ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНІКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Магістр»

«Розробка десктопного додатку для створення нотаток з хмарною синхронізацією»

(тема кваліфікаційної роботи українською мовою)

«Development of a desktop application for creating notes with cloud synchronization»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання
спеціальності 122 Комп'ютерні науки
(код, назва спеціальності)
Освітня програма Комп'ютерні науки
(назва)
Степанець Михайло Сергійович
(прізвище, ім'я, по-батькові здобувача)

Керівник к.т.н., доцент Фразе-Фразенко О.О. _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент кафедри інформаційних технологій НУ ОЮА, Гура В.І.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій
№ ___ від ___ _____ 2024 р.

Завідувачка кафедри

_____ _____
(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № ____
протокол № ___ від _____ 2024 р.

Оцінка _____ / _____ / _____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ _____
(підпис) (прізвище,ім'я)

Одеса 2024

АНОТАЦІЯ

У кваліфікаційній роботі розробляється тема «Розробка десктопного додатку для створення нотаток з хмарною синхронізацією».

Мета роботи – підвищення зручності та безпеки управління нотатками шляхом створення десктопного додатку, що поєднує локальне зберігання з функцією синхронізації через хмарну базу даних. Особлива увага приділяється захисту даних шляхом їх автоматичного шифрування, а також зручності користувача через розробку інтуїтивно зрозумілого інтерфейсу.

У результаті роботи створено програмний продукт, який забезпечує багатофункціональне управління текстовими та мультимедійними записами, їхню безпечну синхронізацію та доступ із різних пристроїв. Додаток використовує сучасні технології для ефективного зберігання даних і підтримує кросплатформність завдяки інтеграції з хмарними сервісами.

ABSTRACT

The qualification work explores the topic "Development of a desktop application for note-taking with cloud synchronization".

The aim of the work is to enhance the convenience and security of note management by developing a desktop application that combines local storage with cloud database synchronization functionality. Special focus is placed on data protection through automatic encryption and user convenience with an intuitive interface design.

As a result of the research, a software product was developed, providing multifunctional management of textual and multimedia notes, secure synchronization, and access from various devices. The application employs modern technologies for efficient data storage and supports cross-platform functionality through integration with cloud services.

ЗМІСТ

ТЕРМІНИ, СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Архітектура веб-додатка –організація компонентів, модулів та взаємодії між ними веб-додатка.

Клієнт – це програмне забезпечення (зазвичай веб-браузер), що звертається до веб-сервера для отримання веб-сторінок та інших ресурсів.

БД – База даних.

C# (C-Sharp) – сучасна об'єктно-орієнтована мова програмування.

SQLite – легка система керування базами даних (СКБД) без сервера.

MySQL – потужна реляційна система керування базами даних з відкритим кодом.

СКБД – система керування базами даних.

.NET – програмна платформа, розроблена Microsoft, що забезпечує інструменти та середовище виконання для створення різноманітних додатків

IDEF0 –Integration Definition for Function Modeling.

SADT – Structured Analysis and Design

API – Application Programming Interface

Хешування – метод шифрування даних.

JSON – JavaScript Object Notation.

GUI – Graphical User Interface.

Клієнт-серверна архітектура – модель взаємодії в системах, де клієнт запитує ресурси або послуги, а сервер надає їх.

Інкапсуляція – передбачає приховування деталей реалізації та забезпечення доступу через інтерфейси.

ВСТУП

У сучасному світі інформація стала ключовим ресурсом, що потребує надійного, зручного та швидкого інструмента для її збереження й обробки. Саме цю роль виконують додатки для ведення нотаток. Вони створені для оптимізації особистої та професійної діяльності, надаючи користувачам можливість впорядковувати ідеї, відстежувати задачі та зберігати важливу інформацію в одному місці. Завдяки стрімкому розвитку технологій, такі додатки поступово перетворилися з простих текстових редакторів на багатофункціональні платформи, здатні синхронізувати дані, працювати з мультимедійним контентом і забезпечувати захист інформації. Розуміння основних принципів та методологій розробки інформаційних систем дозволить ефективно використовувати їх потенціал і побудувати майбутнє, в якому інформація стане ще доступнішою, швидкою та силами трансформувати світ навколо.

Кожен користувач має власні потреби й очікування від подібного програмного забезпечення. Деякі шукають простоту та швидкість, як це пропонує Google Keep, тоді як інші надають перевагу потужним інструментам для організації, таким як Evernote або Notion. Попри широкий вибір подібних рішень, всі вони мають як переваги, так і недоліки, які залежать від специфіки застосування.

Організація інформації та збереження даних є невід'ємною частиною сучасного життя, як у професійній сфері, так і в повсякденному використанні. Усе більше користувачів звертаються до цифрових інструментів, таких як додатки для ведення нотаток, які пропонують зручний спосіб упорядкування ідей, відстеження завдань і швидкого доступу до записів. Попит на такі програми обумовлений зростаючими потребами в мобільності, доступності та безпеці інформації.

У рамках даної роботи було проведено детальний аналіз існуючих додатків для ведення нотаток, що включав вивчення їхніх функцій, переваг і недоліків. До уваги взято як популярні хмарні рішення, такі як Evernote, Google Keep, так і локальні додатки з можливістю синхронізації. Аналіз базувався на ключових аспектах, важливих для користувачів: простота використання, можливості інтеграції, доступність офлайн-режиму, захищеність даних і багатофункціональність.

Метою роботи є створення десктопного додатку для ведення нотаток, що поєднує зручність локального зберігання з можливістю хмарної синхронізації, забезпечуючи мобільність і захищеність даних.

Об'єктом дослідження є процес управління особистою та робочою інформацією за допомогою цифрових інструментів.

Предметом дослідження є методи і технології розробки десктопних додатків із хмарною інтеграцією для забезпечення безперервного доступу до даних.

Кваліфікаційна робота магістра містить 7 таблиць, 23 рисунків та 9 джерел посилань.

1 ОПИС ВИБОРУ АРХІТЕКТУРИ ТА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ВЕБ-СЕРВІСУ

1.1 Аналіз загальної архітектури додатку

Розробка десктопного додатку для нотаток з хмарною синхронізацією – це завдання, яке вирішує питання збереження та управління особистою чи робочою інформацією в цифровому вигляді. У сучасному світі інформація надходить з багатьох джерел і різними способами: робочі проєкти, особисті записи, зустрічі, ідеї, списки справ тощо. У зв'язку з цим виникає необхідність організувати та систематизувати цю інформацію, щоб вона була легко доступною з будь-якого пристрою та в будь-який час.

Для багатьох користувачів важливо мати єдине місце, де вони можуть зберігати та управляти своїми записами, при цьому не турбуючись про безпеку даних чи можливість втрати інформації. Однак, звичайний локальний додаток для нотаток обмежений лише одним пристроєм. Це ускладнює доступ до нотаток з інших пристроїв та створює ризик втрати даних у разі пошкодження чи втрати комп'ютера. Тому поєднання можливостей десктопного додатку з хмарною синхронізацією стає важливим рішенням для забезпечення надійного зберігання нотаток.

Основна мета цього проєкту – створити десктопний додаток для нотаток, який дозволить користувачам легко створювати, редагувати, видаляти та переглядати свої записи. На відміну від багатьох інших рішень, цей додаток фокусується на простоті та ефективності роботи з нотатками, забезпечуючи швидку синхронізацію між локальним сховищем та хмарою. Така синхронізація дозволить отримувати доступ до нотаток з різних пристроїв, що підвищить зручність та мобільність роботи з даними.

Додаток передбачає можливість створення нотаток з різноманітним вмістом, а також їх просте редагування та видалення за потреби.

Десктопний додаток було реалізовано на мові програмування C#. Цей вибір обумовлений численними перевагами, які надає ця мова та її екосистема. C# є однією з найпотужніших і найпопулярніших мов програмування для розробки десктопних додатків завдяки її зручному синтаксису, підтримці об'єктно-орієнтованого підходу та інтеграції з .NET Framework, який надає величезний набір інструментів і бібліотек для швидкої й ефективної реалізації складних функцій.

Основною перевагою використання C# є його багатофункціональність, що дозволяє створювати програми з багатим інтерфейсом користувача, забезпечувати високу продуктивність і підтримувати роботу з базами даних. Крім того, завдяки інтегрованому середовищу розробки Visual Studio, процес розробки додатка стає інтуїтивно зрозумілим і зручним, що дозволяє розробникам фокусуватися на реалізації функціоналу, а не на вирішенні технічних нюансів.

Десктопний додаток, створений на C#, підтримує кросплатформеність завдяки .NET Core та .NET 6, що дозволяє запускати його на різних операційних системах. Це забезпечує більшу гнучкість і доступність для користувачів. Крім того, мова C# має багату підтримку роботи з базами даних, що було критично важливо для реалізації функцій збереження нотаток як локально, так і в хмарі. Завдяки бібліотекам, таким як System.Data.SQLite і MySql.Data, вдалося ефективно впровадити взаємодію з локальною та хмарною базами даних, забезпечуючи швидкий доступ до інформації та її синхронізацію.

Ще однією важливою перевагою є підтримка сучасних методів шифрування, що гарантує безпеку даних користувачів. Мова C# дозволяє легко інтегрувати алгоритми хешування паролів і шифрування контенту нотаток, завдяки чому реалізовано захист даних як у локальній базі, так і під час передачі до хмари.

Таким чином, вибір C# для реалізації десктопного додатка дозволив створити стабільний, функціональний та інтуїтивно зрозумілий продукт, який відповідає сучасним вимогам і очікуванням користувачів.

Десктопний додаток для ведення нотаток реалізовано з використанням дворівневої архітектури, яка є поширеним типом клієнт-серверної архітектури, але з особливим підходом до взаємодії з базами даних. Основна робота з даними відбувається на рівні локальної бази SQLite, що забезпечує високу швидкість доступу та мінімальні затримки при обробці даних. Хмарна база даних MySQL виконує роль резервного сховища для синхронізації даних і забезпечення доступу до них із різних пристроїв.

Дворівнева архітектура включає два основних компоненти:

- клієнт – десктопний додаток, що виконується на пристрої користувача, забезпечує інтерфейс для створення, редагування й видалення нотаток. Клієнт також відповідає за передачу SQL-запитів до серверної бази даних і обробку отриманих результатів;
- база даних – сервер бази даних My SQL, що зберігає всі текстові нотатки, зображення, дані користувачів і забезпечує зручний доступ до них за запитами клієнта.

1.2 Характеристика обраної системи управління локальною базою даних

Під час розробки додатку «Нотатки», для локального збереження нотаток було обрано вбудовану систему керування базами даних SQLite, яка відзначається своєю легкістю, самодостатністю та високою надійністю.

SQLite — це вбудована бібліотека, яка реалізує самодостатній безсерверний механізм транзакційної бази даних SQL без конфігурації. Код для SQLite знаходиться у відкритому доступі, тому його можна безкоштовно

використовувати для будь-яких цілей, комерційних чи приватних. SQLite — це найпоширеніша база даних у світі з більшою кількістю програм, ніж ми можемо порахувати, включаючи кілька гучних проектів.

SQLite — це вбудований механізм бази даних SQL. На відміну від більшості інших баз даних SQL, SQLite не має окремого серверного процесу. SQLite читає та записує безпосередньо у звичайні дискові файли. Повна база даних SQL із кількома таблицями, індексами, тригерами та представленнями даних міститься в одному дисковому файлі. Формат файлу бази даних є крос-платформним — ви можете вільно копіювати базу даних між 32-розрядними та 64-розрядними системами або між архітектурами big-endian та little-endian. Ці функції роблять SQLite популярним вибором як формат файлу програми[1].

Основні характеристики SQLite:

- вбудованість: SQLite функціонує як бібліотека, яку можна безпосередньо вбудувати в додаток, що усуває потребу в окремому сервері бази даних;
- самодостатність: вся база даних зберігається в одному файлі, що спрощує управління та перенесення даних;
- відсутність налаштувань: SQLite не потребує складної конфігурації або адміністрування, що робить її зручною для розробників;
- підтримка стандарту SQL: SQLite підтримує більшість стандарту SQL-92, що забезпечує знайомий синтаксис для розробників та спрощує міграцію даних.

Переваги використання SQLite:

- швидкість роботи: завдяки відсутності мережевих затримок та оптимізованій архітектурі, SQLite забезпечує високу швидкість читання та запису даних[2];

- простота використання: відсутність необхідності в налаштуванні та адмініструванні робить SQLite зручним вибором для розробників, особливо на початкових етапах проєкту[2];
- портативність: база даних зберігається в одному файлі, що спрощує її перенесення між різними системами та резервне копіювання.

Недоліки використання SQLite:

- обмежена підтримка багатокористувацького доступу: SQLite не підходить для систем з високим рівнем одночасних записів, оскільки підтримує лише один запис одночасно;
- обмежена масштабованість: для великих проєктів з великими обсягами даних та високими вимогами до продуктивності краще використовувати інші СКБД, такі як MySQL або PostgreSQL.

Варто зазначити, що в межах даного використання СКБД SQLite, згадані недоліки жодним чином не впливають негативно на роботу додатку. Це зумовлено тим, що система розрахована на одного користувача, а навантаження на базу даних буде мінімальним, що виключає можливість перевантаження через надмірну кількість запитів. Таким чином, обрана СКБД ідеально відповідає потребам додатку та забезпечує його стабільну й ефективну роботу.

1.3 Характеристика обраної системи управління хмарною базою даних

Під час розробки додатку «Нотатки», для хмарного збереження нотаток було обрано потужну та гнучку СКБД – MySQL, яка підходить для широкого спектра застосувань, від невеликих додатків до великих корпоративних систем. Її висока продуктивність, надійність та активна спільнота роблять її привабливим вибором для багатьох проєктів. Однак, перед вибором MySQL варто врахувати специфічні вимоги проєкту та можливі обмеження системи.

MySQL — найпопулярніша у світі система керування базами даних з відкритим кодом, яка використовується для зберігання та управління даними в різноманітних програмних додатках. Вона забезпечує високу продуктивність, надійність та масштабованість, що робить її ідеальним вибором для проєктів будь-якого масштабу — від невеликих особистих проєктів до критично важливих корпоративних систем[3].

Вона відзначається високою продуктивністю, надійністю та гнучкістю, що робить її одним із найпоширеніших виборів серед розробників.

