BAB II

LANDASAN TEORI

2.1. Sistem Pendukung Keputusan (SPK)

Turban (1990) dan Turban & Aronson (2001) menyebutkan bahwa konsep Sistem Pendukung Keputusan (SPK) muncul pertama kali pada awal tahun 1970-an oleh Scott-Morton. Mereka mendefinisikan SPK sebagai suatu sistem interaksi berbasis komputer yang dapat membantu para pengambil keputusan dalam menggunakan data dan model untuk memecahkan persoalan yang bersifat tidak terstruktur[8].

Dari definisi tersebut, dapat diindikasikan empat karakteristik utama dari SPK, yaitu:

1. SPK menggabungkan data dan model menjadi satu bagian.
2. SPK dirancang untuk membantu para manajer (pengambil keputusan) dalam proses pengambilan keputusan dari masalah yang bersifat semi struktural (atau tidak terstruktur).
3. SPK lebih cenderung dipandang sebagai penunjang penilaian manajer dan sama sekali bukan untuk menggantikannya.
4. Teknik SPK dikembangkan untuk meningkatkan efektivitas dari pengambil keputusan.

Definisi lain dari SPK menurut Minch dan Burns dalam Eriyatno (1998) adalah konsep spesifik sistem yang menghubungkan komputerisasi informasi dengan para pengambil keputusan sebagai pemakainya. Karakteristik pokok yang melandasi teknik SPK adalah:

a. Interaksi langsung antara komputer dengan pengambil keputusan.
b. Dukungan menyeluruh dari keputusan bertahap ganda.
c. Suatu sintesa dari konsep yang diambil dari berbagai bidang, antara lain ilmu komputer, psikologi, intelegensia buatan, ilmu sistem dan ilmu manajemen.
d. Mempunyai kemampuan adaptif terhadap perubahan kondisi dan kemampuan berevolusi menuju sistem yang lebih bermanfaat.

Pengembangan teknik pendukung keputusan melalui sistem ini ditujukan untuk membantu manajer pada proses pengambilan keputusan yang umumnya bersifat semi struktural. SPK digunakan sebagai penunjang penilaian manajer dan sama sekali bukan untuk menggantikannya. Teknik SPK dikembangkan untuk meningkatkan efektivitas dari pengambil keputusan. Efektivitas mencakup identifikasi dari apa yang harus dilakukan dan menjamin bahwa kriteria yang dipilih relevan dengan tujuan.

2.2. *Analytic Hierarchy Process* (AHP)

Proses Hirarki Analitik (*Analytic Hierarchy Process* – AHP) dikembangkan oleh Dr. Thomas L. Saaty dari Wharton School of Business pada tahun 1970-an untuk mengorganisasikan informasi dan *judgement* dalam memilih alternatif yang paling disukai (Saaty, 1983). Dengan menggunakan AHP, suatu persoalan yang akan dipecahkan dalam suatu kerangka berpikir yang terorganisir, sehingga memungkinkan dapat diekspresikan untuk mengambil keputusan yang efektif atas persoalan

tersebut. Persoalan yang kompleks dapat disederhanakan dan dipercepat proses pengambilan keputusannya[8]. (Marimin, 2004)

Prinsip kerja AHP adalah penyederhanaan suatu persoalan kompleks yang tidak terstruktur, stratejik, dan dinamik menjadi bagian-bagiannya, serta menata dalam suatu hirarki. Kemudian tingkat kepentingan setiap variabel diberi nilai numerik secara subjektif tentang arti penting variabel tersebut secara relatif dibandingkan dengan variabel yang lain. Dari berbagai pertimbangan tersebut kemudian dilakukan sintesa untuk menetapkan variabel yang memiliki prioritas tinggi dan berperan untuk mempengaruhi hasil pada sistem tersebut.

Secara grafis, persoalan keputusan AHP dapat dikonstruksikan sebagai diagram bertingkat, yang dimulai dengan goal/sasaran, lalu kriteria level pertama, subkriteria dan akhirnya alternatif.

