

**LAPORAN LENGKAP**  
**PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK**



**OLEH :**

**NAMA : HERTIN ODE**  
**NIM : F1G120040**  
**KELAS : B (GENAP)**

**ASISTEN PENGAMPUH :**  
**WAHID SAFRI JAYANTO**

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

**2021**

## HALAMAN PENGESAHAN

### LAPORAN LENGKAP



OLEH :

NAMA : HERTIN ODE

NIM : F1G120040

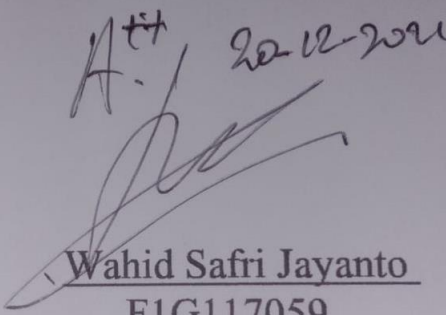
KELAS : B (GENAP)

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

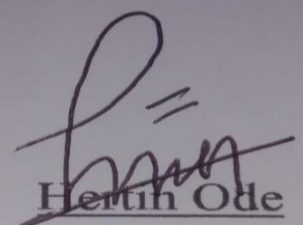
Kendari, 20 Desember 2021

Menyetujui

Asisten Praktikan

Ast 20-12-2021  
  
Wahid Safri Jayanto  
F1G117059

Praktikan

  
Hertin Ode  
F1G120040

## KATA PENGANTAR



Puji syukur kita panjatkan kehadiran Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga saya dapat menyelesaikan tugas yang berjudul “*Oop, Laravel, Xampp, Composer, Class Object dan Method*” tepat pada waktunya.

Adapun tujuan dari penulisan laporan ini adalah untuk memenuhi tugas akhir pada praktikum Pemrograman Berorientasi Objek (*PBO*). Selain itu, laporan ini juga bertujuan untuk menambah wawasan bagi para pembaca dan juga bagi penulis.

Saya mengucapkan terima kasih kepada Kak **Wahid Safri Jayanto** selaku asisten pembimbing yang telah memberikan tugas ini sehingga dapat menambah wawasan dan pengetahuan pada mata kuliah *PBO* ini.

Saya juga mengucapkan terima kasih kepada semua pihak yang tidak dapat saya sebutkan semua, terima kasih atas bantuannya sehingga saya dapat menyelesaikan laporan lengkap ini.

Saya menyadari, laporan yang saya tulis ini masih jauh dari kata sempurna. Oleh karena itu, Saya sangat mengharapkan kritik dan saran yang membangun demi kesempurnaan laporan ini.

Kendari, 20 Desember 2021

HERTIN ODE

## DAFTAR ISI

	Halaman
COVER .....	i
HALAMAN PENGESAHAN.....	ii
KATA PENGANTAR .....	iii
DAFTAR ISI.....	iii
DAFTAR GAMBAR .....	vi
DAFTAR TABEL.....	vii
PRAKTIKUM 1 .....	8
1.1 Pendahuluan .....	8
1.2 Waktu dan Tempat .....	9
1.3 Alat dan Bahan .....	9
1.4 Pengenalan <i>PBO</i> .....	9
1.5 Pengenalan <i>Object Oriented Programing (OOP)</i> .....	11
1.6 Pengenalan <i>PHP</i> .....	11
PRAKTIKUM 2 .....	14
2.1 Pengertian <i>Class, Object, Property, dan Method</i> .....	14
2.1.1 <i>Class</i> .....	14
2.1.2 <i>Property</i> .....	14
2.1.3 <i>Object</i> .....	15
2.2 Enkapsulasi ( <i>Encapsulation</i> ) .....	16
2.3 <i>Protected</i> .....	17
2.4 <i>Private</i> .....	18

2.5 <i>Constructor</i> dan <i>Destructor</i> .....	18
2.5.1 <i>Constructor</i> .....	18
2.5.2 <i>Destructor</i> .....	18
2.6 <i>Interface</i> .....	19
PRAKTIKUM 3 .....	20
3.1 Pengertian <i>CRUD</i> .....	20
3.2 <i>Project CRUD</i> .....	22
3.2.1 Halaman Utama.....	22
3.2.2 Halaman <i>Member</i> .....	22
3.2.3 Halaman <i>Manager</i> .....	23
PRAKTIKUM 4 .....	24
4.1 Pendahuluan .....	24
4.2 <i>ERD (Entity Relationship Model)</i> .....	24
4.3 Data Flow Diagram ( <i>DFD</i> ) .....	25
4.4 <i>User Interface</i> .....	28
4.4.1 Halaman Utama.....	28
4.4.2 Halaman Daftar Harga .....	28
4.4.3 Halaman Pendaftaran .....	29
4.4.4 Halaman <i>Login</i> .....	30
4.5 Kesimpulan.....	31
4.6 Saran.....	32
DAFTAR PUSTAKA .....	33