Основні характеристики MySQL:

- реляційна модель даних: MySQL використовує реляційну модель, що дозволяє зберігати дані в таблицях та встановлювати зв'язки між ними, забезпечуючи структуроване та ефективне управління інформацією[4];
- кросплатформеність: MySQL може працювати на різних операційних системах, включаючи Windows, Linux, macOS та інші, що забезпечує гнучкість у виборі платформи для розгортання[4];
- підтримка стандарту SQL: MySQL підтримує більшість стандарту SQL-99, що забезпечує знайомий синтаксис для розробників та спрощує міграцію даних між різними СКБД[5];
- масштабованість: MySQL здатна обробляти великі обсяги даних та підтримувати велику кількість одночасних підключень, що робить її придатною для використання в системах з високим навантаженням[5];
- розширюваність: завдяки модульній архітектурі, MySQL підтримує різні механізми зберігання даних (storage engines), такі як InnoDB, MyISAM та інші, що дозволяє налаштовувати СКБД відповідно до специфічних вимог проєкту[5].

Переваги використання MySQL:

- висока продуктивність: оптимізована для швидкої обробки запитів, MySQL забезпечує швидкий доступ до даних та ефективне виконання операцій читання та запису;
- надійність та стійкість: підтримка транзакцій та механізмів відновлення після збоїв забезпечує цілісність даних та мінімізує ризик їх втрати;
- гнучкість налаштувань: можливість вибору різних механізмів зберігання даних та налаштування параметрів сервера дозволяє оптимізувати роботу СКБД під конкретні потреби додатку;
- широка підтримка спільноти: велика кількість документації, форумів та ресурсів, а також активна спільнота розробників сприяють швидкому вирішенню проблем та обміну досвідом.

Недоліки використання MySQL:

- обмежена підтримка складних запитів: у порівнянні з іншими СКБД, такими як PostgreSQL, MySQL може мати обмеження у виконанні складних запитів та підтримці деяких розширених функцій;
- ліцензійні обмеження: хоча MySQL є відкритим програмним забезпеченням, деякі розширені функції доступні лише в комерційних версіях, що може вимагати додаткових витрат.

MySQL є потужною та гнучкою СКБД, яка підходить для широкого спектра застосувань. Її висока продуктивність, надійність та активна підтримка спільноти роблять її привабливим вибором для розробників, які шукають ефективне рішення для управління даними.

1.4 Аналіз та опис мови програмування для написання проєкту

C# — одна з найпопулярніших мов програмування, створена для розробки додатків у рамках платформи .NET. Вона була представлена корпорацією

Microsoft у 2000 році як сучасна об'єктно-орієнтована мова, яка забезпечує простоту використання та широкі можливості для створення різноманітних програм і програмного забезпечення[6].

C# є статично типізованою мовою, що підтримує принципи об'єктно-орієнтованого програмування (ООП), а також функціональне, імперативне й подієво-орієнтоване програмування. Вона стала стандартом для розробки бізнес-додатків, веб-сервісів, мобільних додатків, ігор та інтеграції з іншими платформами завдяки своїй багатофункціональності та інтеграції з екосистемою .NET.

Крім того, C# пропонує широкий набір можливостей і інструментів, що робить його універсальним і потужним вибором для програмування в різних сферах. Мова підходить для розробки настільних застосунків, веб-додатків, мобільних програм і створення ігор, зокрема за допомогою таких технологій, як Unity[7].

Основні характеристики C#:

- статична типізація: типи даних визначаються під час компіляції, що допомагає виявляти помилки на ранніх етапах розробки та забезпечує надійність коду;
- компіляція в проміжний код: C# компілюється у байт-код (Intermediate Language, IL), який виконується за допомогою середовища Common Language Runtime (CLR). Це забезпечує високу продуктивність та платформну незалежність;
- інтеграція з .NET: мова має доступ до великої стандартної бібліотеки .NET, що спрощує вирішення завдань, пов'язаних з файловими операціями, мережею, базами даних та графікою;
- кросплатформність: завдяки .NET Core та .NET 5/6, C# підтримує створення додатків для Windows, macOS, Linux та мобільних платформ;

– модульність та масштабованість: підтримка сучасних концепцій ООП та багатопотоковості дозволяє створювати додатки будь-якої складності.

Основним конкурентом у виборі мови програмування була мова Python. Python — потужна та проста у вивченні мова програмування, яка поєднує елегантний синтаксис із динамічною типізацією. Завдяки ефективним структурам даних високого рівня та інтерпретованому характеру, Python ідеально підходить для створення сценаріїв і швидкої розробки додатків у різних сферах на багатьох платформах[8]. Вона широко використовується в різних сферах, таких як аналіз даних, машинне навчання, автоматизація процесів, веб-розробка та створення сценаріїв. Завдяки своїй легкості в освоєнні, Python часто стає першим вибором для новачків у програмуванні, а також для швидкої розробки прототипів.

Використання Python у деяких випадках могло б забезпечити більш простий підхід до розробки, ніж C#. Це зумовлено наявністю величезної кількості модулів і бібліотек, які спрощують вирішення специфічних завдань. Наприклад, бібліотеки, такі як NumPy, Pandas та TensorFlow, дозволяють ефективно працювати з обчисленнями, аналізом даних та штучним інтелектом, а Flask і Django спрощують створення веб-додатків. Крім того, Python є кросплатформеною мовою, що дозволяє запускати програми на різних операційних системах без значних змін у коді.

Ще однією перевагою Python є його розвинена спільнота та велика кількість документації. Завдяки активності спільноти Python пропонує безліч готових рішень для широкого спектра задач, що може значно скоротити час на розробку. Простий синтаксис цієї мови полегшує підтримку коду, що є важливим фактором для командної роботи над проєктами.

Однак, попри свої численні переваги, Python має певні недоліки. Через те, що мова є інтерпретованою, програми на Python зазвичай працюють повільніше, ніж програми, написані на компільованих мовах, таких як C#. Це може стати

критичним фактором у проєктах, де потрібна висока продуктивність. Крім того, динамічна типізація Python може призводити до помилок, які виявляються лише під час виконання програми, що ускладнює її відлагодження в складних системах.

Загалом Python є потужним інструментом для багатьох завдань, але його недоліки можуть стати вирішальними у випадках, коли потрібна висока швидкість виконання, статична типізація чи інтеграція з платформою, як-от .NET. Це стало одним із ключових факторів, чому для даного проєкту було обрано саме C#, а не Python.

C# є однією з найпотужніших мов програмування для розробки десктопних додатків. Вона активно використовується в корпоративному середовищі, а також у розробці програмних рішень для Windows. Однією з ключових переваг C# є підтримка широкого спектру інструментів для створення GUI. Бібліотеки, такі як Windows Forms та Windows Presentation Foundation (WPF), дозволяють легко створювати інтуїтивні та функціональні інтерфейси, що спрощує взаємодію користувача з додатком. Мова C# також надає широкі можливості для роботи з мережею та API, що полегшує синхронізацію даних між клієнтським додатком та сервером.

Серед недоліків C# можна виділити відносно складний синтаксис у порівнянні з Python та необхідність написання більшого обсягу коду для реалізації деяких завдань. Також більшість інструментів та бібліотек C# орієнтовані на екосистему Windows, тому розробка на інших платформах (таких як Linux або macOS) може вимагати додаткових налаштувань. Python, у свою чергу, може зіткнутися з обмеженою продуктивністю для складних проєктів та дещо обмеженим вибором інструментів для створення інтерфейсів порівняно з C#. Втім, його простота та швидкий цикл розробки роблять Python чудовим вибором для багатьох завдань, особливо коли необхідно швидко створити та протестувати прототип додатку.

2 АНАЛІЗ РИНКОВИХ РІШЕНЬ ІЗ СУМІЖНОЮ ФУНКЦІОНАЛЬНІСТЮ

Для того щоб визначити основні функціональні вимоги до розроблюваного застосунку «Нотатки», було здійснено аналіз аналогічних програмних продуктів. Нижче наведено приклади популярних застосунків для нотаток із їх описом, перевагами та недоліками.

«Evernote» це популярний багатофункціональний застосунок для створення й організації нотаток, що підходить як для особистого, так і для професійного використання.

«Evernote» – це універсальна програма для створення нотаток, розроблена як для особистого, так і для спільного використання. Він може похвалитися дружнім інтерфейсом із розширеним редагуванням тексту, потужними функціями організації, такими як блокноти та теги, а також бездоганною інтеграцією на кількох пристроях. Він також пропонує веб-вирізки, розпізнавання рукописного тексту та можливості аудіозапису, що робить його комплексним інструментом для захоплення та зберігання різних типів інформації. Хмарне сховище Evernote гарантує, що нотатки будуть доступні з будь-якого місця та створюватимуться безпечні резервні копії[9].

Основний інтерфейс десктопного та мобільного додатку «Evernote» представлено на рис. 1.

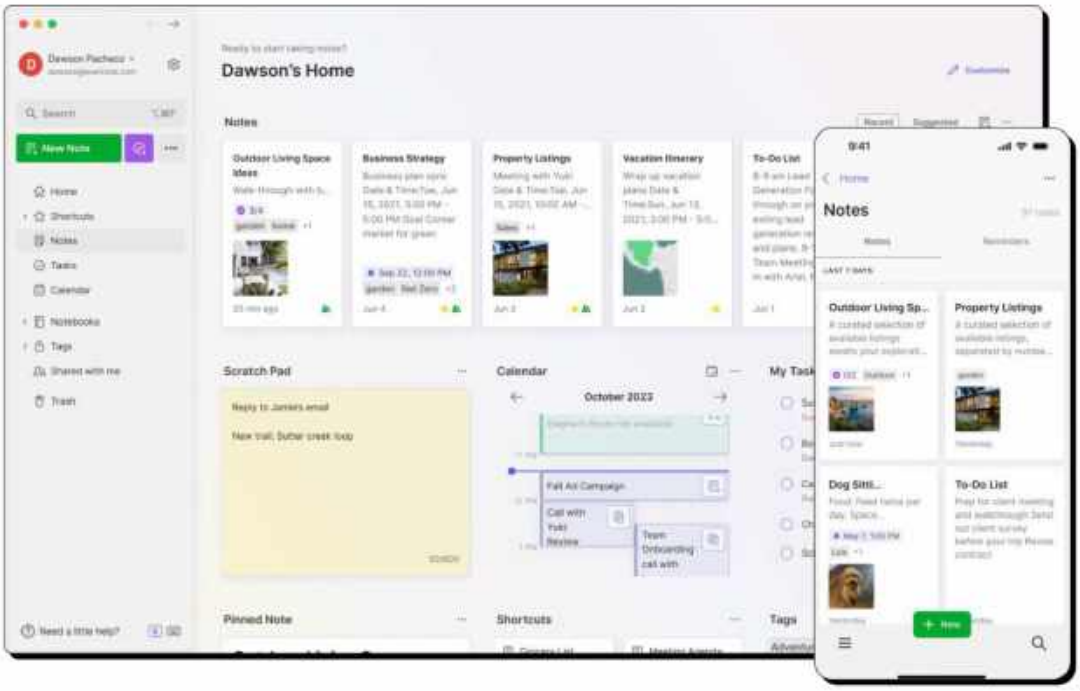

Рисунок 1 – Основний інтерфейс десктопного та мобільного додатку «Evernote»

Основні можливості додатку «Evernote»:

- створення текстових і мультимедійних нотаток;
- організація нотаток за допомогою тегів і категорій;
- підтримка синхронізації даних між пристроями;
- можливість спільного доступу до нотаток із командою;
- інтеграція з іншими сервісами, такими як Google Drive, Slack, Microsoft Teams.

До переваг додатку «Evernote» можна віднести:

- універсальність: підтримка текстових, аудіо- та графічних нотаток, що дозволяє зберігати різноманітну інформацію в одному місці;

- потужна система організації: використання тегів, категорій та блокнотів для структуризації даних, що спрощує пошук та управління інформацією;
- синхронізація між пристроями: автоматичне оновлення даних на всіх пристроях користувача, забезпечуючи доступ до нотаток у будь-який час;
- інтеграція з іншими сервісами: можливість підключення до Google Drive, Slack, Microsoft Teams та інших платформ для розширення функціональності.

До недоліків, в свою чергу можна віднести:

- обмеження безкоштовного плану: ліміт на кількість пристроїв та обсяг завантажень, що може бути незручним для активних користувачів;
- висока вартість преміум-версії: для доступу до повного набору функцій необхідно оформити платну підписку, що може бути дорогим для деяких користувачів;
- залежність від Інтернету: для повноцінної роботи та синхронізації потрібне стабільне підключення до мережі.

Наступний додаток це Google Keep, це мінімалістичний додаток для швидкого створення заміток, інтегрований з екосистемою Google.

Google Keep — це мінімалістичний, але функціональний додаток для створення та організації нотаток, розроблений компанією Google.

Google Keep пропонує простий і інтуїтивно зрозумілий інтерфейс, який підходить як для особистого, так і для командного використання. Основна перевага додатку полягає в його інтеграції з іншими сервісами Google, такими як Google Docs та Google Drive, що дозволяє зручно синхронізувати інформацію між різними платформами.

Цей застосунок дозволяє створювати текстові записи, списки, додавати зображення та навіть використовувати голосові замітки, які автоматично перетворюються в текст. Завдяки функції кольорового кодування користувачі можуть організовувати свої нотатки, роблячи їх візуально впорядкованими. Розглянути приблизний інтерфейс можна на рис. 2.

