

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ДЕСКТОПНОГО ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ
НОТАТОК
DESKTOP SOFTWARE DEVELOPMENT FOR NOTE TAKING

Виконав(ла): студент(ка) 4 курсу, групи КНТ-210
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

КОВЕРСУН О.С.
(ПРИЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК Т.В.
(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.
(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КОВЕРСУНА Олександра Сергійовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка десктопного застосунку для створення нотаток. Desktop Software Development for Note Taking

керівник проєкту (роботи) к.т.н., доцент, ФЕДОРОНЧАК Тетяна Василівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “24” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір і обґрунтування структури системи. 3. Основні рішення реалізації компонентів системи. 4. Експлуатація та тестування.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ФЕДОРОНЧАК Т.В., доцент		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання « 15 » квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір і обґрунтування структури системи.	2 тиждень	Розділ 2
4	Основні рішення реалізації компонентів системи.	3-4 тижні	Розділ 3
5	Експлуатація та тестування.	5 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
7	Нормоконтроль та рецензування.	7 тиждень	
8	Захист роботи.	8 тиждень	

Студент(ка)

Підпис Олександр КОВЕРСУН
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

Тетяна ФЕДОРОНЧАК
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 98 с., 6 табл., 35 рис., 3 дод., 13 джерел.

НОТАТКИ, JAVA, JAVA FX, POSTGRES SQL, WINDOWS, ІНФОРМАЦІЯ.

Об'єкт дослідження – програмні засоби для створення нотаток.

Предмет дослідження – десктопні застосунки для створення нотаток.

Мета роботи – розробка десктопного застосунку, що дозволяє користувачам вести роботу з нотатками.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Java, інтегроване середовище розробки IntelliJ IDEA, фреймворк JavaFX, СКБД PostgreSQL, персональний комп'ютер під управлінням операційної системи Microsoft Windows 10.

Результати. Створено програмний Windows-застосунок, який надає можливість користувачам швидко та зручно вести роботу з нотатками.

Висновки. Розроблено програмне забезпечення для зручної організації та управління інформацією у вигляді нотаток для користувачів, воно надає можливості створювати, змінювати, видаляти, шукати, сортувати та закріплювати нотатки.

Галузь використання – організація інформації у вигляді нотаток для власного використання та в будь-яких підприємствах, де є потреба у систематизації внутрішньої інформації та оптимізації доступу до неї.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 98 pages, 6 tables, 35 figures, 3 appendixes, 13 sources.

NOTES, JAVA, JAVAFX, POSTGRESQL, WINDOWS, INFORMATION.

Object of study is software for note-taking.

Subject of study is desktop software for note-taking.

The purpose of the work is to develop desktop software that allows users to work with notes.

Materials, methods, and tools: object-oriented programming, programming language Java, integrated development environment IntelliJ IDEA, JavaFX framework, DBMS PostgreSQL, personal computer under the control of the Microsoft Windows 10 operating system.

Results. The Windows software application is created, which allows users work with notes quickly and conveniently.

Conclusions. Software has been developed for convenient organization and management of information in the form of notes for users, it provides the ability to create, modify, delete, search, sort and pin notes.

The field of use is an information organization in the form of notes for the own usage and in any company that needs to systematize internal information and optimize access to it.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Поняття ведення (створення) нотаток	9
1.2 Види ведення (створення) нотаток.....	10
1.3 Приклади ПЗ для цифрових нотаток.....	14
1.4 Висновки за розділом 1	20
2 Вибір і обґрунтування структури системи	21
2.1 Вибір засобів для проєктування та реалізації	21
2.2 Визначення функціональних вимог	24
2.3 Вибір СКБД та проєктування.....	26
2.4 Структура системи	30
2.5 Висновки за розділом 2	32
3 Основні рішення реалізації компонентів системи.....	33
3.1 Алгоритм функціонування програми.....	33
3.2 Опис розроблених структурних класів	35
3.3 Опис вікон програми.....	43
3.4 Висновки за розділом 3	47
4 Експлуатація та тестування.....	48
4.1 Експлуатація програми.....	48
4.2 Тестування програми	56
4.3 Висновки за розділом 4	57
Висновки	58
Перелік джерел посилань	59
Додаток А Технічне завдання	61
Додаток Б Текст програми	65
Додаток В Слайди презентації.....	92

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- JDK – Java Development Kit;
SQL – Structured Query Language;
ПЗ – програмне забезпечення;
ПК – персональний комп'ютер.

ВСТУП

У наш час, коли людина поглинає велику кількість інформації майже кожен день, зберігати все в голові є складною задачею. Це підкреслюється різноманіттям джерел інформації, таких як: соціальні мережі, телебачення, цифрові медіа, література, тощо. Окрім цього, джерелами інформації є люди, які можуть надавати інформацію яку обов'язково необхідно запам'ятати, проте, через вже вказане різноманіття та/або зайнятість іншими справами, потребуючими розумових здатностей, людина може забувати надану їй інформацію. Зважаючи на це, зараз актуальним є використання інструментів для організації, управління та зберігання важливої інформації.

Одним з популярних методів зберігання такої інформації є використання нотаток. Вони допомагають людям наприклад для запам'ятовування важливих даних або для структурування думок, це дозволяє підвищити продуктивність та ефективність дій.

В цій дипломній роботі розглядається процес розробки програмного забезпечення для створення нотаток. Це ПЗ забезпечує швидкий доступ до збереженої інформації завдяки зручному інтерфейсу користувача та увесь необхідний спектр операцій з нотатками. В якості мови програмування було обрано Java, яка є широко використовуваною мовою програмування, що забезпечує високий рівень безпеки, в поєднанні з фреймворком JavaFX, який надає широкий та зручний для використання інструментарій, це дозволяє створювати ПЗ із високоякісним графічним інтерфейсом користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття ведення (створення) нотаток

Ведення (створення) нотаток є процесом запису інформації з різних джерел та платформ, роблячи нотатки, автор записує суть інформації, звільняючи свій розум від необхідності все згадувати. З появою писемності та зростанням грамотності нотатки традиційно і, майже завжди, робили від руки (часто в блокнотах), але поява програмного забезпечення для ведення нотаток зробила ведення цифрових нотаток можливим і широко поширеним. Ведення нотаток є основоположною навичкою в управлінні особистими знаннями [1].

Якщо казати про короткочасну пам'ять людини, то людина може охопити поглядом близько семи предметів, запам'ятовуючи у середньому від п'яти до дев'яти одиниць інформації, які вона спроможна точно відтворити через кілька десятків секунд після їхнього пред'явлення [2]. Тому, обґрунтовано вважають, що обсяг короткочасної пам'яті становить (7 ± 2) елементи [2]. Саме через малу кількість елементів яку людина може запам'ятати, доцільно використовувати нотатки, які дозволять з плином часу перечитати або переслухати інформацію внесену у них.

Процес ведення нотаток забезпечує краще засвоєння матеріалу завдяки активній участі людини в інформаційному потоці. Записуючи інформацію вона аналізує, перефразує та систематизує інформацію, що сприяє ефективному запам'ятовуванню та розумінню вивченого. Цей процес перетворення інформації на власні слова підвищує здатність до критичного мислення і забезпечує глибоку обробку знань.

Ведення нотаток допомагає заощадити час, оскільки забезпечує швидкий доступ до важливих даних під час підготовки до іспитів або роботи над проєктами. Чітко організовані нотатки дозволяють легко знаходити ключову інформацію, уникаючи потреби переглядати великі обсяги

матеріалу. Такий підхід ефективно спрощує процес вивчення та дозволяє економити час на підготовку та перегляд.

Нотатки також можуть бути корисними для поширення інформації. Одержувачі нотаток можуть використовувати їх для власного навчання чи роботи над спільним проектом.

Також, нотатки можуть виступати інструментом для візуалізації прогресу в досягненні цілей. Нотатки можуть слугувати нагадуванням про статус цілей та мотивувати до виконання цих цілей, особливо коли справа доходить до тривалих і поступових завдань. Можна записувати які просування були зроблені в певних цілях, або які перешкоди не дають виконати їх.

1.2 Види ведення (створення) нотаток

Виділяють такі основні види ведення нотаток: лінійні, нелінійні, цифрові.

Лінійне ведення нотаток – це процес запису інформації в тому порядку, в якому ви її отримуєте, лінійні нотатки зазвичай являють собою хронологічний план лекції або тексту, лінійне ведення нотаток є найпоширенішим засобом ведення нотаток, проте можливість просто розшифрувати все, що говориться або на слайді презентації, досить висока [1].

До лінійного ведення відносять метод ієрархічного запису та метод речень.

Ієрархічний запис зазвичай ведеться зверху донизу сторінкою, використовуючи заголовки та маркери для структурування інформації. Загальна система складається із заголовків, у яких використовуються римські цифри, літери алфавіту й арабські цифри на різних рівнях [1].

Конспектування речень – це просто запис кожної теми у вигляді короткого простого речення. Цей метод добре підходить для швидких уроків,

де розглядається багато інформації. Автор записує кожен нову думку, факт або тему на окремому рядку. Уся інформація записується, але не розподіляється за основними та другорядними темами. Нотатки можуть бути пронумеровані або позначені маркерами, що показують, де починається нова думка [1].

До нелінійного ведення відносять:

- графіки (діаграми);
- інтелект-карти (карти розуму);
- нотатки Корнела;
- SQ3R;
- направляючі нотатки.

Запис нотаток у вигляді графіків або діаграм використовуються для більш наглядного представлення інформації (візуального), їх можна використовувати при порівнянні якоїсь інформації або при пошуку зв'язків між елементами.

Інтелект-карти це ще один візуальний підхід для створення нотаток, часто використовується в управлінні проєктами, ці карти представляють собою деревовидну структуру в якій є центральний головний вузол, від якого розгалужуються інші гілки з підписами, також, часто ці гілки розбиті на категорії які виділені певними кольорами, для більш зрозумілої орієнтації при читанні цієї карти.

Метод Корнелла — популярна система конспектування, розроблена приблизно в 1950-х роках Вальтером Пауком, професором Корнельського університету, цей метод включає в себе поділ ваших нотаток на три розділи: розділ основних приміток, стовпчик підказок і розділ підсумків [3].

Метод SQ3R є акронімом своїх п'яти кроків: survey (оглянути), question (запитати), read (прочитати), recite (переказати), та review (переглянути) [4].

Цей метод заснований на виконанні певних етапів для більш ефективного запам'ятовування прочитаної інформації, спочатку користувач

читає інформацію, яку необхідно запам'ятати, намічає перелік питань для себе, перечитує інформацію вже знаючи питання, та намагається дати на них відповіді, після чого пише нотатки з відповідями на питання та перечитує те що він написав.

Направляючі нотатки являють собою конспекти, зазвичай створені викладачами, які містять пропуски які запропоновано заповнити тим хто слухає викладачів, тим самим мотивуючі слухати уважно та заглиблюватися у суть. Після заповнення таких нотаток вони перетворюються в повноцінний конспект, який може слугувати для допомоги у запам'ятовуванні інформації яку розповів викладач. Завдяки тому що слухач самостійно заповнює ці пропуски це дозволяє більш ефективно запам'ятовувати важливі аспекти розказаного.

Окремою категорією є цифрові нотатки які створюються завдяки ПЗ, розробленому під задачі створення нотаток на електронних пристроях. Такі нотатки можуть навіть не відрізнятися від рукописних нотаток, оскільки в багатьох ПЗ передбачені функції зміни шрифтів, кольору, розмірів, додавання геометричних фігур, малюнків, тощо.

Також, багато застосунків мають властивість кросплатформності, що дозволяє отримати доступ до написаного з будь-яких пристроїв, які підтримуються ПЗ.

Такі види нотаток набирають популярність з ростом впливу ІТ-сфери, все більше людей використовують електронні пристрої для допомоги в роботі, і такі нотатки можуть замінювати звичайні рукописні.

Проте, в кожного з вищеописаних видів нотаток існують свої переваги та недоліки, порівняння цих пунктів можна побачити в табл. 1.1.

Таблиця 1.1 – Переваги та недоліки видів нотаток

Назва виду	Переваги	Недоліки	Примітка
Ієрархічні	Чітка структура, легкість в організації комплексної інформації	Складність використання при неструктурованих даних	Найкраще підходять для структурованих даних і проєктів
Речення	Простота, швидкість запису	Складність у пошуку конкретної інформації при великих обсягах	Добре для швидких заміток, але не для глибокого аналізу
Графіки	Наочність, допомагають визначити взаємозв'язки, тренди	Необхідність навичок у візуалізації даних, не підходять для детальних пояснень	Ефективні для аналізу і представлення кількісних даних
Інтелект-карти	Наочність, легкість розуміння, підвищують креативність	Займає багато часу, не підходять для детальних пояснень	Підходять для мозкового штурму та комплексного планування
Нотатки Корнела	Чітка, організована структура, легкість розуміння	Може займати багато часу при великій кількості нотаток	Підходять при підготовці до іспитів
SQ3R	Найкраще засвоєння інформації	Займає дуже багато часу	Використовуються переважно для глибокого навчального читання
Направляючі	Фокус на ключових моментах, покращують критичне мислення та самоаналіз	Можуть втрачати контекстуальні деталі	Ефективні для переглядів конкретних тем перед іспитами
Цифрові	Мобільність, багатофункціональність та кросплатформність	Залежність від електрики або інтернету	Ідеальні для людей, що ведуть активний цифровий спосіб життя

Як можна побачити з табл. 1.1 різні методи ведення нотаток мають свої унікальні переваги та недоліки, що робить їх підходящими для різних обставин. Наприклад, ієрархічні та нотатки Корнела добре структурують

інформацію, що ідеально для навчання, тоді як інтелект-карти покращують креативне мислення. Цифрові нотатки забезпечують гнучкість і доступність, але вимагають технологічної підтримки. Проте через розвиток комп'ютерних технологій, все більше людей віддають перевагу саме цифровим нотаткам, а саме конкретним програмам які дозволяють створювати їх.

1.3 Приклади ПЗ для цифрових нотаток

Існує дуже велика кількість програм призначених для роботи з нотатками, вони можуть відрізнятися функціоналом, заточеністю до певної групи користувачів або електронних пристроїв. Великі фірми, володіючи значними ресурсами, часто домінують на ринку, але їхні програми дійсно пропонують високий рівень якості функціоналу та користувацького досвіду. Далі пропонується розглянути три ПЗ цифрових нотаток, які не тільки популярні, але й визнані за свою ефективність і зручність у використанні.

