

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №1 з дисципліни
«Системи безпеки програм і даних»

„Основні методи авторизації ”

Виконав(ла)

ІП-11 Тарасюнок Дмитро Євгенович
(шифр, прізвище, ім'я, по батькові)

Перевірів

Іваніщев Б. В.
(прізвище, ім'я, по батькові)

Київ 2024

ЗМІСТ

1	Мета лабораторної роботи	3
2	Завдання	4
2.1	Основне завдання.....	4
2.2	Додаткове завдання.....	4
3	Виконання основного завдання	5
3.1	Огляд basic_auth	5
3.2	Огляд forms_auth	5
3.3	Огляд token_auth	9
4	Виконання додаткового завдання	11
5	Висновок	18

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – роздивитися основні методи авторизації

2 ЗАВДАННЯ

2.1 Основне завдання

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

Запустити кожен з 3 аплікейшенів та зробити скріншоти запитів до серверу.

2.2 Додаткове завдання

Модифікувати token_auth аплікейшен змінивши токен на JWT.

3 ВИКОНАННЯ ОСНОВНОГО ЗАВДАННЯ

3.1 Огляд basic_auth

Для початку встановлюємо фреймворк express, а потім запускаємо сервер.

```
PS D:\KPI Laboratory Works\3.2 Program and data security systems\Laboratory work 1\auth_examples\basic_auth> npm install express
added 64 packages in 2s
12 packages are looking for funding
  run 'npm fund' for details
PS D:\KPI Laboratory Works\3.2 Program and data security systems\Laboratory work 1\auth_examples\basic_auth> node .\index.js
Example app listening on port 3000
```

Сервер слухає на 3000 порті, спробуємо звернутися до нього з логіном DateArt та паролем 2408.

```
PS C:\Users\dmitr> curl -u DateArt:2408 http://localhost:3000
Hello DateArt
```

Бачимо успіх. Спробуємо звернутися з іншим паролем.

```
PS C:\Users\dmitr> curl -u DateArt:2409 http://localhost:3000
Unauthorized
```

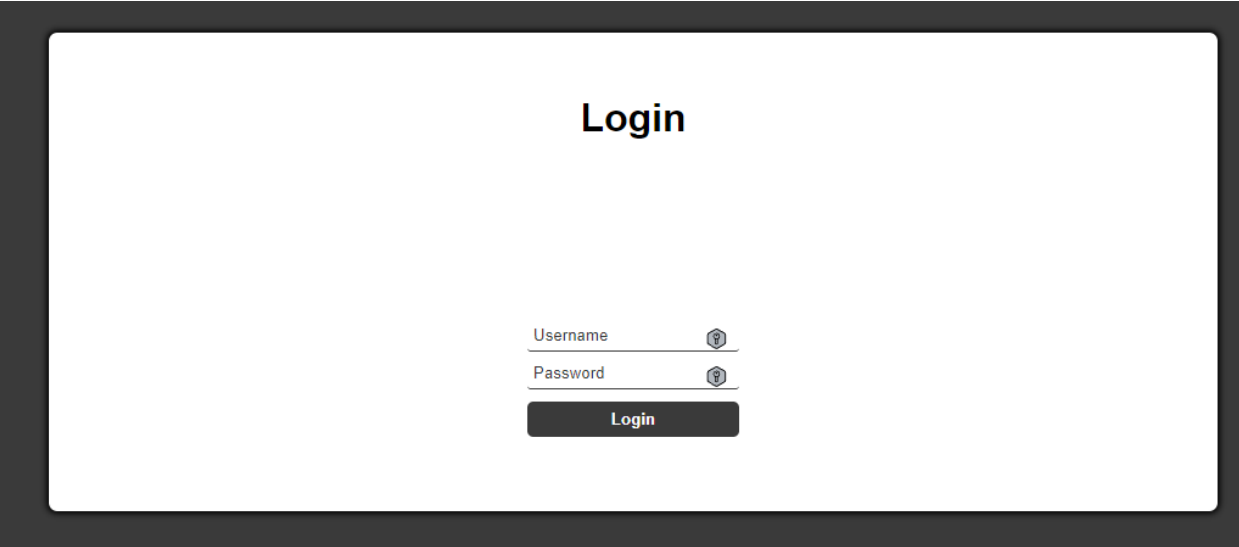
Змінивши пароль, сервер уже не вітає нас.

