

LAPORAN LENGKAP
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK



OLEH :
NAMA : VALENTINE EMAN PUTRI
NIM : F1G120011
KELOMPOK : I (SATU)

ASISTEN PENGAMPU :
WAHID SAFRI JAYANTO (F1G117059)

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

HALAMAN PENGESAHAN

LAPORAN PRAKTIKUM



OLEH :

NAMA : VALENTINE EMAN PUTRI

NIM : F1G120011

Laporan praktikum pemrograman berorientasi objek (PBO) ini disusun untuk memenuhi tugas akhir menyelesaikan kegiatan praktikum pemrograman berorientasi objek (PBO), dan disusun sebagai salah satu syarat lulus mata kuliah pemrograman berorientasi objek (PBO).

Kendari, 19 Desember 2021

Menyetujui :

Asisten Praktikum

Praktikan

18-12-2021

A handwritten signature in black ink, appearing to read "Valentine Eman Putri".

(Wahid Syafri Jayanto)
F1G117059

(Valentine Eman Putri)
F1G120011

KATA PENGANTAR



Puji syukur kami panjat kankehadirat Allah SWT, karena berkat rahmat dan hidayah-nya penyusunan laporan Pemrograman beriorientasi objek dapat di selesaikan dengan tepat waktu tanpa ada halangan yang berarti.

Laporan ini disusun berdasarkan kebutuhan mahasiswa. Dengan demikian, Materi yang dibahas dalam laporan ini sudah selesai dengan kebutuhan mahasiswa. Materi yang kami susun dalam laporan ini kami susun dengan sistematik yang baik dan jelas di tulis dengan bahasa yang mudah dimengerti dan dipahami.

Akhir kata, kami menyadari juga laporan ini tidak lepas dari kekurang. Oleh karenaitu, kami mengharap kritik dan saran dari pengguna laporan ini. Sekian terimakasih,

*wabillahitaufikwalhidayah, WassalamuAlaikum Warahumatullahi
Wabarakatu.*

Kendari, Desember 2021

Penulis

DAFTAR ISI

HALAMAN COVER.....	i
HALAMAN PENGESAHAN.....	ii
KATA PENGANTAR.....	iii
DAFTAR ISI.....	iv
1.1 Pertemuan Pertama (1).....	7
1.1.1 Alat dan bahan.....	7
1.1.2 Pengenalan <i>PBO</i>	7
1.1.3 Pengenalan <i>PHP</i>	9
2.1 Pertemuan ke dua (2).	13
2.1.1 <i>Class</i>	13
2.1.2 <i>Method</i>	13
2.1.3 <i>Property</i>	15
2.1.4 <i>Object</i>	15
2.1.5 <i>Constructor</i>	16
2.1.6 <i>Laravel</i>	17
3.1 Pertemuan ketiga (3)	18
3.1.1 Data Flow Diagram	18
3.1.2 <i>Interface</i>	20
3.1.3 <i>Project CRUD (Create,Read,Update,Delete)</i>	21
4.1 Pertemuan ke empat (4)	26
4.1.1 Pengantar <i>Project</i>	26
4.1.2 ERD (Entity Relationship Diagram).....	4
4.1.3 DFD	26
4.1.4 <i>Interface</i>	28
4.1.5 <i>Project</i>	1
DAFTAR PUSTAKA	8

DAFTAR GAMBAR

Gambar 3. 1 Tampilan Home.....	22
Gambar 3. 2 Tampilan Login Member	22
Gambar 3. 3 Halaman Mengedit Data.....	23
Gambar 3. 4 Halaman Data User	23
Gambar 3. 5 Tampilan Halaman Login Admin	24
Gambar 3. 6 Tampilan Halaman Admin (Manager).....	24
Gambar 3. 7 Halaman Menambah Golongan.....	25
Gambar 3. 8 Halaman Menambah Data Member	25
Gambar 4.1 <i>ERD Kos</i>	27
Gambar 4 .2 <i>DFD Kos</i>	29
Gambar 4. 3 Halaman Utama	4
Gambar 4. 4 Tampilan <i>Login</i>	5
Gambar 4. 5 Halaman Admin	5
Gambar 4. 6 Tampilan Pemilik Kamar Kos	6
Gambar 4. 7 Tampilan Penyewa kamar Kos.....	6

DAFTAR TABEL

Tabel 1.1Tabel penggunaan alat dan bahan	7
--	---

1.1 Pertemuan Pertama (1)

1.1.1 Alat dan bahan.

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

Alat Dan Bahan	Penjelasan
<i>Laptop</i>	Sebagai tempat untuk menyimpan data, untuk mengerjakan projek dan sebagai tempat untuk mengoding.
<i>Xampp</i>	Sebagai penghubung antara <i>chrome</i> dan <i>Visual Studio Code</i> .
<i>Visual Studio Code</i>	Sebagai tempat mengoding sebuah program.
<i>Chrome</i>	Sebagai tempat untuk melihat hasil <i>running</i> dari program yang telah di

Tabel 1.1 Tabel penggunaan alat dan bahan

1.1.2 Pengenalan *PBO*

Menurut *wikipedia*, Pemrograman berorientasi objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Dalam pemrograman berbasis objek, kita dituntut untuk memahami dan memetakan masalah kedalam class serta memecah masalah kedalam class-class yang lebih kecil dan simpel agar solusi yang dibuat lebih spesifik. Selanjutnya,

class-class tersebut akan saling berkomunikasi dan berkolaborasi untuk memecahkan masalah yang kompleks. *Class-class* ini nantinya akan dirubah menjadi objek-objek pada saat runtime.

