

## PEMROGRAMAN WEB BERBASIS SERVICES TUGAS AKHIR – AWONAPAKARYA

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

### PENJELASAN PROGRAM TA AWONAPAKARYA

Team AwonapaKarya membuat Website Berbasis Services dengan study Kasus Website **Website Inventaris Barang dan Jasa 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>

Tautan Server (Backend):

<http://52.199.149.14:2024/>

### 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.

```
// membuat fungsi verify user
export const verifyUser = async (req, res, next) => {
  if (!req.session.userId) {
    return res.status(401).json({ msg: "Mohon login ke akun anda!" });
  }
  // mencari user berdasarkan id
  const user = await User.findOne({
    where: {
      uuid: req.session.userId,
    },
  });
  if (!user) return res.status(404).json({ msg: "User tidak ditemukan" });
  // jika user ditemukan
  req.userId = user.id;
  req.role = user.role;
  next();
};
```

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

```
// membuat fungsi adminOnly
export const adminOnly = async (req, res, next) => {
  const user = await User.findOne({
    where: {
      uuid: req.session.userId,
    },
  });
  if (!user) return res.status(404).json({ msg: "User tidak ditemukan" });
  // jika user bukan admin
  if (user.role !== "admin")
    return res.status(403).json({ msg: "Akses Tidak Di izinkan" });
  next();
};
```

- Terdapat **verifyToken** yang digunakan untuk verifikasi token, jika terdapat token maka user bisa melakukan akses data ke REST API.

```
// membuat fungsi verify token dari jwt
export const verifyToken = (req, res, next) => {
  const authHeader = req.headers["authorization"];
  const token = authHeader && authHeader.split(" ")[1];
  if (token == null) return res.sendStatus(401);
  jwt.verify(token, process.env.ACCESS_TOKEN_SECRET, (err, decoded) => {
    if (err) {
      // console.log(err);
      return res.sendStatus(403);
    }
    req.email = decoded.email;
    next();
  });
};
```

- Terdapat **ketuaJurusan** yang berguna untuk membatasi akses user dengan role ketua jurusan hanya bisa melihat (Read) data saja.

```
// membuat fungsi jika ketua jurusan hanya bisa view data
export const ketuaJurusan = async (req, res, next) => {
  const user = await User.findOne({
    where: {
      uuid: req.session.userId,
    },
  });
  if (!user) return res.status(404).json({ msg: "User Tidak ditemukan" });
  // jika user adalah ketuaJurusan
  if (user.role === "ketuajurusan")
    return res.status(403).json({ msg: "Akses Tidak Di izinkan" });
  next();
};
```

- **Login (Auth.js)**

- 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”.

```
// function login
export const login = async (req, res) => {
  // mencari data user berdasarkan email
  const user = await Users.findOne({
    where: {
      email: req.body.email,
    },
  });
  if (!user) return res.status(404).json({ msg: "User tidak ditemukan" });

  // jika user ditemukan
  const match = await argon2.verify(user.password, req.body.password);

  // jika password tidak betul
  if (!match) return res.status(400).json({ msg: "Password Salah" });

  // jika password cocok
  req.session.userId = user.uuid;
  // get data user
  const uuid = user.uuid;
  const name = user.name;
  const email = user.email;
  const role = user.role;
```

- 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://52.199.149.14:2024/login
Send
200 OK
474 ms
438 B
Just Now

---

Multipart<sup>2</sup> Auth Query Headers<sup>1</sup> Docs

Add	Delete All	Toggle Description
email	wahyu@gmail.com	<input checked="" type="checkbox"/>
password	123456	<input checked="" type="checkbox"/>
email	operator@gmail.com	<input type="checkbox"/>
password	123456	<input type="checkbox"/>
email	budi@gmail.com	<input type="checkbox"/>
password	123456	<input type="checkbox"/>

Preview Headers<sup>12</sup> Cookies<sup>2</sup> Timeline

```

1 {
2   "status": true,
3   "message": "Berhasil Login",
4   "dataUser": {
5     "uuid": "2e1596e0-f5f4-4c77-b2d5-d6c178ea0727",
6     "name": "wahyu",
7     "email": "wahyu@gmail.com",
8     "role": "admin",
9     "accessToken":
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkIjoiaWwMxNTk2ZTAtZjYwMC00Yzc3LWJyZWZlcmMtY2ZlbnNpdjE1IiwibmFtZSI6ImdhH1IiwiaWF0IjO1OjI3YW5dUBnbWpbPbC5jb2IiLCJyb2N1IjoieWRtaW4iLCJpcyQiOiJlE2NzQ4MzI0DMsImV4cCI6MTY3NDgzmg4M30.5dAjqb3uRhRztOQoGlS0Yw2BpU32v-RZZixyEuLYIFXA"
10 }
11 }
```

- Login Ketua Jurusan

POST

<http://52.199.149.14:2024/login>

Send

200 OK

336 ms

478 B

Just Now

Multipart

Auth

Query

Headers

Docs

Add

Delete All

Toggle Description

email	budi@gmail.com			
password	123			
email	operator@gmail.com			
password	123456			
email	budi@gmail.com			
password	123456			

