**BAB II**

**LANDASAN TEORI**

* 1. **Tinjauan Jurnal**

Setelah peneliti melakukan telaah terhadap beberapa penelitian, ada beberapa yang memiliki keterkaitan dengan pembangunan aplikasi absensi menggunakan *iBeacon* sebagai berikut :

Penelitian Xiangjie Li (2016) yang berjudul “*Design and Implementation of Indoor Positioning System Based on iBeacon*” menjelaskan bahwa meningkatnya permintaan penentuan informasi lokasi terutama di dalam ruangan dengan menggunakan perangkat *mobile*. Tujuan dari penelitian ini adalah untuk merancang dan menerapkan sistem informasi lokasi dalam ruangan berbasis *mobile* dengan teknologi Bluetooth *Low Energy* berdasarkan iBeacon. Penelitian ini merancang dan menerapkan sistem penentuan posisi dalam ruangan dengan mengadopsi metode filter Gaussian dan metode filter Kalman yang tidak beralasan untuk mengekstrak sinyal dengan kuat dari perangkat iBeacon. Hasil penelitian ini sistem dapat menunjukkan posisi dengan menggunakan *mobile* terminal dan sistem ini bias berjalan baik di sistem android maupun iOS. Metode ini memiliki kinerja yang lebih baik dibandingkan dengan metode WiFi di karenakan hasil percobaan menunjukkan bahwa kesalahan hanya 4 meter dan sistem ini dapat mencapai posisi yang akurat dan kuat.

Penelitian Maria Varsamou and Theodore Antonakopoulos (2014) yang berjudul “A Bluetooth *Smart Analyzer* in iBeacon *Networks*” mengatakan bahwa baru-baru ini telah diusulkan agar posisi di dalam ruangan dapat dicapai perangkat sensor BLE yang di sebut iBeacons di berbagai lokasi di tempat tertentu. Tujuan dari penelitian ini adalah menyajikan aplikasi berbasis android untuk menganalisis jaringan iBeacon dan menentukan lokasi sinyal terbaik. Penelitian ini juga menunjukkan analisa yang di sajikan untuk mengukur pola radiasi perangkat iBeacon. Metode penelitian ini dengan menggunakan perangkat lunak berbasis Android untuk mengumpulkan statistik mengenai variasi waktu temporal dan variasi spasial RSSI yang diamati di jaringan iBeacon. Penelitian ini juga menunjukkan penerapan analisis ini untuk mengukur pola radiasi perangkat iBeacon. Hasil percobaan menunjukkan bahwa metode pengukuran yang diusulkan memberikan alat yang andal dan berguna untuk menganalisis radiasi perangkat iBeacon.

Penelitian Yang Yang, Zhouchi Li dan Kaveh Pahlavan (2016) yang berjudul “*Using* iBeacon *for Intelligent In-Room Presence Detection*” mengatakan bahwa mengetahui jumlah pengunjung yang ada di dalam ruangan bisa dilakukan dengan teknologi iBeacon. Tujuan dari penelitian ini adalah menutup pintu masuk secara otomatis setelah seseorang masuk atau keluar dari ruangan dan mengirimkan data ke server. Metode yang di gunakan dalam penelitian ini adalah dengan menggunakan *single iBeacon approach* yang hanya menggunakan satu buah iBeacon. Hasil dari penelitian ini adalah menyelidiki dan mengembangkan sistem deteksi keberadaan di dalam ruangan berbasis iBeacon untuk mencatat pengguna di sebuah ruangan. Kinerja optimal dari penelitian yang di lakukan ini bisa mencapai 100%.

Penelitian Jingjing Yang (2015) yang berjudul “*An* iBeacon *based Indoor Positioning Systems for Hospitals*” dalam penelitiannya menjelaskan bahwa membantu pasien menemukan departemen atau tempat mereka di rumah sakit adalah masalah yang perlu dipecahkan oleh rumah sakit dengan segera. Tujuan dari penelitian ini adalah menentukan posisi di dalam ruangan berbasis iBeacon untuk rumah sakit. Penelitian ini pertama-tama adalah menganalisis keuntungan dari iBeacon dibandingkan dengan teknologi penentuan posisi dalam ruangan biasa. Metode yang digunakan dalam penelitian ini adalah mendeteksi keberadaan pasien dan memberikan lokasi dimana pasien itu berada. untuk merekomendasikan departemen atau unit terdekat kepada pasien. Hasil dari penelitian ini adalah, sistem mampu memberitahukan keberadaan departemen di dalam ruangan rumah sakit dan juga keberadaan kamar yang di rekomendasikan kepada pasien.

**Tabel 2.1** Perbedaan Tinjauan Jurnal dengan Penelitian yang diusulkan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nama peneliti | Judul | Metode | Hasil penelitian | Perbedaan |
| Xiangjie Li (2016) | Design and Implementation of Indoor Positioning System Based on iBeacon | Penelitian ini merancang dan menerapkan sistem penentuan posisi dalam ruangan dengan mengadopsi metode filter Gaussian dan metode filter Kalman yang tidak beralasan untuk mengekstrak sinyal dengan kuat dari perangkat iBeacon. | Sistem dapat menunjukkan posisi dengan menggunakan mobile terminal dan sistem ini bias berjalan baik di sistem android maupun iOS. | Metode yang akan digunakan oleh penulis adalah *Double iBeacons Approach* sedangkan metode yang digunakan penulis sebelumnya adalah metode filter Gaussian dan metode filter Kalman. |
| Maria Varsamou and Theodore Antonakopoulos (2014) | A *Bluetooth Smart* Analyzer in iBeacon Networks | Metode penelitian ini dengan menggunakan perangkat lunak berbasis Android untuk mengumpulkan statistik mengenai variasi waktu temporal dan variasi spasial RSSI yang diamati di jaringan iBeacon. | Hasil percobaan menunjukkan bahwa metode pengukuran yang diusulkan memberikan alat yang andal dan berguna untuk menganalisis radiasi perangkat iBeacon. | Metode yang akan digunakan oleh penulis adalah dengan perangkat lunak berbasis iOS dan menyimpan data di *local storage* perangkat iOS. |
| Yang Yang, Zhouchi Li dan Kaveh Pahlavan (2016) | Using iBeacon for Intelligent In-Room Presence Detection | Metode yang di gunakan dalam penelitian ini adalah dengan menggunakan *single iBeacon approach* yang hanya menggunakan satu buah iBeacon. | Hasil dari penelitian ini adalah menyelidiki dan mengembangkan sistem deteksi keberadaan di dalam ruangan berbasis *iBeacon* untuk mencatat pengguna di sebuah ruangan | Metode yang digunakan oleh penulis adalah *double iBeacon approach*. Sementara metode pada penulis sebelumnya dengan menggunakan *single iBeacon approach*. |
| Jingjing Yang (2015) | An iBeacon-based Indoor Positioning Systems for Hospitals | Metode yang digunakan dalam penelitian ini adalah mendeteksi keberadaan pasien dan memberikan lokasi dimana pasien itu berada. | Hasil dari penelitian ini adalah, sistem mampu memberitahukan keberadaan departemen di dalam ruangan rumah sakit dan juga keberadaan kamar yang di rekomendasikan kepada pasien. | Metode yang digunakan oleh penulis adalah mendeteksi kedatangan dan kepergian pengguna sementara penelitian sebelumnya adalah mendeteksi dan memberikan lokasi kepada pengguna. |

* 1. **Konsep Dasar Program**

Dalam penulisan skripsi ini, penulis menggunakan konsep pemrograman terstruktur dikarenakan mudah untuk dimengerti dan memiliki algoritma pemecahan masalah yanag sederhana,  standar dan efektif menurut (Rosa dan Shalahudin, 2011).