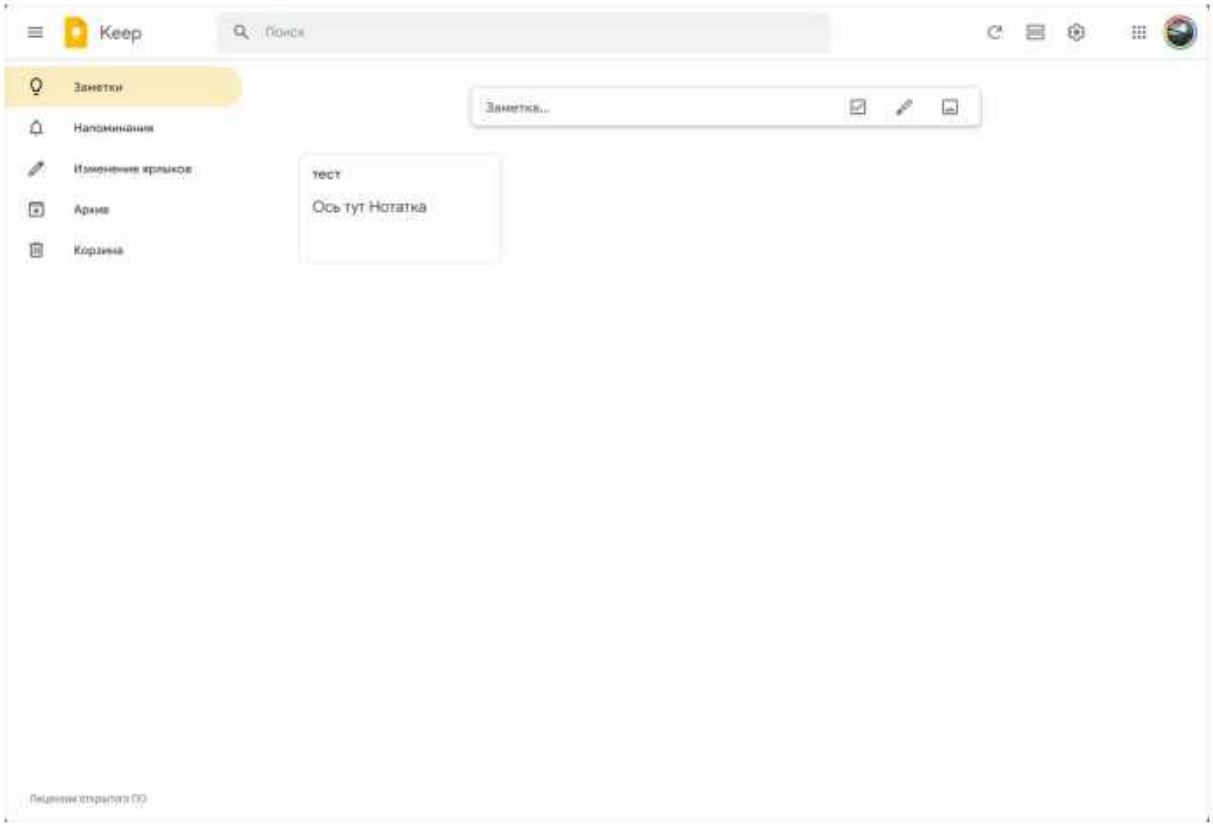

Рисунок 2 – Основний інтерфейс сервісу від Google – Google Keep

Особливість Google Keep — це його хмарна інфраструктура, яка забезпечує миттєвий доступ до нотаток із будь-якого пристрою. Крім того, додаток пропонує функції нагадувань, які активуються на основі часу чи місця, що робить його особливо корисним для планування завдань і організації щоденних справ.

Google Keep також підтримує функцію спільного редагування, що дозволяє працювати над нотатками разом із колегами чи друзями в реальному часі.

Простий дизайн, мінімальна кількість відволікаючих елементів і швидкий доступ до основних функцій роблять цей застосунок ідеальним для тих, хто шукає швидке та зручне рішення для збереження ідей, планів чи важливих заміток.

Інтеграція з екосистемою Google робить Google Keep універсальним інструментом для роботи й особистого використання, гарантуючи швидкий доступ до нотаток у будь-якому місці та в будь-який час.

Основні можливості Google Keep:

- створення різноманітних типів нотаток: користувачі можуть створювати текстові записи, списки, додавати зображення та аудіозаписи, що дозволяє зберігати різноманітну інформацію в одному місці;
- оптичне розпізнавання символів: Google Keep здатний витягувати текст із зображень, що спрощує роботу з друкованими матеріалами та рукописними нотатками;
- транскрипція голосових записів: аудіозаписи автоматично перетворюються в текст, що полегшує їх подальше використання та пошук;
- гнучкий інтерфейс: користувачі можуть обирати між одноколонковим та багатоколонковим відображенням нотаток, залежно від своїх вподобань;
- кольорове кодування та мітки: для кращої організації нотаток можна застосовувати різні кольори та мітки, що полегшує їх категоризацію та пошук;
- закріплення важливих нотаток: можливість закріплювати ключові нотатки у верхній частині списку для швидкого доступу;
- спільна робота в реальному часі: користувачі можуть ділитися нотатками з іншими та спільно редагувати їх, що сприяє ефективній командній роботі;
- інтеграція з Google Docs: нотатки можна легко переносити в Google Docs для подальшого редагування та форматування;

- нагадування за часом та місцем: можливість встановлювати нагадування, прив'язані до конкретного часу або місця, що допомагає не забути важливі завдання;
- синхронізація між пристроями: усі нотатки автоматично синхронізуються між різними пристроями користувача, забезпечуючи доступ до них у будь-який час.

Основні переваги «Google Keep» є:

- простота використання: інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати та редагувати нотатки;
- інтеграція з Google Workspace: синхронізація з іншими сервісами Google, такими як Gmail та Google Drive, що забезпечує зручний обмін даними;
- безкоштовний доступ: усі базові функції доступні безкоштовно без прихованих платежів;
- підтримка голосових заміток: можливість створення нотаток за допомогою голосових команд, що підвищує зручність використання.

До недоліків можна віднести:

- обмежений функціонал: відсутність розширених можливостей для професійного використання, таких як підтримка вкладених папок або розширених форматів;
- залежність від Інтернету: для синхронізації та доступу до даних потрібне підключення до мережі;
- обмежені можливості організації: відсутність підтримки складних структур для організації нотаток, що може бути незручним для користувачів з великою кількістю записів.

Наступним варто розглянути ще один додаток – «Simplenote». Проста програма для записів без зайвих функцій, призначена для обробки звичайних текстових нотаток. Ідеально підійде для швидкої фіксації думок "на ходу". У

Simplenote зручно редагувати тексти, використовуючи розмітку Markdown із публікацією нотаток в інтернеті для загального огляду. Ви можете використовувати теги для розподілу нотаток, щоб за необхідності легко знайти їх у потрібну хвилину. Всі зміни, що вносяться, автоматично фіксуються в історії правок, так що завжди можна повернутися до попередньої версії нотатки для її редагування.

Simplenote – це простий і швидкий додаток для створення текстових нотаток, який підходить для користувачів, що цінують мінімалізм та ефективність. Розроблений компанією Automattic, Simplenote орієнтований на зручність і кросплатформеність, забезпечуючи доступ до нотаток із різних пристроїв, таких як смартфони, планшети, ПК та веб-браузери. Інтерфейс можна розглянути на рис. 3.

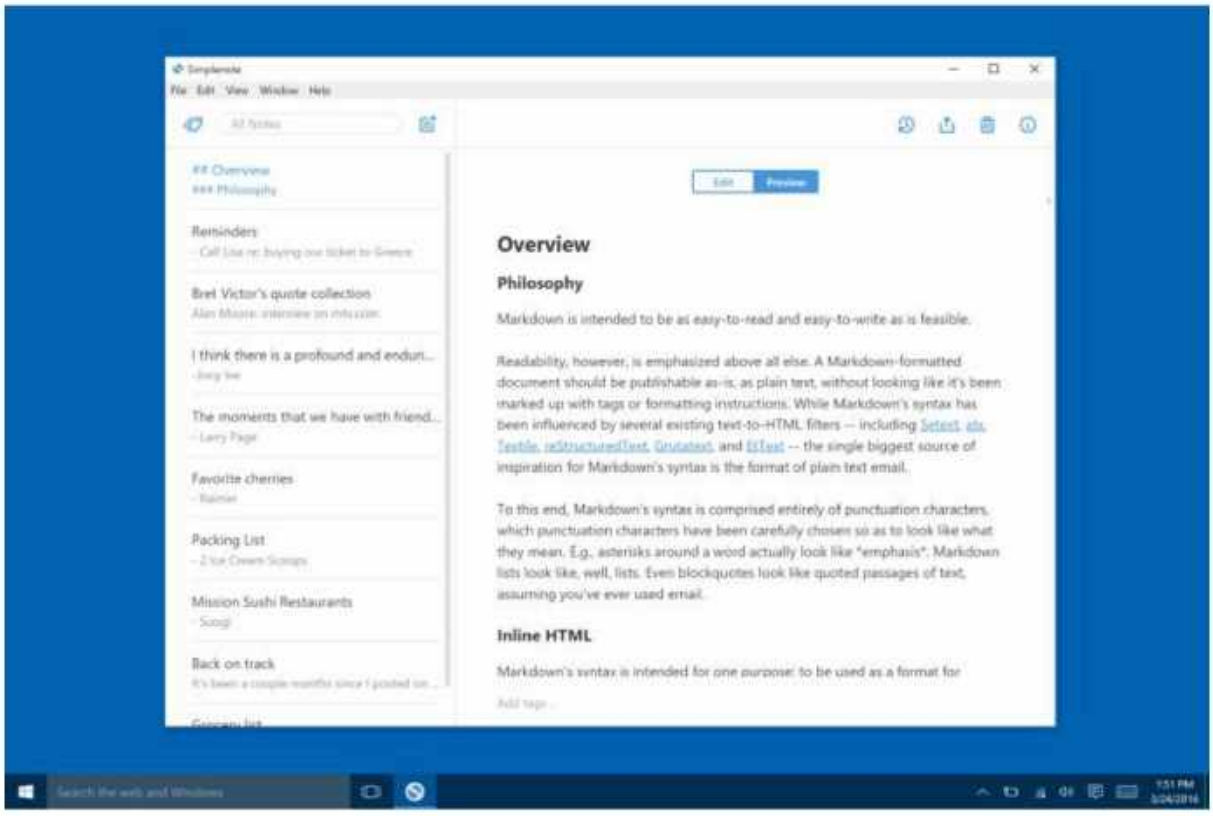

Рисунок 3 – Основний інтерфейс додатку «Simplenote»

Основні можливості Simplenote:

- створення текстових нотаток без зайвих функцій і відволікаючих елементів;
- автоматична синхронізація між пристроями, що забезпечує актуальність даних;
- підтримка тегів для організації нотаток, що дозволяє швидко знаходити потрібну інформацію;
- функція пошуку, яка миттєво знаходить записи за ключовими словами або тегами;
- можливість перегляду історії версій нотаток для відновлення попередніх змін;
- підтримка Markdown для форматування тексту, що спрощує створення структурованих записів;
- можливість спільного редагування нотаток із колегами або друзями.

До основних переваг «Simplenote» можна віднести :

- мінімалістичний інтерфейс, який легко освоїти навіть новачкам;
- кросплатформена доступність – працює на iOS, Android, Windows, macOS, Linux та через веб;
- усі функції надаються безкоштовно, без підписок чи обмежень;
- швидка синхронізація, яка дозволяє працювати над нотатками у реальному часі.

До недоліків можна віднести:

- обмежений функціонал – відсутня підтримка мультимедіа, наприклад, зображень або аудіо;
- відсутність папок для структуризації записів, організація можлива лише через теги;

- прості можливості форматування – навіть із підтримкою Markdown відсутні інструменти для роботи зі складними форматами, наприклад таблицями;
- нотатки зберігаються у незашифрованому вигляді на серверах, що може викликати занепокоєння щодо конфіденційності.

Simplenote — це оптимальне рішення для користувачів, які шукають простий і швидкий спосіб створення текстових записів. Однак для тих, хто потребує більш розширеного функціоналу або підвищеного рівня безпеки, можливо, варто розглянути інші додатки.

3 ПРОЄКТУВАННЯ ДОДАТКУ

3.1 Проєктування за допомогою методології SADT

Використання мови IDEF0 для проєктування

При проєктуванні додатку була обрана методологія функціонального моделювання SADT (Structured Analysis and Design Technique) – стандарт IDEF0.

Методологія функціонального моделювання SADT (Structured Analysis and Design Technique) — це структурований підхід до аналізу і проектування систем, який використовується для опису функцій, процесів і взаємодій між компонентами системи. В основі SADT лежить стандарт IDEF0, який забезпечує формалізовану нотацію для створення діаграм, що візуалізують систему на різних рівнях деталізації.

Методологія SADT базується на використанні діаграм, які складаються з блоків (функцій) і стрілок, що з'єднують ці блоки. Кожна діаграма може містити чотири типи стрілок:

- діяльність – це будь-яка операція, процес або функція, що перетворює вхідні дані у вихідні. Діяльність зображається у вигляді прямокутника, всередині якого зазначається її назва;
- вхідні (Input) дані: дані або інформація, необхідні для початку виконання функції. Це ресурси, які потрапляють у систему або підсистему, щоб бути обробленими;
- вихід (Output): дані, інформація або продукти, що є результатом виконання функції;
- контроль (Control): обмеження, правила або стандарти, які впливають на поведінку функції і визначають, як вона має бути виконана;
- механізм (Mechanism): засоби, ресурси або виконавці (люди, обладнання), необхідні для виконання функції.

Першим етапом створення моделі за методологією SADT є контекстна діаграма. Вона є високорівневим представленням системи, яке показує взаємодію системи із зовнішнім середовищем. Контекстна діаграма надає загальний огляд системи, демонструє її межі та ключові зв'язки з іншими процесами чи системами.

Контекстна діаграма:

- фокусує увагу на взаємодії із зовнішніми факторами;
- не розкриває деталей внутрішньої структури системи;
- слугує основою для подальшого детального моделювання.

Контекстна діаграма є відправною точкою, яка дозволяє визначити основні компоненти системи, окреслити їхні ролі і взаємозв'язки. Вона дозволяє розробнику системи сфокусуватися на ключових аспектах, перш ніж заглиблюватися у специфіку окремих функцій чи підсистем.

Переваги використання методології SADT у розробці додатка:

- **Структурованість:** діаграми IDEF0 забезпечують чітку ієрархію функцій, що дозволяє систематично підходити до аналізу та проектування.
- зрозуміле представлення: візуальні елементи дозволяють легко інтерпретувати складні взаємодії між компонентами;
- аналіз функцій: методологія дозволяє аналізувати і оптимізувати процеси, виявляти слабкі місця і визначати залежності між функціями;
- гнучкість: SADT можна використовувати як для проектування нових систем, так і для аналізу існуючих.

У межах розробки додатку за методологією SADT буде створено кілька рівнів діаграм. На першому рівні контекстна діаграма описуватиме загальну структуру додатку, його основні функції, зв'язки із зовнішніми системами (серверна база даних, інтерфейси користувача) та межі системи.

Контекстна діаграма системи, що розробляється зображена на рис. 4.

