МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»

Група: 4КБ-02

Дипломний проект

здобувача освіти денної форми навчання
КБ.02.16.000.ДП

РЕМІННОГО
АНДРІЯ СЕРГІЙОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Безпека комп'ютерних систем і мереж»**

Група: **4КБ-02**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка програмного модулю аутентифікації клієнта інтернет-магазину

Проектний матеріал складається з пояснювальної записки на 79 сторінках та графічного (презентаційного) матеріалу на 15 аркушах (слайдах)

Дипломник _____ (Ремінний А.С.)

Керівник _____ (Залапін О.І.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «26» червня 2025 р. Протокол ЕК № 5

Оцінка ЕК 5 (відмінно) / 90 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Ремінному Андрію Сергійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка програмного модулю аутентифікації клієнта інтернет-магазину*

затверджена наказом по коледжу від "*14*" *листопада* 2024р. № *246*

2. Термін здачі закінченого проекту

3. Вихідні данні до проекту *1. Реалізувати безпечну систему автентифікації та авторизації користувачів. 2. Інтегрувати сучасні сервіси ідентифікації. 3. Розробити багаторівневу архітектуру з чітким розподілом відповідальностей. 4 Забезпечити використання JWT-токенів та захист від типових атак. 5 Реалізувати логування, аудит доступу та систему сесій.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Аналіз предметної області. 2. Технології та засоби розробки. 3. Проєктування архітектури системи обміну повідомленнями. 4. Робота з JWT-токенами та refresh-токенами 5. Розробка серверної частини 6. Інтеграція з сервісами електронної ідентифікації 7. Реєстрація, вхід, управління сесіями 8. Реалізація модуля автентифікації 9. Економічний розрахунок. 10. Аспекти охорони праці та техніки безпеки

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Титул; Мета роботи; Проблеми, які вирішує механізм авторизації; Опис процесу реєстрації в застосунку; Різновиди токенів для доступу до ресурсів; Загальна структура токенів та їх різновиди; Принцип авторизації в програмній реалізації системи; Підходи до використання асиметричних ключів в шифрування та при підписі; Обґрунтування необхідності оновлення токенів; Механізм оновлення JWT-токенів за допомогою refresh-токена; Принцип роботи стандарту OAuth 2.0; Демонстрація вигляду токена; Головне вікно програми; Висновки

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Залапін О.І.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 12.05.2025

Керівник *Залапін О.І.* _____ (підпис)

Завдання прийняв до виконання *Ремінний А. С.* _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Формування вступу*	15.05.2025	Виконано
2	*Дослідження предметної області*	16.05.2025	Виконано
3	*Вивчення механізмів автентифікації та електронної*	17.05.2025	Виконано
	ідентифікації (Дія.Sign, BankID)		
4	*Аналіз сучасних технологій веб-розробки, вибір стеку*	19.05.2025	Виконано
5	*Проєктування структури бази даних та моделі сесій*	22.05.2025	Виконано
6	*Розробка клієнтської частини*	26.05.2025	Виконано
7	*Розробка серверної частини*	01.06.2025	Виконано
8	*Реалізація JWT-автентифікації та керування сесіями*	06.06.2025	Виконано
9	*Інтеграція з сервісами*	10.06.2025	Виконано
10	*Тестування функціональності*	11.06.2025	Виконано
11	*Підготовка графічних матеріалів*	12.06.2025	Виконано
12	*Економічний розрахунок*	13.06.2025	Виконано
13	*Опис аспектів охорони праці та техніки безпеки*	14.06.2025	Виконано
14	*Підведення висновків*	15.06.2025	Виконано
15	*Підготовка доповіді для захисту*	16.06.2025	Виконано

Дипломник _____ (підпис)

Керівник _____ (підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КБ 02. 16 000. 00 ДП	Дипломний проект		
А4			КБ 02. 16 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КБ 02. 16 000. 00 ДП		
Розробив		Реміний А. С.		16.06.25	Розробка програмного модуля аутентифікації клієнта інтернет-магазину	Літ.	Аркуш 4 / Аркушів 79
Перевірив		Залапін О.І.		16.06.25		Н Д П	
Н. контр.		Петрашова В.І.		16.06.25		ВСП "ОТФК ОНТУ" гр.4КБ-02	
Затверд.		Кривченко Ю.В.		16.06.25			

ЗМІСТ

ВСТУП

Стрімкий розвиток електронної комерції в Україні в якої сьогодні все більше підприємств переходять до онлайн-продажів, створюючи інтернет-магазини з метою розширення своєї клієнтської бази та підвищення конкурентоспроможності. Зростання обсягів електронної торгівлі вимагає не лише зручного і функціонального інтерфейсу користувача, а й високого рівня захисту персональних даних клієнтів. Однією з ключових складових безпеки у сфері електронної комерції є надійна система аутентифікації користувачів.

Сьогоднішні користувачі очікують швидкого, зручного, але водночас безпечного доступу до особистого кабінету, історії замовлень, платіжних даних та іншої конфіденційної інформації. Недостатньо надійна аутентифікація створює ризики несанкціонованого доступу, шахрайства та втрати довіри з боку клієнтів. Саме тому розробка програмного модуля аутентифікації, що відповідає сучасним вимогам захисту, є актуальним та важливим завданням для розробників інтернет-магазинів.

У країнах Європейського Союзу впроваджуються стандартизовані протоколи безпеки, зокрема використання двофакторної аутентифікації, авторизація через соціальні мережі або сервіси єдиного входу (SSO), що значно підвищує рівень захисту та зручність для користувачів. В Україні цифровізація бізнесу також набирає обертів, однак у багатьох інтернет-магазинах все ще використовуються застарілі підходи до аутентифікації, що створює потребу в модернізації. Це відкриває можливості для впровадження сучасних рішень, зокрема інтеграції з державними сервісами, такими як Дія.Sign, або підтримки протоколів OAuth 2.0 та OpenID Connect. Адаптація до європейських практик не лише підвищує безпеку, а й сприяє довірі користувачів до цифрових продуктів українського ринку.

Одним із ефективних рішень є розробка універсального та розширюваного модуля аутентифікації, який можна інтегрувати до інтернет-магазину з урахуванням кращих практик галузі. Такий модуль повинен забезпечувати безпечну реєстрацію, вхід до облікового запису, відновлення пароля, та мати можливість інтеграції з зовнішніми сервісами [1].

Метою даної роботи є створення програмного модуля аутентифікації для клієнтів інтернет-магазину, що забезпечить надійний захист даних користувачів та зручну взаємодію з системою.

Основними завданнями є проєктування архітектури модуля, реалізація функціоналу реєстрації, входу та виходу з системи, та захист від типових загроз (злам пароля, сесійні атаки, SQL-ін'єкції тощо). Для досягнення поставленої мети у розробці використано сучасні технології веб-програмування, зокрема платформу ASP.NET (або PHP, залежно від вашого стеку), систему керування базами даних MySQL, протоколи HTTPS та JWT для захищеної передачі даних [2].

Результатом проєкту є функціональний, безпечний та придатний до інтеграції модуль аутентифікації, що може бути використаний у реальних інтернет-магазинах з метою підвищення рівня інформаційної безпеки та довіри клієнтів до сервісу.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Огляд сучасного стану предметної галузі

1.1.1 Актуальність використання автентифікації

У XXI столітті світ переживає справжню цифрову трансформацію, що охоплює всі сфери суспільного життя від освіти та охорони здоров'я до економіки й торгівлі. Особливо стрімкого розвитку зазнала електронна комерція, яка сьогодні є однією з найбільш динамічних і прибуткових галузей цифрового ринку. За оцінками eMarketer, у 2023 році світові обсяги електронної торгівлі перевищили $6 трильйонів, а до 2027 року очікується подальше зростання на рівні 56% [1]. Водночас розширення онлайн-продажів супроводжується підвищеними вимогами до інформаційної безпеки, зокрема захисту персональних даних клієнтів, безпечної авторизації та зручного доступу до облікових записів.

Інтернет-магазини як цифрові платформи для продажу товарів і послуг потребують не лише функціонального інтерфейсу та якісного сервісу, а й високого рівня довіри з боку користувачів. За даними дослідження компанії Statista, понад 70% користувачів покидають сайт або не завершують реєстрацію, якщо не впевнені у безпеці своїх персональних даних. Одним з основних аспектів формування довіри є реалізація ефективної та захищеної системи аутентифікації. Сьогодні соїть саме питання ідентифікації та перевірки прав користувачів набуває першочергового значення для розробників сучасних веб-рішень.

Актуальність обраної теми зумовлена кількома ключовими чинниками. По-перше, зростаюча кількість кібератак та витоків даних потребує постійного вдосконалення механізмів аутентифікації. За даними IBM Security, у 2022 році середня вартість витоку персональних даних становила $4.35 мільйона, а більшість випадків були пов'язані з недостатньо надійними системами доступу [2]. По-друге, значна частина малих та середніх підприємств в Україні використовує застарілі підходи до аутентифікації, обмежуючись лише логіном і паролем, що підвищує ризики несанкціонованого доступу та підриває довіру клієнтів до онлайн-торгівлі, що допомогає кінцевому споживачу.

Окрім безпеки, важливу роль відіграє і зручність користування. Сучасний інтернет-користувач очікує швидкої авторизації через електронну пошту, номер телефону, облікові записи в соціальних мережах або навіть біометричні методи. Водночас важливо забезпечити баланс між зручністю та захищеністю, уникнувши складних процедур, які можуть відштовхнути потенційного покупця. У цьому контексті актуальним є впровадження універсальних модулів аутентифікації, які б поєднували гнучкість налаштувань, інтеграцію з зовнішніми сервісами (OAuth2, Google/Facebook login, тощо), давая відповідали б стандартам безпеки OWASP та GDPR [3].

З технічної точки зору, реалізація модуля аутентифікації є невід'ємною частиною архітектури будь-якого веб-застосунку. Вона включає процеси реєстрації, авторизації, управління сесіями, відновлення паролю, перевірку ролей користувача та контроль доступу до конфіденційної інформації. У складних комерційних системах модуль аутентифікації також взаємодіє з платіжними шлюзами, службами доставки, аналітичними інструментами, CRM-системами, що вимагає високого рівня надійності, масштабованості та підтримки сучасних протоколів безпеки.

Сучасна нормативно-правова база, зокрема Закон України «Про захист персональних даних», та міжнародні стандарти ISO/IEC 27001, зобов'язують суб'єктів електронної комерції впроваджувати технічні й організаційні заходи безпеки. Будемо вважати, що розробка програмного модуля аутентифікації не лише має прикладне значення для покращення взаємодії з клієнтами, а й виконує функцію відповідності чинному законодавству.

Окремої уваги заслуговує тенденція до уніфікації процесів реєстрації та авторизації, яка реалізується через модульні підходи, наприклад, за допомогою JWT (JSON Web Tokens), які забезпечують зручну авторизацію у SPA-додатках. Використання токенів дозволяє уникати збереження сесій на сервері, що покращує масштабованість системи, знижує навантаження та підвищує ефективність обробки запитів. Крім того, використання JWT спрощує інтеграцію сторонніх сервісів і мікросервісних компонентів, забезпечуючи єдиний механізм

автентифікації для всієї екосистеми додатка.

Ще одним аргументом на користь вибору цієї теми є можливість практичного застосування розробленого модуля у реальному середовищі, наприклад, як частини дипломного стартапу або MVP-проєкту. Універсальність такого модуля дає змогу легко адаптувати його до різних платформ, мов програмування або систем управління контентом (CMS), що значно розширює сферу його використання.

Тому зосередження було на вибір теми «Розробка програмного модуля аутентифікації клієнта інтернет-магазину» зумовлений поєднанням високої практичної цінності, актуальності для цифрової економіки, науково-технічного інтересу та потреби підвищення кібербезпеки в електронній комерції. Розробка такого модуля дозволяє не лише набути глибших знань з інформаційної безпеки, архітектури веб-додатків та сучасних протоколів, але й зробити внесок у створення більш захищених та зручних онлайн-сервісів для українських користувачів.

У контексті поточних глобальних викликів, зокрема масової діджиталізації бізнесу та переходу до віддаленого обслуговування клієнтів, питання безпечної аутентифікації набуває нового значення. Пандемія COVID-19 суттєво прискорила ці процеси: за даними Digital 2022 Global Overview Report, кількість онлайн-покупок у світі зросла більш ніж на 25% у порівнянні з довоєнним 2019 роком. В Україні також спостерігається сталий ріст цифрової економіки: частка e-commerce у структурі ВВП поступово зростає, а кількість онлайн-замовлень у малому та середньому бізнесі подекуди перевищує 60% від загального обсягу продажів.

Однак разом з активним розвитком цифрової торгівлі зростає і кількість загроз. Українська асоціація з IT-безпеки (UA-CERT) у 2023 році зафіксувала понад 100 тис. інцидентів, пов’язаних із зломами особистих акаунтів, крадіжкою даних або шахрайськими операціями з банківськими картками. Найбільш уразливою ланкою у більшості випадків залишається саме процес аутентифікації коли зловмисник може підібрати пароль або скористатися вразливістю в логіці сесійного доступу.

Станом на сьогодні в Україні відсутні загальнонаціональні вимоги або технічні регламенти щодо стандартів аутентифікації в приватних онлайн-сервісах.

Деякі великі платформи як-от Prom.ua, Rozetka чи Monobank впровадили двофакторну аутентифікацію, яка зараз зосередженна на хешування та обробки запитів до серверу.

Абсолютна більшість інтернет-магазинів, особливо у сегменті малого бізнесу, або взагалі не мають розгалуженої системи контролю доступу, або користуються мінімалістичними рішеннями без врахування стандартів OWASP або NIST.

На рис. 1.1 зображений механізм автентифікації та передачі захищених даних між клієнтом та сервером.

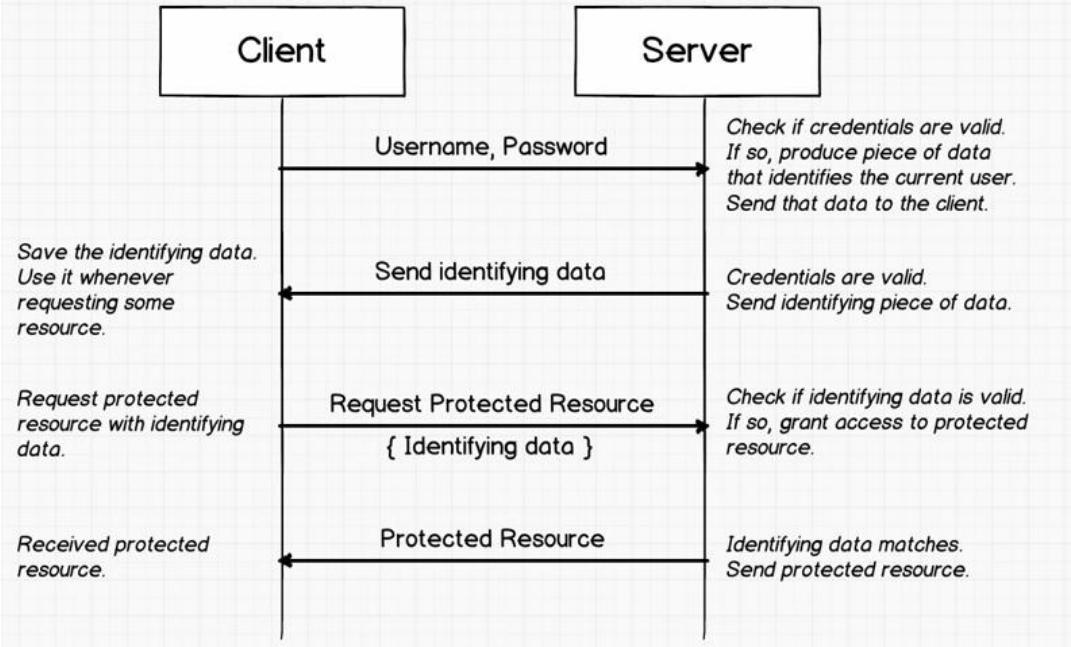


Рисунок 1.1. Приклад макету для запиту доступу до серверу

Окремим фактором мотивації виступає потреба в локалізованих рішеннях. Значна частина open-source бібліотек, які використовуються українськими розробниками, створені у США або Європі, не мають українськомовної документації, не враховують особливості локальних ринків або специфіку правових вимог щодо персональних даних. Розробка власного модуля, який відповідає як технічним, так і законодавчим вимогам України, дає змогу створити інструмент, придатний до широкого використання в реальних умовах, зокрема у локальних стартапах, онлайн-магазинах та CRM-системах. Основну увагу потрібно поставити на оптимізацію цього етапу дозволяє значно підвищити коефіцієнт конверсії це важливий показник у бізнес-аналітиці на якому

зовереджени усі компанії світу [4].

Важливою перевагою розробки модуля аутентифікації є те, що він легко масштабований і може стати базовою частиною будь-якого веб-проєкту. Наприклад, у майбутньому на його основі можна розширити функціонал до повноцінної системи управління користувачами (user management system), з розмежуванням прав доступу, журналом активності, API для зовнішніх інтеграцій або навіть реалізацією авторизації через державний сервіс Дія.Sign або BankID. Обраний модуль може стати стартовим компонентом для складніших систем, які відповідають вимогам держсектору або B2B-ринку.

1.1.2 Принципи роботи систем електронної аутентифікації: Дія.Sign, BankID та міжнародні аналоги

Процес аутентифікації користувачів є невіддільною частиною безпечної взаємодії громадян із державними та приватними онлайн-сервісами. Забезпечення достовірної ідентифікації особи є фундаментальним елементом сучасної цифрової інфраструктури, адже від цього залежить рівень довіри до електронних послуг, а також захист персональних даних і конфіденційної інформації.

В Україні, як і в багатьох країнах світу, активно розвиваються національні платформи електронної ідентифікації, серед яких важливе місце займають Дія.Sign та BankID НБУ. Ці системи забезпечують високий рівень безпеки та сумісність із багатьма державними сервісами, що робить їх надзвичайно популярними серед громадян. Завдяки цим інструментам, користувачі можуть здійснювати електронну ідентифікацію за допомогою електронного підпису або через підтвердження особи в банку, який є учасником відповідної системи. Це особливо актуально в умовах надзвичайних ситуацій або при обмеженому доступі до установ.

Окрім зручності для громадян, такі платформи сприяють підвищенню ефективності роботи державних органів, зменшенню паперового документообігу та оптимізації внутрішніх процесів. Використання сучасних технологій у сфері аутентифікації дозволяє значно скоротити час на отримання послуг і знижує ризик шахрайських дій. На рис. 1.2 зображена схема запиту на аутентифікацію до стороніх сервісів, такі як Дія.Sign та BankID.

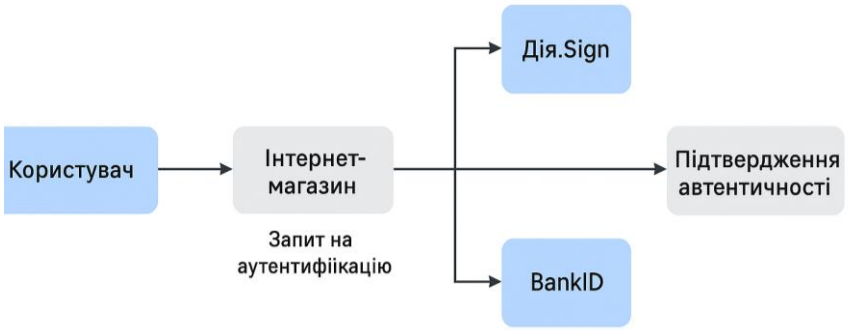


Рисунок 1.2. Схема запиту на аутентифікацію до сервісів

Дія.Sign – сервіс кваліфікованого електронного підпису, реалізований Міністерством цифрової трансформації України в межах мобільного застосунку "Дія". Його принцип роботи ґрунтується на використанні хмарного зберігання ключів електронного підпису, що дозволяє громадянину підписувати документи, підтверджувати дії або проходити аутентифікацію в один дотик через мобільний додаток.

Типова схема роботи Дія.Sign виглядає так:

1. Користувач ініціює дію (наприклад, вхід в особистий кабінет або підписання договору);
2. Система надсилає запит до сервера Дія.Sign на виконання підпису/аутентифікації;
3. Користувач отримує push-сповіщення у застосунку "Дія" з проханням підтвердити дію;
4. Після підтвердження (зазвичай через біометрію або PIN-код), виконується криптографічна операція з використанням збереженого ключа;
5. Сторона, яка надіслала запит, отримує підтвердження дійсності підпису або факту аутентифікації.

Модифікація кодової бази для Дія.Sign забезпечує повноцінну юридичну силу підпису та високий рівень захисту за стандартами ЄС (eIDAS), що робить його універсальним інструментом не лише для державних сервісів, а й для бізнесу. Його інтеграція в інформаційні системи дозволяє автоматизувати процеси підписання документів, значно скорочуючи час та витрати. Наявність простого інтерфейсу та підтримці широкого кола форматів, Дія.Sign стає зручним рішенням для користувачів з різним рівнем технічної підготовки.

