

**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 № 3 з дисципліни  
«Безпека програмного забезпечення»

**“Авторизаційний протокол OAuth2”**

**Виконав(ла)**

*ІІ-13 Бабіч Денис*

\_\_\_\_\_  
(шифр, прізвище, ім'я, по батькові)

**Перевірів(ла)**

*Соколовський В. В.*

\_\_\_\_\_  
(шифр, прізвище, ім'я, по батькові)

Київ 2024

## ЛАБОРАТОРНА РОБОТА № 3

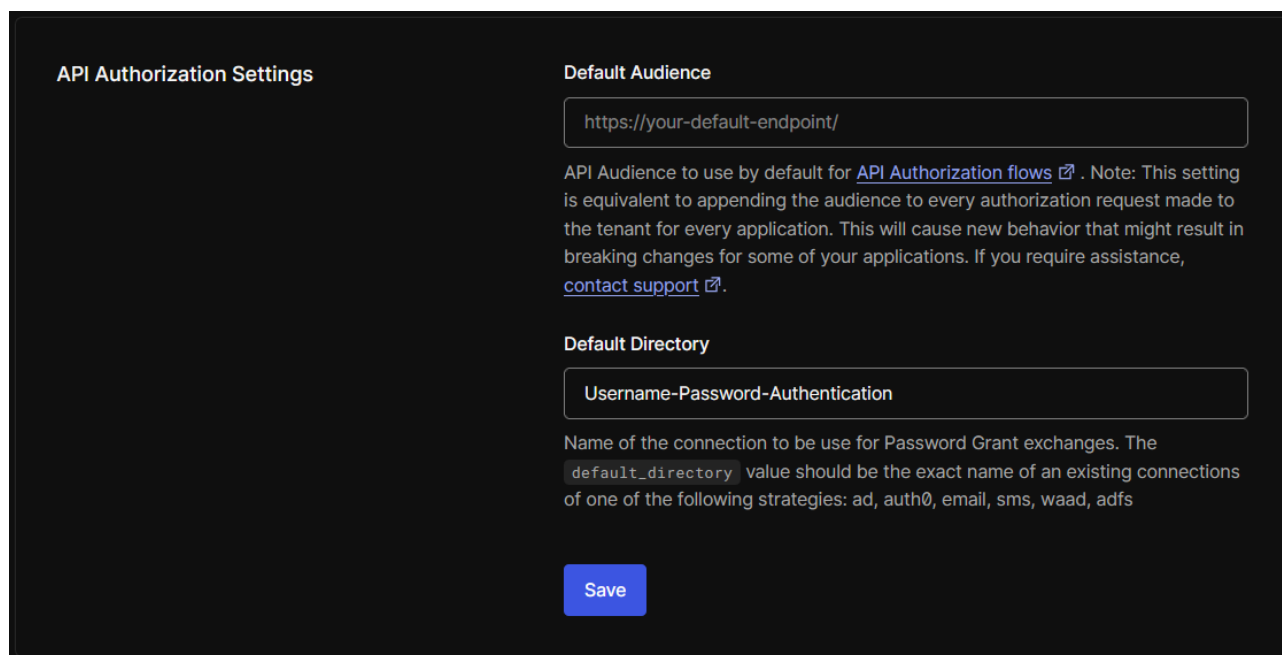
**Тема роботи:** Авторизаційний протокол OAuth2.

**Мета роботи:** Засвоєння базових навичок OAuth2 авторизаційного протокола.

**Основне завдання:**

1. Використовуючи наведені налаштування з лабораторної роботи № 2 зробити запит на отримання user token (попередньо створеного в лабораторній роботі 2).

Для виконання лабораторної роботи був використаний сервіс <https://auth0.com/>, у якому був створений тестовий застосунок та були визначені endpoint'и. Важливим елементом було додавання стандартного способу аутентифікації користувача, налаштування якого зображено на рисунку 1.1.



