

# **LAPORAN UJIAN AKHIR SEMESTER**

## **PENGEMBANGAN BACKEND**



**Dosen Pengampu :**

M. Taufiq M.Kom

**Disusun Oleh :**

Putri Sahma Dewi / 3124101293

**PRODI D3 MANAJEMEN INFORMATIKA**

**STIKOM PGRI BANYUWANGI**

**TAHUN 2025**

## 1. npm init -y

```
Microsoft Windows [Version 10.0.22631.5624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lenovo\OneDrive\Documents\Semester 2\Pengembangan Backend\UASBackend>npm init -y
Wrote to C:\Users\lenovo\OneDrive\Documents\Semester 2\Pengembangan Backend\UASBackend\package.json:

{
  "name": "uasbackend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Penjelasan : Perintah ini digunakan untuk inialisasi sebuah proyek Node.js dengan membuat file package.json secara otomatis.

## 2. npm install

```
C:\Users\lenovo\OneDrive\Documents\Semester 2\Pengembangan Backend\UASBackend>npm install

up to date, audited 1 package in 2s

found 0 vulnerabilities
```

Penjelasan : Perintah ini digunakan untuk menginstal semua library atau package yang tercantum dalam file package.json.

## 3. npm install bcrypt bcryptjs cookie-parser dotenv express express-validator helmet jsonwebtoken mysql2 sequelize

```
C:\Users\lenovo\OneDrive\Documents\Semester 2\Pengembangan Backend\UASBackend>npm install bcrypt bcryptjs cookie-parser dotenv express express-validator helmet jsonwebtoken mysql2 sequelize

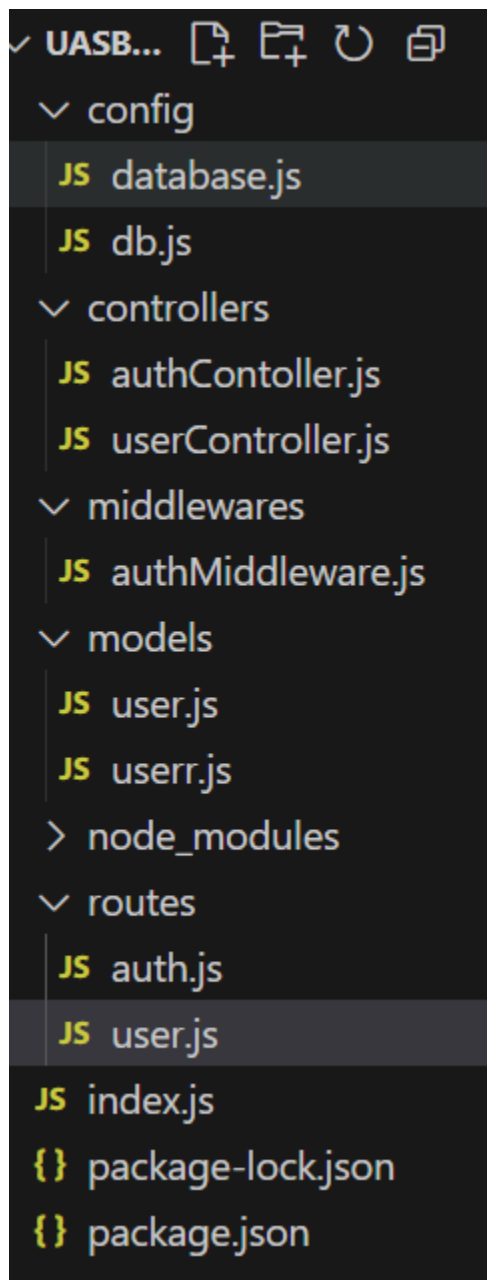
added 118 packages, and audited 119 packages in 34s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Perintah ini digunakan untuk menginstal package yang dibutuhkan dalam pengembangan backend

4. Membuat folder dan isi masing – masing folder



5. Membuat isi file **database.js** pada folder config

```
const { Sequelize } = require('sequelize');
// require('dotenv').config();

const sequelize = new Sequelize('userstopupstreaming', 'root', '', {
  host: 'localhost',
  dialect: 'mysql',
  port: 3306,
});

module.exports = { sequelize};
```

