**BAB II**

**TINJAUAN PUSTAKA**

1. **Pengertian dan Ruang Lingkup Pembangunan Aplikasi**
2. **Pengertian Pembangunan**

Pengertian Pembangunan menurut Kamus Besar Bahasa Indonesia (KBBI) Edisi Ketiga (2002:103) adalah cara, proses, perbuatan membangun.

1. **Pengertian Aplikasi**

Aplikasi berasal dari kata *application* yang berarti penerapan, lamaran, atau penggunaan. Secara istilah aplikasi adalah suatu program komputer (perangkat lunak) yang dibuat untuk melakukan tugas/fungsi tertentu yang diinginkan oleh pemakai.

Dalam Wikipedia dikatakan bahwa aplikasi perangkat lunak adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna.

Menurut Sutanta Edhy dalam buku yang berjudul *Pengantar Teknologi Informasi*, menyebutkan bahwa: “*Software* aplikasi merupakan perangkat lunak yang dikembangkan untuk digunakan pada bidang tertentu.” (2005:21).

1. **Pengertian Pembangunan Aplikasi**

Pembangunan aplikasi merupakan suatu mekanisme dalam penciptaan aplikasi (*software*), sehingga aplikasi tersebut dapat digunakan untuk mengolah data menjadi informasi yang bermanfaat serta dapat dijadikan sebagai bahan pengambilan keputukan suatu organisasi dalam mencapai suatu tujuan.

1. **Pengertian Parkir**

Parkir adalah keadaan tidak bergerak suatu kendaraan yang bersifat sementara karena ditinggalkan oleh pengemudinya. Secara hukum dilarang untuk parkir di tengah jalan raya; namun parkir di sisi jalan umumnya diperbolehkan. Fasilitas parkir di bangun bersama-sama dengan kebanyakan gedung. Termasuk dalam pengetian parkir adalah setiap kendaraan yang berhenti di tempat-tempat tertentu baik yang dinyatakan dengan rambu lalu lintas ataupun tidak, serta tidak semata-mata untuk kepentingan menaikkan dan/atau menurunkan orang dan/atau barang.

1. **Pengertian UML (*Unified* *Modeling* *Language*)**

UML (*Unified* *Modeling* *Language*) merupakan sebuah bahasa pemodelan yang digunakan untuk menspesifikasikan, menvisualisasikan, membangun serta mendokumentasikan seluruh rancangan sistem perangkat lunak (*software*). Pemodelan ini digunakan untuk menyederhanakan permasalahan agar lebih mudah dipahami. Penggunaan model ini juga bertujuan untuk mengidentifikasi bagian-bagian yang termasuk ke dalam lingkungan sistem dan bagaimana hubungan antara sistem dengan subsistem maupun lingkungan luarnya.

Selain itu UML (*Unified* *Modeling* *Language*) merupakan metodologi kolaborasi antara metoda-metoda Booch, OMT (*Object* *Modeling* *Technique*), serta OOSE (*Object* *Oriented* *Software* *Engineering*) dan beberapa metoda lainnya. UML merupakan metodologi yang paling sering digunakan saat ini untuk mengadaptasi maraknya penggunaan bahasa “pemogramana berorientasi objek” (OOP). UML menyediakan notasi-notasi yang mampu membantu dalam memodelkan sistem dari berbagai sudut.

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use* *Case* *Diagram,* merupakan diagram yang digunakan untuk memodelkan proses bisnis.
2. *Conceptual* *Diagram,* merupakan diagram yang digunakan untuk memodelkan konsep-konsepyang ada di dalam aplikasi.
3. *Sequence* *Diagram*, merupakan diagram yang digunakan untuk memodelkan pengiriman pesan (messages) antar objek.
4. *Collaboration* *Diagram*, merupakan diagram yang digunakan untuk memodelkan interaksi antar objek.
5. *State* *Diagram*, merupakan diagram yang digunakan untuk memodelkan perilaku objek di dalam sistem.
6. *Activity* *Diagram*, merupakan diagram yang digunakan untuk memodelkan perilaku *Use* *Cases* dan *Objects* di dalam sistem.
7. *Class* *Diagram*, merupakan diagram yang digunakan untuk memodelkan struktur kelas.
8. *Object Diagram,* merupakan digram yang digunakan untuk memodelkan struktur *Object*.
9. *Component Diagram,* merupakan digram yang digunakan untuk memodelkan komponen *Object*.
10. *Deployment Diagram,* merupakan diagram yang digunakan untuk memodelkan distribusi aplikasi.