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

[Save](#)

Рисунок 1.1 – Встановлення стандартного підключення за допомогою юзернейму та паролю

Для отримання токена доступу до ресурсу був використаний готовий код з лабораторної роботи № 2, який виконує POST запит до ендпоінту, отримавши access token. Відповідний код, за допомогою якого було створено звернення можна побачити на рисунках 1.2 – 1.3.

```
def get_access_token(url: str, payload: str) → str:
    connection = http.client.HTTPSConnection(url)
    headers = {'content-type': "application/json"}
    connection.request("POST", "/oauth/token", payload, headers)
    data = connection.getresponse().read()
    return json.loads(data.decode("utf-8"))['access_token']

def create_user(url: str, access_token: str, payload: str) → Tuple[str, int]:
    headers = {
        "Accept": "application/json",
        "Content-Type": "application/json",
        "Authorization": f"Bearer {access_token}"
    }

    connection = http.client.HTTPSConnection(url)
    connection.request("POST", "/api/v2/users", payload, headers)
    result = connection.getresponse()
    data = result.read()

    return data.decode("utf-8"), result.status
```

Рисунок 1.2 – Функції для створення користувача та отримання токена доступу

```
url = "dev-hfzb6seuth5jesyp.eu.auth0.com"

payload_token = json.dumps({
    "client_id": "sTuTja4U43sC1MyNrNaSZBJDN0gSurHF",
    "client_secret": "Kt85pHTXRZ_OnzhspNmTF-2jSVCvYdULQcILR-Y1sY6LM1Cb5qSC9DrhL1E1l6jC",
    "audience": "https://dev-hfzb6seuth5jesyp.eu.auth0.com/api/v2/",
    "grant_type": "client_credentials"
})

access_token = get_access_token(url, payload_token)

payload_user = json.dumps({
    "email": "denys.babich@test.com",
    "user_metadata": {},
    "blocked": False,
    "email_verified": False,
    "app_metadata": {},
    "given_name": "Denys",
    "family_name": "Babich",
    "name": "Denys",
    "nickname": "NgenX",
    "picture": "https://i.etsystatic.com/26254942/r/il/f54dcc/4754579609/il_300x300.4754579609_aps7.jpg",
    "connection": "Username-Password-Authentication",
    "password": "BabichDenysPassword123@",
    "verify_email": False,
})

response_data, status_code = create_user(url, access_token, payload_user)

print(f"CODE: {status_code}")
print(f"PAYLOAD: {response_data}")
```

Рисунок 1.3 – Основний модуль, у якому виконуються запити

```
CODE: 201
PAYLOAD: {"blocked":false,"created_at":"2024-10-05T20:51:11.197Z","email":"denys.babich@test.com","email_verified":false,"family_name":"Babich","given_name":"Denys","identities":[{"connection":"Username-Password-Authentication","user_id":"6701a6bf35224c86a3b9fd53","provider":"auth0","isSocial":false}],"name":"Denys","nickname":"NgenX","picture":"https://i.etsystatic.com/26254942/r/il/f54dcc/4754579609/il_300x300.4754579609_aps7.jpg","updated_at":"2024-10-05T20:51:11.197Z","user_id":"auth0|6701a6bf35224c86a3b9fd53","user_metadata":{}}
```

Рисунок 1.4 – Отриманий статус код 201, що означає успішне створення користувача

Для подальшого виконання роботи також потрібно додати варіанти аутентифікації до віддаленого ресурсу, тому були встановлені відповідні налаштування grant types, за допомогою яких визначається паплайн та способи обробки токенів доступу клієнта до віддаленого ресурсу. Отримані налаштування можна побачити на рисунку 1.5.

