Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Застосунок для ведення нотаток

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-6
Єгор КУЗНЕЦОВ
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. Галина МАЙСТРЕНКО
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Кузнецову Єгору Руслановичу
(прізвище, ім’я, по батькові)

1. Тема роботи Застосунок для ведення нотаток

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи 1) платформа для розробки: Windows;
2) фреймворк для побудови інтерфейсу: XML; 3) архітектурний шаблон: MVVM;
4) місце зберігання даних: локальна база даних (PostgreSQL).

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз існуючих рішень у сфері ведення цифрових нотаток;
2) обґрунтування платформи, мови програмування і програмних засобів для реалізації;
3) проєктування архітектури десктопного додатку та бази даних;
4) програмна реалізація проекту за допомогою C#, WPF та .Net;
5) тестування програмного продукту та оцінка його функціональних можливостей;
6) висновки та перспективи подальшого розвитку цифрового нотатника.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 11 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз існуючих методів вирішення задачі	27.05.25 – 29.05.25	
2	Вибір програмного забезпечення та	30.05.25 – 31.05.25	
	інструментів розробки		
3	Проєктування архітектури застосунку	01.06.25 – 03.06.25	
4	Розробка логіки	04.06.25 – 06.06.25	
5	Розробка графічного інтерфейсу користувача	07.06.25 – 08.06.25	
6	Реалізація взаємодії	09.06.25 – 10.06.25	
7	Тестування застосунку	11.06.25 – 12.06.25	
8	Подання кваліфікаційної роботи керівникам	13.06.25 – 14.06.25	
	для попереднього захисту		
9	Подання кваліфікаційної роботи на	15.06.25 – 16.06.25	
	рецензування		

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Галина МАЙСТРЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 62 с., 3 таблиці, 18 рисунків, 13 джерел

C#, WPF, POSTGRESQL, ФАЙЛОВА СИСТЕМА, ОПЕРАЦІЇ З ІНФОРМАЦІЄЮ, MVVM, UI/UX DESIGN, ВІКОННИЙ ДОДАТОК, MVVM.

Кваліфакаційна робота присвячена розробці додатку для ефективної організації та управління інформацією, який дозволить користувачам зберігати та швидко отримувати доступ до важливих даних.

У ході виконання кваліфакаційної роботи було проаналізовано існуючі рішення у сфері організації цифрових нотаток, визначено їхні переваги та недоліки. На основі отриманих результатів розроблено власну архітектуру програмного забезпечення з використанням патерну MVVM, що забезпечує чітке розділення логіки додатку, інтерфейсу користувача та даних.

Реалізацію виконано на мові програмування C# із використанням фреймворку WPF, що дозволило створити сучасний та адаптивний інтерфейс. Для зберігання інформації використано систему керування базами даних PostgreSQL, що гарантує надійність і масштабованість при роботі з великими обсягами даних. Також було реалізовано функціонал збереження даних у файлову систему для резервного копіювання та офлайн-доступу. Також було приділено увагу UI/UX-дизайну: реалізовано можливість зміни шрифтів та їх розмірів, що робить додаток більш гнучким та зручним для різних категорій користувачів. А також додану світла та темна теми додатку для більш зручного користування різними користувачами.

ABSTRACT

Bachelor’s thesis: 62 p., 3 tables, 18 figures, 13 sources

C#, WPF, POSTGRESQL, FILE SYSTEM, INFORMATION OPERATIONS, MVVM, UI/UX DESIGN, WINDOW APPLICATION, MVVM.

The qualification work is devoted to the development of an application for the efficient organisation and management of information, which will allow users to store and quickly access important data.

In the course of the qualification work, the existing solutions in the field of digital note organisation were analysed, their advantages and disadvantages were identified. Based on the results obtained, we developed our own software architecture using the MVVM pattern, which ensures a clear separation of application logic, user interface and data.

The implementation was carried out in the C# programming language using the WPF framework, which allowed us to create a modern and adaptive interface. The PostgreSQL database management system was used to store information, which guarantees reliability and scalability when working with large amounts of data. We also implemented the functionality of saving data to the file system for backup and offline access. We also paid attention to UI/UX design: we implemented the ability to change fonts and their sizes, which makes the application more flexible and convenient for different categories of users. We also added light and dark themes to the application for more convenient use by different users.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ООП – об’єктно орієнтоване програмування

CLR– середовище виконання загальної мови (англ., Common Language Runtime)

ID – документ посвідчення особи (англ., identity document)

SQL – мова структурованих запитів (англ., Structured query language)

WPF – графічна підсистема Windows (англ., Windows Presentation Foundation)

XAML – розширена мова розмітки застосунків (англ., Extensible Application Markup Language)

ВСТУП

Сучасний ринок програмного забезпечення характеризується великою кількістю комп'ютерних програм, що відрізняються унікальними функціональними особливостями, перевагами та недоліками. Їх розробка спрямована на автоматизацію широкого спектра завдань та підвищення ефективності користувацької роботи за допомогою графічних інтерфейсів. На сьогоднішній день існує значна кількість програм, призначених для ефективного управління та організації інформації. Кожна з них має свій підхід до організації інформації.

Деякі спеціалізуються на великих об’ємах даних, структуруючи дані та зв’язуючи їх за допомогою посилань, що дозволяє створювати складні бази даних. Такі рішення ідеально підходять для систем управління знаннями та дослідницьких платформ.

Інші концентруються на корпоративних рішеннях: вони забезпечують централізоване адміністрування та збереження даних, автоматизуючи процеси узгодження та включають в себе версіонування документів, а також інтегрують бізнес-системи, такі як ERM та CRM.

Також існують програми, орієнтовані на повсякденне використання, які пропонують простий інтерфейс, мінімальні налаштування та швидке завантаження. Що є легким та доступним для щоденного використаня для студентів, офісних робітників та фрілансерів.

Ключовою метою даного проекту є створення програмного додатку для організації інформації, який забезпечить користувачам зручні засоби для управління нотатками, простий, інтуїтивно зрозумілий інтерфейс та швидкий доступ для потрібної інформації. Додаток надасть функціонал для перегляду, редагування та структурування записів, тим самим спрощуючи щоденну рутину.

Технологічною основою для розробки програми слугують мова програмування C#, підсистема WPF та платформа .Net, що дозволяє створити

гнучкий та функціонально насичений інтерфейс. Існуючі аналоги подібних додатків демонструють варіативність у наборі можливостей, дизайні та показниках продуктивності, що дозволяє користувачам здійснювати вибір на основі індивідуальних потреб.

Однак, незважаючи на різноманітний вибір програм, не всі програми можуть поєднати у собі зручність, доступність та швидкодію. Саме тому під час розробки цього програмного забезпечення було прийнято рішення зосередитись на поєднані легкості в використані, функціональності та швидкодії. Також велика увага приділялася інтерфейсу користувача, він мав бути простий, сучасний та інтуїтивно зрозумілий для користувача. Програма мала стати рішенням щоденних проблем при нагадуваннях на рівні з професійними проблемами у зберіганні інформації.

1 АНАЛІЗ ПРЕДМЕТЕНОЇ ОБЛАСТІ

1.1 Аналіз конкурентів

1.1.1 Додаток Notion

Notion – це робочий простір, який може стати програмою для текстових нотаток, календарем чи списком завдань [1]. Його головна особливість полягає у гнучкості, яка дозволяє підлаштовувати програму згідно до власних потреб. Notion є популярним серед різноманітних людей, таких як студенти, фрілансери, фахівці з IT, а також невеличкі команди, яким потрібен простий та гнучкий інструмент для командної роботи.

Notion є програмою, яка може налаштовуватися під різноманітні задачі. У додатку можна створювати сторінки та підсторінки. Кожна з них є універсальною одиницею, їх можна редагувати, підкріплювати один до одного та дублювати. На кожній сторінці можна розміщувати текст, таблиці, заголовки, календарі, чек-листи, фотографії та поєднувати з іншими сервісами (Google Drive, GitHub, тощо).

Notion має зручний та сучасний інтерфейс, який підтримує декілька різних тем оформлення. Структура програми нагадує складену файлову систему – користувач створює сторінки, які складуються у ієрархію сторінок, що дозволяє організовувати дані. Також є підтримка праці над документом декількома учасниками.

Приклади використання:

- студенти користуються Notion для створення конспектів, слідкуванням за строками здачі та виконання завдань;
- фрілансери створюють розклад, описи задач та ведуть документацію;
- розробники можуть вести технічну документації, створювати сторінки для обміну ідеями та планувати задачі.

Переваги та недоліки додатку представлено у таблиці 1.1.

Таблиця 1.1 – Переваги та недоліки додатку Notion

Переваги	Недоліки
Гнучкий та універсальний інтерфейс	Можливість праці лише в онлайн режимі
Широкі можливості редагування інформації	Повільне завантаження великих сторінок
Доступна інтеграція інших сервісів(Google, Microsoft, тощо)	Стислий доступ до всіх можливостей у безкоштовній версії
Кросплатформеність	Проблеми з експортом великих об’ємів інформації
Можливість роботи у команді	

Notion є програмою з платними можливостями, а саме доступно 4 тарифні плани:

- Free – базовий набір можливостей;
- Plus – розширені можливості використання;
- Business – бізнес варіант для команд з ролями;
- Enterprise – корпоративний варіант з підвищеним контролем безпеки.

Програма має величезну користувальницьку базу, яка ділиться шаблонами, відео та інструкціями для навчання, тайм-менеджменту, опановування платформи та проектного менеджменту. Це значно спрощує опанування платформи та покращує досвід новачків у використання Notion.

Загалом Notion – це зручний додаток, який поєднує в собі декілька програм в одну. Він подобається користувачам своєю гнучкість та простотою в використані.

1.1.2 Додаток Obsidian

Obsidian – це потужна програма для нотаток, яка працює локально з файлами у вигляді системі лінків [2]. Вона дозволяє користувачам

створювати глибоко пов'язану мережу записів, використовуючи внутрішні посилання для з'єднання нотаток. Програма візуалізує ці зв'язки у вигляді інтерактивного графу, допомагаючи структурувати знання гнучким та нелінійним способом.

Obsidian є програмою яка дозволяє користувачу легко користуватися своїми нотатками за допомогою системи Markdown-файлів, які зберігаються у так званих сховищах, та зберігаються локально на пристрої користувача. За допомогою цього вся інформація завжди доступна користувачеві оффлайн чи онлайн. Головною функцією Obsidian є можливість зв’язувати нотатки між собою за допомогою внутрішніх гіперпосилань. Зв’язувати між собою можна будь-які нотатки, використовуючи прямі чи зворотні посилання. Крім того, є можливість візуалізувати граф нотаток у вигляді інтерактивного дерева нотаток. Така структура покращує орієнтування при великих кількостях інформації.

Obsidian має мінімалістичний інтерфейс, який можна повністю підлаштовувати під різні потреби. Користувачі можуть обирати серед великої кількості тем та є можливість змінювати порядок та розташування панелей. Також в додатку є плагіни, які дозволяють розширювати функціонал програми за допомогою сторонніх чи внутрішніх інтеграцій, перетворюючи Obsidian у щоденник, конспект чи календар.

Приклади використання:

- студенти можуть вести конспекти, додаючи уточнення до окремих розділів;

- фрілансери можуть утворювати графи завдань до окремих клієнтів, плануючи свій час;

- дослідники мають можливість створювати пов’язані нотатки наукових тем та різноманітних джерел;

- програмісти можуть створювати документацію, розділяючи окремі теми, при одночасному зберіганні їх в одному файлі.

Переваги та недоліки додатку Obsidian наведено у вигляді таблиці 1.2.

Таблиця 1.2 – Переваги та недоліки додатку Obsidian

Переваги	Недоліки
Повна підтримка оффлайн роботи	Відсутність вбудованої можливості командної роботи
Можливість візуалізації зв’язків у графі	Доволі складний інтерфейс для новачків
Система плагінів та можливості інтеграції сторонніх сервісів	Синхронізація доступна лише с платної підпискою
Майже повний безкоштовний доступ	Складне освоєння програми

Програма є безкоштовною у своєму використані, але має декілька видів підписок:

- Free – безкоштовна підписка з повним функціоналом;
- Obsidian Sync – надає можливість синхронізації між пристроями;
- Obsidian Publish – надає можливість публікувати нотатки у вебайти.

