

**LAPORAN**  
**PEMOGRAMAN BERORIENTASI OBJEK**



**OLEH:**

**NAMA : FITRIANA ASDHI**

**NIM : F1G120019**

**KELOMPOK : I (SATU)**

**ASISTEN PENGAMPU:**

**WAHID SAFRI JAYANTO**

**PROGRAM STUDI S1 ILMU KOMPUTER**  
**JURUSAN MATEMATIKA**  
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**  
**UNIVERSITAS HALU OLEO**  
**2021**

**HALAMAN PENGESAHAN**  
**LAPORAN PRATIKUM**



OLEH:

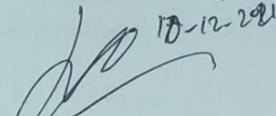
FITRIANA ASDHI (F1G120019)

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, 13 Desember 2021

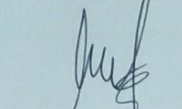
Menyetujui,

Asisten Pratikum,

  
Wahid Safri Jayanto

F1G117059

Pratikan,

  
Fitriana Asdhi

F1G120019

## 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 Programing</i> (OOP)..... | 3    |
| 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> .....                              | 4    |
| 1.5.4 <i>Polymorphism</i> .....                             | 4    |
| 1.6 Kelebihan OOP Pada PHP.....                             | 5    |
| 1.7 Kekurangan OOP pada PHP.....                            | 5    |
| PRATIKUM 2.....   | 7    |
| 2.1 Pengertian Class, Object, Property, dan Method.....     | 7    |
| 2.1.1 <i>Class</i> .....                                    | 7    |
| 2.1.2 <i>Property</i> .....                                 | 8    |
| 2.1.3 <i>Method</i> .....                                   | 8    |

|  |    |
|--|----|
| 2.1.4 <i>Object</i> .....                          | 9  |
| 2.2 Enkapsulasi ( <i>Encapsulation</i> ) .....     | 9  |
| 2.2.1 <i>Public</i> .....                          | 10 |
| 2.2.2 <i>Protected</i> .....                       | 10 |
| 2.2.3 <i>Private</i> .....                         | 11 |
| 2.3 <i>Constructor</i> dan <i>Destructor</i> ..... | 11 |
| 2.3.1 <i>Constructor</i> .....                     | 11 |
| 2.3.2 <i>Destructor</i> .....                      | 12 |
| 2.4 <i>Interfaces</i> .....                        | 12 |
| PRATIKUM 3.....                                    | 13 |
| 3.1 Model data berbasis objek .....                | 13 |
| 3.1.1 <i>Entity Relationship model (ERD)</i> ..... | 13 |
| 3.1.2 <i>Semantic model</i> .....                  | 13 |
| 3.2 Model data berbasis <i>record</i> .....        | 14 |
| 3.2.1 <i>Relational Model</i> .....                | 14 |
| 3.2.2 <i>Hierarchical model</i> .....              | 14 |
| 3.2.3 <i>Network Model</i> .....                   | 15 |
| 3.3 <i>Entity</i> (Entitas).....                   | 15 |
| 3.4 Atribut.....                                   | 15 |
| 3.4.1 Jenis Atribut .....                          | 15 |
| 3.5 <i>Relationship</i> .....                      | 16 |
| 3.6 Pengenalan <i>Crud</i> .....                   | 17 |
| 3.6.1 <i>Create</i> .....                          | 17 |
| 3.6.2 <i>Read</i> .....                            | 18 |
| 3.6.3 <i>Update</i> .....                          | 19 |

|  |    |
|--|----|
| 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</i> (DFD).....    | 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.....         | 219 |
| 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 1.1 Alat & Bahan ..... | 2 |
|------------------------------|---|

## **KATA PENGANTAR**

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Penyayang. Kami panjatkan puja dan puji syukur kehadiran-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 hikmah 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 dikenal 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 penanganannya 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 dilaksanakan 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.  | <i>Xampp</i>   | Sebagai penghubung antara <i>chrome</i> dan <i>sublime</i> .                                      |
| 3.  | <i>Sublime</i> | Sebagai tempat mengoding sebuah program.  |
| 4.  | <i>Chrome</i>  | Sebagai tempat untuk melihat hasil <i>running</i> dari program yang telah di buat.                |

**Table 1.1 Alat & Bahan**

## 1.4 Pengenalan *Object Oriented Programing* (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 di panggil berulang-ulang kali.
- e. tergantung keperluan.
- f. Lebih menghemat waktu.
- g. Aplikasi lebih mudah di kembangkan.
- 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 cara penulisannya.

