

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

Лабораторна робота №1
з дисципліни
«Безпека програмного забезпечення»

Виконав:
студент групи ІП-05
Гапій Денис Едуардович

Перевірив:
Іваніщев Б. В.

Київ 2023

Мета: «Роздивитись основні методи авторизації»

Завдання 1: basic_auth

Лістинг index.js:

```
const express = require('express')
const app = express()
const port = 3000
app.use((req, res, next) => {
  console.log('\n=====');
  const authorizationHeader = req.get('Authorization');
  console.log('authorizationHeader', authorizationHeader);
  if (!authorizationHeader) {
    res.setHeader('WWW-Authenticate', 'Basic realm="Ukraine"');
    res.status(401);
    res.send('Unauthorized');
    return;
  }
  const authorizationBase64Part = authorizationHeader.split(' ')[1];

  const decodedAuthorizationHeader = Buffer.from(authorizationBase64Part,
'base64').toString('utf-8');
  console.log('decodedAuthorizationHeader', decodedAuthorizationHeader);

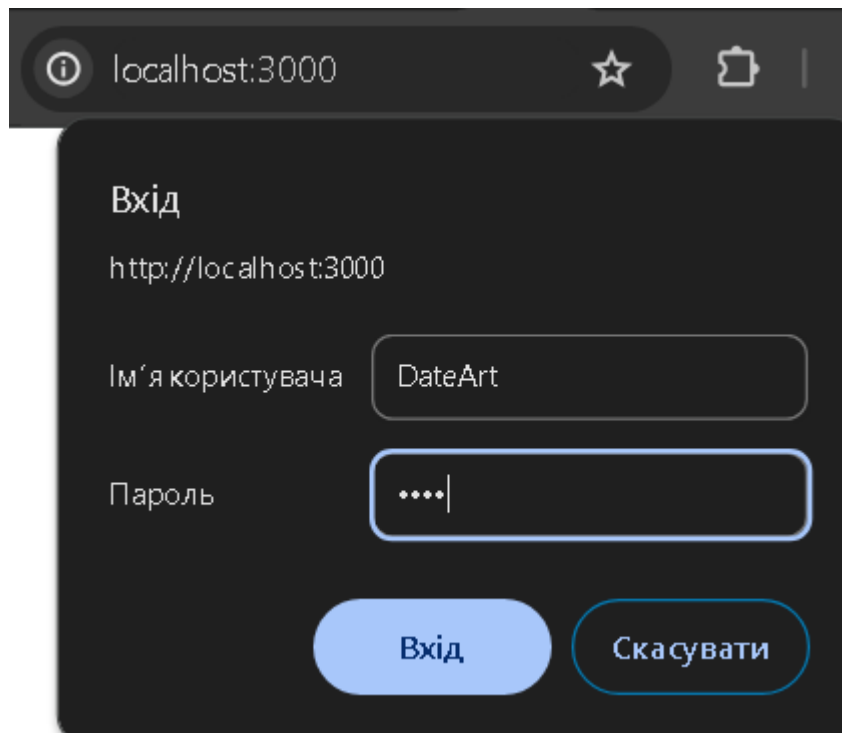
  const login = decodedAuthorizationHeader.split(':')[0];
  const password = decodedAuthorizationHeader.split(':')[1];
  console.log('Login/Password', login, password);

  if (login == 'DateArt' && password == '2408') {
    req.login = login;
    return next();
  }
  res.setHeader('WWW-Authenticate', 'Basic realm="Ukraine"');
  res.status(401);
  res.send('Unauthorized');
});
app.get('/', (req, res) => {
  res.send(`Hello ${req.login}`);
})
app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Страт локального серверу та прослуховування на 3000 порті:

```
PS S:\Dev\Studying\KPI-Studying\7th semester\Software Security\lab1> cd .\basic_auth\
PS S:\Dev\Studying\KPI-Studying\7th semester\Software Security\lab1\basic_auth> node .\index.js
● Example app listening on port 3000
```

У вікні вводимо 'коректні' ім'я користувача та пароль:



Вхід

http://localhost:3000

Ім'я користувача

Пароль

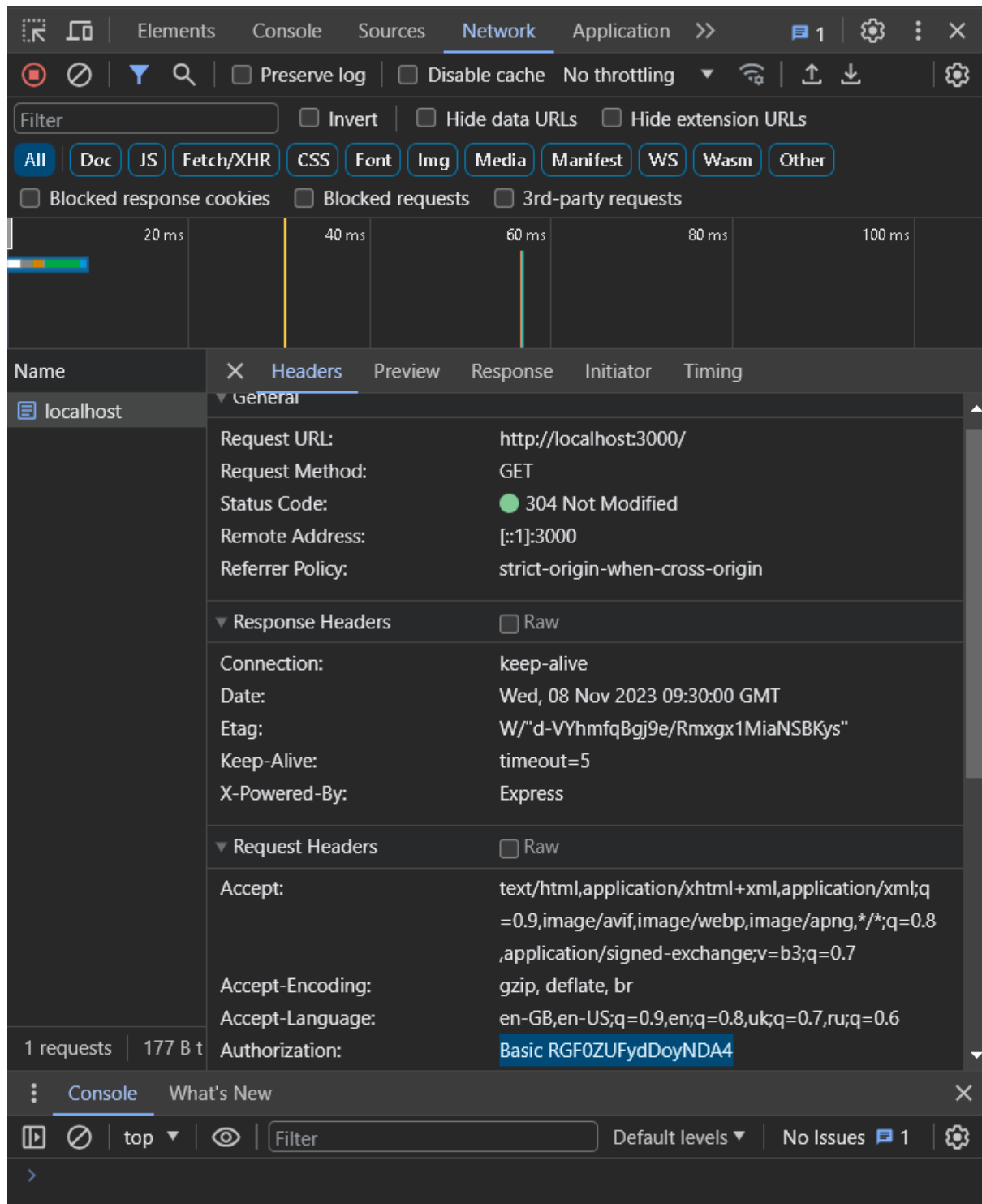
При натисканні на 'Вхід' в консоль виводиться заголовок аутентифікації для конкретного користувача:

```
=====
[]
authorizationHeader undefined

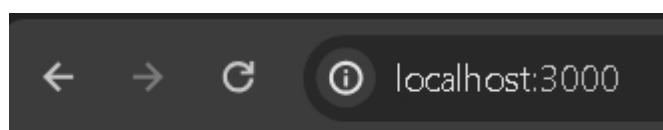
=====

authorizationHeader Basic RGF0ZUFydDoyMDA4
decodedAuthorizationHeader DateArt:2408
Login/Password DateArt 2408
```

З браузера можна звірити успішність авторизації, та порівняти заголовки:

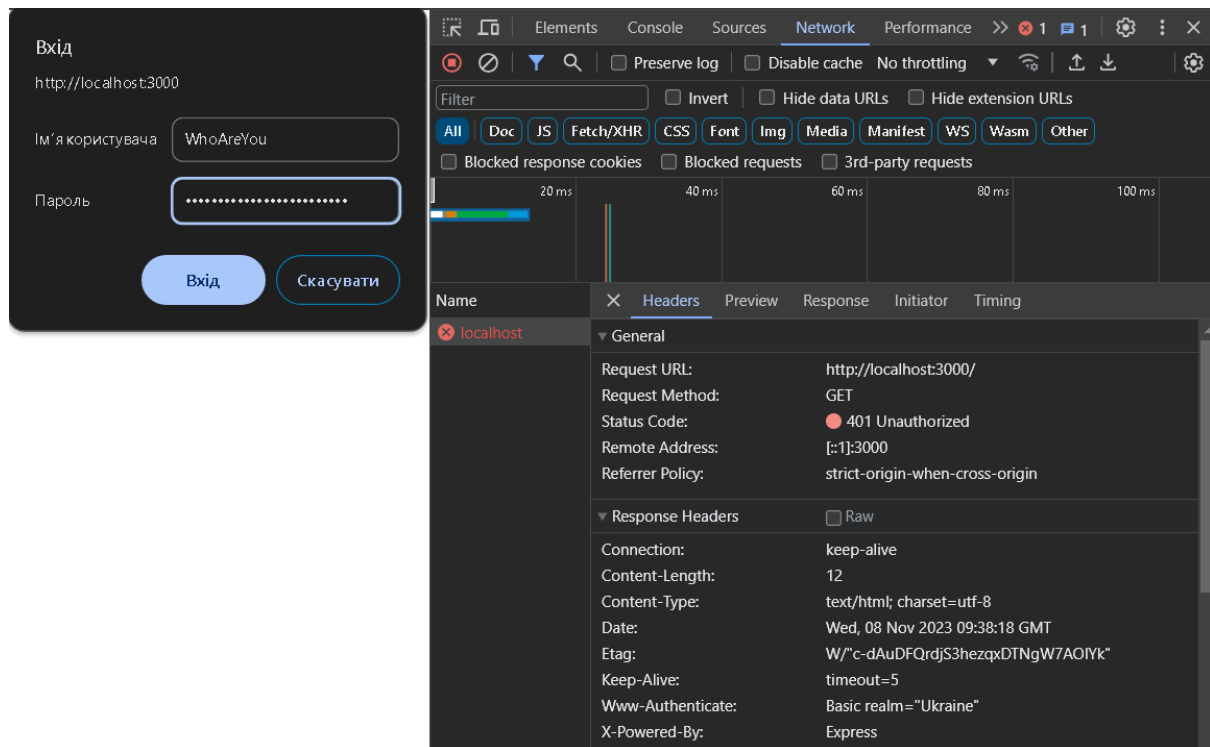


Представлення сайту вітає нашого користувача з успішним входом:



Hello DateArt

Натомість, якщо ми будемо вводити 'некоректні дані' то такого повідомлення не отримаємо:



Завдання 2: forms_auth

Лістинг index.js:

```
const uuid = require('uuid');
const express = require('express');
const cookieParser = require('cookie-parser');
const onFinished = require('on-finished');
const bodyParser = require('body-parser');
const path = require('path');
const port = 3000;
const fs = require('fs');
const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(cookieParser());
const SESSION_KEY = 'session';
class Session {
  #sessions = {}
  constructor() {
    try {
      this.#sessions = fs.readFileSync('./sessions.json', 'utf8');
      this.#sessions = JSON.parse(this.#sessions.trim());
      console.log(this.#sessions);
    } catch (e) {
      this.#sessions = {};
    }
  }
}
```

```

    #storeSessions() {
      fs.writeFileSync('./sessions.json', JSON.stringify(this.#sessions),
'utf-8');
    }
    set(key, value) {
      if (!value) {
        value = {};
      }
      this.#sessions[key] = value;
      this.#storeSessions();
    }
    get(key) {
      return this.#sessions[key];
    }

    init(res) {
      const sessionId = uuid.v4();
      res.set('Set-Cookie', `${SESSION_KEY}=${sessionId}; HttpOnly`);
      this.set(sessionId);

      return sessionId;
    }
    destroy(req, res) {
      const sessionId = req.sessionId;
      delete this.#sessions[sessionId];
      this.#storeSessions();
      res.set('Set-Cookie', `${SESSION_KEY}=; HttpOnly`);
    }
  }
}
const sessions = new Session();
app.use((req, res, next) => {
  let currentSession = {};
  let sessionId;

  if (req.cookies[SESSION_KEY]) {
    sessionId = req.cookies[SESSION_KEY];
    currentSession = sessions.get(sessionId);
    if (!currentSession) {
      currentSession = {};
      sessionId = sessions.init(res);
    }
  } else {
    sessionId = sessions.init(res);
  }
  req.session = currentSession;
  req.sessionId = sessionId;
  onFinish(req, () => {
    const currentSession = req.session;
    const sessionId = req.sessionId;
    sessions.set(sessionId, currentSession);
  });
  next();
});
app.get('/', (req, res) => {
  console.log(req.session);
});

```

```

    if (req.session.username) {
      return res.json({
        username: req.session.username,
        logout: 'http://localhost:3000/logout'
      })
    }
    res.sendFile(path.join(__dirname+'/index.html'));
  })
  app.get('/logout', (req, res) => {
    sessions.destroy(req, res);
    res.redirect('/');
  });
  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) => {
      if (user.login == login && user.password == password) {
        return true;
      }
      return false
    });
    if (user) {
      req.session.username = user.username;
      req.session.login = user.login;