BankID НБУ це система дистанційної ідентифікації користувачів на основі даних, що надаються банками, у яких особа має відкритий рахунок. Створена Національним банком України, ця система дозволяє швидко підтвердити особу в онлайн-сервісах без необхідності завантаження документів. Використання BankID НБУ підвищує зручність доступу до цифрових послуг та сприяє розвитку електронного урядування в Україні, забезпечує безпеку при використанні додатку для громадян.

Механізм дії системи BankID НБУ розпочинається з вибору користувачем відповідного способу входу на сайті або в мобільному застосунку. Серед запропонованих варіантів автентифікації він обирає авторизацію через BankID. Після цього система автоматично перенаправляє користувача на захищену сторінку обраного ним банку.

На цьому етапі користувач проходить авторизацію в інтернет-банкінгу, використовуючи свої звичні облікові дані. Після успішного входу він підтверджує згоду на передачу персональних даних сервісу, до якого здійснюється вхід. Така згода є обов'язковою умовою для продовження процесу ідентифікації, адже вона гарантує, що дані будуть передані лише з відома користувача.

Після підтвердження банк надсилає необхідну інформацію зокрема, прізвище, ім'я, по батькові, ІПН, дату народження та інші дані у зашифрованому вигляді до цільового сервісу. Цей процес відбувається із дотриманням сучасних стандартів шифрування та кібербезпеки. Отримавши ці дані, сайт або застосунок може підтвердити особу користувача, після чого надає доступ до особистого кабінету або автоматично створює новий обліковий запис, якщо користувач реєструється вперше.

Варто зазначити, що BankID не передає жодної банківської інформації, такої як залишок на рахунках, історія транзакцій чи інші фінансові дані. Система оперує виключно тією інформацією, яка необхідна для ідентифікації особи, що повністю відповідає вимогам Закону України «Про захист персональних даних» і міжнародним стандартам конфіденційності. Такий підхід дозволяє значно спростити доступ до онлайн-сервісів, зберігаючи при цьому високий рівень

безпеки та контролю з боку користувача [6].

Український досвід не є унікальним. У Європі, США та Азії подібні технології давно стали стандартом:

1. eIDAS (ЄС) це сучастна європейська ініціатива, яка забезпечує взаємне визнання електронної ідентифікації між країнами ЄС. Кожен громадянин може використовувати свою eID для доступу до сервісів в інших країнах;
2. BankID (Швеція, Фінляндія) у цих країнах BankID є головним інструментом входу до банківських, медичних та державних порталів. Користувач ідентифікується за допомогою застосунку або спеціального ключа, підв'язаного до особистої ID-картки;
3. Mobile ID (Естонія) національна система, яка дозволяє ідентифікувати особу та підписувати документи прямо з телефону, використовуючи SIM-карту.

Їх впровадження дозволяє не лише підвищити рівень довіри до електронної комерції, а й знизити ризики шахрайства, оптимізувати процеси реєстрації та підписання документів, зменшити навантаження на адміністраторів сервісів. Інтеграція таких рішень у модуль аутентифікації інтернет-магазину є кроком до сучасного, безпечного та зручного цифрового бізнесу [7].

Таблиця 1.1. Порівняльна характеристика систем аутентифікації

Критерій	**Дія.Sign**	**BankID НБУ**	**Google / Facebook Login**
Тип системи	Електронний підпис / хмарна аутентифікація	Система віддаленої ідентифікації через банк	Соціальна ідентифікація через OAuth 2.0
Організація-власник	Міністерство цифрової трансформації України	Національний банк України	Google LLC / Meta Platforms Inc.
Дані, що передаються	ЕЦП, підтвердження дії, мінімальні ідентифікаційні дані	ПІБ, ІПН, адреса, дата народження тощо	Ім'я, email, аватар, (опційно: друзі, місцезнаходження)
Стандарт безпеки	eIDAS, КЕП, GDPR	Стандарти НБУ, SSL, GDPR	OAuth 2.0, OpenID Connect, GDPR

Продовження таблиці 1.1

Наявність токенів / підпису	Кваліфікований електронний підпис (КЕП)	Без підпису, лише передача даних	OAuth-токен доступу
Швидкість інтеграції	Середня (потрібна реєстрація у Дія.API)	Низька (договір із НБУ, доступ до тестсередовища)	Висока (готові бібліотеки, швидкий старт)
Вимоги до користувача	Наявність застосунку «Дія» та смартфона	Обліковий запис у банку-учаснику системи	Обліковий запис Google або Facebook
Відповідність українському законодавству	Повна (державна платформа)	Повна (державна платформа)	Часткова (не є державною системою, дані за кордоном)
Застосування у державних проєктах	Так	Так	Ні
Можливість використання у бізнесі	Так	Так	Так
Можливість підписання документів	Так (КЕП через мобільний застосунок)	Ні	Ні
Механізм підтвердження	Біометрія або PIN через застосунок «Дія»	Вхід до онлайн-банкінгу + згода на передачу даних	Вхід у соціальну мережу + підтвердження доступів

Як показано в таблиці, кожен із розглянутих способів автентифікації має свої особливості та сферу застосування. Дія.Sign забезпечує найвищий рівень юридичної сили завдяки використанню кваліфікованого електронного підпису (КЕП), що робить його ідеальним рішенням для підписання угод, договорів та інших юридично значущих документів.

BankID НБУ, своєю чергою, дозволяє здійснити зручну дистанційну ідентифікацію без потреби у фізичній присутності або наданні сканів документів. Проте цей механізм не призначений для підписання документів, а лише підтверджує особу користувача.

Найпростіший у реалізації спосіб входу через акаунти Google або Facebook. Він значно спрощує доступ до сервісу, однак має нижчий рівень захисту та не надає

можливості підтвердження юридичної особи чи накладання підпису.

1.1.3 Огляд та аналіз технічної інтеграції сервісів у веб-додаток

Одним із ключових аспектів створення надійного та зручного цифрового сервісу є інтеграція сторонніх систем аутентифікації, які дозволяють підтверджувати особу користувача без потреби ручного введення великої кількості персональних даних. В умовах зростаючої уваги до інформаційної безпеки та відповідності законодавству про захист персональних даних, використання національних та міжнародних платформ ідентифікації таких як Дія.Sign та BankID НБУ стає надзвичайно актуальним.

Реалізація таких інтеграцій у веб-додаток дає змогу не лише підвищити рівень довіри з боку користувачів, а й значно оптимізувати процеси реєстрації та входу, скоротивши кількість дій, які повинен виконати користувач. Крім того, інтеграція із сервісами державного рівня дозволяє підтвердити достовірність даних, що виводить роботу електронної комерції на якісно новий рівень.

Усі зовнішні платформи аутентифікації, включно з Дія.Sign та BankID, мають API (Application Programming Interface) це набір доступних програмних інтерфейсів, через які веб-додаток може взаємодіяти з відповідним сервісом. Інтеграція відбувається через стандартні HTTP-запити, обмін JSON-даними та використання токенів авторизації для захисту запитів.

Загальна схема виглядає наступним чином:

1. Користувач натискає кнопку «Увійти через Дія» або «BankID».
2. Веб-додаток надсилає запит до API сервісу (наприклад, до https://id.gov.ua/api/auth).
3. Користувач перенаправляється на сторінку авторизації обраного сервісу.
4. Після підтвердження (наприклад, у застосунку "Дія" або через банк), система отримує токен аутентифікації або набір даних.
5. Сервер веб-додатку перевіряє цей токен через backend, використовуючи секретний ключ.
6. У разі успіху користувач вважається автентифікованим, створюється сесія або видається JWT-токен для подальшої роботи.

Дія.Sign надає офіційне API, описане у документації Мінцифри, яке дозволяє розробникам інтегруватися через REST-запити. Щоб розпочати роботу, необхідно зареєструвати веб-додаток у реєстрі Дія.Sign, після чого надаються облікові ключі client_id та secret. Далі розробник отримує токен доступу через стандартний endpoint /oauth/token, що відкриває можливість подальшої взаємодії з сервісом x8ї.

Після авторизації можна ініціювати підпис або підтвердження особи, скориставшись endpoint'ом /sign/request. У цьому запиті вказується, яку саме дію має підтвердити користувач: вхід, транзакцію чи підпис документа. Користувач отримує сповіщення у застосунку «Дія», де підтвердження може відбуватися через FaceID або PIN-код.

Після завершення дії на стороні користувача, API повертає відповідь із результатом підпису або ідентифікації. Отримані дані можна перевірити на сервері, зокрема провести перевірку цифрового підпису, щоб гарантувати їхню справжність і цілісність.

Технічно інтеграція може бути реалізована як middleware або як окремий service-клас у фреймворках ASP.NET Core, Laravel, Django тощо. Основну увагу слід приділити обробці callback-подій, які приходять від Дії, та налаштуванню SSL-сертифікатів, бо обмін завжди має відбуватися по захищеному каналу.

Інтеграція з BankID НБУ дещо складніша через потребу укладання офіційного договору з Національним банком України. Після цього розробнику надаються тестові облікові дані та специфікація API.

Процес аутентифікації користувача за допомогою системи BankID НБУ відбувається поетапно та ґрунтується на принципах безпечної передачі персональних даних через банківські установи.

На першому етапі веб-додаток ініціює запит на проходження аутентифікації через BankID. Це зазвичай реалізовано у вигляді кнопки «Увійти через BankID» або подібного елемента на сторінці входу до особистого кабінету. Після натискання на відповідну кнопку користувач потрапляє на сторінку, де йому пропонується вибрати банк зі списку установ, які беруть участь у системі BankID НБУ та зможуть авторизувати користувача.

Після вибору банку користувача автоматично перенаправляють на офіційний сайт або сторінку входу інтернет-банкінгу обраного банку.

Як слід зазначити можна проходити звичну для себе процедуру автентифікації, наприклад, за допомогою логіну та паролю, одноразового SMS-коду або двофакторної авторизації не дуже швидкий етап. Важливо зазначити, що саме банк виступає гарантом достовірності особи, оскільки дані про клієнта були вже верифіковані на етапі відкриття банківського рахунку.

Після успішного входу в банківську систему, користувачу відображається запит на дозвіл передачі персональних даних до зовнішнього сервісу, яким у цьому випадку виступає веб-додаток. До таких даних можуть входити прізвище, ім’я, по батькові, реєстраційний номер облікової картки платника податків (РНОКПП), дата народження, зареєстроване місце проживання тощо. Користувач має можливість погодитись або відхилити запит і лише у випадку згоди передача інформації буде здійснена.

Наступним етапом є передача зашифрованих персональних даних банком до API сервісу, який їх запитав. Дані шифруються відповідно до протоколів SSL/TLS, що виключає можливість їх перехоплення чи підробки. Веб-додаток, у свою чергу, отримує ці дані у структурованому форматі (зазвичай JSON), після чого використовує їх для оновлення вже існуючого профілю користувача.

Якщо такий акаунт уже існує в системі (наприклад, з тією ж електронною адресою або РНОКПП), він може бути автоматично прив’язаний до авторизованої особи.

Як видно на практиці, BankID забезпечує швидку, безпечну та юридично захищену ідентифікацію, значно спрощуючи процес входу до веб-додатку, зменшуючи потребу у ручному введенні даних і підвищуючи загальну зручність користування [9].

Додатковою перевагою такого підходу є мінімізація ризиків фальсифікації особи, оскільки передані дані мають офіційне походження та підтверджені банківською установою. Важливою частиною є реалізація верифікації підпису та логіки опрацювання помилок наприклад, у випадку, якщо користувач скасував

запит або банк відмовив у передачі даних. Також обов'язковою умовою є підтримка персональних політик конфіденційності, з якими користувач має ознайомитися перед наданням згоди.

Інтеграція подібних сервісів вимагає не лише розробки основної логіки взаємодії, а й належної побудови систем безпеки. Зокрема, важливо впровадити механізми логування, аудит доступу та своєчасне сповіщення адміністратора у разі виявлення підозрілої активності, що дозволить оперативно реагувати на потенційні загрози.

Для досягнення мети, рекомендується реалізувати обмеження строку дії токенів (TTL), перевірку IP-адрес клієнтів, та захист від міжсайтових запитів через валідацію CSRF-токенів. З погляду захисту персональних даних, важливо зберігати лише мінімально необхідну інформацію відповідно до вимог GDPR.

Технічна інтеграція сервісів електронної ідентифікації, таких як Дія.Sign та BankID НБУ, є важливим етапом цифрової трансформації українського бізнесу. Вона дозволяє досягти вищого рівня безпеки, зручності та відповідності стандартам європейського електронного урядування. Упровадження таких рішень сприяє побудові довіри між користувачем і платформою, знижує ризики шахрайства та автоматизує бюрократичні процеси, які раніше потребували фізичної присутності. Саме тому впровадження подібної інтеграції не лише технічне завдання, а й стратегічна перевага для будь-якого сучасного веб-додатку.

1.2 Побудова системи автентифікації

1.2.1 Побудова та аналіз архітектури застосунку

У процесі розробки веб-застосунку для аутентифікації клієнтів інтернет-магазину було обрано архітектурний підхід, який відповідає принципам багатошарової (багаторівневої) архітектури, а саме N-layer architecture, що є загальноприйнятим підходом у розробці корпоративних застосунків на базі платформи .NET.

N-layer architecture, або багаторівнева архітектура, є популярним шаблоном проєктування програмного забезпечення, що передбачає поділ застосунку на логічно

незалежні шари. Архітектура веб-додатку дозволяє підвищити гнучкість, масштабованість і підтримуваність коду, оскільки кожен шар виконує чітко визначену роль.

Першим шаром зазвичай є презентаційний рівень, або інтерфейс користувача, який відповідає за взаємодію з користувачем та відображення даних і приймання введених даних. Далі йде шар бізнес-логіки, що містить основні правила й алгоритми роботи застосунку, реалізуючи ключову функціональність згідно з вимогами домену.

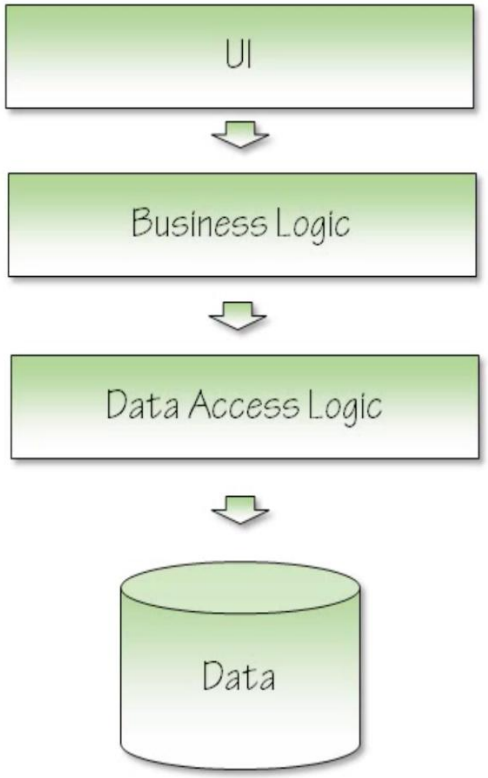


Рисунок 1.3. Приклад обміну даними у межах N-Layer

Використання багатошарової архітектури забезпечує високий рівень модульності, масштабованості, розширюваності, повторного використання коду та тестованості. Чітко визначеним межам між шарами стає можливим вільне оновлення або заміна реалізації окремого шару без необхідності змін в інших частинах системи.

Шар доступу до даних забезпечує взаємодію з джерелами зберігання інформації, наприклад, з базами даних, а також ізолює бізнес-логіку від деталей реалізації зчитування або збереження даних. Нарешті, на найнижчому рівні

базується фізична база даних, де безпосередньо зберігаються всі дані, необхідні для функціонування застосунку. Структурованість архітектури дозволяє спростити тестування, підтримку та розвиток системи.

Аналіз структури та вмісту проєкту ShopAuth дозволяє впевнено стверджувати, що в ньому реалізовано класичну багаторівневу архітектуру типу N-layer. Усі основні компоненти розподілені між трьома логічними рівнями, кожен з яких виконує свою чітко визначену роль у загальній архітектурі системи.

Перший рівень Presentation Layer відповідає за взаємодію з користувачем. У цьому проєкті він представлений частково або передбачений до реалізації у вигляді окремого фронтенду на базі сучасних UI-технологій, таких як React. Його основна функція полягає у прийомі запитів і передачі їх до внутрішніх сервісів.

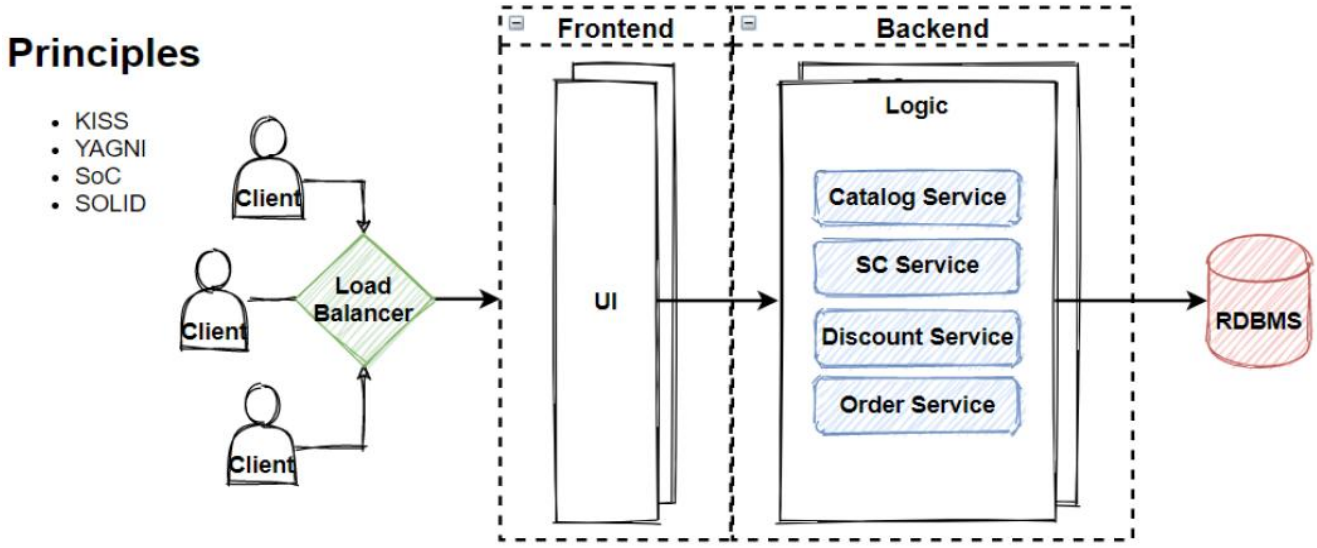


Рисунок 1.4. Принципи запиту даних з клієнтської частини за допомогою багатошарової архітектури

Цей шар є найбільш технічно насиченим у межах проєкту та реалізований у вигляді окремого проєкту під назвою ShopAuth.DataAccess. Його основне завдання забезпечення взаємодії з базою даних. Центральним компонентом тут є файл DataContext.cs, який реалізує контекст Entity Framework Core і слугує посередником між кодом застосунку та таблицями бази даних.

Варто зазначити, що Entity Framework Core створює шар поміж застосунком та базою даних, що, з одного боку, покращує безпеку застосунку, однак тим самим зменшує швидкодію, тому для виконання самих запитів буде використовуватися

з’єднання з базою напряму та з використанням мови запитів SQL.

Структури таблиць описані за допомогою класів моделей, зокрема User.cs та UserSession.cs, які відображають відповідні сутності у базі. Для організації доступу до даних використовується патерн репозиторію, реалізований у класах DataRepository.cs та IDataRepository.cs. Він дозволяє створити абстракцію над механізмами збереження та отримання даних, ізолюючи їх від бізнес-логіки застосунку.

Бізнес-логіка зосереджена у класах, що реалізують доменні моделі, зокрема User і UserSession, та відповідають за обробку основних сценаріїв автентифікації. Цей рівень забезпечує логічну цілісність і централізовану реалізацію ключових процесів, зберігаючи незалежність від деталей зберігання чи відображення даних.

Шар доступу до даних реалізований як окремий проєкт ShopAuth.DataAccess, у якому містяться репозиторії, контекст бази даних та моделі сутностей. Цей рівень забезпечує ізоляцію бізнес-логіки від фізичного сховища даних і дозволяє гнучко змінювати або розширювати механізми взаємодії з базою без впливу на інші частини системи.

Подібне розділення не лише спрощує модульне тестування, а й забезпечує гнучкість у подальшому розвитку проєкту. Наприклад, за потреби можна безболісно перейти на іншу систему керування базами даних, додати підтримку реплікації або впровадити кешування, не змінюючи основну логіку застосунку.

Наразі частина бізнес-логіки реалізується безпосередньо у класах типу DataRepository, однак існуюча архітектура дозволяє легко винести цю логіку в окремий проєкт, наприклад, ShopAuth.Business. У цьому окремому шарі можуть бути зосереджені такі аспекти, як валідація введених користувачем даних, обробка аутентифікації (зокрема, порівняння хешованих паролів та генерація токенів), керування сесіями, визначення правил доступу.