Preview

Headers

Cookies

Timeline

```

1 {
2   "status": true,
3   "message": "Berhasil Login",
4   "dataUser": {
5     "uuid": "d0445363-081e-4e64-9817-90927b97f7fd",
6     "name": "budi anduk keren",
7     "email": "budi@gmail.com",
8     "role": "ketuajurusan",
9     "accessToken":
10    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkIjoia0NDUzNjMtMDEg
11    xZS00ZTY0LTk4MTctOTAsImJpdOTdmN2kiIiwibmFtZSI6ImJ1ZGkgYW5kdWsga2Vy
12    ZW4iLCJlbWFPbGkiOiJ1ZG1AZ21haWwY29tIiwicm9sZSI6ImVhbnVydXNhbGki
13    LCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3
14    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
15    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
16    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
17    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
18    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
19    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
20    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
21    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
22    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
23    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
24    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
25    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
26    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
27    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
28    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
29    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
30    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
31    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
32    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
33    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
34    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
35    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
36    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
37    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
38    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
39    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
40    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
41    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
42    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
43    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
44    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
45    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
46    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
47    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
48    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
49    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
50    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
51    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
52    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
53    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
54    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
55    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
56    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
57    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
58    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
59    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
60    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
61    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
62    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
63    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
64    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
65    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
66    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
67    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
68    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZm9kZj4i
69    MDEgZm9kZj4iLCJ1dHdCI6MTY3MDEgZ
```

- Login Operator

**POST**
Just Now ▾


---

**http://52.199.149.14:2024/login**
Send ▾

200 OK
329 ms
440 B

---

Multipart<sup>2</sup> ▾
Auth ▾
Query
Headers<sup>1</sup>
Docs

Preview ▾
Headers<sup>12</sup>
Cookies<sup>2</sup>
Timeline

---

Add	Delete All	Toggle Description
email	udah@gmail.com	▾ ✓ 🗑️
password	12345678	▾ ✓ 🗑️
email	operator@gmail.com	▾ 🗑️
password	123456	▾ 🗑️
email	budi@gmail.com	▾ 🗑️
password	123456	▾ 🗑️

```

1 {
2   "status": true,
3   "message": "Berhasil Login",
4   "dataUser": {
5     "uuid": "41613ddd-2006-47d2-8472-e8b7e310ad45",
6     "name": "udah",
7     "email": "udah@gmail.com",
8     "role": "operator",
9     "accessToken":
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkIjoiNDUyMTNkZGQzMjAwbWl0bnQyLTg0NiIsInZiIjoiaWwibmFtZSI6InVkeWYngilCj1lbWpCbC16InVkeWYnAAZ21haWwvY29tIiwicm9sZSI6Im9wZXJhdG9yIiwiaWF0IjoxNjc0ODMyMzU2LjCLjEHaiojE2NzQ4MzY5NTZ9.Aa53iAoWfoObU-aQ3yHg4JDVmAucVQ8uDlQhFi2Ew"
10 }
11 }
```

- Check Session User

	←T→	sid	expires	data	createdAt	updatedAt
<input type="checkbox"/>	Edit Copy Delete	tNiNmub_HLqSVohNLAbtPsvFckuQkuJi	2023-01-27 22:41:52	{ "cookie": { "originalMaxAge": 1200000, "expires": "202...	2023-01-27 22:21:52	2023-01-27 22:21:52
<input type="checkbox"/>	Edit Copy Delete	VltwXwdklSg8haqTJkmp56LYJc-J6yf	2023-01-27 22:41:47	{ "cookie": { "originalMaxAge": 1200000, "expires": "202...	2023-01-27 22:21:47	2023-01-27 22:21:47

- Check Refresh Token

name	email	password	role	refresh_token
wahyu	wahyu@gmail.com	\$argon2id\$v=19\$m=65536,t=3,p=4\$SM4PaauHJ4L+hzRbYYJ...	admin	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkjoiM...
budi anduk keren	budi@gmail.com	\$argon2id\$v=19\$m=65536,t=3,p=4\$sJ19MzdKBrJfPqCxET1l...	ketuajurusan	NULL
udah	udah@gmail.com	\$argon2id\$v=19\$m=65536,t=3,p=4\$RQYmaw4hWweOS2xoPDF...	operator	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkjoiM...

- Index.js yang digunakan untuk manajemen session menggunakan

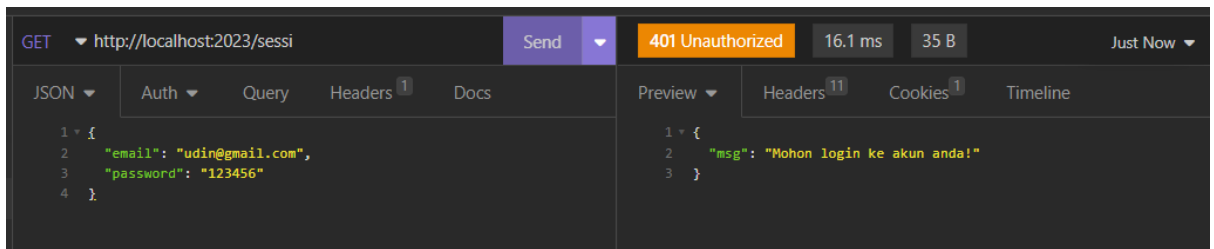
```
import session from "express-session";
import SequelizeStore from "connect-session-sequelize";
import cookieParser from "cookie-parser";
```

```
// 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
});

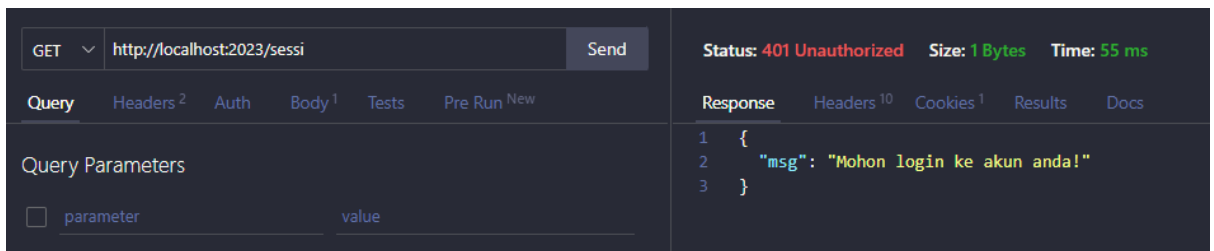
// membuat table session di database
// store.sync();

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

- Ketika Session sudah habis
  - Login User

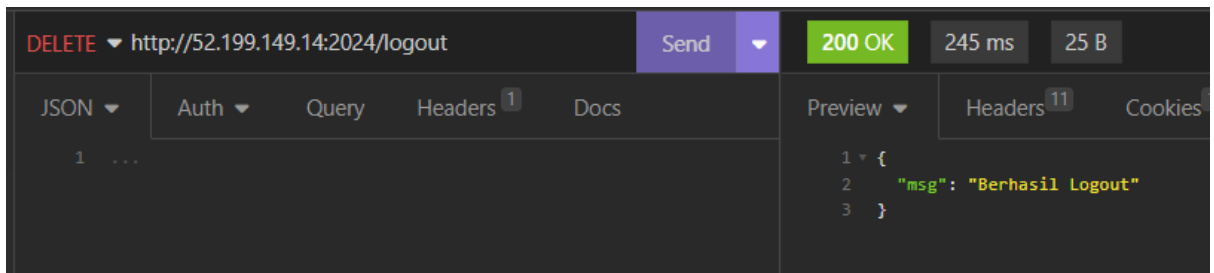


- Login Admin



- 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

- **Add User**

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