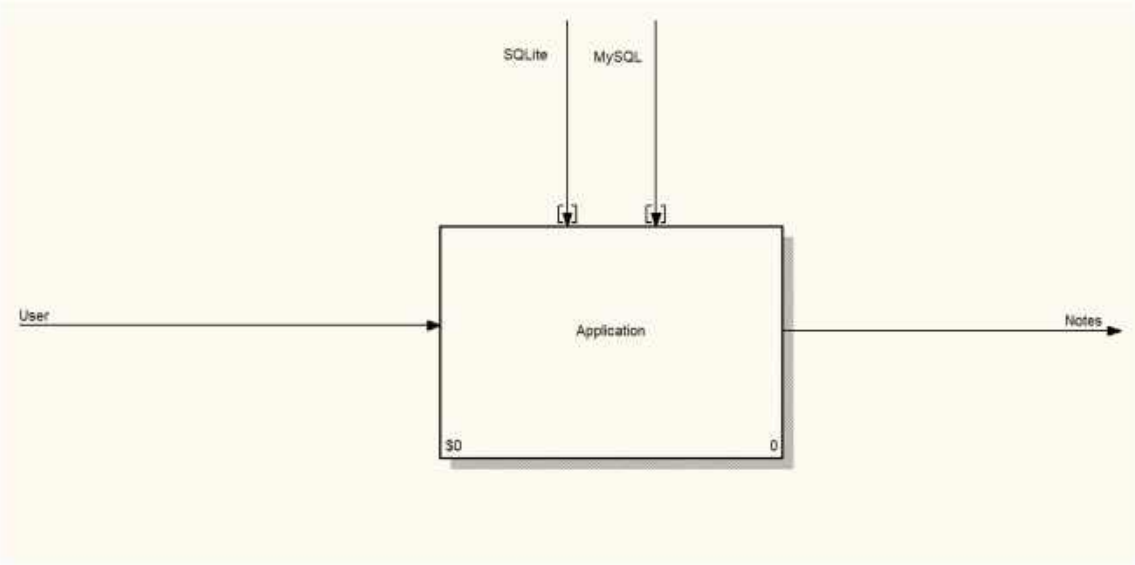

Рисунок 4 – Контекстна діаграма додатку «Нотатки»

На контекстній діаграмі відображено головний процес. На вході користувач, котрий спершу проходить авторизацію, а після вже працює з додатками.

Після того, як процес ідентифіковано як цілісну сутність, його подрібнюють на дрібніші складові (здійснюється декомпозиція). Декомпозиція – це процедура поділу системи на менші елементи та підзадачі з метою глибшого аналізу та кращого розуміння її структури і функцій. Такий підхід дає змогу конкретизувати моделі та створити більш деталізовані функціональні структури для кожної частини програми. Декомпозиція спрощує системний аналіз, проєктування та документування, розчленовуючи складні системи на більш керовані й прозорі елементи. Кожен щабель декомпозиції гарантує глибшу деталізацію, що дає змогу точніше визначити функції, процеси і взаємозв'язки в системі. На різних рівнях декомпозиції синтаксис, уживаний у функціональній

моделі, може залишатися ідентичним, але здатен модифікуватися для відображення докладнішої інформації про складові системи.

Діаграма декомпозиції розробки додатку «Нотатки» (див.рис. 5).

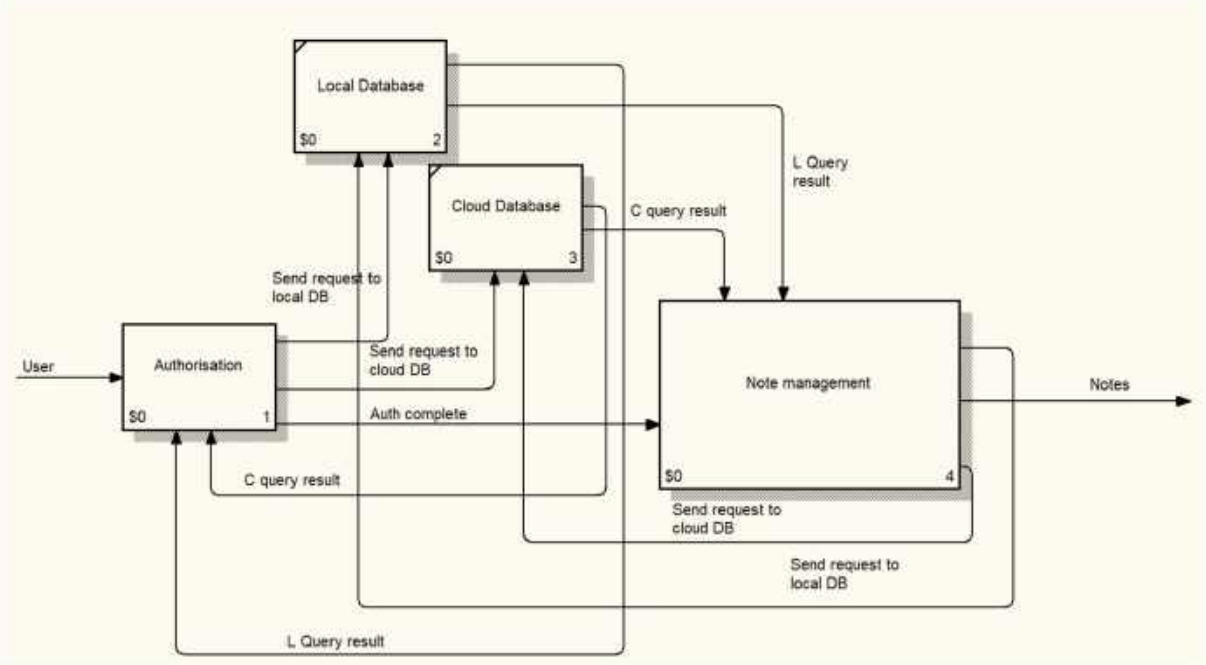

Рисунок 5 – Діаграма декомпозиції розробки додатку «Нотатки»

Діаграма декомпозиції блоку «Авторизації» наведена на рис.6.

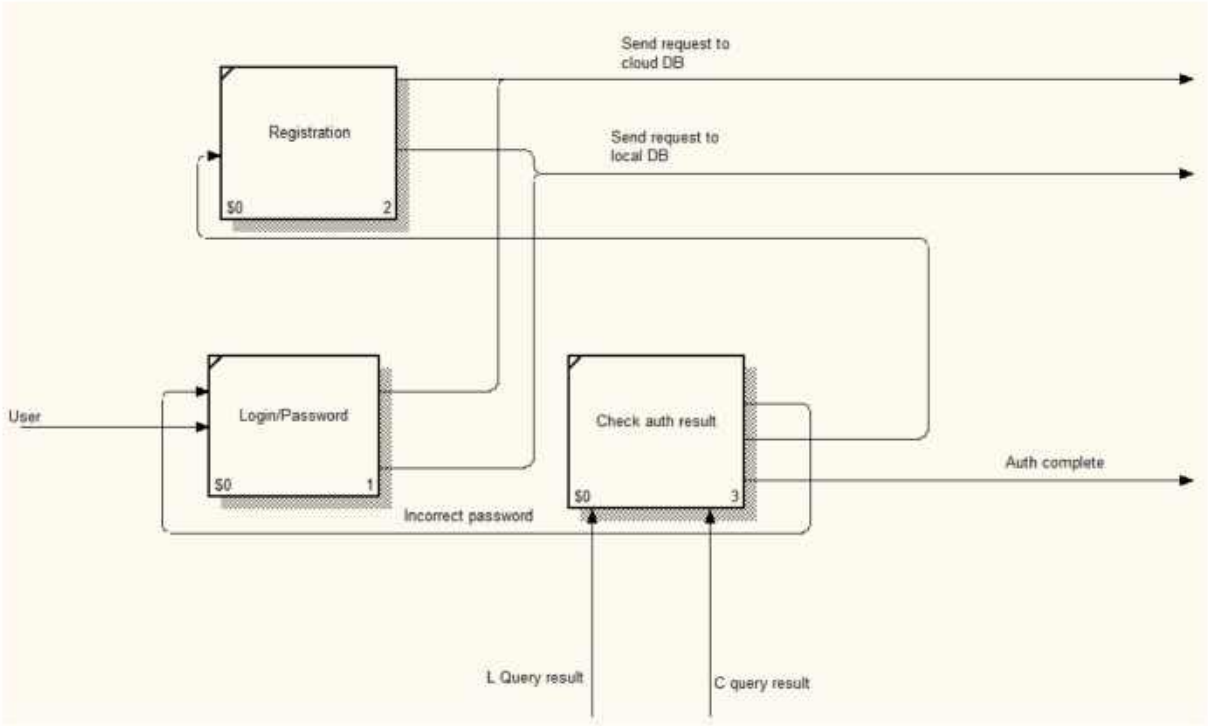

Рисунок 6 – Діаграма декомпозиції блоку «Авторизації»

Діаграма декомпозиції блоку «Менеджмент додаток» наведена на рис. 7.

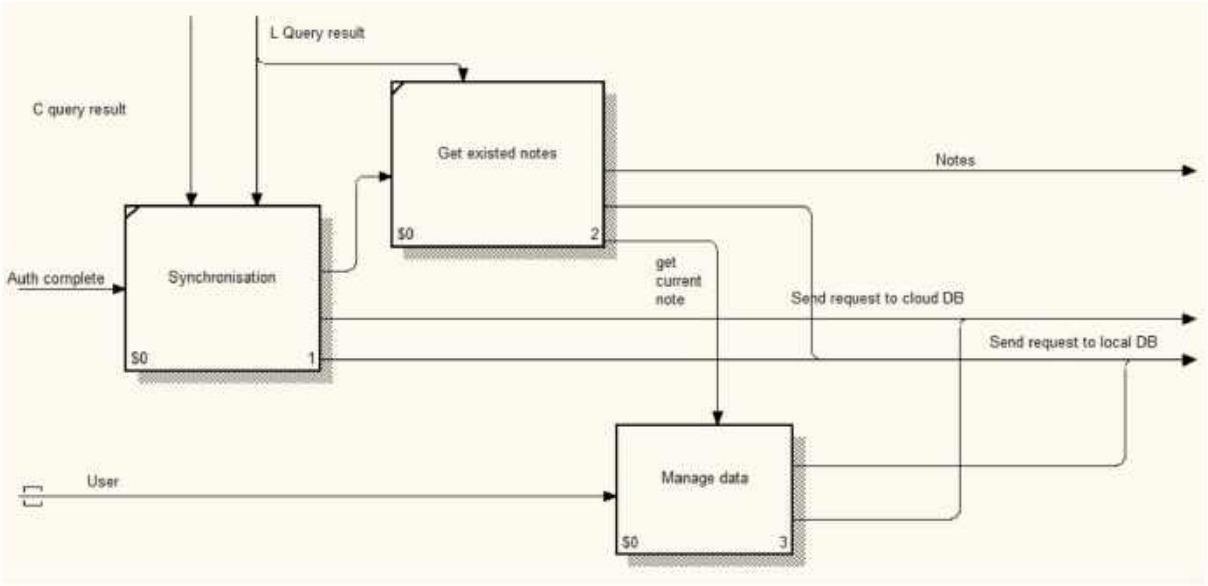

Рисунок 7 – Діаграма декомпозиції блоку «Менеджмент додаток»

Діаграма декомпозиції блоку «Менеджмент даних» наведена на рис. 8.

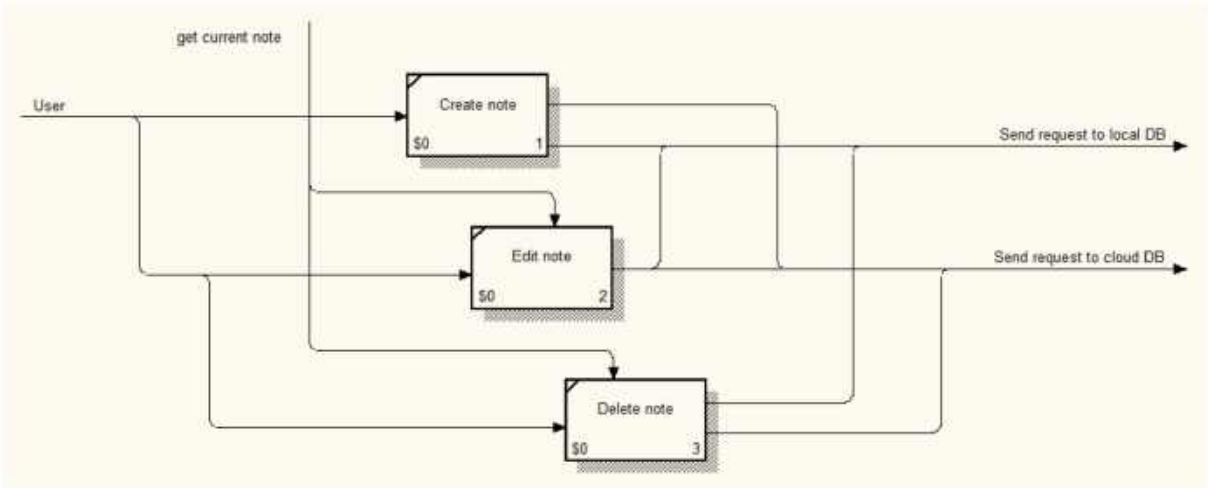

Рисунок 8 – Діаграма декомпозиції блоку «Менеджмент даних»

3.2 Проєктування функціональних вимог локальної БД додатку

Для представлення логічної структури додатку «Нотатки» була обрана модель «Сутність-зв'язок».

Сутність-зв'язок (entity-relationship) – це методологія моделювання даних, яка широко використовується у сфері проєктування баз даних та інформаційних систем. Вона дозволяє візуалізувати структуру даних, представити взаємозв'язки між об'єктами та упорядкувати інформацію, що зберігається в базі даних. Цей підхід надає можливість розробникам та аналітикам побудувати ефективну та зрозумілу модель даних, яка відображає реальні взаємозв'язки між об'єктами.

Основна суть цієї методології полягає у відображенні реальних об'єктів, понять чи явищ у вигляді сутностей, а їх взаємозв'язків – через зв'язки між цими сутностями. Сутності є центральними об'єктами моделювання, які можна ідентифікувати, описати певними властивостями та зберегти в базі даних. Водночас зв'язки показують, як ці об'єкти взаємодіють між собою, створюючи логічну структуру для збереження і роботи з даними.

Для побудови моделі сутність-зв'язок використовуються кілька ключових елементів, кожен з яких має своє значення та роль:

У моделі сутність-зв'язок використовуються такі основні елементи:

- сутність (entity) – модель об'єкта, поняття чи явища, яке має набір унікальних характеристик, що описують його властивості; зазвичай зображується у вигляді прямокутника з назвою, яка вказує на сутність;
- атрибут (attribute) – властивість сутності, яка характеризує її та надає додаткову інформацію; кожна сутність може мати один або кілька атрибутів, які допомагають детальніше описати об'єкт;
- зв'язок (relationship) – модель відношення або взаємодії між сутностями; зв'язки можуть бути однонаправленими або двонаправленими, а також

мати різні типи, зокрема один-до-одного, один-до-багатьох чи багато-до-багатьох;

- кардинальність (cardinality) – характеристика зв'язку, яка визначає кількість сутностей, що можуть бути пов'язані одна з одною; наприклад, кардинальність один-до-багатьох вказує, що одна сутність може бути пов'язана з кількома іншими, а багато-до-багатьох означає багатосторонній зв'язок.

