

PEMROGRAMAN WEB BERBASIS SERVICES TUGAS AKHIR – AWONAPAKARYA

Wahyu Pambudi – 19312302 (Backend)
Muhammad Ali Nasir – 19312248 (FrontEnd)

PENJELASAN PROGRAM

Team AwonapaKarya membuat Website Berbasis Services dengan study Kasus Website **Website Inventaris Barang Sekolah** yang dikembangkan menggunakan Framework Javascript. Untuk backend menggunakan ExpressJs, sedangkan untuk Frontend menggunakan ReactJS.

Berikut ini tautan repository Github :

<https://github.com/wahyupambudi/TA-GAB2-AwonapaKarya>

PENJELASAN PROGRAM BACKEND

Program Backend di kembangkan menggunakan NodeJs Framework ExpressJS. Menggunakan dependencies :

```
"dependencies": {  
  "argon2": "^0.30.2", (hash password)  
  "connect-session-sequelize": "^7.1.5", (session login user)  
  "cookie-parser": "^1.4.6", (parser cookie jwt)  
  "cors": "^2.8.5", (CORS)  
  "dotenv": "^16.0.3", (untuk menyimpan variable .env)  
  "express": "^4.18.2", (framework express)  
  "express-fileupload": "^1.4.0", (express file upload)  
  "express-session": "^1.17.3", (session express)  
  "jsonwebtoken": "^9.0.0", (JWT Express)  
  "mysql2": "^2.3.3",  
  "sequelize": "^6.28.0"  
}
```

- **Middleware/AuthUser.js**
 - Untuk mengakses **REST API Server** (data barang/jasa dan data user). User diharuskan **login** terlebih dahulu, disini menggunakan middleware. Di dalam file **AuthUser.js** terdapat fungsi **verifyUser** jika session user tidak ditemukan maka akan menampilkan pesan “mohon login ke akun anda” hal ini berlaku disemua routers.

- Selain itu terdapat fungsi **adminOnly**, yang akan digunakan untuk routers / dashboard admin dalam melakukan manajemen user.

The screenshot shows the implementation of the `verifyUser` middleware in `AuthUser.js`. The code checks if the user is logged in and if the user is an admin. If not, it returns a 401 status with a message. The REST client on the right shows a GET request to `http://localhost:2023/barangs/` resulting in a 401 Unauthorized response with the message: "Mohon login ke akun anda!"

```

1 import User from "../models/UserModel.js";
2
3 export const verifyUser = async (req, res, next) => {
4   if (!req.session.userId) {
5     return res.status(401).json({ msg: "Mohon login ke akun anda!" });
6   }
7   const user = await User.findOne({
8     where: {
9       uuid: req.session.userId,
10     },
11   });
12   if (!user) return res.status(404).json({ msg: "User tidak ditemukan" });
13   // jika user ditemukan
14   req.userId = user.id;
15   req.role = user.role;
16   next();
17 };
18
19 export const adminOnly = async (req, res, next) => {
20   const user = await User.findOne({
21     where: {
22       uuid: req.session.userId,
23     },
24   });
25   if (!user) return res.status(404).json({ msg: "User tidak ditemukan" });
26   // jika user bukan admin
27   if (user.role !== "admin")
28     return res.status(403).json({ msg: "Akses Di Tolak" });
29   next();
30 };

```

GET `http://localhost:2023/barangs/` Send

Query Parameters

Status: 401 Unauthorized Size: 1 Bytes Time: 249 ms

Response

```

1 {
2   "msg": "Mohon login ke akun anda!"
3 }

```

- Terdapat **verifyToken** yang digunakan untuk verifikasi token, jika terdapat token maka user bisa melakukan akses data ke REST API.
- Terdapat `ketuaJurusan` yang berguna untuk membatasi akses user dengan role ketua jurusan hanya bisa melihat (Read) data saja.
-

The screenshot shows the implementation of the `verifyToken` middleware and the `ketuaJurusan` route in `AuthUser.js`. The `verifyToken` function checks if the token is valid and if the user is a department head. The `ketuaJurusan` function restricts access to department heads. The REST client on the right shows a GET request to `http://localhost:2023/barangs/` resulting in a 401 Unauthorized response with the message: "Mohon login ke akun anda!"

```

20 // membuat fungsi verify token dari jwt
21 export const verifyToken = (req, res, next) => {
22   const authHeader = req.headers["authorization"];
23   const token = authHeader && authHeader.split(" ")[1];
24   if (token == null) return res.sendStatus(401);
25   jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, (err, decoded) => {
26     if (err) {
27       // console.log(err);
28       return res.sendStatus(403);
29     }
30     req.email = decoded.email;
31     next();
32   });
33 };
34
35 // membuat fungsi jika ketua jurusan hanya bisa view data
36 export const ketuaJurusan = async (req, res, next) => {
37   const user = await User.findOne({
38     where: {
39       uuid: req.session.userId,
40     },
41   });
42   if (!user) return res.status(404).json({ msg: "User Tidak ditemukan" });
43   // jika user adalah ketuaJurusan
44   if (user.role === "ketuaJurusan")
45     return res.status(403).json({ msg: "Akses Tidak Di izinkan" });
46   next();
47 };