## DAFTAR GAMBAR

	Halaman
<b>Gambar 3.1</b> Halaman Utama .....	22
<b>Gambar 3.2</b> Halaman <i>Member</i> .....	22
<b>Gambar 3.3</b> Halaman <i>Manager</i> .....	23
<b>Gambar 4.1</b> <i>Erd</i> Kos_kosan.....	25
<b>Gambar 4.2</b> Diagram Konteks .....	26
<b>Gambar 4.3</b> Diagram level 1.....	27
<b>Gambar 4.4</b> Halaman Utama .....	28
<b>Gambar 4.5</b> Halaman Daftar Harga.....	28
<b>Gambar 4.6</b> Halaman Pendaftaran.....	29
<b>Gambar 4.7</b> Halaman <i>Login</i> .....	30

## DAFTAR TABEL

	Halaman
<b>Tabel 1.1</b> Alat dan Bahan .....	9

# PRAKTIKUM 1

## 1.1 Pendahuluan

Secara garis besar, bahasa pemrograman komputer adalah sebuah alat yang dipakai oleh para *programmer* komputer untuk menciptakan program aplikasi yang digunakan untuk berbagai macam keperluan. Pada tahap awal dikenal beberapa jenis bahasa pemrograman, bahasa ini berbasis teks dan berorientasi linear contohnya: Bahasa *BASIC*, Bahasa *Clipper*, Bahasa *Pascal*, Bahasa *cobol*. (Wibowo Kadek,2015)

*Object Oriented Programming (OOP)* merupakan pengembangan dari pemrograman prosedural, *OOP* merupakan model pemrograman menggunakan obyek. *OOP* menggunakan *class* dan obyek yang dapat digunakan berulang ulang, apabila ada pengembangan sebuah sistem tinggal membuat obyek baru, sehingga ketika terjadi permasalahan lebih mudah untuk mengatasinya. *OOP* sangat efektif untuk mengembangkan sistem yang kompleks. *OOP* berbasis obyek, fungsi fungsi yang ada dibagi dalam beberapa *class*, sehingga penanganannya menjadi lebih mudah, selain itu apabila terjadi *bug* pada program, dapat lebih mudah memnemukan dan memperbaikinya.( April Firman,2017)

*PHP (Hypertext preprocessor)* yaitu bahasa pemrograman *web server-side* yang bersifat *open source*. *PHP* merupakan *script* yang terintegrasi dengan *HTML* dan berada pada server (*server side HTML embedded scripting*). *PHP* adalah *script* yang digunakan untuk membuat halaman website yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Mekanisme ini menyebabkan informasi yang diterima client selalu baru



atau *up to date*. Semua *script* PHP dieksekusi pada server dimana *script* tersebut dijalankan. (Joni Karman,2018)

## 1.2 Waktu dan Tempat

Kegiatan praktikum Pemograman Berorientasi Objek ini dimulai dari tanggal 30 September sampai 2 Desember dilaksanakan setiap hari kamis pukul 10.00-12.00 WITA di Laboratorium Aljabar lantai 3 gedung A, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Halu Oleo, Kendari.

## 1.3 Alat dan Bahan

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

Alat Dan Bahan	Penjelasan
Leptop	Sebagai tempat untuk menyimpan data, untuk mengerjakan projek dan sebagai tempat untuk mengoding.
<i>Xampp</i>	Sebagai penghubung antara <i>chrome</i> dan sublime.
<i>Sublime</i>	Sebagai tempat mengoding sebuah program.
<i>Chrome</i>	Sebagai tempat untuk melihat hasil <i>running</i> dari program yang telah dibuat.

**Tabel 1.1** Alat dan Bahan

## 1.4 Pengenalan PBO

Pemrograman Berorientasi Objek (*Object Oriented Programming*) merupakan paradigma pemrograman yang berorientasikan kepada 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. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Pada jaman sekarang, banyak bahasa pemrograman yang mendukung *OOP*.

*OOP* adalah paradigma pemrograman yang cukup dominan saat ini, karena mampu memberikan solusi kaidah pemrograman modern. Meskipun demikian, bukan berarti bahwa pemrograman prosedural sudah tidak layak lagi. *OOP* diciptakan karena dirasakan masih adanya keterbatasan pada bahasa pemrograman tradisional. Konsep dari *OOP* sendiri adalah semua pemecahan masalah dibagi ke dalam objek. Dalam *OOP* data dan fungsi-fungsi yang akan mengoperasikannya digabungkan menjadi satu kesatuan yang dapat disebut sebagai objek. Proses perancangan atau desain dalam suatu pemrograman merupakan proses yang tidak terpisah dari proses yang mendahului, yaitu analisis dan proses yang mengikutinya. Pembahasan mengenai orientasi objek tidak akan terlepas dari konsep objek seperti *inheritance* atau penurunan, *encapsulation* atau pembungkusan, dan *polymorphism* atau kebanyakan. Konsep-konsep ini merupakan fundamental dalam orientasi objek yang perlu sekali dipahami serta digunakan dengan baik, dan menghindari penggunaannya yang tidak tepat.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung *OOP* mengklaim bahwa *OOP* lebih

mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan *OOP* lebih mudah dikembangkan dan dirawat.

