

LAPORAN LENGKAP
PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK



OLEH :

NAMA : FATHIRA AZZAHRA

NIM : F1G120003

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 : FATHIRA AZZAHRA

NIM : F1G120003

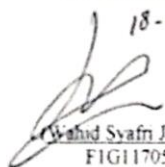
Laporan praktikum pemrograman berorientasi objek (PBO) ini disusun untuk memenuhi tugas akhir menyelesaikan kegiatan praktikum pemrograman berorientasi objek (PBO), dan disusun sebagai salah satu syarat lulus mata kuliah pemrograman berorientasi objek (PBO).

Kendari, 18 Desember 2021

Menyetujui :

Asisten Praktikum

18-12-2021


(Wahid Syafri Javanto)
F1G117059

Praktikan


(Fathira Azzahra)
F1G120003

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 juga laporan ini tidak lepas dari kekurangan. Oleh karenanya, kami mengharap kritik dan saran dari pengguna laporan ini. Sekian terimakasih,

wabillahirrahmatilrahman walrahim, WassalamuAlaikum Warahumatullahi Wabarakatu.

Kendari, Desember 2021

Penulis

DAFTAR ISI

LAPORAN LENGKAP	i
HALAMAN PENGESAHAN LAPORAN PRAKTIKUM... Error! Bookmark not defined.	
KATA PENGANTAR.....	ii
DAFTAR ISI.....	iv
DAFTAR TABEL.....	vii
DAFTAR GAMBAR.....	viii
PRAKTIKUM 1	1
1.1 Alat dan bahan.....	1
1.2 Pengenalan <i>PBO</i>	1
1.3 Pengenalan <i>PHP</i>	3
1.3.1 Sejarah Bahasa Pemograman <i>PHP</i>	3
1.3.3 Kekurangan Bahasa Pemograman <i>PHP</i>	6
PRAKTIKUM 2	8
2.1 <i>Class</i>	8
2.2 <i>Method</i>	8
2.3 <i>Property</i>	9
2.4 <i>Object</i>	10

2.5 Constructor.....	10
2.6 Laravel.....	11
PRAKTIKUM 3	13
3.1 Model data berbasis object	13
3.1.1 Model Data <i>ERD</i> (<i>Entity Relationship Diagram</i>).....	13
3.1.2 Model Data <i>Semantic</i>	13
3.2 Model Data Berbasis <i>Record</i>	14
3.2.1 Model <i>Database Hirarki</i>	14
3.2.2 Model <i>Database Jaringan</i>	15
3.2.3 Model <i>Database Relational</i>	15
3.3 <i>CRUD</i>	16
3.3.1 <i>Create</i>	16
3.3.2 <i>Read</i>	17
3.3.3 <i>Update</i>	17
3.3.4 <i>Delete</i>	18
3.4 Penjelasan Projek <i>CRUD</i>	18
3.4.1 Halaman <i>login</i>	18
3.4.2 Halaman <i>Member</i>	19
3.4.3 Halaman <i>Manager</i>	19
PRAKTIKUM 4	21

4.1. <i>Project akhir (CRUD Sistem Informasi Penyewaan Kamar Kos)</i>	27
4.2 Entity Relationship Diagram	21
4.2.1 <i>ERD Sistem Penyewaan Kamar Kos</i>	21
4.3 <i>Data Flow Diagram</i>	22
4.3.1 <i>Diagram Level 0 (Diagram Konteks)</i>	23
4.2.2 <i>Data Flow Diagram Level 1</i>	24
4.4 <i>Interface</i>	26
DAFTAR PUSTAKA	33

DAFTAR TABEL

Tabel 1.1 Alat dan Bahan.....	1
--------------------------------------	---

DAFTAR GAMBAR

Gambar 3.1 <i>Entity Relationship Diagram</i>	13
Gambar 3.2 <i>Sematic Model</i>	14
Gambar 3.3 <i>Hirarki Model</i>	14
Gambar 3.4 <i>Model Database Jaringan</i>	15
Gambar 3.5 <i>Model Database Relational</i>	15
Gambar 3.6 <i>Halaman Login</i>	18
Gambar 3.7 <i>Halaman Member</i>	19
Gambar 3.8 <i>Halaman Manager</i>	19
Gambar 4.1 <i>ERD Sistem Penyewaan Kamar Kos</i>	22
Gambar 4.2 <i>Diagram Konteks</i>	24
Gambar 4.3 <i>Diagram Flow Level</i>	25
Gambar 4.4 <i>Halaman Login member</i>	27
Gambar 4.5 <i>Tampilan Registrasi Member</i>	28
Gambar 4.6 <i>Tampilan Halaman Welcome Member</i>	28
Gambar 4.7 <i>Tampilan Halaman Daftar Kos</i>	29
Gambar 4.8 <i>Tampilan Daftar Menyewa Kamar</i>	30
Gambar 4.9 <i>Tampilan Welcome Admin</i>	30
Gambar 4.10 <i>Tampilan Halaman Pemilik Kos</i>	31
Gambar 4.11 <i>Tampilan Halaman Daftar Kamar Kos</i>	31
Gambar 4.12 <i>Tampilan Data Pembayaran Sewa</i>	32

PRAKTIKUM 1

1.1 Alat dan Bahan.

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

Alat Dan Bahan	Penjelasan
Laptop	Sebagai tempat untuk menyimpan data untuk mengerjakan proyek 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 dibuat.