* + 1. **Pemrograman Terstruktur**

Pemrograman terstruktur adalah konsep atau paradigm atau sudut pandang pemrograman yang membagi-bagi program berdasarkan fungsi-fungsi atau prosedur-prosedur yang dibutuhkan program komputer. Modul-modul (pembagian program) biasanya dibuat dengan mengelompokkan fungsi-fungsi dan prosedur-prosedur yang diperlukan sebuah proses tertentu.Fungsi-fungsi dan prosedur-prosedur ditulis secara sekunsial atau terurut dari atas ke bawah sesuai dengan kebergantungan antarfungsi atau prosedur (fungsi atau prosedur yang dapat dipakai oleh fungsi atau prosedur dibawahnya harus yang sudah ditulis atau dideklarasikan di atasnya). Pemodelan pada pemrograman terstruktur dibagi berdasarkan fungsi-fungsi dan prosedur-prosedur. Oleh karena itu, pemodelan pada pemrograman terstruktur lebih fokus kepada cara bagaimana memodelkan data dan fungsi-fungsi atau prosedur-prosedur yang harus dibuat. Jenis paradigm pemrograman yang dapat digunakan untuk membuat program, baru setelah itu ditentukan paradigm pemrograman apa yang akan digunakan (Rosa dan Shalahudin, 2011).

* + 1. **Konsep Dasar Terstruktur**

Pemrograman terstruktur adalah proses yang berorientasi kepada teknik yang digunakan untuk merancang dan menulis program secara jelas dan konsisten. Pemodelan Data merupakan suatu teknik yang berorientasi kepada data dengan menunjukkan sistem hanya datanya saja terlepas dari bagaimana data tersebut akan diproses atau digunakan untuk menghasilkan informasi. Rekayasa Informasi merupakan perpaduan dari pemodelan data dan proses, juga memberikan penekanan baru terhadap pentingnya perencanaan sistem informasi (Rosa dan Shalahudin, 2011).

* + 1. **Aplikasi *Mobile***

Menurut (Lim, 2015) Selama lima tahun terakhir, perangkat mobile, seperti *smartphone*, dan tablet telah jauh lebih populer dari pada perangkat *desk-based* tradisional seperti komputer pribadi dan laptop. Sejak iOS telah dirilis pada iPhone pada 2007, dan Android telah dirilis pada berbagai *smartphone* dan tablet pada tahun 2008, Sistem Operasi paling populer yang berjalan pada perangkat komputasi telah menjadi sistem operasi *mobile*. Dengan demikian, semakin banyak aplikasi yang berjalan pada Sistem Operasi mobile daripada Sistem Operasi desktop.

* + 1. **Pengertian Aplikasi**

Menurut Nazrudin Safaat H (2012) aplikasi adalah satu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas. Misalnya termasuk perangkat lunak perusahaan,software akuntansi, perkantoran, grafis perangkat lunak dan pemutar media. Dapat disimpulkan bahwa aplikasi merupakan software yang berfungsi untuk melakukan berbagai bentuk pekerjaan atau tugas-tugas tertentu seperti penerapan, penggunaan dan penambahan data.Program aplikasi merupakan program siap pakai. Program yang direka untukmelaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh-contoh aplikasi ialah program pemproses kata dan Web Browser. Aplikasi akan menggunakan sistem operasi (OS) komputer dan aplikasi yang lainnya yang mendukung.

Klasifikasi aplikasi dapat dibagi menjadi 2 (dua) yaitu:

1. Aplikasi software spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
2. Aplikasi paket, dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu.
   * 1. **Pengertian Aplikasi *Mobile***

Menurut Nazrudin Safaat H (2012) aplikasi mobile berasal dari kata *application* dan *mobile*. *Application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju sedangkan *mobile* dapat di artikan sebagai perpindahan dari suatu tempat ke tempat yang lain.

Maka aplikasi *mobile* dapat di artikan sebuah program aplikasi yang dapat

dijalankan atau digunakan walaupun pengguna berpindah–pindah dari satu tempat ke tempat yang lain serta mempunyai ukuran yang kecil. Aplikasi mobile ini dapat di akses melalui perangkat nirkabel, pager, PDA, telepon seluler, smartphone, dan perangkat sejenisnya.

* + 1. **Karakteristik Perangkat *Mobile***

Menurut Nazrudin Safaat H (2012) Perangkat *mobile* memiliki banyak jenis dalam hal ukuran, desain dan layout, tetapi mereka memiliki karakteristik yang sangat berbeda dari sistem desktop. Berikut karakteristik perangkat *mobile*, diantaranya :

1. Ukuran yang kecil

Perangkat *mobile* memiliki ukuran yang kecil. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.

1. Memory yang terbatas

Perangkat *mobile* juga memiliki memory yang kecil, yaitu primary (RAM) dan secondary (disk). Pembatasan ini adalah salah satu faktor yang mempengaruhi penulisan program untuk berbagai jenis dari perangkat ini. Dengan pembatasan jumlah dari memory, pertimbangan-pertimbangan khusus harus diambil untuk memelihara pemakaian dari sumber daya yang ma hal ini.

1. Daya proses yang terbatas

Sistem *mobile* tidaklah setangguh rekan mereka yaitu desktop. Ukuran, teknologi dan biaya adalah beberapa faktor yang mempengaruhi status dari sumber daya ini. Seperti harddisk dan RAM, Anda dapat menemukan mereka dalam ukuran yang pas dengan sebuah kemasan kecil.

1. Mengkonsumsi daya yang rendah

Perangkat *mobile* menghabiskan sedikit daya dibandingkan dengan mesin

desktop. Perangkat ini harus menghemat daya karena mereka berjalan pada keadaan dimana daya yang disediakan dibatasi oleh baterai-baterai.

1. Kuat dan dapat diandalkan

Karena perangkat *mobile* selalu dibawa kemana saja, mereka harus cukup kuat untuk menghadapi benturan-benturan, gerakan, dan sesekali tetesan-tetesan air.

1. Konektivitas yang terbatas

Perangkat *mobile* memiliki bandwith rendah, beberapa dari mereka bahkan

tidak tersambung. Kebanyakan dari mereka menggunakan koneksi *wireless*.

1. Masa hidup yang pendek

Perangkat-perangkat konsumen ini menyala dalam hitungan detik kebanyakan dari mereka selalu menyala. Coba ambil kasus sebuah handphone, mereka booting dalam hitungan detik dan kebanyakan orang tidak mematikan handphone mereka bahkan ketika malam hari. PDA akan menyala jika anda menekan tombol power mereka.

* + 1. **Bluetooth**

Menurut (www.bluetooth.com/Pages/Basic.aspx, 31-07-2017) Bluetooth merupakan teknolog komunikasi nirkabel jarak pendek yang menggunakan gelombang radio dalam melakukan pertukaran data. Bluetooth dikenal dengan karakteristiknya yang simple, hemat energi dan tingkat keamanannya yang tinggi. Sampai dengan hari ini, walaupun sudah termasuk teknologi kuno, Bluetooth masih ada hampir disemua alat komunikasi modern. Ketika pertama kali diciptakan oleh Ericsson pada tahun 1994 bluetooth hanyalah sebagai alternatif wireless dari kabel RS 232. Kini Bluetooth dapat kita gunakan untuk bertukar data yang lebih kompleks seperti musik, video dan gambar. Untuk dapat saling berkomunikasi pertama-tama 2 device yang ingin berkomunikasi harus menyalakan Bluetooth lalu melakukan *pairing* satu sama lain. Ketika sudah terhubung (melakukan *pairing*) maka sudah dapat terjadi komunikasi/pertukaran data, 2 *device* yang saling terhubung ini juga akan menciptakan suatu jaringan komputer yang disebut dengan istilah PAN(*Personal Area Network*). Untuk jarak yang dapat dijangkau oleh bluetooth sendiri cukup bervariasi, mulai dari 1 meter sampai dengan 100 meter, tergantung dari jenis gelombang radio yang digunakan.

