Nama: Yazid Dhiaulhaq Ismail

Kelas: 3DB03

NPM : 31120173

UJIAN GOLANG INTERMEDIATE

1. Apa yang dimaksud dengan crud? (10 poin)

CRUD adalah singkatan dari Create, Read, Update, dan Delete. Ini adalah empat operasi dasar yang digunakan dalam pengelolaan data dalam sistem database atau aplikasi berbasis data. Setiap operasi memiliki makna berikut:

- Create (Membuat): Operasi ini digunakan untuk membuat entitas data baru dalam database. Misalnya, dalam aplikasi toko online, operasi Create dapat digunakan untuk membuat entri baru untuk produk yang akan dijual.
- Read (Membaca): Operasi ini digunakan untuk membaca atau mengambil data yang ada dari database. Misalnya, dalam aplikasi toko online, operasi Read dapat digunakan untuk menampilkan daftar produk yang tersedia kepada pengguna.
- Update (Memperbarui): Operasi ini digunakan untuk memperbarui data yang sudah ada dalam database. Misalnya, dalam aplikasi toko online, operasi Update dapat digunakan untuk mengubah harga atau deskripsi produk yang sudah ada.
- Delete (Menghapus): Operasi ini digunakan untuk menghapus data yang tidak diperlukan lagi dari database. Misalnya, dalam aplikasi toko online, operasi Delete dapat digunakan untuk menghapus produk yang tidak lagi tersedia atau tidak diperlukan.
- 2. Sebutkan struktur crud pada go Language! (10 poin)

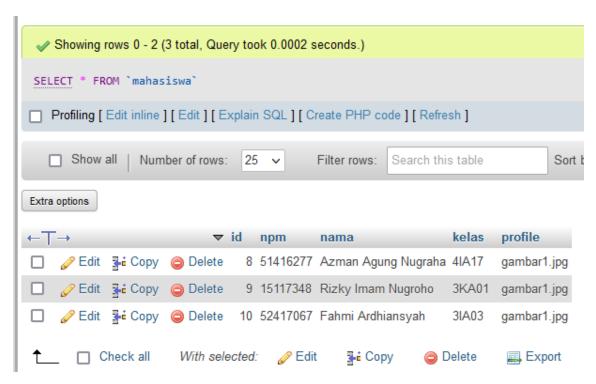
Dalam bahasa Go, struktur CRUD (Create, Read, Update, Delete) biasanya diimplementasikan menggunakan operasi-operasi dasar dalam paket database/sql. Berikut adalah contoh struktur CRUD dalam Go Language:

- a. Create (Membuat Data):
 - Membuka koneksi ke database menggunakan sql.Open.
 - Membuat kueri INSERT baru dengan db.Prepare dan db.Exec.
 - Menangani kesalahan yang terjadi saat menyimpan data baru ke database.
- b. Read (Membaca Data):
 - Membuka koneksi ke database menggunakan sql.Open.

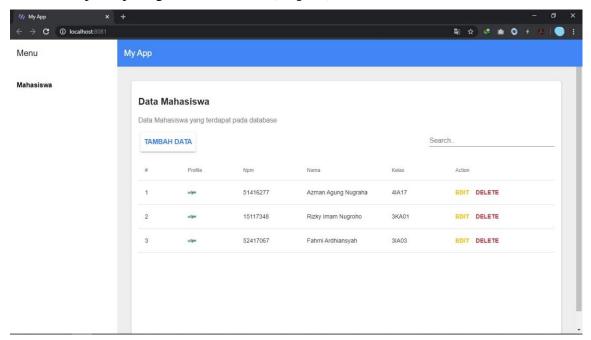
- Membuat kueri SELECT dengan db.Query atau db.QueryRow.
- Menggunakan Scan untuk membaca hasil query dan mengisi data ke struct atau variabel.
- Menangani kesalahan yang terjadi saat membaca data dari database.
- c. Update (Mengubah Data):
 - Membuka koneksi ke database menggunakan sql.Open.
 - Membuat kueri UPDATE dengan db.Prepare dan db.Exec.
 - Menangani kesalahan yang terjadi saat mengubah data di database.
- d. Delete (Menghapus Data):
 - Membuka koneksi ke database menggunakan sql.Open.
 - Membuat kueri DELETE dengan db.Prepare dan db.Exec.
 - Menangani kesalahan yang terjadi saat menghapus data dari database
- 3. Jelaskan fungsi dan tujuan go build serta go run! (10 poin)
 - Go build: Perintah ini digunakan untuk mengkompilasi paket dan dependensinya.
 - Go run: Perintah ini digunakan untuk mengkompilasi dan menjalankan program Go.
- 4. Membuat dan menginput data pada table database! (20 poin)

id	npm	Nama	Kelas	Profile
1	51416277	Azman Agung Nugraha	4IA17	gambar1.jpg
2	15117348	Rizky Imam Nugroho	3KA01	gambar1.jpg
3	52417067	Fahmi Ardhiansyah	3IA03	gambar1.jpg





