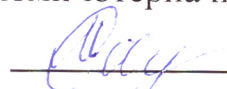


Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект 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 доцент кафедри КІ та КБ, к.пед.н., доцент

 Олена Головня

Рецензент професор кафедри КІ та КБ, д.т.н., доцент

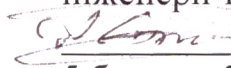
 Володимир Воротніков

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 комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко
5 березня 2024 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Мурсалов Владислав Радікович

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

затверджена Наказом університету від 5 березня 2024 р. №97

Термін здачі студентом закінченої роботи 9 червня 2024 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

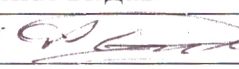
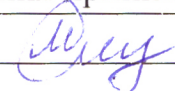
Об'єктом дослідження є системи автентифікації для онлайн-магазинів.

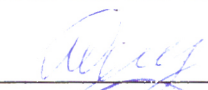
Предметом дослідження виступає підсистема автентифікації на базі протоколу OpenID Connect з використанням різних провайдерів ідентифікації.

Обладнання та програмне забезпечення:

1. Okta Workspace Identity Cloud та Auth 0
2. ASP.NET MVC з MS SQL Server
3. Операційні системи Windows
4. Visual Studio Community / Visual Studio Code

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Андрій Єфіменко	5.03.24 	5.03.24 

Студент  Владислав Мурсалов

Керівник  Олена Головня

Календарний план

№ З/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	05.03.2024	Виконано
2	Написання вступної частини кваліфікаційної роботи	06.03.2024 - 12.03.2024	Виконано
3	Аналіз теоретичних основ автентифікації	13.03.2024 - 02.04.2024	Виконано
4	Проектування рішення	03.04.2024 - 23.04.2024	Виконано
5	Імплементация рішення	24.04.2024 - 21.05.2024	Виконано
6	Тестування проекту	22.05.2024 - 01.06.2024	Виконано
7	Оформлення кваліфікаційної роботи	02.06.2024 - 06.06.2024	Виконано
8	Підготовка презентації до захисту	07.06.2024 - 09.06.2024	Виконано

Студент  Владислав Мурсалов

Керівник  Олена Головня

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи складається з вступу, трьох розділів, висновків, списку використаної літератури та додатків.

Текстова частина викладена на 71 сторінці друкованого тексту. Список використаних джерел містить 17 найменувань і займає 2 сторінки. В роботі наведено 44 рисунки. Загальний обсяг роботи – 75 сторінок.

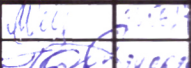
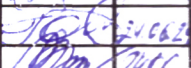
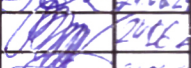
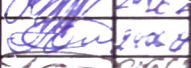
Метою кваліфікаційної роботи є розробка проекту підсистеми автентифікації онлайн-магазину на базі протоколу Open ID Connect та різних провайдерів ідентифікації.

Було проаналізовано теоретичні основи автентифікації з використанням технології Single Sign-On (SSO). Було розглянуто недоліки традиційних методів автентифікації, таких як паролі, та переваги SSO, включаючи зручність використання та централізоване адміністрування. Окрему увагу приділено протоколам SAML та OIDC, їх особливостям, перевагам та сферам застосування, з акцентом на гнучкість і зручність інтеграції OIDC.

Після чого було спроектовано рішення для автентифікації онлайн-магазину. Сформульовано вимоги до проекту на основі бізнес-логіки онлайн-магазину, з урахуванням стратегічних потреб та наявного функціоналу. Оглянуто технологічний стек, та визначено способи інтеграції обраних провайдерів ідентифікації з існуючою кодовою базою. Докладно розглянуто етапи імплементації рішення, включаючи динамічну реєстрацію IdP, автентифікацію та розмежування ролей користувачів.

В результаті було розроблено та успішно протестовано проект, який за допомогою протоколу OIDC здатний ідентифікувати користувачів інтернет-магазину на основі різних провайдерів ідентифікації.

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

					ККІК.КР.Б – 123 – 17 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Проект підсистеми автентифікації онлайн-магазину на базі протоколу Open ID Connect та різних провайдерів ідентифікації	Літ.	Арк.	Аркуші
Розроб.		Мурсалов В.Р.		2024.06.24			4	75
Керівник		Головня О.С.		2024.06.24		ФІКТ, Гр. КІ-20-1		
Реценз.		Воротніков В.В.		2024.06.24				
Н. контр.		Головня О.С.		2024.06.24				
Затверд.		Сфіменко А.А.		2024.06.24				

ANNOTATION

The explanatory note for the course project 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 presented on 71 pages of printed text. The list of references includes 17 titles and occupies 2 pages. The work contains 44 figures. The total volume of the work is 75 pages.

The aim of the qualification work is to develop a project for an authentication subsystem for an online store based on the Open ID Connect protocol and various identity providers.

The theoretical foundations of authentication using Single Sign-On (SSO) technology were analyzed. The disadvantages of traditional authentication methods, such as passwords, and the advantages of SSO, including ease of use and centralized administration, were considered. Special attention was paid to the SAML and OIDC protocols, their features, advantages, and areas of application, with an emphasis on the flexibility and ease of OIDC integration.

Next, a solution for the authentication of the online store was designed. The project requirements were formulated based on the business logic of the online store, taking into account strategic needs and existing functionality. The technological stack was reviewed, and methods for integrating the chosen identity providers with the existing code base were determined. The implementation stages of the solution were examined in detail, including dynamic IdP registration, authentication, and user role segregation.

As a result, a project was developed and successfully tested that can identify online store users using the OIDC protocol and various identity providers.

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

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ВСТУП	9
РОЗДІЛ 1 АНАЛІЗ ТЕОРЕТИЧНИХ ОСНОВ АВТЕНТИФІКАЦІЇ ЗА ДОПОМОГОЮ SINGLE SIGN-ON	11
1.1 Автентифікація, авторизація та ідентифікація в сучасній ІТ-індустрії.	11
1.2 Недоліки традиційної системи автентифікації та переваги Single Sign-on	13
Зручність використання	13
Адміністрування та підтримка	15
1.3 Протоколи для реалізації SSO	16
1.3.1 Протокол SAML	17
1.3.2 Протоколи OAuth 2.0 та OIDC	22
1.3.3 Порівняння протоколів для реалізації SSO	28
Висновки до розділу 1	31
РОЗДІЛ 2 ПРОЕКТУВАННЯ РІШЕННЯ	32
2.1. Формування вимог до проекту відповідно до бізнес-логіки онлайн-магазину	32
2.1.1 Стратегічні потреби від проекту	32
2.1.2 Огляд наявного функціоналу для проектування	36
2.1.3 Сформовані вимоги	37
2.2 Проектування рішення	37
2.2.1 Огляд технологічного стеку інтернет-магазину	37
2.2.2 Проектування рішення під наявну кодову базу	40
2.2.3 Проектування взаємодії між основними компонентами алгоритму	42
2.4 Імплементація рішення	44
2.4.1. Динамічна реєстрація IdP	44
2.4.2. Автентифікація через IdP	49
2.4.3. Вихід з IdP	52
2.4.4. Розмежування ролей автентифікованого користувача	53
Висновки до розділу 2	53

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3 ТЕСТУВАННЯ ПРОЕКТУ

54

3.1 Тестування роботоздатності інтернет-магазину	55
3.2. Налаштування Окта як серверу автентифікації	57
3.3. Налаштування Auth0 як серверу автентифікації	63
3.4 Тестування автентифікації за допомогою Окта	66
3.5 Тестування автентифікації за допомогою Auth0	69
Висновки до розділу 3	72

ВИСНОВКИ

73

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

74

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

SSO – Single sign-on

IDaaS – Identity as a Service

SaaS –Software as a Service

OIDC – Open ID Connect

IdP – Identity Provider

RP – Relying Party

JWT – JavaScript Web Token

SP - Service Provider

MFA – Multi-Factor Authentication

OTP – One Time Password

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

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 підтверджує достовірність 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 різних провайдерів ідентифікації.

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

Визначена мета обумовлює наступні завдання.

- Провести аналіз наявних 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.
- Розробити, налаштувати та протестувати проект підсистема автентифікації онлайн-магазину на базі протоколу OpenID Connect та різних провайдерів автентифікації.

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

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

Апробація результатів:

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Мурсалов В.Р., Головня О.С.. Single Sign-On для Інтернет-магазину за допомогою протоколу OpenId Connect. Тези XIV Міжнародної науково-технічної конференції "Інформаційно-комп'ютерні технології - 2024", 28-29 березня 2024 року. Житомир : «Житомирська політехніка», 2024. С. 71-72.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ ТЕОРЕТИЧНИХ ОСНОВ АВТЕНТИФІКАЦІЇ ЗА ДОПОМОГОЮ SINGLE SIGN-ON

1.1 Автентифікація, авторизація та ідентифікація в сучасній ІТ-індустрії

У сучасному світі, де інформаційні технології грають центральну роль у функціонуванні бізнесу, урядових установ і особистому житті, питання забезпечення безпеки даних стають надзвичайно актуальними. Автентифікація, авторизація та ідентифікація є ключовими процесами у цій сфері, що забезпечують надійний контроль доступу до інформаційних ресурсів і захист від несанкціонованого доступу. Ці процеси, кожен з яких виконує специфічну роль, взаємодіють між собою, створюючи багатоварову систему безпеки. В умовах швидкого розвитку технологій та зростання кіберзагроз їх значення стає ще більш очевидним.

Автентифікація є фундаментальним процесом, який забезпечує підтвердження особи користувача або системи, що звертається до ресурсу. Від правильності автентифікації залежить безпека всієї системи, оскільки саме цей процес встановлює, чи є користувач тим, за кого він себе видає. Традиційні методи автентифікації, такі як паролі, вже не забезпечують достатнього рівня захисту через їх уразливість до атак типу брутфорс, фішинг і соціальної інженерії. У зв'язку з цим, значну популярність набули багатофакторна автентифікація (Multi-Factor Authentication, MFA) та біометричні методи, які поєднують кілька факторів для підтвердження особи. Використання одноразових паролів (One Time Password, OTP), які генеруються на основі часу або подій, значно ускладнює завдання для зловмисників. Застосування біометричних даних, таких як відбитки пальців, розпізнавання обличчя або сканування райдужної оболонки ока, підвищує рівень безпеки завдяки унікальності цих характеристик.

Авторизація є наступним кроком після автентифікації і відповідає на питання: "Що користувач має право робити?". Це процес надання або обмеження доступу до ресурсів або даних на основі прав та привілеїв, встановлених для користувача. Авторизація може бути реалізована на різних рівнях, включаючи

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

рольову авторизацію, де користувачам надаються певні ролі з відповідними правами доступу, та дискреційний контроль доступу, де власник ресурсу вирішує, хто має до нього доступ. У корпоративних середовищах часто застосовується мандатний контроль доступу, де політики безпеки визначають, хто може отримати доступ до інформації залежно від рівня секретності та інших критеріїв. Це забезпечує додатковий рівень захисту і дозволяє гнучко налаштовувати доступ до ресурсів.

Ідентифікація є початковим етапом процесу управління доступом і передбачає визначення особи користувача перед автентифікацією. Користувач пред'являє унікальну інформацію, таку як ім'я користувача або ідентифікаційний номер, яка дозволяє системі розпізнати його як конкретну особу. Ідентифікація є критично важливим етапом, оскільки від неї залежить правильність подальшої автентифікації та авторизації. Без надійної ідентифікації система не зможе точно визначити, хто саме намагається отримати доступ до ресурсів, що може призвести до серйозних порушень безпеки.

У сучасних умовах індустрії питання автентифікації, авторизації та ідентифікації набувають нового значення у зв'язку з розвитком хмарних технологій та мобільних пристроїв. Хмарні сервіси, такі як Amazon Web Services (AWS), Microsoft Azure і Google Cloud Platform, пропонують зручні і масштабовані рішення, але вони також вимагають надійних механізмів управління доступом. Забезпечення безпеки в хмарних середовищах потребує особливого підходу, оскільки дані зберігаються на віддалених серверах, доступ до яких може бути здійснений з будь-якої точки світу. Це вимагає використання передових методів автентифікації та авторизації, таких як OpenID Connect та SAML, які забезпечують безпечний обмін інформацією між користувачами і сервісами.

Мобільні пристрої також додають нових викликів у сфері автентифікації та авторизації. Зростання популярності мобільних додатків вимагає розробки методів, які забезпечують безпечний доступ до даних з мобільних пристроїв. Використання біометричних методів, таких як відбитки пальців та розпізнавання обличчя, стало

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

стандартом для багатьох сучасних смартфонів. Однак, це також вимагає додаткових заходів для захисту біометричних даних від компрометації.

Конфіденційність даних є ще одним важливим аспектом сучасних реалій індустрії. Використання біометричних методів автентифікації та зберігання особистих даних вимагає високого рівня захисту та дотримання нормативних вимог, таких як Загальний регламент захисту даних (GDPR) в Європейському Союзі. Забезпечення конфіденційності даних потребує комплексного підходу, включаючи шифрування даних, регулярні аудити безпеки та навчання користувачів щодо безпечного поводження з інформацією.

Автентифікація, авторизація та ідентифікація є ключовими складовими забезпечення безпеки в сучасних інформаційних системах. Вони забезпечують надійний контроль доступу до ресурсів, допомагають захистити конфіденційну інформацію та знизити ризики несанкціонованого доступу. Використання сучасних технологій та підходів до управління доступом дозволяє підвищити рівень безпеки та зручності для користувачів, знижуючи ризики несанкціонованого доступу до ресурсів та даних. Їх правильна реалізація та інтеграція в інформаційні системи є ключем до забезпечення надійного захисту даних та безперебійного функціонування бізнесу і урядових установ. Сучасні виклики вимагають постійного вдосконалення методів та технологій, що дозволяє зберігати високий рівень безпеки в умовах швидкого розвитку інформаційних технологій та зростання кіберзагроз.