Berikut akan dijelaskan 4 macam diagram yang paling sering digunakan dalam membangun aplikasi berorientasi objek, yaitu *use* *case* *diagram*, *activity* *diagram*, *sequence* *diagram*, dan *class* *diagram*.

1. ***Use*** ***Case*** ***Diagram***

Use case pada dasarnya merupakan unit fungsionalitas korehen yang diekspresikan sebagai transaksi-transaksi yang terjadi antara actor dan sistem. Use case diagram digunakan untuk mendefinisikan suatu bagian perilaku sistem yang bersifat korehen tanpa perlu menyingkap struktur internal sistem / perangkat lunak yang sedang dikembangkan.

Adapun simbol-simbol yang digunankan dalam Use Case Diagram:

1. ***Actor***

*Actor* merupakan suatu simbol yang menunjukkan pelaku yang berinteraksi dengan sistem.

Berikut adalah simbol *Actor*:



**Gambar 2.1** Simbol *Actor* dalam *Use* *Case* *Diagram*

1. ***Use Case***

*Use Case* merupakan simbol yang menunjukkan suatu fungsi yang bisa dilakukan.

Berikut adalah simbol Use Case:



**Gambar 2.2** Simbol *Use Case* dalam *Use Case Diagram*

1. ***Association***

*Association* merupakan suatu simbol yang digunakan untuk menghubungkan antara objek satu dengan objek yang lainnya.

Berikut adalah simbol *Association*:



**Gambar 2.3** Simbol *Association* dalam *Use Case Diagram*

1. ***Extend***

*Extend* merupakan simbol relasi *use* *case* tambahan ke sebuah *use* *case*. Dimana *use* *case* yang ditambahkan tersebut dapat berdiri sendiri walau tanpa *use* *case* tambahan

Berikut adalah simbol *Extend*:



**Gambar 2.4** Simbol *Extend* dalam *Use Case Diagram*

1. ***Include***

*Include* merupakan suatu simbol yang digunakan untuk menggambarkan relasi *use* *case* tambahan ke sebuah *use case*. Dimana *use* *case* yang ditambahkan tersebut memerlukan *use* *case* ini untuk menjalankan fungsinya.

Berikut merupakan simbol *Include*:



**Gambar 2.5** Simbol *Include* dalam *Use Case Diagram*

1. ***System Boundary***

*System Boundary* merupakan suatu simbol yang menunjukkan batasan dari suatu sistem.

Berikut adalah simbol *System Boundary*:



**Gambar 2.6** Simbol *System Boundary* dalam *Use Case Diagram*

1. ***Generalization***

*Generalization* merupakan suatu simbol yang menggambarkan hubungan generalisasi dan spesialisasi antar dua use case. menunjukkan objek induk yang berbagi perilaku dengan objek anak**.**

Berikut adalah simbol *Generalization*:



**Gambar 2.7** Simbol *Generalization* dalam *Use Case Diagram*

1. ***Activity*** ***Diagram***

*Activity* diagram merupakan bentuk khusus dari *state machine* yang bertujuan memodelkan komputasi-komputasi dan aliran-aliran kerja yang terjadi dalam sistem / perangkat lunak yang sedang dikembangkan. Activity diagram menggabarkan aliran kerja (*workflow*) atau aktivitas kerja dari sebuah sistem.