AHP memungkinkan pengguna untuk memberikan nilai bobot relatif dari suatu kriteria majemuk (atau alternatif majemuk terhadap suatu kriteria) secara intuitif, yaitu dengan melakukan perbandingan berpasangan (*pairwise comparisons*). Dr. Thomas L. Saaty, pembuat AHP, kemudian menentukan cara yang konsisten untuk mengubah perbandingan berpasangan/*pairwise*, menjadi suatu himpunan bilangan yang merepresentasikan prioritas relatif dari setiap kriteria dan alternatif.

Contoh soal penerapan AHP[11].

Ada 4 faktor pemilihan pekerjaan, yaitu : lokasi, prospek, resiko, dan gaji. Nilai perbandingan berpasangan dibuat sebagai berikut :

$$\begin{array}{r} \text{Lokasi} \\ \text{Lokasi} \\ \text{Prospek} \\ \text{Resiko} \\ \text{Gaji} \end{array} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 5 \end{bmatrix} \Rightarrow \text{konsisten} \Rightarrow \begin{array}{c} \\ \text{L} \\ \text{P} \\ \text{R} \\ \text{G} \end{array} \begin{array}{c} \begin{array}{cccc} \text{L} & \text{P} & \text{R} & \text{G} \end{array} \\ \begin{bmatrix} 1 & 1/2 & 1/3 & 1/5 \\ 2 & 1 & 1/3 & 1/4 \\ 3 & 3 & 1 & 1/2 \\ 5 & 4 & 2 & 1 \end{bmatrix} \end{array}$$

Apabila A adalah matriks perbandingan berpasangan yang, maka vektor bobot yang berbentuk :

$$(A)(w^T) = (n)(w^T)$$

dapat didekati dengan cara:

- menormalkan setiap kolom j dalam matriks A, sedemikian hingga :

$$\sum_i a_{ij} = 1$$

sebut sebagai A'.

- untuk setiap baris i dalam A', hitunglah nilai rata-ratanya:

$$w_i = \frac{1}{n} \sum_j a'_{ij}$$

dengan w_i adalah bobot tujuan ke-i dari vektor bobot.

	L	P	R	G
L	1	1/2	1/3	1/5
P	2	1	1/3	1/4
R	3	3	1	1/2
G	5	4	2	1
Jml	11	8,5	3,67	1,95

Maka setelah dilakukan normalisasi, menjadi :

	L	P	R	G	Rata2
L	0,091	0,059	0,091	0,103	0,086
P	0,182	0,118	0,091	0,128	0,130
R	0,273	0,353	0,273	0,256	0,288
G	0,455	0,471	0,545	0,513	0,496
Jml	1	1	1	1	1

Kemudian nilai vektor bobot yang didapat adalah :

W = [0,086; 0,130; 0,288; 0,496]

Misalkan A adalah matriks perbandingan berpasangan, dan w adalah vektor bobot, maka konsistensi dari vektor bobot w dapat diuji sebagai berikut :

- Hitung : $(A)(w^T)$
- Hitung : $t = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{elemen\ ke{-}i\ pada\ (A)(w^T)}{elemen\ ke{-}i\ pada\ w^T}\right)$
- Hitung indeks konsistensi : $CI = \frac{t-n}{n-1}$
- Jika CI = 0 maka A konsisten; jika $\frac{CI}{RI_n} \leq 0{,}1$ maka A cukup konsisten; dan jika $\frac{CI}{RI_n} > 0{,}1$ maka A sangat tidak konsisten.

Indeks random RI_n adalah nilai rata-rata CI yang dipilih secara acak pada A dan diberikan sebagai :

N	2	3	4	5	6	7	...
RI_n	0	0,58	0,90	1,12	1,24	1,32	...