#### 2.1.1 Class

*Class* di dalam OOP digunakan 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.

### **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.

### **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.

### **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.



### 2.1.5 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.

## 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.

### **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.

### **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.

## **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.

### **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.

## **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*.

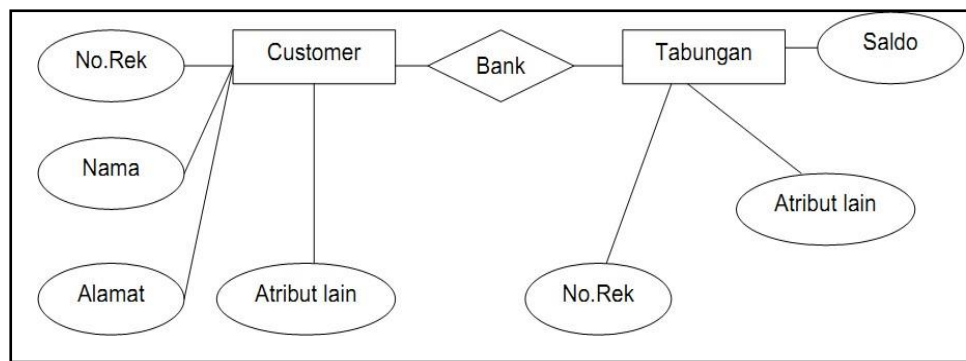
## PRATIUM 3

### 3.1 Model data berbasis objek

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

#### 3.1.1 *Entity Relationship model (ERD)*

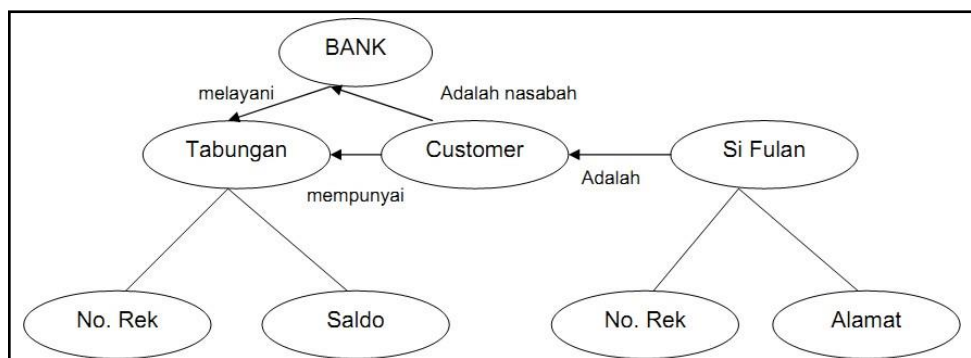
Merupakan model untuk menjelaskan hubungan antar data dalam basis data berdasarkan persepsi bahwa *real world* terdiri dari objek-objek dasar yang mempunyai hubungan / relasi antara objek tersebut.



