Міністерство освіти і науки України

Державний університет «Житомирська політехніка»

Факультет інформаційно-комп'ютерних технологій

Кафедра комп’ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему: «Проєкт підсистеми автентифікації онлайн-магазину на базі протоколу Open ID Connect та різних провайдерів ідентифікації»

Виконав студент 4-го курсу групи КІ-20-1  
спеціальності 123 «Комп’ютерна інженерія»

Владислав Мурсалов

Керівник посада, науковий ступінь, вчене звання,

Олена Головня

Рецензент посада, науковий ступінь, вчене звання

Імʼя Прізвище

Житомир – 2023

Державний університет «Житомирська політехніка»

Факультет інформаційно-комп’ютерних технологій

Кафедра комп’ютерної інженерії та кібербезпеки

Спеціальність 123 «Комп’ютерна інженерія»

Освітня програма «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач   
кафедри компʼютерної   
інженерії та кібербезпеки

\_\_\_\_\_\_\_\_\_ Андрій Єфіменко

1 березня 2022 р.

**ЗАВДАННЯ**

на випускну кваліфікаційну роботу бакалавра

Студент: Мурсалов Владислав Радікович

Тема роботи: «Проєкт підсистеми автентифікації онлайн-магазину на базі протоколу Open ID Connect та різних провайдерів ідентифікації»

затверджена Наказом університету від 1 березня 2023 р. № \_\_\_\_

Термін здачі студентом закінченої роботи 10 червня 2023 р.

Вихідні дані роботи (зазначається предмет і об’єкт дослідження):   
вихідні дані для роботи;

Обладнання та програмне забезпечення:

1. … (Обладнання, операційні системи, програмне забезпечення тощо)

2. …

3. …

4. …

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Консультант | Підпис, дата | |
| Завдання видав | Завдання прийняв |
| 1 | Керівник чи інші |  |  |
| 2 | викладачі, які |  |  |
| 3 | вас консультували |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Імʼя Прізвище

Керівник \_\_\_\_\_\_\_\_\_\_\_ Імʼя Прізвище

**Календарний план**

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назви етапів випускної кваліфікаційної роботи | Термін виконання етапів роботи | Примітка |
| 1 | Постановка завдання |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Владислав Мурсалов

Керівник \_\_\_\_\_\_\_\_\_\_\_ Олена Головня

АНОТАЦІЯ

Завданням кваліфікаційної роботи була розробка проекту підсистеми автентифікації онлайн-магазину на базі протоколу Open ID Connect та різних провайдерів ідентифікації.

Пояснювальна записка до курсового розробка Проекту інфраструктури для онлайн-магазину складається з вступу, трьох розділів, висновків, списку використаної літератури та додатків.

Текстова частина викладена на 27 сторінках друкованого тексту. Список використаних джерел містить 12 найменувань і займає 1 сторінку. В роботі наведено 31 рисунки. Загальний обсяг роботи – 30 сторінок.

Метою кваліфікаційної роботи є прозробка проєкт підсистеми автентифікації онлайн-магазину на базі протоколу Open ID Connect та різних провайдерів ідентифікації

В результаті було розроблено проект, який за допомогою протоколу OIDC здатний ідентифікувати користувачів інтернет магазину за допомогою різних провайдерів ідентифікації.

Ключові слова: OIDC, SAML, ASP.NET MVC, OKTA, AUTH0.

ANNOTATION

The task of the qualification work was to develop a project of an online store authentication system based on the Open ID Connect protocol and various identity providers.

The explanatory note to the course development of the Infrastructure Project for an online store consists of an introduction, three chapters, conclusions, a list of references and appendices.

The text part is set out on 27 pages of printed text. The list of references includes 12 titles and occupies 1 page. The paper contains 31 figures. The total volume of the work is 30 pages.

The purpose of the qualification work is to design an online store authentication subsystem based on the Open ID Connect protocol and various identification providers.

As a result, a project was developed that is able to identify users of an online store using the OIDC protocol with the help of various identification providers.

Keywords: OIDC, SAML, ASP.NET MVC, OKTA, AUTH0.

ЗМІСТ

[ВСТУП 8](#_Toc168260437)

[РОЗДІЛ 1 АНАЛІЗ СИСТЕМ АВТЕНТИФІКАЦІЇ іМАЗАЗИНІВ 9](#_Toc168260438)

[1.1 Недоліки традиційної системи автентифікації 9](#_Toc168260439)

[1.2 Протоколи для реалізації Single sign-on. SAML vs OIDC 10](#_Toc168260440)

[1.3 Детальний огляд роботи OIDC 14](#_Toc168260441)

[1.4 Огляд найпопулярніших IdP 14](#_Toc168260442)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ РІШЕННЯ 15](#_Toc168260443)

[2.2 Назва підрозділу 15](#_Toc168260444)

[РОЗДІЛ 3 ТЕСТУВАННЯ ІНФРАСТРУКТУРИ 18](#_Toc168260445)

[3.1 Тестування роботоздатності інтернет магазину 18](#_Toc168260446)

[3.2. Налаштування Okta як серверу автентифіації 20](#_Toc168260447)

[3.3. Налаштування Auth0 як серверу автентифіації 26](#_Toc168260448)

[3.4 Тестування автентифікації за допомогою Okta 30](#_Toc168260449)

[3.5 Тестування автентифікації за допомогою Auth0 32](#_Toc168260450)

[ВИСНОВКИ 35](#_Toc168260451)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 36](#_Toc168260452)

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

SSO – Single sign-on

OIDC – Open ID Connect

IdP – Identity Provider

RP – Relying Party

JWT – JavaScript Web Token

# ВСТУП

**Актуальність теми.** У сучасному світі інтернет-торгівлі питання безпеки та зручності користування стають все більш важливими для користувачів та власників онлайн-магазинів. Одним з ключових аспектів забезпечення безпеки є автентифікація користувачів, яка дозволяє переконатися у достовірності особи, що здійснює вхід на сайт. Одним з найсучасніших та найбільш ефективних способів реалізації автентифікації є використання технології Single sign-on, який забезпечує безпечний та стандартизований спосіб аутентифікації через різні провайдери ідентифікації.

**Метою кваліфікаційної роботи** є прозробка проєкт підсистеми автентифікації онлайн-магазину на базі протоколу Open ID Connect та різних провайдерів ідентифікації.

Визначена мета обумовлює наступні завдання:

* + Провести аналіз існуючих методів автентифікації та їх переваг і недоліків
  + Дослідити протокол OpenID Connect та його можливості для автентифікації
  + Розробити архітектуру підсистеми автентифікації для онлайн-магазину з урахуванням обраних провайдерів ідентифікації.
  + Обрати провайдерів ідентифікації, які підтримують OpenID Connect, та провести їх порівняльний аналіз.

**Об’єктом дослідження** є Об'єктом дослідження є системи автентифікації для онлайн-магазинів. Предметом дослідження виступає підсистема автентифікації на базі протоколу OpenID Connect з використанням різних провайдерів ідентифікації.

**Предметом дослідження** є протокол OIDC, його інтеграція з .NET та сервісами авторизації Okta та Auth0.

У процесі виконання кваліфікаційної роботи було розроблено проект інтеграції інтернет магазину з різними сервісами авторизації по протоколу OIDC.

# **РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ АВТЕНТИФІКАЦІЇ В ОНЛАЙН-МАГАЗИНАХ**

## **1.1 Недоліки традиційної системи автентифікації та користь Single Sign-on**

Традиційна система автентифікації має кілька недоліків порівняно з технологією Single Sign-On (SSO). Нижче розглянемо основні з них:

**1.1.1. Зручність використання**

Зручність використання є одним із ключових аспектів, що відрізняє технологію SSO від традиційної системи автентифікації. Для користувача це означає суттєве полегшення в процесі взаємодії з різними системами та додатками.

У традиційній системі автентифікації кожен додаток або система має свої власні облікові дані. Користувачу необхідно щоразу вводити ім'я користувача та пароль для кожного ресурсу, до якого він бажає отримати доступ. Це створює ряд незручностей.

Наприклад, користувачеві доводиться запам'ятовувати численні комбінації імен користувачів і паролів, що може призвести до використання однакових паролів для різних систем або до збереження паролів у ненадійних місцях. Наприклад, користувач може мати окремі облікові записи для корпоративної електронної пошти, внутрішніх порталів, систем управління проектами, різних SaaS-додатків тощо.

До тогож, постійна необхідність вводити паролі призводить до втоми від паролів (password fatigue), що може спричинити використання простих або повторних паролів, що знижує загальну безпеку.

SSO кардинально змінює цей процес. Завдяки використанню єдиного облікового запису для доступу до всіх необхідних систем та додатків, користувачі мають змогу значно спростити собі життя.

Користувач входить в систему лише один раз, використовуючи єдиний набір облікових даних. Після цього він автоматично отримує доступ до всіх інтегрованих додатків без необхідності повторного введення логіну та паролю.

SSO дає можливість імплементувати безшовну інтеграцію, Після автентифікації користувач може переміщуватися між різними системами і додатками без переривання робочого процесу. Наприклад, після входу в корпоративний портал користувач може миттєво перейти до системи управління проектами, CRM-системи або електронної пошти, не вводячи знову свої облікові дані.

Це також вирішує проблему password fatigue, адже зманшується кількість запитів на відновлення паролів, оскільки користувачам легше запам'ятати один надійний пароль замість десятків різних.

**1.1.2. Адміністрування та підтримка**

Адміністрування та підтримка користувачів є важливими аспектами управління інформаційними системами, і вони суттєво відрізняються у традиційній системі автентифікації та у системах.

У традиційній системі автентифікації кожна система або додаток має свої власні облікові дані та механізми автентифікації. Це означає, що адміністратори повинні окремо керувати обліковими записами для кожної системи, що викликає ряд проблем.

Найголовніше проблема – проблема децентралізованих даних. Адміністраторам доводиться створювати, змінювати та видаляти облікові записи для кожної системи окремо. Наприклад, при прийнятті нового співробітника на роботу адміністратору потрібно вручну створювати облікові записи в системі електронної пошти, в CRM, в системі управління проектами тощо. Це трудомісткий процес, який збільшує навантаження на ІТ-відділ. До того ж підтримка актуальності облікових даних користувачів у різних системах може бути складною задачею. Наприклад, якщо користувач змінює своє ім’я або посаду, ці зміни потрібно вручну оновити у всіх системах. Відсутність автоматичної синхронізації може призвести до розбіжностей в даних та потенційних проблем з доступом.

Оскільки процеси створення та керування обліковими записами виконуються вручну, зростає ймовірність недостовірності даних. Наприклад, адміністратор може забути створити обліковий запис в одній з систем або неправильно налаштувати права доступу, що призведе до затримок у роботі користувача або до проблем з безпекою, займе багато часу і буте джерелом потенційних помилок, особливо в великих організаціях з численними користувачами і системами.

Використання технології SSO значно спрощує адміністрування та підтримку користувачів, забезпечуючи централізоване управління обліковими записами та доступами. Ось як це виглядає:

SSO вирішує проблема децентралізованих даних, дозволяючи адміністраторам керувати обліковими записами користувачів з однієї централізованої системи. Наприклад, при прийнятті нового співробітника на роботу адміністратор створює один обліковий запис на сервері адентичності, який автоматично надає доступ до всіх необхідних систем та додатків. Це значно скорочує час та зусилля, необхідні для налаштування доступів.

До того ж зміни цього облікового записа автоматично синхронізуються у всіх системах, інтегрованих з SSO. При зміні ролей або обов’язків співробітника адміністратор вносить зміни лише в одному місці, і ці зміни автоматично застосовуються до всіх відповідних систем. Це зменшує ймовірність помилок та підвищує ефективність управління доступами. Крім того, при звільненні співробітника адміністратор може миттєво відкликати всі доступи, просто деактивувавши один обліковий запис.

Завдяки спрощенню процесу автентифікації та централізованому керуванню обліковими записами, значно зменшується кількість запитів на відновлення паролів та інших запитів до ІТ-відділу. Користувачам не потрібно запам'ятовувати кілька паролів, що зменшує кількість інцидентів, пов’язаних з забутими паролями.

Також комерційні рішення для реалізації Single Sign-On (SSO) часто відзначаються детально розробленими системами аудиту та моніторингу. Вони забезпечують ефективний аналіз подій, виявлення незвичайної активності та оперативну реакцію на потенційні загрози.Протоколи для реалізації Single sign-on. SAML vs OIDC

## **Протоколи для реалізації SSO. SAML vs OIDC**

Порівняння протоколів SAML (Security Assertion Markup Language) та OIDC (OpenID Connect) є важливим аспектом при виборі систем автентифікації та авторизації для сучасних веб-додатків. Обидва протоколи мають свої унікальні особливості, переваги та недоліки, що робить їх придатними для різних сценаріїв використання.

### Основи та призначення

SAML - це XML-базований стандарт для обміну даними автентифікації та авторизації між різними доменами безпеки. Він був розроблений для великих організацій та підприємств, де потрібна взаємодія між різними системами і службами, що використовують єдиний набір облікових даних користувачів. SAML використовується головним чином у випадках, коли потрібна міждоменна автентифікація, наприклад, між постачальниками послуг (Service Providers, SP) та постачальниками ідентичностей (Identity Providers, IdP).

OIDC - це протокол автентифікації, побудований поверх OAuth 2.0, який використовує JSON замість XML. Він був розроблений для полегшення інтеграції автентифікації з сучасними веб-додатками, мобільними додатками та API. OIDC підтримує більш легку та гнучку модель автентифікації, яка краще підходить для динамічних та розподілених систем.

### Архітектура та компоненти

Протокол SAML (Security Assertion Markup Language) є складним та багатокомпонентним стандартом для обміну даними автентифікації та авторизації між різними доменами безпеки. Для того, щоб зрозуміти роботу цього протоколу, необхідно детально розглянути три основні компоненти: Assertions, Protocols та Bindings.

**### Assertions (Твердження)**

Assertions (твердження) у SAML є основним носієм інформації, яку обмінюють між постачальниками ідентичностей (IdP) та постачальниками послуг (SP). Твердження містять дані про автентифікацію користувача, його атрибути та рішення щодо авторизації. Твердження у SAML зазвичай структуровані у вигляді XML-документів.

У протоколі SAML (Security Assertion Markup Language) твердження (Assertions) відіграють ключову роль у передачі інформації про автентифікацію, атрибути користувача та рішення щодо авторизації. Твердження забезпечують стандартизований спосіб обміну даними між постачальниками ідентичностей (IdP) та постачальниками послуг (SP). Розглянемо детально три основні типи тверджень у SAML: Authentication Assertions, Attribute Assertions та Authorization Decision Assertions.

### Authentication Assertions

Authentication Assertions (Твердження автентифікації) містять інформацію про автентифікацію суб’єкта (користувача) у певний момент часу за допомогою певного методу автентифікації. Це твердження використовується для підтвердження того, що користувач був успішно автентифікований системою постачальника ідентичностей (IdP).

Структура Authentication Assertion включає такі елементи, як момент часу автентифікації (AuthnInstant), який вказує точний час проведення автентифікації, та метод автентифікації (AuthnContext), який описує спосіб автентифікації, наприклад, використання паролю або сертифіката. Це твердження також може містити інформацію про місцезнаходження суб'єкта під час автентифікації, наприклад, IP-адресу (SubjectLocality).

### Attribute Assertions

Attribute Assertions (Атрибутивні твердження) пов’язані з атрибутами суб’єкта, такими як ім'я, електронна адреса, роль у системі або будь-яка інша інформація, яка може бути використана для ідентифікації та авторизації користувача. Атрибутивні твердження дозволяють передавати додаткові дані про користувача, які можуть бути використані для надання персоналізованих послуг або для прийняття рішень щодо доступу до ресурсів.

Ці твердження включають елементи, що описують конкретні атрибути (Attribute), кожен з яких має назву, тип та значення. Атрибути можуть бути закодовані у різних форматах, залежно від специфічних вимог та використовуваних схем даних. Атрибутивні твердження забезпечують гнучкість у передачі детальної інформації про користувача, що є корисним у складних системах з багатьма різнорідними ресурсами.

### Authorization Decision Assertions

Authorization Decision Assertions (Твердження про рішення авторизації) містять інформацію про рішення щодо надання або відмови в доступі до певного ресурсу для суб'єкта. Це твердження використовується для передачі результатів перевірки прав доступу до ресурсів, на підставі політик безпеки, визначених системою постачальника послуг (SP).

Твердження про рішення авторизації включають елементи, що описують дію (Action), яку користувач хоче виконати (наприклад, читання, запис, видалення), ресурс (Resource), до якого користувач намагається отримати доступ, та рішення (Decision), яке може бути дозволом (Permit), забороною (Deny) або невизначеністю (Indeterminate). Це твердження також може містити додаткову інформацію (Evidence), що підтверджує підстави для прийняття рішення.

Узагальнюючи, кожен з трьох типів тверджень у SAML відіграє важливу роль у забезпеченні безпечного та ефективного обміну інформацією між різними системами. Authentication Assertions підтверджують факт автентифікації користувача, Attribute Assertions передають додаткову інформацію про користувача, а Authorization Decision Assertions визначають права доступу користувача до ресурсів. Разом ці твердження забезпечують повний спектр функцій, необхідних для комплексного управління ідентичностями та доступом у сучасних інформаційних системах.

**### Protocols (Протоколи)**

Protocols (протоколи) у SAML визначають правила обміну твердженнями між IdP та SP. Вони описують формати запитів і відповідей, які використовуються для отримання тверджень, а також інші дії, такі як запити атрибутів або авторизації. Протоколи SAML також реалізовані у вигляді XML-документів.