Obsidian має доволі велику міжнародну спільноту, яка розробляє та ділиться численними шаблонами, плагінами, макросами, досвідом з ведення нотаток. На різних форумах можна знайти навчальні відео чи вказівки для ведення нотаток у Obsidian для новачків та не тільки.

Підсумовуючи, Obsidian це зручна програма для користувачів, яким необхідно структурувати дані та мати змогу легко переглядати великі об’єми інформації.

1.1.3 Додаток Microsoft OneNote

Microsoft OneNote – це цифровий записник, розроблений корпорацією Microsoft, який дозволяє користувачам створювати та впорядковувати нотатки [3]. Програма є частиною пакету Microsoft Office, але також доступна як окремий безкоштовний застосунок OneNote. Також він тісно взаємодіє з іншими продуктами Microsoft, такими як Word, Excel, Outlook та

OneDrive. Цей записник орієнтований як на індивідуальних користувачів, так і на корпоративних чи навчальних.

OneNote є імітацією паперового блокнота, користувач створює блокнот, який ділиться на розділи, а вони в свою чергу діляться на сторінки. Така структура дозволяє навіть новачка легко та швидко опанувати додаток. На сторінки користувач може додавати фото, відео, аудіо, файли, текст та навіть рукописні замітки. Саме так, OneNote має підтримку рукописного вводу, за допомогою планшетів чи стилусів. Також додаток має автоматичні збереження змін та надає доступ до історії редагувань, що має особливу важливість для командної роботи.

OneNote має класичний для продуктів Microsoft інтерфейс, з стрічкою швидкого доступу та вкладками. Додаток доступний як для комп'ютерів так і для мобільних пристроїв. Кожна дія в OneNote автоматично зберігається до OneDrive та синхронізується з іншими пристроями. Користувачі мають можливість працювати над одним файлом в реальному часі.

Приклади використання:

- викладачі можуть надавати учням завдання та контролювати виконання у режимі реального часу;
- офісні працівники проводять робочі зустрічі, записують нотатки з переговорів та зберігають чек-листи проєктів;
- використання у побуті – планування відпусток та подій разом з сім'єю.

OneDrive як і інші програми має як безплатну, так і платні підписки, але різницею є те, що сплачуючи за OneDrive, користувач сплачує за весь пакет програм Microsoft 365:

- особистий тариф;
- сімейний план;
- бізнес план.

Усі три тарифи є однаковими у можливостях, надаючи повний доступ до всіх можливостей програм з пакету Microsoft 365. Різниця, лише в людях

які оформлюють підписку (з назв підписок зрозуміло для кого вона передбачається). Переваги та недоліки представлені у таблиці 1.3.

Таблиця 1.3 – Переваги та недоліки Microsoft OneNote

Переваги	Недоліки
Інтеграція з продуктами Microsoft	Стислі можливості форматування
Легкий у використанні та освоєнні	Проблеми роботи при великих об’ємах інформації
Підтримка рукописного вводу	Проблеми з навігацією у великій кількості сторінок/розділів
Синхронізація у режимі реального часу через OneDrive	Проблеми з налаштуванням додаткових форматів даних
Можливість командної роботи	

Як і інші схожі програми, OneNote має широку аудиторію використання. На різноманітних форумах можна знайти шаблони та відео для спрощення та покращення роботи з програмою.

OneNote – ця додаток, що є одним із найкращих варіантів для командної роботи. Також він залишається одним з найзручніших для новачків, бо його опановування є доволі легким. Хоча він поступається в кастомізації та плагінах своїм конкурентам, але його офісна інтеграція є незаперечною.

1.1.4 Висновки стосовно конкурентів

Проаналізувавши додаток Oraton у порівнянні з такими відомими інструментами для нотаток, як Obsidian, Notion та Microsoft OneNote, можна зробити наступні висновки, особливо підкреслюючи унікальні переваги Oraton : його простоту та швидкість. У світі, де багато програмних додатків прагнуть запропонувати максимальну кількість функцій, що іноді призводить

до ускладнення інтерфейсу та зниження продуктивності, Oraton обирає інший шлях. Він робить ставку на мінімалізм та ефективність, що є його головною конкурентною перевагою.

Порівняно з Obsidian: Obsidian відомий своїми потужними можливостями для створення зв'язаних баз знань, глибокою кастомізацією та орієнтацією на локальне зберігання даних. Це робить його чудовим інструментом для впорядкування великих об’ємів інформації. Однак, ця потужність може супроводжуватися певним порогом входження та необхідністю витрачати час на налаштування та освоєння. Oraton натомість пропонує негайну готовність до роботи. Його сила – у можливості миттєво зафіксувати думку чи ідею без будь-яких перешкод, що є критичним для користувачів, які цінують швидкість захоплення інформації понад усе.

Порівняно з Notion: Notion є багатофункціональною платформою, що об'єднує нотатки, управління проектами та спільну роботу. Це справжній "комбайн" для організації інформації. Проте, для простих завдань ведення нотаток Notion іноді може здаватися надлишковим або навіть повільним через свій широкий функціонал. Oraton, завдяки своїй легкості та оптимізації, виграє у сценаріях, де потрібно швидко щось записати, не відволікаючись на складні структури чи завантаження важкого інтерфейсу.

Порівняно з Microsoft OneNote: OneNote пропонує гнучкий, інтуїтивно зрозумілий досвід, схожий на цифровий блокнот, з хорошою інтеграцією в екосистему Microsoft. Однак, "Oraton" може запропонувати ще більш сфокусований та швидкий досвід для тих, хто шукає максимальної простоти та миттєвої реакції програми. Якщо OneNote – це багатосторінковий зошит з різними розділами, то Oraton – це кишеньковий нотатник.

Завершуючи цей розділ хочеться додати, що Oraton позиціонується як ідеальне рішення для користувачів, які втомилися від складності та нагромадженості багатьох сучасних додатків і шукають інструмент, що дозволяє зосередитися на головному – на своїх думках та ідеях. Його ключові переваги – простота у використанні та виняткова швидкість роботи – роблять

Oraton чудовим вибором для:

- фіксування нотаток "на ходу";
- ведення щоденних записів, де важлива оперативність;
- користувачів, які цінують мінімалістичний дизайн та інтуїтивно зрозумілий інтерфейс;
- тих, кому потрібен надійний та спритний інструмент без зайвих функцій, що відволікають.

1.2 Постановка задачі

1.2.1 Обґрунтування необхідності розробки

Сучасний світ переповнений різноманітною інформацією і щоб ефективно її застосувати, для початку її потрібно впорядкувати. Для цього на ринку вже існує декілька рішень (Notion, Obsidian, OneNote тощо), але жоден з них не поєднує в собі водночас простоту, швидкість роботи, офлайн-доступ і можливість адаптації під вимоги користувачів.

З цього виходить, що є необхідність у створенні нового настільного застосунку, який буде поєднувати в собі усі плюси різних додатків, надаючи можливість роботи в офлайн за допомогою зручного та сучного інтерфейсу.

1.2.2 Мета та завдання

Метою даної кваліфікаційної роботи є розробка десктопного застосунку з використанням технології WPF та .Net для створення та редагування нотаток. Головною цілю є розробка інтуїтивно зрозумілого, швидкого та локально організованого програмного додатку.

Для досягнення поставленої мети було розроблено список завдань:

- провести аналіз предметної області за для виявлення проблем;
- сформулювати вимоги до програмного забезпечення;

- розробити архітектуру системи;
- реалізувати графічний інтерфейс за допомогою WPF;
- забезпечити повноцінну інтеграцію з базою даних;
- розробити функціонал;
- протестувати застосунок на відповідність технічним та функціональним вимогам;
- підготувати документацію та інструкцію для користувача.

1.2.3 Цільова аудиторія

Додаток буде орієнтований на:

- студентів (для ведення конспектів та слідкуванням за строками здачі робіт);
- фрілансерів та офісних працівників (для організації задач та заміток із зустрічей);
- використання у побуті (для запису задач на тиждень, списку продуктів чи запису гарних місць для поїздки).

1.2.4 Вимоги до інтерфейсу та функціональності

Інтерфейс додатку має бути прости для розуміння й одночасно мати сучасний вид. Основні вимоги:

- мінімалістичний дизайн;
- підтримка світлої та темної теми;
- можливість швидких переходів між нотатками;
- можливість редагування нотаток із застосуванням форматування;
- можливість створення, редагування та видалення нотаток;
- збереження нотаток у локальній базі даних;
- резервне копіювання даних.

1.2.5 Очікуванні результати

У результаті розробки додатку буде розроблено стабільний застосунок, який буде відповідати основним вимогам та надавати користувачеві логічно-зрозумілий інтерфейс, який користувач може адаптувати до себе, ефективну систему зберігання та відтворення інформації на локальному комп’ютері. Застосунок буде доступний одразу без додаткових налаштувань чи залежностей.

1.3 Проблематика існуючих рішень

Попри велику кількість існуючих рішень, орієнтованих на організацію нотаток та зберігання інформації, багато з них не задовольняють потреби певних категорій користувачів повністю. При аналізі існуючих продуктів (Notion, Obsidian, OneNote) було визначено певні недоліки, які створюють проблеми при використані програм, як з боку зручності так і функціональності.

Більша кількість популярних програм залежать від онлайн синхронізації та не можуть надати користувачеві більшу свободу. Перш за все це стосується програм, які напряму залежать від доступу в інтернет та не працюють без нього. Такі програми сильно обмежують користувачів, які мають погане інтернет-з’єднання, чи користувачів, які мають швидко завантажувати великі об’єми інформації. Через це може сповільнюватися ефективність роботи.

Більшість з наявних програмних рішень мають занадто складний функціонал та інтерфейс, який потребує тривалого навчання та адаптації до нової програми. Такі інтерфейси відбивають бажання до їх використання, бо звичайному користувачеві потрібен інтуїтивно зрозумілий інтерфейс, на який йому не потрібно витрачати час для зрозуміння.

Перевантаження інтерфейсу функціями вимагає високих затрат

оперативної пам’яті, що може сильно сповільнювати роботи у таких програмах на слабких чи застарілих комп’ютерах. Особливо помітним це стає, коли користувач працює з великими обсягами інформації.

За для запобігання перевантаження ресурсів комп’ютера, деякі програми використовують хмарні сховища, але це надає наступну проблему – проблему з безпекою та конфіденційністю інформації. Певна кількість користувачів можуть зберігати особисту чи комерційну інформацію у цих програмах, яка одразу відсилається у хмарне сховище. З одного боку це надає можливість для синхронізації на різних пристроях. А з іншого, дані зберігаються на сторонніх серверах, часто поза країною користувача, що ускладнює контроль над конфіденційністю та цілісністю цієї інформації.

Попри глобалізації програмного забезпечення, багато популярних сервісів мають певні проблеми з підтримкою української мови. Це створює незручності для україномовних користувачів під час навчання чи праці. Ця проблема є суттєвою через те що сервіси можуть неправильно зчитувати україномовний текст та створювати певні труднощі під час навчання чи роботи.

Аналіз існуючих рішень виявив низку проблем пов’язаних з різними секторами використання програм. Це створює умови для розробки додатку, у якому будуть продумані рішення для вирішення цих проблем.

2 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Мова C#

C# – це сучасна, об'єктно-орієнтована мова програмування, розроблена компанією Microsoft як частина її ініціативи .NET Framework. Вона була представлена у 2000 році Андерсом Хейлсбергом. C# була створена як мова, що поєднує потужність C++ з простотою Visual Basic, і швидко здобула популярність серед розробників.

Однією з ключових особливостей C# [4] є її тісна інтеграція з платформою .NET. Це означає, що програми, написані на C#, виконуються у спеціальному середовищі виконання, відомому як Common Language Runtime (CLR). CLR надає такі важливі сервіси, як автоматичне керування пам'яттю (збирання сміття), безпека типів та обробка винятків. C# є строго типізованою мовою, що допомагає виявляти помилки на етапі компіляції, а не під час виконання програми.

