**BAB III  
LANDASAN TEORI**

1. Sistem Informasi Geografis

Sistem Informasi Geografis (SIG) adalah sistem informasi khusus yang mengelola data yang memiliki informasi spasial (bereferensi keruangan). Atau dalam arti yang lebih sempit, adalah sistem komputer yang memiliki kemampuan untuk membangun, menyimpan, mengelola dan menampilkan informasi bereferensi geografis, misalnya data yang diidentifikasi menurut lokasinya, dalam sebuah database.

Sistem Informasi Geografis adalah kumpulan yang terorganisir dari perangkat keras, perangkat lunak, data geografi, dan personel yang didesain untuk memperoleh, menyimpan, memperbaiki, memanipulasi, menganalisis, dan menampilkan semua bentuk informasi yang bereferensi geografi [8].

Data sistem informasi geografis adalah salah satu komponen krusial dan penting. Di dalam SIG terdapat dua jenis data yaitu data spasial dan data atribut/non spasial.

Data spasial adalah data yang menggambarkan suatu dimensi ruang. Beberapa tipe data spasial antara lain titik, garis, dan polygon.

Titik merupakan representasi grafis yang paling sederhana. Representasi ini tidak memiliki dimensi tetapi dapat diidentifikasi di atas peta dan dapat ditampilkan pada layar monitor. Pada skala tertentu biasanya titik digunakan untuk menggambarkan letak suatu kota, letak suatu bangunan atau ojek-objek lainnya.

Format titik memiliki ciri-ciri yaitu koordinat tunggal, tanpa panjang, tanpa luasan. Contoh dari format titik lokasi kecelakaan, letak pohon, lokasi gedung. Garis merupakan bentuk linier yang akan menghubungkan beberapa titik atau paling sedikit dua titik. Biasanya digunakan untuk menggambarkan suatu objek berdimensi satu.Contoh penggunaan garis pada SIG adalah jaringan jalan, jaringan saluran air, jaringan telepon dan lain sebagainya.Format garis memiliki ciri-ciri yaitu koordinat titik awal dan akhir, mempunyai panjang, tanpa luasan. Contoh dari format garis : jalan, sungai. Bentuk poligon biasanya digunakan untuk mempresentasikan suatu objek berdimensi dua. Suatu wilayah penggunaan lahan suatu tempat adalah entitas yang umumnya digambarkan dengan bentuk poligon. Formatpoligon memiliki ciri-ciri yaitu koordinat dengan titik akhir sama dengan titik awal, mempunyai panjang, mempunyailuasan. Contoh dari format poligon : persil tanah, wilayah,tutupan lahan dan lain-lain.

Data atribut adalah data yang mendeskripsikan data spasial. Biasanya data atribut adalah data berbentuk teks. Data atribut dapat dideskripsikan dengan dua cara, yaitu kualitatif dan kuantitatif. Dalam deskripsi kualitatif maka data atribut akan mendeskripsikan tipe atau klasifikasi suatu objek. Sedangkan secara kuantitatif, data atribut akan dideskripsikan berdasarkan tingkatan [6].

1. Codeigniter

*CodeIgneiter* (CI) adalah *framework* pengembangan aplikasi (*Aplication Development Framework*) dengan menggunakan PHP, suatu kerangka untuk bekerja atau membuat program dengan menggunakan PHP yang lebih sistematis. Pemrogram tidak perlu membuat program dari awal (*from scracth*), karena CI menyediakan sekumpulan librari yang banyak yang diperlukan untuk menyelesaikan pekerjaan yang umum, dengan menggunakan antarmuka dan struktur logika yang sederhana untuk mengakses librarinya. Pemrogram dapat memfokuskan diri pada kode yang harus dibuat untuk menyelesaikan suatu pekerjaan [7].

*CodeIgniter* adalah sebuah *framework* PHP yang dapat membantu mempercepat *developer* dalam pengembangan aplikasi *web* berbasis PHP dibanding jika menulis semua kode program dari awal [8].

*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 opensorce*.
2. Ditulis Menggunakan PHP 4 Meskipun *CodeIgniter* dapat berjalan di PHP 5, namun sampai saat ini kode program *CodeIgniter* masih dibuat dengan menggunakan PHP
3. 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*.URL yang Sederhana

1. Secara default, URL yang dihasilkan *CodeIgniter* sangat bersih dan *Serach Engine Friendly* (SEF).
2. 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*.