```
// membuat fungsi tambah user
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();

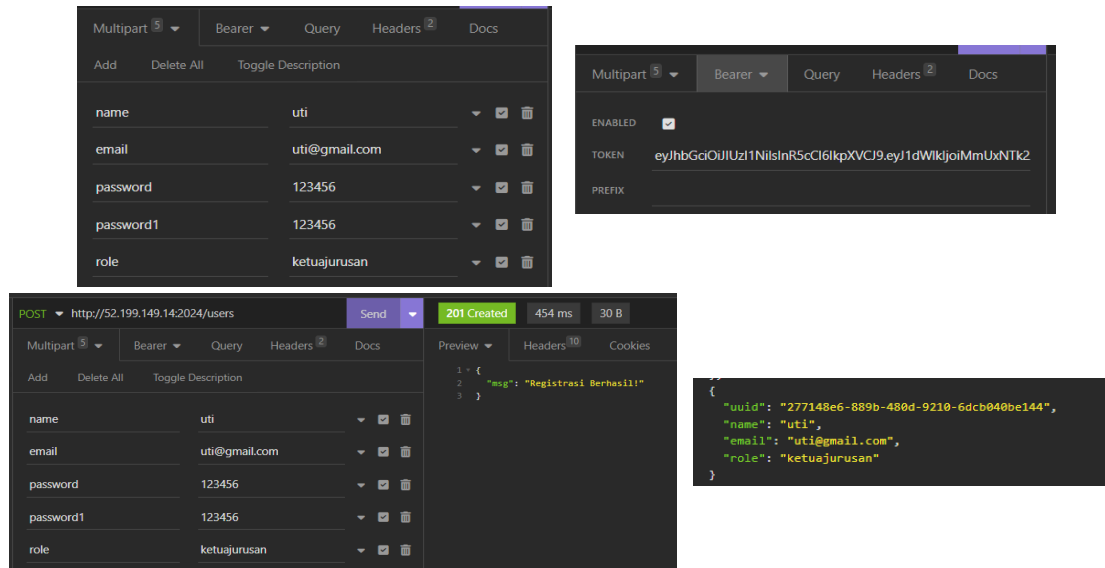
  // perulangan untuk cek apakah email ada yang sama
  for (let i = 0; i <= response.length; i++) {
    let dataEmail = response[i]?.email;
    if (dataEmail === email) {
      return res.status(400).json({ msg: "Registrasi Tidak Berhasil" });
    }
  }

