МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

Про виконання лабораторної роботи №1
З дисципліни “Безпека програмного забезпечення”
На тему “Основні методи авторизації ”

Виконали:
Студенти групи ІП-15
Мєшков А. І.

Перевірила:
пос. Соколовський В. В.

Київ 2024

ЛАБОРАТОРНА РОБОТА №1

Завдання: Викачати репозиторій з лекціями
https://github.com/Kreolwolf1/auth_examples

Запустити кожен з 3 аплікейшенів та зробити скріншити запитів до серверу.
Для отримання додаткового балу: модифікувати token_auth аплікейшен змінивши токен на JWT.

ХІД РОБОТИ

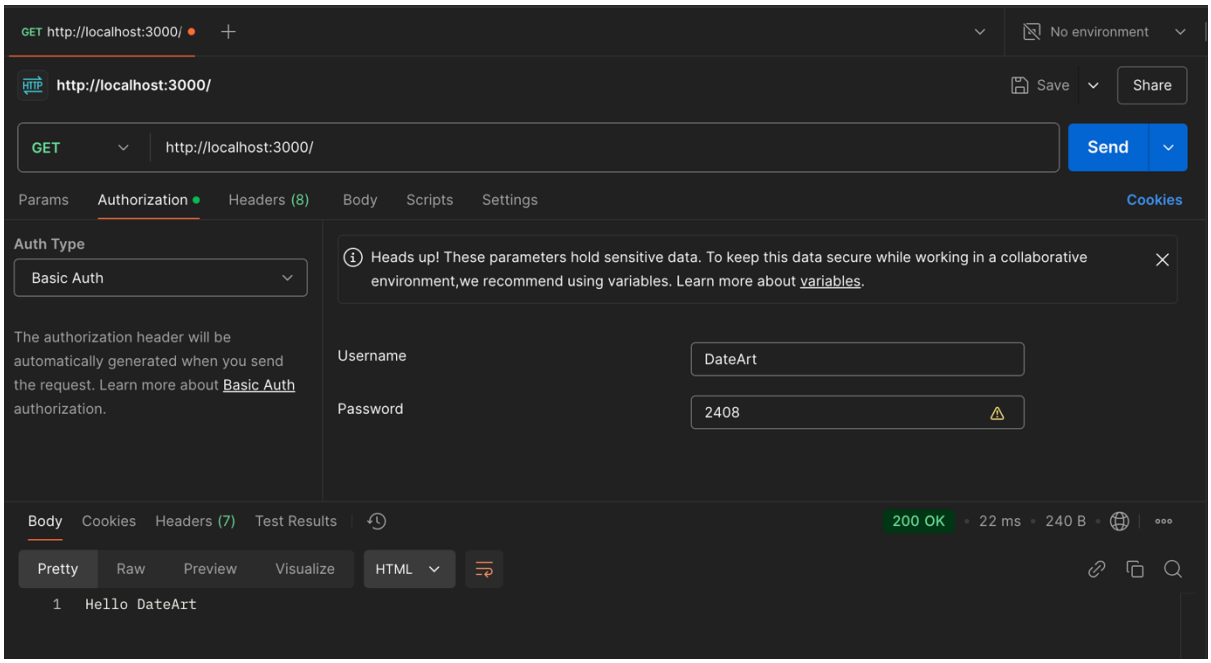
1. Basic Auth

Index.js

```
const express = require('express')
const app = express()
const port = 3000

app.use((req, res, next) => {
    console.log('\n==============================================================\n');

    const authorizationHeader = req.get('Authorization');
    console.log('authorizationHeader', authorizationHeader);

    if (!authorizationHeader) {
        res.setHeader('WWW-Authenticate', 'Basic realm="Ukraine"');
        res.status(401);
        res.send('Unauthorized');
        return;
    }

    const authorizationBase64Part = authorizationHeader.split(' ')[1];

    const decodedAuthorizationHeader = Buffer.from(authorizationBase64Part,
'base64').toString('utf-8');
    console.log('decodedAuthorizationHeader', decodedAuthorizationHeader);

    const login = decodedAuthorizationHeader.split(':')[0];
    const password = decodedAuthorizationHeader.split(':')[1];
    console.log('Login/Password', login, password);

    if (login == 'DateArt' && password == '2408') {
        req.login = login;
        return next();
    }

    res.setHeader('WWW-Authenticate', 'Basic realm="Ukraine"');
    res.status(401);
    res.send('Unauthorized');
});

app.get('/', (req, res) => {
    res.send(`Hello ${req.login}`);
})

app.listen(port, () => {
    console.log(`Example app listening on port ${port}`)
```

```
})
```