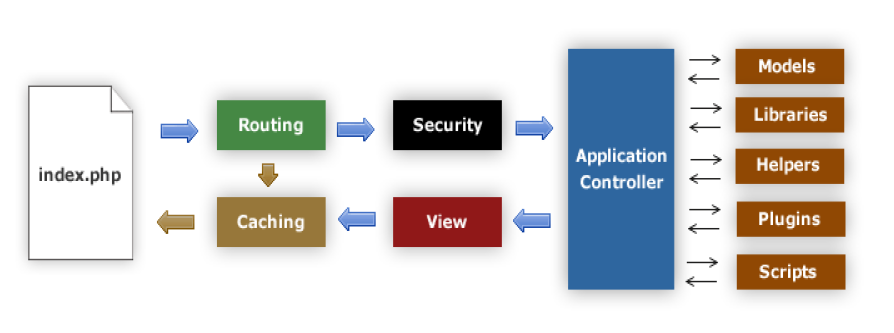
1. Tidak Memerlukan *Template Engine*

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/).

Gambar 3. 1 Application Flowchart CodeIgniter

Keterangan :

*1. Index.php* berfungsi sebagai *front controller*, menginisialisasi *base resorce* untuk menjalankan *CodeIgniter*.

*2. Router* memerikasa HTTP *request* untuk menentukan apa yang harus dilakukan dengannya.

3. Jika *Cache* aktif, maka hasilnya akan langsung dikirimkan ke *browser* dengan mengabaikan aliran data normal.

*4. Security*. Sebelum *Controller* dimuat, HTTP *request* dan data yang dikirimkan *user* akan difilter untuk keamanan.

*5. Controller* memuat *model*, *core* *libraries*, *plugins*, *helpers* dan semua *resource* yang diperlukan untuk memproses *request*.

6. Akhirnya *View* yang dihasilkan akan dikirimkan ke *browser*. Jika *Cache* aktif, maka *View* akan disimpan sebagai *Cache* dahulu, sehingga pada *request* berikutnya langsung dapat ditampilkan.

1. Database MySQL

MySQL adalah salah satu perangkat lunak *Database* *Managemen*t *System (*DBMS*)* yang sering digunakan saat ini, yang didistribusikan secara gratis di bawah lisensi GPL *(GNU General Public License).* Sehingga setiap orang mudah untuk mendapatkan dan bebas untuk menggunakan MySQL, [9]

MySQL adalah sebuah program *database* *server* yang mampu menerima dan mengirimkan datanya sangat cepat, *multi user* serta menggunakan peintah dasar SQL *( Structured Query Language* ), [12]*.*

MySQL merupakan dua bentuk lisensi, yaitu *FreeSoftware* dan *Shareware*. MySQL yang biasa kita gunakan adalah MySQL *FreeSoftware* yang berada dibawah Lisensi GNU/GPL ( *General Public License* ).

MySQL Merupakan sebuah *database* *server* yang *free*, artinya kita bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya. MySQL pertama kali dirintis oleh seorang programmer *database* bernama Michael Widenius . Selain *database* *server*, MySQl juga merupakan program yang dapat mengakses suatu *database* MySQL yang berposisi sebagai *Server*, yang berarti program kita berposisi sebagai *Client*. Jadi MySQL adalah sebuah *database* yang dapat digunakan sebagai *Client* mupun *server*.

*Database* MySQL merupakan suatu perangkat lunak *database* yang berbentuk *database* relasional atau disebut *Relational* *Database* *Management* *System* ( RDBMS ) yang menggunakan suatu bahasa permintaan yang bernama SQL (*Structured Query Language* ).

1. Unified Modeling Language (UML)

UML *(Unified Modeling Language)* adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma (berorientasi objek). Pemodelan *(modeling)* sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami. Diagram – Diagram UML [10]:

1. *Use Case Diagram*

Alat komunikasi tingkat tinggi untuk mewakili persyaratan sistem. Diagram menunjukkan interaksi antara pengguna dan entitas *eksternal* lainnya dengan sistem yang sedang dikembangkan.

|  |  |  |
| --- | --- | --- |
| Simbol | Nama | Keterangan |
|  | *Actor* | Pelaku, seseorang atau apa saja yang berhubungan dengan sistem yang sedang dibangun. |
|  | *Use case* | Menggambarkan bagaimana seseorang menggunakan sistem. |
|  | Relasi | Hubungan dengan *use case* lain, maupun relasi antara aktor dan *usecase*. |
|  | Batasan Sistem | Untuk memperlihatkan batasan sistem dalam diagram *use case*. |
|  | *Dependency* | Suatu hubungan *semantic* antara dua *things* dimana perubahan pada suatu *thinks* (*independent*) mungkin mempengaruhi *semantic* *things* (*independent*) lain. |
|  | *Realization* | Hubungan antara antarmuka yang tesedia secara umum (*interface* atau *usecase*) dengan penerapan detail dari antarmuka (*class*, *package*, atau *reliazation*). |

Tabel 3.1 UML

1. *Activity Diagram*

Menangkap alur dari sebuah sistem, termasuk tindakan utama dan poin keputusan. Diagram ini berguna untuk mendokumentasikan proses bisnis.