Pengujian terhadap konsistensi matriks A dilakukan sebagai berikut:

- $(A)(w^T) = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/5 \\ 2 & 1 & 1/3 & 1/4 \\ 3 & 3 & 1 & 1/2 \\ 5 & 4 & 2 & 1 \end{bmatrix}\begin{bmatrix} 0{,}086 \\ 0{,}130 \\ 0{,}288 \\ 0{,}496 \end{bmatrix} = \begin{bmatrix} 0{,}346 \\ 0{,}522 \\ 1{,}184 \\ 2{,}022 \end{bmatrix}$
- $t = \frac{1}{4}\left(\frac{0{,}346}{0{,}086} + \frac{0{,}522}{0{,}130} + \frac{1{,}184}{0{,}288} + \frac{2{,}022}{0{,}496}\right) = 4{,}057$
- $CI = \frac{4{,}057-4}{3} = 0{,}019$
- Untuk n = 4, diperoleh $RI_4 = 0{,}90$; sehingga :

  $\frac{CI}{RI_n} = \frac{0{,}019}{0{,}900} = 0{,}021 \leq 0{,}1$; maka matriks A dikatakan cukup konsisten.

Misalkan ada n tujuan dan m alternatif pada AHP, maka proses perankingan alternatif dapat dilakukan melalui langkah-langkah sebagai berikut :

i. Untuk setiap tujuan i, tetapkan matriks perbandingan berpasangan A, untuk m alternatif.

ii. Tentukan vektor bobot untuk setiap A_i yang merepresentasikan bobot relatif dari setiap alternatif ke-j pada tujuan ke-i (s_{ij}).

iii. Hitung total skor :

$s_j = \sum_i (s_{ij})(w_i)$

iv. Pilih alternatif dengan skor tertinggi.

Misalkan ada 4 alternatif pekerjaan yang akan dipilih, yaitu A, B, C, dan D. Pada tujuan pertama **(Lokasi yang baik)**, matriks perbandingan berpasangan yang ditetapkan adalah :

	A	B	C	D
A	1	1/2	5	1/5
B	2	1	7	1/2
C	1/5	1/7	1	1/9
D	3	2	9	1
Jml	$6\frac{1}{5}$	$3\frac{9}{14}$	22	$1\frac{17}{18}$

Setelah dilakukan normalisasi :

	A	B	C	D	Rata2
A	0,161	0,137	0,227	0,171	0,174
B	0,322	0,275	0,312	0,257	0,293
C	0,320	0,040	0,045	0,057	0,044
D	0,484	0,549	0,409	0,514	0,489

Sehingga : s_{11} = 0,174; s_{12} = 0,293; s_{13} = 0,044; dan s_{14} = 0,489.

Pada tujuan kedua **(Prospek yang baik)**, matriks perbandingan berpasangan yang ditetapkan adalah :

	A	B	C	D
A	1	9	5	2
B	1/9	1	1/9	1/9
C	1/5	9	1	1/2
D	1/2	9	2	1
Jml	$1\frac{73}{90}$	28	$8\frac{1}{9}$	$3\frac{11}{18}$

Setelah dilakukan normalisasi :

	A	B	C	D	Rata2
A	0,552	0,321	0,616	0,554	0,511
B	0,061	0,036	0,014	0,031	0,035
C	0,110	0,321	0,123	0,138	0,173
D	0,276	0,321	0,217	0,277	0,280

Sehingga : s_{21} = 0,511; s_{22} = 0,035; s_{23} = 0,173; dan s_{24} = 0,280.

Pada tujuan ketiga **(Resiko yang lebih ringan)**, matriks perbandingan berpasangan yang ditetapkan adalah :

	A	B	C	D
A	1	6	1/2	1/2
B	1/6	1	1/6	1/8
C	2	6	1	2
D	2	8	1/2	1
Jml	$5\frac{1}{6}$	21	$2\frac{1}{6}$	$3\frac{5}{8}$

Setelah dilakukan normalisasi :

	A	B	C	D	Rata2
A	0,194	0,286	0,231	0,138	0,212
B	0,032	0,048	0,077	0,034	0,048
C	0,387	0,286	0,462	0,552	0,422
D	0,387	0,381	0,231	0,276	0,319

Sehingga : s_{31} = 0,212; s_{32} = 0,048; s_{33} = 0,422; dan s_{34} = 0,319.