Tabel 1.1 Alat dan Bahan

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 memecahkan 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.

1.3 Pengenalan *PHP*

1.3.1 Sejarah Bahasa Pemograman *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*.

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-*instal*

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. Kelebihan Bahasa Pemrograman *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, 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 menggunakannya atau setidaknya pernah sekedar mencobanya. Tutorial untuk memulai belajar *PHP* pun dengan mudah ditemukan dengan mengetikkan kata kunci pada mesin pencari.

4. Sempel

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.

1.3.3 Kekurangan Bahasa Pemograman *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 mempunyai 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*.

PRAKTIKUM 2

2.1 Class

Class adalah cetakan atau *blueprint* dari objek. Di dalam *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

}

?>
```

2.2 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.

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
    }
}

?>
```

2.3 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 $merek;
    var $ukuran;
}

?>
```

2.4 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_duniaikom, dan lain-lain.. Objek dari *class* laptop akan memiliki seluruh ciri-ciri laptop, yaitu *property* dan *method*.

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();
?>
```

2.5 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() )
    }
}
```

2.6 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*. Ditahun 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*.

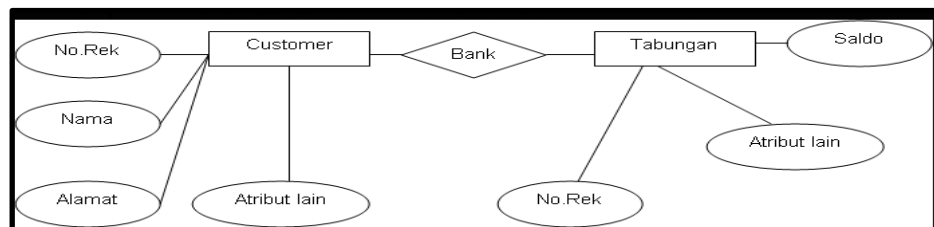
PRAKTIKUM 3

3.1 Model data berbasis object

Model data berbasis objek ini adalah model data yang menyiapkan setiap node / chartnya dengan basis objek database. Dengan menggunakan konsep seperti *entitas*, *attribute* dan *relasi*, objek yang dimaksud adalah sebuah *entitas*.

3.1.1 Model d\ata *ERD (Entity Relationship Diagram)*.

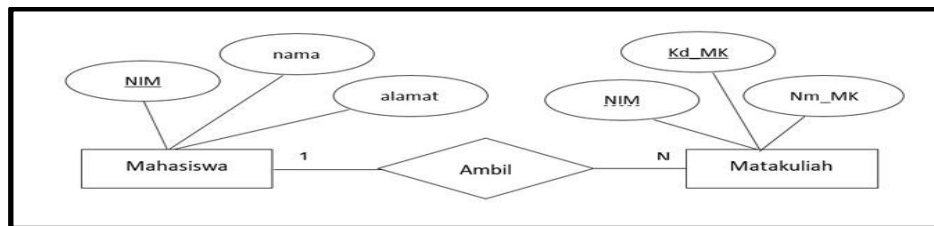
ERD adalah salah satu model data berbasis objek yang paling sering digunakan. Jenis dan bentuk *ERD* dari tahun ke tahun pun berbeda beda. *ERD* adalah cara penggambaran *real case* yang terjadi sesuai kasusnya. Dengan *ERD* kita bisa menggambarkan bagaimana *entitas* satu bisa terhubung dengan *entitas* lainnya.



Gambar 3.1 *Entity Relationship Diagram*

3.1.2 Model Data *Semantic*

Model data *semantic* adalah relasi antar obyek yg dinyatakan dengan kata kata (*semantic*).



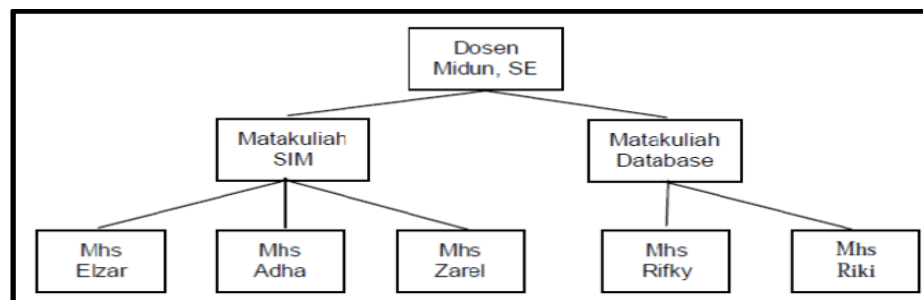