1.3.1 Microsoft OneNote

Першим головним існуючим рішенням створення цифрових нотаток на ПК є OneNote від загальновідомої компанії Microsoft. Ця програма є складовою пакету Microsoft Office.

У програмі OneNote можна виконувати такі дії [5]:

- ввести нотатки або записувати аудіо на ноутбук;
- замальовувати та писати ідеї на планшеті;
- додавати зображення з телефону;
- миттєво шукати нотатки;
- вільно переміщувати нотатки по сторінці;
- упорядковувати ці сторінки в розділи;
- зберігати розділи в одному або кількох блокнотах;

- переключитися на пристрої та продовжити з місця, на якому ви зупинилися;
- надавати спільний доступ до блокнотів іншим користувачам, щоб усі могли одночасно переглядати й робити свій внесок.

OneNote зберігає інформацію на сторінках, організованих у розділи всередині блокнотів. Корпорація Microsoft розробила цей користувацький інтерфейс так, щоб він нагадував кільцевий швидкозшивач з вкладками, в якому користувач може безпосередньо робити нотатки та збирати матеріал з інших програм. Блокноти OneNote збирають, організовують і поширюють, можливо, неопубліковані матеріали – на відміну від текстових, які часто націлені на публікацію в той чи інший спосіб. Різниця проявляється у певних функціях і характеристиках OneNote, а саме що сторінки можуть бути довільного розміру і також не існує примусового уніфікованого макету чи структури сторінок [6].

Інтерфейс програми можна побачити на рис. 1.1.

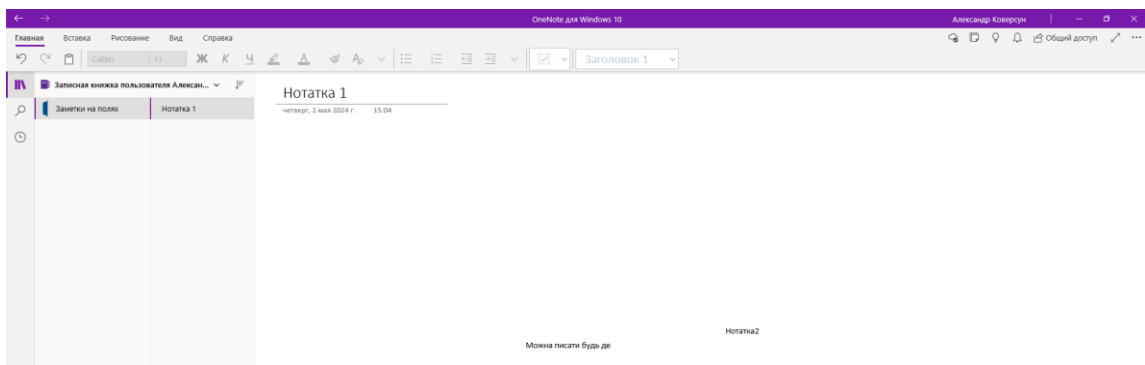


Рисунок 1.1 – Інтерфейс OneNote

Користувачі можуть переміщати сторінки у підшивці та коментувати їх за допомогою стилуса, текстового редактора чи інструментів для малювання. Користувачі можуть додавати вбудовані мультимедійні записи та гіперпосилання. Вони також можуть додавати вбудований вміст, наприклад, відео з YouTube. OneNote також інтегрує функції пошуку та індексації у сховище графіки та аудіо у вільній формі. Також, можна шукати зображення

(наприклад, знімки екрану, вбудовані скани документів, фотографії) на предмет наявності тексту. Додатково, існує можливість витягувати і копіювати тексти з фотографій і документів за допомогою оптичного розпізнавання символів [6].

Звісно, OneNote підтримує інтеграцію з іншими програмами Microsoft Office. Це означає, що користувачі можуть легко імпортувати таблиці або діаграми з Excel або фрагменти тексту з Word документів прямо до своїх нотаток, що дозволяє створювати комплексні нотатки.

Усі описані функції роблять OneNote дуже корисним інструментом для людей, які потребують ефективного інструменту для зберігання інформації, спільної роботи або організації персональних даних.

1.3.2 Google Keep

Другим ПЗ яке буде розглянуто є результатом роботи ще однієї дуже відомої фірми – Google, а саме – Google Keep.

Google Keep входить до складу безкоштовного вебпакету Google Docs, на відміну від OneNote Google Keep є вебзастосунком, а не десктопним застосунком, також як і у OneNote є мобільні версії для iOS та Android.

Google Keep дозволяє користувачам робити різні види нотаток, включаючи: тексти, списки, зображення та аудіо. Користувачі можуть встановлювати нагадування, які інтегровані з Google Now, з параметрами часу або місця. Текст із зображень можна витягувати за допомогою технології оптичного розпізнавання символів. Голосові записи, створені за допомогою Кеер, автоматично транскрибуються. Кеер може перетворювати текстові нотатки в контрольні списки. Користувачі можуть вибирати між представленням в один стовбець або у вигляді багатьох стовбців. Нотатки можуть бути позначені кольором: білим, червоним, помаранчевим, жовтим, зеленим, бірюзовим, синім або сірим. Користувачі можуть натиснути кнопку «Копіювати в Google Doc», яка автоматично копіює весь текст у новий

документ Google Docs. Користувачі можуть створювати нотатки і списки голосом. Нотатки можуть бути категоризовані за допомогою ярликів, список яких можна знайти на навігаційній панелі додатку [7].

На рис. 1.2 можна побачити інтерфейс Google Keep.

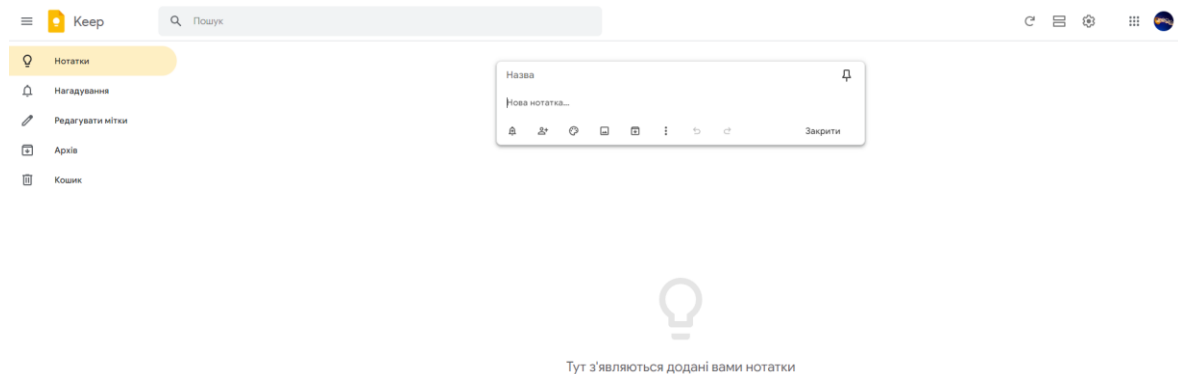


Рисунок 1.2 – Інтерфейс Google Keep

Google Keep є швидким інструментом для нотаток, який можна порівняти з цифровими стікерами. На відміну від OneNote, який функціонує як повноцінний цифровий блокнот із складними можливостями організації нотаток, Google Keep пропонує більш простий підхід. Це робить його ідеальним варіантом для швидкого зберігання думок і ідей на ходу.

Інтеграція з іншими застосунками і простота використання роблять Google Keep відмінним інструментом для людей, які цінують швидкість і ефективність при роботі з нотатками.

1.3.3 Evernote

Останнім рішенням, що буде розглянуто, є Evernote, цей програмний продукт має як окремий застосунок для ПК так і застосунки для мобільних пристроїв і також має вебверсію, функціонал Evernote схожий на розглянутих до цього конкурентів.

Має можливість інтеграції з великою кількістю застосунків такими як: Google Drive, Slack, Outlook, MS Teams, Zapier, Gmail тощо [8].

Проте одним з мінусів є те що в безкоштовній версії функціонал урізаний та наприклад розміри нотатки в окремоті та усіх нотаток в цілому, не дуже великий, цього повинно вистачити середньостатистичному користувачу, проте при активному створенні нотаток, може виникнути необхідні покупки «Personal» або «Professional» тарифів, також є окремий тариф «Teams» для командної роботи, включаючи можливість надання доступу зі специфічними дозволами на редагування або перегляд, що може бути корисним у корпоративному середовищі.

Також, існують вже створені шаблони нотаток, які можна використовувати не створюючи їх з «нуля», такі як:

- трекер для схуднення;
- встановлення цілей;
- трекер заявок на роботу;
- профіль персонажа;
- контрольний список гуртожитку;
- облік коштів.

Інтерфейси десктопного та мобільного застосунків можна побачити на рис. 1.3.

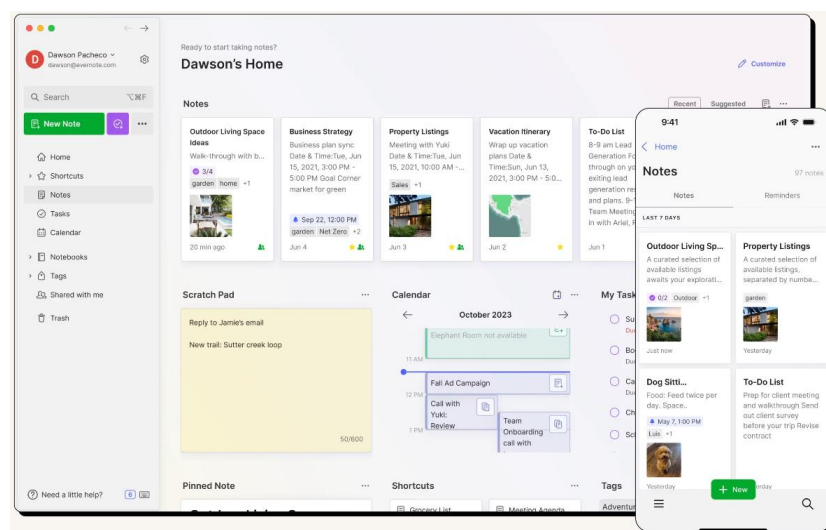


Рисунок 1.3 – Інтерфейс Evernote [8]

Очевидно, що кожне з цих ПЗ має свої мінуси та плюси та знайде попит в певних груп людей, в табл. 1.2 представлено порівняння описаних рішень створення цифрових нотаток.

Таблиця 1.2 – Порівняння ПЗ для цифрових нотаток

Критерій порівняння	Microsoft OneNote	Google Keep	Evernote
Принципи організації	Блокноти, секції блокнотів, групи секцій, теги	Теги, кольори	Блокноти, теги, стеки
Наявність синхронізації	Так	Так	Так
Наявність вебкліпінгу	Так	Лише посилання та заголовки вебсторінок	Так
Розпізнавання ручного вводу	Так	Ні	Так
Наявність пошуку	Так	Так	Так
Перевірка правопису	Так	Так	Так
Можливість друку	Так	Ні	Так
Можливість писати «чорнилами»	Так	Ні	Не можна поєднувати текст і «чорнила»
Вихідний формат	Pdf, onepkg, docx, xps	Експорт до Google Docs (pdf, doc, docx, odt)	Html, enex, pdf
Вартість	Безкоштовне, але повна інтеграція з Office вимагає підписки на Microsoft 365	Безкоштовне	Безкоштовне, але з обмеженнями які прибираються з платними тарифами

1.4 Висновки за розділом 1

В даному розділі було досліджено предметну область, визначено ключові поняття, пов'язані з об'єктом дослідження.

В подробицях розібрано існуючі види ведення нотаток, проаналізовано їх переваги та недоліки, визначено їх приблизні галузі застосування.

Більш детально розглянуто вид нотаток який відноситься до об'єкта дослідження, а саме – цифровий вид нотаток за допомогою трьох прикладів ПЗ які дозволяють створювати такі нотатки. Розібрано кожен з програм, їх функції, галузі застосування, створено порівняльну таблицю цих програм.

На основі дослідження предметної області та зважаючи на порівняльну таблицю та більший функціонал саме у десктопних застосунків (Microsoft OneNote, Evernote) порівняно з вебзастосунком (Google Keep) було прийнято рішення про створення десктопного застосунку для створення нотаток. Основна увага приділяється ефективному захисту та структуризації інформації.

В результаті аналізу предметної області було визначено вимоги до розроблюваного ПЗ та поставлено завдання на розробку (див. додаток А).

2 ВИБІР І ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ

2.1 Вибір засобів для проєктування та реалізації

Порівнюючи мови програмування для створення десктопних застосунків з інтерфейсом, часто звертають увагу на такі параметри:

- швидкість виконання коду;
- зручність розробки;
- доступність бібліотек, фреймворків;
- безпекова складова.

Порівняння, яке зроблено в табл. 2.1 показало що всі мови програмування володіють високими можливостями для створення графічних застосунків, але за сукупністю переваг було обрано мову Java, тому що єдиним мінусом який може виникати – це швидкість виконання коду, вона є високою, але Сі-подібні мови працюють швидше за рахунок наявності ручного керування пам'яттю та вказівників, проте з цього можуть виникати проблеми переповнення пам'яті, проблеми доступу до невиділеної або звільненої пам'яті, також при використанні покажчиків можна випадково виділити масив неправильної довжини, з виходом за межі.

Java це одна з найбільш популярних мов програмування: за останніми оцінками, з нею працюють понад 12 мільйонів ІТ-інженерів у світі. Використовуючи Java, можна створити будь-яку програму: інтернет-магазини, програмне забезпечення для ноутбуків, ігрових консолей, наукових суперкомп'ютерів та інших пристроїв [9]. Java є типобезпечною мовою. Типобезпека гарантує відсутність у програмах помилок, які важко знайти і які пов'язані з неправильною інтерпретацією пам'яті комп'ютера. Це робить процес розробки надійнішим і передбачуваним, а отже швидшим [10].

У мові Java є чіткі певні стандарти на введення-виведення, графіку, геометрію, діалог, доступ до баз даних і інших типових застосувань. Завдяки цим особливостям, застосунки на Java мають значно кращу

кросплатформність, ніж C++ [10]. Усі ці факти дозволяють обрати мову Java для проєктування та розробки ПЗ для створення нотаток.