Pada tujuan keempat **(Gaji yang lebih tinggi)**, matriks perbandingan berpasangan yang ditetapkan adalah :

	A	B	C	D
A	1	1/9	1/4	1/6
B	9	1	2	1
C	4	1/2	1	1/2
D	6	1	2	1
Jml	20	$2\frac{11}{18}$	$5\frac{1}{4}$	$2\frac{2}{3}$

Setelah dilakukan normalisasi :

	A	B	C	D	Rata2
A	0,050	0,043	0,048	0,063	0,051
B	0,450	0,383	0,312	0,375	0,397
C	0,200	0,191	0,190	0,188	0,192
D	0,300	0,383	0,381	0,375	0,360

Sehingga : s_{41} = 0,051; s_{42} = 0,397; s_{43} = 0,192; dan s_{44} = 0,360.

Matriks skor setiap alternatif pada semua tujuan adalah :

	A	B	C	D
L	0,174	0,293	0,044	0,489
P	0,511	0,035	0,173	0,280
R	0,212	0,048	0,422	0,319
G	0,051	0,397	0,192	0,360

Vektor bobot yang telah diperoleh sebelumnya :

W = [0,086; 0,130; 0,288; 0,496]

Skor total setiap alternatif adalah :

s_1 = (0,174)(0,860) + (0,511)(0,130) + (0,212)(0,288) + (0,051)(0,496)

= 0,168

s_2 = (0,293)(0,860) + (0,035)(0,130) + (0,048)(0,288) + (0,397)(0,496)

= 0,240

s_3 = (0,044)(0,860) + (0,173)(0,130) + (0,422)(0,288) + (0,192)(0,496)

= 0,243

s_4 = (0,489)(0,860) + (0,280)(0,130) + (0,319)(0,288) + (0,360)(0,496)

= 0,349

Karena skor total alternatif D (s_4) paling besar, maka alternatif D yang paling dipilih.

2.3. Peminatan

Peminatan berasal dari kata minat yang berarti kecenderungan atau keinginan yang cukup kuat berkembang pada diri individu yang terarah dan terfokus pada terwujudkannya suatu kondisi dengan mempertimbangkan kemampauan dasar, bakat, minat, dan kecenderungan pribadi individu[2].

2.4. Program Studi

Program studi adalah kesatuan rencana belajar yang diselenggarakan atas dasar suatu kurikulum dan ditujukan agar mahasiswa dapat menguasai pengetahuan, keterampilan, dan sikap yang sesuai dengan sasaran kurikulum[5]. (Departemen Pendidikan Nasional, 2000)

2.5. Perancangan

Menurut John Burch dan Gary Grudnitski, perancangan adalah penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah dari satu kesatuan yang utuh dan berfungsi[7].

2.6. *Passing Grade*

Passing grade didefinisikan sebagai nilai minimum yang harus didapatkan untuk bisa lulus ujian. Nilai kelulusan untuk program ujian ditentukan melalui proses pengaturan standar. Secara resmi memang perguruan tinggi tidak memberikan batasan minimal nilai yang harus dicapai untuk menentukan kelulusan seorang calon mahasiswa di sebuah perguruan tinggi. Akan tetapi sebuah perkiraan *passing grade* masih tetap dibutuhkan agar seorang cama bisa memperkirakan capaian skor yang harus diraih agar lolos PTN via SBMPTN[10].