6. Membuat isi file **db.js** pada folder config

```
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('userstopupstreaming', 'root', '', {
  host: 'localhost',
  dialect: 'mysql',
});

module.exports = { sequelize};
```

## 7. Membuat isi file **authController.js** pada folder controllers

```
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const { validationResult } = require('express-validator');
const User = require('../models/user');

Windsurf: Refactor | Explain | Generate JSDoc | ✕
exports.register = async (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) return res.status(400).
    json({ errors: errors.array() });

  const { email, password, role } = req.body;
  const hashedPassword = await bcrypt.hash(password, 10);
  try {
    const user = await User.
      create({ email, password: hashedPassword, role: role || 'user' });
    res.status(201).json({ message: 'User created', user });
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
};

Windsurf: Refactor | Explain | Generate JSDoc | ✕
exports.login = async (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) return res.status(400).
    json({ errors: errors.array() });

  const { email, password } = req.body;
  const user = await User.findOne({ where: { email } });
  if (!user || !await bcrypt.compare(password, user.password)) {
    return res.status(401).json({ message: 'Invalid credentials' });
  }

  const token = jwt.sign({ id: user.id, role: user.role },
    'SECRET_KEY', { expiresIn: '1h' });
  res.cookie('token', token, { httpOnly: true });
  json({ message: 'Login successful' });
};

Windsurf: Refactor | Explain | Generate JSDoc | ✕
exports.logout = (req, res) => {
  res.clearCookie('token').json({ message: 'Logged out' });
};
```

8. Membuat isi file **UserController.js** pada folder controller

```
const express = require('express');
const router = express.Router();
const User = require('../models/user');

// GET semua user
router.get('/', async (req, res) => {
  const user = await User.findAll();
  res.json(user);
});

// GET user by ID
router.get('/:id', async (req, res) => {
  const { id } = req.params;
  const user = await User.findById(id);
  if (user) {
    res.json(user);
  } else {
    res.status(404).json({ message: 'User not found' });
  }
});

// POST tambah user
router.post('/', async (req, res) => {
  const { email, password, role } = req.body;
  try {
    const newUser = await User.create({ email, password, role });
    res.status(201).json(newUser);
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});

// PUT update User
router.put('/:id', async (req, res) => {
  const { id } = req.params;
  const { email, password, role } = req.body;
  try {
    const update = await User.update(
      { email, password, role },
      { where: { id } }
    );
    res.json({ message: 'User updated', update });
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});

// DELETE User
router.delete('/:id', async (req, res) => {
  const { id } = req.params;
  try {
    await User.destroy({ where: { id } });
    res.json({ message: 'User deleted' });
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});

module.exports = router;
```

9. Membuat isi file **authMiddleware.js** pada folder middlewares

```
const jwt = require('jsonwebtoken');  
  
Windsurf: Refactor | Explain | Generate JSDoc | X  
exports.protect = (req, res, next) => {  
  const token = req.cookies.token;  
  if (!token) return res.status(401).json({ message: 'Unauthorized' });  
  
  try {  
    const user = jwt.verify(token, 'SECRET_KEY');  
    req.user = user;  
    next();  
  } catch (err) {  
    res.status(401).json({ message: 'Invalid token' });  
  }  
};  
  
Windsurf: Refactor | Explain | Generate JSDoc | X  
exports.adminOnly = (req, res, next) => {  
  if (req.user.role !== 'admin') return res.status(403).  
    json({ message: 'Forbidden' });  
  next();  
};
```

10. Membuat isi file **user.js** pada folder models

```
const { DataTypes } = require('sequelize');  
const { sequelize } = require('../config/db');  
  
const User = sequelize.define('User', {  
  email: {  
    type: DataTypes.STRING,  
    unique: true,  
    allowNull: false,  
  },  
  password: {  
    type: DataTypes.STRING,  
    allowNull: false,  
  },  
  role: {  
    type: DataTypes.STRING,  
    defaultValue: 'user',  
  },  
});  
module.exports = User;
```