Модель сутність-зв'язок дозволяє оптимізувати процес створення баз даних, забезпечуючи структуроване бачення організації даних. Вона сприяє полегшенню спільної роботи між різними фахівцями, такими як аналітики та технічні спеціалісти, і допомагає знизити ризик помилок під час впровадження баз даних.

Для локальної бази даних додатку «Нотатки» виділимо наступні сутності:

- користувачі (users) – описує облікові записи користувачів, які мають унікальне ім'я, хеш паролю та додаткову інформацію для захисту (сіль);
- нотатки (notes) – представляє текстові записи, створені користувачами, які мають заголовок, вміст, дати створення та оновлення, а також статус синхронізації з віддаленою базою даних;
- зображення нотаток (note_images) – зберігає дані зображень, прикріплених до нотаток, включаючи їх порядок у межах кожної нотатки;
- черга синхронізації (sync_queue) – фіксує зміни, які потрібно синхронізувати між локальною та віддаленою базою даних, із зазначенням типу операції, наприклад створення, оновлення чи видалення.

Кожна сутність повинна мати атрибути або їхні комбінації, які дозволяють однозначно ідентифікувати кожен екземпляр сутності. Такі атрибути

називаються первинними ключами. Під час проєктування бази даних були визначені сутності та їхні атрибути.

Сутність «Користувачі (users)» представляє облікові записи користувачів додатку. Вона містить наступні атрибути (табл. 1):

- id – унікальний ідентифікатор користувача, первинний ключ (integer, primary key, autoincrement);
- username – ім'я користувача, яке повинно бути унікальним (text, not null, unique);
- password_hash – хеш паролю користувача для забезпечення безпеки (text, not null);
- salt – сіль для хешування паролю, використовується для посилення захисту (text, not null).

Таблиця 1 – «Користувачі (users)»

№	Атрибут	Тип
1	2	3
1	id	integer
2	username	text
3	password_hash	text
4	salt	text

Сутність «Нотатки (notes)» зберігає текстові записи, створені користувачами. Вона містить наступні атрибути (табл. 2):

- id – унікальний ідентифікатор нотатки, первинний ключ (integer, primary key, autoincrement);
- user_id – ідентифікатор користувача, якому належить нотатка, зовнішній ключ до таблиці users (integer, not null);
- remote_id – ідентифікатор нотатки в віддаленій базі даних, якщо вона синхронізована (integer, null);
- title – заголовок нотатки (text, not null);

– content – текстовий вміст нотатки (text, null);
– created_at – дата і час створення нотатки (datetime, default current_timestamp);
– updated_at – дата і час останнього оновлення нотатки (datetime, null);
– sync_status – статус синхронізації.

Таблиця 2 – Сутність «Нотатки (notes)»

№	Атрибут	Тип
1	2	3
1	id	integer
2	user_id	integer
3	remote_id	integer
4	title	text
5	content	text
6	created_at	datetime
7	updated_at	datetime
8	sync_status	integer

Сутність «Зображення нотаток(note_images)» містить прикріплені до нотаток зображення. Має наступні атрибути (табл. 3):

– id – унікальний ідентифікатор зображення, первинний ключ (integer, primary key, autoincrement);
– note_id – ідентифікатор нотатки, до якої прикріплено зображення, зовнішній ключ до таблиці notes (integer, not null);
– image_data – дані зображення у бінарному форматі (blob, null);
– position – позиція зображення у межах нотатки, використовується для впорядкування (integer, null).

Таблиця 3 – Сутність «Зображення нотаток(note_images)»

№	Атрибут	Тип
1	2	3
1	id	integer
2	note_id	integer
3	image_data	blob
4	position	integer

Сутність «Черга синхронізації (sync_queue)» зберігає дані про зміни, які очікують синхронізації між локальною та віддаленою базами даних. Сутність містить наступні атрибути (табл. 4):

- id – унікальний ідентифікатор запису в черзі, первинний ключ (integer, primary key, autoincrement);
- note_id – ідентифікатор нотатки, що потребує синхронізації (integer, null);
- sync_type – тип операції, яка виконується:
- "create" – створення нової нотатки;
- "update" – оновлення існуючої нотатки;
- "delete" – видалення нотатки (text, not null);
- is_image – вказує, чи стосується запис зображення;
- image_index – індекс зображення, якщо запис стосується зображення (integer, null).

Таблиця 4 – Сутність «Зображення нотаток(note_images)»

№	Атрибут	Тип
1	2	3
1	id	integer
2	note_id	integer
3	sync_type	text
4	is_image	boolean
5	image_index	integer

Після визначення основних сутностей та їхніх атрибутів для локальної бази даних, можна встановити зв'язки між цими сутностями. Локальна база даних представлена у вигляді моделі «сутність-зв'язок», де зв'язки між сутностями мають тип «один-до-багатьох».

Діаграма «сутність-зв'язок» для локальної бази даних представлена на рис.9.

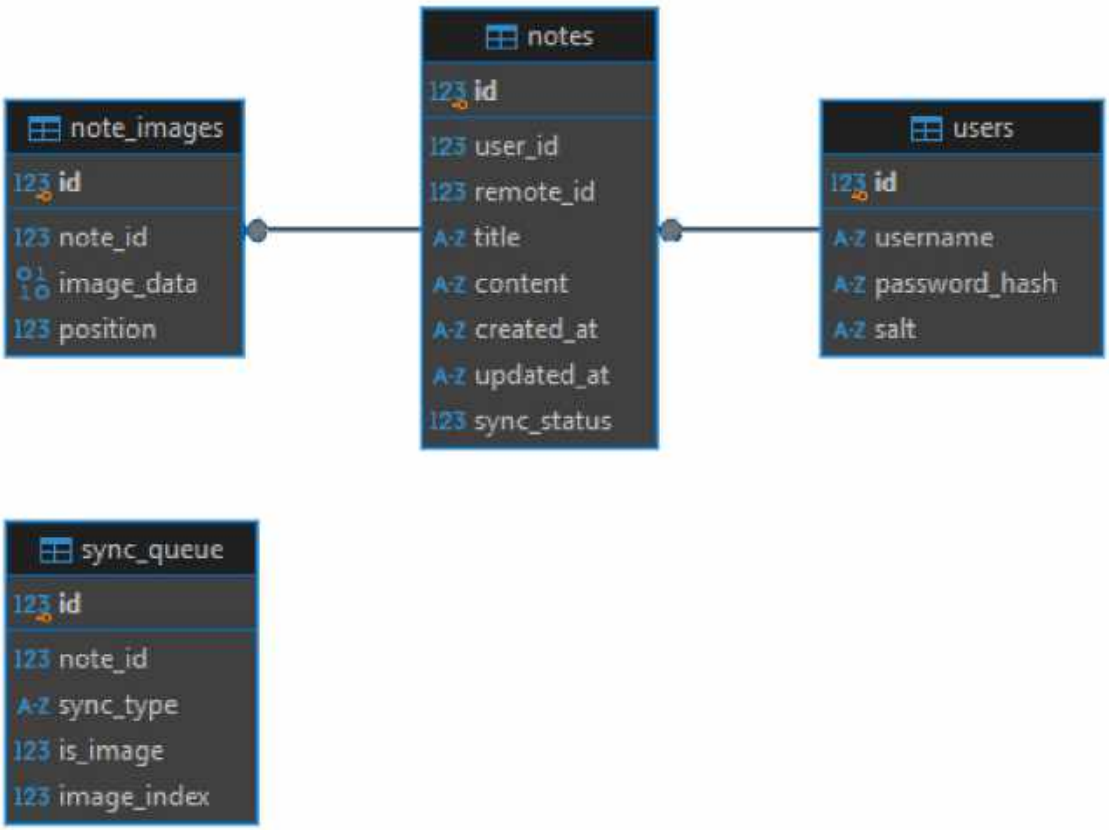

Рисунок 9 – Діаграмма «сутність зв'язок»

На основі проведеного аналізу та розробки діаграми «сутність-зв'язок» для локальної бази даних можна зробити такі висновки. База даних є добре структурованою системою, що охоплює основні сутності, їхні атрибути та зв'язки між ними. Діаграма «сутність-зв'язок» слугує основою для створення та подальшого розвитку бази даних, яка відповідає вимогам локальної системи збереження даних.

3.3 Проєктування функціональних вимог хмарної БД додатку

Для хмарної бази даних додатку «Нотатки» ми також використовуємо підхід моделювання даних на основі моделі «сутність-зв'язок». Цей підхід дозволяє описувати об'єкти та їх взаємозв'язки у структурованій та зрозумілій формі, що є основою для розробки ефективної бази даних. У хмарній базі даних модель «сутність-зв'язок» забезпечує чітке представлення даних, необхідних для зберігання та синхронізації між клієнтськими пристроями та сервером.

Основними сутностями хмарної бази даних є об'єкти, які відображають ключові елементи роботи додатку. Взаємозв'язки між цими сутностями встановлюються для забезпечення цілісності даних, а також для реалізації функцій синхронізації, збереження даних і керування користувачами. Кожна сутність містить набір атрибутів, які описують її властивості та виконують роль ідентифікаторів. Це забезпечує унікальність кожного екземпляра сутності та дозволяє ефективно обробляти запити до бази даних.

Використання моделі «сутність-зв'язок» у хмарному середовищі дозволяє створити масштабовану, надійну та продуктивну систему зберігання даних. У даному випадку було визначено такі основні сутності:

- користувачі (users) – описує облікові записи користувачів, які мають унікальне ім'я та хеш паролю для забезпечення автентифікації користувача. Кожен користувач має унікальний ідентифікатор (id), ім'я користувача (username) та хешований пароль (password_hash). Ця сутність є основою для управління доступом до бази даних;

- нотатки (notes) – представляє текстові записи, створені користувачами. Кожна нотатка має унікальний ідентифікатор (id), посилання на користувача (user_id), якому вона належить, заголовок (title), текстовий вміст (content), дату створення (created_at) та дату останнього оновлення

(updated_at). Ця сутність дозволяє зберігати та управляти текстовою інформацією у хмарному середовищі;

– зображення нотаток (note_images) – зберігає дані зображень, прикріплених до нотаток. Включає унікальний ідентифікатор (id), зв'язок із нотаткою (note_id), до якої належить зображення, дані зображення (image_data) та позицію зображення в межах нотатки (position). Ця сутність додає можливість прикріплювати візуальний контент до текстових записів.

Під час проєктування хмарної бази даних були визначені ці сутності та їхні атрибути, щоб забезпечити надійність зберігання, синхронізацію даних та їх доступність для користувачів із будь-якого пристрою.

Сутність «Користувачі (users)» представляє облікові записи користувачів додатку. Вона містить наступні атрибути (табл. 5):

– id – унікальний ідентифікатор користувача, первинний ключ (integer, primary key, autoincrement);
– username – ім'я користувача, яке повинно бути унікальним (varchar(50), not null, unique);
– password_hash – хеш паролю користувача для забезпечення безпеки (varchar(255), not null).

Таблиця 5 – «Користувачі (users)»

№	Атрибут	Тип
1	2	3
1	id	integer
2	username	varchar(50)
3	password_hash	varchar(255)

Сутність «Нотатки (notes)» представляє текстові записи, створені користувачами додатку. Вона містить наступні атрибути (табл. 6):

- id – унікальний ідентифікатор нотатки, первинний ключ (integer, primary key, autoincrement);
- user_id – посилання на користувача, якому належить нотатка (integer, not null);
- title – заголовок нотатки (varchar(255), not null);
- content – текстовий вміст нотатки (text, nullable);
- created_at – дата і час створення нотатки (datetime, not null, default current_timestamp);
- updated_at – дата і час останнього оновлення нотатки (datetime, nullable).

Таблиця 6 – «Нотатки (notes)»

№	Атрибут	Тип
1	2	3
1	id	integer
2	user_id	integer
3	title	varchar(255)
4	content	text
5	created_at	datetime
6	updated_at	datetime

Сутність «Зображення нотаток (note_images)» зберігає дані зображень, прикріплених до нотаток. Вона містить наступні атрибути (табл. 7):

- id – унікальний ідентифікатор зображення, первинний ключ (integer, primary key, autoincrement);
- note_id – посилання на нотатку, до якої прив'язане зображення (integer, not null);
- image_data – дані зображення у бінарному форматі (longblob, nullable);
- position – позиція зображення в межах нотатки для впорядкування (integer, nullable).

Таблиця 7 «Зображення нотаток (note_images)»

№	Атрибут	Тип
1	2	3
1	id	integer
2	note_id	integer
3	image_data	longblob
4	position	integer

Після визначення основних сутностей та їхніх атрибутів для хмарної бази даних, можна встановити зв'язки між цими сутностями. Хмарна база даних представлена у вигляді моделі «сутність-зв'язок», де зв'язки між сутностями мають тип «один-до-багатьох».

Діаграма «сутність-зв'язок» для хмарної бази даних представлена на рис. 10.

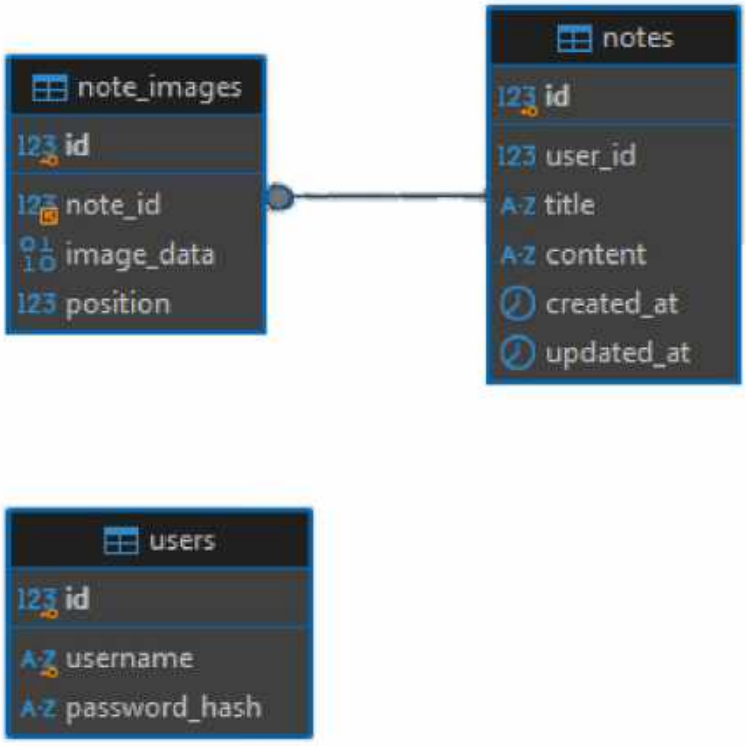

Рисунок 10 – Діаграмма «сутність зв'язок»

На основі проведеного аналізу та розробки діаграми «сутність-зв'язок» для хмарної бази даних додатку «Нотатки» можна зробити такі висновки. База даних являє собою добре структуровану систему, яка охоплює основні сутності, їхні атрибути та зв'язки між ними. Розроблена модель бази даних враховує ключові елементи додатку «Нотатки», такі як користувачі, текстові записи та прикріплені зображення. Зв'язки між сутностями дозволяють ефективно організувати зберігання та доступ до даних, забезпечуючи їхню цілісність та узгодженість. Діаграма «сутність-зв'язок» слугує основою для створення та подальшого розвитку бази даних, яка відповідає потребам хмарного середовища та підтримує синхронізацію між клієнтськими пристроями.