```

GET `http://localhost:2023/barangs/` Send

Query Parameters

Status: 401 Unauthorized Size: 1 Bytes Time: 249 ms

Response

```

1 {
2   "msg": "Mohon login ke akun anda!"
3 }

```

- **Login (Auth.js)**

```

TA > server > controllers > Auth.js > login > user > where
1  import User from "../models/UserModel.js";
2  import argon2 from "argon2";
3
4  // function login
5  export const login = async (req, res) => {
6    const user = await User.findOne({
7      where: {
8        email: req.body.email,
9      },
10    });
11    // jika user tidak ditemukan
12    if (!user) return res.status(404).json({ msg: "User tidak ditemukan" });
13
14    // jika user ditemukan
15    const match = await argon2.verify(user.password, req.body.password);
16
17    // jika password tidak betul
18    if (!match) return res.status(400).json({ msg: "Password Salah" });
19
20    // jika password cocok
21    req.session.userId = user.uuid;
22    // get data user
23    const uuid = user.uuid;
24    const name = user.name;
25    const email = user.email;
26    const role = user.role;
27    res.status(200).json({
28      status: true,
29      message: "Berhasil Login",
30      dataUser: { uuid, name, email, role },
31    });
32  };

```

- Proses login dengan mencari user berdasarkan email, jika tidak ditemukan akan

menampilkan pesan “user tidak ditemukan”

- Melakukan pemeriksaan password dengan verify menggunakan argon2. Jika password tidak sesuai maka menampilkan pesan “password salah”.
- Jika berhasil maka akan melanjutkan proses berupa login dan akan membuat session. Dan akan menampilkan data berupa JSON, yaitu data user, id, name, email dan role user.

Update Proses Login

```

const accessToken = jwt.sign(
  { uuid, name, email, role },
  // proses env dari file .env
  process.env.ACCESS_TOKEN_SECRET,
  {
    expiresIn: "20m",
  }
);

```

Digunakan untuk mendapatkan akses token jwt dari file .env, token tersebut bertahan selama 20 menit bersamaan dengan session login user.

```
// membuat variabel refreshToken dari data .env
const refreshToken = jwt.sign(
  { uuid, name, email, role },
  // proses env dari file .env
  process.env.REFRESH_TOKEN_SECRET,
  {
    expiresIn: "1d",
  }
);

// update refresh token ke database
await Users.update(
  { refresh_token: refreshToken },
  {
    where: {
      uuid: req.session.userId,
    },
  }
);

// membuat http only cookie
res.cookie("refreshToken", refreshToken, {
  httpOnly: true,
  maxAge: 24 * 60 * 60 * 1000,
});

res.status(200).json({
  status: true,
  message: "Berhasil Login",
  dataUser: { uuid, name, email, role, accessToken },
});
};
```

- Refresh Token berguna untuk memperbarui token setelah waktu dari access token habis, di dalam proses ini tidak perlu login ulang karena token sudah diberikan akses 1 hari.
- Namun jika pada session sudah habis waktunya maka tetap harus login ulang.

- Login Admin

POST ▼ http://109.123.238.13:2024/login Send 200 OK 2.26 s 438 B Just Now ▼

Multipart ² ▼	Auth ▼	Query	Headers ¹	Docs	Preview ▼	Headers ¹²	Cookies ²	Timeline
Add	Delete All	Toggle Description						
email	wahyu@gmail.com	▼	✓	🗑️	<pre> 1 { 2 "status": true, 3 "message": "Berhasil Login", 4 "dataUser": { 5 "uuid": "ca4c9cc3-f57a-43ad-81fc-3205ea61cd5e", 6 "name": "wahyu", 7 "email": "wahyu@gmail.com", 8 "role": "admin", 9 "accessToken": 10 "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkIjoieTYE0YzljYzZtZjUyM0M2FkLTgzMmMtMzIwNmVhbnJfj2DVl1IiwibmFtZSI6IndhaH11IiwiaWF0Ij0jYUh5dUlnbmFpbC5jb20iLCJyb2x1IjoieWRtaW4iLCJpYXQiOjE2NzQyMTQzMjksImV4cCI6MTY3NDIxNTpYOX0.8wtQE51Djjep3sqNf846WLn132mpq-UHxX9k-uTAR-0" 11 } </pre>			
password	123456	▼	✓	🗑️				
email	operator@gmail.com	▼	■	🗑️				
password	123456	▼	■	🗑️				
email	budi@gmail.com	▼	■	🗑️				
password	123456	▼	■	🗑️				