Таблиця 2.1 – Порівняння мов програмування

Критерій порівняння	Java	C#	C++	Python
Швидкість виконання коду	Висока	Висока	Висока, проте потребує більше часу на розробку через ручне керування пам'яттю	Середня, через те що Python – інтерпретована мова
Зручність розробки	Висока	Висока	Середня	Висока
Garbage Collector	+	+	-	+
Підтримка Unicode	Так	Так	Так, але вимагає використання окремих типів, напр. wstring	Так
Доступність бібліотек, фреймворків	Велика різноманітність	Велика різноманітність	Велика різноманітність, але частина може бути написана на C, що може викликати деякі проблеми	Велика різноманітність
Безпекова складова	Висока	Висока	Висока, але потрібно уважно слідкувати за виділеною пам'яттю та покажчиками	Висока
Кросплатформність	Повна	Часткова	Часткова	Повна
Підтримка розробниками	Висока	Висока	Висока	Висока

У наш час двома популярними Java GUI-фреймворками є JavaFX та Swing, хоча кожен із них має свої переваги та недоліки, обидва часто

використовуються в розробці інтерфейсів для Java-застосунків, в табл. 2.2 наведено порівняння цих фреймворків, як можна побачити з таблиці JavaFX надає більш сучасні засоби для розробки, підтримується на мобільних пристроях, є більш актуальним та зручним, відповідно цього порівняння вибір зроблено на користь JavaFX.

Таблиця 2.2 – Порівняння Swing та JavaFX

Критерій порівняння	JavaFX	Swing
Дата виходу	Грудень 2008	Березень 1998
Гнучкість	Можна використовувати як для десктопних застосунків так і для веброзробки	Орієнтований на десктопні застосунки
Графіка	2D, 3D	2D
Сумісність	Підтримує Windows, Linux, MacOS, Android, iOS	Підтримує Windows, Linux, MacOS
Зручність використання	Висока	Середня
Продуктивність роботи	Висока	Середня
Актуальність	Активно підтримується та розвивається	Не оновлюється

JavaFX – це платформа клієнтських додатків нового покоління з відкритим вихідним кодом для настільних, мобільних і вбудованих систем, побудованих на Java. Це спільна робота багатьох людей і компаній з метою створення сучасного, ефективного і повнофункціонального інструментарію для розробки повноцінних клієнтських додатків [11].

JavaFX-застосунки можуть бути орієнтовані на десктопні, мобільні та вбудовані системи. Бібліотеки та програмне забезпечення доступні протягом усього життєвого циклу програми [11].

Проектувати «вікна» програми дуже зручно за допомогою вбудованого інструмента Scene Builder, завдяки ньому можна маніпулювати різними об'єктами які можна використати у вікнах програми, це можна робити як безпосередньо в IDE так і в окремому застосунку.

2.2 Визначення функціональних вимог

Після визначення з інструментами розробки ПЗ, було виділено функціональні вимоги яким має відповідати програма:

- користувач повинен мати можливість зареєструватися;
- користувач повинен мати можливість увійти до застосунку після реєстрації;
- користувач повинен мати можливість поставити прапорець «запам'ятати мене» під час входу;
- користувач повинен мати можливість вийти зі свого акаунта;
- користувач повинен мати можливість створювати нотатки;
- користувач повинен мати можливість додавати заголовок нотатки;
- користувач повинен мати можливість змінювати атрибути тексту нотаток (змінення розміру шрифту, сімейства, кольору, оформлення у вигляді маркованих та нумерованих списків, вирівнювання);
- користувач повинен мати можливість бачити усі свої нотатки після створення і при повторних входах у свій акаунт;
- користувач повинен мати можливість додавати картинки до своїх нотаток;
- користувач повинен мати можливість обирати картинки які йому необхідно додати;

- користувач повинен мати можливість додати лише картинку в якості нотаток, без тексту;
- користувач повинен мати можливість бачити дату створення нотаток;
- користувач повинен мати можливість шукати нотатки за їх текстом, заголовком та за їх датою створення;
- користувач повинен мати можливість скасовувати пошук;
- користувач повинен мати можливість відсортувати нотатки за їх заголовком та за їх датою створення;
- користувач повинен мати можливість закріплювати та відкріплювати нотатки;
- користувач повинен мати можливість редагування тексту нотаток;
- користувач повинен мати можливість видалення нотатки;
- система повинна зберігати інформацію про усіх зареєстрованих користувачів;
- система повинна зберігати логін користувача якщо він натиснув прапорець «запам'ятати мене» під час входу;
- система повинна зберігати дані нотаток для конкретного користувача;
- система повинна завантажувати та відображати нотатки користувача для якого здійснено вхід;
- система повинна відображати результати пошуку;
- система повинна відображати повідомлення про помилки, якщо вони є.

Виходячи з цих функціональних вимог було розроблено можливі сценарії дій користувача відповідно до системи (ПЗ), це представлено у вигляді діаграми прецедентів на рис. 2.1.



Рисунок 2.1 – Діаграма взаємодії користувача і системи

2.3 Вибір СКБД та проєктування

З діаграми взаємодії користувача і системи зрозуміло, що необхідно зберігати і заповнювати інформацію про всіх зареєстрованих користувачів, та інформацію про створені ними нотатки, індивідуально для кожного користувача. Зважаючи на невелику кількість сутностей можна було б подумати що для цієї задачі підійде файлова система, тому що вона надає простий спосіб зберігати дані без додаткових залежностей. Проте бази даних забезпечують набагато більш ефективні можливості для керування даними

завдяки індексації, структурованій організації та оптимізованим алгоритмам запиту.

Також була б проблема коли нотатки для кожного користувача зберігалися лише на тому ж пристрої де він увійшов в акаунт і, відповідно, якщо він би втратив доступ до цього пристрою, або спробував увійти у свій акаунт з іншого пристрою, він би не побачив усіх своїх нотаток, база даних ліквідує ці недоліки. Додатково база даних є набагато захищеним варіантом, вона забезпечує контроль доступу на рівні користувачів, дозволяючи визначати ролі та права кожного користувача для доступу до даних – це запобігає несанкціонованому перегляду або зміні даних.

Бази даних також спрощують пошук, фільтрацію та обробку великих обсягів інформації, особливо коли йдеться про одночасний доступ різних користувачів. Вони підтримують транзакції, дозволяючи забезпечити цілісність даних, навіть за наявності помилок у процесі запису.

З огляду на переваги баз даних, було розглянуто 3 популярні СКБД, а саме: SQLite, MySQL та PostgreSQL, щоб визначити, що найкраще підійде при створенні ПЗ, що буде розроблятися.

В табл. 2.3 показано порівняння вищезгаданих СКБД, з цієї таблиці можна побачити що SQLite призначена для нескладних структур даних, з нескладними типами даних, наприклад в ній немає підтримки типу дати і вона може зберігатися лише як текст, вона краще працює з невеликими обсягами даних. Головним мінусом є те що вона не має окремого серверу, отже якщо розроблюване ПЗ під'єднати до цієї бази, то дані будуть зберігатися лише локально, що дуже погано. MySQL та PostgreSQL є клієнт-серверними СКБД, MySQL більше орієнтована на швидкі операції між сутностями баз, PostgreSQL більше орієнтована на потужності інструментів та дуже надійну цілісність зберігання даних.

Проте ця потужність коштує швидкості, тому що при кожному під'єднаному клієнті до бази даних створюється окремий процес який займає приблизно 10 мб оперативної пам'яті [12].

Таблиця 2.3 – Порівняння SQLite, MySQL, PostgreSQL

Параметр	SQLite	MySQL	PostgreSQL
Архітектура	Вбудована в застосунок, без сервера	Клієнт-сервер	Клієнт-сервер
Вартість	Безкоштовна	Є безкоштовні і платні варіації	Безкоштовна
Сумісність	Вбудовується в багато мов програмування	Широко підтримується, має драйвери	Широко підтримується, має драйвери
Надійність	Середня	Висока	Дуже висока, завдяки підвищеному рівню стійкості та цілісності
Продуктивність	Швидка з невеликими обсягами даних	Швидка	Зменшується при великій кількості процесів
Масштабованість	Обмежена обсягом пам'яті	Масштабується горизонтально	Масштабується вертикально та горизонтально
Підтримка транзакцій	Підтримує ACID	Підтримує ACID, реплікацію	Підтримує ACID, реплікацію

Порівнюючи переваги та недоліки розглянутих СКБД, було зроблено вибір на користь PostgreSQL, оскільки при маніпуляціях обліковими даними користувачів головний пріоритет віддається цілісності даних під час операцій, а не швидкості, також звичайно потрібна саме клієнт-серверна база, яка добре працює з великими обсягами даних, масштабується, має можливості до резервного копіювання, не обмежена обсягом пам'яті, підтримується багатьма мовами програмування.

PostgreSQL сумісна з широким спектром мов програмування та платформ. Це означає, що якщо вам коли-небудь знадобиться перенести базу

даних на іншу операційну систему або інтегрувати її з певним інструментом, то з базою даних PostgreSQL це, швидше за все, буде простіше, ніж з іншою СКБД [12].

Також, PostgreSQL здатна ефективно обробляти декілька завдань одночасно, ця характеристика відома як паралелізм. Вона досягає цього без блокування читання завдяки реалізації Multiversion Concurrency Control (MVCC), яка забезпечує атомарність, узгодженість, ізоляцію та довговічність транзакцій, також відому як ACID-сумісність [12].

Додатково, останні бенчмарк-тести показують, що PostgreSQL може зрівнюватися або навіть наближатися до MySQL за швидкістю роботи [12].

Після вибору СКБД завданням було спроектувати як буде виглядати база даних, за допомогою якої буде здійснюватися зв'язок з ПЗ, де будуть зберігатися усі необхідні дані та звідки вони будуть записуватися.

Було створені дві головні таблиці в базі даних, які пов'язані між собою.

Таблиця users включає поля:

- id (integer) – первинний ключ;
- username (varchar) – унікальне поле для логіна користувача;
- password (varchar) – поле хешованого пароля користувача;
- remember_me (boolean) – прапорець «запам'ятати мене», за замовчуванням false, якщо прапорець при вході користувача буде натиснутий, зміниться на true, при повторних запусках буде здійснено автоматичний вхід до акаунта.

Таблиця notes:

- id (integer) – первинний ключ;
- user_id (integer) – зовнішній ключ, автор нотаток (зв'язок один-до-багатьох з таблицею users), один користувач може мати багато нотаток;
- text (text) – текст нотатки (може містити у собі шлях до картинок, містить у собі атрибути тексту);
- date (date) – дата створеної нотатки;

- pinned (boolean) – статус закріплення нотатки, за замовчуванням false, якщо користувач закріпить нотатку статус зміниться на true;
- original_index (integer) – індекс позиції на якій розміщено нотатку в таблицю нотаток, це необхідно для того, щоб при відкріпленні нотатка поверталася на свою оригінальну позицію;
- title (text) – заголовок нотатки, якщо заголовок не додано, сюди буде поміщено перші 5 слів з тексту нотатки.

Схему бази даних можна побачити на рис. 2.2.

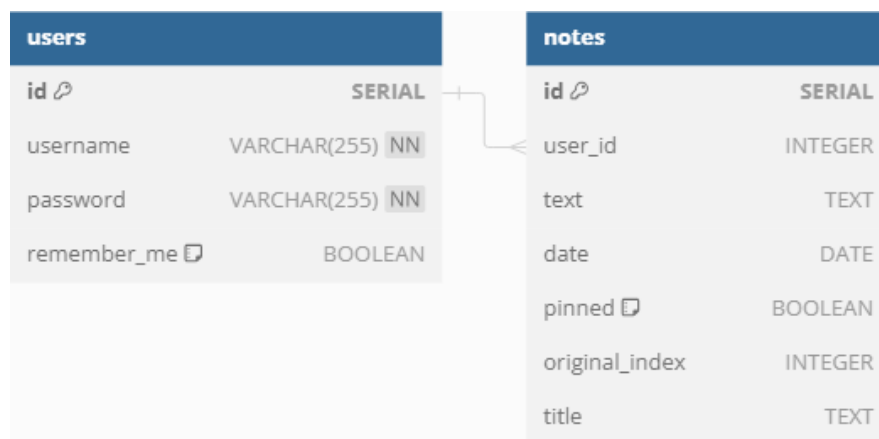


Рисунок 2.2 – Схема бази даних

2.4 Структура системи

За задумом готове ПЗ має виглядати наступним чином: користувач відкриває програму, перед ним з'являється головне вікно з полями для вводу логіну та пароля, якщо він ще незареєстрований йому запропоновано це зробити натисканням на кнопку «Sign Up», при натисканні на неї повинно відкритися інше вікно – вікно реєстрації, при успішній реєстрації користувача, його дані будуть поміщені в базу даних, в таблицю users. Після успішної реєстрації це вікно повинно закриватися.

Далі, коли ще неавтентифікований користувач введе дані для входу та натисне на кнопку «Login» введені ним дані будуть звірятися з даними в базі даних, якщо усі дані співпадуть, то вхід буде успішно здійснено, також він

зможє натиснути на прапорець «Remember me», якщо він хоче щоб вхід в акаунт при повторному запуску був здійснений автоматично, без повторного вводу облікових даних.

При потраплянні до вікна з нотатками, уся інформація про нотатки відповідно до автентифікованого користувача буде завантажуватися з бази даних, з таблиці notes.

Таким чином, необхідно розробити три графічних вікна, створити базу даних з необхідними таблицями, розробити логіку класу-контролера, за допомогою якого буде працювати увесь функціонал ПЗ, потрібно розробити окремі класи User та Note для створення об'єктів відповідних класів, для коректної обробки та зберігання даних про користувачів та їх нотаток у базі даних. Розроблену структуру проєкту можна побачити на рис. 2.3.

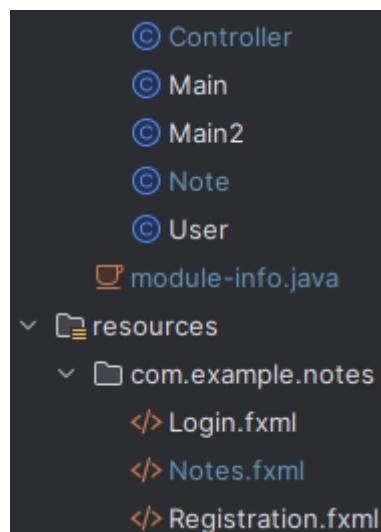


Рисунок 2.3 – Структура проєкту

Файли формату fxml є реалізацією графічних вікон програми, Login – головне вікно, де користувачу пропонується або зареєструватися або увійти у вже існуючий акаунт, Registration – вікно реєстрації, Notes – вікно нотаток користувача, доступне лише при успішному вході користувача.