1.2 Недоліки традиційної системи автентифікації та переваги Single Sign-on

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

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.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

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 призводить до низки 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 того ж, 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 користувач 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 зменшується кількість запитів на відновлення паролів, оскільки користувачам легше запам'ятати один надійний пароль замість десятків різних.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Адміністрування та підтримка

Адміністрування та підтримка користувачів є важливими аспектами управління інформаційними системами, і вони суттєво відрізняються у традиційній системі автентифікації та у SSO-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 IT-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 буде 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

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

роботу адміністратор створює один обліковий запис на сервері ідентичності (identity server), 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 запису 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.

1.3 Протоколи для реалізації SSO

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 стандарт, який базується на XML 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

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачів. SAML використовується передусім у випадках, коли потрібна міждоменна автентифікація, наприклад, між постачальниками послуг (Service Providers, SP) та постачальниками ідентичності (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.

1.3.1 Протокол SAML

Протокол SAML є складним та багатокomпонентним 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.

Специфікація SAML [16, 11] визначає три різні види тверджень, які може створити органом SAML. Усі вони пов'язані з певним суб'єктом. Специфікація визначає наступні види тверджень.

- Аутентифікація: суб'єкт твердження був аутентифікований певним способом у певний час.
- Атрибут: суб'єкт твердження пов'язаний із наданими атрибутами.
- Рішення про авторизацію: запит на надання суб'єкту твердження доступу до вказаного ресурсу був задоволений або відхилений

Assertions (Підтвердження) - це пакет інформації, який містить нуль або більше тверджень, зроблених органом SAML. Підтвердження у SAML є основним носієм інформації, обмін якою відбувається між IdP та SP. Підтвердження містять дані про автентифікацію користувача, його атрибути та рішення щодо авторизації. Підтвердження у SAML зазвичай структуровані у вигляді XML-документів. Розглянемо детально три основні типи підтверджень у SAML: Authentication Assertions, Attribute Assertions та Authorization Decision Assertions.

Authentication Assertions (Підтвердження автентифікації) містять інформацію про автентифікацію суб'єкта (користувача) у певний момент часу за

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

допомогою певного методу автентифікації. Це підтвердження 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 підтвердження також може містити інформацію про місцезнаходження суб'єкта під час автентифікації, наприклад, IP-адресу (SubjectLocality).

Attribute Assertions (Атрибутивні підтвердження) 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 підтвердження дозволяють передавати додаткові дані про користувача, які можуть бути використані для надання персоналізованих послуг або для прийняття рішень щодо доступу до ресурсів.

Ці підтвердження 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 підтвердження забезпечують гнучкість у передачі детальної інформації про користувача, що є корисним у складних системах з багатьма різномірними ресурсами.

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

Підтвердження 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), заборобою (Deny) або невизначеністю

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

(Indeterminate). Це підтвердження також може містити додаткову інформацію (Evidence), що підтверджує підстави для прийняття рішення.

Отже, 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) 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 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 (SLO) «забезпечує протокол обміну повідомленнями, за допомогою якого всі сеанси, надані певним сеансовим повноваженням, завершуються майже одночасно» [16, 59]. 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

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

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 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 наступних трьох 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.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Bindings

Протокол SAML використовує механізми прив'язки (Binding), щоб інтегрувати свої протоколи з різними транспортними механізмами, такими як HTTP та SOAP. Ці прив'язки забезпечують стандартизований спосіб передачі SAML-повідомлень між IdP та SP. Основними видами зв'язувань у SAML є HTTP Redirect Binding, HTTP POST Binding та SOAP Binding.

HTTP Redirect Binding використовується для передачі SAML-повідомлень через URL-параметри у HTTP-запитах методом GET. Цей метод прив'язки 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 форматі 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 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 використовується для передачі SAML-повідомлень через протокол SOAP, що забезпечує надійну та захищену передачу даних між

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

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.

1.3.2 Протоколи OAuth 2.0 та OIDC

OAuth 2.0

Протокол OAuth 2.0 є авторизаційним фреймворком, який дозволяє третім сторонам отримувати обмежений доступ до HTTP-сервісів, діючи від імені користувача. Основна мета OAuth 2.0 полягає в розділенні ролей клієнта та власника ресурсу, що дозволяє уникнути необхідності передачі облікових даних користувача третім сторонам.

У протоколі OAuth 2.0 визначено чотири основні ролі.

1. **Resource Owner** - це «суб'єкт, здатний надати доступ до захищеного ресурсу. Коли власником ресурсу є людина, її називають кінцевим користувачем» [12, 6]. Це може бути користувач веб-додатка, який надає дозвіл на доступ до своїх даних, таких як фотографії, контакти або інша особиста інформація. Власник ресурсу є основним суб'єктом, який може делегувати обмежені права доступу до своїх ресурсів іншим додаткам або сервісам.

2. **Resource Server** - це «сервер, на якому розміщені захищені ресурси, здатний приймати та відповідати на запити до захищених ресурсів за допомогою токенів доступу» [12, 6]. Він відповідає за перевірку дійсності токенів доступу, які надаються клієнтами, і забезпечує доступ до ресурсів у відповідності до отриманих дозволів. Наприклад, якщо клієнт намагається отримати доступ до фотографій

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

користувача, сервер ресурсу перевіряє токен доступу, щоб переконатися, що він дійсний і має відповідні права доступу.

3. Клієнт (Client) - це одаток, що «робить запити до захищеного ресурсу від імені власника ресурсу та з його дозволу» [12, 6] . Це може бути веб-додаток, мобільний додаток або будь-яка інша програма, яка виконує дії від імені користувача. Клієнт спочатку взаємодіє з сервером авторизації, щоб отримати дозвіл на доступ, а потім використовує отриманий токен доступу для взаємодії з сервером ресурсу. Наприклад, мобільний додаток, який хоче отримати доступ до контактів користувача, повинен спочатку отримати токен доступу від сервера авторизації.

4. Authorization Server це сервер, що «видає токени доступу клієнту після успішної аутентифікації власника ресурсу та отримання авторизації» [12, 6].. Він взаємодіє з власником ресурсу, щоб отримати дозвіл на видачу токена доступу клієнту. Сервер авторизації аутентифікує клієнта та власника ресурсу, а потім видає токен доступу, який клієнт може використовувати для доступу до ресурсів на сервері ресурсу. Сервер авторизації також може видавати токени оновлення, які дозволяють клієнтам отримувати нові токени доступу після закінчення строку дії попередніх токенів.

OAuth 2.0 Grants

OAuth 2.0 Authorization Framework описує декілька грантів (grant types), які дозволяють різним типам клієнтів отримувати токени доступу для доступу до захищених ресурсів.

1. Authorization Code Grant - це «запис, що представляє дозвіл власника ресурсу (на доступ до його захищених ресурсів), який використовується клієнтом для отримання токена доступу» [12, 8] і є найпоширенішим і найбезпечнішим методом отримання токенів доступу. Цей метод передбачає, що клієнт (наприклад, веб-додаток) спочатку переспрямовує користувача на сервер авторизації, де користувач автентифікується і підтверджує дозвіл на доступ до ресурсів. Після успішної аутентифікації та авторизації сервер видає авторизаційний код, який клієнт отримує через переспрямування користувача назад до клієнта. Потім клієнт

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

використовує цей код для запиту токена доступу від сервера авторизації. Цей процес зменшує ризик викрадення токена, оскільки сам токен не передається через браузер.

2. **Implicit Grant** - це «спрощений потік коду авторизації, оптимізований для клієнтів, реалізований у браузері за допомогою мови сценаріїв, наприклад, JavaScript» [12, 8]. Цей грант схожий на Authorization Code Grant, але спрощує процес, оскільки токен доступу видається безпосередньо в результаті авторизаційного запиту, без потреби в обміні на авторизаційний код. Цей метод часто використовується для клієнтів, які є веб-додатками або односторінковими додатками (SPA), де зберігання секретів клієнта небезпечно. Однак цей підхід менш безпечний, оскільки токен доступу передається через браузер і може бути викрадений.

3. **Resource Owner Password Credentials Grant**. «Облікові дані власника ресурсу (тобто ім'я користувача та пароль) можуть бути використані безпосередньо як дозвіл на отримання токена доступу» [12, 9]. Цей метод передбачає, що користувач передає свої облікові дані клієнту, який потім відправляє їх на сервер авторизації для отримання токена доступу. Хоча цей підхід може бути корисним у випадках, коли клієнт і сервер авторизації належать одній організації або для додатків, які довіряють клієнту, він менш безпечний і не рекомендований для загального використання, оскільки вимагає передачі облікових даних через мережу.

4. **Client Credentials Grant**. «Облікові дані клієнта (або інші форми автентифікації клієнта) можуть бути використані для надання повноважень, коли сфера повноважень обмежена захищеними ресурсами, які контролюються клієнтом, або захищеними ресурсами, попередньо узгодженими з сервером авторизації» [12, 9]. У цьому випадку клієнт запитує токен доступу, використовуючи свої власні облікові дані (ідентифікатор клієнта і секрет клієнта). Цей підхід часто використовується для сервер-серверних інтеграцій, де клієнт потребує доступу до ресурсів, які не залежать від конкретного користувача.

Access Token

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

«Access token - це облікові дані, які використовуються для доступу до захищених ресурсів» [12, 10]. Є ключовим елементом в OAuth 2.0, який надає клієнту право доступу до захищених ресурсів на сервері ресурсів. Ці токени можуть мати різні формати, включаючи JWT, і містять інформацію про права доступу, термін дії та інші атрибути. Сервер ресурсів перевіряє дійсність токена і, якщо він є дійсним, надає доступ до запитуваних ресурсів.

Безпека в OAuth 2.0

OAuth 2.0 забезпечує високий рівень безпеки, оскільки Access Token можуть бути підписані та зашифровані, що захищає їх від підробки та несанкціонованого доступу. Протокол також рекомендує використовувати захищені з'єднання HTTPS для передачі токенів, щоб запобігти їх перехопленню.

Відповідно до протоколу процес авторизації відбувається наступним чином [12, 7]:

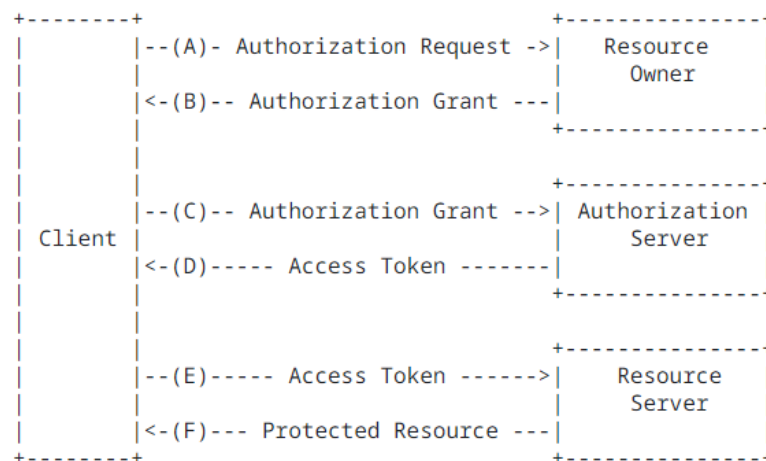


Рис. 1.3.1: Абстрактний алгоритм протоколу

(А) Запит авторизації - клієнт звертається до власника ресурсу з запитом на авторизацію. Цей запит може бути зроблений безпосередньо власнику ресурсу, як показано на схемі, або, що частіше, опосередковано через сервер авторизації, який виступає посередником.

(В) Надання авторизаційного гранту - власник ресурсу надає клієнту авторизаційний грант. Авторизаційний грант - це обліковий запис, що представляє авторизацію власника ресурсу і виражений за допомогою одного з чотирьох типів грантів, визначених у специфікації, або за допомогою розширеного типу грантів.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Тип авторизаційного гранту залежить від методу, який використовує клієнт для запиту авторизації, та типів, які підтримує сервер авторизації.

(А) Клієнт запитує дозвіл у власника ресурсу. Запит на авторизацію може бути зроблений безпосередньо до власника ресурсу (як показано на рис. 1.3.1), або краще опосередковано через сервер авторизації як посередника.

(В) Клієнт отримує дозвіл на авторизацію, який є обліковим записом, що представляє дозвіл власника ресурсу, виражений за допомогою одного з чотирьох типів дозволів, визначених у цій специфікації, або за допомогою типу розширення дозволу. Тип дозволу залежить від методу, що використовується клієнтом для запиту дозволу, та типів, що підтримуються сервером авторизації.

(С) Клієнт запитує маркер доступу, автентифікуючись на сервері авторизації та надаючи дозвіл на авторизацію.

(D) Сервер авторизації автентифікує клієнта і перевіряє дозвіл на авторизацію, і якщо він дійсний, видає маркер доступу.

(Е) Клієнт запитує захищений ресурс у сервера ресурсів ресурсу від сервера ресурсу та автентифікується, пред'являючи маркер доступу.

(F) Сервер ресурсів перевіряє маркер доступу і, якщо він дійсний, обслуговує запит.

«Протокол OpenID Connect розширює OAuth 2.0, додаючи шар автентифікації, який дозволяє клієнтам перевіряти особу кінцевого користувача, використовуючи дані, отримані від сервера авторизації.» [1]. Для досягнення цієї мети OIDC використовує кілька ключових компонентів, таких як ID Token, UserInfo Endpoint, Authorization Endpoint та Token Endpoint.

ID Token

Основне нововведення, яке OpenID Connect додає до OAuth 2.0 для забезпечення автентифікації користувачів, полягає у введенні ID Token. ID Token - це маркер безпеки, який містить твердження про автентифікацію кінцевого користувача сервером авторизації при використанні клієнта, а також потенційно інші запитувані клейми. ID Token представлений у вигляді веб-токену у форматі JSON Web Token (JWT). [8].