Протокол SAML (Security Assertion Markup Language) має кілька важливих аспектів, серед яких виділяються протоколи Single Logout, Authentication Request та Assertion Query and Request. Кожен з цих протоколів виконує специфічні функції в межах SAML та забезпечує різні аспекти безпеки та управління сесіями.

### Single Logout Protocol

Протокол Single Logout (SLO) у SAML дозволяє одночасно завершувати всі сесії, надані певним центром сесій, майже одночасно. Цей протокол використовується, коли користувач виходить із системи на одній з учасниць сесії або безпосередньо у центрі сесій. Протокол також може бути використаний для завершення сесії у випадку тайм-ауту. Причина виходу може бути вказана через атрибут Reason.

Коли користувач ініціює процес виходу на одній з учасниць сесії, ця учасниця повинна надіслати повідомлення `<LogoutRequest>` до центру сесій, який надав твердження, що містить автентифікаційний запис, пов'язаний з цією сесією. Якщо користувач ініціює вихід у центрі сесій, або ж учасниця сесії надсилає запит на вихід до центру сесій з вказаним користувачем, центр сесій повинен надіслати повідомлення `<LogoutRequest>` до кожної учасниці сесії, якій він надав твердження з автентифікаційними записами в межах поточної сесії з користувачем. Центр сесій повинен спробувати зв'язатися з якомога більшою кількістю цих учасниць, завершити власну сесію з користувачем та, нарешті, повернути повідомлення `<LogoutResponse>` до запитуючої учасниці сесії, якщо така є.

### Authentication Request Protocol

Протокол запиту автентифікації (Authentication Request Protocol) дозволяє учасниці сесії надіслати запит на автентифікацію користувача до центру автентифікації. У відповідь центр автентифікації повертає відповідне твердження, що містить результат автентифікації. Цей процес включає кілька ключових етапів:

1. Учасниця сесії надсилає запит на автентифікацію користувача.

2. Центр автентифікації обробляє запит і проводить автентифікацію користувача.

3. У разі успішної автентифікації центр автентифікації генерує твердження, яке містить інформацію про автентифікацію, і надсилає його назад до учасниці сесії.

Цей протокол забезпечує надійну та захищену передачу автентифікаційної інформації між учасницями сесії та центром автентифікації.

### Assertion Query and Request Protocol

Протокол запиту та отримання тверджень (Assertion Query and Request Protocol) дозволяє учасниці сесії надіслати запит на отримання певних тверджень від центру автентифікації. Цей протокол використовується для отримання інформації, що відповідає певним критеріям, визначеним у запиті.

Процес складається з таких етапів:

1. Учасниця сесії формує запит на отримання тверджень, що відповідають певним критеріям.

2. Центр автентифікації отримує та обробляє запит.

3. Центр автентифікації формує відповідь, що містить запитувані твердження, і надсилає її назад до учасниці сесії.

Цей протокол дозволяє гнучко та ефективно отримувати необхідну автентифікаційну інформацію, що сприяє підвищенню загальної безпеки та управління сесіями в межах SAML.

Отже, протоколи SAML, такі як Single Logout, Authentication Request та Assertion Query and Request, забезпечують важливі функції для безпечного управління автентифікацією та сесіями. Кожен з них виконує свою специфічну роль, сприяючи загальній ефективності та надійності системи автентифікації на основі SAML.

**### Bindings (Зв’язування)**

Протокол SAML (Security Assertion Markup Language) використовує механізми зв’язування (Bindings), щоб інтегрувати свої протоколи з різними транспортними механізмами, такими як HTTP та SOAP. Ці зв’язування забезпечують стандартизований спосіб передачі SAML-повідомлень між постачальниками ідентичностей (IdP) та постачальниками послуг (SP). Основними видами зв’язувань у SAML є HTTP Redirect Binding, HTTP POST Binding та SOAP Binding.

### HTTP Redirect Binding

HTTP Redirect Binding використовується для передачі SAML-повідомлень через URL-параметри у HTTP-запитах методом GET. Цей метод зв’язування часто застосовується для ініціації автентифікації з браузера користувача. У цьому випадку запит на автентифікацію формується IdP та перенаправляється на SP, де він обробляється. Основною перевагою HTTP Redirect Binding є те, що він дозволяє передавати запити без потреби в криптографічному захисті самої передачі, оскільки запити та відповіді передаються через захищене з’єднання HTTPS.

При використанні цього методу, SAML-запит кодується у формі Base64 та передається у вигляді параметра URL. Потім браузер користувача перенаправляється на IdP, де користувач проходить автентифікацію. Після успішної автентифікації IdP формує відповідь SAML, кодує її та передає назад SP через браузер користувача.

### HTTP POST Binding

HTTP POST Binding використовується для передачі SAML-повідомлень через тіла HTTP-запитів методом POST. Цей метод забезпечує можливість передачі великих обсягів даних, таких як SAML Assertions, які перевищують обмеження на довжину URL. HTTP POST Binding часто використовується для передачі SAML-повідомлень з серверу на сервер або для передачі даних, які потребують більшого рівня захисту та цілісності.

При використанні HTTP POST Binding, SAML-повідомлення кодується у формі Base64 та включається у тіло HTTP-запиту. Потім цей запит надсилається на IdP для обробки. Після успішної автентифікації IdP формує відповідь SAML, кодує її та передає назад SP через тіло HTTP-відповіді.

### SOAP Binding

SOAP Binding використовується для передачі SAML-повідомлень через протокол SOAP, що забезпечує надійну та захищену передачу даних між серверами. SOAP є протоколом обміну структурованими повідомленнями у розподілених системах, і він підтримує різні транспортні протоколи, включаючи HTTP та SMTP. SOAP Binding забезпечує високий рівень захисту та гнучкості, що робить його придатним для складних корпоративних середовищ.

При використанні SOAP Binding, SAML-повідомлення інкапсулюється у SOAP-повідомлення, яке передається через транспортний протокол, наприклад, HTTP. Це повідомлення може бути підписано та зашифровано, що забезпечує автентифікацію та цілісність даних. SOAP Binding часто використовується у випадках, коли необхідна взаємодія між різними системами у рамках корпоративної мережі або між організаціями.

Протокол OpenID Connect (OIDC) розширює OAuth 2.0, додаючи шар автентифікації, який дозволяє клієнтам перевіряти особу кінцевого користувача, використовуючи дані, отримані від сервера авторизації. Для досягнення цієї мети OIDC використовує кілька ключових компонентів, таких як ID Token, UserInfo Endpoint, Authorization Endpoint та Token Endpoint.

### ID Token

ID Token є одним із ключових компонентів протоколу OIDC. Це JSON Web Token (JWT), який містить твердження (claims) про автентифікацію кінцевого користувача сервером авторизації. Цей токен надається клієнту під час процесу автентифікації та містить інформацію, таку як ідентифікатор випускача токену (issuer), ідентифікатор користувача (subject), аудиторію токену (audience), час випуску та закінчення терміну дії токену. ID Token також може містити інші твердження, які запитує клієнт, що дозволяє отримувати додаткову інформацію про користувача, таку як його електронну пошту або ім'я.

### UserInfo Endpoint

UserInfo Endpoint є захищеним ресурсом, який, отримуючи токен доступу від клієнта, повертає авторизовану інформацію про кінцевого користувача. Клієнт надсилає запит до UserInfo Endpoint, додаючи токен доступу у заголовок запиту. У відповідь сервер надає JSON-об'єкт, що містить затверджені твердження про користувача, наприклад, його ім'я, електронну адресу та інші профільні дані. Це дозволяє клієнту отримувати оновлену інформацію про користувача після початкової автентифікації.

### Authorization Endpoint

Authorization Endpoint використовується для ініціації процесу автентифікації та отримання коду авторизації. Клієнт надсилає запит на авторизацію до цього кінцевого пункту, вказуючи необхідні параметри, такі як відповідь типу (response type), ідентифікатор клієнта (client ID), перенаправлення URI (redirect URI) та scope, який включає 'openid'. Після автентифікації користувача сервер авторизації надає клієнту код авторизації, який може бути обміняний на токени доступу та ID Token на Token Endpoint.

### Token Endpoint

Token Endpoint використовується для обміну коду авторизації на токени доступу та ID Token. Клієнт надсилає запит до цього кінцевого пункту, включаючи код авторизації, ідентифікатор клієнта, секрет клієнта (client secret) та перенаправлення URI. У разі успішної перевірки сервер авторизації повертає токени доступу та ID Token, які клієнт може використовувати для доступу до захищених ресурсів та для отримання інформації про користувача з UserInfo Endpoint.

### Механізми автентифікації та авторизації

У SAML автентифікація відбувається шляхом передачі SAML Assertions від IdP до SP через браузер користувача. Ці твердження містять інформацію про успішну автентифікацію користувача та інші атрибути, необхідні для авторизації.

OIDC, з іншого боку, використовує токени доступу та оновлення, що надаються через OAuth 2.0. ID Token містить інформацію про автентифікацію користувача та може бути перевірений безпосередньо SP без потреби у додаткових запитах до IdP. Це забезпечує більш швидку та ефективну автентифікацію.

### Безпека та шифрування

SAML використовує XML Signature та XML Encryption для забезпечення цілісності та конфіденційності даних. Це дозволяє захистити SAML Assertions від підробки та несанкціонованого доступу під час передачі між IdP та SP.

OIDC використовує JSON Web Signature (JWS) та JSON Web Encryption (JWE) для забезпечення безпеки токенів. JWT токени можуть бути підписані та зашифровані, що забезпечує високий рівень безпеки при їх передачі та зберіганні.

### Сумісність та інтеграція

SAML є більш складним у налаштуванні та інтеграції, оскільки вимагає глибокого розуміння XML та пов'язаних з ним технологій. Він також часто використовується у великих корпоративних середовищах, де необхідна висока ступінь безпеки та контроль доступу.

OIDC є більш гнучким та простим у використанні завдяки своєму JSON формату та тісній інтеграції з OAuth 2.0. Це робить його ідеальним для використання у сучасних веб-додатках та мобільних додатках, де важлива швидкість та простота інтеграції.

### Підсумок

Узагальнюючи, SAML та OIDC мають свої унікальні переваги та недоліки. SAML підходить для великих підприємств та організацій з високими вимогами до безпеки та міждоменної автентифікації. OIDC краще підходить для сучасних веб-додатків, мобільних додатків та API, де важливі швидкість та гнучкість інтеграції.

Остаточний вибір між SAML та OIDC залежить від конкретних потреб та вимог вашої організації, а також від того, наскільки важлива простота інтеграції та підтримка сучасних технологій автентифікації.

## **1.3** **Огляд найпопулярніших рішень на роль IdP**

### Порівняння Auth0, Okta та Azure AD

Auth0, Okta та Azure Active Directory (Azure AD) є провідними рішеннями для управління ідентичністю та доступом (IAM), кожне з яких пропонує унікальні функції та переваги для різних бізнес-потреб.

Auth0

Auth0 - це популярна платформа для управління ідентифікацією та доступом, яка пропонує широкий спектр функцій для аутентифікації та авторизації користувачів.

### Переваги Auth0

1. \*\*Гнучкість та налаштування:\*\* Auth0 підтримує різноманітні методи аутентифікації, включаючи соціальні логіни, єдиний вхід (SSO), багатофакторну аутентифікацію (MFA) та адаптивну аутентифікацію. Це дозволяє налаштувати систему під конкретні потреби вашого бізнесу.

2. \*\*Простота інтеграції:\*\* Платформа пропонує безліч SDK та API для легкої інтеграції з існуючими додатками та сервісами. Це спрощує процес впровадження Auth0 у ваші проекти.

3. \*\*Безпека:\*\* Auth0 забезпечує високий рівень безпеки завдяки підтримці MFA, адаптивній аутентифікації та захисту від атак brute force. Платформа також має вбудовані механізми для виявлення підозрілої активності та реагування на загрози.

4. \*\*Документація та підтримка:\*\* Auth0 пропонує детальну документацію, приклади коду та активну спільноту користувачів, що полегшує розробку та налаштування. Крім того, платформа забезпечує технічну підтримку для вирішення будь-яких проблем.

5. \*\*Масштабованість:\*\* Auth0 легко масштабувати відповідно до зростання бізнесу. Платформа може обробляти велику кількість запитів на аутентифікацію, що робить її ідеальною для великих компаній.

#### Okta

\*\*Особливості:\*\*

Okta пропонує потужні можливості для управління ідентичністю, включаючи SSO, MFA, управління доступом до API, а також моніторинг активності користувачів. Платформа підтримує широкий спектр інтеграцій і забезпечує високу безпеку завдяки багатошаровим методам аутентифікації та управління доступом.

\*\*Переваги:\*\*

Okta відзначається зручністю використання та простотою впровадження. Його інтеграції з різними додатками та сервісами забезпечують ефективне управління доступом і поліпшення безпеки. Користувачі також високо цінують інтерфейс та підтримку платформи.

\*\*Недоліки:\*\*

Деякі користувачі зазначають, що Okta може мати складності в налаштуванні для великих організацій. Існують також зауваження щодо структури ціноутворення та потреби в додаткових можливостях для налаштування.

#### Azure AD

\*\*Особливості:\*\*

Azure AD інтегрується з іншими продуктами Microsoft, такими як Office 365, що забезпечує безшовне управління користувачами та підвищення продуктивності. Платформа надає можливості SSO, MFA, управління доступом на основі ролей та умови доступу, що робить її потужним інструментом для забезпечення безпеки.

\*\*Переваги:\*\*

Azure AD високо цінується за інтеграцію з екосистемою Microsoft, що спрощує управління ідентичністю та доступом. Серед сильних сторін платформи виділяються її безпека, можливості автоматизації процесів і централізоване управління користувачами.

\*\*Недоліки:\*\*

Azure AD може бути складним у налаштуванні та синхронізації з іншими системами, особливо при використанні гібридного середовища з локальним Active Directory. Також деякі користувачі відзначають потребу в поліпшенні документації та інтерфейсу.

### Висновок

Вибір між Auth0, Okta та Azure AD залежить від конкретних потреб вашої організації. Auth0 відмінно підходить для компаній, що шукають гнучкість та налаштування. Okta є потужним інструментом для управління доступом з широкими можливостями інтеграції та безпеки. Azure AD найкраще підходить для організацій, що вже використовують екосистему Microsoft і потребують глибокої інтеграції з продуктами Microsoft.

Для детальнішого порівняння, ви можете ознайомитися з додатковою інформацією на ресурсах:

- [SaaSworthy](https://www.saasworthy.com/compare/auth0-vs-okta-vs-azure-ad-vs-aws-iam?pIds=2935,2940,5977,5991)

- [JumpCloud](https://jumpcloud.com)

# РОЗДІЛ 2 ПРОЕКТУВАННЯ РІШЕННЯ

## **2.1. Формування вимог до проектувідповідно до бізнес правил інтернет-магазину**

Бізнес логіка інтернет-магазину побудована на принципі SaaS, або програмне забезпечення як послуга (Software as a Service), - це модель доставки програмного забезпечення, при якій програми знаходяться на серверах хост-провайдера та доступні через Інтернет. Користувачі можуть отримати доступ до цих програм через веб-браузер або інтерфейс за допомогою мережі Інтернет, не завантажуючи або встановлюючи їх на свої пристрої.

Використання моделі **SaaS** для інтернет-магазину має декілька переваг. Вона забезпечує швидку та легку реалізацію, оскільки не вимагає встановлення або налаштування складного програмного забезпечення на стороні клієнта.

Ми розглядаємо моделі **B2C** (Business-to-Consumer) та **B2B** (Business-to-Business), що є ключовими концепціями в сфері SaaS та електронної комерції. B2C визначає взаємодію між компанією та кінцевими споживачами, тоді як B2B описує взаємодію між компаніями, де одна компанія надає товари або послуги іншій компанії.

Наше рішення розраховане одразу на обидва ці сценарії. У моделі B2C, доступ до магазину надається для безпосереднього використання будь-якими кінцевими споживачами. Наприклад, популярні онлайн-магазини, такі як Amazon або eBay, діють у цьому сегменті. Такі платформи надають широкий вибір товарів та послуг, які можна придбати онлайн, та забезпечують різноманітні функції для зручного пошуку, вибору та оплати товарів.

У моделі B2B, електронна комерція використовуються для покращення бізнес-процесів та взаємодії між компаніями. В цьому плані, наша платформа позіціонується як система де один бізнес, може замовити продукт у іншого на різних умоах та підставах. Така платформа надає можливості для оптимізації бізнес-процесів, автоматизації операцій та полегшення спільної роботи між компаніями.

Імплементуючи SSO в наш інтернет магазин, потрібно враховувати що замовник у магазину буде не один, а декілька. Відповідно критично важлива бізнес вимога - зробити динамічну інтеграцію IdP з інтернет магазином.

## **2.2 Проектування рішення**

### 2.2.1 Огляд технологічного стеку інтернет магазину

Веб-магазин був розроблений на основі фреймворку ASP.NET MVC, адже він надає потужні інструменти для швидкого та ефективного створення складних веб-додатків. Цей фреймворк вбудовує в себе широкий набір функцій, які спрощують розробку, такі як маршрутизація, контролери, моделі та представлення.

Крім того, ASP.NET MVC інтегрований з технологіями та інструментами, такими як Entity Framework для роботи з базами даних та з бібліотекою ASP.NET Identity для внутрішньої автентифікації та авторизації користувачі.

