

**LAPORAN
PEMOGRAMAN BERORIENTASI OBJEK**



OLEH:

NAMA : FUAD AQIL ALBAARI

NIM : F1G120005

KELOMPOK : I (SATU)

ASISTEN PENGAMPU:

WAHID SAFRI JAYANTO

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HALU OLEO

2021

HALAMAN PENGESAHAN

LAPORAN PRAKTIKUM



OLEH:

FUAD AQIL ALBAARI (F1G120005)

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

Kendari, 2 Desember 2021

Asisten Pratikum,
20 - 12 - 2021
Wand Safri Jayanto
F1G117059

Pratikan,
12 -
Fuad Aqil Albaari
F1G120005

DAFTAR ISI

LAPORAN PEMOGRAMAN BERORIENTASI OBJEK	i
HALAMAN PENGESAHAN	ii
DAFTAR ISI.....	iii
DAFTAR GAMBAR	vi
DAFTAR TABEL.....	vii
KATA PENGANTAR	viii
PRATIKUM 1.....	1
1.1 Pendahuluan	1
1.2 Waktu dan Tempat	2
1.3 Alat dan Bahan.....	2
1.4 Pengenalan <i>Object Oriented Programming</i> (OOP)	2
1.5 Prinsip OOP.....	3
1.5.1 <i>Encapsulation</i>	3
1.5.2 <i>Abstraction</i>	3
1.5.3 <i>Inheritance</i>	3
1.5.4 <i>Polymorphism</i>	4
1.6 Kelebihan OOP Pada PHP	4
1.7 Kekurangan OOP pada PHP	5
PRATIKUM 2.....	6
2.1 Pengertian Class, Object, Property, dan Method	6
2.1.1 <i>Class</i>	6
2.1.2 <i>Property</i>	6
2.1.3 <i>Method</i>	7

2.1.4 <i>Object</i>	7
2.2 Enkapsulasi (<i>Encapsulation</i>).....	8
2.2.1 <i>Public</i>	8
2.2.2 <i>Protected</i>	9
2.2.3 <i>Private</i>	9
2.3 <i>Constructor</i> dan <i>Destructor</i>	10
2.3.1 <i>Constructor</i>	10
2.3.2 <i>Destructor</i>	10
2.4 <i>Interfaces</i>	11
PRATIKUM 3.....	12
3.1 Model data berbasis objek.....	12
3.1.1 <i>Entity Relationship model (ERD)</i>	12
3.1.2 <i>Semantic</i> model.....	12
3.2 Model data berbasis <i>record</i>	13
3.2.1 <i>Relational</i> Model	13
3.2.2 <i>Hierarchical</i> model.....	13
3.2.3 <i>Network</i> Model	14
3.3 <i>Entity</i> (Entitas).....	14
3.4 Atribut	14
3.4.1 Jenis Atribut.....	14
3.5 <i>Relationship</i>	15
3.6 Pengenalan <i>Crud</i>	16
3.6.1 <i>Create</i>	16
3.6.2 <i>Read</i>	17
3.6.3 <i>Update</i>	18

3.6.4 <i>Delete</i>	20
PRATIKUM 4	21
4.1 Membuat Sistem Penyewaan Kamar Kos	21
4.1.1 ERD Sistem Penyewaan Kamar Kos	21
4.1.2 <i>Data Flow Diagram (DFD)</i>	22
4.1.3 User Interfaces	24
4.1.4 <i>Login</i>	25
4.1.5 <i>Dashboard</i>	26
4.2 Menampilkan Data Yang Berelasi.....	27
DAFTAR PUSTAKA	30

DAFTAR GAMBAR

Gambar 3.1 ERD	13
Gambar 3.2 Sematic model	13
Gambar 3.3 Relation model.....	14
Gambar 3.4 Hierarchical model	14
Gambar 3.5 Network model	15
Gambar 3.6 Create.....	18
Gambar 3.7 Read.....	19
Gambar 3.8 Update.....	19
Gambar 4.1 ERD Sistem Penyewaan Kamar Kos.....	19
Gambar 4.2 Diagram Konteks.....	23
Gambar 4.3 Diagram Flow Level 1	23
Gambar 4.4 Login	25
Gambar 4.4 Gagal Login	25
Gambar 4.5 Dashboard.....	26
Gambar 4.6 Read Data Penyewa	26
Gambar 4.7 Tambah Data	27
Gambar 4.8 Update Data	27
Gambar 4.9 Inner Join Table Pemilik & Table Kamar.....	28
Gambar 4.10 Tambah Penyewa.....	28
Gambar 4.11 Tampilan Penyewa Kamar	29

DAFTAR TABEL

Table I. Alat & Bahan	2
--	----------

KATA PENGANTAR

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi maha penyayang. Kami panjatkan puja dan puji syukur kehadirat-Nya. Yang telah melimpahkan rahmat dan hidayatnya kepada kami, sehingga kami dapat menyelesaikan Laporan Praktikum Pemograman Berorientasi Objek ini.

Adapun laporan ini kami telah usahakan semaksimal mungkin dan tentunya dengan bantuan berbagai pihak, sehingga dapat memperlancar pembuatan laporan ini. Untuk itu kami tak lupa pula menyampaikan banyak terimakasih kepada semua pihak yang telah membantu dalam pembuatan laporan ini.

Namun tidak lepas dari semua itu kami menyadari sepenuhnya bahwa ada kekurangan baik dari segi penyusun bahasa dan segi lainnya. Oleh karena itu dengan lapang dada dan tangan terbuka kami membuka selebar-lebarnya bagi pembaca yang ingin memberi saran dan kritik kepada kami sehingga kami dapat memperbaiki laporan ini.

Akhirnya penyusun mengharapkan semoga dari laporan ini tentang Sistem Pemograman Berbasis Web dapat diambil hikma dan manfaatnya sehingga dapat memberikan inspirasi terhadap pembaca.