Gambar 3.1 ERD

#### 3.1.2 *Semantic model*

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



Gambar 3.2 *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*

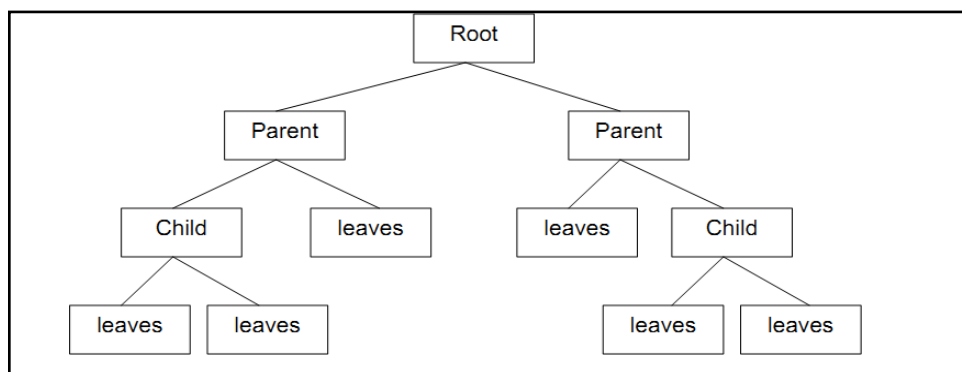
Menjelaskan tentang hubungan logik antar data dalam basis data dengan memvisualisasikan ke dalam bentuk tabel-tabel yang terdiri dari sejumlah baris dan kolom yang menunjukkan atribut tertentu.

| MAHASISWA |       |
|-----------|-------|
| Nomhs     | Nama  |
| 00351234  | Fulan |
| 01351346  | Badu  |
| 02351370  | Ayu   |

Gambar 3.3 *Relation model*

#### 3.2.2 *Hierarchical model*

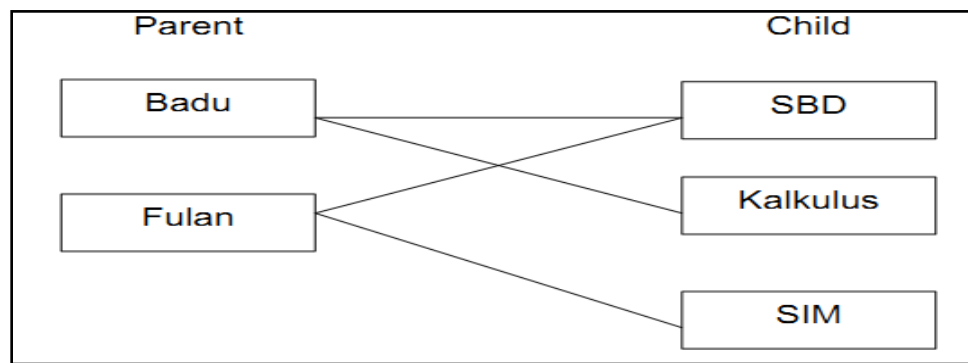
Menjelaskan tentang hubungan logik antar data dalam basis data dalam bentuk hubungan bertingkat (hirarki), Elemen penyusunnya disebut node, yang berupa rinci data, *agregat data*, atau *record*.



Gambar 3.4 *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 3.5 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

- Key (Identifiers)* Atribut yang digunakan untuk menetapkan bagian yang unik dari sebuah entitas.
- Descriptor* Atribut yang digunakan untuk menspesifikasikan karakteristik yang non-unik dari bagian entitas.
- 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

Menggambarakan hubungan antara satu atau lebih entitas, yang digambarkan dalam bentuk *diamond*. Biasanya menunjukkan hubungan: *one-to-one*, *one-to-many*, dan *many-to-many*. Menunjukan 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 duahimpunan 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 dan *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.



The image shows a web form with the following elements:

- Nama Pemilik**: A text input field.
- Alamat**: A text input field.
- Foto Kos**: A file upload field with a "Choose File" button and the text "No file chosen".
- Biaya**: A text input field.
- simpan**: A green button to save the data.

**Gambar 3.6 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 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.

Sistem Informasi Penyewaan Kamar Kos

Dashboard  
Daftar Kamar Kos  
Daftar Penyewa Kamar  
Pemilik Kos  
Kamar Kos  
Penyewa Kos  
Admin  
Log Out

### Pemilik Data Pemilik

Tambah Pemilik

| ID | Nama Pemilik  | Alamat         | Foto Kos  | Biaya Kamar          | Opsi                                       |
|----|---------------|----------------|---|----------------------|--|
| 1  | Rosmiati      | Baruga         |  | Rp 350.000,00        | <a href="#">Edit</a> <a href="#">Hapus</a> |
| 2  | Andrianto     | Jalan Bridgen  |  | Rp 450.000,00        | <a href="#">Edit</a> <a href="#">Hapus</a> |
| 3  | Setiabudi     | Lorong Pelangi |  | Rp. 450.000,00/bulan | <a href="#">Edit</a> <a href="#">Hapus</a> |
| 4  | Siska Melinda | Jalan Damai    |  | Rp. 500.000,00/bulan | <a href="#">Edit</a> <a href="#">Hapus</a> |

**Gambar 3.7 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.

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*.

