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 №4

Безпека ПЗ

Тема: Протокол OAuth2

Виконав
студент групи ІП-11:
Панченко С. В.

Перевірив:
Курченко О. А.

Київ 2024

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Засвоїти базові навички OAuth2 авторизаційного протоколу.

2 ЗАВДАННЯ

1. Основне завдання

Використовуючи наведені налаштування з лабораторної роботи 2 - 3 та приведених запитів модифікувати додаток

https://github.com/Kreolwolf1/auth_examples/tree/main/token_auth

Використовуючи перевірку користувача та отримання токена з auth0 (password grant type)

Надати код модифікованого додатка.

2. Додаткове завдання

Додатково розширити додаток створенням користувача та перевіркою життя

токена (у разі близького завершення – оновити токен використовуючи refresh-token grant type)

3 ВИКОНАННЯ

3.1 Основне завдання

Лабораторну роботу виконаємо з допомогою TypeScript. Код наведено у додатках.

Щоб перевірити правильність роботи, авторизуємося з даними panchenko.serhii@lll.kpi.ua та IP11_NEW_Password на рисунку 3.1.

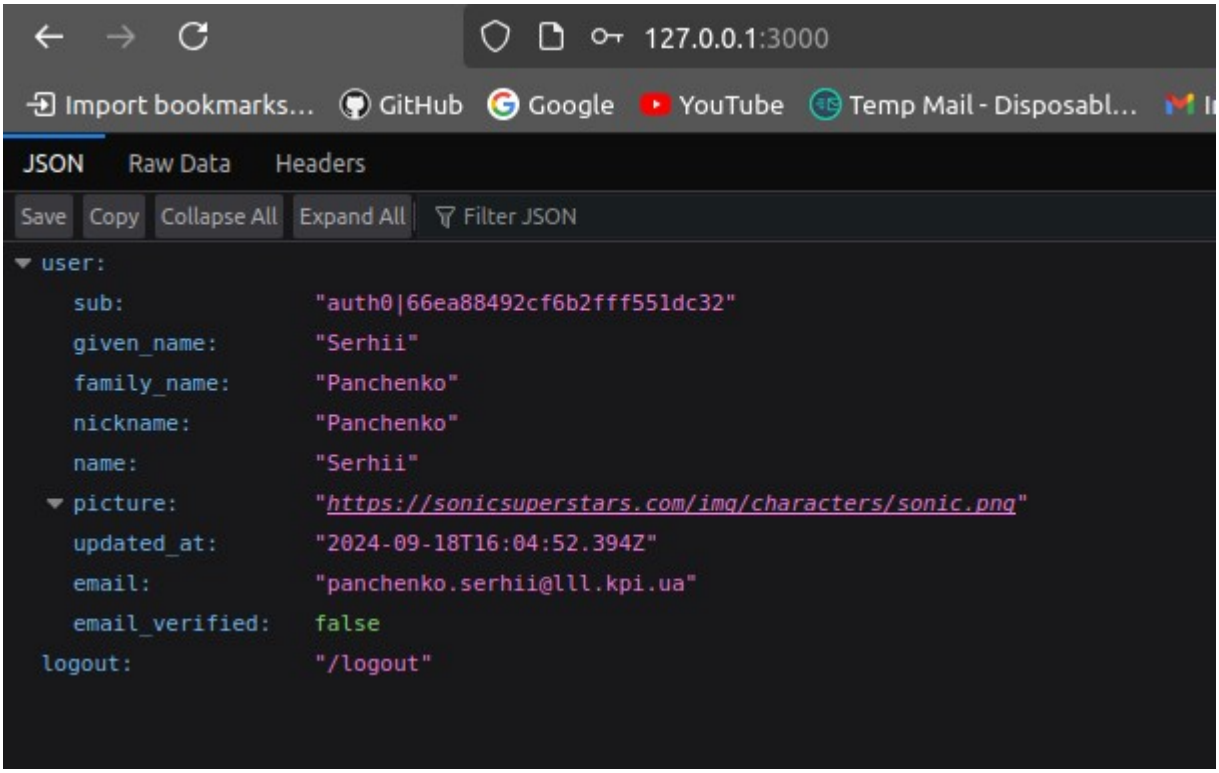

Рисунок 3.1 — Приклад успішної авторизації

3.2 Додаткове завдання

Додали форму реєстрації нижче та спробуємо зареєструватися на рисунку 3.2 з паролем IP11Panchenko1234&&$#.