Сфера застосування C# дуже широка [5]. Вона активно використовується для розробки вебдодатків за допомогою ASP.NET, настільних додатків для Windows за допомогою Windows Forms або WPF, мобільних додатків за допомогою Xamarin, що також дозволяє створювати кросплатформні рішення для iOS та Android, а також для розробки ігор на популярному Unity [6]. Окрім цього, C# знаходить своє застосування у хмарних обчисленнях, розробці сервісів та корпоративних рішень.

2.2 Платформа .Net

.NET – це безкоштовна, кросплатформна платформа з відкритим вихідним кодом для розробки різноманітних типів додатків. Створена компанією Microsoft, .NET надає розробникам інструменти та бібліотеки для створення різноманітних додатків та сервісів таких як: вебдодатки, мобільні

застосунки, десктопні програм, хмарні сервіси, ігри та багато іншого [7].

Ключовими компонентами .NET є середовище виконання, відоме як Common Language Runtime (CLR) для класичного .NET Framework, або CoreCLR для сучасних версій .NET (.NET Core, .NET 5 та новіші). Це середовище відповідає за керування виконанням коду, включаючи швидку компіляцію (Just-In-Time compilation), керування пам'яттю за допомогою збирання сміття, безпеку та обробку винятків. Іншою важливою складовою є широкий набір бібліотек класів, що надає величезну кількість готових функціональних можливостей, від роботи з файлами та мережею до криптографії та маніпуляції даними.

З часом .NET значно еволюціонувала. Поява .NET Core ознаменувала перехід до кросплатформенності та відкритого вихідного коду. Починаючи з .NET 5, Microsoft уніфікувала свої .NET платформи (.NET Framework, .NET Core, Xamarin) під єдиним брендом ".NET". Кожен новий реліз (наприклад, .NET 6, .NET 7, .NET 8 і так далі) приносить покращення продуктивності, нові можливості та розширену підтримку сучасних технологій.

.NET підтримує декілька мов програмування, серед яких найпопулярнішими є C#, F# та Visual Basic .NET. Це дає розробникам гнучкість у виборі інструменту [8], який найкраще відповідає їхнім навичкам та вимогам проєкту. Завдяки потужним інструментам розробки, таким як Visual Studio та Visual Studio Code, великій активній спільноті та постійній підтримці з боку Microsoft, .NET залишається однією з провідних платформ для створення сучасного програмного забезпечення.

2.3 Графічна підсистема WPF

WPF (Windows Presentation Foundation) – це сучасна графічна підсистема від Microsoft, призначена для створення настільних застосунків під Windows. Вона є частиною .NET та дозволяє розробникам будувати інтерфейси користувача з високим ступенем гнучкості та візуальної

привабливості.

WPF [9] відокремлює логіку інтерфейсу від його представлення завдяки використанню XAML (Extensible Application Markup Language). Це дозволяє дизайнерам та розробникам працювати паралельно: дизайнери створюють візуальну частину у XAML [10], а розробники реалізують логіку на C# або іншій .NET-мові.

Однією з головних переваг WPF є підтримка розширених графічних можливостей. Платформа використовує DirectX, що дозволяє створювати складну графіку, анімації, векторні зображення, 3D-об'єкти та візуальні ефекти з меншою затратою ресурсів, ніж класичні WinForms.

WPF підтримує шаблони та стилі, що дає можливість централізовано змінювати вигляд елементів без модифікації їхньої логіки. Це важливо для підтримки єдиного дизайну в усьому застосунку.

Іншою важливою особливістю є система прив'язки даних, яка дозволяє ефективно зв’язувати інтерфейс з джерелами даних. Це особливо корисно у поєднанні з патерном MVVM (Model-View-ViewModel), який є загальноприйнятим архітектурним підходом для WPF-застосунків.

WPF також підтримує механізми розміщення елементів, які дозволяють створювати адаптивні інтерфейси. Завдяки цьому програма може коректно відображатися на різних екранах з різною роздільною здатністю.

2.4 Об’єктно-реляційна база даних PostgreSQL

PostgreSQL – це потужна об'єктно-реляційна система керування базами даних з відкритим вихідним кодом. Вона була створена як продовження проєкту Ingres у Каліфорнійському університеті в Берклі, а перша її версія вийшла у 1996 році. PostgreSQL [11] підтримує більшість стандартів SQL та має розширені можливості, які виходять за межі класичних реляційних баз даних.

Вона дозволяє працювати як зі структурованими даними (у вигляді

таблиць), так і з напівструктурованими або складними типами даних. Серед її особливостей – підтримка транзакцій, збережених процедур, тригерів, перевірок цілісності, індексів різних типів та референційної цілісності.

PostgreSQL орієнтована на розширюваність. Це означає, що користувачі можуть створювати власні типи даних, функції, оператори та навіть оптимізатори запитів. Вона має багату систему розширень, яка дозволяє додавати нові функції без модифікації ядра.

Система підтримує одночасну роботу багатьох користувачів завдяки механізму багатоверсійності (MVCC), який забезпечує високу продуктивність при паралельному виконанні запитів. Безпека реалізується через ролі, політики доступу, шифрування та інші засоби контролю.

PostgreSQL активно використовується у сфері веброзробки, фінансів, телекомунікацій, науки та аналітики даних. Вона сумісна з багатьма операційними системами, включаючи Linux, Windows та macOS, підтримується великою міжнародною спільнотою розробників.

2.5 Архітектурний патерн MVVM

MVVM (Model-View-ViewModel) – це архітектурний патерн, який широко використовується під час розробки WPF-додатків. З його допомогою можна чітко розділити внутрішні складові програми на інтерфейс, логіку та роботу з даними [12]. Це сприяє покращенню зручності розробки, читання, тестування та підтримки коду.

MVVM складається з 3 основних компонентів:

- Model (Модель) – це рівень, що відповідає логіку та дані програми. Цей рівень складається з класів, які реалізують логіку, взаємодію з базою даних та вебсервісами;

- View (Представлення) – цей рівень відповідає за інтерфейс користувача. Ця частина коду не містить логіки, а лише представлення та інтерфейси, які автоматично змінюються під вказівками змін ViewModel;

- ViewModel (Модель представлення) – цей рівень слугує посередником між View і Model. Тут реалізуються команди (RelayCommand) та інтерфейс INotifyPropertyChanged, за для відсилання вказівок у View.

Загалом MVVM слугує для покращення читабельності коду та спрощення роботи з ним. Він спрощує тестування логіки, полегшує повторні використання компонентів. Допомагає при розробці великих проектів.

2.6 Мова розмітки XAML

XAML (eXtensible Application Markup Language) – це мова розмітки [10], яка використовується при програмуванні в технологіях WPF, Xamarin, .Net та інші для опису інтерфейсу користувача.

XAML дозволяє описувати елементи інтерфейсу (Button, Label, Class, Grid та інші) у вигляді вкладених елементів (лістинг 2.1). Це дозволяє спростити написання інтерфейсу, бо для створення цих елементів, достатньо оголосити їх у XAML-коді. Також XAML-код дозволяє виконувати прив’язку елементів інтерфейсу до властивостей ViewModel. Це також відноситься і до подій, наприклад можна приєднати до кнопки якусь подію (лістинг 2.2).

Лістинг 2.1 – Приклад прив’язки елементів інтерфейсу до властивостей

```
<ListBox ItemsSource="{Binding Notes}">
```

Лістинг 2.2 – Приклад прив’язки елементів інтерфейсу до подій

```
<Button Content="Зберегти"
  Command="{Binding SaveNoteCommand}"
  Width="100"
  Height="30"
  Margin="10,0,0,0"
  Click="Button_Click"/>
```

Також в XAML є підтримка сторонніх ресурсів, наприклад, додавання фону додатку за допомогою стороннього зображення. Файли XAML під собою мають ще один файл, відповідаючий за оброблення подій та логіку.

Наприклад файл "MainPage.xaml" має додатково ще один підфайл "MainPage.xaml.cs". Загалом XAML дуже зручна мова розмітки, вона надає чіткий поділ інтерфейсу та логіки та легко інтегрується в MVVM, дозволяючи створювати гнучкі та адаптивні інтерфейси.

2.7 Парадигма об’єктно-орієнтованого програмування

ООП (Об’єктно-орієнтоване програмування) – це одна з парадигм програмування [13], яка складається з "об’єктів". Об’єкти поділяються на атрибути та методи. ООП має 4 основні принципи: інкапсуляція, наслідування, поліморфізм та абстракція.

Інкапсуляція – це принцип за допомогою якого відокремлюються методи та атрибути від інших частин програми. Вона допомагає контролювати доступ до даних, захищаючи об’єкт від неправильного використання. В C# це відбувається за допомогою модифікатору доступу private або інших.

Наслідування – це механізм успадковування методів та атрибутів одного класу іншим. Це дозволяє багаторазово використовувати код, не переписуючи його. У C# це відбувається за допомогою конструкції, яка наведена у лістингу 2.3.

Лістинг 2.3 – Механізм наслідування

```
class Book {
          public string Title {get; set;}
}
class PaperBook : Book {
          public string Quality {get; set;}
}
```

Поліморфізм – це здатність методів поводитись по-різному відносно вхідних даних. Існує два види поліморфізму: перевантаження та перевизначення. Перевантаження – це опис двох методів з однаковим ім’ям, але різними параметрами. А перевизначення – це два методи, перший

знаходиться в класі-"батька", а інший з таким же ім’ям в класі-"нащадку". Визиваючи той чи інший клас, розробник буде використовувати різні методи.

Абстракція – це така система, при якій виділяються загальні характеристики та методи об’єктів, приховуючи деталі реалізації. Виконується вона при створені спочатку інтерфейсу, за допомогою ключового слова "interface" перед ім’ям класу, а потім реалізується сам клас з описом методів.

ООП допомагає підвищити читабельність коду за допомогою його розбиття на класи. Це допомагає використовувати код по декілька разів, не переписуючи його.

3 СТРУКТУРА ДОДАТКУ

3.1 Загальна структура програми

Загальна структура програми розроблена за допомогою технології MVVM, що надає зручний огляд до всіх файлів (рисунок 3.1).

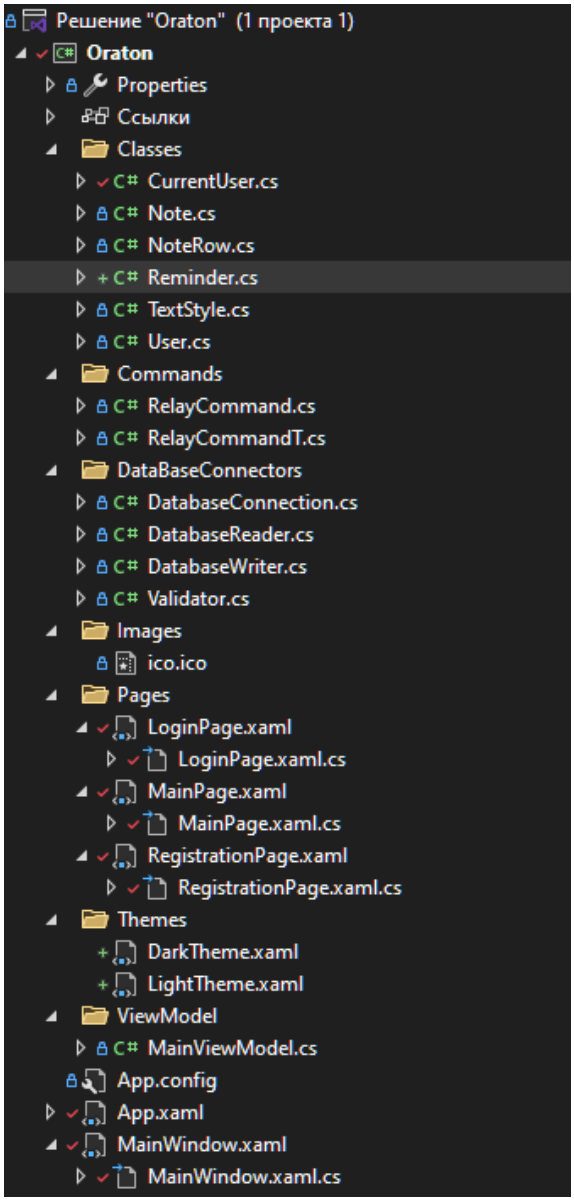


Рисунок 3.1 – Схема проєкту Oraton

Весь проєкт чітко поділений на 6 частин, які чітко виділені та відповідають за коректну роботу різних частин програми.

