**Хід роботи**

1. Підняти локально IAM (Casdoor) (Docker).

З використанням Docker було запущено такі контейнери, як Casdoor, база даних MySQL для Casdoor та nginx. Nginx використовується, щоб забезпечити безпечний зв’язок з casdoor з використанням TLS. Сертифікати було згенеровано за допомогою OpenSSL.

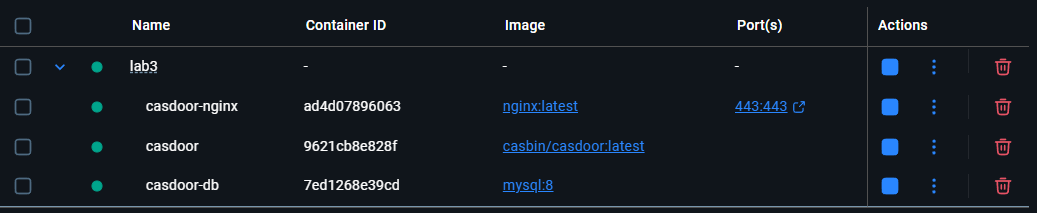


Рисунок 3.1 – Запущені контейнери

**Лістинг 3.1. docker-compose.yaml**

services:

db:

image: mysql:8

container\_name: casdoor-db

environment:

- MYSQL\_ROOT\_PASSWORD=123456

- MYSQL\_DATABASE=casdoor

casdoor:

image: casbin/casdoor:latest

container\_name: casdoor

environment:

- RUNNING\_IN\_DOCKER=true

- dataSourceName=root:123456@tcp(db:3306)/

volumes:

- ./conf:/conf/

depends\_on:

- db

restart: on-failure

nginx:

image: nginx:latest

container\_name: casdoor-nginx

ports:

- "443:443"

volumes:

- ./conf/nginx.conf:/etc/nginx/nginx.conf:ro

- ./conf/cert.pem:/conf/cert.pem:ro

- ./conf/key.pem:/conf/key.pem:ro

depends\_on:

- casdoor

restart: on-failure

**Лістинг 3.2. nginx.conf**

events { }

http {

server {

listen 443 ssl;

server\_name localhost;

ssl\_certificate /conf/cert.pem;

ssl\_certificate\_key /conf/key.pem;

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_ciphers HIGH:!aNULL:!MD5;

location / {

proxy\_pass http://casdoor:8000;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_set\_header X-Forwarded-Host $host;

proxy\_set\_header X-Forwarded-Port $server\_port;

proxy\_redirect off;

proxy\_set\_header Connection "";

}

}

}

**Лістинг 3.3. app.conf (конфіг-файл Casdoor)**

httpport = 8000

appname = casdoor

runmode = dev

copyrequestbody = true

driverName = mysql

dbName = casdoor

tableNamePrefix =

showSql = false

redisEndpoint =

defaultStorageProvider =

isCloudIntranet = false

authState = "casdoor"

socks5Proxy = "127.0.0.1:10808"

verificationCodeTimeout = 10

initScore = 0

logPostOnly = true

isUsernameLowered = false

origin =

originFrontend =

staticBaseUrl = "https://cdn.casbin.org"

isDemoMode = false

batchSize = 100

enableErrorMask = false

enableGzip = true

inactiveTimeoutMinutes =

ldapServerPort = 389

ldapsCertId = ""

ldapsServerPort = 636

radiusServerPort = 1812

radiusDefaultOrganization = "built-in"

radiusSecret = "secret"

quota = {"organization": -1, "user": -1, "application": -1, "provider": -1}

logConfig = {"adapter":"file", "filename": "logs/casdoor.log", "maxdays":99999, "perm":"0770"}

initDataNewOnly = false

initDataFile = "./init\_data.json"

frontendBaseDir = "../cc\_0"

1. Налаштування Casdoor.