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей токен надається клієнту під час процесу автентифікації та містить інформацію, таку як ідентифікатор видавця токена (issuer), ідентифікатор користувача (subject), аудиторію токена (audience), час випуску та закінчення терміну дії токена. ID Token також може містити інші клейми, які запитує клієнт, що дозволяє отримувати додаткову інформацію про користувача, таку як його електронну пошту або ім'я.

JWT — це стандартний спосіб передачі даних між двома сторонами у вигляді JSON-об'єктів. Ці дані можуть бути підписані за допомогою секретного ключа (HMAC) або асиметричного ключа (RSA). Підписані JWT гарантують цілісність даних та захист від несанкціонованого доступу.

Структура JWT складається з трьох частин, розділених крапками:

1. Header - секція яка містить інформацію про тип токена та алгоритм підпису або шифрування.
2. Payload – секція з корисними даними, що містить клейми про користувача та інші дані. Це основна частина, де зберігається інформація.
3. Signature - підпис створюється за допомогою алгоритму, зазначеного в заголовку, та секретного ключа або приватного ключа.

Header і Payload кодуються у форматі Base64Url, після чого ці частини об'єднуються разом з підписом, що формує повний JWT.

JWT має кілька ключових переваг, зокрема наступні.

1. Завдяки використанню формату JSON та кодуванню Base64Url, JWT має невеликий розмір, що робить його зручним та компактним для передачі через URI та заголовки HTTP.
2. Вся необхідна інформація для перевірки автентичності та авторизації міститься всередині самого токена, що зменшує кількість запитів до серверів автентифікації та робить токен самодостатнім.
3. JWT може бути підписаний та зашифрований, що забезпечує захист від підробки та несанкціонованого доступу.

UserInfo Endpoint

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

«UserInfo Endpoint - є захищеним ресурсом OAuth 2.0, який повертає клейми про аутентифікованого кінцевого користувача» [8] 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 клейми 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 авторизації відповідає за перевірку справжності кінцевого користувача. Це здійснюється шляхом перенаправлення браузера користувача до цієї точки на сервері авторизації, де проходить процес автентифікації та надання дозволу, використовуючи параметри запиту, визначені в OAuth 2.0, а також додаткові параметри, встановлені OpenID Connect. [8]. Клієнт надсилає запит на авторизацію до цього ендпоінту, вказуючи необхідні параметри, такі як тип відповіді (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 обмінаний на токени доступу та ID Token на Token Endpoint.

Token Endpoint використовується для обміну коду авторизації на токени доступу та ID Token. Клієнт надсилає запит до цього ендпоінту, включаючи код авторизації, ідентифікатор клієнта, секрет клієнта (client secret) та URI перенаправлення. 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.

1.3.3 Порівняння протоколів для реалізації SSO

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.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

OIDC, з іншого боку, використовує токени доступу та оновлення (refresh token), 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.

Сфери використання OIDC

1. Веб-додатки. Однією з головних сфер використання OIDC є автентифікація користувачів у веб-додатках. Веб-додатки можуть швидко та безпечно перевіряти особу користувача та отримувати необхідну інформацію про нього, що робить OIDC ідеальним для інтерактивних сайтів та платформ.

2. Мобільні додатки. OIDC є надзвичайно корисним для мобільних додатків, оскільки забезпечує легку автентифікацію користувачів через мобільні інтерфейси. Використання OIDC дозволяє розробникам інтегрувати безпечну автентифікацію з різними мобільними платформами, що спрощує процес логіну та покращує досвід

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачів. Мобільні додатки можуть використовувати ID Token для перевірки автентичності та отримання профільної інформації користувача.

3. Single Page Applications (SPA) також широко використовують OIDC для автентифікації користувачів. Оскільки SPA вимагають динамічної взаємодії з сервером без перезавантаження сторінки, OIDC дозволяє легко реалізувати автентифікацію та управління сесіями. Це забезпечує безперебійний та безпечний користувацький досвід у таких додатках.

4. Взаємодія між сервісами (API). OIDC використовується для автентифікації та авторизації доступу до API. Завдяки підтримці OAuth 2.0, OIDC дозволяє розробникам легко інтегрувати захист API, надаючи токени доступу, які можуть бути використані для авторизації запитів до серверів. Це особливо корисно для мікросервісної архітектури, де взаємодія між сервісами потребує високого рівня безпеки та контролю доступу.

5. Електронна комерція (e-commerce). OIDC ідеально підходить для інтеграції в системи електронної комерції, забезпечуючи безпечну та зручну автентифікацію користувачів. Використовуючи OIDC, онлайн-магазини можуть легко впроваджувати функціонал єдиного входу (Single Sign-On), покращуючи користувацький досвід та забезпечуючи безпечний доступ до облікових записів клієнтів.

Сфери використання SAML

1. Корпоративне використання. Корпорації використовують SAML для реалізації єдиної системи входу (SSO) у своїх внутрішніх системах. Це дозволяє співробітникам отримувати доступ до різноманітних додатків та ресурсів, використовуючи один набір облікових даних. SAML забезпечує високий рівень безпеки через підписання та шифрування тверджень, що гарантує цілісність та конфіденційність переданих даних. Крім того, SAML підтримує складні політики автентифікації, що дозволяє корпоративним IT-відділам ефективно керувати доступом до ресурсів.

2. Урядові та освітні установи. SAML також широко використовується в урядових установах для забезпечення безпечної автентифікації громадян та

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

співробітників. Уряди використовують SAML для інтеграції різних державних порталів, що дозволяє громадянам отримувати доступ до різних державних послуг, використовуючи одну пару облікових даних. Це підвищує зручність та безпеку, знижуючи необхідність у створенні окремих облікових записів для кожного сервісу.

За підсумками порівняння протоколів SAML та OIDC для впровадження в онлайн-магазині, було обрано OIDC як простий і гнучкий протокол, що підтримує сучасні технології.. SAML забезпечує високу безпеку багато функціоналу, але його складність може бути зайвою для онлайн-магазину, де важливі швидкість та легкість інтеграції. Таким чином, OIDC є кращим вибором для забезпечення автентифікації та авторизації в онлайн-магазині.

Висновки до розділу 1

У першому розділі було проведено глибокий аналіз теоретичних основ автентифікації з використанням технології Single Sign-On (SSO). Було визначено, що традиційні методи автентифікації, такі як використання паролів, вже не забезпечують належного рівня безпеки через вразливість до атак. Було показано, що використання багатофакторної автентифікації та біометричних методів підвищує рівень безпеки. Також розглянуто протоколи SAML та OIDC, де OIDC було визначено як більш гнучкий та зручний для інтеграції у сучасні веб-додатки та мобільні додатки. Таким чином, SSO є перспективною технологією, яка дозволяє забезпечити зручний та безпечний доступ до ресурсів, знижуючи ризики несанкціонованого доступу та покращуючи загальний рівень безпеки системи.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2 ПРОЕКТУВАННЯ РІШЕННЯ

2.1. Формування вимог до проекту відповідно до бізнес-логіки онлайн-магазину

2.1.1 Стратегічні потреби від проекту

Основою бізнеслогіки інтернет-магазину є модель SaaS. «Програма як послуга (англ. Software as a service, SaaS) — модель поширення програм споживачам, при якій постачальник розробляє вебпрограму, розміщує її й керує нею (самостійно або через третіх осіб) з метою використання її замовниками через інтернет» [17]. Замовники платять не за володіння програмами як такими, а за їх використання. У цій моделі програмне забезпечення розміщується на серверах постачальника і керується ним, що дозволяє клієнтам використовувати його без необхідності встановлювати або підтримувати його на своїх власних комп'ютерах чи серверах.

Однією з головних переваг SaaS є те, що вона знижує витрати на ІТ-інфраструктуру та обслуговування. Це особливо актуально для e-commerce рішень, які часто потребують складних і потужних програмних рішень для управління своїми операціями. Використання такого підходу дозволить замовнику зосередитися на основному бізнесі, не відволікаючись на технічні аспекти підтримки програмного забезпечення.

Для нас SaaS має кілька важливих переваг. По-перше, це швидкий запуск. У порівнянні з традиційними моделями, на базі SaaS продукт можна розгорнути значно швидше, оскільки не вимагатиме складної інсталяції та налаштування на місці. Це особливо важливо для бізнесів, які прагнуть швидко вийти на ринок і почати продавати свої товари своїм покупцям.

По-друге, це масштабованість. Замовники можуть змінювати розмір своїх SaaS-рішень відповідно до зростання або скорочення бізнесу. SaaS-платформи часто пропонують гнучкі плани підписки, які дозволяють збільшувати або зменшувати обсяг використання ресурсів без значних витрат чи зусиль. Це означає,

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

що замовники можуть легко адаптувати свої технологічні потреби до мінливих умов ринку.

По-третє, це доступ до новітніх технологій та оновлень. SaaS-постачальники регулярно оновлюють свої продукти, додаючи нові функції та виправляючи виявлені проблеми. Клієнти автоматично отримуватимуть доступ до оновлень, що дозволяє їм залишатися на передовій технологічних інновацій без додаткових витрат.. Це особливо важливо в умовах швидкоплинного розвитку електронної комерції, де конкурентна перевага може бути отримана завдяки впровадженню нових технологій.

Крім того, SaaS-рішення часто інтегруються з іншими сервісами та платформами, що дозволяє онлайн-магазинам створювати комплексні екосистеми для управління всіма аспектами бізнесу. Наприклад, інтеграція з платіжними системами, логістичними сервісами, CRM-системами та маркетинговими інструментами дозволяє автоматизувати багато процесів та підвищити ефективність роботи.

Важливим аспектом SaaS є також зручність у користуванні. Більшість SaaS-платформ розроблені з урахуванням потреб користувачів, що забезпечує інтуїтивно зрозумілий інтерфейс і легкість у використанні. Це означає, що навіть співробітники без глибоких технічних знань можуть швидко навчитися працювати з програмним забезпеченням, що знижує витрати на навчання та підтримку користувачів.

Загалом, використання SaaS для онлайн-магазину є стратегічним рішенням, що дозволяє знизити витрати, підвищити гнучкість, забезпечити безпеку та доступність, а також швидко впроваджувати нові технології. Це забезпечує конкурентоспроможність та ефективність онлайн-магазинів в умовах динамічного ринку електронної комерції.

Відповідно до обраної стратегії використання хмарних рішень, для розвитку автентифікації магазину буде використано модель **Identity as a Service (IDaaS)** — це модель розповсюдження послуг з управління ідентифікацією користувачів, яка надається через хмару. Ця модель включає в себе автентифікацію, авторизацію,

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

керування доступом та інші функції, пов'язані з ідентифікацією, які допомагають організаціям захистити свої дані та ресурси. Використання IDaaS для SaaS-рішень матиме численні переваги, особливо для замовників, які прагнуть забезпечити безпеку своїх даних та забезпечити зручний доступ для своїх користувачів.

Однією з основних причин, чому варто використовувати IDaaS для SaaS, є підвищення рівня безпеки. У сучасному цифровому світі захист даних стає критично важливим аспектом діяльності будь-якого бізнесу. Замовники, що працюють з великими обсягами особистих та фінансових даних клієнтів, повинні забезпечити високий рівень безпеки, щоб уникнути витоків даних та кіберзагроз. IDaaS-рішення за замовчуванням пропонують багато рівнів захисту, включаючи багатофакторну автентифікацію, шифрування даних та інші механізми безпеки, що знижують ризики несанкціонованого доступу.

Ще однією важливою перевагою IDaaS є централізоване управління ідентифікаціями. Це особливо корисно для наших замовників, які мають велику кількість користувачів і потребують ефективних засобів для управління доступом. Централізоване управління дозволяє адміністраторам легко створювати, змінювати та видаляти облікові записи користувачів, а також надавати або обмежувати доступ до різних ресурсів на основі ролей та привілеїв. Це значно полегшує процеси управління та забезпечує кращу контрольованість доступу.

До того ж ми можемо виграти від зниження витрат на управління ідентифікацією. Замість того, щоб інвестувати у власні системи безпеки та управління доступом, ми можемо скористатися послугами постачальника IDaaS, що забезпечить всі необхідні функції в рамках одного рішення. Це дозволяє значно знизити витрати на підтримку та розвиток ІТ-інфраструктури, одночасно отримуючи доступ до найсучасніших технологій та експертизи.

Також, важливим аспектом використання IDaaS для нас є відповідність вимогам законодавства та регуляторних норм. Багато країн та регіонів мають суворі правила щодо захисту персональних даних, такі як GDPR у Європейському Союзі. Використання IDaaS дозволяє онлайн-магазинам легше дотримуватися цих вимог, оскільки постачальники IDaaS часто інтегрують у свої рішення необхідні

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

механізми для забезпечення відповідності. Це допомагає уникнути можливих штрафів та зберегти довіру клієнтів.

Крім того, IDaaS забезпечує гнучкість та масштабованість, що є критично важливими для нас. Клієнти можуть швидко зростати або змінювати свої потреби. IDaaS-рішення можна легко адаптувати до змін у кількості користувачів або нових вимог до доступу, що дозволяє бізнесу швидко реагувати на нові виклики. Ця гнучкість дозволяє онлайн-магазину легко інтегрувати нові сервіси або додатки без необхідності значних змін у своїй інфраструктурі.

Важливою перевагою IDaaS є можливість інтеграції з іншими хмарними та локальними системами. Це дозволяє нашим клієнтам створювати єдину екосистему для управління ідентифікацією, яка охоплює всі їхні ресурси та сервіси.

У замовника може бути свій власний IDaaS, що забезпечує управління ідентифікацією користувачів та доступом до ресурсів. Використання власного IDaaS дозволить легко інтегрувати його в наявну інфраструктуру, знижуючи залежність від сторонніх постачальників. Це також надасть більший контроль над безпекою даних та управлінням користувачами, забезпечуючи гнучкість у налаштуванні політик доступу відповідно до потреб бізнесу. Власний IDaaS може бути адаптований до специфічних вимог компанії, покращуючи ефективність та безпеку процесів автентифікації та авторизації.