Setiap *class* dalam *OOP* mempunyai *method* atau fungsi serta *property*. *Method* dalam *class* secara mudah diartikan sebagai segala kemampuan dari *class* atau apa saja yang dapat dilakukan oleh sebuah *class*. Sedangkan *property* adalah segala sesuatu yang dimiliki oleh *class*. Dalam *OOP*, *property* dan *method* dalam *class* saling bekerjasama membangun sebuah solusi dari suatu masalah. Dalam beberapa referensi, *method* disebut juga sebagai *function* sedangkan *property* sering disebut juga sebagai *attribute*. Sehingga Anda tidak perlu bingung bila nantinya dibuku lain, Anda menemui istilah *function* dan *attribute* sebagai pengganti *method* dan *property*. Yang perlu Anda pahami bahwa *function* atau *method* adalah fitur atau kemampuan dari sebuah *class* sedangkan *property* atau *attribute* adalah segala sesuatu yang dimiliki oleh sebuah *class*.

Kelebihan Pemrograman Berbasis Objek Pemrograman berbasis objek atau biasa disebut *OOP*, memiliki banyak keunggulan dibandingkan paradigma pemrograman lainnya. Keunggulan pemrograman berbasis objek antara lain sebagai berikut:

- a. Modularitas: program yang dibuat dapat dipecah menjadi modul-modul yang lebih kecil dan nantinya digabungkan menjadi solusi yang utuh.
- b. Fleksibilitas: karena setiap solusi dibuat dalam bentuk *class*, ketika terjadi perubahan maka hanya *class* tersebut saja yang perlu dirubah.
- c. Ekstensibilitas: penambahan *method* atau *property* dapat dilakukan dengan sangat mudah.

- d. *Reuse*: *class* dapat digunakan berkali-kali untuk proyek maupun modul yang lain.
- e. Mudah dimaintain: karena setiap *class* berdiri sendiri, maka untuk memaintain *class* tersebut jauh lebih mudah.
- f. Keamanan *code*: adanya visibilitas memberikan fitur keamanan dimana *developer* lain tidak bisa dengan bebas menggunakan fitur yang ada pada sebuah objek.
- g. Waktu *development* lebih cepat: karena *reusable* otomatis dapat mempersingkat waktu pengembangan program

Selain kelebihan, pemrograman berbasis objek juga mempunyai kekurangan antara lain sebagai berikut: *Learning curve* yang lumayan tinggi, ukuran program jauh lebih besar dan pemakaian *memory* lebih besar.

1.1.3 Pengenalan PHP

- a. Sejarah Bahasa Pemrograman PHP

Sejarah Bahasa Pemrograman PHP Menurut *wikipedia*, 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. I. (Triwansyah yuliano, 2007).

Pengenalan *PHP* 20 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 meng-install *software server*. Pada saat buku ini ditulis, *PHP* telah mencapai versi 7.2 dengan penambahan ekstensi dan perbaikan performa yang menjanjikan. Berikut adalah info grafis tentang sejarah dan perkembangan *PHP* serta ekosistemnya dari awal hingga tahun 2015 yaitu ketika *PHP* 7 atau yang juga

dikenal dengan *PHP Next Generation (PHPNG)* dirilis publik. (Triwansyah yuliano, 2007).

b. Kelebihan Bahasa Pemograman *PHP*

Sebagai bahasa pemrograman, *PHP* memiliki banyak kelebihan antara lain:

1. Komunitas yang besar

Tidak dapat dipungkiri bahwa komunitas *PHP* sangat besar dan tersebar diseluruh dunia. Di Indonesia saja ada banyak komunitas yang berafiliasi dengan *PHP* baik itu pembahasan *PHP* secara umum maupun pembahasan secara khusus misalnya tentang *framework*. Di facebook ada group *PHP Indonesia* yang membahas *PHP* secara umum, dan ada pula *Symfony Framework* Indonesia yang membahas secara khusus tentang *framework Symfony*. Tidak hanya itu, di Telegram, WhatsApp pun banyak bertebaran group yang membahas tentang *PHP*.

2. *Resources* yang melimpah

Dikarenakan komunitasnya yang besar, tentu saja akan berdampak pada kemudahan mencari *resources* yang berhubungan dengan *PHP* baik itu permasalahan yang sering terjadi, *library*, *software*, *CMS* hingga *framework PHP* banyak sekali bertebaran dan dengan mudah dapat ditemukan dengan googling.

3. Mudah dipelajari

PHP adalah bahasa pemrograman sejuta umat. Hampir semua orang yang pernah bergelut dengan dunia *Web Development* pasti pernah menggunakan atau setidaknya pernah sekedar mencobanya. Tutorial

untuk memulai belajar PHP pun dengan mudah ditemukan dengan mengetikkan kata kunci pada mesin pencari.

4. Simpel

PHP itu simpel. *Syntax*-nya sangat sederhana dan mudah sekali dipelajari. Saking simpelnya, untuk memulai belajar PHP kita tidak perlu melakukan setting apapun, cukup *instal* paket *software* seperti *XAMPP* atau *WAMP* maka Anda sudah dapat memulai belajar *PHP*.

5. Mudah dan murah untuk *deployment*

Untuk men-deploy program *PHP* sangatlah mudah, kita cukup meng-upload ke *server hosting* yang harga juga sangat terjangkau bahkan ada yang gratis. Dan masih banyak lagi kelebihan lainnya.

c. Kekurangan Bahasa Pemrograman *PHP*

Banyak orang yang bilang kekurangan utama *PHP* adalah bahwa *PHP* bahasa yang *weak type* dimana sebuah *variable* tidak memiliki tipe data sehingga menyulitkan ketika melakukan *debugging*. *Weak type* ini menyebabkan terjadinya *juggling* dimana sebuah *variable* yang tadinya berisi nilai *integer* misalnya dapat berubah menjadi berisi nilai *string* atau bahkan tipe data lainnya.

