



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

## **Лабораторна робота №6**

**Безпека ПЗ**

**Тема:** Протокол OAuth2

Виконав

студент групи ІП-11:

Панченко С. В.

Перевірив:

Курченко О. А.

Київ 2024

## TABLE OF CONTENTS

1 Мета лабораторної роботи.....	3
2 Завдання.....	4
3 Виконання.....	5
3.1 Додаткове та основне завдання.....	5
Висновок.....	7
ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ.....	8

# 1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Засвоїти базові навички OAuth2 протоколу авторизації.

## 2 ЗАВДАННЯ

### 1. Основне завдання

Розширити Лабораторну роботу 4, змінивши логін сторінку на стандартну від SSO провайдера, для цього, треба зробити перенаправлення на API\_DOMAIN <https://kpi.eu.auth0.com/authorize> та додатково додати параметри вашого додатку client\_id, redirect\_uri, response\_type=code, response\_mode=query [https://kpi.eu.auth0.com/authorize?client\\_id=JlvCO5c2IBHIAe2patn6l6q5H35qxti0&redirect\\_uri=http%3A%2F%2Flocalho%3A3000&response\\_type=code&response\\_mode=query](https://kpi.eu.auth0.com/authorize?client_id=JlvCO5c2IBHIAe2patn6l6q5H35qxti0&redirect_uri=http%3A%2F%2Flocalho%3A3000&response_type=code&response_mode=query)

Надати код рішення.

### 2. Додаткове завдання

Додатково розширити застосунок обробкою перенаправлення та отриманням користувача токена за допомогою code grant type. <https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow>

## 3 ВИКОНАННЯ

### 3.1 Додаткове та основне завдання

Створимо нову кінцеву точку **/login**, що відповідатиме за перенаправлення, а також кінцеву точку **/callback**, що надсилає токен при успішній авторизації.

Додамо **/callback** у список callback-ів в налаштуваннях додатку на рисунку 3.1.

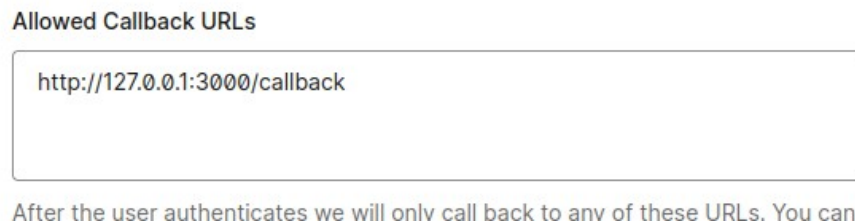


Рисунок 3.1 — Додавання нового елементу в список дозволених callback-ів

Зберігаємо токен, якщо він присутній, у sessionstorage. Повний код index.html, index.ts можна побачити в додатках. При вході відображається сторінка авторизації провайдера на рисунку 3.2.

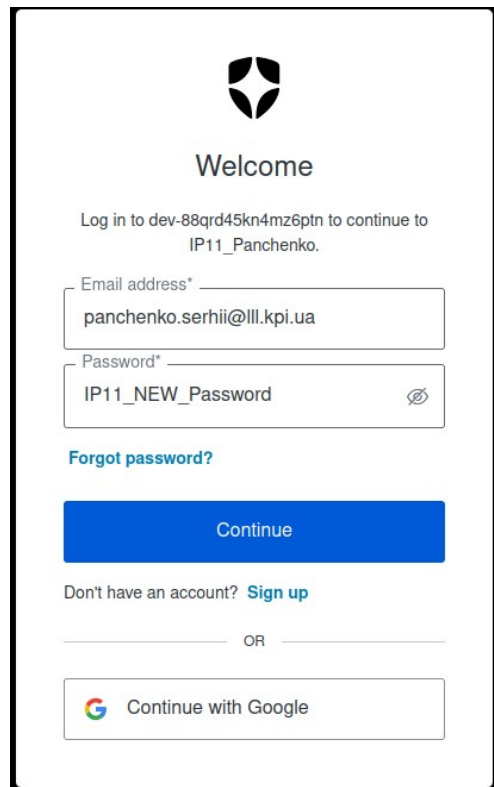


Рисунок 3.2 — Сторінка авторизації провайдера

Після вводу даних нас перенаправлять на підтвердження доступу до профілю на рисунку 3.3.