4 РОЗРОБКА І ТЕСТУВАННЯ ФУНКЦІОНАЛУ ДОДАТКУ

Додаток «Нотатки» створений для управління текстовими записами користувачів у хмарному середовищі. Він забезпечує створення, редагування, збереження та синхронізацію даних між клієнтськими пристроями та хмарною базою даних. Основна мета додатку — надати користувачам зручний інструмент для організації текстових даних, який є доступним у будь-який час і з будь-якого пристрою.

Розгляд проєкту слід почати з аналізу вікна авторизації, адже це перший елемент інтерфейсу, який бачить користувач під час запуску додатку. Авторизація є важливим етапом роботи програми, оскільки саме вона забезпечує доступ до персональних даних і створює враження про зручність та функціональність додатку. Візуальне оформлення цього вікна представлено на рис. 11.

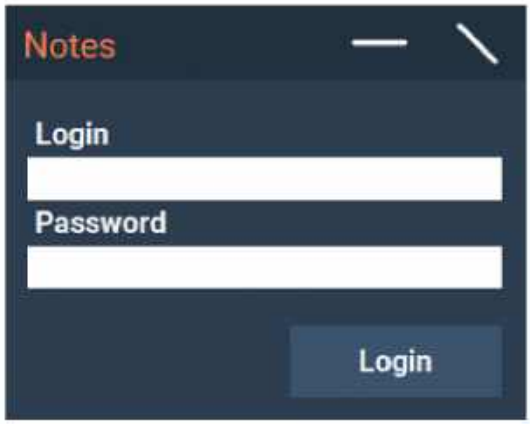

Рисунок 11 – Вигляд вікна авторизації

Додаток надає користувачеві простий та інтуїтивно зрозумілий інтерфейс. Вікно авторизації складається з двох полів для введення даних — логіну та паролю, а також кнопки для підтвердження авторизації. Дизайн цього вікна

створений таким чином, щоб мінімізувати час, необхідний для входу в систему, та зменшити ймовірність помилок під час введення даних.

Після натискання кнопки «Авторизуватися» програма виконує перевірку наявності облікового запису користувача у локальній та хмарній базах даних. Ця перевірка є ключовою, адже вона визначає подальший сценарій роботи програми:

- користувач знайдений у локальній та хмарній базах даних;
- користувач знайдений лише у хмарній базі даних;
- користувач не знайдений ні в локальній, ні в хмарній базах даних;
- хмарна база даних недоступна.

Реалізація у коді виглядає наступним чином:

```
private async void btnLogin_Click(object sender, EventArgs e)
 {
 string username = txtUsername.Text;
 string password = txtPassword.Text;

 var localLoginTask = Task.Run(() =>
UserSession.LoginLocal(username, password));
 var remoteLoginTask = Task.Run(() => UserSession.Login(username,
password));
 int[] loginResults = await Task.WhenAll(localLoginTask,
remoteLoginTask);

 int localLoginStatus = loginResults[0];
 int remoteLoginStatus = loginResults[1];

 if (remoteLoginStatus == 1 && localLoginStatus == 1)
 {
 // Successful login to both databases
 UserSession.loginStatus = 1;
 this.Close();
 }
 else if (remoteLoginStatus == 1 && localLoginStatus == 0)
 {
 // Successful login to the remote DB, but the local user is not
found
 if (UserSession.RegisterLocal(username, password))
 {
 // After registering locally, try to log in locally
 if (UserSession.LoginLocal(username, password) == 1)
 {
 UserSession.loginStatus = 1;
```

```
 MessageBox.Show("Local registration was successful!", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
 this.Close();
 }
 else
 {
 MessageBox.Show("Local login failed after registration.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
 UserSession.loginStatus = -1;
 }
 }
 }
 else if (remoteLoginStatus == -1 && localLoginStatus == 1)
 {
 // No connection to the remote DB, but successful local login -
working offline
 UserSession.loginStatus = 0;
 MessageBox.Show("No connection to the remote server. Working
offline.", "Could be better", MessageBoxButtons.OK,
MessageBoxIcon.Information);
 this.Close();
 }
 else if (remoteLoginStatus == 0 && localLoginStatus == 1)
 {
 // User found locally but not in the remote DB
 DialogResult result = MessageBox.Show("You are registered locally
but not in the cloud. Do you want to register in the cloud?",
"Registration", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
 if (result == DialogResult.Yes)
 {
 if (UserSession.Register(username, password))
 {
 UserSession.loginStatus = 1;
 MessageBox.Show("Cloud registration was successful!", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);
 this.Close();
 }
 else
 {
 MessageBox.Show("Cloud registration failed.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
 }
 }
 else
 {
 this.Close();
 }
 }
 else if (remoteLoginStatus == 0 && localLoginStatus == 0)
```

```
 {
 // Both statuses indicate the user is not found — propose
registration
 DialogResult result = MessageBox.Show("User not found. Do you want
to register?", "Registration", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
 if (result == DialogResult.Yes)
 {
 var localRegisterTask = Task.Run(() =>
UserSession.RegisterLocal(username, password));
 var remoteRegisterTask = Task.Run(() =>
UserSession.Register(username, password));
 bool[] registerResults = await Task.WhenAll(localRegisterTask,
remoteRegisterTask);

 if (registerResults[0] || registerResults[1])
 {
 MessageBox.Show("Registration successful!", "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);

 // Perform automatic login after registration
 localLoginTask = Task.Run(() => UserSession.LoginLocal(username,
password));
 remoteLoginTask = Task.Run(() => UserSession.Login(username,
password));
 loginResults = await Task.WhenAll(localLoginTask,
remoteLoginTask);

 localLoginStatus = loginResults[0];
 remoteLoginStatus = loginResults[1];

 if (remoteLoginStatus == 1 && localLoginStatus == 1)
 {
 // Successful login to both databases after registration
 UserSession.loginStatus = 1;
 this.Close();
 }
 else if (remoteLoginStatus == -1 && localLoginStatus == 1)
 {
 UserSession.loginStatus = 0;
 MessageBox.Show("Working offline.", "Information",
MessageBoxButtons.OK, MessageBoxIcon.Information);
 this.Close();
 }
 else
 {
 MessageBox.Show("Login failed after registration.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
 }
```

```
 }
 else
 {
 MessageBox.Show("Registration failed.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
 }
 }
 }
 }
```

Коли програма виявляє, що обліковий запис користувача існує як у локальній, так і в хмарній базах даних, користувач отримує повідомлення про успішну авторизацію. Після цього йому відкривається доступ до всіх його даних, які синхронізовані між двома базами. Цей сценарій дозволяє користувачу негайно почати роботу в додатку без необхідності завантаження чи створення додаткових записів.

Якщо програма виявляє, що обліковий запис користувача існує в базі даних, але введений пароль не збігається, користувач отримає повідомлення про помилку авторизації. У повідомленні зазначається, що пароль введено неправильно, і пропонується спробувати знову, приклад повідомлення можна фіксувати на рис.12. Для підвищення безпеки система може передбачати обмежену кількість спроб введення паролю, після чого користувача може бути тимчасово заблоковано. Цей підхід допомагає запобігти несанкціонованому доступу до облікового запису.

```
// Method for logging in
 public static int Login(string username, string password)
 {
 try
 {
 using (var conn = new MySqlConnection(connectionString))
 {
 conn.Open();

 // Check if the user exists
 var cmd = new MySqlCommand("SELECT id, password_hash FROM users
WHERE username = @username", conn);
 cmd.Parameters.AddWithValue("@username", username);

 using (var reader = cmd.ExecuteReader())
 {
```

```
 if (reader.Read())
 {
 // User found, verify the password
 int userId = reader.GetInt32("id");
 string storedPasswordHash = reader.GetString("password_hash");
 string get_Hash = HashPassword(password);
 if (storedPasswordHash == get_Hash)
 {
 SetUser(userId, username, get_Hash); // Set user data
 return 1; // Successful login
 }
 else
 {
 // Password does not match
 MessageBox.Show("Incorrect password. Please try again.", "Login
Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
 return 0; // Invalid password
 }
 }
 else
 {
 return 0; // User not found
 }
 }
 }
 }
 catch (MySqlException ex)
 {
 MessageBox.Show("Database connection error: " + ex.Message,
"Connection Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
 return -1; // Database connection error
 }
 }
```

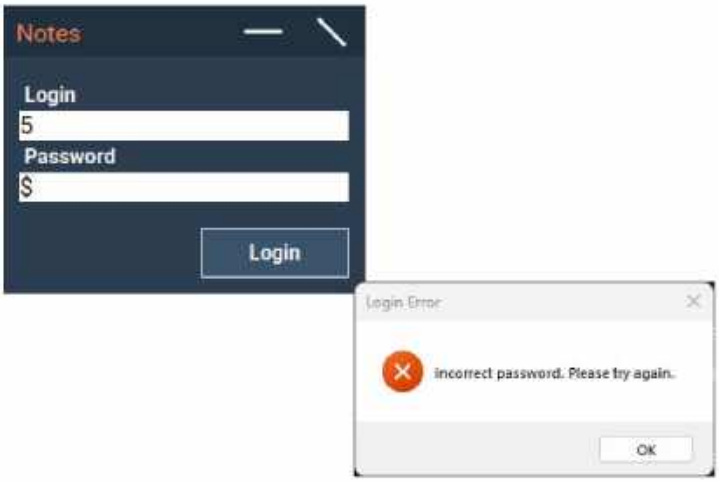

Рисунок 12 – Повідомлення про помилку

Якщо обліковий запис користувача відсутній у локальній базі даних, але існує у хмарній, програма виводить спливаюче вікно. Додаток створює новий запис у локальній базі, завантажує всі нотатки та пов'язані зображення з хмари.

```
public void SyncFromRemoteToLocal()
 {
 if(UserSession.loginStatus <= 0)
 {
 return;
 }
 int userId = UserSession.UserId;
 int localUserId = GetLocalUserIdByUsername(UserSession.Username);
 if (localUserId == 0)
 {
 //MessageBox.Show("The user with the given name was not found in
the local database.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
 return;
 }

 using (var remoteConn = new
MySqlConnection(UserSession.ConnectionString))
 {
 remoteConn.Open();
 var cmd = new MySqlCommand("SELECT id, title, content, created_at,
updated_at FROM notes WHERE user_id = @userId", remoteConn);
 cmd.Parameters.AddWithValue("@userId", userId);
 using (var reader = cmd.ExecuteReader())
 {
 while (reader.Read())
 {
 int remoteNoteId = reader.GetInt32("id");
 string title = reader.GetString("title");
 string content = reader.GetString("content");
 DateTime createdAt = reader.GetDateTime("created_at");
 DateTime? updatedAt =
reader.IsDBNull(reader.GetOrdinal("updated_at")) ? (DateTime?)null
: reader.GetDateTime("updated_at");
 string htmlContent = content;
 string passwordHash = UserSession.Hash;
 byte[] encryptionKey = Main.GenerateKeyFromHash(passwordHash);
 string decryptedContent = Main.Decrypt(htmlContent,
encryptionKey);
 if (!NoteExistsInLocal(remoteNoteId))
 {
 var images = LoadNoteImagesFromRemote(remoteNoteId);
```

```
 SaveNoteToLocalDatabase(localUserId, remoteNoteId, title,
decryptedContent, createdAt, updatedAt, images);
 }
 else if (NoteNeedsUpdate(remoteNoteId, updatedAt))
 {
 var images = LoadNoteImagesFromRemote(remoteNoteId);
 UpdateLocalNoteContent(remoteNoteId, title, decryptedContent);
 }
 }
 }
 }
 }
```

Так як контент віддаленної бази даних зашифровано, то при синхронізації контент розшифровується.

```
public static string Decrypt(string encryptedText, byte[] key)
 {
 byte[] fullCipher = Convert.FromBase64String(encryptedText);

 using (Aes aes = Aes.Create())
 {
 aes.Key = key;

 // Извлекаем IV
 byte[] iv = new byte[16];
 Buffer.BlockCopy(fullCipher, 0, iv, 0, iv.Length);
 aes.IV = iv;

 using (ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key,
aes.IV))
 {
 // Извлекаем зашифрованный текст
 byte[] cipherText = new byte[fullCipher.Length - iv.Length];
 Buffer.BlockCopy(fullCipher, iv.Length, cipherText, 0,
cipherText.Length);

 byte[] decryptedBytes = decryptor.TransformFinalBlock(cipherText,
0, cipherText.Length);
 return Encoding.UTF8.GetString(decryptedBytes);
 }
 }
 }public static string Decrypt(string encryptedText, byte[] key)
 {
 byte[] fullCipher = Convert.FromBase64String(encryptedText);

 using (Aes aes = Aes.Create())
 {
```

```
aes.Key = key;

// Извлекаем IV
byte[] iv = new byte[16];
Buffer.BlockCopy(fullCipher, 0, iv, 0, iv.Length);
aes.IV = iv;

using (ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key,
aes.IV))
{
// Извлекаем зашифрованный текст
byte[] cipherText = new byte[fullCipher.Length - iv.Length];
Buffer.BlockCopy(fullCipher, iv.Length, cipherText, 0,
cipherText.Length);

byte[] decryptedBytes = decryptor.TransformFinalBlock(cipherText,
0, cipherText.Length);
return Encoding.UTF8.GetString(decryptedBytes);
}
}
}
```

Якщо обліковий запис користувача відсутній у локальній базі даних, але існує у хмарній, програма виводить спливаюче вікно. Додаток створює новий запис у локальній базі, завантажує всі нотатки та пов'язані зображення з хмари. Після завершення процесу користувач автоматично авторизується та може працювати з додатком у звичайному режимі. Приклад повідомлення про успішну реєстрацію користувача зображено на рис.13. Якщо ж при якихось обставинах обліковий запис не зможе створитись, або не зможе авторизуватись після створення запису, то буде виведено відповідне вікно помилки, рис.14.