### **1.5 Pengenalan *Object Oriented Programing (OOP)***

*OOP* adalah singkatan dari *Object Oriented Programming*. *OOP* merupakan metode pemrograman yang lebih berorientasi pada objek. maksudnya pemrograman yang lebih terpusat pada objek. sehingga akan lebih sangat memudahkan kita didalam membuat aplikasi sebenar nya *OOP* lebih di dukung pada pemrograman JAVA dan C--. tetapi di *PHP* sudah sangat didukung pada versi PHP5.

### **1.6 Pengenalan *PHP***

Bagi yang belum tahu, *PHP* singkatan dari *Hypertext Preprocessor*. Sedangkan pengertian dari *PHP* adalah bahasa pemrograman sisi server yang digunakan dalam pembuatan situs web bersama dengan *CSS* dan *HTML*. *PHP* mengubah situs web dari statis menjadi lebih dinamis dan mengubah konten serta fungsi situs web yang lebih interaktif untuk keperluan pengguna.

*PHP* merupakan bahasa pemrograman yang populer hingga saat ini mengalahkan beberapa bahasa pemrograman lainnya, termasuk *ASP.NET*. hasil Berdasarkan survei dari *W3Techs.com*, *PHP* mendapatkan prosentase 78,9% mengalahkan bahasa pemrograman lainnya. Tentu ini prosentase yang besar jika dibandingkan dengan lainnya. Memang secara fungsi *PHP* bukan yang terbaik jika dibandingkan dengan pemrograman web lainnya, tetapi secara pengguna *PHP* masih menjadi nomor satu.

Menurut sejarahnya, *PHP* pertama kali muncul tahun 1994 diciptakan oleh Dr Leonardo Bernart. Awalnya *PHP* memiliki singkatan “Personal Home Page Tools”, selanjutnya *PHP* diganti nama menjadi FI (Form Interpreter). Sejak kemunculan *PHP* versi 3.0, nama *PHP* kembali digunakan dengan singkatan menjadi “*Hypertext Preprocessor*” hingga sekarang ini.

Pada survei yang dilakukan bulan Desember tahun 1999, sudah ada lebih dari sejuta *website* yang menggunakan *PHP* termasuk diantaranya *website NASA*, *RedHat* dan *Mitsubishi*. Untuk sekarang ini *website* yang menggunakan *PHP* sudah terhitung lagi jumlahnya.

Dalam pembuatan situs web, sebenarnya dengan menggunakan *HTML* dan *CSS* saja sudah bisa menjadi situs web, tetapi situs web yang dibuat bersifat statis. Nah dengan menggunakan beberapa fungsi yang ada di *PHP*, *website* bisa berubah menjadi dinamis. Fungsi yang ada dalam *PHP* biasa disebut *CRUD*, *CRUD* kepanjangan dari *Create*, *Read*, *Update* dan *Delete*. Berikut penjelasan lengkapnya

- 1.) *Create* adalah fungsi yang digunakan untuk membuat data baru dalam *website*. Contoh saat Anda melakukan registrasi baru ke *website*, nah inilah yang membuat data baru.
- 2.) *Read* adalah fungsi yang digunakan untuk membaca atau bisa juga menampilkan data yang berada di database. Kemudian akan ditampilkan sesuai dari permintaan pengguna.
- 3.) *Update* adalah fungsi untuk melakukan edit data dari dalam *database*. Contoh saat melakukan edit profil pengguna.

- 4.) *Delete* adalah fungsi yang digunakan untuk menghapus database. Contoh Anda menghapus profil, komentar dan tindakan lainnya.

## PRAKTIKUM 2

### 2.1 Pengertian *Class*, *Object*, *Property*, dan *Method*

#### 2.1.1 *Class*

Kelas (*Class*) merupakan kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh “*class of dog*” adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku atau turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah *class* secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah *class* sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan *OOP*). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

Contoh *Syntax*:

```
class Mahasiswa {  
    String nim;           //deklarasi variabel atau  
    atribut  
    String nama;         //deklarasi variabel atau  
    atribut  
}
```

#### 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 hanyalah variabel yang terletak di dalam *class*. Semua aturan dan tipe data yang biasa dimasukkan ke dalam variabel, bisa juga dimasukkan ke dalam properti. Aturan tata cara penamaan properti sama dengan variabel aturan penamaan.