- Index.js yang digunakan untuk manajemen session menggunakan

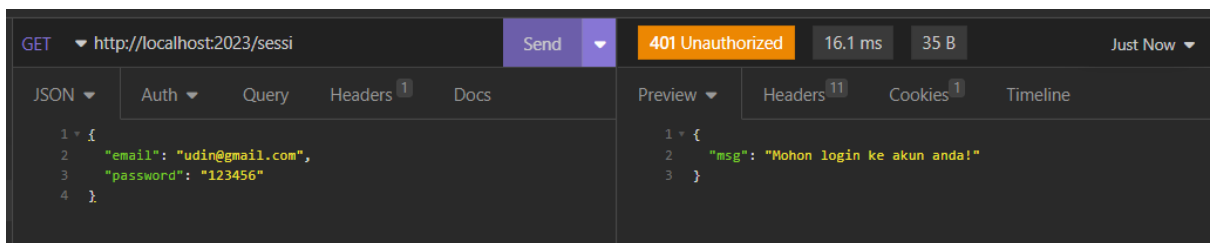
```
import session from "express-session";
import SequelizeStore from "connect-session-sequelize";
```

```
// digunakan untuk sessions
const sessionStore = SequelizeStore(session.Store);
const store = new sessionStore({
  db: db,
  checkExpirationInterval: 43200000, // interval hapus dari database setiap 12 jam
  expiration: 900000, // waktu session 15 menit
});

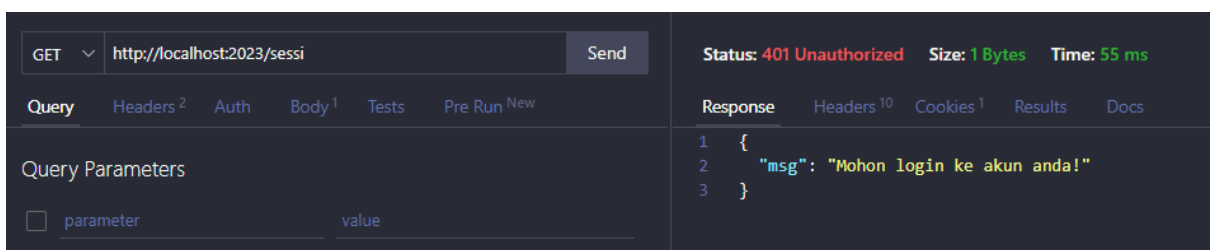
// membuat table session di database
// store.sync();

// definisikan session
app.use(
  session({
    secret: process.env.SESS_SECRET,
    resave: false,
    saveUninitialized: true,
    store: store,
    cookie: {
      // auto jika menggunakan http/s
      secure: "auto",
      maxAge: 900000,
    },
  })
);
```

- Ketika Session sudah habis
 - Login User



- Login Admin



- Logout User (auth.js)

```
// function logout
export const logout = (req, res) => {
  req.session.destroy((err) => {
    // jika gagal logout
    if (err) return res.status(400).json({ msg: "Tidak dapat Logout" });
    // jika berhasil logout
    res.status(200).json({ msg: "Berhasil Logout" });
  });
};
```

CRUD Pada Data User (Admin, User) pada controller Users.js

- Adduser

Untuk menambah user, disini harus login sebagai admin, karena tujuan nya untuk inventaris data barang atau jasa/barang.

```
export const createUser = async (req, res) => {
  // proses destruct
  const { name, email, password, password1, role } = req.body;

  // validasi jika user sama
  const response = await User.findAll();

  // perulangan untuk cek apakah email ada yang sama
  for (let i = 0; i <= response.length; i++) {
    let dataEmail = response[i].email;
    if (dataEmail === email) {
      return res.status(400).json({ msg: "Registrasi Tidak Berhasil" });
    }
  }

  // validasi jika password kosong atau tidak sama
  if (password === "" || password === null) {
    return res.status(400).json({ msg: "Password Tidak Boleh Kosong" });
  } else if (password !== password1) {
    return res.status(400).json({ msg: "Password Tidak Cocok" });
  }

  const hashPassword = await argon2.hash(password);
  try {
    await User.create({
      name: name,
      email: email,
      password: hashPassword,
      role: role,
    });
    res.status(201).json({ msg: "Registrasi Berhasil!" });
  } catch (error) {
    res.status(400).json({ msg: error.message });
  }
};
```