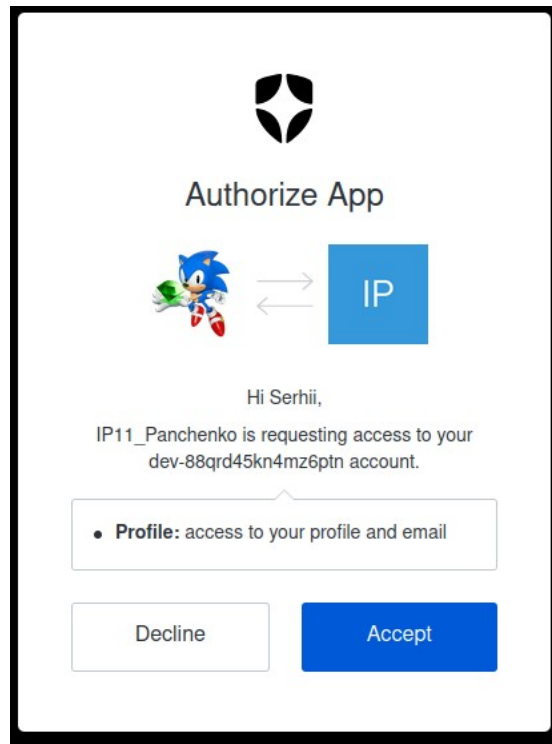


Рисунок 3.3 — Надання доступу до профілю

Після натиснення “Асерт” ми перейдемо на сторінку користувача на рисунку 3.4.

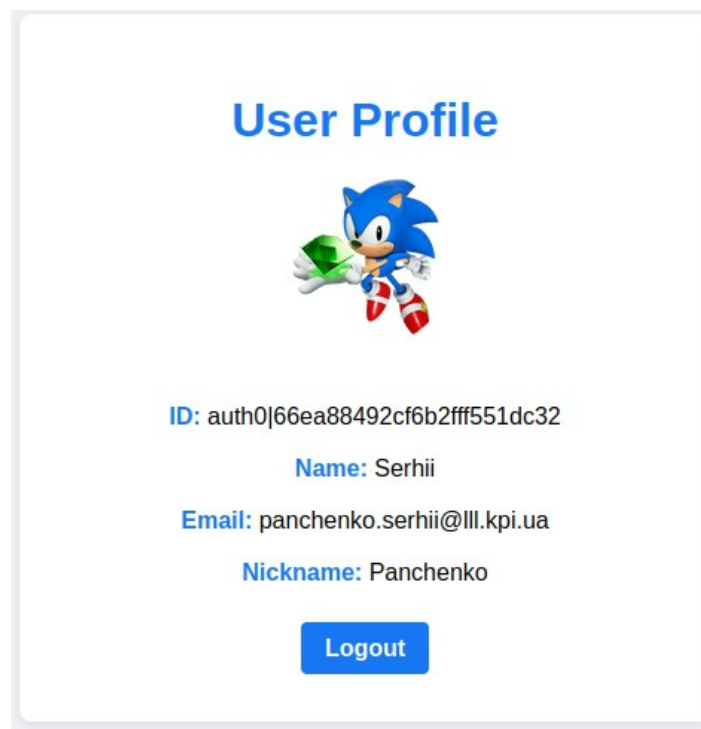


Рисунок 3.4 — Сторінка профілю користувача

## ВИСНОВОК

У процесі виконання лабораторного завдання було проведено суттєву модифікацію системи автентифікації. Замість стандартної сторінки входу, було впроваджено механізм перенаправлення на сторінку авторизації зовнішнього провайдера. Для повноцінної реалізації цього підходу, в рамках додаткового завдання, було розроблено та інтегровано нову кінцеву точку (endpoint) для обробки зворотного виклику (callback) від провайдера автентифікації. Після успішного проходження процедури авторизації, веб-додаток тепер здатний отримувати та відображати розширений набір даних про користувача, забезпечуючи більш персоналізований досвід взаємодії.

# ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

*Тексти програмного коду*  
(Найменування програми (документа))

*Жорсткий диск*  
(Вид носія даних)

(Обсяг програми (документа), арк.)

*Студента групи ІП-11 4 курсу*  
*Панченка С. В*



## **.env.ts**

```
SESSION_KEY=Authorization
AUDIENCE=https://dev-88qrd45kn4mz6ptn.us.auth0.com/api/v2/
AUTH0_DOMAIN=dev-88qrd45kn4mz6ptn.us.auth0.com
AUTH0_CLIENT_ID=zdzhN4dIDmIuptnArPhrZ4TEQAkQNGDH
AUTH0_CLIENT_SECRET=3e54llBevPi1cNu22Yr7ggExIntYXkJgs7_BZ50TleTF1GFnt3ei0T
rSlPkTLK4a
PORT=3000
```

