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 №6 з дисципліни
«Системи безпеки програм і даних»

«OAuth2»

Виконав(ла) *ІП-11 Сідак Кирил Ігорович*
(шифр, прізвище, ім'я, по батькові)

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

Київ 2024

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – засвоювання базових навичок роботи з OAuth2 протоколом.

2 ЗАВДАННЯ

2.1 Основне завдання

Розширити Лабораторну роботу 4, змінивши логін сторінку на стандартну від SSO провайдера, для цього, треба зробити редірект на API_DOMAIN https://kpi.eu.auth0.com/authorize та додатково додати параметри Вашого аплікейшена client_id, redirect_uri, response_type=code, response_mode=query

https://kpi.eu.auth0.com/authorize?client_id=JIvCO5c2IBHlAe2patn6l6q5H35qxti0&r

edirect_uri=http%3A%2F%2Flocalhost%3A3000&response_type=code&response_mo de=query

Надати код рішення.

2.2 Додаткове завдання

Додатково розшири аплікайшен обробкою редіректа та отриманням юзер токена за допомогою code grant type. https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow

3 ВИКОНАННЯ ОСНОВНОГО ЗАВДАННЯ

Для виконання роботи було обрано варіант створення нового ендпоінту який відповідає за редірект на сторінку авторизації. Такий спосіб було обрано для подальшого виконання додаткового завдання:

```
require('dotenv').config()
const express = require('express')
const axios = require('axios')
const path = require('path')
const { auth } = require('express-oauth2-jwt-bearer')

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

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

const checkJwt = auth({
  audience: process.env.AUDIENCE,
  issuerBaseURL: `https://${process.env.AUTH0_DOMAIN}/`,
})

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

app.get('/login', (req, 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('/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 })
```