В якості бази даних інтернет магазин використовує Microsoft SQL Server (MSSQL) адже вона ідеально інтегрується з ASP.NET. Ця база даних зберігає інформацію про продукти, їх характеристики, ціни, а також дані про системних користувачів, що мають доступ до адміністративних функцій магазину, налаштування системи, інформацію про замовлення та інше.

### 2.2.2 Проектування рішення під існуючу кодову базу

Як вже було зазначено, інтернет магазин використовує систему інтифікації яка потавляється з ASP.NET.

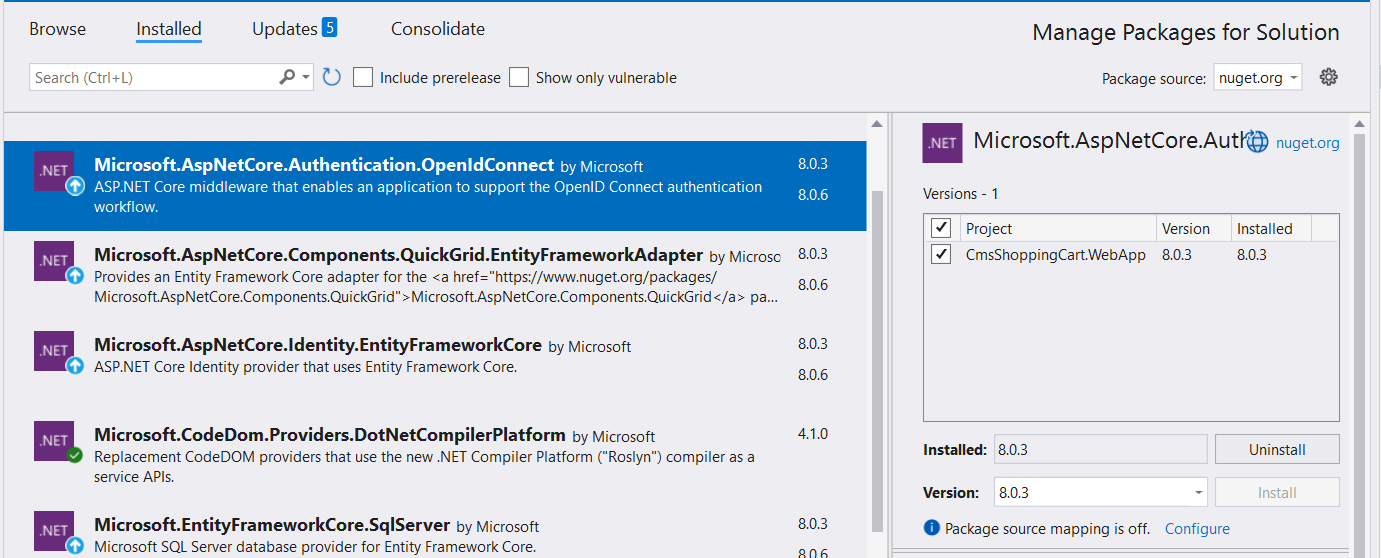
Відповідно, будемо використовувати офіційну бібліотеку для OIDC від Microsoft, яка постачається разом зі стандартною системою автентифікації, та гарно інтегрується з існуючою системою автентифікації інтернет магазину. Ця бібліотека відома як Microsoft.AspNetCore.Authentication.OpenIdConnect.

Одним з традиційних підходів збереження конфігурацій в ASP.NET є конфігураційний файл appsettings.json. Зазвичай адміністратор перед тим як підняти додаток змінює цей конфігураційний файл, тому має сенс зберігати конфігурацію провайдерів інтеифікації саме там.

## **2.4 Імплементація рішення**

### 2.4.1. Динамічна реєстрація IdP

1. Спочатку, додамо бібліотеку до проекту, за допомогою NuGet Package Manager:



*Мал. 2.4.1.1 Втсановлення бібліотеки для протоколу OIDC*

В XML налаштуваннях проекту можемо бачити що посилання на пакунок успішно додалося до проекту:

<Project Sdk="Microsoft.NET.Sdk.Web">

<PropertyGroup>

<TargetFramework>net8.0</TargetFramework>

</PropertyGroup>

<ItemGroup>

<Compile Remove="wwwroot\lib\ckeditor\\*\*" />

<Content Remove="wwwroot\lib\ckeditor\\*\*" />

<EmbeddedResource Remove="wwwroot\lib\ckeditor\\*\*" />

<None Remove="wwwroot\lib\ckeditor\\*\*" />

</ItemGroup>

<ItemGroup>

<PackageReference Include="Auth0.AspNetCore.Authentication" Version="1.4.1" />

<PackageReference Include="Microsoft.AspNetCore.Authentication.OpenIdConnect" Version="8.0.3" />

<PackageReference Include="Microsoft.AspNetCore.Components.QuickGrid.EntityFrameworkAdapter" Version="8.0.3" />

<PackageReference Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore" Version="8.0.3" />

<PackageReference Include="Microsoft.CodeDom.Providers.DotNetCompilerPlatform" Version="4.1.0" />

<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="8.0.3" />

<PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.3">

<PrivateAssets>all</PrivateAssets>

<IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>

</PackageReference>

</ItemGroup>

</Project>

2. Тепер імплементуємо динамічне зчитування та налаштовування налаштовує систем автентифікації з використанням. Нам потрібно зчитати з файлу конфігурації наступну структуру:

1. Autority – посилання на сервер авторизації, що в результаті буде автентифіковувати кінцевого користувача.
2. ClientId – валідний для серверу автентифікації ідентифікатор клієнта, що налаштовується на самому сервері ідетифікації, відповідно до його документації.
3. Secret – секрет, за допомогою якого наш сервер авторизації буде розпізнавати що це саме наш інтернет магазин а не зловмисник.
4. CallbackPath – шлях, на який повернеться користувач від IdP з результами авторизації користувача.
5. LoginButtonImageUrl – посилання на картинку, що буде відображена в кнопці для авторизації.
6. LoginButtonText – текст який буде відображено на кнопці автентифікації.

Ці властивості будуть спільними для всіх інтеграцій, тому структура нашого налаштування буде виглядати наступним чином:

"SSO": {

"OIDCProdider1": {

"Authority": "OIDCProvider1 authority URL",

"ClientId": "OIDCProvider1 application client id",

"ClientSecret": "secret",

"CallbackPath": "/OIDCProvider1/signin-oidc",

"LoginButtonImageUrl": "~/media/OIDCProvider1-logo.png",

"LoginButtonText": "Login With OIDCProvider1"

},

" OIDCProdider2": {

"Authority": "OIDCProvider2 authority URL",

"ClientId": "OIDCProvider2 application client id",

"ClientSecret": "secret",

"CallbackPath": "/ OIDCProvider2/signin-oidc",

"LoginButtonImageUrl": "~/media/OIDCProvider2-logo.png",

"LoginButtonText": "Login With OIDCProvider2"

}

}

Додмо клас який буде використовується для збереження конфігураційних параметрів, необхідних для налаштування OpenID Connect провайдерів:

public class SSOConfiguration

{

public string Authority { get; set; }

public string ClientId { get; set; }

public string ClientSecret { get; set; }

public string CallbackPath { get; set; }

public string LoginButtonImageUrl { get; set; }

public string LoginButtonText { get; set; }

}

Зчитування конфігурації:

var b = new AuthenticationBuilder(services);

var providers = Configuration["SSO"];

var config = JsonSerializer.Deserialize<Dictionary<string, SSOConfiguration>>(providers);

Створюється екземпляр `AuthenticationBuilder`, який використовується для додавання та налаштування автентифікаційних схем у додатку, в нього треба передати services типу IserviceCollection. Він є частиною системи впровадження залежностей (Dependency Injection) в ASP.NET Core. Це інтерфейс, який представляє контейнер для реєстрації служб, необхідних вашому додатку.

З конфігураційного файлу зчитуються параметри для всіх SSO провайдерів і десеріалізуються в словник, де ключ - це ім'я провайдера, а значення - об'єкт `SSOConfiguration`.

Далі напишемо код, що дозволяє налаштувати декілька провайдерів OpenID Connect для автентифікації користувачів. Кожен провайдер буде налаштовуватися з використанням своїх параметрів:

foreach (var provider in config)

{

var providerName = provider.Key;

var providerConfig = provider.Value;

b.AddOpenIdConnect(providerName, o =>

{

o.Authority = providerConfig.Authority;

o.ClientId = providerConfig.ClientId;

o.ClientSecret = providerConfig.ClientSecret;

o.CallbackPath = providerConfig.CallbackPath;

o.Scope.Add("email");

o.UsePkce = true;

o.ResponseType = OpenIdConnectResponseType.Code;

o.SignInScheme = IdentityConstants.ApplicationScheme;

o.ClaimActions.Add(new SimpleClaimAction("AuthenticationType", providerName));

o.TokenValidationParameters.RoleClaimType = "account\_role";

o.Events = new OpenIdConnectEvents()

{

OnRedirectToIdentityProviderForSignOut = context =>

{

var logoutUri = "/";

context.Response.Redirect(logoutUri);

context.HandleResponse();

return Task.CompletedTask;

}

};

});

}

- Зчитується ім'я провайдера (`providerName`) та його конфігурація (`providerConfig`).

- Додається схема автентифікації OpenID Connect для цього провайдера з налаштуванням властивостей:

- Встановлюються параметри `Authority`, `ClientId`, `ClientSecret`, `CallbackPath` які беруться з конфігурації провайдера.

- Додається скоуп `email` для запитів до OpenID провайдера. Це означає, що додаток запитує доступ до електронної пошти користувача. За замовчуванням, OpenIdConnectOptions додаються скоупи openid та profile.

- Використовується PKCE (Proof Key for Code Exchange) для додаткової безпеки.

- Встановлюється тип відповіді як `Code`.

- Визначається схема входу. Саме схема ApplicationScheme використовується для авторизації в інтернет магазині і на ній зав’язана існуюча логіка.

- Визначається account\_role як клейм для ролі в токені.

- Налаштовуються подія для обробки виходу з системи, Це потрібно щоб перед тим як перенаправити користувача до IdP вказати на який шлях в в інтернет магазині, потрібно повернути кристувача, а саме на головну.

Також додається клейм `AuthenticationType` в токен, в якому закодовується назва провайдеру. Це потрібно для того щоб клієнтський код зміг ідентифікувати що поточний користувач автентифікувався за допомогою OIDC. За замовчуванням ClaimActions не має такого механізму хардкоду значення в токен. Тому ми додамо свій власний ClaimAction, маємо:

public class SimpleClaimAction : ClaimAction

{

private string SavedValue { get; set; }

public SimpleClaimAction(string claimType, string valueType) : base(claimType, valueType)

{

SavedValue = valueType;

}

public override void Run(JsonElement userData, ClaimsIdentity identity, string issuer)

{

identity.AddClaim(new Claim(ShopClaimTypes.AuthenticationScheme, SavedValue));

}

}

Робота цього класу досить проста. Конструктор приймає два параметри: `claimType` і `valueType`, які використовуються для визначення типу та значення клейма. Викликається базовий конструктор класу `ClaimAction` з переданими параметрами. Значення `valueType` присвоюється властивості `SavedValue`.

Метод Run є перевизначеним методом з базового класу `ClaimAction` і викликається, коли відбувається обробка клеймів. В методі створюється новий клейм за допомогою значення `SavedValue`, яке було збережено під час ініціалізації об'єкта `SimpleClaimAction`. Клейм додається до інформації про автентифікацію користувача.

### 2.4.2. Автентифікація через IdP

Спочатку визначимо, які властивості будемо передавати представленню. Все що нам потрібно це схема автентифікації, задля того щоб потім вказати за яким провайдером ми будемо автентифікуватися, текст кнопки та логотип провайдера, щоб відрендерити ту кнопку на яку користувач натисне для автентифікації. Для цього настпний клас:

public class IdentiryProviderViewModel

{

public string AuthenticationScheme { get; set; }

public string LoginButtonText { get; set; }

public string LoginButtonImageUrl { get; set; }

}

Тепер створимо метод, який буде, обробляти запит на сторінку входу:

[AllowAnonymous]

public IActionResult Login(string returnUrl)

{

var oidcModels = new List<OIDCIdentiryProviderViewModel>();

var providers = configuration["SSO"];

var configurations = JsonSerializer.Deserialize<Dictionary<string, SSOConfiguration>>(providers);

foreach (var configuration in configurations)

{

var providerName = configuration.Key;

var providerConfig = configuration.Value;

oidcModels.Add(new OIDCIdentiryProviderViewModel()

{

AuthenticationScheme = providerName,

LoginButtonText = providerConfig.LoginButtonText,

LoginButtonImageUrl = providerConfig.LoginButtonImageUrl

}); ;

}

var loginViewModel = new LoginViewModel()

{

ReturnUrl = returnUrl,

IdentityProviders = oidcModels

};

return View(loginViewModel);

}

Метод має атрибут [AllowAnonymous], що дозволяє доступ до цього методу без аутентифікації. Метод приймає параметр returnUrl, який використовується для перенаправлення користувача після успішного входу.

У методі створюється список об'єктів типу OIDCIdentiryProviderViewModel. Далі з конфігурації зчитуються налаштування провайдерів єдиного входу (SSO) у вигляді рядка JSON. Цей JSON десеріалізується для подальшого використання.

У циклі foreach обробляється кожен провайдер зі словника конфігурацій. Для кожного провайдера створюється новий об'єкт OIDCIdentiryProviderViewModel з відповідними властивостями: схема аутентифікації, текст кнопки входу та URL зображення кнопки входу. Цей об'єкт додається до списку oidcModels.

Після обробки всіх провайдерів створюється об'єкт LoginViewModel, куди встановлюються returnUrl та список провайдерів аутентифікації (oidcModels).

В результаті метод повертає подання (View) з цим LoginViewModel, що буде використовуватись для відображення сторінки входу.

Модифікуємо представлення. Після основної форми входу додамо динамічне створення форм для входу через налаштовані IdP, перелічених у властивості IdentityProviders моделі. Для кожного провайдера створюється окрема форма з прихованими полями для ReturnUrl та AuthenticationScheme, а також кнопкою, яка містить зображення та текст для відповідного провайдера. Також не мало важливо вказати що при підтвердженні форми буде викликатися asp-action – LoginWithSSO Виглядати це буде наступним чином:

@model CmsShoppingCart.WebApp.Models.Authentication.LoginViewModel

@{

ViewData["Title"] = "Login";

}

<h1>Login</h1>

<div class="row">

<div class="col-md-10">

<**form** **asp-action**="Login">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<**input** **type**="hidden" **asp-for**="ReturnUrl" />

<div class="form-group">

<**label** **asp-for**="Email" class="control-label"></**label**>

<**input** **asp-for**="Email" class="form-control" />

<**span** **asp-validation-for**="Email" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Password" class="control-label"></**label**>

<**input** **asp-for**="Password" class="form-control" />

<**span** **asp-validation-for**="Password" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Log in" class="btn btn-primary" />

</div>

</**form**>

</div>

@foreach (var provider in Model.IdentityProviders)

{

<div class="col-md-10">

<**form** **asp-controller**="Account" **asp-action**="LoginWithSSO">

<**input** **type**="hidden" **asp-for**="ReturnUrl" />

<**input** **type**="hidden" **asp-for**="AuthenticationScheme" **value**="@provider.AuthenticationScheme" />

<div class="form-group">

<button type="submit" class="btn btn-outline-secondary">

<img src="@provider.LoginButtonImageUrl" width="48" alt="Auth0 logo" />

<span>@provider.LoginButtonText</span>

</button>

</div>

</**form**>

</div>

}

</div>

Імплементація методу LoginWithSSO:

[AllowAnonymous]

[HttpPost]

[ValidateAntiForgeryToken]

public async Task LoginWithSSO(string authenticationScheme, string returnUrl = "/")

{

if (authenticationScheme.IsNullOrEmpty())

throw new ArgumentException("Scheme cannot be null or empty", nameof(authenticationScheme));

var authenticationProperties = new LoginAuthenticationPropertiesBuilder()

.WithRedirectUri(returnUrl)

.Build();

await HttpContext.ChallengeAsync(

authenticationScheme,

authenticationProperties

);

}

Цей метод обролює вже сам запит. Він має три атрибути: [AllowAnonymous] дозволяє доступ до цього методу без аутентифікації, [HttpPost] вказує, що метод обробляє тільки POST-запити, а [ValidateAntiForgeryToken] забезпечує захист від підробки міжсайтових запитів (CSRF).

Метод приймає два параметри: `authenticationScheme`, який вказує на схему аутентифікації, і `returnUrl`, який задає URL для перенаправлення після успішного входу, за замовчуванням встановлений на "/".

Перший крок у методі перевіряє, чи `authenticationScheme` не є порожнім або null, і якщо це так, викидає ArgumentException з відповідним повідомленням.

Далі створюється об'єкт `authenticationProperties` з використанням класу `LoginAuthenticationPropertiesBuilder`. Цей об'єкт налаштовується з редіректом на `returnUrl` після успішної аутентифікації.

Нарешті, метод викликає `HttpContext.ChallengeAsync`, передаючи `authenticationScheme` і `authenticationProperties`. Це ініціює процес аутентифікації через вказану схему, що може включати перенаправлення користувача до IdP.