2.7. Aplikasi Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance,* konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia[9]. (Nazruddin Safaat, 2014)

Aplikasi Android ditulis dalam bahasa pemrograman java. Kode java dikompilasi bersama dengan data *file resource* yang dibutuhkan oleh aplikasi, di mana prosesnya di-*package* oleh tools yang disebut “apt tools” ke dalam paket Android sehingga menghasilkan file dengan ekstensi apk. File

apk itulah yang disebut dengan aplikasi, dan nantinya dapat di install di perangkat *mobile*.

2.8. *Web Service*

Web service adalah komponen perangkat lunak disimpan pada suatu komputer yang dapat diakses oleh aplikasi (atau komponen perangkat lunak lainnya) di komputer lain melalui jaringan. *Web service* berkomunikasi menggunakan teknologi seperti XML, JSON, dan HTTP[4].

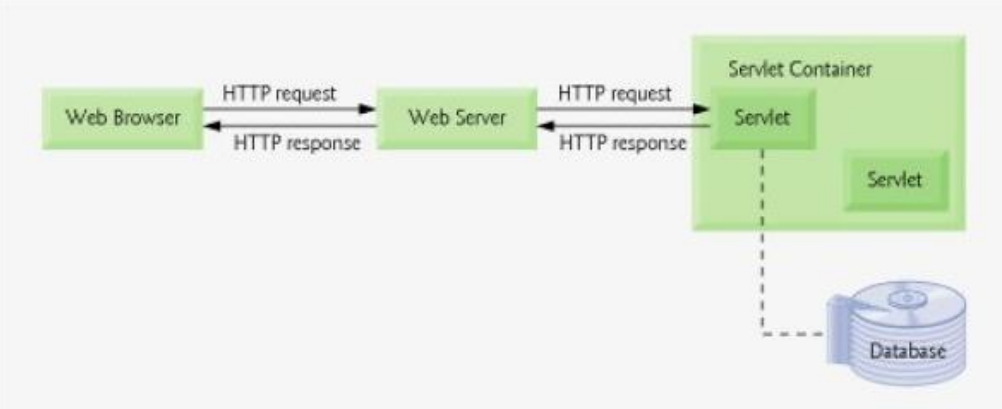
Web service digunakan sebagai suatu fasilitas yang disediakan oleh suatu website untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (*service*) yang disediakan oleh suatu sistem yang menyediakan *web service*.

2.9. Java Servlet

Servlet adalah sebuah class dalam bahasa pemgrograman Java yang digunakan untuk meningkatkan kapabilitas dari server sebagai host dari aplikasi yang diakses melalui *request-response programming* model (diadaptasi dari tutorial J2EE). Servlet adalah sebuah class java yang mengimplementasi *interface* Servlet dan menerima *request* yang berasal dari class Java, *web client*, atau servlet lain yang membangkitkan *response*. "Servlet" juga dipanggil sebagai HTTP Servlet. Hal ini disebakan karena servlets biasanya digunakan dengan HTTP, akan tetap servlet bukanlah merupakan salah satu spesifikasi spesifik dari protokol *client-server*.

Servlets dan JSP telah menjadi sangat populer sehingga mereka sekarang didukung langsung, atau dengan pihak ketiga *plug-in* oleh sebagian besar server web utama dan server aplikasi yang mengeksekusi aplikasi untuk menghasilkan halaman web dinamis dalam menanggapi permintaan. Beberapa server web populer dan server aplikasi termasuk Sun Java System Application Server[3].

Servlets menunjukkan komunikasi antara klien dan server melalui protokol http. Klien mengirimkan permintaan http ke server. Servlet container menerima permintaan dan mengarahkan untuk diproses oleh servlet yang sesuai. Servlet melakukan pengolahan, yang mungkin termasuk berinteraksi dengan *database* atau komponen server lain, seperti servlet atau JSP lainnya. Servlet mengembalikan hasilnya kepada klien di *form* dari HTML, XHTML atau XML dokumen untuk menampilkan di browser. Format data lain, seperti yang gambar dan data biner, juga bisa kembali lagi.