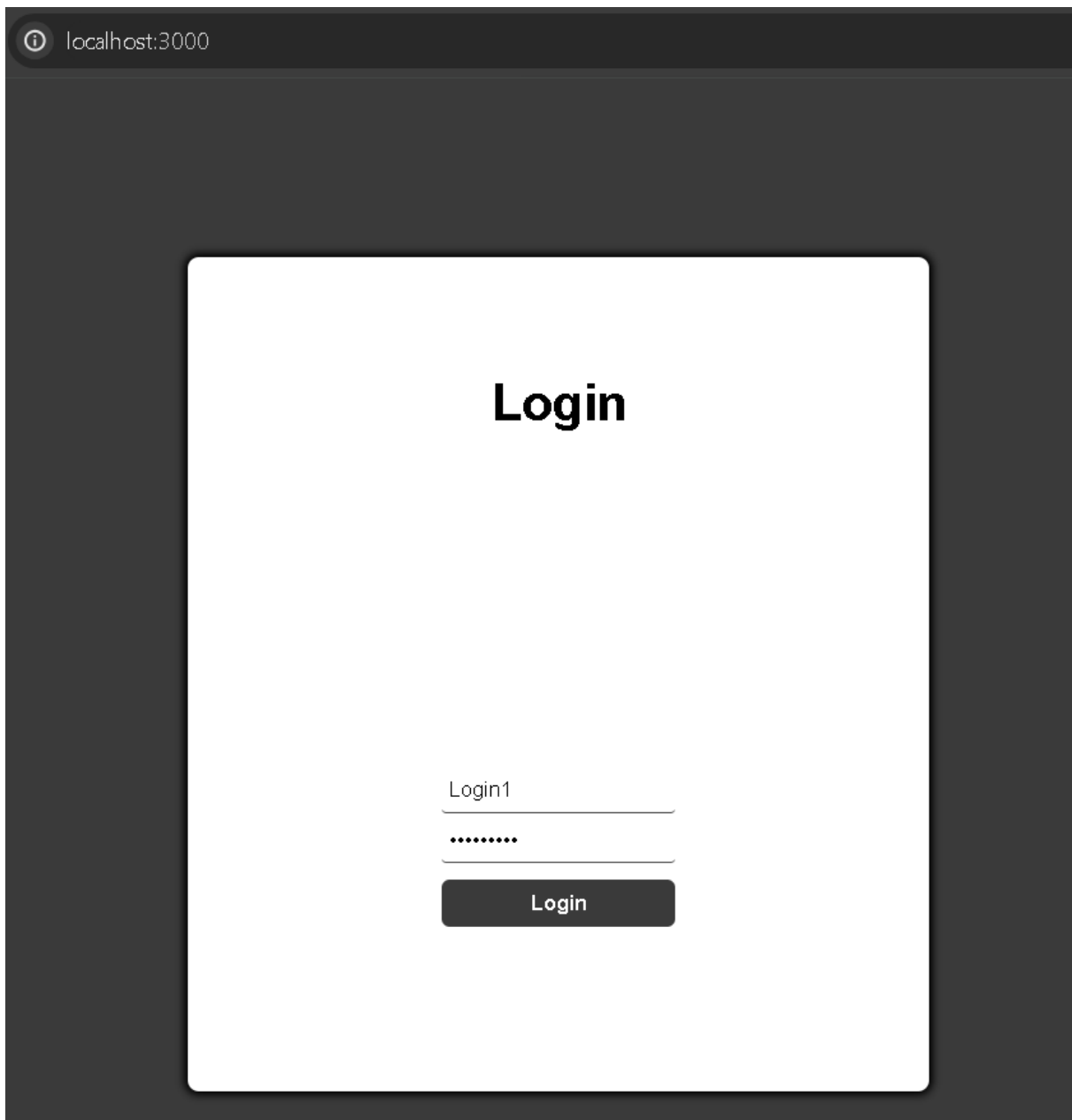
      res.json({ username: login });
    }
    res.status(401).send();
  });

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

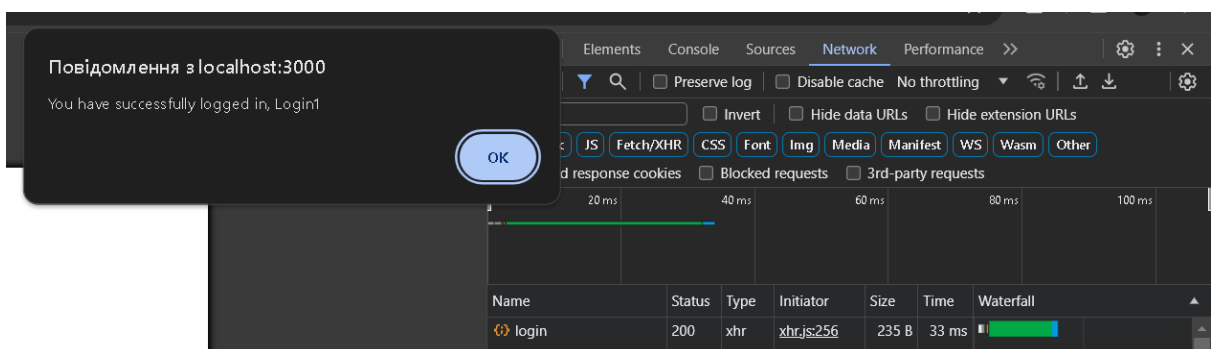
```

Старт локального серверу та прослуховування на 3000 порті.

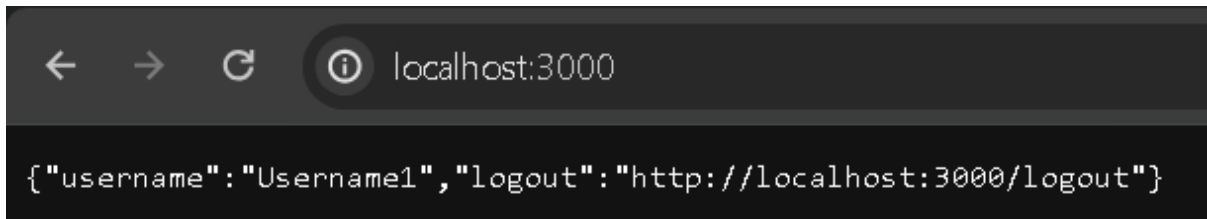
У вікні-формі вводимо 'коректні' ім'я користувача та пароль:



Після отримуємо повідомлення про успішний вхід:



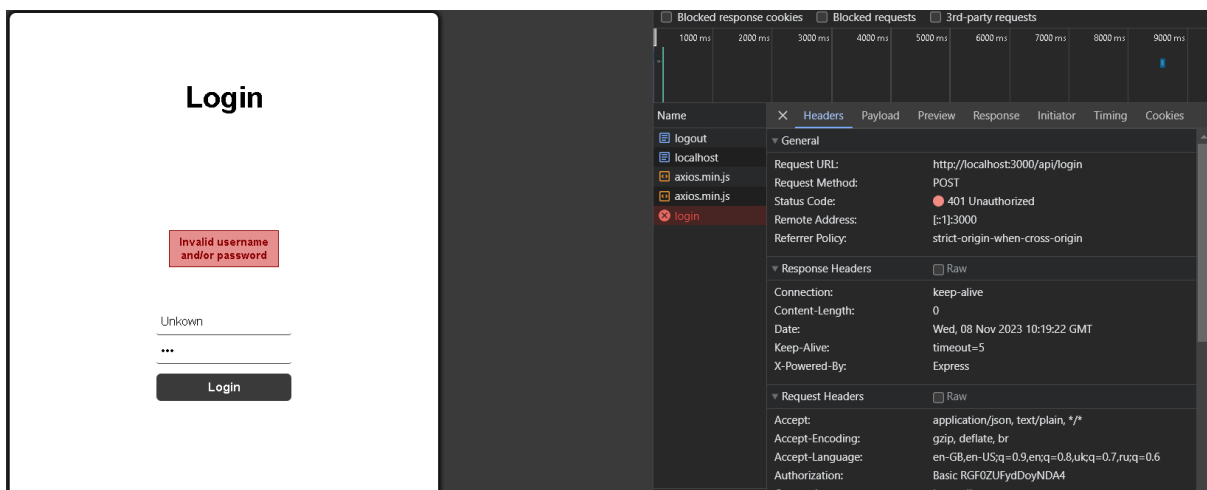
Після підтвердження отримуємо ім'я користувача та посилання на вихід з профілю у форматі json:



У консолі терміналу, в свою чергу, можемо спостерігати етапи перед входом, сам вхід, та вихід, що супроводжується найменуванням користувача та його логіном:

```
● Example app listening on port 3000
{}
{ username: 'Username1', login: 'Login1' }
{}
{ username: 'Username', login: 'Login' }
[]
```

У випадку введення некоректних даних для входу, отримуємо помилку:



Усі сесії зберігаються у файлі sessions.json:

```
JS index.js  {} sessions.json x
lab1 > forms_auth > {} sessions.json > ...
1 [{"9d3add08-80d5-4c46-941b-82b8c896df4f":{"username":"Username1",
  "login":"Login1"},"e10ccc39-154e-40e6-8656-7bbdea9a1263":{"username":"Username",
  "login":"Login"},"0d9b25a2-ecda-43b8-81ad-bb61c349889d":{}}]
```

Завдання 3: token_auth

Лістинг index.js:

```
const uuid = require('uuid');
const express = require('express');
const onFinished = require('on-finished');
const bodyParser = require('body-parser');
```

```

const path = require('path');
const port = 3000;
const fs = require('fs');

const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
const SESSION_KEY = 'Authorization';
class Session {
  #sessions = {}
  constructor() {
    try {
      this.#sessions = fs.readFileSync('./sessions.json', 'utf8');
      this.#sessions = JSON.parse(this.#sessions.trim());

      console.log(this.#sessions);
    } catch (e) {
      this.#sessions = {};
    }
  }
  #storeSessions() {
    fs.writeFileSync('./sessions.json', JSON.stringify(this.#sessions),
'utf-8');
  }

  set(key, value) {
    if (!value) {
      value = {};
    }
    this.#sessions[key] = value;
    this.#storeSessions();
  }

  get(key) {
    return this.#sessions[key];
  }

  init(res) {
    const sessionId = uuid.v4();
    this.set(sessionId);

    return sessionId;
  }
  destroy(req, res) {
    const sessionId = req.sessionId;
    delete this.#sessions[sessionId];
    this.#storeSessions();
  }
}
const sessions = new Session();
app.use((req, res, next) => {
  let currentSession = {};
  let sessionId = req.get(SESSION_KEY);

  if (sessionId) {
    currentSession = sessions.get(sessionId);
  }
});

```

```

        if (!currentSession) {
            currentSession = {};
            sessionId = sessions.init(res);
        }
    } else {
        sessionId = sessions.init(res);
    }
    req.session = currentSession;
    req.sessionId = sessionId;
    onFinish(req, () => {
        const currentSession = req.session;
        const sessionId = req.sessionId;
        sessions.set(sessionId, currentSession);
    });
    next();
});

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

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

const users = [
    {
        login: 'Login',
        password: 'Password',
        username: 'Username',
    },
    {
        login: 'Denys',
        password: 'Password1',
        username: 'Hapii',
    }
]

app.post('/api/login', (req, res) => {
    const { login, password } = req.body;

    const user = users.find((user) => {
        if (user.login == login && user.password == password) {
            return true;
        }
        return false
    });
});

if (user) {

```

```

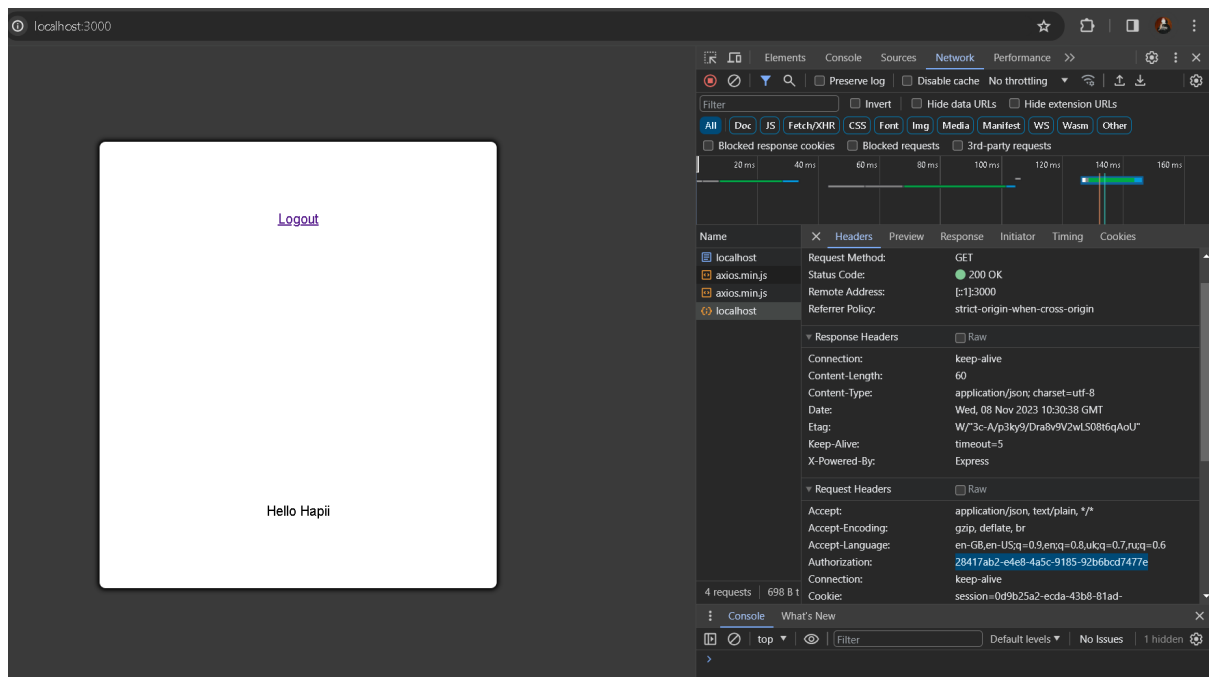
    req.session.username = user.username;
    req.session.login = user.login;

    res.json({ token: req.sessionId });
  }

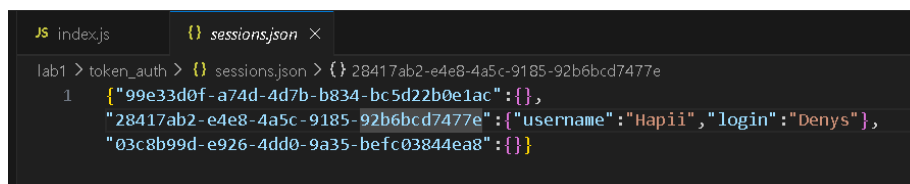
  res.status(401).send();
});
app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})

```

Повторюємо кроки з попередньої реалізації. Успішна авторизація:



Збереження сесій (в якому видно що 'номер' авторизації користувача Нарії співпадає):



Завдання 4: jwt_auth

Лістинг index.js:

```

const jwt = require('jsonwebtoken')
const uuid = require('uuid');
const express = require('express');
const onFinished = require('on-finished');
const bodyParser = require('body-parser');
const path = require('path');

```

```

const port = 3000;
const fs = require('fs');
const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
const indexHtmlPath = path.join(__dirname + "/index.html");
const sessionKey = 'authorization';
const secretKey = 'secret_jwt';
class Session {
  #sessions = {}

  constructor() {
    try {
      this.#sessions = fs.readFileSync('./sessions.json', 'utf8');
      this.#sessions = JSON.parse(this.#sessions.trim());

      console.log(this.#sessions);
    } catch (e) {
      this.#sessions = {};
    }
  }

  #storeSessions() {
    fs.writeFileSync('./sessions.json', JSON.stringify(this.#sessions),
'utf-8');
  }

  set(key, value) {
    if (!value) {
      value = {};
    }
    this.#sessions[key] = value;
    this.#storeSessions();
  }

  get(key) {
    return this.#sessions[key];
  }

  init(res) {
    const sessionId = uuid.v4();
    this.set(sessionId);

    return sessionId;
  }

  destroy(req, res) {
    const sessionId = req.sessionId;
    delete this.#sessions[sessionId];
    this.#storeSessions();
  }
}

const sessions = new Session();
app.use((req, res, next) => {
  let currentSession = {};
  let sessionId = req.get(sessionKey);

  if (sessionId) {

```

```

        currentSession = sessions.get(sessionId);
        if (!currentSession) {
            currentSession = {};
            sessionId = sessions.init(res);
        }
    } else {
        sessionId = sessions.init(res);
    }
    req.session = currentSession;
    req.sessionId = sessionId;
    onFinish(req, () => {
        const currentSession = req.session;
        const sessionId = req.sessionId;
        sessions.set(sessionId, currentSession);
    });
    next();
});
app.get('/', (req, res) => {
    try {
        let token = req.headers.authorization?.replace("Bearer ", "");

        const tokenPayload = jwt.verify(token, secretKey);

        const user = users.find(({ login }) => login === tokenPayload.login);

        if (user) {
            return res.json({
                username: user.username,
                logout: "http://localhost:3000/logout",
            });
        }
    } catch (e) {
        res.status(401);
    }

    res.sendFile(indexHtmlPath);
});

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

const users = [
    {
        login: 'Login',
        password: 'Password',
        username: 'Username',
    },
    {
        login: 'Denys',
        password: 'Password1',
        username: 'Hapii',
    }
]

app.post('/api/login', (req, res) => {

```

```

const { login, password } = req.body;

const user = users.find((user) => {
  if (user.login == login && user.password == password) {
    return true;
  }
  return false
});

if (user) {
  req.session.username = user.username;
  req.session.login = user.login;

  const token = jwt.sign({ login },
    secretKey,
    { expiresIn: "10m" });

  console.log(token);
  return res.json({ token: token });
}

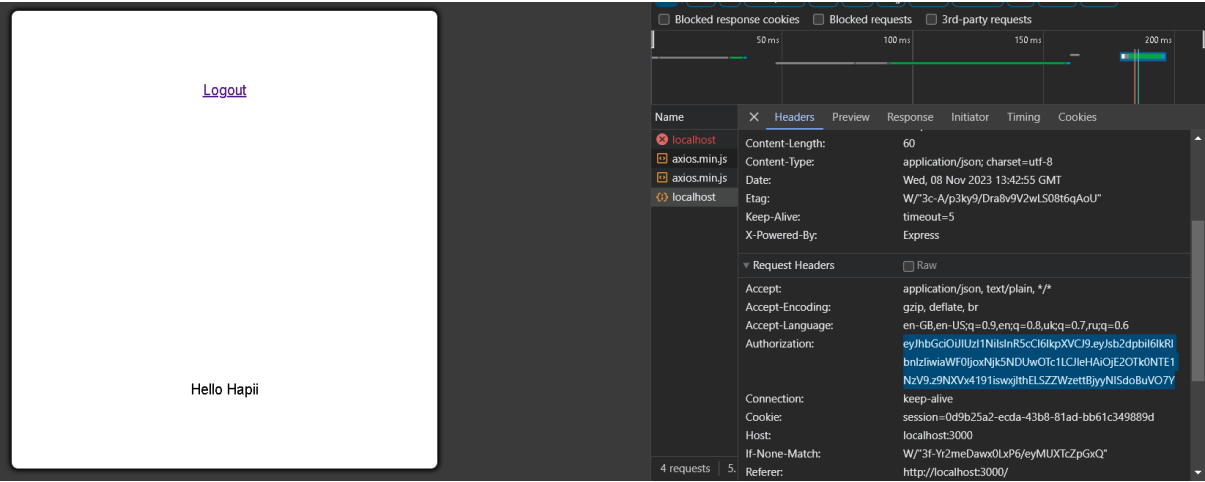
res.status(401).send();
});
app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})

```

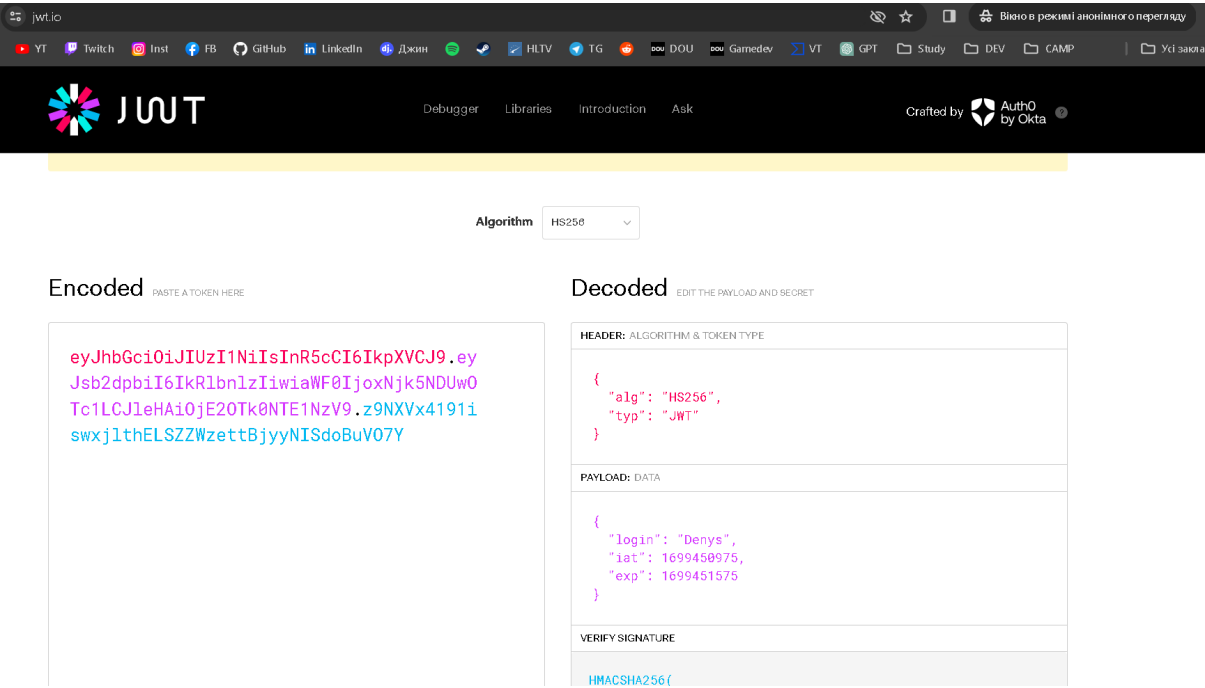
Зайдемо як другий користувач:



Успішна авторизація з jwt як токен:



Перевірка декодування на платформі jwt.io:



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

Лабораторна робота №2
з дисципліни
«Безпека програмного забезпечення»

Виконав:
студент групи ІП-05
Гапій Денис Едуардович

Перевірив:
Іваніщев Б. В.

Київ 2023

Перевірка валідності токєну:

[illegible]

Завдання 2: Створити юзера з власним email в системі

Лістинг create-user.js:

```
const request = require("request-promise");
const dotenv = require("dotenv");
dotenv.config();
async function createUser() {
  const raw = {
    email: "gapiy.denis@gmail.com",
    user_metadata: {},
    blocked: false,
    email_verified: false,
    app_metadata: {},
    given_name: "Gapiyka",
    family_name: "Gapiyka",
    name: "Gapiyka",
    nickname: "Gapiyka",
    picture:
      "https://unity.com/sites/default/files/styles/social_media_sharing/public/2022-02/U_Logo_White_CMYK.jpg",
    user_id: "5123121",
    connection: "Username-Password-Authentication",
    password: "gap41K!mamut",
    verify_email: false,
  };
  const requestOptions = {
    method: 'POST',
    url: `${process.env.AUTH0_URL}/api/v2/users`,
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${process.env.AUTH0_TOKEN}`,
    },
  };
}
```

```

    },
    body: JSON.stringify(raw),
  });
  try {
    const response = await request(requestOptions);
    console.log(response);
  } catch (error) {
    console.error("Error creating user:", error.message);
  }
}

createUser();

```

Створення користувача:

```

PS S:\Dev\Studying\KPI-Studying\7th semester\Software Security\lab2> node .\app\create-user.js
{"blocked":false,"created_at":"2023-11-08T14:56:22.411Z","email":"gapiy.denis@gmail.com","email_verified":false,"family_name":"Gapiyka","given_name":"Gapiyka","identities":[{"user_id":"5123121","connection":"Username-Password-Authentication","provider":"auth0","isSocial":false}],,"name":"Gapiyka","nickname":"Gapiyka","picture":"https://unity.com/sites/default/files/styles/social_media_sharing/public/2022-02/U_Logo_White_CMYK.jpg","updated_at":"2023-11-08T14:56:22.412Z","user_id":"auth0|5123121","user_metadata":{}}

```

Завдання 3: зробити власний акаунт в auth0

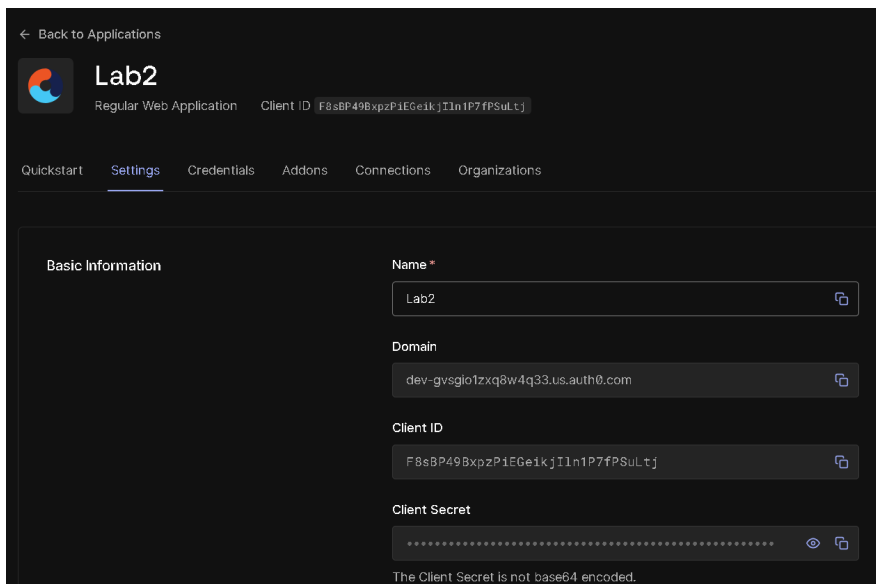
Оновлений лістинг .env:

```

AUTH0_URL=https://dev-gvsgio1zxq8w4q33.us.auth0.com
AUTH0_TOKEN=
AUTH0_CLIENT_ID=F8sBP49BxpzPiEGeikjIln1P7fPSuLtj
AUTH0_CLIENT_SECRET=L5U3ifZocZxrRlH0km3aJBqabREZYhBQHihfBdgiQj0K0cQqdFsPCzarPB
Uzh5Rh

```

Власний application:



Доступи в API:

Lab2

Authorized ☒

Client Id: `F8sBP49BxpzPiEGeikjI1n1P7fPSuLtlj`

Select which permissions (scopes) should be granted to this client:

Grant ID

`cgr_9EWI72QZbeuxNFdC`

Permissions

Select: All None

☐ read:client_grants

☐ create:client_grants

☐ delete:client_grants

☐ update:client_grants

☒ read:users

☒ update:users

☐ delete:users

☒ create:users

☐ read:users_app_metadata

☐ update:users_app_metadata

[illegible]