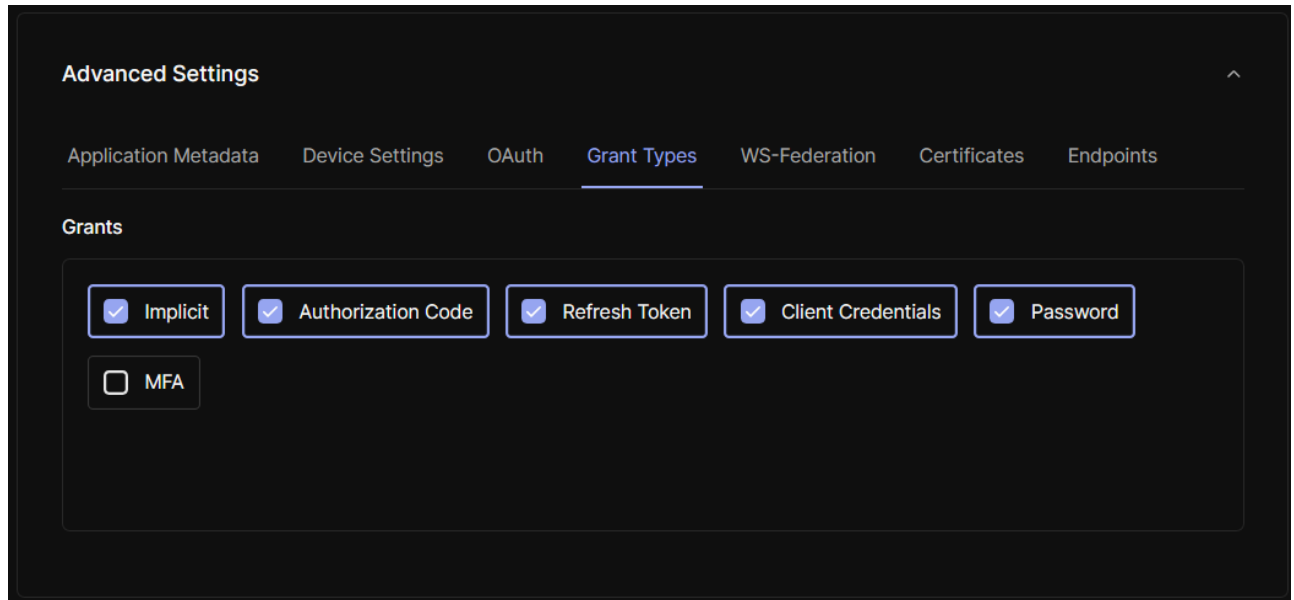


Рисунок 1.5 – Налаштування grant types для застосунку

Implicit Grant використовується у додатках, де токен доступу передається безпосередньо клієнту, без участі серверної частини. Це відбувається за сценарієм, коли користувач вводить свої облікові дані через сторінку авторизації і токен доступу (access token) повертається безпосередньо клієнту.

Authorization Code є найпоширенішим типом для серверних додатків, де токен доступу передається через код авторизації. Користувач вводить свої облікові дані на сторінці авторизації, після успішної авторизації сервер видає код авторизації і повертає його клієнту (зазвичай через редирект). Клієнт обмінює цей код на токен доступу та refresh token через сервер.

Refresh Token використовується для оновлення токена доступу після того, як час його дії закінчився, або став недійсним. Цей grant-тип зазвичай використовується разом із Authorization Code.

Client Credentials використовується для авторизації між сервісами без участі користувача, додаток сам себе авторизує за допомогою своїх облікових даних (client\_id і client\_secret), відправляючи їх на сервер авторизації.

Password передбачає, що клієнт отримує ім'я користувача і пароль шляхом реєстрації, або напряму від іншого користувача та отримує доступ до віддаленого ресурсу, шляхом авторизації через них.

Самі запити на отримання токена користувача та інші операції виконуватимуться за допомогою онлайн-сервісу Postman, який надає зручний графічний інтерфейс для створення та налаштування подібних запитів. Для отримання user-токену у тіло запиту передаються ім'я, пароль користувача, секретний заголовок запиту і доступ виконується з grant типом через пароль. Сам запит для отримання user-токену показаний на рисунку 1.6.