Перша та друга частини відповідають за інтерфейс користувача, у них прописано інтерфейс, шрифти, розміри та розміщення на вікнах. Ці частини, котрі бачить користувач, визначають досвід користувача, тобто наскільки користувачу подобається зовнішній вигляд додатку.

Наступні 3 частини є класами "Model", тобто класами, які є логікою програми. Тут прописаний зв’язок з БД, виклики команд та подій, а також класи, які зберігають в собі всі дані.

Остання частина є "ModelView" частиною, у ній прописані зміни інтерфейсу під час різних подій та взаємодії інтерфейсу з логікою додатку. Саме тут знаходяться функції за допомогою яких здійснюється звертання до даних (зі сторони інтерфейсу). Як приклад можливо зазначити порівняння чи заповнення нотаток. В свою чергу логіка додатку бере ці дані з БД та направляє в ModelView за для подальших дій.

3.2 Класи даних

3.2.1 Класи даних для бази даних

Розглянемо 4 класи програми, які тісно пов’язані з базою даних (рисунок 3.2) та мають відповідні таблиці в ній. Ці класи слугують для записи інформації з БД та зворотного запису в БД:

- Note – відповідає за нотатки кожного користувача;
- NoteRows – відповідає за текст в нотатках;
- TextStyle – відповідає за збереження стилей та розміру тексту;
- User – відповідає за користувачів в БД.

Ці класи є класами, які мають приймати в себе дані з бази даних, а також зберігати в собі зміни та нових користувачів перед відправкою їх до БД. На лістингу 3.1 представлено клас TextStyle, який зберігає в собі дані про стилі та розміри певних відділів тексту.

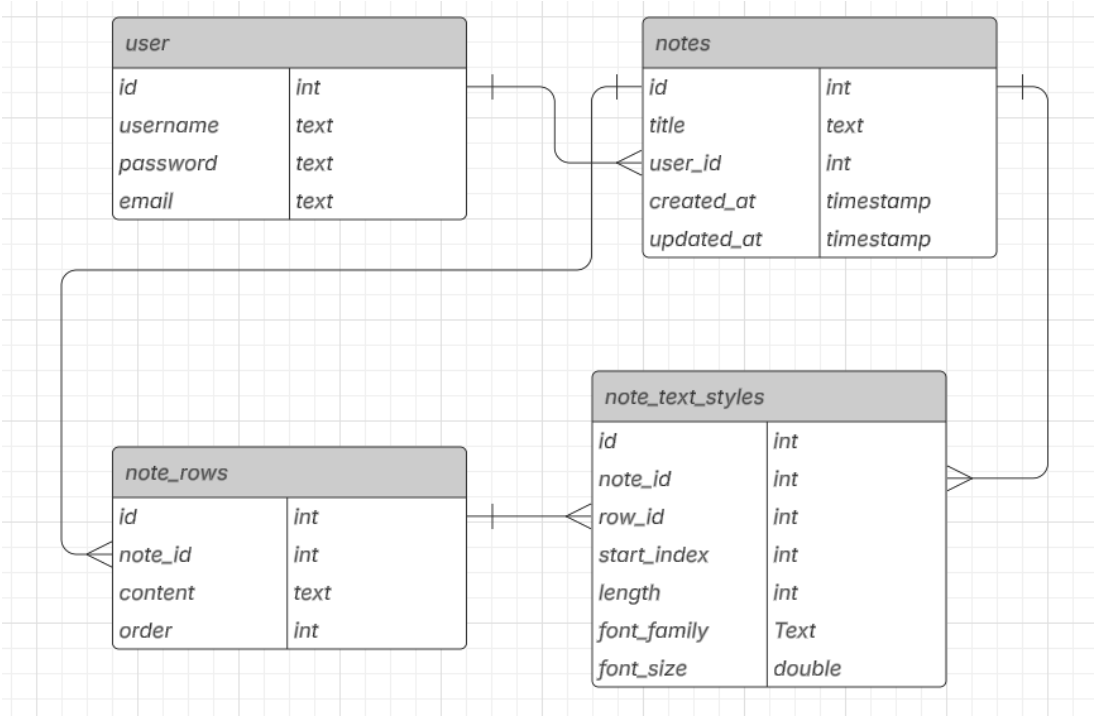


Рисунок 3.2 – Схема бази даних

Лістинг 3.1 – Клас TextStyle

```
public class TextStyle
{
public int Id { get; set; }
public int NoteId { get; set; }
public int RowId { get; set; }
public int StartIndex { get; set; }
public int Length { get; set; }
public string? FontFamily { get; set; }
public float? FontSize { get; set; }
}
```

3.2.2 Класи даних для поточної роботи

Також в програмі є ще два класи з даними CurrentUser та Reminder.

Клас CurrentUser (лістинг 3.2) відповідає за збереження поточного користувача та інформації про нього, що має критично важливу роль для коректної роботи додатку. Завдяки цьому класу забезпечується пошук та відображення коректних нотаток певного користувача, також цей клас відповідає за коректний запис та оновлення нотаток конкретного користувача. Цей клас допомагає запобігати проблем з перемішувань нотаток

між користувачами та забезпечує коректний доступ користувачеві саме до його нотаток.

Лістинг 3.2 – Клас CurrentUser

```
internal class CurrentUser
{
    public static User User { get; set; }

    public static void SetUser(User user)
    {
        User = user;
    }

    public static bool IsAuthenticated()
    {
        return User != null;
    }
}
```

Клас Reminder (лістинг 3.3) є класом, що відповідає за повідомлення та нагадування, що є основною функціональністю програми. Він керує таймером з відліком часу до повідомлення користувача, а також забезпечує коректне збереження часу нагадування разом з необхідним текстом. Завдяки цьому класу, користувач може самостійно обирати час та день повідомлення та додавати необхідний текст до кожного з них. Саме цей клас допомагає користувачеві слідкувати за дедлайнами та ефективно розподіляти свій час. Ці два класи є провідниками для коректної роботи додатку. Вони забезпечують коректну роботу додатку та підтримують взаємодію з іншими частинами програми, за допомогою збереження даних про поточного користувача.

Лістинг 3.3 – Частина класу Reminder

```
public class Reminder
{
    private DateTime reminderTime;
    private Timer timer;
    public Reminder(DateTime time, string message)
    {
        reminderTime = time;
        timer = new Timer(1000);
```

```
        timer.Elapsed += (s, e) => CheckReminder(message);
        timer.Start();
    }
    private void CheckReminder(string message)
     {
        if (DateTime.Now >= reminderTime)
          {
            ShowNotification("Нагадування", message);
            timer.Stop();
}
}
```

3.3 Команди

У цю частину входить два класи команд RelayCommand (лістинг 3.4) та RelayCommand<T> (лістинг 3.5), які відповідають за реалізацію команд, дозволяючи додавати логіку виконання та умови доступності команд.

Перший клас RelayCommand є звичайним представником класу команд, який виконує команду з одним параметром типу "object". Він включає в себе основну логіку для виконання команд та перевірки їх на доступність. Він перевіряє умови активації команди та сам зміст виконаної команди. Цей клас є зручним при виконані команд, які не є складними та не потребують особливих параметрів.

Лістинг 3.4 – Клас RelayCommand

```
internal class RelayCommand : ICommand{
    public event EventHandler CanExecuteChanged;
    public Action<object> _Execute { get; set; }
    public Predicate<object> _CanExecute { get; set; }
    public RelayCommand(Action<object> execute,
Predicate<object> canExecute){
        _Execute = execute;
        _CanExecute = canExecute;
    }
    public bool CanExecute(object parameter){
        return _CanExecute(parameter);
    }
    public void Execute(object parameter){
        _Execute(parameter);
    }
}
```

Інший клас RelayCommand<T> (лістинг 3.5) є більш гнучкою версією свого попередника. Він дозволяє використовувати широкий спектр параметрів, надаючи більші можливості при розробці команди, оскільки може приймати різні параметри, такі як рядки, числа, списки та інші.

Обидва класи використовуються за для реалізації команд, дозволяючи відокремити логіку виконання від інтерфейсу користувача та покращити тестованість та масштабованість програми.

Лістинг 3.5 – Клас RelayCommand<T>

```
internal class RelayCommand<T> : ICommand
{
    private readonly Action<T> _execute;
    private readonly Predicate<T> _canExecute;

    public RelayCommand(Action<T> execute, Predicate<T>
canExecute = null)
    {
        _execute = execute ?? throw new
ArgumentNullException(nameof(execute));
        _canExecute = canExecute;
    }
    public bool CanExecute(object parameter) =>
_canExecute?.Invoke((T)parameter) ?? true;
    public void Execute(object parameter) =>
_execute((T)parameter);
    public event EventHandler CanExecuteChanged;
}
```

3.4 Класи для зв’язку з базою даних

Загалом у програмі є три основні класи, які забезпечують обмін інформацією між БД та самою програмою. Кожен з них відповідає за конкретну функцію для коректної роботи обміну:

- DataBaseConnection (лістинг 3.6) – цей клас є найважливішим. Завдяки ньому встановлюється з’єднання з базою даних для подальшої роботи. Сам клас окрім з’єднання з БД, є також перевіряльником статусу підключення, що застерігає від збоїв у програмі та дає можливість одразу вирішувати проблему без зупинки програми чи перевірки з’єднання;

- DatabaseReader – цей клас відповідає за зчитування даних із бази даних. Він містить методи для отримання користувачів, перевірки унікальності логіна та пошти, завантаження нотаток та їхніх рядків, а також завантаження стилів та розмірів уже існуючого тексту. Клас зчитує інформацію з БД, перевіряючи її на унікальність та забезпечуючи правильність порядку зчитування, без перемішування даних різних користувачів за для її запису у певні класи. На лістингу 3.7 зображена функція зчитування рядків та запису їх до класу NoteRow;

- DatabaseWriter – цей клас відповідає за зворотній процес, за запис даних у БД з певних класів. Він містить методи для запису нових користувачів, новоутворених нотаток та змін у нотатках, відслідковує нову інформацію порівнючи її з вже існуючою. У випадку утворення нового користувача, він перевіряє його унікальність порівнюючи імена користувачів та адреса електронних пошт. Під час запису змін у нотатках, він повністю перезаписує нотатку за для уникнення дублювання інформації чи пропуску якихось змін. На лістингу 3.8 зображено метод для запису даних користувача в словник. Після додавання даних до словника "data", викликається інший метод (лістинг 3.9), який відповідає за запис цих даних у БД;

- Validator (лістинг 3.10) – цей клас не відповідає за зв’язок з БД, але також грає велику роль у цьому проєкті: він перевіряє введені дані електронної пошти за загальними стандартами та не допускає невірного або неповного введення.

Лістинг 3.6 – Клас DataBaseConnection

```
internal class DatabaseConnection
{
    private static readonly string ConnectionString =
"Host=localhost;Port=5432;Username=postgres;Password=123456;Data
base=postgres";

    public static NpgsqlConnection GetConnection()
    {
        var connection = new NpgsqlConnection(ConnectionString);
        connection.Open();
```

```
        return connection;
    }
}
```

Лістинг 3.7 – Функція GetNoteRows

```
public List<NoteRow> GetNoteRows(){
    var noteRows = new List<NoteRow>();
    using (var connection = DatabaseConnection.GetConnection()){
        string query = "SELECT id, note_id, content, \"order\"
FROM note_rows";
        using (var command = new NpgsqlCommand(query,
connection))
        using (var reader = command.ExecuteReader()){
            while (reader.Read()){
                var row = new NoteRow{
                    Id = reader.GetInt32(0),
                    NoteId = reader.GetInt32(1),
                    Content = reader.GetString(2),
                    Order = reader.GetInt32(3)};
                noteRows.Add(row);}}
        return noteRows;}}
```

Лістинг 3.8 – Метод InsertUser

```
public void InsertUser(User user)
{
    var data = new Dictionary<string, object>
{
    { "username", user.Username },
    { "password", user.Password },
    { "email", user.Email }
};
    InsertData("users", data);
}
```

Лістинг 3.9 – Метод InsertData

```
using (var connection = DatabaseConnection.GetConnection())
    {
        var columns = string.Join(", ", data.Keys);
        var parameters = string.Join(", ",
GetParameterNames(data.Keys));
        string query = $"INSERT INTO {tableName} ({columns})
VALUES ({parameters})";
        using (var command = new NpgsqlCommand(query,
connection))
        {
            foreach (var kvp in data)
            {
                command.Parameters.AddWithValue(kvp.Key,
```

```
kvp.Value);
            }
            command.ExecuteNonQuery();}}
```

Лістинг 3.10 – Клас Validator

```
internal class Validator
{
    public static bool IsValidEmail(string email)
    {

        if (string.IsNullOrWhiteSpace(email))
            return false;

        string pattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-
]+\.[a-zA-Z]{2,}$";

        return Regex.IsMatch(email, pattern);
    }
}
```

Загалом вся система чітко слідкує за правильністю даних, уникає дублікатів користувачів. У разі створення двох однакових нотаток не виникає жодних суперечок даних, через те що, кожна нотатка має унікальний ID, через який можна чітко визначити який текст нотатки куди відноситься.