На додаток до всіх технічних переваг, використання IDaaS може також сприяти покращенню взаємодії з клієнтами. Завдяки більш зручним та безпечним методам автентифікації кінцеві клієнти замовників отримують покращений досвід користування, що може підвищити їхню лояльність та задоволеність. Це особливо важливо для нас, адже ми прагнемо забезпечити високий рівень обслуговування та створити позитивний імідж на ринку.

Таким чином, використання підходу IDaaS в онлайн-магазині є стратегічним кроком, що забезпечує безпеку, зручність, ефективність та відповідність законодавчим вимогам. Це дозволить зосередитися на основному бізнесі, знижуючи витрати та ризики, одночасно забезпечуючи найвищий рівень обслуговування для наших клієнтів.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

2.1.2 Огляд наявного функціоналу для проектування

Основна бізнес-логіка онлайн-магазину полягає в управлінні товарами, категоріями та користувачами. Користувачі з відповідними ролями можуть переглядати доступні товари, додавати їх до кошика, оформляти замовлення та здійснювати покупки. Адміністратори мають можливість створювати нових користувачів, редагувати їхні дані та видаляти облікові записи.

Управління товарами та категоріями

Товари організовані за категоріями, що дозволяє користувачам легко знаходити потрібні продукти. Адміністратори мають можливість додавати нові товари, редагувати їхні характеристики, включаючи назву, опис, ціну та зображення, а також видаляти непотрібні товари. Категорії також можуть бути створені, редаговані та видалені адміністраторами, що дозволяє підтримувати актуальність та структуру каталогу товарів.

Управління сторінками

Адміністратори можуть створювати та редагувати статичні сторінки сайту, такі як інформаційні сторінки, сторінки з умовами використання, політикою конфіденційності тощо. Це забезпечує можливість швидкої адаптації контенту під потреби користувачів та підтримки актуальної інформації на сайті.

Управління користувачами та ролями

Адміністратори мають можливість управляти обліковими записами користувачів, включаючи створення нових користувачів, редагування існуючих даних та видалення облікових записів. Користувачі можуть мати різні ролі, що визначає їх права доступу до певних функцій системи. Наприклад, адміністратори мають повний доступ до управління контентом та налаштувань системи, тоді як звичайні користувачі можуть переглядати товари та здійснювати покупки.

Оформлення замовлень та управління кошиком

Користувачі можуть додавати товари до кошика, переглядати вміст кошика, змінювати кількість товарів або видаляти їх з кошика. Після завершення покупок користувачі можуть оформити замовлення. Система зберігає інформацію про

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

замовлення, що дозволяє адміністраторам обробляти їх, оновлювати статуси замовлень та здійснювати інші необхідні операції для виконання замовлень.

Бізнес логіка цього онлайн-магазину побудована таким чином, щоб забезпечити зручність користувачів, ефективне управління контентом та товарами, а також високий рівень безпеки та інтеграції з зовнішніми сервісами. Система дозволяє адміністраторам гнучко налаштовувати та управляти магазином, а користувачам - легко знаходити та купувати товари.

2.1.3 Сформовані вимоги

1. Онлайн-магазин повинен підтримувати інтеграцію з зовнішніми IDaaS сервісами через протокол OpenID Connect. Ця інтеграція повинна дозволити користувачам автентифікуватися через будь-яких провайдерів з підтримкою протоколу OIDC, без втрати наявного функціоналу магазину.

2. Система повинна підтримувати динамічну конфігурацію IDaaS, що дозволить нашим адміністратору легко додати або замінити один IdP на інший.

3. Система автентифікації повинна забезпечувати безпечний доступ до всіх ресурсів та функціоналу онлайн-магазину, визначених у пункті 2.1.2.

4. Весь попередній функціонал онлайн-магазину повинен бути збережений.

2.2 Проектування рішення

2.2.1 Огляд технологічного стеку інтернет-магазину

Основою для нашого інтернет-магазину був обраний **ASP.NET MVC** – це одна з популярних платформ для створення веб-додатків, яка була розроблена компанією Microsoft. Вона надає можливість будувати веб-додатки з використанням шаблону проектування MVC, що сприяє розділенню логіки програми, користувацького інтерфейсу та обробки даних. Розглянемо основні можливості ASP.NET MVC детально.

Модель (Model) у MVC-додатку «представляє стан додатку та будь-яку бізнес-логіку або операції, які він повинен виконувати» [11]. Модель взаємодіє з базою даних та іншими джерелами даних, обробляючи їх і забезпечуючи доступ до

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

даних для контролера та представлення. У ASP.NET MVC модель може бути реалізована за допомогою Entity Framework, який дозволяє працювати з базами даних за допомогою об'єктів. Це забезпечує високу абстракцію і зручність роботи з даними. [11]

Контролер (Controller) – це компоненти, які «обробляють взаємодію з користувачем, працюють з моделлю і, зрештою, обирають вигляд для візуалізації». Контролер отримує вхідні дані від користувача через URL-запити, обробляє ці дані, звертається до моделі для отримання необхідних даних і передає їх у представлення для відображення. Контролери в ASP.NET MVC зазвичай реалізуються як класи, що успадковуються від базового класу Controller, і містять методи-дії (action methods), які відповідають на різні запити користувача.

Представлення (View) – «відповідають за представлення контенту через користувацький інтерфейс» [11]. Представлення в ASP.NET MVC зазвичай реалізуються за допомогою Razor-синтаксису, що дозволяє поєднувати HTML з C# кодом. Razor-вид представлення дозволяє створювати динамічні веб-сторінки, де дані з моделі можуть бути інтегровані в HTML-шаблони для відображення.

Однією з ключових переваг ASP.NET MVC є чітке розділення обов'язків між компонентами моделі, представлення та контролера. Це забезпечує високу підтримуваність і масштабованість веб-додатків. Розділення обов'язків дозволяє різним розробникам працювати над різними частинами програми незалежно один від одного, що значно прискорює процес розробки.

ASP.NET MVC підтримує безліч протоколів та стандартів для роботи з даними та взаємодії з іншими системами. Одним з найпоширеніших протоколів є HTTP (Hypertext Transfer Protocol), який використовується для передачі даних між клієнтом і сервером. ASP.NET MVC надає розширені можливості для обробки HTTP-запитів, включаючи підтримку GET, POST, PUT, DELETE та інших методів HTTP. Це дозволяє створювати RESTful веб-сервіси, які можуть взаємодіяти з іншими веб-додатками та мобільними додатками.

В якості бази даних інтернет-магазин використовує **Microsoft SQL Server (MSSQL)**. Microsoft SQL Server є потужною реляційною базою даних, яка надає

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

широкий спектр можливостей для підтримки онлайн-магазинів. Цей сервер баз даних забезпечує високу продуктивність, масштабованість та безпеку, що робить його ідеальним вибором для обробки великих обсягів даних і підтримки складних бізнес-процесів.

Однією з ключових можливостей Microsoft SQL Server є його здатність обробляти великі обсяги транзакцій в режимі реального часу. Онлайн-магазини часто стикаються з великим потоком запитів на покупку, оновлення запасів, обробку платежів і управління клієнтськими даними. SQL Server забезпечує високу продуктивність завдяки використанню індексів, кешування запитів та оптимізації виконання запитів, що дозволяє швидко обробляти тисячі запитів одночасно.

SQL Server підтримує різні типи даних і забезпечує потужні механізми для управління складними структурами даних. Це дозволяє зберігати і обробляти дані про товари, категорії, клієнтів, замовлення, оплати та інші аспекти онлайн-магазину. Завдяки підтримці реляційних і нереляційних даних, SQL Server дозволяє створювати гнучкі та ефективні схеми баз даних, що відповідають специфічним потребам нашого бізнесу.

Для забезпечення високої доступності та безперервності бізнесу, SQL Server підтримує різні механізми резервного копіювання та відновлення даних. Ви можете налаштувати автоматичне резервне копіювання даних і використовувати функції журналу транзакцій для відновлення бази даних до будь-якої точки часу в разі збою. Крім того, SQL Server підтримує реплікацію даних і кластеризацію, що дозволяє забезпечити високу доступність і відмовостійкість нашої бази даних навіть у разі виходу з ладу одного з серверів.

Ще однією важливою можливістю SQL Server є підтримка масштабованості. З ростом бізнесу і збільшенням обсягів даних, SQL Server дозволяє легко масштабувати базу даних, додаючи нові сервери або розподіляючи навантаження між наявними. Для забезпечення стабільної роботи нашого онлайн-магазину навіть при значних пікових навантаженнях можна використовувати горизонтальне та вертикальне масштабування.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

SQL Server надає потужні засоби для автоматизації управління базою даних. Ви можете використовувати SQL Server Agent для планування і виконання регулярних завдань, таких як резервне копіювання, обслуговування індексів, очищення даних та інших. Це дозволяє знизити витрати на адміністрування бази даних і забезпечити стабільну і безперебійну роботу нашого онлайн-магазину.

Окрім того, SQL Server підтримує різні протоколи для доступу до даних, такі як T-SQL (Transact-SQL), який є розширенням стандартного SQL і надає додаткові можливості для управління транзакціями, процедур, тригерів та інших об'єктів бази даних. Це забезпечує гнучкість і потужність при розробці складних запитів і бізнес-логіки для нашого онлайн-магазину.

2.2.2 Проектування рішення під наявну кодову базу

Для імплементації одноразового входу SSO обрано бібліотеку Microsoft.AspNetCore.Authentication з огляду на те, що проект написаний на ASP.NET. Ця бібліотека пропонує потужний інструментарій для створення безпечних, масштабованих і зручних для користувачів систем автентифікації.

Однією з основних причин використання Microsoft.AspNetCore.Authentication для імплементації SSO є її глибока інтеграція з ASP.NET. Це дозволяє нам легко і швидко налаштувати автентифікацію без необхідності писати великий обсяг коду. Вбудовані механізми та можливості конфігурації забезпечують високу гнучкість і дозволяють адаптувати систему автентифікації під специфічні потреби проекту. Крім того, підтримка протоколів автентифікації таких як OpenID Connect і OAuth 2.0, Microsoft.AspNetCore.Authentication забезпечує за замовчуванням.

Microsoft.AspNetCore.Authentication також надає потужні можливості для управління сесіями користувачів. Це включає автоматичне оновлення токенів доступу, відстеження активних сесій і забезпечення безпечного завершення сесій. Завдяки цьому, розробники можуть бути впевнені, що їх додаток завжди буде відповідати найвищим стандартам безпеки і забезпечувати надійний захист персональних даних користувачів.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Ще одним важливим аспектом є підтримка багаторазових автентифікаційних схем. Це означає, що за допомогою Microsoft.AspNetCore.Authentication можна легко реалізувати різні методи автентифікації для різних частин додатку або для різних типів користувачів. Наприклад, для внутрішніх користувачів компанії можна використовувати корпоративний SSO на базі Active Directory, тоді як для зовнішніх користувачів - автентифікацію через соціальні мережі. Це забезпечує високу гнучкість і дозволяє створювати зручні та безпечні рішення, які відповідають потребам різних категорій користувачів.

Активна підтримка та оновлення з боку Microsoft є ще однією вагомою причиною для вибору Microsoft.AspNetCore.Authentication. Ми можемо бути впевнені, що вони використовують сучасні і надійні рішення, які регулярно оновлюються для усунення вразливостей і додавання нових функцій. Це дозволяє завжди залишатися на крок попереду потенційних загроз і забезпечувати стабільну роботу додатку.

До того ж, за допомогою конфігураційних файлів і вбудованих механізмів, ми можемо налаштувати одноразовий вхід з мінімальними зусиллями. 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 ідентифікації саме там. Це робить процес інтеграції простим і швидким.

Таким чином, використання Microsoft.AspNetCore.Authentication для імплементації одноразового входу в проєкті на базі ASP.NET надає численні переваги. Висока інтеграція з платформою, підтримка сучасних стандартів безпеки, зменшення кількості необхідного коду, потужні можливості управління сесіями, підтримка багаторазових автентифікаційних схем, висока продуктивність і масштабованість, а також активна підтримка з боку Microsoft роблять цю бібліотеку популярним вибором для розробників, які прагнуть створювати безпечні, зручні і ефективні рішення.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2.3 Проектування взаємодії між основними компонентами алгоритму

Існують два основних підходи для реалізації автентифікації через сторонні сервіси, такі як OIDC: Popur і Redirect. Popur відкриває невелике вікно для введення облікових даних, що дозволяє користувачу залишатися на тій самій сторінці. Redirect, навпаки, перенаправляє користувача на сторінку провайдера автентифікації, а після завершення процесу повертає його назад. Обидва підходи мають свої переваги та недоліки, що впливають на користувацький досвід і технічну реалізацію.

Ми оберемо підхід з переадресацією на сторінку IdP через наступні причини.

1. Переадресація дозволяє користувачам бачити, що вони вводять свої дані безпосередньо на сайті провайдера (наприклад, Facebook або Google), що підвищує довіру до вашого магазину, адже користувачі бачать, що їхні дані не передаються третім особам.

2. Попапи можуть викликати проблеми на мобільних пристроях і різних браузерях, оскільки вони можуть бути заблоковані, що призводить до плутанини у користувачів. Redirect працює стабільніше на всіх пристроях і браузерях.

3. Переадресація не потребує складної обробки станів або відслідковування різних сценаріїв відкриття попапів, що спрощує підтримку і розвиток функціоналу.