- membuat user dengan req body dari inputan
- Terdapat proses untuk pengecekan apakah ada email yang sama atau tidak.
- Jika password kosong atau tidak sama akan menampilkan pesan json.
- Jika berhasil maka password akan di hash menggunakan argon2.
- kemudian akan menjalankan proses create user.

POST <https://localhost:2023/users> Send Status: 201 Created Size: 30 Bytes Time: 266 ms

Query Headers 2 Auth Body 1 Tests Pre Run New

Json Xml Text Form Form-encode GraphQL Binary

Json Content Format

```
1 {
2   "name": "ketuajurusan",
3   "email": "adam@gmail.com",
4   "password": "123456",
5   "password1": "123456",
6   "role": "user"
7 }
```

Response Headers 3 Cookies Results Docs

```
1 {
2   "msg": "Registrasi Berhasil!"
3 }
```

- Update User
 - Untuk update user menggunakan method Patch. dengan mendapatkan id user menggunakan findOne berdasarkan id user.
 - Jika user tidak ditemukan tampilkan pesan json.
 - Kemudian menerima input dari body
 - Membuat kondisi untuk password jika password kosong maka akan menggunakan password yang lama, jika password di update maka akan melakukan update password baru dan akan melakukan hash password ulang.
 - Melakukan kondisi pengecekan email berdasarkan id
 - Jika email dan id sama maka akan melakukan update.

```

72 export const updateUser = async (req, res) => {
73   const user = await User.findOne({
74     where: {
75       uuid: req.params.id,
76     },
77   });
78
79   // mendapatkan id user untuk dibuat kondisi email
80   let dataIdOne = user.dataValues.id;
81
82   // jika user tidak ditemukan
83   if (!user) return res.status(404).json({ msg: "User tidak ditemukan" });
84
85   // mendapatkan inputan dari user
86   const { name, email, password, password1, role } = req.body;
87
88   // membuat variabel untuk hashpassword
89   let hashPassword;
90   if (password === "" || password === null) {
91     // user dari variabel const user
92     hashPassword = user.password;
93   } else {
94     hashPassword = await argon2.hash(password);
95   }
96
97   // validasi jika user sama
98   const response = await User.findAll();
99   // perulangan untuk cek apakah email ada yang sama
100  for (let i = 0; i <= response.length; i++) {
101    let dataId = response[i].id;
102    let dataEmail = response[i].email;
103    // kondisi jika email sama dan data id tidak sama
104    if (dataEmail === email && dataId !== dataIdOne) {
105      return res.status(400).json({ msg: "Update Tidak Berhasil" });
106    }
107    // kondisi jika password tidak sama
108    else if (password !== password1) {
109      return res.status(400).json({ msg: "Password Tidak Cocok" });
110    }
111    // jika email sama dan data id user sama
112    else if (dataEmail === email || dataId === dataIdOne) {
113      try {
114        // lakukan update user
115        await User.update(
116          {
117            name: name,
118            email: email,
119            password: hashPassword,
120            role: role,
121          },
122          {
123            where: {
124              // user dari variabel const user
125              id: user.id,
126            },
127          }
128        );
129        res.status(200).json({ msg: "User Berhasil Update!" });
130      } catch (error) {
131        res.status(400).json({ msg: error.message });
132      }
133    }
134  }
135 }
  
```

- Delete User
 - Untuk delete user menggunakan method delete.
 - Jika user tidak ditemukan menampilkan json berupa User tidak ditemukan.
 - Jika user ditemukan berdasarkan uuid maka akan melakukan destroy atau menghapus user.


```
export const deleteUser = async (req, res) => {  
  const user = await User.findOne({  
    where: {  
      uuid: req.params.id,  
    },  
  });  
  if (!user) return res.status(404).json({ msg: "User tidak ditemukan" });  
  
  try {  
    await User.destroy({  
      where: {  
        // user dari variabel const user  
        id: user.id,  
      },  
    });  
    res.status(200).json({ msg: "User Berhasil Di Hapus!" });  
  } catch (error) {  
    res.status(400).json({ msg: error.message });  
  }  
};
```