```
PS S:\Dev\Studying\KPI-Studying\7th semester\Software Security\lab2> node .\app\create-user.js
{"blocked":false,"created_at":"2023-11-08T15:48:40.547Z","email":"gapiy.denise@gmail.com","email_verified":false,"family_name":"Gapiyka","given_name":"Gapiyka","identities":[{"user_id":"5123121","connection":{"Username-Password-Authentication","provider":"auth","isSocial":false},"name":"Gapiyka","nickname":"Gapiyka","picture":"https://unity.com/sites/default/files/styles/social_media_sharing/public/2022-02/U_Logo_White_CMYK.jpg"},"updated_at":"2023-11-08T15:48:40.547Z","user_id":["auth","5123121"],"user_metadata":{}}
```

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

Лабораторна робота №3
з дисципліни
«Безпека програмного забезпечення»

Виконав:
студент групи ІП-05
Гапій Денис Едуардович

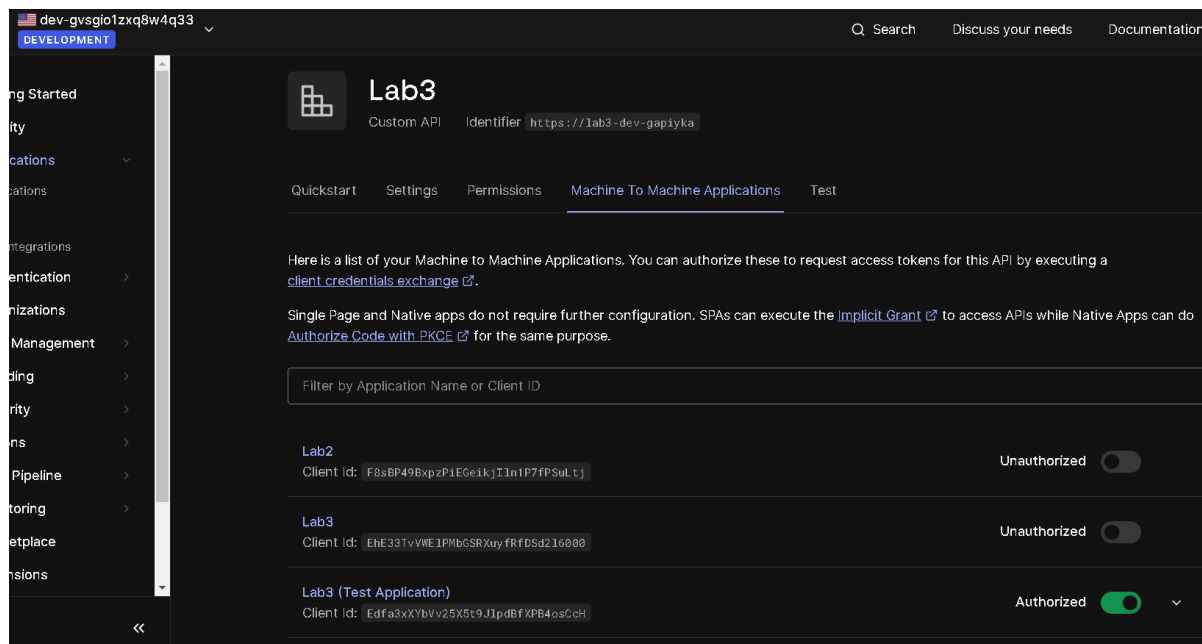
Перевірив:
Іваніщев Б. В.

Київ 2023

Мета: «Засвоєння базових навичок OAuth2 авторизаційного протокола»

Завдання 1: зробити запит на отримання user token

Створені API (для offline_access) та application для запитів:



Лістинг main.js:

```
const request = require("request");
const dotenv = require("dotenv");
dotenv.config();

const options = {
  method: 'POST',
  url: `${process.env.AUTH0_URL}/oauth/token`,
  headers: { 'content-type': 'application/json' },
  form: {
    client_id: process.env.AUTH0_CLIENT_ID,
    client_secret: process.env.AUTH0_CLIENT_SECRET,
    code: process.env.AUTH0_CODE,
    audience: `${process.env.AUTH0_AUDIENCE}`,
    grant_type: "authorization_code",
    redirect_uri: "https://www.google.com/",
    scope: "offline_access"
  }
};
```

```
request(options, function (error, response, body) {
  if (error) throw new Error(error);

  console.log(body);
});
```

Отримання токетну з kpi.eu.auth0.com

[illegible]

Лістинг .env для сховних ключів / токенів / посилань:

```
AUTH0_URL=https://dev-gvsgio1zxq8w4q33.us.auth0.com  
AUTH0_AUDIENCE=https://lab3-dev-gapiyka  
AUTH0_TOKEN=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9IAoIMPZCmVxjeHTEphVV9pcHJDZXFiRLlVMjS9eyJpc3MiOiJodHRwcovL2Rldi1ndnNnaW8xenhxOHc0cTMzLnVzMfIdGgwLmNvbS8iLCJzdWIiOiJFVGZhM3hyYWwJdWJ1WDV0OUpscGRGRCzlHQQJRvc0NJSEBjbGlbnRzIiwiaWFkIjoiaHR0cHM6Ly9kZXltYyZ3ZzZ2lvMXp4cTh3NHczMy5lcys5hdXRocMC5jb20vYXBPL3YyLyIsImhhbmE6MTY5OTU1MDDezcwiZWxiOjoxNXkiNSNmNTMtMLCJhenAiOiJFVGZhM3hyYWwJdWJ1WDV0OUpscGRGRCzlHQQRvc0NJSCSIinNjb3BlbiJoicmVhZDpjbjGlbnRfZ3JhbnRzIGNyZF0ZTpjbGlbnRfZ3JhbnRzIGRlbGV0ZTpjbGlbnRfZ3JhbnRzIHVwZGF0ZTpjbGlbnRfZ3JhbnRzIHllYWQ6dXN1cnMgdXBkYXRloNvZZXJzIGRlbGV0ZTpjc2VyYyBjcmVhdGU6dXN1cnMgcmlvdDpjbjGlbnRfY3JlZGVudG1hbHMgY3JlYXRlOmNsawVuDF9jcmVkZW50aWFscyB1cGRhdGU6Y2xpZW50X2NyZWRIbnRpYWxzIGRlbGV0ZTpjbGlbnRfY3JlZGVudG1hbHMiLCJndHKiOiJJbjGlbnQtY3JlZGVudG1hbHMifQ.Ig8XK1fhbbP6b0ie6m2P1p8pxcr7IBA55xwQRJluQXgdgHGaiDIfoY32Z4JET_YiewoQIBIJXkWKhQRJQwmrDCOVQj5deV9AfkSUHLVcqCr8wiJSPG6yIXf0LGNDgsucqeHmocM5bzBs0zDPwkWiGrV1LOZRQBEPKJPmq7VBjBdk0oqEmXspzhU4xBABTon38W4wi0oS0SYK8TG-zcuE2zeEIIVBUrnaG086Ym9WMWDSsh3iT_A0vsqsqiJgf2lxCjUes-IW4Anw93gfpFEtQUAXDuulWEazZnHX1-Sz2STeqg3I_Uh3CuKBdlCSzymmeffuiMgyEG9xpQ6Hh2w  
AUTH0_CLIENT_ID=Edfa3xxYbvV25X5t9JlpdBfxPB4osCcH  
AUTH0_CLIENT_SECRET=6rjAp8BF7I7srePG4qx1Q4U0MZjkT00cf86G1ONRCJL5as1IXv4dfziaQVe0tbk  
AUTH0_CODE=QtyglzmIx_36Qi6JYesnpA80ygEb8rhbsrlIg7siFo5x6  
AUTH0_REFRESH=4fcGPYZ9ba3ir0vfKiVPYC4QxvQRR COSX8FWz-NeYUMZ
```

Посилання для отримання коду:

https://dev-gvsgio1zxq8w4q33.us.auth0.com/authorize?response_type=code&client_id=Edfa3xXYbVv25X5t9JlpdBfXPB4osCch&redirect_uri=https://www.google.com/&scope=offline_access

Отримуємо код в адресі:

← → ↻ 🔍 https://www.google.com/?code=QtglZxMlX_36Qi6JYesNpA80yGEb8rhbsr1lg7slfo5x6

Отриманий рефреш токен після запуску main.js:

Завдання 2: Отримати оновлений токен використовуючи refresh-token grant type

```
const request = require("request");
const dotenv = require("dotenv");
dotenv.config();
const options = {
  method: 'POST',
  url: `${process.env.AUTH0_URL}/oauth/token`,
  headers: { 'content-type': 'application/json' },
  form: {
    refresh_token: process.env.AUTH0_REFRESH,
    client_id: process.env.AUTH0_CLIENT_ID,
    client_secret: process.env.AUTH0_CLIENT_SECRET,
    code: process.env.AUTH0_CODE,
    audience: `${process.env.AUTH0_AUDIENCE}`,
    grant_type: "refresh_token",
    scope: "offline_access"
  }
};

request(options, function (error, response, body) {
  if (error) throw new Error(error);

  console.log(body);
});
```

[illegible]

Завдання 3: зробити запит до API для зміни пароля

```
const request = require("request");
const dotenv = require("dotenv");

dotenv.config();

const options = {
  method: "PATCH",
  url: `${process.env.AUTH0_URL}/api/v2/users/auth0%7C12344443434`,
}
```

```
//auth0|12344443434
headers: {
  "content-type": "application/json",
  authorization: `Bearer ${process.env.AUTH0_TOKEN}`
},
form: {
  "connection": "Username-Password-Authentication",
  "password": "new!Password--1-12313123"
}
};

request(options, function (error, response, body) {
  if (error) throw new Error(error);

  console.log(body);
});
```

Успішна зміна:

```
{ "error": "Invalid grant", "error_description": "Wrong client or password" }
PS S:\Dev\Studying\KPI-Studying\7th semester\Software Security\lab3> node .\app\change-pass.js
{"blocked":false,"created_at":"2023-11-08T18:13:23.404Z","email":"gapiyka@gmail.com","email_verified":false,"family_name":"Gapiyka2","given_name":"Ga
piyka2","identities":[{"user_id":"12344443434","provider":"auth0","connection":"Username-Password-Authentication","isSocial":false}], "name":"Gapiyka2
","nickname":"Gapiyka2","picture":"https://unity.com/sites/default/files/styles/social_media_sharing/public/2022-02/U_Logo_White_CMYK.jpg","updated_a
t":"2023-11-08T19:12:49.813Z","user_id":"auth0|12344443434","user_metadata":{},"last_ip":"77.87.40.220","last_login":"2023-11-08T19:12:49.813Z","logi
ns_count":1}
```

Зміна пароля в історії користувача:

The screenshot shows the Auth0 user management interface. The user profile for 'Gapiyka2' is displayed, with the user ID 'auth0|12344443434'. The 'History' tab is selected, showing a log of events. The log indicates a successful password change a few seconds ago and a failed login attempt 3 minutes ago. The interface includes a sidebar with navigation options like 'Getting Started', 'Activity', 'Applications', 'Authentication', 'Organizations', 'User Management', 'Branding', 'Security', 'Actions', 'Auth Pipeline', 'Monitoring', 'Marketplace', 'Extensions', and 'Settings'. The top right has a search bar and a link to 'Discuss your needs'.

Event	When	App	Identity Provider	From
Success Change Password	a few seconds ago	N/A	Username-Pass...	IP: 77.87.40.220, Kyiv, U
Failed Login (wrong password)	3 minutes ago	Lab3 (Test Appl...	Username-Pass...	IP: 77.87.40.220, Kyiv, U

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

Лабораторна робота №4
з дисципліни
«Безпека програмного забезпечення»

Виконав:
студент групи ІП-05
Гапій Денис Едуардович

Перевірів:
Іваніщев Б. В.

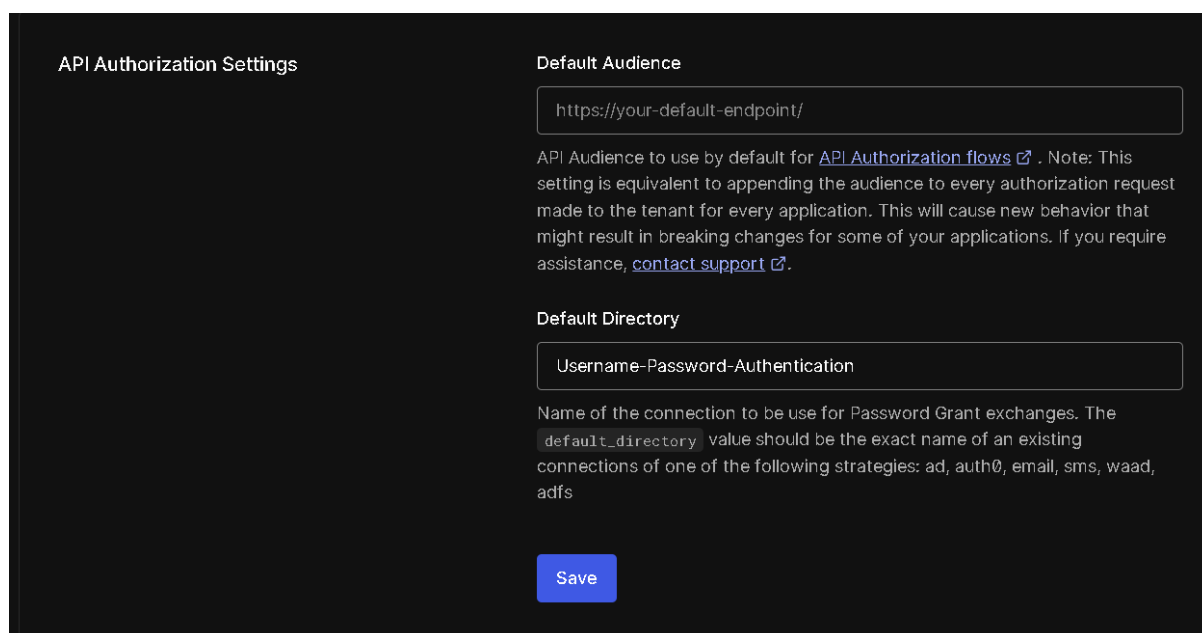
Київ 2023

Мета: «Засвоєння базових навичок OAuth2 авторизаційного протокола»

Завдання 1: Використовуючи наведені налаштування та приведені запитів модифікувати аплікейшн.

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

Перед початком, у налаштуваннях варто додати змінну в 'дефолтну директорію':



API Authorization Settings

Default Audience

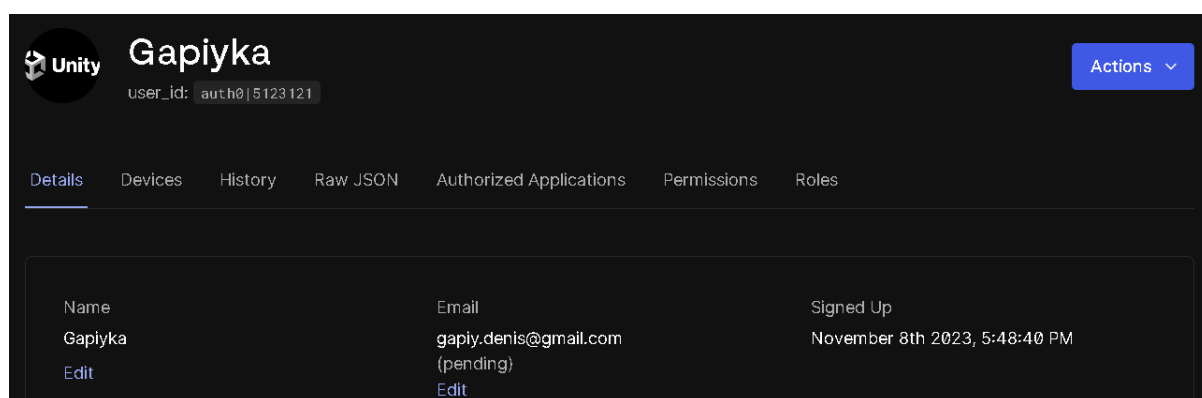
API Audience to use by default for [API Authorization flows](#). Note: This setting is equivalent to appending the audience to every authorization request made to the tenant for every application. This will cause new behavior that might result in breaking changes for some of your applications. If you require assistance, [contact support](#).

Default Directory

Name of the connection to be used for Password Grant exchanges. The `default_directory` value should be the exact name of an existing connection of one of the following strategies: ad, auth0, email, sms, waad, adfs

[Save](#)

Заготовлений користувач на auth0:



Unity **Gapiyka** [Actions](#)

user_id: auth0|5123121

[Details](#) [Devices](#) [History](#) [Raw JSON](#) [Authorized Applications](#) [Permissions](#) [Roles](#)

Name	Email	Signed Up
Gapiyka	gapiy.denis@gmail.com	November 8th 2023, 5:48:40 PM
Edit	(pending)	

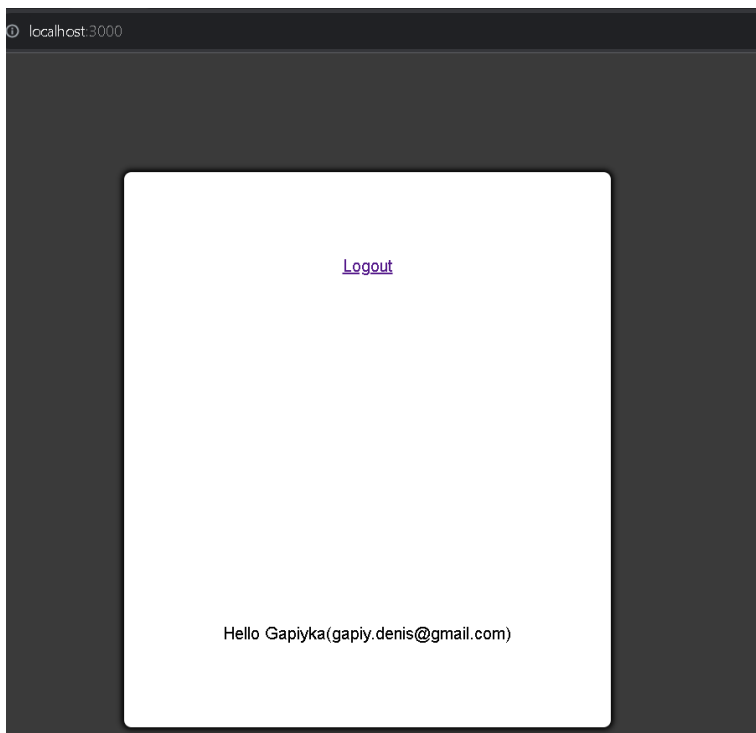
Вікно входу, таке ж як було у auth_token:

Login

Login

[Don't have an account? Sign up](#)

Приклад успішного входу:

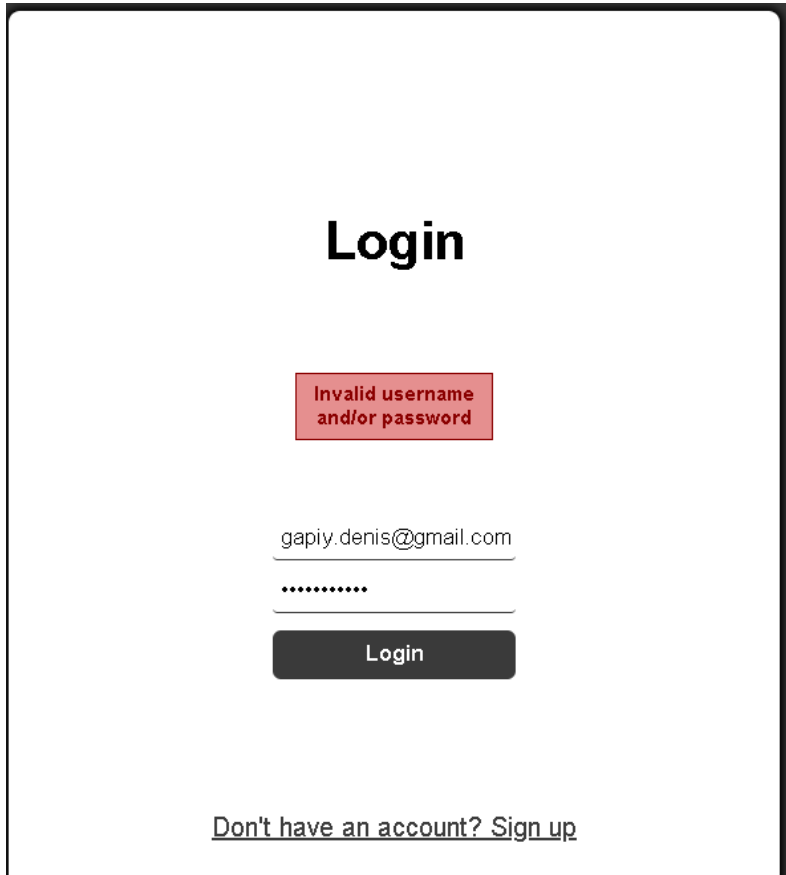


```
}  
User with id auth0|5123121 authorized by Access Token  
[
```

Вихід з акаунту:

```
User with id auth0|5123121 authorized by Access  
User with id auth0|5123121 successfully logout
```

Приклад не успішного входу:



Login

Invalid username
and/or password

gapiy.denis@gmail.com

.....

Login

[Don't have an account? Sign up](#)

```
Error: Auth0 user-token: 403 Forbidden {"error":"invalid_grant","error_description":"Wrong email or password."}  
    at Object.getUserAccessToken (S:\Dev\Studying\KPI-Studying\7th semester\Software Security\lab4\user-token.js:17:15)  
    at process.processTicksAndRejections (node:internal/process/task_queues:95:5)  
    at async S:\Dev\Studying\KPI-Studying\7th semester\Software Security\lab4\index.js:86:13
```

Лістинг index.html - представлення веб сторінки (що є частково модифікованою версією auth_token):

```
<!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">  
    <a href="/logout" id="logout">Logout</a>  
    <h1 id="login-header">Login</h1>  
    <div id="login-error-msg-holder">  
      <p id="login-error-msg" class="login-error-msg">Invalid username  
<span id="error-msg-second-line">and/or
```

```

        password</span></p>
        <p id="login-server-msg" class="login-error-msg"></p>
    </div>
    <form id="login-form" action="/api/login" method="post">
        <input type="text" name="login" id="username-field"
class="login-form-field" placeholder="Login">
        <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 class="redirect-link" id="redirect-link" href="/register">Don't
have an account? Sign up</a>
</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;
}

#login-form-submit:disabled {background-color: #900000;}

.redirect-link {
  color: #3d3d3d;
}
</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 loginSeverMsg = document.getElementById('login-server-msg');
const logoutLink = document.getElementById('logout');
const redirectLink = document.getElementById('redirect-link');
redirectLink.style.display = '';

const hourToRefreshInMSec = 4 * 60 * 60 * 1000;

const toggleLoading = (isLoading) => {
  loginForm.style.display = isLoading ? 'none' : '';
};

let token;
let expiresDate;

```



```

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

if (token) {
  const tokenValidTimeMsec = expiresDate - hourToRefreshInMsec;
  console.log(tokenValidTimeMsec);
  if (Date.now() >= tokenValidTimeMsec) {
    toggleLoading(true);
    axios.get('/api/refresh', {
      headers: {
        Authorization: `Bearer ${token}`,
      }
    }).then((response) => {
      toggleLoading(false);
      sessionStorage.setItem('session',
JSON.stringify(response.data));
    });
  }

  toggleLoading(true);
  axios.get('/', {
    headers: {
      Authorization: `Bearer ${token}`,
    }
  }).then((response) => {
    toggleLoading(false);
    const { username } = response.data;

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

      loginForm.remove();
      loginErrorMsg.remove();
      loginHeader.remove();

      mainHolder.append(`Hello ${username}`);
      logoutLink.style.opacity = 1;
      redirectLink.style.display = 'none';
    }
  });
}

logoutLink.addEventListener("click", (e) => {
  e.preventDefault();
  axios({ method: 'get', url: '/logout', headers: { Authorization:
`Bearer ${token}` } }, { withCredentials: true });
  sessionStorage.removeItem('session');
  location.reload();
});

loginButton.addEventListener("click", (e) => {
  e.preventDefault();

```

```

const login = loginForm.login.value;
const password = loginForm.password.value;

toggleLoading(true);
axios({
  method: 'post',
  url: '/api/login',
  data: {
    login,
    password
  }
}).then((response) => {
  toggleLoading(false);
  sessionStorage.setItem('session', JSON.stringify(response.data));
  location.reload();
}).catch((err) => {
  toggleLoading(false);
  console.log(err)
  if (err.response.data) {
    const { waitTime } = err.response.data;
    const waitSec = waitTime / 1000;
    loginSeverMsg.innerText = `Try after ${waitSec} seconds`;
    loginSeverMsg.style.opacity = 1;
    loginButton.disabled = true;
    setTimeout(() => {
      loginButton.disabled = false;
      loginSeverMsg.innerText = '';
      loginSeverMsg.style.opacity = 0;
    }, waitTime)
  }
  loginErrorMsg.style.opacity = 1;
});
})
</script>
</html>

```

Лістинг index.js, що відповідає за вхідну точку застосунку, перенаправлення та дії між субдиректоріями і тд:

```

'use strict';

const express = require('express');
const bodyParser = require('body-parser');
const cookieParser = require('cookie-parser');
const path = require('path');
const httpConstants = require('http-constants');

const { port, localTokenPath, sessionKey } = require('./config');
const appToken = require('./app-token');
const userToken = require('./user-token');
const userModel = require('./user-crud');
const AttemptManager = require('./attempt-manager');
const DataBase = require('./database');
require('dotenv').config();

```

```

const attemptsManager = new AttemptManager();
const tokensStorage = new DataBase(path.join(localTokenPath));
const app = express();

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

const SESSION_KEY = sessionKey;

app.use((req, res, next) => {
  try {
    const authorizationHeader = req.get(SESSION_KEY);
    const accessToken = authorizationHeader.split(' ')[1];
    const payload = userToken.getPayloadFromToken(accessToken);
    if (payload) {
      req.userId = payload.sub;
      console.log(`User with id ${req.userId} authorized by Access
Token`);
    } else {
      console.log('Not valid authorization header');
    }
  } catch { }

  next();
});

app.get('/', async (req, res) => {
  if (req.userId) {
    const userData = await userModel.getUserById(req.userId);

    return res.json({
      username: `${userData.name}(${userData.email})`,
      logout: 'http://localhost:3000/logout',
    });
  }
  res.sendFile(path.join(__dirname + '/index.html'));
});

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

app.get('/logout', async (req, res) => {
  try {
    const userId = req.userId;

    if (!userId) {
      return res.status(httpConstants.codes.UNAUTHORIZED).send();
    }

    console.log(`User with id ${userId} successfully logout`);
    await tokensStorage.deleteByKey(userId);
    res.clearCookie('refreshToken');
    res.redirect('/');
  }
});

```

```

    } catch (err) {
      console.error(err);
      res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
    }
  });

app.post('/api/login', async (req, res) => {
  const { login, password } = req.body;
  if (!attemptsManager.canLogin(login))
    return res
      .status(httpConstants.codes.UNAUTHORIZED)
      .json({ waitTime: attemptsManager.waitTime });

  try {
    const { accessToken, expiresIn, refreshToken } =
      await userToken.getUserAccessToken(login, password);

    const { sub: userId } = userToken.getPayloadFromToken(accessToken);
    tokensStorage.upsert(userId, { refreshToken });

    console.log(`User with id ${userId} (${login}) successfully login`);
    res.cookie('refreshToken', refreshToken, { httpOnly: true });
    res.json({
      token: accessToken,
      expiresDate: Date.now() + expiresIn * 1000,
    });
  } catch (err) {
    console.error(err);
    res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
  }
});

app.get('/api/refresh', async (req, res) => {
  try {
    const userId = req.userId;
    const { refreshToken } = req.cookies;

    if (!userId) return
    res.status(httpConstants.codes.UNAUTHORIZED).send();

    const { refreshToken: refreshTokenDb } =
      tokensStorage.getData(userId);
    if (refreshToken === refreshTokenDb) {
      const { accessToken, expiresIn } = await
      userToken.refreshUserToken(
        refreshToken
      );
      console.log(`Refresh token for user with id ${req.userId}`);
      res.json({
        token: accessToken,
        expiresDate: Date.now() + expiresIn * 1000,
      });
    }

    res.status(httpConstants.codes.UNAUTHORIZED).send();
  } catch (err) {

```

```

        console.error(err);
        res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
    }
});

app.post('/api/register', async (req, res) => {
    try {
        const userOptions = req.body;
        const user = await userModel.createUser(userOptions);

        console.log(
            `User with id ${user.user_id} (${user.email}) successfully
registered`
        );
        res.json({ redirect: '/' });
    } catch (err) {
        console.error(err);
        res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
    }
});

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

    const appAccessToken = await appToken.getAppAccessToken();

    console.log({ appAccessToken });
});

```

Лістинг request-options.js, що відповідає за шаблони запитів:

```

'use strict';
const httpConstants = require('http-constants');
const uuid = require('uuid');
const config = require('./config');
const getAppTokenOptions = () => ({
    method: httpConstants.methods.POST,
    url: `https://${config.domain}/oauth/token`,
    headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
    form: {
        client_id: config.clientId,
        client_secret: config.clientSecret,
        audience: config.audience,
        grant_type: 'client_credentials',
    },
});

const getUser = (name, surname, nickname, login, password) => ({
    email: login,
    user_metadata: {},
    blocked: false,
    email_verified: false,
    app_metadata: {},
    given_name: name,
    family_name: surname,
    name: `${name} ${surname}`,

```

