

Contents

BAB 1 PENDAHULUAN	2
1.1 Latar Belakang	2
1.2 Tujuan	2
BAB 2 PEMBAHASAN.....	3
2.1 JavaScript.....	3
2.2 Node.js	3
2.2.1 NPM.....	3
2.3 React.js.....	3
2.4 ExpressJS	4
2.5 CORS	4
2.6 Axios	4
2.7 MariaDB	4
2.8 XAMPP.....	5
2.9 Struktur File	5
BAB 3 ANALISA DAN PERANCANGAN	6
3.1a Daftar Versi	6
3.1b Setup Database.....	6
3.2 Setup Struktur File	6
3.3 Perancangan Program	8
3.3.1 Frontend	8
3.3.2 Backend.....	11
3.4 Output Program.....	17
3.4.1 Kegiatan Create data	18
3.4.2 Kegiatan Update Data	19
3.4.3 Kegiatan Delete Data	20
3.4.4 Kegiatan Get Data.....	20
BAB 4 PENUTUP	21
4.1 Kesimpulan	21
4.2 Saran	21

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi saat ini sangatlah cepat. Teknologi informasi memegang peranan yang sangat penting dalam kehidupan manusia, termasuk dalam pengelolaan gudang senjata. Batas-batas yang ada karena perbedaan ruang dan waktu dapat ditembus dengan keberadaan teknologi khususnya internet. Dengan media internet, pengelolaan gudang senjata dapat dilakukan secara lebih efisien dan efektif.

Awalnya, pengelolaan gudang senjata dilakukan secara manual. Hal ini menyebabkan proses pengelolaan menjadi lebih lama dan rentan terjadi kesalahan. Dengan adanya teknologi informasi, pengelolaan gudang senjata dapat dilakukan secara otomatis. Hal ini akan memudahkan petugas gudang senjata dalam melakukan tugasnya, sehingga proses pengelolaan menjadi lebih cepat dan akurat.

Saat ini, banyak gudang senjata yang telah menerapkan teknologi informasi dalam pengelolaannya. Hal ini menunjukkan bahwa penggunaan teknologi informasi dalam pengelolaan gudang senjata telah menjadi kebutuhan.

1.2 Tujuan

Adapun penulisan ini bertujuan untuk mempermudah sistem pengelolaan penyimpanan senjata pada gudang senjata militer.

BAB 2 PEMBAHASAN

2.1 JavaScript

JavaScript adalah bahasa pemrograman untuk menciptakan berbagai fungsi pada web. Pemrograman ini mengubah web yang awalnya statis, dapat memiliki fitur yang dinamis seperti pembaharuan fitur, kegiatan yang interaktif, grafik 2D/3D, dll. Jika dibandingkan dengan HTML dan CSS, HTML adalah bahasa markup yang mendefinisikan struktur konten dari sebuah web, CSS mendefinisikan style dari konten tersebut, lalu JavaScript mendefinisikan *behavior* dari konten tersebut. Berfungsi layaknya seperti bahasa pemrograman, JavaScript dapat menyimpan variabel, melakukan berbagai operasi, bahkan melakukan programming OOP.

Diatas JavaScript yang sudah tertanam pada klien, dibuatlah berbagai fungsionalitas. Kumpulan fungsionalitas ini disebut API, atau Application Programming Interfaces, libraries, framework, dan khususnya yang akan digunakan kali ini yaitu sebuah Runtime Environment Node.js.

2.2 Node.js

Node.js merupakan salah satu platform pengembang yang dapat digunakan untuk membuat aplikasi berbasis Cloud. Node.js dikembangkan dari engine JavaScript yang dibuat oleh Google untuk browser Chrome ditambah dengan libuv serta beberapa pustaka lainnya. Node.js menggunakan JavaScript sebagai bahasa pemrograman dan event-driven, non-blocking I/O (asynchronous) model yang membuatnya ringan dan efisien. Node.js memiliki fitur built-in HTTP server library yang menjadikannya mampu menjadi sebuah web server tanpa bantuan software lainnya seperti Apache dan Nginx.

Pada dasarnya, Node.js adalah sebuah runtime environment dan script library. Sebuah runtime environment adalah sebuah software yang berfungsi untuk mengeksekusi, menjalankan dan mengimplementasikan fungsi-fungsi serta cara kerja inti dari suatu bahasa pemrograman. Sedangkan script library adalah kumpulan, kompilasi atau bank data berisi skrip/kode-kode pemrograman.