* + 1. **Bluetooth *Low Energy***

Bluetooth *Low Energy* sering disebut dengan Bluetooth *Smart*, bluetooth yang membutuhkan daya rendah dalam kerjanya (*A VeriFone White Paper*, 2013). Dikatakan dalam (http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons, 31-07-2017) bahwa hanya dengan sumber daya dari baterai berbentuk koin (seperti baterai bios komputer) sebuah modul BLE dapat bertahan hingga 3 tahun. BLE adalah sebuah *Personal Area Network* (PAN) yang memancarkan sinyal data dengan jarak tertentu. Syarat yang harus dimiliki oleh sebuah perangkat sehingga dia bisa bekerja dalam kategori BLE adalah bahwa perangkat tersebut harus sudah tertanamkan bluetooth versi 4.0. Beberapa perangkat dengan bluetooth versi ini seperti iPhone 4s atau seri di atasnya. Tidak hanya versi bluetooth saja yang disyaratkan, untuk iPhone atau iPad juga harus telah terinstal versi iOS 7. Sedangkan untuk *smartphone* Android versi sistem operasi yang terinstal yaitu minimal Jelly Bean versi 4.3.

* + 1. **iBeacon**

iBeacon adalah implementasi dari teknologi BLE yang memungkinkan perangkat iOS atau Android untuk bisa mendeteksi sinyal yang dipancarkan oleh iBeacon tersebut dan memberikan informasi dimanakah posisi perangkat tersebut terhadap iBeacon (http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons, 31-07-2017). Tetapi posisi yang dimaksud bukanlah posisi secara fisik seperti lintang dan bujur, melainkan posisi relatif *receiver* terhadap iBeacon apakah dia dalam range *immediate*, *near* atau *far*. Untuk bisa mengetahui estimasi posisi *receiver* terhadap iBeacon secara pasti maka perlu adanya peta secara lokal, yaitu menentukan sebuah lokasi atau ruang atau tempat tertentu, kemudian dari lokasi tersebut dipetakan dalam bentuk koordinat. Dengan meletakkan beberapa iBeacon pada koordinat tertentu dalam lokasi tersebut akan dapat digunakan untuk mengetahui estimasi posisi *receiver* terhadap iBeacon. *Positioning* dengan cara seperti ini bisa dikatakan berbasis *context aware*.

* + 1. **Sistem Operasi iOS pada *Smartphone***

Adelphia (2015) mendefinisikan iOS adalah sistem operasi yang dikembangkan oleh perusahaan Apple untuk ponsel iPhone, tetapi kemudian berkembang dan dapat digunakan ke dalam perangkat Apple yang lainnya seperti iPod Touch, Apple TV dan iPad. Menurut Adelphia (2015) sistem operasi ini bersifat tertutup dan hanya bisa digunakan oleh perangkat Apple, jadi anda tidak akan menemukan sistem operasi iOS

pada perangkat serupa dengan merek lain. Didalam iOS juga terdapat komponen *abstraction layers*, yaitu lapisan sistem iOS yang terbagi menjadi empat bagian, seperti *framework* yang berfungsi untuk membangunkan *user* ke hardware (Adelphia, 2015). Menurut Adelphia (2015) dalam iOS juga terdapat komponen Darwin yang masih satu

keluarga dengan UNIX, beberapa lapisan dalam iOS adalah:

1. Lapisan yang berhubungan langsung ke hardware atau disebut juga *Core OS Layer*.
2. Bagian yang berisi layanan yang membentuk sistem dasar semua aplikasi, yaitu *Core Service Layer*.
3. Sementara itu, bagian untuk grafis disebut Media Layer. Lapisan ini terbentuk untuk mengarahkan audio, video, dan teknologi grafis lainnya menjadikan iOS kaya akan multimedia.
4. Yang terakhir adalah *Cocoa Touch layer*, bagian ini berfungsi untuk interaksi antara user dan optimasi *focus*.

Keempat bagian tersebut bersatu dalam iOS. Sistem operasi ini juga dikonsep untuk dapat bekerja dengan baik dilayar sentuh (Adelphia, 2015). Menurut Adelphia (2015) ada beberapa fitur menarik yang bisa anda temukan di sistem operasi iOS dalam iPhone, diantaranya:

1. *User friendly.*
2. Kemampuan untuk bekerja secara *multi tasking.*
3. Desain yang elegan.
4. Banyak pengembang yang memberikan aplikasi untuk diunduh melalui App Store.
5. Ukuran memory yang cukup besar. Sampai saat buku ini ditulis masih tersedia ukuran 8-128 GB. Sulit terserang oleh virus.
6. *Upgrade* sistem operasi dapat dengan mudah dilakukan melalui *smartphone* ataupun PC.

Namun menurut Adelphia (2015) banyak hal yang mungkin tidak disukai oleh beberapa individu ketika hendak memakai sistem operasi ini, diantaranya adalah:

1. Harga yang relatif mahal. Belum diketahui sebabnya kenapa merek Apple selalu mempunyai harga yang diatas rata-rata.
2. Belum tersedia pemasangan memori tambahan (eksternal).
3. Tidak dapat melakukan modifikasi terhadap sistem operasi.
4. Output suara masih terlalu kecil.

Berikut ini daftar versi iPhone beserta iOS yang digunakan dari awal kemunculannya hingga sekarang (Adelphia, 2015)

**Tabel 2.2** Perkembangan iPhone

|  |  |  |  |
| --- | --- | --- | --- |
| Versi | iOS versi | Prosesor | Kamera |
| iPhone | iOS 3.1.3 | 620 Mhz | 2 MP |
| iPhone 3G | iOS 4.2.1 | 412 Mhz arm 11 | 2 MP |
| iPhone 3Gs | iOS 6.0 | 600 Mhz Cortex A8 | 3 MP |
| iPhone 4 | iOS 6.0 | 1 Ghz Cortex A8 | 5 MP |
| iPhone 4s | iOS 6.0 | Dual Core 1 Ghz A9 | 8 MP |
| iPhone 5 | iOS 6.0 | Dual Core 1,2 Ghz | 8 MP |
| iPhone 5s | iOS 7.0 | Dual Core 1,3 Ghz | 8 MP |
| iPhone 6 | iOS 8.0 | Dual Core 1,4 Ghz | 8 MP |

* + 1. ***Database***

Menurut Connolly dan Begg (2010:65), *database* adalah sekumpulan data tersebar yang berhubungan secara logis, dan penjelasan dari data ini dirancang untuk memenuhi kebutuhan informasi dari suatu organisasi.

Menurut Inmon (2011:493), *database* adalah sekumpulan data yang saling berhubungan yang disimpan (biasanya dengan redudansi yang terkontrol dan terbatas) berdasarkan skema. Sebuah *database* dapat melayani *single* atau *multiple applications*.

Menurut Gottschalk dan Saether dalam jurnal (2010:41), *database* adalah sekumpulan data yang terorganisir untuk mendukung banyak aplikasi secara efisien dengan memusatkan data dan mengontrol *data redundant.*

Berdasarkan definisi-definisi yang dijabarkan oleh para ahli di atas, maka dapat disimpulkan *database* adalah sekumpulan data yang saling berhubungan dan terorganisir yang disimpan berdasarkan skema dengan memusatkan data dan mengontrol *data redundant* untuk memenuhi kebutuhan informasi dari suatu organisasi.