Gambar 3.2 Sematic model

3.2 Model Data Berbasis *Record*

Model data ini berbeda dari model data berbasis objek. Model data ini mengambil nodenya berdasarkan *record-record* yang di perlukan dari database. *Record* sendiri adalah rekaman-rekaman data yang tersimpan di database. Contoh-contoh model data berbasis *record* yaitu :

3.2.1 Model Database *Hirarki*

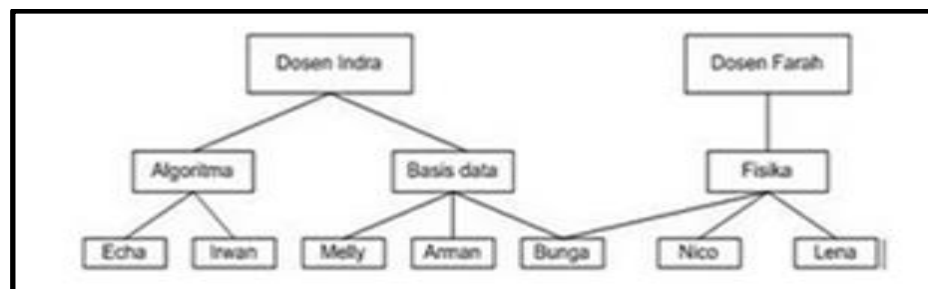
Model data ini disajikan dari kumpulan *record* dan relasi yang digambarkan seperti bentuk pohon (*tree*). Model data ini memungkinkan satu *node* hanya untuk memiliki satu orang tua.



Gambar 3.3 Hirarki Model

3.2.2 Model Database Jaringan

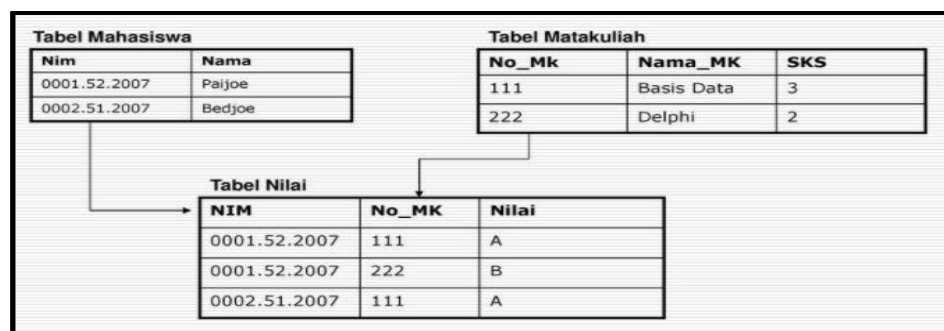
Network data model hampir menyerupai model data hirarki. Namun pada model data ini, memungkinkan satu node bisa memiliki lebih dari satu orang tua.



Gambar 3.4 Model Database Jaringan

3.2.3 Model Database Relational.

Model *database* yang disajikan dalam bentuk tabel yang terdiri dari kolom dengan nama yang unik dan baris-baris yang menyimpan data yang berbeda. Model data ini digambarkan berdasarkan *recordnya* dan yang paling sering digunakan untuk memudahkan perancangan sebuah database.



Gambar 3.5 Model Database Relational

3.3 *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. Nantinya, akan ditampilkan dalam *browser* atau aplikasi pada perangkat komputer *user*. Terdapat empat *poin* penting dari akronim fungsi *CRUD* untuk mengembangkan perangkat lunak, baik berbasis *web* maupun *mobile*.

3.3.1 *Create*

Fungsi *CRUD* yang pertama adalah *create*, dimana anda dapat memungkinkan untuk membuat *record* baru pada sistem basis data. Jika anda sering menggunakan *SQL*, maka sering disebut dengan istilah *insert*.

Sederhananya, Anda dapat membuat tabel atau data baru sesuai atribut dengan memanggil fungsi *create*. Akan tetapi, biasanya hanya

posisi *administrator* saja yang dapat menambahkan atribut lain ke dalam tabel itu sendiri.

3.3.2 Read

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

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

3.3.3 Update

Fungsi *CRUD* yang ketiga adalah *update*, dimana berfungsi untuk memodifikasi data atau *record* yang telah tersimpan di dalam *database*. Namun, Anda perlu untuk mengubah beberapa informasi terlebih dahulu agar dapat mengubah *record* sesuai kebutuhan Anda.

Untuk pengisian *update data* Anda juga perlu menyesuaikan nilai atribut sesuai dengan *form* yang tersedia agar tidak ada kesalahan saat pemrosesan data di dalam *server*.

3.3.4 Delete

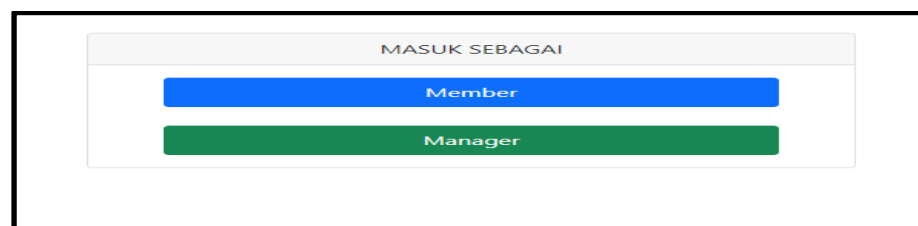