Gambar 2.1. Arsitektur Servlet

2.9.1. Siklus Hidup Servlet

Siklus hidup Sebuah servlet dimulai ketika memuat servlet kontainer ke dalam memori normal, dalam tanggapan terhadap permintaan pertama servlet. Sebelum servlet dapat menangani permintaan itu, container memanggil method init servlet. Setelah init selesai eksekusi, servlet dapat merespon permintaan pertamanya. Semua permintaan ditangani oleh method *service* servlet, yang menerima permintaan tersebut, memprosesnya dan mengirim respon ke klien. Selama siklus hidup servlet, method *service* disebut sekali per permintaan. Setiap permintaan baru biasanya ditangani di thread eksekusi yang terpisah (dikelola oleh container servlet) di mana method service mengeksekusi. Ketika container servlet berakhir (misalnya, ketika container servlet membutuhkan lebih banyak memori atau ketika dimatikan), servlet destroy method dipanggil untuk melepaskan sumber daya servlet.

Paket servlet mendefinisikan dua kelas abstrak yang mengimplementasikan antarmuka *class* servlet GenericServlet (dari paket javax.servlet) dan *class* HttpServlet (dari paket javax.servlet.http). Kelas ini menyediakan implementasi default beberapa method servlet. Kebanyakan servlet memperpanjang baik GenericServlet atau HttpServlet dan menimpa beberapa atau semua method-nya. GenericServlet adalah protokol servlet independen, sementara HttpServlet menggunakan protokol HTTP untuk menukar informasi antara server dan klien. HttpServlet digunakan pada web.

Jika servlet perlu mengimplementasikan protokol selain HTTP, mereka dapat memperpanjang GenericServlet. Untuk memperpanjang GenericServlet, servlet harus mengesampingkan method service abstrak. Untuk memperpanjang HttpServlet, servlet harus menimpa setidaknya satu method yang dideklarasikan di kelas HttpServlet.

Method kunci dalam setiap servlet adalah *service*, yang menerima kedua objek ServletRequest dan objek ServletResponse. Kedua objek ini menyediakan akses ke input dan output stream yang memungkinkan servlet untuk membaca data dari dan mengirim data ke klien.

2.9.2. Http Servlet Class

Servlet secara khusus memperpanjang class HttpServlet, yang menimpa method *service* untuk membedakan antara permintaan diterima dari klien web browser. Dua jenis HTTP request yang paling umum (juga dikenal sebagai method request) adalah get dan post. Get request mendapat (atau memperoleh kembali) informasi dari server. Request tersebut sering mengambil sebuah dokumen HTML atau gambar. Sedangkat post request mem-*posting* (atau mengirimkan) data ke server, seperti informasi otentikasi atau data dari sebuah form yang mengumpulkan input pengguna. Biasanya, post request digunakan untuk mengirim pesan kepada kelompok berita atau forum diskusi, melalui input pengguna untuk

proses penanganan data dan menyimpan atau memperbarui data pada server.

Class HttpServlet mendefinisikan method doGet dan doPost untuk merespon permintaan get dan post dari klien. Method ini disebut oleh method service, yang dipanggil oleh container servlet ketika permintaan tiba di server. Method service pertama menentukan jenis permintaan, kemudian memanggil method yang tepat untuk menangani permintaan seperti itu.

Method doGet dan doPost menerima sebagai argumen objek HttpServletRequest dan objek HttpServletResponse yang memungkinkan interaksi antara klien dan server. Method HttpServletRequest memungkinkan akses ke data yang diberikan sebagai bagian dari permintaan. Method HttpServletResponse membuatnya mudah untuk menghasilkan servlet untuk klien web.

2.9.3. Http Servlet Request Interface

Setiap panggilan untuk doGet atau doPost untuk HttpServlet menerima sebuah objek yang mengimplementasikan antarmuka HttpServletRequest. Container servlet menciptakan objek HttpServletRequest dan melewati ke method service servlet. Objek ini berisi permintaan klien dan menyediakan method yang memungkinkan servlet untuk memproses permintaan tersebut. Beberapa method dari antarmuka ServletRequest memperpanjang antarmuka HttpServletRequest.