Kendari

Penyusun

PRATIKUM 1

1.1 Pendahuluan

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

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

PHP (*Hypertext preprocessor*) yaitu bahasa pemrograman *web server-side* yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berada pada server (*server side HTML embedded scripting*).PHP adalah *script* yang digunakan untuk membuat halaman website yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Mekanisme ini menyebabkan informasi yang diterima client selalu baru atau *up to date*. Semua *script* PHP dieksekusi pada server dimana *script* tersebut dijalankan. (Joni Karman,2018)

1.2 Waktu dan Tempat

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

1.3 Alat dan Bahan

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

No.	Alat dan Bahan	Kegunaan
1.	Laptop	Sebagai tempat untuk menyimpan data, untuk mengerjakan projek dan sebagai tempat untuk mengoding.
2.	Xampp	Sebagai penghubung antara chrome dan sublime.
3.	Sublime	Sebagai tempat mengoding sebuah program.
4.	Crome	Sebagai tempat untuk melihat hasil running dari program yangtelah di buat.

Table 1. Alat & Bahan

1.4 Pengenalan *Object Oriented Programming* (OOP)

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

1.5 Prinsip OOP

Dalam *object oriented programming*, dikenal empat prinsip yang menjadi dasar penggunaannya. Merangkum *Freecodecamp*, keempat prinsip OOP tersebut adalah sebagai berikut:

1.5.1 Encapsulation

Encapsulation atau pengkapsulan adalah konsep tentang pengikatan data atau metode berbeda yang disatukan atau “dikapsulkan” menjadi satu unit data. Maksudnya, berbagai objek yang berada dalam *class* tersebut dapat berdiri sendiri tanpa terpengaruh oleh yang lainnya. *Encapsulation* dapat mempermudah pembacaan kode. Hal tersebut terjadi karena informasi yang disajikan tidak perlu dibaca secara rinci dan sudah merupakan satu kesatuan. Proses *enkapsulasi* mempermudah untuk menggunakan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci.

1.5.2 Abstraction

Prinsip selanjutnya yaitu *abstraction*. Prinsip ini sendiri berarti memungkinkan seorang *developer* memerintahkan suatu fungsi, tanpa harus mengetahui bagaimana fungsi tersebut bekerja. Lebih lanjut, *abstraction* berarti menyembunyikan detail latar belakang dan hanya mewakili informasi yang diperlukan untuk dunia luar. Ini adalah proses penyederhanaan konsep dunia nyata menjadi komponen yang mutlak diperlukan. Seperti kala menggunakan handphone, kamu cukup memberikan suatu perintah, tanpa tahu bagaimana proses terlaksananya perintah tersebut.

1.5.3 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.

1.5.4 Polymorphism

Prinsip terakhir dalam OOP adalah *polymorphism*. Pada dasarnya *polymorphism* adalah kemampuan suatu pesan atau data untuk diproses lebih dari satu bentuk. Salah satu ciri utama dari OOP adalah adanya *polymorphism*. Tanpa hal ini, suatu pemrograman tidak bisa dikatakan sebagai OOP. *Polymorphism* sendiri adalah konsep di mana suatu objek yang berbeda-beda dapat diakses melalui *interface* yang sama. Sebagai contoh, kamu memiliki fungsi untuk menghitung luas suatu benda, sementara benda tersebut berbentuk segitiga, lingkaran, dan persegi. Tentu, ketiga benda tersebut memiliki rumus perhitungan tersendiri. Dengan *polymorphism*, kamu dapat memasukkan fungsi perhitungan luas ke tiga benda tersebut, dengan tiap benda memiliki metode perhitungannya sendiri. Ini tentu akan mempermudah perintah yang sama untuk beberapa *class* atau *subclass* tertentu.

1.6 Kelebihan OOP Pada PHP

Sebenarnya kelebihan ini tidak hanya berlaku di PHP saja. di semua bahasa pemrograman juga berefek. Berikut ini adalah kelebihan menggunakan OOP dalam membuat aplikasi :

- a. *Syntax* lebih terstruktur.
- b. Terekomendasi.
- c. Sangat efektif jika di gunakan untuk membuat aplikasi yang berskala besar.
- d. Lebih mudah dan menghemat waktu, karena fungsi/*function* bisa dipanggil berulang-ulang kali.
- e. tergantung keperluan.
- f. Lebih menghemat waktu.
- g. Aplikasi lebih mudah dikembangkan.
- h. Dan lebih banyak lagi kelebihan dari OOP yang akan teman-teman rasakan sendiri pada saat mulai menggunakannya.

1.7 Kekurangan OOP pada PHP

- a. Tidak efisien, Menggunakan OOP akan lebih memakan daya pada *CPU* yang digunakan. Oleh karenanya, sangat disarankan untuk menggunakan perangkat terbaru saat melakukan pengembangan dengan OOP.
- b. Membutuhkan manajemen data yang ketat, Hal lain yang dikeluhkan oleh *developer* mengenai OOP adalah perlunya kontrol yang cukup ketat terhadap kode-kode tersebut. Hal ini karena OOP akan memunculkan beberapa kode-kode baru jika terdapat kode-kode yang kurang berfungsi dengan baik.
- c. Kemungkinan duplikasi, Dengan berbagai kemudahan yang diberikan oleh OOP, mengembangkan program baru dari yang telah ada sebelumnya akan jadi lebih mudah. Namun, hal ini justru membuat berbagai *project* yang dibuat akan terasa seperti sekadar duplikasi saja.

PRATIKUM 2

2.1 Pengertian Class, Object, Property, dan Method

Class, object, property dan *method* adalah pondasi dasar dari membangun aplikasi menggunakan struktur OOP. jika diibaratkan membangun sebuah rumah, maka *class, object, property* dan *method* adalah pilar-pilar dan bahan penyokong nya. selain penjelasannya, akan disertakan juga contoh dan carapenulisannya.