Contoh *Sintax*:

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

### 2.1.3 *Object*

*Object* atau Objek adalah *hasil cetak* dari kelas , atau hasil '*konkrit*' dari kelas . Jika menggunakan analogi *class* laptop , maka objek dari *class* laptop bisa berupa: *laptop\_andi*, *laptop\_anto*, *laptop\_duniaikom*, dan lain-lain. Objek dari kelas laptop akan memiliki seluruh ciri-ciri laptop , yaitu *property* dan *method* -nya.

Proses '*mencetak*' objek dari kelas ini disebut dengan '*instansiasi*' (atau *instantiation* dalam bahasa *inggris* ). Pada *PHP*, proses *instansiasi* dilakukan dengan menggunakan kata kunci '*baru*'. Hasil kelas cetakan akan disimpan dalam *variabel* untuk selanjutnya digunakan dalam proses program. Sebagai contoh, berikut adalah cara

membuat objek *laptop\_andi* dan *laptop\_anto* yang dibuat dari kelas *laptop* :

Contoh *Sintax*:

```
<?php
class laptop {
    //... isi dari class laptop
}
$laptop_andi = new laptop();
```

## 2.2 Encapsulation

Enkapsulasi (*encapsulation*) adalah sebuah metoda untuk mengatur struktur *class* dengan cara menyembunyikan alur kerja dari *class* tersebut. Struktur *class* yang dimaksud adalah *property* dan *method*. Dengan enkapsulasi, kita bisa membuat pembatasan akses kepada *property* dan *method*, sehingga hanya *property* dan *method* tertentu saja yang bisa diakses dari luar *class*. Enkapsulasi juga dikenal dengan istilah ‘*information hiding*’. Dengan enkapsulasi, kita bisa memilih *property* dan *method* apa saja yang boleh diakses, dan mana yang tidak boleh diakses. Dengan menghalangi kode program lain untuk mengubah *property* tertentu, *class* menjadi lebih terintegrasi, dan menghindari kesalahan ketika seseorang ‘mencoba’ mengubahnya. *Programmer* yang merancang *class* bisa menyediakan *property* dan *method* khusus yang memang ditujukan untuk diakses dari luar. Untuk membatasi hak akses kepada *property* dan *method* di dalam sebuah *class*, *Objek Oriented Programming* menyediakan 3 kata kunci, yakni *Public*, *Protected* dan *Private*. Kata kunci ini diletakkan sebelum nama *property* atau sebelum nama *method*.



Ketika sebuah *property* atau *method* dinyatakan sebagai *public*, maka seluruh kode program di luar class bisa mengaksesnya, termasuk *class* turunan.

```
<?php

// buat class laptop class
laptop {

    // buat public property
    public $pemilik;

    // buat public method
    public function hidupkan_laptop() { return
```

### 2.3 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.

```
<?php

// buat class laptop
class laptop {

    // buat protected property
    protected $pemilik;

    protected function hidupkan_laptop() { return
        "Hidupkan Laptop";

    }

}
```

## 2.4 Private

Hak akses terakhir dalam konsep enkapsulasi adalah *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.

```
<?php
// buat class komputer
class komputer {

    // property dengan hak akses protected

    private $jenis_processor = "Intel Core i7-4790 3.6Ghz";

    public function tampilkan_processor() { return
        $this->jenis_processor;

    }

}
```

## 2.5 Constructor dan Destructor

### 2.5.1 Constructor

*Constructor* adalah *method* khusus yang akan dijalankan secara otomatis pada saat sebuah objek dibuat (instansiasi), yakni ketika perintah “*new*” dijalankan. *Constructor* biasa digunakan untuk membuat proses awal dalam mempersiapkan objek, seperti memberi nilai awal kepada *property*, memanggil *method internal* dan beberapa proses lain yang digunakan untuk mempersiapkan objek.

### 2.5.2 Destructor

*Destructor* adalah *method* khusus yang dijalankan secara otomatis pada saat sebuah objek dihapus. Di dalam PHP, seluruh objek sebenarnya

```
public function _destruct() {
    echo "Ini berasal dari Destructor Laptop";
}
```

sudah otomatis dihapus ketika halaman PHP selesai diproses. Tetapi kita juga dapat menghapus objek secara manual. *Destructor* biasanya dipakai untuk membersihkan beberapa variabel, atau menjalankan proses tertentu sebelum objek dihapus.

## **2.6 Interface**

*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 disingkat dengan ‘*Interface*’ saja. 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*.

## PRAKTIKUM 3

### 3.1 Pengertian *CRUD*

*CRUD* adalah singkatan yang berasal dari *Create*, *Read*, *Update*, dan *Delete*, dimana keempat istilah tersebut merupakan fungsi utama yang nantinya diimplementasikan ke dalam basis data. Empat poin tersebut mengindikasikan bahwa fungsi utama melekat pada penggunaan *database* relasional beserta aplikasi yang mengelolanya, seperti *Oracle*, *MySQL*, *SQL Server*, dan lain – lain.