Node.js dibangun menggunakan JavaScript dan C++, terdapat arsitektur serta fungsi dari Google V8 di dalamnya yang berfungsi sebagai compiler ditulis dalam C++ dan library Libuv bekerja untuk menangani operasi asynchronous I/O dan main event loop.

2.2.1 NPM

NPM merupakan paket manager untuk Node.js, yang ditemukan pada tahun 2009 sebagai suatu proyek terbuka untuk membantu pengembang Javascript saling berbagi kode paket modul. NPM juga merupakan sebuah kode perintah untuk memungkinkan pengembang menggunakan dan mempublikasi paket modul.

2.3 React.js

ReactJS atau React merupakan open-source Javascript library untuk mengembangkan antarmuka pengguna yang lebih interaktif dan mempermudah developer dalam perancangan aplikasi. ReactJS digunakan untuk menangani view layer pada aplikasi single-page dan aplikasi mobile. ReactJS dikelola oleh Facebook, Instagram, dan komunitas para developer (Khuat, 2018).

ReactJS berusaha untuk memberikan kecepatan, kesederhanaan, dan skalabilitas. Beberapa fitur ReactJS yang biasa dikenal adalah JSX atau Javascript XML. JSX adalah extension untuk sintaksis ECMAScript. JSX membantu developer pada saat mengembangkan UI di dalam Javascript, dan juga dapat membantu developer ketika sedang melakukan error debugging (Khuat, 2018).

Konsep MVC atau Model View Control pada ReactJS hanya mempresentasikan bagian View saja, dan ini adalah bagian yang terbaik dari ReactJS karena lebih sederhana (Nursaid dkk., 2020).

2.4 ExpressJS

ExpressJS merupakan minimal framework yang sangat fleksibel, memungkinkan pengguna untuk membuat laman server HTML, statik file, layanan web dengan akses REST API atau aplikasi hybrid yaitu selain pengguna mempunyai akses melalui REST API juga dapat memiliki akses pada laman HTML.

2.5 CORS

CORS, singkatan dari *Cross-Origin Resource Sharing*, adalah kebijakan keamanan yang diterapkan oleh browser web untuk melindungi pengguna dari potensi ancaman keamanan yang muncul ketika permintaan HTTP dilakukan dari suatu domain (asal) ke domain lain (tujuan). Kebijakan ini membantu mencegah serangan seperti *Cross-Site Request Forgery* (CSRF) dan *Cross-Site Scripting* (XSS).

Jika halaman web di satu domain mencoba membuat permintaan HTTP ke sumber daya di domain lain, browser umumnya akan menerapkan kebijakan CORS untuk memastikan bahwa sumber daya tersebut hanya dapat diakses oleh domain yang sah. Ini dilakukan dengan menambahkan dan memeriksa *HTTP headers* khusus pada permintaan dan respons HTTP.

Beberapa header CORS umum meliputi:

- **Origin:** Menentukan asal dari permintaan.
- **Access-Control-Allow-Origin:** Menentukan domain yang diizinkan untuk mengakses sumber daya.
- **Access-Control-Allow-Methods:** Menentukan metode HTTP yang diizinkan saat mengakses sumber daya.
- **Access-Control-Allow-Headers:** Menentukan jenis *header* yang diizinkan selama permintaan aktual di prangkat asal (browser).
- **Access-Control-Allow-Credentials:** Menentukan apakah browser harus menyertakan *credentials* (seperti cookie atau *HTTP authentication*) saat membuat permintaan asinkron (mis., dari JavaScript).

Jika server tidak mengonfigurasi CORS dengan benar, browser akan mencegah permintaan dari halaman web di domain yang berbeda. Ini dapat menyebabkan masalah ketika Anda mencoba mengakses sumber daya dari domain yang berbeda, seperti saat mengembangkan aplikasi web frontend yang berkomunikasi dengan backend yang berbeda domain.

Untuk mengatasi kendala CORS, server backend harus dikonfigurasi untuk mengizinkan akses dari domain tertentu. Hal ini dapat dilakukan dengan mengatur header **Access-Control-Allow-Origin** pada server. Jika Anda bekerja dengan Node.js dan Express, misalnya, Anda dapat menggunakan library seperti **cors** untuk menyederhanakan konfigurasi CORS.

2.6 Axios

Axios adalah library OpenSource untuk JavaScript yang memberi fungsionalitas klien HTTP. Axios dapat merequest dan menerima respond data melalui HTTP. Dengan Axios, bagian Frontend dapat berkomunikasi dengan backend.

2.7 MariaDB

MariaDB adalah sistem manajemen basis data (*database management system* atau DBMS) yang *open-source* dan dikembangkan sebagai *fork* dari MySQL. Kedua produk ini awalnya berasal dari MySQL, namun perbedaan utama antara keduanya berkaitan dengan pengembangan, komunitas, dan fokus.