3.2 Огляд forms_auth

Встановлюємо залежності та запускаємо сервер

```
PS D:\KPI Laboratory Works\3.2 Program and data security systems\Laboratory work 1\auth_examples\forms_auth> npm install express uuid cookie-parser body-parser
added 67 packages in 2s
13 packages are looking for funding
  run 'npm fund' for details
PS D:\KPI Laboratory Works\3.2 Program and data security systems\Laboratory work 1\auth_examples\forms_auth> node index.js
Example app listening on port 3000
```

Заходимо в браузері за адресою <http://localhost:3000>



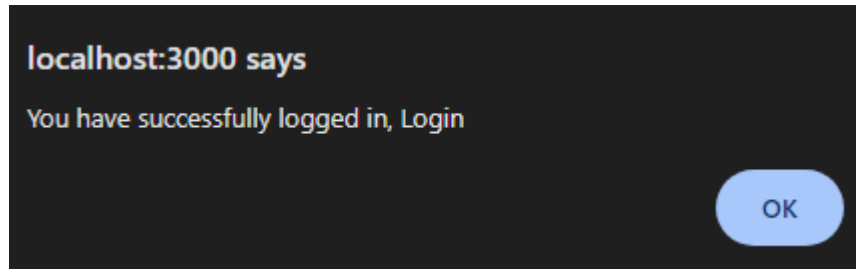
Login

Username

Password

Login

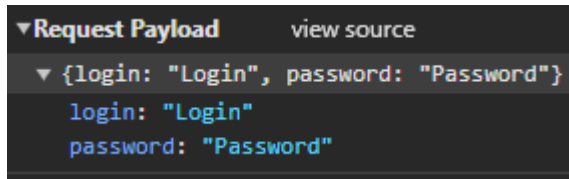
Бачимо вікно авторизації. Вводимо логін Login та пароль Password (узяті з коду серверу) та логінімося.



Усе вдалося. Розглянемо запити детальніше через DevTools у Chrome.

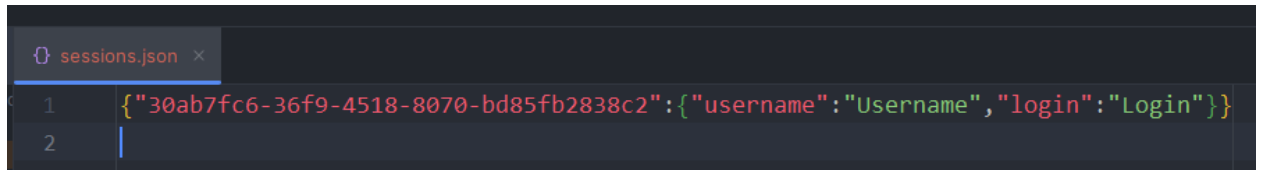
▼ General	
Request URL:	http://localhost:3000/api/login
Request Method:	POST
Status Code:	● 200 OK
Remote Address:	[::1]:3000
Referrer Policy:	strict-origin-when-cross-origin
▼ Response Headers <input type="checkbox"/> Raw	
Connection:	keep-alive
Content-Length:	20
Content-Type:	application/json; charset=utf-8
Date:	Sun, 03 Mar 2024 09:28:28 GMT
Etag:	W/"14-0fqzPGGcLZFMZIE3m2VyMUMvVI8"
Keep-Alive:	timeout=5
X-Powered-By:	Express
▼ Request Headers <input type="checkbox"/> Raw	
Accept:	application/json, text/plain, */*
Accept-Encoding:	gzip, deflate, br, zstd
Accept-Language:	en-US;q=0.9,ru-UA;q=0.8,ru;q=0.7,uk-UA;q=0.6,uk;q=0.5,ru-RU;q=0.4,pl;q=0.3
Connection:	keep-alive
Content-Length:	39
Content-Type:	application/json
Cookie:	session=30ab7fc6-36f9-4518-8070-bd85fb2838c2
Host:	localhost:3000
Origin:	http://localhost:3000
Referer:	http://localhost:3000/
Sec-Ch-Ua:	"Chromium";v="122", "Not(A:Brand";v="24", "Google Chrome";v="122"
Sec-Ch-Ua-Mobile:	?0
Sec-Ch-Ua-Platform:	"Windows"
Sec-Fetch-Dest:	empty
Sec-Fetch-Mode:	cors
Sec-Fetch-Site:	same-origin