Запити:



2. Forms auth

Index.html

```
const express = require('express')
const app = express()
const port = 3000

app.use((req, res, next) => {
    console.log('\n==============================================================\n');

    const authorizationHeader = req.get('Authorization');
    console.log('authorizationHeader', authorizationHeader);

    if (!authorizationHeader) {
        res.setHeader('WWW-Authenticate', 'Basic realm="Ukraine"');
        res.status(401);
        res.send('Unauthorized');
        return;
    }

    const authorizationBase64Part = authorizationHeader.split(' ')[1];

    const decodedAuthorizationHeader = Buffer.from(authorizationBase64Part,
'base64').toString('utf-8');
```

```
    console.log('decodedAuthorizationHeader', decodedAuthorizationHeader);

    const login = decodedAuthorizationHeader.split(':')[0];
    const password = decodedAuthorizationHeader.split(':')[1];
    console.log('Login/Password', login, password);

    if (login == 'DateArt' && password == '2408') {
        req.login = login;
        return next();
    }

    res.setHeader('WWW-Authenticate', 'Basic realm="Ukraine"');
    res.status(401);
    res.send('Unauthorized');
});

app.get('/', (req, res) => {
    res.send(`Hello ${req.login}`);
})

app.listen(port, () => {
    console.log(`Example app listening on port ${port}`)
})
```

Index.js

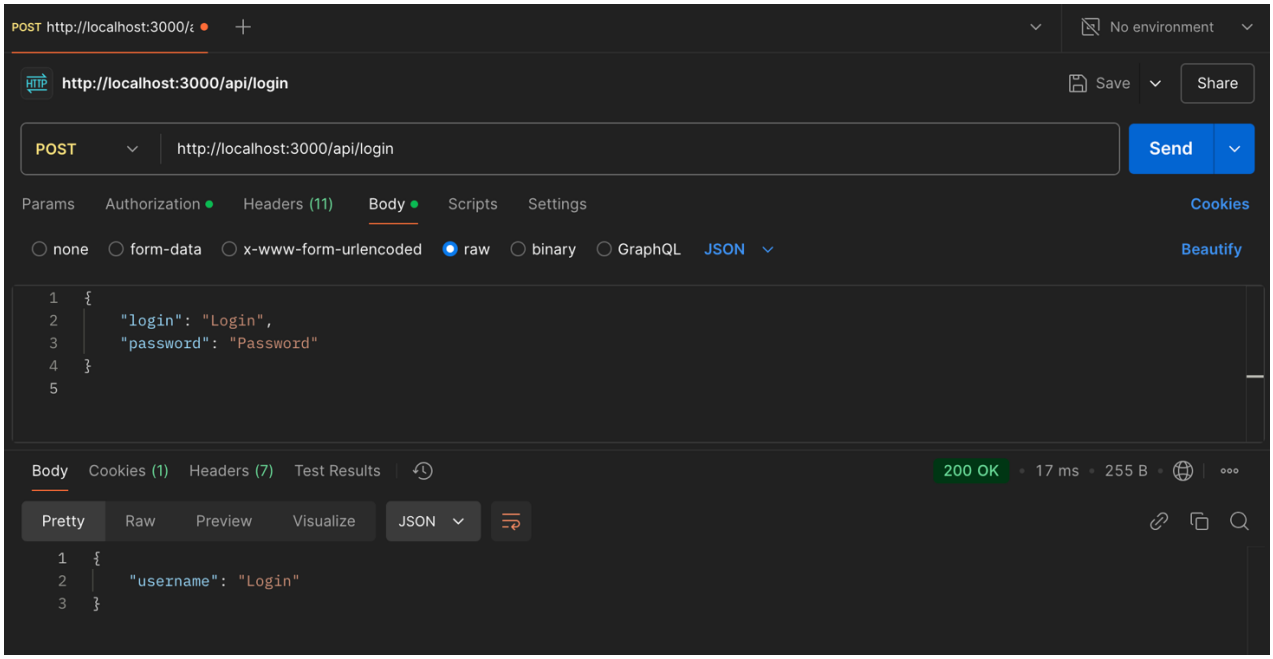
```
const uuid = require('uuid');
const express = require('express');
const cookieParser = require('cookie-parser');
const onFinished = require('on-finished');
const bodyParser = require('body-parser');
const path = require('path');
const port = 3000;
const fs = require('fs');

const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(cookieParser());

const SESSION_KEY = 'session';

class Session {
    #sessions = {}

    constructor() {
        try {
            this.#sessions = fs.readFileSync('./sessions.json', 'utf8');
            this.#sessions = JSON.parse(this.#sessions.trim());
```

