

LAPORAN LENGKAP
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK



OLEH :

NAMA : JANUARDIN DANU
NIM : F1G120022

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:

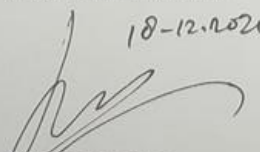
JANUARDIN DANU (F1G120022)

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.

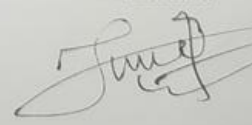
Kendari, 18⁹ Desember 2021

Menyetujui

Asisten Praktikum

18-12-2021

Wahid Safri Jayanto
F1G117059

Praktikan


Januardin Danu
F1G120022

KATA PENGANTAR



Puji syukur kami panjat kankehadirat Allah SWT, karena berkat rahmat dan hidayah-nya penyusunan laporan Pemrograman berorientasi 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 sistematis yang baik dan jelas di tulis dengan bahasa yang mudah dimengerti dan dipahami.

Akhir kata, kami menyadari "tak ada gading yang retak" juga laporan ini tidak lepas dari kekurangan. Oleh karenanya, kami mengharap kritik dan saran dari pengguna laporan ini. Sekian terimakasih, *wabillahaufikwalhidayah, WassalamuAlaikum Warahumatullahi Wabarakatu.*

Kendari, Desember 2021

Penyusun

DAFTAR ISI

HALAMAN PENGESAHAN.....	ii
KATA PENGANTAR	iii
DAFTAR ISI.....	iv
DAFTAR TABEL.....	vi
DAFTAR GAMBAR	vii
1.1 Pertemuan I.....	1
1.1.1 Alat dan bahan	1
1.1.2 Pengenalan PBO	1
1.1.3 Pengenalan PHP.....	3
2.1 Pertemuan II	5
2.1.1 Class.....	5
2.1.2 Method	5
2.1.3 Constructor.....	7
2.1.4 <i>Modifier</i>	8
2.1.5 Property.....	9
2.1.6 Object.....	9
2.1.7 Atribut.....	10
2.1.8 <i>Composer</i>	10
2.1.9 Laravel	11
2.1.10 Constructor dan Destructor	12
2.1.11 Abstract Class dan Abstract Method	15
2.1.12 Inheritance	16
2.1.13 Recursive Function	17
3.1 Pertemuan ke III	18
3.1.1 CRUD	18

3.1.2 Projek <i>CRUD</i> data member	19
4.1 Pertemuan IV	22
4.1.1 ERD	22
4.1.2 DFD	24
4.1.3 Interface	26
DAFTAR PUSTAKA	30

DAFTAR TABEL

Tabel 1.1	Tabel alat dan Bahan.....	1
------------------	---------------------------	---

DAFTAR GAMBAR

Gambar 3.1 Halaman login sebagai.....	19
Gambar 3.2 Halaman login.....	20
Gambar 3.3 Halaman Manager.....	20
Gambar 3.4 Halaman Member.....	21
Gambar 4.1 <i>ERD</i> Penyewaan Kamar Kos.....	23
Gambar 4.2 DFD Level 0.....	25
Gambar 4.3 DFD Level 1.....	25
Gambar 4.4 Halaman Utama Danone Kos.....	27
Gambar 4.5 Halaman Data Pemilik.....	27
Gambar 4.6 Halaman Perintaan Sewa.....	28
Gambar 4.7 Halaman Daftar Penyewa.....	29

1.1 Pertemuan I

1.1.1 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.
Sublime	Sebagai tempat mengoding sebuah program.
<i>Chrome</i>	Sebagai tempat untuk melihat hasil <i>running</i> dari program yang telah di

Tabel 1.1 penggunaan alat dan bahan

1.1.2 Pengenalan PBO

Pemrograman *berorientasi* objek (*Object Oriented Programming* atau disingkat *OOP*) adalah paradigma pemrograman yang berorientasikan kepada objek yang merupakan suatu metode dalam pembuatan program, dengan tujuan untuk menyelesaikan kompleksnya berbagai masalah program yang terus meningkat. Objek adalah *entitas* yang memiliki *atribut*, karakter (*bahaviour*) dan kadang

kala disertai kondisi (*state*). 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 menganalisis 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. (Douglas, 1992).