Terdapat empat *poin* penting dari akronim fungsi *CRUD* untuk mengembangkan perangkat lunak, baik berbasis *web* maupun *mobile*. Fungsi *CRUD* yang pertama adalah *create*, dimana anda dapat memungkinkan untuk membuat *record* baru pada sistem basis data. Jika anda sering menggunakan *SQL*, maka sering disebut dengan istilah *insert*. Sederhananya, anda dapat membuat tabel atau data baru sesuai atribut dengan memanggil fungsi *create*. Akan tetapi, biasanya hanya posisi administrator saja yang dapat menambahkan atribut lain ke dalam tabel itu sendiri.

Fungsi yang kedua adalah *read*, berarti memungkinkan anda untuk mencari atau mengambil data tertentu yang berada di dalam tabel dengan membaca nilainya. Fungsi *read* mempunyai kesamaan dengan fungsi *search* yang biasa anda temukan dalam berbagai perangkat lunak.

Hal yang perlu anda lakukan adalah dengan menggunakan kata kunci (*keyword*) untuk dapat menemukan file *record* dengan bantuan *filter data* berdasarkan kriteria tertentu.

Fungsi *CRUD* yang ketiga adalah *update*, dimana berfungsi untuk memodifikasi data atau *record* yang telah tersimpan di dalam *database*. Namun, anda perlu untuk mengubah beberapa informasi terlebih dahulu agar dapat mengubah *record* sesuai kebutuhan anda. Untuk pengisian *update data* anda juga perlu menyesuaikan nilai atribut sesuai dengan *form* yang tersedia agar tidak ada kesalahan saat pemrosesan data di dalam *server*.

Fungsi yang terakhir adalah *delete*, dimana ketika anda tidak membutuhkan sebuah *record* lagi, maka data tersebut perlu untuk dihapus. Sehingga, anda perlu untuk menggunakan fungsi *delete* untuk memproses aktivitas tersebut. Beberapa *software* terkait *database* relasional mengizinkan anda untuk menggunakan *soft* dan *hard delete*. Untuk *soft delete* berfungsi untuk memperbarui status baris yang menunjukkan bahwa data akan dihapus meskipun informasi tersebut tetap ada. Sedangkan, untuk *hard delete* bertujuan untuk menghapus catatan pada basis data secara permanen.

### 3.2 Project CRUD

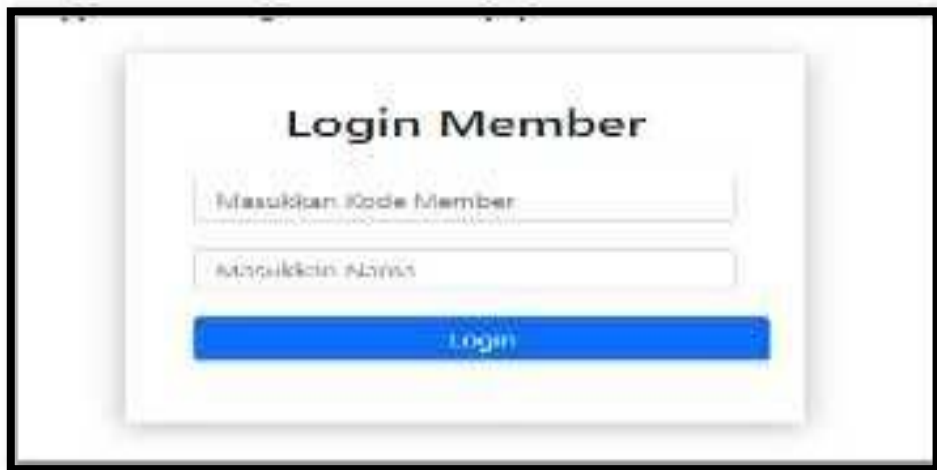
#### 3.2.1 Halaman Utama



**Gambar 3.1** Halaman Utama

Pada halaman utama ini member dan manager. Jika ingin masuk sebagai member klik tombol member dan jika ingin masuk sebagai manager tinggal klik manager.

#### 3.2.2 Halaman *Member*



**Gambar 3.2** Halaman *Member*

Pada halaman member ini terdapat login member, member tinggal memasukkan nama dan memasukkan kode member.

### 3.2.3 Halaman *Manager*



**Gambar 1.3** Halaman *Manager*

Pada halaman *manager* juga terdapat login *manager*, *manager* tinggal memasukkan *username* dan *password*.

