

LAPORAN LENGKAP
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK



OLEH :
NAMA : LAODE MUHAMMAD ILHAM
SETIAWAN
NIM : F1G120006
KELOMPOK : DUA (2)

ASISTEN PENGAMPU :
WAHID SAFRI JAYANTO

PROGRAM STUDI S1 ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HALU OLEO
KENDARI
2021

HALAMAN PENGESAHAN
LAPORAN LENGKAP



OLEH :
LA ODE MUHAMMAD ILHAM SETIAWAN (F1G12006)

Menerangkan bahwa apa yang tertulis dalam laporan lengkap ini adalah benar dan dinyatakan telah memenuhi syarat.

Kendari, Desember 2021

Menyetujui

ASISITEN

A
WAHID SAFRI JAYANTO
F1G117059

PRAKTIKAN

M. ILHAM SETIAWAN
F1G120006

20-12-2021

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Puji syukur kehadirat Allah *subhaanallahu wata'ala*, karena berkat taufik dan hidayah-Nya, sehingga penulis dapat menyelesaikan laporan lengkap praktikum Pemograman Berorientasi Objek dalam rangka memenuhi salah satu syarat kelulusan dan untuk mengikuti ujian praktikum Pemograman Berorientasi Objek.

Dalam penyusunan laporan lengkap ini, penulis banyak memperoleh bimbingan, masukan dan dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis mengucapkan terima kasih khususnya kepada dosen mata kuliah Pemograman Berorientasi Objek, **Bambang Pranomo**, S.Si., MT. yang telah memberikan dasar teori yang berguna selama praktikum ini serta kepada asisten Praktikum Pemograman Berorientasi Objek, **Wahid Safri Jayanto** atas arahan dan koreksi kepada penulis sehingga laporan lengkap ini dapat selesai seperti yang diharapkan. Penulis tak lupa pula mengucapkan terimakasih kepada teman-teman yang telah berbagi suka duka selama praktikum berlangsung serta semua pihak yang telah memberikan bantuan dan motivasinya kepada penulis hingga laporan lengkap ini dapat selesai.

Penulis menyadari bahwa laporan lengkap praktikum ini masih jauh dari kesempurnaan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran dari para pembaca demi kesempurnaan laporan lengkap ini.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Kendari, Desember 2021

DAFTAR ISI

LAPORAN LENGKAP	PRAKTIKUM PEMROGRAMAN
BERORIENTASI OBJEK	i
HALAMAN PENGESAHAN LAPORAN LENGKAP	ii
KATA PENGANTAR.....	iii
DAFTAR ISI.....	iv
DAFTAR TABEL	viii
DAFTAR GAMBAR	ix
DAFTAR PROGRAM	x
PRAKTIKUM 1	1
1.1. Alat dan Bahan	1
1.2. Pengenalan Pemograman Berorientasi Objek (PBO).....	1
1.3. Prinsip OOP.....	3
1.4. Pengenalan Mengenai PHP	4
PRAKTIKUM 2	7
2.1. Pengenalan <i>Class, Object, Property</i> dan <i>Method</i>	7
2.1.1 Class OOP.....	7
2.1.2 Property	7
2.1.3 Method.....	8

2.1.4 Object.....	9
2.2. Enkapsuli (<i>Encapsulation</i>)	10
2.2.1 Public	11
2.2.2 Private	12
2.2.3 Protected	12
2.3. <i>Constructor</i> dan <i>Destructor</i>	12
2.3.1 Constructor	12
2.3.2 Descrtor.....	13
2.4. <i>Interfaces</i>	14
2.5. <i>Laravel</i>	15
2.6. <i>Composer</i>	19
PRAKTIKUM 3	21
3.1. Model Data Bebasis Objek.....	21
3.1.1 Entity Relationship model	21
3.1.2 Binary Model.....	21
3.1.3 Semantic Model	21
3.2. Model Data Berbasis <i>Record</i>	22
3.2.1 Model Data Hirarki.....	22
3.2.2 Model Data Jaringan.....	23
3.2.3 Model Data Rasional	24

3.3. Komponen <i>ERD</i>	25
3.3.1 Entity	25
3.3.2 Atribut.....	26
3.3.3 Relasi	27
3.4. Pengenalan <i>CRUD</i>	27
3.4.1 Create.....	28
3.4.2 Read.....	28
3.4.3 Update.....	28
3.4.4 Delete.....	29
PRAKTIKUM 4	30
4.1. <i>ERD</i> projek penyewaan kamar	30
4.2. Pengenalan <i>Data Flow Diagram</i> (DFP).....	31
4.2.1 Diagram Level 0 (Diagram Konteks)	32
4.2.2 DFP Diagram Level 1.....	33
4.2.3 Perbedaan DFP Level 0 dan DFP Level 1	33
4.3. Proses Pembuatan Sistem Penyewaan Kamar.....	34
4.3.1 Halaman utama	34
4.3.2 Halaman Daftar Kamar.....	35
4.3.3 Halaman Galeri.....	36
4.3.4 Halaman penyewaan.....	36

Kesimpulan dan Saran.....	37
4.4. Kesimpulan.....	37
4.4.1 Saran.....	39
DAFTAR PUSTAKA	40

DAFTAR TABEL

Tabel 1.1 Alat dan Bahan.....	1
Tabel 2.1 Fitur <i>Laravel</i>	19

DAFTAR GAMBAR

Gambar 3.1 Model Data Hirarki	22
Gambar 3.2 Model Data Jaringan	23
Gambar 3.3 Struktur <i>CRUD</i>	28
Gambar 4.1 ERD penyewaan kamar	31
Gambar 4.2 Diagram Konteks.....	33
Gambar 4.3 <i>DFD Level 1</i>	33
Gambar 4.4 Halaman Utama.....	34
Gambar 4.5 Halaman Utama Lanjutan.....	35
Gambar 4.6 Halaman Daftar Kamar	35
Gambar 4.7 Halaman Galeri	36
Gambar 4.8 Halaman Penyewaan	36

DAFTAR PROGRAM

Program 2.1 <i>Syntax Class</i>	7
Program 2.2 <i>Syntax Property</i>	8
Program 2.3 <i>Syntax Method OOP</i>	9
Program 2.4 <i>Syntax Object</i>	10
Program 2.5 <i>Public Encapsulation</i>	11
Program 2.6 <i>Syntax Private Encapsulation</i>	12

PRAKTIKUM 1

1.1. Alat dan Bahan

Adapun alat dan bahan yang di gunakan pada praktikum kali ini adalah sebagai berikut:

Alat dan Bahan	penjelasan
Laptop	Sebagai tempat untuk menyimpan data, untuk mengerjakan projek dan sebagai tempat untuk mengoding.
Xampp	Sebagai penghubung antara <i>chrome</i> dan sublime.
Sublime text	Sebagai tempat mengoding sebuah program.
<i>Chrome</i> atau <i>web browser</i>	Sebagai tempat untuk melihat hasil <i>running</i> dari program yang telah di buat.

Tabel 1.1 Alat dan Bahan

1.2. Pengenalan Pemrograman Berorientasi Objek (PBO)

Bahasa pemrograman adalah bahasa khusus yang memungkinkan seseorang programer memberi tahu komputer untuk melakukan sesuatu, dengan mengatakannya dengan tepat bagaimana melakukan hal itu. Seorang pemrogram menulis kode sumber program, dan menjalankan program khusus, yang disebut compiler, yang mengubah kode sumber menjadi sesuatu yang dapat dimengerti oleh komputer (Frank, 2016). Kegiatan menggunakan algoritma pada bahasa pemrograman menjadi sebuah program komputer adalah pemrograman (Yasin, 2012).

Pemograman Berorientasi Object atau dalam bahasa inggris lebih dikenal dengan *Object Oriented Programming* (OOP) adalah sebuah paradigma dalam pemograman yang menyelesaikan masalah program dengan menyediakan objek-objek(terdiri dari beberapa *attribute* dan *method*) yang saling berkaitan dan disusun kedalam satu kelompok atau yang disebut dengan *class*. Nantinya objek-objek tersebut akan saling berinteraksi untuk menyelesaikan masalah program yang rumit.

Pemrograman berorientasi objek ditemukan pada Tahun 1960, dimana berawal dari suatu pembuatan program yang terstruktur (*structured programming*). Metode ini dikembangkan dari bahsa C dan Pascal. Dengan program yang terstruktur inilah untuk pertama kalinya kita mampu menulis program yang begitu sulit dengan lebih mudah.

Ide dasar pada OOP adalah mengkombinasikan data dan fungsi untuk mengakses data menjadi sebuah kesatuan unit yang dikenal dengan nama objek. Objek adalah struktur data yang terdiri dari bidang data dan metode bersama dengan interaksi mereka untuk merancang aplikasi dan program komputer. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

Menurut Gata (2012:7) OOP (*Object Oriented Programing*), merupakan suatu cara atau paradigm pemrograman yang berorientasikan kepada objek. Pemrograman berorientasi objek ditemukan pada Tahun 1960, dimana berawal dari suatu pembuatan program yang terstruktur (*structured programming*).

Metode ini dikembangkan dari bahsa C dan Pascal. Dengan program yang terstruktur inilah untuk pertama kalinya kita mampu menulis program yang begitu sulit dengan lebih mudah.

Ide dasar pada OOP adalah mengkombinasikan data dan fungsi untuk mengakses data menjadi sebuah kesatuan unit yang dikenal dengan nama objek. Objek adalah struktur data yang terdiri dari bidang data dan metode bersama dengan interaksi mereka untuk merancang aplikasi dan program komputer. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

1.3. Prinsip OOP

Prinsip OOP ada 3, yaitu *encapsulation*, *inheritance*, serta *polymorphism*. *Encapsulation* adalah mekanisme pemrograman yang membungkus kode dan data yang dimanipulasi dan menjaganya supaya terhindar dari interferensi dan penggunaan yang tidak perlu. Salah satu caranya dengan membentuk objek. Sedangkan manfaat encapsulation adalah :

- a. Menyembunyikan implementasi detil sebuah *class*, dan menyediakan *public method*
- b. Memaksa pengguna untuk menggunakan method untuk mengakses data
- c. Membuat kode lebih terpelihara

Inheritance memungkinkan programer meletakkan member yang sama dalam satu *class* dan *class-class* lain dapat mewarisi member tersebut. *Class* yang

mengandung member yang sama dari beberapa *class* lain dinamakan *superclass* atau parent *class*. *Class* yang mewarisi dinamakan *subclass* atau child *class*. Inheritance menghasilkan *class hierarchy*. Hasil dari pendefinisian suatu class yang lain disebut sebagai subclass. Definisi *subclass* dapat juga digunakan menjadi subclass yang lain. Proses pendefinisian suatu class berdasarkan class yang lain disebut dengan pewarisan (*inheritance*) menurut Sidik (2012:526).

Polymorphism artinya mempunyai banyak bentuk. Dua objek atau lebih dikatakan sebagai *polymorphic*, bila objek-objek itu mempunyai antar muka yang identik namun mempunyai perilaku-perilaku yang berbeda

1.4. Pengenalan Mengenai PHP

Pada awalnya PHP merupakan kependekan dari *Personal Home Page* (Situs personal). PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama *Form Interpreted* (FI), yang wujudnya berupa sekumpulan skrip yang digunakan untuk mengolah data formulir dari web. Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FI. Dengan perilisan kode sumber ini menjadi sumber terbuka, maka banyak pemrogram yang tertarik untuk ikut mengembangkan PHP.

Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini, *interpreter* PHP sudah diimplementasikan dalam program C. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP/FI secara signifikan. Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang *interpreter* PHP menjadi lebih bersih, lebih baik, dan lebih cepat.

Kemudian pada Juni 1998, perusahaan tersebut merilis *interpreter* baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0 dan singkatan PHP diubah menjadi akronim berulang *PHP*: *Hypertext Preprocessing*. Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi. Pada Juni 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek. *Server web* bawaan ditambahkan pada versi 5.4 untuk mempermudah pengembang menjalankan kode PHP tanpa menginstall software server.

PHP atau Hypertext Preprocessor merupakan bahasa pemrograman yang bersifat dinamis yang di desain khusus untuk *web development* atau pengembangan web. PHP memiliki sifat *Server-Side* karena PHP dijalankan atau di eksekusi dari sisi *server* bukan pada komputer *client*. PHP di jalankan melalui aplikasi *web browser* sama halnya seperti HTML.

Dalam *website* dinamis atau pun interaktif, bahasa pemrograman PHP dipakai sebagai media untuk mempersingkat tatanan bahasa pemrograman HTML dan CSS. Dalam pembuatan website yang berisi data siswa misalnya. Dengan menggunakan bahasa pemrograman HTML dan CSS, maka dibutuhkan baris kode yang sangat panjang (sesuai dengan jumlah data siswa yang ingin diinput).

sedangkan dengan menggunakan bahasa pemrograman PHP, baris kode yang dibutuhkan dapat dipersingkat hingga menjadi beberapa baris saja.

Selain dapat mempersingkat *script* bahasa pemrograman, PHP juga dapat digunakan untuk menginput data ke sistem *database*, mengkonversi halaman yang berisi text menjadi dokumen PDF, melaksanakan manajemen *cookie* dan *session* dalam berbagai macam aplikasi, menghasilkan gambar, dan berbagai macam kegunaan lainnya.

PRAKTIKUM 2

2.1. Pengenalan *Class, Object, Property* dan *Method*

2.1.1 *Class OOP*

Class adalah wadah berisi pemodelan suatu objek, mendeskripsikan karakteristik dan fungsi objek tersebut. Karena *class* merupakan wadah yang akan digunakan untuk menciptakan objek tersebut, maka *Class* harus diciptakan terlebih dahulu [Hermawan, 2004].

Syntax Class OOP

```
<?php  
class Fruit {  
    // code goes here...  
}  
?  
?>
```

Program 2.1 *Syntax Class*

2.1.2 *Property*

Property (atau disebut juga dengan atribut) adalah data yang terdapat dalam sebuah *class*. Melanjutkan analogi tentang laptop, *property* dari laptop bisa berupa merk, warna, jenis processor, ukuran layar, dan lain-lain.

Jika anda sudah terbiasa dengan program PHP, *property* ini sebenarnya hanyalah *variabel* yang terletak di dalam *class*. Seluruh aturan dan tipe data yang biasa diinput ke dalam *variabel*, bisa juga diinput kedalam *property*. Aturan tata cara penamaan *property* sama dengan aturan penamaan *variabel*.

Berikut adalah contoh penulisan *class* dengan penambahan *property* :

```
<?php  
class laptop {  
    var $pemilik;  
    var $merk;  
    var $ukuran_layar;  
    // lanjutan isi dari class laptop...  
}  
?>
```

Program 2.2 Syntax Property

2.1.3 *Method*

Method adalah kumpulan program yang mempunyai nama. *Method* merupakan sarana bagi *programmer* untuk memecah program menjadi bagian-bagian yang kecil agar jadi lebih kompleks sehingga dapat di gunakan berulang-ulang.

Method merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu *object*. *Method* didefinisikan pada *class* akan tetapi dipanggil melalui *object*.

Method pada dasarnya adalah *function* yang berada di dalam *class*. Seluruh fungsi dan sifat *function* bisa diterapkan ke dalam *method*, seperti argumen/parameter, mengembalikan nilai (dengan keyword *return*), dan lain-lain.

Contoh syntax *method* pada oop

```
<?php  
class Fruit {  
    // Properties  
    public $name;  
    public $color;  
  
    // Methods  
    function set_name($name) {  
        $this->name = $name;  
    }  
    function get_name() {  
        return $this->name;  
    }  
}  
?>
```

Program 2.3 Syntax Method OOP

2.1.4 *Object*

Object adalah gabungan antara beberapa data dan fungsi yang masing-masing bekerja bersama-sama dan tidak dapat dipisahkan. Gabungan dari data dan fungsi tersebut akan membentuk suatu *object-object* yang aktif. Dari kumpulan beberapa *object* yang sama akan membentuk struktur baru yang disebut *class*. (Thomas, 2001)

Contoh *syntax object* pada php

```
<?php  
class Fruit {  
    // Properties  
    public $name;  
    public $color;  
  
    // Methods  
    function set_name($name) {  
        $this->name = $name;  
    }  
    function get_name() {  
        return $this->name;  
    }  
}  
  
$apple = new Fruit();  
$banana = new Fruit();  
$apple->set_name('Apple');  
$banana->set_name('Banana');  
  
echo $apple->get_name();  
echo "<br>";  
echo $banana->get_name();  
?>
```

Program 2.4 *Syntax Object*

2.2. Enkapsuli (*Encapsulation*)

Encapsulation adalah mekanisme membungkus sebuah data pada sebuah object. Dalam istilah lain seringkali disebut "*Information Hiding*". Pada PHP terdapat tiga modifier yang dapat diimplementasikan untuk melakukan pembungkusan data yaitu *private*, *protected* dan *public*. *Modifier* tersebut

digunakan untuk mendefinisikan tingkat visibilitas sebuah data(properti) atau fungsi (metode) yang ada di dalam *class*(Aziz.2005).

Ada 3 Kunci dalam oop untuk mengatur hak akses dari suatu *property* dan *method*, yaitu :

2.2.1 *Public*

Public, Ketika sebuah property atau method dinyatakan sebagai public, maka seluruh kode program di luar *class* bisa mengaksesnya, termasuk *class* turunan. Berikut adalah contoh penulisan *public property* dan *public method* dalam PHP.

Contoh syntax enkapsulasi publik pada OOP

```
<?php

// buat class laptop
class laptop {

    // buat public property
    public $pemilik;

    // buat public method
    public function hidupkan_laptop() {
        return "Hidupkan Laptop";
    }
}
?>
```

Program 2.5 Public Encapsulation

2.2.2 *Private*

Private, Jika sebuah *property* atau *method* di-set sebagai *private*, maka satu-satunya yang bisa mengakses adalah *class* itu sendiri. *Class* lain tidak bisa mengaksesnya, termasuk *class* turunan.

Sebagai contoh, berikut adalah hasil yang di dapat jika kita mengakses *property* dan *method* dengan *level private* :

```
<?php
// buat class komputer
class komputer {
    // property dengan hak akses protected
    private $jenis_processor = "Intel Core i7-4790 3.6Ghz";
    public function tampilan_processor() {
        return $this->jenis_processor;
    }
}
?>
```

Program 2.6 *Syntax Private Encapsulation*

2.2.3 *Protected*

Protected, jika sebuah *property* atau *method* dinyatakan sebagai *protected*, berarti *property* atau *method* tersebut tidak bisa diakses dari luar *class*, namun bisa diakses oleh *class* itu sendiri atau turunan *class* tersebut.

Apabila kita mencoba mengakses *protected property* atau *protected method* dari luar *class*, akan menghasilkan *error*.

2.3. *Constructor* dan *Destructor*

2.3.1 *Constructor*

Menurut Aziz (2005:23) *Constructor* adalah *method* yang akan dipanggil pertama kali setiap pembuatan sebuah object dari suatu *class*

Constructor biasa dipakai untuk membuat proses awal dalam persiapan *object*, seperti memberi nilai kepada data member, memanggil *member function internal* serta beberapa proses lain yang dirasa perlu.

Sebuah *constructor* tidak mengembalikan nilai sehingga tidak perlu menulis tipe data sebelum nama function. *Constructor* juga harus di set sebagai public, jika tidak kita tidak bisa men-instansiasi *class* tersebut.

Ciri lain dari *constructor* adalah, hanya boleh ada satu *constructor* di setiap *class*.

2.3.2 *Desctructor*

Menurut Aziz (2005:23) *Destructor* adalah *method* yang akan dipanggil terakhir kali setiap pembuatan sebuah *object* dari suatu *class*.

Destructor adalah *member function* khusus yang dijalankan secara otomatis pada saat sebuah objek dihapus.

Salah satu kasus dimana perlu *destructor* adalah jika kode program kita mengakses memori komputer secara langsung (menggunakan *pointer*), maka *destructor* bisa dipakai untuk “melepaskan” ruang memori tersebut agar tidak terjadi *memory leak*.

Memory leak adalah istilah *programming* dimana sebuah aplikasi menggunakan *memory* RAM terlalu besar dan terus membesar sepanjang program berjalan. *Memory leak* merupakan *bug* yang harus dihindari, tapi selama kita tidak menggunakan *pointer*, seharusnya tidak ada masalah.

Mirip seperti *constructor*, *destructor* juga tidak butuh tipe data kembalian. Dan dalam sebuah *class* hanya bisa terdapat satu member function *destructor*.

Berbeda dengan *constructor*, *destructor* tidak bisa menerima parameter / argumen.

2.4. *Interfaces*

Secara sederhana, *Object Interface* adalah sebuah ‘kontrak’ atau perjanjian implementasi *method*. Bagi *class* yang menggunakan *object interface*, *class* tersebut harus mengimplementasikan ulang seluruh *method* yang ada di dalam interface. Dalam pemrograman objek, penyebutan *object interface* sering disingkan dengan ‘*Interface*’ saja.

Jika anda telah mempelajari abstract *class*, maka *interface* bisa dikatakan sebagai bentuk lain dari abstract *class*. Walaupun secara konsep teoritis dan tujuan penggunaannya berbeda.

Sama seperti abstract *class*, *interface* juga hanya berisi *signature* dari *method*, yakni hanya nama *method* dan parameternya saja (jika ada). Isi dari *method* akan dibuat ulang di dalam *class* yang menggunakan *interface*.

Jika kita menganggap abstract *class* sebagai ‘kerangka’ atau ‘*blue print*’ dari *class-class* lain, maka *interface* adalah implementasi *method* yang harus ‘tersedia’ dalam sebuah objek. *Interface* tidak bisa disebut sebagai ‘kerangka’ *class*.

Menyambung analogi kita tentang *class* komputer, *interface* bisa dicontohkan dengan ‘*mouse*’, atau ‘*keyboard*’. Di dalam *interface mouse*, kita bisa membuat method seperti *klik_kiri()*, *klik_kanan()*, dan *double_klik()*. Jika *class laptop* ‘menggunakan’ *interface mouse*, maka *class* tersebut harus membuat ulang *method* *klik_kiri()*, *klik_kanan()*, dan *double_klik()*.

2.5. *Laravel*

Laravel merupakan *Framework* PHP yang menekankan pada kesederhanaan dan fleksibilitas pada desainnya. *Laravel* dirilis dibawah lisensi MIT dengan sumber kode yang disediakan di Github. Sama seperti framework PHP lainnya, *Laravel* dibangun dengan basis MVC (*Model-View-Controller*). *Laravel* dilengkapi *command line tool* yang bernama “Artisan” yang bisa digunakan untuk packging bundle dan instalasi *bundle*. Framework *Laravel* dibuat oleh Taylor Otwell, proyek *Laravel* dimulai pada April 2011. Awal mula proyek ini dibuat karena Otwell sendiri tidak menemukan *framework* yang *upto-date* dengan versi PHP. Mengembangkan *framework* yang sudah ada juga bukan merupakan ide yang bagus karena keterbatasan sumber daya. Dikarenakan beberapa keterbatasan tersebut, Otwell membuat sendiri framework dengan nama *laravel*. Oleh karena itu *laravel* mensyaratkan PHP versi 5.3 keatas. (Rohman, 2014).