```

    nickname,
    picture: config.pictureUrl,
    user_id: uuid.v4(),
    connection: 'Username-Password-Authentication',
    password,
    verify_email: false,
  });
const getUserCreateOptions = (authorization, user) => ({
  method: httpConstants.methods.POST,
  url: `https://${config.domain}/api/v2/users`,
  headers: {
    'Content-Type': 'application/json',
    Authorization: authorization,
  },
  body: JSON.stringify(user),
});
const getUserTokenOptions = (username, password) => ({
  method: httpConstants.methods.POST,
  url: `https://${config.domain}/oauth/token`,
  headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
  form: {
    grant_type: 'password',
    audience: config.audience,
    client_id: config.clientId,
    client_secret: config.clientSecret,
    scope: 'offline_access',
    username: username,
    password: password,
  },
});
const getRefreshUserTokenOptions = (refreshToken) => ({
  method: httpConstants.methods.POST,
  url: `https://${config.domain}/oauth/token`,
  headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
  form: {
    grant_type: 'refresh_token',
    client_id: config.clientId,
    client_secret: config.clientSecret,
    refresh_token: refreshToken,
  },
});
const getUserGetOptions = (authorization, userId) => ({
  method: httpConstants.methods.GET,
  url: `https://${config.domain}/api/v2/users/${userId}`,
  headers: {
    Authorization: authorization,
  },
});
module.exports = {
  getUser,
  getAppTokenOptions,
  getUserCreateOptions,
  getUserTokenOptions,
  getRefreshUserTokenOptions,
  getUserGetOptions,
};

```

Лістинг database.js, для псевдо бд, що зберігатиме токен:

```
'use strict';
const fsp = require('fs/promises');
const fs = require('fs');
class DataBase {
  #data;
  constructor(path = './tokens.json') {
    this.path = path;
    try {
      this.#data = fs.readFileSync(this.path, 'utf8');
      this.#data = JSON.parse(this.#data.trim());
    } catch (e) {
      this.#data = {};
    }
  }

  async upsert(key, data) {
    this.#data[key] = data || {};
    await this.store();
  }

  getData(key) {
    if (this.#data[key]) return this.#data[key];
    return null;
  }

  deleteByFind(callback) {
    for (const key in this.#data) {
      if (callback(this.#data[key])) {
        this.#data[key] = {};
        return key;
      }
    }
  }

  find(callback) {
    for (const key in this.#data) {
      if (callback(this.#data[key])) {
        return key;
      }
    }
  }

  async deleteByKey(key) {
    if (key) delete this.#data[key];
    await this.store();
  }

  async store() {
    try {
      const buffer = JSON.stringify(this.#data);
      await fsp.writeFile(this.path, buffer);
    } catch (err) { }
  }
}
module.exports = DataBase;
```

Лістинг app-token.js, що відповідає за логіку отримання токена, збереження, оновлення:

```
'use strict';
const httpConstants = require('http-constants');
const fsp = require('fs/promises');
const requestCallback = require('request');
const { promisify } = require('util');
const config = require('./config');
const options = require('./request-options');
const hourInSec = 60 * 60;
const request = promisify(requestCallback);
const defaultTokenOptions = options.getAppTokenOptions();

const readTokenInfo = async () => {
  try {
    const buffer = await fsp.readFile(config.localTokenPath, 'utf-8');
    const json = JSON.parse(buffer);
    if (json.expiryDate <= Date.now()) {
      return null;
    }

    return json.tokenInfo;
  } catch (err) {
    return null;
  }
};

const storeTokenInfo = async (tokenInfo) => {
  try {
    const tokenValidTimeMsec = (tokenInfo.expires_in - hourInSec) * 1000;
    const buffer = JSON.stringify({
      tokenInfo,
      expiryDate: Date.now() + tokenValidTimeMsec,
    });
    await fsp.writeFile(config.localTokenPath, buffer);
  } catch (err) {
    return null;
  }
};

const getAppAccessToken = async (tokenOptions = defaultTokenOptions) => {
  let tokenInfo = await readTokenInfo();

  if (!tokenInfo) {
    const tokenResponse = await request(tokenOptions);
    if (tokenResponse.statusCode !== httpConstants.codes.OK) {
      const { statusCode, statusMessage, body } = tokenResponse;
      console.dir({ statusCode, statusMessage, body });
      return;
    }

    tokenInfo = JSON.parse(tokenResponse.body);
    await storeTokenInfo(tokenInfo);
  }

  return tokenInfo;
};
```



```
module.exports = {
  readTokenInfo,
  storeTokenInfo,
  getAppAccessToken,
};
```

Лістинг user-token.js, яка в свою чергу вже відповідає за токени конкретно користувача, якого ми будемо авторизовувати:

```
'use strict';
const httpConstants = require('http-constants');
const requestCallback = require('request');
const { promisify } = require('util');
const jwt = require('jsonwebtoken');
const options = require('./request-options');
const request = promisify(requestCallback);
const getUserAccessToken = async (username, password) => {
  const userTokenOptions = options.getUserTokenOptions(username, password);

  const accessTokenUserResponse = await request(userTokenOptions);
  if (accessTokenUserResponse.statusCode !== httpConstants.codes.OK) {
    const { statusCode, statusMessage, body } = accessTokenUserResponse;
    throw new Error(
      `Auth0 user-token: ${statusCode} ${statusMessage} ${body}`
    );
  }
  const response = JSON.parse(accessTokenUserResponse.body);
  return {
    accessToken: response.access_token,
    expiresIn: response.expires_in,
    refreshToken: response.refresh_token,
  };
};
const refreshUserToken = async (refreshToken) => {
  const refreshTokenOptions =
options.getRefreshUserTokenOptions(refreshToken);
  const refreshTokenResponse = await request(refreshTokenOptions);
  if (refreshTokenResponse.statusCode !== httpConstants.codes.OK) {
    const { statusCode, statusMessage, body } = refreshTokenResponse;
    throw new Error(
      `Auth0 user-token: ${statusCode} ${statusMessage} ${body}`
    );
  }

  const response = JSON.parse(refreshTokenResponse.body);
  return {
    accessToken: response.access_token,
    expiresIn: response.expires_in,
  };
};
const getPayloadFromToken = (token) => {
  try {
    return jwt.decode(token);
  } catch (error) {
    return null;
  }
};
```

```

    }
  };
  module.exports = {
    getUserAccessToken,
    getPayloadFromToken,
    refreshUserToken,
  };
};

```

Лістинг user-crud.js, для взяття користувача по ІД чи його створення:

```

'use strict';

const requestCallback = require('request');
const { promisify } = require('util');
const httpConstants = require('http-constants');

const options = require('./request-options');
const appToken = require('./app-token');
const request = promisify(requestCallback);

const getUserById = async (userId) => {
  const { access_token, token_type } = await appToken.getAppAccessToken();
  const authorizationHeader = `${token_type} ${access_token}`;
  const userGetOptions = options.getUserGetOptions(authorizationHeader,
userId);

  const userResponse = await request(userGetOptions);
  if (userResponse.statusCode !== httpConstants.codes.OK) {
    const { statusCode, statusMessage, body } = userResponse;
    throw new Error(
      `Auth0 user-token: ${statusCode} ${statusMessage} ${body}`
    );
  }

  const response = JSON.parse(userResponse.body);

  return response;
};

const createUser = async (userInput) => {
  const { access_token, token_type } = await appToken.getAppAccessToken();
  const authorizationHeader = `${token_type} ${access_token}`;

  const { name, surname, nickname, login, password } = userInput;
  const user = options.getUser(name, surname, nickname, login, password);
  const isValidUser = Object.keys(user).reduce(
    (acc, key) => acc && user[key] !== undefined,
    true
  );
  if (!isValidUser) {
    throw new Error(
      `Not all fields of registration: ${statusMessage} ${body}`
    );
  }
}

```


localhost:3000/register

Register

Name

Surname

Nickname

Email

Password

Register

[Already have an account? Sign in](#)

Заповнюємо форму:

autho

register

lab4


lab4@gmail.com

.....

Register

Успішно створюємо нового користувача:

```
at async 3: (dev) (Studying (K1-Studying (7th Semester (Software Security (lab4 (index.js:88:13)
User with id auth0|27e3b8fa-24af-48d2-94bb-dbd24476068 (lab4@gmail.com) successfully registered
```


 **autho register**
lab4@gmail.com

Username-Password-Authenti... 0 never

...

І відповідно спробуємо зайти в нього для підтвердження працездатності:

```
User with id auth0|27e3b8fa-24af-48d2-94bb-dbd24476068 (lab4@gmail.com) successfully login
User with id auth0|27e3b8fa-24af-48d2-94bb-dbd24476068 authorized by Access Token
```

Name	Connection	Logins	Latest Login
 autho register lab4@gmail.com	Username-Password-Authenti...	1	a few seconds ...

Оскільки решту коду де використовується створення користувача, запити ітд вже було описано в попередніх лістингах, залишилось лише описати новостворену сторінку у подібному до головного вікна стилі, яка додатково має форму для даних та виконує відповідні запити для реєстрації. Лістинг register.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Register</title>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
</head>

<body>
  <main id="main-holder">
    <h1 id="register-header">Register</h1>

    <div id="register-error-msg-holder">
      <p id="register-server-msg" class="register-error-msg"></p>
    </div>

    <form id="register-form" action="/api/login" method="post">
      <input type="text" name="name" id="name-field"
class="register-form-field" placeholder="Name">
      <input type="text" name="surname" id="surname-field"
class="register-form-field" placeholder="Surname">
      <input type="text" name="nickname" id="nickname-field"
class="register-form-field" placeholder="Nickname">
      <input type="email" name="login" id="email-field"
class="register-form-field" placeholder="Email">
      <input type="password" name="password" id="password-field"
class="register-form-field"
        placeholder="Password">
      <input type="submit" value="Register" id="register-form-submit">
    </form>

    <a class="redirect-link" href="/">Already have an account? Sign in</a>
  </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;
}

#register-error-msg-holder {
  width: 100%;
  height: 100%;
  display: grid;
  justify-items: center;
  align-items: center;
}

.register-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;}

#register-form {
  align-self: flex-start;
  display: grid;
  justify-items: center;
  align-items: center;
}

.register-form-field::placeholder {color: #3a3a3a;}
```

```

.register-form-field {
  border: none;
  border-bottom: 1px solid #3a3a3a;
  margin-bottom: 10px;
  border-radius: 3px;
  outline: none;
  padding: 0px 0px 5px 5px;
}

#register-form-submit {
  width: 100%;
  padding: 7px;
  border: none;
  border-radius: 5px;
  color: white;
  font-weight: bold;
  background-color: #3a3a3a;
  cursor: pointer;
  outline: none;
}

#register-form-submit:disabled {
  background-color: #900000;
}

.redirect-link {
  color: #3d3d3d;
}
</style>

<script>
const registerForm = document.getElementById("register-form");
const registerButton = document.getElementById("register-form-submit");
const registerSeverMsg = document.getElementById("register-server-msg");
const logoutLink = document.getElementById("logout");
const hourInMSec = 60 * 60 * 1000;

const toggleLoading = (isLoading) => {
  registerForm.style.display = isLoading ? 'none' : '';
};

registerButton.addEventListener("click", (e) => {
  e.preventDefault();
  registerSeverMsg.innerHTML = '12212';
  const name = registerForm.name.value;
  const surname = registerForm.surname.value;
  const nickname = registerForm.nickname.value;
  const login = registerForm.login.value;
  const password = registerForm.password.value;

  toggleLoading(true);
  axios({
    method: 'post',
    url: '/api/register',
    data: {
      name,

```



```
        surname,  
        nickname,  
        login,  
        password  
    }  
}).then((response) => {  
    toggleLoading(false);  
    if (response.data.redirect === '/') {  
        window.location = response.data.redirect;  
    }  
}).catch((err) => {  
    toggleLoading(false);  
    console.log(err)  
    if (err.response.data) {  
        alert(err.response.data);  
    }  
});  
})  
</script>  
  
</html>
```

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

Лабораторна робота №5
з дисципліни
«Безпека програмного забезпечення»

Виконав:
студент групи ІП-05
Гапій Денис Едуардович

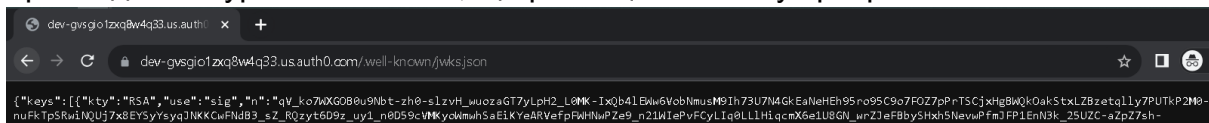
Перевірів:
Іваніщев Б. В.