Selain *weak type*, *PHP* juga mempunya kekurangan lain yaitu inkonsistensi *API* (*Application Programming Interface*). *API* disini bukan *Web API* yang mengembalikan *json* tapi *API* disini adalah fungsi bawaan dari *PHP* yang menjadi *interface* atau tatap muka antara kita sebagai *developer* dan bahasa pemrograman *PHP* itu sendiri. Contoh paling mudah dari

ketidakkonsistenan *PHP* adalah dalam hal penamaan fungsi misalnya antara fungsi *substr* dan *str_replace*.

2.1 Pertemuan ke dua (2).

2.1.1 Class

Class adalah cetakan atau *blueprint* dari objek. Didalam class terdapat *property* dan *method*. Dalam OOP, *class* merupakan kerangka dasar yang harus dibuat sebelum kita membuat real *object*.

Untuk membuat sebuah *class* pada *PHP* kita menggunakan *keyword class* diikuti nama dari *class* tersebut. Sebagai contoh kita akan membuat sebuah *class* Mobil , maka kita dapat membuatnya sebagai berikut:

Contoh *syntax*:

```
<?php  
//Cara penulisan class OOP PHP - www.malasngoding.com  
class nama_class{  
    //isi dari class ini  
}  
?>
```

2.1.2 Method

Method adalah segala sesuatu yang dapat dilakukan oleh *class* atau *object*. Sama seperti *property*, *method* juga memiliki visibilitas serta dapat memiliki parameter. Parameter dapat memiliki nilai awal atau *default value*. Bila parameter tidak memiliki *default value* maka parameter tersebut dianggap sebagai mandatory parameter. Ada 4 (Empat) bagian dasar yang dimiliki metode antara lain:

- 1.Nama *metode*

2. *Tipe Objek* atau *tipe primitive* yang dikembalikan metode.
3. Daftar parameter.
4. Badan atau isi metode.

Tiga bagian pertama mengindikasikan informasi penting tentang metode itu sendiri. Dengan kata lain, nama metode tersebut=metode lain dalam program. Dalam java kita dapat memiliki metode-metode berbeda yang memiliki nama sama tetapi berbeda tipe kembalian atau daftar argumennya, sehingga bagian-bagian definisi metode ini menjadi penting. Ini disebut *overloading* metode(proses yang berlebihan pada suatu metode). Untuk menjalankan program yang memiliki sifat *polymorphism* tersebut, diperlukan suatu kemampuan *overloading*, yaitu suatu kemampuan untuk menentukan fungsi yang mana yang harus digunakan atau dijalankan jika terdapat nama fungsi yang sama. *Polimorfisme* bisa diartikan seperti kemampuan suatu *variable* untuk mengubah perangai sesuai dengan objek hasil *instansiasi* yang digunakan. *Polimorfisme* membiarkan lebih dari 1 objek dari *sub class* *sub class* dan diperlakukan sebagai objek dari *super class* tunggal.

Contoh Syntax:

```
<?php  
  
//Cara penulisan class dan property OOP PHP -  
//www.malasngoding.com  
class mobil{  
    // property oop  
    var $warna;  
    var $merk;  
    var $ukuran;  
  
    //method oop
```

2.1.3 *Property*

Property adalah sebuah variabel dapat digunakan dalam lingkup *class* atau *object*. *Property* sering disebut juga sebagai segala sesuatu yang dimiliki oleh *class*. *Property* memiliki visibilitas serta dapat memiliki nilai *default*.

Contoh *Syntax* :

```
<?php  
//Cara penulisan class dan property OOP PHP -  
www.malasngoding.com  
class mobil{  
    var $warna;  
    var $merk;  
    var $ukuran;  
}  
?>
```

2.1.4 *Object*

Object atau Objek adalah hasil cetak dari *class*, atau hasil ‘konkrit’ dari *class*. Jika menggunakan analogi *class* laptop, maka objek dari *class* laptop bisa berupa: laptop_andi, laptop_anto, laptop_duniaIlkom, dan lain-lain. Objek dari

class laptop akan memiliki seluruh ciri-ciri laptop, yaitu *property* dan *method*-nya. Proses ‘mencetak’ objek dari *class* ini disebut dengan ‘instansiasi’ (atau *instantiation* dalam bahasa inggris). Pada PHP, proses instansiasi dilakukan dengan menggunakan *keyword* ‘*new*’. Hasil cetakan *class* akan disimpan dalam variabel untuk selanjutnya digunakan dalam proses program.

Contoh *Syntax* :

```
<?php  
//Cara penulisan class dan property OOP PHP -  
www.malasngoding.com  
class mobil{  
    //isi class  
}  
$mobil = new mobil();  
?  
$laptop_andi = new laptop();  
$laptop_anto = new laptop();  
?>
```

2.1.5 Constructor

Constructor adalah sebuah *method* khusus yang dieksekusi ketika sebuah *class* diinstansiasi. *Constructor* digunakan untuk mempersiapkan object ketika *keyword* *new* dipanggil. Dalam *constructor* kita dapat melakukan apapun yang kita dapat lakukan pada method biasa namun tidak bisa mengembalikan *return value*.