Keunggulan dari *activity* *diagram* adalah bahwa diagram lebih mudah dipahami dibandingkan dengan skenario. Pada dasarnya *activity* *diagram* menggambarkan skenario secara grafis. *Activity* *diagram* juga sesungguhnya mirip/serupa dengan diagram alir (*flowchart*). Yang membedakan hanyalah *swimlane* (garis tegas dari atas ke bawah) yang menunjukan suatu state berada pada objek/kelas tertentu.

Berikut simbol yang digunakan dalam Activity Diagram.

1. ***Initial*** ***State***

*Initial* *State* merupkan suatu simbol yang menunjukkan titik awal memulai suatu aktivitas.

Berikut adalah *Initial* *State* simbol:



**Gambar 2.8** Simbol *Initial* *State* dalam *Activity Diagram*

1. ***Final*** ***State***

*Final* *State* merupakan suatu simbol yang menunjukkan titik akhir untuk mengakhiri suatu activitas.

Berikut adalah simbol *Final* *State*:



**Gambar 2.9** Simbol *Final* *State* dalam *Activity Diagram*

1. ***Decision***

*Decision* merupakan suatu simbol yang menunjukkan percabangan, menunjukan pilihan (sesuatu yang harus dipilih).

Berikut adalah simbol *Decision*:



**Gambar 2.10** Simbol *Decision* dalam *Actiivity Diagram*

1. ***Fork And Join***

*Fork* *And* *Join*merupakan suatu simbol yang menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan parallel menjadi satu.

Berikut adalah simbol *Fork* *And* *Join*:



**Gambar 2.11** Simbol *Fork* *And* *Join* dalam *Activity Diagram*

1. ***Activity***

*Activity* merupakan suatu simbol yang menggambarkan aktivitas yang berjalan, yang dilakukan oleh sistem

Berikut adalah simbol *Activity*:



**Gambar 2.12** Simbol *Activity* dalam *Activity Diagram*

1. ***Swimlane***

*Swimlane* merupakan suatu simbol yang memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Berikut adalah simbol *Swimlane*:



**Gambar 2.13** Simbol *Swimlane* dalam *Activity Diagram*

1. ***Sequence*** ***Diagram***

*Sequence diagram* adalah suatu diagram yang menggambarkan interaksi antar objek dan mengindikasikan komunikasi diantara objek-objek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh objek-objek yang melakukan suatu tugas atau aksi tertentu.

*Sequence* *diagram* mendeskripsikan bagaimana entitas dalam system berinteraksi, termasuk pesan yang digunakan saat interaksi. Sequence diagram berhubungan erat dengan Use Case diagram, dimana 1 Use Case akan menjadi 1 Sequence Diagram.

Berikut simbol-simbol yang digunakan dalam *Sequence* *Diagram*.

1. ***Actor***

*Actor* merupakan suatu simbol yang menunjukkan pelaku yang berinteraksi dengan sistem

Berikut adalah simbol *Actor*:



**Gambar 2.14** Simbol *Actor* dalam *Sequence Diagram*

1. ***Object***

*Object* atau yang biasa disebut partisipan merupakan *instance* dari sebuah *class*.

Berikut adalah *Object* simbol:



**Gambar 2.15** Simbol *Object* dalam *Sequence Diagram*

1. ***Lifeline***

*Lifeline* merupakan suatu simbol yang menunjukkan waktu mulai dan berakhirnya pesan. Mengindikasikan keberadaan sebuah objek dalam basis waktu.

Berikut adalah simbol *Lifeline*:



**Gambar 2.16** Simbol *Lifeline* dalam *Sequence Diagram*

1. ***Activation***

*Activation* merupakan suatu simbol yang berfungsi untuk mengindikasikan sebuah objek yang melakukan sebuah aksi.

Berikut adalah simbol *Activation*:

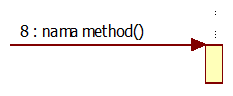


**Gambar 2.17** Simbol *Activation* dalam *Sequence Diagram*

1. ***Call Message***

*Call* *Message* merupakan suatu simbol yang menyatakan suatu objek atau memanggil operasi / *method* yang ada pada objek lain atau dirinya sendiri.