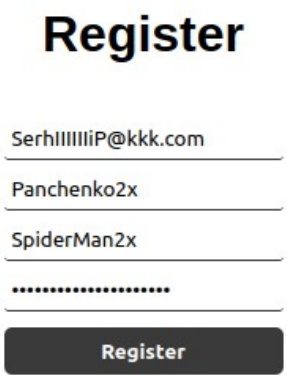

Рисунок 3.2 — Форма реєстрації

Як бачимо на рисунку 3.3, реєстрація успішна.

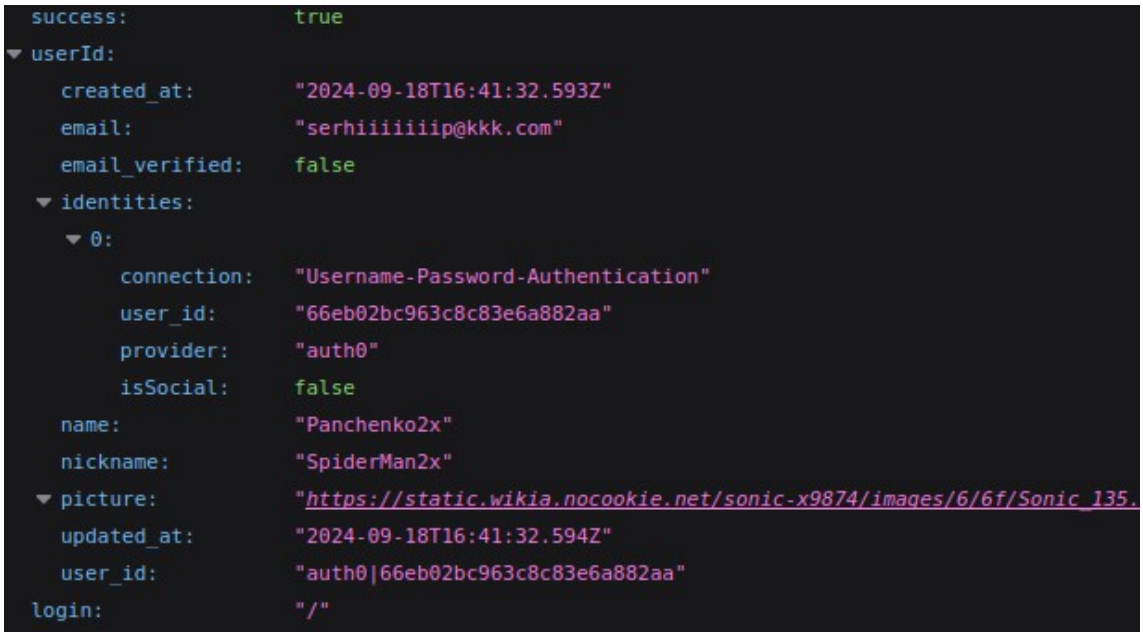

Рисунок 3.3 — Приклад успішної реєстрації

Спробуємо авторизуватися на рисунку 3.4.

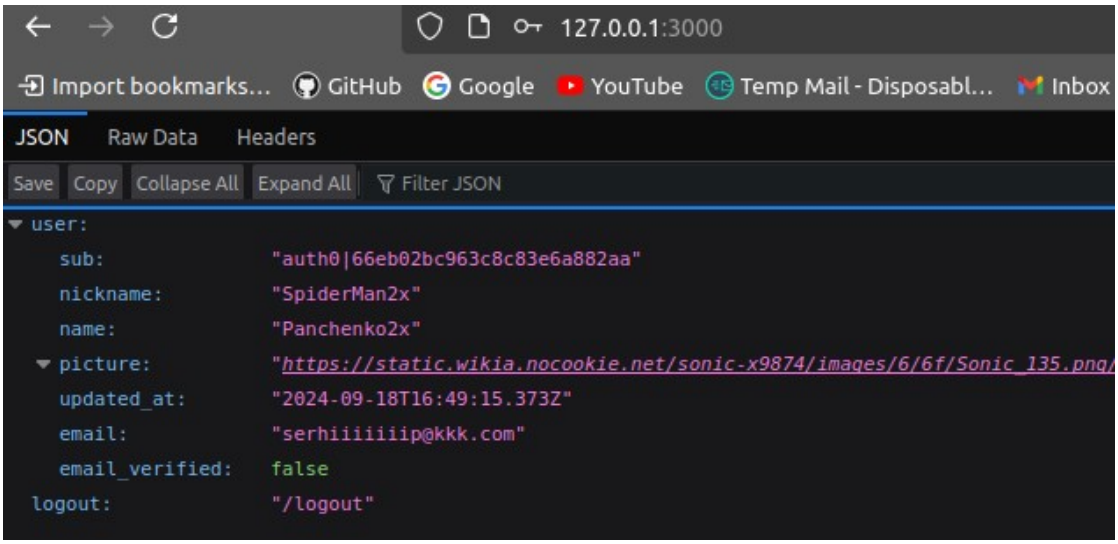

Рисунок 3.4 — Приклад успішної авторизації

Також додана функція **refreshToken, код якої можна переглянути в додатках.** При виконанні запитів виконується перевірка, а за 5 хв до відтермінування токену він оновлюється. Також є логування на рисунку 3.5.