2.1.1 Class

Class di dalam OOP di gunakan untuk membuat sebuah kerangka kerja. bisa di katakan sebagai *library class* berisi *property* dan *method*. jadi ibaratnya *class* adalah sebuah wadah yang menyimpan *property* dan *method*. dan *object* yang di hasilkan biasanya berdasarkan isi dari *class*.

Di dalam PHP, penulisan *class* diawali dengan *keyword class*, kemudian diikuti dengan nama dari *class*. Aturan penulisan nama *class* sama seperti aturan penulisan variabel dalam PHP, yakni diawali dengan huruf atau *underscore* untuk karakter pertama, kemudian boleh diikuti dengan huruf, *underscore* atau angka untuk karakter kedua dan selanjutnya. Isi dari *class* berada dalam tanda kurung kurawal.

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

2.1.2 Property

Property (atau disebut juga dengan atribut) adalah data yang terdapat dalam sebuah *class*. Melanjutkan analogi tentang laptop, *property* dari laptop bisa berupa merk, warna, jenis *processor*, ukuran layar, dan lain-lain.

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

2.1.3 Method

Method adalah tindakan yang bisa dilakukan di dalam *class*. Jika menggunakan analogi *class* laptop kita, maka contoh *method* adalah menghidupkan laptop, mematikan laptop, mengganti cover laptop, dan berbagai tindakan lain. *Method* pada dasarnya adalah *function* yang berada di dalam *class*. Seluruh fungsi dan sifat *function* bisa diterapkan ke dalam *method*, seperti argumen/parameter, mengembalikan nilai (dengan keyword *return*), dan lain-lain.

```
<?php  
class laptop {  
    function hidupkan_laptop() {  
        //... isi dari method hidupkan_laptop  
    }  
  
    function matikan_laptop() {  
        //... isi dari method matikan_laptop  
    }  
  
    ... //isi dari class laptop  
}  
?>
```

2.1.4 Object

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

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

2.2 Enkapsulasi (*Encapsulation*)

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

2.2.1 *Public*

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

```
<?php

// buat class
laptopclass
laptop {

    // buat public
    propertypublic
    $pemilik;

    // buat public method
    public function hidupkan_laptop() {
        return "Hidupkan Laptop";
    }
}
```

2.2.2 Protected

Jika sebuah *property* atau *method* dinyatakan sebagai *protected*, berarti *property* atau *method* tersebut tidak bisa diakses dari luar *class*, namun bisa diakses oleh *class* itu sendiri atau turunan *class* tersebut.

```
<?php

// buat class
laptopclass
laptop {

    // buat protected
    propertyprotected
    $pemilik;

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

2.2.3 Private

Hak akses terakhir dalam konsep enkapsulasi adalah *private*. Jika sebuah *property* atau *method* di-set sebagai *private*, maka satu-satunya yang bisa mengakses adalah *class* itu sendiri. *Class* lain tidak bisa mengaksesnya, termasuk *class* turunan.

```
<?php  
  
// buat class  
komputerclass  
komputer {  
  
    // property dengan hak akses protected  
    private $jenis_processor = "Intel Core i7-4790  
3.6Ghz";  
  
    public function tampilan_processor() {  
        return $this->jenis_processor;  
    }  
}
```

2.3 Constructor dan Destructor

2.3.1 Constructor

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

```
public function __construct(){  
    echo "Ini berasal dari Constructor Laptop";  
}
```

2.3.2 Destructor

Destructor adalah *method* khusus yang dijalankan secara otomatis pada saat sebuah objek dihapus. Di dalam PHP, seluruh objek sebenarnya sudah otomatis dihapus ketika halaman PHP selesai diproses. Tetapi kita juga dapat menghapus objek secara manual. *Destructor* biasanya dipakai untuk membersihkan beberapa variabel, atau menjalankan proses tertentu sebelum objek dihapus.

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

2.4 Interfaces

Object Interface adalah sebuah ‘kontrak’ atau perjanjian implementasi *method*. Bagi *class* yang menggunakan *object interface*, *class* tersebut harus mengimplementasikan ulang seluruh *method* yang ada di dalam *interface*. Dalam pemrograman objek, penyebutan *object interface* sering disingkat dengan ‘*Interface*’ saja. Sama seperti *abstract class*, *interface* juga hanya berisi *signature* dari *method*, yakni hanya nama *method* dan parameternya saja (jika ada). Isi dari *method* akan dibuat ulang di dalam *class* yang menggunakan *interface*.