Laravel fokus di bagian end-user, yang berarti fokus pada kejelasan dan kesederhanaan, baik penulisan maupun tampilan, serta menghasilkan fungsionalitas aplikasi *web* yang bekerja sebagaimana mestinya. Hal ini membuat *developer* maupun perusahaan menggunakan *framework* ini untuk membangun apa pun, mulai dari proyek kecil hingga skala perusahaan kelas atas.

Laravel mengubah pengembangan website menjadi lebih elegan, ekspresif, dan menyenangkan, sesuai dengan jargonnya “*The PHP Framework For Web Artisans*”. Selain itu, *laravel* juga mempermudah proses pengembangan website

dengan bantuan beberapa fitur unggulan, seperti *Template Engine*, *Routing*, dan *Modularity*.

Laravel menawarkan beberapa keuntungan ketika Anda mengembangkan website menggunakan dasar framework ini

- a. Pertama, website menjadi lebih *scalable* (mudah dikembangkan).
- b. Kedua, terdapat *namespace* dan tampilan yang membantu Anda untuk mengorganisir dan mengatur sumber daya website.
- c. Ketiga, proses pengembangan menjadi lebih cepat sehingga menghemat waktu karena *Laravel* dapat dikombinasikan dengan beberapa komponen dari framework lain untuk mengembangkan website.

Laravel mempunyai berbagai macam fitur yang tidak semua *framework* menyediakannya. Apalagi *laravel* adalah framework yang modern sehingga Anda dapat melakukan berbagai hal menggunakan *framework* ini seperti proses otentifikasi terbaru

<i>Blade template engine</i>	<p><i>Laravel</i> menggunakan <i>Blade</i>. <i>Blade</i> merupakan template <i>engine</i> untuk mendesain layout yang unik. Layout yang didesain dapat digunakan di tampilan lain sehingga menyediakan konsistensi desain dan struktur selama proses pengembangan. Dibandingkan dengan <i>template engine</i> lain, <i>Blade</i> mempunyai kelebihan: tidak membatasi pengembang untuk menggunakan kode PHP biasa di dalam tampilan; desain tampilan <i>blade</i> akan tetap di-<i>cache</i> sampai dengan ada modifikasi.</p>
------------------------------	--

<i>Routing</i>	Di <i>laravel</i> , semua <i>request</i> dipetakan dengan bantuan rute. Dasar dari routing adalah merutekan <i>request</i> ke kontroler terkait. <i>Routing</i> ini dianggap dapat mempermudah pengembangan website dan meningkatkan performanya. Setidaknya ada tiga kategori <i>routing</i> di <i>laravel</i> , yaitu <i>basic routing</i> , <i>route parameters</i> , dan <i>named routes</i> .
<i>Modularity</i>	Seperti yang sudah dibahas di bagian sebelumnya, di dalam <i>Laravel</i> terdapat kumpulan modul dan <i>library</i> yang terkait dengan <i>composer</i> . Fitur ini membantu Anda untuk menyempurnakan dan meningkatkan fungsionalitas dari website yang dibangun, serta mempermudah proses <i>update</i> .
<i>Testability</i>	<i>Laravel</i> dibangun dengan fitur proses pengecekan yang cukup lengkap. <i>Framework</i> ini mendukung proses pengecekan dengan PHPUnit dan file phpunit.xml yang dapat disesuaikan dengan aplikasi web yang sedang dibangun. <i>Framework</i> ini juga dibangun menggunakan metode pembantu yang nyaman. Metode ini memungkinkan Anda untuk menguji website secara ekspresif
<i>Query builder and ORM</i>	<i>Laravel</i> database query builder menyediakan antarmuka yang lancar untuk membuat dan menjalankan <i>database query</i> . Fitur ini dapat

	digunakan untuk menjalankan berbagai operasi database di dalam <i>website</i> dan mendukung berbagai sistem <i>database</i> .
<i>Authentication</i>	<i>Laravel</i> membuat pengimplementasian otentikasi menjadi sangat sederhana. Seluruh proses konfigurasi otentikasi sudah berjalan secara otomatis. Anda bisa menemukan file konfigurasi otentikasi ini di ‘config/auth.php’. Di dalam file ini terdapat beberapa opsi otentifikasi yang sudah terdokumentasikan dengan baik dan sewaktu-waktu dapat Anda sesuaikan dengan kebutuhan sistem.
<i>Schema builder</i>	<i>Class laravel Schema</i> menyediakan database <i>agnostic</i> untuk memanipulasi tabel. <i>Schema</i> ini berjalan baik di berbagai tipe <i>database</i> yang didukung <i>Laravel</i> dan mempunyai API yang sama di seluruh sistem.
<i>Configuration management features</i>	Seluruh file konfigurasi <i>Laravel</i> disimpan di dalam direktori config. Setiap opsi didokumentasikan dengan baik. Jadi Anda tidak perlu khawatir untuk mengubah setiap konfigurasi yang tersedia.
<i>E-mail Class</i>	<i>Laravel</i> menyediakan API beberapa library SwiftMailer yang cukup populer dengan koneksi ke SMTP, Postmark, Mailgun, SparkPost, Amazon SES, dan sendmail. Fitur ini memungkinkan Anda untuk mengirimkan <i>email</i> dengan cepat

	melalui aplikasi lokal maupun layanan <i>cloud</i> .
<i>Redis</i>	<i>Laravel</i> menyediakan API beberapa library SwiftMailer yang cukup populer dengan koneksi ke SMTP, Postmark, Mailgun, SparkPost, Amazon SES, dan sendmail. Fitur ini memungkinkan Anda untuk mengirimkan email dengan cepat melalui aplikasi lokal maupun layanan <i>cloud</i> .
<i>Event and Command Bus</i>	<i>Laravel</i> Command Bus menyediakan metode pengumpulan tugas yang dibutuhkan aplikasi supaya dapat berjalan secara simpel dan perintah yang mudah dimengerti.

Tabel 2.1 Fitur *Laravel*

2.6. *Composer*

Composer adalah *tools dependency manager* pada PHP, *Dependency* (ketergantungan) sendiri diartikan ketika project PHP yang kamu kerjakan masih membutuhkan atau memerlukan library dari luar. *Composer* berfungsi sebagai penghubung antara project PHP kamu dengan library dari luar. Jika Bahasa pemrograman PHP menggunakan *Composer* sebagai *dependency manager*, Maka sama halnya seperti Ruby yang menggunakan Gem, Java menggunakan Maven and Gradle dan seluruh komunitas JS berfokus pada npm.

Dependency manager memungkinkan kamu untuk membuat dan mengambil *library* pada *project* PHP kamu pada library packagist.org.

Packagist.org sendiri merupakan situs yang menyediakan banyak *library* yang bisa kamu gunakan. Dengan bantuan tools tersebut kamu bisa terhubung pada situs packagist.org dan kamu dapat mengambil dan mengupload *library*.

PRAKTIKUM 3

3.1. Model Data Bebasis Objek

Model data berbasis objek menggunakan konsep entitas, atribut dan hubungan antar entitas. Dan model ini terdiri dari :

3.1.1 *Entity Relationship model*

Model untuk menjelaskan hubungan antar data dalam basis data berdasarkan suatu persepsi bahwa real word terdiri dari objek-object dasar yang mempunyai hubungan atau relasi antara objek-objek tersebut E-R MODEL berisi ketentuan /aturan khusus yang harus dipenuhi oleh isi database. Aturan terpenting adalah *Mapping Cardinalities*, yang menentukan jumlah entity yang dapat dikaitkan dengan entity lainnya melalui relationship-set.

Mapping cardinality adalah hubungan antara entitas terhadap entitas dimana diantaranya terdapat relasi atau relationship

3.1.2 *Binary Model*

Binary model adalah model data yang memperluas definisi dari *entity*, bukan hanya *atributnya* tetapi juga tindakan-tindakannya.

3.1.3 *Semantic Model*

Hampir sama dengan *Entity Relationship model* dimana relasi antara objek dasar tidak dinyatakan dengan simbol tetapi menggunakan kata-kata (*Semantic*). Semantik Data Model adalah salah satu jenisnya dimana relasi antar objek dasar tidak dinyatakan dengan simbol tetapi dengan kata-kata (*Semantic*).

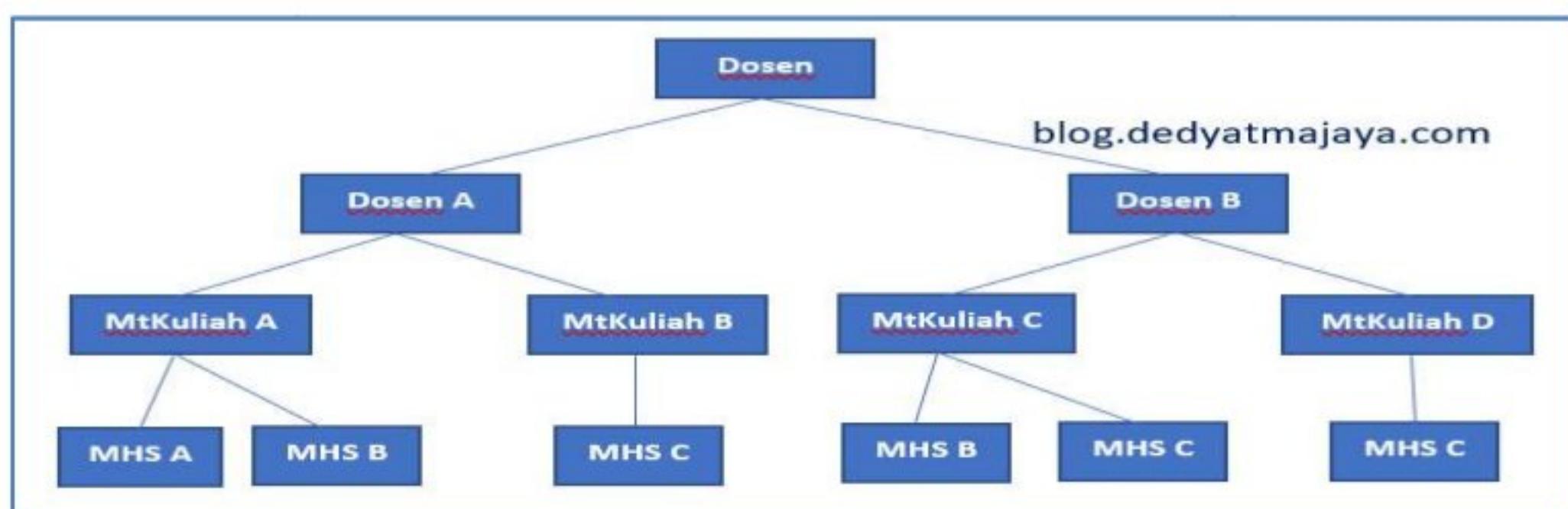
3.2. Model Data Berbasis *Record*

Model ini berdasarkan pada *record* untuk menjelaskan kepada *user* tentang hubungan *logic* antar data dalam basis data. Berbeda dengan *Object Based Data Model* (Model Data Berbasis Object), Model Data ini digunakan untuk menguraikan struktur logika keseluruhan dari suatu database, juga digunakan untuk menguraikan implementasi dari sistem database (*higher level description of implementation*)

Pada jenis model data berbasis *record* yaitu himpunan data dan prosedur atau relasi yang menjelaskan hubungan logik antar data dalam suatu basis data model yang didasarkan pada *record*. Pada jenis model data ini terdapat beberapa bagian yaitu, *Hierarchycal model*, *Network model*, *Relational model*.