Спочатку було створено нову організацію university. Після цього було додано новий застосунок labs, де було налаштовано Redirect URLs відповідно до бекенду веб застосунку. Також був доданий новий користувач, який належить організації university та застосунку labs.

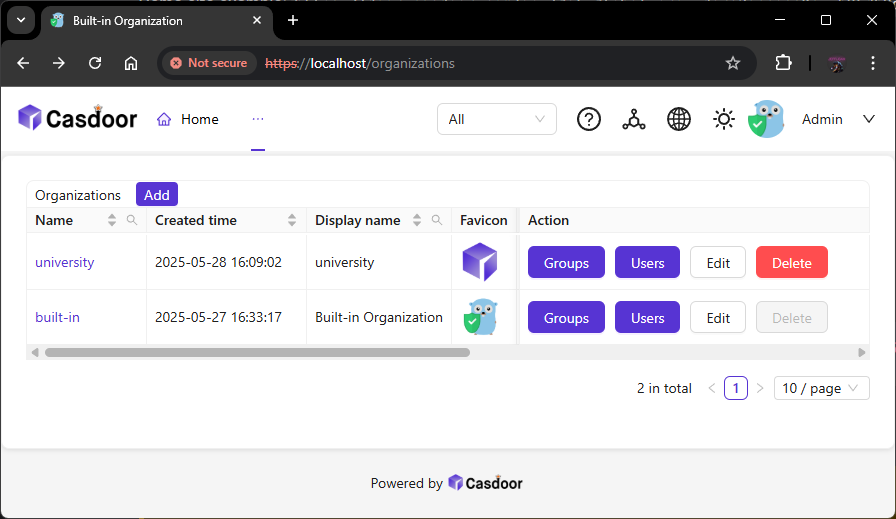


Рисунок 3.2 – Нова організація

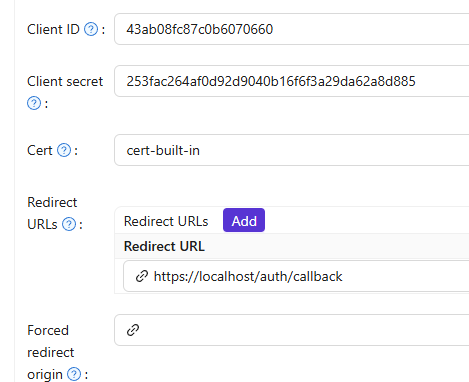


Рисунок 3.3 – Налаштування Redirect URLs

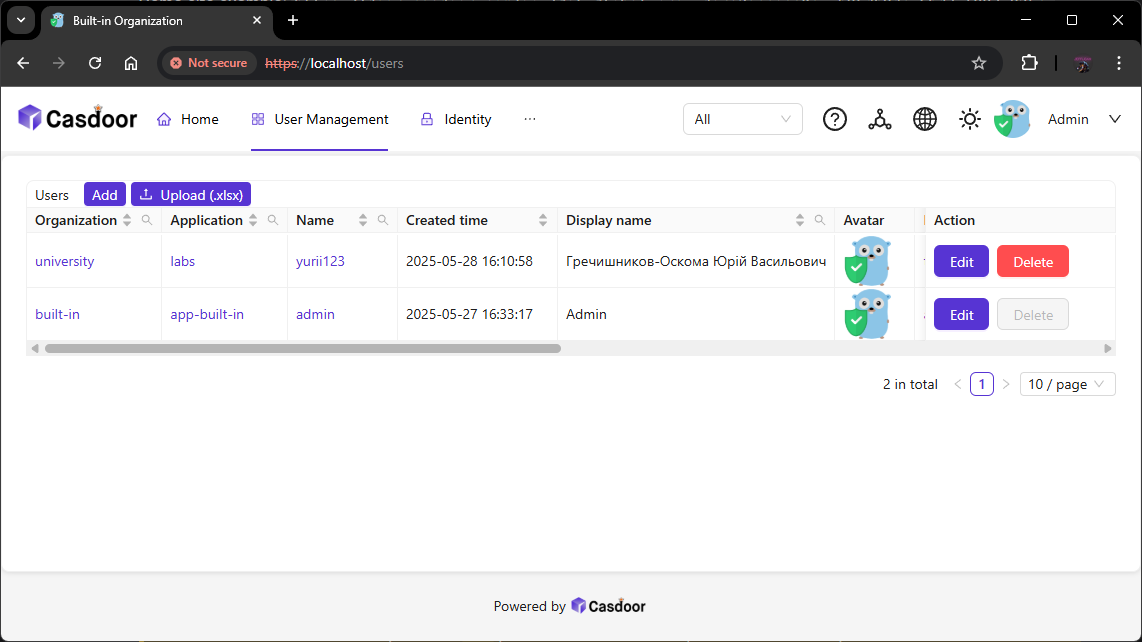


Рисунок 3.4 – Новий користувач у списку

1. Створення веб-застосунку.

Для створення програми було використано [ASP.NET](http://asp.net). Клас AuthController відповідає за логін. Коли фронтенд надсилає запит **auth/login**, то відбувається