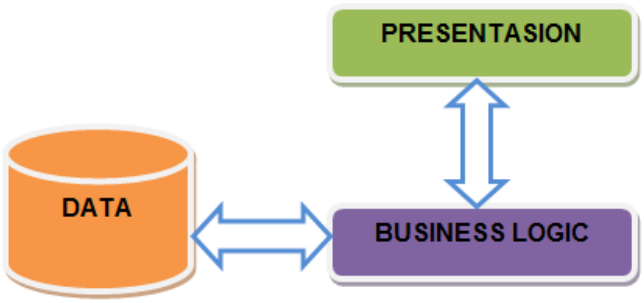
2.9.4. Http Servlet Response Interface

Setiap panggilan untuk doGet atau doPost untuk HttpServlet menerima sebuah objek yang mengimplementasikan antarmuka HttpServletResponse. Container servlet menciptakan objek HttpServletResponse dan melewati ke method service servlet. Objek ini menyediakan method yang memungkinkan servlet untuk merumuskan tanggapan untuk klien. Beberapa method dari antarmuka ServletRequest memperpanjang antarmuka HttpServletResponse.

2.10. iBATIS

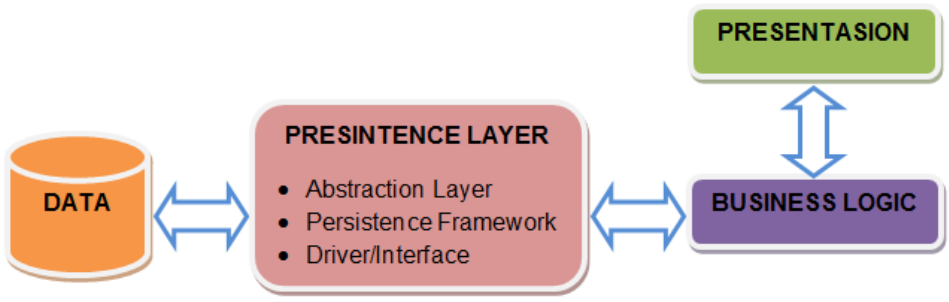
iBATIS merupakan sebuah *persintence framework* yang mengotomatisasi penjembatanan atau pemetaan (*mapping*) antara database SQL dan objek-objek di dalam Java[6].

Pada sebuah sistem aplikasi *client-server* dengan arsitektur *three-tier* dikenal tiga buah lapisan (*layer*) yang disebut sebagai *presentasion, business logic,* dan *data.*



Gambar 2.2. Arsitektur *Three-Tier*

Seiring dengan berkembangnya evolusi pemrograman dan ditemukannya teknik-teknik yang baru yang berdasarkan pengalaman diakui dapat meningkatkan efisiensi serta kemudahan maka diperkenalkan sebuah lapisan persintence yang berada di antara *busines logic* dan data sehingga yang semula kedua lapisan itu langsung berinteraksi maka sekarang tersedia jembatan di tengah-tengahnya yang mengatur komunikasi keduanya secara tidak langsung.



Gambar 2.3. Arsitektur dengan *Presintance Layer*

Perlu dipahami bahwa persistence layer adalah konsep yang umum pada pengembangan sebuah aplikasi dan tidak terkait secara spesifik dengan sebuah teknologi atau produk tertentu. Selain diaplikasikan pada lingkungan Java, *persistence layer* juga biasa diterapkan pada platform yang lain seperti .NET.