Fungsi yang terakhir adalah *delete*, dimana ketika Anda tidak membutuhkan sebuah *record* lagi, maka data tersebut perlu untuk dihapus. Sehingga, Anda perlu untuk menggunakan fungsi *delete* untuk memproses aktivitas tersebut.

Beberapa *software* terkait *database* relasional mengizinkan anda untuk menggunakan *soft* dan *hard delete*. Untuk *soft delete* berfungsi untuk memperbarui status baris yang menunjukkan bahwa data akan dihapus meskipun informasi tersebut tetap ada.

3.4 Penjelasan Proyek *CRUD*

Disini saya akan menjelaskan proyek saya tentang *CRUD member* dan golongan serta saya akan menampilkan gambar beserta keterangannya.

3.4.1 Halaman *login*

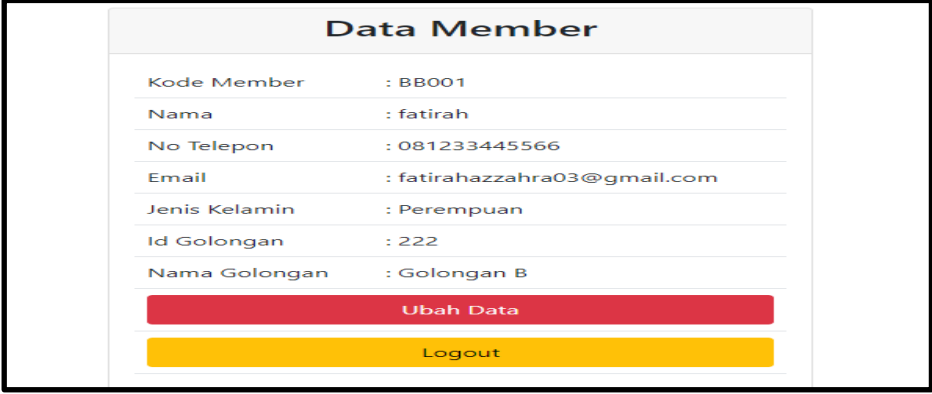


Gambar 3.6 Halaman *Login*

Keterangan :

Pada halaman ini kita diminta untuk masuk melalui *member* maupun *manager*

3.4.2 Halaman *Member*



The screenshot shows a web form titled "Data Member". It contains several input fields with pre-filled data, followed by two action buttons: "Ubah Data" (red) and "Logout" (yellow).

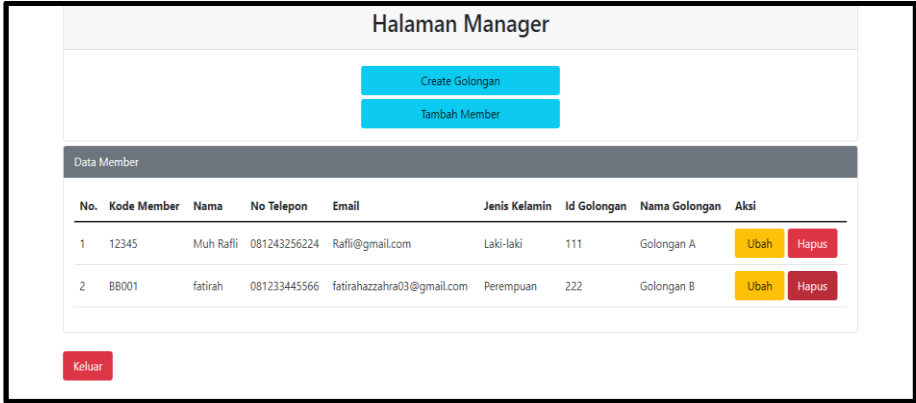
Data Member	
Kode Member	: BB001
Nama	: fatirah
No Telepon	: 081233445566
Email	: fatirahazzahra03@gmail.com
Jenis Kelamin	: Perempuan
Id Golongan	: 222
Nama Golongan	: Golongan B
Ubah Data	
Logout	

Gambar 3.7 Halaman *Member*

Keterangan :

Ketika kita masuk sebagai *member* kita akan di arahkan ke halaman *member* dimana pada halaman *member* terdapat data data dari *member* itu sendiri baik dari kode *member*, nama *member* dll. Di halaman *member* terdapat tombol untuk mengubah data *member* dan *log out* untuk keluar dari halaman *member* itu sendiri.

3.4.3 Halaman *Manager*



The screenshot shows a web interface titled "Halaman Manager". It features two blue buttons at the top: "Create Golongan" and "Tambah Member". Below these is a table titled "Data Member" with columns for No., Kode Member, Nama, No Telepon, Email, Jenis Kelamin, Id Golongan, Nama Golongan, and Aksi. The table contains two rows of member data. At the bottom left, there is a red "Keluar" button.

Halaman Manager								
Create Golongan								
Tambah Member								
Data Member								
No.	Kode Member	Nama	No Telepon	Email	Jenis Kelamin	Id Golongan	Nama Golongan	Aksi
1	12345	Muh Rafli	081243256224	Rafli@gmail.com	Laki-laki	111	Golongan A	Ubah Hapus
2	BB001	fatirah	081233445566	fatirahazzahra03@gmail.com	Perempuan	222	Golongan B	Ubah Hapus

[Keluar](#)

Gambar 3.8 Halaman *Manager*

Keterangan :