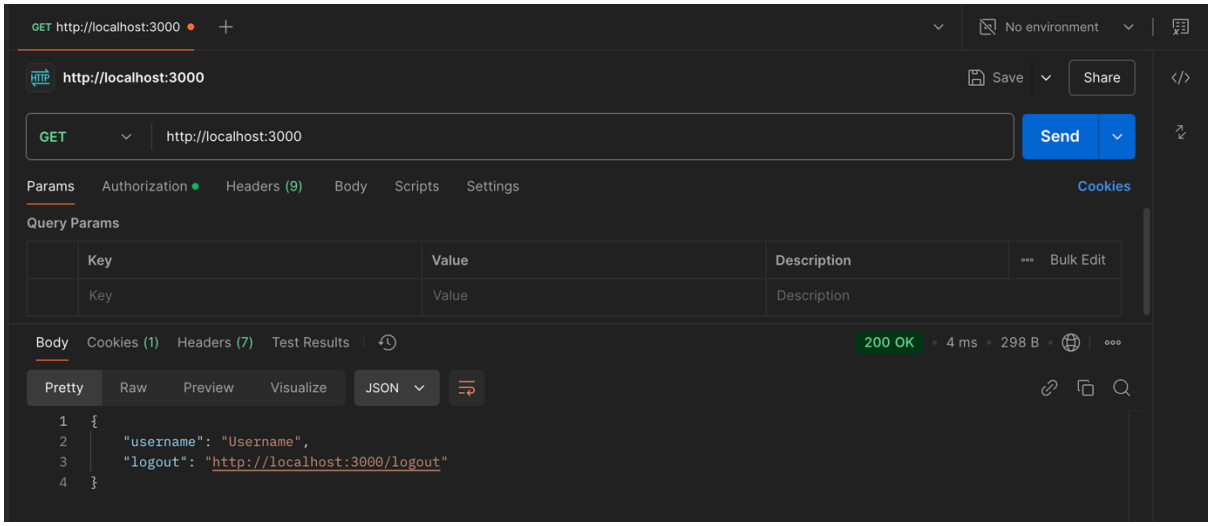
```
            console.log(this.#sessions);
        } catch(e) {
            this.#sessions = {};
        }
    }

    #storeSessions() {
        fs.writeFileSync('./sessions.json', JSON.stringify(this.#sessions), 'utf-
8');
    }

    set(key, value) {
        if (!value) {
            value = {};
        }
        this.#sessions[key] = value;
        this.#storeSessions();
    }

    get(key) {
        return this.#sessions[key];
    }

    init(res) {
        const sessionId = uuid.v4();
        res.set('Set-Cookie', `${SESSION_KEY}=${sessionId}; HttpOnly`);
        this.set(sessionId);

        return sessionId;
    }

    destroy(req, res) {
        const sessionId = req.sessionId;
        delete this.#sessions[sessionId];
        this.#storeSessions();
        res.set('Set-Cookie', `${SESSION_KEY}=; HttpOnly`);
    }
}

const sessions = new Session();

app.use((req, res, next) => {
    let currentSession = {};
    let sessionId;

    if (req.cookies[SESSION_KEY]) {
        sessionId = req.cookies[SESSION_KEY];
        currentSession = sessions.get(sessionId);
        if (!currentSession) {
            currentSession = {};
            sessionId = sessions.init(res);
```