* + 1. ***Database Management System* (DBMS)**

Menurut (Hoffer, Prescott, & Topi, 2011: 11), *Database Management System* (DBMS) merupakan sebuah sistem perangkat lunak yang dapat membuat membuat, menciptakan dan memelihara akses ke pengguna *database*. Menurut Kroenke and Auer (2012:11) DBMS adalah suatu program komputer yang digunakan untuk membuat, memproses, dan mengelola database.

Menurut Connolly & Begg (2010:66) *Database Management System* (DMS) adalah sebuah *software system* yang memungkinkan user mendefinisi, membentuk dan mengatur *database* dan megendalikan akses ke *database* DBMS berinteraksi dengan pengguna aplikasi program dan *database*.

DBMS menyediakan fasilitas :

1. Data *Definition Language* (DDL), yang berguna untuk menspesifikasikan tipe data, struktur dan *constraint* data.
2. Data *Manipulation Language* (DML), yang berguna untuk memberikan fasilitas *query* data.
3. Pengendalian akses *database*, antara lain mengontrol :
4. Keamanan sistem : Mencegah user yang tidak memiliki hak untuk mengakses *database*.
5. Integrasi Sistem : Menjaga konsistensi data. Jadi, dari beberapa pengertian diatas dapat disimpulkan bahwa DBMS merupakan sebuah sistem perangkat lunak yang memungkinkan pengguna untuk dapat membuat, menciptakan, memelihara dan mengontrol akses dari *database* ke pengguna *database*.
   * 1. ***MySQL***

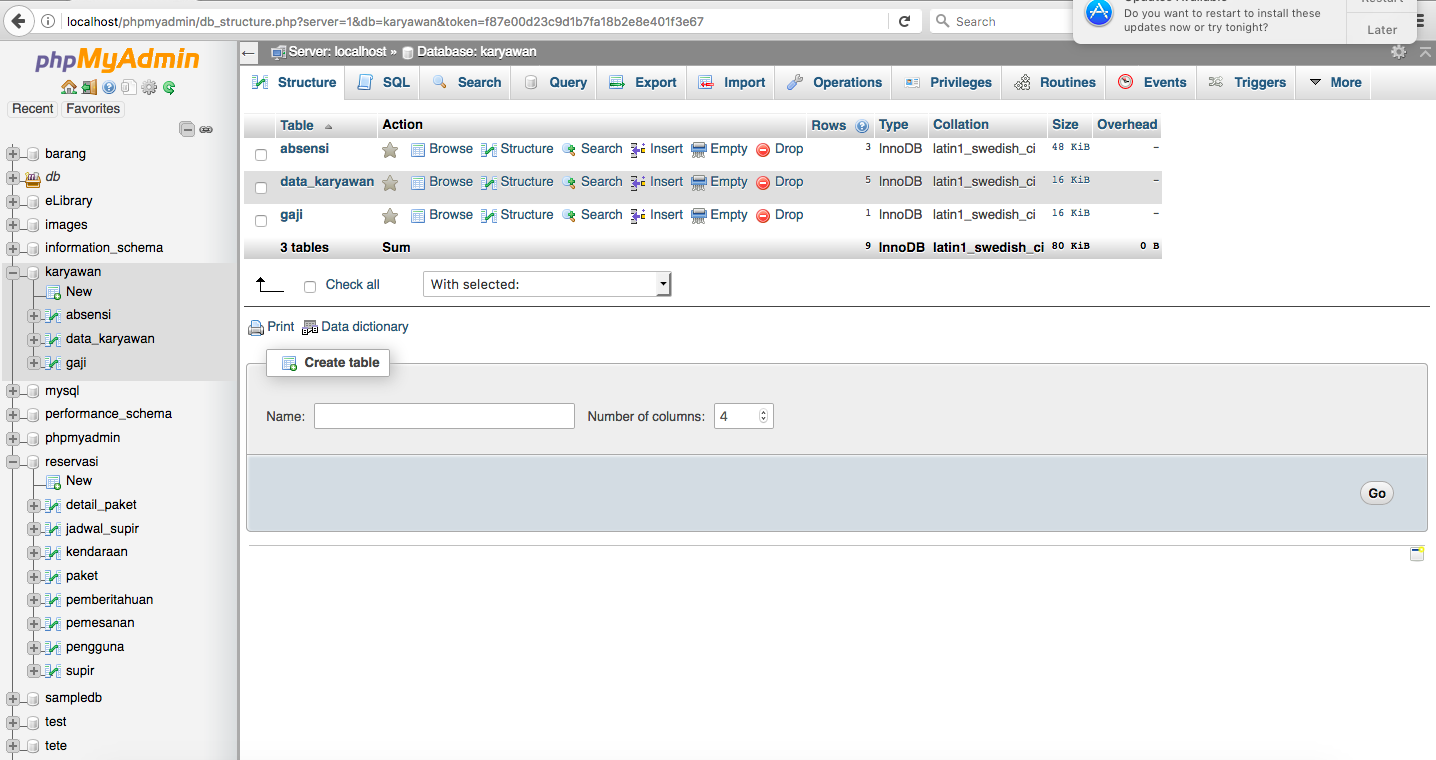
Menurut Sidik (2012, p.333), *MySql* merupakan *software database* yang termasuk paling popouler di lingkungan *Linux*, kepopuleran ini karena ditunjang karena performansi *query* dari *databasenya* yang saat itu bisa dikatakan paling cepat, dan jarang bermasalah. *MySql* telah tersedia juga di lingkungan *Windows*. *PHP* secara *default* telah mendukung *MySql* karena *MySql* telah dimiliki oleh Oracle, dimana mengembangkan database yang murni *open source* dan *freeware* dengan nama *MariaDB*.

*MySql* diciptakan di negara Swedia oleh perusahaan *MySql AB*. Adapun masing-masing nama yang berjasa dalam menciptakan *MySql* adalah David Axmark, Alan Larsson, dan Michael “Monty” Widenius. Perangkat lunak ini tersebar luas secara gratis karena memiliki lisensi *GNU General Public License*. Sampai sekarang, tercatat ada beberapa bersinkronisasi dengan MySql, seperti C, C++, C#, bahasa pemrograman yang cukup populer *Eiffel*, bahasa pemrograman *Smalltalk*, bahasa pemrograman *Java*, bahasa pemrograman *Lisp, Perl, PHP*, bahasa pemrograman *Python, Ruby, REALbasic*, dan *Tcl*.

Menurut Saputra, Subagio, dan Saluky (2012, p.7), *MySQL* bekerja menggunakan *SQL Language* (*Structure Query Language*), yang diartikan bahwa *MySQL* merupakan standard penggunaan database di dunia untuk pengolahan data. Pada umumnya, perintah yang paling sering digunakan dalam *MySQL* adalah *SELECT, INSERT, UPDATE,* dan *DELETE*. *SQL* juag menyediakan perintah untuk membuat *database*, *field*, ataupun *index* untuk menambah atau menghapus data. *MySQL* juga merupakan database yang mampu berjalan di semua sistem operasi, *powerful*, dan sangat mudah untuk dipelajari, dan hosting servernya juga banyak mengadopsi *MySQL* sebagai *standard database*. Ada juga beberapa kelebihan *MySQL* yaitu :

1. Bersifat open source, yang memiliki kemampuan untuk dapat dikembangkan lagi.
2. Menggunakan bahasa SQL, yang merupakan standar bahasa dunia dalam pengolahan data.
3. Super performance dan reliable, pemrosesan database-nya sangat cepat dan stabil.
4. Sangat mudah dipelajari
5. Memiliki dukungan support pengguna MySQL.
6. Mampu lintas platform, dapat berjalan di berbagai sistem operasi.
7. Multiuser, di mana SQLdapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami konflik.
   * 1. ***PHP My Admin***

Bunafit (2013:15), *PHPMyAdmin* adalah aplikasi manajemen *database* *server MySQL* berbasis web. Dengan aplikasi phpMyAdmin kita bias mengelola *database* sebagai *root* atau juga sebagai *user* biasa, kita bisa membuat *database* baru, megelola *database* dan melakukan operasi perintah-perintah *database* secara lengkap seperti saat kita di *MySQL* *Promp*. Berikut tampilan dari *PHP MyAdmin* :



sumber : *PHP MyAdmin*

**Gambar 2.1** Tampilan *PHP MyAdmin*

* + 1. ***PHP***

*PHP* merupakan singkatan dari *PHP Hypertext Preprocessor*, ia merupakan bahasa berbentuk skrip yang ditempatkan dalam *server* dan diproses di server. Secara khusus *PHP* dirancang untuk membentuk aplikasi *web* dinamis artinya ia dapat membentuk suatu tampilan berdasarkan permintaan terkini.