## **PRAKTIKUM 4**

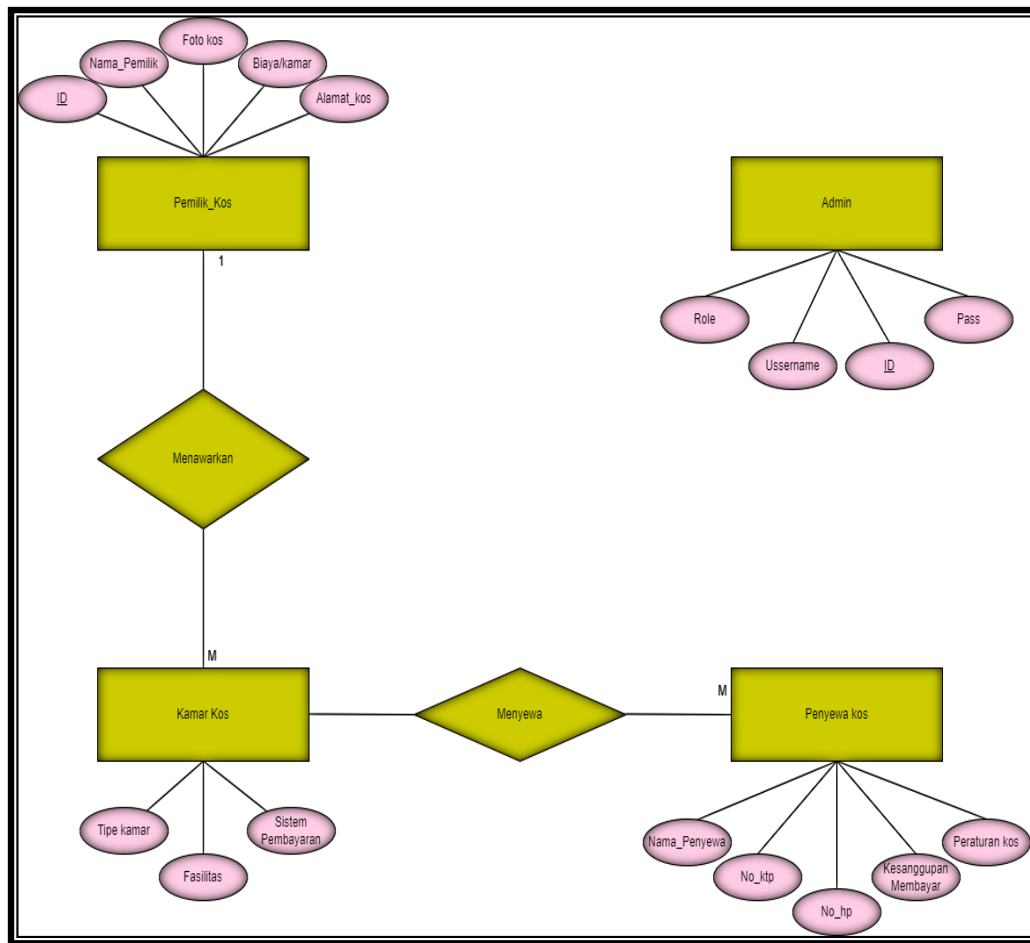
### **4.1 Pendahuluan**

Pada praktikum keempat ini kami dibimbing oleh kak wahid membuat sistem penyewaan kamar kost. Pada praktikum kali ini juga dijelaskan mengenai *erd*. Pada pembuatan system crud, pertama kita membuat model data berbasis objek terlebih dahulu, dimana fungsi dari model data ini yaitu untuk untuk menjelaskan hubungan antar data dalam basis data berdasarkan persepsi bahwa *real world* terdiri dari objek-objek dasar yang mempunyai hubungan / *relasi* antara objek tersebut.

### **4.2 ERD (Entity Relationship Model)**

*ERD (Entity Relationship Diagram)* adalah suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. *ERD* untuk memodelkan struktur data dan hubungan antar data, untuk menggambarkannya digunakan beberapa notasi dan simbol. Pada dasarnya ada tiga komponen yang digunakan, yaitu : Entitas, atribut hubungan / relasi derajat relasi atau kardinalitas rasio.





**Gambar 2.1** *Erd Kos\_kosan*

Berdasarkan gambar 4 menjelaskan model data yang digunakan yaitu *Entity Relationship model (ERD)*, Merupakan model untuk menjelaskan hubungan antar data dalam basis data berdasarkan persepsi bahwa *real world* terdiri dari objek-objek dasar yang mempunyai hubungan / relasi antara objek tersebut.