Contoh Program:

```
class Kotak {  
    double panjang;  
    double lebar;  
    double tinggi;  
    //Mendefenisikan constructor dengan parameter  
    kotak(double p, double l, double t) {  
        panjang = p;  
        lebar = l;  
        tinggi = t;  
    }  
    double hitungVolume() {  
        return (panjang * lebar * tinggi)  
    }  
}  
class DemoConstructor2 {  
    public static void main(String[] args) {  
        kotak k1, k2;  
        k1 = new kotak(4, 3, 2)  
        k2 = new kotak (6, 5, 4)  
        system.out.println("volume k1 = " +  
        k1.hitungVolume())  
        system.out.println("volume k2 = " +  
        k2.hitungVolume() )  
}
```

2.1.6 *Laravel*

Laravel adalah satu-satunya *framework* yang membantu Anda untuk memaksimalkan penggunaan *PHP* di dalam proses pengembangan website. *PHP* menjadi bahasa pemrograman yang sangat dinamis, tapi semenjak adanya *Laravel*, dia menjadi lebih *powerful*, cepat, aman, dan simpel. Setiap rilis versi terbaru, *Laravel* selalu memunculkan teknologi baru di antara *framework PHP* lainnya. *Laravel* diluncurkan sejak tahun 2011 dan mengalami pertumbuhan yang cukup eksponensial. Di tahun 2015, *Laravel* adalah *framework* yang paling banyak mendapatkan bintang di *Github*. Sekarang *framework* ini menjadi salah satu yang populer di dunia, tidak terkecuali di Indonesia. *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 *projek* kecil hingga skala perusahaan kelas atas. *Laravel* mengubah pengembangan website menjadi lebih *elegan*, *ekspressif*, dan menyenangkan, sesuai dengan jargonya “*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*.

3.1 Pertemuan ketiga (3)

3.1.1 Data Flow Diagram

a. Pengertian

Diagram Arus Data atau yang sering disebut sebagai Data Flow Diagram (DFD) merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi yang dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program.

DFD sendiri juga digunakan untuk memberikan gambaran sistem secara keseluruhan hingga batasan sistem, sumber-sumber dan tujuan data, proses data, arus data dan media penyimpanan dengan memanfaatkan simbol-simbol dalam DFD. Sehingga DFD ini dapat menggambarkan analisa maupun

rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program.

b. Komponen-Komponen DFD

1) Entitas (*Entity*)

Entitas merupakan sumber data atau menjadi tujuan data yang dibedakan dengan garis panah atau keterangan yang berkaitan. Istilah lain dari entitas adalah terminator. Terdapat dua jenis terminator; yaitu terminator sumber (*source*) yang merupakan terminator yang menjadi sumber; dan terminator tujuan yang merupakan terminator yang menjadi tujuan data / informasi sistem.

2) Proses (*Process*)

Proses adalah suatu manipulasi terhadap data, menggambarkan bagian dari sistem yang mentransformasikan *input* menjadi *output*. Pemberian nama proses dilakukan dengan menggunakan kata kerja transitif (kata kerja yang membutuhkan obyek) dan setiap prosesnya diberikan nama dan nomor proses.

3) Media Penyimpanan (*Data Store*)

Komponen ini digunakan untuk membuat model sekumpulan paket data dan diberi nama dengan kata benda jamak, misalnya Buku. Data *store* ini biasanya berkaitan dengan penyimpanan-penyimpanan, seperti file atau databas, yang berkaitan dengan penyimpanan secara komputerisasi, misalnya file disket, file harddisk, file pita magnetik. Data *store* juga berkaitan dengan penyimpanan secara manual seperti buku alamat, file folder, dan agenda. Suatu data store dihubungkan dengan alur data hanya pada komponen proses, tidak komponen DAD lainnya. Alur data dibedakan menjadi dua macam yaitu

alur data dari data *store* yang berarti sebagai pembacaan atau pengaksesan data; dan alur data ke data store yang berarti sebagai pengupdatean data, seperti menambah data baru, menghapus baru, menghapus, atau mengubah/memodifikasi data. Dengan kata lain, proses alur data bertanggung jawab terhadap perubahan yang terjadi pada data *store*.

4) Arus Data (Data Flow)

Sekelompok elemen data yang berhubungan secara logis yang bergerak dari satu titik atau proses ke titik satu titik atau proses yang lain. Suatu arus data digambarkan dengan anak panah, yang menunjukkan arah menuju ke dan keluar dari suatu proses. Arus data ini digunakan untuk menerangkan perpindahan data atau paket data/informasi dari satu bagian sistem ke bagian lainnya.

3.1.2 *Interface*

Interface adalah wadah dari kumpulan *method* yang bersifat *abstrak* atau tidak memiliki implementasi. Sedangkan *method* yang didefinisikan di dalam *interface* tersebut akan diimplementasikan oleh *class* yang mengimplementasikan *interface* tersebut. *Interface* merupakan bentuk perluasan dari kelas *abstrak*. Selain *method*, *interface* juga dapat berisi sekumpulan *variable*, namun *variable* yang dideklarasikan di dalam *interface* harus bersifat final (nilainya tidak dapat diubah /konstan). Sebagai contoh : Dalam kehidupan nyata dapat diketahui ada manusia yang bekerja sebagai tentara, penyanyi, pengacara, dan sebagainya, tentunya manusia-manusia tersebut selain harus memiliki *method* standard sebagai seorang manusia, juga harus memiliki *method* yang sesuai dengan pekerjaannya. Dengan demikian untuk membuat objek manusia yang bekerja sebagai penyanyi, harus

dibuat kelas yang merupakan turunan kelas manusia yang meng-implementasikan *interface* penyanyi.

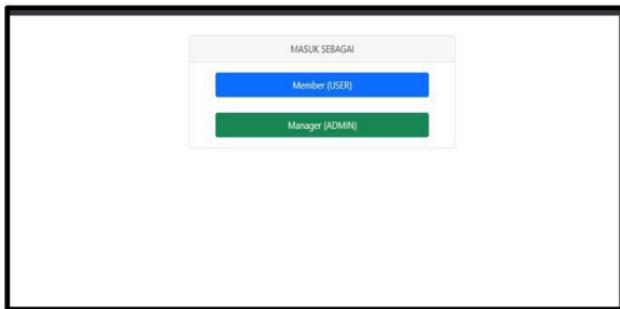
3.1.3 Project CRUD (*Create, Read, Update, Delete*).