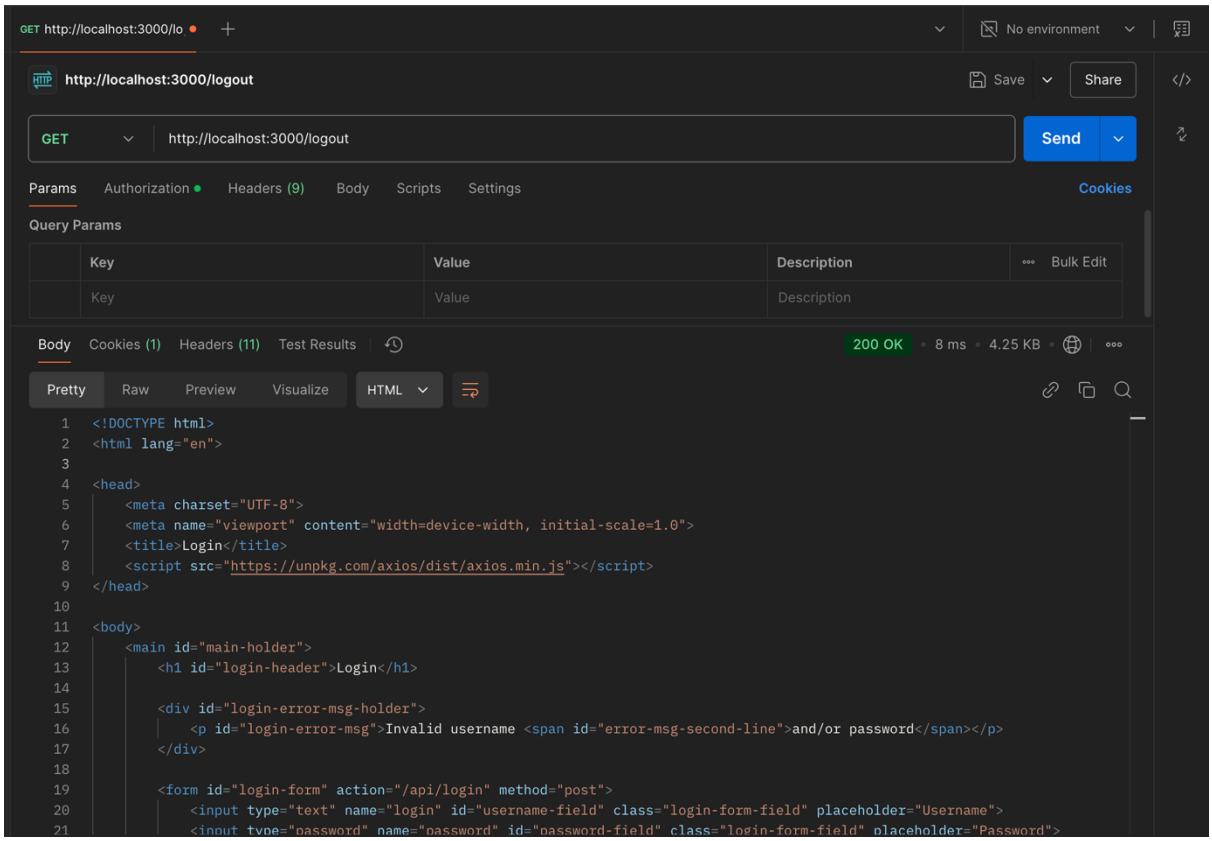
```
        }
    } else {
        sessionId = sessions.init(res);
    }

    req.session = currentSession;
    req.sessionId = sessionId;

    onFinished(req, () => {
        const currentSession = req.session;
        const sessionId = req.sessionId;
        sessions.set(sessionId, currentSession);
    });

    next();
});

app.get('/', (req, res) => {
    console.log(req.session);

    if (req.session.username) {
        return res.json({
            username: req.session.username,
            logout: 'http://localhost:3000/logout'
        })
    }
    res.sendFile(path.join(__dirname+'/index.html'));
})

app.get('/logout', (req, res) => {
    sessions.destroy(req, res);
    res.redirect('/');
});

const users = [
    {
        login: 'Login',
        password: 'Password',
        username: 'Username',
    },
    {
        login: 'Login1',
        password: 'Password1',
        username: 'Username1',
    }
]

app.post('/api/login', (req, res) => {
    const { login, password } = req.body;
    console.log(login, password, "-------------")
```

```
    const user = users.find((user) => {
        if (user.login == login && user.password == password) {
            return true;
        }
        return false
    });

    if (user) {
        req.session.username = user.username;
        req.session.login = user.login;

        res.json({ username: login });
    }

    res.status(401).send();
});

app.listen(port, () => {
    console.log(`Example app listening on port ${port}`)
})
```