перенаправлення на Casdoor для авторизації. Якщо авторизація проходить успішно, то сервер запитує access token, який зберігається у cookie, після чого відбувається повернення до початкової сторінки. Якщо користувач авторизований, то він може отримати конфіденційну інформацію. Щоб отримати інформацію про користувача – надсилається запит **user/data**, який перевіряє access token за допомогою Casdoor і якщо він правильний, то інформація з jwt токену декодується і повертається користувачеві.

Параметри OIDC зберігаються у файлі appsettings.json.

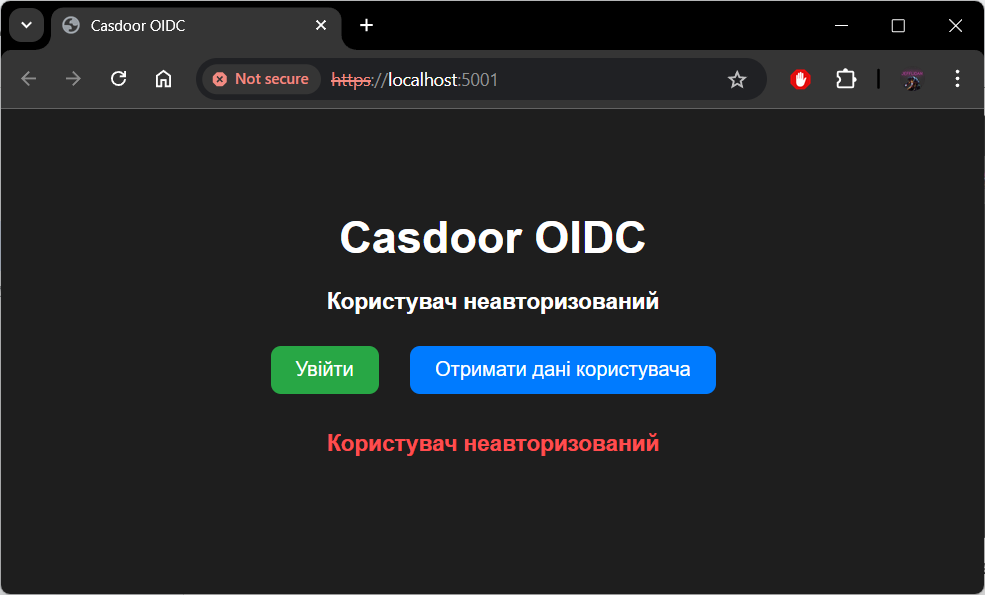


Рисунок 3.5 – Головна сторінка з помилкою, тому що користувач намагався отримати дані без попередньої авторизації

