2. TINJAUAN PUSTAKA

2.1. Program

Menurut Sukrisno, program merupakan ekspresi, kata dan juga pernyataan yang disusun & dirangkai menjadi satu kesatuan dalam prosedur, berupa urutan langkah, guna menyelesaikan masalah yang diimplementasikan dengan memakai bahasa pemrograman sehingga bisa dieksesuksi oleh komputer.

Menurut Sugiyono (2005:21), program merupakan suatu rangkaian instruksi-instruksi dalam bahasa komputer yang disusun secara logis dan sistematis.

Menurut Kamus Besar Bahasa Indonesia (KBBI), program yang diciptakan secara khusus sehingga memungkinkan komputer melakukan fungsi tertentu.

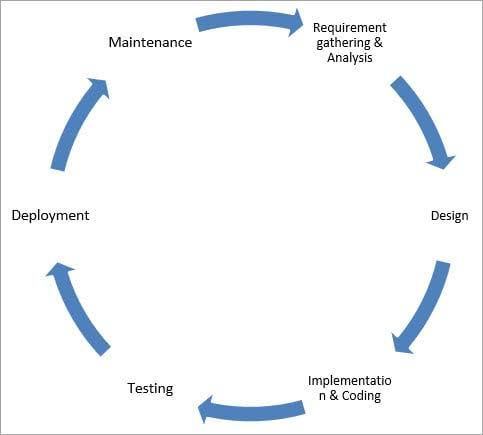
2.2. Systems Development Life Cycle (SDLC)

Menurut Prof. Dr. Sri Mulyani, AK., CA. (2017) SDLC adalah proses logika yang digunakan oleh seorang analis sistem untuk mengembangkan sebuah sistem informasi yang melibatkan requirments, validation, training, dan pemilik sistem.

SDLC digunakan untuk membangun suatu sistem informasi agar dapat berjalan sesuai dengan apa yang diharapkan.

SDLC terdiri dari beberapa tahapan, yang umum diajarkan pada mata pelajaran rekayasa perangkat lunak atau analis sistem, ia terdiri dari 6 tahapan, yakni:

* *Planning* (Perencanaan)
* *Analysis* (Analisis)
* *Design* (Desain)
* *Implementation*
* *Testing & Integration* (pengetesan dan pengintegrasian)
* *Maintenance* (perawatan)



Gambar 2.1 Siklus SDLC

SDLC berisi tahapan-tahapan yang dikembangkan untuk tujuan tertentu. Berikut ini tujuh tahapan yang harus dilewati.

**1. Tahapan Analisis Sistem**

Tahapan pertama, yaitu analisis sistem. Pada tahap ini, sistem akan dianalisis bagaimana akan dijalankan nantinya. Hasil analisis berupa kelebihan dan kekurangan sistem, fungsi sistem, hingga pembaharuan yang dapat diterapkan.

Bagian ini termasuk dalam bagian perencanaan. Bagian lain yang termasuk dalam perencanaan ialah alokasi sumber daya, perencanaan kapasitas, penjadwalan proyek, estimasi biaya, dan penetapan.

Dengan demikian, hasil dari tahap perencanaan ialah rencana proyek, jadwal, estimasi biaya, dan ketentuan. Idealnya manajer proyek dan pengembang dapat bekerja maksimal pada tahap ini.

**2. Tahapan Perancangan Sistem**

Setelah persyaratan dipahami, perancang dan pengembang dapat mulai mendesain *software*. Tahapan ini akan menghasilkan *prototype* dan beberapa *output* lain meliputi dokumen berisi desain, pola, dan komponen yang diperlukan untuk mewujudkan proyek tersebut.

Setelah spesifikasi, kemudian dilakukan perancangan sistem sebagai tahapan kelanjutannya. Tahap ini ialah tahap di mana seluruh hasil analisis dan pembahasan tentang spesifikasi sistem diterapkan menjadi rancangan atau cetak biru sebuah sistem.

Tahap ini disebut sebagai cetak biru, di mana sistem sudah siap untuk dikembangkan mulai dari implementasi, analisis sistem, hingga tenaga pendukung sistem yang akan dikembangkan.

**3. Tahap Pembangunan Sistem**

Pengembangan sistem ialah tahap di mana rancangan mulai dikerjakan, dibuat, atau diimplementasikan menjadi sistem yang utuh dan dapat digunakan. Jika diibaratkan bangunan, tahap ini merupakan tahap membangun.