На рис. 2.2.3.2 зображено процес автентифікації та авторизації з використанням протоколу OpenID Connect за сценарієм Redirect:

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

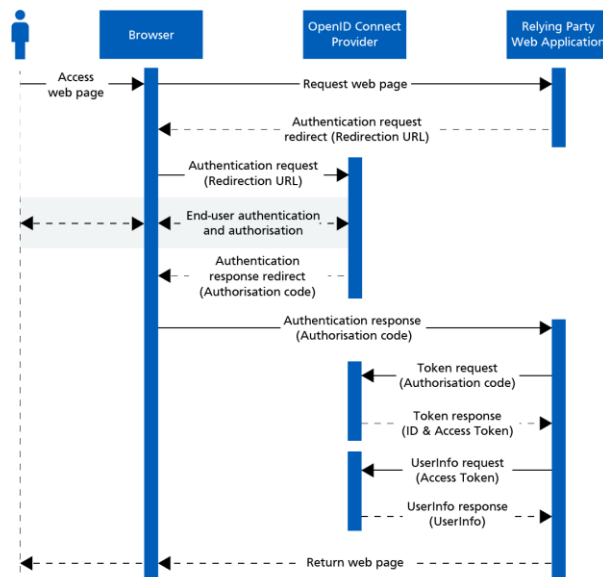


Рис. 2.2.3.1 схема автентифікації за допомогою OIDC за сценарієм Redirect [5]

Детальніше розглянемо процес, зображений на рис. 2.2.3.1.

1. Access web page - користувач за допомогою браузера звертається до веб-сторінки у Веб-додатку (Relying Party Web Application).

2. Request web page – магазин отримує запит на веб-сторінку від браузера.

3. Authentication request redirect (Redirection URL) - магазин переадресовує браузер користувача на провайдера OpenID Connect для автентифікації, відправляючи запит на автентифікацію.

4. Authentication request (Redirection URL) - браузер надсилає запит на автентифікацію до провайдера OpenID Connect, використовуючи наданий Redirection URL.

5. End-user authentication and authorization - провайдер OpenID Connect проводить автентифікацію та авторизацію користувача. Це може включати введення користувачем облікових даних або надання згоди на доступ до своїх даних.

6. Authentication response redirect (Authorization code) - після успішної автентифікації та авторизації провайдер OpenID Connect переадресовує браузер користувача назад до веб-додатку з авторизаційним кодом (Authorization code).

7. Authentication response (Authorization code) - магазин отримує авторизаційний код від браузера.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

8. Token request (Authorization code) - веб-додаток надсилає запит на отримання токенів (Token request) до провайдера OpenID Connect, використовуючи авторизаційний код.

9. Token response (ID & Access Token) - провайдер OpenID Connect відповідає веб-додатку, надсилаючи ID токен та Access токен.

10. UserInfo request (Access Token) - веб-додаток надсилає запит на отримання інформації про користувача (UserInfo request) до провайдера OpenID Connect, використовуючи Access токен.

11. UserInfo response (UserInfo) - провайдер OpenID Connect відповідає веб-додатку, надаючи інформацію про користувача (UserInfo).

12. Return web page - браузер користувача отримує доступ до веб-сторінки у веб-додатку, вже будучи автентифікованим.

2.4 Імплементация рішення

2.4.1. Динамічна реєстрація IdP

1. Спочатку додамо бібліотеку до проекту за допомогою NuGet Package Manager (рис. 2.4.1.1).

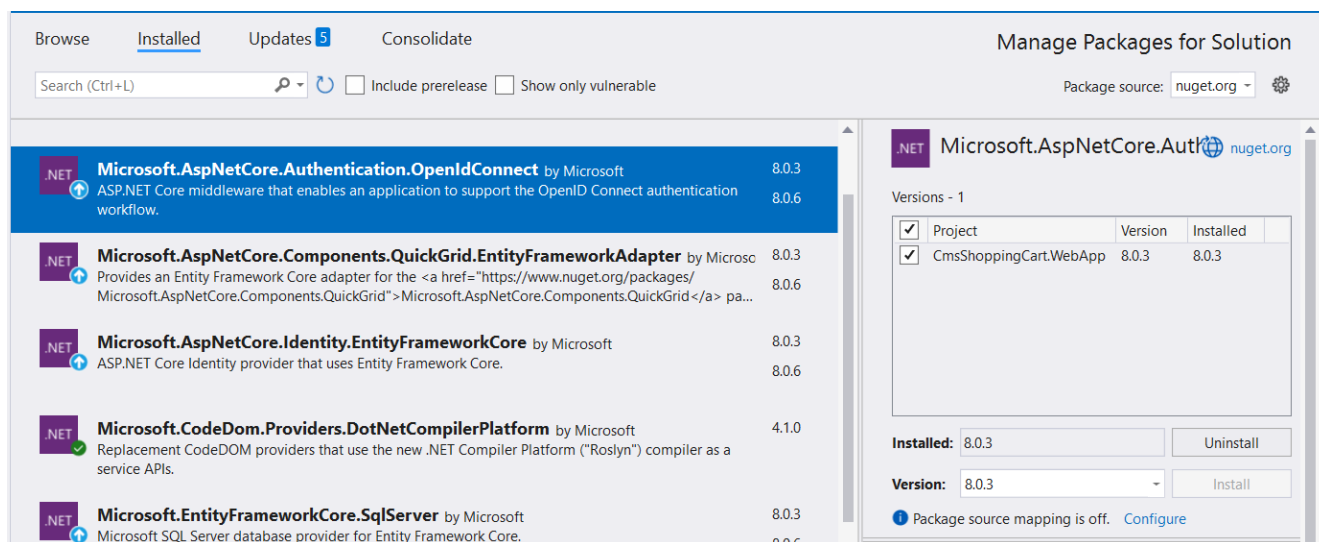


Рис. 2.4.1.1 Встановлення бібліотеки для протоколу OIDC

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

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

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<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 IdP. Нам потрібно зчитати з файлу конфігурації наступну структуру:

1. Authority – посилання на сервер авторизації, що в результаті буде автентифікувати кінцевого користувача.
2. ClientId – валідний для серверу автентифікації ідентифікатор клієнта, що налаштовується на самому сервері ідентифікації, 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 результатами авторизації користувача.
5. LoginButtonImageUrl – посилання на картинку, що буде відображена в кнопці для авторизації.
6. LoginButtonText – текст який буде відображено на кнопці автентифікації.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

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

```
"SSO": {
  "OIDCProvider1": {
    "Authority": "OIDCProvider1 authority URL",
    "ClientId": "OIDCProvider1 application client id",
    "ClientSecret": "secret",
    "CallbackPath": "/OIDCProvider1/signin-oidc",
    "LoginButtonImageUrl": "~/media/OIDCProvider1-logo.png",
    "LoginButtonText": "Login With OIDCProvider1"
  },
  "OIDCProvider2": {
    "Authority": "OIDCProvider2 authority URL",
    "ClientId": "OIDCProvider2 application client id",
    "ClientSecret": "secret",
    "CallbackPath": "/OIDCProvider2/signin-oidc",
    "LoginButtonImageUrl": "~/media/OIDCProvider2-logo.png",
    "LoginButtonText": "Login With OIDCProvider2"
  }
}
```

Додамо клас, який буде використовувється 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 нашого 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`.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

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.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

- Встановлюється тип відповіді як `Code`.
- Визначається схема входу. Саме схема `ApplicationScheme` використовується для авторизації в інтернет-магазині, і на ній зав'язана наявна логіка.
- Визначається `account_role` як клейм для ролі в токени.
- Налаштовуються подія для обробки виходу з системи, Це потрібно, щоб перед тим, як перенаправити користувача до IdP, вказати, на який шлях в інтернет-магазині, потрібно повернути користувача, 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.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

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 згенерувати ту кнопку, на яку користувач натисне для автентифікації. Для цього використано наступний клас:

```
public class IdentityProviderViewModel
{
    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<OIDCIdentityProviderViewModel>();

    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 OIDCIdentityProviderViewModel()
        {
            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.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

У методі створюється список об'єктів типу `OIDCIdentityProviderViewModel`. 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 `OIDCIdentityProviderViewModel` з відповідними властивостями: схема автентифікації, 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. Під час підтвердження 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">
```

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <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">
                    
                    <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 оброблює 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.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

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 API-запит - [POST]
`https://{domain}/account/loginWithSSO`

2.4.3. Вихід з IdP

Для виходу з IdP модифікуємо вже ~~наявний 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 внутрішню сесію.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

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 захист контролеру:

```
[Authorize(Roles = "Admin")]
[Area("Admin")]
public class ShopController : Controller
{ }
```

Так буде виглядати захист окремого методу контролеру:

```
[Area("Shop")]
public class ShopController1 : Controller
{
    [Authorize(Roles = "Customer")]
    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>
}
```

Висновки до розділу 2

Другий розділ був присвячений проектуванню рішення для автентифікації в онлайн-магазині на основі протоколу OpenID Connect. Було визначено стратегічні та функціональні вимоги до проекту, а також розглянуто бізнес-логіку онлайн-магазину. Основною перевагою використання SaaS та IDaaS моделей було

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

визначено зниження витрат на ІТ-інфраструктуру та підвищення рівня безпеки завдяки централізованому управлінню ідентифікацією користувачів. Проектування рішення включало інтеграцію з зовнішніми сервісами ідентифікації, підтримку мультифакторної автентифікації та диференційованого доступу до функцій і даних онлайн-магазину. В результаті було створено архітектуру системи, яка дозволяє ефективно керувати доступом та забезпечувати високий рівень безпеки користувачів.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3 ТЕСТУВАННЯ ПРОЕКТУ

3.1 Тестування роботоздатності інтернет-магазину

Для тестування інфраструктури автентифікації онлайн-магазину було використано сервіси Okta та Auth0. Налаштування цих сервісів можна замінити будь-якими іншими IdP. Це робиться для того щоб перевірити правильність роботи нашої розробки з протоколом OpenID Connect та переконатися, що ми дійсно можемо динамічно налаштувати будь-який IdP.

Спочатку нам потрібно сконфігурувати всі IdP в конфігураційному файлі appsettings.json. Для тестування налаштуємо провайдери Okta та Auth0. Детальні налаштування провайдерів наведено в пунктах 3.2 та 3.3. Відповідно, нас цікавить поле “SSO”, в якому потрібно на кожний IdP додати наступні налаштування (рис. 3.1.1)

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "CmsShoppingCartContext": "Data Source=DESKTOP-SOMLU0M;Initial Catalog=CmsShoppingCartContext;Integrated Security=True;Connect Timeout=30;Encrypt=True;Trust Server Certificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"
  },
  "SSO": {
    "Auth0": {
      "Authority": "https://dev-2ax1yy8k40jn3rc4.us.auth0.com",
      "ClientId": "yJ8KAQC6DluyVRfH1ZnmrDM26j3fOKGg",
      "ClientSecret": "PmkPU7SE3Mc-Fik800-1Yqr02uUqwithtErcdHz5gKacF2b0e0IV8Vj58c%GTy",
      "CallbackPath": "/Auth0/signin-oidc",
      "ImageUrl": "~/media/sso-icons/auth0-logo.png",
      "LoginButtonText": "Login with Auth0"
    },
    "Okta": {
      "Authority": "https://dev-23347671.okta.com/oauth2/default",
      "ClientId": "0oagm239w2gkM6hCH5d7",
      "ClientSecret": "WmpoEmqYsv9-UwRjaobsq-HC3meEwrgBOTCnd4mvB%NnHvSR1AZZ20nB3RJrjx",
      "CallbackPath": "/okta/signin-oidc",
      "ImageUrl": "~/media/sso-icons/okta-logo.png",
      "LoginButtonText": "Login with Okta"
    }
  }
}
```

Рис. 3.1.1 Конфігурація двох IdP в конфігураційному файлі appsettings.json

Після розробки технології нам потрібно перевірити, чи немає помилок виконання при запуску додатку, тому переходимо в теку з проектом, відкриваємо в цій теці консоль та прописуємо команду «dotnet build» (рис. 3.1.2).

```
C:\Study\Graduate Work\CmsShoppingCart\CmsShoppingCart>dotnet build
MSBuild version 17.8.3195e7f5a3 for .NET
Determining projects to restore...
C:\Study\Graduate Work\CmsShoppingCart\CmsShoppingCart.WebApp.csproj : warning NU1701: Package 'Microsoft.CodeDom.Providers.DotNetCompilerPlatform 4.1.0' was restored using '.NETFramework,Version=v4.6.1, .NETFramework,Version=v4.6.2, .NETFramework,Version=v4.7, .NETFramework,Version=v4.7.1, .NETFramework,Version=v4.7.2, .NETFramework,Version=v4.8, .NETFramework,Version=v4.8.1' instead of the project target framework 'net8.0'. This package may not be fully compatible with your project.
All projects are up-to-date for restore.
C:\Study\Graduate Work\CmsShoppingCart\CmsShoppingCart.WebApp.csproj : warning NU1701: Package 'Microsoft.CodeDom.Providers.DotNetCompilerPlatform 4.1.0' was restored using '.NETFramework,Version=v4.6.1, .NETFramework,Version=v4.6.2, .NETFramework,Version=v4.7, .NETFramework,Version=v4.7.1, .NETFramework,Version=v4.7.2, .NETFramework,Version=v4.8, .NETFramework,Version=v4.8.1' instead of the project target framework 'net8.0'. This package may not be fully compatible with your project.
CmsShoppingCart.WebApp -> C:\Study\Graduate Work\CmsShoppingCart\CmsShoppingCart\bin\Debug\net8.0\CmsShoppingCart.WebApp.dll
Build succeeded.
```

Рис. 3.1.2 Успішна збірка проекту

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Далі, щоб підняти застосунок, потрібно прописати команду «dotnet run». Можемо бачити, що застосунок успішно запустився, та адресу, на якій знаходиться наш онлайн-магазин (рис. 3.1.3).