**Nama Pemilik**

Helmi Adam

**Alamat**

Lorong Salangga

**Foto Kos**

[Choose File](#) No file chosen

**Biaya**

Rp. 470.000,00/bulan

[simpan](#)

**Gambar 3.8 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.

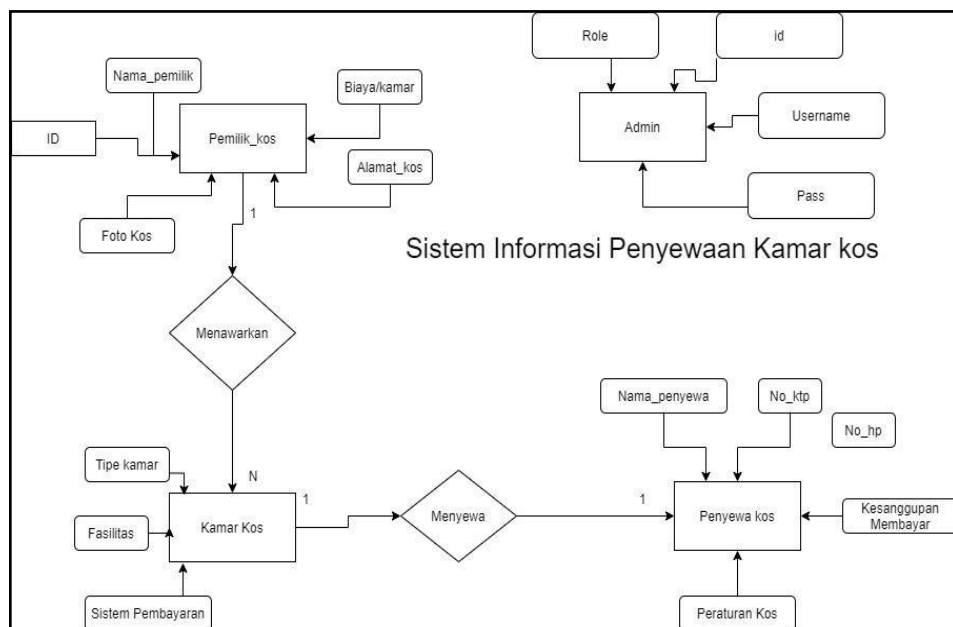
Sedangkan, untuk *hard delete* bertujuan untuk menghapus catatan pada basis data secara permanen.