Controller – клас який реалізує увесь функціонал проєкту.

Note, User – класи для зручного керування об'єктів відповідних класів та взаємодії з базою даних.

Main, Main2 – файли які містять функції main необхідні для запуску.

Module-info.java – файл модулів проєкту які використовуються, підключено драйвер sql.

2.5 Висновки за розділом 2

В цьому розділі визначено основний інструментарій необхідний для розробки ПЗ.

Проведено порівняння мов програмування, обґрунтовано вибір у бік мови програмування Java, порівняно два фреймворки цієї мови програмування, вибір зроблено на користь JavaFX.

Визначено функціональні вимоги до ПЗ, спроектовано схему взаємодії користувача і ПЗ. Відповідно функціональних вимог визначено необхідність використання бази даних.

Для роботи з базою даних порівняно СКБД, обрано PostgreSQL.

Продумано та визначено структуру проєкту, що розроблюється.

3 ОСНОВНІ РІШЕННЯ РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм функціонування програми

За структурною схемою, описаною в розділі 2, було створено схему алгоритму функціонування програми, її можна побачити на рис. 3.1.

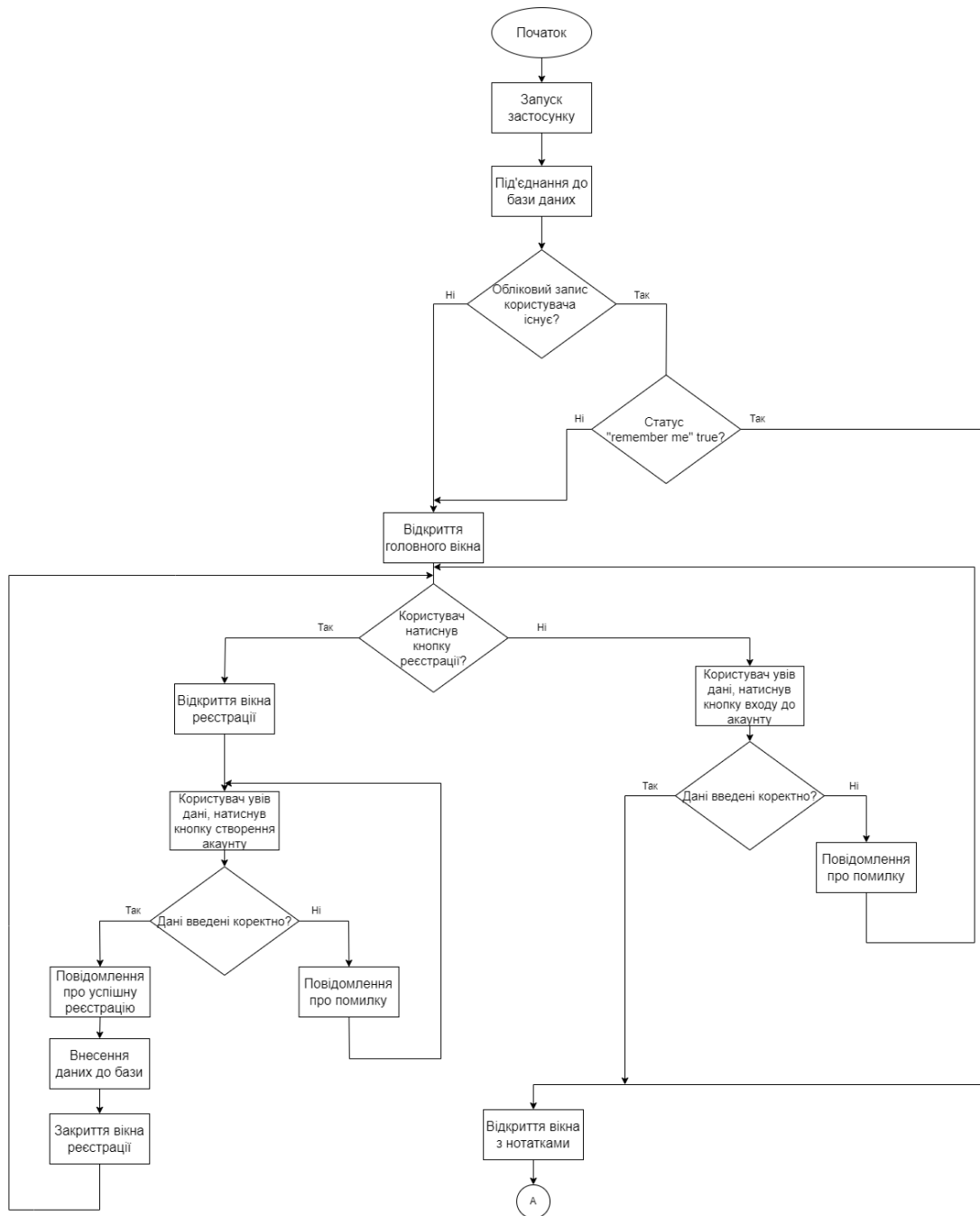


Рисунок 3.1а – перша частина рисунку 3.1

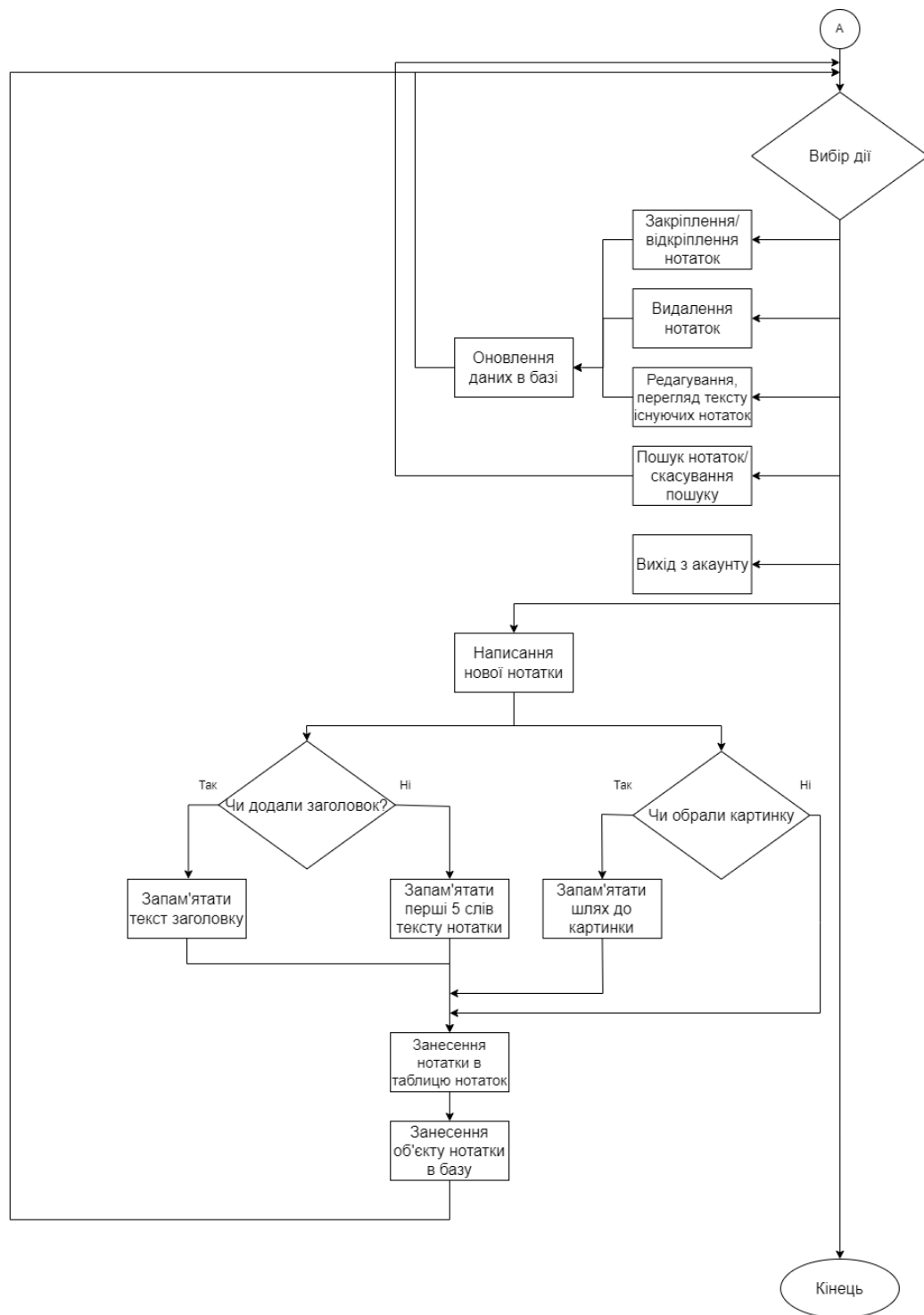


Рисунок 3.1б – друга частина рисунку 3.1

За розробленою схемою алгоритму буде простіше створювати ПЗ, а саме чітко уявляти логіку програми, її структуру та потоки даних між компонентами. Це спрощує ідентифікацію можливих помилок та слабких місць на початку процесу розробки, що може заощадити час та ресурси під час подальшої реалізації проєкту.

3.2 Опис розроблених структурних класів

Клас User створений для представлення даних користувача, що зберігаються в базі даних, в таблиці users, це дає змогу легко маніпулювати цими даними в коді, без використання неструктурованих запитів і масивів. Цей клас має три поля: id, username та password; також в ньому наявні гетери та сетери для цих полів, конструктор за замовчуванням та ще два конструктора, перший з атрибутом id, другий – без нього, це зроблено для того що, коли користувач ще не зареєстрований, його облікових даних ще немає в базі, відповідно немає id, тому для розміщення об'єктів які відповідають за користувача який щойно зареєструвався і поміщення його в базу буде використано саме об'єкт без id, де він і отримає свій унікальний id, а якщо користувач вже зареєстрований то його дані (дані про створені ним нотатки) будуть завантажені за допомогою вже привласненому йому id, це дозволяє гнучко керувати об'єктами цього класу.

Використання класу User спростило розробку програми, оскільки будь-які зміни в структурі даних користувача можна було зробити шляхом модифікації цього класу, не зачіпаючи більшу частину решти коду програми.

Клас Notes створений для представлення даних про нотатки, що зберігаються в базі даних, в таблиці notes. Цей клас має поля id, text, date, pinned, original_index, title. Також в цьому класі є методи доступу (гетери та сетери) для кожного з полів. Також в цьому класі є три конструктори, перший конструктор за замовчуванням, для створення порожніх об'єктів, наступні конструктори схожі за логікою як і в класі User, один конструктор з id = 0, він потрібен для нотаток які ще не були створені користувачем і, відповідно, не були додані в базу даних, другий конструктор містить усі атрибути класу, і він використовується при маніпуляціях з нотатками які вже були додані до бази даних, як і в випадку з класом User це дозволяє гнучко взаємодіяти між об'єктами (нотатками) для кожного користувача.

Клас `Controller` є основним і наймасштабнішим серед розроблених класів, він відповідає за бекенд частину програми та реалізує такі методи:

- `getDatabaseConnection()` – відповідає за під'єднання до створеної бази даних, в цьому методі вказується логін та пароль автора бази даних та адресу за якою необхідно під'єднуватися (`jdbc:postgresql://localhost:5432/notes`);

- `openLoginWindow()` – відповідає за відкриття головного вікна застосунку, в ньому виставляється назва вікна, розміри, вікно відкривається за допомогою файлу формату `fxml`, а саме `Login.fxml`;

- `openRegistrationWindow()` – відповідає за відкриття вікна реєстрації, в ньому виставляється назва вікна, розміри, вікно відкривається за допомогою файлу формату `fxml`, а саме `Registration.fxml`;

- `isValidString(String str)` – відповідає за перевірку тексту (приймає рядок), введеного в поля для вводу у головному вікні та у вікні реєстрації, має вигляд регулярного виразу (`str.matches("[a-zA-Z0-9!@#%&*?+-]+$")`), дозволяє лише використання латинських літер та додаткових спеціальних символів;

- `isUsernameTaken(String username)` – перевіряє строку з логіном який введено при спробі реєстрації на наявність в базі даних, якщо такий користувач вже є у базі повертає `true`, інакше `false`, це реалізовано `query`-запитом «`SELECT COUNT(*) FROM users WHERE username = ?`» тобто в якості `username` підставляється переданий рядок і шукається кількість збігів, якщо збіг знайдено це означає що такий користувач вже існує;

- `hashPassword(String password)` – відповідає за хешування пароля користувача, зроблено за допомогою об'єкту типу `MessageDigest` та алгоритму шифрування `SHA-256`, алгоритм є бібліотечним, з `java.security`. Вхідний пароль перетворюється в масив байтів із використанням кодування `UTF-8`, а потім застосовується метод `digest` об'єкта `MessageDigest`. Це створює хеш-значення у вигляді масиву байтів. Отриманий рядок, що представляє хеш-значення, повертається з методу.