2.8 XAMPP

XAMPP adalah *open-source cross-platform* yang menyediakan paket perangkat lunak yang mendukung pengembangan dan pengujian aplikasi web lokal. Nama "XAMPP" sendiri merupakan singkatan yang mewakili komponen utama dalam paket ini:

1. **X** - Sistem Operasi (Cross-platform, bisa digunakan di berbagai sistem operasi seperti Windows, Linux, Mac)
2. **A** - Server Web Apache
3. **M** - Bahasa Skrip (PHP, Perl)
4. **P** - Database (MySQL)
5. **P** - Pengolah (PHPMyAdmin)

XAMPP dirancang untuk menyederhanakan pengaturan dan konfigurasi server web lokal sehingga pengembang dan desainer web dapat dengan mudah membuat dan menguji situs web mereka tanpa harus terhubung ke server web eksternal. Ini sangat berguna selama tahap pengembangan dan pengujian proyek web sebelum mereka diunggah ke server produksi.

Beberapa komponen utama yang disertakan dalam XAMPP:

- **Apache HTTP Server:** Server web yang populer dan andal yang mendukung protokol HTTP. Apache digunakan untuk meng-host dan menyajikan situs web lokal Anda.
- **MySQL:** Sistem manajemen basis data relasional (*Relational Database Management System* atau RDBMS) yang digunakan untuk menyimpan dan mengelola data. Walau disebut menggunakan MySQL, sebenarnya instalasi XAMPP yang terbaru menggunakan MariaDB. Perbedaan ini penting diperhatikan karena implementasi yang digunakan untuk menyambung backend dengan server berbeda.
- **PHP, Perl, dan bahasa skrip lainnya:** Bahasa-bahasa skrip yang digunakan untuk mengembangkan aplikasi web dinamis.
- **phpMyAdmin:** Antarmuka web berbasis PHP yang memungkinkan pengelolaan database MySQL melalui antarmuka grafis. Ini mempermudah administrasi database tanpa harus menggunakan baris perintah SQL.

XAMPP menyediakan lingkungan pengembangan web lokal yang lengkap dan mudah digunakan. Ini sangat berguna untuk menguji proyek web di lingkungan lokal sebelum mereka diunggah ke server produksi. Selain itu, XAMPP mendukung beberapa sistem operasi, menjadikannya alat yang fleksibel dan dapat digunakan di berbagai platform.

2.9 Struktur File

Aplikasi Gudang Senjata ini memerlukan implementasi Frontend dan Backend. Karena itu, project secara garis besar dibagi menjadi dua bagian:

- server (backend)
- client (frontend)

BAB 3 ANALISA DAN PERANCANGAN

Proyek yang dibuat adalah perancangan frontend dan backend aplikasi CRUD (Create, Update, Read dan Delete) dari database “manifest_gudang_senjata”. Pada proyek ini, database hanya akan menampung satu tabel yaitu “manifest_senjata”. Perancangan akan dilakukan dengan IDE Visual Studio Code.

3.1a Daftar Versi & Github

Berikut adalah daftar versi dari library, Runtime Environment, dan aplikasi-aplikasi lain yang digunakan:

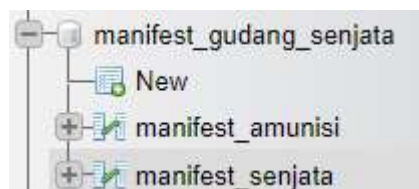
1. Visual Studio Code 1.73.1
2. GitHub Desktop
3. XAMPP Control Panel v.3.3.0
4. Server MariaDB: Server version: 10.4.32-MariaDB (Bawaan dari XAMPP)
5. Apache/2.4.58 (Win64) (Bawaan dari XAMPP)
6. Node.js 20.9.0
7. Body-parser 1.20.2 (didapat melalui npm)
8. Express 4.18.2 (didapat melalui npm)
9. CORS 2.8.5 (didapat melalui npm)
10. MariaDB 3.2.2 (didapat melalui npm)
11. Axios 1.6.2 (didapat melalui npm)

Link ke repository GitHub: <https://github.com/GrenArisaka/SimpleGudangSenjata>

3.1b Setup Database

Server database dijalankan dengan MariaDB pada XAMPP. Berikut struktur dari database dan tabel.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 nama	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3 varian	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	4 amunisi	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	5 tahun	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	6 jumlah	int(11)			No	None			Change Drop More



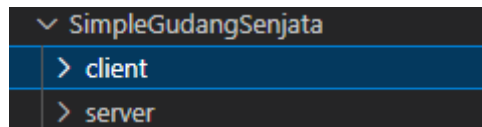
Catatan: abaikan manifest_amunisi.