Бачимо, що це був POST запит за адресою localhost:3000/api/login. Подивимося payload.



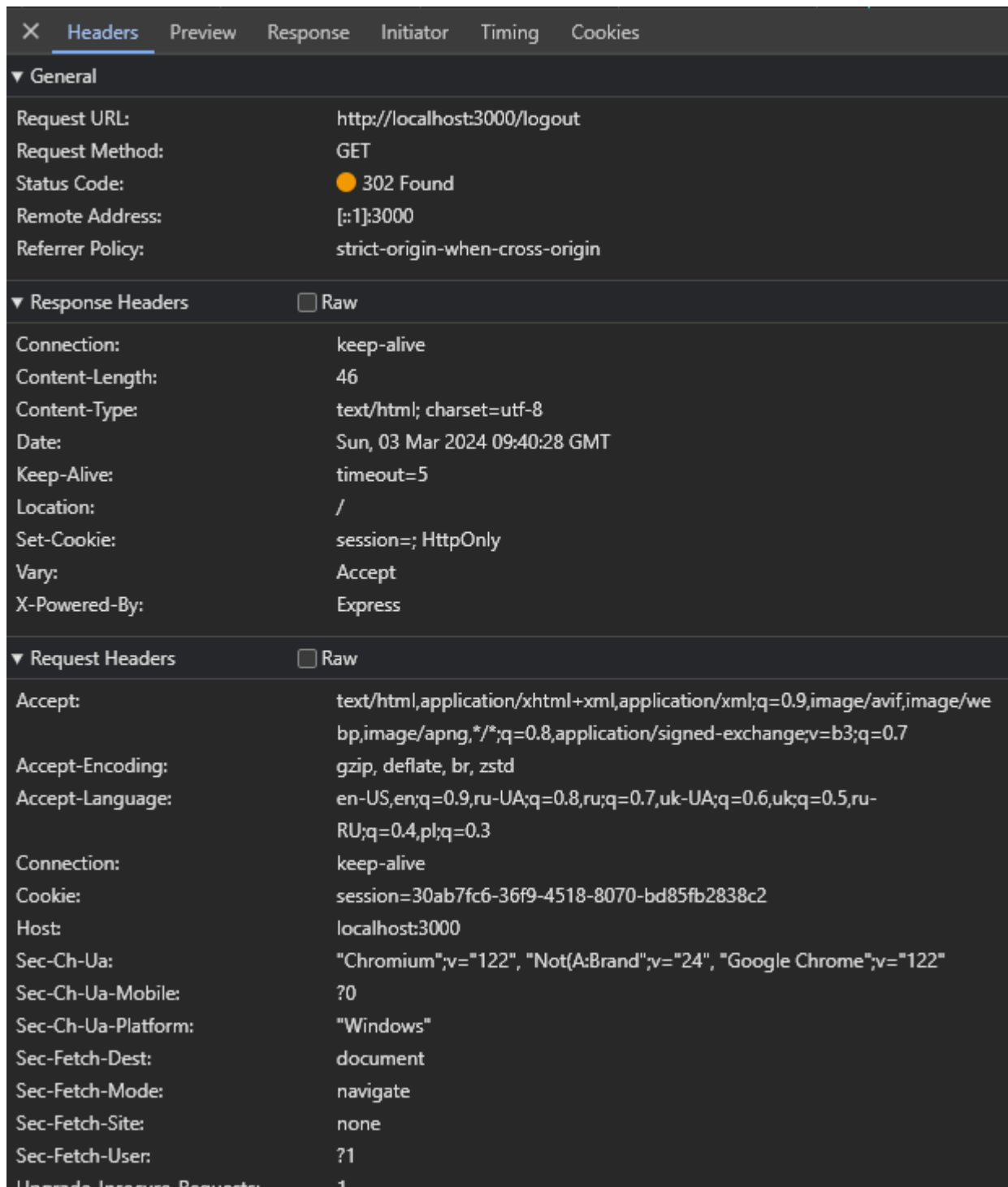
```
▼ Request Payload view source
{login: "Login", password: "Password"}
login: "Login"
password: "Password"
```