1.1.3 Pengenalan 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 perilsan *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 *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. (Triwansyah yuliano, 2007).

2.1 Pertemuan II

2.1.1 Class

Class merupakan suatu *blueprint* atau cetakan untuk menciptakan suatu *instant* dari *object* yang biasa digunakan untuk membuat kerangka kerja. *Class* juga merupakan grup suatu *object* dengan kemiripan *attributes/properties*, *behaviour* dan relasi ke *object* lain. (Gunadarman, 2013)

Contoh *syntax*:

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

2.1.2 Method

Method atau disebut juga tingkah laku merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu *object*. *Method* didefinisikan sebuah aksi yang terdapat didalam sebuah class. misalnya method pada class mobil adlah contohnya maju, berhenti, belok dan lain- lain. Penulisan method pada class OOP adalah dengan menuliskan syntax function di awalnya .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:

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 perangkat 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. (Gunadarman, 2013)

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

Construktor 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 :

- 1) Nama *Constructor* sama dengan nama *Class*.
- 2) Tidak ada *return type* yang diberikan kedalam *Constructor Signature*.
- 3) Tidak ada return statement, didalam tubuh *constructor*.

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()
}
```

2.1.4 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* (dapat berupa *simple adverb* , *adverb phrase*, *clause*, *preposition phrase*,*infinitive*), kata ini menerangkan *verb*, *adjective* atau *adverb* lain. (Gunadarman, 2013)

Contoh Program:

```
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. (Andre 2015).

Contoh Syntax:

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

2.1.6 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_duniaikom, 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. . (Andre 2015).

Contoh *Syntax*:

```
<?php
class laptop {
    //... isi dari class 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 (Andre 2015).

2.1.8 Composer

Composer merupakan *tool* yang di dalamnya terdapat *dependencies* dan, kumpulan *library*. Seluruh *dependencies* disimpan menggunakan format *file composer.json* sehingga dapat ditempatkan di dalam folder utama *website*. Inilah mengapa *composer* terkadang dikenal

dengan *dependencies management*. *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. (Nurul Huda, 2020).

Cara Pengunannya:

```
<?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 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 *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*. (Yasin k, 2019).

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*. Muncul pertanyaan, kenapa *constructor* tidak dapat mengembalikan *return value*? Ya jelas lah tidak bisa mengembalikan *return value*, kan *keyword new* itu sudah mengembalikan berupa *object* dari *class* yang diinstansiasi. Masa kemudian *constructor* mengembalikan lagi nilai yang sesuai? Misalnya, kita punya *class A* maka ketika menginisiasi *class A* tersebut dengan *keyword new* kedalam variable \$a maka saat itu sebenarnya telah mengembalikan nilai berupa *object A* ke dalam *variable \$a* tersebut. Bagaimana jadinya jika didalam *constructor* kita dapat mengembalikan nilai dan kemudian membuat *constructor* dengan mengembalikan nilai integer 1 misalnya. Maka yang terjadi ketika X. *Constructor* dan *Destructor* 79 kita melakukan *instansiasi class A* dan fungsi *constructor* dipanggil, alih-alih kita mendapatkan *object A* yang ada kita justru mendapatkan integer 1 . . (Ahmad Muhardian 2019).

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*. (Ahmad Muhardian 2019).

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. Seperti pada pembahasan sebelumnya tentang *class Connection* dimana kita harus membuat implementasi dari *class* tersebut dengan *meng-extends*-nya menjadi *MySQLConnection* dan *PostgreSQLConnection*. Karena *abstract class* harus diimplementasikan melalui proses pewarisan, maka dalam *abstract class* berlaku aturan-aturan yang ada pada konsep pewarisan yang telah kita bahas sebelumnya. 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*. (Andre, 2018).

Syntax Abstract Class:

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

Sama seperti *abstract class*, *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 kurang 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*. (Andre, 2018).

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*. (Andre, 2018).