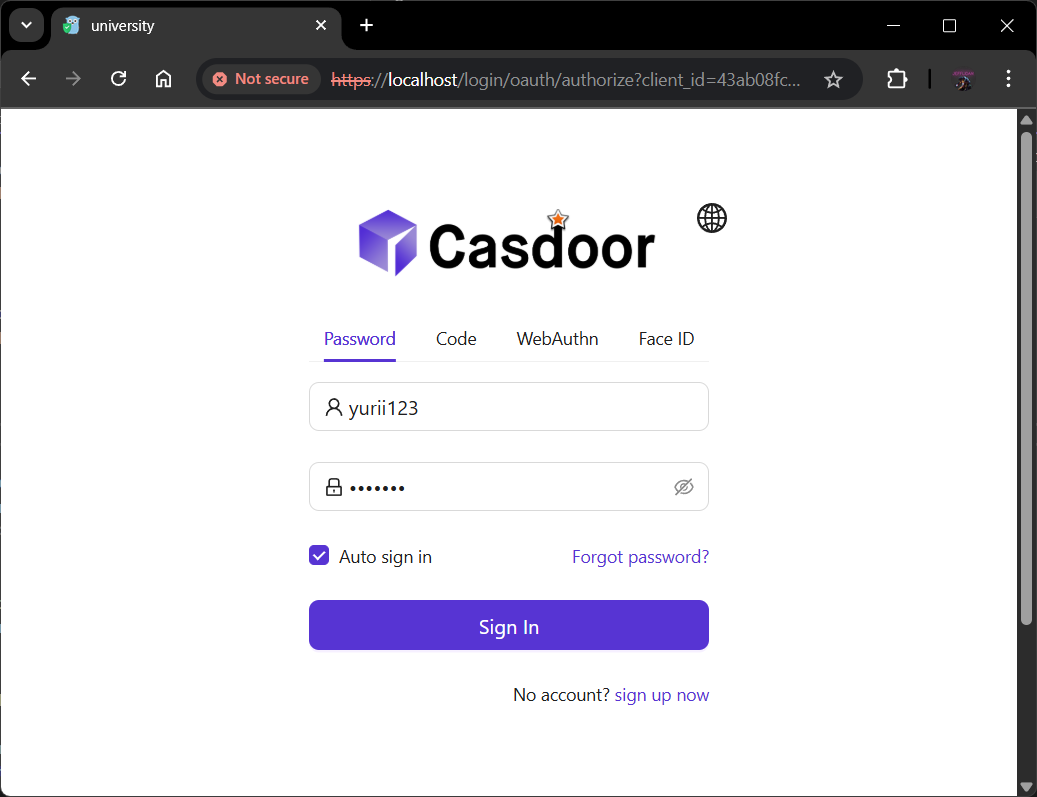


Рисунок 3.6 – Сторінка авторизації Casdoor, на яку користувача була перенаправлено після натискання на кнопку “Увійти”