## **index.ts**

```
import * as dotenv from 'dotenv';
import * as express from 'express';
import axios from 'axios';
import * as path from 'path';

dotenv.config();

const app = express();
const port = process.env.PORT || 3000;

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

app.get('/', (_, res) => {
  res.sendFile(path.join(__dirname, 'index.html'));
});

app.get('/login', (_, res) => {
  const authUrl =
    `https://${process.env.AUTH0_DOMAIN}/authorize?` +
    `client_id=${encodeURIComponent(process.env.AUTH0_CLIENT_ID || '')}&`
+
    `redirect_uri=${encodeURIComponent('http://localhost:3000/callback')}&` +
    `response_type=code&` +
    `response_mode=query&` +
    `scope=openid profile email`;
  res.redirect(authUrl);
});

app.get('/callback', async (req, res) => {
  const { code } = req.query;
  try {
    const response = await axios.post(
```

```

    `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
    new URLSearchParams({
      grant_type: 'authorization_code',
      client_id: process.env.AUTH0_CLIENT_ID || '',
      client_secret: process.env.AUTH0_CLIENT_SECRET || '',
      code: code as string,
      redirect_uri: 'http://localhost:3000/callback',
    }),
    {
      headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
    }
  );
  const { access_token } = response.data;
  res.redirect(`/?token=${access_token}`);
} catch (error) {
  console.error('Error exchanging code for tokens:', error);
  res.status(500).send('Internal Server Error');
}
});

app.get('/api/userinfo', async (req, res) => {
  const token = req.headers['authorization'];
  try {
    const response = await axios({
      method: 'get',
      url: `https://${process.env.AUTH0_DOMAIN}/userinfo`,
      headers: {
        'content-type': 'application/json',
        Authorization: token,
      },
    });
    res.json({ success: true, user: response.data });
  } catch (e) {
    console.log(e);
    res.status(500).json({ success: false, error: 'Failed to fetch user
info' });
  }
});

function handleLogout(req: express.Request, res: express.Response) {
  req.session.destroy(() => {
    res.redirect('/');
  });
}

```

```
app.get('/logout', handleLogout);

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

## **index.html.**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
  <title>User Profile</title>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  <style>
    body {
      font-family: 'Arial', sans-serif;
      background-color: #f0f2f5;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
    }
    #main-holder {
      background-color: white;
      border-radius: 8px;
      box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
      padding: 2rem;
      width: 100%;
      max-width: 400px;
      text-align: center;
    }
    #logout {
      display: inline-block;
      margin-top: 1rem;
      padding: 0.5rem 1rem;
      background-color: #1877f2;
      color: white;
      text-decoration: none;
      border-radius: 4px;
```

```

        font-weight: bold;
        transition: background-color 0.3s ease;
    }
    #logout:hover {
        background-color: #166fe5;
    }
    .profile-pic {
        width: 120px;
        height: 120px;
        border-radius: 50%;
        object-fit: cover;
        margin-bottom: 1rem;
    }
    .user-info {
        margin-bottom: 0.5rem;
    }
    .user-info span {
        font-weight: bold;
        color: #1877f2;
    }
    h1 {
        color: #1877f2;
        margin-bottom: 1rem;
    }
</style>
</head>
<body>
    <main id="main-holder">
        <h1>User Profile</h1>
        <!-- User info will be inserted here -->
    </main>

    <script>
        const mainHolder = document.getElementById('main-holder');
        const queryParams = new URLSearchParams(window.location.search);
        const setToken = queryParams.get('token');
        const session = sessionStorage.getItem('session');
        let token;

        if (setToken) {
            sessionStorage.setItem('session', JSON.stringify({ token:
setToken }));

            token = setToken;
            const newUrl = window.location.pathname;

```

```

        window.history.pushState('', 'Profile', newUrl);
    }

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

    if (!token) {
        location.href = '/login';
    }

    if (token) {
        axios.get('/api/userinfo', {
            headers: {
                Authorization: `Bearer ${token}`,
            },
        })
        .then((response) => {
            const { user } = response.data;
            if (user) {
                const userInfoDiv = document.createElement('div');
                userInfoDiv.innerHTML = `
                    
                    <p class="user-info"><span>ID:</span> $
{user.sub}</p>
                    <p class="user-info"><span>Name:</span> $
{user.name}</p>
                    <p class="user-info"><span>Email:</span> $
{user.email}</p>
                    <p class="user-info"><span>Nickname:</span> $
{user.nickname}</p>
                    <a href="/logout" id="logout">Logout</a>
                `;
                mainHolder.appendChild(userInfoDiv);

                const logoutLink = document.getElementById('logout');
                logoutLink.addEventListener('click', (e) => {
                    e.preventDefault();
                    sessionStorage.removeItem('session');
                    location.reload();
                });
            }
        })
    }
}

```

```
        .catch((error) => {
            console.error('Error fetching user info:', error);
            mainHolder.innerHTML += '<p>Error loading user
information. Please try again later.</p>';
        });
    }
</script>
</body>
</html>
```