3.2.1 Model Data Hirarki

Model Data Hirarki (Hirarkis) atau biasa disebut model pohon (*Tree*) atau dapat pula disebut dengan istilah orang tua dan anak (*Parent and Child*). Terdapat juga istilah simpul (bercirikan kotak atau lingkaran). Simpul yang berada diatas yang terhubung ke simpul pada level dibawahnya disebut orang tua. Perbedaannya adalah, *record-record* diorganisasikan sebagai tree (pohon) dari pada grafik.

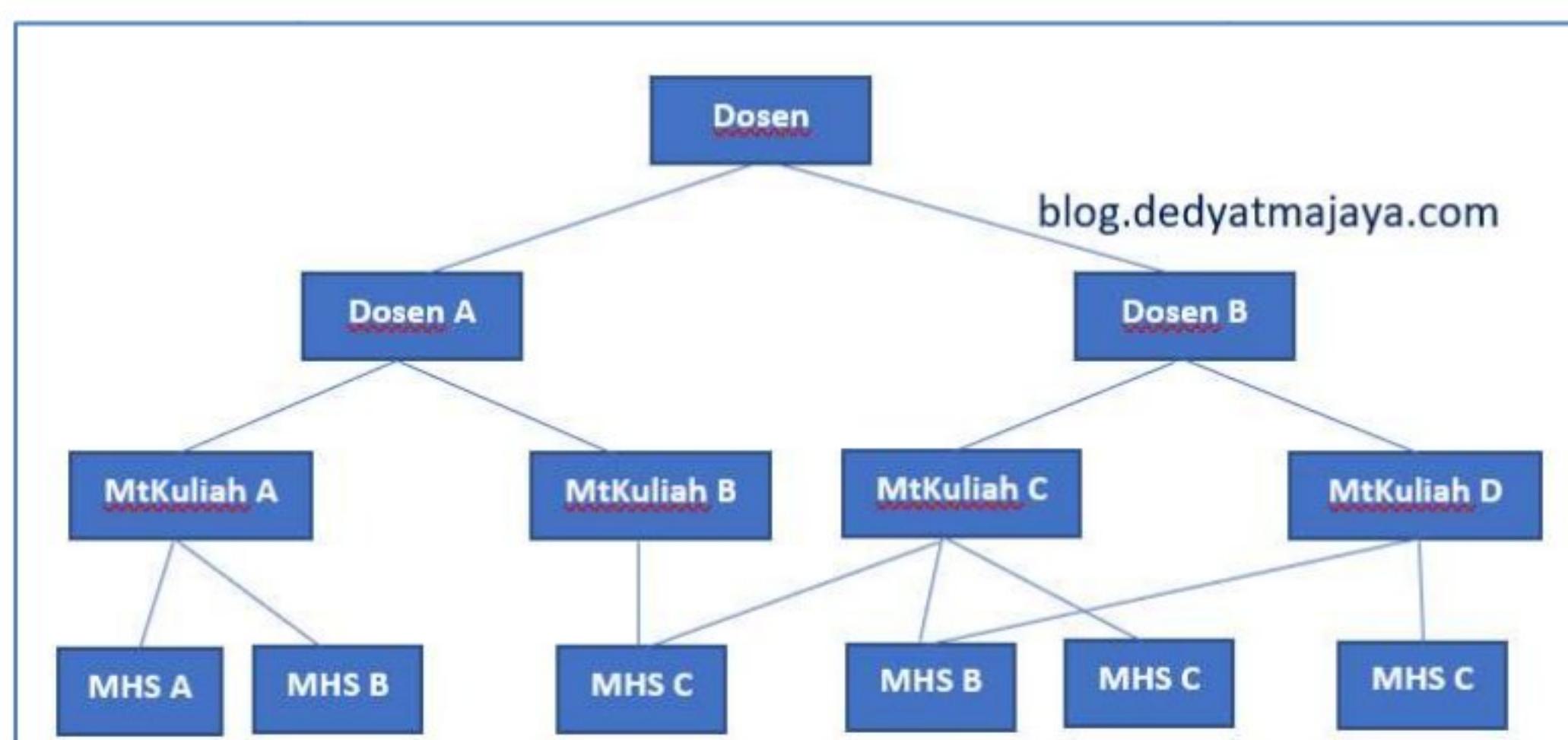


Gambar 3.1 Model Data Hirarki

- a. Kelebihan basis data hirarki
 - Data dapat dengan cepat dilakukan *retrieve*.
 - Integritas data mudah dilakukan pengaturan.
- b. Kekurangan basis data hirarki
 - Pengguna harus sangat familiar dengan struktur basis data.
 - Terjadi redundansi data.

3.2.2 Model Data Jaringan

Model data jaringan (*Network*) adalah pengembangan dari model data hirarkis, melihat kekurangan dari model hirarkis tersebut. Pada model jaringan diperkenankan bahwa sebuah *child-record* bisa memiliki lebih dari satu *parent-record*. Pada implementasinya berarti antara *parent-record* dan *child-record* diperlukan penghubung (*link* atau *pointer*) yang bisa satu arah atau dua-arah.



Gambar 3.2 Model Data Jaringan

Dengan model jaringan ini maka informasi dimana seorang mahasiswa dapat mengambil beberapa matakuliah (*pointer* dari *record* mahasiswa tersebut ke beberapa *record-kuliah*) dan juga informasi bahwa satu matakuliah dapat di-

programkan oleh banyak mahasiswa (*pointer* dari *record*-kuliah ke beberapa *record*-mahasiswa) keduanya dapat di-representasikan.

- a. Kelebihan basis data jaringan
 - Data lebih cepat diakses
 - *User* dapat mengakses data dimulai dari beberapa tabel
 - Mudah untuk memodelkan basis data yang komplek
 - Mudah untuk membentuk query yang komplek dalam melakukan *retrieve data*.
- b. Kekurangan basis data jaringan
 - Struktur basis datanya tidak mudah untuk dilakukan modifikasi
 - Perubahan struktur basis data yang telah didefinisikan akan mempengaruhi program aplikasi yang mengakses basis data
 - User harus memahami struktur basis data.

3.2.3 Model Data Rasional

Model Data Relasional (*Relational*) berbeda dengan model jaringan & hierarki. Pada model data relasional pemodelan menggunakan tabel untuk merepresentasikan data & relasi antar data. Setiap tabel terdiri atas kolom, dan setiap kolom mempunyai nama variable tertentu. Inti dari model ini adalah relasi, yang dimisalkan sebagai himpunan dari *record*. Pada model relasional, skema atau deskripsi data pada model relasi ditentukan oleh nama, nama dari tiap *field* (Atribut atau kolom), dan tipe dari tiap *field*.

- a. Kelebihan basis data relasional
 - Data sangat cepat diakses

- Struktur basis data mudah dilakukan perubahan
 - Data direpresentasikan secara logika, *user* tidak membutuhkan bagaimana data disimpan.
 - Mudah untuk membentuk *query* yang komplek dalam melakukan *retrieve data*
 - Mudah untuk mengimplementasikan integritas data
 - Data lebih akurat
 - Mudah untuk membangun dan memodifikasi program aplikasi
 - Telah dikembangkan *Structure Query Language* (SQL).
- b. Kekurangan basis data relasional
- Kelompok informasi/*tables* yang berbeda harus dilakukan *joined* untuk melakukan *retrieve data*
 - User harus familiar dengan relasi antar tabel
 - User harus belajar SQL.

3.3. Komponen *ERD*

3.3.1 *Entity*

Kumpulan objek yang dapat diidentifikasi secara unik atau saling berbeda. Biasanya, simbol dari entitas adalah persegi panjang. Selain itu, ada juga “Entitas Lemah” yang dilambangkan dengan gambar persegi panjang kecil di dalam persegi panjang yang lebih besar. Disebut entitas lemah karena harus berhubungan langsung dengan entitas lain sebab dia tidak dapat teridentifikasi secara unik.

Entitas atau *entity* didalam *database* adalah benda,orang,tempat,unit,objek atau hal lainnya yang mempresentasikan data, dan tersebut akan disimpan ke dalam pangkal data. beberapa properti entitas data mewakili hubungan antar entitas lain, contohnya entitas ‘pekerjaan’ dapat memiliki properti ‘klien’ yang merujuk padaa entitas ‘klien’ sehingga ketika anda membuat pekerjaan baru maka anda bisa memilih klien yang terhubung dengan pekerjaan tersebut. setiap entitas terdiri dari beberapa atribut cotohnya entitas karyawan maka memiliki atribut nama_karyawan,alamat,dan id.

setiap entitas memerlukan atribut kunci utama (*primary key*) atau kunci asing (*foreign key*). kunci utama adalah sebuah atribut yang unik alias isinya tidak boleh sama dalam satu atribut, misalkan entitas siswa memiliki atribut NIS.

3.3.2 *Atribut*

Komponen kedua dari ERD adalah atribut. Setiap entitas pasti mempunyai elemen yang disebut atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Atribut kunci merupakan hal pembeda atribut dengan entitas. Gambar atribut diwakili oleh simbol elips dan terbagi menjadi beberapa jenis:

1. Atribut kunci (*key*): atribut yang digunakan untuk menentukan entitas secara unik. Contoh: NPWP, NIM (Nomor Induk Mahasiswa).
2. Atribut simpel: atribut bernilai tunggal yang tidak dapat dipecah lagi (*atomic*). Contoh: Alamat, tahun terbit buku, nama penerbit.
3. Atribut multivilai (*multivalue*): atribut yang memiliki sekelompok nilai untuk setiap entitas instan. Contoh: nama beberapa pengarang dari sebuah buku pelajaran.

4. Atribut gabungan (*composite*): atribut yang terdiri dari beberapa atribut yang lebih kecil dengan arti tertentu. Contoh: nama lengkap yang terbagi menjadi nama depan, tengah, dan belakang.
5. Atribut *derivatif*: atribut yang dihasilkan dari atribut lain dan tidak wajib ditulis dalam diagram ER. Contoh: usia, kelas, selisih harga.