```

C:\Windows\System32\cmd.exe - dotnet run
C:\Study\Graduate Work\CmsShoppingCart\CmsShoppingCart>dotnet run
Building...
C:\Study\Graduate Work\CmsShoppingCart\CmsShoppingCart\CmsShoppingCart.WebApp.csproj : warning NU1701: Package 'Microsoft.CodeDom.Providers.DotNetCompilerPlatform 4.1.0' was restored using '.NETFramework,Version=v4.6.1, .NETFramework,Version=v4.6.2, .NETFramework,Version=v4.7, .NETFramework,Version=v4.7.1, .NETFramework,Version=v4.7.2, .NETFramework,Version=v4.8, .NETFramework,Version=v4.8.1' instead of the project target framework 'net8.0'. This package may not be fully compatible with your project.
C:\Study\Graduate Work\CmsShoppingCart\CmsShoppingCart\CmsShoppingCart.WebApp.csproj : warning NU1701: Package 'Microsoft.CodeDom.Providers.DotNetCompilerPlatform 4.1.0' was restored using '.NETFramework,Version=v4.6.1, .NETFramework,Version=v4.6.2, .NETFramework,Version=v4.7, .NETFramework,Version=v4.7.1, .NETFramework,Version=v4.7.2, .NETFramework,Version=v4.8, .NETFramework,Version=v4.8.1' instead of the project target framework 'net8.0'. This package may not be fully compatible with your project.
Info: Microsoft.EntityFrameworkCore.Database.Command[20101]
      Executed DbCommand (15ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
      SELECT CASE
        WHEN EXISTS (
          SELECT 1
          FROM [Pages] AS [p]) THEN CAST(1 AS bit)
        ELSE CAST(0 AS bit)
      END
Info: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[63]
      User profile is available. Using 'C:\Users\Vladyslav\AppData\Local\ASP.NET\DataProtection-Keys' as key repository and Windows DPAPI to encrypt keys at rest.
Info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:5001
Info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
Info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
Info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
Info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Study\Graduate Work\CmsShoppingCart\CmsShoppingCart
  
```

Рис.. 3.1.3 Успішний запуск з консолі

Тепер переходимо на головну сторінку, щоб упевнитися, що все добре (рис. 3.1.4).

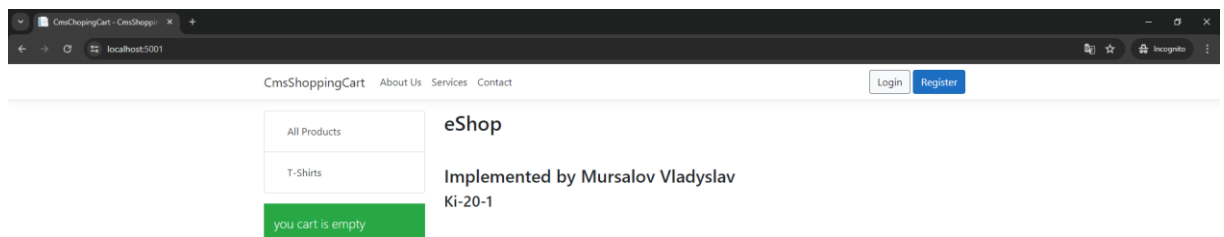


Рис. 3.1.4.Головна сторінка магазину

Тепер перейдемо на сторінку авторизації, щоб упевнитися, чи додалися посилання на авторизацію на наші два провайдери (рис. 3.1.5).

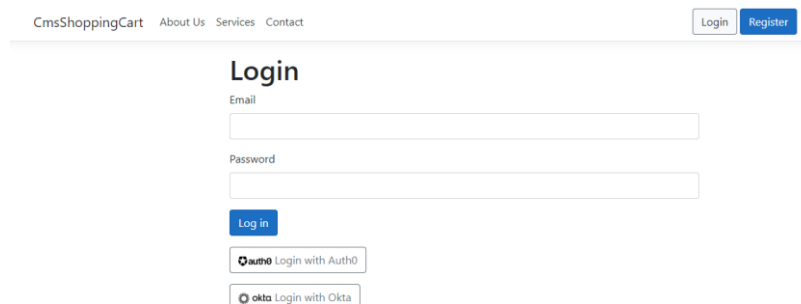


Рис. 3.1.5 Сторінка авторизації

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

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2. Налаштування Okta як серверу автентифікації

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 Okta (рис. 3.2.1).

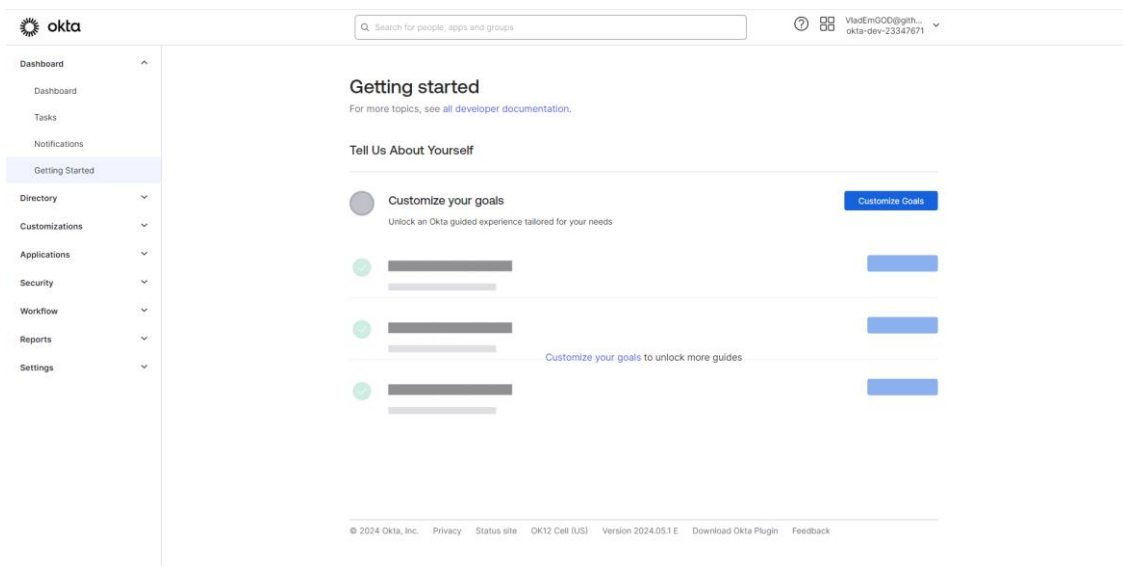


Рис. 3.2.1 Директорія Okta

Тепер нам потрібно створити інтеграцію для нашого інтернет-магазину. Необхідно перейти у панелі керування до вкладки "Applications". Тут ми можемо бачити список всіх застосунків. Натискаємо кнопку "Create App Integration", і Okta запропонує вибрати тип застосунку. Вибираємо протокол Open Id Connect. Та вибираємо шаблон "Web application" (рис. 3.2.2).

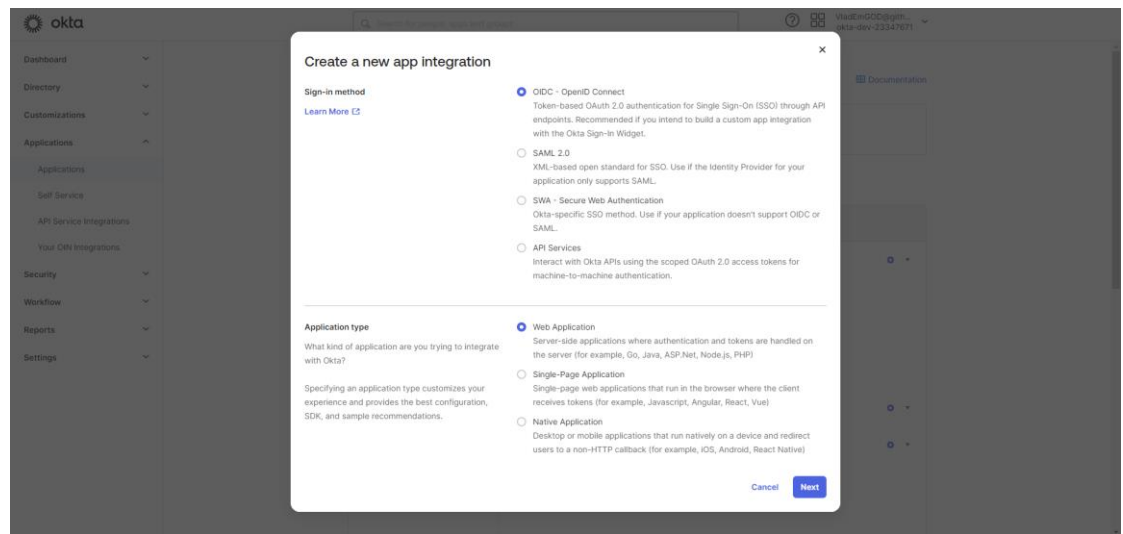


Рис. 3.2.2 Створення нової інтеграції для магазину в директорії

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Після створення застосунку можемо бачити потрібні нам згенеровані client-id та client secret (рис. 3.2.3).

The screenshot shows the 'Okta Degree App' configuration page. At the top, there's a 'Back to Applications' link. Below it, the app is marked as 'Active' with a dropdown menu and a 'View Logs' button. The main navigation bar includes 'General', 'Sign On', 'Assignments', 'Okta API Scopes', and 'Application Rate Limits'. The 'General' tab is selected, showing 'Client Credentials' and 'Client Secrets'. Under 'Client Credentials', the 'Client ID' is '0oagm239w2gKW6hCH5d7'. The 'Client authentication' section has 'Client secret' selected. The 'Proof Key for Code Exchange (PKCE)' section has 'Require PKCE as additional verification' selected. The 'CLIENT SECRETS' section shows a table with one secret generated on 'Apr 22, 2024' with an 'Active' status.

Рис. 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 (рис. 3.2.4). 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.

The screenshot shows the 'LOGIN' configuration page. It has two main sections: 'Sign-in redirect URIs' and 'Sign-out redirect URIs'. The 'Sign-in redirect URIs' section has a checkbox for 'Allow wildcard * in login URI redirect.' and a text input field containing 'https://cmsshoppingcart.mursalov.net/Okta/signin-oidc'. The 'Sign-out redirect URIs' section has a text input field containing 'http://cmsshoppingcart.mursalov.net/signout-callback-oidc'. There are also fields for 'Login initiated by' (set to 'App Only') and 'Initiate login URI'.

Рис. 3.2.4. Налаштування Sign-in та Sing-Out Redirect URL

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Тепер додамо першого користувача. Для додавання користувача в Okta потрібно увійти до панелі керування Okta та перейти до вкладки "People". Натиснути кнопку "Add Person" та заповнити наступну форму, де потрібно вказати необхідні дані (рис. 3.2.5).

Рис. 3.2.5. Створення користувача в провайдері

Далі потрібно додати нового користувача до цієї інтеграції, щоб він 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 (рис. 3.2.6).

Рис.. 3.2.6. Прив'язка користувача до відповідного додатку

Тепер потрібно додати користувачу роль в інтернет-магазині. Okta дозволяє надати користувачу роль в адмін директорії але ця роль не передається в Id токен.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Але рішення є, в Okta можна сконфігурувати свої клейми. Потрібно перейти по шляху Directory > Profile Editor (рис. 3.2.7).

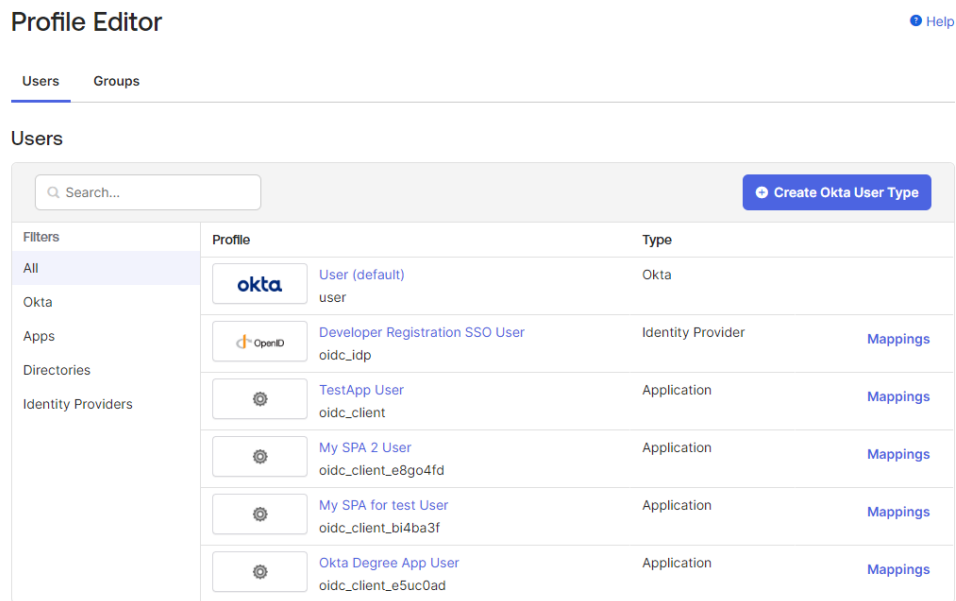


Рис.. 3.2.7. Створення користувача в провайдері