Запити:







3. Token auth with JWT

Index.html

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Login</title>
        <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    </head>
```

```
    <body>
        <main id="main-holder">
            <a href="/logout" id="logout">Logout</a>

            <h1 id="login-header">Login</h1>

            <div id="login-error-msg-holder">
                <p id="login-error-msg">Invalid username <span id="error-msg-
second-line">and/or password</span></p>
            </div>

            <form id="login-form" action="/api/login" method="post">
                <input type="text" name="login" id="username-field" class="login-
form-field" placeholder="Username">
                <input type="password" name="password" id="password-field"
class="login-form-field" placeholder="Password">
                <input type="submit" value="Login" id="login-form-submit">
            </form>

        </main>
    </body>


    <style>
        html {
            height: 100%;
        }

        body {
            height: 100%;
            margin: 0;
            font-family: Arial, Helvetica, sans-serif;
            display: grid;
            justify-items: center;
            align-items: center;
            background-color: #3a3a3a;
        }

        #logout {
            opacity: 0;
        }

        #main-holder {
            width: 50%;
            height: 70%;
            display: grid;
            justify-items: center;
            align-items: center;
            background-color: white;
            border-radius: 7px;
```

```
        box-shadow: 0px 0px 5px 2px black;
    }

    #login-error-msg-holder {
        width: 100%;
        height: 100%;
        display: grid;
        justify-items: center;
        align-items: center;
    }

    #login-error-msg {
        width: 23%;
        text-align: center;
        margin: 0;
        padding: 5px;
        font-size: 12px;
        font-weight: bold;
        color: #8a0000;
        border: 1px solid #8a0000;
        background-color: #e58f8f;
        opacity: 0;
    }

    #error-msg-second-line {
        display: block;
    }

    #login-form {
        align-self: flex-start;
        display: grid;
        justify-items: center;
        align-items: center;
    }

    .login-form-field::placeholder {
        color: #3a3a3a;
    }

    .login-form-field {
        border: none;
        border-bottom: 1px solid #3a3a3a;
        margin-bottom: 10px;
        border-radius: 3px;
        outline: none;
        padding: 0px 0px 5px 5px;
    }

    #login-form-submit {
        width: 100%;
        padding: 7px;
```

```
            border: none;
            border-radius: 5px;
            color: white;
            font-weight: bold;
            background-color: #3a3a3a;
            cursor: pointer;
            outline: none;
        }
    </style>