## 11. Membuat isi file **userr.js** pada folder models

```
const { DataTypes } = require('sequelize');
const { sequelize } = require('../config/database');

const User = sequelize.define('User', {
  email: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false,
  },
  password: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  role: {
    type: DataTypes.STRING,
    defaultValue: 'user',
  },
}, {
  tableName: 'users',
  timestamps: false,
});
module.exports = User;
```

## 12. Membuat isi file **auth.js** pada folder routes

```
const express = require('express');
const { body } = require('express-validator');
const router = express.Router();
const { register, login, logout } =
  require('../controllers/authController');
const { protect, adminOnly } =
  require('../middlewares/authMiddleware');

router.post('/register',
  body('email').isEmail(),
  body('password').isLength({ min: 6 }),
  register
);

router.post('/login',
  body('email').isEmail(),
  body('password').notEmpty(),
  login
);

router.post('/logout', logout);

router.get('/admin', protect, adminOnly, (req, res) => {
  res.json({ message: 'Welcome Admin!' });
});

module.exports = router;
```



13. Membuat isi file **user.js** pada folder routes

```
const express = require("express");
const router = express.Router();
const userController = require("../controllers/userController");

router.get("/", userController.get);
router.get("/:id", userController.get);
router.post("/", userController.post);
router.put("/:id", userController.put);
router.delete("/:id", userController.delete);

module.exports = router;
```

14. Membuat isi file **index.js**

```
const express = require("express");
const cookieParser = require("cookie-parser");
const helmet = require("helmet");
const { sequelize } = require("../config/db");
const authRoutes = require("../routes/auth");
const userRouter = require("../controllers/userController");

const app = express();
app.use(express.json());
app.use(cookieParser());
app.use(helmet());

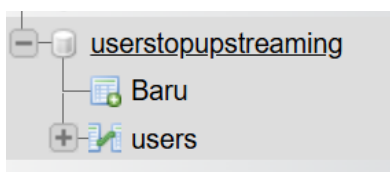
app.use("/api/user", userRouter);

app.use("/api/auth", authRoutes);
sequelize.sync().then(() => console.log("Database synced"));
app.listen(3000, () => console.log("Server running on http://localhost:3000));
```

15. Membuat database di **MySQL**

```
1 CREATE DATABASE usersTopUpStreaming;
```

Tampilan :









## 16. Node index.js

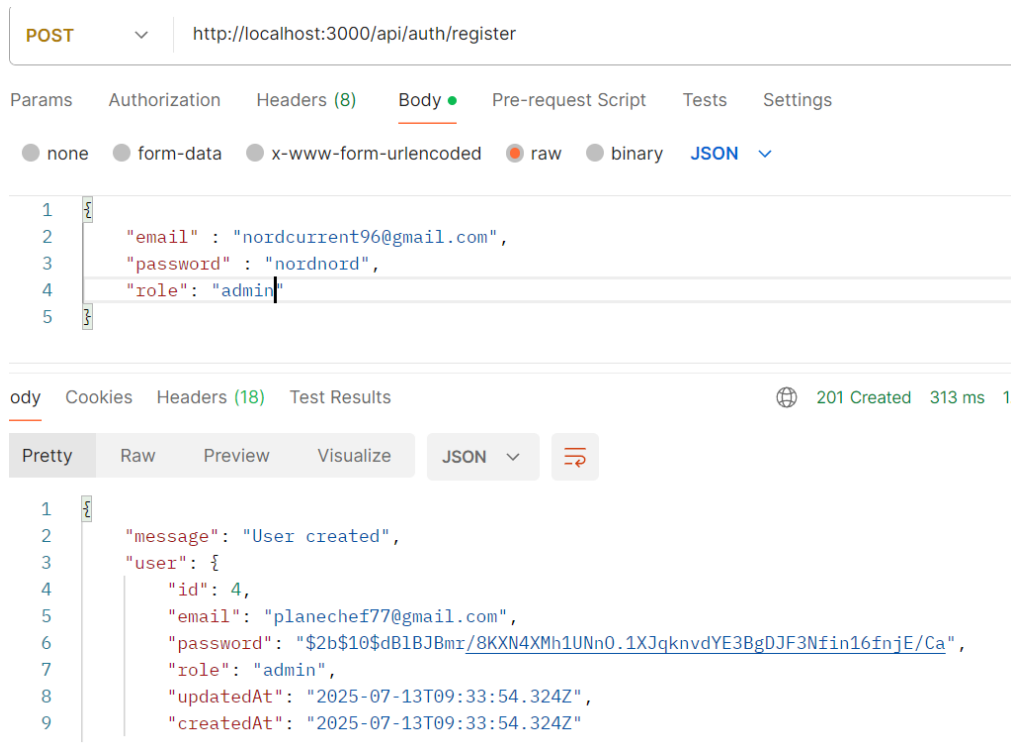
```
C:\Users\lenovo\OneDrive\Documents\Semester 2\Pengembangan Backend\16bcak>node index.js
Server running on http://localhost:3000
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'Users' AND TABLE_SCHEMA = 'userstopupstreaming'
Executing (default): CREATE TABLE IF NOT EXISTS `Users` (`id` INTEGER NOT NULL auto_increment , `email` VARCHAR(255) NOT NULL UNIQUE, `password` VARCHAR(255) NOT NULL, `role` VARCHAR(255) DEFAULT 'user', `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `Users`
Database synced
```