3.5 Вікна додатку

LoginPage – ця сторінка відповідає за вхід у систему, на ній користувач вводить своє ім’я та пароль для подальшого доступу до системи. Ця сторінка є сторінкою автентифікації, перевіряючи надані користувачем дані, вона надає доступ до певних нотаток. Ця сторінка організує безпеку програми перевіряючи користувача та надає можливості використання програми лише зареєстрованим користувачам.

При натисканні кнопки входу (лістинг 3.11), посилається запит на перевірку вірності даних, порівнюються дані введені користувачем з даними з БД, й залежачи від того чи співпадають дані чи ні, відкривається основна сторінка програми, чи з’являється повідомлення, що сповіщує про невірність введених даних.

Лістинг 3.11 – Метод Button_Click_1

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    string username = txtUsername.Text;
    string password = txtPassword.Password;
    DatabaseReader dbReader = new DatabaseReader();
    User user = dbReader.GetUserByCredentials(username,
password);
    if (user != null){
        CurrentUser.SetUser(user);
        this.NavigationService.Navigate(new MainPage());
    }
    else{
        lblMessage.Text = "Невірний логін або пароль!";
        lblMessage.Visibility = Visibility.Visible;
     }
}
```

MainPage – це основна сторінка додатку, на ній користувач може побачити свої нотатки: на ній він може переглядати зміст нотаток. На цій сторінці користувачу надається повний доступ над нотатками: можливості створення нових, видалення застарілих, змінення існуючих нотаток, збереження інформації та створення повідомлень. Ця сторінка є ключовим інтерфейсом, за допомогою якого користувач отримує доступ до повного функціоналу програми.

Методи цієї сторінки відповідають за зчитування інформації при натисканні кнопок, вибору нотаток, зчитування введеного тексту, зчитування тексту, якого користувач виділив та виклику методів з класу логіки. На лістингу 3.12 показано початок класу, який включає в себе оголошення об’єкту класу логіки, конструктор для ініціалізації компонентів вікна та завантаження даних про нотатки та їх зміст.

Лістинг 3.12 – Частина класу MainPage

```
public partial class MainPage : Page
{
    private MainViewModel _viewModel;

    public MainPage()
    {
        InitializeComponent();
```

```
        _viewModel = new MainViewModel();
        DataContext = _viewModel;
        _viewModel.LoadNotes();

        _viewModel.PropertyChanged += ViewModel_PropertyChanged;
    }
```

RegistrationPage – ця сторінка відповідає за реєстрацію користувача у системі. У цьому вікні користувач вводить свої дані, такі як e-mail, ім'я користувача та пароль. У разі підтвердження унікальності введених даних, створюється новий обліковий запис, який в подальшому можна використовувати для праці з нотатками на основній сторінці додатку. Ця сторінка є першим кроком користувача в роботі з програмою.

Методи класу цієї сторінки відповідають за чітке слідкуванням унікальності введених даних та запису користувача до бази даних. Після введення даних усі зайві пробіли будуть автоматично видалено, дані перевірено на унікальність, а e-mail буде додатково перевірний на правильність введення. Після усіх перевірок буде створено новий обліковий запис (лістинг 3.13), додано його до БД, а користувач буде сповіщений про результат.

Лістинг 3.13 – Клас Button_Click

```
private void Button_Click(object sender, RoutedEventArgs e)
 {
     string username = usernameField.Text.Trim();
     string email = emailField.Text.Trim();
     string password = passwordField.Text.Trim();
…
            User newUser = new User
            {
                Username = username,
                Email = email,
                Password = password
            };

            DatabaseWriter dbWriter = new DatabaseWriter();
            dbWriter.InsertUser(newUser);

            MessageBox.Show("Реєстрація успішна!", "Успіх",
MessageBoxButton.OK, MessageBoxImage.Information);
            this.NavigationService.Navigate(new LoginPage());
```

MainWindow – це перше та головне вікно програми, саме на нього потрапляє користувач при запуску додатку. Тут прописаний загальний вид додатку та головні елементи навігації. Починаючи звідси користувач може змінювати теми додатку. Це вікно є точкою входу користувача в програму та надає можливості для реєстрації нового облікового запису чи входу в вже існуючий.

Клас цього вікна відповідає за ініціалізацію вікна, наявність чи відсутність панелі меню та заголовкової панелі вікна, а також за саму роботу заголовкової панелі вікна. На лістингу 3.14 показаний один з методів класу, а саме метод, який відповідає за зміну теми вікна.

Лістинг 3.14 – Метод SetTheme

```
private void SetTheme(string theme)
{
    string uri = theme == "Світла"
        ? "Themes/LightTheme.xaml"
        : "Themes/DarkTheme.xaml";

    Application.Current.Resources.MergedDictionaries.Clear();
    Application.Current.Resources.MergedDictionaries.Add(new
ResourceDictionary
    {
        Source = new Uri(uri, UriKind.Relative)
    });

    DarkThemeItem.IsChecked = theme == "Темна";
    LightThemeItem.IsChecked = theme == "Світла";
}
```

3.6 Клас логіки

MainViewModel – є серцем програми, саме тут прописана основна логіка, більшість методів та взаємодії логіки з інтерфейсом. Саме в цьому класі прописана можливість переходу між вікнами програми та виведення інформації. Цей клас слугує зв’язною точкою між усіма відділами програми. Тут прописані виклики методів взаємодії з БД за для подальшого виведення даних у вікна, нотаток у перелік нотаток певного, авторизованого

користувача, а їх змісту у відповідне поле для редагування. На лістингу 3.15 наведено метод для виведення списку нотаток

Лістинг 3.15 – Метод для виведення списку нотаток

```
public void LoadNotes()
{
  Notes.Clear();
  foreach (var note in _databaseReader.GetNotes())
  {
      Notes.Add(note);
   }
}
```

У лістингу 3.16 вказано головний цикл для виведення тексту нотаток, у цьому циклі звіряються літери та положення кожної літери задля додавання до них стилів. Після додання стилів, відбувається порядкове заповнення текстового поля даними з БД.

Лістинг 3.16 – Цикл з методу LoadNoteContent

```
    var textStylesForRow = textStyles.Where(s => s.RowId ==
row.Id).OrderBy(s => s.StartIndex).ToList();
    foreach (var style in textStylesForRow)
     {
        if (style.StartIndex > currentIndex)
          {
            paragraph.Inlines.Add(new
Run(row.Content.Substring(currentIndex, style.StartIndex -
currentIndex)));}
        var styledRun = new
Run(row.Content.Substring(style.StartIndex, style.Length));
        if (!string.IsNullOrEmpty(style.FontFamily)){
            styledRun.FontFamily = new
FontFamily(style.FontFamily);}
        if (style.FontSize.HasValue){
            styledRun.FontSize = style.FontSize.Value;}
        paragraph.Inlines.Add(styledRun);
        currentIndex = style.StartIndex + style.Length;}
    if (currentIndex < row.Content.Length)
     {
        paragraph.Inlines.Add(new
Run(row.Content.Substring(currentIndex)));}
    flowDocument.Blocks.Add(paragraph);}
NoteContentDocument = flowDocument;
}
```

Цей клас містить логіку створення (лістинг 3.17), видалення (лістинг 3.18), редагування, перейменування (лістинг 3.19) та збереження нотаток.

Лістинг 3.17 – Метод створення нової нотатки

```
private void CreateNewNote()
{
    int currentUserId = CurrentUser.User.Id;
    var newNote = new Note
    {   Title = "Нова замітка",
        UserId = currentUserId,
        CreatedAt = DateTime.Now,
        UpdatedAt = DateTime.Now
    };
    _databaseWriter.InsertNote(newNote);
    Notes.Add(newNote);
    SelectedNote = newNote;
    LoadNotes();
}
```

Лістинг 3.18 – Метод видалення нотатки

```
private void DeleteNote()
{
    if (SelectedNote == null) return;
    var result = MessageBox.Show(
        "Ви впевнені, що хочете видалити цю нотатку?",
        "Видалення нотатки", MessageBoxButton.YesNo,
MessageBoxImage.Warning);
    if (result == MessageBoxResult.Yes)
    {
        _databaseWriter.DeleteNote(SelectedNote.Id);
        Notes.Remove(SelectedNote);
        SelectedNote = Notes.FirstOrDefault();
        CommandManager.InvalidateRequerySuggested();
    }
}
```

Лістинг 3.19 – Метод перейменування нотатки

```
private void RenameNote(){
    if (SelectedNote == null) return;
    var newTitle = Microsoft.VisualBasic.Interaction.InputBox(
        "Введіть нову назву нотатки:", "Перейменування",
SelectedNote.Title);
    if (!string.IsNullOrWhiteSpace(newTitle) && newTitle !=
SelectedNote.Title)
    {   SelectedNote.Title = newTitle;
```

```
        _databaseWriter.UpdateNote(SelectedNote);
        OnPropertyChanged(nameof(SelectedNote));
        LoadNotes();
    }
}
```

Через цей клас викликаються методи створення повідомлень. Зберігаючи нотатку, користувач активує подію для її перезапису в БД, це наглядно показано на лістингу 3.20.

Лістинг 3.20 – Основний цикл збереження

```
for (int i = 0; i < newLines.Length; i++){
    if (i < rows.Count){
        if (rows[i].Content != newLines[i]){
            rows[i].Content = newLines[i];
            updatedRows.Add(rows[i]);}}
    else{   var newRow = new NoteRow{
            NoteId = SelectedNote.Id,
            Content = newLines[i],
            Order = i};
        newRows.Add(newRow);}}
```

Також у цьому класі прописана взаємодія курсору з полем для введення інформації нотаток (лістинг 3.21) за для змінення розміру тексту чи його стилю.

Лістинг 3.21 – Метод оновлення положення курсору

```
private int _noteTextBoxCaretIndex;
public int NoteTextBoxCaretIndex
{
    get => _noteTextBoxCaretIndex;
    set
    {        _noteTextBoxCaretIndex = value;
        OnPropertyChanged();    }
}
```

Ще одна з найважливіших подій разом з методом прописані тут. Вона відповідає за відправку повідомлень до View частини інтерфейсу (лістинг 3.22), які сповіщають інтерфейс, що значення певної властивості змінилося.

Лістинг 3.22 – Подія та метод

```
public event PropertyChangedEventHandler PropertyChanged;
protected void OnPropertyChanged([CallerMemberName] string
propertyName = null)
{
    PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
}
```

3.7 Файли кастомізації

Загалом проект містить 1 головний файл кастомізації та 3 додаткові. Файл ico.ico – це іконка програми, яка відображається в лівому верхньому куті програми та в панелі задач на комп’ютері.

Далі йде два файли DarkTheme.xaml та LightTheme.xaml. Вони відповідають за теми додатку, а саме світлу та темну. В них зберігається інформація про кольорові значення (лістинг 3.23) для всіх частин інтерфейсу, а також пензли (лістинг 3.24) для прив’язки кольорів до елементів інтерфейсу. Ці файли дозволяють централізовано керувати кольорами в інтерфейсу, не переписуючи код для кожного окремого випадку. Це зроблено для зручної зміни тем у програмі.

Лістинг 3.23 – Кольорові значення темної теми

