

LAPORAN LENGKAP
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK



OLEH :

NAMA : HAMKA RAHMAN

NIM : F1G120020

KELOMPOK : II (DUA)

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

HALAMAN PENGESAHAN LAPORAN PERAKTIKUM



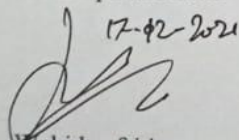
OLEH:
HAMKA RAHMAN (F1G120020)

Laporan praktikum Pemrograman Berorientasi *Object* ini disusun sebagai tugas akhir menyelesaikan praktikum Pemrograman Berorientasi *Object* sebagai salah satu syarat lulus matakuliah Pemrograman Berorientasi *Object*. Menerangkan bahwa yang tertulis dalam laporan lengkap ini adalah benar dan dinyatakan telah memenuhi syarat.

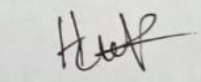
Kendari, 12. Desember 2021

Menyetujui

Asisten praktikum

17-12-2021

Wahid safri jayanto
F1G117059

Peraktikan


Hamka rahman
F1G120020

DAFTAR ISI

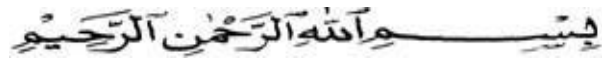
Halaman pengesahan laporan praktikum	ii
Daftar isi.....	iii
Daftar gambar.....	iv
kata pengantar	v
1.1 Pertemuan pertama.....	6
1.1.1 Alat dan bahan.....	6
1.1.2 Pengenalan PBO	6
1.1.3 Pengenalan <i>php</i>	9
1.2 Pertemuan ke dua	10
2.1.1 <i>Class</i>	10
2.1.2 <i>Method</i>	10
2.1.3 <i>Constructor</i>	11
2.1.4 <i>Modifier</i>	12
2.1.5 <i>Property</i>	13
2.1.6 <i>Object</i>	13
2.1.7 <i>Atribut</i>	14
2.1.8 <i>Composer</i>	14
2.1.9 <i>Laravel</i>	15
2.1.10 <i>Constructor dan Destructor</i>	16
2.1.11 <i>Abstract Class dan Abstract Method</i>	17
2.1.12 <i>Interface</i>	19
2.1.13 <i>Recursive Function</i>	19
1.3 Pertemuan ke tiga.....	21

3.1.1 Projek tantang php <i>crud</i>	21
3.1.2 Projek akhir	23
1.4 Pertemuan ke empat	26
4.1.1 <i>ERD</i>	26
4.1.2 DFD	28
4.1.3 <i>Interface</i>	30
Daftar pustaka	32

DAFTAR GAMBAR

Gambar 1 tampilan awal	21
Gambar 2 halaman login	21
Gambar 3 setelah berhasil login	22
Gambar 4 pemberitahuan setelah berhasil login	22
Gambar 5 halaman awal	23
Gambar 6 halaman login	23
Gambar 7 setelah login	24
Gambar 8 daftar kamar kos	24
Gambar 9 daftar orang yang sudah menyewa	25
Gambar 10 erd tugas akhir	26
Gambar 11 contoh dfd level 0	29
Gambar 12 contoh dfd level 1	30
Gambar 13 contoh dfd level 2	Error! Bookmark not defined.

KATA PENGANTAR



Assalamu'alaikum Warahmatullahi Wabarakatuh, Puji syukur mari kita panjatkan kehadiran Allah swt yang telah memberikan kesehatan dan kekuatan sehingga laporan ini bisa selesai tepat pada waktunya.

Shalawat serta salam tidak lupa selalu kita haturkan untuk junjungan nabi agung kita, yaitu nabi Muhammad salallahu 'alaihi wassalam karena beliau adalah yang membaw kita dari alam kegelapan menuju alam yang terang benerang seperti apa yang kita rasakan saat ini.

Penulis sangat mengucapkan banyak terima kasih kepada semua pihak yang terlibat dalam pengerjaan laporan ini semoga laporan ini dapat menambah pengetahuan dan pengalaman bagi pembaca. Bahkan penulis berharap lebih jauh lagi agar laporan ini bisa pembaca praktekkan dalam kehidupan sehari-hari.

Bagi penulis sebagai penyusun merasa bahwa masih banyak kekurangan dalam penyusunan laporan ini karena keterbatasan pengetahuan dan pengalaman. Untuk itu saya sangat mengharapkan kritik dan saran yang membangun dari pembaca demi kesempurnaan laporan ini ini.

Kendari, Desember 2021

Penulis

1.1 Pertemuan pertama

1.1.1 Alat dan bahan

Adapun alat dan bahan yang di gunakan pada praktikum kali ini yaitu:

a. Laptop

Sebagai tempat untuk menyimpan data, untuk mengerjakan proyek dan sebagai tempat untuk mengoding.

b. Xampp

Sebagai penghubung antara *chrome* dan *vscode*/menyediakan *server* gratis.

c. Google chrome

Sebagai tempat untuk melihat hasil *running* dari program yang telah di buat.

d. Vscode

Sebagai tempat mengoding sebuah program.