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. Jika dihubungkan dengan tampilan antarmuka (*interface*), maka peran *CRUD* sebagai fasilitator berkaitan dengan tampilan pencarian dan perubahan informasi dalam bentuk formulir, tabel, atau laporan. Nantinya, akan ditampilkan dalam *browser* atau aplikasi pada perangkat komputer *user*. Istilah ini pertama kali diperkenalkan oleh James Martin pada tahun 1983 dalam bukunya yang berjudul “*Managing the Database Environment*”.

Secara konseptual, data diletakkan di lokasi penyimpanan sehingga konten dapat diperbarui dan dibaca. Sebelum *file* penyimpanan dibaca oleh sistem, maka lokasi perlu dibuat dan dialokasikan dengan konten. Untuk beberapa poin yang tidak diperlukan dapat dihapus agar tidak membebani sistem dan *storage* yang telah dialokasikan. Contoh penggunaan *CRUD* dalam *project*:

a. Tampilan *Home*

Tampilan pertama pada *browser* maka muncul tampilan halaman *home* yang mana pada tampilan awal ini terdapat *login member (user)* dan *login manager (admin)*. Berikut tampilan menu *Home* pada browser.

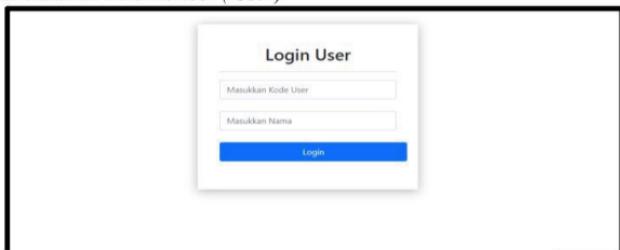


Gambar 3. 1 Tampilan *Home*

b. Tampilan *Login Member (User)*

Selanjutnya, ketika kita mengklik tulisan *member (user)* pada tampilan awal tadi, maka kita akan diarahkan ke tampilan *login user*. Pada tampilan *login user* kita bisa langsung *login* dengan cara mengisikan *username* dan *password*.

c. Tampilan Halaman Data *Member (User)*



Gambar 3. 2 Tampilan *Login Member*

Setelah melakukan *login* maka akan tampil halaman data diri dari Data *user*.

Berikut Tampilan *data user* sebagai berikut :

The screenshot shows a form titled "Data USER". It contains the following data:

Kode User	: AA001
Nama	: Nana
No Telepon	: 087654334876
Email	: nana@gmail.com
Jenis Kelamin	: laki-laki
Id Golongan	: 100
Nama Golongan	: Golongan Z

Below the table are two buttons: "Ubah Data" (in red) and "Logout" (in yellow).

Gambar 3. 3 Halaman *Data User*

Jika kita ingin mengedit data diri, maka kita hanya perlu menekan button ubah *data* pada *data user* dan akan muncul halaman sebagai berikut :

The screenshot shows a form titled "Ubah Data". It contains the following fields:

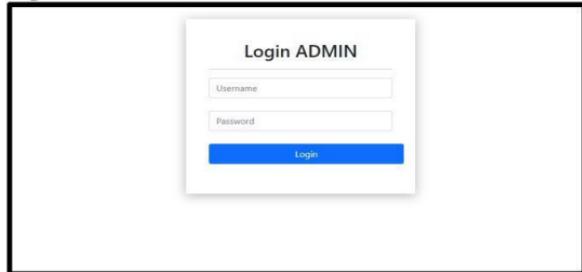
- Nama:
- Kode User:
- No Telepon:
- Email:
- Jenis Kelamin:
- Nama Golongan:

Below the fields is a "Ubah Data" button.

Gambar 3. 4 Halaman Mengedit Data

d. Tampilan Login Admin

Ketika kita sebagai *admin* maka kita bisa login dengan cara mengetikkan “*localhost/sabrina/login*. Pada tampilan *login admin* disini kita diarahkan untuk mengisi *username* dan *password* yang mana *username* dan *password* kita ambil di *database* pada tabel *admin* yang telah kita buat. Berikut tampilan halaman *login admin*.



The screenshot shows a simple login form titled "Login ADMIN". It contains two input fields: "Username" and "Password", both with placeholder text. Below the fields is a large blue rectangular button labeled "Login".

Gambar 3. 5 Tampilan Halaman *Login Admin*

e. Tampilan Halaman *Admin (Manager)*

Pada tampilan halaman manager disini kita bisa melihat keseluruhan data member kita. Saat ada member ingin mengubah datanya maka kita bisa mengedit data yang member inginkan. Ketika member keluar maka kita bisa menghapus data *member* tersebut.



The screenshot displays a table titled "Halaman Manager" under the heading "Data USER". The table has columns: No., Kode User, Name, No Telepon, Email, Jenis Kelamin, Id Golongan, Nama Golongan, and Akai. There are four rows of data. At the bottom right of the table, there are two red rectangular buttons labeled "Ubah" and "Hapus".