```
<?php
interface mouse
{
    //...isi dari interface mouse
}
?>
```

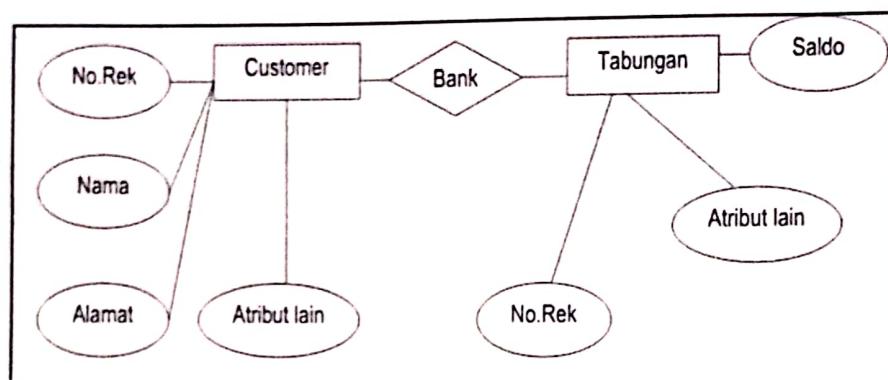
PRATIKUM 3

3.1 Model data berbasis objek

Merupakan himpunan data dan relasi yang menjelaskan hubungan logika berdasarkan objek antar data dalam suatu basis data.

3.1.1 Entity Relationship model (ERD)

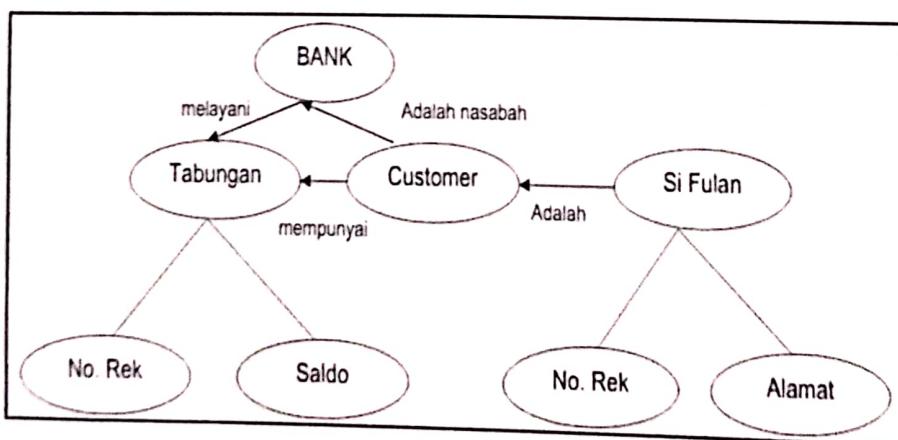
Merupakan model untuk menjelaskan hubungan antar data dalam basis data berdasarkan persepsi bahwa *real world* terdiri dari objek-objek dasar yang mempunyai hubungan / relasi antara objek tersebut.



Gambar 1, ERD

3.1.2 Semantic model

Merupakan model dimana relasi antar objek dinyatakan dengan kata-kata (*semantic*).



Gambar 2 Semantic model

3.2 Model data berbasis record

Model ini mendasarkan pada *record* untuk menjelaskan kepada *user* tentang hubungan logik antar data dalam basis data.

3.2.1 Relational Model

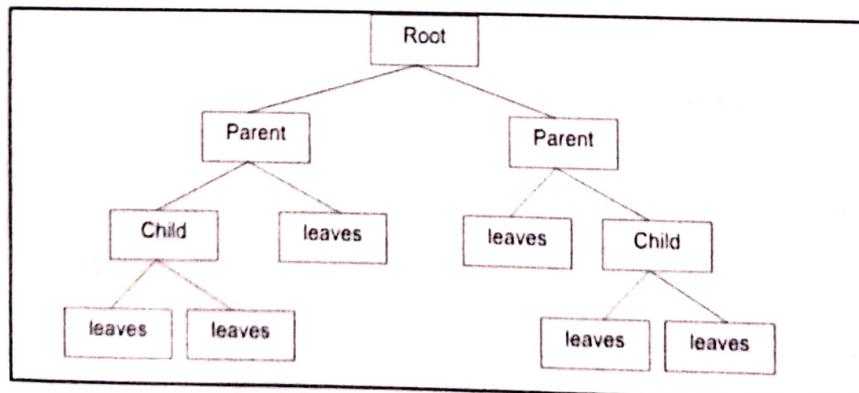
Menjelaskan tentang hubungan logik antar data dalam basis data dengan memvisualisasikan ke dalam bentuk tabel-tabel yang terdiri dari sejumlah baris dan kolom yang menunjukkan atribut tertentu.

MAHASISWA	
Nomhs	Nama
00351234	Fulan
01351346	Badu
02351370	Ayu

Gambar 3 Relation model

3.2.2 Hierarchical model

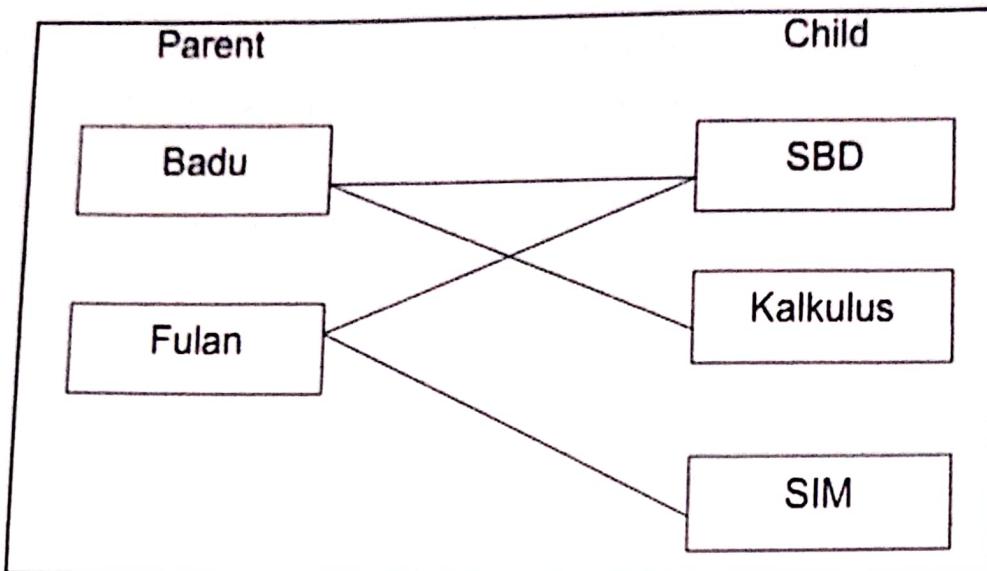
Menjelaskan tentang hubungan logik antar data dalam basis data dalam bentuk hubungan bertingkat (hirarki), Elemen penyusunnya disebut node, yang berupa rinci data, *agregat data*, atau *record*.



Gambar 4 Hierarchical model

3.2.3 Network Model

Hampir sama dengan model hirarki, dan digambarkan sedemikian rupa sehingga *child* pasti berada pada level yang lebih rendah daripada *parent*.



Gambar 5 Network model

3.3 Entity (Entitas)

Objek data yang utama dimana informasi dikumpulkan. Biasanya menunjukkan orang, tempat, benda, atau kejadian yang bersifat informasional.

3.4 Atribut

Atribut Adalah karakteristik /properti/ciri yang ada di dalam entitas, yang menghasilkan deskripsi detail mengenai entitas. Bagian dari sebuah atribut yang ada di dalam sebuah entitas atau *relationship* disebut nilai atribut.

3.4.1 Jenis Atribut

- a. *Key (Identifiers)* Atribut yang digunakan untuk menetapkan bagian yang unik dari sebuah entitas.
- b. *Descriptor* Atribut yang digunakan untuk menspesifikasi karakteristik yang non-unik dari bagian entitas.
- c. *Atribut Simple* Atribut sederhana yang tidak dapat dibagi dalam beberapa bagian