В результаті, ми маємо апі запит - [POST] https://{domain}/account/loginWithSSO

### 2.4.3. Вихід з IdP

Для виходу з IdP модифікуємо вже існуючий метод Logout.

public async Task<IActionResult> Logout()

{

var authenticationProperties = new LogoutAuthenticationPropertiesBuilder()

.WithRedirectUri(Url.Action("Index", "Home"))

.Build();

var authClaim = User.Claims.FirstOrDefault(c => c.Type == ShopClaimTypes.AuthenticationScheme);

if (authClaim is not null)

await HttpContext.SignOutAsync(authClaim.Value, authenticationProperties);

await HttpContext.SignOutAsync();

return Redirect("/");

}

Для створення параметрів перенаправлення користувача після виходу з IdP, використовується `LogoutAuthenticationPropertiesBuilder`. Метод `WithRedirectUri` вказує, що після виходу користувач буде перенаправлений на сторінку "Index" контролера "Home", тобто на головну сторінку.

Далі іде пошук клейму, який має тип `ShopClaimTypes.AuthenticationScheme`, адже в ньому зберігається інформація про IdP, через який користувача автентифіковано.

Якщо інформація про зовнішній IdP знайдена, викликається метод `SignOutAsync` для відключення користувача з IdP.

Якщо клейм не знайдений,то цей користувач автентифікувався через нашу внутрішню систему автентифікації, тому викликаємо метод SignOutAsync без параметрів, що за замовчуванням завершить внутрішн сесію.

### 2.4.4. Розмежування ролей автентифікованого користувача

Розмежування доступу у ASP.NET MVC забезпечується на рівні контролерів за допомогою атрибутів [Authorize], які можуть бути застосовані як до всього контролера, так і до окремих дій (методів) у контролері. Атрибут [Authorize] дозволяє обмежити доступ до ресурсів лише для автентифікованих користувачів або для користувачів з певними ролями або політиками.

Так буде виглядати захит контроллеру:

[Authorize(Roles = "Admin")]

[Area("Admin")]

public class ShopController : Controller

{ }

Так буде виглядати захит окремого методу контроллеру:

[Area("Shop")]

public class ShopController1 : Controller

{

[Authorize(Roles = "Editor")]

public async Task<IActionResult> Index()

{

return View();

}

}

Вся авторизація ASP.NET MVC побудована на http контексті і великим плюсом нашої імплементації є можливість користуватися стандартними інструментами автентифікації фреймворку.

Для представлень Razor доступ до інформації про ролі буде досягатись наступним чином:

@if (User?.Identity?.IsAuthenticated @@ (User.IsInRole("Admin") | User.IsInRole("Editor"))

{

<li class="nav-item ml-1">

<**a** class="btn btn-success"

**asp-area** = "Admin" **asp-controller**="Pages" **asp-action**="Index">Admin Area</**a**>

</li>

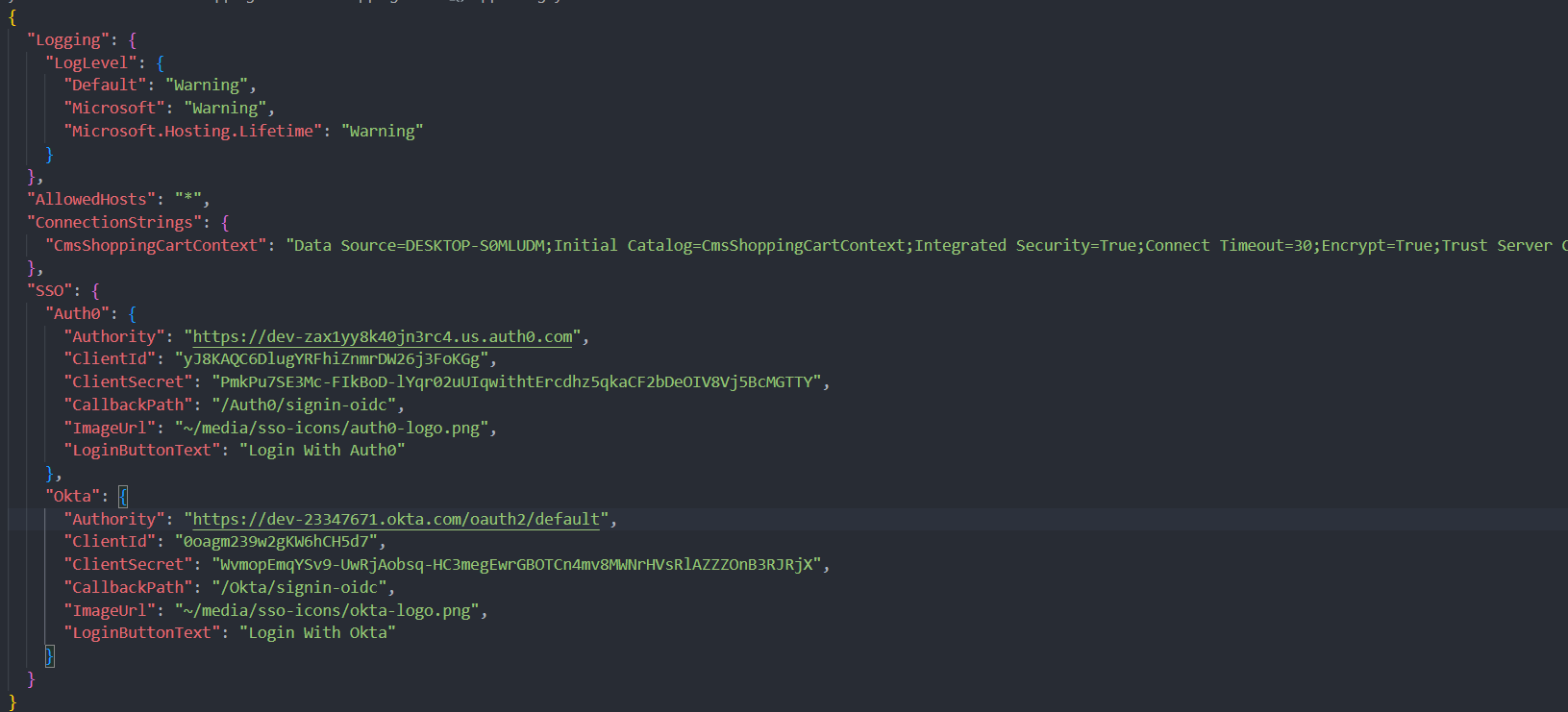
}

# РОЗДІЛ 3 ТЕСТУВАННЯ ІНФРАСТРУКТУРИ

### 3.1 Тестування роботоздатності інтернет магазину

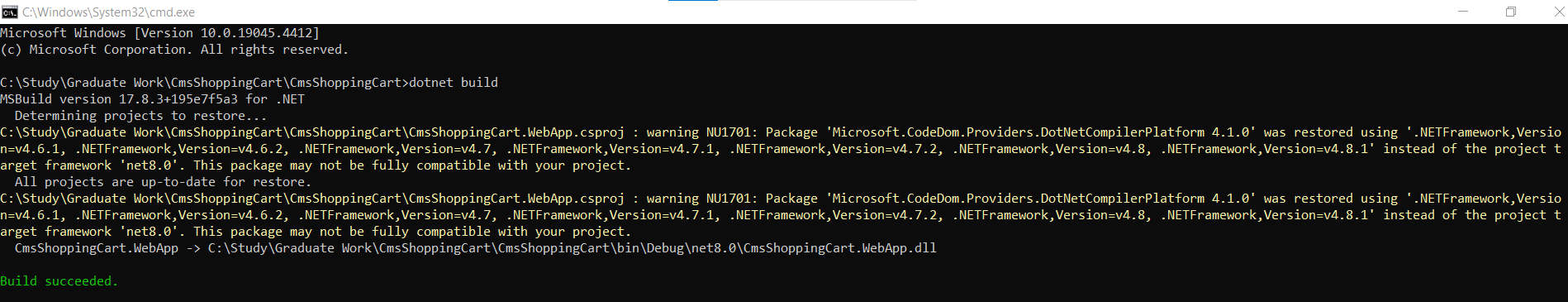
Спочатку нам потрібно сконфігурувати всі IdP в конфіграційному файлі appsettings.json. Для тестування налаштуємо провайдери Okta та Auth0. Детальні налаштування провайдерів приведено в пунктах 3.2 та 3.3 . Відповідно нас цікавить поле “SSO”, в якому потрібно на кожний IdP додати наступні налаштування:

1. Autority – посилання на сервер авторизації, що в результаті буде автентифіковувати кінцевого користувача.
2. ClientId – валідний для серверу автентифікації ідентифікатор клієнта, що налаштовується на самому сервері ідетифікації, відповідно до його документації.
3. Secret – секрет, за допомогою якого наш сервер авторизації буде розпізнавати що це саме наш інтернет магазин а не зловмисник.
4. CallbackPath – шлях, на який повернеться користувач від IdP з результами авторизації користувача.
5. LoginButtonImageUrl – посилання на картинку, що буде відображена в кнопці для авторизації.
6. LoginButtonText – текст який буде відображено на кнопці автентифікації.



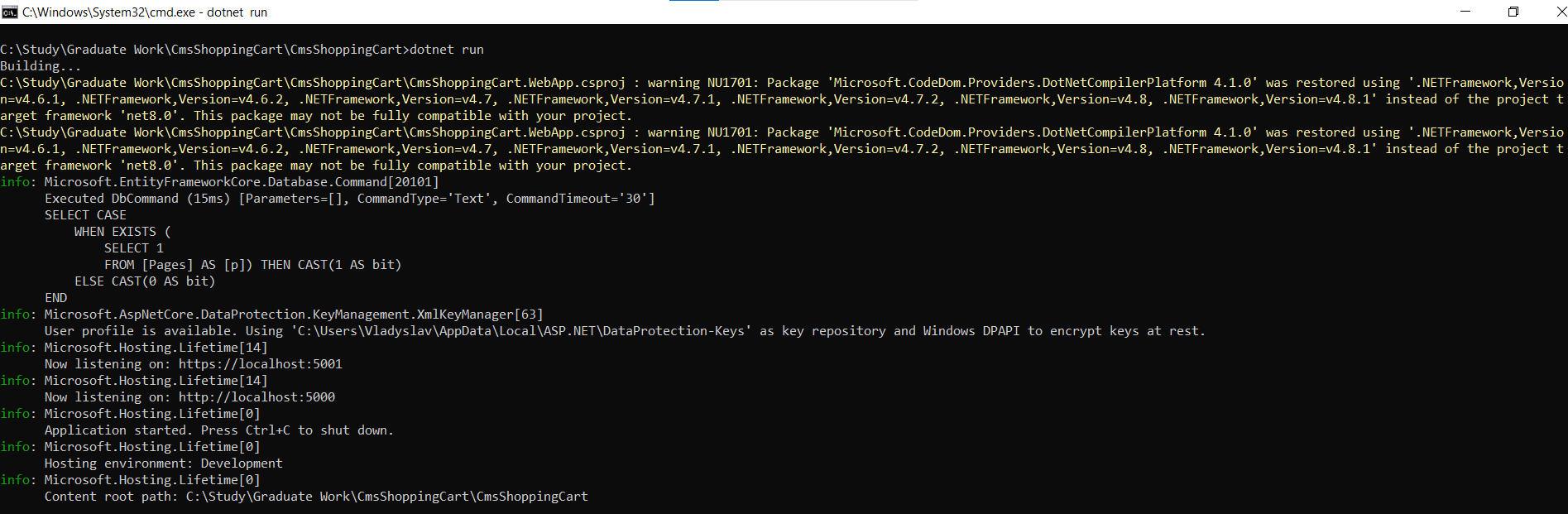
*Мал. 3.1.1 Конфігурація двох IdP в конфіграційному файлі appsettings.json*

Після розробки технології, нам потрібно перевірити чи немає run time помилок при запуску додатку, пому переходимо в теку з проектом, відкриваємо в цій теці консоль, та прописуємо команду «dotnet build»:



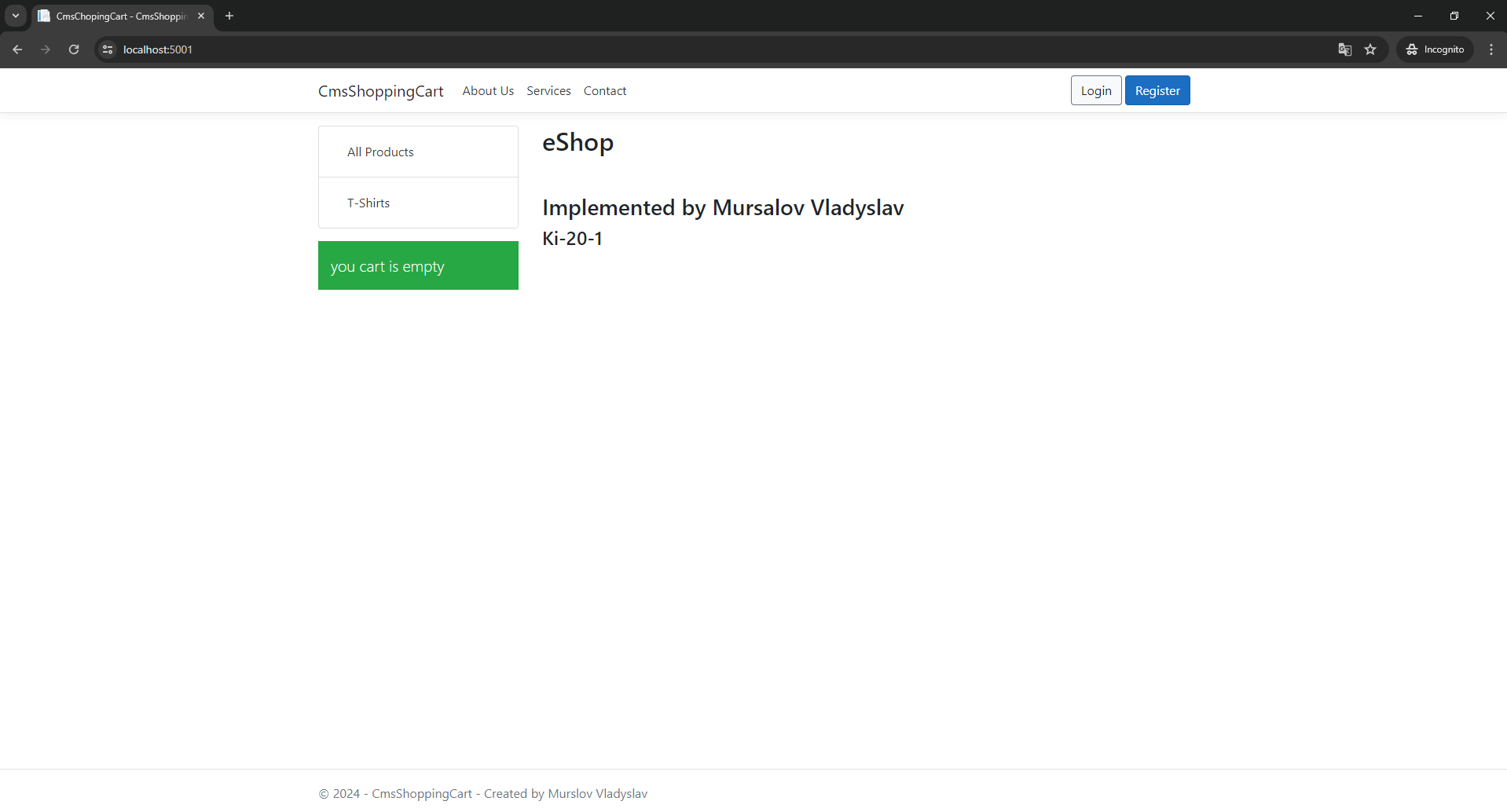
*Мал. 3.1.2 Успішна збірка проекту*

Далі щоб підняти застосунок потрібно прописати команду «dotnet run». Можемо бачити що застосунк успішно запутився та адресу на якій знаходиться наш інтернет-магазин:



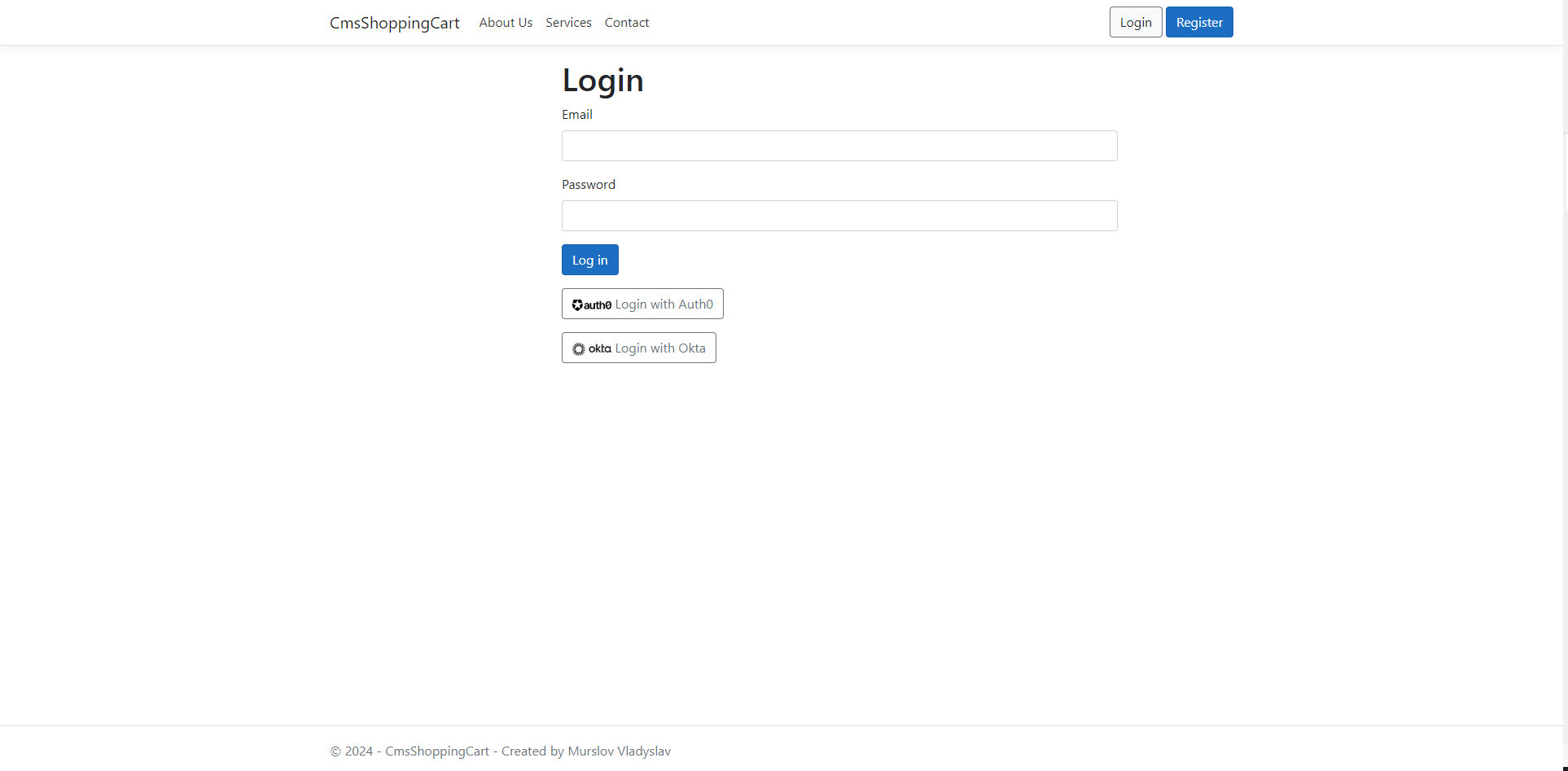
*Мал. 3.1.3 Успішний запуск з консолі*

Тепер переходимо на головну сторінку, щоб запевнитися що все добре.