1.1.2 Pengenalan PBO

Pemrograman berorientasi *objek* atau *object oriented programming* (OOP) merupakan suatu pendekatan pemrograman yang menggunakan *object* dan *class*. OOP memberikan kemudahan dalam pembuatan sebuah program, keuntungan yang didapat apabila membuat Program berorientasi objek atau *object oriented programming*.

a. Reusability

kode yang dibuat dapat digunakan kembali,

b. Extensibility

pemrogram dapat membuat metode baru atau mengubah yang sudah ada sesuai yang diinginkan tanpa harus membuat kode dari awal

c. *Maintainability*

kode yang sudah dibuat lebih mudah untuk dikelola

(Ramahdani,2015)

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*). Pemrograman berorientasi objek ditemukan pada Tahun 1960, dimana berawal dari suatu pembuatan program yang terstruktur (*structured programming*). Metode ini dikembangkan dari bahasa 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.

Pemrograman berorientasi objek dalam melakukan pemecahan suatu masalah tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh sebuah departemen yang memiliki seorang manager, sekretaris, petugas administrasi data dan lainnya. Jika manager ingin memperoleh data dari bagian administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.

Pemrograman berorientasi objek bekerja dengan baik ketika dibarengi dengan *Objek-Oriented Analysis And Design Process (OOAD)*. Jika membuat program berorientasi objek tanpa *OOAD*, seperti membangun rumah tanpa terlebih dahulu penganalisis apa saja yang dibutuhkan oleh rumah itu, tanpa perencanaan, tanpa *blue-print*, tanpa menganalisis ruangan apa saja yang diperlukan, beberapa besar rumah yang akan dibangun dan sebagainya.

1.1.3 Pengenalan *php*

PHP adalah pemrograman interpreter yaitu proses penerjemahan baris kode sumber menjadi kode mesin yang dimengerti komputer secara langsung pada saat baris kode dijalankan. PHP disebut sebagai pemrograman Server Side Programming, hal ini dikarenakan seluruh prosesnya dijalankan pada server tidak dijalankan pada client. PHP merupakan suatu bahasa dengan hak cipta terbuka atau yang juga dikenal dengan istilah Open Source, yaitu pengguna dapat mengembangkan kode fungsi PHP dengan kebutuhannya.(Hartono,2017)

1.2 Pertemuan ke dua

2.1.1 Class

Class merupakan suatu *blueprint* atau cetakan untuk menciptakan suatu *instant* dari *object*. *Class* juga merupakan grup suatu *object* dengan kemiripan *attributes/properties*, *behaviour* dan relasi ke *object* lain.

Contoh *syntax*

```
<?php

//Cara penulisan class OOP PHP - www.malasngoding.com
class nama_class{

    //isi dari class ini

}

?>
```

2.1.2 Method

Method merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu *object*. *Method* didefinisikan pada *class* akan tetapi dipanggil melalui *object*. Metode menentukan perilaku objek, yakni apa yang terjadi ketika objek itu dibuat serta berbagai operasi yang dapat dilakukan objek sepanjang hidupnya. Ada 4 (Empat) bagian dasar yang dimiliki metode antara lain:

- a. Nama *metode*
- b. *Tipe* Objek atau *tipe primitive* yang dikembalikan *metode*.
- c. Daftar *parameter*.
- d. Badan atau isi metode.

Contoh *syntax*

```
<?php

//Cara penulisan class dan property OOP PHP -
www.malasngoding.com
class mobil{
    // property oop
    var $warna;
    var $merek;
    var $ukuran;

    //method oop
    function maju(){
        //isi method
    }

    function berhenti(){
        //isi mehod
    }
}

?>
```

2.1.3 *Constructor*

Constructor adalah *Constructor* merupakan suatu method yang akan memberikan nilai awal pada saat suatu objek dibuat. Pada saat program dijalankan. (Gunadarman, 2013) , *constructor* akan bekerja dengan *constructor*, hal mendasar yang perlu diperhatikan, yaitu :

- a. Nama *Constructor* sama dengan nama *Class*.
- b. Tidak ada *return type* yang diberikan kedalam *Constructor Signature*.
- c. Tidak ada return statement, didalam tubuh *constructor*.

Contoh syntax

```
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() )
    }
}
```

2.1.4 Modifier

Modifier adalah kata *phrase* atau *clause* yang berfungsi sebagai *adjective* atau *adverb* yang menerangkan kata atau kelompok kata lain. Sebagai *adjective* dan *adverb* ketika berfungsi sebagai *adjective* (dapat berupa simple *adjective*, *adjective phrase*, *clause participle*, *infinitive*), *modifier* menerangkan *noun*, sedangkan ketika berfungsi sebagai *adverb* kata ini menerangkan *verb*, *adjective* atau *adverb* lain.