```
    <Color x:Key="BackgroundBaseColor">#1E1E2F</Color>
    <Color x:Key="TextBaseColor">#FFFFFF</Color>
    <Color x:Key="AccentBaseColor">#4CAF50</Color>
    <Color x:Key="InputBaseColor">#2D2D40</Color>
    <Color x:Key="BorderBaseColor">#4A4A70</Color>
    <Color x:Key="ErrorBaseColor">#FF6B6B</Color>
```

Лістинг 3.24 – Пензли світлої теми

```
    <SolidColorBrush x:Key="WindowBackground"
Color="{StaticResource BackgroundBaseColor}"/>
    <SolidColorBrush x:Key="TextColor" Color="{StaticResource
TextBaseColor}"/>
    <SolidColorBrush x:Key="AccentColor" Color="{StaticResource
AccentBaseColor}"/>
    <SolidColorBrush x:Key="InputBackground"
Color="{StaticResource InputBaseColor}"/>
    <SolidColorBrush x:Key="BorderBrushColor"
```

```
Color="{StaticResource BorderBaseColor}"/>
    <SolidColorBrush x:Key="ErrorColor" Color="{StaticResource
ErrorBaseColor}"/>
    <SolidColorBrush x:Key="MenuBackground" Color="#F0F0F0"/>
```

Головним файлом проєкту є App.xaml (лістинг 3.25), який відповідає за коректне підключення ресурсів, також в ньому зберігається інформація про параметри запуску програми. Файл App.xaml є головним елементом, який пов’язує описи тем з програмою та визначає порядок запуску вікон.

Лістинг 3.25 – Код файлу App.xaml

```
<Application x:Class="Oraton.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:Oraton"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="Themes/DarkTheme.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>

