

**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 №4 з дисципліни  
«Системи безпеки програм і даних»

«Протокол OAuth2»

**Виконав(ла)**

ІП-11 Сідак Кирил Ігорович  
(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
4	Виконання додаткового завдання.....	11
5	Висновок .....	15

## 1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – засвоювання базових навичок OAuth2 авторизаційного протокола.

## 2 ЗАВДАННЯ

### 2.1 Основне завдання

Використовуючи наведені налаштування з лабораторної роботи 2 - 3 та приведених запитів модифікувати аплікейшен [https://github.com/Kreolwolf1/auth\\_examples/tree/main/token\\_auth](https://github.com/Kreolwolf1/auth_examples/tree/main/token_auth)

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

Надати код модифікованного аплікейшена.

### 2.2 Додаткове завдання

Додатково розшири аплікайшен створенням юзера та перевіркою життя токена (у разі близького завершення – оновити токен використовуючи refresh-token grant type)

### 3 ВИКОНАННЯ ОСНОВНОГО ЗАВДАННЯ

Для виконання завдання було створено новий репозиторій lab4 та додано файл .env, для безпечного використання змінних. Зміст файлу:

```
SESSION_KEY = Authorization
AUTH0_DOMAIN = dev-o0271wx4nn4u0i0k.us.auth0.com
AUTH0_CLIENT_ID = XAOzNA98H0u80yP43YAmqKe0ItntJasW
AUTH0_CLIENT_SECRET =
1eYfwJVweqbM_Qv5_xvGBPwX6PJU8XZaziErodefavtjLtXBI1rsnwGZWmJIkZdk
PORT=3000
```

Для подальшої модифікації додано файли з папки token\_auth. В модифікованому коді, зменшено візуальний функціонал, вивід та особистий кабінет тепер мають вигляд json відповіді (для того щоб не звертати увагу на зовнішній вигляд). Перероблений index.js:

```
require('dotenv').config()
const express = require('express')
const axios = require('axios')
const path = require('path')
const session = require('express-session')

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

app.use(express.json())
app.use(express.urlencoded({ extended: true }))
app.use(
  session({
    secret: 'secret',
    resave: false,
    saveUninitialized: true,
    cookie: { secure: false },
  })
)

app.get('/', async (req, res) => {
  if (req.session.tokens) {
    try {
      const { access_token } = req.session.tokens
      const response = await axios.get(
        `https://${process.env.AUTH0_DOMAIN}/userinfo`,
        {
          headers: {
            Authorization: `Bearer ${access_token}`,
          },
        },
      )
    }
  }
})
```

```

    )

    return res.json({
      user: response.data,
      logout: '/logout',
    })
  } catch (error) {
    console.error('Error:', error.response?.data || error.message)
    req.session.destroy()
  }
}

res.sendFile(path.join(__dirname, 'index.html'))
})

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

app.post('/api/login', async (req, res) => {
  try {
    const { login, password } = req.body
    const response = await axios({
      method: 'post',
      url: `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
      headers: { 'content-type': 'application/x-www-form-urlencoded' },
      data: new URLSearchParams({
        grant_type: 'password',
        username: login,
        password: password,
        client_id: process.env.AUTH0_CLIENT_ID,
        client_secret: process.env.AUTH0_CLIENT_SECRET,
        audience: `https://${process.env.AUTH0_DOMAIN}/api/v2/`,
        scope: 'offline_access openid profile email',
      }),
    })

    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.response?.data || error.message)
    res.status(401).send('Login failed')
  }
})

```

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

А також перероблений 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>

    </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');

  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>

```

Для перевірки заходимо на localhost:3000 та намагаємось авторизуватись з даними користувача в auth0 (kyrylo.sidak@yopmail.com: IP11Sidak2):



The screenshot shows a JSON object representing a user profile and a logout endpoint. The 'user' object contains the following fields:

- sub:** "auth0|6634aef7671f9eeb7d04ed70"
- given\_name:** "Kyrylo"
- family\_name:** "Sidak"
- nickname:** "Sidak"
- name:** "Kyrylo"
- picture:** "https://i.pinimg.com/originals/e1/4c/ae/e14cae2f0f44121ab4e3506002ba1a55.jpg"
- updated\_at:** "2024-05-04T14:39:56.141Z"
- email:** "kyrylo.sidak@yopmail.com"
- email\_verified:** true

The JSON also includes a **logout** field with the value **"/logout"**.

Ми бачимо, що користувача було авторизовано. При оновленні сторінки сесія залишається, вихід з акаунту виконується за допомогою /logout.

## 4 ВИКОНАННЯ ДОДАТКОВОГО ЗАВДАННЯ

Для реєстрації користувача до файлу index.html було додано нову форму реєстрації:

```
<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>
```

В index.js створено новий ендпоінт для реєстрації:

```
app.post('/api/register', async (req, res) => {
  const { email, password, name, nickname } = req.body

  try {
    // Отримання токену
    const authData = await axios.post(
      `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' },
      }
    )

    // Запит реєстрації
    const userResponse = await axios.post(
      `https://${process.env.AUTH0_DOMAIN}/api/v2/users`,
      {
        email: email,
        password: password,
        connection: 'Username-Password-Authentication',
        verify_email: true,
        name: name,
        nickname: nickname,
        picture:
```

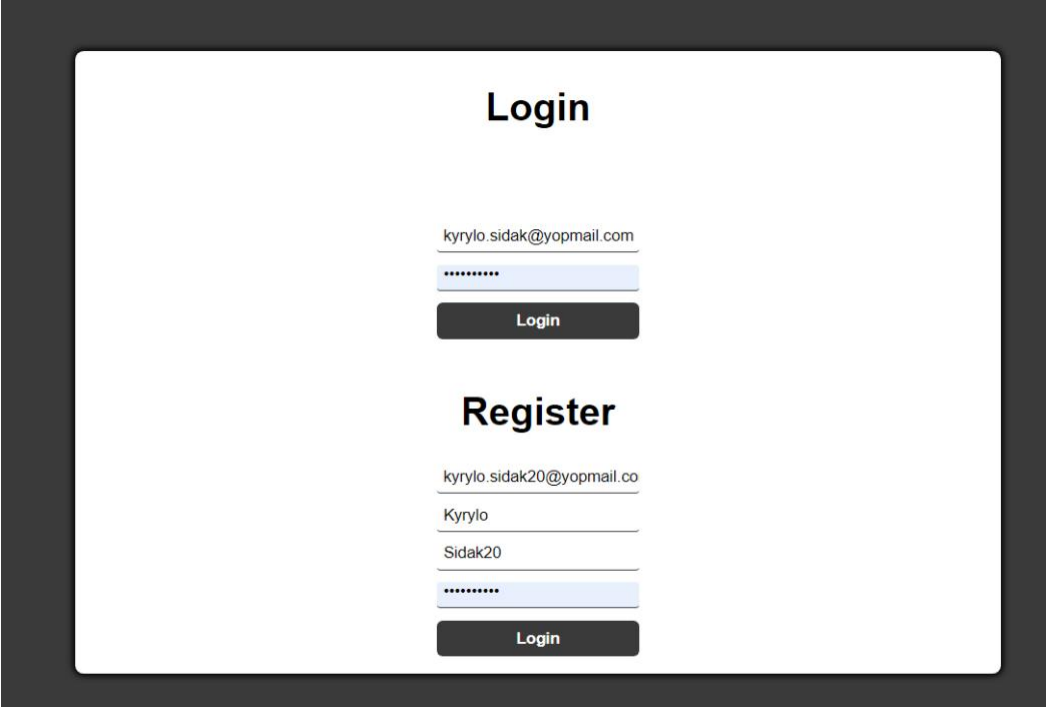
```

        'https://i.pinimg.com/originals/e1/4c/ae/e14cae2f0f44121ab4e3506002ba1a
55.jpg',
      },
      {
        headers: {
          Authorization: `Bearer ${authData.data.access_token}`,
          'content-type': 'application/json',
        },
      },
    ],
  )
  res.status(201).json({
    success: true,
    userId: userResponse.data,
    login: '/',
  })
} catch (error) {
  console.error('Registration failed:', error.response?.data || error.message)
  res.status(400).json({
    success: false,
    error: error.response?.data,
  })
}
}
})

```

Процес реєстрації було спрощено, тому всі інші поля прописані в тілі запиту (наприклад зображення).

Намагаємось зареєструвати аккаунт:



The screenshot displays a web interface with two main sections: 'Login' and 'Register'. The 'Login' section has a text input for the email 'kyrylo.sidak@yopmail.com' and a password input field with masked characters. Below these is a dark 'Login' button. The 'Register' section has three text inputs: the first contains 'kyrylo.sidak20@yopmail.co', the second contains 'Kyrylo', and the third contains 'Sidak20'. There is also a password input field with masked characters and a dark 'Login' button at the bottom.