Berikut adalah simbol *Call* *Message*:

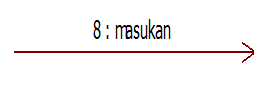


**Gambar 2.18** Simbol *Call* *Message* dalam *Sequence Diagram*

1. ***Send Message***

*Send Message* merupakan suatu simbol yang menyatakanbahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah kepada objek yang dikirim

Berikut adalah simbol *Send Message*:



**Gambar 2.19** Simbol *Send Message* dalam *Sequence Diagram*

1. ***Return Message***

*Return* *Message* merupakan suatu simbol yang menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau *method* menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian

Berikut adalah simbol *Return* *Message*:

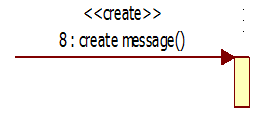


**Gambar 2.20** Simbol *Return* *Message* dalam *Sequence Diagram*

1. ***Create Message***

*Create Message* merupakan suatu simbol yang menyatakan suatu objek dapat membuat objek lain, arah panah mengarah pada objek yang dibuat.

Berikut adalah simbol *Create Message*:

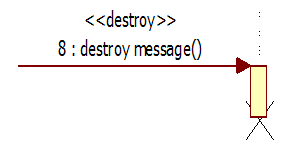


**Gambar 2.21** Simbol *Create Message* dalam *Sequence Diagram*

1. ***Destroy Message***

*Destroy Message* merupakan suatu simbol yang menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri.

Berikut adalah simbol *Destroy Message*:

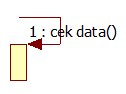


**Gambar 2.22** Simbol *Destroy Message* dalam *Sequence Diagram*

1. ***Self- Message***

*Self- Message* merupakan simbol yang menunjukkan komunikasi ke dalam sebuah objek itu sendiri.

Berikut adalah simbol:



**Gambar 2.23** Simbol *Self- Message* dalam *Sequence Diagram*

1. ***Class*** ***Diagram***

Diagram kelas atau *class diagram* menunjukkan interaksi antar kelas di dalam sistem. Diagram kelas mengandung informasi dan tingkah-laku segala sesuatu yang berkaitan dengan informasi tersebut. Berikut kegunaan diagram kelas.

1. Mengelompokkan obyek-obyek menjadi kelas – kelas berarti mengabstraksikan masalah yang sedang dihadapi.
2. Definisi-definisi common (seperti nama kelas dan atribut) cukup disimpan sekali per instan kelas (objek).

Berikut simbol-simbol yang digunakan dalam diagram kelas.

1. ***Association***

*Association* merupakan suatu simbol yang menghubungkan objek satu dengan objek yang lainnya.

Berikut adalah simbol *Association*:



**Gambar 2.24** Simbol *Association* dalam *Class Diagram*

1. ***Aggregation***

*Aggregation* merupkan suatu simbol yang menunjukkan hubungan kelas induk dan anak, kelas anak masih bisa berdiri sendiri

Berikut adalah simbol *Aggregation*:



**Gambar 2.25** Simbol *Aggregation* dalam *Class Diagram*

1. ***Composition***

*Composition* merupakan suatu simbol yang menunjukkan hubungan kelas induk dan anak, kelas anak tidak bisa berdiri sendiri.

Berikut adalah simbol *Composition*:



**Gambar 2.26** Simbol *Composition* dalam *Class Diagram*

1. ***Dependency***

*Dependency* merupakan suatu simbol yang menunjukkan suatu kelas yang bergantung pada kelas lainnya, perubahan pada kelas yang berdiri sendiri dapat mempengaruhi kelas yang bergantung padanya.

Berikut adalah simbol *Dependency*:



**Gambar 2.27** Simbol *Dependency* dalam *Class Diagram*

1. ***Realization***

*Realization* merupakan suatu simbol yang menunjukkan realisasi dari suatu kelas, digunakan untuk menunjukkan kelas implement dari suatu interface.