Ketika masuk sebagai *manager* kita akan di arahkan ke halaman *manager*, pada halaman *manager* terdapat tombol untuk menambahkan golongan dan *member*. Kemudian pada halaman *manager* terdapat informasi mengenai kode *member*, nama no telpoon, email, jenis kelamin, id golongan, nama golongan dan aksi. Aksi ini dia berfungsi untuk kita dapat mengubah data di halaman *manager* ataupun menghapus data dihalaman *manager*, serta dihalaman *manager* juga terdapat tombol keluar yang berfungsi untuk kita dapat keluar dari halaman *manager* tersebut

PRAKTIKUM 4

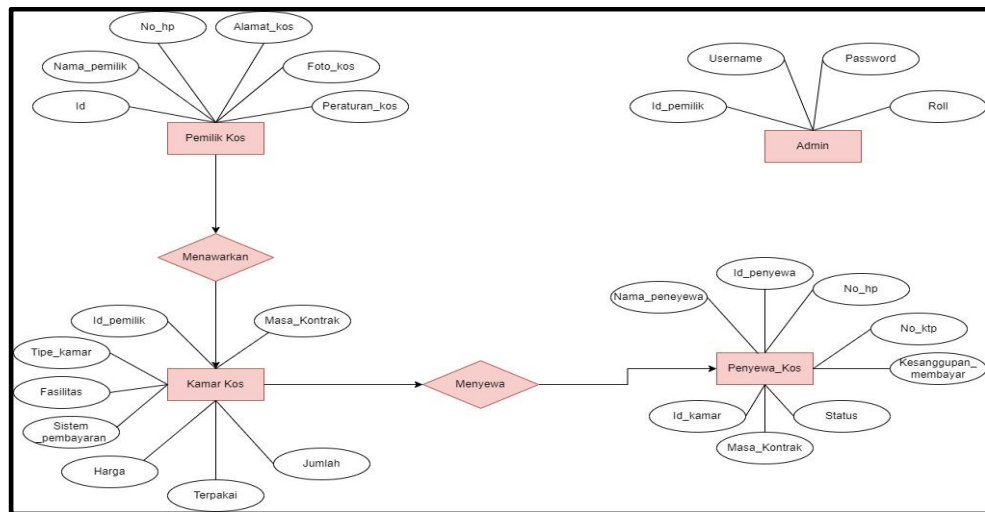
Pada kegiatan praktikum 4 menjelaskan *project* mengenai aplikasi berbasis *web* tentang kos dan disini juga saya akan menampilkan gambar pada *project* ini beserta penjelasan dan disertai dengan erdnya.

4.1 Entity Relationship Diagram

ERD adalah pemodelan data atau sistem dalam *database* yang sudah sering digunakan oleh banyak lembaga. Fungsinya *ERD* adalah untuk memodelkan struktur dan hubungan antar data yang relatif kompleks. Keberadaan sistem *ERD* sangat penting untuk perusahaan dalam mengelola data yang dimilikinya.

4.2.1 ERD Sistem Penyewaan Kamar Kos

Pada pembuatan sistem *CRUD*, pertama kita membuat model data berbasis objek terlebih dahulu, dimana fungsi dari model data ini yaitu untuk untuk menjelaskan hubungan antar data dalam basis data berdasarkan persepsi bahwa *real world* terdiri dari objek-objek dasar yang mempunyai hubungan / relasi antara objek tersebut.



Gambar 4.1 ERD Sistem Penyewaan Kamar Kos

Gambar 4.1 menjelaskan model data yang digunakan yaitu *Entity Relationship model (ERD)*, Merupakan model untuk menjelaskan hubungan antar data dalam basis data berdasarkan persepsi bahwa *real world* terdiri dari objek-objek dasar yang mempunyai hubungan / relasi antara objek tersebut.

ERD project , terdapat 4 entitas yaitu *admin*, pemilik kos, kamar kos, penyewa kos yang mana tiap entitas terdapat atribut-atribut. *ERD* diatas juga merupakan kardinalitas relasi banyak ke banyak (*Many to Many*) setiap elemen dari entitas A berhubungan maksimal banyak elemen pada entitas B demikian sebaliknya.

4.2 Data Flow Diagram

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.

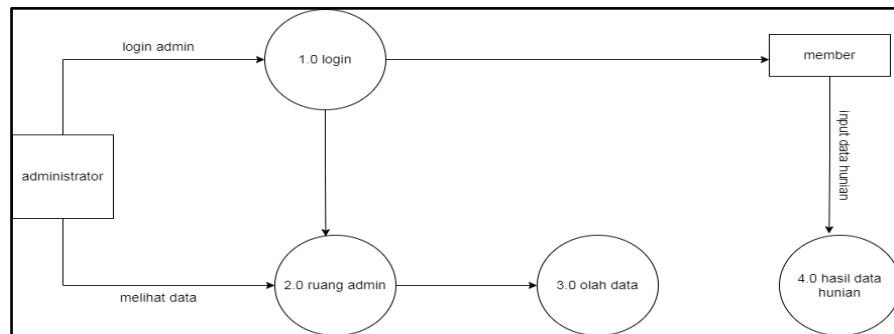
DFD sendiri juga digunakan untuk memberikan gambaran sistem secara keseluruhan hingga batasan sistem, sumber-sumber dan tujuan data, proses data, arus data dan media penyimpanan dengan memanfaatkan simbol-simbol dalam DFD. Sehingga DFD ini dapat menggambarkan analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program.

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

4.3.1 Diagram Level 0 (Diagram Konteks)