#### 4.3 Data Flow Diagram (DFD)

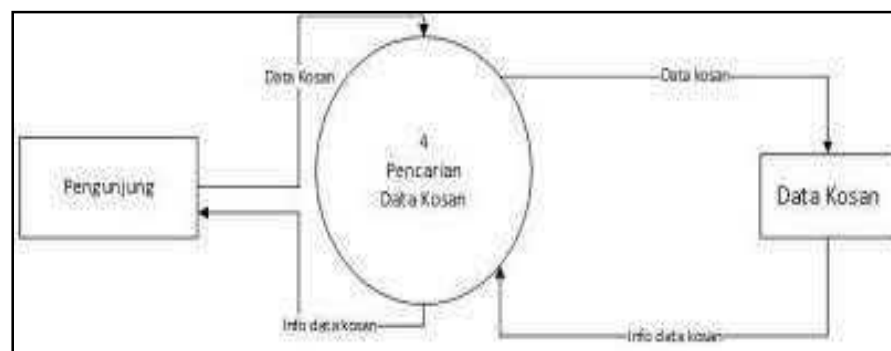
*DFD* adalah suatu diagram yang menggambarkan aliran data dari sebuah proses yang sering disebut dengan sistem informasi. Di dalam *data flow diagram* juga menyediakan informasi mengenai *input* dan *output* dari tiap entitas dan proses

itu sendiri.

Dalam diagram alir data juga tidak mempunyai kontrol terhadap *flow* - nya, sehingga tidak adanya aturan terkait keputusan atau pengulangan. Bentuk penggambaran berupa data *flowchart* dengan skema yang lebih spesifik. *Data flow diagram* berbeda dengan UML (*Unified Modelling Language*), dimana hal mendasar yang menjadi pembeda antara kedua skema tersebut terletak pada *flow* dan *objective* penyampaian informasi di dalamnya.

#### 1.) Diagram Level 0 (Diagram Konteks)

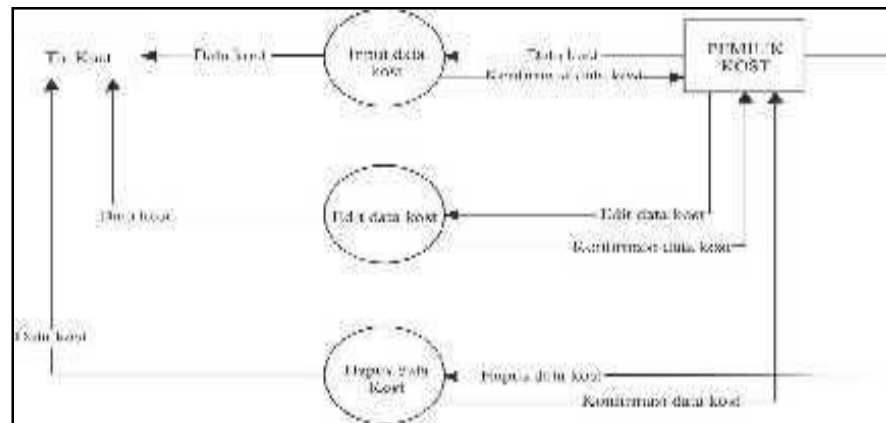
Diagram level 0 atau bisa juga diagram konteks adalah level diagram paling rendah yang menggambarkan bagaimana sistem berinteraksi dengan external entitas. Pada diagram konteks akan diberikan nomor untuk setiap proses yang berjalan, umumnya mulai dari angka 0 untuk start awal. Semua entitas yang ada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

## 2.) Data Flow 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** Diagram level 1

Ada perbedaan antara 2 level *DFD* tersebut yang perlu diketahui, berikut ini perbedaannya:

- 1). *DFD* level 0 hanya menggambarkan sistem secara basic saja.
- 2). *DFD* level 0 hanya menjelaskan aliran data dari *input* sampai *output*.
- 3.) *DFD* level 1 menggambarkan aliran data yang lebih kompleks pada setiap prosesnya yang kemudian terbentuklah data store dan aliran data.
- 4.) *DFD* level 1 menggambarkan sistem secara sebagian atau seluruhnya secara mendetail.

#### 4.4 User Interface

Disini saya akan menjelaskan bagaimana proyek saya yang berjudul kost Putri,disini juga saya akan menampilkan gambar beserta keterangan dan *Erdnya*.

##### 4.4.1 Halaman Utama

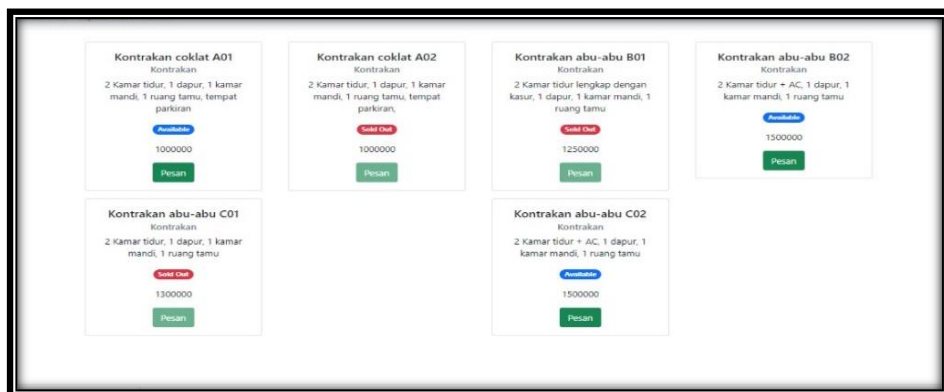


**Gambar 4.4** Halaman Utama

Keterangan:

Pada halaman utama akan menampilkan gambar dari gedung dan lokasi *kost* tersebut, dan kemudian pada sudut kanan atas tulisan login itu digunakan untuk masuk kehalaman berikutnya.