No.	Kode User	Name	No Telepon	Email	Jenis Kelamin	Id Golongan	Nama Golongan	Akai
1	AA001	velia	082242568791	velia97@gmail.com	perempuan	111	Golongan A	Ubah Hapus
2	AA001	Nana	087654334876	nana@gmail.com	Laki-laki	444	(Golongan D)	Ubah Hapus
3	BB001	Joeypoin	081234543210	joeyhyn14@gmail.com	Laki-laki	222	Golongan B	Ubah Hapus
4	CC001	karma	082243675611	karmar@gmail.com	perempuan	333	Golongan C	Ubah Hapus

Gambar 3. 6 Tampilan Halaman *Admin (Manager)*

f. Tampilan Menambah Golongan

Pada tampilan ini kita bisa menambahkan golongan baru sesuai yg kita inginkan. Pada tampilan menambah golongan disini kita diarahkan untuk

Tambah Golongan Baru

- Id Golongan :
- Nama Golongan :
-

[Kembali](#)

Gambar 3. 7 Halaman Menambah Golongan

mengisi Id golongan dan nama golongan. Selanjutnya, jika sudah mengisi kita bisa langsung mengklik button tambah golongan.

g. Tampilan Menambah Data Member

Pada tampilan ini kita bisa menambahkan data member baru sesuai yg kita inginkan. Pada tampilan menambah data member disini kita diarahkan untuk mengisi nama member, kode user, no. telepon, email, jenis kelamin dan golongan. Selanjutnya, jika sudah mengisi kita bisa langsung mengklik button tambah data.

Tambah Data Member

- Nama :
- Kode User :
- No Telepon :
- Email :
- Jenis Kelamin :
- Golongan :
-

[Kembali](#)

Gambar 3. 8 Halaman Menambah Data Member

4.1 Pertemuan ke empat (4)

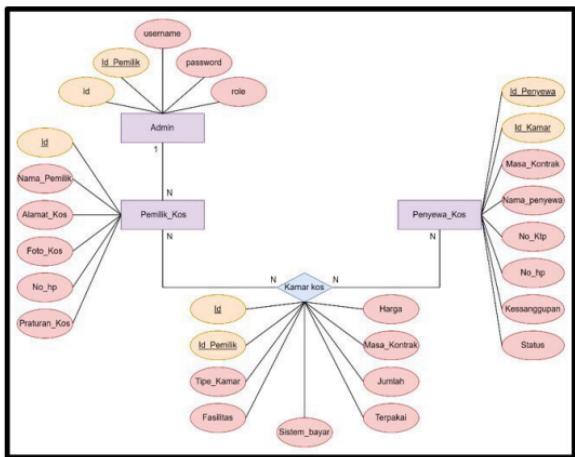
4.1.1 Pengantar Project

Pada pembuatan *system crud*, pertama kita membuat model data berbasis objek terlebih dahulu, dimana fungsi dari model data ini yaitu 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.

Project Ini memiliki beberapa menu, yaitu: *home*, data kos, data pemilik kos, admin. Menu *home* sendiri terdapat tombol masuk pemilik kos dan masuk sebagai costumer. Lalu di menu data kos, pemilik kos bisa menambahkan informasi tentang kos-kosannya. Di menu lainnya berisi tentang proses instalasi dari *project* ini sendiri.. Lalu menu *logout* jika ingin keluar dari aplikasi ini.

4.1.2 ERD (*Entity Relationship Diagram*)

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.



Gambar 4.1 ERD Kos

Penjelasan dan keterangan:

1. Admin

Admin atau Administrator adalah orang yang bertugas atau ditugaskan untuk mengelola, memeriksa dan memasukkan data-data yang dibutuhkan dan kemudian menampilkan kedalam sistem informasi sewa rumah kost dan kontrakan. Pada sistem ini, administrator mengelola dan mengontrol secara penuh transaksi yang dilakukan *user* atau member.

2. Penyewa Kos

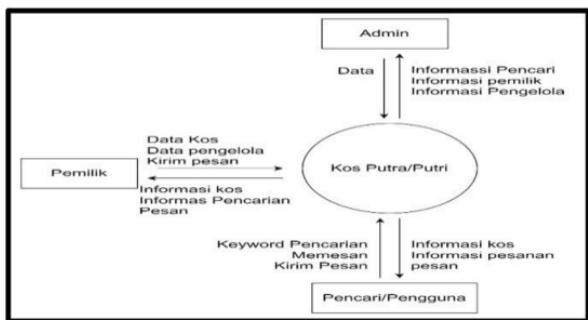
Penyewa Kos adalah orang-orang menggunakan fasilitas dari sistem informasi sewa rumah kost dan kontrakan berbasis web untuk membantu memenuhi

kebutuhan mencari informasi rumah kost dan kontrakan yang ada di Asrama Kuning. Member atau *user* dapat memesan dan menyewa kamar atau hunian dan dapat mengajukan komplain atau keluhan terhadapa masalah yang terjadi misalnya perihal listrik padam atau pun masalah air.

4.1.3 DFD

DFD merupakan alat bantu dalam menggambarkan atau menjelaskan sistem yang sedang berjalan logis. Dalam sumber lain dikatakan bahwa *DFD* ini merupakan salahsatu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, *DFD* adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem. *DFD* ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah di komunikasikan oleh profesional sistem kepada pemakai maupun pembuat program. Suatu yang lazim bahwa

ketika menggambarkan sebuah sistem kontekstual *data flow diagram* yang akan pertama kali muncul adalah interaksi antara sistem dan *entitas* luar.



Gambar 4 .2 DFD Kos

4.1.4 DFD Level 0

Diagram level 0 merupakan diagram berjenjang, diagram level 0 sangat menunjang dari pembuatan laporan secara mendetail, menjelaskan jalannya sistem yang dibangun dan turunan dari diagram level 0 ini terdapat diagram detail yang menggambarkan lebih terperinci lagi.