Tahap ini memakan waktu cukup lama karena akan muncul kendala-kendala baru yang mungkin dapat menghambat jalannya pengembangan sistem. Pada tahapan ini, perancangan bisa saja berubah karena satu atau banyak hal.

Tahap selanjutnya ialah memproduksi perangkat lunak di bawah proses pengembangan. Menurut metodologi yang sudah digunakan, tahap ini dapat dilakukan dengan cepat. *Output* yang dihasilkan pada tahap ini ialah perangkat lunak yang telah berfungsi dan siap diuji.

**4. Tahap Pengujian Sistem**

Sesudah sistem selesai dikembangkan, sistem harus melalui pengujian sebelum digunakan atau dikomersialisasikan. Tahap pengujian sistem harus dijalankan untuk mencoba apakah sistem yang dikembangkan dapat bekerja optimal atau tidak.

Pada tahap ini, ada beberapa hal yang harus diperhatikan, seperti kemudahan penggunaan sampai pencapaian tujuan dari sistem yang sudah disusun sejak perancangan sistem dilakukan. Jika ada kesalahan, tahap pertama hingga keempat harus diperbarui, diulangi, atau pun dirombak total.

Tahap tes SDLC ialah bagian paling penting dalam rangkaian pembuatan sebuah perangkat lunak. Karena sangat tidak mungkin mempublikasikan sebuah *software* tanpa melalui pengujian terlebih dahulu.

Beberapa pengujian yang harus dilewati, antara lain kualitas kode, tes fungsional, tes integrasi, tes performa, dan tes keamanan.

Untuk memastikan pengujian berjalan teratur dan tidak ada bagian yang terlewati, tes dapat dilakukan menggunakan perangkat *Continuous Integration* seperti *Codeship*.

Dari tahap ini, akan dihasilkan perangkat lunak yang telah dites dan siap untuk disebarkan ke dalam proses produksi.

**5. Implementasi**

Implementasi dan pemeliharaan merupakan tahap akhir dalam pembuatan SDLC. Di tahap ini sistem sudah dibuat, diuji coba, dan dipastikan dapat bekerja optimal.

Setelah tahap pembuatan selesai, dilakukan implementasi dan pemeliharaan oleh pengguna. Pemeliharaan sangat penting untuk memastikan sistem bekerja dengan optimal setiap saat.

Untuk implementasi, langkah yang harus dilakukan adalah sebagai berikut.

* Melakukan survei dan penilaian terhadap kelayakan sistem yang sudah dikembangkan.
* Menganalisis dan mempelajari sistem yang sudah ada dan sedang berjalan.
* Melakukan pemecahan masalah dalam pengembangan sistem.
* Menentukan penggunaan *hardware* dan *software* yang tepat.
* Merancang dan mengembangkan sistem baru.
* Memelihara dan meningkatkan sistem yang baru jika diperlukan.

Fase ini disebut juga sebagai tahap penyebaran. Pada tahap ini, *software* disebarkan setelah melewati proses yang melibatkan beberapa persetujuan manual. Tahap ini dilakukan sebelum menurunkan *software* ke produksi.

Proses penyebaran dapat dilakukan menggunakan Application Release Automation (ARA) sebelum masuk ke proses produksi. *Output* yang didapat dari tahap ini ialah perangkat lunak yang siap untuk diproduksi secara massal.

**6. Pemeliharaan Sistem**

Pemeliharaan sistem yang sudah dibuat sangat penting untuk referensi di kemudian hari. Pemeliharaan ialah tahap akhir yang menjadi permulaan fase yang baru yaitu penggunaan.

SDLC belum berakhir di tahap ini. *Software* yang dihasilkan harus terus dipantau untuk memastikan ia berjalan sempurna.

Celah dan kerusakan yang ditemukan pada proses produksi harus dilaporkan dan diselesaikan. Jika ditemukan sebelum diproduksi massal, ini akan lebih baik daripada menyelesaikan dengan merombak semuanya dari awal ke akhir.

2.3. UML (Unified Modelling Languange)

UML atau “Unified Modelling Language” adalah suatu metode permodelan secara visual yang berfungsi sebagai sarana perancangan sistem berorientasi objek.