Бачимо що авторизаційні дані передаються просто в сирому JSON вигляді. Оглянемо файл sessions.json.



```
sessions.json x
1 {"30ab7fc6-36f9-4518-8070-bd85fb2838c2":{"username":"Username","login":"Login"}}
2 |
```

Бачимо, що тут з'явилася саме та сесія, яку нам повернув сервер. Спробуємо розлогінитися.



Бачимо, що сервер повернув заголовок Set-Cookie, у якому видається наша сесія. Цікаво те, що з файлу session.json не просто не зникла наша сесія, а ще й додалися дві нові. Скоріше за все, це баг самого сервера.


```
sessions.json x JS index.js
1 {
2   "bb1d8388-50f4-46ba-a80c-2f37e26aa054": {
3     "username": "Username",
4     "login": "Login"
5   },
6   "30ab7fc6-36f9-4518-8070-bd85fb2838c2": {
7     "username": "Username",
8     "login": "Login"
9   },
10  "a74edc46-fdf0-405c-8eb0-59ae2df28fe1": {}
11 }
```

3.3 Огляд token_auth

Встановлюємо всі залежності та запускаємо сервер

```
PS D:\KPI Laboratory Works\3.2 Program and data security systems\Laboratory work 1\auth_examples\token_auth> npm install express uuid body-parser on-finished
added 65 packages in 1s
13 packages are looking for funding
  run 'npm fund' for details
PS D:\KPI Laboratory Works\3.2 Program and data security systems\Laboratory work 1\auth_examples\token_auth> node index.js
Example app listening on port 3000
```

Для даного методу авторизації будемо працювати через curl, оскільки браузер не хоче коректно відображати відповіді від сервера.

Робимо запит на авторизацію, знову передаючи логін та пароль у вигляді JSON.

```
PS C:\Users\dmtr> curl -X POST http://localhost:3000/api/login -H "Content-Type: application/json" -d '{"login":"Login", "password":"Password"}'
{"token":"c194f52c-dfb8-46b0-85e2-88cf9b89962a"}
```

Подивимося, що відбулося з файлом sessions.json.

```
{
  "7848bd98-20ce-4c83-9c28-ffe3df79a1ab": {},
  "4fceaa1f-022c-495b-8da5-ae37ec110afa": {},
  "1462ab34-6866-403b-985a-9d15d2f1a0e9": {
    "username": "Username",
    "login": "Login"
  },
  "ce785b9e-d6a7-4e19-9718-1a8a5e095ee9": {},
  "1b9c6433-68b9-45e7-98ae-3f012352e649": {
    "username": "Username",
    "login": "Login"
  },
  "c194f52c-dfb8-46b0-85e2-88cf9b89962a": {
    "username": "Username",
    "login": "Login"
  },
  "06813fec-cd55-4171-a141-65f02fb935aa": {}
}
```

Знову створилася велика кількість сесій, у тому числі й наша. Спробуємо тепер розлогінитися. Вказуємо токен уже не як куки, а в окремому хедері Authorization.

```
PS C:\Users\dmityr> curl -X GET http://localhost:3000/logout -H "Authorization: c194f52c-dfb8-46b0-85e2-88cf9b89962a"
Found. Redirecting to /
```

Цікаво те, що сесія знову не видалилася з файлу sessions.json, хоча її було передано вірно. Сервер локально видаляє сесію зі свого асоціативного масиву, але ці зміни не записуються у файл, через що й відбувається така ситуація. Однак, виправлення даної помилки не входить у заддання даної лабораторної роботи.