– `showAlert(String message)` – відповідає за виведення повідомлення при помилках, текст помилки передається зі строки яку приймає метод, об'єкту типу `alert` дається заголовок та відносно нього викликається метод `JavaFX showAlertAndWait()` для того щоб повідомлення не закрилося без дії користувача;

– `registerUser(String username, String password)` – відповідає за внесення облікових даних користувача до бази даних (таблиці `users`), зроблено за допомогою `query`-запиту `INSERT INTO users (username, password) VALUES (?, ?);`

– `@FXML registerUser()` – відповідає за обробку даних введених у поля для вводу вікна реєстрації, містить перевірки на коректність введення даних користувачем, поля не можуть бути порожніми, довжина логіну та пароля не може бути довше ніж 24 символи, довжина пароля має бути мінімум 8 символів, у вікні реєстрації пароль потрібно ввести двічі, тому також перевіряється щоб паролі у двох полях для вводу співпадали. Також в ньому викликаються методи `isUsernameTaken(String username)` та `isValidString(String str)` для відповідних перевірок, якщо усі перевірки пройдені успішно викликається метод `registerUser(String username, String password)`, викликається метод `showAlert(String message)` повідомляючи про успішну реєстрацію та вікно реєстрації закривається, якщо знайдено якісь помилки то також викликається метод `showAlert(String message)`, але вікно реєстрації не закривається;

– `authenticateUser(String username, String hashedPassword)` – відповідає за автентифікацію користувача, зроблено за допомогою `query`-запиту «`SELECT id, username FROM users WHERE username = ? AND password = ?`», повертає об'єкт типу `User`, в цьому методі йде під'єднання до бази даних та перевірка чи є дані які вказує користувач у вікні логіну в базі, окрім логіну також перевіряється хеш пароля;

– `@FXML loginValidate(ActionEvent event)` – відповідає за перевірку тексту введеного у поля вікна логіну (головного вікна), в ньому

викликаються методи `isValidString(String str)`, `showAlert(String message)` при помилках у вводі та також метод `authenticateUser(String username, String hashedPassword)`, якщо усі поля введені за стандартом валідного рядка та такий користувач вже зареєстрований і хеш його пароля відповідає хешу в базі тоді змінній `currentUsername` передається логін який введено та викликається метод `openNotesWindow()`, а це вікно закривається. Також якщо `rememberMeCheckBox` був натиснутий тоді викликається метод `updateLoginState(String username, boolean rememberMe)`;

– `updateLoginState(String username, boolean rememberMe)` – відповідає за зміну статусу `remember_me` в базі (таблиці `users`) на `true` відповідно до певного користувача, зроблено за допомогою query-запиту «UPDATE users SET remember_me = ? WHERE username = ?»;

– `checkLoginState(Stage primaryStage)` – відповідає за перевірку авто-входу користувача, це зроблено за допомогою query-запиту «SELECT id, username, password FROM users WHERE remember_me = true LIMIT 1», йде під'єднання до бази даних, в таблиці `users` йде пошук користувача дані якого відповідають останнім введенням та перевіреним методом `authenticateUser`, перевіряється якщо в цього користувача статус `remember_me = true` тоді відносно цього користувача спрацює метод `openNotesWindow(Stage stageToClose)`;

– `openNotesWindow(Stage stageToClose)` – відповідає за відкриття вікна з нотатками для користувача з `currentUsername`, в ньому виставляється назва вікна, розміри, вікно відкривається за допомогою файлу формату `fxml`, а саме `Notes.fxml`, в цьому методі створюється об'єкт контролера в нього поміщується `currentUsername` та викликається метод `initialize()` відповідно до цього поля, тобто відкривається вікно з нотатками для конкретного користувача логін якого вказано при заході;

– `initialize(String username)` – відповідає за ініціалізацію основних компонентів інтерфейсу конкретного користувача для роботи з нотатками та налаштування початкового стану цих компонентів, в цьому методі

створюється список об'єктів `Note`, `notesList`, в якому будуть зберігатися нотатки завантажені з бази даних, якщо вони є. Цей список поміщується в таблицю `notesTableView`, яка є візуальним елементом вікна з нотатками.

Також, в цьому методі створюються об'єкти `WebView` які поміщуються в кожен комірок таблиці з нотатками для відображення заголовку нотатки, якщо заголовок не додано буде відображено перші 5 слів з тексту нотатки (метод `extractTitle`).

При натисканні на заголовок нотатки відображається окремо вікно для редагування та перегляду тексту нотаток, яке містить кнопку для додавання картинок (додавання картинок що при створенні нотаток що при їх зміні відбувається за допомогою методу `@FXML chooseImage`), `htmlEditor` для зміни атрибутів та текстове поле для зміни заголовку, при закритті цього вікна усі зміни застосовуються.

Після цього йде створення колонок таблиці, колонка для закріплення та відкріплення нотаток створюється методом `setupPinColumn()` після чого створюються колонки тексту та дати, вони додаються в `notesTableView`.

Після підготовки таблиці у змінну `userId` передається `Id` користувача який здійснив вхід, цей `Id` визначається методом `getCurrentUserId()`, відносно цього `id` викликається метод `loadNotesFromDatabase(int userId)`;

– `getCurrentUserId()` – відповідає за визначення `Id` користувача, зроблений за допомогою `query`-запиту «`SELECT id FROM users WHERE username = ?`», до цього методу передається `currentUsername` та після під'єднання до бази даних та перегляду таблиці `users` повертається `id` користувача, якщо користувач з `currentUsername` існує;

– `extractTitle(String text)` – потрібен для тих випадків коли заголовок не додано, тоді з усього тексту нотатки беруться п'ять перших слів і поміщують ці в якості заголовку;

– `@FXML chooseImage()` – відповідає за відкриття вікна з вибором файлів формату `.png`, `.jpg` або `.gif` на ПК користувача цей файл поміщується у `htmlEditor`. При зміні або додаванні нотаток можна комбінувати

зображення і текст, кількість доданих зображень не обмежена, в ньому викликається метод `insertImageAtCursor`.

- `insertImageAtCursor(HTML editor, String html)` – відповідає за знаходження положення курсору в `editor` (де вводиться текст нотаток та змінюються їх атрибути) та поміщення обраної картинки на місце курсору;

- `@FXML addNote()` – відповідає за додавання об'єкту нотатки в таблицю нотаток, в рядки заголовку та тексту поміщується вміст з поля для вводу заголовку та з `noteEditor`, створюється об'єкт `newNote` типу `Note`, у сетери заголовку, тексту нотатки та позиції, за якою нотатку буде розміщено у таблиці, вносяться відповідні дані. Далі перевіряється якщо текст у полі для вводу не порожній (метод `isNoteEmpty`) цей об'єкт поміщується у список `notesList`. Якщо текст порожній про це користувача буде повідомлено.

Після чого у змінну `userId` передається `Id` користувача який здійснив вхід, цей `Id` визначається методом `getCurrentUserId()`, відносно цього `id` викликається метод `saveNoteToDatabase(newNote, userId)`;

- `isNoteEmpty(String text)` – перевіряє `noteEditor` в який вноситься текст нотатки на наявність тексту (текстом вважається і картинка);

- `@FXML saveNoteToDatabase(Note note, int userId)` – цей метод відповідає за збереження нотатки до бази даних, це зроблено за допомогою query-запиту «INSERT INTO notes (user_id, text, date, pinned, original_index, title) VALUES (?, ?, ?, ?, ?, ?)», йде під'єднання до бази даних, в таблицю `notes` поміщуються дані, за допомогою сетерів, передані об'єктом `note` відповідно до `userId` певного користувача;

- `loadNotesFromDatabase(int userId)` – цей метод відповідає за завантаження інформації про нотатки користувача з певним `userId`, це зроблено за допомогою query-запиту «SELECT id, text, date, pinned, original_index, title FROM notes WHERE user_id = ?», в цьому методі йде під'єднання до бази даних, в базі даних в таблиці `notes` обираються нотатки в яких `user_id` відповідає переданому у цей метод, ці нотатки поміщуються в список об'єктів `Note notes` який буде повернено в якості результату, також

перед поверненням цього списку йде сортування закріплених та незакріплених нотаток, закріплені нотатки будуть поміщені у таблицю зверху, незакріплені будуть поміщені знизу в порядку зростання `originalIndex`;

– `setupPinColumn()` – цей метод відповідає за створення колонки закріплення та відкріплення нотаток, в цій колонці в кожній створеній нотатки буде з'являтися кнопка `pinButton` за замовчуванням вона буде у положенні з текстом `Pin`, тобто не закріпленою, при натисканні на неї текст зміниться на `Unpin`, в обробнику подій цієї кнопки створюється об'єкт типу `Note` в неї зчитується її індекс, цей об'єкт видаляється зі списку `noteList`, якщо нотатку намагаються закріпити вона поміщується нагору усього списку нотаток в таблиці, якщо нотатку намагаються відкріпити то вона розміщується за своїм оригінальним індексом, з яким вона була додано до таблиці методом `addNote()`.

Після чого викликається метод `updateNoteInDatabase()` та метод `sortAndUpdateView()`;

– `sortAndUpdateView()` – цей метод схожий на сортування в методі `loadNotesFromDatabase`, там це відбувається при завантаженні з бази даних, тут це відбувається для поточних об'єктів таблиці, остання закріплена нотатка йде догори, при відкріпленні повертається на свою початкову позицію в таблиці;

– `updateNoteInDatabase(Note note, int noteId)` – цей метод викликається при зміні об'єкту нотатки, наприклад при закріпленні або відкріпленні або при зміні тексту або заголовку нотатки, це зроблено за допомогою `query`-запиту «`UPDATE notes SET text = ?, date = ?, pinned = ?, original_index = ?, title = ? WHERE id = ?`», йде під'єднання до бази даних, в таблицю `notes` вставляються дані об'єкту нотатки відповідно певного `userId`, цей метод зроблений за аналогією методу збереження нотатки до бази;

– `@FXML searchNotes()` – цей метод відповідає за пошук нотаток у таблиці за текстом, заголовком або за датою створення, текст введений у

поле для пошуку зчитується в рядок `searchText`, дата обрана за допомогою `datePicker` поміщується у змінну дати `selectedDate`, створюється список нотаток `filteredNotes` та йде пошук введеного тексту у якості збігу підрядка в рядку нотатки які містять у собі введений текст поміщуються у список `filteredNotes`, за датою шукається повний збіг, після чого виводяться у таблицю `notesTableView`, якщо таких нотаток не знайдено ця таблиця буде порожньою;

- `@FXML undoSearch()` – цей метод відповідає за скасування запиту пошуку, поле для пошуку та дата в `datePicker` очищується, в таблицю `notesTableView` передається `notesList`;

- `@FXML deleteNote()` – метод відповідає за видалення нотатки з таблиці нотаток, користувача за допомогою повідомлення звіщають про те чи впевнений він що хоче видалити обрану нотатку, при натисканні на `ok` спрацьовує метод `deleteNoteFromDatabase(int noteId)` та нотатка методом `remove` прибирається з таблиці нотаток;

- `deleteNoteFromDatabase(int noteId)` – цей метод відповідає за видалення нотатки з бази даних, це зроблено за допомогою `query`-запиту «DELETE FROM notes WHERE id = ?», йде під'єднання до бази даних, з таблиці `notes` видаляється нотатка з переданим у метод `noteId`;

- `@FXML handleLogout(ActionEvent event)` – цей метод відповідає за обробку виходу з акаунта, в метод `updateLoginState` відповідно до активного користувача передається статус `rememberMe = false`, вікно з нотатками закривається, відкривається головне вікно.

Класи `Main` та `Main2` необхідні для запуску програми. Обидва класи містять метод `main`, в класі `Main` метод `main` викликає метод `launch`, який ініціалізує графічний інтерфейс користувача. Також у цьому класі створюється об'єкт `FXMLLoader`, що завантажує FXML-файл головного вікна (`login.fxml`) і передає його до класу `Controller`.

Клас `Main2` використовується для запуску програми, коли в проєкті включено сторонні бібліотеки або модулі (наприклад, `SQL`). Це пов'язано з

тим, що під час пакування проєкту в виконуваний JAR-файл, спроба запакувати клас з методом `main` який містить виклик `launch` може викликати помилки через специфіку завантаження залежностей проєкту. Тому, для пакування використовується метод `main` класу `Main2`, що передає аргументи `args` з `Main` без додаткових викликів до графічного інтерфейсу.

3.3 Опис вікон програми

Головне вікно програми, створене за допомогою `Login.fxml`, можна побачити на рис. 3.2.

В цьому вікні присутні елементи типу `Label`:

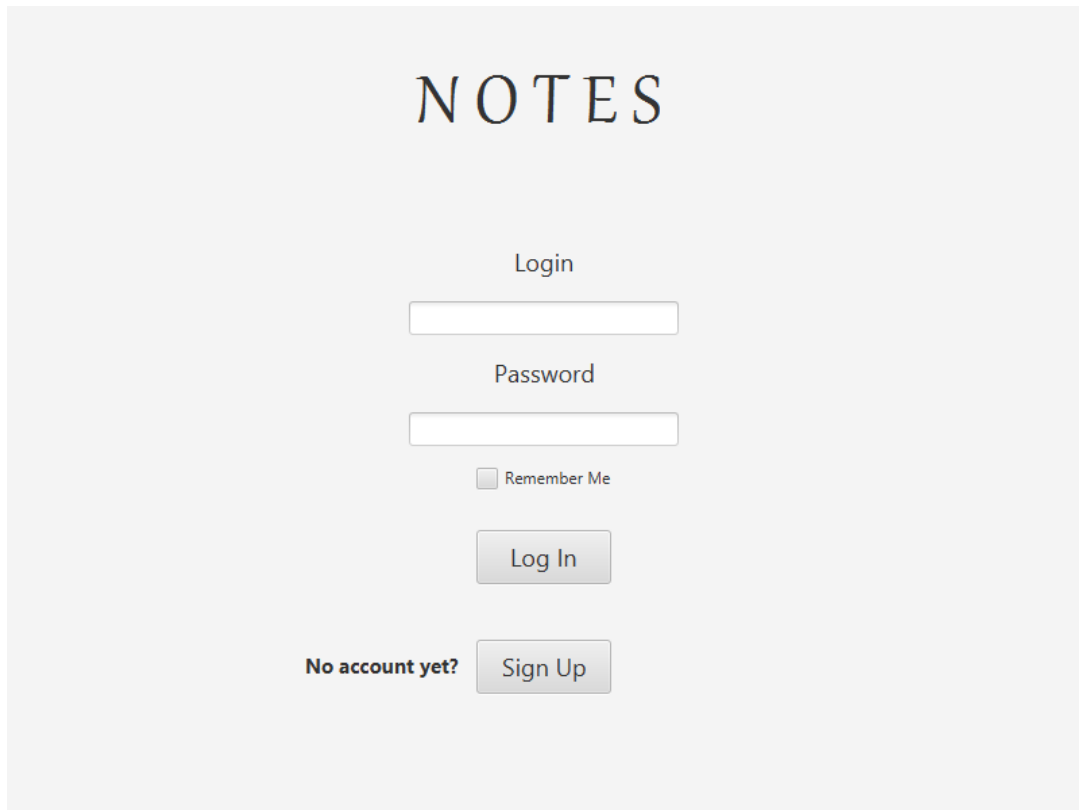
- заголовок зверху вікна, `NOTES`;
- підписи полів для вводу, `Login` та `Password`;
- підпис до чекбоксу, `Remember Me`;
- підпис біля кнопки, `No account yet?`.

Також присутні елементи типу `Button`, кнопки відкриття вікна реєстрації та входу в акаунт, елемент типу `CheckBox`, елементи типу `textField` та `passwordField` для вводу логіну та пароля відповідно.