  // 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 });
  }
};
```

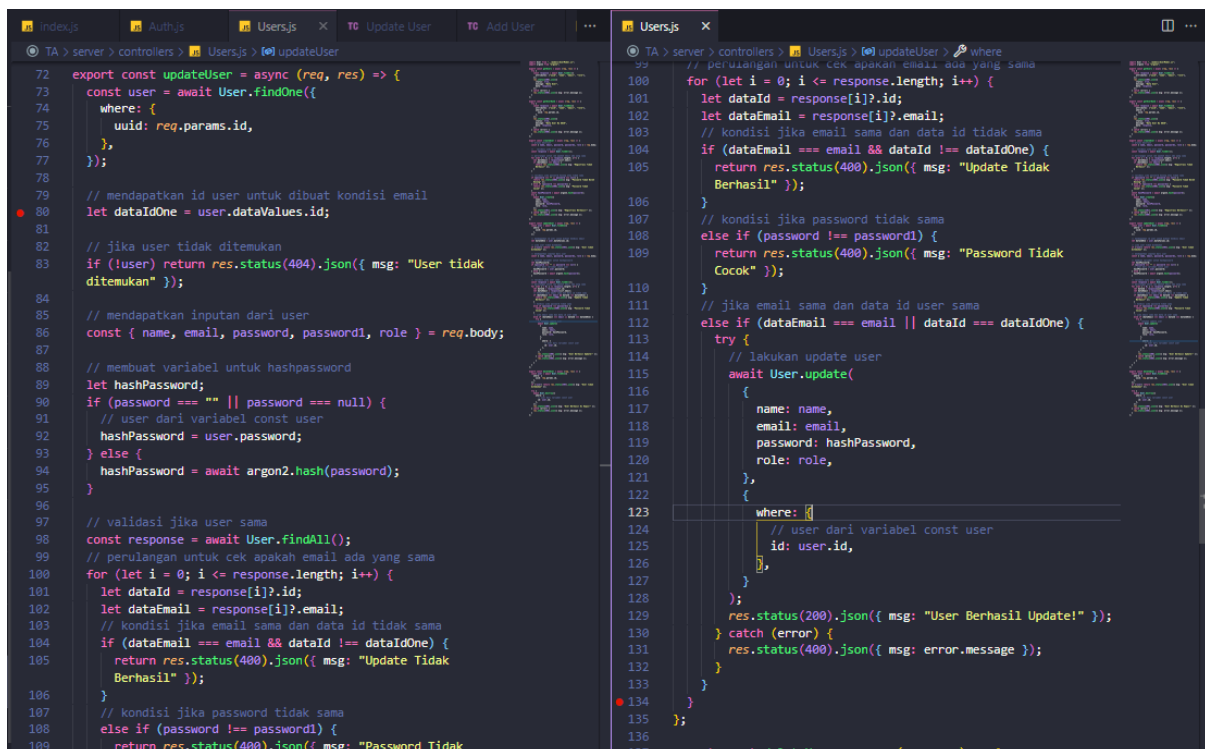
- membuat user dengan req body dari inputan
- Terdapat proses untuk pengecekan apakah ada email yang sama atau tidak.
- Jika password kosong atau tidak sama akan menampilkan pesan json.
- Jika berhasil maka password akan di hash menggunakan argon2.
- kemudian akan menjalankan proses create user.



- Untuk membuat user menggunakan Insomnia, kita akan menggunakan JWT dengan Autentikasi Bearer dan memasukkan Token yang didapatkan ketika user login.



## • Update User

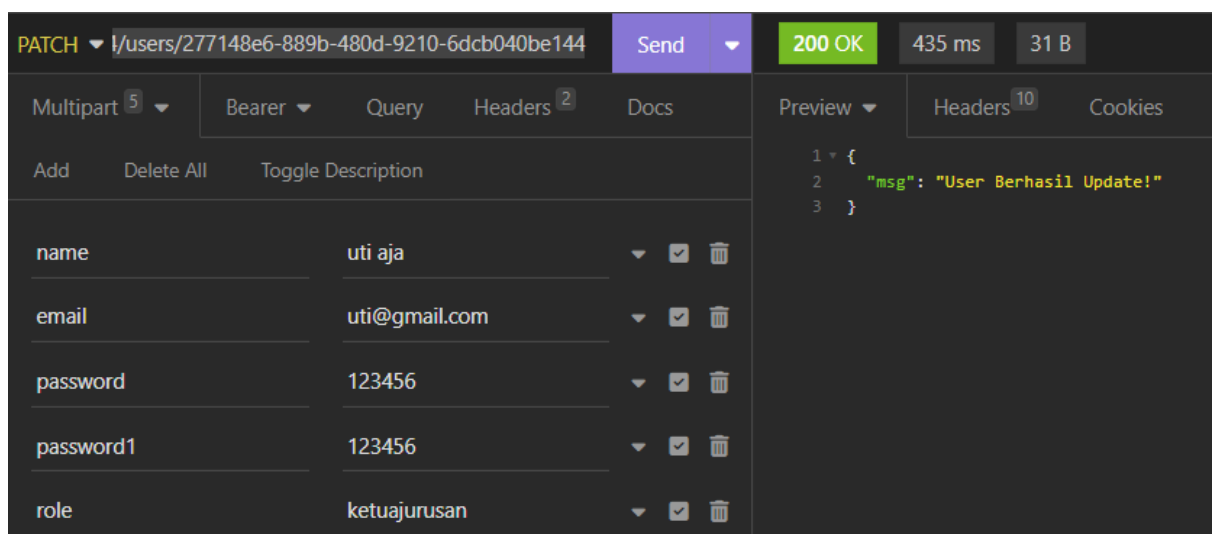


- Untuk update user menggunakan method Patch. dengan mendapatkan id user menggunakan `findOne` berdasarkan id user.
- Jika user tidak ditemukan tampilkan pesan json.
- Kemudian menerima input dari body

- 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.

Untuk update menggunakan uuid dari user dan tetap menggunakan autentikasi bearer token, seperti pada contoh berikut ini :

<http://52.199.149.14:2024/users/277148e6-889b-480d-9210-6dcb040be144>



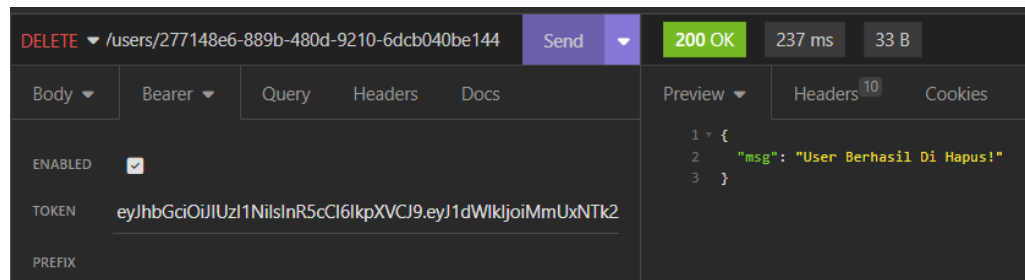
## • Delete User

```
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 });
  }
};
```

- 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.



## CRUD Pada Data Barang pada controller Barangs.js

### • Get Semua Data Barang

- Jika login sebagai admin, maka akan dapat melakukan proses CRUD seluruh data barang, yang di input oleh admin ataupun oleh user.
- Jika login sebagai operator, user dapat melihat data dari semua inputan user, kemudian dapat melakukan CRUD hanya pada data yang di input oleh operator, tidak ada akses untuk merubah data dari Admin.
- Jika login sebagai Ketua Jurusan, hanya dapat melihat seluruh data dari user saja. Tidak melakukan tambah data, perubahan, atau hapus data.