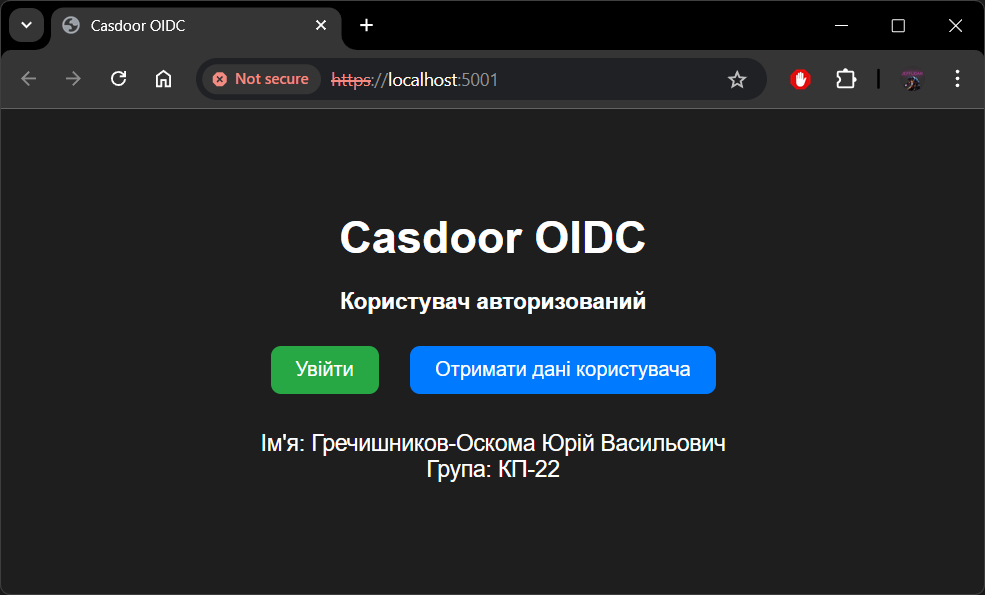


Рисунок 3.7 – Головная сторінка після авторизації та натискання на кнопку “Отримати дані”

**Лістинг 3.4. Файл Program.cs**

using Microsoft.AspNetCore.Authentication.Cookies;

using System.Security.Cryptography.X509Certificates;

var builder = WebApplication.CreateBuilder(args);

var cert = new X509Certificate2("certs/devcert.pfx", "devpass");

builder.WebHost.ConfigureKestrel(options =>

{

options.ListenLocalhost(5001, listenOptions =>

{

listenOptions.UseHttps(cert);

});

});

builder.Services.AddAuthentication(options =>

{

options.DefaultScheme = CookieAuthenticationDefaults.AuthenticationScheme;

})

.AddCookie();

builder.Services.AddAuthorization();

builder.Services.AddControllers();

builder.Services.Configure<OAuthOptions>(builder.Configuration.GetSection("OAuth"));

var app = builder.Build();

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.MapFallbackToFile("index.html");

app.Run();

**Лістинг 3.5 Файл AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.WebUtilities;

using Microsoft.Extensions.Options;

using System.Text.Json;

[ApiController]

[Route("auth")]

public class AuthController : ControllerBase

{

private readonly OAuthOptions \_oauthOptions;

public AuthController(IOptions<OAuthOptions> options)

{

\_oauthOptions = options.Value;

}

[HttpGet("login")]

public IActionResult Login([FromQuery] string returnUrl = "/")

{

var redirectUri = Url.Action(nameof(Callback), "Auth", null, Request.Scheme)!;

var authorizeUrl = QueryHelpers.AddQueryString(

\_oauthOptions.AuthorizeEndpoint,

new Dictionary<string, string?>

{

["client\_id"] = \_oauthOptions.ClientId,

["redirect\_uri"] = redirectUri,

["response\_type"] = "code",

["scope"] = \_oauthOptions.Scope,

["state"] = returnUrl

});

return Redirect(authorizeUrl);

}

[HttpGet("callback")]

public async Task<IActionResult> Callback([FromQuery] string code, [FromQuery] string? state)

{

var redirectUri = Url.Action(nameof(Callback), "Auth", null, Request.Scheme)!;

using var http = new HttpClient();

var tokenRequest = new HttpRequestMessage(HttpMethod.Post, \_oauthOptions.TokenEndpoint)

{

Content = new FormUrlEncodedContent(new Dictionary<string, string>

{

["grant\_type"] = "authorization\_code",

["code"] = code,

["redirect\_uri"] = redirectUri,

["client\_id"] = \_oauthOptions.ClientId,

["client\_secret"] = \_oauthOptions.ClientSecret

})

};

var tokenResponse = await http.SendAsync(tokenRequest);

tokenResponse.EnsureSuccessStatusCode();

var payload = await tokenResponse.Content.ReadAsStringAsync();

using var doc = JsonDocument.Parse(payload);

if (!doc.RootElement.TryGetProperty("access\_token", out var accessTokenElement))

return BadRequest("No access\_token found in the token response.");

string accessToken = accessTokenElement.GetString() ?? string.Empty;

var cookieOptions = new CookieOptions

{

HttpOnly = true,

Secure = true,

SameSite = SameSiteMode.Lax,

Expires = DateTimeOffset.UtcNow.AddHours(1)