Contoh syntax

```
Public class bank balance
{
public String owner
public int balance
public bank_balance(String name, int dollars )
{
owner = name;
if(dollars > = 0)
balance = dollars;
else
dollars =0;
}
}
```

2.1.5 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.

Contoh syntaxnya

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

2.1.6 Object

Object atau *Object* adalah *hasil cetak* dari *class*, atau hasil ‘*konkrit*’ dari *class*. Jika menggunakan *analogi class laptop*, maka *object* dari *class laptop* bisa berupa. 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 *xytaxnya*

```
<?php
class laptop {
    //... isi dari class laptop
}

$laptop_andi = new laptop();
$laptop_anto = new laptop();
?>
```

2.1.7 Atribut

Atribut merupakan nilai data yang terdapat pada suatu *object* di dalam *class*. *Attribute* mempunyai karakteristik yang membedakan *object* yang satu dengan *object* yang lainnya. Contoh : pada *Class* Buah terdapat *attribute*: warna, berat. Misalkan pada *object* mangga: warna berisi kuning dan berat 0.5 kg dan pada *object* apel : warna merah dan berat 0.6 kg.

2.1.8 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. *Composer* adalah *package-manager* (di level aplikasi) untuk bahasa pemrograman *PHP*. Menawarkan standarisasi cara pengelolaan *libraries* dan *software dependencies* dalam proyek *PHP*.

Dengan *Composer* kita tidak perlu repot-repot lagi *mendownload source code* pustaka yang kita butuhkan secara manual, lalu memasangnya di aplikasi kita, lalu *mengupdate*-nya secara manual jika ada versi baru. Itu semua tidak perlu lagi karena *Composer* bisa menangani semua proses tersebut dengan mudah.

Cara penggunaan

```
<?php
// misalkan ini adalah file index.php

require_once __DIR__ . '/vendor/autoload.php';

$fb = new \Facebook\Facebook([
    'app_id' => '{app-id}',
    'app_secret' => '{app-secret}',
    'default_graph_version' => 'v2.10',
    //'default_access_token' => '{access-token}', //
    optional
]);
```

2.1.9 *Laravel*

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 *fungsi* 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*.

2.1.10 Constructor dan Destructor

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*.

Destructor adalah sebuah *method* khusus yang dieksekusi ketika sebuah *object* dihapus dari *memory*. Secara mudah, *destructor* adalah kebalikan dari *constructor*. Sama seperti pada *constructor*, *PHP* juga akan membuat *destructor* tanpa parameter dan tanpa *logic* jika kita tidak mendefinisikan *destructor* secara *eksplisit*. Berbeda dengan *constructor* yang dapat memiliki parameter, *destructor* tidak dapat memiliki parameter dan hanya dapat berisi *logic*.

Contoh *syntax Denstructor*:

```
class User {  
    public:  
        User( String *username ); // <-- ini constructor  
        ~User(); // <-- ini destructor.  
    private:  
        String username;  
        String password;  
};
```

Contoh syntax Constructor:

```
package konstruktor;

public class User {

    public String username;

    public String password;

    public User(String username, String password){

        this.username = username;

        this.password = password;

    }

}

class DemoConstructor{

    public static void main(String[] args) {

        User petani = new User("petanikode", "kopi");

        System.out.println("Username: " +

petani.username);

        System.out.println("Password: " +

petani.password);

    }

}
```

2.1.11 Abstract Class dan Abstract Method

Abstract class adalah sebuah *class* dalam *OOP* yang tidak dapat diinstansiasi atau dibuat *object*-nya. *Abstract class* biasanya berisi fitur-fitur dari sebuah *class* yang belum implementasikan. Didalam sebuah abstract class kita dapat membuat *property* dan *method* yang nantinya dapat digunakan oleh *child class*. Tentu saja *property* dan *method* yang dapat digunakan oleh *child class* adalah *property* dan *method* yang memiliki *visibilitas protected* dan *public*.

Syntax Abstract Class:

```
<?php
abstract class komputer {
    // isi dari class komputer
}
?>
```

abstract method adalah sebuah *method* yang harus diimplementasikan oleh *child class*. *Abstract method* hanya ada pada *abstract class* dan *interface* (akan dibahas secara terpisah). Bila biasanya setiap *method* yang kita buat pasti mempunyai kurawal {}, pada *abstract method* hal tersebut tidak dapat ditemui karena *abstract method* adalah sebuah *method* yang tidak memiliki *body* atau badan *method*. Pada *child class*, *abstract method* harus didefinisikan ulang dan kita tidak dapat menggunakan *keyword parent* untuk memanggil *abstract method* pada *parent class*. Bila kita melakukan hal tersebut maka akan terjadi *error*.