Selama perancangan aplikasi, server database menyala.

3.2 Setup Struktur File

Pada bagian ini, akan dibuat struktur file dasar dari proyek dan juga metode untuk menginstall library-library yang diperlukan.

Dibuat sebuah folder client untuk aplikasi frontend dengan ReactJS lalu server untuk aplikasi backend.



Dengan terminal, jalankan pada folder client:

```
SimpleGudangSenjata\client> npx create-react-app .
```

Dengan terminal baru, jalankan pada folder server:

```
SimpleGudangSenjata\server> npm init
```

Command pada terminal server akan meminta input, tekan tombol enter sampai selesai. Setelah itu, pada terminal server, jalankan perintah ini untuk menginstall library express, cors, mariadb dan body-parser.

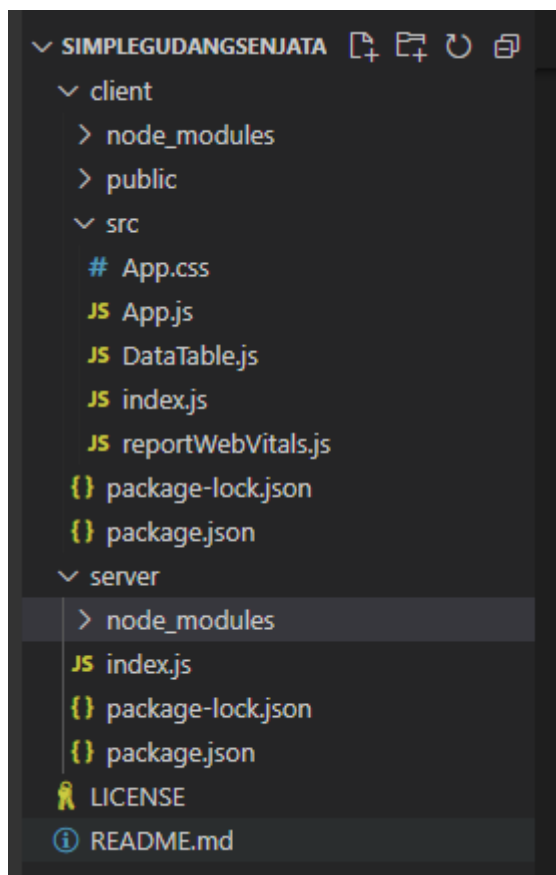
```
npm i express cors mariadb body-parser
```

Kemudian, pada terminal client, jalankan command berikut untuk menginstall axios.

```
SimpleGudangSenjata\server> npm i axios
```

Lalu, ada beberapa file di client yang tidak diperlukan, yaitu index.css, logo.svg, dan setupTests.js maka ketiga file tersebut dihapus. Tambahkan sebuah file DataTable.js yang akan menjadi komponen penampung tabel dari database.

Kemudian, pada server, buat sebuah file utama backend yaitu index.js. Berikut hasil akhir struktur file dari proyek ini.



3.3 Perancangan Program

Program ini akan menghandle kegiatan Create, Read, Update dan Delete dari tabel manifest_senjata pada database manifest_gudang_senjata yang sudah dijelaskan pada bagian sebelumnya. Terdapat sebuah bagian form yang terdiri dari komponen HTML <input> yang berfungsi sebagai input dari field-field yang ada pada tabel manifest_senjata. Kemudian adapun sebuah tombol edit untuk mengedit nilai data yang sudah ada (hanya tahun dan jumlah) berdasarkan varian yang diinput. Ada juga sebuah tombol dan textbox khusus untuk mencari dan mendelete varian yang dispesifikasi pengguna aplikasi.

Diluar bagian form tersebut, ada sebuah bagian yang akan mendisplay isi tabel. Aplikasi menyediakan tombol refresh untuk merefresh tabel yang sudah dimanipulasi.