Шар взаємодії з користувачем відповідає за прийом запитів і повернення відповідей у структурованому вигляді. Технологія .NET дозволяє реалізовувати цей рівень як API або UI (наприклад, на основі React, Blazor, Razor Pages тощо). Основне завдання цього шару обробка HTTP-запитів, передача даних у бізнес-

логіку (BLL) і формування відповідей у вигляді, зрозумілому кінцевому користувачу або зовнішньому клієнту.

Для кращого розуміння загальної взаємодії між компонентами в межах багаторівневої архітектури, на рис. 1.5 представлено структуру типового підходу до побудови N-layer застосунків. Такий підхід дає змогу підтримувати масштабованість, зручність супроводу та можливість незалежної еволюції кожного з шарів системи.

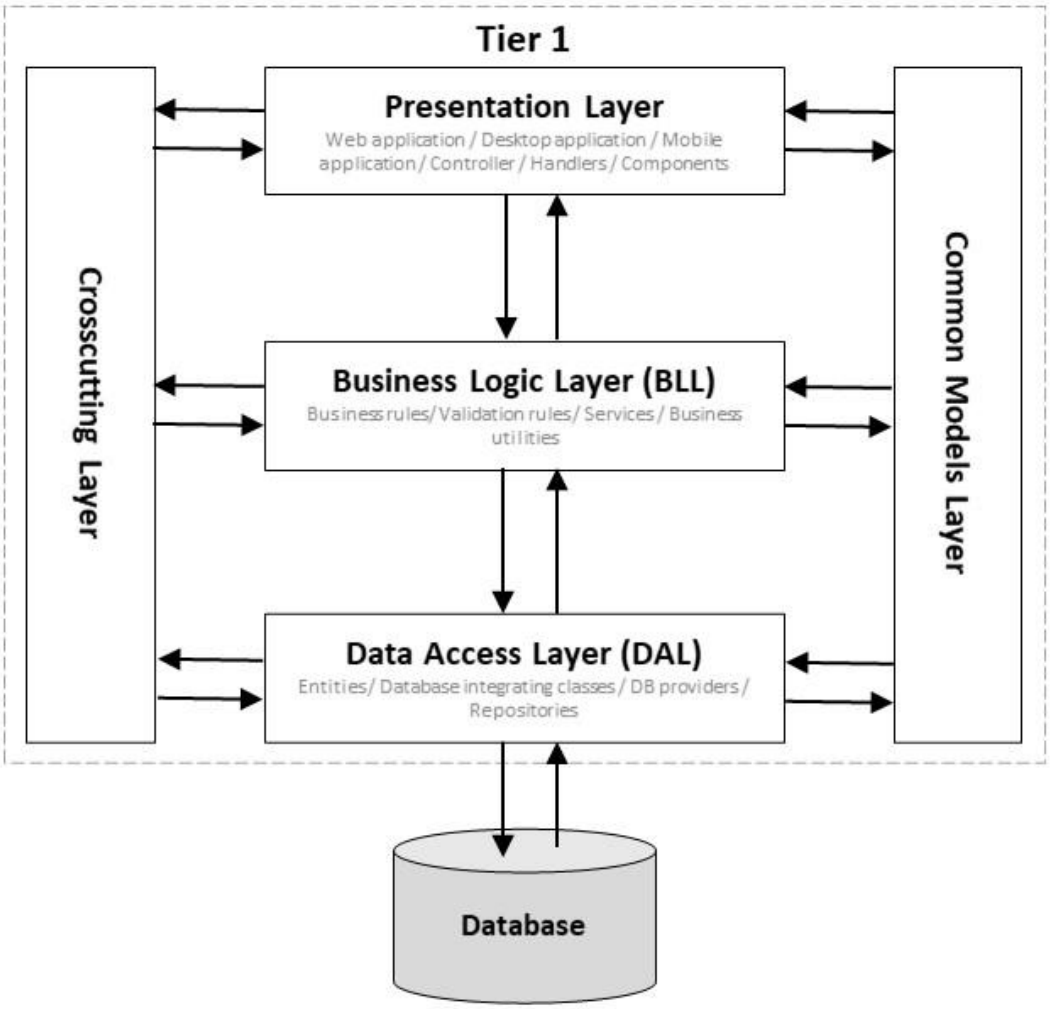


Рисунок 1.5. Поділ системи на логічні рівні

На наведеній діаграмі чітко відображено поділ системи на логічні рівні, кожен з яких виконує власну функцію у процесі обробки запитів і взаємодії з базою даних. На верхньому рівні розташовується Presentation Layer у якому шар презентації, який відповідає за інтерфейс користувача: це можуть бути вебсторінки, мобільні додатки, API-контролери чи будь-які інші компоненти, з якими безпосередньо взаємодіє користувач. Саме цей шар ініціює запити та

отримує відповіді, які згодом відображаються у зручній формі.

У межах такої архітектури презентаційний шар виступає своєрідним посередником між користувачем та внутрішньою логікою застосунку. Він приймає вхідні запити, викликає відповідні методи бізнес-логіки для обробки цих запитів і повертає результат у зручному форматі та як правило, у вигляді HTML-сторінок або JSON-відповідей для API. Мая ці алгоритми користувач отримує швидкий і структурований зворотний зв'язок у зручному для нього вигляді. Така організація дозволяє легко адаптувати інтерфейс до різних типів клієнтів вебдодатків, мобільних застосунків або зовнішніх сервісів.

Однією з основних причин вибору багаторівневої архітектури стала її практична ефективність і відповідність сучасним вимогам до побудови надійних та масштабованих програмних систем.

Однією з ключових переваг такої архітектури є модульність**,** де кожен логічний шар реалізує чітко визначену функціональність, що дозволяє ізолювати зміни, спростити налагодження та забезпечити незалежне тестування. Наприклад, можна змінити реалізацію доступу до бази даних, не зачіпаючи основну бізнес-логіку або користувацький інтерфейс, що є надзвичайно важливим у довготривалих проєктах з активною фазою підтримки.

Не менш важливою перевагою є масштабованість та чіткому розділенню відповідальностей у проєкт легко інтегрувати нові функціональні можливості, розширити обробку запитів, додати нові сервіси або API без необхідності кардинальної перебудови існуючої структури. Архітектура дозволяє «рости» як горизонтально (через поділ відповідальностей між модулями), так і вертикально (додаючи нові технології або рівні обробки даних).

Ще одним вагомим аргументом на користь багатошарової архітектури є можливість повторного використання компонентів. Зокрема, бізнес-логіка, винесена в окремий шар, може бути використана одночасно як у веб-версії інтернет-магазину, так і в мобільному застосунку або API, який взаємодіє з партнерськими сервісами. Це значно скорочує витрати на розробку та супровід кількох версій одного й того ж продукту.

Також архітектурна модель сприяє підвищенню рівня безпеки, оскільки дозволяє чітко розділити доступ до даних та обробку запитів. Такий поділ виключає або мінімізує ризик неконтрольованої взаємодії між шарами, що може призвести до витоків інформації чи несанкціонованого доступу до конфіденційних ресурсів.

Зрештою, застосування N-layer архітектури позитивно впливає на простоту обслуговування. Певній логічній організації та структурованості коду, розробник може швидко знайти та усунути проблему в межах певного шару, не ризикуючи порушити загальну роботу системи. Це критично важливо у командній роботі, коли над проєктом працює кілька фахівців одночасно. Як ми можемо бачити, вибір саме багаторівневої архітектури є цілком виправданим і повністю відповідає поставленим вимогам щодо якості, гнучкості та надійності програмного забезпечення. Приклад логічної схеми шарів зображено на рис. 1.7:

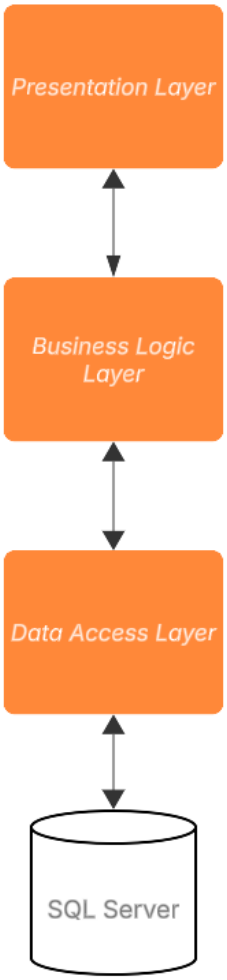


Рисунок 1.6. Схема логічних шарів в проєкті

У межах дипломного проєкту було створено програмну реалізацію, яка організована у вигляді структурованого багаторівневого застосунку відповідно до принципів N-layer архітектури. Код поділений на окремі проєкти (підкаталоги), кожен з яких виконує чітко визначену роль у загальній системі. Така організація не лише спрощує орієнтацію в коді, але й дозволяє ефективно дотримуватися принципів SOLID та забезпечити масштабованість рішення.

```
ShopAuth/
├── ShopAuth.sln
├── ShopAuth.DataAccess/
│   ├── DataContext.cs
│   ├── DataRepository.cs
│   ├── IDataRepository.cs
│   ├── Entities/
│   │   ├── User.cs
│   │   └── UserSession.cs
├── ShopAuth.DataAccess.Migrations/
│   ├── Program.cs
│   ├── appsettings.json
│   ├── appsettings.Development.json
```

Основний файл рішення ShopAuth.sln який забезпечує централізоване керування кількома проєктами всередині застосунку та дає змогу легко відкрити весь застосунок у середовищі розробки (наприклад, Visual Studio або Rider). Каталог ShopAuth.DataAccess/ відповідає за шар доступу до даних: у ньому міститься контекст бази даних DataContext.cs, реалізація патерну репозиторію (DataRepository.cs) та відповідний інтерфейс IDataRepository.cs, що слугує контрактом для взаємодії з базою.

Також тут розміщено каталог Entities/, у якому зберігаються сутності доменної моделі User та UserSession. Вони безпосередньо відповідають таблицям бази даних і описують основні об’єкти, з якими працює система аутентифікації.

Окремий проєкт ShopAuth.DataAccess.Migrations/ призначений для ініціалізації та керування міграціями бази даних за допомогою Entity Framework Core. Файл Program.cs містить стартову логіку для запуску консолі міграцій або відладки контексту. Конфігураційні файли проекту таки як appsettings.json та appsettings.Development.json використовуються для збереження налаштувань підключення до бази даних, середовища розгортання, логування тощо. Такий поділ дозволяє легко налаштувати окреме середовище для розробки, тестування або продакшену, не змінюючи базові конфігурації.

Реалізація N-layer архітектури у межах розробки модуля аутентифікації дозволяє досягти високого рівня організації коду, спрощує підтримку та масштабування проєкту, та закладає надійну основу для майбутнього розширення функціональності. Такий підхід є рекомендованим при створенні середніх і великих застосунків, що потребують довгострокового супроводу та розвитку, і повністю відповідає сучасним практикам програмної інженерії у сфері веб-розробки.

1.2.2 Побудова, структура та логіка взаємодії з базою даних

Проєктування бази даних є одним із ключових етапів створення веб-застосунку, особливо коли йдеться про модуль аутентифікації користувачів, де критично важливими є цілісність, захищеність та коректна організація даних.

На рисунку 1.7 представлено початковий етап проєктування структури бази даних у вигляді діаграми сутностей:

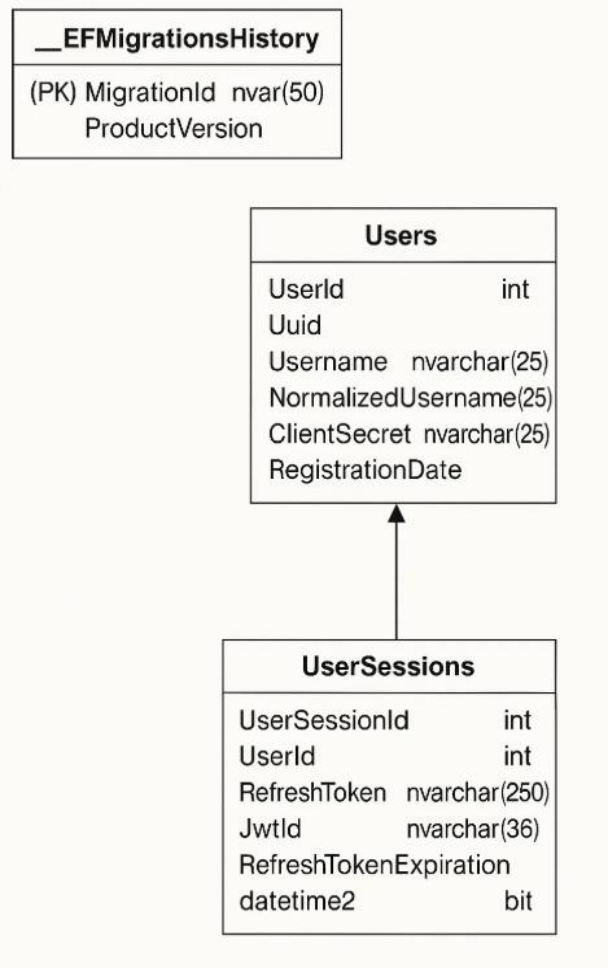


Рисунок 1.7. Структура схема бази даних

Сутність Users є основною у системі автентифікації та містить набір ключових полів, що описують кожного користувача. Центральним елементом є поле UserId, яке виступає первинним ключем і має тип ціле число. Воно унікально ідентифікує кожен запис у таблиці.

Відповідна модель зберігання дозволяє забезпечити ефективну ідентифікацію користувача, управління його сесіями та реалізацію механізмів безпечного доступу до системи. На цьому етапі визначаються основні сутності, з якими взаємодіє застосунок, та типи даних, зв'язки, ключі, обмеження та обов'язковість заповнення полів.

У межах розробленої системи передбачено дві базові сутності, які реалізовані у вигляді таблиць реляційної бази даних: Users та UserSessions. Таблиця Users зберігає основну інформацію про зареєстрованих користувачів, а таблиця UserSessions відповідає за керування активними сесіями та токенами доступу.

Окрім цього, у структурі передбачено поле Uuid унікальний ідентифікатор, який використовується для зовнішніх посилань і не дублює внутрішній числовий ключ. Імена користувача зберігаються у полях Username та NormalizedUsername, де останнє використовується для спрощеного пошуку. Обидва поля мають обмеження у 25 символів, що гарантує їхню компактність та уніфікованість.

Для забезпечення безпеки автентифікації кожен користувач має пов'язаний ClientSecret приватний ключ, який використовується у процесах ідентифікації або підпису. Дата реєстрації користувача фіксується в полі RegistrationDate, що дозволяє відслідковувати часові рамки появи облікового запису в системі.

Сутність UserSessions відіграє роль зберігання інформації про авторизаційні сесії користувачів. Вона має поле UserSessionId як первинний ключ і зовнішній ключ UserId, що встановлює зв'язок із таблицею Users. Такий підхід дозволяє системі підтримувати багаторазові сесії для одного користувача.

Кожен запис у UserSessions зберігає дані про видані токени: RefreshToken та JwtId, що ідентифікують поточну сесію. Також фіксується дата закінчення дії токена у полі RefreshTokenExpiration. Поле Redeemed є логічним прапорцем, який позначає, чи був відповідний токен вже використаний, що дозволяє забезпечити

додатковий рівень контролю над повторним використанням токенів та забезпечити захист у випадку компрометації токенів.

Удосконалена версія структури бази даних, яка була реалізована у фінальній версії застосунку, наведена на рис. 1.8.

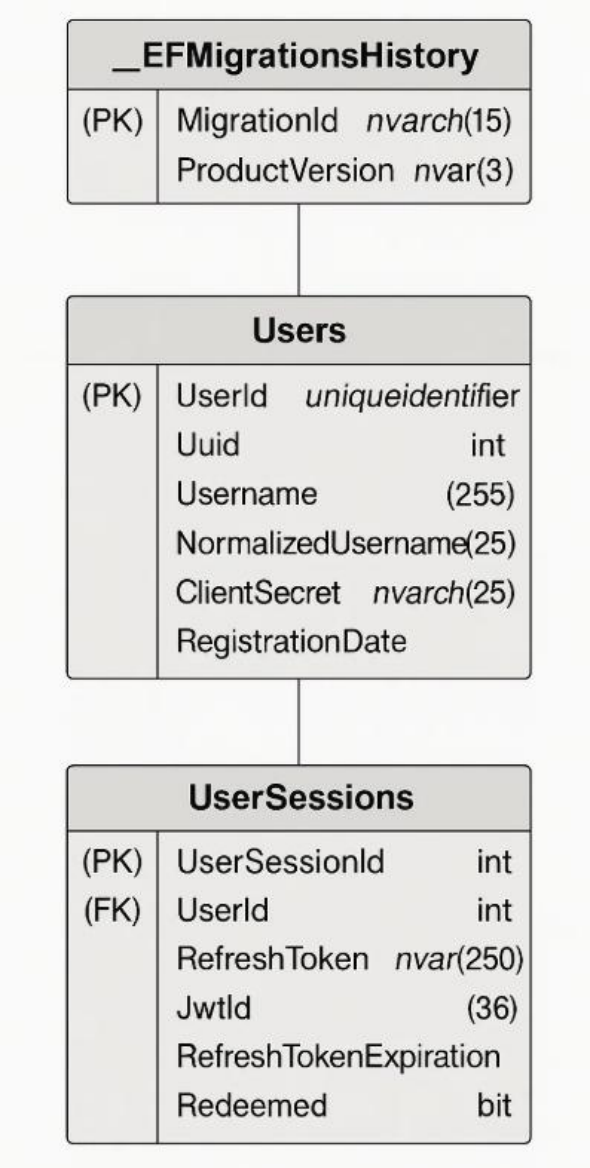


Рисунок 1.8. EF діаграма бази даних

Окрему увагу також приділено типам даних, які було підібрано відповідно до специфіки обробки. Наприклад, використання nvarchar для токенів і логінів забезпечує коректне зберігання символів будь-якого алфавіту, включаючи підтримку Unicode, що критично важливо для інтернаціоналізації. Поля типу bit для зберігання логічних значень дозволяють суттєво зменшити об’єм даних і прискорити фільтрацію в SQL-запитах. Bit є зручним способом представити логічні значення в базах даних.

Як видно з фінальної структури бази даних, у процесі проєктування було впроваджено низку вдосконалень, спрямованих на підвищення унікальності, зручності використання та підтримки структурної цілісності. Одним із ключових покращень стало переведення поля UserId на тип uniqueidentifier, що дозволяє забезпечити глобальну унікальність ідентифікаторів користувачів у системі, що особливо важливо в умовах масштабування або інтеграції з іншими сервісами.

Доповненням до технічної структури варто зазначити, що така побудова моделі є загальноприйнятою практикою у розробці безпечних багаторівневих застосунків. По окремому зберіганню сесій стає можливим реалізувати такі функції, як примусове завершення активного сеансу, авторизація на декількох пристроях, аудит дій користувача та контроль одночасного входу. Це важливо не лише з точки зору зручності, а й з точки зору кібербезпеки, адже більшість атак на систему автентифікації відбуваються саме через вразливості в управлінні сесіями.

Ще однією важливою зміною стало збільшення максимальної довжини полів Username та NormalizedUsername до 255 символів. Це дало змогу врахувати більшу гнучкість у введенні імен користувачів, підтримуючи різні алфавіти та складніші комбінації символів, не обмежуючи користувача надмірно короткими значеннями.

До сутності сесій було додано логічне поле Redeemed, яке виконує функцію індикатора статусу використання токена. Воно дозволяє системі фіксувати, чи був refresh-токен уже застосований, що унеможливлює його повторне використання та посилює безпеку механізмів автентифікації.

Усі зовнішні ключі у структурі бази даних чітко позначені як (FK), а первинні як (PK), що є хорошою практикою для підтримання ясності в структурі зв’язків між таблицями. Таке маркування спрощує читання схеми, полегшує подальше обслуговування бази та дозволяє уникати помилок при створенні запитів або модифікації структури.

Що стосується нормалізації, спроєктована база даних повністю відповідає першій нормальній формі (1NF), згідно з якою всі поля є атомарними, тобто не містять множинних або складених значень. Кожен атрибут у таблицях має одне конкретне значення для кожного запису.

Крім того, структура задовольняє вимоги другої (2NF) та третьої нормальної форми (3NF). Це означає, що всі неключові атрибути залежать від усього первинного ключа, та не мають транзитивних залежностей кожне поле залежить виключно від первинного ключа. Такий рівень нормалізації забезпечує логічну узгодженість, уникає надлишковості даних та робить базу більш зручною для обслуговування й масштабування.

Таке структурування дозволяє досягти логічної цілісності, запобігти дублюванню даних та забезпечити ефективність доступу до інформації при зростанні навантаження. Реалізація сесій у вигляді окремої таблиці відкриває простір для впровадження складніших механізмів безпеки, таких як багатофакторна аутентифікація, обмеження за часом дії, управління активними токенами тощо.