```
// membuat fungsi untuk getBarang
export const getBarangs = async (req, res) => {
  try {
    let response;
    // req.role berasal dari middleware ketika login
    if (req.role) {
      response = await Barang.findAll({
        attributes: [
          "uuid",
          "kd_brg",
          "nm_brg",
          "spek_brg",
          "jml_brg",
          "kondisi_brg",
          "tgl_buy_brg",
          "harga_brg",
          "image",
          "url",
        ],
        // memasukkan nama, email dari model User
        include: [
          {
            model: User,
            attributes: ["name", "email"],
          },
        ],
      });
    }
    res.status(200).json(response);
  } catch (error) {
    res.status(500).json({ msg: error.message });
  }
};
```

- Untuk mendapatkan seluruh data menggunakan `findAll` dari seluruh table barang
- Ambil seluruh field di dalam array `attributes` dan akan ditampilkan di dalam json.
- Termasuk memasukkan user yang melakukan input data tersebut

- Mendapatkan **Barang** berdasarkan id barang atau uuid

```
// membuat fungsi untuk getBarang berdasarkan id/uuid barang
export const getBarangById = async (req, res) => {
  try {
    const barang = await Barang.findOne({
      where: {
        uuid: req.params.id,
      },
    });
    // jika barang tidak ditemukan
    if (!barang) return res.status(404).json({ msg: "Data Tidak Ditemukan" });
  }
}
```

GET http://52.199.149.14:2024/jbarangs/ecbb98cc-e65f-4 Send 200 OK 249 ms 418 B Just Now

Body Bearer Query Headers Docs Preview Headers 10 Cookies Timeline

ENABLED ☒

TOKEN eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1dWlkIjoiMmUxNTk2

PREFIX

```
1 {
2   "uuid": "ecbb98cc-e65f-4aaa-a0d0-834e9d113a4c",
3   "kd_jbrg": "kd02",
4   "nm_jbrg": "Tinta Printer",
5   "spek_jbrg": "Black",
6   "jml_jbrg": 10,
7   "kondisi_jbrg": "Bagus",
8   "tgl_buy_jbrg": "2022-11-10T00:00:00.000Z",
9   "harga_jbrg": 600000,
10  "image": "1674835118403-6a4787bb5ead614031e93aff573eb893.png",
11  "url": "http://52.199.149.14:2024/images/jbarang/1674835118403-6a4787bb5ead614031e93aff573eb893.png",
12  "user": {
13    "name": "udah",
14    "email": "udah@gmail.com"
15  }
16 }
```

- **Create Barang**
  - Mendapatkan inputan dari form yang berisi data barang dari user (req body)
  - Kemudian melakukan pemeriksaan untuk kode barang, bahwa kode barang tidak boleh sama.

```
// membuat fungsi untuk tambah Barang
export const createBarang = async (req, res) => {
  // mendapatkan data input dari form
  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" });
    }
  }
}
```

- Tahap berikutnya yaitu melakukan pengecekan pada file yang di upload

```
// check jika file kosong
if (req.files === null)
  return res.status(400).json({ msg: "Tidak ada file yang di upload" });

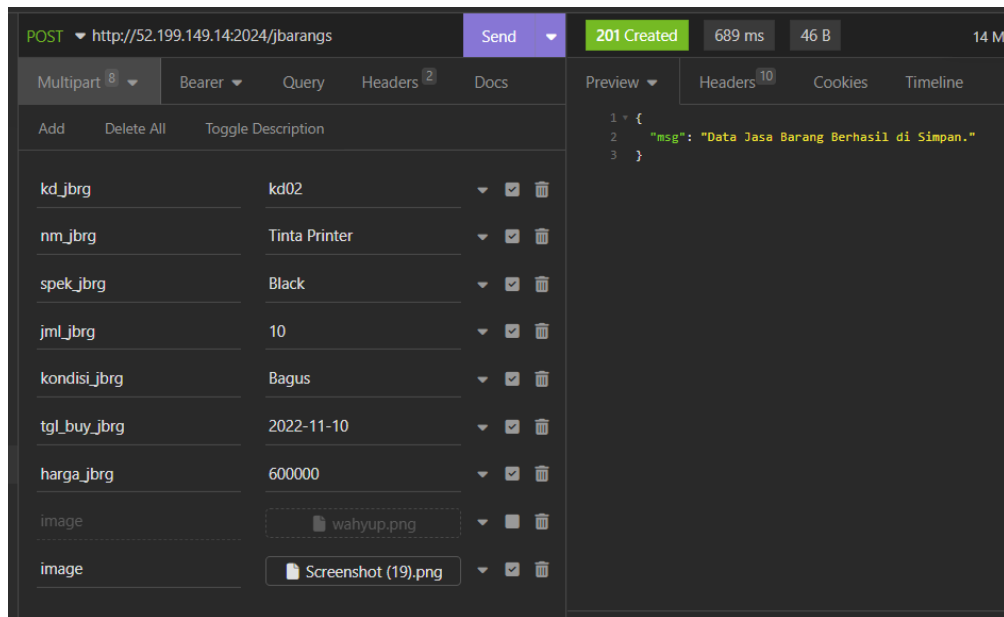
// membuat variabel untuk menampung data gambar
const file = req.files.image;
const fileSize = file.data.length;
const ext = path.extname(file.name);
const fileName = Date.now() + "-" + file.md5 + ext;
const url = `${req.protocol}://${req.get("host")}/images/barang/${fileName}`;
const allowedType = [".png", ".jpg", ".jpeg"];

// membuat kondisi jika variabel allowedType dan jika fileSize
if (!allowedType.includes(ext.toLowerCase()))
  return res.status(422).json({ msg: "Gambar tidak sesuai" });
if (fileSize > 5000000)
  return res.status(422).json({ msg: "Ukuran gambar harus dibawah 5MB" });
```

- Jika file kosong akan menampilkan json msg
- Jika file tidak sesuai untuk ekstensi akan menampilkan pesan json
- Jika file lebih dari 5MB akan menampilkan pesan json
- Jika kondisi terpenuhi maka akan menjalankan proses membuat data barang.
- Berdasarkan user yang menginput data maka dibutuhkan userId.