Definisi UML adalah sebagai suatu bahasa yang sudah menjadi standar pada visualisasi, perancangan, dan juga pendokumentasian sistem aplikasi. Saat ini UML menjadi bahasa standar dalam penulisan *blue print software* (arsitektur).

Menurut (Rosa-Salahuddin, 2011:113), *Unified Modelling Language* atau UML merupakan salah satu standar bahasa yang banyak digunakan di dunia industri untuk menggambarkan kebutuhan (*requirement*), membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (PBO). Berikut ini jenis-jenis dari UML, antara lain:

1. *Use Case Diagram*. Suatu urutan interaksi yang saling berkaitan antara sistem dan aktor. Use case dijalankan melalui cara menggambarkan tipe interaksi antara user
2. *Activity Diagram*. Salah satu jenis diagram pada UML yang dapat memodelkan metode apa saja yang terjadi pada sebuah sistem.
3. *Sequence diagram*. Salah satu jenis diagram pada UML yang menggambarkan hubungan objek yang berdasarkan urutan waktu. Sequence diagram dapat menjelaskan tahapan atau urutan yang harus dilakukan agar dapat menghasilkan sesuatu seperti pada use case diagram.
4. *Class diagram.* Salah satu jenis diagram pada UML yang dipakai untuk menampilkan paket-paket maupun kelas-kelas yang ada pada sebuah sistem yang akan digunakan.
5. *Statemachine diagram*. Salah satu jenis diagram pada UML yang menjelaskan perubahan keadaan maupun transisi suatu objek pada sistem.
6. *Communication diagram*. Salah satu jenis diagram pada UML yang dapat menjelaskan proses terjadinya suatu aktivitas dan diagram ini juga menggambarkan interaksi antara objek yang ada pada sebuah sistem.
7. *Deployment diagram*. Salah satu diagram pada UML yang menunjukan tata letak sebuah program secara fisik. dapat juga diartikan untuk menampilkan bagian-bagian aplikasi yang terdapat pada perangkat keras dan dipakai untuk menerapkan suatu sistem dan hubungan antara komponen hardwere.
8. *Component diagram*. Salah satu jenis diagram pada UML yang menjelaskan softwere pada suatu sistem.
9. *Object diagram*. Salah satu jenis diagram pada UML yang menjelaskan objek-objek pada suatu sistem dan hubungan lainnya.
10. *Composite structure diagram*. Salah satu jenis diagram pada UML yang menjelaskan struktur internal dari (*component, class,* dan *use case*), termasuk hubungan pengklasifikasian ke bagian lain dari sebuah program.
11. *Interaction Overview diagram*. Salah satu jenis diagram pada UML yang berfungsi untuk menvisualisasikan hubungan dan kerjasama antara activity diagram dengan sequence diagram.
12. *Package diagram*. Salah satu jenis diagram pada UML berfungsi untuk mengumpulkan kelas dan juga menunjukan bagaimana elemen model akan disusun serta menggambarkan ketergantungan antara paket-paket.
13. *Timing diagram*. Salah satu jenis diagram pada UML berfungsi sebagai bentuk lain dari interaksi diagram, dimana fokus yang paling utamanya kepada waktu.

Adapun macam-macam dari UML dibagi menjadi 3, antara lain:

* *Structure diagram*. Kumpulan diagram yang berfungsi untuk menjelaskan suatu struktur statis dari sistem yang dimodelkan.
* *Behaviour diagram*. Kumpulan diagram yang digunakan untuk menjelaskan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
* *Interaction diagram*. Kumpulan diagram yang berfungsi untuk menjelaskan interaksi sistem dengan sistem lain maupun antar sistem pada sebuah sistem.

Berikut ini notasi-notasi UML pada di tabel 2.1.

TABEL 2.1 Simbol UseCase Diagram

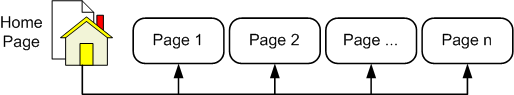
|  |  |
| --- | --- |
| Simbol | Keterangan |
|  | *Actor :* Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi. |
|  | Generalisasi/*generalization* : Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
|  | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor. |
|  | Relasi *use case* tambahan ke sebuah *case* dimana *use case* yang ditambahkan memiliki nama depan yang sama dengan *use case* yang di tambahkan. |
|  | Komunikasi antara *actor* dan *use case* yang berpatisipasi pada *use case* atau *use case* memiliki interaksi dengan *actor.* |
|  | Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang di tambahkan memerlukan *use case* ini untuk menjalankan fungsional atau sebagai syarat dijalankan *use case.* |