У підсумку, обрана модель бази даних не лише відповідає вимогам безпеки та нормалізації, а й гнучка до майбутнього масштабування додавання нових сутностей, пов’язаних з ролями, історією активності або логуванням, не вимагатиме перегляду існуючої логіки зберігання даних.

1.2.3 Реалізація механізму автентифікації та авторизації у застосунку

У сучасному веб-розробленні впровадження ефективного та захищеного механізму автентифікації й авторизації є основою побудови будь-якої інформаційної системи, пов’язаної з обробкою персональних даних, управлінням доступом та забезпеченням безпеки.

У межах розробленого веб-застосунку для аутентифікації клієнтів інтернет-магазину було реалізовано модуль, який базується на принципах REST-архітектури, використанні JWT (JSON Web Token), системи оновлення токенів та хешування паролів з урахуванням сучасних вимог безпеки.

Функціональність автентифікації в застосунку реалізована через сервіс AuthorizationService, який відповідає за обробку запитів, що надходять із контролера AuthorizationController. Цей сервіс взаємодіє з шаром доступу до даних за допомогою репозиторіїв користувачів і сесій, що забезпечує чітке розмежування обов’язків та дотримання принципів багаторівневої архітектури.

Користувач має змогу виконувати дві основні дії: зареєструвати новий обліковий запис через метод Register або здійснити вхід у систему з подальшим отриманням токена доступу за допомогою методу Login. Ці операції охоплюють базову логіку автентифікації та формують ядро взаємодії клієнта.

При успішній автентифікації система генерує пару токенів: JWT-токен доступу та refresh-токен, що зберігається у базі даних у таблиці UserSessions. Механізм дозволяє користувачу залишатися в системі протягом певного часу без повторної авторизації.

Реєстрація користувача реалізована в методі Register класу AuthorizationService. Спочатку перевіряється, чи існує користувач із такою ж електронною адресою (username), після чого відбувається створення нового користувача з хешуванням пароля.

```
public async Task<TokenDto> Register(UserCreationDto dto)
{
    var normalizedUsername = dto.Username.ToUpper();
    var existing = await
_repository.GetUserByNormalizedUsername(normalizedUsername);
    if (existing != null)
        throw new Exception("User already exists");

    var user = new User
    {
        Username = dto.Username,
        NormalizedUsername = normalizedUsername,
        ClientSecret = dto.Password.Hash(),
        RegistrationDate = DateTime.UtcNow,
        Uuid = Guid.NewGuid()
    };

    await _repository.CreateUser(user);
    return await GenerateTokens(user);
}
```

Ключова роль тут належить розширенню PasswordExtension.Hash(), яке перетворює введений пароль у захищений хеш. Якщо дивитись уважно, навіть у разі витоку даних, паролі залишаються недоступними для зловмисників.

Метод Login відповідає за автентифікацію користувача. Він перевіряє наявність користувача за нормалізованим ім'ям, а потім порівнює збережений хеш із наданим паролем.

```
public async Task<TokenDto> Login(UserAuthorizationDto dto)
{
    var user = await
_repository.GetUserByNormalizedUsername(dto.Username.ToUpper());
    if (user == null || !dto.Password.Verify(user.ClientSecret))
```

```
            throw new Exception("Invalid credentials");

        return await GenerateTokens(user);
    }
```

Як видно з коду, перевірка пароля виконується методом Verify, який порівнює збережений хеш і введений пароль. У разі успішного входу викликається метод GenerateTokens, який видає два токени – AccessToken та RefreshToken, які необхідні для доступу до ресурсів.

Метод GenerateTokens генерує JWT-токен із вбудованими клеймами (claims) та зберігає refresh-токен у базі даних. Як видно з коду, процес перевірки пароля здійснюється за допомогою методу Verify, який порівнює введений користувачем пароль зі збереженим хешем у базі даних. Якщо перевірка проходить успішно, система викликає метод GenerateTokens, що відповідає за видачу нової пари токенів. Зокрема, GenerateTokens створює JWT-токен із вбудованими клеймами (claims) та записує відповідний refresh-токен до бази даних для подальшої перевірки та оновлення.

```
    private async Task<TokenDto> GenerateTokens(User user)
    {
        var accessToken = _tokenService.GenerateJwt(user);

        var session = new UserSession
        {
            UserId = user.UserId,
            RefreshToken = Guid.NewGuid().ToString(),
            JwtId = accessToken.JwtId,
            CreatedAt = DateTime.UtcNow,
            RefreshTokenExpiration = DateTime.UtcNow.AddDays(7),
            Redeemed = false
        };

        await _repository.CreateSession(session);

        return new TokenDto
        {
            Token = accessToken.Token,
            Expires = accessToken.Expires,
            RefreshToken = session.RefreshToken
        };
    }
```

JWT-токен створюється з урахуванням унікального ідентифікатора JwtId, а сам refresh-токен зберігається разом із мітками часу в базі. Це дає змогу реалізувати механізм продовження сесії та відкликання токена.

Коли токен доступу (JWT) завершує свою дію, користувач може надіслати запит на оновлення, використовуючи refresh-токен. Метод Refresh перевіряє

дійсність токена та створює нову пару токенів:

```
public async Task<TokenDto> Refresh(RefreshTokenDto dto)
{
    var session = await _repository.GetSessionByRefreshToken(dto.RefreshToken);
    if (session == null || session.Redeemed || session.RefreshTokenExpiration <
DateTime.UtcNow)
        throw new Exception("Invalid refresh token");

    session.Redeemed = true;
    await _repository.UpdateSession(session);
    var user = await _repository.GetUserById(session.UserId);
    return await GenerateTokens(user);
}
```

Вбудовавши механізми змінній Redeemed забезпечується одноразове використання токена, що є сучасною практикою для запобігання повторним запитам з компрометованим токеном. Контролери веб-застосунку позначаються атрибутами [Authorize], що вказує на необхідність авторизованого доступу. Фільтр автоматично зчитує JWT-токен із заголовка запиту, перевіряє його підпис і строк дії. Сервіс ITokenService використовує налаштування токену з конфігурації:

```
public JwtDto GenerateJwt(User user)
{
    var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_jwtSettings.Secret));
    var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);
    var expires = DateTime.UtcNow.AddMinutes(_jwtSettings.TokenLifetimeMinutes);

    var token = new JwtSecurityToken(
        claims: new[] {
            new Claim(JwtRegisteredClaimNames.Sub, user.UserId.ToString()),
            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
        },
        expires: expires,
        signingCredentials: creds);

    return new JwtDto
    {
        Token = new JwtSecurityTokenHandler().WriteToken(token),
        Expires = expires,
        JwtId = token.Id
    };
}
```

Архітектурно модуль автентифікації спроєктований як ізольований сервіс, що має чітко визначену зону відповідальності. Такий підхід дозволяє легко масштабувати, розширювати й обслуговувати функціонал автентифікації незалежно від інших частин застосунку. Крім того, завдяки чіткій структурі цей модуль добре підходить для модульного тестування, що сприяє підвищенню надійності системи.

Однією з ключових переваг є використання інтерфейсів, таких як

IAuthorizationService та ITokenService. Це забезпечує слабке зв’язування між компонентами та дозволяє за потреби впроваджувати альтернативні реалізації, наприклад, перейти на протокол OAuth 2.0 або додати інтеграцію з зовнішніми провайдерами ідентифікації, такими як Google чи Microsoft.

Безпека реалізована на кількох рівнях. Паролі користувачів проходять хешування перед збереженням, що унеможливлює компрометацію у разі витоку бази. Використання одноразових refresh-токенів і обмежений термін дії JWT-токенів зменшують вікно для потенційних атак. Окрему увагу приділено перевірці цифрового підпису кожного токена, давая можливості відкликання refresh-токенів через логічне поле Redeemed, яке відслідковує їхній статус.

Розробка, у межах проєкту реалізовано повноціннею, захищеною та масштабованою механізмами автентифікації і авторизації на основі токенів, що відповідає сучасним стандартам веб-безпеки. Цей модуль може бути легко використаний у будь-якому подібному застосунку та розширений для підтримки авторизації за ролями, інтеграції з BankID або Дія.Sign, при запровадженні більш складних стратегій контролю доступу (RBAC, Claims-based, Policy-based). Повний код сервісу авторизації знаходиться в додатку А.

1.2.4 Обробка JWT-токенів та захист захищених маршрутів

Однією з ключових складових реалізованого механізму автентифікації є логіка роботи з токенами, зокрема з JWT (JSON Web Token). Цей тип токенів активно використовується у сучасних веб-додатках як легкий, масштабований і безпечний спосіб ідентифікації користувача протягом сесії.

Після успішної реєстрації або авторизації сервер формує відповідь, що містить два токени, необхідні для подальшої роботи клієнта із захищеними ресурсами. Основним із них є accessToken токен доступу, який має короткий строк дії, зазвичай обмежений однією годиною. Його використовують для автентифікації запитів до API протягом активної сесії.

Другим є refreshToken, який дозволяє поновити сесію без необхідності повторного введення пароля. Завдяки цьому користувач може залишатися авторизованим протягом тривалого часу, навіть після завершення строку дії

accessToken’a, при цьому зберігаючи зручність і безпеку взаємодії із системою. Це особливо зручно для користувачів, які часто повертаються до застосунку, оскільки дозволяє уникнути повторної авторизації під час кожного відвідування.

Токени мають наступну структуру:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiIxMjM0NSIsIm5iZiI6MTY3ODU5NzYwMC
wiZXhwIjoxNjc4NjAxMjAwfQ.1UdSpmHFkW4z9X_R_zFVqBg3EYwM3vymrPy5P2DbU9s
```

На стороні клієнта accessToken зберігається в localStorage, а під час кожного запиту до захищеного маршруту додається до заголовків:

```
fetch('/api/products', {
  headers: {
    'Authorization': `Bearer ${token}`
  }
});
```

На сервері використовується проміжне програмне забезпечення (middleware),

яке перевіряє дійсність токена:

```
public class JwtMiddleware
{
    private readonly RequestDelegate _next;
    private readonly IConfiguration _configuration;

    public JwtMiddleware(RequestDelegate next, IConfiguration configuration)
    {
        _next = next;
        _configuration = configuration;
    }

    public async Task Invoke(HttpContext context, IUserService userService)
    {
        var token =
context.Request.Headers["Authorization"].FirstOrDefault()?.Split(" ").Last();

        if (token != null)
        {
            try
            {
                var principal = JwtTokenHelper.ValidateToken(token,
_configuration);
                context.User = principal;
            }

            catch (SecurityTokenException)
            {
                context.Response.StatusCode = 401;
                await context.Response.WriteAsync("Invalid token");
                return;
            }
        }

        await _next(context);
    }
}
```

У разі відсутності або недійсності токена користувач отримує відповідь 401 Unauthorized. Це запобігає несанкціонованому доступу до приватних API. Даний механізм забезпечує базовий рівень контролю доступу.

Якщо accessToken втрачає чинність (закінчується строк дії), клієнт може надіслати запит на /auth/refresh разом із refreshToken, за допомогою якого може замінити прострочений токен на новий:

```
{
  "refreshToken": "8qafaX+MSfC4K54MSxFZC5q..."
}
```

Сервер перевіряє токен у базі даних (таблиця UserSessions) і, якщо він не прострочений і не використаний раніше (Redeemed = false), видає новий accessToken і оновлює сесію. Механізм гарантує, що кожен refreshToken можна використати лише один раз, що істотно знижує ризик зловживань у разі його компрометації.