Penjelasan : Perintah ini digunakan untuk menjalankan file utama (index.js) dari server Express. File ini berfungsi sebagai entry point yang menginisialisasi server, middleware, dan koneksi ke database. Setelah dijalankan, server akan aktif dan siap menerima permintaan dari client.

## 17. Tampilan MySQL setelah terhubung dengan project

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
<input type="checkbox"/> 1	<b>id</b> 🔑	int(11)			Tidak	Tidak ada		AUTO_INCREMENT	 Ubah  Hapus Lainnya
<input type="checkbox"/> 2	<b>email</b> 📧	varchar(255)	utf8mb4_general_ci		Tidak	Tidak ada			 Ubah  Hapus Lainnya
<input type="checkbox"/> 3	<b>password</b>	varchar(255)	utf8mb4_general_ci		Tidak	Tidak ada			 Ubah  Hapus Lainnya
<input type="checkbox"/> 4	<b>role</b>	varchar(255)	utf8mb4_general_ci		Ya	user			 Ubah  Hapus Lainnya
<input type="checkbox"/> 5	<b>createdAt</b>	datetime			Tidak	Tidak ada			 Ubah  Hapus Lainnya
<input type="checkbox"/> 6	<b>updatedAt</b>	datetime			Tidak	Tidak ada			 Ubah  Hapus Lainnya

## 18. Ujicoba di postman : Registrasi admin



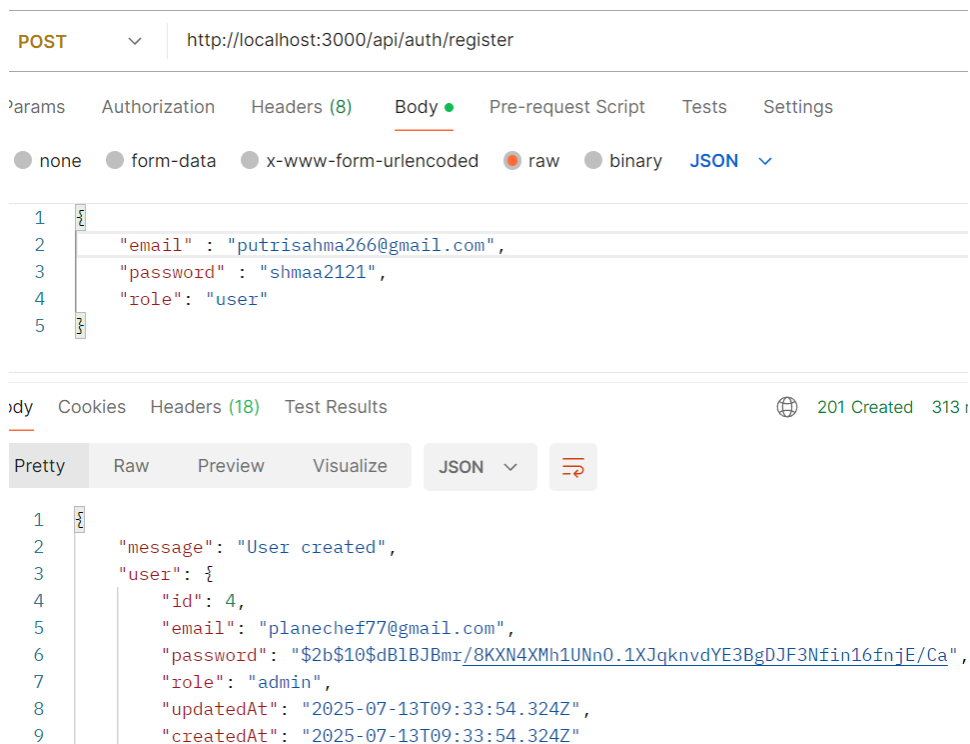
The image shows the Postman interface for a POST request to `http://localhost:3000/api/auth/register`. The request body is a JSON object with the following fields:

```
{
  "email": "nordcurrent96@gmail.com",
  "password": "nordnord",
  "role": "admin"
}
```