2.4. Struktur Navigasi

Struktur navigasi pada *Website* adalah sebuah halaman web atau dokumen web yang buat yang mempunyai struktur yang jelas serta terusun dengan teratur. Rancangan struktur navigasi pada web digunakan untuk untuk mengkaitkan halaman web dengan halaman web yang saling keterhubungan satu sama lain dengan ***hypertext.*** Terdapat beberapa bentuk struktur navigasi yang biasa digunakan pada rancangan struktur navigasi :

2.4.1 Struktur Navigasi Linear

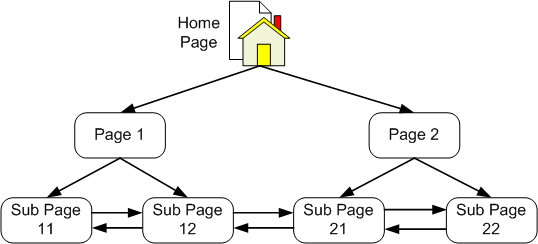
Struktur Linear disajikan dalam bentuk berurutan. Struktur ini tidak memerlukan rancangan bentuk interface yang kompleks dan Usability Testing yang banyak. Implementasi struktur Linear memudahkan penjelajahan dokumen website dan pemahaman konteks website karena pengguna akhir tidak diberi banyak pilihan untuk menjelajah ke dokumen website yang lain. Contoh struktur navigasi yang ditunjukkan pada gambar 2.2.



Gambar 2.2 Struktur Navigasi Linear

2.4.2 Kombinasi Struktur Linear dan Struktur Hirarki

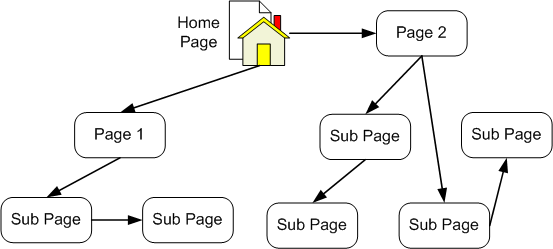
Kombinasi struktur Linear dan Struktur Hierarchy, merupakan struktur yang biasa digunakan dalam sebuah rancangan sistem berbasis web. Rancangan suatu website memiliki keterhubugan dokumen website yang disesuaikan dengan kebutuhan customer. Contoh struktur navigasi yang ditunjukkan pada gambar 2.3.



Gambar 2.3 Kombinasi Struktur Navigasi Linear dan Struktur Hirarki

2.4.3 Struktur Rangkaian / Jaringan

Struktur rangkaian atau jaringan merupakan keterhubungan satu dokumen website yang mempunyai sedikit atau tidak ada struktur secara keseluruhan. Hypertext dibangun dengan hierarki yang jelas terlihat dan yang tersusunan bertingkat-tingkat. Contoh struktur navigasi yang ditunjukkan pada gambar 2.4



Gambar 2.4 Struktur Navigasi Hirarki

2.5 HTML (HyperText Markup Languange)

HTML merupakan singkatan dari *Hypertext Markup Languange*. Tujuan utama pengembangan HTML adalah untuk menghubungkan satu halaman web dengan halaman web lainnya. HTML merupakan bahasa dasar web berfungsi untuk menampilkan berbagai komponen web[4].HTML akan dikirimkan oleh *webserver* *application* dan di-render pada browser untuk dijadikan tampilan pada *web browser*. Penulisan HTML terdiri atas tag – tag tertentu yang memiliki fungsi dan maksud masing. Setiap tak memiliki pembuka dan penutup. Penutup tag pada HTML ditulis menggunakan tanda / setelah kurung buka. Penulisan tag HTML juga harus menggunakan kata yang sama persis, termasuk besar kecilnya huruf.