Kelahiran *PHP* bermula ketika Rasmus Lerdorf membuat sejumlah skrip perl yang dapat mengamati siapa saja yang melihat-lihat daftar riwayat hidupnya., yakni pada tahun 1994. Skrip ini selanjutnya dikemas menjadi *tool* yang disebut dengan “*Personal Home Page*”. Pada tahun 1995, Rasmus menciptakan *PHP/FI* Versi 2. Pada versi inilah pemrograman dapat menempelkan kode terstruktur di dalam tag *HTML*. *PHP* juga dapat berkomunikasi dengan *database* dan melakukan banyak perhitungan yang kompleks. Pada saat ini *PHP* cukup poluler sebagai piranti pemrograman *Web*, terutama dilingkungan *Linux*, Walaupun demikian, *PHP* sebenarnya juga dapat berfungsi pada *server-server* yang berbasis *UNIX*, *Windows*, dan *Macintosh*. Pada awalnya *PHP* dirancang untuk diintegrasikan dengan *web server Apache*. Namun belakangan ini *PHP* juga dapat bekerja dengan *Web Server* seperti PWS (*Personal Web Server*), IIS (*Internet Information Server*) dan Xitami (Mehdi, et all. 2007).



Sumber : Sublime

**Gambar 2.2** Contoh Kode PHP

* + 1. **CodeIgniter**

Menurut Hakim (2010:8) CodeIgniter adalah sebuah *framework* PHP yang dapat membantu mempercepat developer dalam pengembangan aplikasi web berbasis PHP dibanding jika menulis semua kode program dari awal.

CodeIgniter pertama kali dibuat oleh Rick Ellis, CEO Ellislab, Inc. (http://ellislab.com), sebuah perusahaan yang memproduksi CMS (*Content Management System*) yang cukup handal, yaitu *Expression Engine* (http://www.expressionengine.com). Saat ini, CodeIgniter dikembangkan dan dimaintain oleh *Expression Engine Development Team*. Adapun beberapa keuntungan menggunakan CodeIgniter, diantaranya:

1. Gratis

CodeIgniter berlisensi dibawah Apache/BSD *open source*.

1. Ditulis Menggunakan PHP 4

Meskipun CodeIgniter dapat berjalan di PHP 5, namun sampai saat ini kode program CodeIgniter masih dibuat dengan menggunakan PHP 4.

1. Berukuran Kecil Ukuran

CodeIgniter yang kecil merupakan keunggulan tersendiri. Dibanding dengan *framework* lain yang berukuran besar.

1. Menggunakan Konsep MVC

CodeIgniter menggunakan konsep MVC yang memungkinkan pemisahan *layer* *application-logic* dan *presentation*.

1. URL yang Sederhana

Secara default, URL yang dihasilkan CodeIgniter sangat bersih dan *Serach Engine Friendly* (SEF).

1. Memiliki Paket *Library* yang Lengkap

CodeIgniter mempunyai *library* yang lengkap untuk mengerjakan operasi-operasi yang umum dibutuhkan oleh sebuah aplikasi berbasis web, misalnya mengakses *database*, mengirim email, memvalidasi form, menangani *session* dan sebagainya.

1. *Extensible*

Sistem dapat dikembangkan dengan mudah menggunakan *plugin* dan *helper*, atau dengan menggunakan *hooks*.

1. Tidak Memerlukan *Template Engine*

Meskipun CodeIgniter dilengkapi dengan *template parser* sederhana yang dapat digunakan, tetapi hal ini tidak mengharuskan kita untuk menggunakannya.

1. Dokumentasi Lengkap dan Jelas

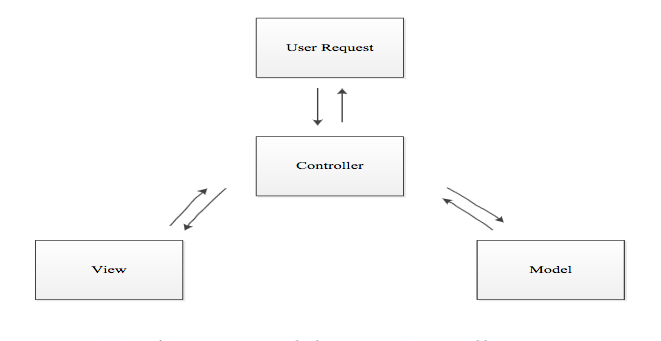
Dari sekian banyak *framework*, CodeIgniter adalah satu-satunya framework dengan dokumentasi yang lengkap dan jelas.

1. Komunitas

Komunitas CodeIgniter saat ini berkembang pesat. Salah satu komunitasnya bisa dilihat di (http://codeigniter.com/forum/).

* + 1. **MVC (*Model-View-Controller*)**

CodeIgniter adalah *framework* PHP yang dibuat berdasarkan kaidah *Model-View-Controller*. Dengan MVC, maka memungkinkan pemisahan antara *layer* *application-logic* dan presentation. Sehingga, dalam sebuah pengembangan web, seorang *programmer* bisa berkonsentrasi pada *core-system*, sedangkan web *designer* bisa berkonsentrasi pada tampilan web. Menariknya, skrip *PHP*, *query* *MySQL*, *Javascript* dan *CSS* bisa saling terpisah, tidak dibuat dalam satu skrip berukuran besar yang membutuhkan *resource* besar pula untuk mengesekusinya. Adapun alur program aplikasi berbasis *framework* Codeigniter dapat dilihat pada gambar 2.3.



**Gambar 2.3** *Model-View-Controller*

Sumber : Hakim (2010: 4) Membangun Web Berbasis PHP dengan *Framework* CodeIgniter

Gambar diatas menerangkan bahwa ketika datang sebuah *user request*, maka akan ditangani oleh *controller*, kemudian *controller* akan memanggil *model* jika memang diperlukan operasi *database*. Hasil dari *query* oleh *model* kemudian akan dikembalikan ke *controller*. Selanjutnya *controller* akan memanggil *view* yang tepat dan mengkombinasikannya dengan hasil *query model*. Hasil akhir dari operasi ini akan ditampilkan dibrowser.

Dalam konteks CodeIgniter dan aplikasi berbasis web, maka penerapan konsep MVC mengakibatkan kode program dapat dibagi menjadi tiga kategori, yaitu :

1. *Model*

Kode program (berupa OOP *class*) yang digunakan untuk memanipulasi *database*.