2.1.12 Inheritance

Inheritance dalam konsep OOP adalah kemampuan untuk membentuk class baru yang memiliki fungsi turunan atau mirip

dengan fungsi yang ada sebelumnya. Konsep ini menggunakan sistem hierarki atau bertingkat. Maksudnya, semakin jauh turunan atau subclass-nya, maka semakin sedikit kemiripan fungsinya.

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.

3.1 Pertemuan ke III

3.1.1 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. Jika dihubungkan dengan tampilan antarmuka (interface), maka peran CRUD sebagai fasilitator berkaitan dengan tampilan pencarian dan perubahan informasi dalam bentuk formulir, tabel, atau laporan.

Terdapat empat poin penting dari akronim fungsi CRUD untuk mengembangkan perangkat lunak, baik berbasis web maupun mobile.

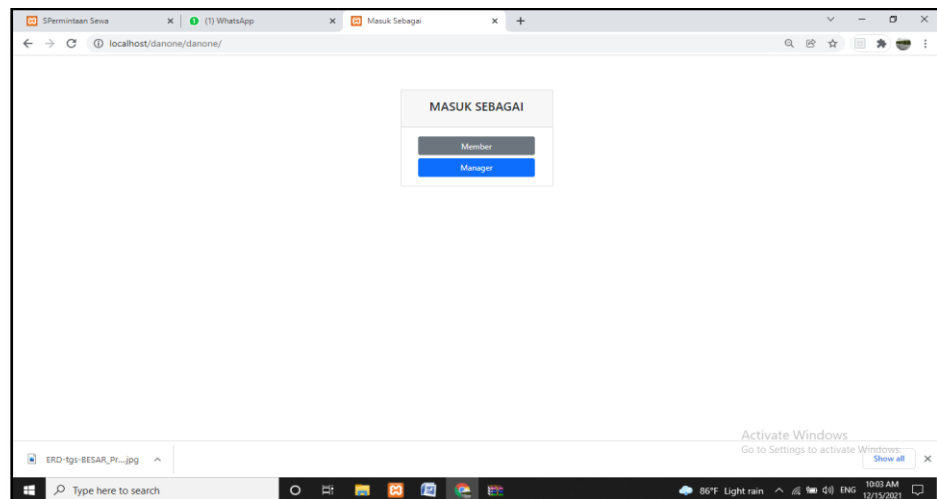
- 1) **CREATE** Fungsi CRUD yang pertama adalah create. Fungsi ini memungkinkanmu membuat record baru dalam database. Dalam aplikasi SQL, fungsi create sering disebut juga sebagai insert.
- 2) **READ** Fungsi read hampir mirip dengan fungsi search. Fungsi ini memungkinkan kamu untuk mencari dan mengambil data tertentu dalam tabel dan membaca nilainya.
- 3) **UPDATE** Untuk memodifikasi record yang telah tersimpan di database, fungsi CRUD yang bisa kamu gunakan adalah fungsi update.

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

3.1.2 Projek *CRUD* data member

Disini saya akan menjelaskan bagaimana proyek saya yang tentang crud ,disini juga saya akan menampilkan gambar beserta keterangannya.

1. Halaman Login Sebagai

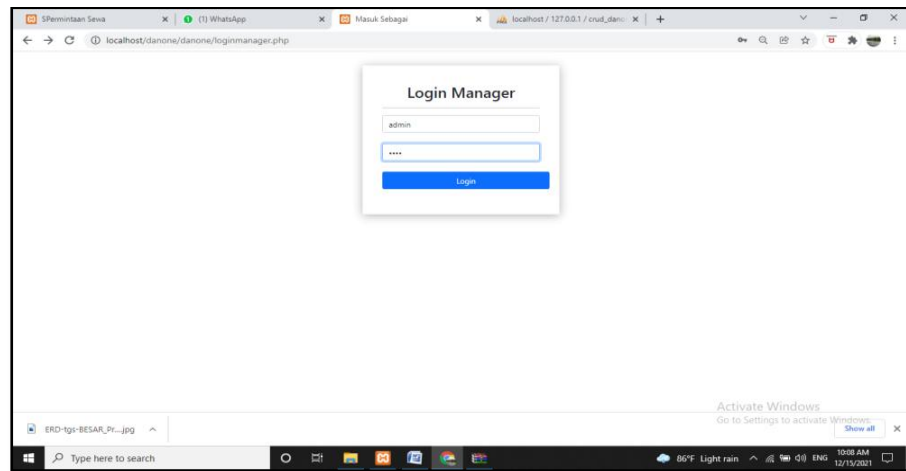


Gambar 3.1 Halaman Login Sebagai

Keterangan:

Pada halaman ini terdapat dua pilihan login yaitu login sebagai manager atau sebagai manager.

2. Halaman Login Manager

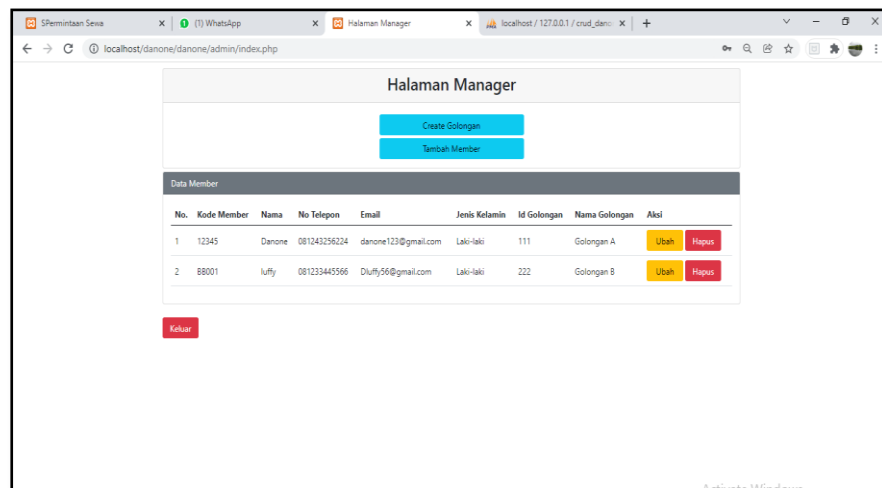


Gambar 3.2 Halaman *login*.

Keterangan:

Setelah login pada halaman ini kita akan diarahkan ke halaman manager.

3. Halaman Manager

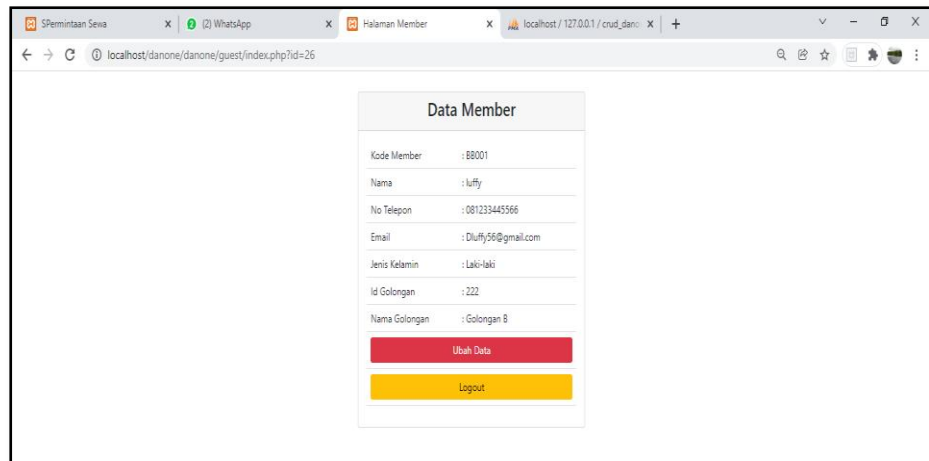


Gambar 3.3 Halaman Manager

Keterangan:

Pada halaman manager ini kita bisa melakukan *create*, *read*, *update*, dan *delete* (*CRUD*) pada data member.