Contoh Syntax Abstract Method:

```
<?php
abstract class komputer {
    abstract public function lihat_spec();
}
?>
```

Kegunaan *Abstract Class* dan *Abstract Method* Yaitu Secara mudah *abstract class* dan *abstract method* berguna untuk memastikan *child class* memiliki fitur-fitur yang telah ditentukan sebelumnya. *Abstract class* akan sangat berguna pada saat kita membahas tentang *type hinting* atau parameter hinting. Dengan *abstract class* dan *abstract*

method kita bisa lebih percaya diri ketika memanggil sebuah *method* karena dapat dipastikan *method* tersebut dimiliki *child class*.

2.1.12 Interface

Dalam pemrograman berbasis objek, *interface* adalah sebuah *class* yang semua *method*-nya adalah *abstract method*. Karena semua *method*-nya adalah *abstract method* maka *interface* pun harus diimplementasikan oleh *child class* seperti halnya pada *abstract class*. Hanya saja bila kita sebelumnya menggunakan *keyword extends* untuk mengimplementasikan sebuah *abstract class*, maka pada *interface* kita menggunakan *keyword implements* untuk mengimplementasikan sebuah *interface*. *interface* akan *valid* jika dimasukkan kedalam *method* yang menggunakan *interface* tersebut sebagai *type hinting* atau *parameter casting*.

2.1.13 Recursive Function

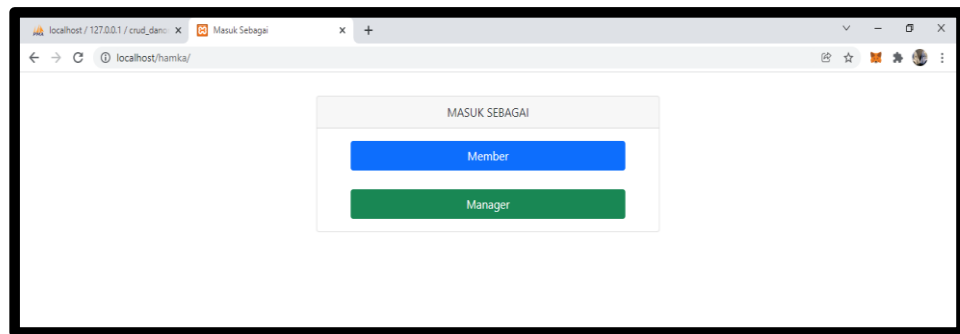
Recursive function adalah sebuah *function* yang memanggil dirinya sendiri dalam badan *function*-nya. *Recursive function* biasanya dipakai untuk menyelesaikan permasalahan yang mempunyai pola dasar yang berulang seperti perhitungan *faktorial*. Keuntungan menggunakan *recursive function* adalah mempersingkat *code* yang kita tulis. Namun yang perlu diperhatikan adalah bahwa kita harus benar-benar paham bagaimana *function* tersebut bekerja. Jika kita tidak paham bagaimana *nested call* yang terjadi didalam *recursive function* bisa saja bukan solusi singkat yang didapat tapi justru permasalahan yang justru kita sama sekali tidak mengetahui bagaimana cara mengatasinya. *Recursive function* sangat perlu dipelajari dan dipahami oleh programmer karena dalam banyak kasus *recursive*

function terbukti mampu menyelesaikan permasalahan yang *kompleks* dan dinamis.

1.3 Pertemuan ke tiga

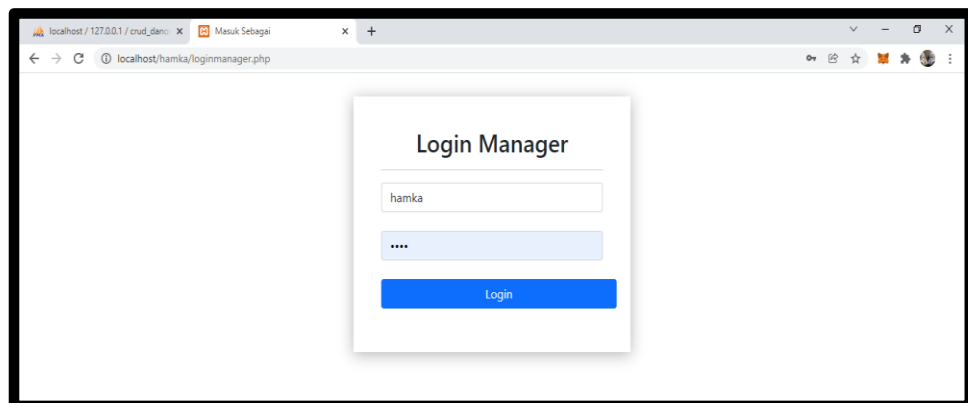
3.1.1 Proyek tantang php crud

Saya akan menjelaskan proyek/tugas yang di yang dimana tugasnya yaitu membuat *php crud*.