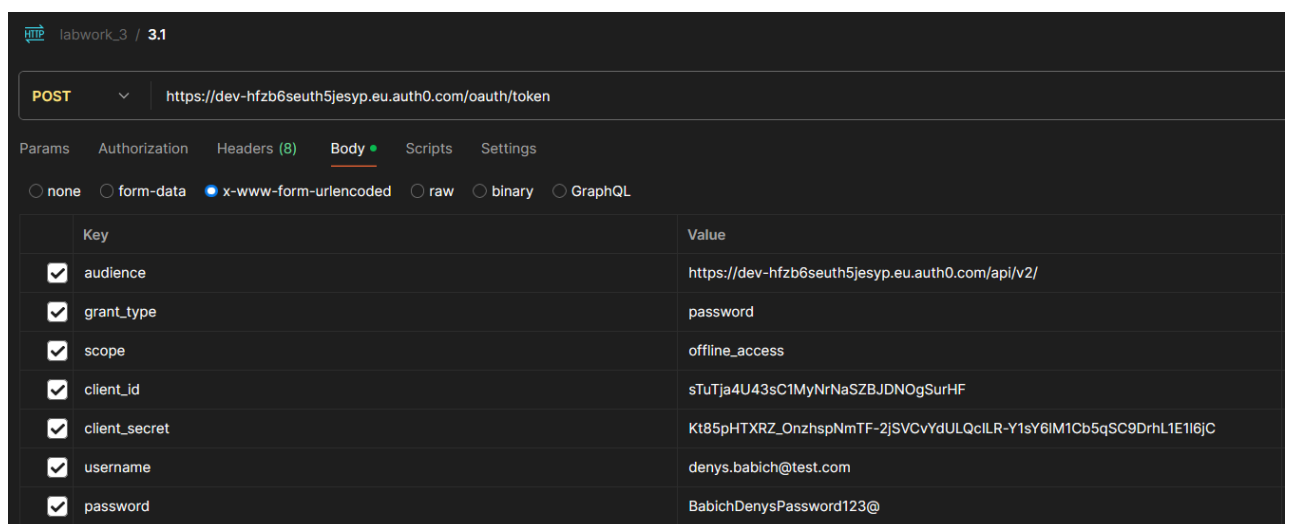


Рисунок 1.6 – Отримане тіло запиту до сервісу

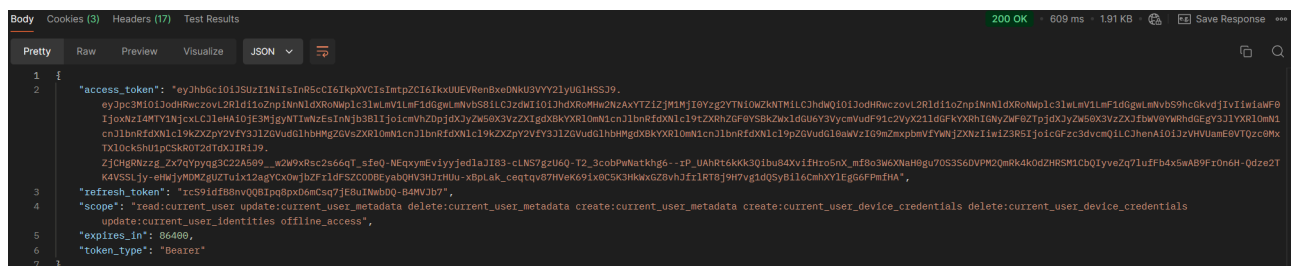


Рисунок 1.7 – Отримана успішна відповідь

Отримана відповідь від сервісу включає сам токен доступу, та інші поля, зокрема scope, яке вказує на отримані права доступу, які має наданий токен

доступу та refresh\_token, який використовується для отримання нового токenu доступу після того, як закінчиться термін дії поточного access\_token.

```
{
  "access_token":
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IkkxUUEVRenBxeDNkU3VY
    Y2lyUGlhSSJ9.eyJpc3MiOiJodHRwczovL2Rldi1oZnpiNnNldXRoNWplc3lwLmV1
    LmFldGgwLmNvbS8iLCJzdWUiOiJhdXRoMHw2NzAxYTZiZjM1MjI0Yzg2YTNi
    OWZkNTMiLCJhdWQiOiJodHRwczovL2Rldi1oZnpiNnNldXRoNWplc3lwLmV1L
    mFldGgwLmNvbS9hcGkvdjIvIiwiaWF0IjoxNzI4MTY1NjcxcjE4IiwiaGVhZCI6ImJyYXN
    TIwNzEsInNjb3BlIjoicmVhZDpjdXJyZW50X3VzZXIgdXBkYXRlOmN1cnJlbnRf
    dXNlcl9tZXRhZGF0YSBkZWxldGU6Y3VycmVudF91c2VyX21ldGFkYXRhIGNy
    ZWF0ZTpjdXJyZW50X3VzZXJfbWV0YWRhdGEgY3JlYXRlOmN1cnJlbnRfdXN
    lcl9kZXZpY2VfY3JlZGVudGlhbHMgZGVsZXRIOmN1cnJlbnRfdXNlcl9kZXZpY
    2VfY3JlZGVudGlhbHMgdXBkYXRlOmN1cnJlbnRfdXNlcl9pZGVudGl0aWVzIG9
    mZmxpbmVfYWNjZXNzIiwiaWF0IjoxNzI4MTY1NjcxcjE4IiwiaGVhZCI6ImJyYXN
    TQzc0MxTXlOck5hU1pCSkROT2dTdXJIRiJ9.ZjCHgRNzzg_Zx7qYpyqg3C22A50
    9__w2W9xRsc2s66qT_sfeQ-NEqsymEviyyjedlaJI83-cLNS7gzU6Q-T2_3cobPwNat
    khg6--rP_UAhRt6kKk3Qibu84XvifHro5nX_mf8o3W6XNaH0gu7OS3S6DVPM2Q
    mRk4kOdZHRSM1CbQIyveZq7lufFb4x5wAB9FrOn6H-Qdze2TK4VSSLjy-eHWjy
    MDMZgUZTuix12agYCxOwjbZFrldFSZCODBEyabQHV3HJrHUu-xBpLak_ceqtq
    v87HVeK69ix0C5K3HkWxGZ8vhJfrlRT8j9H7vgldQSyBil6CmhXYIEgG6FPmfH
    A",
  "refresh_token": "rcS9idfB8nvQQBIpq8pxD6mCsq7jE8uINwbDQ-B4MVJb7",
  "scope":
    "read:current_user                                update:current_user_metadata
    delete:current_user_metadata                      create:current_user_metadata
    create:current_user_device_credentials             delete:current_user_device_credentials
    update:current_user_identities offline_access",
  "expires_in": 86400,
  "token_type": "Bearer"
```