Тут відображені профілі користувачів для всіх інтеграцій, створених у директорії. Спочатку необхідно відредагувати профіль користувача за замовчуванням. Цей профіль є базою для всіх інтеграцій, оскільки він містить найбільшу кількість даних у директорії. Потім інтеграції вибирають необхідні дані з цього профілю. Таким чином, створюємо новий клейм account_role (рис. 3.2.8).

Account role

"Regular account" or "Account manager" or "Sub-account"

Data type: string

Display name: Account role

Variable name: user.account_role

Description: "Regular account" or "Account manager" or "Sub-account"

Enum: ☒ Define enumerated list of values

Display name	Value
Admin	Admin
Editor	Editor
Customer	Customer

+ Add Another

Рис.. 3.2.8. Створення користувача в провайдері

Налаштовуємо тип даних string і для зручності заздалегідь визначаємо три ролі, після чого можемо бачити клейм в профілі користувача за замовчуванням (рис. 3.2.9).

Рис.. 3.2.9. Створення користувача в провайдері

Тепер потрібно визначити такий атрибут персонально для нашої інтеграції. Заходимо в профіль користувача інтеграції, додаємо атрибут з іменем змінної account_role, описом та читабельним іменем (рис. 3.2.10).

Рис. 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” (рис. 3.2.11).

Рис.. 3.2.11. Створення користувача в провайдері

Можемо бачити дві вкладки: перша - це сторінка з зіставлення атрибутів користувача інтеграції з атрибутом користувача за замовчуванням, друга – навпаки. Відповідно, зіставляємо 2 створених атрибути (рис. 3.2.12).



Рис. 3.2.12. Створення користувача в провайдері

На цьому налаштування можна вважати закінченим. Тепер можемо побачити структуру Id токена для цього користувача при автентифікації у цьому додатку. Треба перейти в Security > Api > Authorization servers, перейти в потрібний нам сервер та обрати опцію Token Preview. Після чого вибираємо створений додаток, тип автентифікації – Authorization code, тільки що створеного користувача та передаємо потрібні області доступу. Переконаємося, що інформація у токені правильна (рис. 3.2.13).

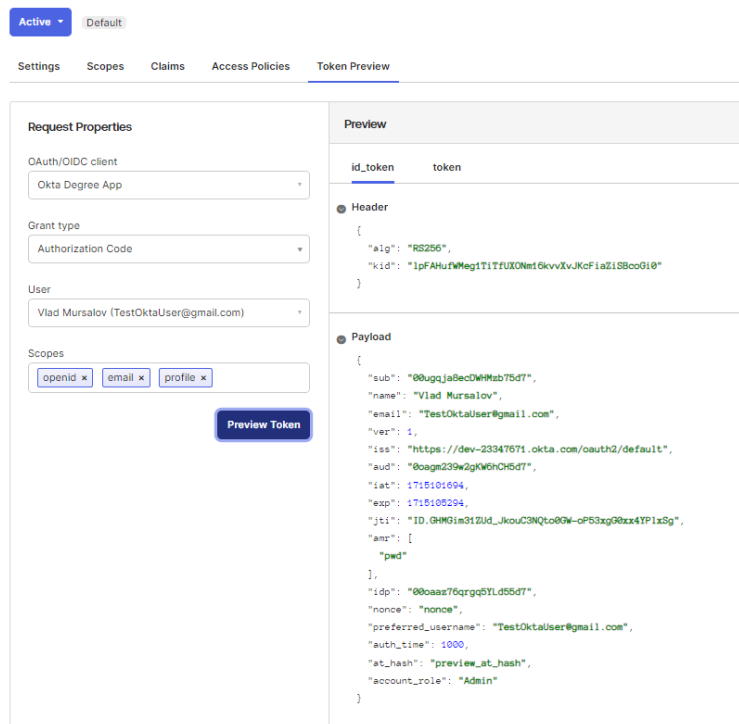


Рис. 3.2.13. Прив'язка користувача до відповідного додатку

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3. Налаштування Auth0 як серверу автентифікації

Як і в Okta, в Auth0 потрібно пройти базову автентифікацію на сервісі та створити першу директорію. Потім потрібно створити додаток з використанням шаблону “Regular Web Applications” (рис. 3.3.1).

Create application

Name *

My App

You can change the application name later in the application settings.

Choose an application type



Native

Mobile, desktop, CLI and smart device apps running natively.

e.g.: iOS, Electron, Apple TV apps



Single Page Web Applications

A JavaScript front-end app that uses an API.

e.g.: Angular, React, Vue



Regular Web Applications

Traditional web app using redirects.

e.g.: Node.js Express, ASP.NET, Java, PHP



Machine to Machine Applications

CLIs, daemons or services running on your backend.

e.g.: Shell script

Cancel

Create

Рис. 3.3.1. Створення нового додатку-інтеграції для Auth0

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Після створення додатку-інтеграції можемо бачити домен нашого серверу автентифікації, ідентифікатор цього додатку та секрет (рис. 3.3.2).

Degree project app
Regular Web Application Client ID yJ8KAQC6D1ugYRFhiZnmrDW26j3FoKGg

Quickstart **Settings** Credentials Addons Connections Organizations

Basic Information

Name *
Degree project app

Domain
dev-zax1yy8k40jn3rc4.us.auth0.com

Client ID
yJ8KAQC6D1ugYRFhiZnmrDW26j3FoKGg

Client Secret

The Client Secret is not base64 encoded.

Description
Auth 0 degree project app
A free text description of the application. Max character count is 140.

Рис. 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 (рис. 3.3.3).

Application Login URI
https://myapp.org/login

In some scenarios, Auth0 will need to redirect to your application's login page. This URI needs to point to a route in your application that should redirect to your tenant's `/authorize` endpoint. [Learn more](#)

Allowed Callback URLs
https://cmsdomain.com/Auth0/signin-oidc

After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol (`https://`) otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol `https://`. You can use [Organization URL](#) parameters in these URLs.

Allowed Logout URLs
https://cmsdomain.com/signout-callback-oidc,
https://cmsdomain.com/signout-callback-oidc

Comma-separated list of allowed logout URLs for redirecting users post-logout. You can use wildcards at the subdomain level (`*.google.com`). Query strings and hash information are not taken into account when validating these URLs. [Learn more about logout](#)

Allowed Web Origins
https://cmsdomain.com

Comma-separated list of allowed origins for use with [Cross-Origin Authentication](#), [Device Flow](#), and [web message response mode](#), in the form of `<scheme> "://" <host> [ ":" <port> ]`, such as `https://login.mydomain.com` or `http://localhost:3000`. You can use wildcards at the subdomain level (e.g.: `https://*.contoso.com`). Query string and hash information are not taken into account when validating

Рис. 3.3.3. Налаштування маршрутів для працездатності OIDC

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Створення користувачів:

Переходимо в User management > Users та тиснемо на кнопку Create new user та вводимо необхідні дані (рис. 3.3.4).

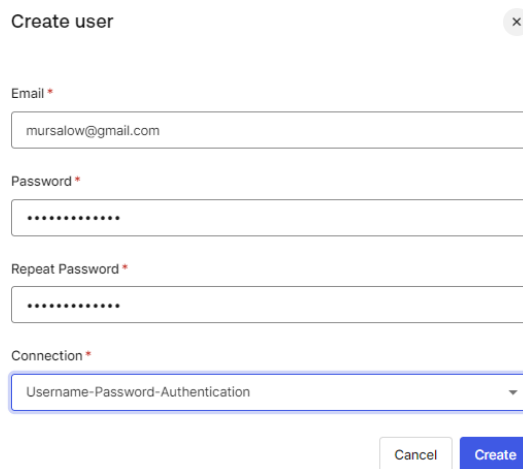


Рис. 3.3.4. Створення користувачів в Auth0

Налаштування додаткових клеймів в Id токен

Як і в Okta, в Auth0 роль користувача налаштувати можна, але ця роль розповсюджується на адміністрування в директорії і не передається в Id токен. Тому, як і в випадку з Okta, 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 клейм account_role з відповідним значенням до метаданих користувача в форматі JSON (рис. 3.3.5).

Metadata

User Metadata (user_metadata)