*Мал. 3.1.3 Головна сторінка магазину*

Тепер перейдемо на сторінку авторизації, щоб запевнитися чи додались посилання на авторизацію на наші 2 провайдери, маємо:



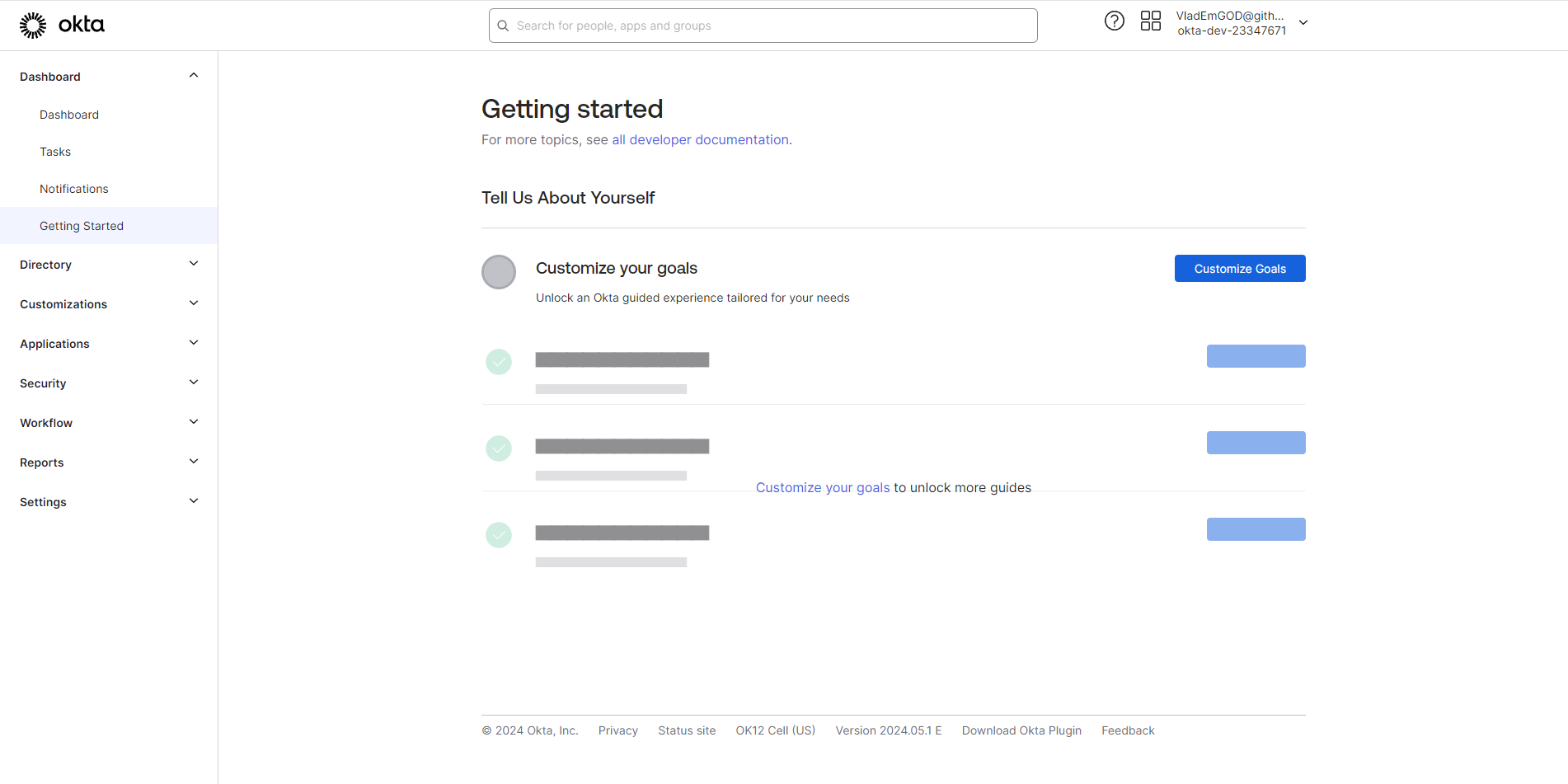
*Мал. 3.1.3 Сторінка авторизації*

Можемо бачити що кнопки успішно додалися і ми налаштували все правильно.

### 3.2. Налаштування Okta як серверу автентифіації

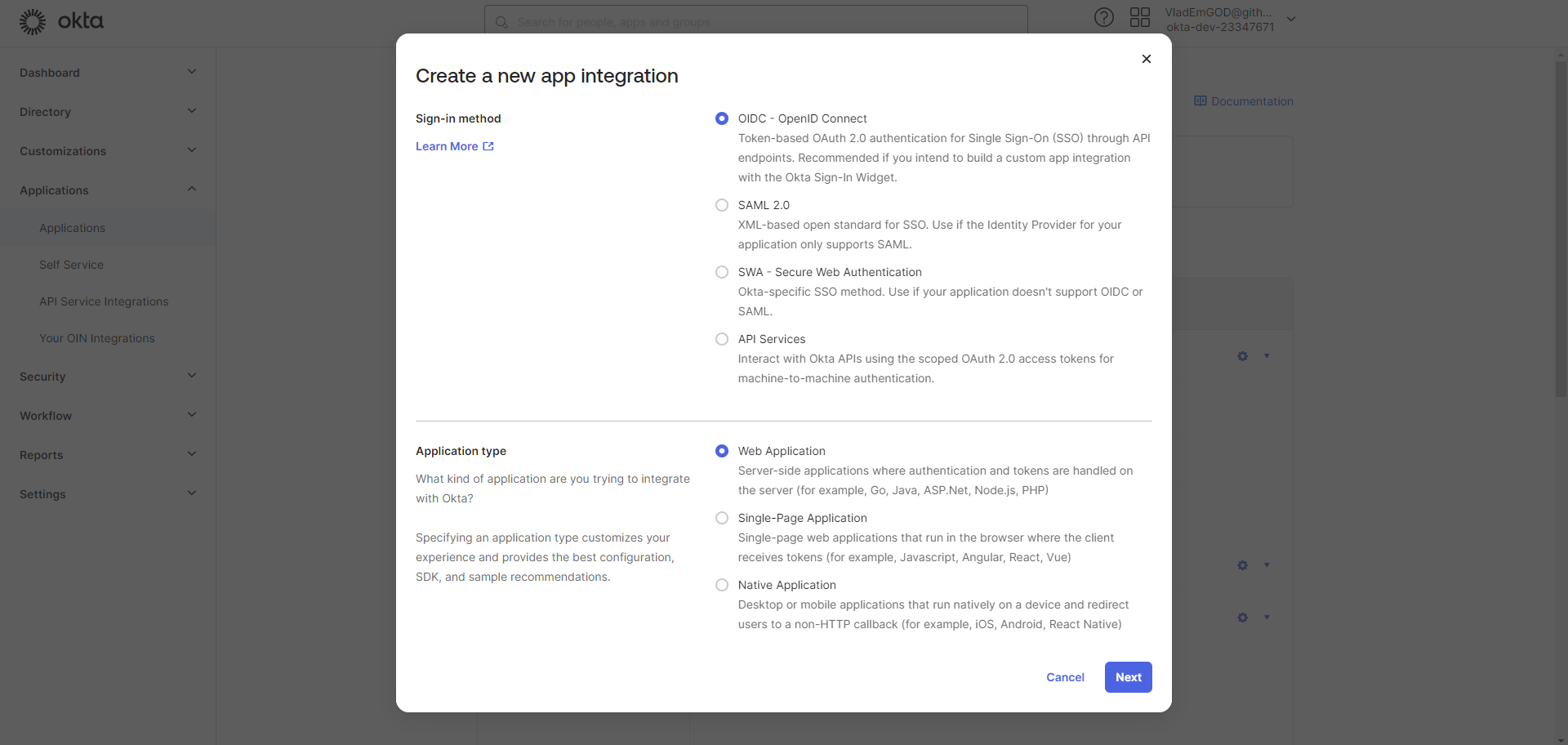
**Налаштування інтеграції**

Спочатку потрібно пройти стандарту процедуру автентифікації на офіційному сайті <https://www.okta.com>, де потрібно буде ввести всю необхідну для сервісу персональну інформацію. Після чого ми зможемо потрапити в директорію:



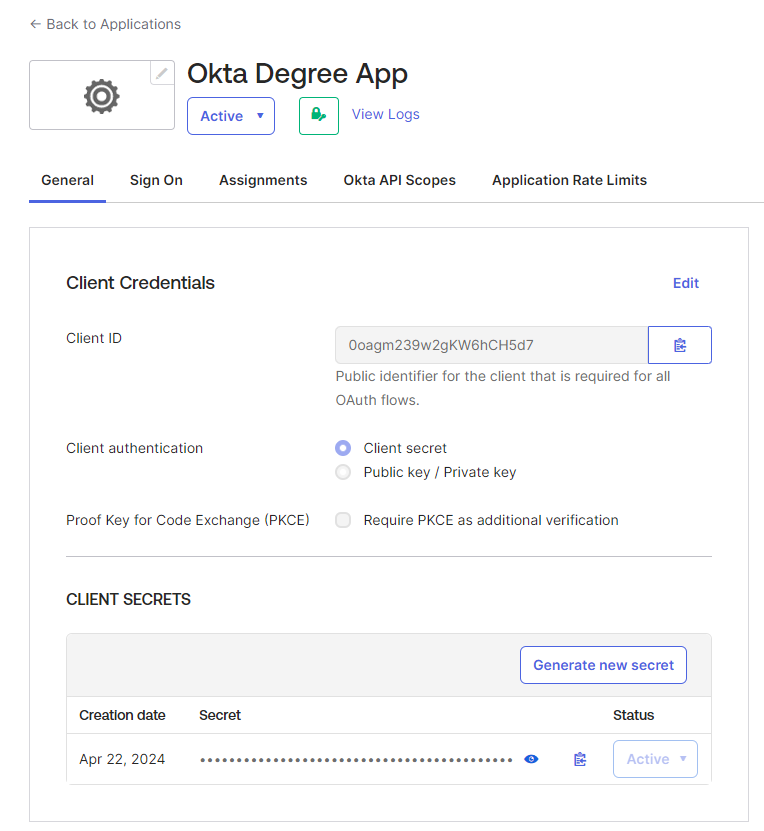
*Мал. 3.2.1 Директорія Okta*

Тепер нам потрібно стоврити інтеграцію для нашого інтернет магазину. треба буде перейти у панелі керування до вкладки "Applications". Тут ми можемо бачити список всіх застосунків. Натискаємо кнопку "Create App Integration", і Okta запропонує нам вибрати тип застосунку. Вибираємо протокол Open Id Connect. Та вибираємо темпейт "Web application".



*Мал. 3.2.2 Створення нової інтеграції для магазину в директорії*

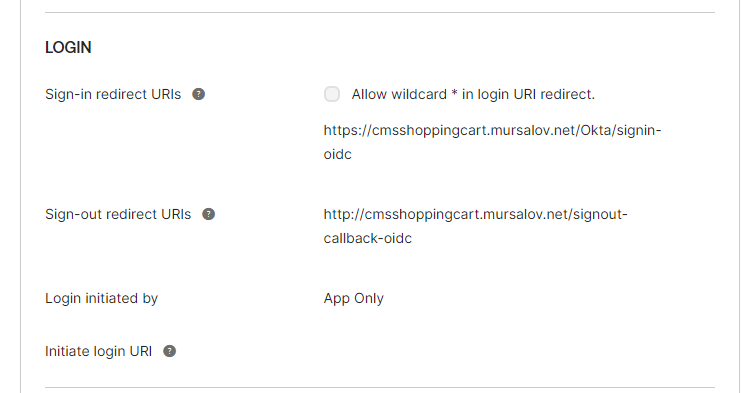
Після створення застосунку, можемо бачити потрібні нам згенеровані client-id та client secret.



*Мал. 3.2.3. Створена інтеграція в провайдері*

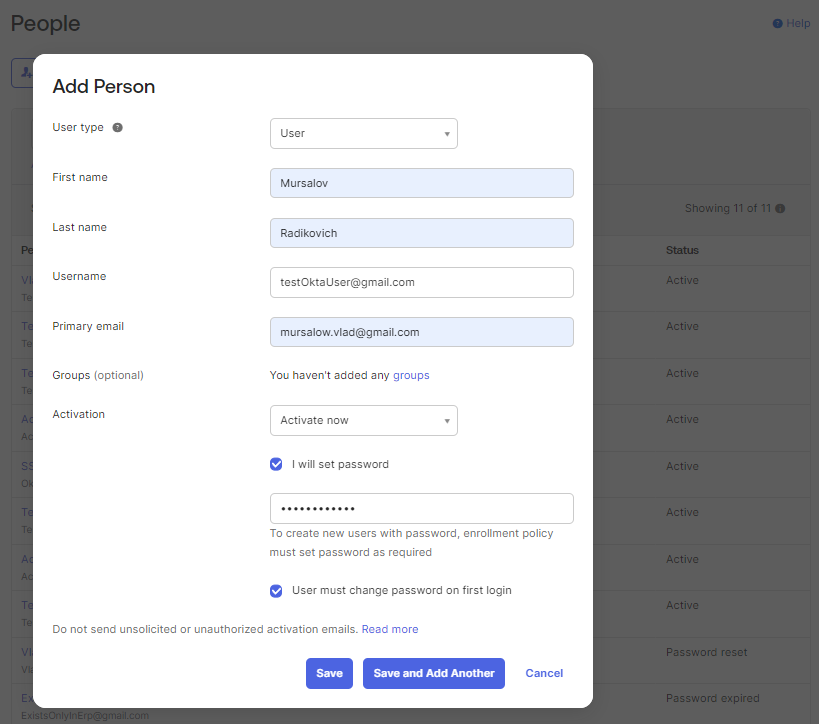
Далі для застосунку потрібно налаштувати URL-адреси. Ці URL-адреси вказуються для спрямування користувачів після упізнання та успішної або неуспішної автентифікації. Sign-in Redirect URL – це URL, на який користувач буде перенаправлений після успішної аутентифікації на сервері OIDC.

Sign-out Redirect URL - це URL, на який користувач буде перенаправлений після виходу з системи або розлогінення.