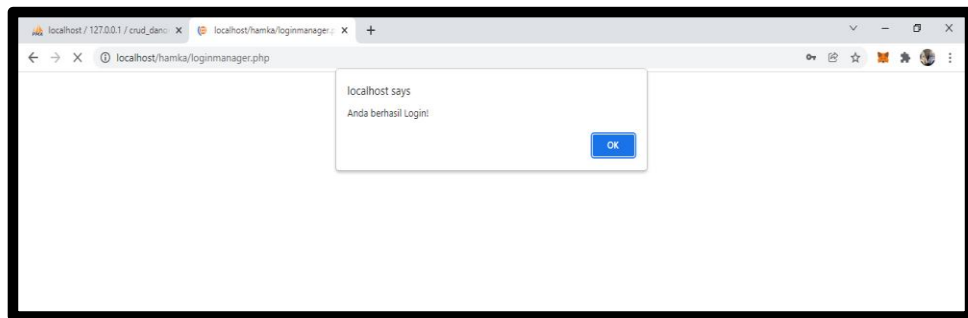
Gambar 1 tampilan awal

Awal pertama masuk disini ada dua pilihan yaitu apakah ingin *login meneger* atau *membre*.misalnya kita *login manager* maka tampilan nya seperti pada gambar 2

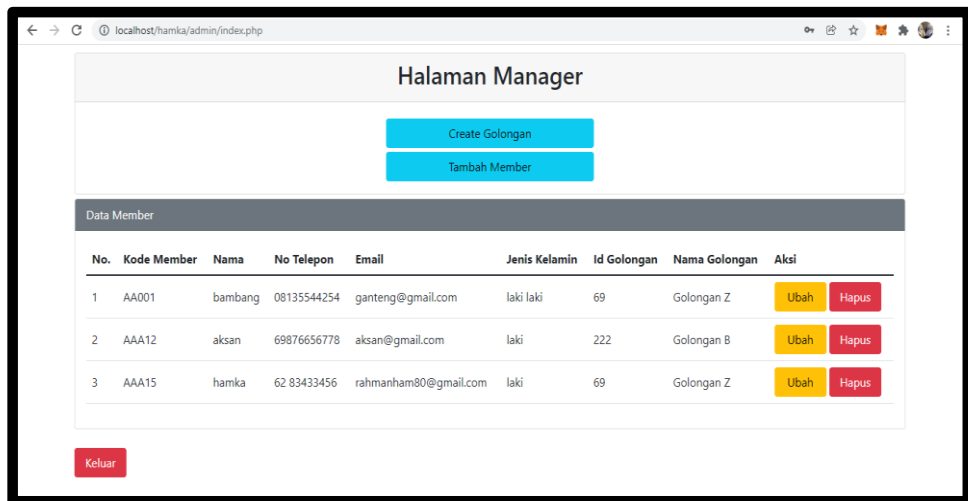


Gambar 2 halaman login

Login sebagai *maneger* di sini ada perintah untuk memasukan nama dan *password* yang sudah ada sebelumnya.setelah selesai memasukan nama dan *password* maka akan muncul tampilan semacam pemberitahuan bahwa kita sudah berhasil *login*/masuk seperti pada gambar 3.



Gambar 4 pemberitahuan setelah berhasil *login* setelah *password* yang kita masukan sudah benar maka kita langsung di arahkan ke halaman selanjutnya seperti pada gambar 3.



Gambar 3 setelah berhasil *login* saya mengambil contoh di halaman *maneger* ini setelah berhasil *login* di sini ada data yang di mna data tersebut bisa kita ubah,hapus,tambah karena projek ini mengusun tema (*php crud*).

3.1.2 Proyek akhir

Tugas akhir ini mengusun tema tempat sewa kos-kosan.seperti pada gambar di bawah ini ada daftar kamar kos

[Login](#)

Daftar Kamar Kos

Kamar Type A	
Fasilitas	ac + televisi
Sistem Pembayaran	cash
Harga	3000000
Masa Kontrak	6 Bulan
Kamar Tersedia	2
Informasi Pemilik	
Pilih	

Kamar Type B	
Fasilitas	kipas angin + rak sepatu
Sistem Pembayaran	cash

Gambar 5 halaman awal

Seperti yang kita lihat pada gambar 5 pada pojok kiri atas ada tombol untuk *login* itu khusu bagi yang ingin menyewa dan ingin melakukan pembayaran

[Login](#)

Login

hamka r

....