```
// jika kondisi benar maka akan melakukan proses simpan data
file.mv(`./public/images/barang/${fileName}`, async (err) => {
  if (err) return res.status(500).json({ msg: err.message });
  // 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,
      image: fileName,
      url: url,
      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

```
// membuat fungsi update barang
export const updateBarang = async (req, res) => {
  // 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" });
  }

  // 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;
}
```

- Untuk mendapatkan barang yang akan di update maka akan melakukan seleksi berdasarkan uuid data barang
- Jika uuid sesuai maka akan menampilkan data barang
- Jika tidak sesuai akan menampilkan json message

```
// kode untuk manajemen gambar
let fileName;
// check jika gambar null maka langsung update data selain image
if (req.files === null) {
  fileName = barang.image;
  // console.log(fileName);
}
// jika gambar tidak kosong melakukan proses update gambar
else {
  const file = req.files.image;
  const fileSize = file.data.length;
  const ext = path.extname(file.name);
  fileName = Date.now() + "-" + file.md5 + ext;
  // console.log(fileName);

  const allowedType = [".png", ".jpg", ".jpeg"];

  // membuat kondisi jika variabel allowedType dan jika fileSize
  if (!allowedType.includes(ext.toLowerCase()))
    return res.status(422).json({ msg: "Gambar tidak sesuai" });
  if (fileSize > 5000000)
    return res.status(422).json({ msg: "Ukuran gambar harus dibawah 5MB" });
}
```

- Jika file kosong akan menampilkan json msg
- Jika file tidak sesuai untuk ekstensi akan menampilkan pesan json
- Jika file lebih dari 5MB akan menampilkan pesan json

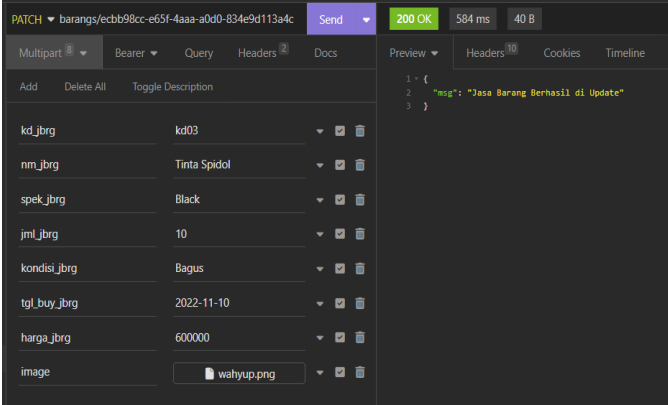
```
// hapus gambar
const filepath = `./public/images/barang/${barang.image}`;
fs.unlinkSync(filepath);

// pindahkan gambar ke folder public
file.mv(`./public/images/barang/${fileName}`, async (err) => {
  if (err) return res.status(500).json({ msg: err.message });
});
```

- Jika gambar akan di update maka gambar yang lama akan dihapus dan gambar yang baru akan dipindahkan ke folder public/images/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.

- Contoh proses update dari admin untuk data barang dari user udah (operator)

```
{
  "uuid": "ecbb98cc-e65f-4aaa-a0d0-834e9d113a4c",
  "kd_jbrg": "kd02",
  "nm_jbrg": "Tinta Printer",
  "spek_jbrg": "Black",
  "jml_jbrg": 10,
  "kondisi_jbrg": "Bagus",
  "tgl_buy_jbrg": "2022-11-10T00:00:00.000Z",
  "harga_jbrg": 600000,
  "image": "1674835118403-6a4787bb5ead614031e93aff573eb893.png",
  "url": "http://52.199.149.14:2024/images/jbarang/1674835118403-6a4787bb5ead614031e93aff573eb893.png",
  "user": {
    "name": "udah",
    "email": "udah@gmail.com"
  }
}
```



REST Client PATCH request to `barang/ecbb98cc-e65f-4aaa-a0d0-834e9d113a4c`. The request body is a JSON object with product details. The response is a 200 OK status with a message: "Jasa Barang Berhasil di Update".

```
{
  "uuid": "ecbb98cc-e65f-4aaa-a0d0-834e9d113a4c",
  "kd_jbrg": "kd03",
  "nm_jbrg": "Tinta Spidol",
  "spek_jbrg": "Black",
  "jml_jbrg": 10,
  "kondisi_jbrg": "Bagus",
  "tgl_buy_jbrg": "2022-11-10T00:00:00.000Z",
  "harga_jbrg": 600000,
  "image": "1674836373116-51f9f03ce67fb6721ddd95716ebf20c7.png",
  "url": "http://52.199.149.14:2024/images/jbarang/1674836373116-51f9f03ce67fb6721ddd95716ebf20c7.png",
  "user": {
    "name": "udah",
    "email": "udah@gmail.com"
  }
}
```

- 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.



```
// membuat fungsi delete barang
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" });
  }
}
```

- Mencari data berdasarkan uuid, jika tidak ditemukan akan menampilkan error berupa json.

```
// req.role berasal dari middleware ketika login
if (req.role === "admin") {
  // hapus gambar
  const filepath = `./public/images/barang/${barang.image}`;
  fs.unlinkSync(filepath);
  await Barang.destroy({
    where: {
      id: barang.id,
    },
  });
} else {
  // jika user id dan barang user id tidak sama
  if (req.userId !== barang.userId)
    return res.status(403).json({ msg: "Akses Tidak ditemukan" });
  // jika kondisi terpenuhi
  // hapus gambar
  const filepath = `./public/images/barang/${barang.image}`;
  fs.unlinkSync(filepath);
  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 });
}
```