Tabel tag-tag HTML dapat dilihat pada Tabel 2.4 Tag-tag HTML :

Tabel 2.2 Tag pada HTML

|  |  |
| --- | --- |
| Tak HTML | Keterangan |
| <!DOCTYPE> | Tag untuk menentukan tipe dokumen |
| <html>..</html> | Tag untuk membuat sebuah dokumen HTML |
| <title> .. </title> | Tag untuk membuat judul dari sebuah halaman |
| <body>.. </body> | Tag untuk membuat tubuh dari sebuah halaman |
| <h1>to<h6> | Tag menformat text menjadi judul tipe 1-6 (terkecil-terbesar) |
| <p> .. </p> | Tag membuat paragraf |
| <br> | Memasukkan satu baris putus |
| <hr> | Tag membuat perubahan dasar kata dalam isi |
| <!-- --> | Tag untuk membuat komentar |
| <form> .. </form> | Tag untuk membuat sebuah form HTML untuk input pengguna |
| <input> | Tag untuk membuat sebuah kontrol input |
| <textarea> | Tag untuk membuat sebuah tombol yang dapat diklik |
| <label> .. </label> | Tag untuk membuat sebuah label untuk sebuah elemen <input> |
| <img> | Tag untuk membuat gambar |
| <a> .. </a> | Tag untuk membuat hyperlink |
| <link> | Tag untuk membuat hubungan antara dokumen dan sumber daya ekternal(seperti link ke stylesheet) |
| <nav> | Tag untuk membuat navigasi link |
| <ul> | Tag untuk membuat daftar selain nomor |
| <li> | Tag untuk membuat sebuah item daftar |
| <table> </table> | Tag untuk membuat tabel |
| <caption> | Tag untuk membuat caption table |
| <th> .. </th> | Tag untuk membuat sebuah sel header tabel |
| <tr> .. </tr> | Tag untuk membuat baris dalam sebuah tabel |
| <td> .. </td> | Tag untuk membuat sel dalam sebuah tabel |
| <thead>.. <thead> | Mengelompokan isi header dalam sebuah tabel |
| <tbody>.. </tbody> | Mengelompokan isi tubuh sebuah tabel |
| </head> | Tag untuk membuat informasi tentang dokumen |
| <script>..</script> | Tag untuk membuat script sisi klien |

2.6 CSS (Cascading Style Sheet)

CSS atau *Cascading Style Sheet* adalah kumpulan perintah yang digunakan untuk menjelaskan tampilan sebuah halaman situs web dalam *mark-up languange*. *Mark-up languange* atau bahasa markah adalah bahasa pemograman yang biasanya digunakan membuat webs. CSS memberikan cara yang mudah dan efisien bagi pemograman untuk menentukan tata letak halaman web dan mempercanik halaman dengan elemen desain seperti warna, sudut bulat, gradien, dan animasi[5].

CSS beroperasi melalui tag <style> dengan *atribut class* warna. Dengan adanya CSS pada HTML tersebut maka pengaturan warna teks akan menjadi lebih mudah. bahwa CSS akan menghemat waktu dengan perintah-perintah yang efisien. Hal ini bisa terjadi karena CSS sendiri dikembangkan untuk bisa mengubah tampilan laman website tanpa harus mengganti isi konten.

Dengan menguasai CSS dapat lebih mudah mengendalikan website. sehingga tidak harus pusing-pusing mengutak-atik HTML saat harus memprogram ulang tampilan laman. Bahkan pada CSS tingkat lanjut, bisa mengatur website lewat medium yang lain seperti suara. Keuntungan lain dari belajar CSS adalah tingkat compatibility yang tinggi. CSS kompatibel dengan kebanyakan browsers dan sangat *user-friendly*.

2.7 PHP (Hypertext Preprocessor)

PHP merupakan bahasa pemograman untuk membuat atau mengembangkan situs web statis atau situs web dinamis atau aplikasi Web[6]. PHP digunakan karena untuk membuat website dinamis bisa digunakan untuk menyimpan data ke dalam database, membuat halaman yang dapat berubah-ubah sesuai dengan *input user*, memproses form, dll.

Dalam penulisan PHP yang benar diawali dengan “<?php” dan diakhiri dengan “?>”. Dan di dalam File PHP juga dapat berisi tag seperti HTML dan skrip sisi klien seperti JavaScript.