- d. *Atribut Komposit* Atribut yang dapat dibagi lagi dalam beberapa bagian contoh : Alamat; yang terdiri dari Negara, Propinsi dan Kota
- e. *Atribut Single-valued*, Atribut yang memiliki paling banyak satu nilai untuk setiap baris data
- f. *Multi-valued attributes*, Atribut yang dapat diisi dengan lebih satu nilai tetapi jenisnya sama. Contoh : Nomor Telp, Alamat, Gelar
- g. Atribut Turunan, Atribut yang diperoleh dari pengolahan dari atribut lain yang berhubungan. Contoh : Umur, IP

3.5 Relationship

Menggambarkan hubungan antara satu atau lebih entitas, yang digambarkan dalam bentuk *diamond*. Biasanya menunjukkan hubungan: *one-to-one*, *one-to-many*, dan *many-to-many*. Menunjukkan banyaknya himpunan entitas yang saling berelasi. Jenis derajat relasi:

- a. *Unary Degree* (Derajat Satu) melibatkan sebuah entitas yang berelasi dengan dirinya sendiri.
- b. *Binary Degree* (Derajat Dua) Himpunan relasi melibatkan dua himpunan entitas. Secara umum himpunan relasi dalam sistem basis data adalah *binary*.
- c. *Ternary Degree* (Derajat Tiga) Himpunan relasi memungkinkan untuk melibatkan lebih dari dua himpunan entitas.

3.6 Pengenalan 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*. Istilah ini pertama kali diperkenalkan oleh James Martin pada tahun 1983 dalam bukunya yang berjudul “*Managing the Database Environment*”.

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

3.6.1 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.

Nama Pemilik

Alamat

Foto Kos

Biaya

Simpan

Gambar 6 *Create*

3.6.2 *Read*

Fungsi yang kedua adalah *read*, berarti memungkinkan anda untuk mencari atau mengambil data tertentu yang berada di dalam tabel dengan membacanya. 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 *filterdata* berdasarkan kriteria tertentu.

No.	Nama Pemilik	Alamat	Foto Kos	Biaya	Opsi
1	Heini Adam	Lorong Salangga		Rp. 470.000,00/bulan	
2	Muhi Amal Anugra S	Lorong Pelangi		Rp. 520.000,00/bulan	
3	Muhamar Aminan	Jalan Lumbia - Lumbia		Rp. 580.000,00/bulan	
4	Ahdai Al Murad	Jalan Kelapa		Rp. 400.000,00/bulan	
5	Musdekan Syahru	Jalan Lumbia - Lumbia		Rp. 350.000,00/bulan	

Gambar 7 *Read*

3.6.3 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*.

The screenshot shows a web-based form for updating a record. The form fields include:

- Nama Pemilik:** Helmi Adam
- Alamat:** Lorong Salangga