3.3.3 Relasi

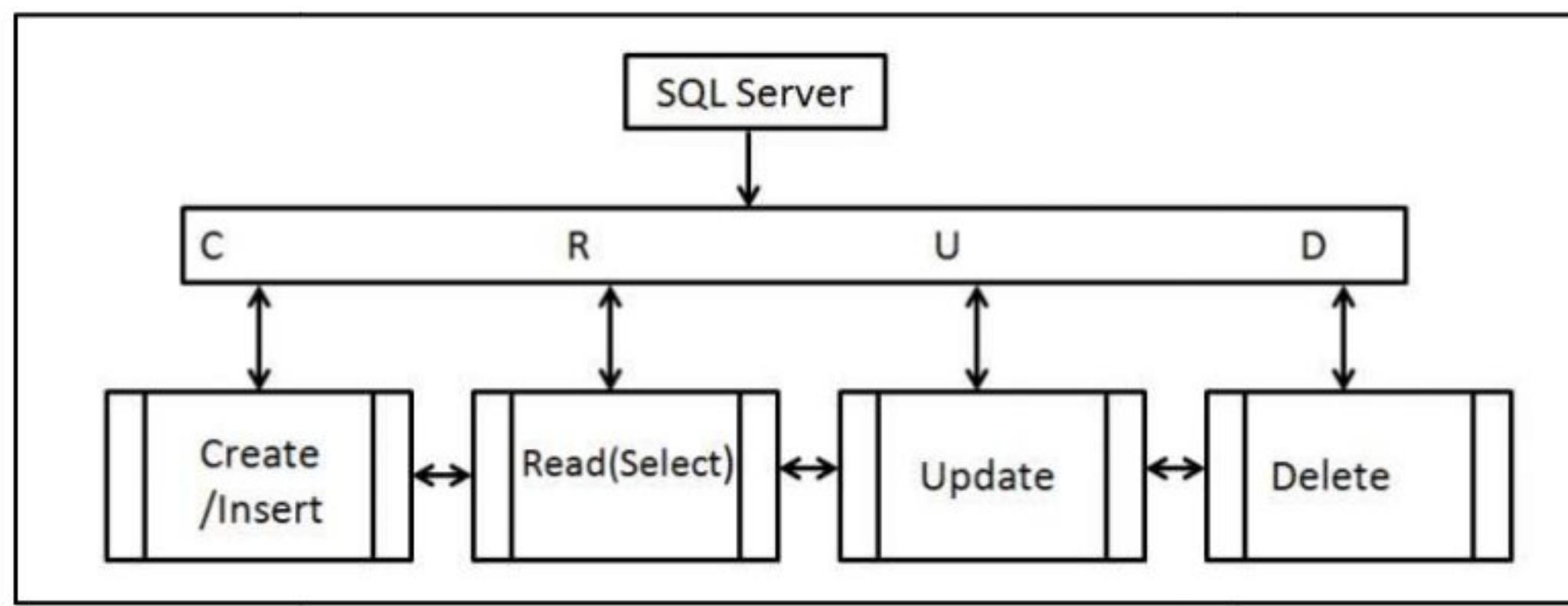
Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Gambar relasi diwakili oleh simbol belah ketupat. Relasi juga terbagi menjadi beberapa jenis:

1. *One to one*: setiap entitas hanya bisa mempunyai relasi dengan satu entitas lain. Contoh: siswa dengan nomor induk siswa
2. *One to many*: hubungan antara satu entitas dengan beberapa entitas dan sebaliknya. Contoh: guru dengan murid dan sebaliknya.
3. *Many to many*: setiap entitas bisa mempunyai relasi dengan entitas lain, dan sebaliknya. Contoh: siswa dan ekstrakurikuler.

3.4 Pengenalan *CRUD*

CRUD adalah singkatan dari *create, read, update, and delete*. Keempat poin ini menurut Techopedia merupakan fungsi-fungsi utama yang diimplementasikan dalam aplikasi database.

Akronim CRUD mengidentifikasi semua fungsi utama yang melekat pada database relasional dan aplikasi yang digunakan untuk mengelolanya, termasuk Oracle Database, Microsoft SQL Server, MySQL, dan lainnya.



Gambar 3.3 Struktur CRUD

3.4.1 *Create*

Fungsi CRUD yang pertama adalah *create*. Fungsi ini memungkinkanmu membuat *record* baru dalam database. Dalam aplikasi SQL, fungsi *create* sering disebut juga sebagai *insert*.

Kamu dapat membuat baris baru dan mengisinya dengan data yang sesuai dengan setiap atribut. Tetapi, hanya administrator yang dapat menambahkan atribut baru ke tabel itu sendiri.

3.4.2 *Read*

Fungsi *read* hampir mirip dengan fungsi *search*. Fungsi ini memungkinkan kamu untuk mencari dan mengambil data tertentu dalam tabel dan membaca nilainya.

Kamu dapat menemukan *record* yang diinginkan menggunakan kata kunci, atau dengan memfilter data berdasarkan kriteria yang diinginkan.

3.4.3 *Update*

Untuk memodifikasi *record* yang telah tersimpan di *database*, fungsi CRUD yang bisa kamu gunakan adalah fungsi *update*.

Namun, kamu perlu mengubah berbagai informasi terkait agar bisa memodifikasi *record* yang diinginkan secara utuh.

Record yang ada dalam *database* harus diubah dan semua nilai atribut diubah untuk mencerminkan karakteristik baru yang diinginkan.

3.4.4 *Delete*

Ketika ada *record* atau data yang tidak lagi dibutuhkan dalam *database*, fungsi CRUD yang digunakan adalah fungsi *delete*. Fungsi ini dapat digunakan untuk menghapus data tersebut.

Beberapa aplikasi database relasional mungkin mengizinkan kamu untuk melakukan hard delete atau soft delete.

Hard delete akan secara menghapus catatan dari database permanen. Sementara, soft delete hanya akan memperbarui status baris untuk menunjukkan bahwa data telah dihapus meskipun data tersebut tetap ada dan utuh.

PRAKTIKUM 4

Pada kesempatan kali ini saya membahas projek yang telah dibuat yaitu sistem penyewaan kamar yang dimana pembuatan projek ini menggunakan sistem *CRUD (Create, Read, Update, Delete)*.

Dalam pengembangan projek yang dibuat, saya banyak memperoleh bimbingan, masukan dan dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini saya mengucapkan terima kasih khususnya kepada dosen mata kuliah Pemograman Berorientasi Objek, Bambang Pranomo, S.Si., MT. yang telah memberikan dasar teori yang berguna selama praktikum ini serta kepada asisten Praktikum Pemograman Berorientasi Objek, Wahid Safri Jayanto atas arahan dan koreksi kepada *developer* projek yang dibuat dapat selesai seperti yang diharapkan..

Pada pengembangan sistem ini juga tidak luput dari yang namanya kesalahan baik itu *human error* maupun kesalahan pada *device* maka dari itu projek ini memerlukan kritik serta saran yang membangun agar lebih mudah dalam pengembangannya

4.1. *ERD* projek penyewaan kamar

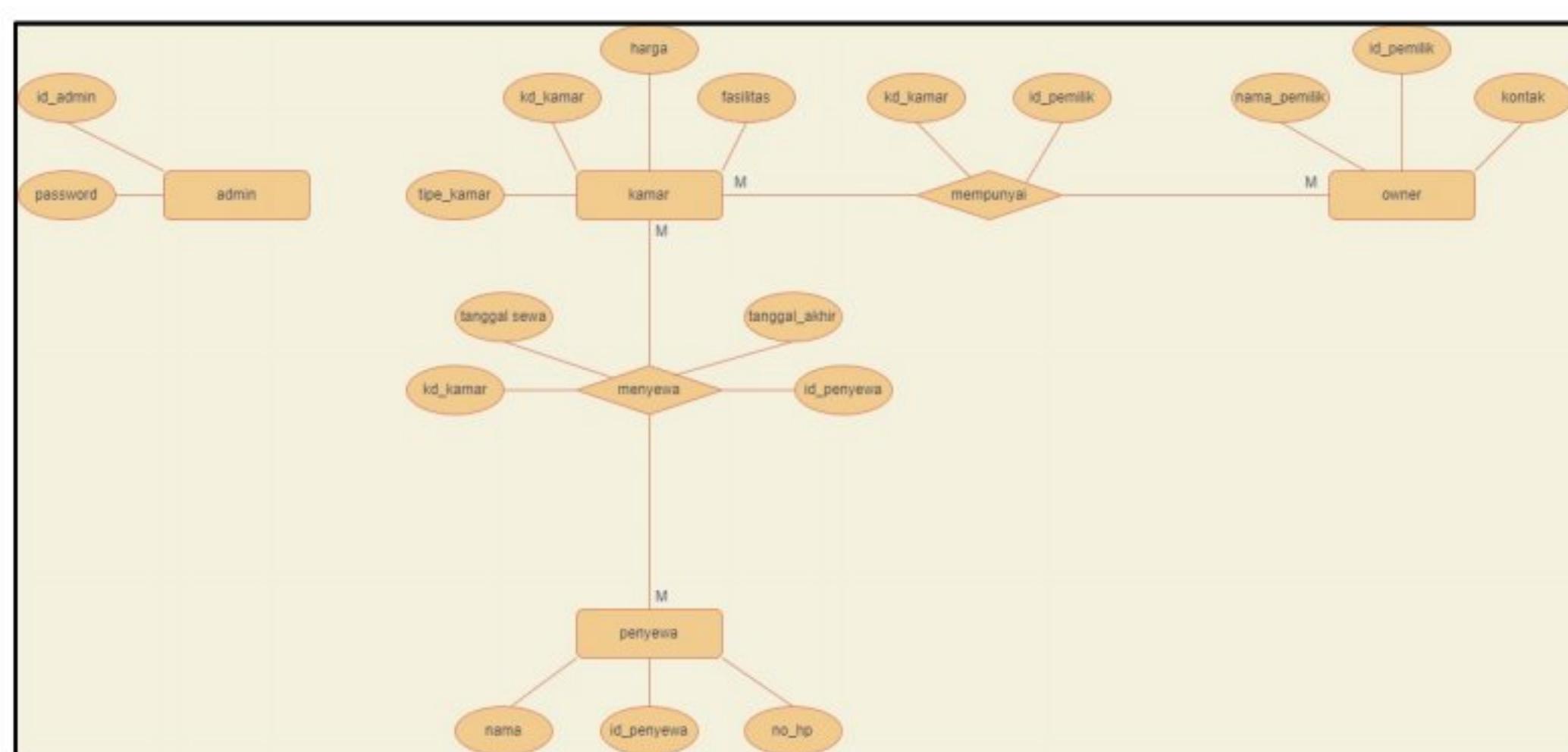
ERD dari projek kali ini mempunyai relasi *many to many* dari setiap tabelnya kecuali tabel admin. Pada *ERD* ini lebih berfokus pada penyewa kamar, yang dimana kamar yang disewakan menampilkan data yang sangat spesifik.

Tabel kamar sendiri direlasikan dengan tabel pemilik menggunakan tabel mempunyai yang dimana pada tabel pada tabel mempunyai berisi *id_kamar*(tabel kamar) dan *id_pemilik* (tabel pemilik) tabel kamar sendiri *memiliki field*

`id_kamar`, `tipe_kamar`, `harga`, dan juga fasilitas. Sedangkan tabel pemilik memiliki *field* `id_pemilik`, `nama_pemilik` dan kontak.

Tabel kamar juga direlasikan dengan tabel penyewa dengan relasi *many to many* dengan tabel penghubung menyewa. Untuk tabel penyewa sendiri memiliki *field* `nama`, `id_penyewa` dan `no_hp`.

Sementara untuk tabel admin sendiri tidak berelasi dengan tabel lainnya dan merupakan tabel yang berdiri sendiri tapi memiliki tujuan mengawasi sistem.



Gambar 4.1 ERD penyewaan kamar

4.2. Pengenalan *Data Flow Diagram* (DFP)

DFD merupakan salah satu bentuk dokumentasi sistem yang menggambarkan proses dalam sistem itu dan komponen-komponennya, serta arus data atau informasi yang mengalir di antara komponen-komponen tersebut. Para analis menggunakan modeling ERD dan DFD untuk merepresentasikan kebutuhan dan persyaratan sistem secara grafis, namun dalam penyusunannya tidak secara eksplisit dikaitkan dengan konsep proses bisnis (Hollander, 2000).