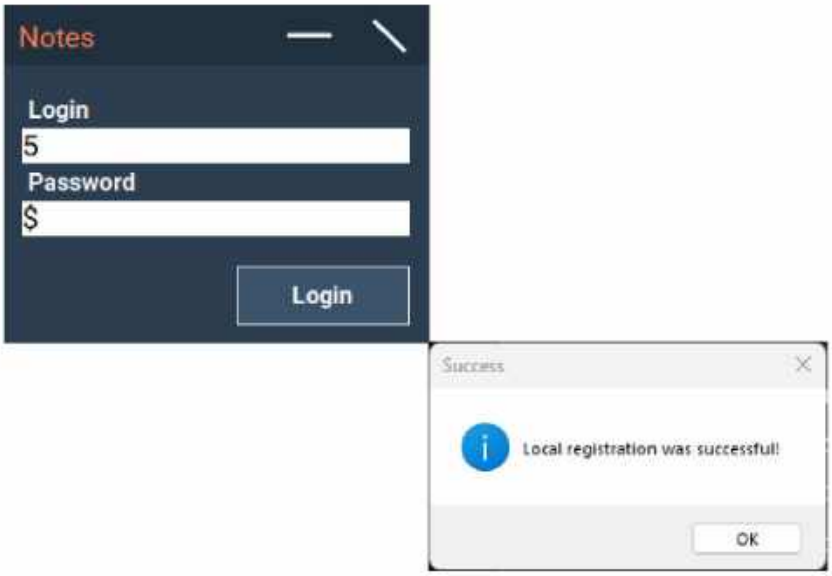

Рисунок 13 – Повідомлення про успішну реєстрацію

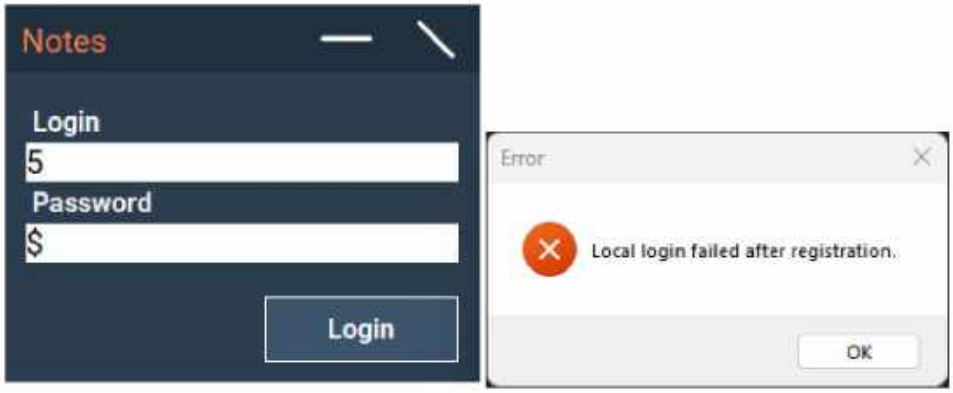

Рисунок 14 – Повідомлення про невдалу реєстрацію або авторизацію користувача

У випадку, коли обліковий запис користувача не знайдено в жодній базі даних, програма пропонує створити новий обліковий запис, рис.15.

```
public static bool Register(string username, string password)
 {
 using (var conn = new MySqlConnection(connectionString))
 {
 conn.Open();

 // Check if a user with this username already exists
```

```
 var checkCmd = new MySqlCommand("SELECT COUNT(*) FROM users WHERE
username = @username", conn);
 checkCmd.Parameters.AddWithValue("@username", username);
 var userExists = Convert.ToInt32(checkCmd.ExecuteScalar()) > 0;

 if (userExists)
 {
 // User with this username already exists
 return false;
 }

 // Hash the password
 string passwordHash = HashPassword(password);

 // Add the new user
 var cmd = new MySqlCommand("INSERT INTO users (username,
password_hash) VALUES (@username, @passwordHash)", conn);
 cmd.Parameters.AddWithValue("@username", username);
 cmd.Parameters.AddWithValue("@passwordHash", passwordHash);
 cmd.ExecuteNonQuery();

 return true; // Registration successful
 }
 }
```

На екрані з'являється вікно реєстрації, де користувач може ввести логін та пароль для створення облікового запису. Після успішної реєстрації новий користувач отримує повідомлення про успішну реєстрацію, та доступ до функціоналу додатку, і його дані автоматично зберігаються у хмарній базі даних, рис.16.

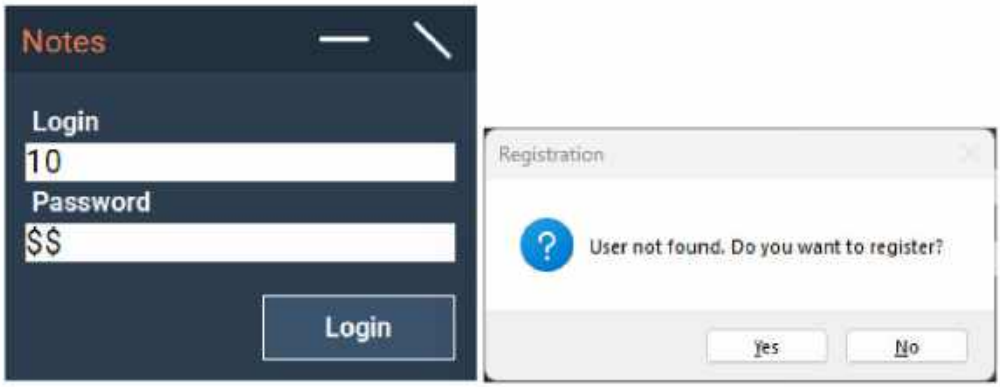

Рисунок 15 – Пропозиція реєстрації нового користувача

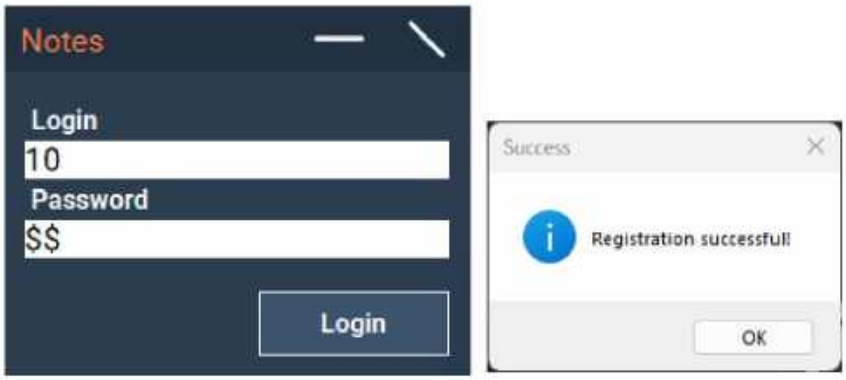

Рисунок 16 – Повідомлення про успішну реєстрацію нового користувача

Якщо додаток не може встановити з'єднання з хмарною базою через відсутність інтернет-з'єднання або інші технічні проблеми, програма переходить у режим офлайн. Користувач отримує повідомлення про відсутність підключення до сервера та інформацію про те, що всі його дані будуть зберігатися лише у локальній базі даних, рис.17. У цьому режимі доступ до вже існуючих нотаток забезпечується виключно через локальну базу, а всі дії (створення, редагування чи видалення нотаток) логуються. Після наступного запуску додатку з активним інтернет-з'єднанням програма виконує процес двосторонньої синхронізації. Усі зміни, зроблені в режимі офлайн, автоматично передаються до хмарної бази даних. Одночасно з цим нові чи оновлені дані з хмари завантажуються до локальної бази. Це забезпечує узгодженість інформації між локальним і хмарним середовищем, що дозволяє уникнути втрати даних.

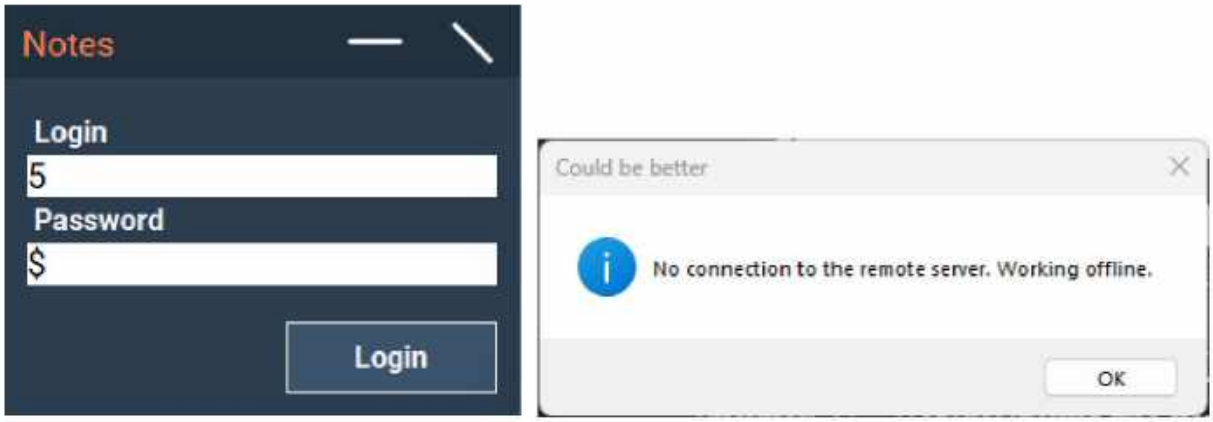

Рисунок 17 – Повідомлення про недоступність віддаленого серверу.

Такий підхід гарантує стабільну роботу додатку в будь-яких умовах, забезпечуючи комфорт і безпеку для користувачів. Уся ця логіка роботи спрямована на те, щоб забезпечити максимальну зручність для користувачів. Додаток дозволяє працювати як в онлайн-, так і в офлайн-режимах, забезпечуючи безперервність роботи. Усі зміни синхронізуються автоматично, що створює відчуття плавності та зручності під час використання. Завдяки вбудованим механізмам реєстрації та синхронізації додаток гарантує стабільність, безпеку й доступність даних у будь-яких умовах.

Інтерфейс програми «Нотатки» можна побачити на рис. 18. Він виконаний у мінімалістичному стилі, який спрямований на створення максимально комфортного середовища для користувача. Основний акцент зроблений на простоті, зрозумілості та легкості взаємодії з додатком. Відсутність зайвих елементів дозволяє уникнути перевантаження інтерфейсу, що особливо важливо для більшості користувачів, які можуть відчувати дискомфорт через надлишок функцій чи складність у навігації.

Рисунок 18 – Інтерфейс програми «Нотатки»

Простота дизайну сприяє інтуїтивній зрозумілості. У програмі використовується обмежена кількість кнопок і полів, щоб залишити тільки найнеобхідніші функції. Це дозволяє зосередити увагу користувача на основних завданнях і забезпечити легкість у використанні додатку. У наведеному інтерфейсі присутні такі кнопки:

- кнопка створення нової нотатки;
- кнопка збереження нотатки;
- кнопка видалення нотатки;
- стандартні кнопки для згортання та закриття додатку.

Усі зазначені кнопки додатку розташовані у верхній частині інтерфейсу, що можна побачити на рис. 19. Такий підхід забезпечує користувачеві зручний доступ до основних функцій програми, оскільки верхнє розташування є інтуїтивно зрозумілим і легкодоступним.

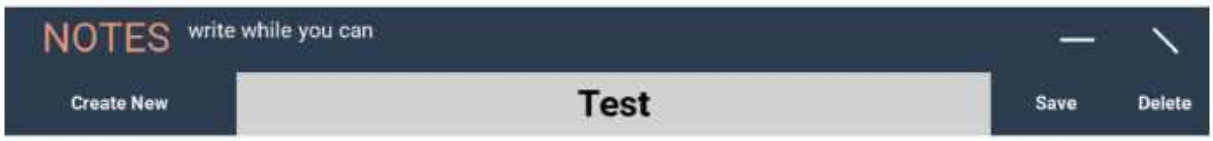

Рисунок 19 – Усі кнопки додатку зверху додатку

Для роботи з існуючими нотатками передбачений список, де користувач може вибрати потрібну йому нотатку для перегляду чи редагування. Список реалізований таким чином, щоб користувач міг швидко знаходити свої записи, спираючись на їх заголовки, рис. 20. Нижче вказано код для отримання списку усіх нотаток даного користувача.

```
public async Task LoadUserNotesAsync()
 {
 listViewNotes.Items.Clear();

 try
 {
 using (var conn = new SQLiteConnection($"Data
Source={localDb._dbPath};Version=3;"))
 {
 await conn.OpenAsync();
 var cmd = new SQLiteCommand("SELECT id, title, created_at FROM
notes WHERE user_id = @userId", conn);
 cmd.Parameters.AddWithValue("@userId", localUserId);

 using (var reader = await cmd.ExecuteReaderAsync())
 {
 while (await reader.ReadAsync())
 {
 int noteId = reader.GetInt32("id");
 string title = reader.GetString("title");
 string dateCreated =
reader.GetDateTime("created_at").ToString("dd.MM.yyyy");

 ListViewItem item = new ListViewItem(title)
 {
 Tag = noteId
 };
 item.SubItems.Add(dateCreated);
 listViewNotes.Items.Add(item);
 }
 }
 }
 }
 catch (Exception ex)
 {
 MessageBox.Show($"Error loading notes: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error); }}
```

Рисунок 20 – Панель з списком усіх нотатків користувача

Основна частина інтерфейсу складається з двох ключових полів. У верхній частині розташоване поле заголовка, яке є редагованим. Користувач може змінювати назву своєї нотатки, вносячи до неї відповідні зміни. Це поле завжди знаходиться в зручній зоні видимості, що робить роботу з ним швидкою та ефективною, рис. 21.

Під полем заголовка розташоване основне поле для введення вмісту нотатки, рис.22. Це місце, де користувач може записувати текст, а також додавати зображення. Для зручності роботи з зображеннями передбачено кілька способів їх додавання. Зображення можна вставити через буфер обміну або перетягнути в поле за допомогою функції drag&drop. Такий підхід забезпечує універсальність у роботі з мультимедійним контентом і дозволяє користувачам адаптувати процес під свої потреби.