CRUD Pada Data Barang pada controller Barangs.js

- Get Semua Data Barang
 - Jika login sebagai admin, maka akan menampilkan seluruh data barang, yang di input oleh admin ataupun oleh user.
 - Jika login sebagai user data yang ditampilkan hanya berupa data yang di inputkan oleh user saja.

```

Status: 200 OK   Size: 780 Bytes   Time: 40 ms

Response  Headers 9  Cookies  Results  Docs

1  [
2  {
3    "uuid": "dc3bd68d-aad3-4135-9222-ef5ec29c1031",
4    "kd_brg": "kd1",
5    "nm_brg": "Komputer",
6    "spek_brg": "Keren Betul",
7    "jml_brg": 10,
8    "kondisi_brg": "Bagus",
9    "tgl_buy_brg": "2022-10-10T00:00:00.000Z",
10   "harga_brg": 100000,
11   "user": {
12     "name": "Wahyu p",
13     "email": "wahyup@gmail.com"
14   }
15 },
16 {
17   "uuid": "4208dc3c-c16a-4de8-aac2-6f3d31effdb3",
18   "kd_brg": "kd5",
19   "nm_brg": "Komputer Baru",
20   "spek_brg": "Keren Betul",
21   "jml_brg": 10,
22   "kondisi_brg": "Bagus",
23   "tgl_buy_brg": "2022-10-10T00:00:00.000Z",
24   "harga_brg": 100000,
25   "user": {
26     "name": "Budi Keren Aja",
27     "email": "budii@gmail.com"
28   }
29 },
30 {
31   "uuid": "bb4d28cc-983e-437d-88ac-ad7a6da11d3a",
32   "kd_brg": "kd2",
33   "nm_brg": "Komputer Udin",
34   "spek_brg": "Keren Betul",
35   "jml_brg": 10,
36   "kondisi_brg": "Bagus",

```

```

Preview  Headers 10  Cookies  Timeline

1  [
2  {
3    "uuid": "bb4d28cc-983e-437d-88ac-ad7a6da11d3a",
4    "kd_brg": "kd2",
5    "nm_brg": "Komputer Udin",
6    "spek_brg": "Keren Betul",
7    "jml_brg": 10,
8    "kondisi_brg": "Bagus",
9    "tgl_buy_brg": "2022-10-10T00:00:00.000Z",
10   "harga_brg": 100000,
11   "user": {
12     "name": "Udin",
13     "email": "udin@gmail.com"
14   }
15 }
16 ]

```

- Mendapatkan Barang berdasarkan id barang dan user yang login menggunakan op dari sequelize

```

where: {
  // select berdasarkan id dan user id yang login
  [Op.and]: [{ id: barang.id }, { userId: req.userId }],
},
include: [
  {
    model: User,
    attributes: ["name", "email"],
  },
],

```

```

Status: 200 OK   Size: 266 Bytes   Time: 69 ms

Response  Headers 9  Cookies  Results  Docs

1  {
2    "uuid": "4208dc3c-c16a-4de8-aac2-6f3d31effdb3",
3    "kd_brg": "kd5",
4    "nm_brg": "Komputer Baru",
5    "spek_brg": "Keren Betul",
6    "jml_brg": 10,
7    "kondisi_brg": "Bagus",
8    "tgl_buy_brg": "2022-10-10T00:00:00.000Z",
9    "harga_brg": 100000,
10   "user": {
11     "name": "Budi Keren Aja",
12     "email": "budii@gmail.com"
13   }
14 }