Data Flow Diagram (DFD) memberikan tampilan secara visual tentang aliran data dan informasi dari suatu sistem. Visual dari DFD ini mengambarkan

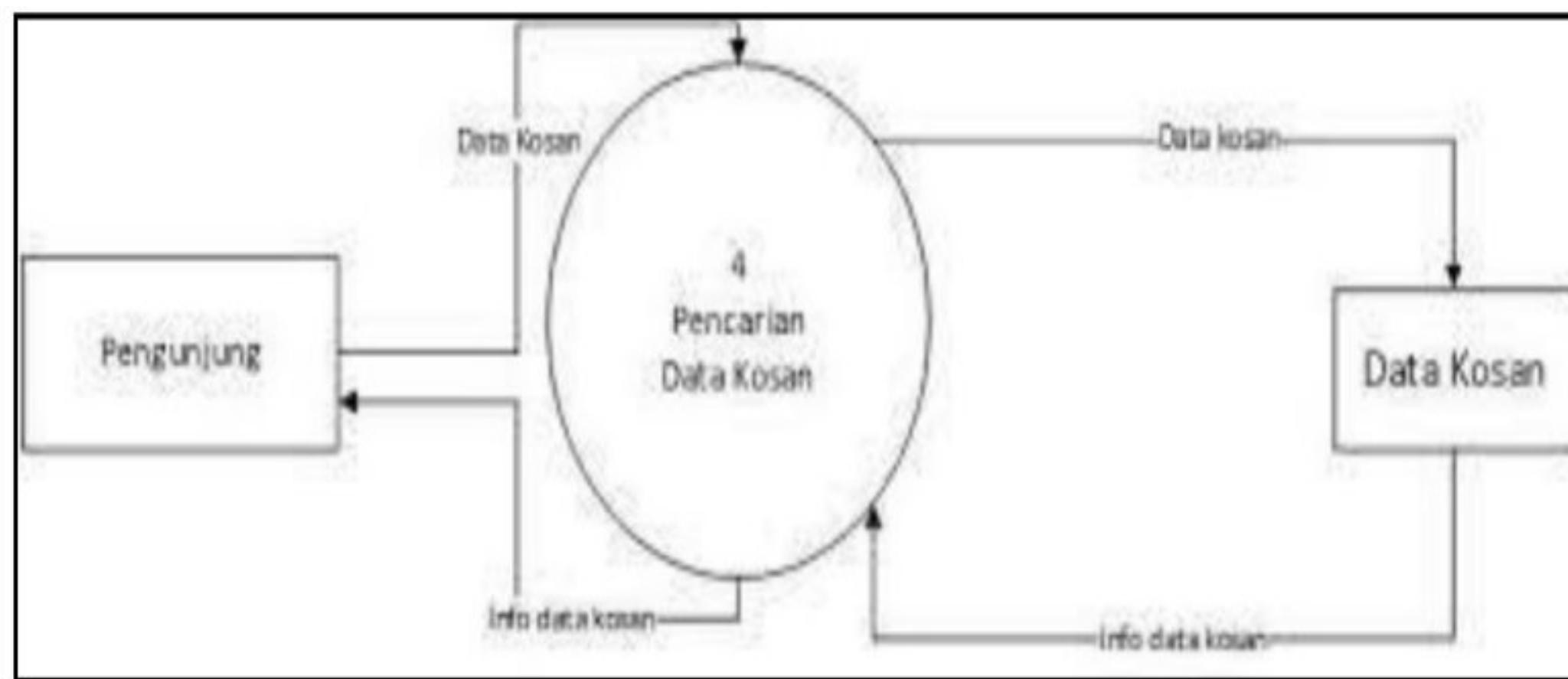
siapa saja yang terlibat pada sistem tersebut dari start sampai finish. DFD sering kali dipakai untuk mengambarkan suatu sistem yang sudah ada atau sistem baru yang akan dikembangkan. Bagi Anda yang belum mengetahui DFD adalah suatu diagram yang dibuat menggunakan notasi-notasi untuk mengambarkan aliran data dari sistem.

DFD pertama kali penggunaannya dipopulerkan oleh Larry Constantine dan Ed Yourdon dalam structured analysis and design technique. Notasi DFD memiliki acuan pada teori grafik yang awalnya digunakan untuk penelitian operasional adlam permodelan alur kerja dalam suatu organisasi. Terdapat banyak simbol yang mengambarkan DFD, satu dengan simbol lain memiliki fungsi dan kegunaan yang berbeda. Pada tahun 1979 Tom DeMarco menyimpulkan komponen DFD terdiri dari Entitas, Proses, Aliran dan Data Store.

4.2.1 *Diagram Level 0 (Diagram Konteks)*

Diagram level 0 atau bisa juga diagram konteks adalah level diagram paling rendah yang mengambarkan bagaimana sistem berinteraksi dengan external entitas. Pada diagram konteks akan diberikan nomor untuk setiap proses yang berjalan, umumnya mulai dari angka 0 untuk start awal.

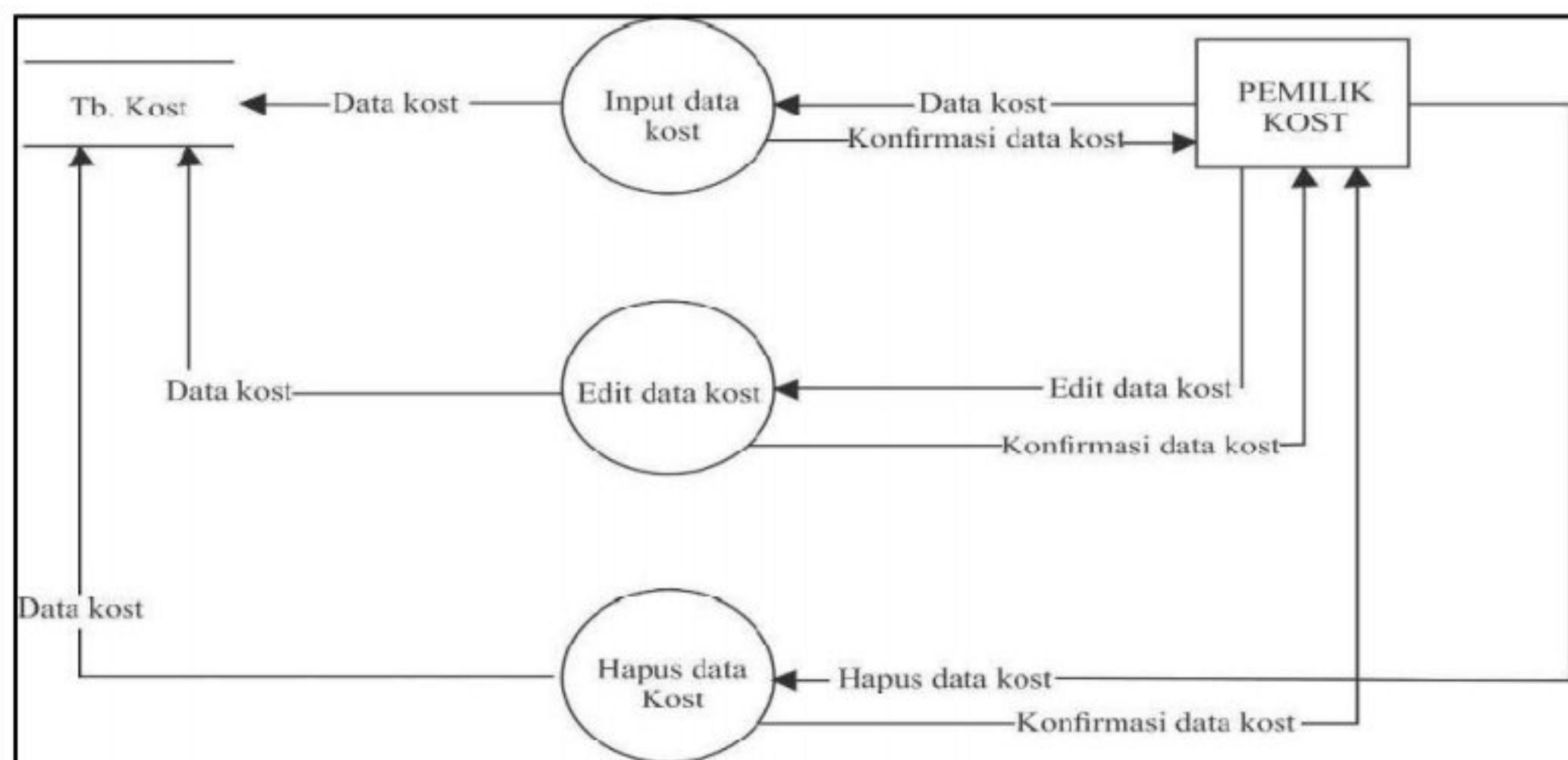
Semua entitas yang ada pada diagram konteks termasuk juga aliran datanya akan langsung diarahkan kepada sistem. Pada diagram konteks ini juga tidak ada informasi tentang data yang tersimpan dan tampilan diagramnya tergolong sederhana



Gambar 4.2 Diagram Konteks

4.2.2 DFP Diagram Level 1

DFD level 1 adalah tahapan lebih lanjut tentang DFD level 0, dimana semua proses yang ada pada DFD level 0 akan dirinci dengan lengkap sehingga lebih lengkap dan detail. Proses-proses utama yang ada akan dipecah menjadi sub-proses.



Gambar 4.3 DFD Level 1

4.2.3 Perbedaan DFP Level 0 dan DFP Level 1

Ada perbedaan antara 2 level DFD tersebut yang perlu Anda ketahui, berikut ini perbedaannya:

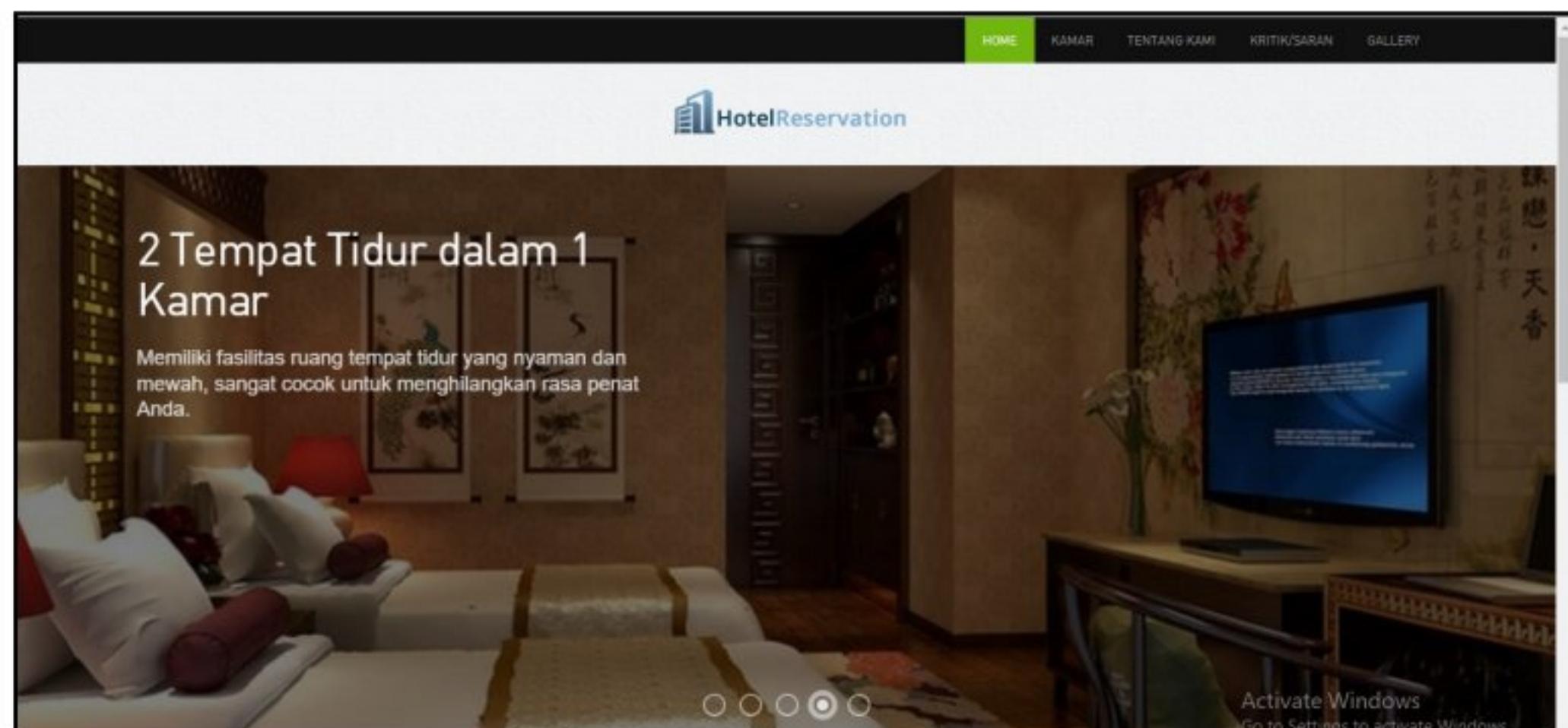
- DFD level 0 hanya mengambarkan sistem secara basic saja.