```
CREATE TABLE [UserSessions] (
  [UserSessionId] INT PRIMARY KEY IDENTITY,
  [UserId] UNIQUEIDENTIFIER NOT NULL,
  [RefreshToken] NVARCHAR(250) NOT NULL,
  [JwtId] NVARCHAR(36) NOT NULL,
  [RefreshTokenExpiration] DATETIME2 NOT NULL,
  [Redeemed] BIT NOT NULL DEFAULT 0,
  FOREIGN KEY (UserId) REFERENCES Users(UserId)
);
```

У системі автентифікації refreshToken обробляється з особливою обережністю для підвищення безпеки. Він зберігається виключно на сервері, на відміну від accessToken, який може передаватися клієнту. Такий підхід значно знижує ризики, пов’язані з XSS-атаками, оскільки зловмисник не зможе дістати refreshToken навіть у разі отримання доступу до браузера користувача.

JWT-токени, які видаються користувачу, мають цифровий підпис, сформований на основі секретного ключа. Це гарантує їхню цілісність і автентичність без знання ключа підробити токен неможливо. Можемо впевненно, система може довіряти кожному токену, що надійшов, лише у разі успішної верифікації його підпису.

Кожен refreshToken має обмежений строк дії, після завершення якого він автоматично втрачає чинність. До того ж реалізований механізм одноразового використання щойно токен був використаний для оновлення сесії, його статус

змінюється на Redeemed = true, що унеможливлює повторне застосування. Це забезпечує додатковий рівень захисту від повторного використання скомпрометованих токенів.

Реалізований механізм на основі JWT є ефективним, зручним і захищеним способом автентифікації користувачів. Він дозволяє масштабувати систему, інтегрувати мобільні або SPA-клієнти та мінімізувати навантаження на сервер при повторних перевірках прав доступу. Поділ обов'язків між accessToken і refreshToken забезпечує баланс між зручністю користування і безпекою, а чітке логічне розділення на шари контролю доступу дозволяє легко розширювати систему в майбутньому. Система залишається гнучкою, придатною до розвитку та відповідає сучасним вимогам до безпечної авторизації в веб-застосунках. Це рішення добре масштабується як для невеликих проєктів, так і для складних корпоративних систем з великою кількістю користувачів і клієнтських додатків.

1.3 Аналіз стабільності функціонування шляхом ручного тестування

1.3.1 Тестування системи авторизації

Одним із ключових елементів будь-якого веб-застосунку, пов'язаного з доступом користувачів, є модуль реєстрації та авторизації. В рамках дипломного проєкту було реалізовано та протестовано повний цикл взаємодії користувача з інтерфейсом автентифікації: від створення облікового запису до перевірки вхідних даних і подальшої взаємодії з системою. Тестування здійснювалося вручну з метою підтвердження правильності обробки даних, коректності відображення помилок та безпеки операцій.

Мета тестування полягає в тому, щоб переконатися в правильності та надійності роботи механізмів автентифікації. Одним із ключових завдань є перевірка, що користувач може успішно зареєструвати новий обліковий запис лише за умови, що обране ім'я є унікальним. Це гарантує відсутність конфліктів і дозволяє системі точно ідентифікувати кожного користувача.

Особливу увагу під час тестування приділяють тому, що паролі зберігаються

виключно у хешованому вигляді. Крім того, перевіряється, що вхід у систему можливий лише за правильного поєднання імені користувача та пароля. Це критично важливо для запобігання несанкціонованому доступу до особистих даних.

Окремо тестується обробка помилок система повинна коректно реагувати на неправильні або некоректні введення, виводячи зрозумілі повідомлення для користувача. Також перевіряється стабільність роботи у випадках як правильних, так і неправильних даних, що дозволяє впевнено стверджувати про надійність і витривалість реалізованої логіки автентифікації.

Форма авторизації представлена двома обов’язковими полями "Ім’я користувача" та "Пароль", та кнопками керування. Інтерфейс побудований так як, що користувач одразу орієнтується в необхідних діях.

Також важливо перевірити, що логіка автентифікації працює незалежно від контексту виконання як при взаємодії з API, так і в межах можливого користувацького інтерфейсу. Тестування охоплює як позитивні сценарії (успішна реєстрація, правильний вхід), так і негативні спроби повторної реєстрації.

Не менш критичним є тестування обробки граничних значень, таких як надто короткі чи довгі імена користувачів, некоректні формати введення, відсутні поля тощо. Система має коректно реагувати на всі ці випадки, не допускаючи збоїв або непередбачуваної поведінки. Такий підхід дозволяє переконатися, що валідація введених даних реалізована належним чином.

Окремо слід протестувати надійність збереження даних після реєстрації – переконатися, що нові облікові записи зберігаються у базі даних правильно, включно з хешованим паролем, унікальним ідентифікатором та коректною датою створення. Це дозволяє гарантувати цілісність користувацьких записів і правильність роботи усіх пов’язаних механізмів у довготривалому використанні.

Базу даних можна перевірити за допомогою виконання SQL-запиту. Необхідно переконатися, що хеш-функція відпрацьовує правильно та паролі зберігаються в хешованому вигляді.

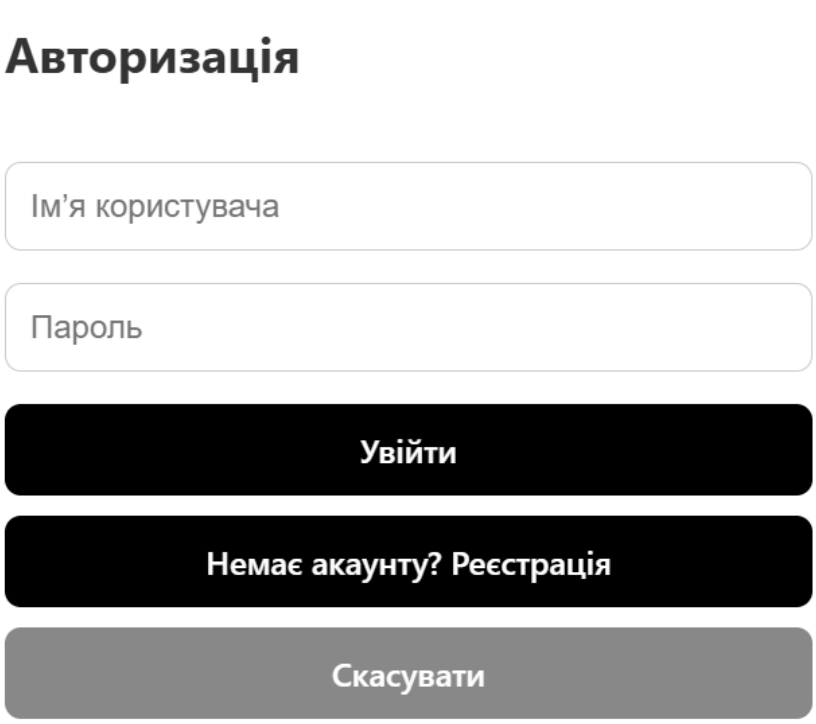


Рисунок 1.9. Приклад форми авторизації

При натисканні кнопки «Немає акаунту» користувач переходить на форму, де необхідно ввести ім’я користувача, пароль та підтвердити пароль. Без введення форма не активує кнопку реєстрації, що підтверджує наявність клієнтської валідації.

Коли користувач натискає кнопку «Немає акаунту? Реєстрація», він потрапляє на форму створення нового облікового запису. У цій формі потрібно вказати ім’я користувача, ввести пароль і повторити його для підтвердження. Такий підхід допомагає уникнути помилок при введенні, зокрема випадкових розбіжностей у паролі.

Форма має вбудовану клієнтську валідацію кнопка реєстрації залишається неактивною, доки всі поля не будуть заповнені. Така реалізація базового рівня перевірки на стороні клієнта, що підвищує зручність користування та зменшує кількість некоректних запитів до сервера.

Сервер також має вбудовану валідацію. Таким чином, навіть якщо потенційному користувачу вдасться оминути клієнтську перевірку або він викличе API напряму, то йому буде відмовлено в реєстрації.

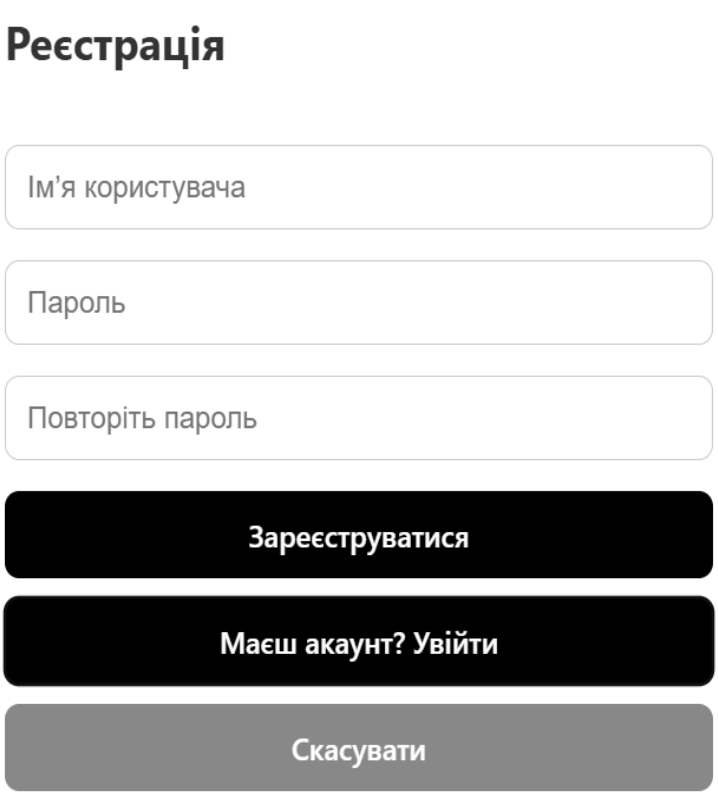


Рисунок 1.10. Вид системи реєстрації

Після заповнення полів, наприклад:

- Ім’я користувача: TestUser1234
- Пароль: ********
- Повторити пароль: ********

користувач натискає кнопку «Зареєструватися». Відправка даних здійснюється методом POST, після чого контролер на сервері обробляє запит і перевіряє, чи не існує вже такого користувача.

Як можна побачити у вкладці Network засобів розробника браузера, після успішної авторизації сервер надсилає клієнту відповідь у форматі JSON, що містить основні елементи для керування сесією. Одним із них є accessToken де JWT (JSON Web Token), який використовується для автентифікації подальших запитів до захищених ресурсів. Він містить у собі вбудовану інформацію (claims) про користувача та підписаний за допомогою секретного ключа, що унеможливлює його підробку.

Разом із accessToken клієнт також отримує refreshToken це окремий токен, який зберігається на сервері та використовується для поновлення accessToken після

завершення його дії без повторного введення пароля. Поле expiresIn вказує на час життя accessToken у секундах; у даному випадку значення становить 3600 секунд, що дорівнює одній годині. Це дозволяє системі точно регулювати термін дії токена й забезпечувати баланс між безпекою та зручністю для користувача.

Система автентифікації повинна забезпечувати баланс між безпекою та зручністю для користувача. Як можна побачити у вкладці Network засобів розробника браузера, після успішної авторизації сервер надсилає клієнту відповідь у форматі JSON, яка містить ключові елементи для керування сесією користувача.

Одним із основних компонентів цієї відповіді є accessToken – токен типу JWT (JSON Web Token). Він використовується для автентифікації наступних запитів до захищених ресурсів. JWT містить у собі вбудовані claims – дані про користувача, а також цифровий підпис, створений на основі секретного ключа. Це забезпечує цілісність токена та захист від підробки чи модифікації.

Окрім accessToken, клієнт отримує також refreshToken цей токен, який не передається з кожним запитом, а зберігається на сервері та використовується для оновлення accessToken після завершення його строку дії. У відповіді також присутнє поле expiresIn, що вказує на тривалість життя accessToken у секундах. У наведеному випадку воно дорівнює 3600 секунд, тобто одній годині. Це дозволяє системі гнучко балансувати між безпекою доступу та зручністю для користувача. Такий токен, на відміну від токена доступу, не несе в собі ніякого корисного навантаження, адже потрібен лише для оновлення основного токену – він генерується за допомогою генератора випадкових чисел, які потім перетворюються в строку. Завдяки такому підходу неможливо відстежити закономірність між токеном доступу та токеном оновлення.

Завдяки такій структурі токенів система може ефективно контролювати сеанси користувачів, запобігати несанкціонованому доступу та реалізовувати механізми відкликання доступу в разі підозрілої активності. Наприклад, якщо refreshToken було скомпрометовано або виявлено спробу повторного використання, система може відмітити його як недійсний (через поле Redeemed) і заблокувати подальші запити.

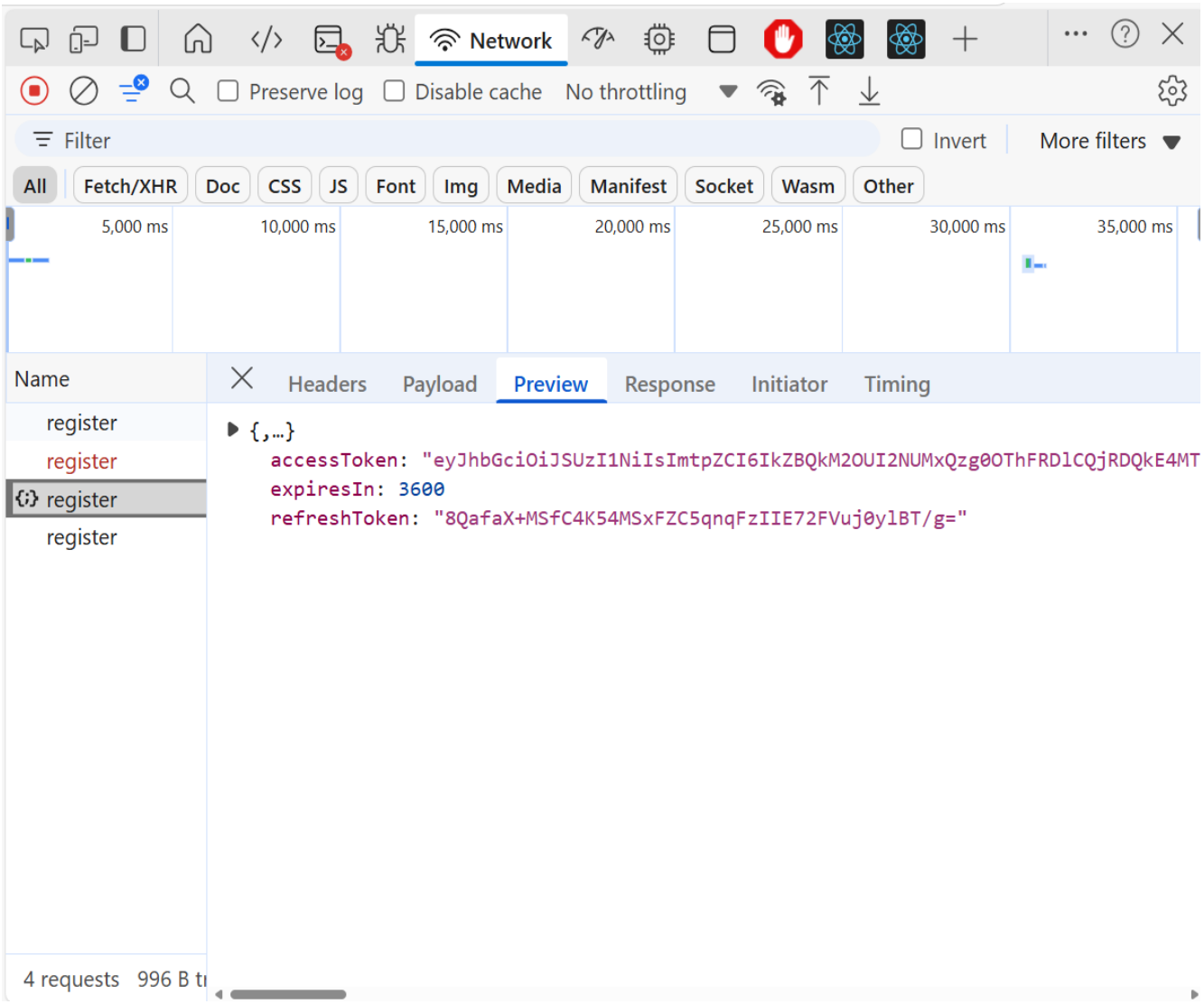


Рисунок 1.11. Приклад токену та даних з серверу

Це свідчить про коректну роботу механізму авторизації через токени. Усі поля повертаються в JSON-форматі з відповідним HTTP-заголовком Content-Type: application/json.

У разі успіху створюється новий обліковий запис, пароль зберігається у базі даних в хешованому вигляді за допомогою функції PasswordExtension.Hash(). Використання хешування дозволяє уникнути зберігання конфіденційних даних у відкритому вигляді. Це підтверджує, що система дотримується сучасних стандартів безпеки під час обробки облікових даних. Усі відповіді сервер формує у форматі JSON, супроводжуючи їх коректним заголовком Content-Type: application/json, що дозволяє клієнту правильно інтерпретувати отримані дані. Завдяки використанню

функції PasswordExtension.Hash() паролі користувачів зберігаються у захищеному

вигляді, що унеможливлює їхнє пряме зчитування навіть у разі доступу до бази даних.

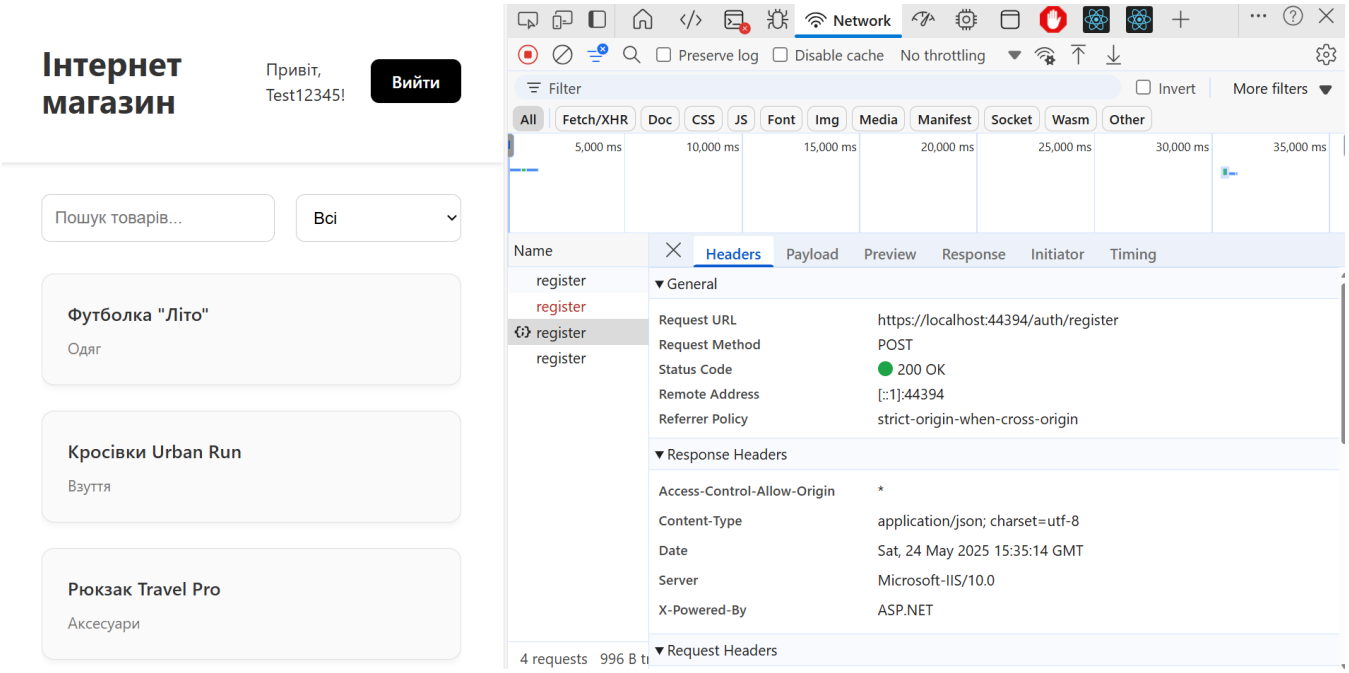


Рисунок 1.12. Приклад успішної авторизації користувача

Якщо ввести ім’я користувача, яке вже було використано, API повертає помилку 400, а система виводить повідомлення "Не вдалося зареєструватись". Це означає, що серверна логіка перевірки унікальності імені користовача спрацьовує належним чином.

У процесі реєстрації критично важливо перевіряти унікальність імені користувача. Без цього система може створити декілька облікових записів із однаковими іменами, що призведе до плутанини, порушення цілісності даних та потенційних проблем безпеки. Реалізована перевірка не лише захищає від дублювання, а й покращує загальний досвід користувача.

Крім цього, така поведінка є яскравим підтвердженням того, що бекенд застосунку ефективно та коректно обробляє виключні ситуації, не допускаючи порушення цілісності даних у базі. Це свідчить про високий рівень відповідності архітектури застосунку сучасним стандартам розробки, зокрема принципам fail-safe програмування. Такий підхід передбачає, що система не просто відмовляється виконувати некоректний запит, а робить це контрольовано, передбачувано і без негативних наслідків для її стану або функціонування. Крім технічної стабільності, важливим аспектом є й те, що система надає зрозуміле пояснення причини відмови,

що допомагає користувачеві або розробнику швидко зрозуміти суть проблеми та скоригувати свої дії. Це значно покращує загальний користувацький досвід, зменшує кількість помилок під час взаємодії з інтерфейсом та сприяє швидшому вирішенню можливих труднощів.

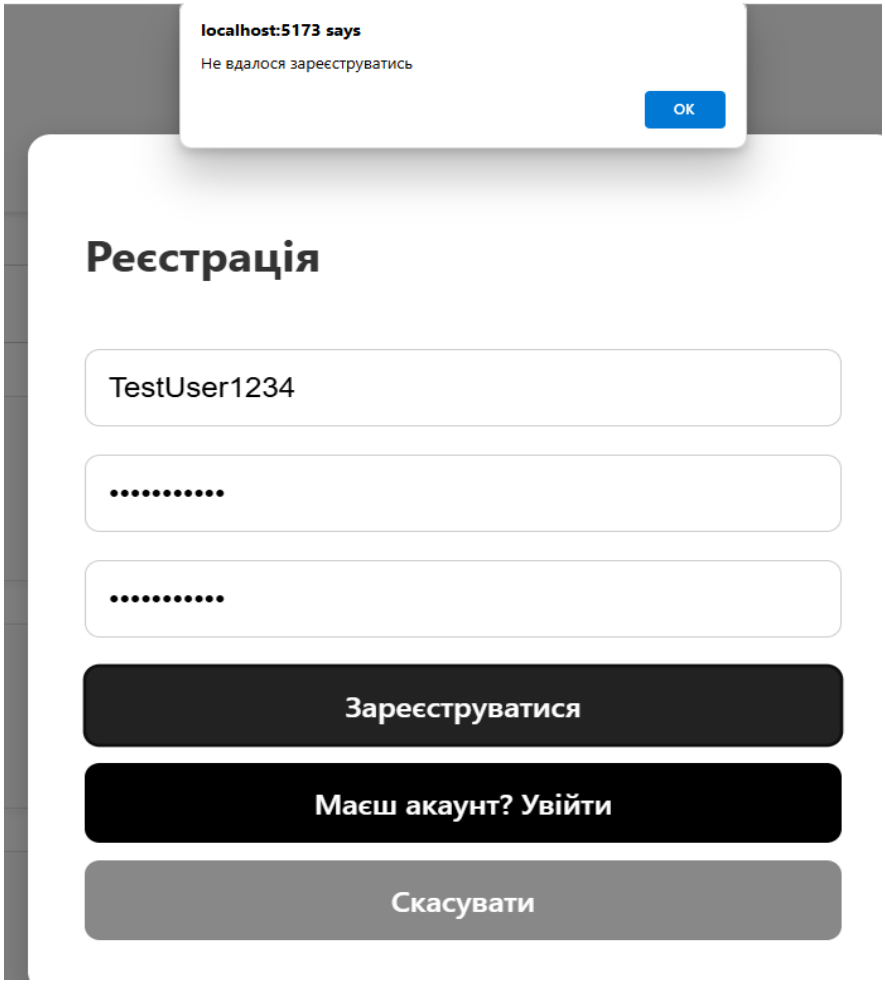


Рисунок 1.13. Повідомлення о проблемах з авторизацію користувача

Після успішної реєстрації користувач переходить на сторінку входу. Введення правильного імені користувача та пароля призводить до успішної авторизації.

При цьому запит потрапляє до методу Login сервісу авторизації, де здійснюється пошук користувача та перевірка пароля через Password.Verify(). Якщо дані правильні, видається токен доступу (JWT) та refresh-токен, які клієнт зберігає для подальшого використання. У разі невдалої спроби входу система повертає відповідь з відповідним кодом помилки, що дозволяє клієнту відобразити повідомлення про неправильні облікові дані.

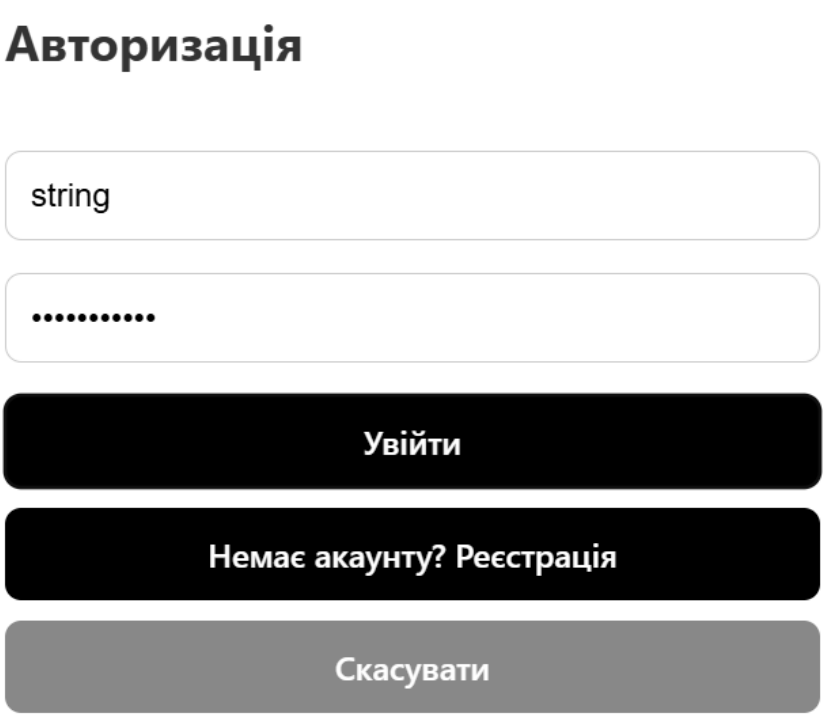


Рисунок 1.14. Приклад невалідних даних

Це свідчить про наявність ефективної перевірки облікових даних і виняткових ситуацій на сервері. Повідомлення про помилки дозволяють користувачу орієнтуватись у причинах відмови. Запит не проходить валідацію та не видає токени, як очікується. Даний підхід забезпечує додатковий рівень безпеки та дозволяє оперативно реагувати на загрози в режимі реального часу, також дає більшу валідацію для нового алгоритму.

У ході ручного тестування було підтверджено, що всі основні форми в застосунку – як для реєстрації, так і для входу – доступні та відображаються коректно. Вони працюють стабільно на різних етапах взаємодії з користувачем, забезпечуючи належний рівень зручності та передбачуваної поведінки інтерфейсу.

Також було зафіксовано наявність як клієнтської, так і серверної валідації. Клієнтська валідація не дозволяє надіслати порожні або некоректні поля, тоді як серверна забезпечує додаткову перевірку даних після надсилання, зокрема перевірку унікальності імені користувача чи правильності пароля. Це гарантує надійність обробки введеної інформації на всіх рівнях.

Було протестовано всі ключові сценарії взаємодії: успішна та невдала реєстрація, авторизація з правильними та неправильними даними, та спроби

повторної реєстрації вже існуючого користувача. Усі ці випадки обробляються системою належним чином із відповідними повідомленнями для користувача.

Окрім функціональної частини, перевірено відповідність усіх HTTP-запитів очікуваним REST-протоколам. Сервер коректно повертає JWT і refresh токени у відповіді, при дотриманні сучасних стандартів безпеки, зокрема хешування паролів, використання CORS, відповідей із кодом 200 та правильного формату заголовків (JSON). Це свідчить про відповідальний підхід до безпеки та якості реалізації.

Можемо впевнено наголосити, що реалізований функціонал реєстрації та авторизації забезпечує стабільну, безпечну та інтуїтивну взаємодію з користувачем, що є критично важливим для будь-якого веб-застосунку, що працює з персональними даними.

1.3.2 Відображення товарів та фільтрація у внутрішній частині веб-застосунку

Після успішної авторизації користувач автоматично перенаправляється до основної частини веб-застосунку сторінки, яка відображає доступні товари у форматі карток. Цей розділ умовно можна вважати внутрішньою частиною системи, доступною лише для автентифікованих користувачів.

Завдяки впровадженню цього механізму забезпечується чітке та надійне розмежування між публічним і захищеним доступом до ресурсів застосунку. Усі запити, що надходять до внутрішніх маршрутів, проходять обов’язкову перевірку на наявність та дійсність accessToken. Це дозволяє з високим ступенем впевненості гарантувати, що доступ до функціональності, яка стосується персоналізованих або чутливих даних, мають виключно авторизовані користувачі.

Подібна перевірка є фундаментальним елементом сучасної системи контролю доступу та відповідає загальноприйнятим стандартам інформаційної безпеки. Вона забезпечує захист від несанкціонованого втручання, спроб обійти систему автентифікації або отримати доступ до конфіденційних даних без відповідних прав.

У практичному вимірі це означає, що лише користувач, який успішно

пройшов процес автентифікації, може виконувати такі дії, як перегляд детальної інформації про товари, додавання їх до кошика, оформлення замовлення чи керування особистими даними. Водночас усі публічні маршрути — наприклад, базовий перегляд каталогу або загальна інформація про магазин — залишаються доступними для всіх відвідувачів без потреби в авторизації.

Цей підхід не лише підвищує безпеку, а й покращує користувацький досвід, оскільки дозволяє уникнути помилок або "битих" сторінок, забезпечуючи логічну і передбачувану поведінку інтерфейсу при різних сценаріях взаємодії. На рис. 1.15 зображена перша сторінка магазину.

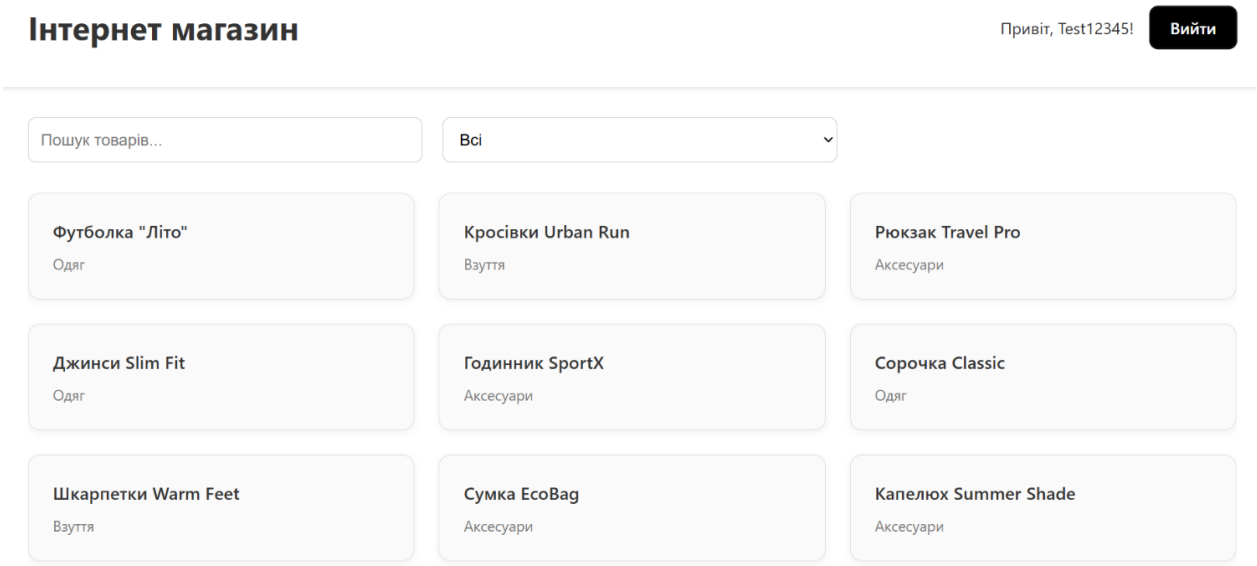


Рисунок 1.15. Перша сторінка магазину

Інтерфейс спроєктований так, щоб забезпечити зручну навігацію, швидкий пошук і можливість фільтрації продуктів за категоріями.

У верхній частині сторінки розташоване привітання користувача з його ім’ям, отриманим із JWT-токена, та кнопка «Вийти», яка завершує сесію. Це наочно демонструє, що система зчитує збережені облікові дані для індивідуалізації інтерфейсу.

Кожен товар подається у вигляді окремої картки, що містить назву, категорію та мінімалістичний дизайн для зручності сприйняття. Стилі адаптовані під мобільні та десктопні пристрої, що підтверджує адаптивність верстки.

Інтерфейс підтримує дві основні взаємодії користувача: фільтрацію товарів за категорією (наприклад, «Одяг», «Аксесуари», «Взуття») та текстовий пошук. Це

дозволяє користувачу швидко знайти необхідний товар серед широкого асортименту.

Крім зручної навігації, інтерфейс враховує потреби персоналізованої взаємодії, що підвищує довіру користувача до системи. Відображення імені користувача, зчитаного з JWT-токена, створює ефект персонального обслуговування, а наявність кнопки «Вийти» дозволяє легко завершити сесію, що важливо як для безпеки, так і для зручності.

Дизайн сторінки зосереджений на доступності: великі кнопки, чітка типографіка та логічна структура розміщення елементів створюють комфортне середовище для користувача. Має адаптивну верстку інтерфейса коректно відображається на різних розмірах екранів, а пошук і фільтрація працюють швидко, без перезавантаження сторінки, що забезпечує сучасний користувацький досвід.

На прикладі зображення 1.16 обрано категорію «Взуття». Відповідно, інтерфейс миттєво відфільтровує всі інші позиції, залишаючи лише ті, що належать до вибраної категорії.

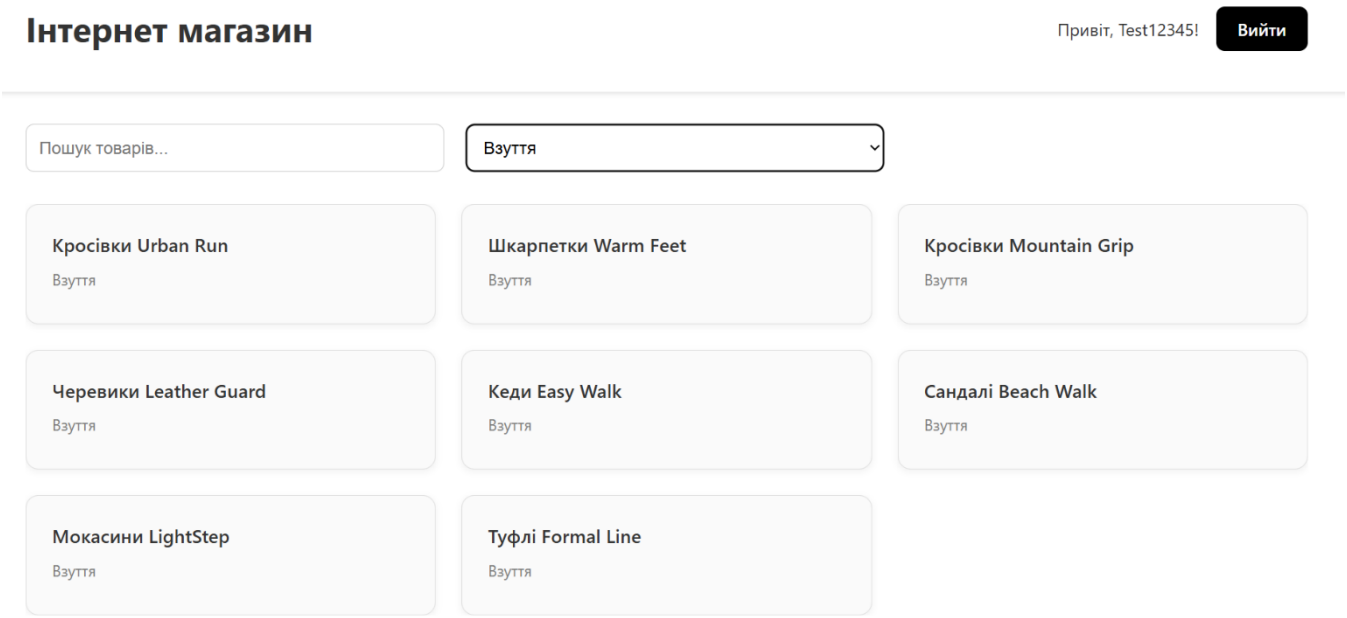


Рисунок 1.16. Приклад використання дроп-боксу для фільтрації товарів

Реалізована логіка передбачає динамічне оновлення списку товарів без перезавантаження сторінки. Також доступна функція пошуку за назвою або ключовим словом. У прикладі 1.17 здійснено пошук за запитом «Кро». Система

повернула лише ті товари, що містять цю частину слова у своїй назві.

Реалізована логіка динамічного оновлення контенту дозволяє користувачу взаємодіяти з каталогом товарів у режимі реального часу без потреби в повному перезавантаженні сторінки. Такий підхід не лише підвищує зручність, а й значно покращує швидкодію інтерфейсу, забезпечуючи плавний та безперервний користувацький досвід.