## PRATIUM 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.

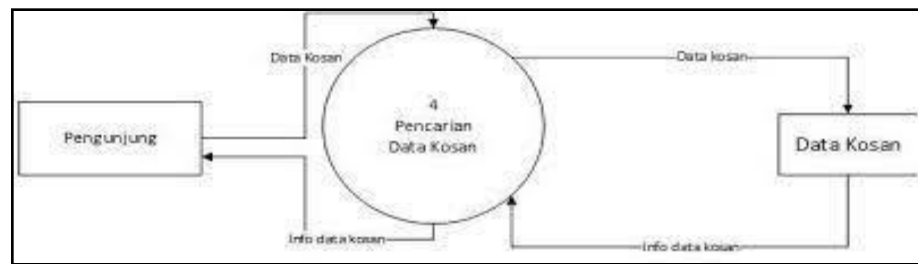
#### **4.1.2 Data Flow Diagram (DFD)**

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)**

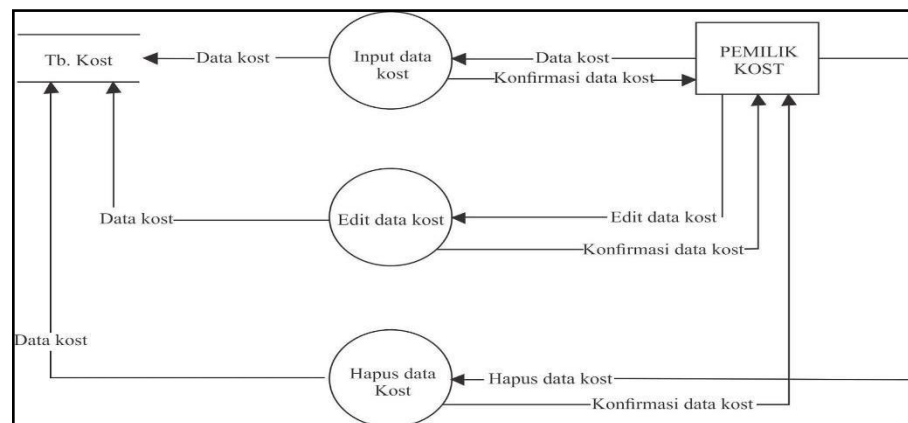
Diagram level 0 atau bisa juga diagram konteks adalah level diagram paling rendah yang menggambarkan bagaimana sistem berinteraksi dengan external entitas. Pada diagram konteks akan diberikan nomor untuk setiap proses yang berjalan, umumnya mulai dari angka 0 untuk start awal. Semua entitas yang ada 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 menggambarkan sistem secara basic saja.
2. DFD level 0 hanya menjelaskan aliran data dari input sampai output.
3. DFD level 1 menggambarkan 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 Interfaces**

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.

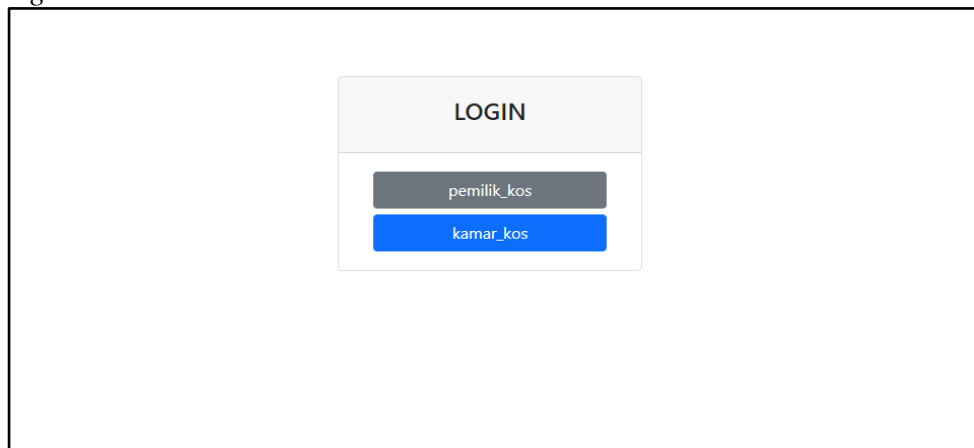
Interface sendiri memiliki fungsi untuk memasukkan pengetahuan baru ke dalam ES yang merupakan basis pengetahuan sistem pakar. Jadi saat sebuah software ataupun hardware baru ditambahkan ke dalam alat digital, interface ini yang pertama kali memberikan informasi. Ini terkait juga dengan sistem interface 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*.