4.1.5 DFD Level 1

Setelah selesai membuat DFD level 0, maka tahap selanjutnya adalah merinci setiap proses yang ada pada DFD level 0, sehingga setiap *event* yang ada dalam suatu proses dapat digambarkan menjadi lebih detil dalam sebuah DFD lagi, yang disebut dengan DFD level 1. DFD level 1 bertujuan untuk memberikan pandangan mengenai keseluruhan sistem dengan lebih mendalam. Proses-proses utama yang ada akan dipecah menjadi sub-proses. *Data store* yang digunakan dalam proses-proses utama juga diidentifikasi dalam DFD level 1. DFD level 1 merupakan lanjutan dari diagram konteks, dimana setiap proses yang berjalan akan

diperinci pada tingkatan ini. Sehingga, proses utama akan dipecah menjadi sub – sub proses yang lebih kecil lagi.

4.1.4 *Interface*

Antarmuka (*Interface*) merupakan mekanisme komunikasi antara pengguna (*user*) dengan sistem. Antarmuka (*Interface*) dapat menerima informasi dari pengguna (*user*) dan memberikan informasi kepada pengguna (*user*) untuk membantu mengarahkan alur penelusuran masalah sampai ditemukan suatu solusi *Interface*, berfungsi untuk menginput pengetahuan baru ke dalam basis pengetahuan sistem pakar (*ES*), menampilkan penjelasan sistem dan memberikan panduan pemakaian sistem secara menyeluruh / *step by step* sehingga pengguna mengerti apa yang akan dilakukan terhadap suatu sistem. Yang terpenting adalah kemudahan dalam memakai / menjalankan sistem, *interaktif, komunikatif*, sedangkan kesulitan dalam mengembangkan / membangun suatu program jangan terlalu diperlihatkan.

Interface yang ada untuk berbagai sistem, dan menyediakan cara :

Input, memungkinkan pengguna untuk memanipulasi sistem. Output, memungkinkan sistem untuk menunjukkan efek manipulasi pengguna.

4.1.5 Tujuan *Interface*

Tujuan sebuah *interface* adalah mengkomunikasikan fitur-fitur sistem yang tersedia agar user mengerti dan dapat menggunakan sistem

tersebut. Dalam hal ini penggunaan bahasa amat *efektif* untuk membantu pengertian, karena bahasa merupakan alat tertua (barangkali kedua tertua setelah gesture) yang dipakai orang untuk berkomunikasi sehari-harinya. Praktis, semua pengguna komputer dan Internet (kecuali mungkin anak kecil yang memakai komputer untuk belajar membaca) dapat mengerti tulisan. Meski pada umumnya panduan interface menyarankan agar ikon tidak diberi tulisan supaya tetap mandiri dari bahasa, namun elemen *interface* lain seperti teks pada tombol, *caption window*, atau teks-teks singkat di sebelah kotak input dan tombol pilihan semua menggunakan bahasa. Tanpa bahasa pun kadang ikon bisa tidak jelas maknanya, sebab tidak semua lambang ikon bisa bersifat *universal*.

Meskipun penting, namun sayangnya kadang penggunaan bahasa, seperti pemilihan istilah, sering sekali dianggap kurang begitu penting. Terlebih dari itu dalam dunia desain *situs Web* yang serba grafis, bahasa sering menjadi sesuatu yang nomor dua ketimbang elemen-elemen *interface* lainnya.

Interface ada dua jenis, yaitu :

- 1) *Graphical Interface* : Menggunakan unsur-unsur multimedia (seperti gambar, suara, video) untuk berinteraksi dengan pengguna.
- 2) *Text-Based* : Menggunakan *syntax/rumus* yang sudah ditentukan untuk memberikan perintah.

4.1.6 PERBANDINGAN INTERFACE

Ada 5 tipe utama interaksi untuk interaction:

- 1) *Direct manipulation* – pengoperasian secara langsung : interaksi langsung dengan objek pada layar. Misalnya *delete file* dengan memasukkannya ke *trash*. Contoh: *Video games*. Kelebihan : Waktu pembelajaran sangat singkat, *feedback* langsung diberikan pada tiap aksi sehingga kesalahan terdeteksi dan diperbaiki dengan cepat. Kekurangan : *Interface* tipe ini rumit dan memerlukan banyak fasilitas pada sistem komputer, cocok untuk penggambaran secara *visual* untuk satu operasi atau objek.
- 2) Menu *selection* – pilihan berbentuk menu : Memilih perintah dari daftar yang disediakan. Misalnya saat *click* kanan dan memilih aksi yang dikehendaki. Kelebihan : tidak perlu ingat nama perintah. Pengetikan minimal. Kesalahan rendah. Kekurangan : Tidak ada logika *AND* atau *OR*. Perlu ada struktur menu jika banyak pilihan. Menu dianggap lambat oleh *expert* dibanding *command language*.
- 3) *Form fill-in* – pengisian *form* : Mengisi area-area pada *form*. Contoh : *Stock control*. Kelebihan : Masukan data yang sederhana. Mudah dipelajari Kekurangan : Memerlukan banyak tempat di layar. Harus menyesuaikan dengan form manual dan kebiasaan.
- 4) *Command language* – perintah tertulis : Menuliskan perintah yang sudah ditentukan pada program. Contoh: *operating system*. Kelebihan : Perintah diketikan langsung pada system. Misal *UNIX, DOS command*.