The response is displayed in the bottom pane, showing a 201 status code and the following JSON body:

```
{
  "message": "User created",
  "user": {
    "id": 4,
    "email": "planechef77@gmail.com",
    "password": "$2b$10$dB1BJBmr/8KXN4XMh1UNn0.1XJqknvdYE3BgDJF3Nfin16fnjE/Ca",
    "role": "admin",
    "updatedAt": "2025-07-13T09:33:54.324Z",
    "createdAt": "2025-07-13T09:33:54.324Z"
  }
}
```

## 19. Ujicoba di postman : Registrasi user



The image shows the Postman interface for a POST request to `http://localhost:3000/api/auth/register`. The request body is a JSON object with the following fields:

```
{
  "email": "putrisahma266@gmail.com",
  "password": "shmaa2121",
  "role": "user"
}
```

The response is displayed in the bottom pane, showing a 201 status code and the following JSON body:

```
{
  "message": "User created",
  "user": {
    "id": 4,
    "email": "planechef77@gmail.com",
    "password": "$2b$10$dB1BJBmr/8KXN4XMh1UNn0.1XJqknvdYE3BgDJF3Nfin16fnjE/Ca",
    "role": "admin",
    "updatedAt": "2025-07-13T09:33:54.324Z",
    "createdAt": "2025-07-13T09:33:54.324Z"
  }
}
```

## 20. Tampilan di MySQL

☐ Tampilkan semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini | Sort by key: Tidak ada

Extra options

				id	email	password	role	createdAt	updatedAt
<input type="checkbox"/>	Ubah	Salin	Hapus	1	putrisahma266@gmail.com	\$2b\$10\$pWsodkwdq1Y47OL6zU4KZO4Pk3TO/sppaNUc0iiywEh...	user	2025-07-13 09:31:35	2025-07-13 09:31:35
<input type="checkbox"/>	Ubah	Salin	Hapus	2	amandamelda17@gmail.com	\$2b\$10\$JnC1J9srD.uTzipOnuTJE.7xPt1C7Mm7H4XuprtU/aw...	user	2025-07-13 09:32:29	2025-07-13 09:32:29
<input type="checkbox"/>	Ubah	Salin	Hapus	3	nordcurrent96@gmail.com	\$2b\$10\$h4frfmJmY5A1G3RtA8wUgOxooOX9KEZYTZ.YIZW2k5q...	admin	2025-07-13 09:33:22	2025-07-13 09:33:22
<input type="checkbox"/>	Ubah	Salin	Hapus	4	planechef77@gmail.com	\$2b\$10\$dBiBJBmr/8KXN4XMh1UNnO.1XJqknvdYE3BgDJF3Nfi...	admin	2025-07-13 09:33:54	2025-07-13 09:33:54

☐ Pilih Semua | Dengan pilihan: Ubah Salin Hapus Ekspor

☐ Tampilkan semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini | Sort by key: Tidak ada