3.3.1 Frontend

Berikut isi file App.js, yaitu file utama desain dengan reactJS.

```
//Author: GrenArisaka (Tristan Suud)
//Frontend ReactJS

import { useState, useEffect } from 'react';
import DataTable from './DataTable';
import axios from 'axios';
import './App.css';

function App() {
  const [nama, setNama] = useState('');
  const [varian, setVarian] = useState('');
  const [tahun, setTahun] = useState('');
  const [amunisi, setAmunisi] = useState('');
  const [jumlah, setJumlah] = useState(0);
  const [varian2, setVarian2] = useState('');
  const [senjataData, setSenjataData] = useState([]);
  //Function to handle post request.
  const handleSenjataCreate = async () =>{
    try{
      const response = await axios.post('http://localhost:8081/senjata/create', {
        nama: nama,
        varian: varian,
        tahun: tahun,
        amunisi : amunisi,
        jumlah: jumlah
      }).then(()=>{console.log("success")});
    } catch (error){
      console.log("whoops, ada error: "+error.message);
    }
  }
  //Function to handle put request.
  const handleSenjataEdit = async () => {
    try {
      const varianToUpdate = varian;
      const updateData = {tahun, amunisi, jumlah};
    }
  }
}
```



```

    const response = await
axios.put(`http://localhost:8081/senjata/put/${varianToUpdate}`, updateData);
    console.log('Update response:', response.data);
  } catch (error)
  {
    console.log("whoops, ada error: "+error.message);
  }
}
//Function to handle delete request.
const handleSenjataDelete = async ()=>{
  const varianToDelete = varian2; // Replace with the actual name you want to
delete

  try {
    const response = await
axios.delete(`http://localhost:8081/senjata/deleteData/${varianToDelete}`);
    console.log('Delete response:', response.data);
    // Optionally, handle UI updates or fetch updated data
  } catch (error) {
    console.error('Error deleting data:', error.message);
  }
}
//Function to handle getting the data.
const handleSenjataGet = async ()=>{
  try{
    const response = await axios.get('http://localhost:8081/senjata/get');
    setSenjataData(response.data);
    console.log('Response Data:', response.data);
  } catch (error){
    console.log("whoops, ada error di get: "+error.message);
  }
}
//ReactJS design
return (
  <div className="App">
    <h1>Manifest Gudang Senjata</h1>
    <div className="container">
      <div className="form">
        <h2>Masukkan Item Baru pada tabel senjata.</h2>
        <label>Nama: </label>
        <input type="text" onChange={(e)=>{setNama(e.target.value);}}></input>
        <label>Varian: </label>
        <input type="text" onChange={(e)=>{setVarian(e.target.value);}}></input>
        <label>Amunisi: </label>
        <input type="text"
onChange={(e)=>{setAmunisi(e.target.value);}}></input>
        <label>Tahun: </label>
        <input type="text" onChange={(e)=>{setTahun(e.target.value);}}></input>

```

```

        <label>Jumlah: </label>
        <input type="number"
onChange={e=>{setJumlah(e.target.value);}}></input>
        <button onClick={handleSenjataCreate}>Add New</button>
        <p>Edit data, data dicari berdasarkan varian.</p>
        <button onClick={handleSenjataEdit}>Edit Data</button>
        <div className='deleteForm'>
            <p>Delete sebuah record berdasarkan varian. Hati-hati.</p>
            <label>Varian: </label>
            <input type="text"
onChange={e=>{setVarian2(e.target.value);}}></input>
            <button onClick={handleSenjataDelete}>Delete</button>
        </div>
    </div>
    <div className="dataholder">
        <button onClick={handleSenjataGet}>Refresh</button>
        <h2>Data Gudang Senjata</h2>
        {senjataData !== null && senjataData.length > 0 ? (
        <DataTable data={senjataData} />
        ) : (
        <p></p>
        )}
    </div>
</div>
);
}
export default App;

```

Berikut isi dari DataTable.js

```

import React from 'react'
import './App.css';
const DataTable = ({ data }) => {
    return (
        <table>
            <thead>
                <tr>
                    { /* Render table headers based on keys of the first data item */}
                    {Object.keys(data[0]).map((key) => (
                        <th key={key}>{key.replace(/(^\\w|\\s\\w)/g, (match) =>
match.toUpperCase())}</th>
                    ))}
                </tr>
            </thead>
            <tbody>
                { /* Render table rows based on data */}
                {data.map((item, index) => (

```

```

        <tr key={index}>
          {Object.values(item).map((value, index) => (
            <td key={index}>{value}</td>
          ))}
        </tr>
      </tbody>
    </table>
  );
};

```

`export default DataTable`

Penjelasan Kode:

- Pada aplikasi ReactJS tersebut, digunakan fungsionalitas useState dari ReactJS untuk menampung nilai-nilai yang field dengan tabel di database. Setiap nilai memiliki nama variabel dari nilai tersebut (nama, varian, tahun, amunisi, jumlah) dan sebuah fungsi perubah nilai (setName, setVarian, setTahun, setAmunisi, setJumlah).
- Setiap field <input> menggunakan event onChange untuk selalu merubah nilai variabel field dengan text yang ditampilkan.
- Ada empat fungsi utama yang berinteraksi dengan database, yaitu handleSenjataCreate, handleSenjataEdit, handleSenjataDelete dan handleSenjataGet.
 - handleSenjataCreate melakukan request ke backend server untuk membuat sebuah record baru dengan memberikan data yang diinput pengguna sebagai isi dari body request.
 - handleSenjataEdit melakukan request ke backend server untuk mengupdate data yang sudah ada berdasarkan varian
 - handleSenjataDelete melakukan request untuk mendelete sebuah record menggunakan varian
 - handleSenjataGet melakukan request untuk mendapatkan seluruh isi tabel manifest_senjata.
- Keempat fungsi tersebut dijalankan melalui event onClick masing masing tombol Add new, Edit, Delete, dan Refresh.
- Digunakan komponen DataTable yaitu komponen untuk menyajikan data dari tabel manifest_senjata.

3.3.2 Backend

Pemrograman terjadi di file index.js pada folder server. Isi dari file tersebut adalah: (Penjelasan ada di comment kode)

```

//Author GrenArisaka (Tristan Suud)
//Backend server for Gudang Senjata.

//imports
/*
express : Node.js web app framework
mariadb : provides connection to the MariaDB database
'manifest_gudang_senjata'
cors : provides HTTP-header based interactions
bodyparser : parses json sent by frontend

```

```

*/

const express = require('express');
const mariadb = require('mariadb');
const cors = require('cors');
const bodyparser = require('body-parser');

const app = express();
const port = 8081;

app.use(cors());
app.use(bodyparser.json());

//connection pool init
const pool = mariadb.createPool({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'manifest_gudang_senjata',
  connectionLimit: 5
});

//create route
app.post("/senjata/create", async (req,res)=>{
  //extract data from request body
  const nama = req.body.nama;
  const varian = req.body.varian;
  const amunisi = req.body.amunisi;
  const tahun = req.body.tahun;
  const jumlah = req.body.jumlah;

  //nullcheck everything
  if (!nama){
    return res.status(400).send('Name is required.');
```

```

    //query
    let connection;
    try {
        connection = await pool.getConnection();
        const rows = await connection.query('INSERT INTO manifest_senjata (nama,
varian, amunisi, tahun, jumlah) VALUES (?, ?, ?, ?, ?)', [nama, varian, amunisi,
tahun, jumlah]);
        console.log('Result:', rows);
        res.send("Values Inserted");
    } catch (error) {
        console.error('Error:', error);
    } finally {
        if (connection) {
            connection.end();
        }
    }
});

//get route
app.get('/senjata/get', async (req, res) => {
    try {
        // Logic Implementation the logic to retrieve data from the database
        const data = await getDataFromDatabase();

        res.json(data);
    } catch (error) {
        console.error('Error retrieving data:', error);
        res.status(500).json({ error: 'Internal Server Error' });
    }
});

//Update route
app.put('/senjata/put/:varian', async (req, res) => {
    const { varian } = req.params;
    const updatedData = req.body;

    //query
    try {
        // Logic Implementation to update the record based on 'varian'
        const result = await updateDataInDatabase(varian, updatedData);
        res.json({ success: true, message: 'Record updated successfully' });
    } catch (error) {
        console.error('Error updating data:', error);
        res.status(500).json({ success: false, error: 'Internal Server
Error' });
    }
});

```

```

//delete route
app.delete('/senjata/deleteData/:varian', async (req, res) => {
  const { varian } = req.params;

  try {
    // Logic Implementation to delete the record based on the name
    const result = await deleteDataFromDatabase(varian);

    res.json({ success: true, message: 'Record deleted successfully' });
  } catch (error) {
    console.error('Error deleting data:', error);
    res.status(500).json({ success: false, error: 'Internal Server
Error' });
  }
});

const deleteDataFromDatabase = async (varian) => {
  //obtain connection
  const connection = await pool.getConnection();
  //query
  try {
    const result = await connection.query('DELETE FROM manifest_senjata
WHERE varian = ?', [varian]);
    return result;
  } catch (error) {
    throw error;
  } finally {
    connection.end();
  }
};

const updateDataInDatabase = async (varian, updatedData) => {
  //setClause would be the data to be updated, a string then provided into
the mysql query.
  const setClause = Object.entries(updatedData)
    .map(([key, value]) => `${key} = '${value}'`)
    .join(', ');
  console.log(setClause);
  const connection = await pool.getConnection();
  const result = await connection.query('UPDATE manifest_senjata SET
'+setClause+' WHERE varian = ?', [varian]);
  console.log(result);
  connection.end();
  return result;
};

const getDataFromDatabase = async () => {
  const connection = await pool.getConnection();
  const result = await connection.query('SELECT * FROM manifest_senjata');