};

Response.Cookies.Append("access\_token", accessToken, cookieOptions);

var returnUrl = !string.IsNullOrEmpty(state) ? state! : "/";

return Redirect(returnUrl);

}

}

**Лістинг 3.6. Файл UserDataController.сs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Options;

using System.Net.Http.Headers;

[ApiController]

[Route("user")]

public class UserDataController : ControllerBase

{

private readonly OAuthOptions \_oauthOptions;

public UserDataController(IOptions<OAuthOptions> options)

{

\_oauthOptions = options.Value;

}

[HttpGet("data")]

public async Task<IActionResult> GetUserData()

{

if (!Request.Cookies.TryGetValue("access\_token", out var accessToken) || string.IsNullOrEmpty(accessToken))

{

return Unauthorized(new { error = "Access token missing" });

}

try

{

var payloadJson = JwtDecoder.DecodeJwtPayload(accessToken);

using var httpClient = new HttpClient();

var response = await httpClient.GetAsync(\_oauthOptions.JwtVerifyEndpoint);

var jwks = await response.Content.ReadAsStringAsync();

if (!JwtDecoder.ValidateJwt(accessToken, jwks))

{

Console.WriteLine("Invalid or expired access token according to Casdoor");

return Unauthorized(new { error = "Invalid or expired access token" });

}

return Ok(payloadJson.RootElement);

}

catch (Exception ex)

{

Console.WriteLine("Error decoding JWT or validating token: " + ex.Message);

return Unauthorized(new { error = "Invalid access token" });

}

}

}

**Лістинг 3.7. Файл JwtDecoder.cs**

using System.IdentityModel.Tokens.Jwt;

using System.Security.Cryptography;

using System.Text;

using System.Text.Json;

using Microsoft.IdentityModel.Tokens;

public class JwtDecoder

{

public static JsonDocument DecodeJwtPayload(string jwt)

{

if (string.IsNullOrEmpty(jwt))

throw new ArgumentException("JWT token is null or empty");

string[] parts = jwt.Split('.');

if (parts.Length != 3)

throw new ArgumentException("Invalid JWT token format");

string payload = parts[1];

string json = Base64UrlDecode(payload);

return JsonDocument.Parse(json);

}

public static bool ValidateJwt(string jwt, string jwksJson)

{

if (string.IsNullOrWhiteSpace(jwt))

throw new ArgumentException("JWT token is null or empty", nameof(jwt));

if (string.IsNullOrWhiteSpace(jwksJson))

throw new ArgumentException("JWKS JSON is null or empty", nameof(jwksJson));

using JsonDocument jwksDoc = JsonDocument.Parse(jwksJson);

var keys = jwksDoc.RootElement.GetProperty("keys");

var handler = new JwtSecurityTokenHandler();

var jwtToken = handler.ReadJwtToken(jwt);

string kid = jwtToken.Header.Kid;

var keyElement = keys.EnumerateArray()

.FirstOrDefault(k => k.TryGetProperty("kid", out var kKid) && kKid.GetString() == kid);

if (keyElement.ValueKind == JsonValueKind.Undefined)

{

keyElement = keys[0];

}

// Extract modulus (n) and exponent (e)

string n = keyElement.GetProperty("n").GetString();

string e = keyElement.GetProperty("e").GetString();

RSA rsa = CreateRsaProviderFromParameters(n, e);

var validationParameters = new TokenValidationParameters

{

ValidateIssuer = false,

ValidateAudience = false,

ValidateLifetime = true,

RequireSignedTokens = true,

ValidateIssuerSigningKey = true,

IssuerSigningKey = new RsaSecurityKey(rsa)

};

try

{

handler.ValidateToken(jwt, validationParameters, out SecurityToken validatedToken);

return true;