4. Halaman Member



Gambar 3.4 Halaman Member

Keterangan:

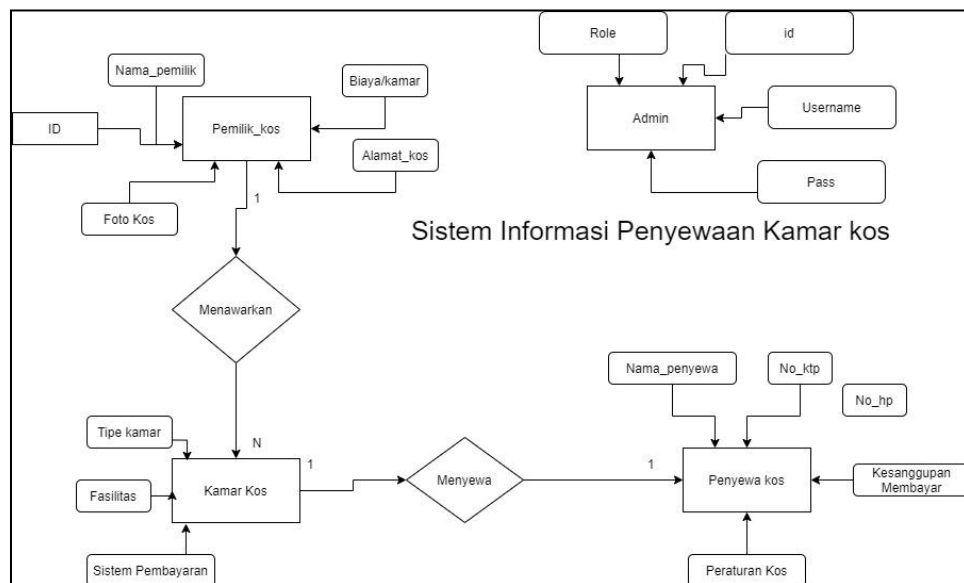
Jika berhasil login sebagai member maka akan masuk ke halaman ini. Halaman ini memuat tentang data member yang login, pada halaman ini juga bisa mengubah data member.

4.1 Pertemuan IV

Pengolahan data pembayaran masih menggunakan sistem konvensional atau manual yaitu melakukan pencatatan kedalam buku catatan pembayaran. Dengan pengolahan sistem manual, kendala yang dihadapi adalah pengecekan data penyewa yang telah membayar maupun yang belum, pengecekan data kamar yang kosong, dan pencarian data penyewa. Dari permasalahan tersebut, maka dibangun Sistem Informasi berbasis web agar dapat digunakan untuk membantu pemilik kos mengolah berbagai administrasi dan keuangan kos. Sistem ini dapat membantu calon penyewa memonitoring kamar kos yang telah terisi, rusak, maupun yang belum terisi, dapat membantu penyewa kos dalam pembayaran kos, dan dapat membantu pemilik kos dalam membuat laporan keuangan.

4.1.1 ERD

ERD adalah suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi.



Gambar 4.1 ERD Penyewaan Kamar Kos

Keterangan :

Berdasarkan Gambar 4.1, dapat dilihat bahwa Pemilik kos memiliki hubungan One To Many dengan Kamar kos yang artinya satu pemilik kos dapat menawarkan banyak kamar kos, begitupun sebaliknya, banyak kamar kos dapat ditawarkan oleh satu pemilik kos. Dan pada Kamar kos memiliki hubungan One To One dengan Penyewa kos yang dimana dapat diartikan bahwa satu kamar kos dapat disewa oleh satu penyewa dan juga sebaliknya, satu penyewa dapat menyewa satu kamar kos

Fungsi penggambaran *Entity Relationship Diagram (ERD)* yang ada saat ini yaitu sebagai berikut :

- 1) Untuk memudahkan kita dalam menganalisis pada suatu basis data atau suatu sistem dengan cara yang cepat dan murah.

- 2) Dapat dilakukan pengujian pada model yang telah dibuat dan dapat mengabaikan proses yang sudah dibuat hanya dengan menggambar *Entity Relationship Diagram (ERD)*.
- 3) Menjelaskan hubungan-hubungan antar data-data dalam basis data berdasarkan objek –objek dasar data yang memiliki hubungan yang dihubungkan oleh suatu relasi.
- 4) Untuk mendokumentasikan data-data yang ada dengan cara *mengidentifikasi* setiap *entitas* dari data-data dan hubungannya pada suatu *Entity Relationship Diagram (ERD)* itu sendiri.