- b. DFD level 0 hanya menjelaskan aliran data dari input sampai output.
- c. DFD level 1 mengambarkan aliran data yang lebih kompleks pada setiap prosesnya yang kemudian terbentuklah data store dan aliran data.
- d. DFD level 1 mengambarkan sistem secara sebagian atau seluruhnya secara mendetail.

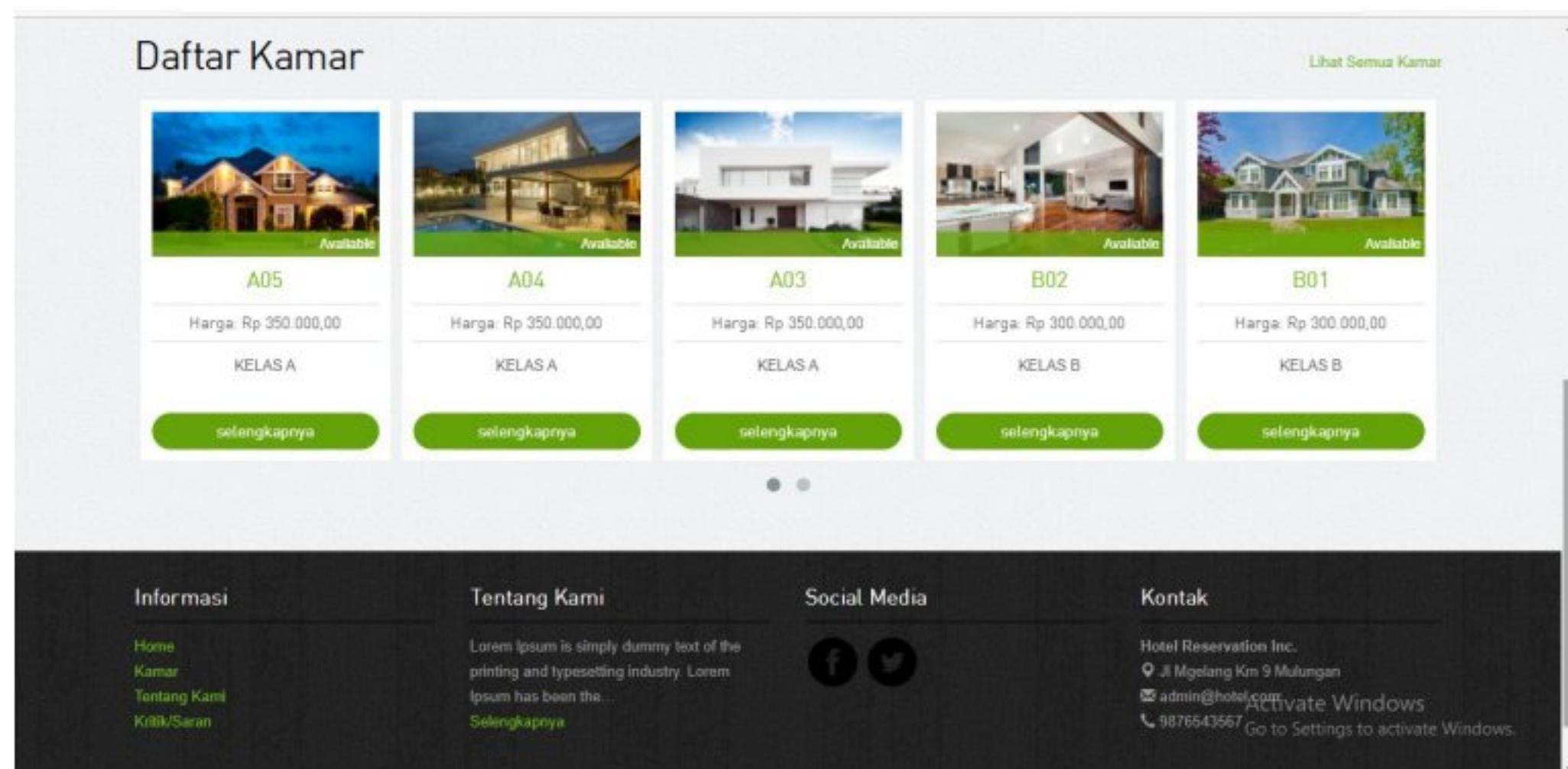
4.3. Proses Pembuatan Sistem Penyewaan Kamar

Disini saya akan menjelaskan bagaimana projek saya yang berjudul penyewaan kamar.

4.3.1 Halaman utama



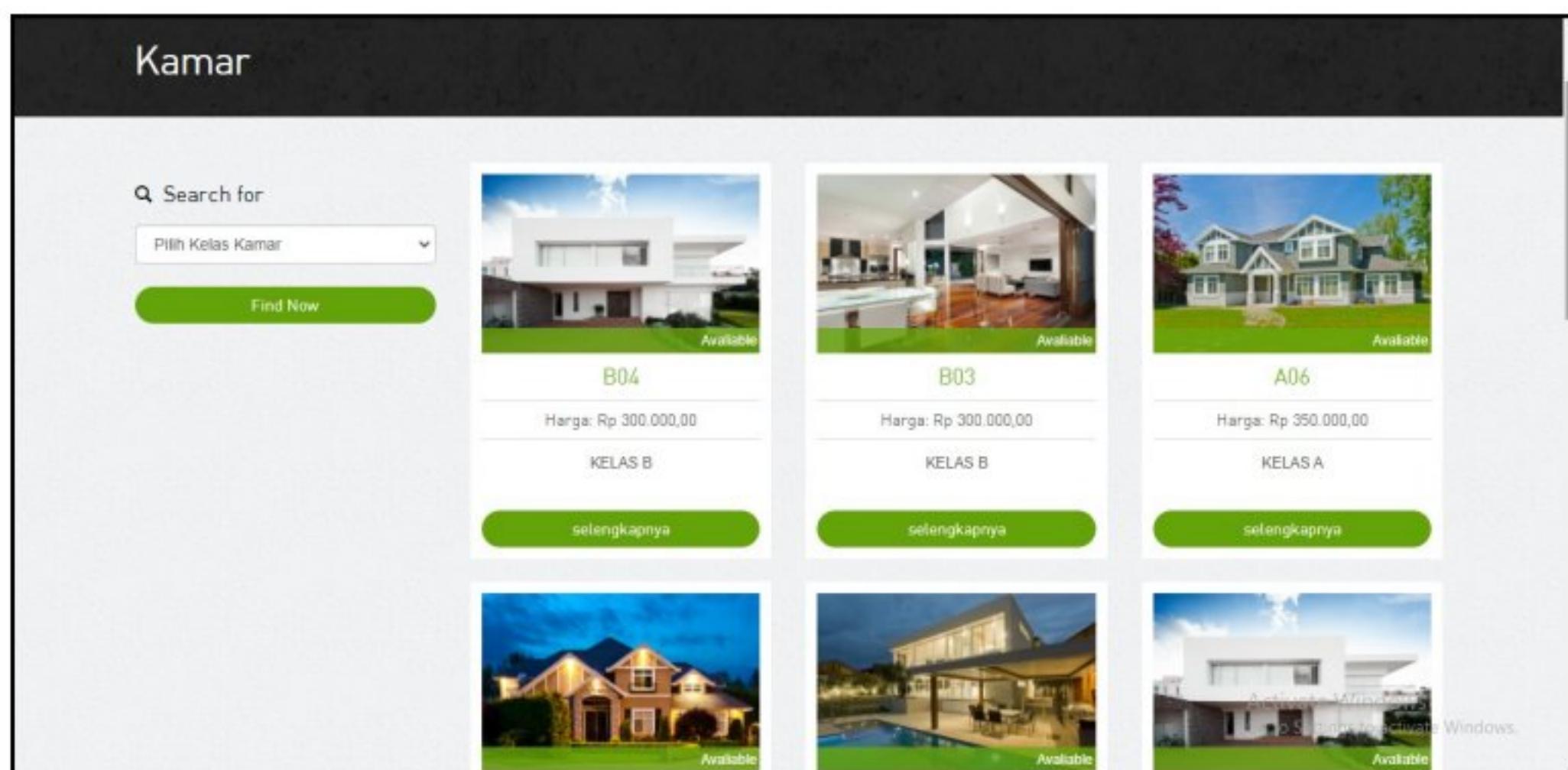
Gambar 4.4 Halaman Utama



Gambar 4.5 Halaman Utama Lanjutan

Pada halaman utama ini memperlihatkan home screen serta informasi tambahan mengenai halaman web yang kami buat, serta juga memberikan informasi mengenai kamar yang kami sewakan pada website ini.