}

catch

{

return false;

}

}

private static string Base64UrlDecode(string input)

{

string output = input.Replace('-', '+').Replace('\_', '/');

switch (output.Length % 4)

{

case 2: output += "=="; break;

case 3: output += "="; break;

}

byte[] buffer = Convert.FromBase64String(output);

return Encoding.UTF8.GetString(buffer);

}

private static byte[] Base64UrlDecodeBytes(string input)

{

string output = input.Replace('-', '+').Replace('\_', '/');

switch (output.Length % 4)

{

case 2: output += "=="; break;

case 3: output += "="; break;

}

return Convert.FromBase64String(output);

}

private static RSA CreateRsaProviderFromParameters(string n, string e)

{

var rsa = RSA.Create();

rsa.ImportParameters(new RSAParameters

{

Modulus = Base64UrlDecodeBytes(n),

Exponent = Base64UrlDecodeBytes(e)

});

return rsa;

}

}

**Лістинг 3.8. Файл OAuthOptions.cs**

public class OAuthOptions

{

public string ClientId { get; set; }

public string ClientSecret { get; set; }

public string AuthorizeEndpoint { get; set; }

public string TokenEndpoint { get; set; }

public string Scope { get; set; }

public string JwtVerifyEndpoint { get; set; }

}

**Лістинг 3.9. Файл appsettings.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"OAuth": {

"ClientId": "43ab08fc87c0b6070660",

"ClientSecret": "253fac264af0d92d9040b16f6f3a29da62a8d885",

"AuthorizeEndpoint": "https://localhost/login/oauth/authorize",

"TokenEndpoint": "https://localhost/api/login/oauth/access\_token",

"Scope": "openid profile email",

"JwtVerifyEndpoint": "https://localhost/.well-known/jwks"

}

}

**Лістинг 3.10. Файл index.html**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="UTF-8">

<title>Casdoor OIDC</title>

<style>

body {

background-color: #1e1e1e;

color: #ffffff;

text-align: center;

padding-top: 50px;

font-size: 18px;

}

h1 {

margin-bottom: 30px;

}

button {

padding: 10px 20px;

font-size: 16px;

margin: 10px;

border: none;

border-radius: 8px;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

opacity: 0.9;

}

button.login {

background-color: #28a745;

color: white;

}

button.data {

background-color: #007bff;

color: white;

}

#output.error {

color: #ff4d4f;

font-weight: bold;

font-family: 'Roboto', sans-serif;

}

#output.success {

color: #ffffff;

white-space: pre-wrap;

font-family: 'Roboto', sans-serif;

}

</style>

</head>

<body>

<h1>Casdoor OIDC</h1>

<button class="login" onclick="login()">Увійти</button>

<button class="data" onclick="getData()">Отримати дані</button>

<pre id="output"></pre>

<script>

async function login() {

window.location.href = "/auth/login";

}

async function getData() {

const output = document.getElementById('output');

output.className = '';

output.textContent = 'Loading...';

try {

const response = await fetch('/user/data', { method: 'GET' });

if (!response.ok) {

const errorData = await response.json();

output.className = 'error';

output.textContent = `Помилка: користувач неавторизований`;

return;

}

const data = await response.json();

output.className = 'success';

output.textContent = `Ім'я: ${data.displayName || 'Unknown'}\nГрупа: ${data.affiliation || 'Unknown'}`;

}

catch (e) {

output.className = 'error';

output.textContent = 'Error: ' + e.message;

}

}

</script>

</body>

</html>

**Висновки:** розробив застосунок з авторизацією з підтримкою автентифікації та авторизації на основі протоколів OAuth 2.0 та OpenID Connect та з використанням Casdoor, який було розгорнуто у Docker з разом з nginx для того, щоб забезпечити безпечну взаємодію. Бекенд частина застосунку була реалізована з використанням ASP.NET. Авторизація здійснюється через JWT-токен, який зберігається у cookie після успішного входу користувача.