1. *View*

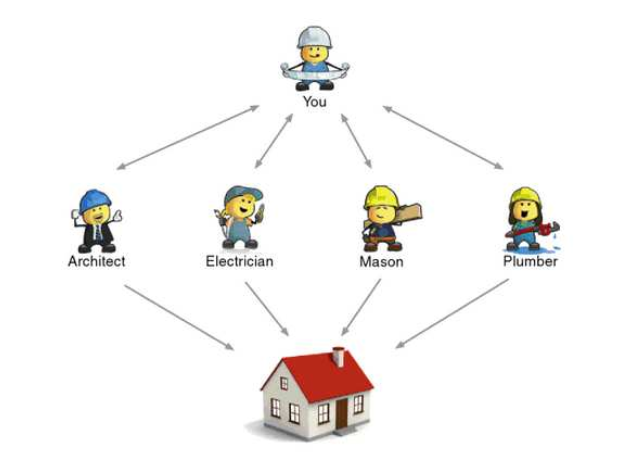
Berupa *template* *html/xml* atau *php* untuk menampilkan data pada *browser*

1. *Controller*

Kode program (berupa OOP *class*) yang digunakan untuk mengontrol aliran aplikasi (sebagai pengontol *model* dan *View*)

* + 1. **API**

API merupakan *software interface* yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk *library* dan menjelaskan bagaimana agar suatu *software* dapat berinteraksi dengan *software* lain. Penjelasan ini dapat dicontohkan dengan analogi apabila akan dibangun suatu rumah. Dengan menyewa kontraktor yang dapat menangani bagian yang berbeda, pemilik rumah dapat memberikan tugas yang perlu dilakukan oleh kontraktor tanpa harus mengetahui bagaimana cara kontraktor menyelesaikan pekerjaan tersebut. Dari analogi tersebut, rumah merupakan *software* yang akan dibuat, dan kontraktor merupakan API yang mengerjakan bagian tertentu dari *software* tersebut tanpa harus diketahui bagaimana prosedur dalam melakukan pekerjaan tersebut (Reddy, 2011).



**Gambar 2.4** Analogi API pada Pembangunan Rumah

(Sumber: API *Design for* C, Reddy, 2011)

*Interface* pada *software* merupakan suatu *entry points* yang digunakan untuk mengakses seluruh *resources* yang terdapat di dalam *software* tersebut. Dengan adanya API, maka terdapat aturan bagaimana *software* dapat berinteraksi dengan *software* lain untuk mengakses *resources* melalui *interface* yang telah tersedia.

Menurut (3Scale Networks, 2011) secara struktural, API merupakan spesifikasi dari suatu *data structure*, *objects*, *functions*, beserta parameter-parameter yang diperlukan untukmengakses *resource* dari aplikasi tersebut. Seluruh spesifikasi tersebut membentuk suatu *interface* yang dimiliki oleh aplikasi untuk berkomunikasi dengan aplikasi lain, dan API dapat digunakan dengan berbagai bahasa *programming*, ataupun hanya dengan menggunakan URL (*Uniform Resource Locator*) yang telah disediakan oleh suatu *website*. API dapat diklasifikasikan menjadi beberapa kategori, hal ini dilihat dari abstraksi apa yang dideskripsikan di dalam sistem. Kategori-kategori ini diantaranya:

**Tabel 2.3** Kategori API

|  |  |  |
| --- | --- | --- |
| **Kategori API** | **Deskripsi** | **Contoh** |
| ***Operating System*** | API yang digunakan untuk fungsi dasar yang dapat dilakukan oleh komputer. Seperti proses I/O, eksekusi program. | API for MS Windows |
| ***Programming Languages*** | API yang digunakan untuk memperluas kapabilitas dalam melakukan eksekusi terhadap suatu bahasa pemrograman. | Java API |
| ***Application Services*** | API yang digunakan untuk mengakses data dan layanan yang disediakan dari suatu aplikasi. | API for mySAP (BAPI/*Business Application Programming Interface*) |
| ***Infrastructure Services*** | Digunakan untuk mengakses infrastruktur dari suatu komputer. Infrastruktur disini adalah komputer beserta *peripheral* seperti *storage*, aplikasi, dan lain-lain. | Amazon EC2 (*Elastic Compute Cloud*) untuk akses untuk *virtual computing* dan Amazon S3 (*Simple Storage Service*) untuk menyimpan data dalam jumlah besar. |
| ***Web Services*** | API yang digunakan untuk mengakses *content* dan layanan yang disediakan oleh suatu *web* *application*. | *Facebook Graph* API yang digunakan untuk mengakses informasi yang dapat dibagikan. |

* + 1. **REST (*Representational State Transfer*)**

REST (*Representational State Transfer*) merupakan jenis arsitektur yang terdapat pada *web* untuk melayani suatu *service*. REST merepresentasikan cara interaksi antara *server* dan *client* untuk melakukan proses pertukaran informasi melalui media yang sama. Dalam suatu jaringan. agar suatu *resource* dapat diakses, maka diperlukan identifikasi dan suatu bentuk manipulasi. Dapat digunakan URI (*Uniform Resource Identifier*) yang digunakan untuk mengidentifikasi *resource* yang ada pada suatu jaringan, dan dapat membuat *resource* menjadi *addressable*, yang berarti *resource* dapat diketahui lokasinya dan dapat dimanipulasi dengan menggunakan suatu aplikasi (Webber, Parastatidis, & Robinson, 2010: 5).

REST dapat digunakan sebagai *interface* dari API untuk mengakses suatu *resource*. API yang mengikuti prinsip dari REST *architecture* memberikan kemudahan bagi *developer* untuk tidak perlu mengetahui bagaimana struktur dari API di dalam *server*. Dalam hal ini, *server* akan memberikan informasi bagaimana agar *client* dapat mengakses *service* melalui API yang telah disediakan.

Penggunaan protokol HTTP pada REST *architecture* untuk komunikasi antara *client* dan *server* terletak pada HTTP *method*, yaitu *GET*, *POST*, *PUT*, dan *DELETE*. *Method* ini dapat digunakan untuk mengakses *resources* yang ada pada *server*, bergantung dari instruksi yang diberikan oleh *server*.

Dengan menggunakan protokol HTTP, URI dapat dijadikan sebagai media yang digunakan untuk mengakses *resource* dari *server*. Hal ini disebut dengan URI *tunneling*.

URI *tunneling* mempergunakan URI untuk mentransfer informasi pada antar sistem yang dalam jaringan dengan melakukan *encode* pada URI itu sendiri. Dengan mengirim HTTP *method* yang telah disebutkan sebelumnya, *server* dapat melakukan eksekusi terhadap suatu program yang menghasilkan atau mengambil suatu *resource* dan mengirimkannya kembali ke *client*. Dalam proses ini terjadi proses *mapping* dari URI menjadi *method call* pada *server* yang dituju (Webber, Parastatidis, & Robinson, 2010: 37).

* + 1. **JSON**

Menurut Deitel (2012, p1303) JSON (*JavaScript Object Notation*) adalah suatu format pertukaran data komputer. Format dari JSON adalah berbasis teks, dapat terbaca oleh manusia, digunakan untuk mempresentasikan struktur data sederhana, dan tidak bergantung dengan bahasa apapun. Biasanya digunakan pada aplikasi Ajax. Format JSON sering digunakan untuk mentransmisikan data terstruktur melalui koneksi jaringan. Secara umum, JSON digunakan untuk mentransmisikan data antara *server* dan aplikasi *web*.

Jenis media internet yang resmi untuk JSON adalah aplikasi/json. Format JSON sering digunakan untuk serialisasi dan mengirimkan data terstruktur melalui koneksi jaringan, terutama untuk pengiriman data antara server dan aplikasi *web* melayani sebagai alternatif ke XML.



**Gambar 2.5** Contoh format JSON

(Sumber: Insomnia)

* + 1. **Macintosh**

Macintosh atau yang biasa disebut Mac adalah salah satu jenis komputer yang diproduksi oleh Apple Computer Inc. Macintosh diperkenalkan pertama kali pada januari 1984. Macintosh adalah komputer pertama yang memperkenalkan sistem antar muka grafis (GUI) (Setiawan, 2008).