}

## 2. Отримати оновлений токен використовуючи refresh-token grant type.

Як уже було зазначено, refresh token використовується для отримання нового токenu доступу, коли закінчується термін дії старого. Для виконання такої операції можна побудувати запит, який показаний на рисунку 1.8.

Key	Value
audience	https://dev-hfzb6seuth5jesyp.eu.auth0.com/api/v2/
grant_type	refresh_token
client_id	sTuTja4U43sC1MyNrNaSZBJDNOgSurHF
client_secret	Kt85pHTXRZ_OnzhspNmTF-2jSVCvYdULQcILR-Y1sY6IM1Cb5qSC9DrhL1E1l6JC
refresh_token	rcS9ldfB8nvQQBlpq8pxD6mCsQ7JE8ulNwbDQ-B4MVJb7

Рисунок 1.8 – Запит для отримання токenu доступу за допомогою refresh-токenu

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImlpZCI6Ikh0UEVrenBxeDNUU3VYV2lyUGlHSSJ9.eyJpc3MiOiJodHRwczovL2RldiI0ZnpInN1dXRoNwplc3lWbVlMfDd6gUmlNbS9hcGkvZjIvIiwiaWF0IjoxNjI0MTY1OTYxLjE1e30sInR5cCI6IkpXVCIsImlpZCI6Ikh0UEVrenBxeDNUU3VYV2lyUGlHSSJ9.eyJpc3MiOiJodHRwczovL2RldiI0ZnpInN1dXRoNwplc3lWbVlMfDd6gUmlNbS9hcGkvZjIvIiwiaWF0IjoxNjI0MTY1OTYxLjE1e30sInR5cCI6IkpXVCIsImlpZCI6Ikh0UEVrenBxeDNUU3VYV2lyUGlHSSJ9",
  "expires_in": 86400,
  "token_type": "Bearer"
}
```

Рисунок 1.9 – Отримана успішна відповідь на продовження авторизації

Тіло такої відповіді так само включає сам access-токен та область дії токenu, яка вказує на отримані права доступу.

### Додаткове завдання:

#### 1. Зробити запит до API для зміни пароля, токен має бути використаний з прикладу client\_credential grant прикладу.

Для того, щоб оновити пароль буде використаний PATCH-запит, проте спочатку необхідно отримати айді клієнта, який може бути здобутий завдяки запиту через токenu доступу, який був зроблений з додаванням grant-типів openid та profile. OpenID вказує на те, що застосунок хоче виконати

автентифікацію користувача (ID-токен). У свою чергу, `profile` означає те, що у тілі відповіді має бути така додаткова інформація, як дані користувача (ім'я, прізвище, аватар, тощо). Відповідний запит можна побачити на рисунку 1.10.

labwork\_3 / 3.3.1

POST

https://dev-hfzb6seuth5jesyp.eu.auth0.com/oauth/token

Params

Authorization

Headers (8)

Body

Scripts

Settings

☐ none

☐ form-data

☒ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

Key	Value
<input checked="" type="checkbox"/> audience	https://dev-hfzb6seuth5jesyp.eu.auth0.com/api/v2/
<input checked="" type="checkbox"/> grant_type	password
<input checked="" type="checkbox"/> scope	offline_access openid profile
<input checked="" type="checkbox"/> client_id	sTuTja4U43sC1MyNrNaSZBJDNOgSurHF
<input checked="" type="checkbox"/> client_secret	Kt85pHTXRZ_OnzhspNmTF-2JSVCvYdULQcILR-Y1sY6IM1Cb5qSC9DrhL1E1I6jC
<input checked="" type="checkbox"/> username	denys.babich@test.com
<input checked="" type="checkbox"/> password	BabichDenysPassword123@

Рисунок 1.10 – Отримане тіло запиту

[illegible]

Рисунок 1.11 – Успішна відповідь від застосунку

Для отримання `userId` необхідно використати `access_token` з відповіді на рисунку 1.11 та підставити у тіло запиту на endpoint з ідентифікатором `/userinfo`. Тіло запиту наведено на рисунку 1.12.

The screenshot shows a REST client interface with a GET request to `https://dev-hfzb6seuth5jesyp.eu.auth0.com/userinfo`. The response body is shown in JSON format:

```
{  "audience": "https://dev-hfzb6seuth5jesyp.eu.auth0.com/api/v2/",  "grant_type": "password",  "client_id": "sTuTja4U43sC1MyNrNaSZBJDNOgSurHF",  "client_secret": "Kt85pHTXRZ_OnzhspNmTF-2jSVCvYdULQcILR-Y1sY6IM1Cb5qSC9DrhL1E1I6JC",  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9ImltpmtpZCI6IkxUUUVRenBxeDNkU3VYYy2lyUGIH..."}
```



Рисунок 1.12 – Тіло запиту для отримання userId

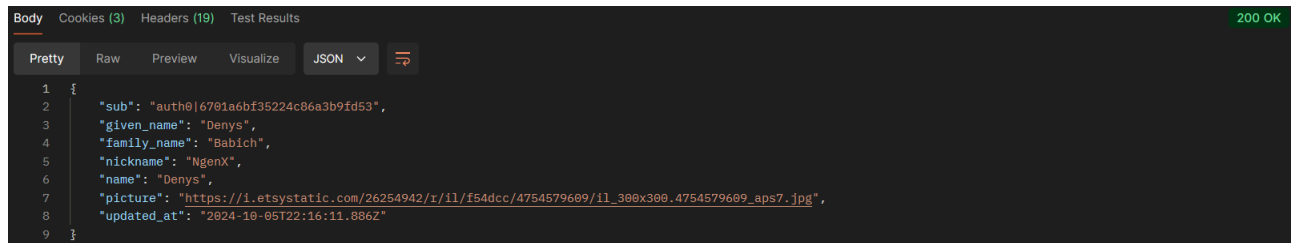


Рисунок 1.13 – Отримана успішна відповідь