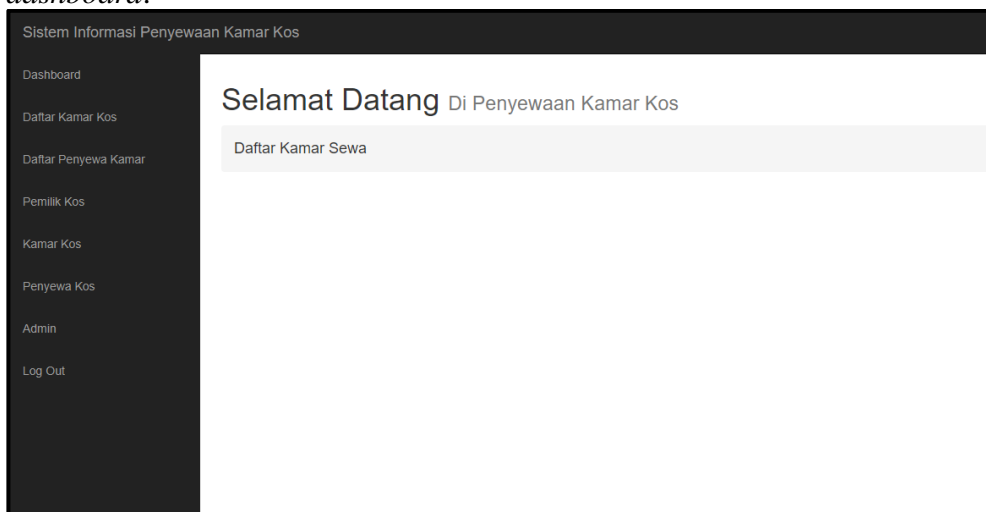
The image shows a login interface within a light gray rectangular frame. At the top center is a white box with the word "LOGIN" in bold black text. Below this box are two stacked buttons. The top button is gray with the text "pemilik\_kos" in white. The bottom button is blue with the text "kamar\_kos" in white.

Gambar 4.4 Login

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

#### 4.1.5 Dashboard

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

The image shows a dashboard interface. On the left is a dark gray sidebar with a list of menu items in white text: "Sistem Informasi Penyewaan Kamar Kos", "Dashboard", "Daftar Kamar Kos", "Daftar Penyewa Kamar", "Pemilik Kos", "Kamar Kos", "Penyewa Kos", "Admin", and "Log Out". The main content area has a white background. At the top, it says "Selamat Datang Di Penyewaan Kamar Kos". Below this is a light gray button labeled "Daftar Kamar Sewa".

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 *erdnya* yang dimana ketiga *tabl* tersebut dibuat sesuai dengan *system crud* Adapun itu tampilan *read, create, update* dan *delete*.

Sistem Informasi Penyewaan Kamar Kos

Dashboard  
Daftar Kamar Kos  
Daftar Penyewa Kamar  
Pemilik Kos  
Kamar Kos  
Penyewa Kos  
Admin  
Log Out

### Pemilik Data Pemilik

Tambah Pemilik

| ID | Nama Pemilik  | Alamat         | Foto Kos | Biaya Kamar          | Opsi                                       |
|----|---------------|----------------|----------|----------------------|--|
| 1  | Rosmiati      | Baruga         |          | Rp 350.000,00        | <a href="#">Edit</a> <a href="#">Hapus</a> |
| 2  | Andrianto     | Jalan Bridgen  |          | Rp 450.000,00        | <a href="#">Edit</a> <a href="#">Hapus</a> |
| 3  | Setiabudi     | Lorong Petangi |          | Rp. 450.000,00/bulan | <a href="#">Edit</a> <a href="#">Hapus</a> |
| 4  | Siska Melinda | Jalan Damai    |          | Rp. 500.000,00/bulan | <a href="#">Edit</a> <a href="#">Hapus</a> |

Gambar 4.6 Read Data Pemilik

Sistem Informasi Kosan

Dashboard  
Kamar Kos  
Lihat Penyewa Kamar  
Data Pemilik  
Data Kamar  
Data Penyewa  
About  
Services  
Logout

### Pemilik

Tambah Data Pemilik

Nama Pemilik

Alamat

Foto Kos  
 No file chosen

Biaya

Gambar 4.7 Tambah Data

Sistem Informasi Kosan

Dashboard  
Kamar Kos  
Lihat Penyewa Kamar  
Data Pemilik  
Data Kamar  
Data Penyewa  
About  
Services  
Logout