- Foto Kos:** A file input field with the placeholder "Choose File" and "No file chosen".
- Biaya:** Rp. 470 000.00/bulan
- Simpan:** A green rectangular button labeled "simpan".

Gambar 8 Update

3.6.4 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.

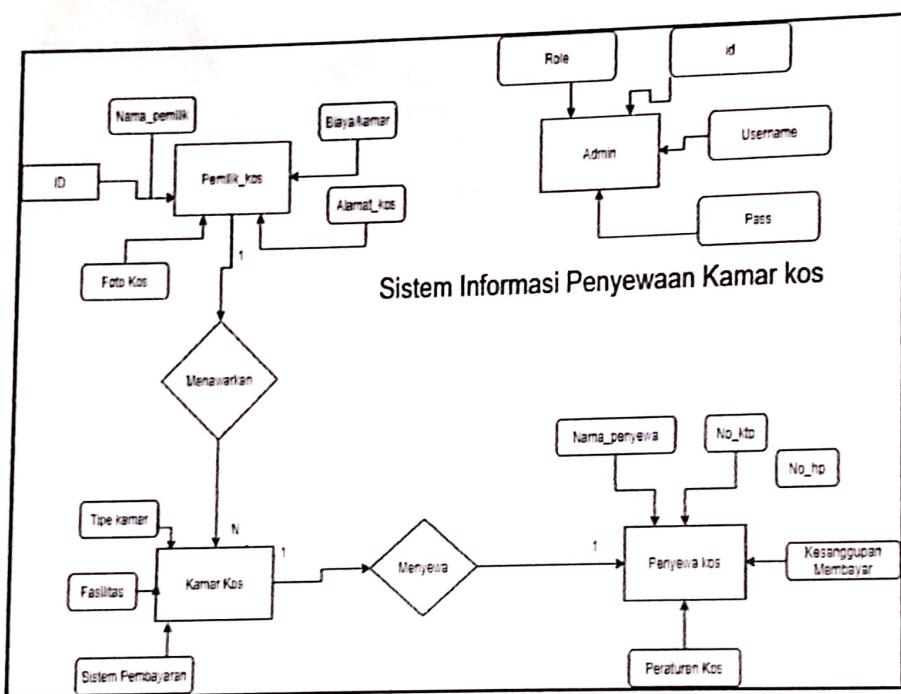
Sedangkan, untuk *hard delete* bertujuan untuk menghapus catatan pada basis data secara permanen.

PRATIKUM 4

4.1 Membuat Sistem Penyewaan Kamar Kos

4.1.1 ERD Sistem Penyewaan Kamar Kos

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



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.

+

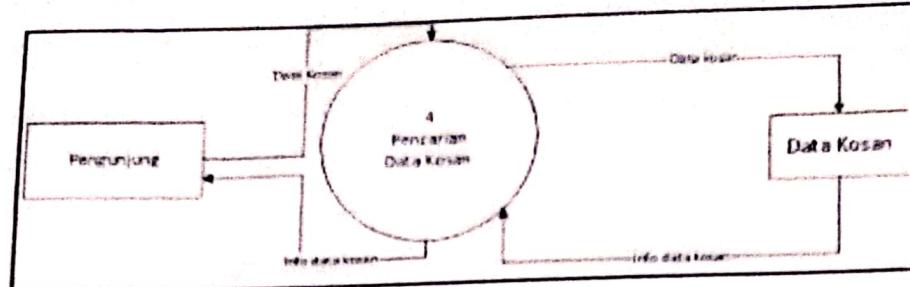
4.1.2 Data Flow Diagram (DFD)

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

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

a. Diagram Level 0 (Diagram Konteks)

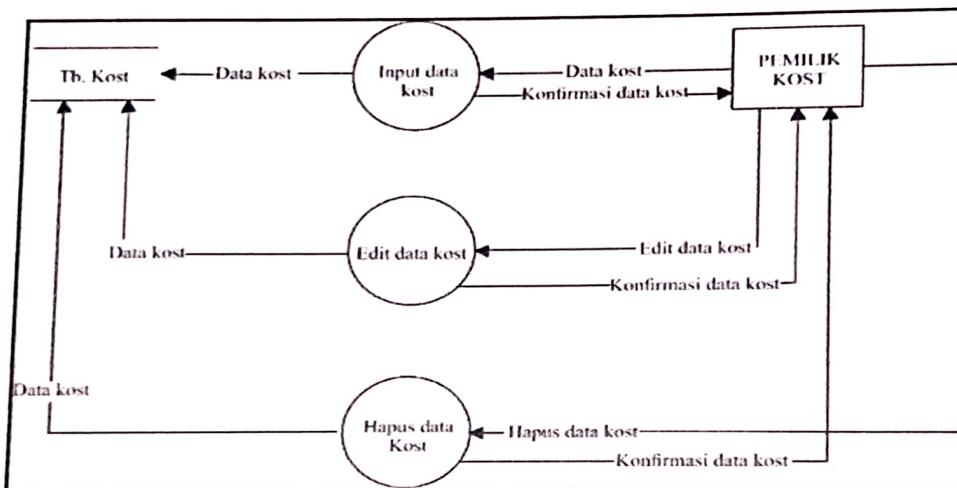
Diagram level 0 atau bisa juga diagram konteks adalah level diagram paling rendah yang menggambarkan bagaimana sistem berinteraksi dengan external entitas. Pada diagram konteks akan diberikan nomor untuk setiap proses yang berjalan, umumnya mulai dari angka 0 untuk start awal. Semua entitas yang ada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



Gambar 4.2 Diagram Konteks

b. 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

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

1. DFD level 0 hanya mengambarkan sistem secara basic saja.
2. DFD level 0 hanya menjelaskan aliran data dari input sampai output.
3. DFD level 1 mengambarkan aliran data yang lebih kompleks pada setiap prosesnya yang kemudian terbentuklah data store dan aliran data.

4. DFD Level 1 menggambarkan sistem secara sebagian atau seluruhnya secara mendetail.

4.1.3 User Interface

Interface atau dalam bahasa Indonesianya adalah antarmuka merupakan garda terdepan bagi suatu alat digital. Hal ini dikarenakan interface merupakan suatu layanan ataupun mekanisme yang diberikan kepada setiap pengguna alat digitalnya. Biasanya layanan ini berbentuk komunikasi antara pengguna (user) terhadap sistem operasi yang terdapat dalam alat digitalnya. Dalam hal ini antarmuka akan memberikan layanan berupa informasi kepada penggunanya sesuai yang dibutuhkan. Nantinya antarmuka ini akan memberikan layanan serta pemecahan masalah sampai masalah tersebut tuntas. Dengan menggunakan antarmuka ini memungkinkan sistem operasi untuk bersentuhan langsung dengan para penggunanya. Tanpa memerlukan hal yang rumit.