```
1 {  
2   "account_role": "Admin"  
3 }
```

Рис.. 3.4.1. Додавання ролі в метадату

Переходимо в Actions > Library та створюємо нову подію. Тиснемо кнопку Create action та вибираємо опцію Build from scratch. В тіло події додаємо код, поданий на рис. 3.4.2.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

← Custom Actions



Account role login action

Login / Post Login, Runtime: Node 18 (Recommended)

```
1 exports.onExecutePostLogin = async (event, api) => {  
2   const userAccountRole = event.user.user_metadata["account_role"];  
3  
4   if (!userAccountRole)  
5     api.idToken.setCustomClaim('account_role', "Customer");  
6  
7   api.idToken.setCustomClaim('account_role', userAccountRole);  
8 };
```

Рис. 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 на вкладку Custom та обираємо створену подію. Перетягуємо подію між початком та кінцем автентифікації користувача (рис. 3.4.3).

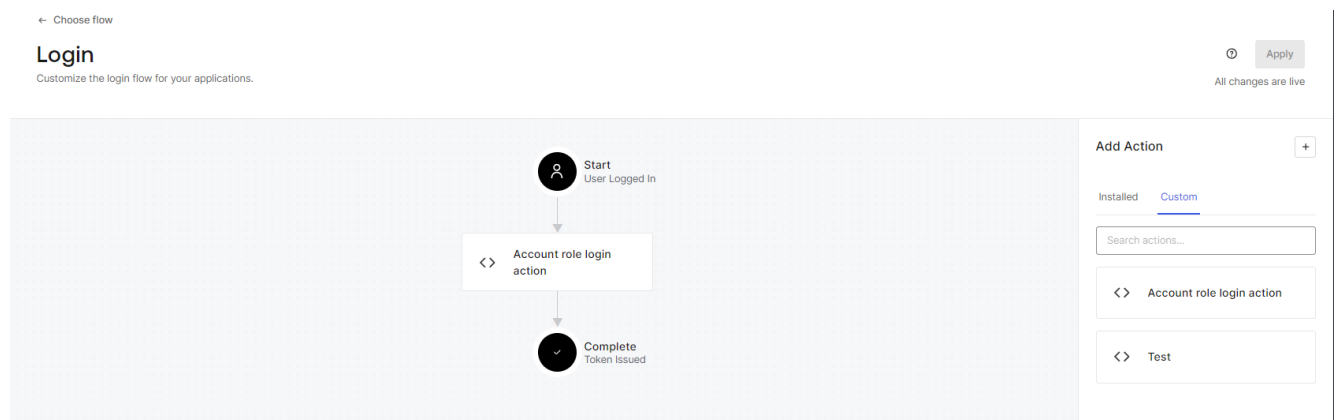


Рис. 3.4.3. Додавання події в процес автентифікації користувача

3.4 Тестування автентифікації за допомогою Okta

Спочатку перевіримо, чи можлива успішна автентифікація користувача. В інтернет-магазині переходимо на сторінку авторизації, натискаємо кнопку «Login with Okta» на сторінці авторизації, потрапляємо уже на сторінку автентифікації Okta (рис. 3.4.1).

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Подключение к 

Войдите в свою okta-dev-23347671 учетную запись, чтобы получить доступ к Okta Degree App



Войти в систему

Имя пользователя

Пароль

☐ Остаться в системе

Войти в систему

[Забыли пароль?](#)
[Справка](#)
[Sign in with GitHub](#)

Рис. 3.4.1. Сторінка авторизації Okta

Стандартно вводимо пошту та пароль на стороні Окта. Потім нас запитують дозволи на доступ до інформації профілю та пошти, дозволяємо це, після чого нас успішно автентифікувало та перенаправило на сторінку профілю (рис. 3.4.2).

[CmsShoppingCart](#)
[About Us](#)
[Services](#)
[Contact](#)

[My Account](#)
[Log out](#)
[Admin Area](#)

[Your orders](#)

Edit Account

User information from identity provider:

Authentication type : Okta
Name identifier : 00ugqja8ecDWHMzb75d7
name : Vlad Mursalov
Email : TestOktaUser@gmail.com
jti : ID.SvcSaSMTJ0xpVNIj9e8DXCDNcraHX6ZT77Pg2DdUqbg
http://schemas.microsoft.com/claims/authnmethodsreferences : pwd
http://schemas.microsoft.com/identity/claims/identityprovider : 00oaaaz76qrgq5YLd55d7
preferred_username : TestOktaUser@gmail.com
auth_time : 1715189693
account_role : Admin
AuthenticationScheme : Okta

Рис. 3.4.2. Інформація про авторизованого користувача

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

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Тепер перевіримо базу даних, спробуємо купити декілька продуктів, Переходимо на сторінку з продуктом та замовляємо його декілька разів в різних кількостях (рис. 3.4.3).

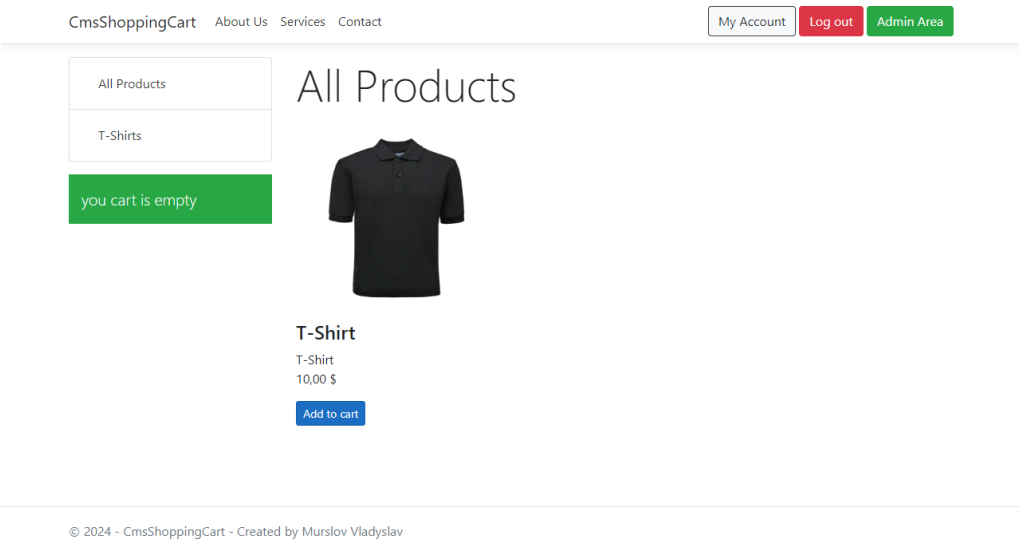


Рис.. 3.4.3. Сторінка з продуктом

Переходимо на сторінку свого профілю, та можемо бачити чотири замовлення (рис. 3.4.4).

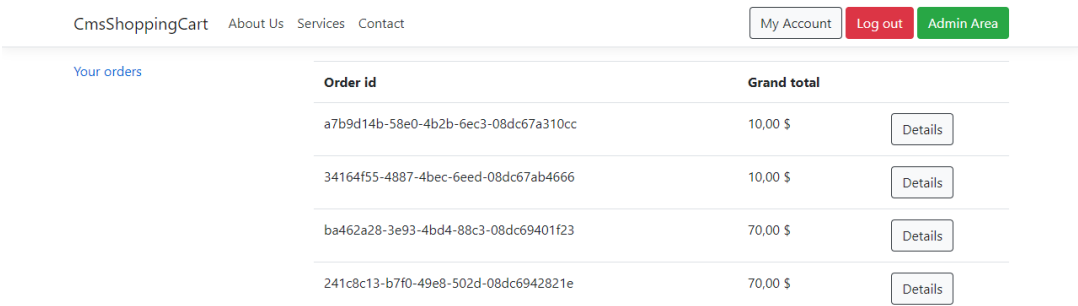


Рис.. 3.4.4. Сторінка з замовленнями

Ці ж чотири замовлення можемо побачити в базі даних, отже вона справно функціонує (рис. 3.4.5).

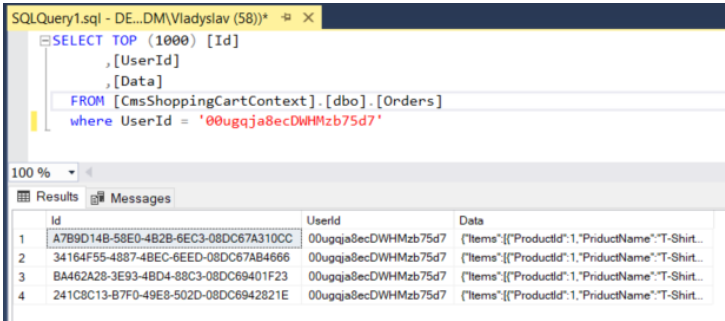


Рис.. 3.4.5. Замовлення в базі даних

3.5 Тестування автентифікації за допомогою Auth0

Так само, перевіримо чи можлива успішна автентифікація користувача. В інтернет-магазині натискаємо кнопку «Login with Auth0» та потрапляємо на сторінку автентифікації Auth0 (рис. 3.5.1).

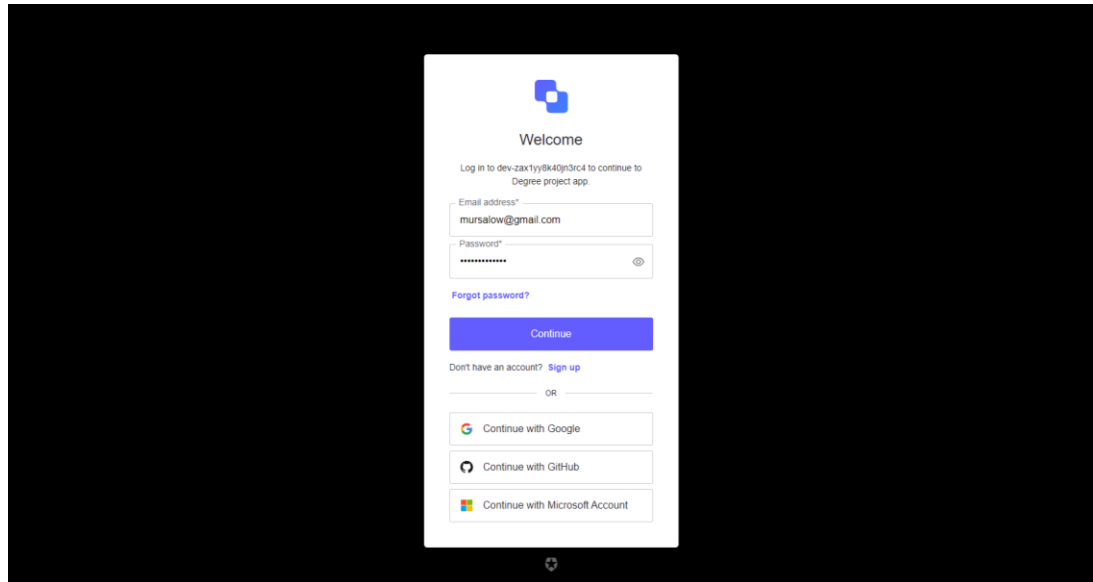


Рис. 3.5.1 Сторінка автентифікації в Auth0

Вводимо наші дані, після чого нас запитують дозволи на доступ до інформації профілю та пошти, дозволяємо це, після чого нас успішно автентифікувало та перенаправило на сторінку профілю (рис. 3.5.2).

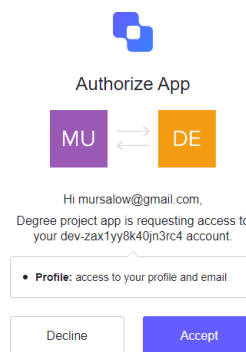


Рис.. 3.5.2. Прийняття областей доступу на доступ інформації в провайдері

Після успішної автентифікації нас Auth0 з результатами автентифікації перенаправляє на сторінку профілю користувача з актуальною інформацією (рис. 3.5.3).

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Edit Account

User information from identity provider:

Authentication type : Auth0
account_role : Admin
nickname : mursalow
name : mursalow@gmail.com
updated_at : 2024-06-02T14:16:23.964Z
Email : mursalow@gmail.com
email_verified : false
Name identifier : auth0 665c62986dce04fae15b0309
sid : woX8JhMvlgDkjg1ZyP4l2dvwQrqssSeT
AuthenticationScheme : Auth0

Рис. 3.5.3. 3.4.3. Отримана інформація користувача з Auth0

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

Спробуємо замовити два продукти. Переходимо на сторінку продукту, замовляємо декілька штук (рис. 3.5.4).

CmsShoppingCart

[About Us](#) [Services](#) [Contact](#)

[My Account](#) [Log out](#) [Admin Area](#)

All Products

T-Shirts

Items in cart: 2

Total: 20,00 \$

[View Cart](#)

[Clear Cart](#)

T-Shirts



T-Shirt

T-Shirt

10,00 \$

[Add to cart](#)

Рис.. 3.5.4. Замовлення продукту в інтернет-магазині

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Після оформлення замовлення, можемо бачити його в списку замовлень (рис. 3.5.5).

CmsShoppingCart	About Us	Services	Contact	My Account	Log out	Admin Area
-----------------	----------	----------	---------	------------	---------	------------

Your orders	
Order id	Grand total
98f66259-29a6-4fec-5a07-08dc83134cf6	30,00 \$
	Details

Рис. 3.5.5. Замовлення на сторінці замовлення

Відповідний запис в базі даних показано на рис. 3.5.6.

SQLQuery1.sql - DE...DM\Vladyslav (55)*		
<pre>SELECT TOP (1000) [Id] ,[UserId] ,[Data] FROM [CmsShoppingCartContext].[dbo].[Orders] where UserId = 'auth0 665c62986dce04fae15b0309'</pre>		
100 %		
Results	Messages	
Id	UserId	Data
1	auth0 665c62986dce04fae15b0309	{ "Items": [{"ProductId": 1, "PriductName": "T-Shirt", "Quantity": 3, "Price": 10.00, "Total": 30.00, ...

Рис. 3.5.6. Замовлення в базі даних

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

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.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до розділу 3

Третій розділ зосереджувався на тестуванні інфраструктури розробленого рішення. Було проведено налаштування та тестування серверів автентифікації Okta та Auth0, а також перевірку роботоздатності онлайн-магазину. Тестування показало, що інтеграція з сервісами ідентифікації забезпечує надійний та безпечний доступ користувачів до ресурсів магазину. Використання протоколу OIDC дозволило значно спростити процес автентифікації та підвищити зручність користувачів. Також було перевірено функціонування мультифакторної автентифікації та механізмів розмежування ролей користувачів. Результати тестування підтвердили ефективність розробленого рішення та його відповідність вимогам проекту, що забезпечує безпечний та зручний доступ до ресурсів онлайн-магазину.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В результаті проведеної роботи вдалося досягти всіх поставлених на початку завдань. Було розроблено проект підсистеми автентифікації онлайн-магазину на базі протоколу Open ID Connect та різних провайдерів ідентифікації. У ході роботи проведено детальний аналіз наявних методів автентифікації, вивчено їх переваги та недоліки. Особливу увагу приділено дослідженню протоколу OpenID Connect, його можливостям та перевагам порівняно з іншими протоколами, такими як SAML.

Розроблено архітектуру підсистеми автентифікації для онлайн-магазину, яка враховує обраних провайдерів ідентифікації, таких як Okta та Auth0. В результаті обрано оптимальне рішення для інтеграції, яке забезпечує високу безпеку, зручність використання та відповідність сучасним вимогам до автентифікації.

Проектування рішення включало формування вимог до системи відповідно до бізнес-логіки онлайн-магазину, розробку взаємодії між основними компонентами системи та імплементацію рішення в наявну кодову базу. Зокрема, реалізовано динамічну реєстрацію провайдерів ідентифікації, автентифікацію через IdP та розмежування ролей автентифікованих користувачів.

Проведене тестування інфраструктури підтвердило працездатність розробленого рішення. Налаштовано сервіси Okta та Auth0 як сервери автентифікації і проведено тестування автентифікації за допомогою цих сервісів. Результати тестування показали, що система здатна забезпечити надійну автентифікацію користувачів та ефективно управляти доступом до ресурсів онлайн-магазину.

Таким чином, всі поставлені завдання були успішно виконані. Розроблене рішення відповідає вимогам до сучасних систем автентифікації, забезпечує високий рівень безпеки, зручність використання та гнучкість у налаштуванні. Проект підсистеми автентифікації на базі протоколу Open ID Connect та різних провайдерів ідентифікації може бути рекомендований для впровадження в інших подібних проектах, що сприятиме підвищенню рівня безпеки та зручності для користувачів.

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Advantages of Software as a Service (SaaS) | IBM. IBM - United States. URL: <https://www.ibm.com/think/insights/software-as-a-service-advantages> (дата звернення: 10.06.2024).
2. App integrations | Okta. *Okta Docs*. URL: https://help.okta.com/en-us/content/topics/apps/apps_apps.htm (дата звернення: 09.06.2024).
3. Auth0 ASP.NET Core MVC SDK Quickstarts: Add Login to your ASP.NET MVC application. *Auth0 Docs*. URL: <https://auth0.com/docs/quickstart/webapp/aspnet-core/interactive> (дата звернення: 09.06.2024).
4. Auth0 ASP.NET Core MVC SDK Quickstarts: Add Login to your ASP.NET MVC application. *Auth0 Docs*. URL: <https://auth0.com/docs/quickstart/webapp/aspnet-core/interactive> (дата звернення: 09.06.2024).
5. Authorization code flow - NHS England Digital. NHS England Digital. URL: <https://digital.nhs.uk/services/care-identity-service/applications-and-services/cis2-authentication/guidance-for-developers/detailed-guidance/authorization-code-flow> (дата звернення: 09.06.2024).
6. Authorize with a specific scheme in ASP.NET Core. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en-us/aspnet/core/security/authorization/limitingidentitybyscheme?view=aspnetcore-8.0> (дата звернення: 09.06.2024).
7. Customize tokens returned from Okta with custom claims | Okta Developer. *Home | Okta Developer*. URL: <https://developer.okta.com/docs/guides/customize-tokens-returned-from-okta/main/> (дата звернення: 09.06.2024).
8. Final: OpenID Connect Core 1.0 incorporating errata set 2. *OpenID - OpenID Foundation*. URL: https://openid.net/specs/openid-connect-core-1_0.html (дата звернення: 09.06.2024).
9. Getting Started with Lock episode 3: Redirect vs Popup Mode. Auth0 - Blog. URL: <https://auth0.com/blog/getting-started-with-lock-episode-3-redirect-vs-popup-mode/> (дата звернення: 10.06.2024).

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		

10. Overview of ASP.NET Core Authentication. *Microsoft Learn: Build skills that open doors in your career.* URL: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/?view=aspnetcore-8.0> (дата звернення: 09.06.2024).

11. Overview of ASP.NET Core MVC. *Microsoft Learn: Build skills that open doors in your career.* URL: <https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0> (дата звернення: 09.06.2024).

12. RFC 6749. The OAuth 2.0 Authorization Framework. На заміну RFC 5849 ; чинний від 2012-10-01. Вид. офіц. Internet Engineering Task Force (IETF), 2012. URL: <https://www.rfc-editor.org/rfc/rfc6749> (дата звернення: 09.06.2024).

13. Sign users in to your web app using the redirect model | Okta Developer. *Home | Okta Developer.* URL: <https://developer.okta.com/docs/guides/sign-into-web-app-redirect/asp-net-core-3/main/> (дата звернення: 09.06.2024).

14. What is IDaaS? Understanding Identity as a Service and Its Applications | Okta. *Employee and Customer Identity Solutions | Okta.* URL: <https://www.okta.com/identity-101/idaas/> (дата звернення: 09.06.2024).

15. What is OpenID vs SAML? Find out the differences - Auth0. *Auth0.* URL: <https://auth0.com/intro-to-iam/saml-vs-openid-connect-oidc> (дата звернення: 09.06.2024).

16. Working Draft 07. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Чинний від 2015-09-15. Вид. офіц. OASIS, 2015. URL: <https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf> (дата звернення: 09.06.2024).

17. Учасники проєктів Вікімедіа. Програмне забезпечення як послуга – Вікіпедія. *Вікіпедія.* URL: https://uk.wikipedia.org/wiki/Програмне_забезпечення_як_послуга (дата звернення: 09.06.2024).

		Мурсалов В.Р.			ККІК.КР.Б – 123 – 17 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		