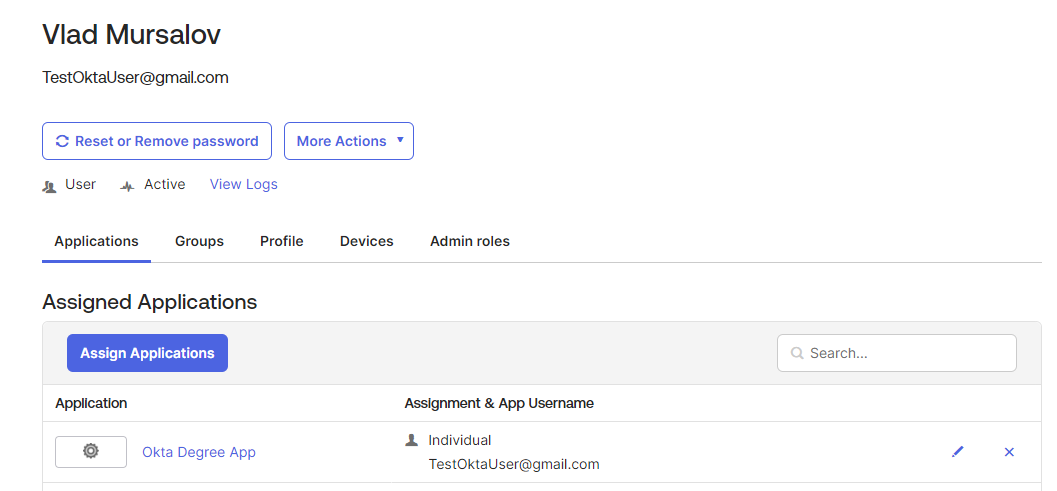
*Мал. 3.2.4. Налаштування Sign-in та Sing-Out Redirect URL*

Тепер додамо першого користувача. Для додавання користувача в Okta нам потрібно увійти до панелі керування Okta та перейти до вкладки "People". Натиснути кнопку "Add Person" та заповнити наступну форму, де потрібно вказати необхідні дані:



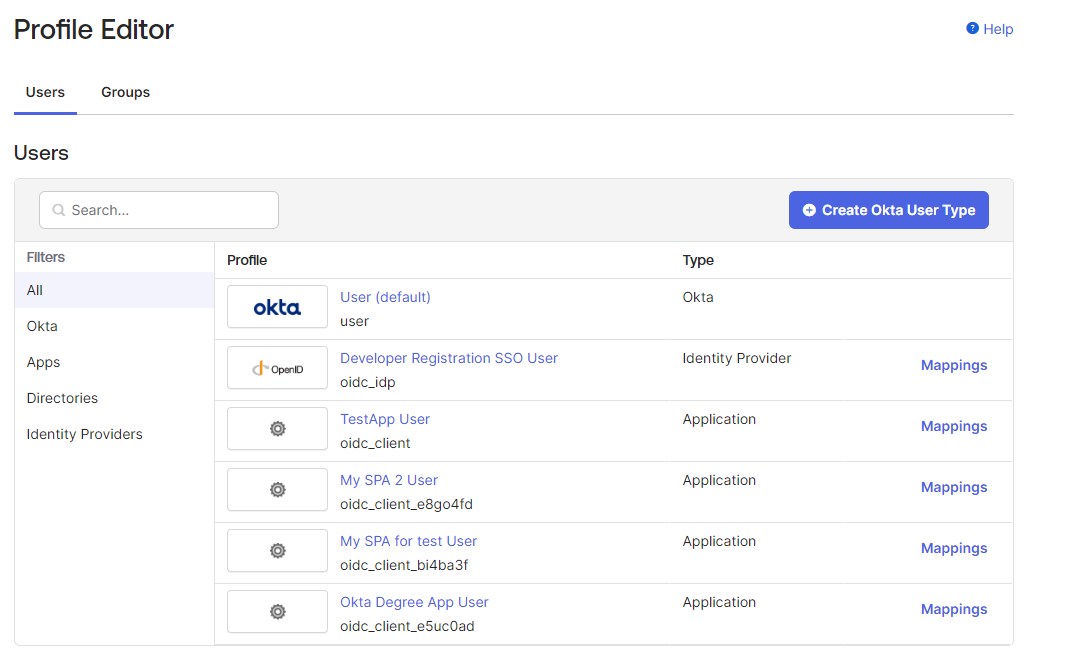
*Мал. 3.2.5. Створення користувача в провайдері*

Далі потрібно додати цього користувача до цієї інтеграції, щоб вім мав змогу через цю інтеграцію автентифікуватися. Спочатку потрібно відкрити потрібний додаток у вкладці "Applications". Потім перейти до вкладки "Assignments" у налаштуваннях застосунка. Тут можна додати користувачів, вибравши їх із списку. Можемо бачити наступне:



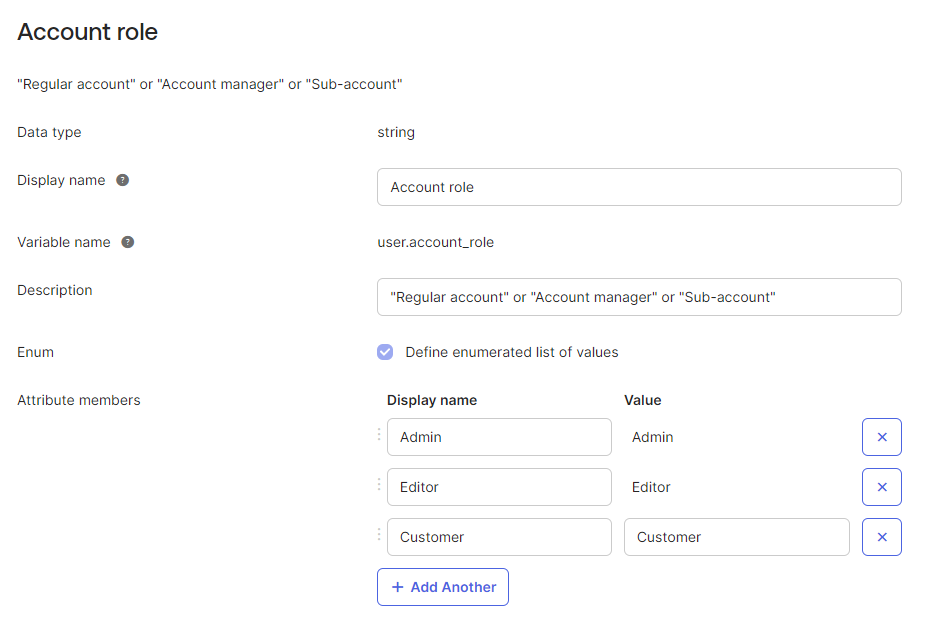
*Мал. 3.2.6. Прив’язка користувача до відповідного додатку*

Тепер потрібно додати користувачу роль в інтернет магазині. Окта дозволяє надати користувачу роль в адмін директорії але ц роль не передається в Id токен. Але рішення є, в окті можна сконфігурувати свої клейми. Потрібно перейти по шляху Directory > Profile Editor:



*Мал. 3.2.7. Створення користувача в провайдері*

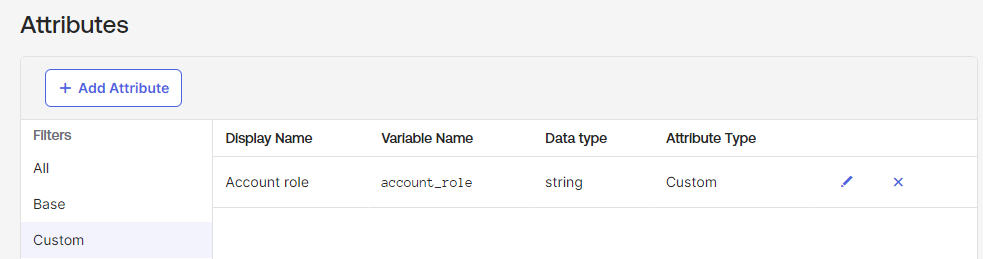
Можемо бачити тут профілі користувачів для всіх інтеграції які створені в директорії. Спочатку нам порібно відредагувати профіль користувача за замовчуванням. Цей профіль є базою для всіх інтеграцій, це користувач з найбільшою кількістью даних в директорії. Потім інтеграції вибирають потрібні їм дані з нього. Отже, створюємо новий клейм account\_role:



*Мал. 3.2.8. Створення користувача в провайдері*

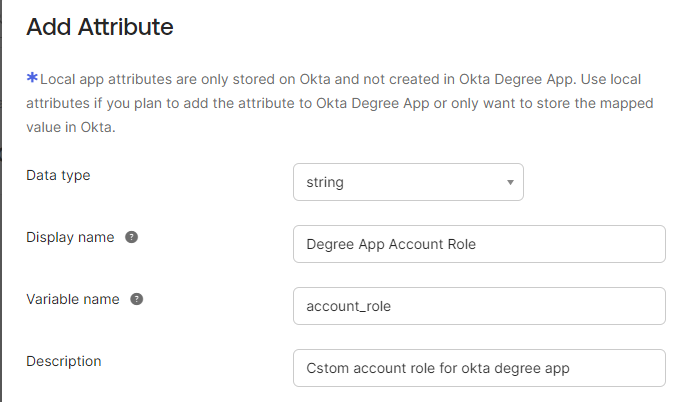
Налаштовуємо тип дляних як string і для зручності заздалегідь визначаємо 3 ролі,

Після чогоможемо бачити клейм в профілі користувача за амовчуванням:



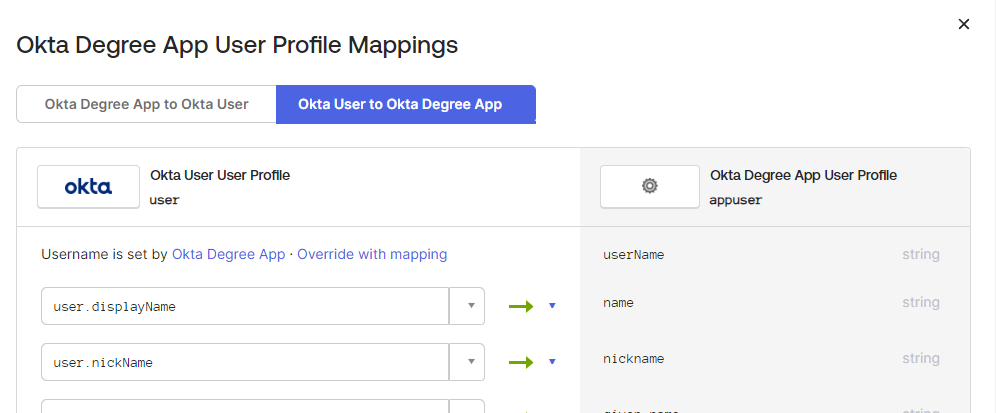
*Мал. 3.2.9. Створення користувача в провайдері*

Тепер портібно визначити такий атрибут персонально для нашої інтеграції. Заходимо в профіль користувача інтеграції, додаємо атрибут з іменем змінної account\_role, описом та читабельним іменем:



*Мал. 3.2.10. Створення користувача в провайдері*

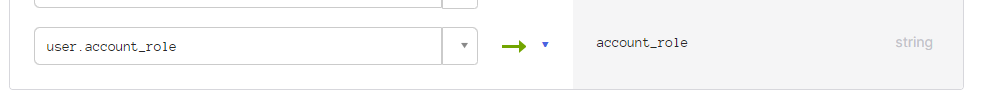
І нарешті потрібно зіставити атрибут профілю користувача за замовчуванням з профілем користувача нашої інтеграції. Для цього нам потрібно зайти по тому ж шляху та натиснути кнопку “Mappings”:



*Мал. 3.2.11. Створення користувача в провайдері*

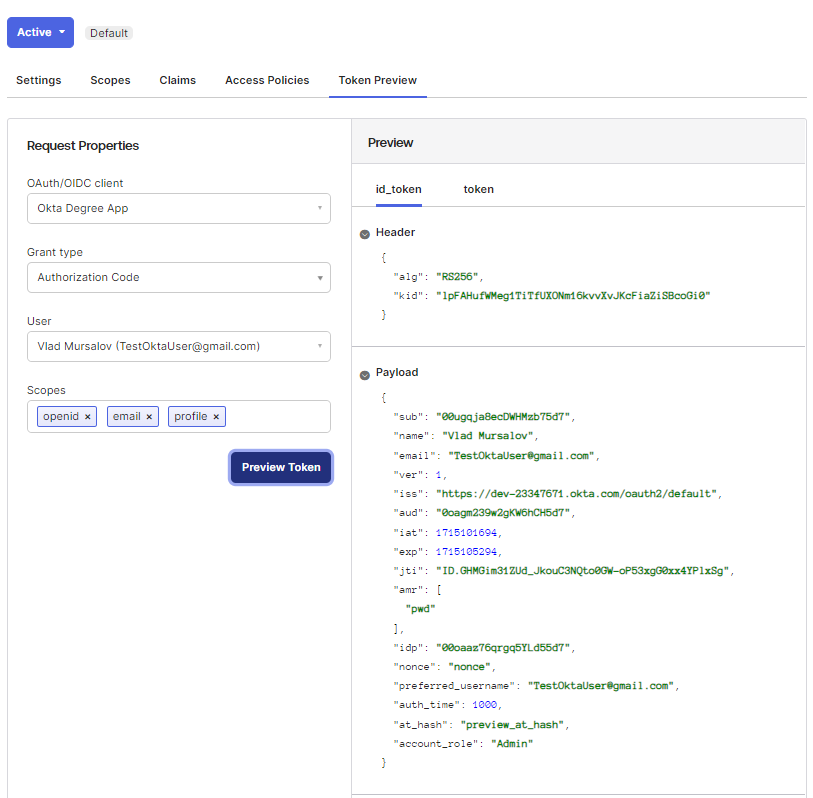
Можемо бачити 2 таби: перша це сторінка з зіставлення атрибутів користувача інтеграції з атрибутом користувача за замовчуванням, друга – навпаки.

Відповідно зіставляємо 2 створених атрибути.



*Мал. 3.2.12. Створення користувача в провайдері*

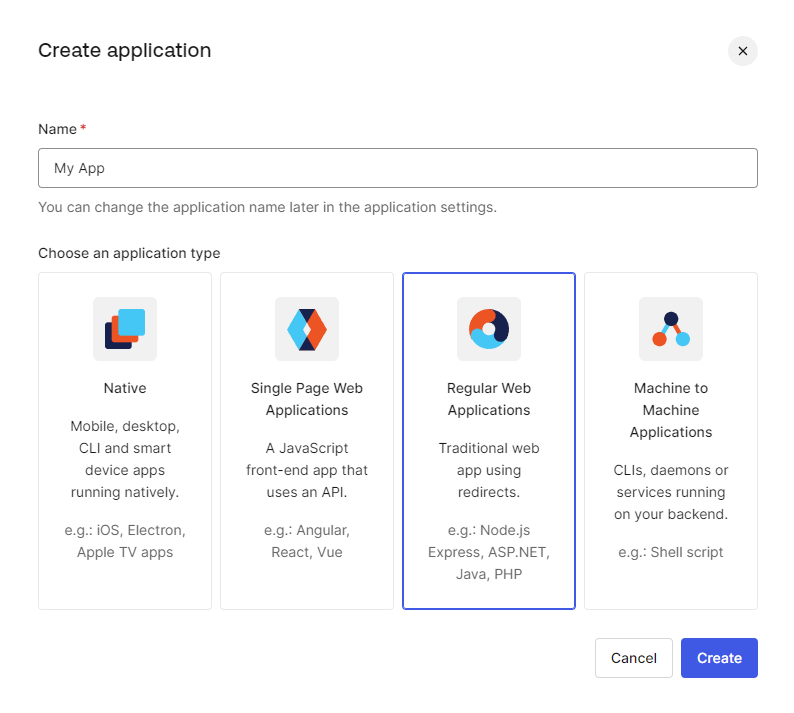
На цьому налаштування можна вважати закінченим. Тепер можемо побачити структуру Id токену для цього користувача при автентифікації до цього додатку. Треба перейти в Security > Api > Authorization servers. Перейти в потрібний нам сервер та обрати фічу Token Preview. Після чого вибираємо створений додаток, тип автентифікації – Authorization code, тільки що створеного користувача та передаємо потрібні скоупи. Переконуємось інформація в токені що токені вірна.