##### 4.4.2 Halaman Daftar Harga



**Gambar 4.5** Halaman Daftar Harga

Keterangan:

Pada halaman ini pelanggan akan disuguhkan dengan berbagai macam harga dan fasilitas kamar *kost* dan pada halaman ini pula pelanggan akan menentukan kamar yang akan diambil sesuai dengan *budget* yang dimiliki.

#### 4.4.3 Halaman Pendaftaran

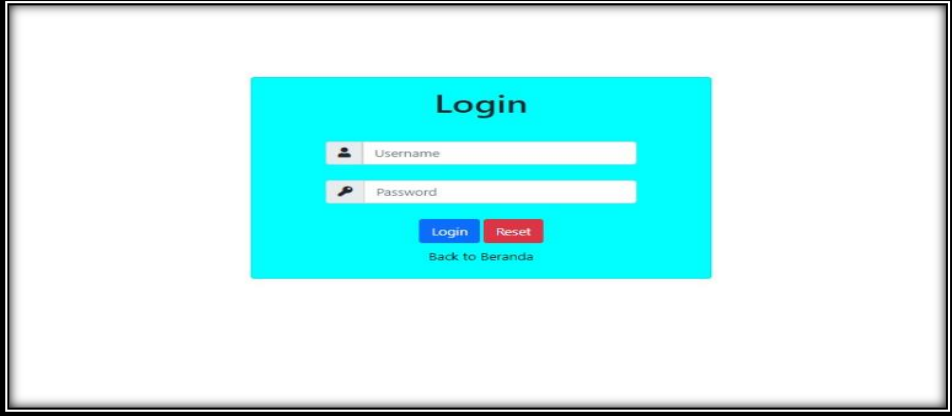
A screenshot of a registration form. It contains three input fields: 'Nama' (Name), 'Email', and 'Comments'. The 'Comments' field has a small icon in the bottom right corner. Below the 'Comments' field is a blue 'Submit' button. The form is enclosed in a light gray border.

**Gambar 4.6** Halaman Pendaftaran

Keterangan:

Pada halaman ini pelanggan akan mendaftarkan dirinya sebagai calon penghuni *kost*.

#### 4.4.4 Halaman *Login*



The image shows a login form centered on a light blue background. The form has a white background and contains the following elements:

- A title "Login" in bold black text.
- A "Username" input field with a user icon on the left.
- A "Password" input field with a key icon on the left.
- A blue "Login" button.
- A red "Reset" button.
- A link "Back to Beranda" below the buttons.

**Gambar 4.7** Halaman Login

Keterangan:

Pada halaman ini jika pelanggan sudah memiliki akun maka otomatis tinggal *username* dan passwordnya untuk masuk kehalaman berikutnya.

#### 4.5 Kesimpulan

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 (*behaviour*) dan kadang kala disertai kondisi (*state*) (Douglas, 1992).

*PHP* atau kependekan dari *Hypertext Preprocessor* adalah salah satu bahasa pemrograman *open source* yang sangat cocok atau dikhususkan 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 berbasis *kode – kode (script)* yang digunakan untuk mengolah suatu data dan mengirimkannya kembali ke *web browser* menjadi *kode HTML*”.

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.

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*).

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.

#### **4.6 Saran**

Dalam pembuatan program, praktikan harus memahami terlebih dahulu materi yang akan diterapkan. Agar pada saat melaksanakan praktikum tidak bingung lagi dengan materi dan proyek-proyek apa yang harus kita kerjakan dan kembangkan.



## DAFTAR PUSTAKA

Wibowo, Kadek .2015.*Pengertian Pemograman Berorientasi Objek*.

Yogyakarta

Ibrahim, Ali. 2009. *Cara Praktis Membuat Website Dinamis Menggunakan*

*Xampp. Neotekno. Jakarta*

Kusrini. 2007. *Strategi Perancangan dan Pengelolaan Basis Data. Andi Offset.*

Yogyakarta

Nugroho, Bunafit. 2005. *Database Relational dengan My-SQL. Andi Offset.*

Yogyakarta

Suyanto, M. 2003. *Startegi Periklaan pada Sistem Crud Perusahaan Top Dunia.*

*Andi Offset. Yogyakarta*

Usdiyanto, Riyeke. 2001. *Framework Crud. Andi Offset. Yogyakarta*