    <script>
        const session = sessionStorage.getItem('session');

        let token;

        try {
            token = JSON.parse(session).token;
        } catch(e) {}

        if (token) {
            axios.get('/', {
                headers: {
                    Authorization: token
                }
            }).then((response) => {
                const { username } = response.data;

                if (username) {
                    const mainHolder = document.getElementById("main-holder");
                    const loginHeader = document.getElementById("login-header");

                    loginForm.remove();
                    loginErrorMsg.remove();
                    loginHeader.remove();

                    mainHolder.append(`Hello ${username}`);
                    logoutLink.style.opacity = 1;
                }
            });
        }


        const loginForm = document.getElementById("login-form");
        const loginButton = document.getElementById("login-form-submit");
        const loginErrorMsg = document.getElementById("login-error-msg");
        const logoutLink =  document.getElementById("logout");

        logoutLink.addEventListener("click", (e) => {
            e.preventDefault();
            sessionStorage.removeItem('session');
            location.reload();
```

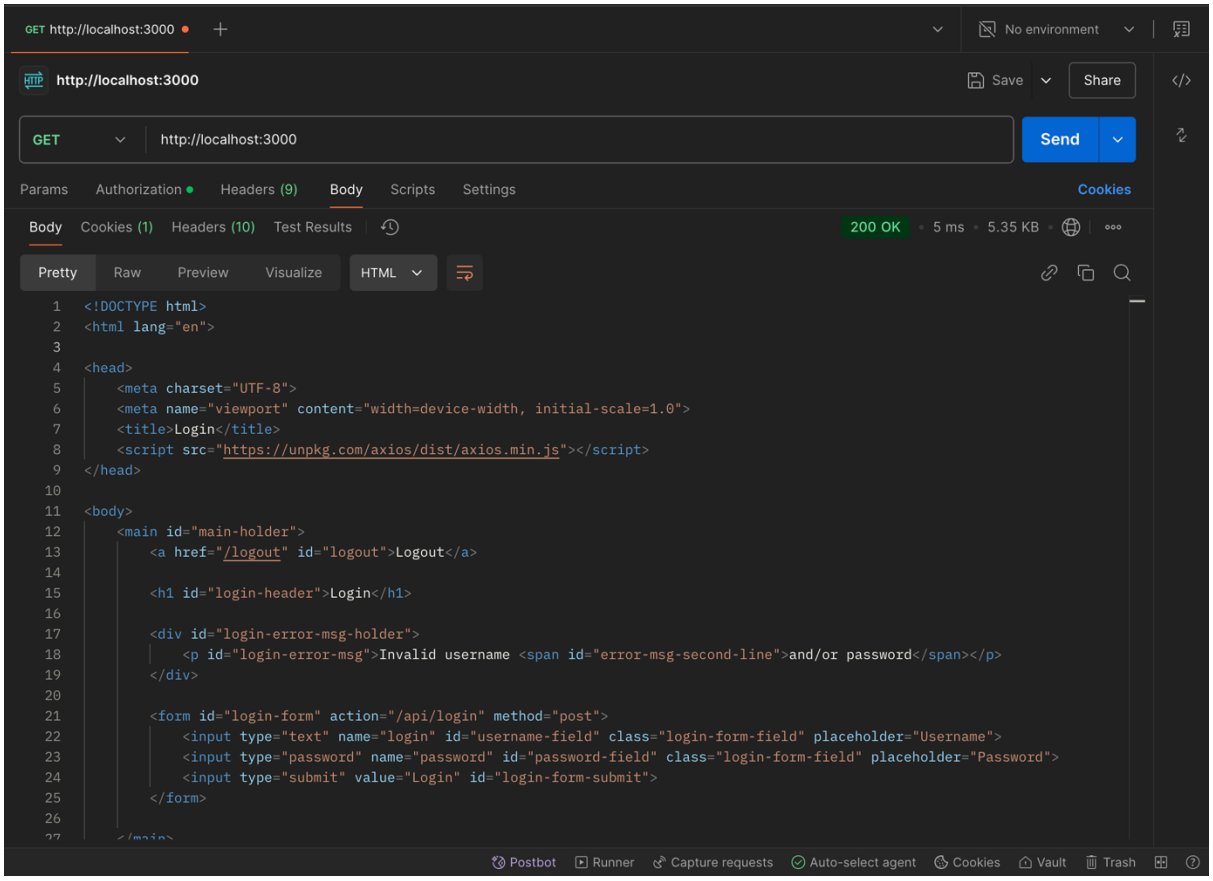
```
        });