Login

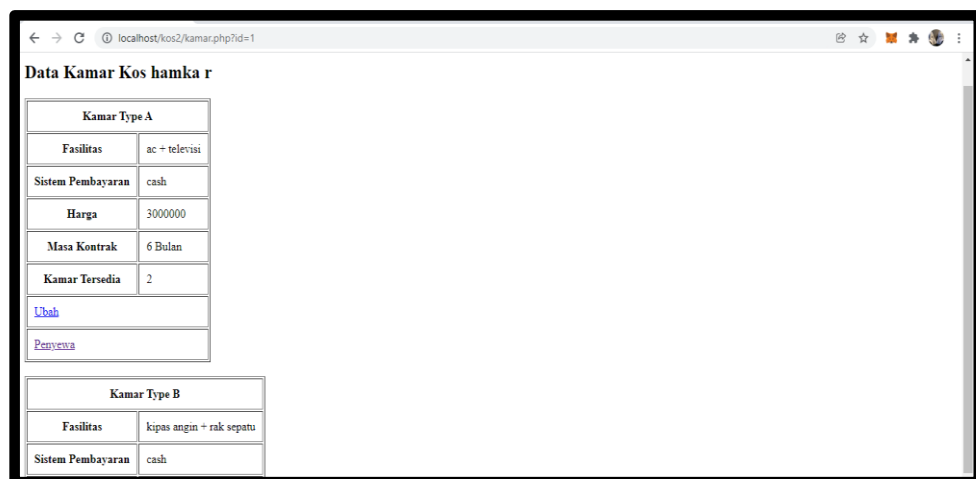
Gambar 6 halaman *login*

Halaman *login* ini terkusus bagi pemilik kos dan yang ingin melakukan pembayaran di sini kita diwajibkan mengisi *username* dan *password*



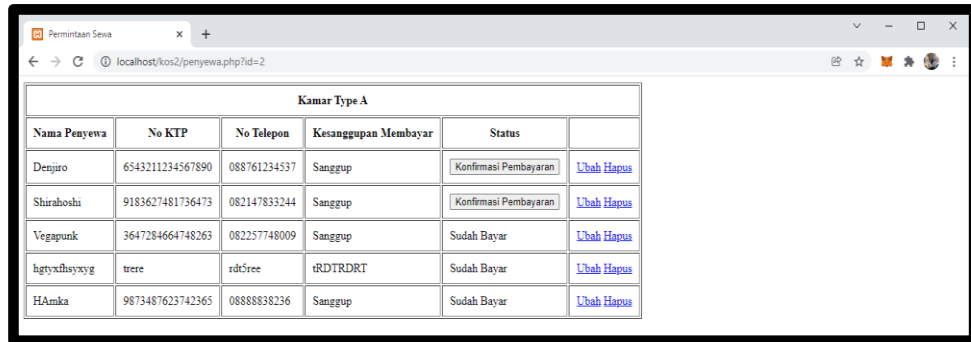
Gambar 7 setelah login

Setelah kita melakukan proses *login*, jika kita pemilik kos maka kita bisa mengubah data yang ada di dalam kita juga bisa cek apakah masih ada kamar yang kosong atau tidak



Gambar 8 daftar kamar kos

halaman ini menampilkan daftar kamar kos hamka r yang tadi kita klik di awal dan terhubung saya *login* sebagai pemilik maka saya bisa mengubah atau megedit apapun yang ingin saya ubah



Kamar Type A					
Nama Penyewa	No KTP	No Telepon	Kesanggupan Membayar	Status	
Denjiro	6543211234567890	088761234537	Sanggup	Konfirmasi Pembayaran	Ubah Hapus
Shirahoshi	9183627481736473	082147833244	Sanggup	Konfirmasi Pembayaran	Ubah Hapus
Vegapunk	3647284664748263	082257748009	Sanggup	Sudah Bayar	Ubah Hapus
hgtyxhsyxyg	trere	rdcfree	tRDTRDRT	Sudah Bayar	Ubah Hapus
HAma	9873487623742365	08888838236	Sanggup	Sudah Bayar	Ubah Hapus