При натисканні на кнопку `Sign Up` йде виклик методу `openRegistrationWindow` класу `Controller`, при введеннях даних та натисканні на кнопку `Log In` йде виклик методу `loginValidate`, після коректного спрацювання цього методу йде виклик методу `openNotesWindow`, також якщо користувач натиснув чекбокс йде виклик методу `updateLoginState`.

На рис 3.3 можна побачити вікно реєстрації створене за допомогою `Registration.fxml`.

На цьому вікні присутні `Label` заголовок зверху вікна, `NOTES`, `Label` підписи полів для вводу, `Login`, `Password` та `Confirm password`. Також присутній елемент типу `Button`, кнопка реєстрації, елементи типу `textField` та два `passwordField` для вводу логіну, пароля та підтвердження пароля відповідно.



NOTES

Login

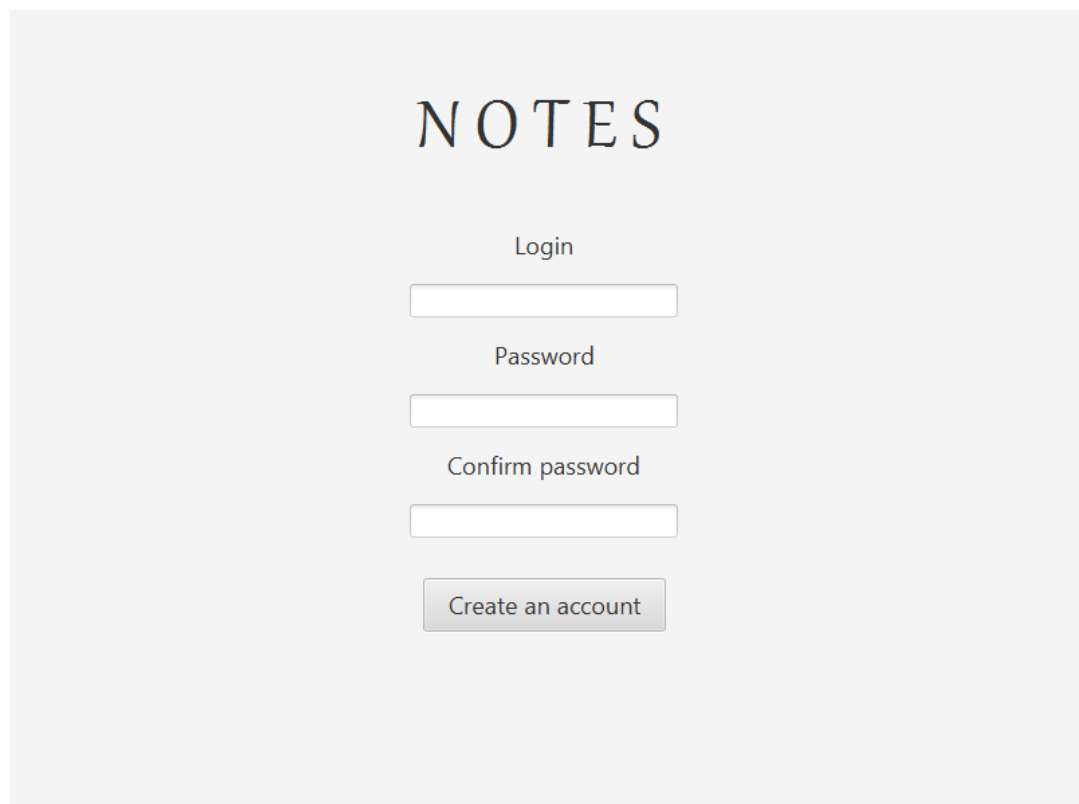
Password

☐ Remember Me

Log In

No account yet? [Sign Up](#)

Рисунок 3.2 – Головне вікно програми



NOTES

Login

Password

Confirm password

Create an account

Рисунок 3.3 – Вікно реєстрації

При натисканні на кнопку спрацює метод @FXML registerUser, якщо реєстрація успішна то це вікно буде закрите.

На рис 3.4 можна побачити останнє вікно програми – вікно з нотатками, створене за допомогою Notes.fxml.

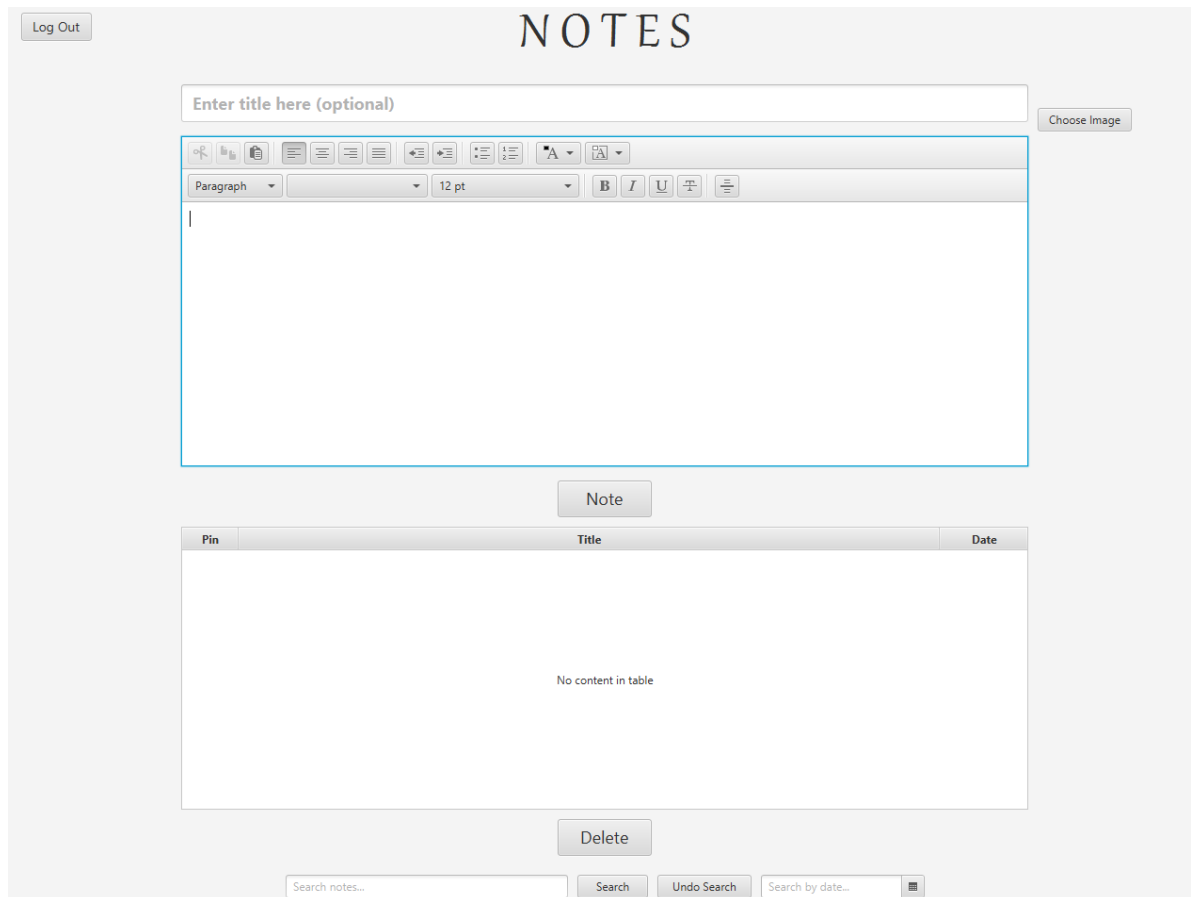


Рисунок 3.4 – Вікно з нотатками

Це вікно складається з:

- елементу типу Label заголовок зверху вікна NOTES;
- елементу типу htmlEditor для вводу тексту нотатки та змінення його атрибутів;
- елементів типу Button, кнопка виходу з акаунту, кнопка вибору зображення, кнопка відправлення нотатки, кнопка видалення, кнопка здійснення пошуку, кнопка скасування пошуку;
- елементу типу textField, з placeholder – Search notes...;
- елементу типу datePicker, з placeholder – Search by date...;

– елементу типу TableView, таблиця з нотатками.

При натисканні на кнопку Log Out спрацьовує метод @FXML handleLogout, при натисканні на кнопку Choose Image спрацьовує метод @FXML chooseImage, при натисканні на кнопку Note спрацьовує метод @FXML addNote, при натисканні на кнопку Delete спрацьовує метод @FXML deleteNote, при натисканні на кнопку Search та Undo Search спрацьовують методи @FXML searchNotes та @FXML undoSearch відповідно, при натисканні на заголовок нотатки в таблиці TableView, відкривається вікно для перегляду та редагування тексту нотатки, його можна побачити на рис. 3.5.

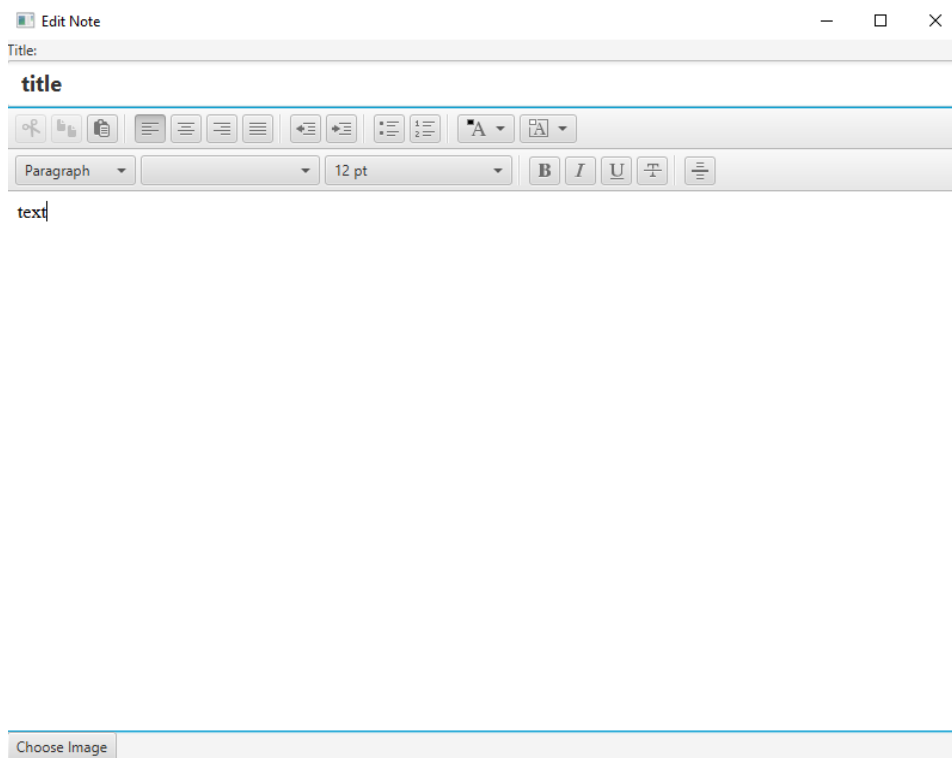


Рисунок 3.5 – Вікно для перегляду та редагування нотаток

Також, звичайно, при завантаженні цього вікна спрацьовує метод loadNotesFromDatabase та при здійсненні маніпуляцій з нотатками, окрім пошуку, це записується в базу методом saveNoteToDatabase та оновлюється методом updateNoteInDatabase.

3.4 Висновки за розділом 3

В цьому розділі було створено та показано схему алгоритму функціонування програми.

Також, було детально описано усі структурні класи, їх методи та призначення для програми. Використання бази даних PostgreSQL з високою надійністю та хешування паролів дозволяють зберігати облікові дані користувачів з великим рівнем захисту.

Продемонстровано вигляд вікон програми, детально описано їх складові та які методи викликаються при взаємодії з їх елементами. Використання мови програмування Java в поєднанні з фреймворком JavaFX дозволили легко та зручно створювати компактні та практичні вікна інтерфейсу.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ

4.1 Експлуатація програми

Для експлуатації розробленого ПЗ необхідно зробити декілька підготовчих кроків.

Необхідно зайти на офіційний сайт oracle [13], зверху обрати категорію products, знизу в категорії Hardware and Software обрати Java, після чого натиснути на кнопку Download Java у правому верхньому куті.

Далі буде запропоновано версії JDK які ще підтримуються компанією, для того щоб запустити розроблене ПЗ необхідна версія JDK 21 і вище, тому необхідно завантажити обрану версію для ОС Windows на якій буде проводитися експлуатація, розробка проводилася на Windows 10, тестування на інших версіях не проводилося. Якщо на пристрої вже встановлена Java, але це версія нижче, тоді потрібно встановити необхідну версію та додати її до змінних середовища:

- в пошуку операційної системи написати «змінні середовища»;
- обрати параметр «Змінні середовища»;
- в системних змінних обрати пункт «Path» та натиснути змінити;
- натиснути кнопку створити та натиснути кнопку «огляд»;
- обрати папку bin в завантаженому JDK;
- перемістити додану змінну нагору, як це показано на рис. 4.1.

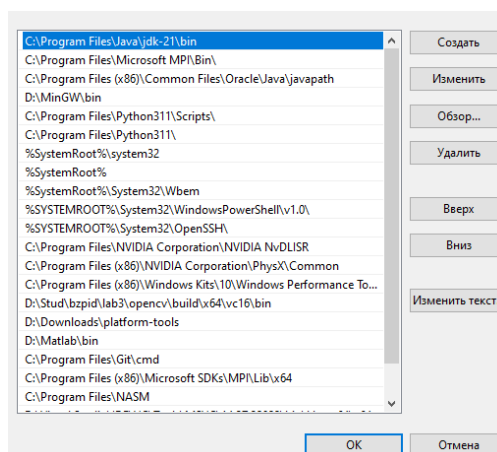


Рисунок 4.1 – Додана змінна з найвищим пріорітетом

Після цих підготовчих дій застосунок можна запускати для експлуатації. Після запуску перед користувачем з'явиться головне вікно програми, якщо в нього ще немає акаунту – його необхідно створити, натиснувши кнопку Sign Up, після чого перед користувачем з'явиться вікно реєстрації, у поля для вводу необхідно ввести логін, за допомогою якого буде здійснюватися вхід в акаунт та двічі написати пароль. Логін та пароль можуть містити лише латинські символи, арабські цифри та спеціальні символи. Пароль користувача має бути довжиною мінімум вісім символів, логін і пароль мають бути максимум двадцять чотири символи. Заповнені дані для реєстрації можна побачити на рис 4.2.