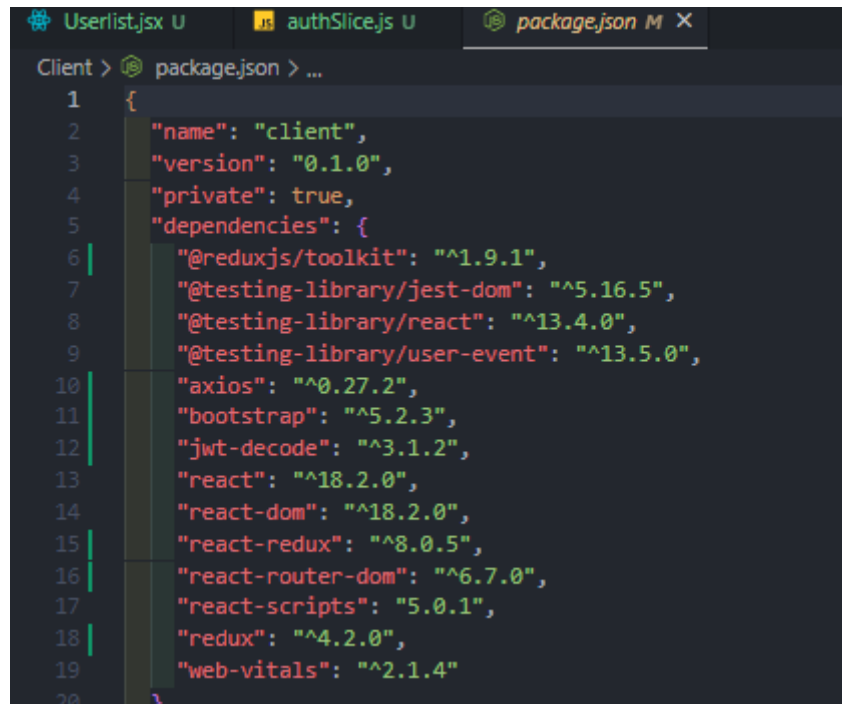
- Jika login sebagai admin maka dapat menghapus untuk data semua user, dan pada proses tersebut terjadi hapus gambar juga.

## PROSES CRUD PADA BARANG DAN JASA BARANG SAMA.

## Penjelasan Program Front End

### PENJELASAN PROGRAM FRONT END

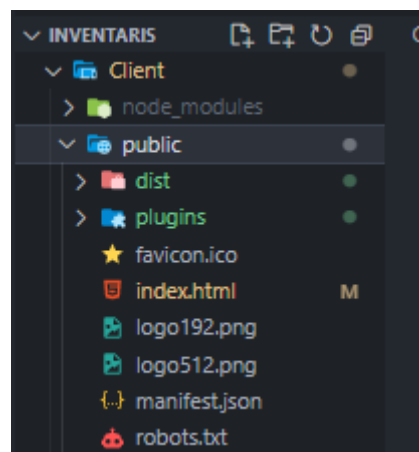
Program FrontEnd di kembangkan menggunakan NodeJs Framework ReactJS. Menggunakan Dependencies :



```

Client > package.json > ...
1  {
2    "name": "client",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@reduxjs/toolkit": "^1.9.1",
7      "@testing-library/jest-dom": "^5.16.5",
8      "@testing-library/react": "^13.4.0",
9      "@testing-library/user-event": "^13.5.0",
10     "axios": "^0.27.2",
11     "bootstrap": "^5.2.3",
12     "jwt-decode": "^3.1.2",
13     "react": "^18.2.0",
14     "react-dom": "^18.2.0",
15     "react-redux": "^8.0.5",
16     "react-router-dom": "^6.7.0",
17     "react-scripts": "5.0.1",
18     "redux": "^4.2.0",
19     "web-vitals": "^2.1.4"
20   }
  
```

Menggunakan Template Adminlte dengan memasukan Folder dist dan plugins



## Membuat Halaman Login dengan Framework Css Bootstrap :

**Halaman Login**

Silahkan Masukan Akun Anda

Mohon login ke akun anda!

✉

🔒

Masuk

```

Client > src > component > Login.jsx > Login
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 "../feature/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   };
  
```

PHP HTML Umum Jaringan Skripsi UI/UX Github Dicoding Indonesia Document Sharing... Slideshare Downloa... Scribd Downloader...
Other bookmarks

**Inventaris Sekolah**

adam

Widgets New

Data <

Belanja <

Charts <

Home Contact
⌵

### Data User

Home / Data User

Tambah Users

No	Nama	Email	Role	Action
1	Muhammad Ali Nasir	muhammadalii2020@gmail.com	Admin	<span style="background-color: #007bff; color: white; padding: 2px 5px; border-radius: 5px; cursor: pointer;">Edit</span> <span style="background-color: #dc3545; color: white; padding: 2px 5px; border-radius: 5px; cursor: pointer;">Hapus</span>

## Halaman Dashboard dan Tabel data Users

```

Client > src > pgs > Dashboard.js > ...
1 import { useDispatch, useSelector } from 'react-redux';
2
3
4 import { useNavigate } from "react-router-dom";
5 import { getMe } from "../feature/authSlice";
6 import Layout from '../Layout';
7 import Welcome from '../component/Welcome';
8
9
10 const Dashboard = () => {
11   const dispatch = useDispatch();
12   const navigate = useNavigate();
13   const { isError } = useSelector((state) => state.auth);
14
15   useEffect(() => {
16     dispatch(getMe());
17   }, [dispatch]);
18
19   useEffect(() => {
20     if (isError) {
21       navigate("/");
22     }
23   }, [isError, navigate]);
24   return (
25     <Layout>
26       <Welcome />
27     </Layout>
28   );
29 };
30
31 export default Dashboard;