Di bawah ini adalah beberapa alasan kuat mengapa banyak orang lebih memilih PHP dan menjadikan PHP menjadi bahasa pemrograman terpopuler :

* 1. PHP adalah merupakan bahasa yang bersifat *open source* dan gratis.
  2. Proses belajar pemrograman PHP yang lebih singkat dibandingkan dengan bahasa lain seperti JSP, ASP dll.
  3. Karena bahasa pemrograman terpopuler, Maka komunitas banyak dokumentasi lebih besar
  4. Sebagian besar *server web hosting* mendukung PHP secara default tidak seperti bahasa lain seperti ASP yang membutuhkan IIS. Ini menjadikan PHP pilihan yang hemat biaya.
  5. PHP diperbarui secara teratur untuk mengikuti tren teknologi terbaru.
  6. PHP adalah bahasa *scripting sisi server* ; yang berarti hanya perlu menginstalnya di server dan komputer klien yang meminta sumber daya dari server tidak perlu menginstal PHP; hanya peramban web saja sudah cukup.
  7. PHP bekerja sama dengan MySQL ; Hal ini tidak berarti dapat menggunakan PHP dengan sistem manajemen basis data lainnya. Anda masih dapat menggunakan PHP dengan Postgres, Peramal, MS SQL Server ODBC dll.
  8. PHP adalah bahasa lintas *platform*; yang berarti dapat pada aplikasi di sejumlah sistem operasi yang berbeda seperti Windows, Linux, Mac OS dll.

2.8 MySql

**MySQL** adalah perangkat lunak sistem manajemen basis data atau database management system (DBMS) yang *multithread* dan *multiuser*. **MySQL** sebuah implementasi dari sistem manajemen basis data relasional(**RDMS**) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Setiap pengguna dapat dengan bebas menggunakan MySQL, akan tetapi perangkat lunak tersebut tidak boleh dijadikan produk komersial.  
 **MySQL** merupakan turunan dari **SQL (**Structured Query Language**).**SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Beberapa perintah SQL yang tersedia, yaitu :

* *CREATE DATABASE*, untuk membuat sebuah *database*.
* *DROP DATABASE*, untuk menghapus sebuah *database*.
* *CREATE TABLE*, untuk membuat sebuah *table*.
* *ALTER TABLE*, untuk memodifikasi sebuah *table*.
* *DROP TABLE*, untuk menghapus sebuah *table*.
* *SELECT*, untuk menampilkan data dari *database*.
* *UPDATE*, untuk memodifikasi data dari *database*.
* *INSERT INTO*, untuk menambah data di *database*.
* *DELETE*, untuk menghapus data dari *database*.

2.9. Framework Bootstrap

Bootstrap pada awalnya dibuat dan dikembangkan oleh pekerja / programmer Twitter, yaitu Mark Octo dan Jacob Thornton sejak 2011.

Pada waktu itu programmer di Twitter menggunakan berbagai alat dan perpustakaan yang mereka kuasai dan sukai untuk melakukan pekerjaan mereka, sehingga tidak ada standardisasi dalam penamaan kelas.

Akibatnya sulit untuk dikelola, oleh karena itu keduanya membuat alat atau kerangka kerja yang digunakan bersama di lingkungan internal Twitter.

Sejak diluncurkan pada Agustus 2011, Bootstrap telah berkembang dari proyek berbasis CSS menjadi kerangka kerja yang lebih lengkap yang juga berisi *plugin* Javascript, ikon, formulir, dan tombol.

Pada Januari 2012, Bootstrap *realease version* 2.0, yang telah menyertakan fitur tata letak responsif, dan sejak itu Bootstrap telah banyak digunakan sehingga menjadi proyek Github yang disalin dengan berat hingga 20.000 kali.

Dua tahun setelah itu, tepatnya pada Agustus 2014 Bootstrap kembali merilis versi terbaru, versi 3.0 yang telah mengakomodasi konsep *Mobile* terlebih dahulu.

Yang berarti dalam pembuatan dan pengembangan layar terkecil terlebih dahulu (*mobile / smartphone*) dan secara bertahap di layar terbesar.

Dan Bootstrap baru-baru ini merilis versi 4.0 (sebelumnya pada tahun 2017 versi beta telah dirilis) sebagai penyempurnaan dokumentasi dan untuk menambahkan beberapa fitur tambahan yang saat ini sedang *booming*.

Yaitu dengan memberikan tampilan yang lebih baik untuk pengembang e-commerce, dan terlihat lebih stabil dan dengan ini maka versi 2.0 dan 3.0 tidak akan lagi didukung atau diperbarui dalam penggunaannya.