## 21. Ujicoba di postman : Admin login "Login juga berlaku bagi user"

**POST** ▼ `http://localhost:3000/api/auth/login`

Params Authorization Headers (9) **Body** ● Pre

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐

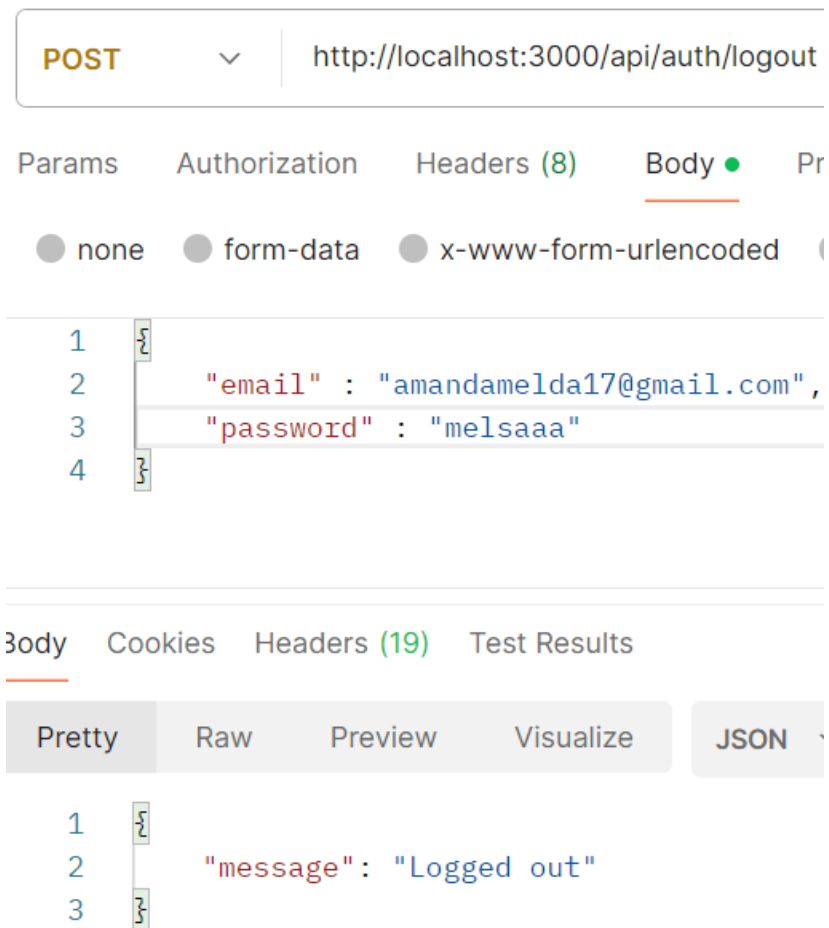
```
1 {
2   "email" : "planechef77@gmail.com",
3   "password" : "airplanechef"
4 }
```

body Cookies (1) Headers (19) Test Results

Pretty Raw Preview Visualize **JSON** ▼

```
1 {
2   "message": "Login successful"
3 }
```

## 22. Ujicoba di postman : Logout



### 23. Welcome admin

GET

⌵

http://localhost:3000/api/auth/admin

Params

Authorization

Headers (9)

Body ●

Pre

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒

1

{

2

"email" : "planechef77@gmail.com",

3

"password" : "airplanechef"

4

}

Body

Cookies (1)

Headers (18)

Test Results

Pretty

Raw

Preview

Visualize

JSON ⌵

1

{

2

"message": "Welcome Admin!"