```

```

        connection.end();

        return result;
    };

//unused function to test connection to database
async function runQuery() {
    let connection;
    try {
        connection = await pool.getConnection();
        const rows = await connection.query('SELECT * FROM manifest_senjata');
        console.log('Result:', rows);
    } catch (error) {
        console.error('Error:', error);
    } finally {
        if (connection) {
            connection.end();
        }
    }
}

//establish server
app.listen(port, () => {
    console.log(`Server berjalan di http://localhost:${port}`);
});

```

Penjelasan Kode:

Penjelasan kode tertera pada berbagai comment dari kode diatas. Namun, berikut beberapa hal yang penting diperhatikan:

- Digunakan MariaDB untuk membuat koneksi ke database.
- Digunakan express framework
- Digunakan Cors
- Body-parser digunakan untuk menerjemahkan json yang didapat dari request frontend.

Untuk menerima request dari axios frontend, diperlukan beberapa route. Pembuatan route terjadi di app.post(), app.put(), app.get() dan app.delete(). Beberapa fungsi tersebut menggunakan fungsi lain yaitu updateDataInDatabase(), getDataFromDatabase, dan deleteDataFromDatabase. Ketiga fungsi tersebut melakukan manipulasi langsung melalui koneksi MariaDB dengan tabel database, atas request dari frontend melalui alamat HTTP masing-masing request, (/senjata/create, /senjata/get, /senjata/put/:varian, /senjata/delete/:varian).

Server dihost sesuai dengan port sesuai dengan variabel port.

Berikut adalah CSS pada frontend (/client/App.css)

```

.App {
  background-color: rgb(214, 248, 219);
  padding: 40px;
  padding-bottom: 60px;
}
h1 {
text-align: center;
font-family: Arial;
}
h2 {
  font-family: Arial;
}
h3 {
  font-family: Arial;
}
.container{
  display: flex;
  flex-direction: row;
  justify-content: center;
}

.form {
  display:flex;
  flex-direction: column;
  background-color: rgb(130, 175, 130);
  margin:5px;
  padding: 5px;
  padding-bottom: 10px;
}

.form input {
  width:200px;
  height: 20px;
  margin:5px;
}

.form button {
  margin-top: 5px;
  margin-left:5px;
  width:150px;
  height:30px
}
.form button:hover{
  cursor:pointer;
}
.dataholder {
  display:flex;

```



```

flex:2;
flex-direction: column;
background-color:rgb(130, 175, 130);
margin: 5px;
padding: 5px;
padding-bottom: 10px;
}
.dataholder button{
margin-top: 5px;
margin-left:5px;
width:150px;
height:30px
}
.dataholder table, th, td{
border: 1px solid;
}
.dataholder th{
text-align:left;
}

.deleteForm {
margin: 5px;
padding: 5px;
padding-bottom: 10px;
background-color:rgb(0, 0, 0);
color:white
}
.deleteForm p{
color:red;
margin: 2px;
padding:2px;
border-radius: 1px;
border-color: white;
}
.dataholder th{
text-align:left;
}

```

3.4 Output Program

Berikut adalah tampilan dari website.



3.4.1 Kegiatan Create data

Berikut adalah hasil screenshot kegiatan create data dan melakukan refresh (**get data**).

Masukkan Item Baru pada tabel senjata.

Nama:

Varian:

Amunisi:
















Tahun:

Jumlah:

Edit data, data dicari berdasarkan varian.

Data Gudang Senjata					
Id	Nama	Varian	Amunisi	Tahun	Jumlah
1	AK-74	AKS-74	5.45x39mm	2010	40
2	AK-74	AK-74M	5.45x39mm	1991	120
3	HK MP5	MP5K	5.56x45mm NATO	2001	76
4	Webley Revolver	Webley Mk. VI	.45 ACP	1965	32
5	M1911	M1911A1	.45 ACP	2010	14

Di phpmyadmin:

← T →			▼	id	nama	varian	amunisi	tahun	jumlah			
<input type="checkbox"/>		Edit		Copy		Delete	1	AK-74	AKS-74	5.45x39mm	2010	40
<input type="checkbox"/>		Edit		Copy		Delete	2	AK-74	AK-74M	5.45x39mm	1991	120
<input type="checkbox"/>		Edit		Copy		Delete	3	HK MP5	MP5K	5.56x45mm NATO	2001	76
<input type="checkbox"/>		Edit		Copy		Delete	4	Webley Revolver	Webley Mk. VI	.45 ACP	1965	32
<input type="checkbox"/>		Edit		Copy		Delete	8	M1911	M1911A1	.45 ACP	2010	14

3.4.2 Kegiatan Update Data

Berikut sebuah record dengan varian “M1911A1” akan diupdate dan diubah nilai “Jumlah” menjadi 500.