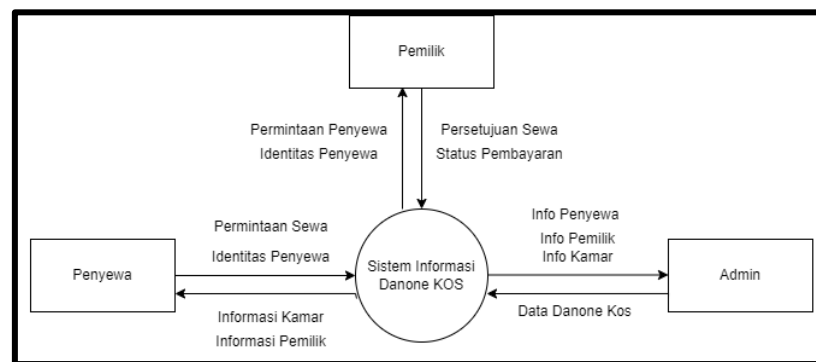
4.1.2 DFD

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.

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

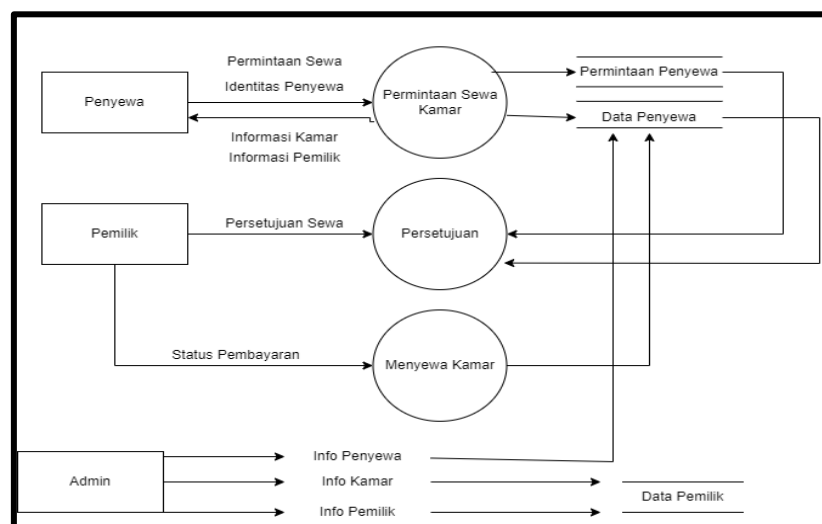
- a. Diagram level 0 diagram konteks) Diagram konteks atau level 0 merupakan diagram dengan tingkatan paling rendah, dimana menggambarkan sistem berinteraksi dengan entitas eksternal. Pada diagram konteks akan diberi nomor untuk setiap proses yang berjalan, dimulai dari angka 0 terlebih dahulu. Jadi, 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.



Gambar 4.2 DFD Level 0

- 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.



Gambar 4.3 DFD Level 1

Secara fundamental, terdapat tiga fungsi dari pembuatan diagram alir data untuk kebutuhan software development. Berikut ini merupakan penjelasan dari masing – masing fungsi di bawah ini.

1. Menyampaikan Rancangan Sistem

Dengan pembuatan DFD, maka proses penyampaian informasi menjadi lebih mudah dengan tampilan visual yang simple dan dapat dimengerti oleh tiap stakeholder. Dimana, data yang disajikan mampu menggambarkan alur data secara terstruktur dengan pendekatan yang lebih efisien.

2. Menggambarkan Suatu Sistem

Fungsi yang kedua, DFD dapat membantu proses penggambaran sistem sebagai jaringan fungsional. Maksudnya adalah, di dalam jaringan terdapat berbagai komponen yang saling terhubung menggunakan alur data.