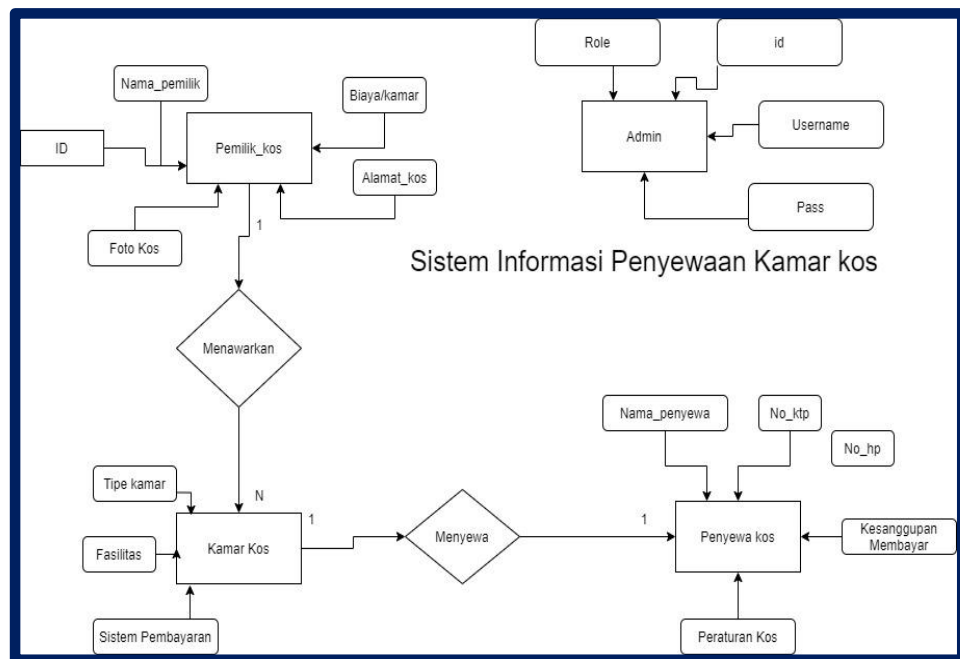
Gambar 9 daftar orang yang sudah menyewa
pada gambar 9 menunjukkan ketika seorang pemilik kos ingin melihat siapa
saya yang menyewa kos miliknya dan melihat bukti pembayaran.

1.4 Pertemuan ke empat

Pertemuan ke empat ini membahas tentang *ERD, DFD* dan *interface*

4.1.1 ERD

Berikut adalah *erd* dari proyek (tugas akhir) peraktikum pemograman berorientasikan *object*



Gambar 10 erd tugas akhir

Penjelasan:

Pada *ERD* proyek kos ini terdiri dari table pemilik kos, kamar kos, penyewa kos dan admin.

- Pemilik kos berelasi dengan kamar kos dan mempunyai hubungan relasinya *one to many* yang artinya satu pemiilik kos bias mempunyai banyak kamar kos sementara
- hubungan antara kamar kos dan penyewa kos berelasi dan mempunyai hubungan *one to one* artinya satu kamar kos bisa ditempati oleh satu penyewa kos, dan

- c. terakhir tabel admin dia berdiri sendiri dia berfungsi sebagai tempat untuk menyimpan *user,id* maupun *password*

ERD merupakan model atau rancangan untuk membuat *database*, supaya lebih mudah dalam menggambarkan data yang memiliki hubungan atau relasi dalam bentuk sebuah desain. Dengan adanya *ER* diagram, maka sistem database yang terbentuk dapat digambarkan dengan lebih terstruktur dan terlihat rapi.

Untuk menyusun sistem database yang tepat, maka kita harus menentukan terlebih dahulu mengenai jenis model data yang akan digunakan. Yang mana, hal tersebut akan sangat berpengaruh nantinya pada pengembangan aplikasi sesuai dengan kebutuhan proyek bisnis. Model ER konseptual sangat berguna untuk mendokumentasikan segala bentuk arsitektur data pada sebuah organisasi. Model ini dapat digunakan untuk satu atau lebih jenis model data logis. Tujuan dari pengembangannya adalah untuk membangun struktur metadata untuk data master entitas dan set ER model logis.

Fungsi dari *erd* itu sendiri yaitu Untuk memudahkan kita dalam menganalisis pada suatu basis data atau suatu sistem dengan cara yang cepat dan murah, Untuk mendokumentasikan data-data yang ada dengan cara mengidentifikasi setiap entitas dari data-data dan hubungannya pada suatu Entity Relationship Diagram (ERD) itu sendiri, dan masih banyak fungsi dari *erd* itu sendiri.

4.1.2 DFD

DFD merupakan 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. Menurut Kenneth Kozar, tujuan dari adanya DFD sendiri adalah sebagai penyedia atau menjembatani antara pengguna dengan sistem.

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.

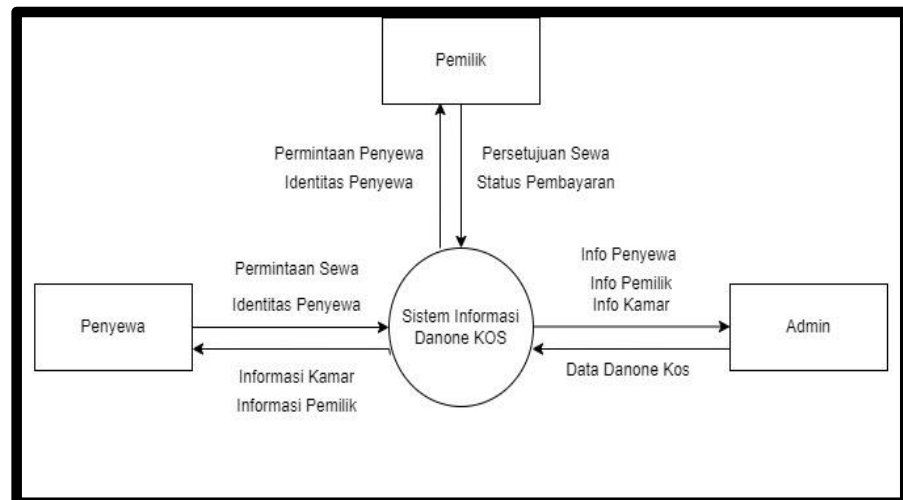
Data flow diagram terbagi menjadi tiga jenis, dimana setiap bagian memiliki peran dan fungsinya masing – masing. Untuk pembuatannya sendiri dapat menyesuaikan kebutuhan proyek dari manajemen tim -nya.