Tabel 3. 1 Simbol Activity Diagram

| **Simbol** | **Nama** | **Keterangan** |
| --- | --- | --- |
|  | *Start State* | Menunjukkan dimulainya suatu *workflow* |
|  | *End State* | Menggambarkan akhir dari pada sebuah *activity* *diagram* |
|  | *Activities* | Menggambarkan sebuah pekerjaan atau tugas dalam *workflow* |
|  | *Decision* | Suatu titik atau *point* pada *activity* diagram yang mengindikasikan suatu kondisi dimana ada kemungkinan perbedaan transisi. |
|  | *State Transition* | Menunjukkan kegiatan apa berikutnya setelah kegitan sebelumnya. |
| swimlanes | *Swimlanes* | Siapa yang bertanggung jawab terhadap aktifitas |
| note | *Note* | Suatu *symbol* yang memerikan batasan dan komentar yang dikaitkan pada suatu elemen atau kumpulan elemen. |

1. *Class Diagram*

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas.

Tabel 3. 2 Simbol Class Diagram

| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| --- | --- | --- | --- |
| 1 |  | *Generalization* | Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
| 2 |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek. |
| 3 |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
| 4 |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 5 |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |
| 6 |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri |
| 7 |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya |

1. Sequence Diagram

Sequence diagram secara khusus menjabarkan sebuah Use Case. Diagram ini menunjukkan sejumlah objek dan pesan yang melewati suatu objek.

Tabel 3. 3 Simbol Sequence Diagram

| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| --- | --- | --- | --- |
| 1 |  | *LifeLine* | Objek *entity*, antar muka yang saling berinteraksi. |
| 2 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |
| 3 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |

1. Google Maps API

Google *Maps* API merupakan perkembangan dari google *Maps*. Dengan menggunakan google *Maps* API ini, dimungkinkan untuk dapat menggunakan google *Maps* di dalam website. Meski awalnya hanya *JavaScript* API, *Maps* API diperluas untuk menyertakan sebuah API untuk aplikasi Adobe Flash. Keberhasilan google *Maps* API telah melahirkan sejumlah pesaing antara lain *Yahoo! Maps API, Bing Maps Platforn, MapQuest Development Platform dan OpenLayers* [11].

1. XAMPP

XAMPP adalah paket stack server web platform cross-platform sumber terbuka dan terbuka yang dikembangkan oleh Teman Apache, yang sebagian besar terdiri dari Apache HTTP Server, database MariaDB, dan juru bahasa untuk skrip yang ditulis dalam bahasa pemrograman PHP dan Perl. XAMPP adalah singkatan dari Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) dan Perl (P). Ini adalah distribusi Apache yang sederhana dan ringan yang membuatnya sangat mudah bagi pengembang untuk membuat server web lokal untuk tujuan pengujian dan penerapan. Semua yang dibutuhkan untuk membuat aplikasi server - server web (Apache), database (MariaDB), dan bahasa scripting (PHP) - disertakan dalam file yang dapat di ekstrak. XAMPP juga cross-platform, yang berarti bekerja sama baiknya di Linux, Mac dan Windows. Karena sebagian besar penyebaran server web sebenarnya menggunakan komponen yang sama seperti XAMPP, ini membuat transisi dari server uji lokal ke server live sangat mudah juga [12].

1. UI/UX

Dalam sebuah sistem komputer memiliki tiga aspek yaitu perangkat keras (hardware), perangkat lunak (software), dan manusia (brainware) yang saling terkait dan berhubungan. User interface atau antarmuka pengguna menggunakan bentuk tampilan grafis yang berhubungan langsung dengan pengguna. Interface adalah sebuah tempat di mana interaksi antara pengguna dan sistem pada antarmuka pengguna adalah pengoperasian dan kendali sistem operasi efektif dan umpan balik dari sistem operasi yang mambantu operator membuat keputusan operasional.

User Interface adalah bagian dari komputer dan perangkat lunak yang orang bisa melihat, mendengar, menyentuh, berbicara atau dimengerti. UI pada intinya memiliki dua komponen yaitu input dan output. Input adalah cara seseorang menyampaikan kebutuhannya atau keinginannya kepada komputer. Perangkat input yang umum adalah keyboard dan mouse. Output adalah cara komputer menyatakan hasil dari perhitungan dan kebutuhan pengguna [13].