```

- Create Barang
 - Mendapatkan inputan dari json yang berisi data barang dari user (req body)
 - Kemudian melakukan pemeriksaan untuk kode barang, bahwa kode barang tidak boleh sama.
 - Jika kondisi terpenuhi maka akan menjalankan proses membuat data barang.
 - Berdasarkan user yang menginput data maka dibutuhkan userId.

```

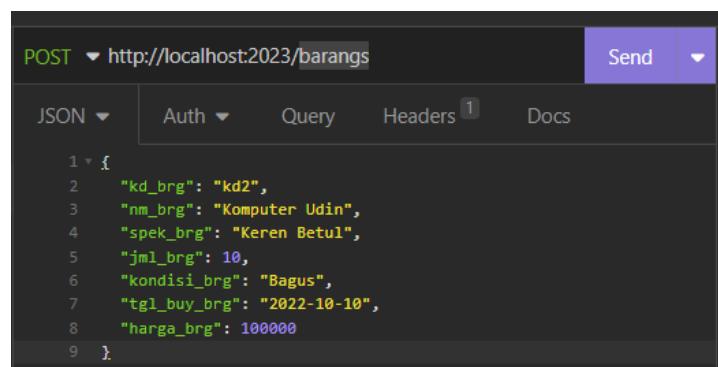
export const createBarang = async (req, res) => {
  const {
    kd_brg,
    nm_brg,
    spek_brg,
    jml_brg,
    kondisi_brg,
    tgl_buy_brg,
    harga_brg,
  } = req.body;

  // mendapatkan semua kode barang
  const getBarangAll = await Barang.findAll();
  for (let i = 0; i < getBarangAll.length; i++) {
    // console.log(getBarangAll[i].kd_brg);
    let new_kd_brg = getBarangAll[i].kd_brg;
    // jika kd_brg sama
    if (new_kd_brg === kd_brg)
      return res.status(500).json({ msg: "Kode Barang Tidak Boleh sama" });
  }

  // proses create barang
  try {
    await Barang.create({
      kd_brg: kd_brg,
      nm_brg: nm_brg,
      spek_brg: spek_brg,
      jml_brg: jml_brg,
      kondisi_brg: kondisi_brg,
      tgl_buy_brg: tgl_buy_brg,
      harga_brg: harga_brg,
      userId: req.userId,
    });
    res.status(201).json({ msg: "Data Barang Berhasil di Simpan." });
  } catch (error) {
    res.status(500).json({ msg: error.message });
  }
};

```

- Contoh method POST yang digunakan untuk input data barang menggunakan JSON.



- Update Barang
 - Untuk mendapatkan barang yang akan di update maka akan melakukan seleksi berdasarkan uuid data barang
 - Mendapatkan input dari user data barang
 - Kemudian jika user sebagai admin maka dapat melakukan update ke semua data barang dari user manapun.

- Jika login sebagai user maka akan melakukan pemeriksaan uuid data user yang dikirim, dan jika kondisi terpenuhi maka akan melakukan Barang.Update.

```

// updateBarang.js
export const updateBarang = async (req, res) => {
  try {
    // mendapatkan kodebarang sesuai id
    const barang = await Barang.findOne({
      where: {
        uuid: req.params.id,
      },
    });
  } catch (error) {
    return res.status(404).json({ msg: "Data Barang Tidak Ditemukan" });
  }

  // jika barang tidak ditemukan
  if (!barang) {
    return res.status(404).json({ msg: "Data Barang Tidak Ditemukan" });
  }

  // jika barang ditemukan ambil data dari req body
  const {
    kd_brg,
    nm_brg,
    spek_brg,
    jml_brg,
    kondisi_brg,
    tgl_buy_brg,
    harga_brg,
  } = req.body;

  // req.role berasal dari middleware ketika login
  if (req.role === "admin") {
    await Barang.update(
      {
        kd_brg,
        nm_brg,
        spek_brg,
        jml_brg,
        kondisi_brg,
        tgl_buy_brg,
        harga_brg,
      },
      {
        where: {
          id: barang.id,
        },
      }
    );
    return res.status(200).json({ msg: "Barang Berhasil di Update" });
  } else {
    // jika user id dan barang user id tidak sama
    if (req.userId !== barang.userId) {
      return res.status(403).json({ msg: "Akses Tidak Ditemukan" });
    }
    // jika kondisi terpenuhi
    await Barang.update(
      {
        kd_brg,
        nm_brg,
        spek_brg,
        jml_brg,
        kondisi_brg,
        tgl_buy_brg,
        harga_brg,
      },
      {
        where: {
          [Op.and]: [{ id: barang.id }, { userId: req.userId }],
        },
      }
    );
    return res.status(200).json({ msg: "Barang Berhasil di Update" });
  }
} catch (error) {
  return res.status(500).json({ msg: error.message });
}

// deleteBarang.js
export const deleteBarang = async (req, res) => {
  try {
    // mendapatkan kodebarang sesuai id
  }

```

- Contoh proses update dari admin untuk data barang dari user budi

```

PATCH http://localhost:2023/barangs/4208dc3c-c16a-4de8-aac2-6f3d31effdb3

{
  "kd_brg": "kd5",
  "nm_brg": "Komputer Keren",
  "spek_brg": "Gaming Spec",
  "jml_brg": 10,
  "kondisi_brg": "Bagus",
  "tgl_buy_brg": "2022-10-10",
  "harga_brg": 100000
}

```

- Delete Barang
 - Proses hapus barang dengan cara mendapatkan data barang dengan findOne berdasarkan barang.id.
 - Jika user sebagai admin maka dapat menghapus seluruh data barang dari user lain.