Berikut adalah simbol *Realization*:



**Gambar 2.28** Simbol *Realization* dalam *Class Diagram*

1. ***Class***

*Class* merupakan himpunan objek-objek yang berbagi atribut serta operasi yang sama.

Berikut adalah simbol *Class*:



**Gambar 2.29** Simbol *Class* dalam *Class* *Diagram*

1. **Perancangan Basis Data**
2. **Pengertian Basis Data**

Fathansah mengemukakan bahwa Basis data dapat didefinisikan dalam beberapa sudut pandang seperti:

1. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
2. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
3. Kumpulan file / tabel / arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.
4. **Sistem Basis Data**

Menurut Fathansah (1999:9-10), sistem basis data adalah sebagai berikut,

“Sistem basis data merupakan sistem yang terdiri atas kumpulan *file* (tabel) yang saling berhubungan (dalam sebuah basis data disebuah komputer) dan sekumpulan program (*DBMS/Database Management System*) yang memungkinkan beberapa pemakai dan / atau program lain untuk mengakses dan memanipulasi *file-file* (tabel-tabel) tersebut”.

Basis Data biasanya terdiri dari beberapa komponen yaitu :

1. Perangkat Keras ( *Hardware* ).
2. Sistem Operasi ( *Operating System* ).
3. Basis Data ( *Database* ).
4. **Relasional Basis Data**

Relasional Basis Data menurut Budi Sutedjo (2002 : 108),

“Relasional basis data berisi kumpulan tabel, dimana setiap tabel mempunyai nama dan struktur yang unik. Dalam setiap tabel, masing-masing *record* data diorganisasikan dalam struktur yang sama dan memiliki kunci yang akan menjadi penghubung antar tabel yang ada dan terkait satu sama lain”.

Dalam Model Relasional, basis data digambarkan dalam bentuk tabel-tabel dua dimensi yang memiliki baris dan kolom yang dihubungkan dengan relasi-relasi. Implementasi paling banyak dari model ini ialah pada definisi data SQL dan bahasa query.

1. **Ruang Lingkup Database SQL Server**

Microsoft SQL Server adalah Desain sistem manajemen basis data relational (RDBMS) untuk menyimpan data yang diorganisasikan sebagai entitas dengan atribut sederhana dan kompleks serta perilaku bisnis. Bahasa query utamanya adalah Transact-SQL yang merupakan implementasi dari SQL standar ANSI/ISO yang digunakan oleh microsoft dan Sybase.

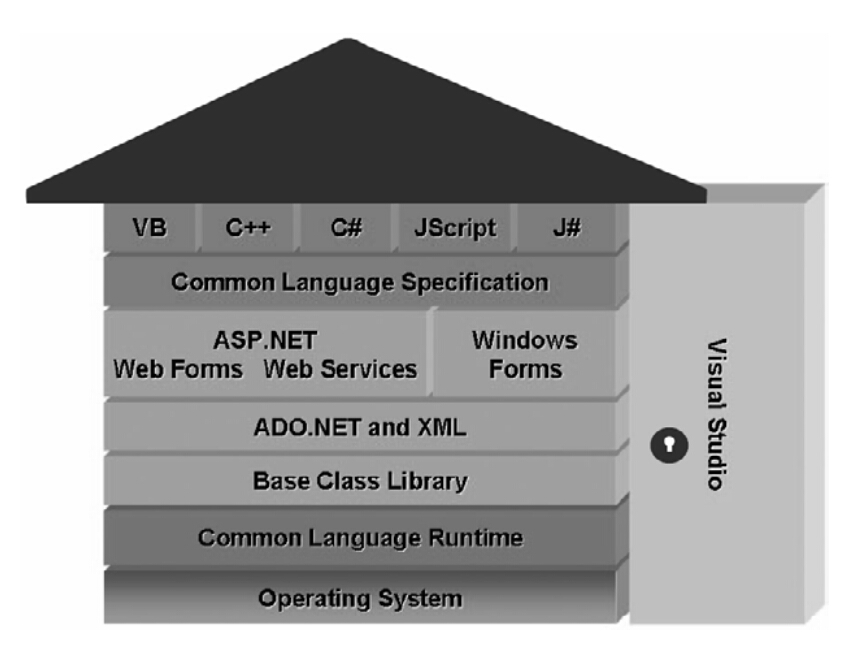
Microsoft SQL Server dan Sybase/ASE dapat berkomunikasi lewat jaringan dengan menggunakan protokol TDS (*Tabular* *Data* *Stream*). Selain dari itu, Microsoft SQL Server juga mendukung ODBC (*Open* *Database* *Connectivity*). Fitur yang lain dari SQL Server ini adalah kemampuan untuk membuat basis data *morroring* dan *clustering*.