Коли користувач обирає певну категорію, наприклад, «Взуття», система одразу фільтрує каталог і миттєво відображає лише ті позиції, які відповідають цьому критерію. Має повну асинхронну обробку запитів, зміни в інтерфейсі відбуваються без затримок, що особливо важливо для великих каталогів із численними товарами.

Окрім фільтрації, доступна функція пошуку, яка реагує на введення тексту вже під час набору. Якщо, наприклад, користувач вводить фрагмент слова, як-от «Кро», система проводить пошук серед усіх назв товарів і повертає лише релевантні результати. Це дозволяє значно скоротити час на навігацію та пришвидшує доступ до потрібної інформації.

Обидві функції фільтрація та пошук працюють разом, дозволяючи комбінувати умови. Користувач може, наприклад, спершу вибрати категорію, а потім уточнити пошук за назвою в межах цієї категорії. Така інтерактивність підвищує ефективність користування застосунком і робить взаємодію з ним інтуїтивно зрозумілою.

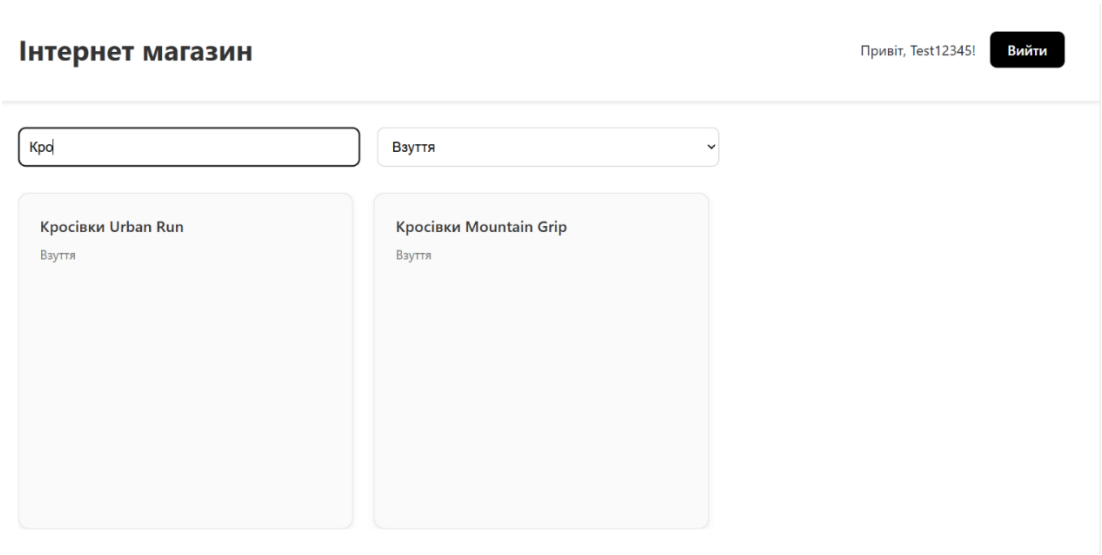


Рисунок 1.17. Фільтрація за допомогою вводу запиту

У результаті ручного тестування внутрішньої частини веб-застосунку підтверджено, що система успішно забезпечує індивідуалізований доступ до функціоналу після входу. Навігація є інтуїтивною, а функції пошуку та фільтрації працюють швидко й стабільно. Інтерфейс побудований з урахуванням UX-принципів, що значно підвищує зручність користування для кінцевого споживача. Реалізація логіки фільтрації на клієнтській стороні дозволила досягти високої швидкодії без навантаження на сервер.

Під час перевірки також слід звернути увагу на сам токен. За допомогою ресурсу jwt.ms дізнаємося приховані в токені Claims. Можна побачити, що він складається з трьох частин: заголовок з інформацією, частина з даними та підпис, в якості якого виступає приватний ключ сертифіката. На рис. 1.18 зображений приклад розшифровки токена одного з користувачів.

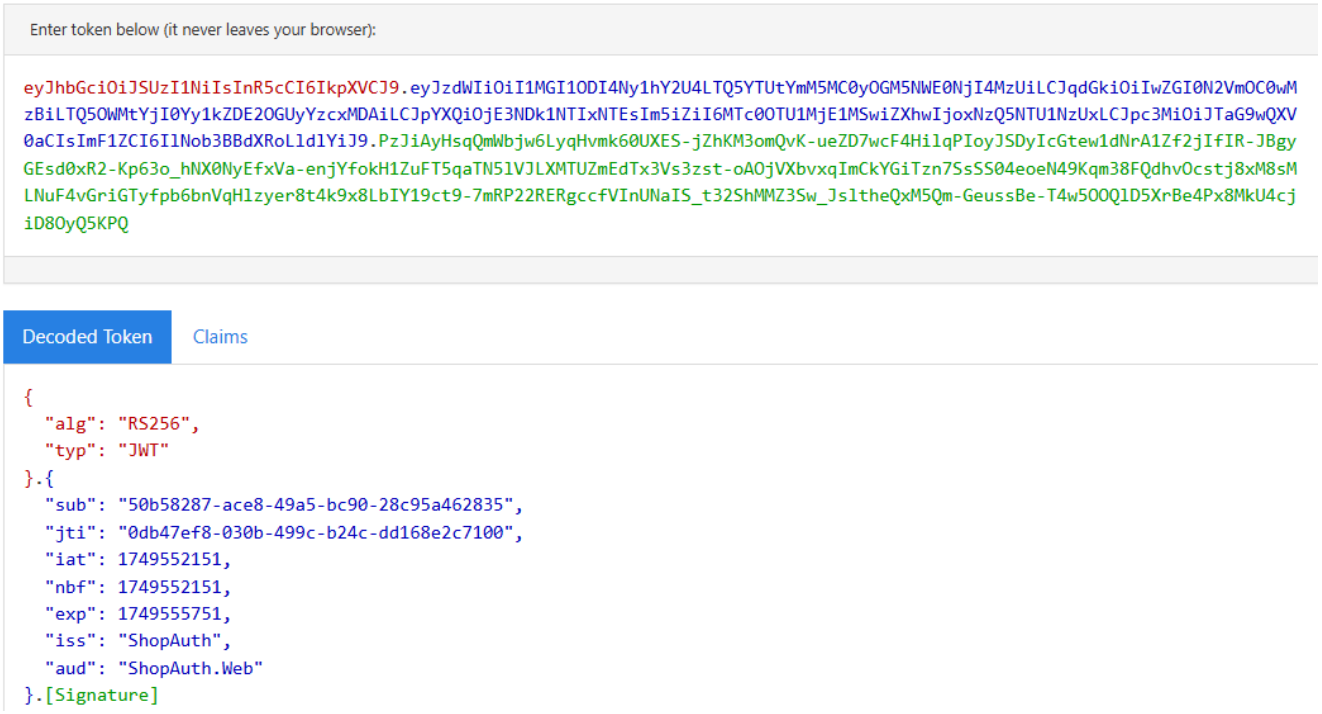


Рисунок 1.18. Приклад токену, що видається сервером

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

У дипломному проєкті створено систему авторизації для клієнтів інтернет-магазину з використанням сучасних технологій створення токенів авторизації за допомогою стандарту OAuth. Застосунок дозволяє забезпечити безпеку авторизованих користувачів за допомого методів шифрування та власно створених сертифікатів безпеки.

2.2 Визначення трудомісткості розробки ПЗ

Тривалість розробки програмного продукту залежить від його обсягу, складності, кваліфікації розробників і встановлених ринком термінів. Метод структурної аналогії дозволяє оцінити обсяг у тисячах умовних машинних команд на основі подібного програмного забезпечення.

Табл. 2.1 містить аналоги ПЗ з подібними функціями; обраний варіант виділено сірим.

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, умовних. машинних командах**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП введення інформації	1060 – 5750

Після вибору аналога з обсягом Vo (умовні машинні команди), трудомісткість визначається за табл. 2.2.

Таблиця 2.2. Обсяг ПП

Обсяг ПП, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262
4.00	283
5.00	306
6.00	330
7.00	357

Продовження таблиці 2.1

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
8.00	385
9.00	414
10.00	445

На основі отриманого значення за довідником визначають укрупнену норму часу, скориговану коефіцієнтом $К_к$ = 0,7–0,8 для умов розробки на комп'ютері:

$$T_{ap} = 229 \text{ x } 0,8 = 183,2 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість визначається для кожного етапу окремо, з урахуванням складності, новизни та використання стандартних модулів, за відповідними формулами:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.2)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.3)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.4)$$

Для розрахунку використовуються такі коефіцієнти:

- Li – частка і-го етапу (табл. 2.3);
- Кн – коефіцієнт новизни (табл. 2.4);
- Кт – коефіцієнт використання типових програм (табл. 2.5).

Наш варіант виділено сірим.

Таблиця 2.3. Питомі коефіцієнти трудомісткості стадії у загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Таблиця 2.4. Значення коефіцієнта новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово новий ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8

Продовження таблці 2.4

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
В	ПП, що має аналог	0,7

Таблиця 2.5. Значення коефіцієнта використання типових програм

Ступінь охоплення реалізованих функцій розробленого ПП типовими програмами, %	**Значення $К_т$**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Тепер розраховуємо трудомісткість для всіх етапів і зводимо у табл. 2.6:

Трудомісткість технічного завдання:

$$T_{тз}=Ta*L_1*K_н=244,8*0,12*0,8= 23,50 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки технічного проєкту:

$$T_{тп}=Ta*L_2*K_н =183,2*0,15*0,8 = 21,98 \text{ (люд/годин)} \quad (2.3)$$

Трудомісткість розробки робочого проєкту:

$$T_{рп}=Ta*L_3*K_н*K_т=183,2*0,58*0,8*0,7= 59,50 \text{ (люд/годин)} \quad (2.4)$$

Для розрахунків визначили обсяг документації по етапах:

- технічне завдання $N_{тз}$=2 (стор);
- розробка ТП $N_{тп}$=43 (стор);
- розробка робочого проєкту $N_{рп}$=5(стор);
- пояснювальна записка відповідно $N_{пз}$=15 (стор).

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин**		
1.ТЗ	$T_{Ртз}$=17,58	$T_{кк}=0,7*N_{тз}$= 0,7*2 =1,4	$T_{нк}=0,15*N_{тз}$ =0,15*2=0,30
2.Розробка ТП	ТРтп=21,98	Ткк=0,7*Nтп= 0,7*43=30,1	Тнк=0,15*Nтп =0,15*43=6,45
3.Розробка РП	ТРрп= 59,50	Ткк=0,7*Nрп= 0,7*5=3,5	Тнк=0,15*Nрп= 0,15*5=0,75
4.Розробка ПЗ	Тпз=1,5*Nпз= 1,5*15 =22,5	Ткк=0,7*Nтз= 0,7*15=10,5	Тнк=0,15*Nпз= 0,15*15 =2,25

Продовження таблиці 2.6

Усього, в т.ч.:	176,51		
- на розробку	Тр=121,56		
- контроль керівника		Ткк=45,5	
- нормоконтроль			Тнк=9,45

2.3 Розрахунок ціни програмного продукту

Розраховуємо основну зарплату виконавців, матеріальні та загальні витрати на розробку ПП. Зарплата наведена в табл. 2.7. З 1 квітня 2024 мінімальна місячна зарплата – 8000 грн, погодинна ставка – 46 грн (згідно зі ст. 8 Закону про Держбюджет України).

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	121,56	55,00	6685,8
2.Контроль керівника	45,5	120,00	5460,00
3.Нормоконтроль	9,45	120,00	1134,00
Усього	-	-	Зо= 13279,8

Розрахуємо матеріальні витрати на розробку ПП та наведемо їх у табл. 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	280,0	280,0	280,0
Разом	-	$В_{мі}$ =280,0	$В_{мі}$ =280,0	$В_{мі}$ =280,0
Транспортно – заготівельні Витрати (10%)				$В_{тр\_з}$ = 0.1 * $В_{м1}$ = 0,1*280 =28,00
Усього				$В_м$=$В_{мі}$+$В_{тр\ з=}$ 308,00

За отриманими даними складена калькуляція планової собівартості ПП, наведена в табл. 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	308,00	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	13279,8	$З_о$ (див. табл. 2.7)
3.Додаткова заробітна плата	1327,98	$З_д = 0.1 * З_о = 13279,8*0,1$
4.Відрахування до єдиного фонду соціального внеску	3213,71	$В_{є.с.в.} = 0.22 * (З_о + З_д) = 0,22*$ (13279,8+1327,98)
5. Накладні витрати	5311,92	$В_{нак.} = 0.4 * З_о = 0.4 * 13279,8$
6. Повна собівартість	23441,41	$С_{пов} = В_м + З_о + З_д + В_{є.с.в.} + В_{нак.} =$ 308,00+13279,8+1327,98+3213,71+ 5311,92

Розмір прибутку розраховується за формулою:

$$П=(С_п*Р)/100 =(23441,41*10)/100=2344,14 \text{ грн.} \quad (2.5)$$

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна розраховується за формулою:

$$Ц_о=С_п+П=23441,41+2344,14=25785,55 \text{ грн.} \quad (2.6)$$

За отриманими даними, ціна реалізації ПП за формулою становить:

$$Ц_р=Ц_о+ПДВ=25785,55+25785,55*0.2=30942,66 \text{ грн.} \quad (2.7)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

3.1 Резюме

Забезпечення безпечних умов праці є не лише важливим чинником створення комфортного середовища для працівників, а насамперед запорукою збереження їхнього здоров'я та працездатності, що безпосередньо впливає на продуктивність і прибутковість підприємства. Досягти високого рівня безпеки можливо лише за умови неухильного дотримання вимог трудового законодавства, державних стандартів України, а також галузевих норм і правил, спрямованих на охорону здоров'я працівників.

У цьому розділі дипломного проєкту розглядаються умови праці програміста (оператора ПК), які повинні бути створені на підприємстві задля безпечної організації робочого процесу, потенційні небезпечні фактори, санітарні умови та інше під час розробки веб-системи.

3.2 Оцінювання загроз і негативних впливів на здоров'я оператора ПК під час розробки ПЗ

Небезпечні фактори це умови, які за певних обставин можуть призвести до виробничих травм або раптового погіршення здоров'я працівника. Якщо ж вплив чинника має поступовий характер і з часом спричиняє хронічні захворювання або зниження працездатності, його класифікують як шкідливий. Водночас тривалий або інтенсивний вплив шкідливих факторів може перетворити їх на небезпечні, особливо за відсутності відповідних заходів безпеки.

Під час роботи за ПК працівник піддається дії таких небезпечних і шкідливих факторів:

- відхилення мікрокліматичних умов від нормативів;
- недостатній рівень освітлення робочого місця;
- імовірність ураження електричним струмом;
- вплив статичної електрики;
- нераціональна організація робочого простору;
- інші можливі чинники небезпеки.

3.3 Нормативи до умов виробничого середовища

Згідно з вимогами Правил охорони праці при роботі з електронно-обчислювальними машинами, робоче місце користувача персонального комп'ютера повинно бути організоване так, щоб сприяти максимально ефективній і безпечній роботі, при цьому мінімізуючи можливий ризик впливу небезпечних факторів. Розглянемо основні вимоги до таких умов.

3.3.1 Вимоги до приміщення для роботи з ПК

У приміщеннях, де розміщені ПК, бажано орієнтувати вікна на північ або північний схід. Віконні отвори мають бути обладнані регульованими жалюзі або шторами, які забезпечують повне затемнення при необхідності. Освітлення в таких приміщеннях повинно відповідати нормам ДБН В.2.5-28-2018 «Природне і штучне освітлення» та включати як природне, так і штучне джерело світла.

Крім того, у виробничих приміщеннях слід передбачити спеціальні зони для відпочинку і психологічного розвантаження працівників.

Кожне робоче місце користувача ПК повинне мати щонайменше 6 м² площі та не менше 20 м³ об'єму повітря. Стіни приміщення мають бути пофарбовані матовою фарбою відповідно до встановлених санітарно-гігієнічних норм.

3.3.2 Освітлення

Робоча зона користувача ПК повинна мати комбіноване освітлення – із застосуванням як природного, так і штучного світла. Для загального освітлення рекомендується використовувати люмінесцентні лампи типу ЛД. Освітленість робочої поверхні має відповідати нормативам і становити в межах 300-500 люкс.

3.3.3 Шум

Під час виконання завдань, що вимагають зосередженості, рівень шуму в приміщенні не повинен перевищувати 50 дБ. Щоб зменшити вплив шуму та вібрацій, обладнання встановлюють на спеціальні амортизуючі підкладки. У випадках, коли джерелом шуму є стіни, їх обробляють звукоізоляційними матеріалами.

3.3.4 Мікроклімат

Відхилення від нормативних показників мікроклімату негативно позначається на працездатності, швидкості реакції працівника та може призводити до збільшення кількості помилок. Тому важливо забезпечити оптимальні умови в робочому приміщенні: температура повітря повинна знаходитись у межах 22-25 °С, відносна вологість – 40-60%, а швидкість руху повітря – 0,1-0,2 м/с.

Приміщення має бути обладнане системами опалення та кондиціювання, які забезпечують рівномірний розподіл тепла, належну вентиляцію та ефективне очищення повітря від пилу і шкідливих домішок.

3.3.5 Електробезпека

Проходження електричного струму через тіло людини може спричинити термічні опіки, електролітичні порушення та біологічні ураження організму. Щоб запобігти ураженням електричним струмом, необхідно дотримуватись таких вимог:

- чітко виконувати правила техніки безпеки та експлуатації електрообладнання;
- унеможливити доступ працівників до відкритих або неізольованих частин, що перебувають під небезпечною напругою;
- використовувати відповідні ізоляційні матеріали для електрозахисту;
- забезпечити надійне заземлення усіх металевих конструкцій, зокрема батарей опалення, водопровідних труб та кабелів із заземленим екраном. При цьому слід застосовувати діелектричні екрани або захисні сітки, щоб унеможливити випадковий контакт оператора персонального комп’.тера із джерелом струму.

3.3.6 Вимоги до організації робочого місця

Робочі місця слід облаштовувати таким чином, щоб у поле зору працівника не потрапляли відбивні поверхні, пряме світло від вікон або джерел освітлення. Відеотермінали рекомендується розташовувати під кутом 90-100° до вікон, щоб природне освітлення надходило збоку. Не допускається розміщення працівника

обличчям або спиною до вікна незалежно від типу освітлення.

Робочий стіл повинен мати можливість регулювання по висоті в діапазоні 680-800 мм і достатню ширину для комфортного доступу до всіх необхідних предметів. Оптимальні розміри: висота – 725 мм, ширина – 600-1400 мм, глибина – 800-1000 мм. Стілець має бути оснащений механізмами для налаштування висоти сидіння, кута нахилу та висоти спинки. Усі регулювання повинні бути простими у використанні, незалежними одне від одного та надійно фіксуватися.

Екран ПК рекомендується розташовувати під кутом близько +30° до вертикальної лінії погляду користувача для зручності візуального сприйняття. Клавіатура повинна знаходитись на столі на відстані 100-300 мм від переднього краю.

Уся організація робочого місця має відповідати ергономічним стандартам, з урахуванням взаємного розташування обладнання для забезпечення комфорту та продуктивності працівника. На рис. 3.1 представлено робоче місце і робоча поза користувача комп'ютера.

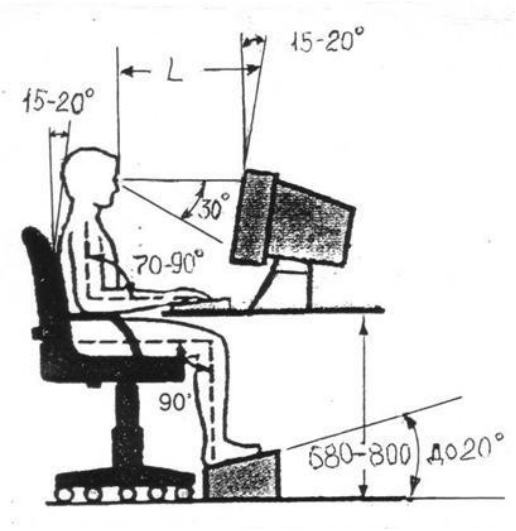


Рисунок 3.1. Робоче місце і робоча поза користувача комп'ютера

Робоче місце і робоча поза користувача комп'ютера включає:

1 – кут екрана; 2 – кут огляду (зору); 3 – відстань огляду; 4 – висота середини екрана; 5 – висота клавіатури; 6 – висота столу; 7 – відстань колін від столу; 8 – підставка для ніг; 9 – підставка для документів; 10 – положення рук; 11 – кут ліктів;

12 – спинка крісла; 13 – підлокітник; 14 – опора для попереку; 15 – кут колін; 16 – кут спинки крісла; 17 – висота сидіння

3.4 Пожежна безпека

Вимоги до пожежної безпеки приміщень, де експлуатуються електричні мережі та електронно-обчислювальні машини, визначаються стандартами ГОСТ 12.1.033-81 та ГОСТ 12.1.004-85. Такі приміщення повинні відповідати категорії Д за пожежною небезпекою, тобто бути виготовленими з негорючих матеріалів та не містити займистих речовин у нормальних умовах експлуатації.

Система пожежної безпеки включає комплекс заходів, до яких належать засоби попередження виникнення пожеж, елементи протипожежного захисту, організаційно-технічні рішення для зниження ризиків. До засобів захисту входять автоматизована пожежна сигналізація, наявність відповідних вогнегасників та забезпечення умов для швидкої та безпечної евакуації персоналу.

Для оперативного гасіння невеликих займання на початковій стадії використовуються первинні засоби пожежогасіння: порошкові та вуглекислотні вогнегасники, пожежні покривала з вогнетривких матеріалів, ящики з піском або ємності з водою. На рис. 3.2 представлено ілюстрацію типового пожежного щита з первинними засобами пожежогасіння.

Рисунок 3.2. Первинні засоби пожежогасіння

ВИСНОВКИ

У результаті виконання дипломного проєкту було розроблено програмний модуль аутентифікації клієнта, який може бути інтегрований до інтернет-магазину для забезпечення безпечного та зручного входу користувачів до системи. Реалізований функціонал включає реєстрацію нового користувача, вхід до особистого кабінету, валідацію облікових даних, зберігання інформації про сесію, має механізм безпечного виходу із системи.

Серверна частина модуля була реалізована з використанням сучасного фреймворку веб-програмування ASP.NET, що дозволяє ефективно обробляти запити та забезпечує надійну взаємодію з базою даних. Для зберігання облікових записів, хешування паролів та керування токенами автентифікації застосовано сучасні засоби безпеки, включаючи використання алгоритму bcrypt та JSON Web Token (JWT).

У процесі розробки було проведено аналіз сучасних підходів до аутентифікації користувачів в інтернет-магазинах, зокрема: використання двофакторної аутентифікації, єдиного входу (SSO) через сторонні сервіси, OAuth-протоколу.

Проведене тестування продемонструвало стабільну роботу модуля при різному навантаженні, та відповідність функціоналу поставленим вимогам. Аутентифікаційний процес виконується швидко. Окрему увагу було приділено перевірці стійкості модуля до типових атак, таких як brute-force та SQL-ін’єкції. Результати тестування підтвердили високу надійність та безпечність реалізованого рішення.

У ході дипломної роботи було успішно реалізовано програмний модуль, який здатен забезпечити надійний рівень захисту даних клієнтів інтернет-магазину. Розроблене рішення може бути використане як базовий компонент для масштабних комерційних проєктів, при цьому має потенціал до подальшого розширення наприклад, додавання двофакторної аутентифікації, інтеграції з соціальними мережами. Модуль відзначається простотою інтеграції, масштабованістю та відповідністю сучасним вимогам безпеки.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Соммервілл, І. Інженерія програмного забезпечення. 10-е вид. – К. : ВНТЛ-Класика, 2021. – 832 с.
2. Крамаренко, С. В.; Ковальчук, Ю. П. Основи безпеки інформаційних систем. – Київ: КНУ імені Тараса Шевченка, 2020. – 248 с.
3. Хільчевський, В. М.; Барановська, І. Ю. Web-програмування: навчальний посібник. – Київ: КНЕУ, 2021. – 336 с.
4. ДСТУ ISO/IEC 27002:2015. Інформаційні технології. Методи захисту. Практичні правила управління інформаційною безпекою. – Київ: ДП «УкрНДНЦ», 2015. – 96 с.
5. RFC 7519: JSON Web Token (JWT). Internet Engineering Task Force (IETF), 2015. – 33 p. [Веб-сайт]. URL: https://datatracker.ietf.org/doc/html/rfc7519 – Дата звернення: 18.05.2025.
6. Open Web Application Security Project (OWASP). Top 10 Web Application Security Risks. – 2023. [Веб-сайт]. URL: https://owasp.org/www-project-top-ten/ – Дата звернення: 18.05.2025.
7. ISO/IEC 27001:2017. Information technology – Security techniques – Information security management systems – Requirements. – Geneva: ISO, 2017. – 30 p.
8. McGraw, Gary. Software Security: Building Security In. – Addison-Wesley, 2006. – 448 p.
9. Welling, Luke; Thomson, Laura. PHP and MySQL Web Development. 5th Edition. – Addison-Wesley Professional, 2016. – 1008 p.
10. Flanagan, David. JavaScript: The Definitive Guide. 7th Edition. – O'Reilly Media, 2020. – 706 p.
11. Duckett, Jon. JavaScript and JQuery: Interactive Front-End Web Development. – Wiley, 2014. – 640 p.
12. OWASP Foundation. OWASP Authentication Cheat Sheet. – 2023. [Веб-сайт]. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html – Дата звернення: 25.05.2025.

ДОДАТОК А. Фрагмент програмного коду авторизації