### Pemilik

Edit Data Pemilik

Nama Pemilik

Alamat

Foto Kos  
 No file chosen

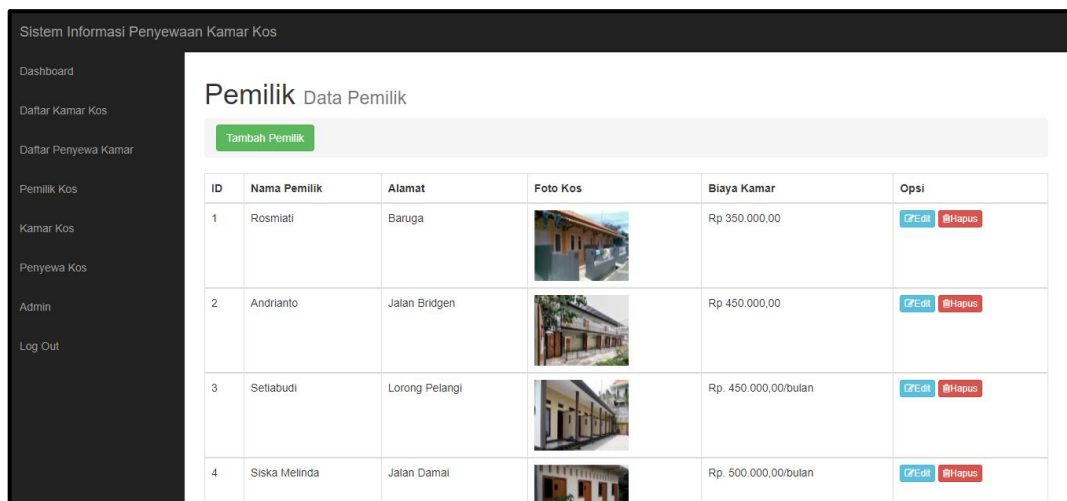
Biaya





Gambar 4.8 Update Data

## 4.2 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 memunculkan 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*.



| ID | Nama Pemilik  | Alamat         | Foto Kos  | Biaya Kamar          | Opsi                                       |
|----|---------------|----------------|---|----------------------|--|
| 1  | Rosmiati      | Baruga         |  | Rp 350.000,00        | <a href="#">Edit</a> <a href="#">Hapus</a> |
| 2  | Andrianto     | Jalan Bridgen  |  | Rp 450.000,00        | <a href="#">Edit</a> <a href="#">Hapus</a> |
| 3  | Setiabudi     | Lorong Pelangi |  | Rp. 450.000,00/bulan | <a href="#">Edit</a> <a href="#">Hapus</a> |
| 4  | Siska Melinda | Jalan Damai    |  | Rp. 500.000,00/bulan | <a href="#">Edit</a> <a href="#">Hapus</a> |

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.

**Sistem Informasi Kosan**

**Penyewa**  
Tambah Data Penyewa

No KTP:

Nama Penyewa:

No Telp:

Kesanggupan Membayar:

Peraturan Kos:

ID Kamar:

**Gambar 4.10 Tambah Penyewa**

Setelah selesai mengisi data penyewa, kita akan langsung akan diarahkan ke tampilan penyewa kamar kos, pada tampilan tersebut kita akan melihat tampilan yang merelasikan 3 tabel. Pada tampilan tersebut kita juga menggunakan juga *query inner join* yang menampilkan semua data yang telah diisi pada ketiga *table*.

**Sistem Informasi Penyewaan Kamar Kos**

**Penyewa Data Penyewa**

| No Ktp           | Nama Penyewa   | No Telp      | Peraturan Kos                                | Kesanggupan Membayar | Opsi   |
|------------------|----------------|--------------|--|----------------------|--|
| 7273746598070004 | Fitriana Asdhi | 082134562789 | Tidak Boleh membawa tamu diatas jam 10 malam | Cash                 | <input type="button" value="Edit"/> <input type="button" value="Hapus"/> |
| 7407432789342690 | Riska Amalia   | 082213457689 | Tidak Boleh membawa tamu diatas jam 10 malam | Cicil                | <input type="button" value="Edit"/> <input type="button" value="Hapus"/> |

**Gambar 4.11 Tampilan Penyewa Kamar**

## 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, Riyeke. 2001. *Framework Crud.* Andi Offset. Yogyakarta