Diagram level 0 atau bisa juga diagram konteks adalah *level* diagram paling rendah yang menggambarkan bagaimana sistem berinteraksi dengan *external* entitas. Pada diagram konteks akan diberikan nomor untuk setiap proses yang berjalan, umumnya mulai dari angka 0 untuk start awal. Semua entitas yang ada pada *diagram* konteks termasuk juga aliran datanya akan langsung diarahkan kepada sistem. Pada *diagram* konteks ini juga tidak ada informasi tentang data

yang tersimpan dan tampilan diagramnya tergolong sederhana.



Gambar 4.2 *Diagram Level 0*

Berdasarkan **Gambar 4.1** *DFD Level 0*, tergambar jelas bahwa administrator bisa melakukan proses *login* sebagai *admin* dan sebagai *member*, jika administrator melakukan proses *login* sebagai *admin* dan selanjutnya ketika *login* berhasil maka administrator berubah jadi *admin* dan diarahkan ke proses ruangan *admin* kemudian bisa mengelolah data, sedangkan jika administrator *login* sebagai *member* maka administrator berubah menjadi *member* dan bisa mengimput data hunian yang akan disewa yang mana data hunian tersebut merupakan hasil olah dari *admin*. Disini juga tergambar bahwa administrator dapat melihat data hasil olah yang dilakukan *admin* tanpa melakukan *login*.

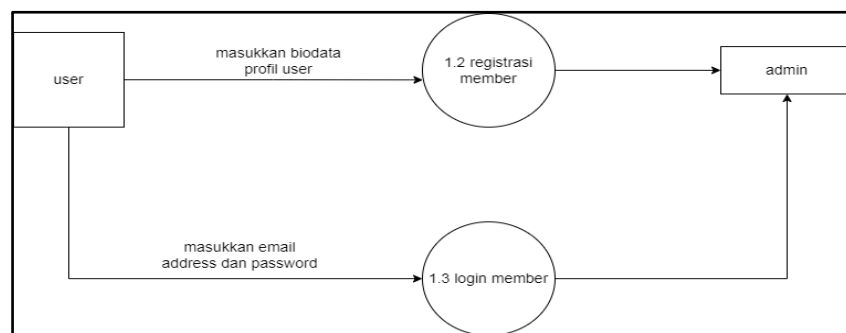
4.2.2 Data Flow Diagram Level 1

DFD level 1 adalah tahapan lebih lanjut tentang DFD *level 0*, dimana semua proses yang ada pada DFD level 0 akan dirinci dengan

lengkap sehingga lebih lengkap dan detail. Proses-proses utama yang ada akan dipecah menjadi sub-proses.

a. *Data Flow Diagram Level 1*

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



Gambar 4.3 *Diagram Flow Level 1*

Berdasarkan **Gambar 4.3** *DFD Level 1*, tergambar jelas proses terjadinya proses registrasi *member* yang mana *user* jika ingin melakukan proses registrasi *member* maka harus memasukkan biodata *profil user* terlebih dahulu kemudian data tersebut akan dikelola oleh *admin*.

Pada gambar diatas juga tergambar jelas proses terjadinya *login member* yang dilakukan setelah proses registrasi *member* berhasil yang mana *user* jika ingin melakukan proses *login* harus

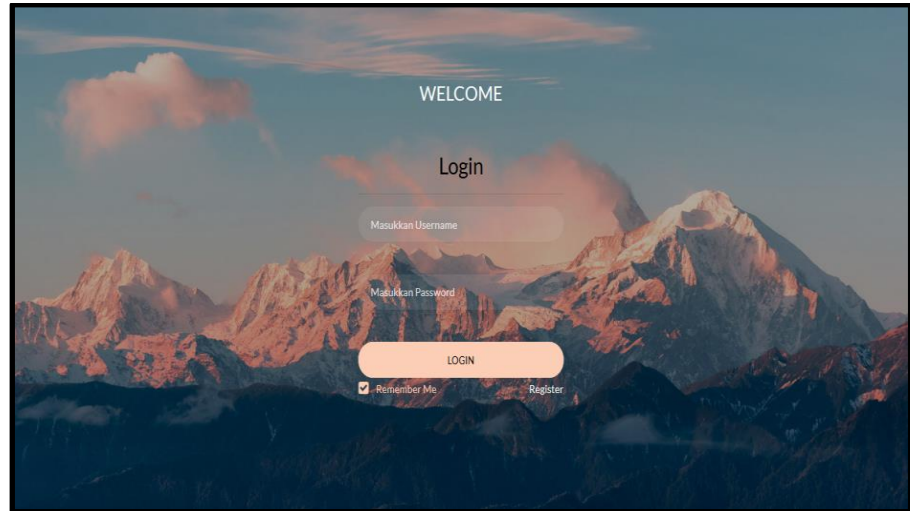
memsukkan *email address* dan *password* terlebih dahulu dan datanya juga akan dikelola oleh *admin*.

4.3 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). 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 diimplementasikan *interface* penyanyi

4.4 Project akhir (CRUD Sistem Informasi Penyewaan Kamar Kos)

a. Tampilan Halaman *Login Member*



Gambar 4.4 Halaman *Login Member*

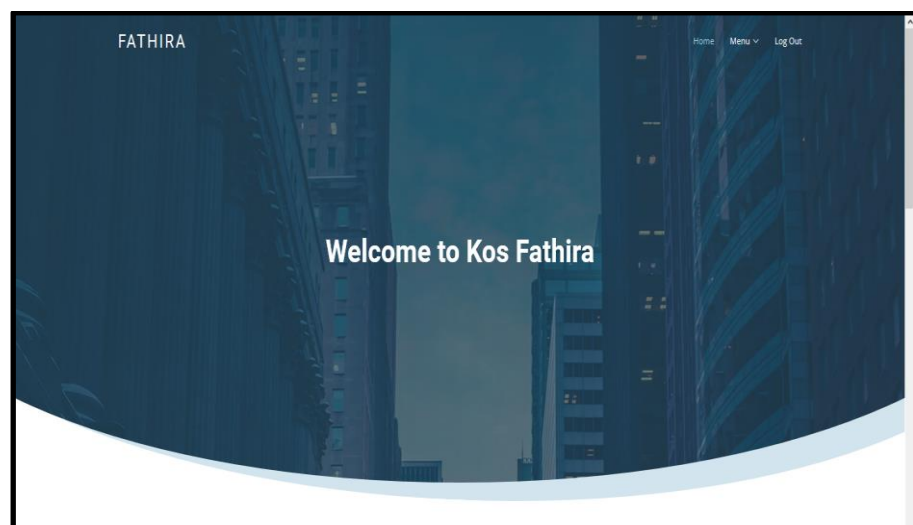
Selanjutnya ketika kita mengklik tulisan *login* pada tampilan awal tadi maka kita akan diarahkan ke tampilan *login member*. Pada tampilan *login member* kita bisa langsung *login* jika kita sudah memiliki akun dengan cara mengisi *username* dan *password*, tetapi jika kita belum memiliki akun maka kita harus registrasi *member* terlebih dahulu, adapun tampilan registrasi *member* sebagai berikut :