3

}

## 24. Proses CRUD : GET

GET ⌵ http://localhost:3000/api/user Send ⌵

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON ⌵ Beautify

```
1 {
2   "email" : "putrisahma266@gmail.com",
3   "password" : "shmaa2121",
4   "role": "user"
5 }
```

Body Cookies Headers (18) Test Results 🌐 200 OK 226 ms 1.32 KB Save Response ⌵

Pretty Raw Preview Visualize JSON ⌵ 🔍

```
1 [
2   {
3     "id": 1,
4     "email": "putrisahma266@gmail.com",
5     "password": "$2b$10$pWsdq1Y470L6zU4KZ04Pk3T0/sppaNuc0iyywEhBL1XeIo8Cu",
6     "role": "user"
7   },
8   {
9     "id": 2,
```

## 25. Proses CRUD : POST

POST ⌵ http://localhost:3000/api/user Send ⌵

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON ⌵ Beautify

```
1 {
2   "email" : "sahmadewi12@gmail.com",
3   "password" : "samasama",
4   "role": "admin"
5 }
```

Body Cookies Headers (18) Test Results 🌐 201 Created 122 ms 957 B Save Response ⌵

Pretty Raw Preview Visualize JSON ⌵ 🔍

```
1 {
2   "id": 5,
3   "email": "sahmadewi12@gmail.com",
4   "password": "samasama",
5   "role": "admin"
6 }
```

## 26. Proses CRUD : PUT

PUT ▼ http://localhost:3000/api/user/5 Send ▼

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary **JSON** ▼ Beautify

```
1 {
2   "email": "sahmadewi12@gmail.com",
3   "password": "salmonaraaaa",
4   "role": "admin"
5 }
```

ody Cookies Headers (18) Test Results 200 OK 63 ms 914 B Save Response ▼

**Pretty** Raw Preview Visualize **JSON** ▼ 🔍

```
1 {
2   "message": "User updated",
3   "update": [
4     1
5   ]
6 }
```

## 27. Proses CRUD : DELETE

DELETE ▼ http://localhost:3000/api/user/4 Send ▼

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary **JSON** ▼ Beautify

```
1 {
2   "email": "sahmadewi12@gmail.com",
3   "password": "salmonaraaaa",
4   "role": "admin"
5 }
```

Body Cookies Headers (18) Test Results 200 OK 21 ms 1.45 KB Save Response ▼

**Pretty** Raw Preview Visualize **JSON** ▼ 🔍

```
18 {
19   "role": "admin"
20 },
21 {
22   "id": 4,
23   "email": "planechef77@gmail.com",
24   "password": "$2b$10$dB1BJBmr/8KXN4XMh1UNn0.1XJqknvdYE3BgDJF3Nfin16fnjE/Ca",
25   "role": "admin"
26 }
```



Tampilan setelah di delete :

The image shows two screenshots of a REST client interface. The top screenshot displays a DELETE request to `http://localhost:3000/api/user/4`. The request body is a JSON object: `{ "email": "sahmadewi12@gmail.com", "password": "salmonaraaaa", "role": "admin" }`. The response is a 200 OK status with a body of `{ "message": "User deleted" }`. The bottom screenshot displays a GET request to `http://localhost:3000/api/user`. The response is a 200 OK status with a body containing two user objects: `{ "id": 3, "email": "nordcurrent96@gmail.com", "password": "$2b$10$h4frfmJmY5A1G3RtA8wUg0xoo0X9KEZYTZ.YIZW2k5qhpZz15oDw.", "role": "admin" }, { "id": 5, "email": "sahmadewi12@gmail.com", "password": "salmonaraaaa" }`.

**DELETE Request**

Method: DELETE  
URL: `http://localhost:3000/api/user/4`  
Body: `{ "email": "sahmadewi12@gmail.com", "password": "salmonaraaaa", "role": "admin" }`

**DELETE Response**

Status: 200 OK  
Time: 137 ms  
Size: 901 B  
Body: `{ "message": "User deleted" }`

**GET Request**

Method: GET  
URL: `http://localhost:3000/api/user`

**GET Response**

Status: 200 OK  
Time: 569 ms  
Size: 1.32 KB  
Body: `{ "id": 3, "email": "nordcurrent96@gmail.com", "password": "$2b$10$h4frfmJmY5A1G3RtA8wUg0xoo0X9KEZYTZ.YIZW2k5qhpZz15oDw.", "role": "admin" }, { "id": 5, "email": "sahmadewi12@gmail.com", "password": "salmonaraaaa" }`

Link github : <https://github.com/putrisyk888/pengembangan-backend/tree/main/Pertemuan%2016>