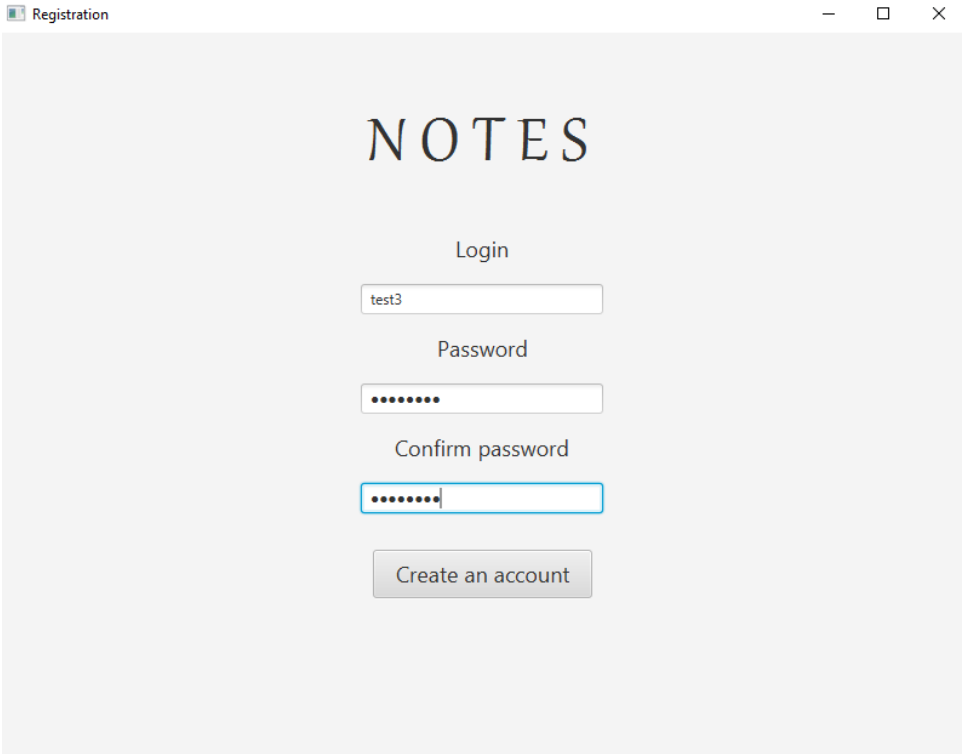


Рисунок 4.2 – Коректно заповнені дані для реєстрації

Після натискання на кнопку Create an account, якщо дані заповнені правильно перед користувачем з'явиться відповідне повідомлення, яке можна побачити на рис. 4.3 та вікно реєстрації закриється.

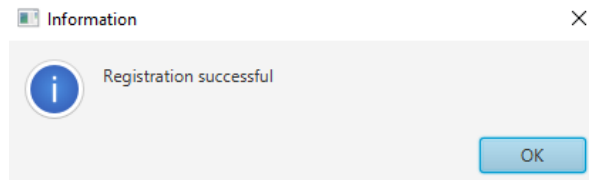


Рисунок 4.3 – Повідомлення про успішну реєстрацію

При наявності помилок з'явиться повідомлення з помилкою, приклади такого повідомлення на рис. 4.4.

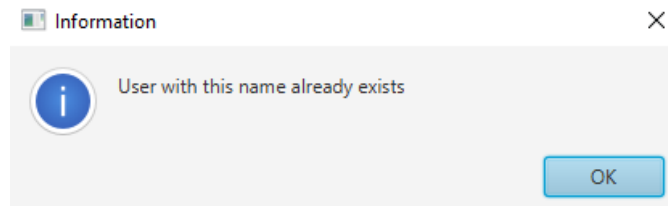


Рисунок 4.4 – Приклад помилки при реєстрації

Далі користувачу необхідно зайти у свій акаунт, за допомогою головного вікна Login, приклад введених даних у цьому вікні на рис. 4.5.

A login window titled 'Login' with standard window controls (minimize, maximize, close) in the top right corner. The main content area has a light gray background and the word 'NOTES' in a large, blue, serif font at the top. Below the title, the word 'Login' is centered. There are two input fields: the first is labeled 'test3' and the second is labeled 'Password' with a masked password '.....'. Below the password field is a checked checkbox labeled 'Remember Me'. At the bottom, there are two buttons: 'Log In' and 'Sign Up'. To the left of the 'Sign Up' button is the text 'No account yet?'. The window has a title bar with the text 'Login' and standard window controls.

Рисунок 4.5 – Коректно заповнені дані для входу в акаунт

Для успішного входу в акаунт користувачу необхідно вказати правильні дані, які він вказав при реєстрації, опціонально він може натиснути на чекбокс Remember Me, для того щоб при повторних запусках програми вхід в акаунт здійснювався автоматично, без повторного введення облікових даних. Якщо дані введено коректно і користувач попередньо зареєструвався вікно для входу закриється, а відкриється вікно з нотатками для конкретного користувача. При помилці входу для користувача буде показано повідомлення, приклад повідомлення можна побачити на рис. 4.6.

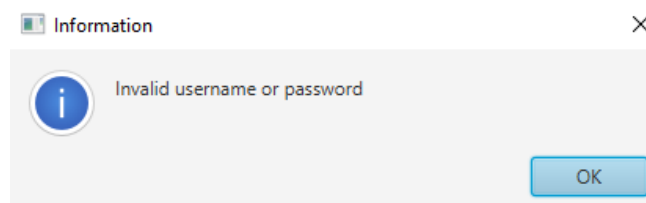


Рисунок 4.6 – Приклад помилки при вході в акаунт

Після входу в акаунт користувач може повноцінно працювати з застосунком. Він може додавати нотатки, при цьому змінюючи атрибути тексту та додаючи картинки, опціонально можна додати заголовок нотатки, який буде відображено в списку нотаток (в таблиці), якщо заголовок не додати буде відображено перші 5 слів з тексту нотатки. Додавання нотатки відбувається за допомогою натискання на кнопку Note, для додавання картинки необхідно натиснути на кнопку Choose Image. Для зміни атрибутів необхідно обирати потрібні функції в панелі над полем для введення тексту. Приклади створених нотаток в таблиці можна побачити на рис. 4.7.

Для перегляду вмісту нотаток та їх змінення, необхідно натиснути на заголовок, після цього відкриється вікно для зміни тексту та перегляду вмісту, на рис. 4.8 показано огляд нотатки зі зміненими атрибутами та картинками.

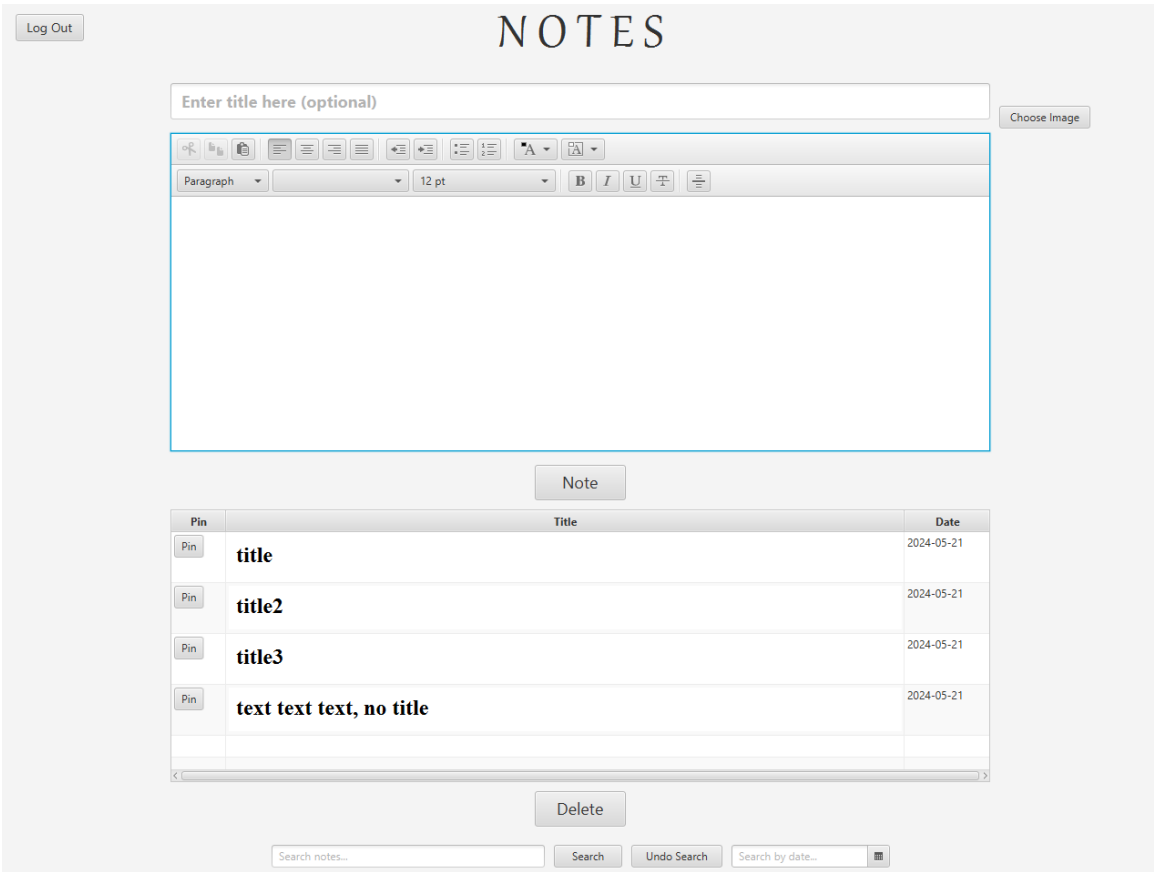


Рисунок 4.7 – Створені нотатки

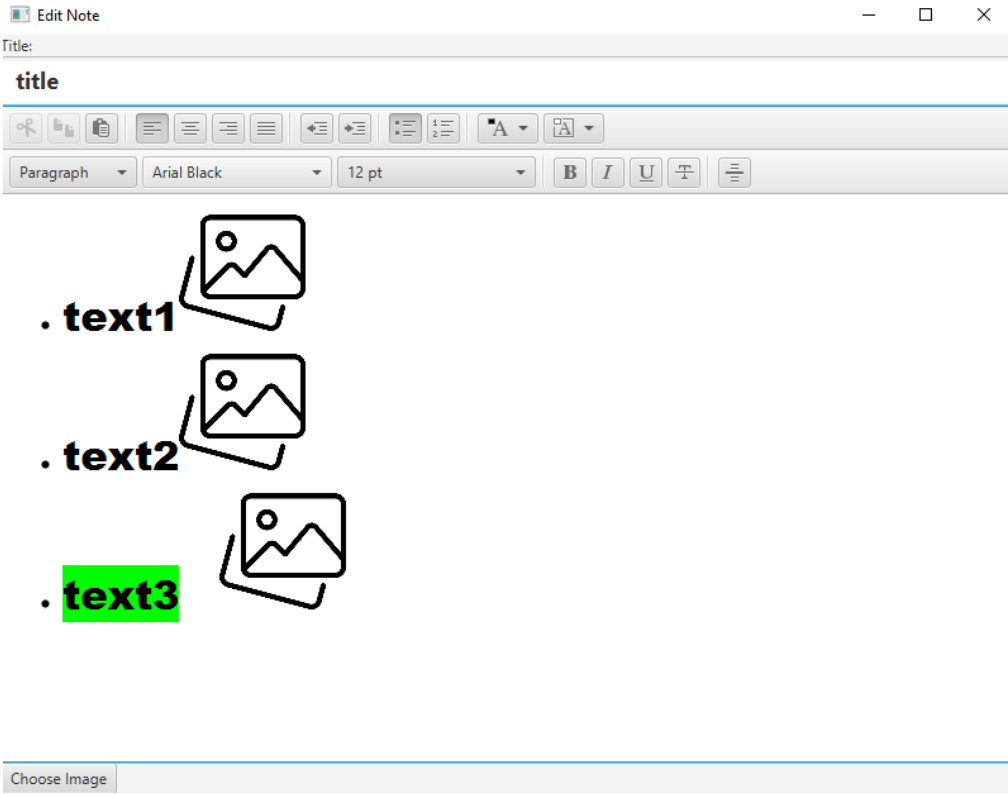


Рисунок 4.8 – Перегляд вмісту нотатки та її редагування

Користувач може закріплювати або відкріплювати нотатки за допомогою натискання клавіш pin, unpin відповідно, закріплені нотатки можна побачити на рис. 4.9. Закріплені нотатки поміщуються догори таблиці.

Також користувач може зробити сортування за заголовком або датою натиснувши на заголовок відповідно стовбця, одне натискання відсортує за зростанням, два натискання за спаданням, третє натискання поверне таблицю в початковий стан, сортування нотаток за заголовком можна побачити на рис. 4.10.

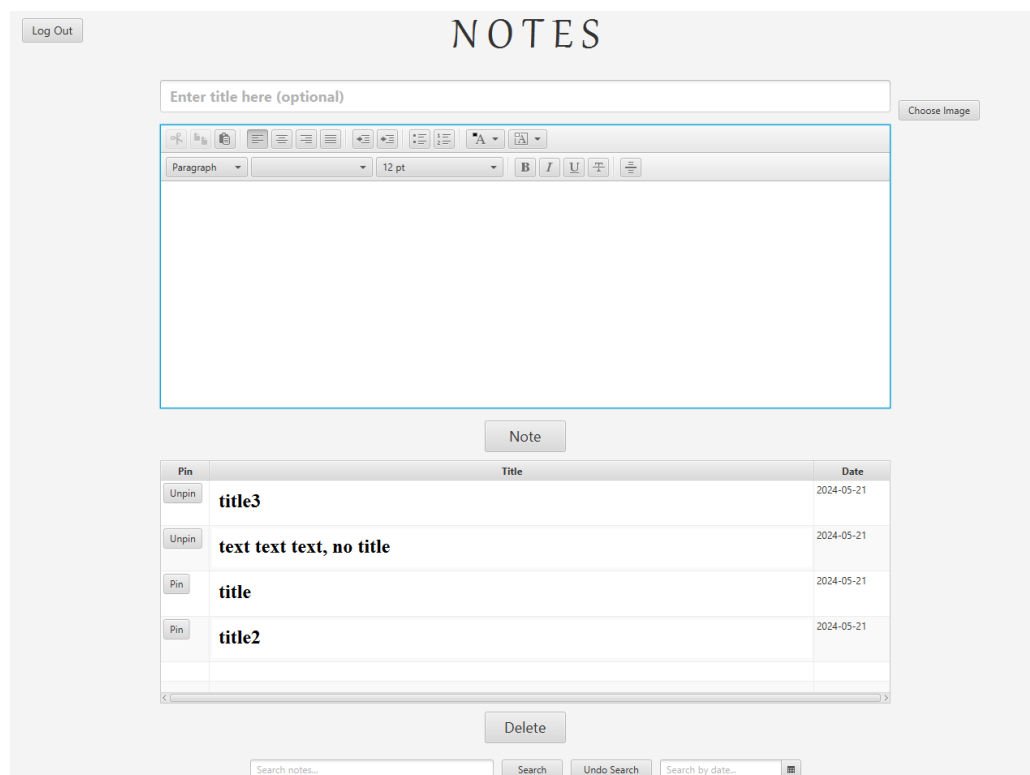


Рисунок 4.9 – Закріплення нотаток

Також, користувач може видаляти нотатки, для цього необхідно обрати нотатку для видалення та натиснути на кнопку Delete, після чого для користувача з'явиться повідомлення, яке можна побачити на рис. 4.11.