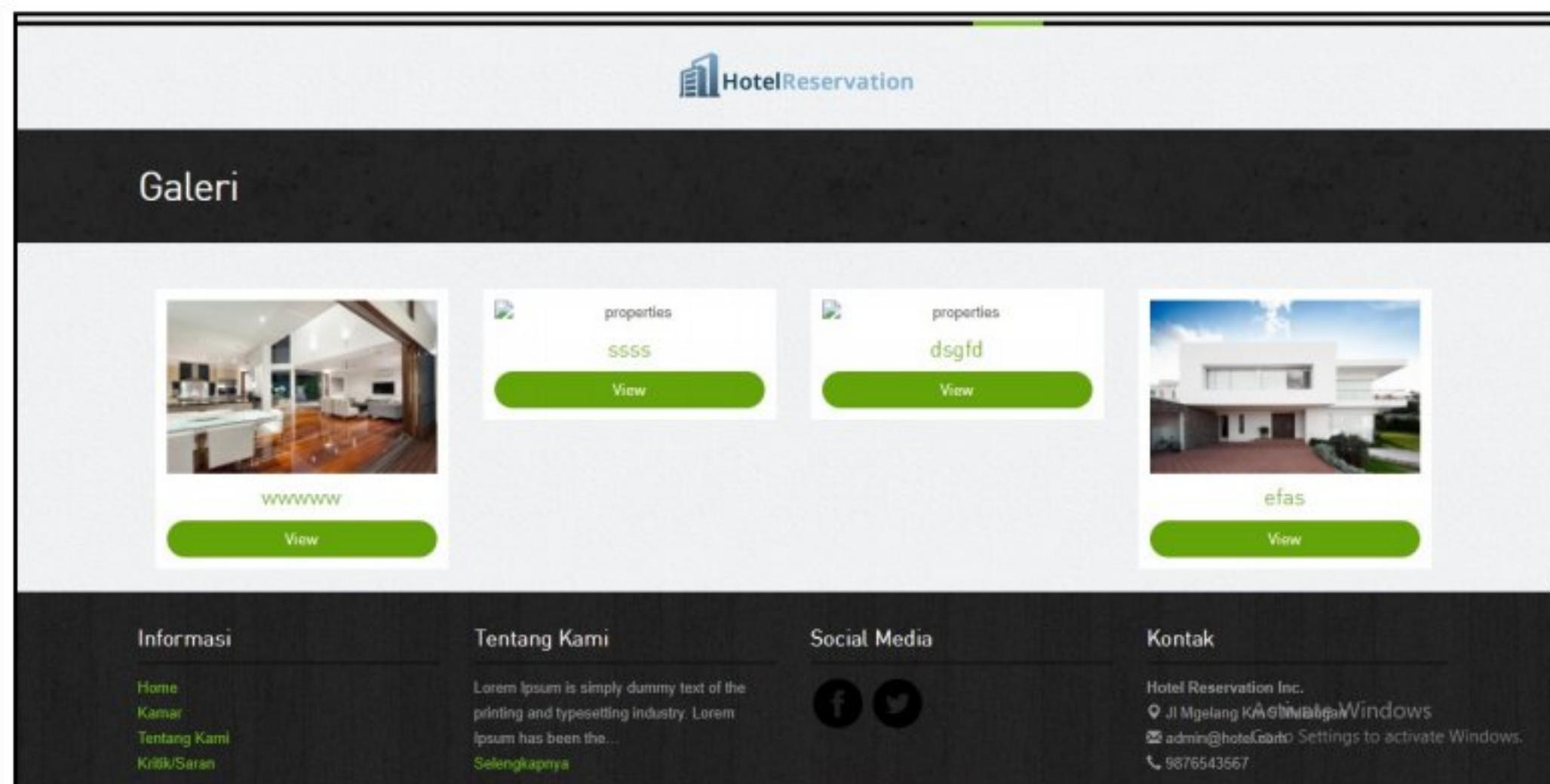
4.3.2 Halaman Daftar Kamar



Gambar 4.6 Halaman Daftar Kamar

Halaman ini memuat daftar kamar yang kami punya secara lengkap, serta dilengkapi dengan informasi mengenai kamar yang disediakan dan juga gambar mengenai kamar tersebut.

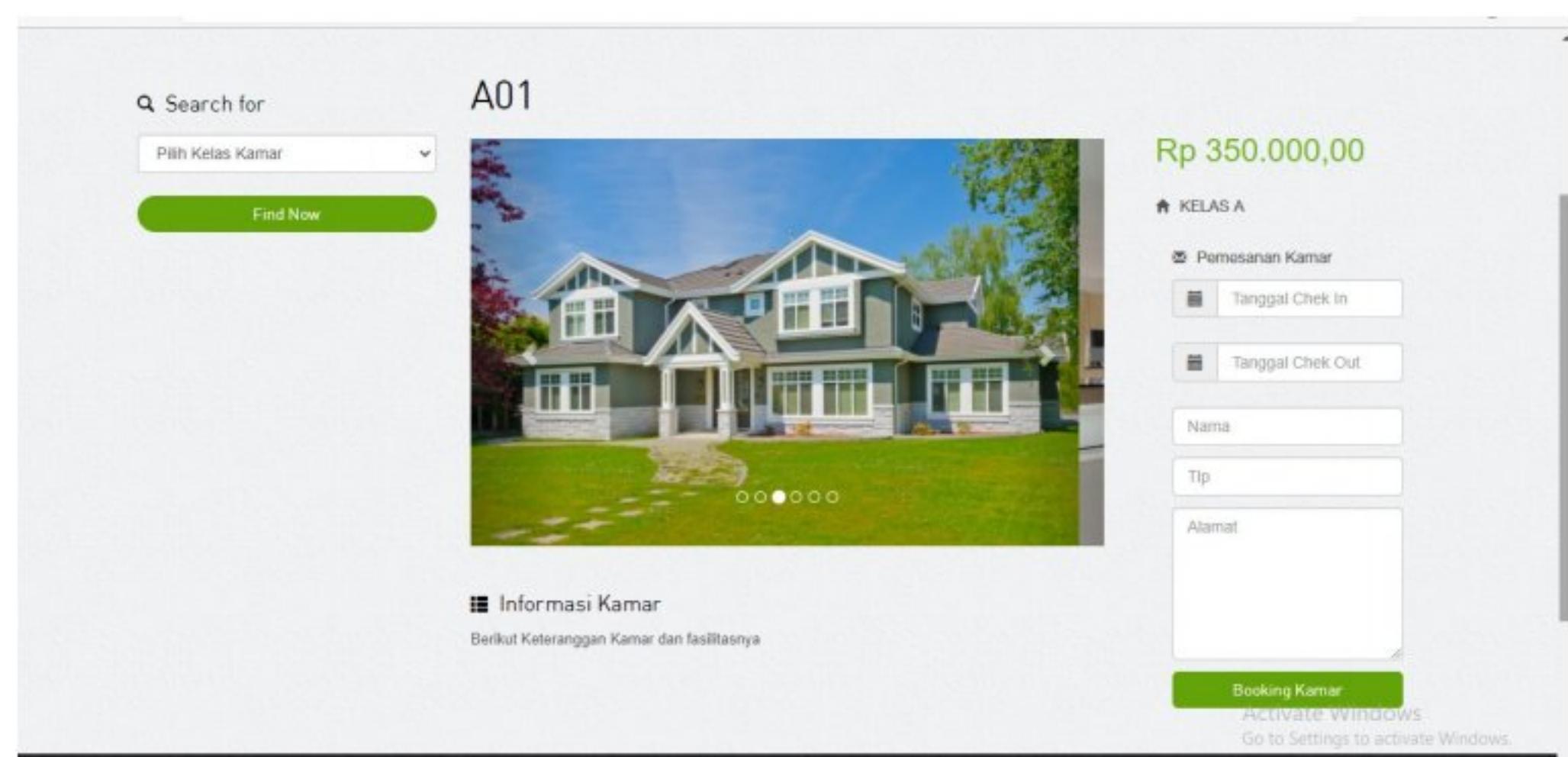
4.3.3 Halaman Galeri



Gambar 4.7 Halaman Galeri

Halaman ini berisi tentang properti yang dimiliki oleh tiap pemilik properti yang terdaftar di database kami.

4.3.4 Halaman penyewaan



Gambar 4.8 Halaman Penyewaan

Halaman ini berisi form penyewaan kamar yang telah kami tawarkan pada website

Kesimpulan dan Saran

4.4. Kesimpulan

- a. **Pemrograman berorientasi objek (Object Oriented Programming)** atau disingkat OOP) adalah paradigma pemrograman yang berorientasikan kepada objek yang merupakan suatu metode dalam pembuatan program, dengan tujuan untuk menyelesaikan kompleksnya berbagai masalah program yang terus meningkat. Objek adalah *entitas* yang memiliki atribut, karakter (*bahaviour*) dan kadang kala disertai kondisi (*state*) (Douglas, 1992).
- b. *PHP* atau kependekan dari *Hypertext Preprocessor* adalah salah satu bahasa pemrograman *open source* yang sangat cocok atau dikhkusukan untuk pengembangan *web* dan dapat ditanamkan pada sebuah skripsi *HTML*. Bahasa *PHP* dapat dikatakan menggambarkan beberapa bahasa pemrograman seperti *C, Java, dan Perl* serta mudah untuk dipelajari. *PHP* merupakan bahasa *scripting server – side*, dimana pemrosesan datanya dilakukan pada sisi *server*. Sederhananya, serverlah yang akan menerjemahkan *skrip* program, baru kemudian hasilnya akan dikirim kepada *client* yang melakukan permintaan. Adapun pengertian lain *PHP* adalah *akronim* dari *Hypertext Preprocessor*, yaitu suatu bahasa pemrograman berbasiskan *kode – kode (script)* yang digunakan untuk mengolah suatu data dan mengirimkannya kembali ke *web browser* menjadi *kode HTML*".

- c. Kelas (*class*). Kelas (*class*) merupakan penggambaran satu *set* objek yang memiliki atribut yang sama. Kelas mirip dengan tipe data ada pemrograman non objek, akan tetapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya. Kelas baru dapat dibentuk lebih spesifik dari kelas ada umumnya. Kelas merupakan jantung dalam pemrograman *berorientasi objek*.
- d. Objek (*Object*). Objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan teknik yang efektif dalam menemukan cara yang tepat dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak. Orientasi objek merupakan teknik pemodelan sistem *riil* yang berbasis objek. Objek adalah entitas yang memiliki *atribut*, karakter dan kadang kala disertai kondisi. Objek mempresentasikan sesuai kenyataan seperti siswa, mempresentasikan dalam bentuk konsep seperti merek dagang, juga bisa menyatakan visualisasi seperti bentuk huruf (*font*).
- e. Abstraksi (*Abstraction*). Kemampuan sebuah program untuk melewati aspek informasi yang diolah adalah kemampuan untuk fokus pada inti permasalahan. Setiap objek dalam sistem melayani berbagai model dari pelaku abstrak yang dapat melakukan kerja, laporan dan perubahan serta berkomunikasi dengan objek lain dalam sistem, tanpa harus menampakkan kelebihan diterapkan.

4.4.1 Saran

Saran yang dapat diberikan pada praktikum ini adalah sebagai berikut :

- a. Bagi praktikan agar mempelajari terlebih dahulu materi yang akan di praktikumkan.
- b. Bagi asisten agar membimbing praktikan dengan baik dan menjelaskan materi lebih baik lagi.
- c. Untuk laboratorium agar dapat bekerja sama dengan baik.

DAFTAR PUSTAKA

- Aziz, M. Farid. 2005. *Object Oriented Programming dengan PHP5*, Jakarta: PT Elex Media Komputindo.
- Frank J. 2016. *Principles of Programming JAVA Level 1*. Xlibris US.
- Hermawan B. 2004. *Menguasai JAVA 2 & Object Oriented Programming*, 1e. Yogyakarta: Andi Offset.
- Hollander, Anita S., Denna, Eric L., Cherrington, J. Owen. (2000). Accounting, *Information Technology, and Business Systems Solutions (2nd ed.)*. New York: Irwin McGraw-Hill.
- Rohman, A. 2014. Mengenal Framework “*Laravel*” (Best PHP Frameworks For 2014). ilmuit.org.
- Sidik, Betha. 2012. *Pemrograman Web dengan PHP*, Bandung: Informatika.
- Sutaji, Deni. 2012. *Sistem Inventory Mini Market dengan PHP & JQUERY*, Yogyakarta: Lokomedia.
- Wu, C. Thomas. 2001. An Introduction to Object-Oriented Programming with Java. Nort America: McGraw Hill
- Yasin V. 2012. Rekayasa Perangkat Lunak Berorientasi Objek, 1e. Jakarta: Mitra Wacana Media.