```
{
  "sub": "auth0|6701a6bf35224c86a3b9fd53",
  "given_name": "Denys",
  "family_name": "Babich",
  "nickname": "NgenX",
  "name": "Denys",
  "picture":
    "https://i.etsystatic.com/26254942/r/il/f54dcc/4754579609/il_300x300.4754579609_aps7.jpg",
  "updated_at": "2024-10-05T22:16:11.886Z"
}
```

Токен доступу для зміни паролю буде виконаний не через самого користувача, а за допомогою grant-типу `client_credentials`, який використовується, коли додаток прагне виконати доступ до ресурсу не від імені користувача, а від себе, зазвичай, така поведінка застосовується для комунікації напряду між клієнтським застосунком та ендпоінтом, що корисно, оскільки не вимагає залучення користувача.

Хоча є й інші альтернативні способи зміни паролю, навіть більш логічні та безпечні, проте, такий доступ через `client_credentials` може застосовуватися у випадку, якщо користувач забув власний пароль й після підтвердження своєї особи альтернативними засобами (через пошту, мобільний телефон, тощо) хоче відновити доступ до свого акаунту.

Запит на отримання токєну доступу можна побачити на рисунку 1.14

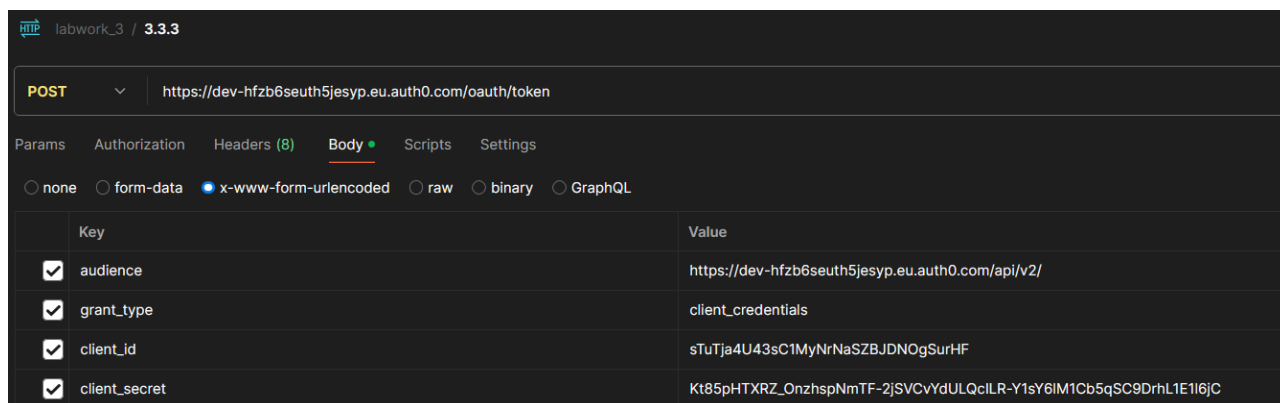


Рисунок 1.14 – Тіло запиту

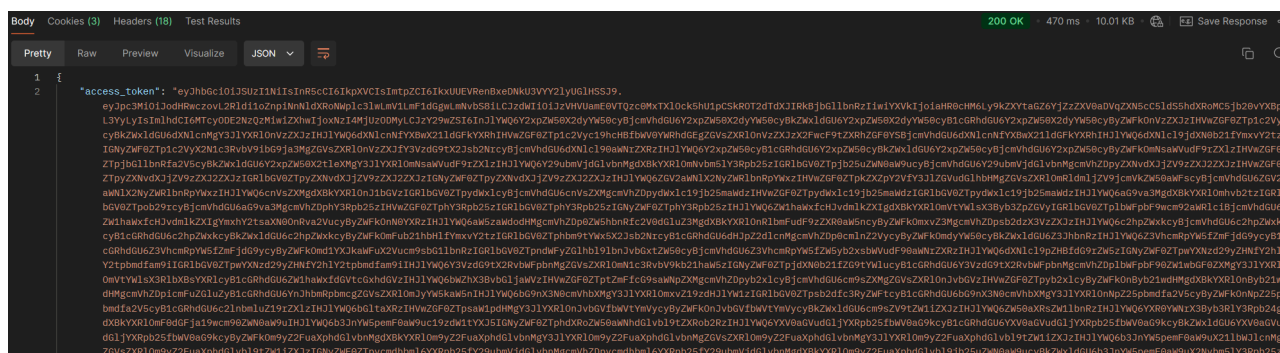


Рисунок 1.15 – Успішна відповідь від серверу

Тепер цей отриманий сесійний токен використовуватиметься для безпосередньої зміни паролю за допомогою PATCH-запиту, де у заголовок у поле `Authorization` передається отримане значення access-токену. Тіло з корисним навантаженням, яке зберігає поле паролю наведено на рисунку 1.16.

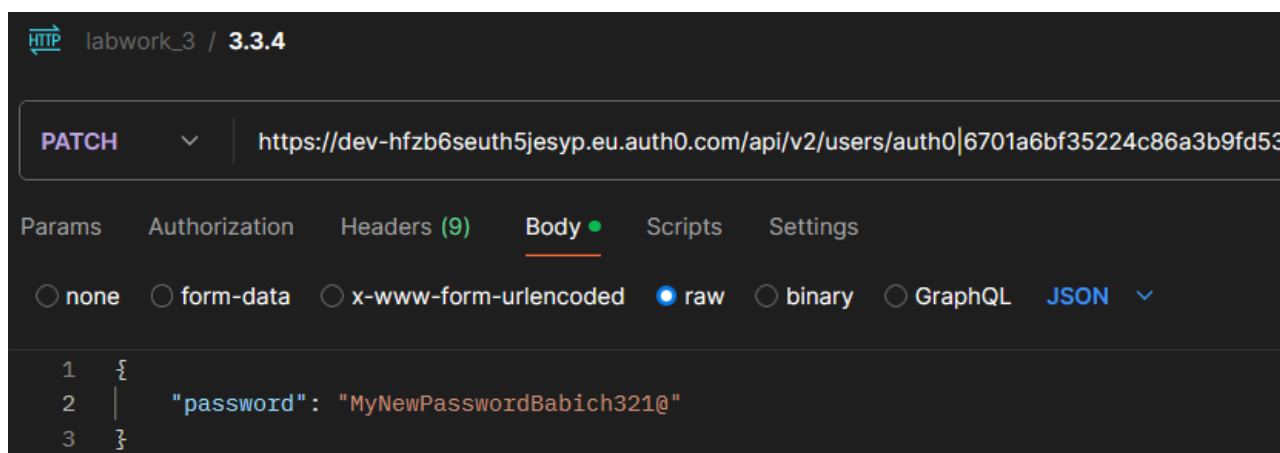