3. Perancangan Model

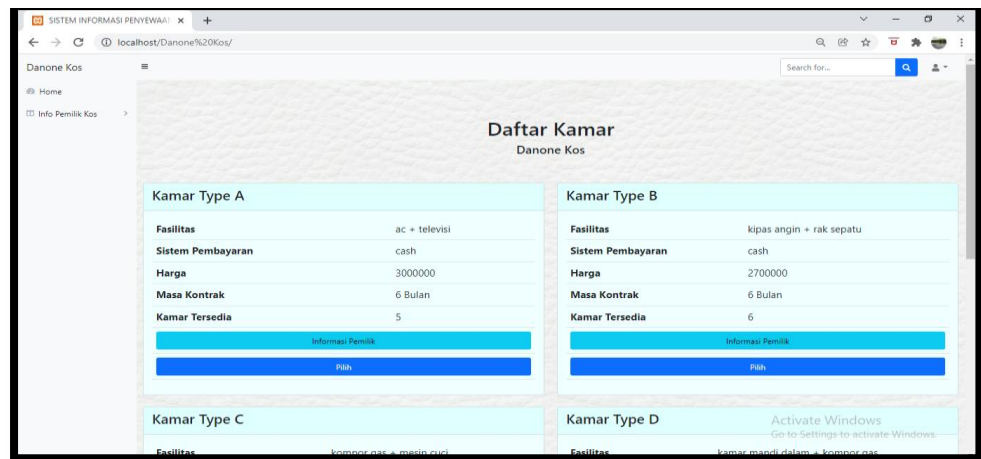
Fungsi yang terakhir, diagram ini juga dapat membuat rancangan model baru dengan menekankan pada fungsi sistem tertentu. Hal tersebut dapat dimanfaatkan untuk melihat bagian yang lebih detail dari diagram alir data tersebut.

4.1.3 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.

1. Halaman Utama

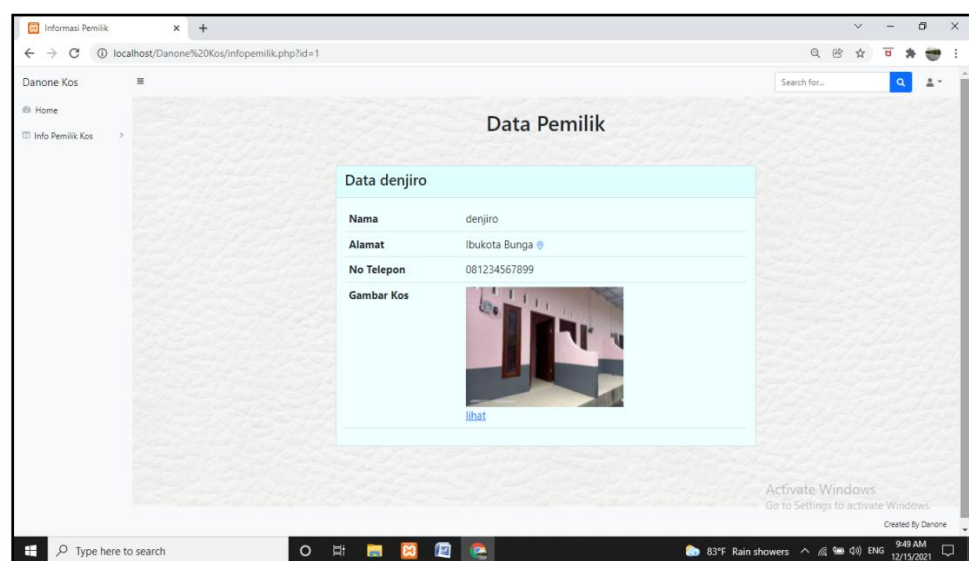


Gambar 4.4 halaman utama Kos Danone

Keterangan:

Pada halaman utama menampilkan pilihan kamar yang disertai dengan informasi kamar dan pemilik kosnya.

2. Halaman Informasi Pemilik Kos



Gambar 4.5 Halaman Data Pemilik

Keterangan:

Halaman ini memuat beberapa informasi tentang pemilik kos yaitu no telepon, gambar kos, alamat dan nama pemilik kos.