A screenshot of a web registration form titled "Register". The form is centered on a blue background. It contains several input fields: "Username", "Password", "Konfirmasi Password", "Nama:", "No KTP:", "No Telepon:", and "Kesanggupan Membayar:". At the bottom of the form is a blue button labeled "Register".

Gambar 4.5 Tampilan Registrasi *Member*

Pada tampilan registrasi *member* disini kita diarahkan untuk mengisi data diri seperti *username*, *password*, konfirmasi *password*, nama, no ktp, no hp, dan kesanggupan membayar. Jika kita sudah mengisi data diri maka bisa langsung daftar dan akan otomatis memiliki akun.

b. Tampilan Halaman *Welcome Member*



Gambar 4.6 Tampilan Halaman *Welcome Member*

Setelah kita memiliki akun dan berhasil *login member* maka kita langsung diarahkan ketampilan halaman *member*. Pada tampilan member ini \ terdapat *button home*, *button menu* yang terdiri daftar kamar kos, *button log out*. Ketika kita mengklik menu bagian daftar kamar kos maka akan muncul tampilan daftar kamar kos.

Daftar Kamar Kos	
Kamar Type A	
Fasilitas	ac + televisi
Sistem Pembayaran	cash
Harga	3000000
Masa Kontrak	6 Bulan
Kamar Tersedia	4
	Informasi Lebih
	Pilih
Kamar Type B	
Fasilitas	tipex angin + rak sepatu
Sistem Pembayaran	cash
Harga	2700000
Masa Kontrak	6 Bulan
Kamar Tersedia	6
	Informasi Lebih
	Pilih
Kamar Type C	
Fasilitas	kompas gas + mesin cuci
Sistem Pembayaran	cash
Harga	2500000
Masa Kontrak	6 Bulan
Kamar Tersedia	4
	Informasi Lebih

Gambar 4.7 Tampilan Halaman Daftar Kos

Pada tampilan daftar kamar kos dihalaman *member* terdapat beberapa *type* kamar kos yang dimana setiap *type* kamar kos memiliki fasilitas, sistem pembayaran, harga, masa kontrak, jumlah kamar serta pemilik yang berbeda. Ketika kita ingin memilih kamar kos *type A* kita langsung mengklik “Pilih”. Setelah mengklik “Pilih” langsung muncul peraturan kos dan mengisi data diri seperti gambar berikut.

Peraturan Kos

1. menghemat air dengan menutup keran setelah menggunakan air
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. mengunci kamar apabila meninggalkan kos

Untuk Menyewa Kamar Kos Silahkan Isi data diri Anda

- Nama Penyewa:
- No KTP:
- No Telepon:
- Kemampuan Membayar:
- Masa Kontrak : 6 Bulan Rp 3000000

[Sewa Kamar Ini](#)

Setelah Mengklik "Sewa Kamar Ini" silahkan lakukan pembayaran pada pemilik kos sebesar Rp 3000000
 Kunci kamar kos akan diberikan oleh pemilik_kos setelah anda menyelesaikan proses pembayaran.
 Untuk melihat informasi pemilik kos silahkan klik link dibawah