Sebuah *persistence layer* bekerja dengan hal-hal yang berhubungan dengan persisting yaitu penyimpanan (*storing*) dan pembacaan (*retrieving*) data ke/dari database. Tujuan yang paling mendasar dari keberadaan sebuah *persistence layer* adalah:

– Mengambil alih pekerjaan-pekerjaan yang berkaitan dengan persisting dari lapisan *business logic* (misalnya perintah SELECT, INSERT dan

UPDATE) sehingga *business logic* bisa sungguh-sungguh fokus dengan rangkaian-rangkaian program yang berhubungan dengan logika bisnis, aturan dan aliran program dari aplikasi yang bersangkutan tanpa perlu dipusingkan dengan detail dari perintah-perintah SQL dari database. Dengan demikian, *business logic* menjadi bebas dan tidak tergantung dengan tipe atau platform database apapun yang digunakan.

- Memisahkan secara jelas antara kode program dengan perintah-perintah SQL dengan cara mengeluarkan semua perintah operasi database dari *business logic*. Tujuannya adalah untuk menghindari program yang campur aduk antara kode program dari aplikasi dan perintah-perintah SQL. Hal ini sangat penting untuk kerapihan dan kemudahan dalam memelihara kode program. Dengan demikian embedded SQL atau inline SQL (perintah-perintah SQL yang berbaur dengan kode program) adalah sesuatu yang ingin dihindari oleh *persistence framework*.

Merujuk pada gambar 2 di atas, tampak bahwa lapisan persistence terbagi menjadi tiga bagian dengan masing-masing fungsi sebagai berikut :

1. Abstraction Layer, berfungsi sebagai antar muka bagi lapisan persistence agar lapisan *business logic* dapat berinteraksi dengannya.
2. Persistence Framework, dalam hal ini adalah iBATIS itu sendiri. Pada prakteknya iBATIS adalah sebuah library (file JAR) yang diasosiasikan dengan sebuah project Java sehingga class-class di dalam iBATIS dapat digunakan.

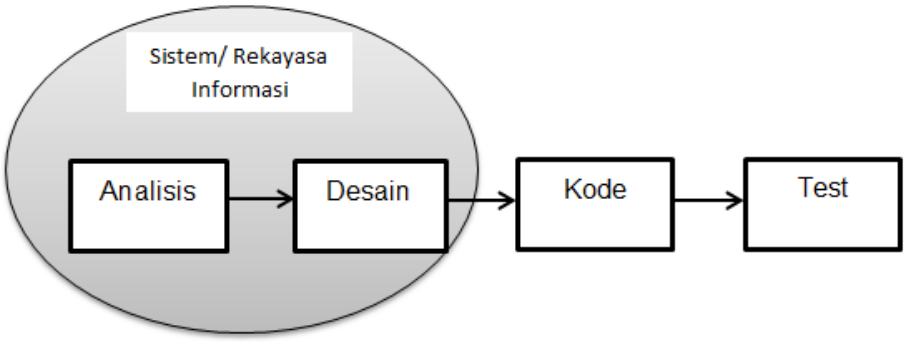
3. Driver/Interface, driver disini adalah antar muka untuk mengakses database. Driver database yang digunakan dalam pemrograman Java adalah JDBC.

iBATIS bukanlah sebuah perkakas O/RM melainkan sebuah perkakas Query Mapping karena iBATIS tidak memetakan struktur table dengan sebuah class secara langsung, namun melalui perintah-perintah SQL (SELECT, INSERT, UPDATE atau DELETE). Pemetaan pada iBATIS memiliki fleksibilitas tinggi, selain itu programmer memiliki kontrol lebih besar terhadap perintah-perintah SQL dan kebebasan lebih dalam kustomisasi.

2.11. Software Engineering Model *Waterfall*

Model SDLC air terjur (*waterfall*) sering juga disebut model sekuensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*) [1]. (Shalahuddin dan Rosa, 2013)

Berikut adalah gambar model air terjun:



Gambar 2.4. Ilustrasi model *waterfall* (Pressman, 2002)

1. Analisis kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh user. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Pembuatan kode program

   Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan

memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung (*support*) atan pemeliharaan (*maintenance*)

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke user. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.