```
namespace ShopAuth.Service;

using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Security.Cryptography.X509Certificates;
using System.Text.RegularExpressions;
using DataAccess;
using DataAccess.Entities;
using DTO;
using Exceptions;
using Extensions;
using Microsoft.Extensions.Options;
using Microsoft.IdentityModel.Tokens;
using Options;

public partial class AuthorizationService(IDataRepository dataRepository,
IOptions<AuthConfig> configuration, X509Certificate2 signingKey) : IAuthorizationService
{
    /// <summary>
    /// Registers a new user in the system, validates input, and generates an
authentication token for the user.
    /// </summary>
    /// <param name="userToCreate">The details of the user to register, including
username, password, and repeated password.</param>
    /// <returns>A <see cref="TokenDto"/> containing the access token, refresh token,
and token expiration time.</returns>
    /// <exception cref="ForbiddenException">
    /// Thrown if the user's passwords do not match, the username is invalid or is
already in use.
    /// </exception>
    public async Task<TokenDto> RegisterUserAsync(UserCreationDto userToCreate)
    {
        if (userToCreate.Password != userToCreate.RepeatedPassword)
        {
            throw new ForbiddenException("Passwords dont match");
        }

        if (userToCreate.Username.Length < 6 || userToCreate.Username.Contains(' '))
        {
            throw new ForbiddenException("Username cannot contain spaces and has to be
at least 8 digits");
        }

        IsValidPassword(userToCreate.Password);

        if (await
dataRepository.IsUserNameOccupiedAsync(userToCreate.Username.ToUpper()).ConfigureAwait(f
alse))
        {
            throw new ForbiddenException($"Username {userToCreate.Username} is
occupied");
        }

        var userEntity = new User
        {
            Uuid = Guid.NewGuid(),
            Username = userToCreate.Username,
            NormalizedUsername = userToCreate.Username.ToUpper(),
            ClientSecret = userToCreate.Password.Hash(),
            RegistrationDate = DateTime.Now
        };

        userEntity.UserId = await
dataRepository.SaveAsync(userEntity).ConfigureAwait(false);
```

```
        var tokenId = Guid.NewGuid().ToString();

        var token = GetToken(userEntity.Uuid, tokenId);

        var userSession = new UserSession
        {
            UserId = userEntity.UserId,
            RefreshToken = token.RefreshToken,
            JwtId = tokenId,
            RefreshTokenExpiration =
DateTime.Now.AddDays(configuration.Value.RefreshTokenExpiresDays),
            Redeemed = false
        };

        await dataRepository.SaveAsync(userSession).ConfigureAwait(false);

        return token;
    }

    /// <summary>
    /// Authorizes a user by validating their credentials and generates an
authentication token if successful.
    /// </summary>
    /// <param name="clientData">The credentials provided by the client, including
username and password.</param>
    /// <returns>A <see cref="TokenDto"/> containing the access token, refresh token,
and token expiration time.</returns>
    /// <exception cref="ForbiddenException">
    /// Thrown if the username is not found or the provided password is incorrect.
    /// </exception>
    public async Task<TokenDto> AuthorizeAsync(UserAuthorizationDto clientData)
    {
        var user = await
dataRepository.GetUserByUsernameAsync(clientData.Username).ConfigureAwait(false);

        if (user == null)
        {
            throw new ForbiddenException("Невірна комбінація логіну та паролю");
        }

        if (!clientData.Password.IsHashStringsEqual(user.ClientSecret))
        {
            throw new ForbiddenException("Невірна комбінація логіну та паролю");
        }

        var tokenId = Guid.NewGuid().ToString();

        var token = GetToken(user.Uuid, tokenId);

        var userSession = new UserSession
        {
            UserId = user.UserId,
            RefreshToken = token.RefreshToken,
            JwtId = tokenId,
            RefreshTokenExpiration =
DateTime.Now.AddDays(configuration.Value.RefreshTokenExpiresDays),
        };

        await dataRepository.SaveAsync(userSession).ConfigureAwait(false);

        return token;
    }

    /// <summary>
    /// Refreshes an access token using a valid access and refresh token pair,
generating a new access token and refresh token.
```

```
    /// </summary>
    /// <param name="accessToken">The expired or soon-to-expire access token.</param>
    /// <param name="refreshToken">The associated refresh token used to validate and
generate new tokens.</param>
    /// <returns>A <see cref="TokenDto"/> containing a new access token, refresh token,
and token expiration time.</returns>
    /// <exception cref="ForbiddenException">
    /// Thrown if the provided tokens are invalid, expired, tampered with, or do not
match the user's stored session
    /// details in the system.
    /// </exception>
    public async Task<TokenDto> RefreshTokenAsync(string accessToken, string
refreshToken)
    {
        var tokenHandler = new JwtSecurityTokenHandler();
        var validationParameters = GetValidationParameters();

        var principal = tokenHandler.ValidateToken(accessToken, validationParameters,
out var securityToken);

        if (securityToken is not JwtSecurityToken jwtSecurityToken ||
            !jwtSecurityToken.Header.Alg.Equals(SecurityAlgorithms.RsaSha256,
                StringComparison.InvariantCultureIgnoreCase))
        {
            throw new ForbiddenException();
        }

        var storedToken = await
dataRepository.GetUserSessionByRefreshTokenAsync(refreshToken);

        if (storedToken == null || storedToken.Redeemed)
            throw new ForbiddenException();

        storedToken.User = await dataRepository.GetUserByIdAsync(storedToken.UserId);


        var jwtId = principal.Claims.FirstOrDefault(x => x.Type ==
JwtRegisteredClaimNames.Jti)?.ToString();

        var userUuid = principal.FindFirst(ClaimTypes.NameIdentifier)?.Value;

        if (jwtId == null || userUuid == null ||
!string.Equals(storedToken.User.Uuid.ToString(), userUuid,
StringComparison.CurrentCultureIgnoreCase) ||
            !string.Equals("jti: " + storedToken.JwtId, jwtId,
StringComparison.CurrentCultureIgnoreCase) || DateTime.Now >
storedToken.RefreshTokenExpiration)
        {
            throw new ForbiddenException();
        }

        var newJwtId = Guid.NewGuid().ToString();

        var token = GetToken(storedToken.User.Uuid, newJwtId);

        var userSession = new UserSession
        {
            UserId = storedToken.User.UserId,
            RefreshToken = token.RefreshToken,
            JwtId = newJwtId,
            RefreshTokenExpiration =
DateTime.Now.AddDays(configuration.Value.RefreshTokenExpiresDays),
        };

        await dataRepository.SaveAsync(userSession);

        storedToken.Redeemed = true;
```

```
        await dataRepository.RedeemRefreshTokenAsync(storedToken.UserSessionId);

        return token;
    }

    /// <summary>
    /// Generates a JSON Web Token (JWT) for a given user, including user-specific
claims and token metadata.
    /// </summary>
    /// <param name="userUuid">The unique identifier (UUID) of the user for whom the
token is being generated.</param>
    /// <param name="jwtId">The unique identifier for the JWT, used to track the token
within the application.</param>
    /// <returns>A <see cref="TokenDto"/> containing the access token, a refresh token,
and the token's expiration time in seconds.</returns>
    private TokenDto GetToken(Guid userUuid, string jwtId)
    {
        var tokenHandler = new JwtSecurityTokenHandler();
        var claims = new List<Claim>
        {
            new(JwtRegisteredClaimNames.Sub, userUuid.ToString()),
            new(JwtRegisteredClaimNames.Jti, jwtId),
            new(JwtRegisteredClaimNames.Iat,
DateTimeOffset.Now.ToUnixTimeSeconds().ToString(), ClaimValueTypes.Integer64),
        };

        var now = DateTime.UtcNow;
        var expiry = now.AddMinutes(configuration.Value.TokenExpiresMinutes);
        var expiresIn = (int)(expiry - now).TotalSeconds;

        var credentials = new SigningCredentials(
            new RsaSecurityKey(signingKey.GetRSAPrivateKey()),
            SecurityAlgorithms.RsaSha256);

        var tokenDescriptor = new SecurityTokenDescriptor
        {
            Subject = new ClaimsIdentity(claims),
            Expires = expiry,
            SigningCredentials = credentials,
            Issuer = configuration.Value.Issuer,
            Audience = configuration.Value.Audience
        };

        var token = tokenHandler.CreateToken(tokenDescriptor);

        var tokenResponse = new TokenDto(tokenHandler.WriteToken(token),
GenerateRefreshToken(), expiresIn);

        return tokenResponse;
    }

    /// <summary>
    /// Generates a secure, unique refresh token using a cryptographic random number
generator.
    /// </summary>
    /// <returns>A base64-encoded string representing the generated refresh
token.</returns>
    private static string GenerateRefreshToken()
    {
        var randomNumber = new byte[32];
        using var rng = System.Security.Cryptography.RandomNumberGenerator.Create();
        rng.GetBytes(randomNumber);
        return Convert.ToBase64String(randomNumber);
    }