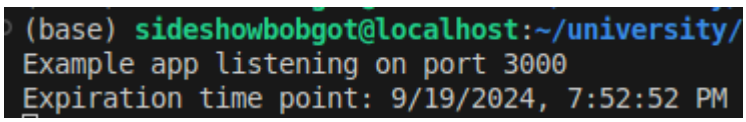

Рисунок 3.5 — Логування часу інвалідації токена

ВИСНОВОК

У результаті лабораторної роботи було суттєво вдосконалено приклад авторизації token_auth. Запити на авторизацію перенесено з постера безпосередньо до додатку. Тепер форма надсилає запити напряму до Auth0. Процеси авторизації та виходу з облікового запису були успішно протестовані та підтверджені.

При виконанні додаткового завдання функціональність додатку розширено новою кінцевою точкою для реєстрації користувачів та функцією перевірки дійсності токена. Крім того, на головну сторінку додано форму реєстрації, що покращує процес залучення нових користувачів.

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
AUTH0_DOMAIN=dev-88qrd45kn4mz6ptn.us.auth0.com
AUTH0_CLIENT_ID=zdzhN4dIDmIuptnArPhrZ4TEQAkQNGDH
AUTH0_CLIENT_SECRET=3e54llBevPi1cNu22Yr7ggExIntYXkJgs7_BZ5OTleTF1GFnT3eiOTrSlPkTLK4a
PORT=3000
```

index.ts.

```
import * as dotenv from 'dotenv';
import * as express from 'express';
import axios, { AxiosError } from 'axios';
import * as path from 'path';
import * as session from 'express-session';

dotenv.config();

const app = express();
const port = process.env.PORT ? parseInt(process.env.PORT, 10) : 3000;

declare module 'express-session' {
  interface SessionData {
    tokens?: {
      access_token: string;
      refresh_token: string;
      expires_in: number;
    };
  }
}

interface LoginBody {
  login: string;
  password: string;
}

interface RegisterBody {
  email: string;
  password: string;
  name: string;
  nickname: string;
}



app.use(express.json());
```

```
      app.use(express.urlencoded({ extended: true }));
      app.use(
        session({
          secret: process.env.SESSION_SECRET || 'default_secret',
          resave: false,
          saveUninitialized: true,
          cookie: { secure: process.env.NODE_ENV === 'production' },
        })
      );

      const expiration_period = 5 * 60 * 1000;

      const refreshToken: express.RequestHandler = async (req, res, next) => {
        const tokens = req.session.tokens;

        if (!tokens) return next();

        const isTokenExpired = (expiresIn: number, bufferPeriod: number) =>
          Date.now() > expiresIn - bufferPeriod;

        const logExpirationDate = (expiresIn: number) => {
          const expirationDate = new Date(expiresIn);
          console.log(`Expiration time point: $
{expirationDate.toLocaleString()}`);
        };

        const refreshAuthToken = async (refreshToken: string) => {
          try {
            const response = await axios.post<{ access_token: string;
expires_in: number }>(
              `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
              new URLSearchParams({
                grant_type: 'refresh_token',
                client_id: process.env.AUTH0_CLIENT_ID!,
                client_secret: process.env.AUTH0_CLIENT_SECRET!,
                refresh_token: refreshToken,
              }),
              { headers: { 'content-type': 'application/x-www-form-urlencoded' }
}
            );

            return {
              access_token: response.data.access_token,
              expires_in: Date.now() + response.data.expires_in * 1000,
```

```
            };
          } catch (error) {
            throw new Error(
              (error as AxiosError).response?.data
                ? JSON.stringify((error as AxiosError).response?.data)
                : (error as Error).message
            );
          }
        };

        if (tokens.expires_in) {
          logExpirationDate(tokens.expires_in);
        }

        if (isTokenExpired(tokens.expires_in, expiration_period)) {
          try {
            const newTokens = await refreshAuthToken(tokens.refresh_token);
            req.session.tokens = {
              ...newTokens,
              refresh_token: tokens.refresh_token,
            };
          } catch (error) {
            console.error('Error refreshing token:', error.message);
            return res.status(401).json({ error: 'Failed to refresh token' });
          }
        }

        next();
      };

      app.use(refreshToken);

      async function handleHomeRoute(req: express.Request, res:
express.Response) {
        const session = req.session;

        if (session.tokens) {
          try {
            const userInfo = await getUserInfo(session.tokens.access_token);
            res.json({
              user: userInfo,
              logout: '/logout',
            });
          } catch (error) {
```