* + 1. **Mac OS**

Mac OS adalah singkatan dari Macintosh *Operating System*, Mac OS adalah sistem operasi yang dibuat oleh Apple ComputerInc. yang dikhususkan untuk komputer Macintosh dan tidak kompatibel dengan PC berbasis IBM, Namun mulai tahun 2006 Mac OS sudah bisa digunakan pada komputer dengan arsitektur x86. Sistem operasi ini pertama kali diperkenalkan bersamaan dengan komputer Macintosh (Setiawan, 2008).

* + 1. **iOS SDK dan Xcode**

iOS SDK (*Software Development Kit*) yang bekerja pada komputer Macintosh menyediakan antarmuka, *tools* dan semua sumber yang digunakan untuk membangun aplikasi iOS. (Apple, iOS Technology Overview, 2010). Apple memberikan sebagian besar sistem untuk antarmuka mereka ke dalam paket yang disebut dengan *framework* atau kerangka kerja. *Framework* adalah sebuah direktori yang berisi *library* dan sumber-sumber seperti *header files*, gambar, aplikasi bantuan, dan lain-lain yang digunakan untuk mendukung *library* yang tersedia. *Framework* digunakan dengan cara *me-link framework* yang diinginkan ke dalam proyek aplikasi. Menghubungkan *framework* dengan proyek yang dibuat memberi akses ke semua fitur dari *framework* tersebut dan juga memungkinkan untuk *development* *tools* mengetahui dimana menemukan suatu *file* tertentu (Apple, iOS Technology Overview, 2010, pp. 14-15).

Komponen yang ada pada SDK termasuk:

* Xcode *Tools* – alat yang mendukung pengembangan aplikasi iOS, termasuk:

1. Xcode – sebuah alat yang mendukung pengembangan untuk mengelola proyek aplikasi dan memungkinkan untuk mengedit, mengkompilasi, menjalankan dan memperbaiki *error* pada kode. Xcode terintegrasi dengan alat pendukung lainnya dan merupakan aplikasi utama yang digunakan selama pengembangan.
2. *Interface Builder* – alat yang digunakan untuk merancang antarmuka secara visual. Objek antarmuka yang dibuat disimpan pada suatu *file* dan dimuat ke aplikasi saat *runtime*.
3. *Instruments* – alat yang digunakan untuk menganalisa perfoma dan mendeteksi kesalahan. *Instruments* bisa digunakan untuk mendapatkan informasi mengenai perilaku *runtime* dari aplikasi dan mengidentifikasikan kemungkinan kesalahan.
4. iOS *Simulator* – aplikasi Mac OS X yang mensimulasikan teknologi iOS, mengizinkan pengembang untuk menguji aplikasi iOS secara lokal pada komputer Macintosh yang digunakan.
5. iOS *Developer Library* – referensi dan dokumentasi konsep yang menjelaskan semua tentang teknologi iOS dan proses pengembangan aplikasi.

Walaupun aplikasi bisa dicoba pada iOS *Simulator*, pengembang juga dimungkinkan untuk menguji aplikasinya ke piranti yang sesungguhnya menggunakan Xcode dan *Instruments*. iOS *Simulator* ideal untuk membangun dan menguji aplikasi secara cepat namun tidak bisa menggantikan pengujian pada piranti sesungguhnya. Untuk mencoba aplikasi ke salah satu piranti iOS, pengembang harus terdaftar pada iOS *Developer* Program (Apple, iOS Technology Overview, 2010, p. 15).

* + 1. **Cocoa**

Cocoa dan Cocoa Touch *framework* yang bekerja untuk Mac OS X dan iOS adalah sebuah kerangka kerja yang terintegrasi ke dalam Xcode. Cocoa itu sendiri merupakan salah satu API (*Application Programming Interface*) berbasis objek milik Apple Inc. API Cocoa yang bertingkat tinggi membuat pengembang lebih mudah untuk menambahkan animasi, jaringan, tampilan *platform* asli, dan karakter untuk aplikasi dengan hanya beberapa baris kode. Cocoa *framework* terdiri dari *libraries*, API dan *runtimes*. Dengan menggunakan Cocoa, pengembang dapat membangun aplikasi dengan cara yang sama dengan Mac OS X. Aplikasi otomatis akan mempunyai karakter dan tampilan yang sama dengan Mac OS X. Cocoa dengan menggunakan Xcode adalah cara yang paling baik untuk membuat aplikasi untuk Mac (Apple, Cocoa: Mac OS X Technology Overview, 2011).

* + 1. **Cocoa Touch**

Cocoa Touch, berbeda dengan Cocoa, merupakan ekstensi dari Cocoa untuk aplikasi pada sistem operasi iOS pada iPhone, iPad dan iPod Touch. Cocoa Touch digunakan untuk menambahkan fungsi sensor gerak dan animasi.

Cocoa Touch mengatur dan mengelola interaksi pengguna pada iOS. Karena Cocoa Touch merupakan ekstensi dari Cocoa, maka semua teknologi Cocoa Touch berasal dari Cocoa. Cocoa Touch dan antarmuka iOS benar-benar dirancang ulang untuk menangani *multi-touch*. Semua fitur antarmuka yang biasa digunakan pada *platform* iOS seperti iPhone, iPad dan iPod Touch telah disediakan pada Cocoa Touch *framework* (Apple, Cocoa: Mac OS X Technology Overview, 2011).

Dibangun di atas paradigma *Model-View-Controller*, Cocoa Touch menyediakan dasar yang kuat untuk menciptakan sebuah aplikasi menarik. Ketika dikombinasikan dengan alat pengembang pada Xcode seperti *Interface Builder*, keduanya akan menjadi mudah dan menyenangkan ketika menggunakan sistem *drag-and-drop* pada saat pembuatan aplikasi.

* + 1. **UIKit**

UIKit adalah sebuah *framework* yang menyediakan kelas-kelas untuk membangun dan mengatur perancangan antarmuka aplikasi iOS. UIKit menyediakan objek, *event handling*, penggambaran model, *windows*, *views*, dan kontrol yang didesain secara spesifik untuk antarmuka layar sentuh (Apple, UIKit Framework References, 2011, p. 26).

* + 1. **Swift**

Swift adalah bahasa pemrograman baru yang dikembangkan oleh Apple untk membangun aplikasi berbasi iOs dan OS X. Swift dibangun menggunakan bahasa pemrograman C dan Objective C. Swift mengadopsi *safe programming pattern* dan menambahkan fitur-fitur modern yang membuat *programming* lebih mudah, lebih fleksibel dan menyenangkan. Swift didukung oleh *Cocoa* dan *Cocoa* *Touch framework* yang telah matang dan banyak disukai.

Swift sudah dikembangkan selama bertahun-tahun. Apple meletakkan dasar untuk Swift dengan meningkatkan *compiler* yang telah ada. *Debugger*, dan infrastruktur, menyederhanakan manajemen memori dengan *Automatic Reference Counting* (ARC).

Swift adalah bahasa pemrograman yang mudah dipelajari untuk pemula, *syntax* yang *expressive* membuat belajar Swift lebih menyenangkan. Swift menyediakan fitur *playgrounds* yang memungkinkan programmer untuk bereksperimen dan langsung melihat hasilnya tanpa perlu *build* dan menjalankan sebuah aplikasi.

Swift menggabungkan bahasa pemikiran pemrograman modern dengan budaya *Engineering* Apple, Apple mengoptimalkan kinerja *compiler* dan proses *development* yang lebih mudah tanpa mengorbankan performa untuk sedekar mebuat “Hello world” ataupun aplikasi kompleks. Ini yang membuat swift sebagai investasi untuk *developer* dan Apple di masa depan (Neuburg, 2014).