[informasi pemilik kos](#)

[Kembali](#)

Gambar 4.8 Tampilan Daftar Menyewa Kamar

c. Halaman *Welcome Admin*

WELCOME

Login

Masukkan Username

Masukkan Password

[LOGIN](#)

☒ Remember Me [Register](#)

Gambar 4.9 Halaman *Welcome Admin*

Selanjutnya ketika kita mengklik tulisan *login* pada tampilan awal *login admin* tadi maka kita akan diarahkan ke tampilan selamat datang di halaman admin. Pada tampilan ini kita bisa melihat keseluruhan data kos kita, mulai dari pemilik kos, daftar kamar kos, data penyewa kos. Berikut ini data detail tampilan keseluruhan data kos :

d. Tampilan halaman pemilik kos



Gambar 4.10 Tampilan Halaman Pemilik Kos

Pada tampilan data hunian kita bisa menambah data hunian kita, mengubah data hunian bahkan kita bisa menghapus data hunian kita.

e. Tampilan Halaman Daftar Kamar Kos



Gambar 4.11 Tampilan Halaman Daftar Kamar Kos

Pada tampilan daftar kamar kos di halaman *member* terdapat beberapa *type* kamar kos yang dimana setiap *type* kamar kos memiliki fasilitas, sistem pembayaran, harga, masa kontrak, jumlah kamar serta pemilik yang berbeda

f. Tampilan Data Pembayaran Sewa

Kamar					
Nama Penyewa	No KTP	No Telepon	Kewangggan Membayar	Status	
Juslok	6543211234567890	08761234567	Sanggup	Konfirmasi Pembayaran	Udah Bayar
Lee Min Ho	1234567890123456	08123456789	Sanggup	Sudah Bayar	Udah Bayar
Iminidii	7654321098765432	08234567890	Sanggup	Konfirmasi Pembayaran	Udah Bayar
Thv	8263540210987654	082441431410	Sanggup	Konfirmasi Pembayaran	Udah Bayar
Song Jong Ki	9183427481736473	082147833244	Sanggup	Konfirmasi Pembayaran	Udah Bayar
Hope	3647284664748263	082257748009	Sanggup	Sudah Bayar	Udah Bayar
Jin	6543211234567888	087854321200	Sanggup	Konfirmasi Pembayaran	Udah Bayar
Suga	983235264787625	082364738763	Sanggup	Sudah Bayar	Udah Bayar
Ran	8742683754732762	37623654454	Mungkin	Sudah Bayar	Udah Bayar
Kim Soo Hyun	7654321098765432	082249140632	Sanggup	Konfirmasi Pembayaran	Udah Bayar
Suho	24323	243243243	trush	Konfirmasi Pembayaran	Udah Bayar

Gambar 4.12 Tampilan Data Pembayaran Sewa

Pada tampilan data pembayaran disini kita bisa melihat keseluruhan data pembayaran sewa dari seluruh *member*. Disini kita juga bisa mengedit, tetapi bukan untuk mengedit data melainkan untuk mengedit status pembayaran.

DAFTAR PUSTAKA

Iksanudin, M.S. 2018. Pemrograman Berbasis Objek Modern. Jakarta.

Yamamah. "Makalah OOP pada PHP". 22 Oktober 2015.

<https://mayazyaze.wordpress.com/2010/07/03/>

[makalah-oop-pada-php/](#)

Yudhanto, Y., Prasetyo.H.A. 2018. Panduan Mudah Belajar Framework Laravel.

Gramedia. Jakarta: 17

<https://www.sekawanmedia.co.id/blog/pengertian-crud/>