dapat mengenali. [14]