        loginButton.addEventListener("click", (e) => {
            e.preventDefault();
            const login = loginForm.login.value;
            const password = loginForm.password.value;


            axios({
                method: 'post',
                url: '/api/login',
                data: {
                    login,
                    password
                }
            }).then((response) => {
                const { username } =  response.data;
                sessionStorage.setItem('session', JSON.stringify(response.data));
                location.reload();
            }).catch((response) => {
                loginErrorMsg.style.opacity = 1;
            });
        })
    </script>
</html>
```

Index.js

```
const express = require('express');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');

const app = express();
const port = 3000;

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

const JWT_SECRET = '23hbhb23ybdundjwnsjdnk,a.c,i2mbu3n2m,lsjdhn8fi2,qpc2tu3w';

const users = [
    { login: 'Login', password: 'Password', username: 'Username' },
    { login: 'Login1', password: 'Password1', username: 'Username1' }
];

const authenticateJWT = (req, res, next) => {
    const authHeader = req.get('Authorization');
    if (authHeader) {
        const token = authHeader.split(' ')[1];
        jwt.verify(token, JWT_SECRET, (err, user) => {
            if (err) {
```

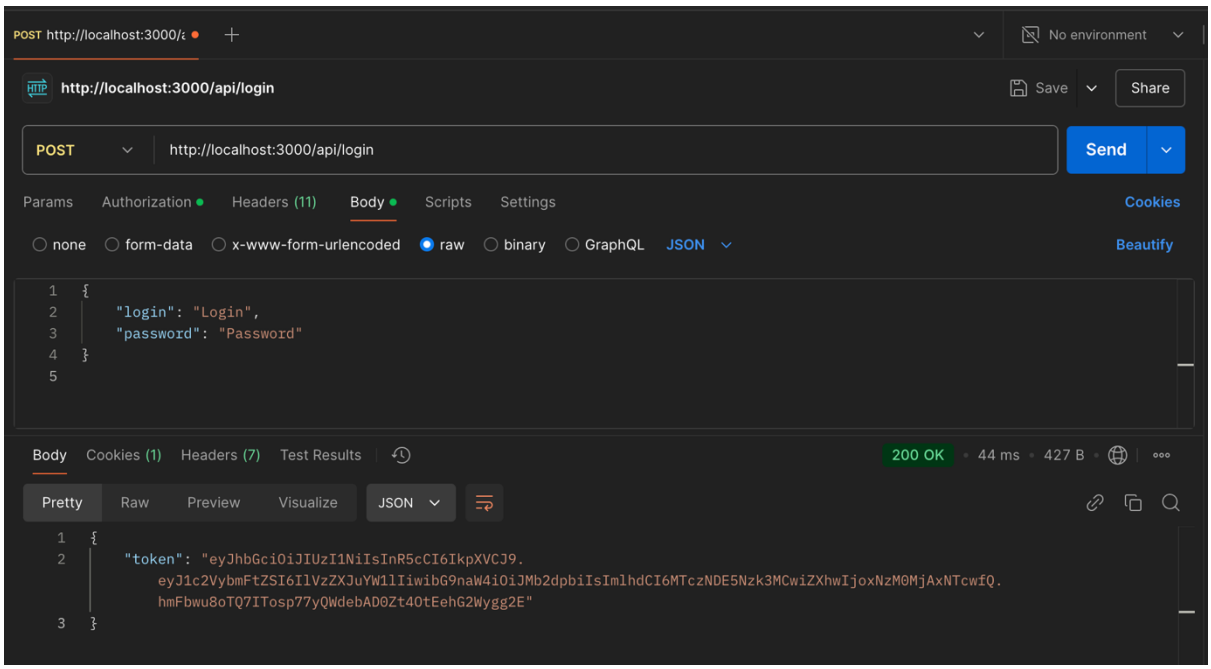
```
                return res.status(403).json({ error: 'Invalid token' });
            }
            req.user = user;
            next();
        });
    } else {
        res.status(401).json({ error: 'Authorization header missing' });
    }
};