- Jika login sebagai user maka hanya dapat menghapus untuk user yang login dan data id yang sama.

```
export const deleteBarang = async (req, res) => {
  try {
    // mendapatkan kodebarang sesuai id
    const barang = await Barang.findOne({
      where: {
        uuid: req.params.id,
      },
    });

    // jika barang tidak ditemukan
    if (!barang)
      return res.status(404).json({ msg: "Data Barang Tidak Ditemukan" });

    // req.role berasal dari middleware ketika login
    if (req.role === "admin") {
      await Barang.destroy({
        where: {
          id: barang.id,
        },
      });
    } else {
      // jika user id dan barang user id tidak sama
      if (req.userId !== barang.userId)
        return res.status(403).json({ msg: "Akses Tidak ditemukan" });
      // jika kondisi terpenuhi
      await Barang.destroy({
        where: {
          [Op.and]: [{ id: barang.id }, { userId: req.userId }],
        },
      });
    }
    res.status(200).json({ msg: "Barang Berhasil di Hapus" });
  } catch (error) {
    res.status(500).json({ msg: error.message });
  }
};
```

Penjelasan Program Front End

PENJELASAN PROGRAM FRONT END

Program FrontEnd di kembangkan menggunakan NodeJs Framework ReactJS. Menggunakan Dependencies :

```
"dependencies": {
  "@reduxjs/toolkit": "^1.9.1",
  "@testing-library/jest-dom": "^5.16.5",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^14.4.3",
  "axios": "^1.2.2",
  "bootstrap": "^5.1.0",
  "bulma": "^0.9.4",
  "react": "^18.2.0",
  "react-bootstrap": "^2.0.0-beta.5",
  "react-bulma-dropdown": "^0.1.10",
  "react-dom": "^18.2.0",
  "react-icons": "^4.7.1",
  "react-redux": "^8.0.5",
  "react-router-dom": "^6.6.1",
  "react-scripts": "5.0.1",
  "web-vitals": "^2.1.4"
},
```

Membuat Halaman Navbar dengan Framework Css Bulma :

```
Navbar.jsx M X
client > src > component > Navbar.jsx > Navbar
1 import React from 'react'
2 import { NavLink, useNavigate } from "react-router-dom";
3
```

Import Navlink untuk mengarahkan setiap Menu dengan React-router-dom

```

client > src > component > Navbar.jsx > Navbar
1 import React from 'react'
2 import { NavLink, useNavigate } from "react-router-dom";
3
4 const Navbar = () => {
5   return (
6     <div>
7       <nav className="navbar is-info is-fixed-top has-shadow" role="navigation" aria-label="main navigation">
8         <div className="navbar-brand">
9           <NavLink to="/dashboard" className="navbar-item" href="https://bulma.io">
10             
11           </NavLink>
12
13           <a role="button" className="navbar-burger burger" aria-label="menu" aria-expanded="false" data-target="navbarBasicExample">
14             <span aria-hidden="true"></span>
15             <span aria-hidden="true"></span>
16             <span aria-hidden="true"></span>
17           </a>
18         </div>
19
20         <div id="navbarBasicExample" className="navbar-menu">
21           <div className="navbar-start">
22
23           </div>
24
25           <div className="navbar-end">
26             <div className="navbar-item">
27               <div className="buttons">
28                 <a className="button is-primary">
29                   <strong>Keluar</strong>
30                 </a>
31               </div>
32             </div>
33           </div>
34         </nav>
35       </div>
36     </div>
37   )
38 }
39
40 export default Navbar

```

Membuat Halaman Sidebar dengan mengimport IoPerson, IoHome dari react-icons/io5

Agar bagian setiap menu memiliki Icon.

```

client > src > component > Sidebar.jsx > ...
1 import React from "react";
2 import { NavLink } from "react-router-dom";
3 import { IoPerson, IoHome, } from "react-icons/io5";
4

```

```

5   const Sidebar = () => {
6     return (
7       <div>
8         <aside class="menu pl-2 has-shadow">
9           <p class="menu-label">
10            General
11          </p>
12          <ul class="menu-list">
13            <li><NavLink to={"/dashboard"}><IoHome/> Dashboard</NavLink></li>
14          </ul>
15          <p class="menu-label">
16            Administration
17          </p>
18          <ul class="menu-list">
19            <li>
20              <a class=""><IoPerson/> Manage Users</a>
21              <ul>
22                <li><NavLink to={"/admin"}>Admin</NavLink></li>
23                <li><NavLink to={"/operator"}>Operator</NavLink></li>
24                <li><NavLink to={"/kajur"}>Ketua Jurusan</NavLink></li>
25              </ul>
26            </li>
27          </ul>
28          <p class="menu-label">
29            Belanja
30          </p>
31          <ul class="menu-list">
32            <li><NavLink to={"/barangm"}>Barang Modal</NavLink></li>
33            <li><a>Habis Pakai</a></li>
34          </ul>
35        </aside>
36      </div>
37    )
38  }
39  }
40  }
41  export default Sidebar
42