```

Memasukan Layout dan Welcome supaya Header dan SideNav bisa tergabung jadi Satu

```

Client > src > pgs > Layout.js > default
1  import React from 'react'
2  //import axios from 'axios';
3  ///import jwt_decode from "jwt-decode";
4  //import { useNavigate } from 'react-router-dom';
5  import Header from '../component/Header'
6  import SideNav from '../component/SideNav'
7
8  const Layout = ({ children }) => {
9    return (
10     //Menggabungkan Beberapa Elemen
11     <React.Fragment>
12       <Header />
13       <SideNav />
14       <div className="column has-background-light">
15         <main>{children}</main>
16       </div>
17     </React.Fragment>
18   );
19 };
20
21 export default Layout;

```

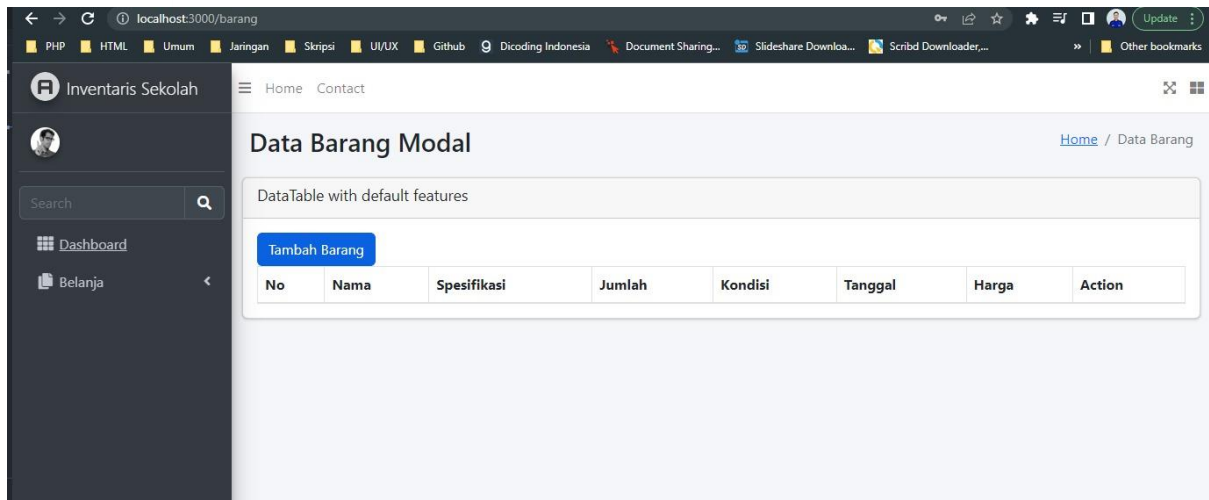
Kemudian di dalam File Layout kita masuk React.Fragment

```

Client > src > component > Welcome.jsx > ...
1  import React from 'react'
2  import { useSelector } from "react-redux";
3
4  const Welcome = () => {
5    const { user } = useSelector((state) => state.auth);
6    return (
7      <div>
8        <h1 className="title">Dashboard</h1>
9        <h2 className="subtitle">
10         Welcome Back <strong>{user && user.user.name}</strong>
11        </h2>
12      </div>
13    );
14  };
15
16  export default Welcome;

```

Dan di file Welcome jika User telah memasukan akun maka akan terbaca sesuai Akun yang dimasukan.



Halaman Barang Modal

```

Client > src > pgs > Barang.jsx > ...
1  import React, { useEffect } from "react";
2  import Layout from "../Layout";
3  import { useDispatch, useSelector } from "react-redux";
4  import { useNavigate } from "react-router-dom";
5  import { getMe } from "../feature/authSlice";
6  import Baranglist from "../component/Baranglist";
7
8  const Barang = () => {
9    const dispatch = useDispatch();
10   const navigate = useNavigate();
11   const { isError } = useSelector((state) => state.auth);
12
13   useEffect(() => {
14     dispatch(getMe());
15   }, [dispatch]);
16
17   useEffect(() => {
18     if (isError) {
19       navigate("/");
20     }
21   }, [isError, navigate]);
22   return (
23     <Layout>
24       <Baranglist />
25     </Layout>
26   );
27 };
28
29 export default Barang;

```

```

</Link>
<table id="example1" className="table table-bordered table-striped">
  <thead>
    <tr>
      <th>No</th>
      <th>Nama</th>
      <th>Spesifikasi</th>
      <th>Jumlah</th>
      <th>Kondisi</th>
      <th>Tanggal</th>
      <th>Harga</th>
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
    {barang.map((barangs, index) => (
      <tr key={barangs.uuid}>
        <td>{index + 1}</td>
        <td>{barangs.kd_brg}</td>
        <td>{barangs.nm_brg}</td>
        <td>{barangs.spek_brg}</td>
        <td>{barangs.jml_brg}</td>
        <td>{barangs.kondisi_brg}</td>
        <td>{barangs.tgl_buy_brg}</td>
        <td>{barangs.harga_brg}</td>
        <td>{barangs.user.name}</td>
        <td>
          <Link
            to={` /barang/edit/${barang.uuid}`}
            className="button is-small is-info"
          >
            Edit
          </Link>
          <button
            onClick={() => deleteBarang(barang.uuid)}
            className="button is-small is-danger"
          >

```

```
Client > src > component > Baranglist.jsx > Baranglist
1  import React, { useState, useEffect } from "react";
2  import { Link } from "react-router-dom";
3  import axios from "axios";
4
5  const Baranglist = () => {
6    const [barang, setBarang] = useState([]);
7
8    useEffect(() => {
9      getBarang();
10     }, []);
11
12     //memanggil data barang
13     const getBarang = async () => {
14       const response = await axios.get("http://localhost:2023/barang");
15       setBarang(response.data);
16     };
17
18     //membuat fungsi hapus barang
19     const deleteBarang = async (barangId) => {
20       await axios.delete(`http://localhost:2023/barang/${barangId}`);
21       getBarang();
22     };
23 }
```