Нижче буде наведено код, для завантаження контенту нотатки. Ця функція викликається, коли користувач обирає з списку одну з нотаток.

```
public async Task DisplayNoteWithImages(int noteId)
 {
 richTextBox1.Clear();
 string htmlContent = LoadNoteContent(noteId);
 Dictionary<int, Image> images = LoadNoteImages(noteId);
 htmlContent = Regex.Replace(htmlContent, @"<\/?(html|body)>", "",
RegexOptions.IgnoreCase);
 var parts = Regex.Split(htmlContent, @"<<IMG_(\d+)>>");

 foreach (var part in parts)
 {
 if (int.TryParse(part, out int imageIndex) &&
images.ContainsKey(imageIndex))
 {
 Clipboard.SetImage(images[imageIndex]);
 richTextBox1.Paste();
 Clipboard.Clear();
 }
 else
 {
richTextBox1.AppendText(System.Web.HttpUtility.HtmlDecode(part));
 }
 }
 }
```

Для коректного збереження нотаток було відділено зображення та текст та збережено з форматуванням html, для зручного зберігання з подальшою можливістю редагування та читання.

```
public string ConvertRichTextBoxToHtml()
 {
 string htmlContent = "<html><body>";
 int currentPosition = 0;
 int imageIndex = 1;

 foreach (var imageData in imagesList)
 {
 int imagePosition = imageData.Position;
 if (imagePosition > currentPosition && imagePosition <=
richTextBox1.Text.Length)
 {
 string textBeforeImage =
richTextBox1.Text.Substring(currentPosition, imagePosition -
currentPosition);
 htmlContent += System.Web.HttpUtility.HtmlEncode(textBeforeImage);
 }

 htmlContent += $"<<IMG_{imageIndex}>>";
 imageIndex++;
```

```
 currentPosition = imagePosition;
 }

 if (currentPosition < richTextBox1.Text.Length)
 {
 htmlContent +=
System.Web.HttpUtility.HtmlEncode(richTextBox1.Text.Substring(curre
ntPosition));
 }

 htmlContent += "</body></html>";
 return htmlContent; }
```

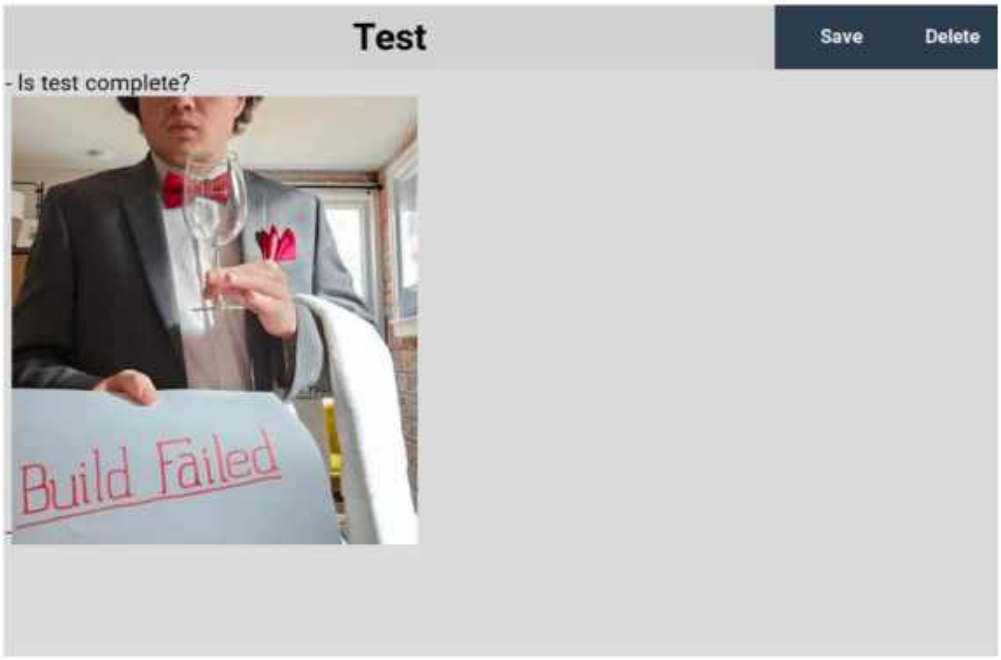

Рисунок 21 – Права частина додатку, де розміщено заголовок нотатку та основне поле з інформацією

Інтерфейс також враховує ергономічні аспекти: усі кнопки та функції розташовані таким чином, щоб забезпечити доступність і зручність навіть для нових користувачів. Мінімалістичний підхід у дизайні сприяє зосередженості на основних завданнях, а відсутність зайвих функцій знижує ймовірність помилок або плутанини під час роботи з додатком.

Кнопка створення є одним із основних інструментів додатку «Нотатки», що дозволяє користувачеві почати роботу над новою нотаткою. Її функціональність реалізована таким чином, щоб забезпечити інтуїтивно зрозумілий і швидкий процес створення нових записів без додаткових кроків чи складних налаштувань.

При натисканні кнопки створення програма автоматично очищує всі активні поля інтерфейсу. Поле заголовка та основне текстове поле стають порожніми, що сигналізує користувачеві про готовність програми до введення нової інформації. Крім того, видаляється прив'язка до ідентифікатора поточної нотатки, яка редагувалася до цього моменту. Це означає, що наступна нотатка буде збережена як новий запис у базі даних.

Алгоритм роботи кнопки створення дуже простий, що робить її зрозумілою навіть для нових користувачів. Вона не виконує жодних складних дій з базами даних, а лише готує інтерфейс для роботи з новою нотаткою. Остаточне збереження нової нотатки виконується за допомогою кнопки збереження, яка додає запис у локальну та хмарну бази даних залежно від їх стану.

Ця кнопка є важливим компонентом додатку, оскільки її простота сприяє комфортному створенню нових записів. Мінімалістичний підхід у її функціоналі допомагає уникнути можливих помилок та спрощує роботу з програмою. Користувач може зосередитися на введенні нового контенту, знаючи, що процес збереження буде виконаний відповідно до поточного стану баз даних за допомогою окремої кнопки збереження.

Кнопка збереження є ключовим елементом додатку «Нотатки», яка дозволяє користувачеві зберігати зміни у створених нотатках, або ж зберігати нові. Її функціональність адаптована до поточного стану баз даних та забезпечує стабільне збереження даних у різних сценаріях роботи додатку.

Якщо обидві бази даних (локальна та хмарна) активні, кнопка збереження записує зміну безпосередньо у дві бази. Процес виконується синхронно: спочатку дані додаються або оновлюються в локальній базі, після чого відправляються у

хмарну базу. У разі успішного завершення операції з хмарною базою програма підтверджує збереження, рис. 22. У таких умовах логування не відбувається, оскільки всі дані одразу синхронізуються між базами.

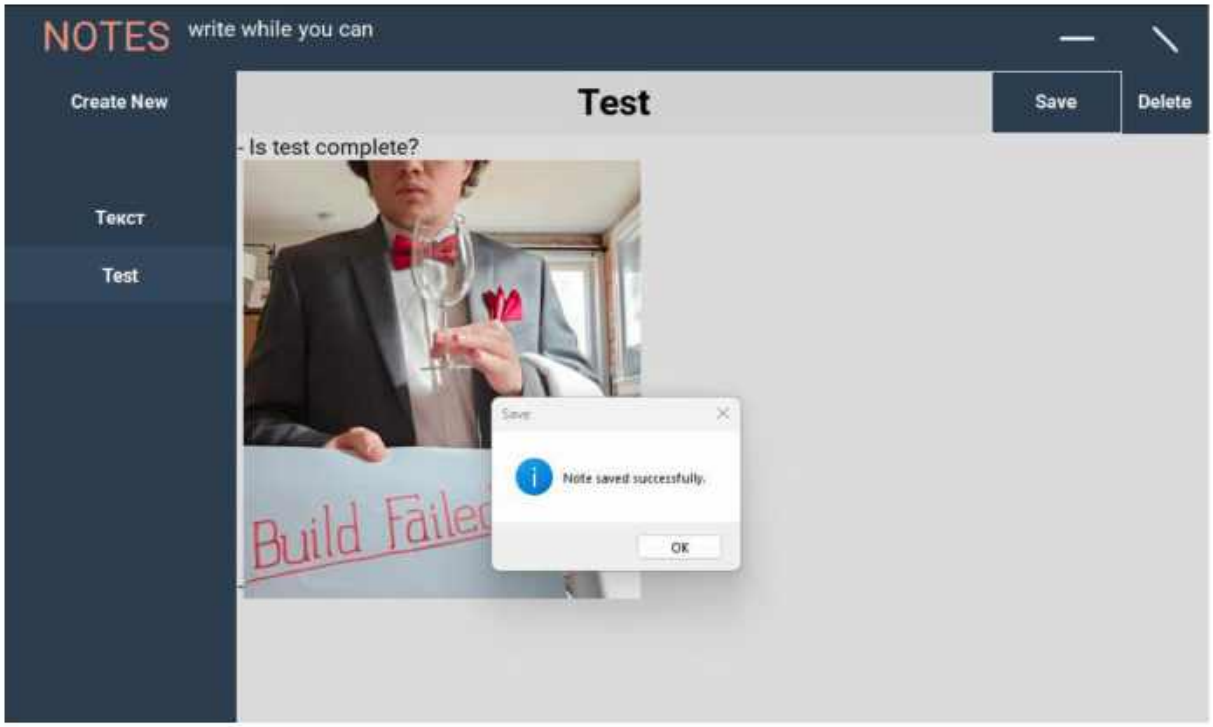

Рисунок 22 – Результат виконання збереження нотатки

Кнопка видалення призначена для повного видалення обраної нотатки з баз даних. Її функціональність передбачає видалення запису з локальної бази, а за наявності активного підключення до хмари — також із хмарної бази.

Коли активні обидві бази, видалення виконується синхронно. Після натискання кнопки програма дасть відкриє вікно з підтвердженням виконання операції, рис. 23, та якщо користувач натисне на кнопку «так», програма видалить нотатку з локальної бази, а потім надсилає відповідний запит на сервер для видалення даних із хмарної бази. Після завершення роботи, програма видасть повідомлення про виконання, рис.24.

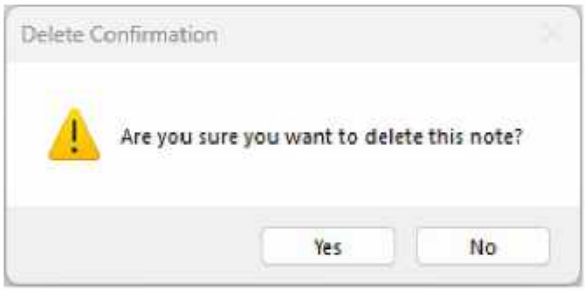

Рисунок 23 – Вікно підтвердження видалення нотатки

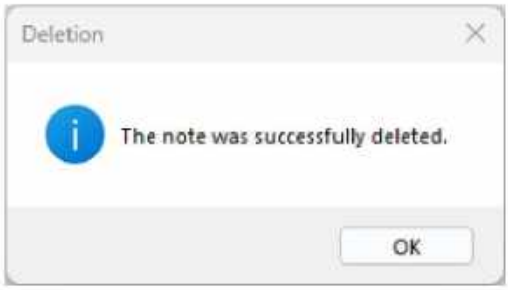

Рисунок 24 – Вікно після успішного видалення нотатки

Якщо програма працює виключно з локальною базою через відсутність доступу до хмари, кнопка збереження записує зміни тільки у локальну базу даних. Усі зміни автоматично логуються, щоб зафіксувати дії користувача. Це необхідно для подальшої синхронізації. Лог містить інформацію про тип операції (створення, оновлення чи видалення), ідентифікатор нотатки та час виконання операції. При наступному вході у додаток, при первинному підключенні до хмарної бази з'єднання відновлюється, всі збережені зміни передаються до хмари автоматично.

ВИСНОВКИ

У процесі розробки програмного забезпечення було проведено детальний аналіз функціональних та технічних вимог, що дало змогу визначити ключові цілі та завдання проєкту. На основі цього аналізу були сформовані чіткі технічні специфікації, які забезпечили систематичний підхід до реалізації поставлених завдань. У ході роботи було використано сучасні технології та інструменти, що відповідають поточним стандартам галузі, а також застосовано передові практики програмування.

Розробка передбачала створення якісного програмного коду, інтеграцію баз даних та проєктування зручного інтерфейсу користувача. Особлива увага приділялася зручності використання системи, щоб забезпечити інтуїтивність і простоту виконання задач для кінцевих користувачів. У результаті вдалося досягти високої функціональності програмного забезпечення, що повністю відповідає початковим вимогам.

Особлива увага приділялася створенню зручного інтерфейсу користувача, який дозволяє швидко орієнтуватися у додатку та виконувати основні завдання: створення, редагування та збереження нотаток. Інтерфейс спроектований таким чином, щоб бути інтуїтивно зрозумілим навіть для недосвідчених користувачів.

Загалом, створення додатку було успішно завершено, відповідно до встановлених вимог та цілей. Результатом роботи є базовий набір функцій, які відповідають початковим потребам користувачів та сприяє ефективній роботі з нотатками. У результаті створення додатку були досягнуті позитивні результати. Програма забезпечує ефективний обмін даними, зберігання та обробку інформації, що сприяє зручному та безпечному збереженню нотаток користувачів.

Розроблений застосунок є базовою платформою для подальшого розвитку. Планується вдосконалення його функціональності шляхом додавання нових можливостей, таких як:

- поліпшення механізмів синхронізації даних між локальною та серверною базами даних;
- розширення підтримки мультимедіа, зокрема можливості зберігання відео чи аудіо в нотатках;
- впровадження функцій пошуку та сортування за категоріями для зручнішої організації інформації;
- вдосконалення серверної частини додатку, а саме додавання окремого API серверу для обробки запитів. Це зменшить навантаження на основний сервер при великій кількості запитів від різних користувачів, та збільшить надійність зберігання даних користувача.

Розробка продемонструвала потенціал для подальшого вдосконалення застосунку, забезпечуючи основу для додавання нових функцій у відповідь на потреби користувачів. Створене програмне забезпечення не лише відповідає початковим технічним вимогам, а й слугує платформою для впровадження додаткових інновацій у майбутньому.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. About SQLite. URL: https://www.sqlite.org/about.html (дата звернення: 01.11.2024)
2. SQLite. URL: https://www.sqlitetutorial.net/what-is-sqlite (дата звернення: 4.11.2023)
3. MySQL. URL: https://www.oracle.com/be/mysql/what-is-mysql/ (дата звернення: 02.11.2024)
4. MySQL: https://dev.mysql.com/doc/ (дата звернення: 01.11.2024)
5. MySQL. URL: https://www.mysql.com/ (дата звернення: 01.11.2024)
6. C# (C Sharp): URL: https://www.geeksforgeeks.org/csharp-programming-language (дата звернення: 10.11.2024)
7. What is C#. URL: https://uk.sharpcoderblog.com/blog/introduction-to-csharp (дата звернення: 10.11.2024)
8. Python : URL: https://docs.python.org/uk/3/tutorial/index.html (дата звернення: 10.11.2024)
9. Evernote. URL: https://www.sqlite.org/about.html (дата звернення: 15.11.2024)
10. Марк Дж. Прайс – C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development. / 2019 – 820 С.
11. Роб Майлз – C# Programming Yellow Book / 2019 – 212С.
12. Джон Вієскас – SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL / 2018 – 960 С