</Application>
```

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

Після запуску додатку "Oraton" користувач побачить головне вікно програми з двома кнопками (рисунок 4.1), які відповідають за реєстрацію та вхід. А також на цьому вікні є два елементи меню "Довідка" та "Тема".

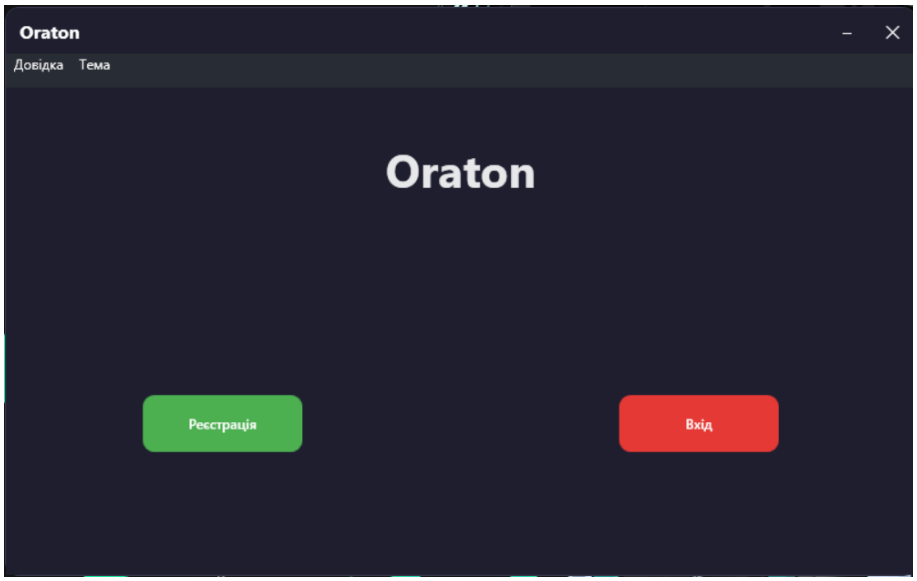


Рисунок 4.1 – Головне вікно програми

Натиснувши на "Довідка" (рисунок 4.2), користувач отрумує інформацію стосовно розробника програми та її версію.

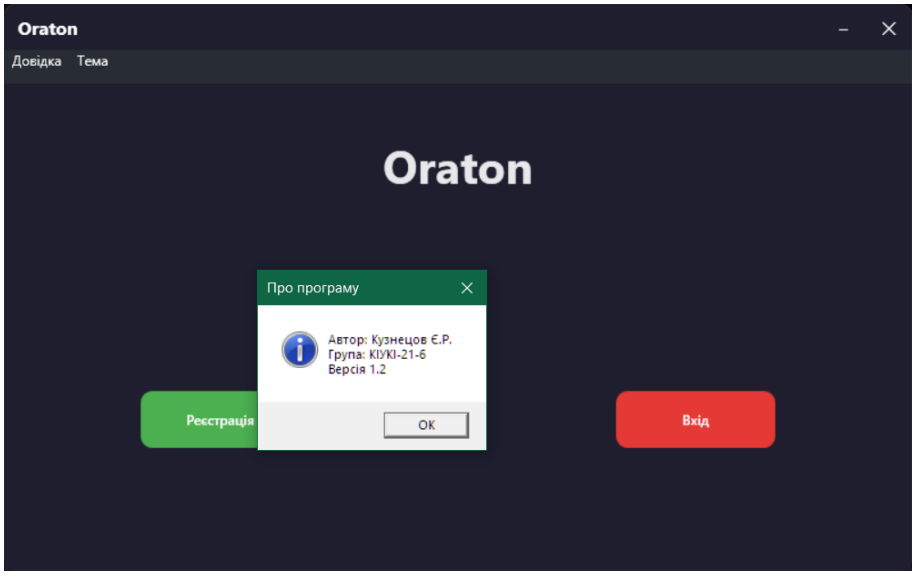


Рисунок 4.2 – Інформація з «Довідка»

У програмі доступні дві теми темна (вона використовується за замовчуванням) та світла. Для їх зміни потрібно натиснути "Тема" (рисунок 4.3) та обрати тему підходящу під потребу користувача. Надалі буде використовуватися темна тема.

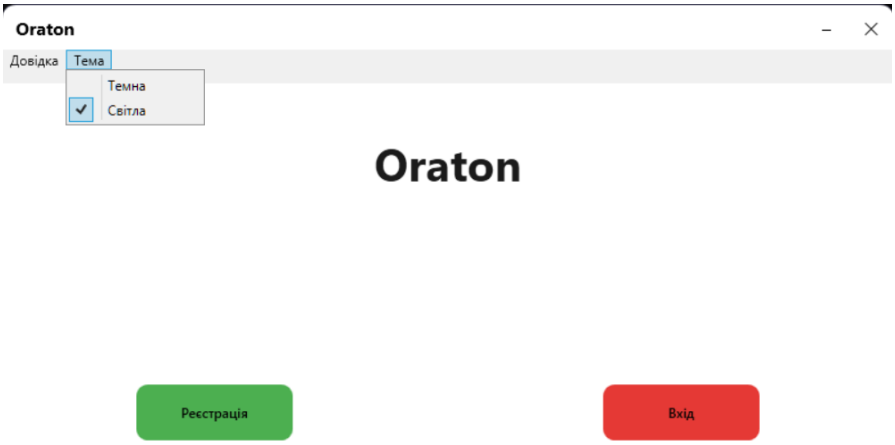


Рисунок 4.3 – Світла тема додатку

При натисканні кнопки "Реєстрація" користувач потрапляє на сторінку реєстрації (рисунок 4.4), де йому потрібно ввести ім’я користувача, Email (електрону пошту) та пароль.

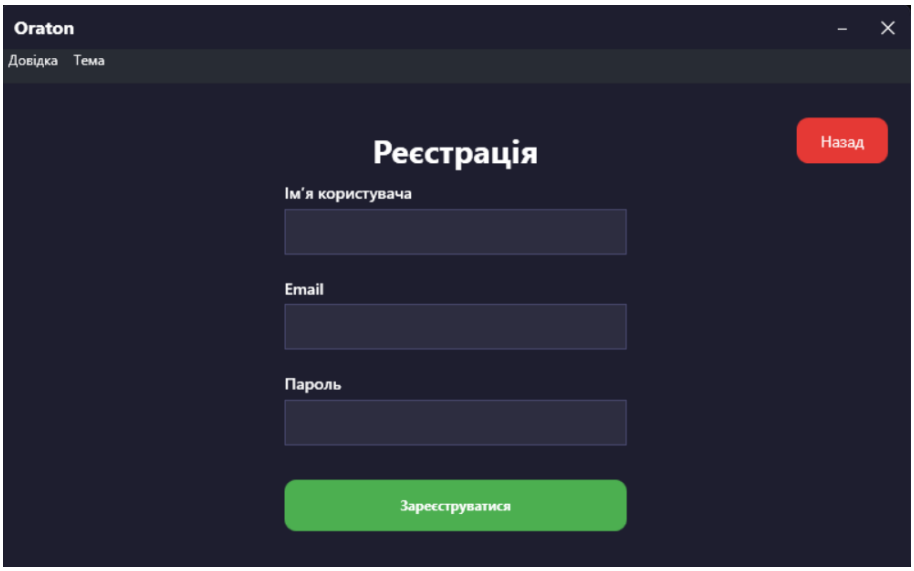


Рисунок 4.4 – Вікно реєстрації

Після введення цих даних користувач побачить віконце з написом "Реєстрація успішно" (рисунок 4.5) та його буде "переведено" до вікна входу (рисунок 4.6).

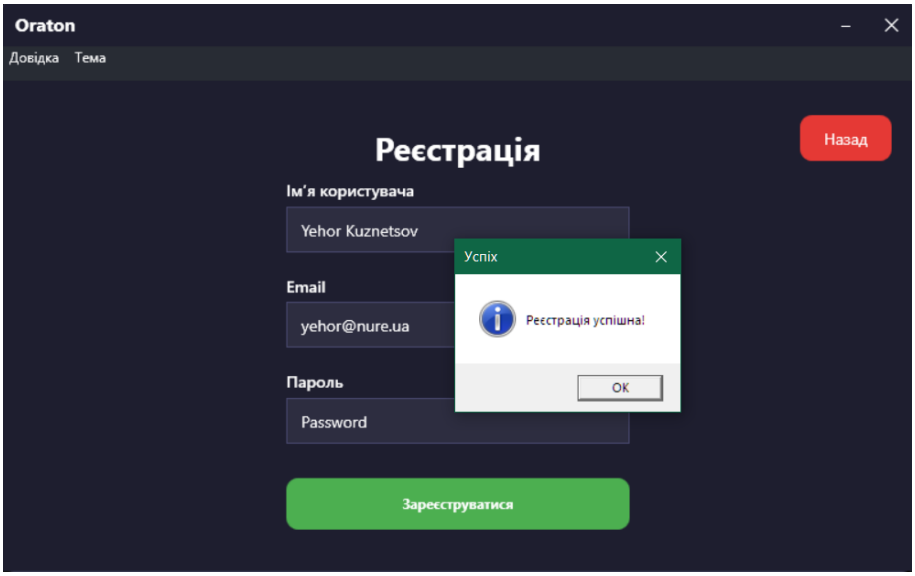


Рисунок 4.5 – Підтвердження успішної реєстрації

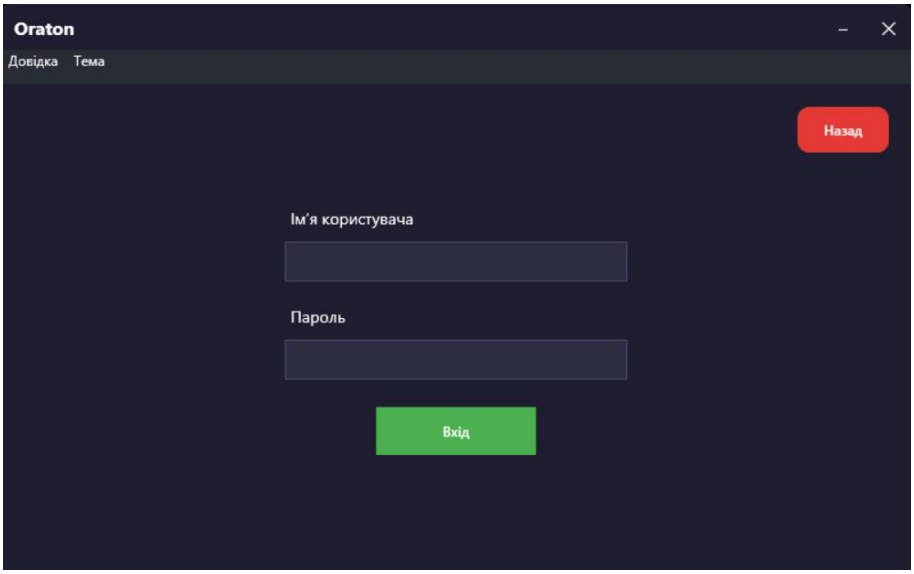


Рисунок 4.6 – Вікно входу в додаток

Якщо при реєстрації користувач введете вже зареєстрований Email чи ім’я користувача, програма сповістить його про це та попросить ввести новий Email (рисунок 4.7) чи ім’я користувача (рисунок 4.8).

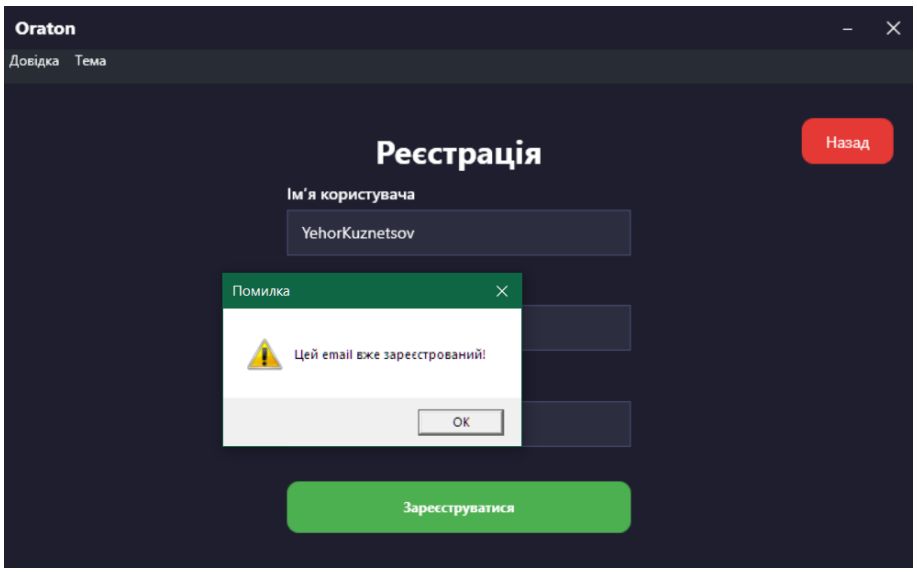


Рисунок 4.7 – Помилка 1

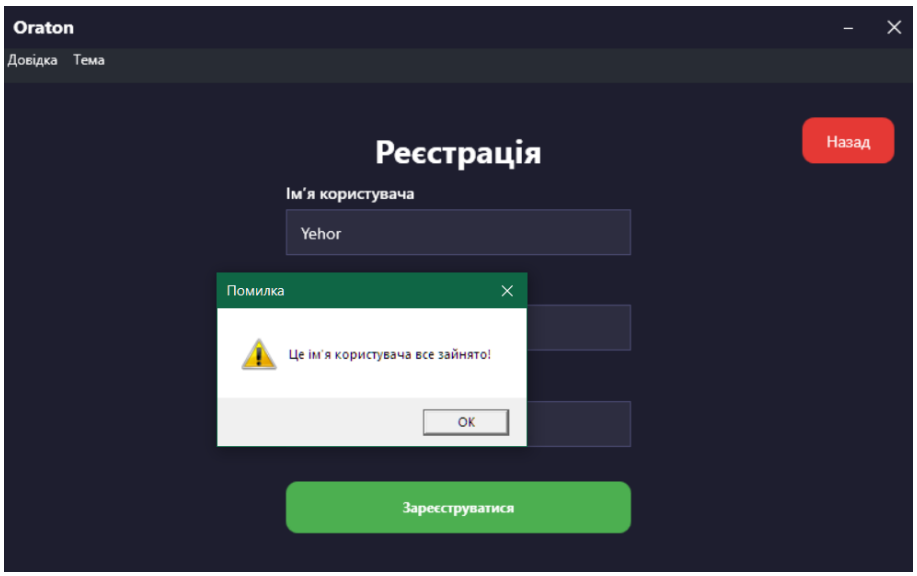


Рисунок 4.8 – Помилка 2

Також на вікно входу можна попасти натиснувши на кнопку «Вхід» на першому вікні програми. В цьому вікні потрібно ввести ім’я користувача та пароль для входу в свій обліковий запис. Після правильного введення даних користувача "перекине" на основне вікно програми (рисунок 4.9).

На цьому вікні розташовані декілька кнопок, список нотаток, поле для вводу замітки та два нерозгорнутих списків, біля одного написано "Шрифт", а біля другого "Розмір шрифту". Під написом "Список нотаток" будуть відображатися всі нотатки користувача. Користувач може обрати одну з них,

натиснути на неї і тоді під написом "Текст нотаток" виведеться вміст обраного нотатку (рисунок 4.10)

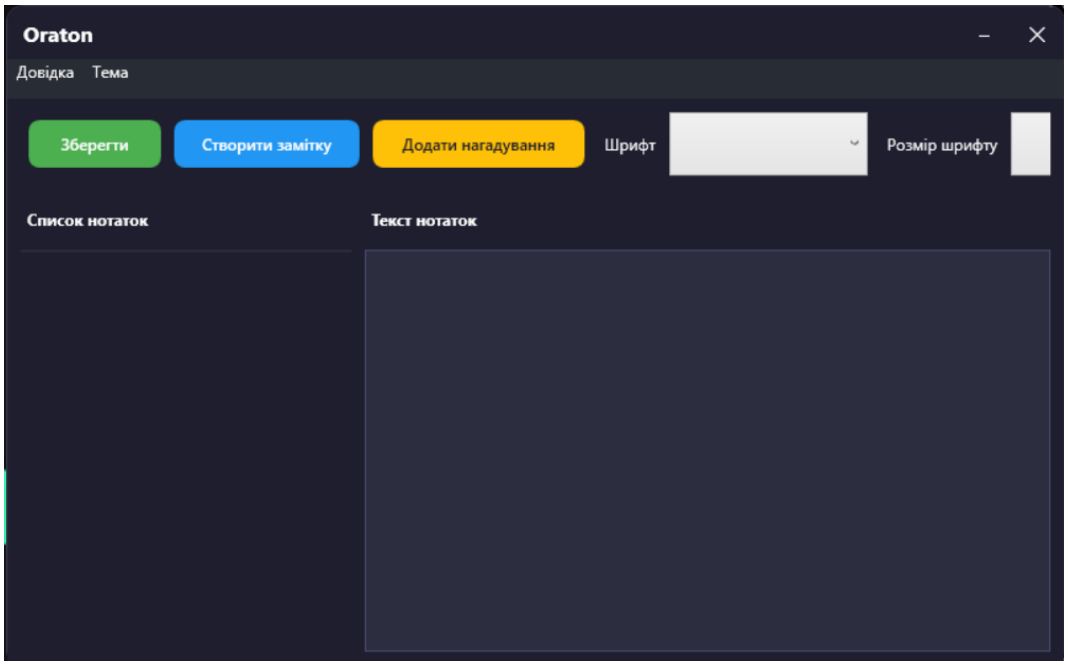


Рисунок 4.9 – Основне вікно програми

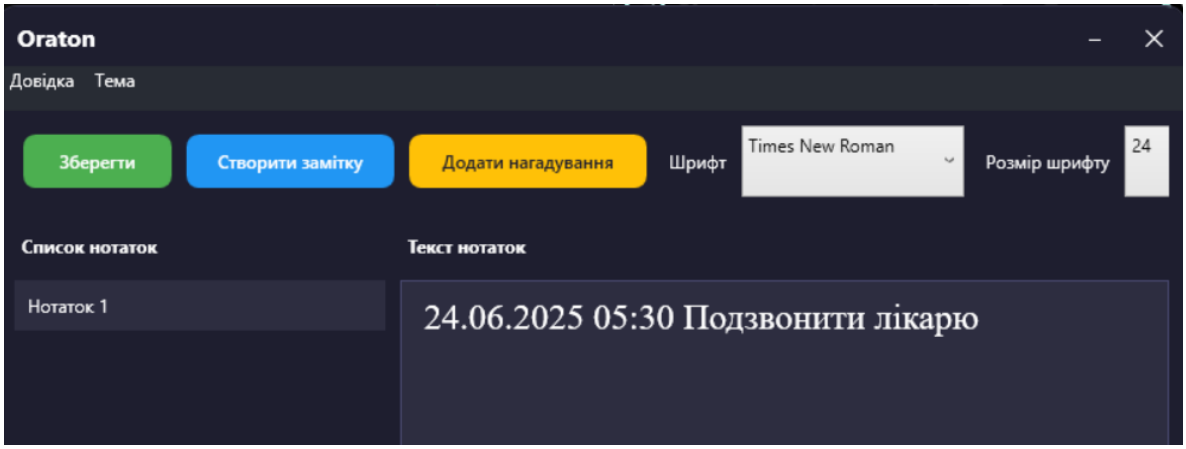


Рисунок 4.10 – Виведення вмісту додатку

Але для початку потрібно створити свою першу нотатку, це можна зробити за допомогою кнопки "Створити замітку". Натиснувши правою кнопкою миші на створену замітку її можна переіменувати чи видалити за допомогою контекстного меню, яке складається з двух пунктів "Переіменувати" та "Видалити" (рисунок 4.11).

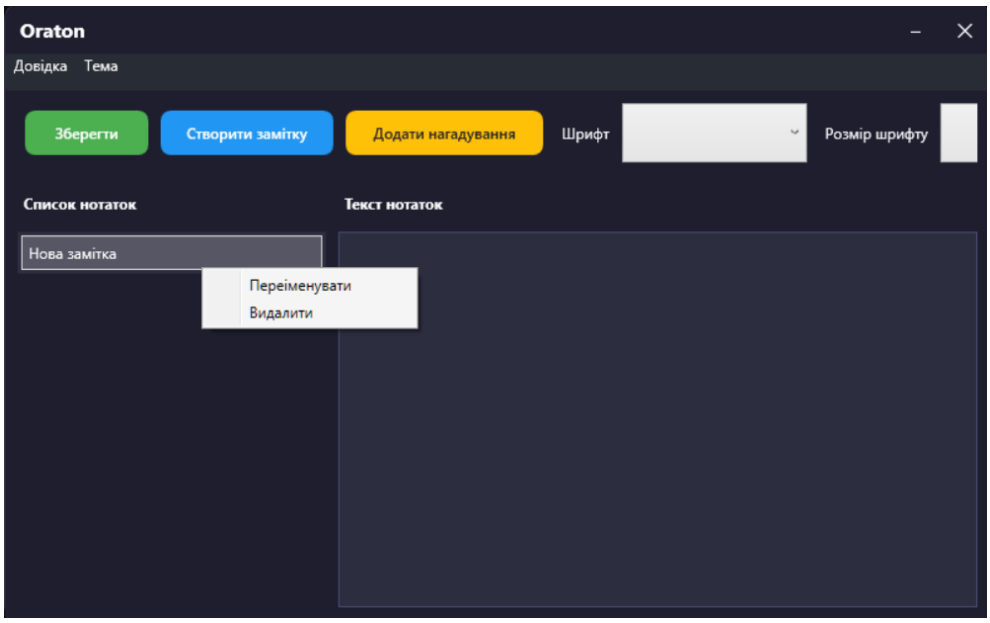


Рисунок 4.11 – Контексне меню

Під написом "Текст нотаток" можна вводити будь-який текст. Після завершення введення тексту, рекомендується власноруч зберегти нотатку за допомогою зеленої кнопки "Зберегти" (рисунок 4.12).

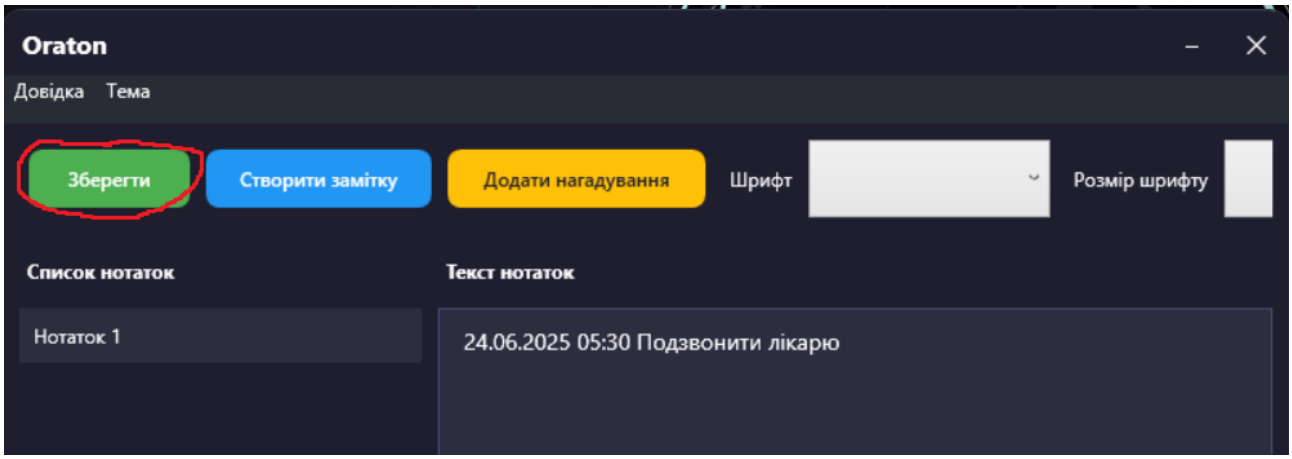
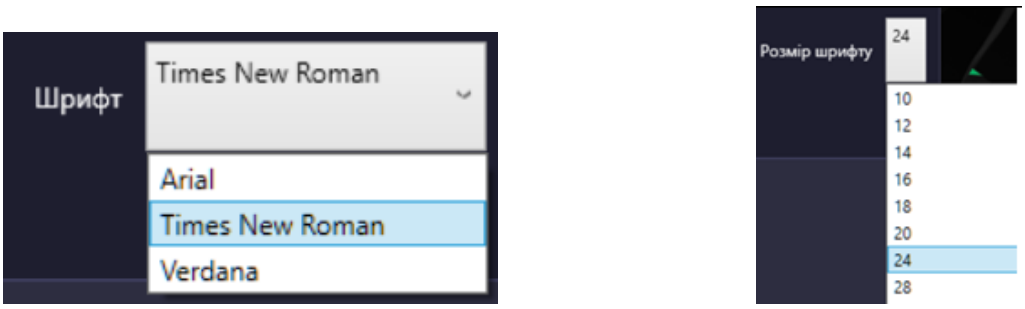


Рисунок 4.12 – Ручне збереження нотатку

Також додаток "Oraton" надає можливість редагування тексту за допомогою різгих шрифтів (рисунок 4.13 а) та зміненя розміру шрифтів (рисунок 4.13 б). Така можливість надає користувачу додаткові можливості форматування зовнішнього вигляду нотаток за своїм бажанням.



а) б)

Рисунок 4.13 – Редагування нотаток: а) типу шрифту; б) розміру шрифту

Створивши нотатку та виділивши час та дату, можна встановити нагадування, використовуючи жовту кнопку "Додати нагадування" (рисунок 4.14), яке вчасно сповістить користувача в зазначений час (рисунок 4.15).