2.9.1 Fungsi Bootstrap

Beberapa fungsi yang dimiliki oleh bootstrap meliputi:

* Bootstrap akan membantu dalam membuat dan merancang situs web yang cepat dan responsif. Artinya, tampilan web yang dibangun memakai bootstrap akan secara otomatis menyesuaikan ukuran layar di browser.
* Bootstrapping juga akan membantu dalam menciptakan dan mengembangkan situs web yang dinamis atau statis. Anda hanya perlu menentukan sesuai dengan kebutuhan anda.
* Bootstrap sudah menyediakan berbagai antarmuka kelas yang berguna untuk membantu situs web menjadi ringan dan cepat saat diakses.
* Boostrap juga membantu dalam membuat pengaturan tata letak halaman di situs web. Ini karena Bootstrap dilengkapi dengan fitur yang disebut kisi.
* Anda juga dapat langsung menambahkan CSS dan kelas sendiri. Dengan begitu, anda bisa mendesain situs web anda sehingga memiliki tampilan yang lebih bervariasi.

2.9.2 Kelebihan Bootstrap

Berikut adalah beberapa kelebihan dan dimiliki oleh Bootstrap, yaitu:

* Dapat digunakan untuk membantu mempercepat saat membuat *front-end* di situs web.
* Bootstrap memiliki tampilan yang modern sehingga dapat membuat situs web anda memiliki penampilan yang menarik.
* Bootstrap akan membuat situs web anda terlihat lebih responsif. Jadi, ketika anda menggunakan ponsel cerdas atau komputer untuk membuat situs web anda, itu akan menyesuaikan dengan ukuran layar.
* Boostrap juga akan membuat situs web anda lebih ringan saat dibuka. Jadi mereka yang ingin mengakses situs web anda tidak perlu menunggu lama. Ini karena sebelumnya proses Bootstrap dilakukan dengan rapi dan sistematis.
* Dengan Bootstrap, anda bisa mendapatkan semua jenis warna, operasi penggunaan, dan juga berbagai variabel yang anda butuhkan. Jadi, anda tidak perlu khawatir lagi karena hasilnya tentu tidak akan berbeda ketika anda membukanya di perangkat yang berbeda.
* Bootstrap dikenal lebih lengkap. Ini karena itu termasuk CSS, HTML dan Javascript.
* *Open source*, artinya anda tidak perlu harus mendapatkan lisensi dan bahkan anda dapat mengembangkan situs web anda sesuai dengan kebutuhan atau kebutuhan anda.
* Kerangka kerja yang dimiliki oleh Bootstrap menggunakan *less*. *Less* adalah teknologi dari CSS yang terkenal dengan kesederhanaannya. Kurang menawarkan banyak fleksibilitas dan kekuatan.

2.10 Microsoft Visual Studio Code

Microsoft Visual Studio Code adalah *one-stop shop* yang memungkinkan kita fokus pada proses pengembangan dan melupakan *tools* baru. Beberapa fitur Visual Studio Code:

1.      *Cross platform* : tersedia di MacOS, Linux dan Windows artinya anda dapat bekerja pada sistem operasi manapun tanpa khawatir belajar *coding tools* yang sama untuk sistem yang berbeda-beda.

2.     *Lightweight* : tak perlu menunggu lama untuk memulai. Anda mengontrol sepenuhnya bahasa, tema, *debugger*, *commands* dan lain-lainnya sesuai keinginan. Ini dapat dilakukan melalui *extentions* untuk bahasa populer seperti Python, Node.js, Java dan lain-lainnya di Visual Studio Code *Marketplace*.

3.      *Powerful editor* : memfungsikan fitur untuk *source code editing* yang sangat produktif, seperti membuat *code snippets*, *IntelliSense*, *auto correct*, dan *formatting*.

4.      *Code Debugging* : salah satu fitur terkeren yang ditawarkan Visual Studio Code adalah membantu anda melakukan *debug* pada kode dengan cara mengawasi kode, variabel, *call stack* dan *expression* yang mana saja.

5*.      Source control* : Visual Studio Code memiliki *integrated source control* termasuk Git *support in-the-box* dan penyedia *source code control* lainnya di pasaran. Ini meningkatkan siklus rilis proyek anda secara signifikan.

6.      *Integrated terminal* : Tiada lagi *multiple windows* dan alt-tabs. Anda dapat melakukan *command-line task* sekejap dan membuat banyak terminal di dalam *editor*.