1. **Ruang Lingkup Pemrograman**
2. **Microsoft .NET Framework**

Framework .Net adalah sebuah platform yang memungkinkan untuk mengembangkan berbagai aplikasi dan library. Platform ini menyediakan compiler dan tools yang memungkinkan untuk membangun, men-debug, dan mengeksekusi aplikasi tersebut. Framework .Net merupakan platform yang menyediakan semua yang dibutuhkan untuk mengembangkan aplikasi – aplikasi yang berjalan di Windows.

Framework .Net mengatur eksekusi aplikasi dengan menyediakan runtime environment yang bisa dikontrol, memiliki banyak layanan, pengaturan memori, dan memonitor serta mengatur keamanan saat aplikasi dieksekusi. Sebelum adanya .Net, aplikasi tidak dieksekusi oleh runtime environment. Aplikasi harus mengatur semua layanannya sendiri, yang sering menyebabkan kode error, masalah keamanan dan kerusakan data. Karena itu, aplikasi menjadi sulit untuk di-debug dan diperbaiki.

Framework .Net menyediakan banyak tools, seperti compilers, debuggers, bahasa-bahasa pemrograman, mesin eksekusi, tools untuk para pengembang dan banyak library. Kumpulan library tersebut dinamakan FCL (*Framework Class Libraries*). Komponen – komponen .Net dapat digambarkan seperti blok – blok bangunan penyusun rumah dalam gambar berikut.



**Gambar 2.30** Komponen – komponen pembentuk .Net

Common Language Runtime (CLR) merupakan komponen dasar pada Framework .Net. CLR mengelola kode yang dieksekusi dan menyediakan abstraksi lapisan antara kode dan sistem operasi. CLR sangat penting bagi fondasi karena merupakan mesin yang menjalankan dan mengatur proses eksekusi kode program.

1. **ASP .Net MVC**

ASP .Net MVC pada dasarnya adalah sebuah framework pengembangan web dari Microsoft, yang menggabungkan fitur arsitektur MVC (Model-View-Controller), ide dan teknik terbaru dari Agile development, dan bagian terbaik yang ada pada ASP .Net platform.

ASP .Net 1.0 dirilis pada tanggal 5 Januari 2002, sebagai bagian dari .Net Framework versi 1.0. pada saat itu, ASP .Net dan Web Forms dianggap sebagai satu kesatuan yang sama. Namun ASP .Net selalu mendukung dua lapisan abstraksi:

* System.Web.UI: lapisan formulir web, terdiri dari server controls, ViewState, dan sebagainya.
* System.Web: menyediakan the basic web stack, termasuk modules, handlers, the HTTP stack, dan lain-lain.

**2.6.2.1 Sejarah ASP .Net MVC**

Saat ASP .Net MVC diumumkan pada tahun 2007, pola MVC menjadi salah satu cara yang paling popular untuk membangun kerangka web.

Pada April 2009 source code ASP .Net MVC dirilis dibawah Microsoft Public License (MS-PL). ASP .Net MVC Framework merupakan framework yang ringan dan sangat dapat diuji yang terintegrasi dengan fitur ASP .Net yang ada.

Beberapa fitur yang terintegrasi ini adalah master pages dan membership-based authentication. MVC framework didefinisikan System.Web.Mvc assembly.