Dalam menjalankan sebuah bisnis, diperlukan sebuah strategi yang tepat agar bisnis tersebut dapat berjalan dengan sukses dan berhasil. User Experience (UX) merupakan salah satu strategi mendesain produk yang berfokus pada perspektif pengguna. Strategi User Experience dibangun dan diimplementasikan bersamaan dengan strategi bisnis dan produk perusahaan untuk melihat produk kita dari perspektif pengguna. Strategi User Experience juga dapat digunakan untuk melihat akan seperti apa interaksi pelanggan dengan perusahaan melalui beragam produk yang telah dihasilkan. belajar-ux-product-design

Terdapat 4 poin utama dalam Strategi User Experience (UX) :

1. Mendefinisikan value yang dapat diberikan ke pengguna dan mengeksplorasi produk apakah dapat mencapai tujuan2 dari bisnis.
2. Spesifikasi tujuan, mengidentifikasi setiap kesempatan yang dapat meningkatkan produk, dan mengeksplorasi setiap fese interaksi pengguna dengan produk untuk mengidentifikasi apakah komponen-komponen produk telah sesuai.
3. Merencanakan pengembangan produk dan peningkatan secara terus-menerus.
4. Mendefinisikan kesuksesan produk dan metode yang digunakan untuk memvalidasi keberhasilan sebuah produk :

Mengimplementasikan User Experience (UX) Strategy membantu kita untuk:

1. Membuat Seluruh tim ikut dalam pengembangan strategi produk yang dibuat
2. Memperjelas pemikiran desain yang abstrak
3. Keputusan desain dan Pengembangan didasari oleh visi yang sudah disetujui bersama
4. Mengurangi waktu karena tidak perlu membangun komponen yang tidak berkontribusi kepada pengguna atau nilai bisnis.

Bagaimanapun juga dalam strategi User Experience (UX) ini, strategi harus tetap fleksibel sesuai dengan metode dan pendekatan yang digunakan dalam pengembangan produk. Sebuah perusahaan harus belajar User Experience (UX) Strategy agar dapat dijalankan secara konsisten karena menentukan keberhasilan desain dari sebuah produk [14].

1. *Usability Testing*

Menurut Nielsen dalam (Aelani dan Falah, 2012) Secara umum *usability* mengacu pada sejauh mana *user* dapat belajar dan menggunakan suatu produk untuk mencapai tujuannya dan sejuh mana kepuasan *user* dalam menggunakan produk tersebut.

*Usability* didefinisikan melalui 5 komponen kualitas yaitu:

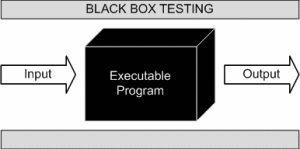
1. *Learnbility*, mengatur semudah apa user dapat mempelajari cara penggunaan produk trsebut untuk pertama kali.
2. *Efficiency*, mengukur secepat apa user dapat melakukan tugasnya.
3. *Memorability,* sejauh mana *user* dapat mengingat langkah-langkah atau proses yang dilakukan dalam mencapai tujuannya.
4. *Error,* sebanyak apa *user* melakukan *error* dan sejauh mana akibat *error* tersebut, serta apakah mudah bagi *user* untuk mengatasi *error* tersebut.
5. *Satisfaction,* bagaimana perasaan *user* ketika menggunakan produk atau tanggapan terhadap desain produk secara keseluruhan.

Inti utama *usability* adalah menjawab pertanyaan, apakah produk tersebut sesuai dengan kebutuhan *user* [15]*.*

1. Black Box Testing

*Black Box Testing* atau Pengujian Kotak Hitam atau juga disebut *Behavioral* *Testing*, berfokus pada persyaratan fungsional dari perangkat lunak. Artinya, teknik *Black Box* *Testing* memungkinkan untuk mendapatkan set kondisi masukan yang sepenuhnya akan melaksanakan semua persyaratan fungsional untuk suatu program [16].

*Black Box* *Testing* bukan merupakan alternatif dari pengujian *White Box Testing*. Sebaliknya, *Black Box* *Testing* adalah pendekatan komplementer yang mungkin untuk mengungkap kelas yang berbeda dari kesalahan daripada metode *White Box* *Testing*. Adapun cara kerja *Black Box Testing* dapat dilihat pada Gambar 3.2 :



Gambar 3. 8 Cara Kerja Black Box Testing

*Black Box Testing* mencoba untuk menemukan kesalahan dalam kategori berikut :

1. Fungsi tidak benar atau hilang.
2. Kesalahan *interface* atau antarmuka.
3. Kesalahan dalam struktur data atau akses *database* eksternal.
4. Kesalahan kinerja atau perilaku
5. Kesalahan inisialisasi dan terminasi.