*Мал. 3.2.13. Прив’язка користувача до відповідного додатку*

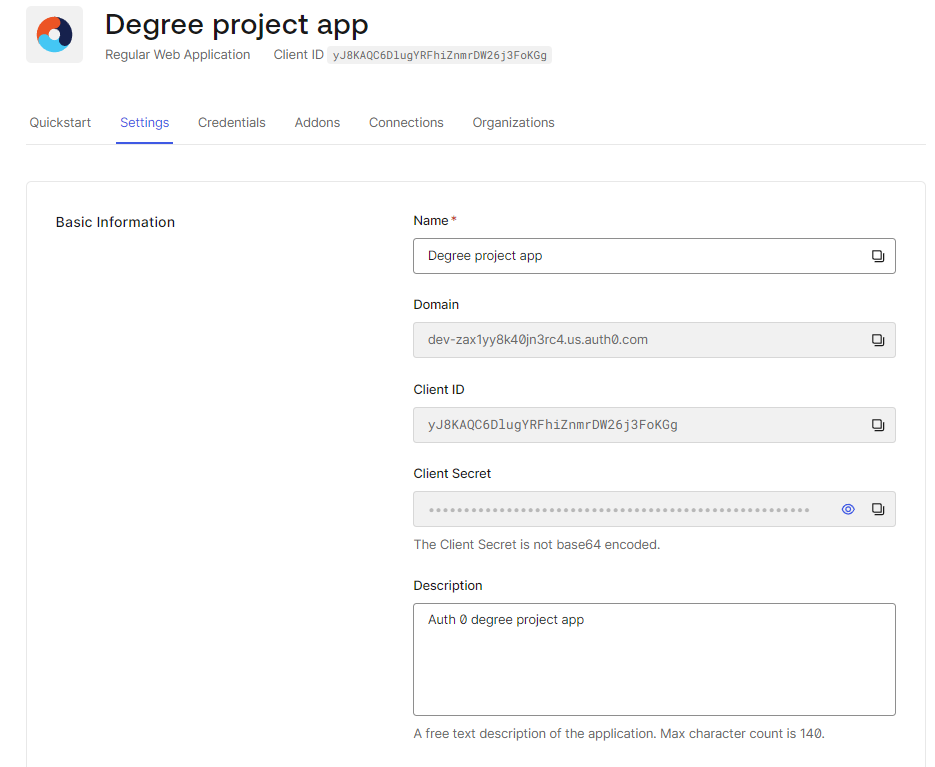
### 3.3. Налаштування Auth0 як серверу автентифіації

Як і в Окті, потрібно пройти базову автентифікацію на сервісі та створити першу директорію. Потім потрібно створити додаток з викристанням шаблону “Regular Web Applications”:



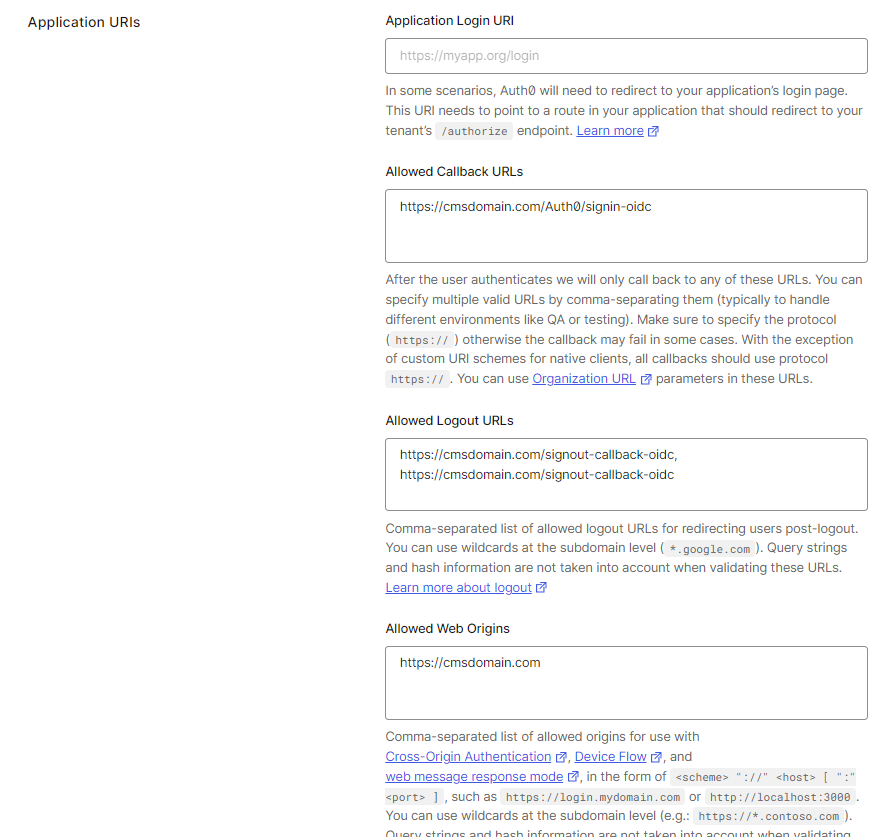
*Мал. 3.3.1. Створення нового додатку-інтеграції для Auth0*

Після створення додатку-інтеграції, відповідно можемо бачити домен нашого серверу автентифікації, ідентифікатор цього додатку та секрет:



*Мал. 3.3.2. Дані для підключення магазину до додатку Auth0*

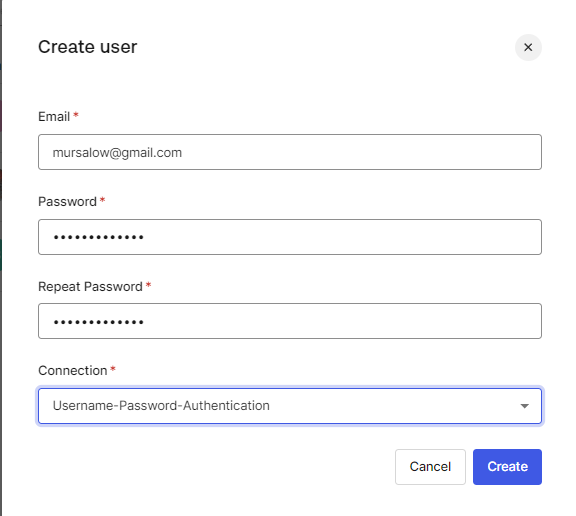
Відповідно до протоколу потрібно ввести маршрути для взаємодії з Auth0 та інтернет магазину. Вони будуються за таким же принципом як і для Окти:



*Мал. 3.3.3. Налаштування маршрутів для працездатності OIDC*

**Створення користувачів:**

Переходимо в User management > Users та тиснемо на кнопку Create new user та вводимо необхідні дані:



*Мал. 3.3.4. Створення користувачів в Auth0*

**Налаштування додаткових клеймів в Id токен:**

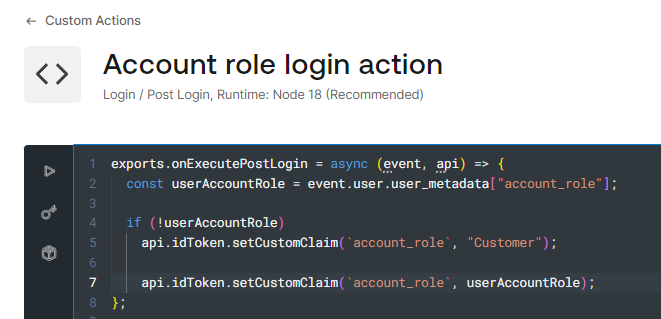
Як і в Окті, роль користувача налаштувати можна, але ця роль розповсюджується на адміністрування в директорії і не передається в Id токен. Тому як і в випаду з Октою нам треба створити власний клейм.

Auth0 має досить зручну систему для модифікацій ходу автентифікації яка називається “Actions”. Кожна подія це програмований код на мові JavaScript. Щоб додати додаткову інформацію в Id токен, переходимо в профіль користувача, спускаємось на до розділу “Metadata” та вводимо додаємо account\_role клейм з відповідним значенням до метадати користувача в форматі JSON:



*Мал. 3.4.1. Додавання ролі в метадату*

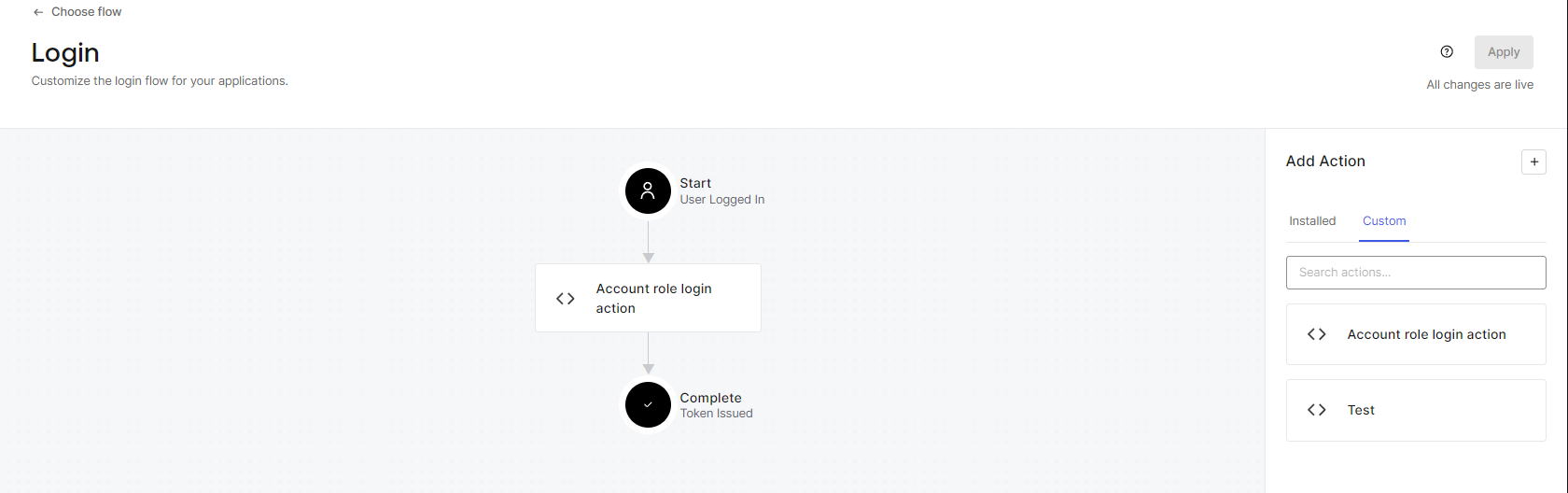
Переходимо в Actions > Library та створюємо нову подію. Тиснемо кнопкуCreate action та вибираємо опцію Build from scratch. В тіло події додаємо наступний код:



*Мал. 3.4.2. Код події для додавання ролі в токен*

Цей код спробує взяти з метаданих користувача його роль, якщо її адміністратор не вказав – будемо вважати що це звичайний покупець.

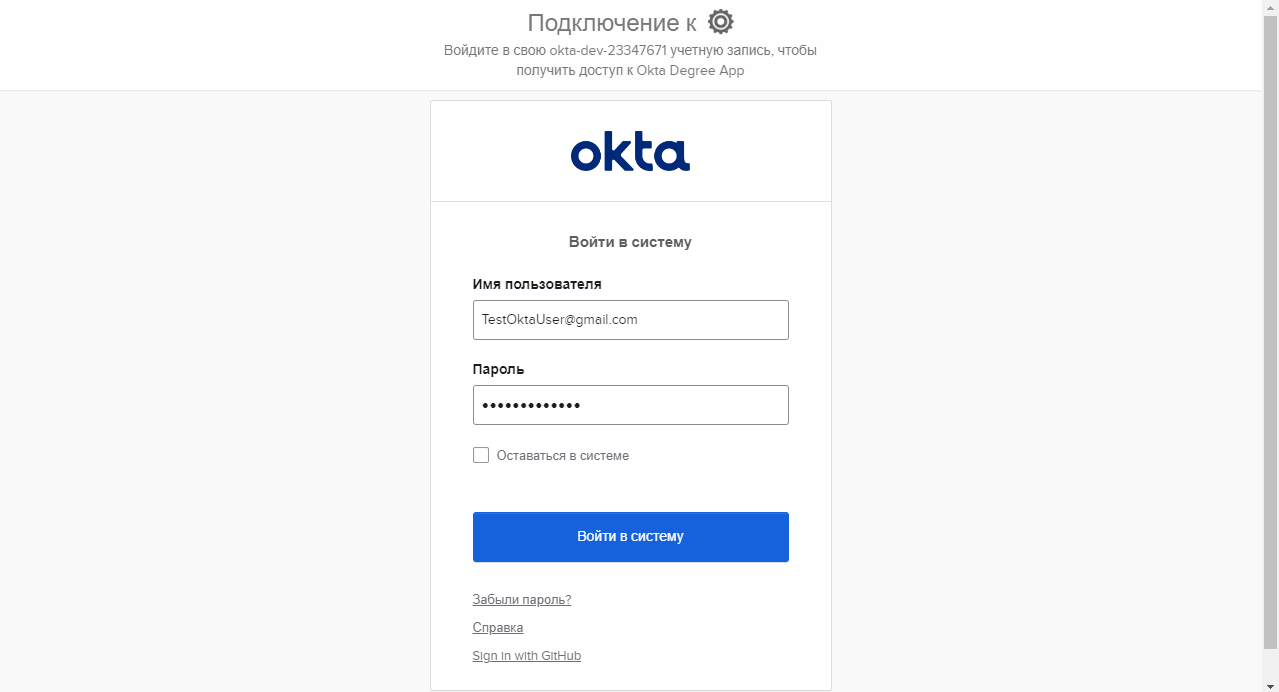
Далі переходимо в Actions > Flows. Обираємо процес Login. В правій панелі “Add action” переходимо в Cumstom табу та обираємо створену подію. Перетягуємо подію між початком та кінцем автентифікації користувача.



*Мал. 3.4.3. Додавання події в процес автентифікації користувача*

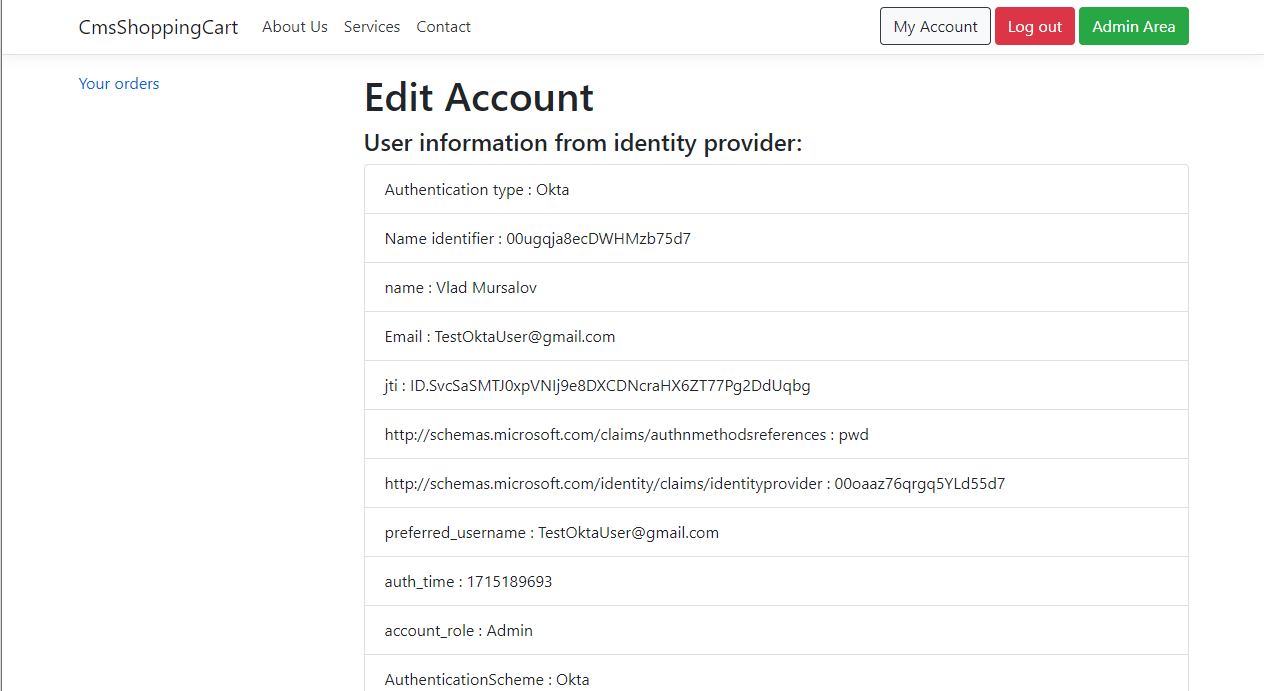
### 3.4 Тестування автентифікації за допомогою Okta

Спочатку перевіримо чи можлива успішна автентифікація користувача. В інтернет магизині, переходимо на сторінку авторизації, натискаємо кнопку «Login with Okta» на сторінці авторизації, потрапляємо уже на сторінку автентифікації Окти:



*Мал. 3.4.1. Сторінка авторизації Okta*

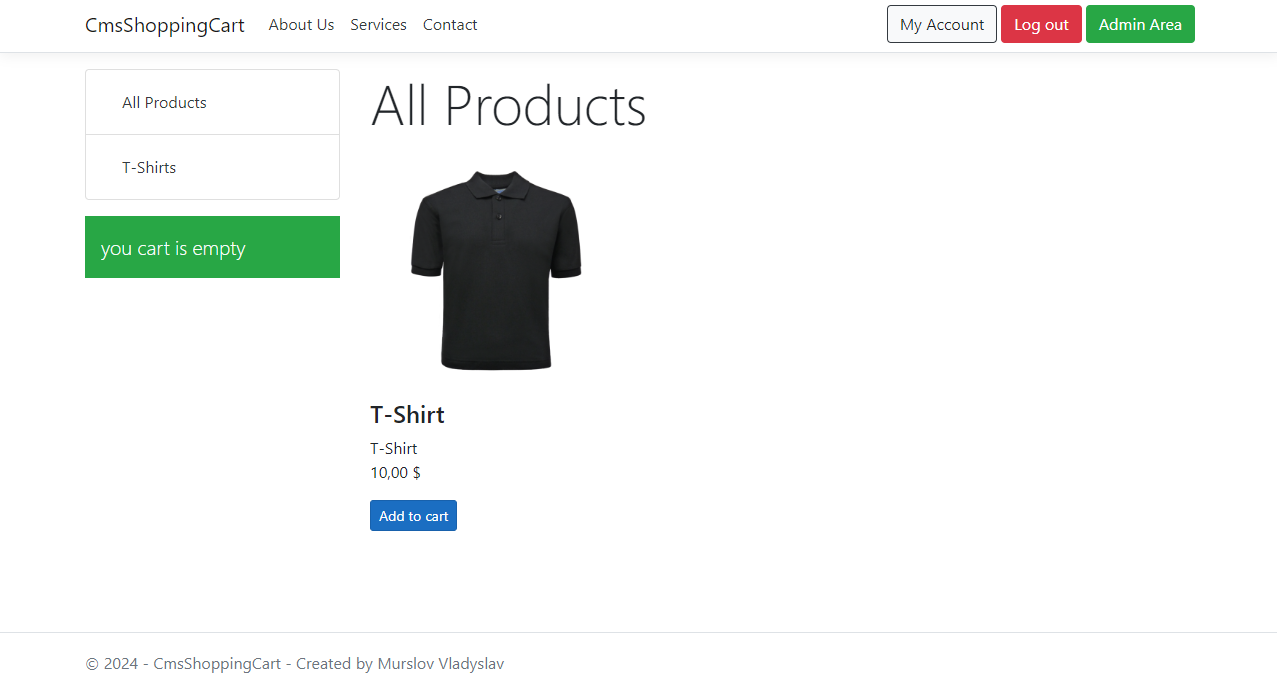
Стандартно вводимо пошту та пароль на стороні Окти. Потім нас запитають дозволи на доступ до інформації профілю та пошти, дозволяємо це, після чого нас успішно автентифікувало та перенаправило на сторінку профілю:



*Мал. 3.4.2. Інформація про авторизованого окристувача*

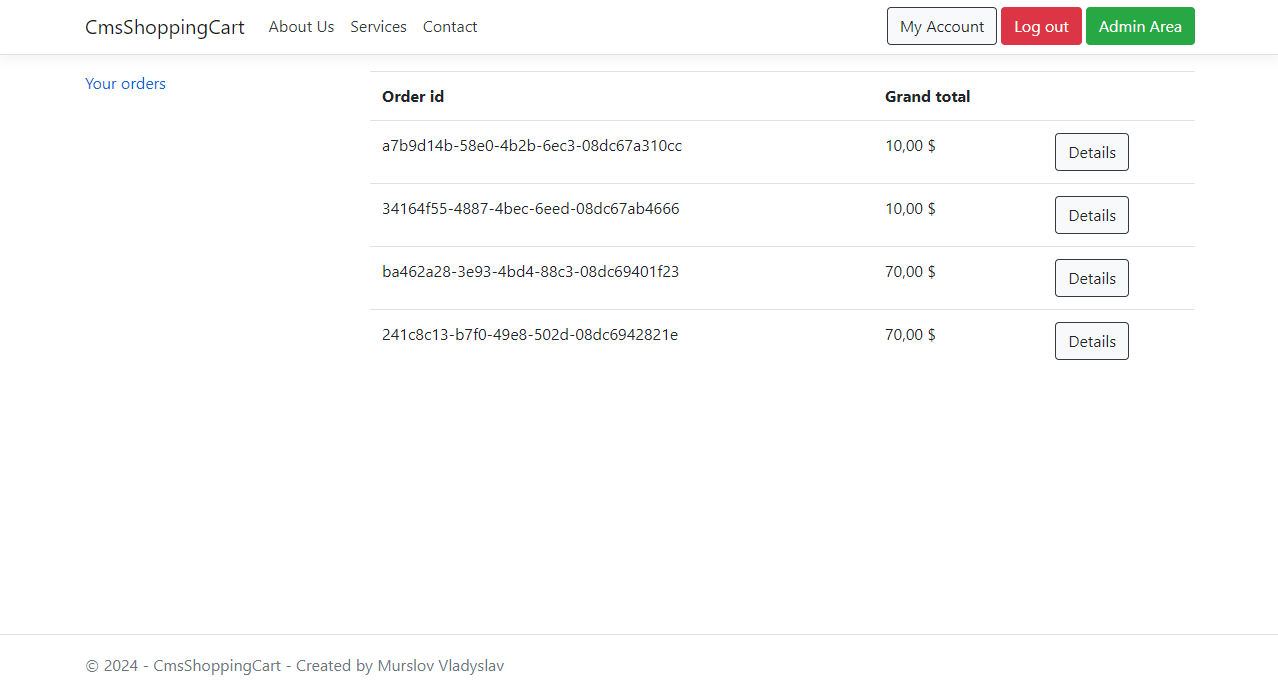
Для демонстрації ми навмисно вивели всю доступну інформацію яку отримує інтернет магазин. Можемо бачити що схема автентифікації – Okta та бачимо створений нами клейм прийшов з правильною ролью.

Тепер перевіримо базу даних, спробуємо купити декілька продуктів, Переходимо на сторінку з продуктом та замовляємо його декіль каразів в різних кількостях:



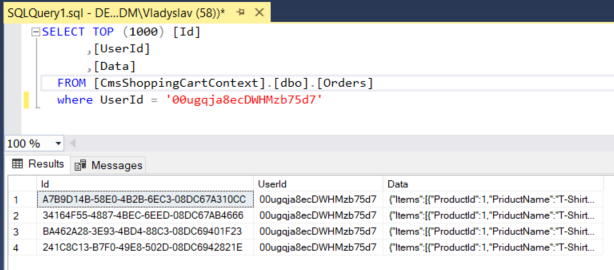
*Мал. 3.4.3. Сторінка з продуктом*

Переходимо на сторінку свого профілю, та можемо бачити 4 замовлення:



*Мал. 3.4.4. Сторінка з замовленнями*

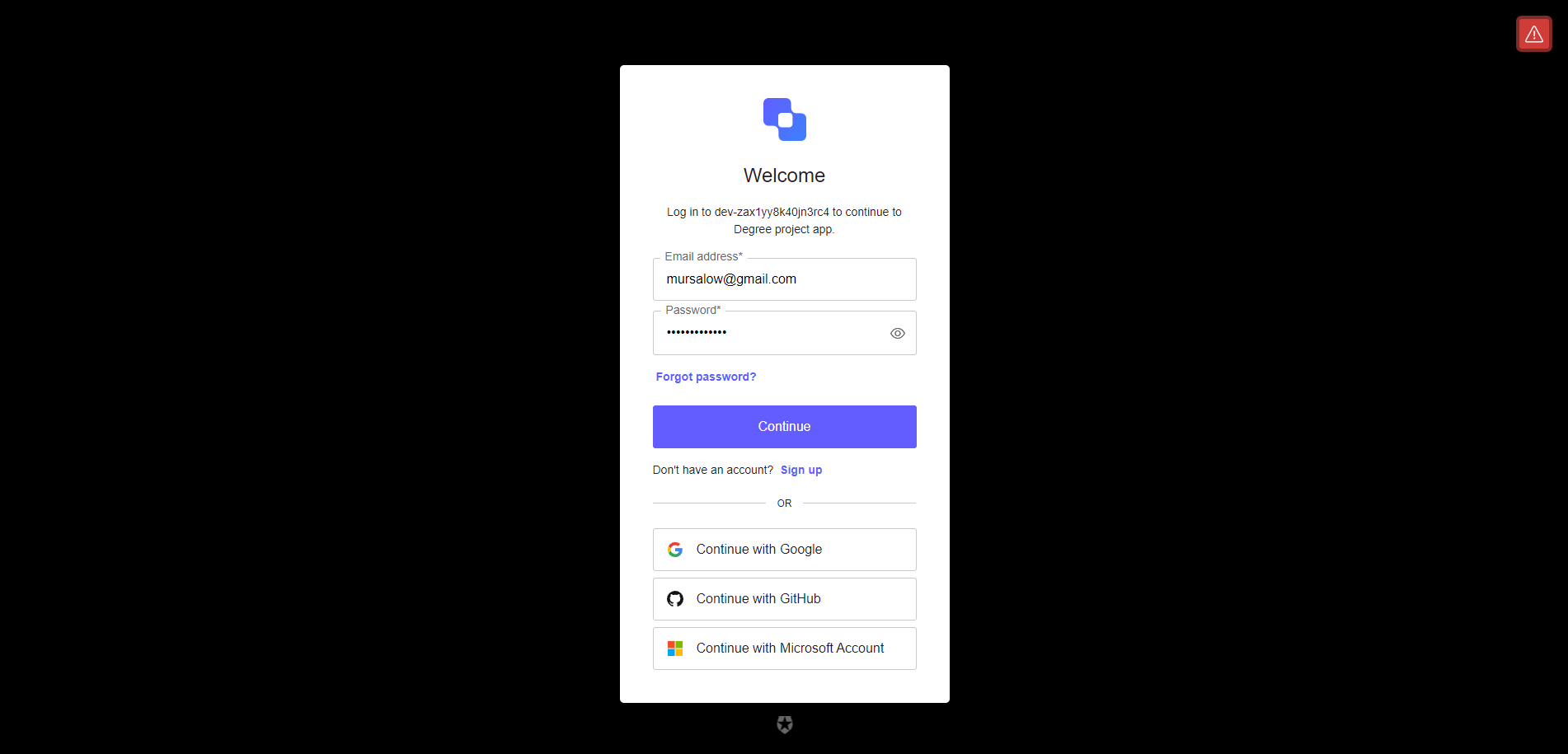
Ці ж 4 замовлення можемо побачити в базі даних, що значить що вона справно функціонує:



*Мал. 3.4.5. Замовлення в базі даних*

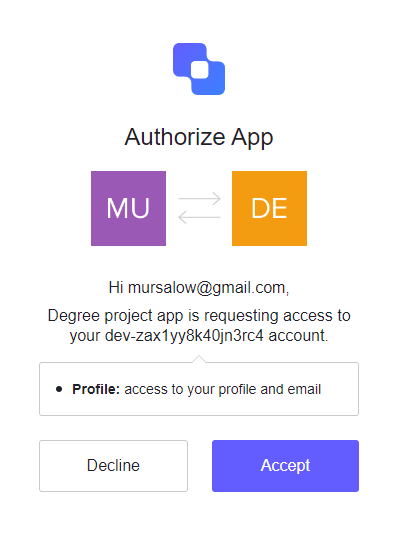
### 3.5 Тестування автентифікації за допомогою Auth0

Нак само, перевіримо чи можлива успішна автентифікація користувача. В інтернет магизині, Натискаємо кнопку «Login with Auth0» та потрапляємо на сторінку автентифікації Auth0:



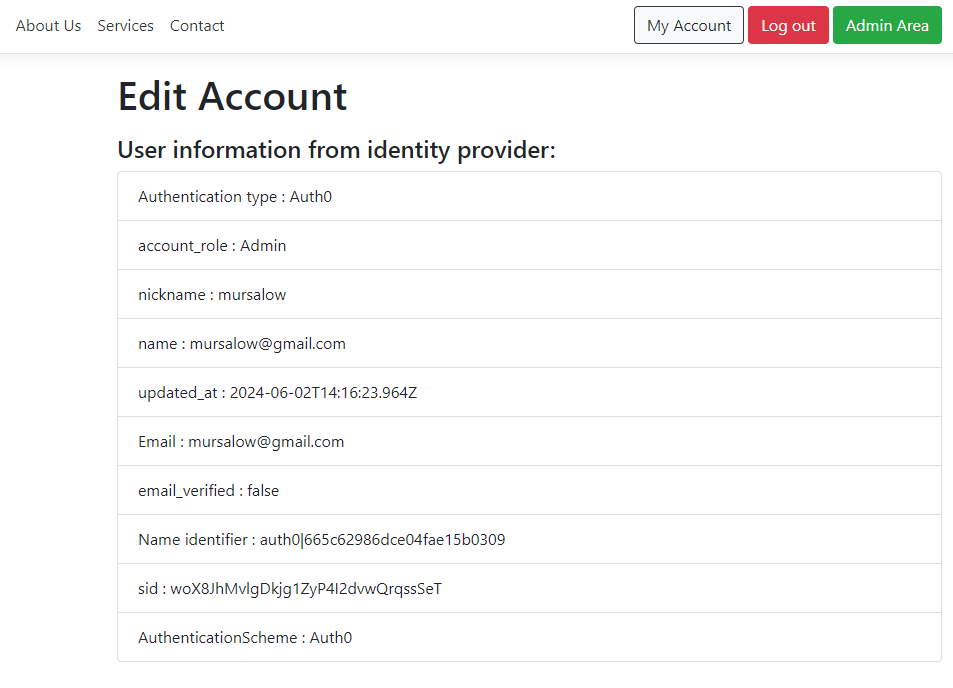
*Мал. 3.4.1. Сторінка автентифікації в Auth0*

Вводимо наші дані, після чого нас запитають дозволи на доступ до інформації профілю та пошти, дозволяємо це, після чого нас успішно автентифікувало та перенаправило на сторінку профілю.



*Мал. 3.4.2. Прийняття скоупів на доступ інформації в провайдері*

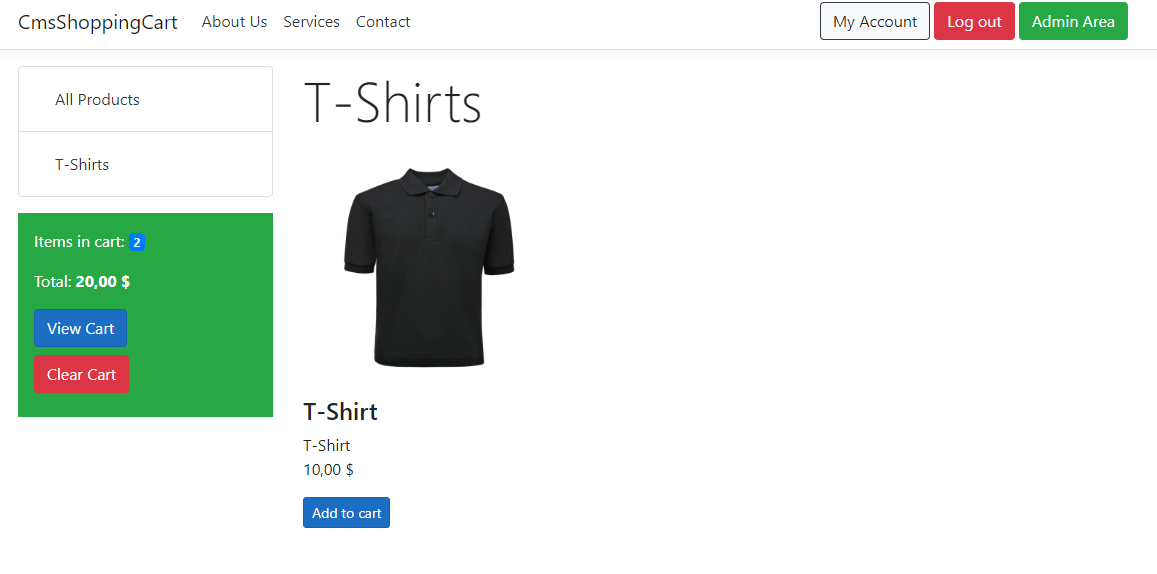
Після успішної автентифікації нас Auth0 з результатами автентифікації перенаправляє на сторінку профілю користувача з актуальною інформацію:



*Мал. 3.4.3. Отримана інформація користувача з Auth0*

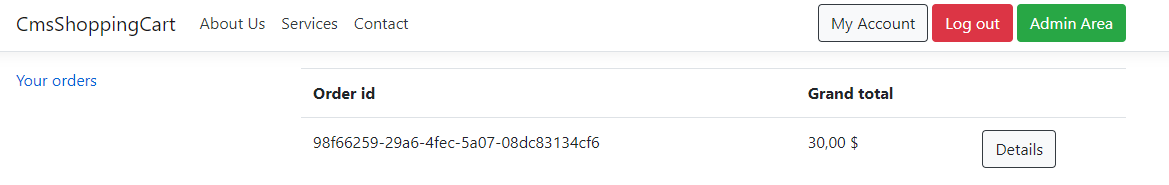
Бачимо що прийшла актуальна інформація, схема автентифікації відповідає налаштованому IdP і ми отримали потрібну нам роль налашнована роль.

Спробуємо замовити 2 продукти, переходимо на сторінку продукту, замовляємо декілька штук.



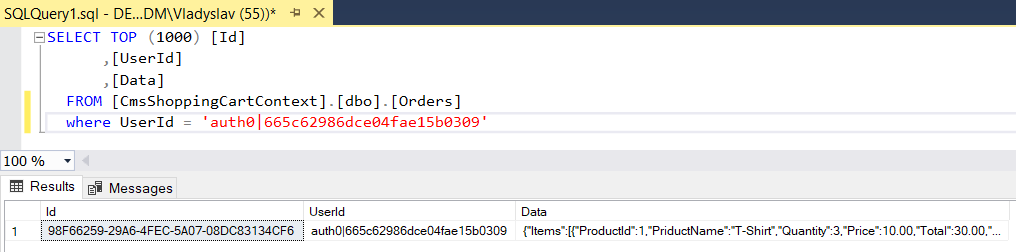
*Мал. 3.4.4. Замовлення продукту в інтернет магазині*

Після оформлення замовлення, можемо бачити його в списку замовлень:



*Мал. 3.4.5. Замовлення на сторінці замовлення*

Відповідний запис в базі даних:



*Мал. 3.4.6. Замовлення в базі даних*

Можемо бачити що це саме те замовлення по його Id. Таков можемо бачити що UserId в базі співпадає з тим Id що ми отримуємо з серверу автентифікації Auth0.

**Висновок**: Проведене тестування показало, що розроблена система динамічного додавання систем ідентифікації працює як задумано. Автентифікація користувачів через такі провайдери, як Okta та Auth0, є надійним та ефективним способом забезпечення безпеки і доступу до веб-додатків.

Під час тестування було продемонстровано:

1. Процедуру налаштування та інтеграції Okta і Auth0 як серверів автентифікації.

2. Процес автентифікації користувачів через обидва провайдери, включаючи прийняття необхідних скоупів та отримання коректних ролей.

3. Успішне створення та обробку замовлень.

# ВИСНОВКИ

Висновки з кваліфікаційної роботи. Констатуємо, яких завдань, поставлених на початку кваліфікаційної роботи, будо досягнуто. За кожним завданням наводимо конкретні результати. Якщо у вас добре написані висновки за кожним розділом, то їх можна взяти за основу висноків з кваліфікаційної роботи.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