5. Buatlah output seperti gambar dibawah! (50 poin)



main.go

my-app.html

```
<app-route route="{{route}}" pattern="[[rootPath]]:page" data="{{routeData}}" tail="{{subroute}}">
<app-drawer-layout fullbleed narrow="{{narrow}}">
 <app-drawer id="drawer" slot="drawer" swipe-open="[[narrow]]">
   <app-toolbar>Menu</app-toolbar>
   <iron-selector selected="[[page]]" attr-for-selected="name" class="drawer-list" role="navigation">
     <a name="mahasiswa" href="[[rootPath]]mahasiswa">Mahasiswa</a>
<!-- <a name="view2" href="[[rootPath]]view2">View Two</a>
 <app-header-layout has-scrolling-region>
   <app-header slot="header" condenses reveals effects="waterfall">
      <app-toolbar>
        <paper-icon-button icon="my-icons:menu" drawer-toggle></paper-icon-button>
        <div main-title>My App</div>
      </app-toolbar>
   </app-header>
   <iron-pages selected="[[page]]" attr-for-selected="name" fallback-selection="view404" role="main">
     <mahasiswa-page name="mahasiswa"></mahasiswa-page>
     <my-view404 name="view404"></my-view404>
```

```
Polymer.setPassiveTouchGestures(true);
107
          class MyApp extends Polymer.Element {
109
            static get is() { return 'my-app'; }
            static get properties() {
              return {
                page: {
                  type: String,
114
                  reflectToAttribute: true,
115
                 observer: '_pageChanged',
116
                },
                routeData: Object,
118
                subroute: Object,
                rootPath: String,
120
              };
            static get observers() {
123
              return [
124
                 '_routePageChanged(routeData.page)',
125
              ];
126
            _routePageChanged(page) {
127
128
              this.page = page || 'mahasiswa';
129
              if (!this.$.drawer.persistent) {
               this.$.drawer.close();
132
            _pageChanged(page) {
134
              const resolvedPageUrl = this.resolveUrl(page + '.html');
135
              Polymer.importHref(
                resolvedPageUrl,
137
                null,
                this. showPage404.bind(this).
```

mahasiswa.html

```
<div class="card"
          <h1>Data Mahasiswa</h1>
           Data Mahasiswa yang terdapat pada database
          <template is="dom-if" if="[[success]]">
            <strong>Message:</strong> [[success]]
          <iron-ajax auto url="{{url}}" method="{{method}}" handle-as="json" content-type="application-json"</pre>
          last-response="{{response}}"></iron-ajax>
51
          <paper-button raised class="link" style="margin-bottom: 20px;" on-click="openAddModal">Tambah Data
          <paper-input-container no-label-float>
            <label slot="label">Search..</label>
            <iron-input slot="input";</pre>
              <input type="text" value="{{keyword::input}}" width="30%">
           <vaadin-grid items="{{response.data}}">
            <vaadin-grid-column width="50px"
              <template class="header">#</template>
                <div>[[index]]</div>
             </vaadin-grid-column>
             <vaadin-grid-column width="80px">
              <template class="header">Profile</template>
               <img src="/img/[[item.profile]]" alt="profile" width="20px">
```

```
<img src="/img/[[item.profile]]" alt="profile" width="20px">
            </vaadin-grid-column>
            <vaadin-grid-column>
              <template class="header">Npm</template>
                <div>[[item.npm]]</div>
            </vaadin-grid-column>
            <vaadin-grid-column width="150px">
              <template class="header">Nama</template>
                <div>[[item.nama]]</div>
            </vaadin-grid-column>
            <vaadin-grid-column>
              <template class="header">Kelas</template>
38
                <div>[[item.kelas]]</div>
            </vaadin-grid-column>
            <vaadin-grid-column width="200px">
              <template class="header">Action</template>
                <paper-button id="[[item.id]]" class="warning" on-click="openEditModal">Edit
                </paper-button>
                <paper-button id="[[item.id]]" class="danger" on-click="openDeleteModal">Delete
                </paper-button>
            </vaadin-grid-column>
```

```
class MahasiswaPage extends Polymer.Element {
            static get is() { return 'mahasiswa-page'; }
            static get properties() {
              return {
                response: {
                 type: Object,
               keyword: {
                type: String,
                 value: ''
               url: {
                computed: '_computeUrl(keyword)'
               method: {
                 type: String,
                 value: "GET"
               success: String,
140
               uid: {
                type: Number,
                 notify: true
143
144
145
147
            _computeUrl(keyword) {
             if (keyword) {
               return ['/mahasiswa/search', keyword].join('/');
              } else {
```

form-add.html

```
type: String,
                             value: '/api/mahasiswa'
                         method: {
                             type: String,
                             value: 'GET'
                         },
                         success: {
                             type: String,
                             notify: true
                         },
                         response: {
                             type: Object,
                             notify: true
                 addMahasiswa() {
                     this.url = '/api/mahasiswa';
                     this.method = 'POST';
                     this.success = 'Data mahasiswa berhasil ditambahkan';
                     this.$.mahasiswaAjax.params = this.formData;
                     this.formData = {};
93
                 _handleInsert(e) {
                     this.method = 'GET';
95
                     this.set('response', JSON.parse(e.detail.response));
97
98
             window.customElements.define(FormAdd.is, FormAdd);
         </script>
```

form-edit.html

```
nocity: crue
                         },
                         response: {
                              type: Object,
                             notify: true
                         },
                         uid: Number,
                 _computeUrl(uid) {
                     return ['/api/mahasiswa', uid].join('/');
83
                 editMahasiswa() {
85
                     this.url = '/api/mahasiswa';
86
                     this.method = 'PUT';
87
                     this.success = 'Data mahasiswa berhasil diubah';
                     this.$.editMahasiswaAjax.params = this.formData;
89
                     this.formData = {};
                 _handleUpdate(e) {
93
                     this.method = 'GET';
                     this.set('response', JSON.parse(e.detail.response));
94
96
                 _handleGet(e) {
98
                     this.set('formData', e.detail.response.data);
```

form-delete.html