Рисунок 1.16 – Тіло запиту з новим паролем

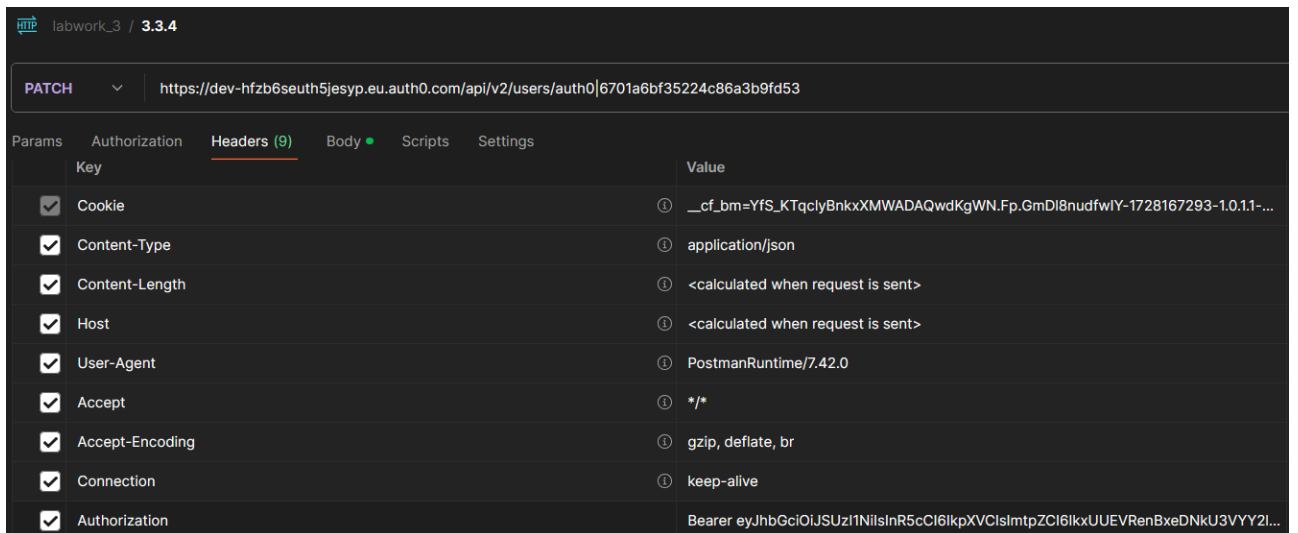


Рисунок 1.17 – Заголовок запиту, де у Authorization занесено токен доступу



Рисунок 1.18 – Отримана успішна відповідь на зміну паролю

Тепер можна виконати перевірку доступу до застосунку, спочатку буде перевірений старий пароль, тіло запиту та негативна відповідь зі статус-кодом 403 наведено на рисунку 1.19.



**Висновок:** У цій лабораторній роботі було успішно засвоєно основні навички роботи з OAuth2 авторизаційним протоколом, зокрема з використанням сервісу Auth0 для керування доступом до ресурсів. Запити та відповіді сервісу оброблялися за допомогою Postman, що дозволило вивчити структуру запитів та отриманих відповідей.

Основним завданням було отримання токена доступу (access token) та його використання для аутентифікації користувача і взаємодії з ресурсами через API. У ході роботи були розглянуті різні варіанти grant types, включаючи Password, Authorization Code, Refresh Token та Client Credentials. Особливу увагу було приділено сценаріям, коли токен доступу передається безпосередньо клієнту або ж через серверну частину. Зокрема, детально описано процес отримання нового токена через Refresh Token після закінчення терміну дії попереднього токена доступу.

Також у рамках додаткового завдання було продемонстровано застосування Client Credentials для зміни пароля користувача. У результаті роботи було виконано успішну зміну пароля користувача, підтверджену отриманими статус-кодами та відповідями сервісу, що свідчить про правильну реалізацію авторизації та керування доступом до ресурсів.