2.11 Black Box Testing

Black box testing adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak. Bila dianalogikan seperti melihat suatu kotak hitam, *kit* hanya bisa melihat penampilan luarnya saja, tanpa tau ada apa dibalik bungkus hitam nya. Sama seperti pengujian *black box*, mengevaluasi hanya dari tampilan luarnya (*interface*nya), fungsionalitasnya. Tanpa mengetahui apa sesungguhnya yang terjadi dalam proses detilnya (hanya mengetahui *input* dan *output*)[11].

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi *unit testing* juga.

Pengujian pada *Black Box* berusaha menemukan kesalahan seperti:

* Fungsi-fungsi yang tidak benar atau hilang.
* Kesalahan *interface.*
* Kesalahan dalam struktur data atau akses database eksternal
* Kesalahan kinerja
* Inisialisasi dan kesalahan terminasi

Teknik khas *Black Box Testing* meliputi :

1. *Decision table*

Decision Tablel adalah cara yang tepat belum kompak untuk model logika rumit, seperti diagram alur dan jika-then-else dan switch-laporan kasus, kondisi mengaitkan dengan tindakan untuk melakukan, tetapi dalam banyak kasus melakukannya dengan cara yang lebih elegan.

1. *All-pairs testing*

*All-pairs testing* atau *pairwise* testing adalah metode pengujian perangkat lunak kombinatorial bahwa, untuk setiap pasangan parameter masukan ke sistem (biasanya, sebuah algoritma perangkat lunak), tes semua kombinasi yang mungkin diskrit parameter tersebut.

Alasan di balik semua *All-pairs* testing ini: yang sederhana dalam sebuah program umumnya dipicu oleh parameter masukan tunggal. Kategori paling sederhana berikutnya bug terdiri dari mereka bergantung pada interaksi antara pasangan parameter, yang bisa ditangkap dengan menguji semua-pasangan. yang melibatkan interaksi antara tiga atau lebih parameter secara progresif kurang umum , sementara pada saat yang sama waktu semakin lebih mahal untuk mencari oleh pengujian mendalam, yang sebagai batas pengujian lengkap semua input yang mungkin.

1. *State transition table*

Dalam teori automata dan logika sekuensial, state transition table adalah tabel yang menunjukkan apa yang negara (atau negara dalam kasus robot terbatas *non-deterministic*) suatu *semi automaton* terbatas atau mesin *finite state* akan pindah ke, berdasarkan kondisi saat ini dan masukan lainnya. Sebuah tabel negara pada dasarnya adalah sebuah tabel kebenaran di mana beberapa input adalah kondisi saat ini, dan output termasuk negara berikutnya, bersama dengan keluaran lain.

*State transition table* adalah salah satu dari banyak cara untuk menentukan mesin negara, cara lain menjadi diagram negara, dan persamaan karakteristik.

1. *Equivalence partitioning*

*Equivalence partitioning* adalah pengujian perangkat lunak teknik yang membagi data masukan dari unit perangkat lunak menjadi beberapa partisi data dari mana *test case* dapat diturunkan. Pada prinsipnya, uji kasus dirancang untuk menutupi setiap partisi minimal sekali. Teknik ini mencoba untuk mendefinisikan kasus uji yang mengungkap kelas kesalahan, sehingga mengurangi jumlah kasus uji yang harus dikembangkan.

1. *Boundry values analysis*

*Boundary value analysis* merupakan suatu teknik pengujian perangkat lunak di mana tes dirancang untuk mencakup perwakilan dari nilai-nilai batas. Nilai-nilai di tepi sebuah partisi kesetaraan atau sebesar nilai terkecil di kedua sisi tepi. Nilai dapat berupa rentang masukan atau keluaran dari komponen perangkat lunak. Karena batas-batas tersebut adalah lokasi umum untuk kesalahan yang mengakibatkan kesalahan perangkat lunak mereka sering dilakukan dalam kasus-kasus uji.

Dokumentasi komponen *software*, mencangkup pemeriksaan dokumen dari *software* itu sendiri, yaitu :

* *Flowchart* yang dibuat.
* Deskripsi input yang digunakan.
* Deskripsi output yang digunakan.
* Deskripsi output yang dihasilkan.
* Kesesuaian penulisan (akurasi).
* Kontrol/kendali terhadap sistem yang dibuat.