Nama:

Varian:

Amunisi:

Tahun:

Jumlah:

Edit data, data dicari berdasarkan varian.

Id	Nama	Varian	Amunisi	Tahun	Jumlah
1	AK-74	AKS-74	5.45x39mm	2010	40
2	AK-74	AK-74M	5.45x39mm	1991	120
3	HK MP5	MP5K	5.56x45mm NATO	2001	76
4	Webley Revolver	Webley Mk. VI	.45 ACP	1965	32
5	M1911	M1911A1	.45 ACP	2010	500

Di phpmysqladmin:

Extra options

					id	nama	varian	amunisi	tahun	jumlah
<input type="checkbox"/>	Edit	Copy	Delete		1	AK-74	AKS-74	5.45x39mm	2010	40
<input type="checkbox"/>	Edit	Copy	Delete		2	AK-74	AK-74M	5.45x39mm	1991	120
<input type="checkbox"/>	Edit	Copy	Delete		3	HK MP5	MP5K	5.56x45mm NATO	2001	76
<input type="checkbox"/>	Edit	Copy	Delete		4	Wheley Revolver	Wheley Mk. VI	.45 ACP	1965	32
<input type="checkbox"/>	Edit	Copy	Delete		8	M1911	M1911A1	.45 ACP	2010	500

3.4.3 Kegiatan Delete Data

Delete data dengan varian M1911A1 pada bagian delete.

Delete sebuah record berdasarkan varian. Hati-hati.

Varian:

Refresh

Data Gudang Senjata

Id	Nama	Varian	Amunisi	Tahun	Jumlah
1	AK-74	AKS-74	5.45x39mm	2010	40
2	AK-74	AK-74M	5.45x39mm	1991	120
3	HK MP5	MP5K	5.56x45mm NATO	2001	76
4	Wheley Revolver	Wheley Mk. VI	.45 ACP	1965	32

Extra options

					id	nama	varian	amunisi	tahun	jumlah
<input type="checkbox"/>	Edit	Copy	Delete		1	AK-74	AKS-74	5.45x39mm	2010	40
<input type="checkbox"/>	Edit	Copy	Delete		2	AK-74	AK-74M	5.45x39mm	1991	120
<input type="checkbox"/>	Edit	Copy	Delete		3	HK MP5	MP5K	5.56x45mm NATO	2001	76
<input type="checkbox"/>	Edit	Copy	Delete		4	Wheley Revolver	Wheley Mk. VI	.45 ACP	1965	32

3.4.4 Kegiatan Get Data

Get data dapat dilakukan dengan menekan tombol refresh pada website.

Refresh

Data Gudang Senjata

Id	Nama	Varian	Amunisi	Tahun	Jumlah
1	AK-74	AKS-74	5.45x39mm	2010	40
2	AK-74	AK-74M	5.45x39mm	1991	120
3	HK MP5	MP5K	5.56x45mm NATO	2001	76
4	Wheley Revolver	Wheley Mk. VI	.45 ACP	1965	32

BAB 4 PENUTUP

4.1 Kesimpulan

Dengan adanya implementasi CRUD ini, pengelolaan gudang senjata menjadi semakin efisien. Walau sederhana, implementasi ini dapat menjadi acuan pertama dalam pengembangan lebih lanjut. Selain itu, mahasiswa menjadi mampu mengupayakan berbagai ilmu, yaitu programming dan desain untuk membuat sebuah aplikasi yang fungsional.

4.2 Saran

Saya sering merasa bahwa video penjelasan pada praktikum sudah bagus dan jelas, namun lebih baik apabila terdapat sebuah file sebagai sarana referensi, atau materi bacaan yang lengkap *dan* untuk masuk ke dalam kuis pretest dan posttest. Saya tekankan pada pretest dan posttest karena biasanya soalnya sedikit, sehingga resiko kesalahannya tinggi, tetapi terkadang soal tersebut terkesan ambigu, seperti berasal dari sebuah materi bacaan yang memang tidak diberikan ke mahasiswa, hanya sebagai bahan ajaran untuk pengajar.