```
            await handleUserInfoError(req, res, error);
          }
        } else {
          res.sendFile(path.join(__dirname, 'index.html'));
        }
      }

      async function handleUserInfoError(req: express.Request, res:
express.Response, error: unknown) {
        console.error('Error:', (error as AxiosError).response?.data || (error
as Error).message);
        await new Promise<void>((resolve) => req.session.destroy(() =>
resolve()));
        res.status(500).json({ error: 'An error occurred while fetching user
info' });
      }

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

      export async function getUserInfo(accessToken: string) {
        const response = await axios.get(
          `https://${process.env.AUTH0_DOMAIN}/userinfo`,
          {
            headers: {
              Authorization: `Bearer ${accessToken}`,
            },
          }
        );
        return response.data;
      }

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

      app.post('/api/login', async (req: express.Request<{}, {}, LoginBody>,
res) => {
        try {
          const { login, password } = req.body;
          const response = await axios.post<{ access_token: string;
refresh_token: string; expires_in: number }>(
```

```
            `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
            new URLSearchParams({
              grant_type: 'password',
              username: login,
              password: password,
              client_id: process.env.AUTH0_CLIENT_ID!,
              client_secret: process.env.AUTH0_CLIENT_SECRET!,
              audience: `https://${process.env.AUTH0_DOMAIN}/api/v2/`,
              scope: 'offline_access openid profile email',
            }),
            {
              headers: { 'content-type': 'application/x-www-form-urlencoded' },
            }
          );

          req.session.tokens = {
            access_token: response.data.access_token,
            refresh_token: response.data.refresh_token,
            expires_in: Date.now() + response.data.expires_in * 1000,
          };
          res.json({ success: true, token: response.data.access_token });
        } catch (error) {
          console.error('Login failed:', (error as AxiosError).response?.data ||
(error as Error).message);
          res.status(401).send('Login failed');
        }
      });

      async function getAuth0Token() {
        const response = await axios.post<{ access_token: string }>(
          `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
          new URLSearchParams({
            grant_type: 'client_credentials',
            client_id: process.env.AUTH0_CLIENT_ID!,
            client_secret: process.env.AUTH0_CLIENT_SECRET!,
            audience: `https://${process.env.AUTH0_DOMAIN}/api/v2/`,
          }),
          {
            headers: { 'content-type': 'application/x-www-form-urlencoded' },
          }
        );
        return response.data.access_token;
      }
```

```
      async function registerUser(req: express.Request<{}, {}, RegisterBody>,
res: express.Response) {
        const { email, password, name, nickname } = req.body;

        try {
          const authToken = await getAuth0Token();
          const user = await createAuth0User(authToken, { email, password, name,
nickname });

          res.status(201).json({
            success: true,
            userId: user,
            login: '/',
          });
        } catch (error) {
          handleError(res, error as Error | AxiosError);
        }
      }

      async function createAuth0User(token: string, userData: Omit<RegisterBody,
'password'> & { password: string }) {
        const response = await axios.post(
          `https://${process.env.AUTH0_DOMAIN}/api/v2/users`,
          {
            ...userData,
            connection: 'Username-Password-Authentication',
            verify_email: true,
            picture:
'https://static.wikia.nocookie.net/sonic-x9874/images/6/6f/Sonic_135.png/
revision/latest/thumbnail/width/360/height/360?cb=20160226182328',
          },
          {
            headers: {
              Authorization: `Bearer ${token}`,
              'content-type': 'application/json',
            },
          }
        );
        return response.data;
      }

      export function handleError(res: express.Response, error: Error |
AxiosError) {
        console.error('Registration failed:', (error as
```

```
AxiosError).response?.data || error.message);
        res.status(400).json({
          success: false,
          error: (error as AxiosError).response?.data || error.message,
        });
      }

      app.post('/api/register', registerUser);

      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>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">
                      <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>

                  <h1 id="register-header">Register</h1>
```

```
                <form id="register-form" action="/api/register" method="post">
                    <input type="email" name="email" id="email" class="login-
form-field" placeholder="Email">
                    <input type="text" name="name" id="name" class="login-
form-field" placeholder="Name">
                    <input type="text" name="nickname" id="nickname"
class="login-form-field" placeholder="Nickname">

                    <input type="password" name="password" id="password-field"
class="login-form-field" placeholder="Password">
                    <input type="submit" value="Login" id="register-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,
#register-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,
        #register-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');

        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");


        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>
```