```

Membuat Halaman Login dengan :

Mengimport useDispatch, useSelector dari react-redux

Menimport LoginUser, reset

```

client > src > component > Login.jsx > ...
1   import React, { useState, useEffect } from "react";
2   import { useDispatch, useSelector } from "react-redux";
3   import { useNavigate } from "react-router-dom";
4   import { LoginUser, reset } from "../features/authSlice";
5

```



```

6  const Login = () => {
7      const [email, setEmail] = useState("");
8      const [password, setPassword] = useState("");
9      const dispatch = useDispatch();
10     const navigate = useNavigate();
11     const { user, isError, isSuccess, isLoading, message } = useSelector(
12         (state) => state.auth
13     );
14
15     useEffect(() => {
16         if (user || isSuccess) {
17             navigate("/dashboard");
18         }
19         dispatch(reset());
20     }, [user, isSuccess, dispatch, navigate]);
21
22     const Auth = (e) => {
23         e.preventDefault();
24         dispatch(LoginUser({ email, password }));
25     };
26

```

```

27     return (
28         <div class="hero is-fullheight">
29             <div class="hero-body is-justify-content-center is-align-items-center">
30                 <div class="columns is-flex is-flex-direction-column box">
31                     <div class="column">
32                         <label for="email">Email</label>
33                         <input class="input is-primary" type="text" placeholder="Email address"/>
34                         <input
35                             type="text"
36                             className="input"
37                             value={email}
38                             onChange={(e) => setEmail(e.target.value)}
39                             placeholder="Email"
40                         />
41                     </div>
42                     <div class="column">
43                         <label for="Name">Password</label>
44                         <input class="input is-primary" type="password" placeholder="Password"/>
45                         <input
46                             type="password"
47                             className="input"
48                             value={password}
49                             onChange={(e) => setPassword(e.target.value)}
50                             placeholder="*****"
51                         />
52                         <a href="/forget.html" class="is-size-7 has-text-primary">forget password?</a>
53                     </div>
54                     <div class="column">
55                         <button class="button is-info is-fullwidth" type="submit">Login</button>
56                     </div>
57                 <div class="has-text-centered">
58                     <p class="is-size-7">Don't have an account? <a href="#" class="has-text-primary">Sign up</a>
59                 </p>
60             </div>
61         </div>
62     </div>
63 </div>
64 )

```

```

client > src > component > Barang.jsx > [?] Barang
1  import React, { useState, useEffect } from "react";
2  import { Link } from "react-router-dom";
3  import axios from "axios";
4

```

```

5   const Barang = () => {
6     const [barang, setBarangm] = useState([]);
7
8     useEffect(() => {
9       getBarangm();
10    }, []);
11
12    const getBarangm = async () => {
13      const response = await axios.get("http://localhost:2023/barangs");
14      setBarangm(response.data);
15    };
16
17    const deleteBarangm = async (barangId) => {
18      await axios.delete(`http://localhost:2023/barangs/${barangId}`);
19      getBarangm();
20    };
21
22    return (
23      <div>
24        <h1 className="title">Barang Modal</h1>
25        <table className="table is-striped is-fullwidth">
26          <thead>
27            <tr>
28              <th>Id</th>
29              <th>Kode Barang</th>
30              <th>Nama Barang</th>
31              <th>Spesifikasi</th>
32              <th>Jumlah Barang</th>
33              <th>Tanggal Pembelian</th>
34              <th>Harga Barang</th>
35            </tr>
36          </thead>

```

```

37     <tbody>
38       {barang.map((Barang, index) => (
39         <tr key={Barang.uuid}>
40           <td>{index + 1}</td>
41           <td>{barang.name}</td>
42           <td>{barang.price}</td>
43           <td>{barang.name}</td>
44           <td>
45             <Link
46               to={` /products/edit/${barang.uuid}`}
47               className="button is-small is-info"
48             >
49               Edit
50             </Link>
51             <button
52               onClick={() => deleteBarangm(barang.uuid)}
53               className="button is-small is-danger"
54             >
55               Delete
56             </button>
57           </td>
58         </tr>
59       )]}
60     </tbody>
61   </table>
62 </div>
63 )
64 }
65
66 export default Barang
67

```