app.get('/', authenticateJWT, (req, res) => {
    res.json({
        message: `Hello, ${req.user.username}!`,
        logout: 'http://localhost:3000/logout'
    });
});


app.post('/api/login', (req, res) => {
    const { login, password } = req.body;

    const user = users.find(u => u.login === login && u.password === password);
    if (user) {
        const token = jwt.sign(
            { username: user.username, login: user.login },
            JWT_SECRET,
            { expiresIn: '1h' }
        );

        res.json({ token });
    } else {
        res.status(401).json({ error: 'Invalid login or password' });
    }
});

app.get('/logout', (req, res) => {
    res.json({ message: 'To logout, simply delete your token on the client side.'
});
});


app.listen(port, () => {
    console.log(`Example app listening on http://localhost:${port}`);
});
```

Запити:


GET http://localhost:3000
No environment
http://localhost:3000
Save
Share
GET
http://localhost:3000
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
Body
Cookies (1)
Headers (10)
Test Results
200 OK
5 ms
5.35 KB
Pretty
Raw
Preview
Visualize
HTML
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login</title>
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
</head>
<body>
<main id="main-holder">
Logout
<h1 id="login-header">Login</h1>

Invalid username and/or password

<form id="login-form" action="/api/login" method="post">
<input type="text" name="login" id="username-field" class="login-form-field" placeholder="Username">
<input type="password" name="password" id="password-field" class="login-form-field" placeholder="Password">
<input type="submit" value="Login" id="login-form-submit">
</form>
Postbot
Runner
Capture requests
Auto-select agent
Cookies
Vault
Trash


POST http://localhost:3000/
No environment
http://localhost:3000/api/login
Save
Share
POST
http://localhost:3000/api/login
Send
Params
Authorization
Headers (11)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"login": "Login",
"password": "Password"
}
Body
Cookies (1)
Headers (7)
Test Results
200 OK
44 ms
427 B
Pretty
Raw
Preview
Visualize
JSON
{
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJ1c2VybmFtZSI6IlVzZXJuYW1lIiwibG9naW4iOiJMb2dpbiIsImlhdCI6MTczNDE5Nzk3MCwiZXhwIjoxNzM0MjAxNTcwfQ.
hmFbwu8oTQ7ITosp77yQWdebAD0Zt4OtEehG2Wygg2E"
}

ВИСНОВКИ

В результаті виконання лабораторної роботи було досліджено основні методи авторизації:

Basic Auth — простий метод аутентифікації, де логін та пароль передаються закодованими в заголовку Authorization. Основний недолік цього методу — логін і пароль можна легко перехопити без використання шифрування.

Forms Auth — реалізація авторизації з використанням сесій та cookies. Користувач проходить аутентифікацію через форму, після чого сервер зберігає інформацію про сесію на стороні клієнта через cookies. Це забезпечує кращу безпеку, але потребує додаткових механізмів управління сесіями.

Token Auth (JWT) — модернізований підхід, що використовує *JSON Web Token* для передачі інформації про користувача. Перевага JWT полягає у відсутності потреби зберігати стан на сервері, оскільки токен містить усю необхідну інформацію. Токен підписується секретним ключем, що гарантує його цілісність і автентичність.

Додатково було модифіковано *token_auth* аплікейшен шляхом інтеграції JWT замість простих токенів. Це покращило безпеку додатка та дозволило впровадити токени з обмеженим терміном дії.

Всі три додатки було успішно запущено, виконано запити до серверів та отримано очікувані результати, що підтверджено скріншотами.

Таким чином, у ході роботи було засвоєно:

- базові принципи авторизації у веб-додатках;
- переваги та недоліки кожного з методів;
- реалізацію сучасних підходів до авторизації з використанням JWT.