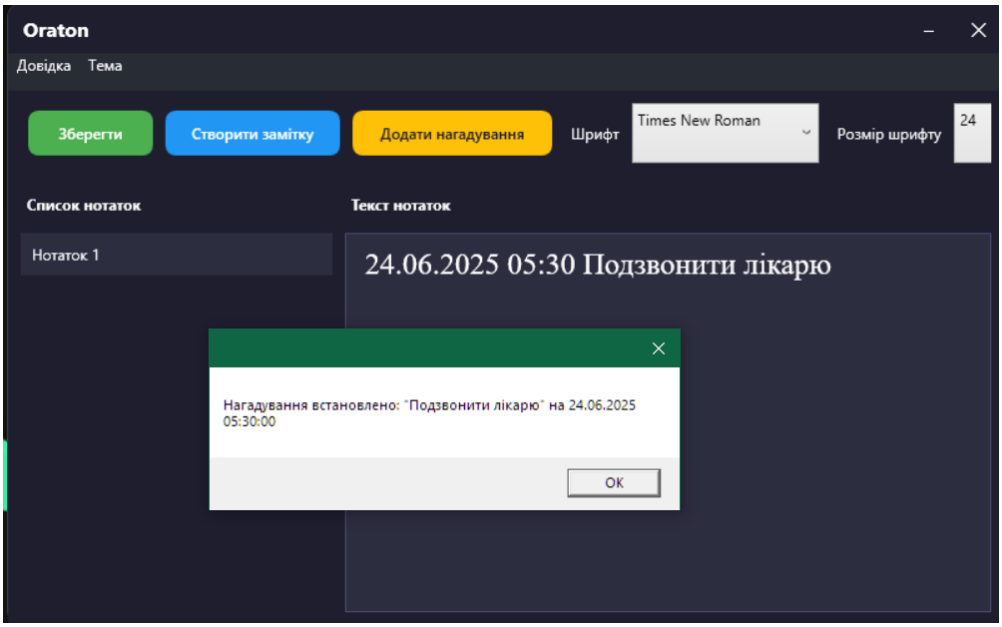


Рисунок 4.14 – Сповіщення встановлено

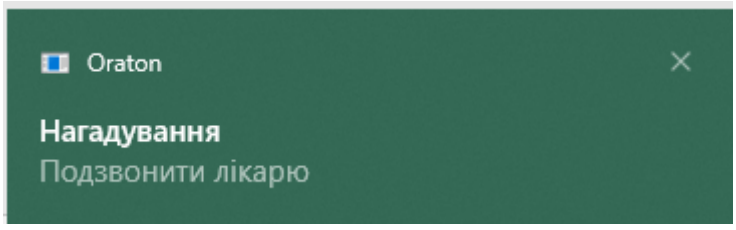


Рисунок 4.15 – Сповіщення спрацювало

Якщо користувач обере неправильно записаний час чи дату, програма сповістить його про неможливість створення повідомлення (рисунок 4.16).

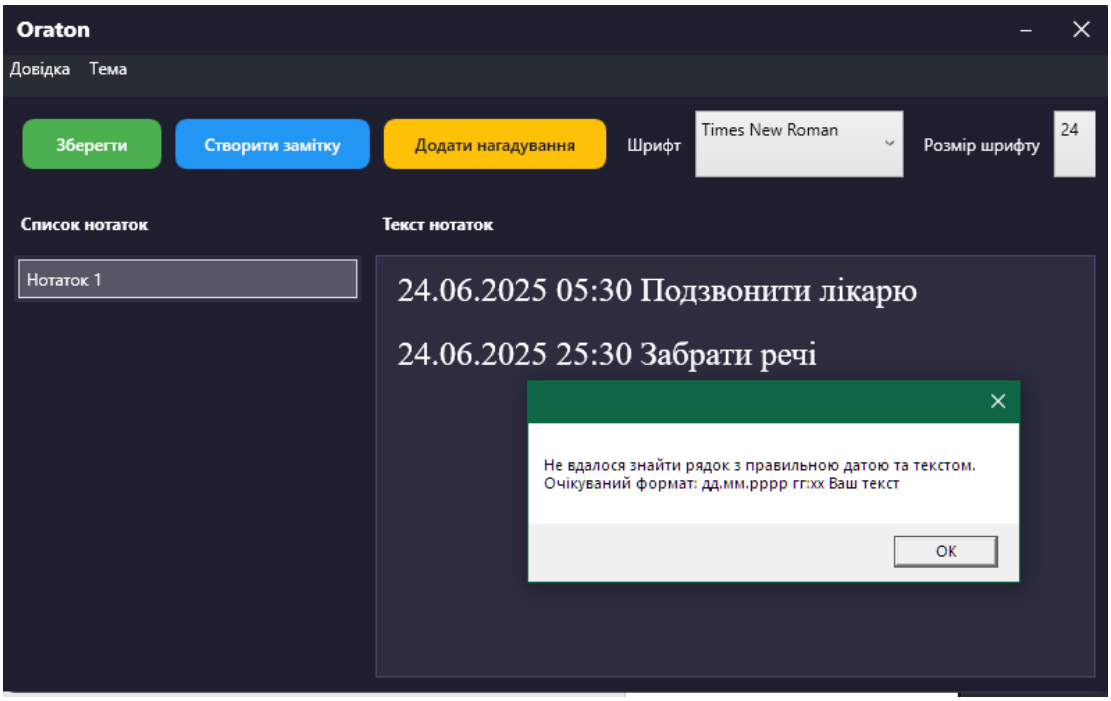


Рисунок 4.16 – Помилка 3

Загалом програма Oraton є інтуітивно зрозумілою, що полегшує взаємопрацю із нею.

ВИСНОВКИ

У процесі виконання кваліфакаційної роботи було створено додаток під назвою «Oraton» – інструмент для зберігання, організації та структурування особистої або робочої інформації, зокрема текстових нотаток. Його головна мета – забезпечити користувача зручним засобом управління записів та повсякденними нотатками у зручному інтерфейсі.

Розробка здійснювалася мовою програмування C# у середовищі Microsoft Visual Studio 2022. Для побудови графічного інтерфейсу було використано Windows Presentation Foundation (WPF), що дозволило створити сучасний, гнучкий та візуально привабливий інтерфейс із підтримкою стилізації, анімацій та адаптивності.

Збереження даних реалізовано на основі бази даних PostgreSQL, що забезпечило надійність та ефективність обробки інформації. Для зручності роботи з базою активно використовувався інструмент DataGrip, який спростив написання SQL-запитів, перегляд структури даних та керування зв’язками між таблицями.

Особливу увагу приділено досвіду користувача – реалізовано можливості створення, редагування та видалення нотаток, пошуку та сортування інформації, створення нагадувань, а також зміну зовнішнього вигляду тексту (наприклад, вибір шрифту), що дозволяє адаптувати інтерфейс під власні потреби.

У ході розробки довелося долати технічні труднощі, пов’язані з реалізацією логіки додатку та побудовою його архітектури. Це вимагало глибокого аналізу, пошуку оптимальних рішень та вдосконалення професійних навичок.

Загалом, реалізація проєкту "Oraton" стала цінним досвідом, що дозволив закріпити знання з програмування, роботи з базами даних і розробки інтерфейсів. Проєкт став важливим етапом у професійному зростанні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Офіційний сайт Notion. URL: https://www.notion.com

2. Офіційний сайт Obsidian. URL: https://obsidian.md

3. Офіційний сайт Microsoft OneNote. URL: https://www.onenote.com/?public=1&wdorigin=ondcauth2&wdorigin=ondc

4. Patrick T. Start to Finish Visual C# 2015. Owani Press, 2016. P. 592.

5. Albahari J., Albahari B. C# 7.0 in a Nutshell: The Definitive Reference. O'Reilly Media, 2017, 1st Edition. P. 1087.

6. Lukosek G. Learning C# by Developing Games with Unity 5.x - Second Edition: Develop your first interactive 2D platformer game by learning the fundamentals of C#. Packt Publishing, 2016, 2nd Edition. P.230.

7. Troelsen A., Japikse P. Pro C# 7: With .NET and .NET Core. Apress, 8st Edition, 2017. P. 1437.

8. Albahari J., Albahari B. LINQ Pocket Reference: Learn and Implement LINQ for .NET Applications. Publisher: O'Reilly Media. 2008. 217 p.

9. Windows Presentation Foundation documentation. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/

10. Nathan A. XAML Unleashed 1st Edition. Publisher: Sams Publishing. 2014. 512 p.

11. Ferrari L., Pirozzi E. Learn PostgreSQL – Second Edition: Use, manage and build secure and scalable databases with PostgreSQL. 2nd ed. Packt Publishing, 2023. 744 p.

12. Nijs P. The MVVM Pattern in .NET MAUI: The definitive guide to essential patterns, best practices, and techniques for cross-platform app development. Publisher: Packt Publishing, 2023. 386 p.

13. Farmer M., Sander R. C# for Object-Oriented Programming Unlock the Power of OOP: A Comprehensive Guide to Mastering C# and Object-Oriented Design. Publisher: Independently published. 2025. 463 p.