* + 1. **Core Data**

Core Data adalah sebuah *framework* yang menyediakan solusi untuk tugas-tugas umum yang berhubungan dengan siklus objek dan pengelolaan objek (Apple, Core Data, 2010, pp. 17-19).

Fitur dari Core Data antara lain:

1. Menyediakan bantuan untuk melakukan *undo* dan *redo*.
2. Mampu mengatur konsistensi dari relasi antar objek.
3. Dapat mengurangi memori yang berkelebihan dari suatu program dengan memuat objek-objek.
4. Melakukan validasi terhadap nilai properti dari sebuah objek secara otomatis.
5. Mengizinkan pengguna mengubah skema yang telah ada dengan lebih mudah.
6. Menggunakan NSFetchedResultsController untuk terintegrasi dengan Cocoa pada Mac OS X.
7. Mengumpulkan *accessor* (*getter* –metoda yang mengembalikan nilai dari *private class*) yang sesuai untuk relasi *to-many*.
8. Mengelompokkan dan mengorganisasi data pada memori dan pada antarmuka.
9. Mendukung penyimpanan objek pada tempat penyimpanan data eksternal.
10. *Meng-compile* *query* dengan baik.
11. Menyediakan fitur *locking* untuk mengatasi permasalahan pada *multi-writer*.

Terdapat banyak alasan mengapa Core Data baik untuk digunakan. Salah satunya adalah jumlah kode yang ditulis untuk mendukung layer model dari aplikasi akan menjadi 50% sampai 70% lebih sedikit (dihitung dari baris kode). Ini dikarenakan oleh fitur-fitur yang disediakan oleh Core Data.

Core Data mempunya kode dimana kualitasnya dijaga melalui tes unit dan sudah digunakan setiap harinya untuk aplikasi-aplikasi yang bervarian. *Framework* ini sudah sangat teroptimisasi. Untuk menyediakan keamanan dan *error-handling* yang baik, Core Data juga menyediakan skalabilitas memori untuk solusi apapun.

Selain dari keuntungan diatas, Core Data juga mampu untuk terintegrasi dengan baik dengan Mac OS X *tool*. *Model design tool* mampu membantu para pengguna untuk membuat skema secara cepat dan mudah. Pengguna juga bisa menggunakan *template* pada *Instruments application* untuk mengukur performa Core Data dan untuk menangani berbagai macam masalah. Pada *desktop* Mac OS X, Core Data juga terintegrasi dengan *Interface Builder* untuk membantu pengguna membangun antarmuka berdasarkan skema model yang dibuat.

Core Data sendiri bukan merupakan sebuah basis data *relasionalatau Relational Database Management System* (RDBMS). Core Data menyediakan infrastuktur untuk mengatur dan menyimpan objek dan mengambil objek tersebut dari sebuah tempat penyimpanan, seperti SQLite dan RealmDb. Core Data tidak menghilangkan keharusan pengguna untuk menulis kode. Walaupun pengguna dapat menggunakan *Data Modeling Tool* dan *Interface Builder* pada Xcode, untuk aplikasi yang lebih nyata, pengguna tetap harus menulis kode.

* + 1. **Alamofire**

Alamofire adalah *package* yang digunakan untuk mempermudah pemanggilan API. Di bahasa pemrograman swift sudah tersedia fungsi untuk pemanggilan API, namun penggunaannya sangat tidak *developer friendly*. Dengan menggunakan Alamofire penulisan kode untuk pemanggilan API jadi lebih singkat dan tentunya lebih mudah untuk dibaca oleh developer. Berikut adalah contoh pemanggilan API menggunakan Alamofire di Aplikasi absensi otomatis pada bagian mengambil data karyawan.

Alamofire.request(url, method: method, parameters: parameters, headers: nil).responseJSON {

response in

print(response)

}

variable parameter url adalah url pemanggilan api dan parameters adalah parameter yang dibutuhkan dalam pemanggilan API. Dari contoh code yang dapat dilihat dengan hanya dengan sedikit baris code sudah bisa melakukan pemanggilan API, menurut penulis penggunaan Alamofire sangat membantu dalam pengembangan aplikasi.

* + 1. **Metode *Waterfall***

Menurut Pressman (2015:42), model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Nama model ini sebenarnya adalah “*Linear Sequential Model*”. Model ini sering disebut juga dengan “*classic life cycle*” atau metode *waterfall*. Model ini termasuk ke dalam model *generic* pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai dalam *Software Engineering* (SE). Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan.

Fase-fase dalam *Waterfall* Model menurut referensi Pressman :

1. *Communication* (*Project Initiation & Requirements Gathering*)

Sebelum memulai pekerjaan yang bersifat teknis, sangat diperlukan adanya komunikasi dengan *customer* demi memahami dan mencapai tujuan yang ingin dicapai. Hasil dari komunikasi tersebut adalah inisialisasi proyek, seperti menganalisis permasalahan yang dihadapi dan mengumpulkan data-data yang diperlukan, serta membantu mendefinisikan fitur dan fungsi *software*. Pengumpulan data-data tambahan bisa juga diambil dari jurnal, artikel, dan internet.

1. *Planning* (*Estimating, Scheduling, Tracking*)

Tahap berikutnya adalah tahapan perencanaan yang menjelaskan tentang estimasi tugas-tugas teknis yang akan dilakukan, resiko-resiko yang dapat terjadi, sumber daya yang diperlukan dalam membuat sistem, produk kerja yang ingin dihasilkan, penjadwalan kerja yang akan dilaksanakan, dan *tracking* proses pengerjaan sistem.

1. *Modeling* (*Analysis & Design*)

Tahapan ini adalah tahap perancangan dan permodelan arsitektur sistem yang berfokus pada perancangan struktur data, arsitektur *software*, tampilan *interface*, dan algoritma program. Tujuannya untuk lebih memahami gambaran besar dari apa yang akan dikerjakan.

1. *Construction* (*Code* & *Test*)

Tahapan Construction ini merupakan proses penerjemahan bentuk desain menjadi kode atau bentuk/bahasa yang dapat dibaca oleh mesin. Setelah pengkodean selesai, dilakukan pengujian terhadap sistem dan juga kode yang sudah dibuat. Tujuannya untuk menemukan kesalahan yang mungkin terjadi untuk nantinya diperbaiki.

1. *Deployment* (*Delivery*, *Support*, *Feedback*)

Tahapan *Deployment* merupakan tahapan implementasi *software* ke *customer*, pemeliharaan *software* secara berkala, perbaikan *software*, evaluasi *software*, dan pengembangan *software* berdasarkan umpan balik yang diberikan agar sistem dapat tetap berjalan dan berkembang sesuai dengan fungsinya. (Pressman, 2015:17)

* + 1. ***White Box Testing***

Menurut Black (2009 :2), *Structural test* (atau yang biasa disebut *white-box tests* dan *glass-box tests*) menemukan *bug* dalam elemen struktur tingkat rendah seperti yang terjadi di tingkatan *code*, *database scemas*, *chips*, *subassemblies* dan *interfaces*. Pengujian struktural ini didasarkan pada bagaimana suatu sistem beroperasi. Contohnya, pengujian struktural akan mengungkapkan tempat penyimpanan *database* masih memiliki ruang penyimpanan *username* sebanyak 80 karakter, tetapi pada kenyataannya hanya memungkinkan menyimpan 40 karakter.

Pengujian struktural melibatkan pengetahuan teknis terperinci dari sistem. Untuk menguji *software*, *tester* membuat pengujian yang paling struktural dengan melihat kode dan struktur data itu sendiri. Untuk pengujian *hardware*, *tester* membuat pengujian struktural untuk membandingkan spesifikasi *chip* untuk pembacaan *oscilloscopes* atau meter tegangan.

* 1. **Metode Algorithma**