4 ВИКОНАННЯ ДОДАТКОВОГО ЗАВДАННЯ

Мною було змінено файл 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">
  <h1 id="login-header">Login</h1>

  <div id="login-error-msg-holder" style="display: none;">
    <p id="login-error-msg">Invalid username and/or
password</p>
  </div>

  <form id="login-form">
    <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>

  <a href="#" id="logout" style="display: none;">Logout</a>
</main>

<script>
  document.getElementById("login-
form").addEventListener("submit", function (e) {
    e.preventDefault();
    const login = document.getElementById("username-
field").value;
    const password = document.getElementById("password-
field").value;

    axios.post('/api/login', {login, password})
      .then(response => {
        sessionStorage.setItem('authToken',
```

```

response.data.token);
        alert('You are now logged in!');
        document.getElementById("logout").style.display
= "block";
        document.getElementById("login-
form").style.display = "none";
    })
    .catch(error => {
        document.getElementById("login-error-msg-
holder").style.display = "block";
    });
});

    document.getElementById("logout").addEventListener("click",
function (e) {
    e.preventDefault();
    sessionStorage.removeItem('authToken');
    alert('You are now logged out!');
    document.getElementById("login-form").style.display =
"block";
    this.style.display = "none";
});

    if (sessionStorage.getItem('authToken')) {
        document.getElementById("login-form").style.display =
"none";
        document.getElementById("logout").style.display =
"block";
    }

    document.addEventListener("DOMContentLoaded", function () {
        const token = sessionStorage.getItem('authToken');
        if (token) {
            axios.get('/', {
                headers: {
                    'Authorization': `Bearer ${token}`
                }
            }).then(response => {
                const {username} = response.data;

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

```

```

        mainHolder.append(`Hello ${username}`);

        loginForm.remove();
        loginErrorMsg.remove();
        loginHeader.remove();
        logoutLink.style.opacity = 1;
    }
    }).catch(error => {
    });
}
});
</script>
</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;
    }

    #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>
</html>
```

Та файл index.js:

```
const express = require('express');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');
const path = require('path');
const port = 3000;

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

const JWT_SECRET_KEY = 'IP11_Tarasonok';

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;
    const user = users.find(user => user.login === login &&
user.password === password);

    if (user) {
        const token = jwt.sign({ username: user.username,
login: user.login }, JWT_SECRET_KEY, { expiresIn: '1h' });
        res.json({ token });
    } else {
        res.status(401).send('Unauthorized');
    }
});
```

```
});

const authenticateToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];

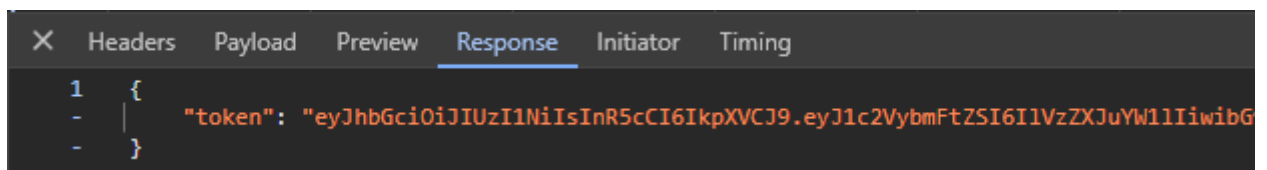
  if (token == null) {
    if (req.path === '/' || req.path === '/api/login')
      return next();
    else return res.sendStatus(401);
  }

  jwt.verify(token, JWT_SECRET_KEY, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
};