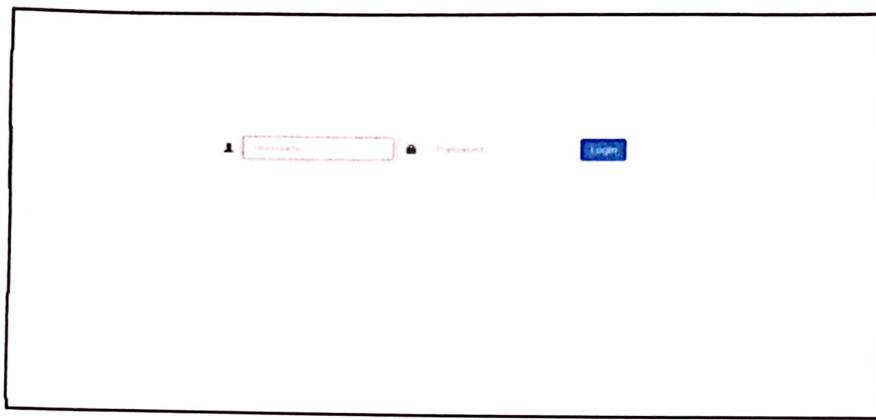
Interface sendiri memiliki fungsi untuk memasukkan pengetahuan baru ke dalam ES yang merupakan basis pengetahuan sistem pakar. Jadi saat sebuah software ataupun hardware baru ditambahkan ke dalam alat digital, interface ini yang pertama kali memberikan informasi. Ini terkait juga dengan sistem interface yakni input dan output yang keduanya sama-sama tentang memberikan efek manipulasi.

Interface ini biasanya didesain seindah mungkin, tapi tetap bersifat compact. Contoh sederhananya yakni pada alat digital

smartphone. Dalam interface ini juga biasanya ditampilkan penjelasan tentang sistem dan bagaimana cara menggunakan sistem tersebut. Karenanya syarat utama desain interface adalah kemudahan.

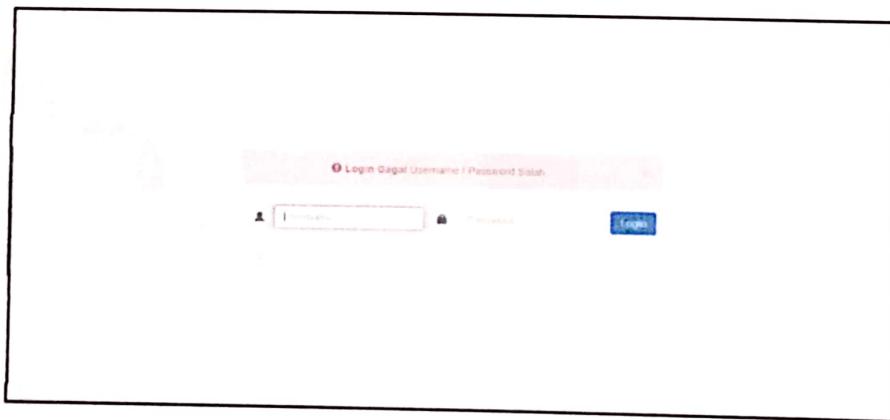
4.1.4 Login

Ketika kita mengakses halaman ini, kemudian akan diarahkan ke tampilan *login*.



Gambar 4.4 Login

Saat admin salah memasukan *username* dan *password* akan muncul peringatan pada header tampilan *login*.

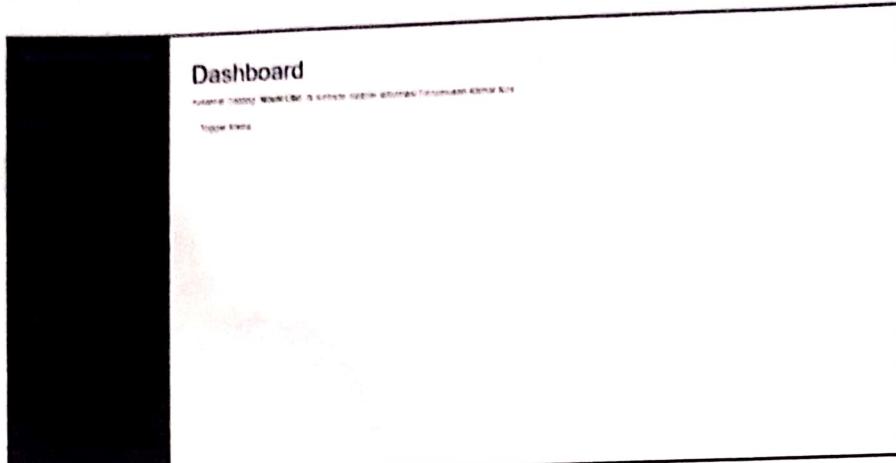


Gambar 4.4 Gagal Login

Tetapi, jika admin sesuai memasukan *username* dan *password* yang sesuai dengan database *table* admin, maka tampilan akan menuju ke halaman *dashboard*.

4.1.5 Dashboard

Jika admin telah sukses melakukan login tampilan akan ke halaman *dashboard*.

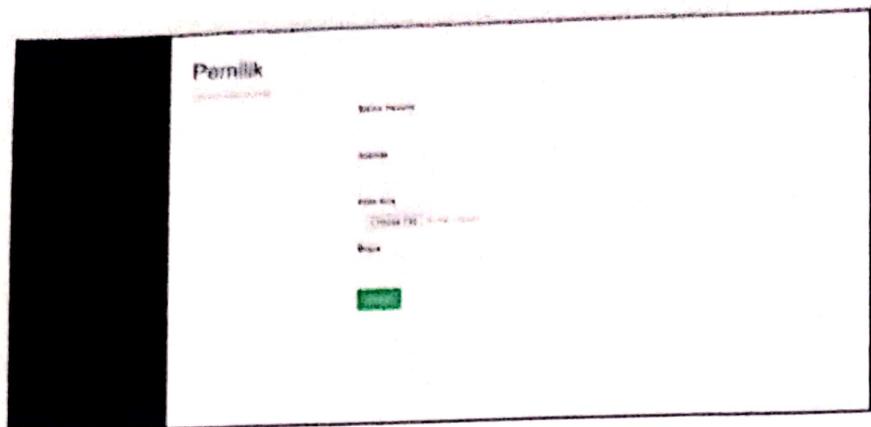


Gambar 4.5 Dashboard

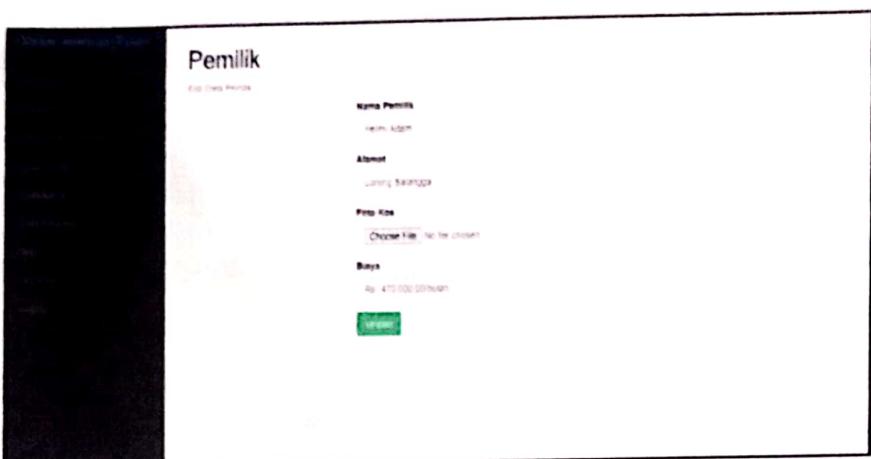
Pada tampilan *dashboard*, kita akan melihat pilihan-pilihan menu seperti, data pemilik, data kamar, dan data penyewa. Ketiga pilihan menu tersebut diisi sesuai dengan isi dari model data *erd*nya yang dimana ketiga *table* tersebut dibuat sesuai dengan *system crud*. Adapun itu tampilan *read*, *create*, *update* dan *delete*.

No.	Nama Pemilik	Alamat	Foto Kos	Buya	Opsi
1	Herm Adam	Lorong Selangga		Rp. 470.000/bulanan	Edit Delete
2	Muti Amali Angga S	Lorong Perlang		Rp. 225.000,00/bulanan	Edit Delete
3	Muamer Amali	Jalan Lumbu - Lumbu		Rp. 580.000,00/bulanan	Edit Delete
4	Azizah Al Muzni	Jalan Keciput		Rp. 410.000,00/bulanan	Edit Delete
5	Muhammad Arifin	Jalan Lumbu - Lumbu		Rp. 340.000,00/bulanan	Edit Delete

Gambar 4.6 Read Data Penyewa



Gambar 4.7 Tambah Data



Gambar 4.8 Update Data

Dan ketika menekan tombol *delete* data yang ada pada pada tampilan read akan terhapus.

4.1 Menampilkan Data Yang Berelasi