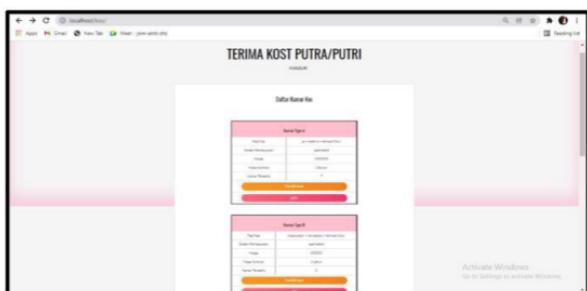
Bisa diterapkan pada terminal yang murah. Kombinasi perintah bisa dilakukan. Misal *copy file* dan *rename nama file*. Kekurangan : Perintah harus dipelajari dan diingat cara penggunaannya, tidak cocok untuk biasa. Kesalahan pakai perintah sering terjadi. Perlu ada sistem pemulihan kesalahan.Kemampuan mengetik perlu.

5) *Natural language* – perintah dengan bahasa alami : Menggunakan bahasa alami untuk mendapatkan hasil. Contoh: *search engine* di Internet. Kelebihan: Perintah dalam bentuk bahasa alami, dengan kosa kata yang terbatas (singkat), misalnya kata kunci yang kita tentukan untuk dicari oleh *search engine*. Ada kebebasan menggunakan kata-kata. Kekurangan: Tidak semua sistem cocok gunakan ini. Jika digunakan maka akan memerlukan banyak pengetikan.

4.1.5 Project

Project membuat *CRUD* kos-kosan

a. Tampilan halaman Utama.

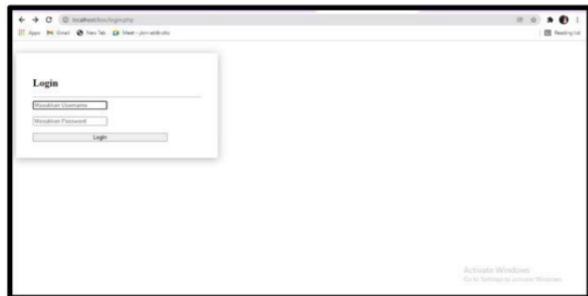


Gambar 4.3 Halaman Utama

Pada halaman Utama saya menampilkan Tempat untuk *Login* serta daftar dari kamar kos yang dimana nantinya penyewa dapat

melihat *type* dari kamar pemilik kos dan dapat melihat siapa pemilik kost tersebut

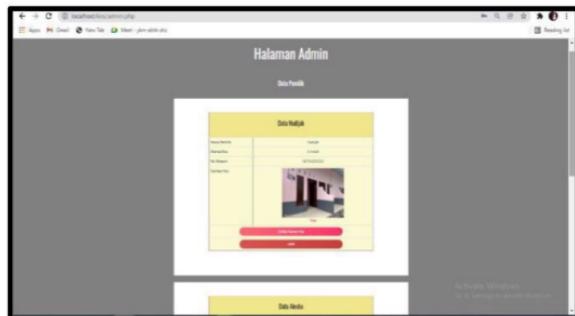
b. Tampilan *Login*



Gambar 4. 4 Tampilan Login

Pada menu *login* kita bisa masuk sebagai admin ataupun sebagai pemilik kos,kita dapat memasukkan *username* dan *password*.

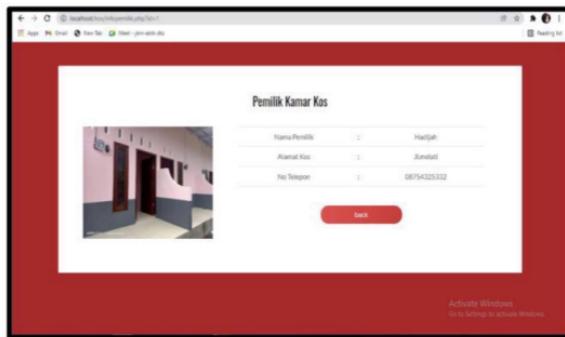
c. Halaman *admin*



Gambar 4. 5 Halaman Admin

Selanjutnya ketika kita mengklik tulisan pemilik kost pada tampilan awal tadi maka kita akan diarahkan ke tampilan pemilik kost. adapun tampilan registrasi member sebagai berikut :

d. Tampilan data pemilik kamar kos.



Gambar 4. 6 Tampilan Pemilik Kamar Kos

Pada tampilan Pemilik kamar kos disini kita dapat melihat data pemilik kos yaitu nama pemilik,alamat pemilik serta nomor telepon pemilik. Selanjutnya Ketika Kembali ke menu awal kita dapat mengklik Pilih untuk menyewa kamar kos,maka akan muncul tampilan seperti dibawah.

e. Tampilan Data penyewa kamar.

A screenshot of a web form titled "Untuk menyewa Kamar Kos Silahkan isi data diri Anda". The form has fields for "Nama Penyewa", "No ATM", "No Telpon", "Karang Anyar Mandeban", "Masa Kontrak" (set to 1 bulan Rp.600000), and a "Sewa Kamar Ini" button. A note at the bottom states: "Silahkan melakukan pembayaran pada pemilik kamar kos sebesar Rp.600000. Kunci kamar kos akan dibuka oleh pemilik, jadi segera unduh dan instal aplikasi pembayaran." (Please make a payment to the room owner's account of Rp.600000. The room key will be unlocked by the owner, so download and install the payment application immediately.)

Gambar 4.7 Tampilan Penyewa kamar Kos

Pada tampilan tersebut kita dapat mengisi data diri sebagai penyewa kamar kos dan melakukan pembayaran sesuai dengan

harga kos dan *type* yang kita pilih dengan cara melakukan pembayaran langsung atau debit kepada pemilik kos tersebut.

DAFTAR PUSTAKA

- Iksanudin, M.S. 2018.Pemrograman Berbasis Objek Modern. Jakarta.
- Yamamah. "Makalah OOP pada PHP". 22 Oktober 2015.
- Yudhanto, Y., Prasetyo.H.A. 2018.Panduan Mudah Belajar Framework Laravel.
Gramedia.Jakarta: 17