Київ 2023

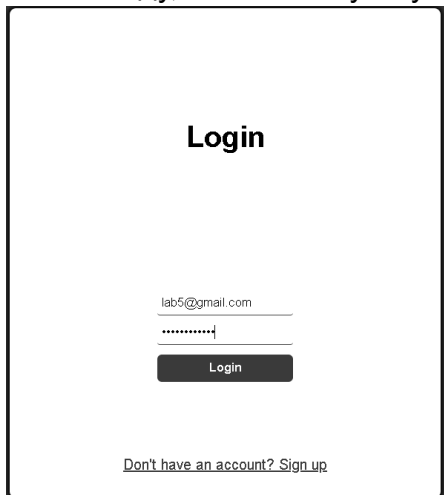
Мета: «Засвоєння базових навичок OAuth2 авторизаційного протокола»

Завдання 1: Розширити Лабораторну роботу 4 перевіркою сигнатури JWT токена. Приклади SDK <https://auth0.com/docs/quickstart/backend>. У випадку асиметричного ключа, public є можливість отримати за посиланням <https://kpi.eu.auth0.com/pem>, або за формулою [https://\[API_DOMAIN\]/pem](https://[API_DOMAIN]/pem)

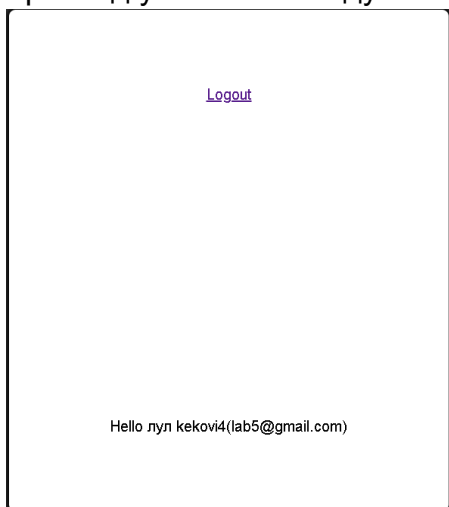
Приклад сигнатури JWT токена, що розміщений на аус розробника:



Вікно входу, таке ж як було у auth_token:



Приклад успішного входу:



```
User with id auth0|5795eadc-4c5c-4dd7-8a2f-8eb2c4a5a12f (lab5@gmail.com) successfully registered
User with id auth0|5795eadc-4c5c-4dd7-8a2f-8eb2c4a5a12f (lab5@gmail.com) successfully login
User with id auth0|5795eadc-4c5c-4dd7-8a2f-8eb2c4a5a12f authorized by Access Token
auth0|5795eadc-4c5c-4dd7-8a2f-8eb2c4a5a12f
```

Приклад згенерованого public key:

```
JS index.js  public.key X JS jwt-utils.js JS public-key.js JS config.js
lab4 > public.key
1  -----BEGIN CERTIFICATE-----
2  MIIDHTCCAgwGAWIBAgIJPCd2FnhmmfcOMA0GCSqGSIb3DQEBCwUAMCwxKjAoBgNV
3  BAMTIWRldi1ndnNnaW8xenhxOHc0cTMzLnVzLmF1dGgwLmNvbTAeFw0yMzA5Mjcx
4  MjM1NTlaFw0zNzA2MDUxMjM1NTlaMCwxKjAoBgNVBAMTIWRldi1ndnNnaW8xenhx
5  OHc0cTMzLnVzLmF1dGgwLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
15  ab/eWeXbFsblURSL+9QztefhIvea2z2Hg+AuHfEiXbiVtOieUN5bbteogs5+EEGy
16  /7eYAlwkGExHbQ2hmpj/YLx40znYzYLG9z4fbTCfXyQfTF2eqBG1Fg1npFb6lfyf
17  JvIT5nYW8T2dxFTQrn+7mSmx+RDvNlVEoYYDwUosRtQbjn//bkvhysv6pnOimhqW
18  V4gOePX3PQ+z6qz/6OKkdxWr4gcm1TnyotxMjh9FD0GX
19  -----END CERTIFICATE-----
```

Лістинг jwt-utils.js, для перевірки сигнатури JWT:

```
'use strict';
const config = require('./config');
const { getPublicKey } = require('./public-key');
const jwt = require('jsonwebtoken');
const verifyOptions = {
  issuer: `https://${config.domain}/`,
  audience: config.audience,
  algorithms: ['RS256'],
};
const verifyToken = async (accessToken) => {
  try {
    const publicKey = config.publicKey || (await getPublicKey());
    config.publicKey = publicKey;

    const payload = jwt.verify(accessToken, publicKey, verifyOptions);

    return payload;
  } catch (err) {
    console.log({ jwtVerifyErrorMsg: err.message });
    return null;
  }
};
module.exports = {
  verifyToken,
```

```
};
```

Лістинг нового файлу public-key.js, для отримання public ключа, у випадку асиметричного:

```
'use strict';
const { promisify } = require('util');
const requestCallback = require('request');
const request = promisify(requestCallback);
const fs = require('fs');
const config = require('./config');
const getPublicKey = async () => {
  if (!fs.existsSync('public.key')) {
    const { body: publicKey } = await
request(`https://${config.domain}/pem`);

    await fs.promises.writeFile('public.key', publicKey, 'utf-8');
    return publicKey;
  }

  const publicKey = await fs.promises.readFile('public.key', 'utf-8');
  return publicKey;
};
module.exports = {
  getPublicKey,
};
```

Оновлення в представлені, лістинг index.html:

```
toggleLoading(true);
axios.get('/userinfo', {
  headers: {
    Authorization: `Bearer ${token}`,
  }
}).then((response) => {
  toggleLoading(false);
  const { username } = response.data;

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

    loginForm.remove();
    loginErrorMsg.remove();
    loginHeader.remove();

    mainHolder.append(`Hello ${username}`);
    logoutLink.style.opacity = 1;
    redirectLink.style.display = 'none';
  }
}).catch(() => {
  toggleLoading(false);
});
}
```

І основні оновлення були у лістингу index.js:

```

'use strict';

const express = require('express');
const bodyParser = require('body-parser');
const cookieParser = require('cookie-parser');
const path = require('path');
const httpConstants = require('http-constants');
const { expressJwt: expressJwt } = require('express-jwt');
const jwksRsa = require('jwks-rsa');
const config = require('./config');
const appToken = require('./app-token');
const userToken = require('./user-token');
const userModel = require('./user-crud');
const AttemptManager = require('./attempt-manager');
const DataBase = require('./database');
const { getPublicKey } = require('./public-key');
const { verifyToken } = require('./jwt-utils');
require('dotenv').config();

const attemptsManager = new AttemptManager();
const tokensStorage = new DataBase(path.join(config.localTokenPath));
const app = express();

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

const SESSION_KEY = config.sessionKey;

const checkJwt = expressJwt({
  secret: jwksRsa.expressJwtSecret({
    cache: true,
    rateLimit: true,
    jwksRequestsPerMinute: 5,
    jwksUri: `https://${config.domain}/.well-known/jwks.json`,
  }),

  audience: config.audience,
  issuer: `https://${config.domain}/`,
  algorithms: ['RS256'],
});

app.use(async (req, res, next) => {
  const authorizationHeader = req.get(SESSION_KEY);
  if (!authorizationHeader) return next();
  const accessToken = authorizationHeader.split(' ')[1];
  const payload = await verifyToken(accessToken);
  if (payload) {
    req.userId = payload.sub;
    console.log(`User with id ${req.userId} authorized by Access Token`);
  } else {
    console.log('Not valid authorization header');
  }

  next();
});

```

```

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

app.get('/userinfo', checkJwt, async (req, res) => {
  if (req.userId) {
    const userData = await userModel.getUserById(req.userId);

    return res.json({
      username: `${userData.name}(${userData.email})`,
      logout: 'http://localhost:3000/logout',
    });
  }
  res.status(httpConstants.codes.UNAUTHORIZED).send();
});

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

app.get('/logout', async (req, res) => {
  try {
    const userId = req.userId;

    if (!userId) {
      return res.status(httpConstants.codes.UNAUTHORIZED).send();
    }

    console.log(`User with id ${userId} successfully logout`);
    await tokensStorage.deleteByKey(userId);
    res.clearCookie('refreshToken');
    res.redirect('/');
  } catch (err) {
    console.error(err);
    res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
  }
});

app.post('/api/login', async (req, res) => {
  const { login, password } = req.body;
  if (!attemptsManager.canLogin(login))
    return res
      .status(httpConstants.codes.UNAUTHORIZED)
      .json({ waitTime: attemptsManager.waitTime });

  try {
    const { accessToken, expiresIn, refreshToken } =
      await userToken.getUserAccessToken(login, password);

    const { sub: userId } = userToken.getPayloadFromToken(accessToken);
    tokensStorage.upsert(userId, { refreshToken });

    console.log(`User with id ${userId} (${login}) successfully login`);
    res.cookie('refreshToken', refreshToken, { httpOnly: true });
    res.json({

```

```

        token: accessToken,
        expiresDate: Date.now() + expiresIn * 1000,
    });
} catch (err) {
    console.error(err);
    res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
}
});

app.get('/api/refresh', async (req, res) => {
    try {
        const userId = req.userId;
        const { refreshToken } = req.cookies;

        if (!userId) return
res.status(httpConstants.codes.UNAUTHORIZED).send();

        const { refreshToken: refreshTokenDb } =
tokensStorage.getData(userId);
        if (refreshToken === refreshTokenDb) {
            const { accessToken, expiresIn } = await
userToken.refreshUserToken(
                refreshToken
            );
            console.log(`Refresh token for user with id ${req.userId}`);
            res.json({
                token: accessToken,
                expiresDate: Date.now() + expiresIn * 1000,
            });
        }

        res.status(httpConstants.codes.UNAUTHORIZED).send();
    } catch (err) {
        console.error(err);
        res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
    }
});

app.post('/api/register', async (req, res) => {
    try {
        const userOptions = req.body;
        const user = await userModel.createUser(userOptions);

        console.log(
            `User with id ${user.user_id} (${user.email}) successfully
registered`
        );
        res.json({ redirect: '/' });
    } catch (err) {
        console.error(err);
        res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
    }
});

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

```



```
const appAccessToken = await appToken.getAppAccessToken();  
const publicKey = await getPublicKey();  
console.log({ appAccessToken });  
});
```

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

Лабораторна робота №6
з дисципліни
«Безпека програмного забезпечення»

Виконав:
студент групи ІП-05
Гапій Денис Едуардович

Перевірив:
Іваніщев Б. В.

Київ 2023

Мета: «Засвоєння базових навичок OAuth2 авторизаційного протокола»

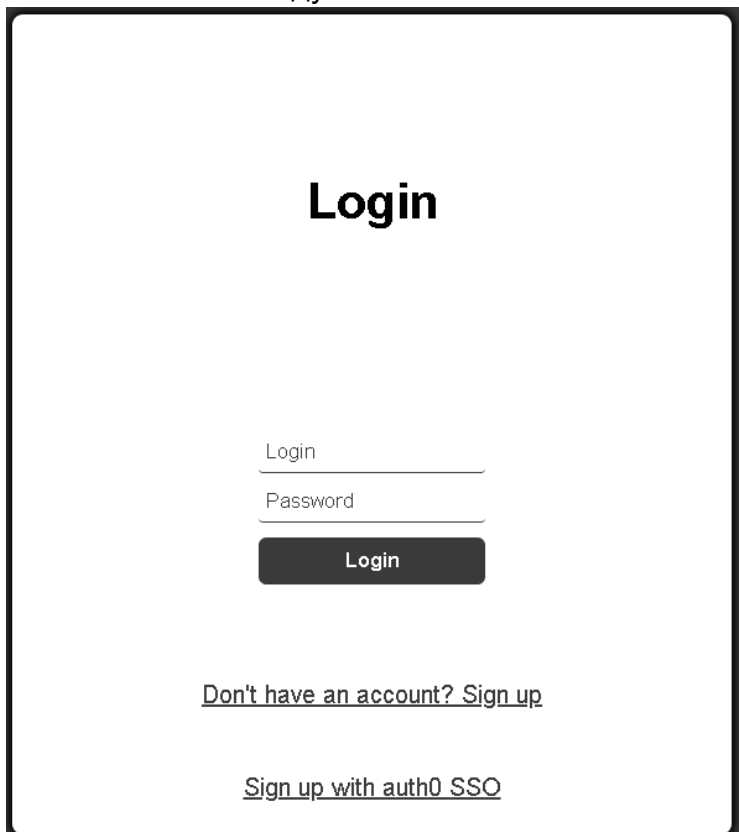
Завдання 1: Розширити Лабораторну роботу 4, змінивши логін сторінку на стандартну від SSO провайдера, для цього, треба зробити редірект на API_DOMAIN

<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=JlvCO5c2IBHIAe2patn6l6q5H35qxti0&redirect_uri=http%3A%2F%2Flocalhost%3A3000&response_type=code&response_mode=query

Завдання 2: додатково розширити аплікейшн обробкою редіректа та отриманням юзер токена за допомогою code grant type

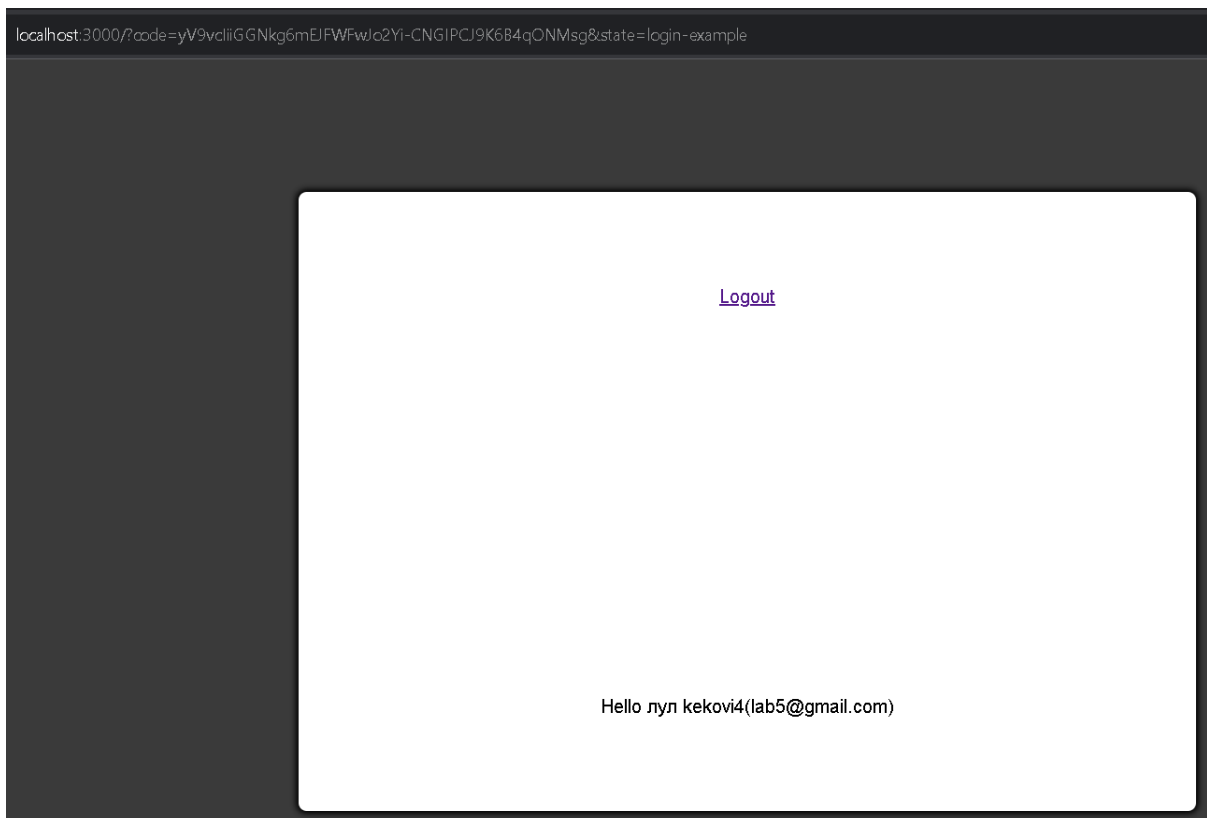
Оновлене вікно входу:



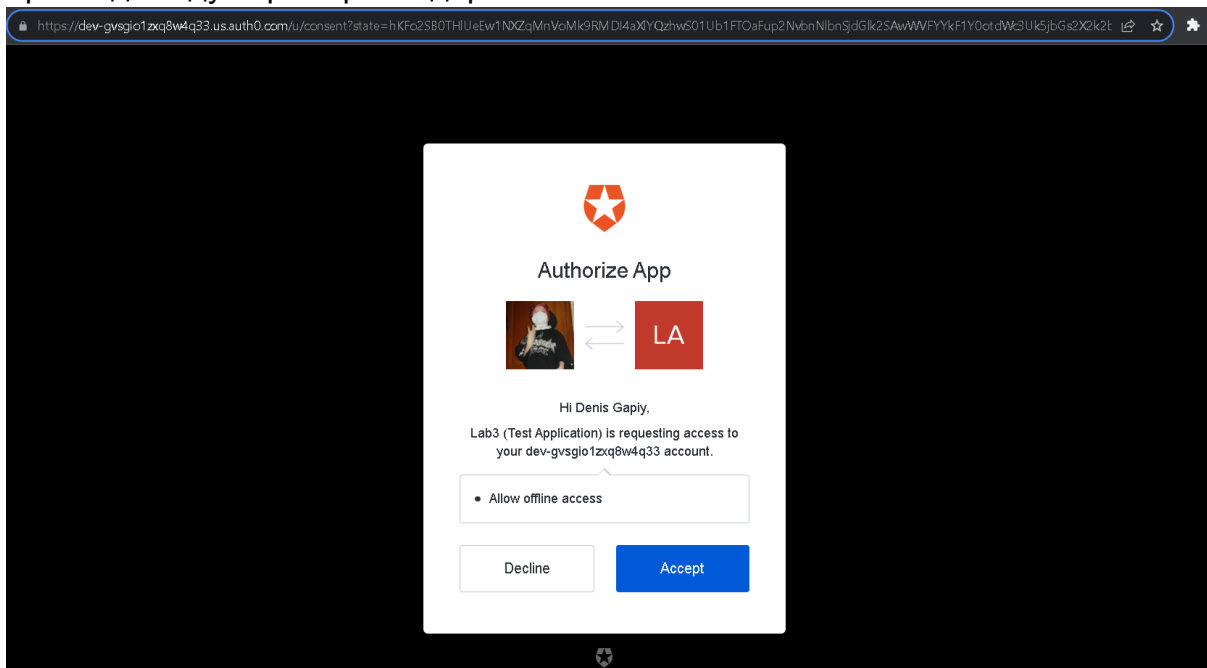
The image shows a login page with the following elements:

- Login** (title)
-
-
- Login** (button)
- [Don't have an account? Sign up](#)
- [Sign up with auth0 SSO](#)

Приклад входу, з використанням code:



Приклад входу через провайдера SSO:



Лістинг нового файлу, auth-code.js, для доступу / токену по коду розробника:

```
'use strict';
const requestCallback = require('request');
const { promisify } = require('util');
const httpConstants = require('http-constants');
const options = require('./request-options');
const request = promisify(requestCallback);
const authByCode = async (code) => {
```

```

const codeAuthOptions = options.getCodeOptions(code);
const tokenResponse = await request(codeAuthOptions);
if (tokenResponse.statusCode !== httpConstants.codes.OK) {
  const { statusCode, statusMessage, body } = tokenResponse;
  throw new Error(
    `Auth0 user-token: ${statusCode} ${statusMessage} ${body}`
  );
}
const response = JSON.parse(tokenResponse.body);
return {
  accessToken: response.access_token,
  expiresIn: response.expires_in,
  refreshToken: response.refresh_token,
};
};
module.exports = {
  authByCode,
};

```

Оновлення в request-options.js:

```

const getCodeOptions = (authorizationCode) => ({
  method: httpConstants.methods.POST,
  url: `https://${config.domain}/oauth/token`,
  headers: {
    'content-type': 'application/x-www-form-urlencoded',
  },
  form: {
    grant_type: 'authorization_code',
    client_id: config.clientId,
    client_secret: config.clientSecret,
    code: authorizationCode,
    redirect_uri: 'http://localhost:3000',
  },
});
module.exports = {
  getUser,
  getAppTokenOptions,
  getUserCreateOptions,
  getUserTokenOptions,
  getRefreshUserTokenOptions,
  getUserGetOptions,
  getCodeOptions
};

```

Оновлення в конфіг файлі:

```

'use strict';
require('dotenv').config();
const config = {
  port: process.env.PORT || 8080,
  sessionKey: 'Authorization',
  domain: process.env.AUTH0_DOMAIN || '',
  clientId: process.env.AUTH0_CLIENT_ID || '',
  clientSecret: process.env.AUTH0_CLIENT_SECRET || '',
  audience: process.env.AUTH0_AUDIENCE || '',
  pictureUrl:

'https://unity.com/sites/default/files/styles/social_media_sharing/public/2022
-02/U_Logo_White_CMYK.jpg',
  localTokenPath: `${__dirname}/token-info.json`,
  refreshTokenViaTimeSec: 500,
  timeToRefreshSec: 23.95 * 60 * 60
}
module.exports = {
  ...config,
  loginUrl:
`https://${config.domain}/authorize?response_type=code&client_id=${config.clie
ntId}&redirect_uri=http://localhost:3000&scope=offline_access&audience=${confi
g.audience}&state=login-example`
};

```

Оновлення в лістингу index.js:

```

'use strict';
const express = require('express');
const bodyParser = require('body-parser');
const cookieParser = require('cookie-parser');
const path = require('path');
const httpConstants = require('http-constants');
const { expressJwt: expressJwt } = require('express-jwt');
const jwksRsa = require('jwks-rsa');
const config = require('./config');
const appToken = require('./app-token');
const userToken = require('./user-token');
const userModel = require('./user-crud');
const AttemptManager = require('./attempt-manager');
const DataBase = require('./database');
const { getPublicKey } = require('./public-key');
const { verifyToken } = require('./jwt-utils');
const { authByCode } = require('./auth-code');
require('dotenv').config();
const attemptsManager = new AttemptManager();
const tokensStorage = new DataBase(path.join(config.localTokenPath));
const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(cookieParser());
const SESSION_KEY = config.sessionKey;
const checkJwt = expressJwt({
  secret: jwksRsa.expressJwtSecret({
    cache: true,
    rateLimit: true,
    jwksRequestsPerMinute: 5,

```

```

        jwksUri: `https://${config.domain}/.well-known/jwks.json`,
    )),
    audience: config.audience,
    issuer: `https://${config.domain}/`,
    algorithms: ['RS256'],
  }));
app.use(async (req, res, next) => {
  const authorizationHeader = req.get('SESSION_KEY');
  if (!authorizationHeader) return next();
  const accessToken = authorizationHeader.split(' ')[1];
  const payload = await verifyToken(accessToken);
  if (payload) {
    req.userId = payload.sub;
    if (payload.exp - config.timeToRefreshSec <= Date.now()) {
      const { refreshToken } = tokensStorage.getData(req.userId);
      const { accessToken, expiresIn } = await
userToken.refreshUserToken(refreshToken);
      res.setHeader('AccessToken', accessToken);
      res.setHeader('expiresDate', Date.now() + expiresIn * 1000);
    }
    console.log(`User with id ${req.userId} authorized by Access Token`);
  } else {
    console.log('Not valid authorization header');
  }

  next();
});
app.get('/', async (req, res) => {
  const queryParams = req.query;

  if (
    queryParams &&
    queryParams.code &&
    queryParams.state === config.state
  ) {
    try {
      const { code } = queryParams;
      const { accessToken, expiresIn, refreshToken } = await
authByCode(code);
      res.setHeader('AccessToken', accessToken);
      res.setHeader('expiresDate', Date.now() + expiresIn * 1000);
      const { sub: userId } =
userToken.getPayloadFromToken(accessToken);
      tokensStorage.upsert(userId, { refreshToken });
    } catch { }
  }
  res.sendFile(path.join(__dirname + '/index.html'));
});
app.get('/userinfo', checkJwt, async (req, res) => {
  if (req.userId) {
    const userData = await userModel.getUserById(req.userId);

    return res.json({
      username: `${userData.name}(${userData.email})`,
      logout: 'http://localhost:3000/logout',
    });
  }
});

```

```

    });
  }
  res.status(httpConstants.codes.UNAUTHORIZED).send();
});
app.get('/register', (req, res) => {
  res.sendFile(path.join(__dirname + '/register.html'));
});
app.get('/logout', async (req, res) => {
  try {
    const userId = req.userId;

    if (!userId) {
      return res.status(httpConstants.codes.UNAUTHORIZED).send();
    }

    console.log(`User with id ${userId} successfully logout`);
    await tokensStorage.deleteByKey(userId);
    res.redirect('/');
  } catch (err) {
    console.error(err);
    res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
  }
});
app.get('/login', async (req, res) => {
  res.redirect(config.loginUrl);
});
app.post('/api/login', async (req, res) => {
  const { login, password } = req.body;
  if (!attemptsManager.canLogin(login))
    return res
      .status(httpConstants.codes.UNAUTHORIZED)
      .json({ waitTime: attemptsManager.waitTime });

  try {
    const { accessToken, expiresIn, refreshToken } =
      await userToken.getUserAccessToken(login, password);

    const { sub: userId } = userToken.getPayloadFromToken(accessToken);
    tokensStorage.upsert(userId, { refreshToken });

    console.log(`User with id ${userId} (${login}) successfully login`);
    res.json({
      token: accessToken,
      expiresDate: Date.now() + expiresIn * 1000,
    });
  } catch (err) {
    console.error(err);
    res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
  }
});
app.get('/api/refresh', async (req, res) => {
  try {
    const userId = req.userId;

    if (!userId) return
    res.status(httpConstants.codes.UNAUTHORIZED).send();
  }
});

```



```

        const { refreshToken: refreshTokenDb } =
tokensStorage.getData(userId);
        if (refreshTokenDb) {
            const { accessToken, expiresIn } = await
userToken.refreshUserToken(
                refreshTokenDb
            );
            console.log(`Refresh token for user with id ${req.userId}`);
            res.json({
                token: accessToken,
                expiresDate: Date.now() + expiresIn * 1000,
            });
        }
        res.status(httpConstants.codes.UNAUTHORIZED).send();
    } catch (err) {
        console.error(err);
        res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
    }
});
app.post('/api/register', async (req, res) => {
    try {
        const userOptions = req.body;
        const user = await userModel.createUser(userOptions);

        console.log(
            `User with id ${user.user_id} (${user.email}) successfully
registered`
        );
        res.json({ redirect: '/' });
    } catch (err) {
        console.error(err);
        res.status(httpConstants.codes.INTERNAL_SERVER_ERROR).send();
    }
});
app.listen(config.port, async () => {
    console.log(`Example app listening on port ${config.port}`);

    const appAccessToken = await appToken.getAppAccessToken();
    const publicKey = await getPublicKey();
    console.log({ appAccessToken });
});

```

І наостанок оновлення в лістинг index.html:

```

. . .
<a class="redirect-link" id="redirect-register" href="/register">Don't have an
account? Sign up</a>
    <a class="redirect-link" id="redirect-login" href="/login">Sign up
with auth0 SSO</a>
. . .
<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 loginSeverMsg = document.getElementById('login-server-msg');
const logoutLink = document.getElementById('logout');
const redirectRegister = document.getElementById('redirect-register');
const redirectLogin = document.getElementById('redirect-login');
redirectRegister.style.display = '';
redirectLogin.style.display = '';
function fetchSimilarHeaders(callback) {
    const request = new XMLHttpRequest();
    request.onreadystatechange = function () {
        if (callback && typeof callback === 'function') {
            if (request.readyState === XMLHttpRequest.DONE) {
                const token = request.getResponseHeader('accesstoken');
                const expiresDate =
+request.getResponseHeader('expiresDate')
                callback({ token, expiresDate });
            }
        }
    };
    request.open('HEAD', document.location, true);
    request.send(null);
}
const hourToRefreshInMSec = 23.95 * 60 * 60 * 1000;

fetchSimilarHeaders(headers => {
    if (headers.token && headers.expiresDate) {
        sessionStorage.setItem('session', JSON.stringify(headers));
        document.location = '/';
    }
});
. . .
if (token) {
    const tokenValidTimeMsec = expiresDate - hourToRefreshInMSec;
    console.log(tokenValidTimeMsec);
    if (Date.now() >= tokenValidTimeMsec) {
        toggleLoading(true);
        axios.get('/api/refresh', {
            headers: {
                Authorization: `Bearer ${token}`,
            }
        }).then((response) => {

```

```

        toggleLoading(false);
        sessionStorage.setItem('session',
JSON.stringify(response.data));
    });
}

toggleLoading(true);
axios.get('/userinfo', {
    headers: {
        Authorization: `Bearer ${token}`,
    }
}).then((response) => {
    toggleLoading(false);
    const { username } = response.data;

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

        loginForm.remove();
        loginErrorMsg.remove();
        loginHeader.remove();

        mainHolder.append(`Hello ${username}`);
        logoutLink.style.opacity = 1;
        redirectRegister.style.display = 'none';
        redirectLogin.style.display = 'none';
    }
    if (response.headers.accesstoken && response.headers.expiresdate)
{
        sessionStorage.setItem(
            'session',
            JSON.stringify({ token: response.headers.accesstoken,
expiresDate: +response.headers.expiresdate })
        );
    }
}).catch(() => {
    toggleLoading(false);
});
}

. . .

```