a. Diagram *level 0* diagram konteks)

Diagram konteks atau level 0 merupakan diagram dengan tingkatan paling rendah, dimana menggambarkan sistem berinteraksi dengan entitas eksternal. Pada diagram konteks akan diberi nomor untuk setiap proses yang berjalan, dimulai dari angka 0 terlebih dahulu. Jadi, untuk setiap aliran data akan langsung diarahkan menuju sistem. Dan ciri dari

diagram level 0 terletak pada tidak adanya informasi yang terkait data yang tersimpan pada *data store*.

Contoh



Gambar 11 contoh dfd level 0

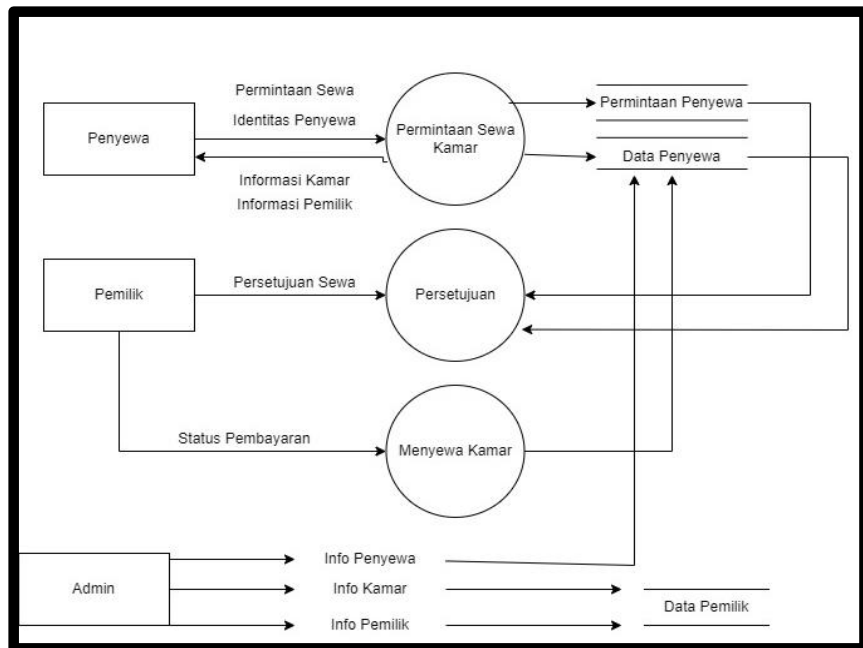
Penjelasan

Pelanggan memasukan data/mengirim data ke aplikasi ojek *online* selanjutnya aplikasi ojek *online* mengirim ke driver ojek *online* dan setelah driver ojek *online* menerima dia langsung mengirim lagi ke aplikasi ojek *online* sebagai konfirmasi order dan setelah itu aplikasi ojek *online* mengirim lagi ke pelanggan.

b. Diagram *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.

Contoh



Gambar 12 contoh dfd level 1

- Pada tahap ini pelanggan akan memasukan biodata diri seperti nama, alamat, no handphone, dan alamat email
- Salon akan menginputkan daftar treatmen seperti potong rambut, perawatan wajah, dan lainnya dengan harga yang sudah ditentukan
- Kemudian sistem akan menyimpan data dari pelanggan untuk memudahkan pelanggan saat melakukan pemesanan
- Selanjutnya admin akan mengakses semua data tersebut ke sistem informasi

4.1.3 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 ada dua jenis, yaitu:

a. Graphical Interface

Menggunakan unsur-unsur multimedia (seperti gambar, suara, video) untuk berinteraksi dengan pengguna.

b. Text-Based

Menggunakan syntax/rumus yang sudah ditentukan untuk memberikan perintah.

DAFTAR PUSTAKA

- Ramadhani C. 2015. Dasar Algoritma & Struktur Data dengan Bahasa JAVA, 1e. Yogyakarta: Andi Offset.
- Hermawan B. 2004. Menguasai JAVA 2 & Object Oriented Programming, 1e. Yogyakarta: Andi Offset
- Harri H, hartono, sukimsn, 2017. Pengembangan Learning Management System (LMS) Untuk Bahasa Pemrograman PHP e-ISSN: 2548-3528 p-ISSN: 2339-1766