3. Halaman Permintaan Sewa Kos

Permintaan Sewa Kos

Peraturan Kos

1. selama masa sewa, penyewa kos berhak menempati kamar dan memanfaatkan fasilitas yang ada di dalam kos
2. menghemat listrik dengan cara mematikan lampu dan peralatan elektronik yang tidak digunakan, terutama saat akan meninggalkan kos
3. menjaga fasilitas kos dengan baik, apabila ada fasilitas atau bagian dari kamar kos yang rusak harap segera menghubungi pengelola kos
4. menjaga ketenangan di kamar kos

Untuk Menyewa Kamar Kos Silahkan Isi data diri Anda

[Kembali](#)

No KTP:

No Telepon:

Kesanggupan Membayar:

Masa Kontrak :
6 Bulan Rp 2900000

Setelah Mengklik "Sewa Kamar Ini" silahkan lakukan pembayaran pada pemilik kos sebesar Rp 2900000
Kunci kamar kos akan diberikan oleh pemilik kos setelah anda menyelesaikan proses pembayaran.

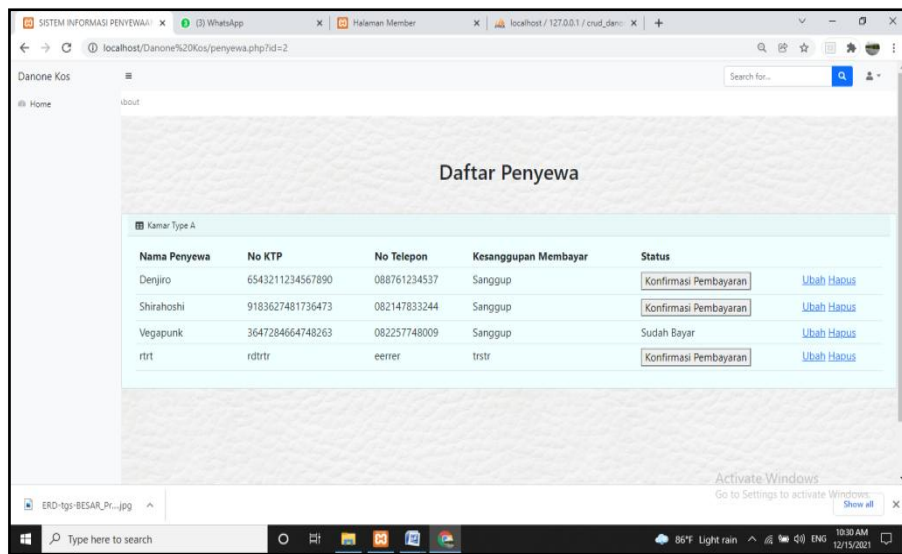
Windows 10

Gambar 4.6 Halaman Permintaan Sewa Kos

Keterangan:

Pada Halaman Permintaan sewa ini menampilkan peraturan kos dan jika ingin menyewa kamar kos maka harus mengisi data diri terlebih dahulu.

4. Halaman penyewa



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	082237748009	Sanggup	Sudah Bayar Ubah Hapus
rtrtr	rdtrtr	eerrrr	trstr	Konfirmasi Pembayaran Ubah Hapus

Gambar 4.7 Halaman Daftar Penyewa

Keterangan:

Pada Halaman ini berisi data data penyewa, pada halaman ii bisa dilihat juga penyewa yang sudah melunasi biaya sewa dan yang belum.

DAFTAR PUSTAKA

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, Rieke. 2001. Framework Crud. Andi Offset. Yogyakarta

<https://smkn1panjalu.sch.id/pengertian-pemrograman-berorientasi-objek-pbo/>

<https://waesalgorny.blogspot.com/2014/11/definisi-class-object-methodatribut.html>

<https://www.niagahoster.co.id/blog/laravel-adalah/>

<https://wikishare27.wordpress.com/pengertian-class-method-constructor-modifierobject-pada-java/>

<https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-class-objectproperty-dan-method/>

<https://www.petanikode.com/java-oop-constructor>