Pada sistem Informasi crud penyewaan kamar kos ini, kita membuat membuat data pada tabel-tabel tersebut berelasi. Disini kita menggunakan *query* yang ada pada database. *Query* yang kita gunakan disini yaitu *Inner Join*, *query* ini memiliki fungsi yang dimana akan mengeksekusi data atau memuncul data yang telah berelasi dalam kata lain *column* atau *field* yang merelasikan mereka sama memiliki data. Berbeda dengan *right join* dan *left join*, *query* ini akan mengeksekusi semua data Adapun itu kedua atau tiga tabel tersebut memiliki data

ataupun tidak memiliki data perintah ini tetap memiliki *result*.

No.	Nama Pemilik	Kamar	Foto Kamar	Type Kamar	Fasilitas	Bonus Pembayaran	Biaya Kamar	Opsi
1	Gede Suharmi	Jadeng Rumah		S x S	Kamar Minimalis	Rp. 110.000,-	Rp. 100.000,-/bulan	<button>Detail</button>
2	Eko Arie Aryana S.	Jadeng Rumah		S x S	Kamar Standar	Rp. 110.000,-	Rp. 100.000,-/bulan	<button>Detail</button>
3	Wulan Jenifer	Wulan Cottages		S x S	Penginapan Kamar	Rp. 110.000,-/bulan	Rp. 100.000,-/bulan	<button>Detail</button>
4	Aditya Alifqur	2020 KOSKAM		S x S	Apt. Double & Apt. Pernikahan	Rp. 1185.000,-	Rp. 450.000,-/bulan	<button>Detail</button>
5	Mardian Suryana	Jadeng Rumah		S x S	Cieng Rumah Dier	Rp. 1174.000,-	Rp. 350.000,-/bulan	<button>Detail</button>

Gambar 4.9 Inner Join Table Pemilik & Table Kamar

Pada pilihan form kamar kos yang tersedia, terlihat tampilan yang merelasikan table pemilik dan kamar, terbukti dari *field/column* yang ada.

Tampilan ini juga memiliki tombol untuk menyewa kamar yang tersedia, Ketika kita menekan tombol sewa halaman akan langsung berganti pada halaman tambah data penyewa.

The form is titled "Penyewa". It contains the following fields:

- Nomor KTP
- Nama Penyewa
- No. Telepon
- Kelangungan Merkantil
 - Male
 - Female
- Perizinan Kos
- ID Kamar
 - Pilih Kamar

Gambar 4.10 Tambah Penyewa

DAFTAR PUSTAKA

- Wibowo, Kadek .2015.*Pengertian Pemograman Berorientasi Objek.*
Yogyakarta
- Ibrahim, Ali. 2009. *Cara Praktis Membuat Website Dinamis Menggunakan Xampp.* Neotekno. Jakarta
- Kusrini. 2007. *Strategi Perancangan dan Pengelolaan Basis Data.* Andi Offset. Yogyakarta
- Nugroho, Bunafit. 2005. *Database Relational dengan My-SQL.* Andi Offset.
Yogyakarta
- Suyanto, M. 2003. *Startegi Periklaan pada Sistem Crud Perusahaan Top Dunia.* Andi Offset. Yogyakarta
- Usdiyanto, Rieneke. 2001. *Framework Crud.* Andi Offset. Yogyakarta