    /// <summary>
```

```
    /// Validates the given password against specified security constraints.
    /// Throws a ForbiddenException if the password does not meet the requirements.
    /// </summary>
    /// <param name="password">The password to validate.</param>
    /// <exception cref="ForbiddenException">
    /// Thrown when the password is null, empty, contains spaces, is less than 8
characters long,
    /// does not contain at least one capital letter, or does not include at least one
digit.
    /// </exception>
    private static void IsValidPassword(string password)
    {
        if (string.IsNullOrWhiteSpace(password))
            throw new ForbiddenException("Password cannot be empty");

        if (password.Length < 8)
            throw new ForbiddenException("Password must be at least 8 digits");

        if (password.Contains(' '))
            throw new ForbiddenException("Password must not contain space");

        if (!LettersConstraint().IsMatch(password))
            throw new ForbiddenException("Password must contain at least 1 capital
letter");

        if (!DigitsConstraint().IsMatch(password))
            throw new ForbiddenException("Password must contain at least 1 digit");
    }

    private TokenValidationParameters GetValidationParameters()
    {
        return new TokenValidationParameters
        {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new RsaSecurityKey(signingKey.GetRSAPublicKey()),
            ValidateIssuer = true,
            ValidIssuer = configuration.Value.Issuer,
            ValidateAudience = true,
            ValidAudience = configuration.Value.Audience,
            ValidateLifetime = false,
            ClockSkew = TimeSpan.Zero
        };
    }

    [GeneratedRegex("[A-Z]")]
    private static partial Regex LettersConstraint();

    [GeneratedRegex("[0-9]")]
    private static partial Regex DigitsConstraint();
}
```

ДОДАТОК Б . Слайди мультимедійної презентації

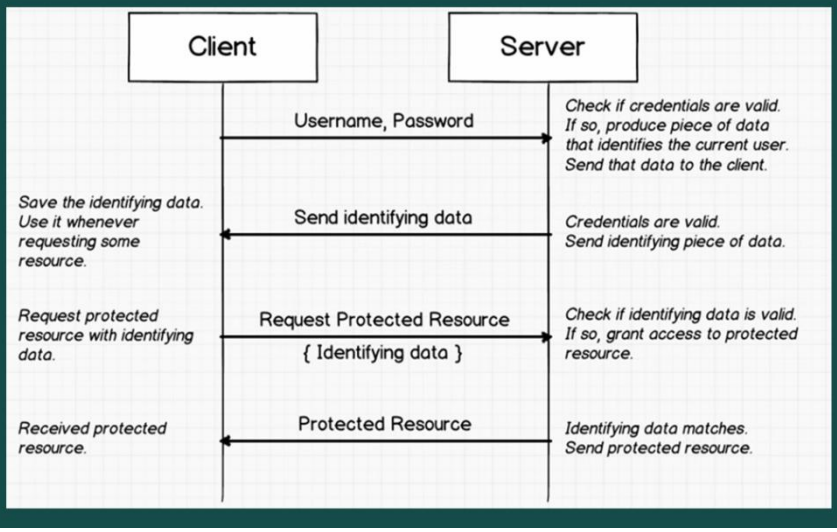
Розробка механізму автентифікації та авторизації клієнта у веб-застосунку інтернет-магазину

Виконав студент: Ремінний А.С.

Керівник: Залапін І.О.

Мета роботи

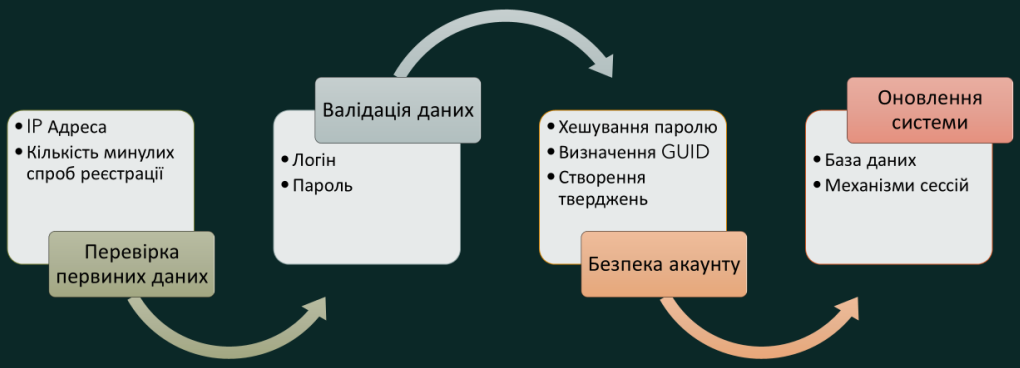
Створити надійний програмний модуль, що забезпечить реєстрацію, захист персональних даних та доступ до закритої частини веб-застосунку тільки для автентифікованих користувачів. У процесі реалізації враховувались вимоги безпеки, масштабованості та зручності використання.



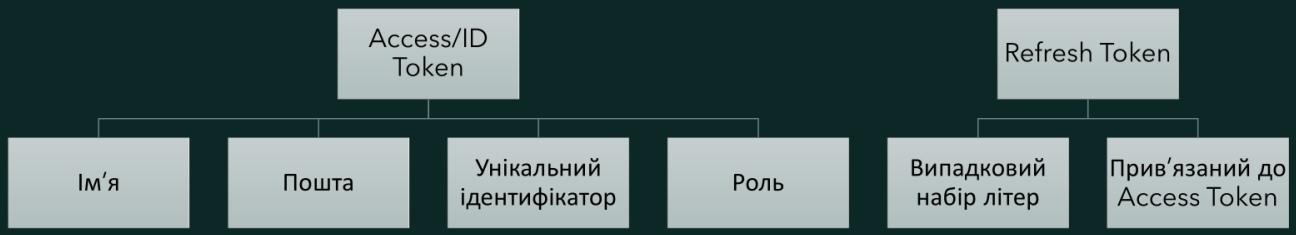
Проблеми, які вирішує механізм авторизації

- массові спроби автоматичних реєстрацій (боти);
- викрадення даних через API;
- навантаження на сервер через сканування зловмисниками;
- блокуванням реального функціоналу через атаки.

Як працює механізм реєстрації



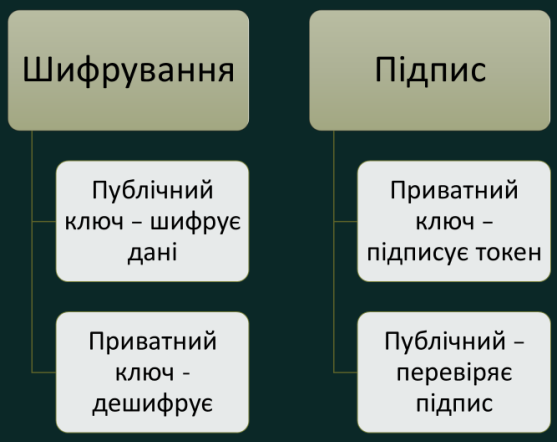
Що таке токени і як вони працюють



Механізм авторизації

- Перевірка даних
- Створення сесії
- Запис тверджень (claims)
- Підпис приватним ключем
- Створення токену оновлення
- Оновлення БД

Різниця в використанні ключів



Чому токен повинен оновлюватися кожну годину

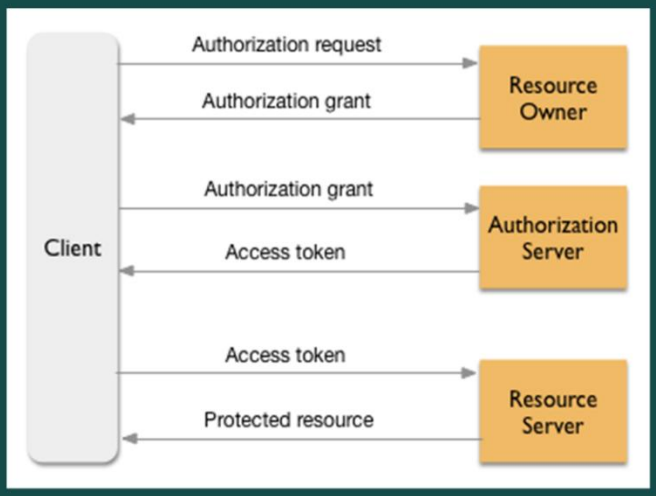
Якщо accessToken «вічний», у разі крадіжки зловмисник зможе користуватись ним без обмежень. Оновлення токенів через refreshToken кожну годину:

- мінімізує ризики злому;
- дозволяє відкликати доступ;
- зменшує навантаження на сервер при перевірці прав доступу;
- забезпечує контрольовану сесію користувача.

Механізм оновлення токенів

- Дешифрування Access Token
- Перевірка ідентифікатора сесії
- Перевірка правильності Refresh Token
- Створення нової пари Access/Refresh Token
- Оновлення БД

Oauth 2.0



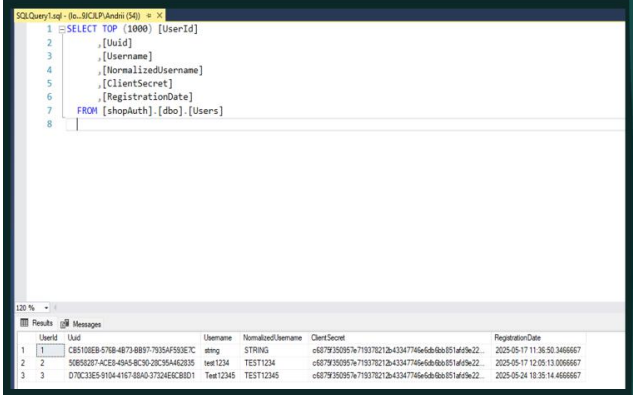
Зовнішній вид токена

Enter token below (it never leaves your browser):


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1MGI1ODI4Ny1hY2U4LTQ5YTUtYmM5MC0yOGM5NWE0NjI4MzUiLCJqdGkiOiIwZGI0N2VmOC0wMzBiLTQ5OWMtYjI0Yy1kZDE2OGUyYzcxMDAiLCJpYXQiOjE3NDk1NTIxNTEsIm5iZiI6MTc0OTU1MjE1MSwiZXhwIjoxNzQ5NTU1NzUxLCJpc3MiOiJTaG9wQXV0aCIsImF1ZCI6IlNob3BBdXRoLldlYiJ9.PzJiAyHsqQmWbjw6LyqHvmk60UXES-jZhKM3omQvK-ueZD7wcF4HilqPIoyJSDyIcGtew1dNrA1Zf2jIfIR-JBgyGEsd0xR2-Kp63o_hNX0NyEfxVa-enjYfokH1ZuFT5qaTN5lVJLXMTUZmEdTx3Vs3zst-oAOjVXbvxqImCkYGiTzn7SsSS04eoeN49Kqm38FQdhvOcstj8xM8sMLNuF4vGriGTyfpb6bnVqHlzyer8t4k9x8LbIY19ct9-7mRP22RERgccfVInUNaIS_t32ShMMZ3Sw_JsltheQxM5Qm-GeussBe-T4w5OOQlD5XrBe4Px8MkU4cjiD8OyQ5KPQ
```

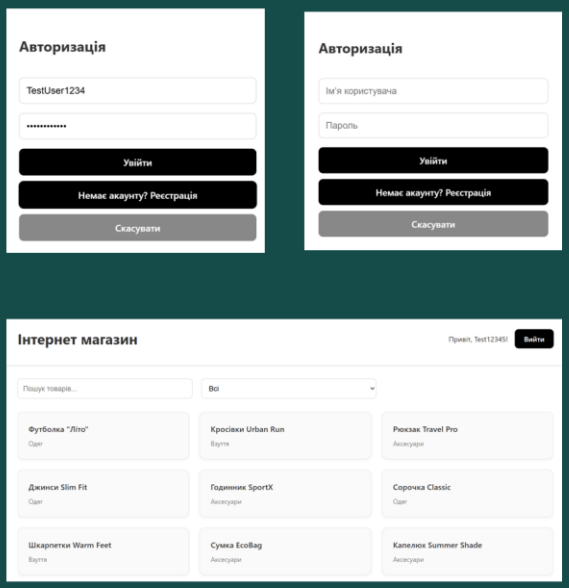

Decoded Token | Claims


```
{
  "alg": "RS256",
  "typ": "JWT"
}.{
  "sub": "50b58287-ace8-49a5-bc90-28c95a462835",
  "jti": "0db47ef8-030b-499c-b24c-dd168e2c7100",
  "iat": 1749552151,
  "nbf": 1749552151,
  "exp": 1749555751,
  "iss": "ShopAuth",
  "aud": "ShopAuth.Web"
}.[Signature]
```

Розташування даних у БД

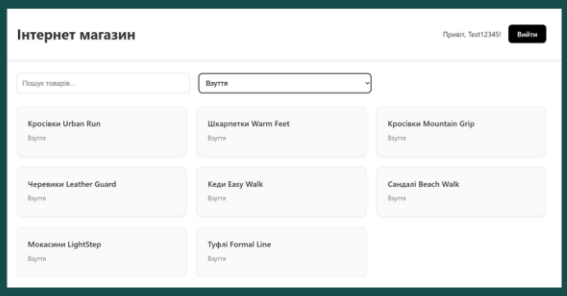
Запити на реєстрацію та авторизацію у проекті



Персоналізований доступ після входу

Після входу користувач бачить свій інтерфейс із кнопкою «Вийти», а також має доступ до товарів магазину, фільтрації та пошуку. Без авторизації – доступ обмежено.

Весь внутрішній функціонал розрахований лише на перевірених користувачів. Це реалізовано як на рівні API, так і в інтерфейсі.



Висновки

1. Реалізований механізм автентифікації захищає систему.
2. Авторизація дозволяє побудувати персоналізований інтерфейс.
3. Використання токенів робить систему масштабованою.
4. Оновлення сесій через refreshToken підвищує рівень безпеки.
5. Забезпечено розділення доступу між публічною та приватною частинами веб-застосунку.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Ремінного Андрія Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітня програма *«Безпека комп'ютерних систем та мереж»*

Керівник дипломного проекту (роботи) *Залапін Олексій Ігорович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка програмного модулю аутентифікації клієнта інтернет-магазину*

Обсяг розрахунково-пояснювальної записки *79* сторінок

Обсяг графічної (презентаційної) частини *15* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі захищеності доступу до ресурсів та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 15 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки добра, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Реалізовано Back-End та Front-end частини модулю авторизації за стандартом OAuth, який відповідає сучасним стандартам в сфері кібербезпеки.*

Проєкт містить повну реалізацію хешування паролів, логіку роботи з токенами, контроль сесій, а також забезпечення одноразовості використання токенів, що відповідає рекомендаціям OWASP.

д) основні недоліки дипломного проекту *Зберігання access-токена у localStorage робить його вразливим до XSS-атак; краще використовувати HttpOnly-cookie. Відсутній механізм блокування облікового запису після кількох невдалих спроб входу — ризик brute-force. Змішано бізнес-логіку з доступом до даних у DataRepository — порушення принципів чистої архітектури.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис: _____

«23» 06 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Ремінного Андрія Сергійовича

(прізвище, ім`я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка програмного модулю аутентифікації клієнта інтернет-магазину*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 79 сторінкок. У пояснювальній записці описано етапи розробки Back-End та Front-End частини програмного модулю аутентифікації клієнта інтернет-магазину засобами ASP.NET та React. Графічна частина складається з окремих слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Ремінний Андрій поступово та послідовно виконував всі етапи, проявляв ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Ремінний Андрій під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою. Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.*

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Ремінний Андрій показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування та криптографії, розробляти, та налаштовувати спеціалізоване програмне забезпечення, оформлювати слайди та складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як MS VS Code, MS VS, SSMS, Microsoft PowerPoint, Microsoft Visio та ін.*

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові керівника дипломного проекту *Залапін Олексій Ігорович*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерної техніки та програмної інженерії*

Підпис _____

«16» червня 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Ремінний Андрій Сергійович,
здобувач освіти гр. 4КБ-02, та

Залапін Олексій Ігорович ,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка програмного модулю аутентифікації клієнта інтернет-магазину» (автор роботи – Ремінний А.С., керівник роботи – Залапін О.І.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Ремінний А.С. /

Керівник _____ / Залапін О.І. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КБ-02

Ремінного Андрія Сергійовича

на тему *Розробка програмного модулю аутентифікації клієнта інтернет-магазину*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з деякмим порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 13.06.2025 р. значення коефіцієнту подібності в роботі становить 13,09%, коефіцієнт цитування – 1,35%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Ремінного А.С.* (П.І.Б.)

проведена « *16* » *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism

Дата звіту 6/13/2025
Дата редагування 6/13/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка програмного модуля аутентифікації клієнта інтернет-магазину

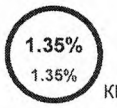
Автор
Ремінний Андрій Сергійович
Науковий керівник / Експерт
Залапін Олексій Ігорович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1



КЦ

25	14986	125062
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	18
Інтервали	A→	0
Мікропробіли	¶	2
Білі знаки	ß	0
Парафрази (SmartMarks)	a	88

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	47 0.31 %
2	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	44 0.29 %
3	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	39 0.26 %
4	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	39 0.26 %
5	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	38 0.25 %

6	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	37 0.25 %
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	36 0.24 %
8	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	34 0.23 %
9	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	34 0.23 %
10	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	29 0.19 %

з домашньої бази даних (0.33 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка 3D-гри у жанрі survival-horror з налаштуваннями рівнів складності [redacted] 6/12/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	49 (4) 0.33 %

з програми обміну базами даних (0.22 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Диплом_Школовой_УКР.pdf [redacted] 5/22/2022 State University of Telecommunications (ННІЗІ)	18 (2) 0.12 %
2	Lb_1_Nasheba_A.docx [redacted] 4/22/2023 Yuriy Fedkovych Chernivtsi National University(CNU) course papers (Deanery)	10 (1) 0.07 %
3	2023_61260002_Stepovanyi_Oleh_Ihorovych_197536 [redacted] 11/22/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	5 (1) 0.03 %

з Інтернету (12.54 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	543 (45) 3.62 %
2	**https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download**	**405 (30) 2.70 %**
3	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**70 (4) 0.47 %**
4	**https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download**	**69 (3) 0.46 %**
5	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	65 (7) 0.43 %
6	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content**	**65 (2) 0.43 %**
7	**https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download**	**64 (3) 0.43 %**
8	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content**	**64 (4) 0.43 %**
9	**https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download**	**60 (3) 0.40 %**
10	https://dotnettutorials.net/lesson/refresh-token-in-asp-net-core-web-api-using-jwt-authentication/	50 (6) 0.33 %

11	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	45 (4) 0.30 %
12	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	43 (3) 0.29 %
13	https://www.c-sharpcorner.com/article/jwt-authentication-with-refresh-tokens-in-net-6-0/	43 (3) 0.29 %
14	**https://www.c-sharpcorner.com/article/jwt-validation-and-authorization-in-net-5-0/**	**39 (2) 0.26 %**
15	**https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download**	**38 (1) 0.25 %**
16	**https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download**	**34 (1) 0.23 %**
17	https://card-file.ontu.edu.ua/bitstreams/12d5c0ab-e979-48f2-a8ec-d5fc31f71fd5/download	32 (4) 0.21 %
18	https://csharp.hotexamples.com/examples/-/LoginDto/-/php-logindto-class-examples.html	31 (5) 0.21 %
19	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	19 (2) 0.13 %
20	http://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:ztb:2011:projekty:letsrun	18 (1) 0.12 %
21	https://stackoverflow.com/questions/76838364/asp-net-core-7-mvc-authentication-using-json-web-tokens	15 (2) 0.10 %
22	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	14 (2) 0.09 %
23	https://card-file.ontu.edu.ua/bitstreams/f9e1bea4-9c6d-4957-b037-04c4fbac9b21/download	13 (1) 0.09 %
24	https://card-file.ontu.edu.ua/bitstreams/55e2b8f2-7d3c-4235-99fc-2be51199b96d/download	13 (2) 0.09 %
25	https://www.cs.ubbcluj.ro/files/curricula/2020/syllabus/IR_sem4_MLR5015_ro_bufny_2020_5156.pdf	11 (1) 0.07 %
26	https://www.c-sharpcorner.com/article/implement-jwt-in-net-core-api/	9 (1) 0.06 %
27	https://card-file.ontu.edu.ua/bitstreams/d42aac6d-ab01-4a74-b9cb-ced2a9eff719/download	7 (1) 0.05 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: **«Безпека комп'ютерних систем і мереж»** **Група:** 4КБ- 02

Дипломний **проект здобувача освіти денної форми навчання КБ.** 02.16.000.ДП

РЕМІННОГО
АНДРІЯ СЕРГІЙОВИЧА м. Одеса
2025 р. МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»
Група: 4 КБ-02

ПОЯСНЮВАЛЬНА ЗАПИСКА
до дипломного проекту на тему: _____
_____ Проектний матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах) Дипломник _____ (Ремінний А. С.) Керівник _____ (Залапін І.О.)

Консультанти:
з економічного розділу _____ (Канський М.Ю.)
з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.) з нормоконтролю