```

{
  "success": true,
  "userId": {
    "created_at": "2024-05-04T14:52:57.101Z",
    "email": "kyrylo.sidak20@yopmail.com",
    "email_verified": false,
    "identities": [
      {
        "connection": "Username-Password-Authentication",
        "user_id": "66364bc917f99a74d3a3bd1c",
        "provider": "auth0",
        "isSocial": false
      }
    ],
    "name": "Kyrylo",
    "nickname": "Sidak20",
    "picture": "https://i.pinimg.com/originals/e1/4c/ae/e14cae2f0f44121ab4e3506002ba1a55.jpg",
    "updated_at": "2024-05-04T14:52:57.101Z",
    "user_id": "auth0|66364bc917f99a74d3a3bd1c"
  },
  "login": "/"
}

```

Реєстрація успішна, ми отримали інформацію про аккаунт та посилання на авторизацію. Тепер спробуємо авторизуватись (kyrylo.sidak20@yopmail.com: IP11Sidak2):

```

{
  "user": {
    "sub": "auth0|66364bc917f99a74d3a3bd1c",
    "nickname": "Sidak20",
    "name": "Kyrylo",
    "picture": "https://i.pinimg.com/originals/e1/4c/ae/e14cae2f0f44121ab4e3506002ba1a55.jpg",
    "updated_at": "2024-05-04T15:07:13.856Z",
    "email": "kyrylo.sidak20@yopmail.com",
    "email_verified": false
  },
  "logout": "/logout"
}

```

Фінальним кроком є додання функції refreshToken:

```

async function refreshToken(req, res, next) {
  const tokens = req.session.tokens;
  if (tokens?.expires_in) {
    const expirationDate = new Date(tokens.expires_in);
    console.log(`Token expires on: ${expirationDate.toLocaleString()}`);
  }
  if (tokens && Date.now() > tokens.expires_in - 5 * 60 * 1000) {
    try {
      const response = await axios({
        method: 'post',
        url: `https://${process.env.AUTH0_DOMAIN}/oauth/token`,
        headers: { 'content-type': 'application/x-www-form-urlencoded' },
        data: new URLSearchParams({
          grant_type: 'refresh_token',
          client_id: process.env.AUTH0_CLIENT_ID,

```

```

        client_secret: process.env.AUTH0_CLIENT_SECRET,
        refresh_token: tokens.refresh_token,
    }},
  });

  req.session.tokens = {
    access_token: response.data.access_token,
    refresh_token: tokens.refresh_token,
    expires_in: Date.now() + response.data.expires_in * 1000
  };
} catch (error) {
  console.error('Error refreshing token:', error.response?.data ||
error.message);
  return res.status(401).json({ error: 'Failed to refresh token' });
}
}
next();
}

app.use(refreshToken);

```

При виконанні будь запитів спочатку виконується перевірка, якщо до кінця дії токена залишилось 5 хв, то токен оновлюється. Також додано вивід в консоль до якої дійсний токен:

```

Example app listening on port 3000
Token expires on: 05.05.2024, 18:17:50

```

## 5 ВИСНОВОК

В результаті лабораторної роботи, було оновлено приклад авторизації `token_auth`. Запити для авторизації перенесенні з `poster` у додаток. Тепер форма надсилає запит до `auth0`. Авторизація та вихід з аккаунту були успішно перевірені. Для спрощення було прибрано візуал.

Під час виконання додаткового завдання, функціонал додатку було розширено новим ендпоінтом для реєстрації та функцією перевірки дійсності токена. Також на головну сторінку була додана форма для реєстрації.