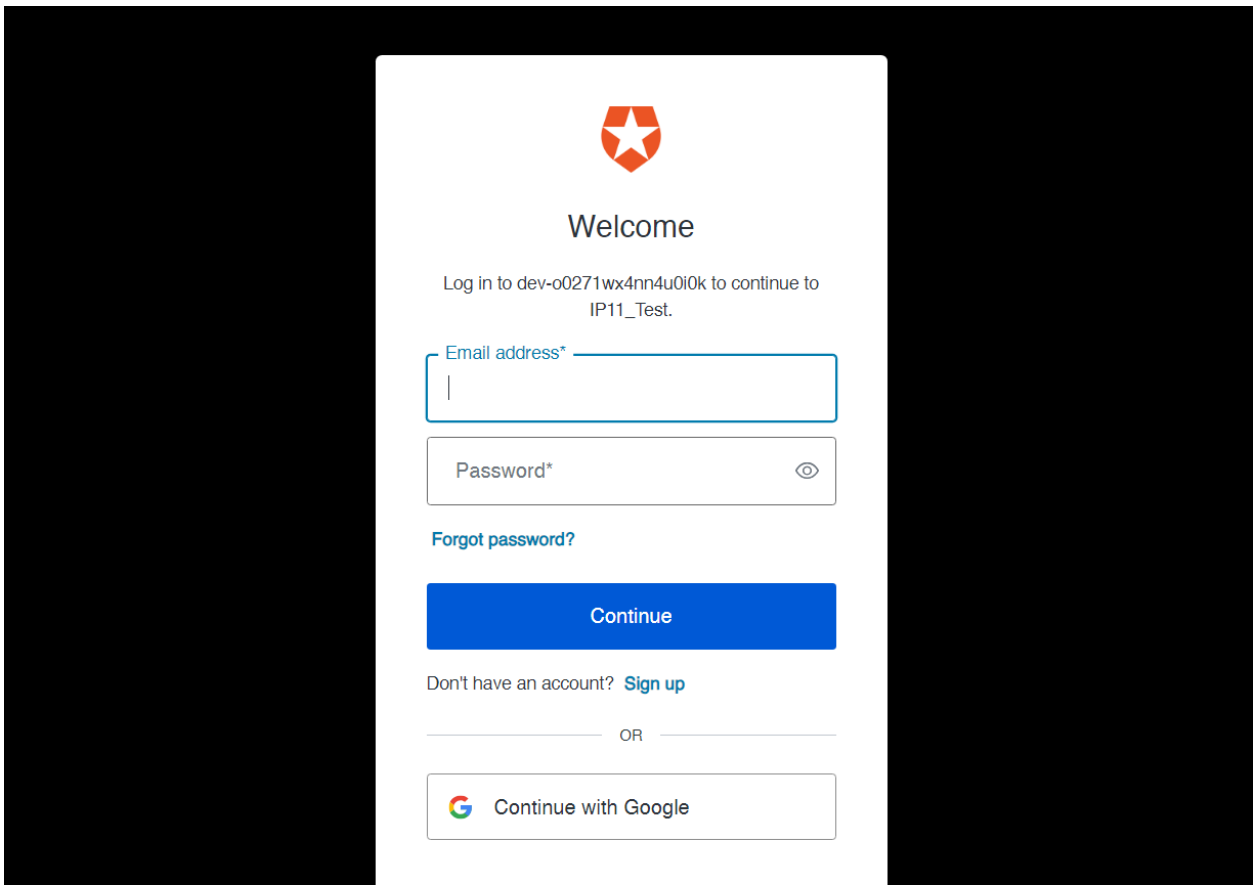
```
  } catch (e) {
    console.log(e)
  }
})

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',
      }),
    })

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

  try {
    // Получение токена
    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' },
      }
    )

    // Запрос регистрации
    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,
    })
  }
})

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

В хтмл було додано на редірект, якщо токен не встановлений:

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

Запускаємо код та перевіряємо:



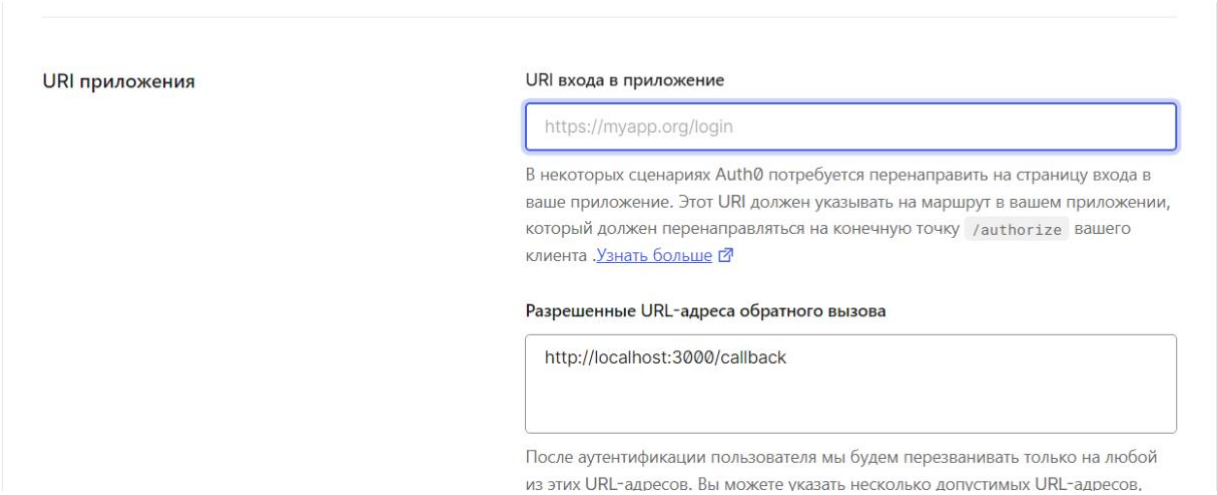
В результаті ми отримали редірект на сторінку авторизації від провайдера.

4 ВИКОНАННЯ ДОДАТКОВОГО ЗАВДАННЯ

Для виконання додаткового завдання, до основого коду додано новий ендпоінт callback, який при успішній авторизації надсилає токен в параметрах запиту на головну сторінку. Код ендпоінту:

```
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,
        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')
  }
})
```

Для того щоб callback працював, переходимо на сторінку налаштувань нашого додатку, та додаємо його в список калбеків:



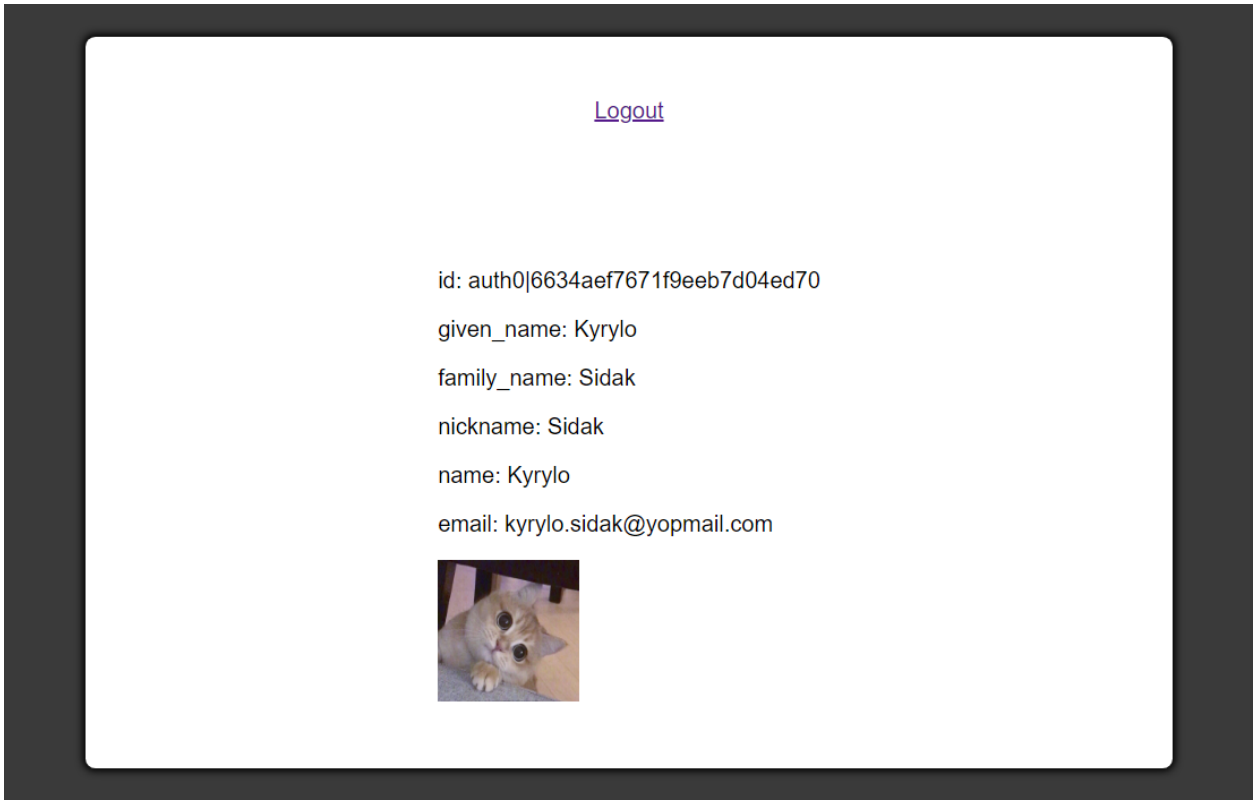
Тепер прибираємо зайве з index.html (форми реєстрації та авторизації) та додаємо обробку токену з параметрів. Якщо токен присутній, то зберігаємо його в sessionstorage, та прибираємо з параметрів в адресній строфі вікна. Інша обробка залишається як і була. Оновлений код:

```
<!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>
    <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;
      }
    </style>
  </head>

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

```
  <script>
    const logoutLink = document.getElementById('logout')

    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 div = document.createElement('div')
            div.innerHTML = `
                <p>id: ${user.sub}</p>
                <p>given_name: ${user.given_name}</p>
                <p>family_name: ${user.family_name}</p>
                <p>nickname: ${user.nickname}</p>
                <p>name: ${user.name}</p>
                <p>email: ${user.email}</p>
                <img src='${user.picture}' alt='Profile pic' width="100"
height="100"/>
            `
            mainHolder.appendChild(div)

            logoutLink.style.opacity = 1
```

```
        }
      })
  }

  logoutLink.addEventListener('click', (e) => {
    e.preventDefault()
    sessionStorage.removeItem('session')
    location.reload()
  })
</script>
</html>
```

В результаті, після авторизації ми бачимо інформацію про користувача, яка залишиться після перезавантаження:



5 ВИСНОВОК

Під час виконання лабораторної роботи, було змінено авторизаційну сторінку на сторінку авторизації провайдеру за допомогою редіректу. Також було прибрано зайвий код з фронтенду, який відповідав за відображення форм.

Виконавши додаткове завдання, було додано новий ендпоінт для обробки callback. Після авторизації сайт знову відображає всю інформацію про користувача.