Pada bulan Maret 2012, Microsoft telah merilis sebagian dari Web stack-nya (termasuk ASP .Net MVC, Razor dan Web API) di bawah lisensi open source (Apache License 2.0). ASP .Net Web Forms tidak disertakan dalam inisiatif ini.

1. **Razor**

Razor adalah sintaks markup yang memungkinkan untuk menanamkan kode berbasis server ke halaman web menggunakan C# dan VB .Net. Razor bukan merupakan bahasa pemrograman, melainkan bahasa markup dari sisi server.

Razor tidak memiliki hubungan dengan ASP .Net karena Razor adalah mesin templating tujuan umum. Razor bisa digunakan dimana saja untuk menghasilkan output seperti HTML. Hanya saja ASP .Net telah menerapkan view engine yang memungkinkan menggunakan Razor dalam aplikasi MVC untuk menghasilkan HTML.

1. **Javascript**

Javascript adalah bahasa pemrograman komputer yang dinamis. Javascript bersifat ringan dan paling umum digunakan sebagai bagian dari halaman web, yang implementasinya memungkinkan client-side script untuk berinteraksi dengan pengguna dan membuat halaman dinamis. Javascript adalah bahasa pemrograman yang ditafsirkan dengan kemampuan berorientasi objek.

Javascript pertama kali dikenal sebagai LiveScript, namun Netscape mengubah namanya menjadi JavaScript. JavaScript membuat penampilan pertamanya di Netscape 2.0 pada tahun 1995 dengan nama LiveScript.

1. **Microsoft Visual Studio**

Microsoft Visual Studio merupakan sebuah [perangkat lunak](https://id.wikipedia.org/wiki/Perangkat_lunak) lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup [kompiler](https://id.wikipedia.org/wiki/Kompiler), [SDK](https://id.wikipedia.org/w/index.php?title=Software_Development_Kit&action=edit&redlink=1), [Integrated Development Environment (IDE)](https://id.wikipedia.org/wiki/Integrated_Development_Environment), dan dokumentasi (umumnya berupa [MSDN Library](https://id.wikipedia.org/w/index.php?title=MSDN_Library&action=edit&redlink=1)). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain [Visual C++](https://id.wikipedia.org/wiki/Visual_C%2B%2B), [Visual C#](https://id.wikipedia.org/w/index.php?title=Visual_C&action=edit&redlink=1), [Visual Basic](https://id.wikipedia.org/wiki/Visual_Basic), [Visual Basic .NET](https://id.wikipedia.org/wiki/Visual_Basic_.NET), [Visual InterDev](https://id.wikipedia.org/w/index.php?title=Visual_InterDev&action=edit&redlink=1), [Visual J++](https://id.wikipedia.org/w/index.php?title=Visual_J%2B%2B&action=edit&redlink=1), [Visual J#](https://id.wikipedia.org/w/index.php?title=Visual_J&action=edit&redlink=1), [Visual FoxPro](https://id.wikipedia.org/wiki/Visual_FoxPro), dan [Visual SourceSafe](https://id.wikipedia.org/w/index.php?title=Visual_SourceSafe&action=edit&redlink=1).

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam bentuk [Microsoft Intermediate Language](https://id.wikipedia.org/wiki/Microsoft_Intermediate_Language) di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi [Silverlight](https://id.wikipedia.org/wiki/Silverlight), aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).

1. **IIS (Internet Information Server)**

Aplikasi IIS (Internet Information Server) adalah aplikasi Visual Basic yang tinggal di server Web dan merespon permintaan dari browser. Aplikasi IIS menggunakan HTML untuk menampilkan antarmuka pengguna dan menggunakan kode Visual Basic yang dikompilasi untuk memproses permintaan dan merespon kejadian di browser.

Aplikasi IIS ditujukan untuk pengembang Visual Basic yang membangun aplikasi berbasis Web, bukan halaman Web. Aplikasi IIS memungkinkan pemrosesan bisnis yang rumit dan mudah dari hampir semua browser atau platform.