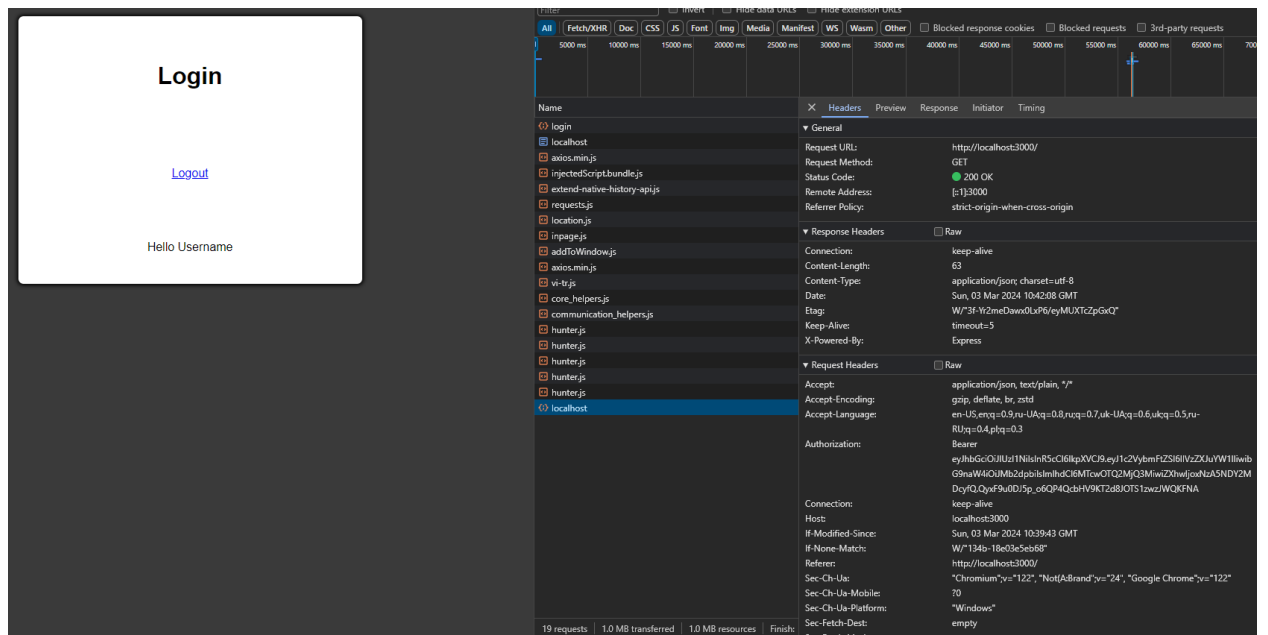
app.get('/', authenticateToken, (req, res) => {
  console.log(req.user);
  if (req.user) {
    return res.json({
      username: req.user.username,
      logout: 'http://localhost:3000/logout'
    });
  }
  res.sendFile(path.join(__dirname + '/index.html'));
});

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

Продемонструємо роботу зміненого способу авторизації на практиці. При переході на сторінку localhost:3000 відкривається вже знайома нам веб-сторінка з формою авторизації. Логінімося з тими ж даними.



В усьому запит такий же, тільки бачимо, що змінився формат токену. Перезавантажимо головну сторінку та побачимо, що буде.



По-перше бачимо, що коректно відобразився юзернейм. Також можна побачити, як саме передається токен: хедер Authorization, а в ньому значення Bearer <наш токен>.

5 ВИСНОВОК

У ході даної лабораторної роботи ми ознайомилися з базовими способами авторизації. У першому випадку ми напряму передавали авторизаційні дані в запитах, у другому випадку – ми отримували айді сесії, передаючи дані у форматі JSON. Айді сесії при цьому зберігався в кукі. У третьому випадку спосіб отримання токена залишився таким же, але зберігатися вже він почав у localStorage. Далі ми змінили третій спосіб авторизації на JWT. Покращили програму тим, що тепер немає необхідності зберігати сесії локально у файлі, як це було в другому та третьому випадках. Тепер токени зберігаються тільки на стороні користувача й валідуються на стороні сервера за допомогою секретного ключа.