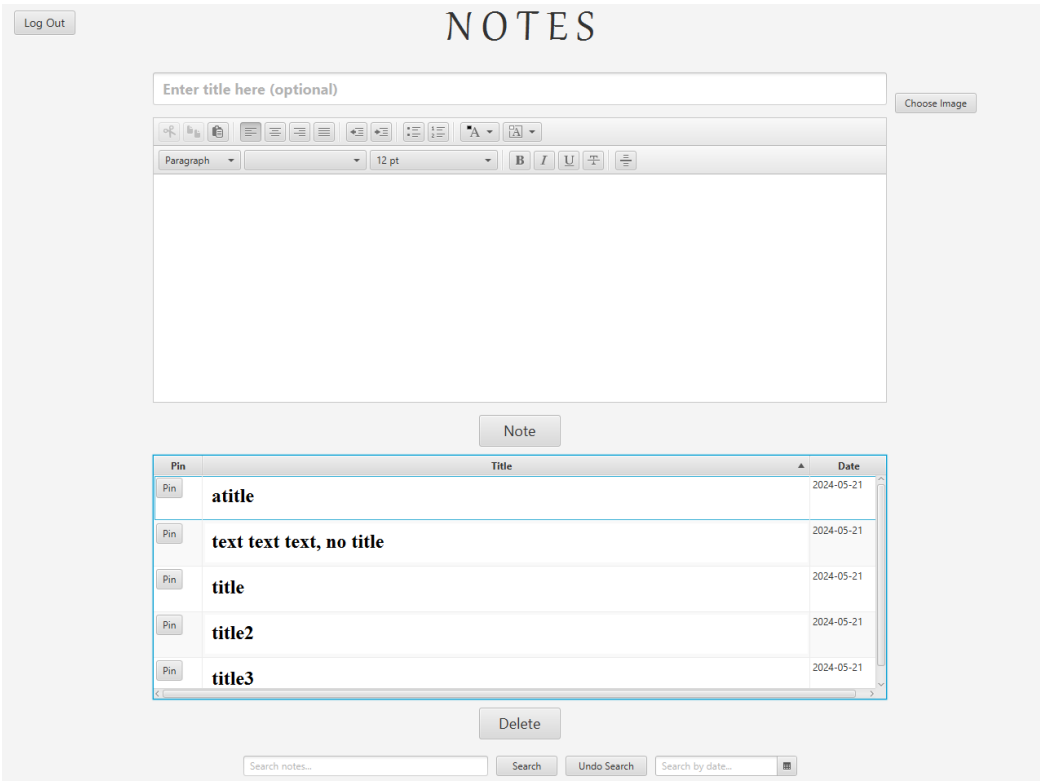


Рисунок 4.10 – Відсортовані нотатки за заголовком

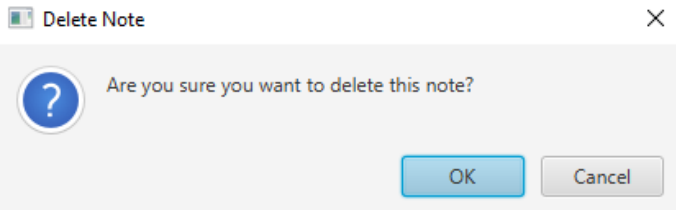


Рисунок 4.11 – Повідомлення про можливе видалення нотатки

При натисканні на ОК нотатку буде видалено зі списку нотаток (таблиці).

Користувач може здійснювати пошук написаних ним нотаток, для цього в поле для пошуку необхідно написати текст який потрібно шукати серед нотаток та натиснути кнопку Search, пошук здійснюється як за текстом нотатки так й за заголовком, приклад пошуку можна побачити на рис. 4.12.

Відображено 2 нотатки, одна містить у заголовку слово текст, інша містить слово text у самому тексті нотатки, якщо переглянути її.

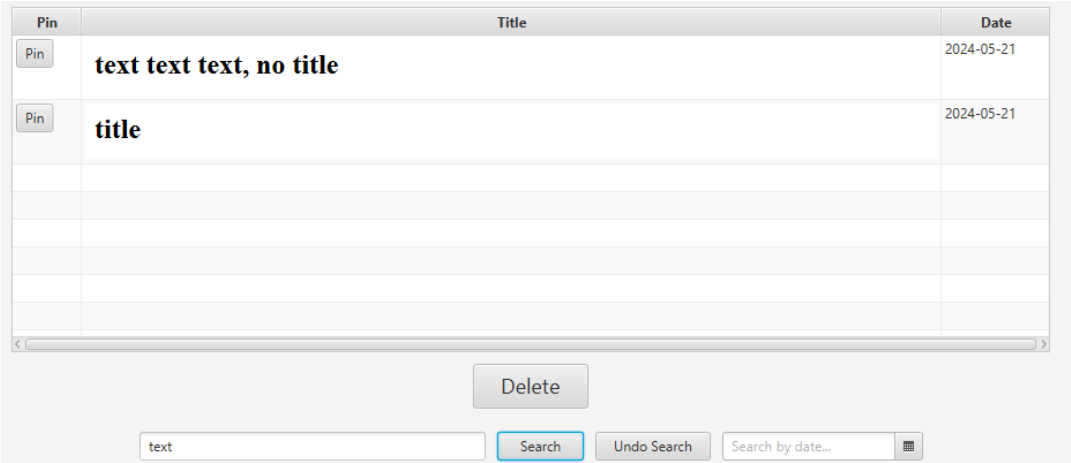


Рисунок 4.12 – Пошук за текстом

Результати пошуку за датою можна побачити на рис. 4.13.

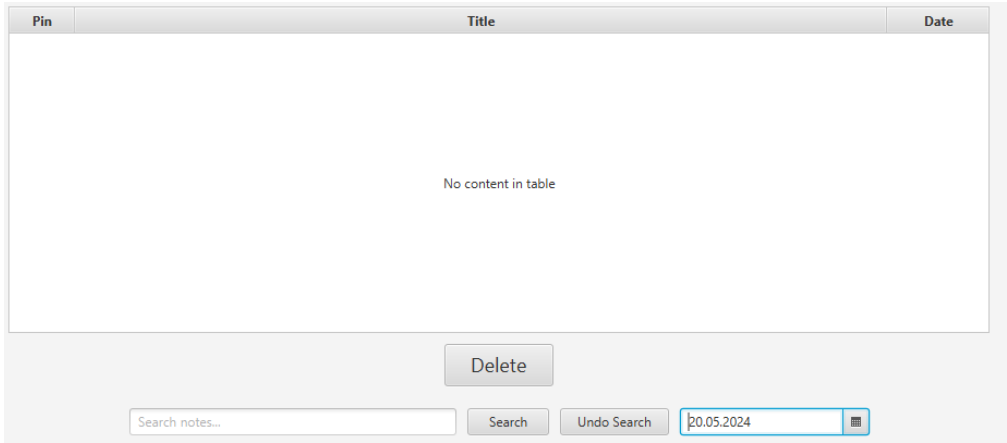


Рисунок 4.13 – Пошук за датою створення

Жодної нотатки не було створено 20.05.2024 тому в таблиці нічого не відображено.

Для скасування пошукового запиту необхідно натиснути на кнопку Undo Search, тоді в таблиці буде відображено усі нотатки.

І, наостанок, користувач може вийти зі свого акаунту натиснувши кнопку Log Out в лівому верхньому куті, після чого він потрапить до головного вікна, для входу в акаунт йому знову потрібно буде ввести дані.

4.2 Тестування програми

Відповідно до вимог до ПЗ, які вказані в технічному завданні та підпункті 2.2 розроблено перевірки, які можна побачити у чеклисті в табл. 4.1, усі перевірки пройдено успішно, що свідчить про коректність реалізації та роботи програми.

Таблиця 4.1 – Чеклист тестування програми

Назва тесту	Результат
1	2
Реєстрація з валідними даними	+
Реєстрація з невалідними даними (порожні поля, вже існуючий логін, не співпадіння пароля в двох полях для вводу, пароль < 8 символів, логін або пароль > 24 символів, не латиниця, цифри, спеціальні символи)	Виводяться відповідні повідомлення про некоректність введених даних
Занесення облікових даних до бази при успішній реєстрації	+
Вхід до акаунту з валідними даними	+
Вхід до акаунту з невалідними даними (порожні поля, неправильний логін або пароль)	Виводяться відповідні повідомлення про некоректність введених даних
Активація функції «запам'ятати мене»	+
Автоматичний вхід при активованій функції «запам'ятати мене»	+
Зміна атрибутів тексту нотатки (коректна зміна розміру, сімейства, вирівнювання, створення списків)	+
Додавання заголовку нотатки	+
Вибір картинки	+
Додавання картинки до тексту	+
Створення нотатки з картинкою та з текстом	+
Створення нотатки без картинки	+
Створення нотатки лише з картинкою	+
Створення порожньої нотатки	Виводиться повідомлення про незаповнене поле

Продовження таблиці 4.1

1	2
Закріплення/відкріплення нотаток	+
Видалення нотатки	+
Сортування нотаток за заголовком	+
Сортування нотаток за датою	+
Пошук нотаток за текстом, заголовком	+
Пошук нотаток за датою	+
Перегляд вмісту нотатки	+
Зміна тексту нотатки	+
Зміна заголовку нотатки	+
Скасування пошуку	+
Збереження інформації про нотатки в базу даних	+
Синхронізація нотаток при повторних входах в акаунт	+
Вихід з акаунту	+
Багатокористувацьке використання	+
Коректність відображення інтерфейсу при зміні розмірів вікна	+

4.3 Висновки за розділом 4

В цьому розділі описано процес експлуатації розробленого ПЗ, написано перелік дій які необхідно зробити для коректної роботи з ним, описано усі можливі дії під час експлуатації.

Створено чеклист перевірок розробленого ПЗ який показав що застосунок коректно реагує на усі коректні та некоректні введені дані, виконує призначений для нього функції, дозволяє працювати з ним багатьом користувачам, коректно відображає елементи інтерфейсу.

У підсумку розроблена програма є зручною для використання, не містить помилок та дозволяє виконувати багато дій з нотатками.

ВИСНОВКИ

В ході виконання дипломної роботи було проведено аналіз предметної області, визначено ключові поняття пов'язані з предметною областю, розібрано види нотаток, їх переваги та недоліки. Цифровий вид ведення нотаток розглянуто за допомогою конкретних існуючих програм: Microsoft OneNote, Evernote, Google Keep. Враховуючи функціонал десктопних застосунків їх було прийнято як орієнтир для розробки.

Розроблено структурну схему проекту, визначено функціональні вимоги до ПЗ, спроектовано схему взаємодії між користувачем та програмою. Розроблено алгоритм функціонування програми та його схему.

Розроблене ПЗ створено з використанням мови програмування Java та фреймворку JavaFX це дозволило зручно та якісно створити бекенд частину проекту та оформити зручний та швидко реагуючий інтерфейс. СКБД PostgreSQL перевагами якої визначено надійність та стійкість даних, допомогла зберігати, обробляти та завантажувати дані зареєстрованих користувачів та дані написаних ними нотаток.

Відносно розробленого ПЗ проведено експлуатацію та тестування які показали що програма коректно функціонує та дозволяє користувачам працювати з нотатками, виконуючи необхідні функції створення нотаток, додавання заголовків, зміни атрибутів шрифту, редагування тексту нотаток, додавання картинок, видалення нотаток, здійснення пошуку за текстом, заголовком та датою, закріплення та відкріплення нотаток, сортування за заголовком та датою. Розроблене ПЗ має потенціал для розвитку та підвищення кількісного фактору у функціоналі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Note-taking. Wikipedia [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Note-taking>.
2. Пам'ять. Вікіпедія [Електрон. Ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Пам%27ять>.
3. Астрід Тран. 5 найкращих методів ведення нотаток на роботі, оновлено у 2024 році [Електрон. ресурс] / Астрід Тран. – Режим доступу: <https://ahaslides.com/uk/blog/note-taking-methods>.
4. SQ3R. Вікіпедія [Електрон. Ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/SQ3R>.
5. Microsoft. Відомості про OneNote [Електрон. Ресурс]. – Режим доступу: <https://support.microsoft.com/uk-ua/office/відео-що-таке-onenote-be6cc6cc-3ca7-4f46-8876-5000f013c563>.
6. Microsoft OneNote. Wikipedia [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Microsoft_OneNote.
7. Google Keep. Wikipedia [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Google_Keep.
8. Evernote [Electronic resource]. – Access mode: <https://evernote.com>.
9. Андрій Козюля. Яку мову програмування краще вибрати для старту в IT? [Електрон. ресурс] / Андрій Козюля. – Режим доступу: <https://happymonday.ua/yaku-movu-programuvannya-vybraty-dlya-startu-v-it>.
10. Порівняння C++ з мовами Java і C# та майбутній розвиток мови [Електрон. ресурс]. – Режим доступу: <https://cpppm12.blogspot.com/2018/05/java-i-c.html>.
11. JavaFX [Electronic resource]. – Access mode: <https://openjfx.io>.
12. Digital Ocean. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems. [Electronic resource]. – Access mode: <https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems>.

13. Oracle [Electronic resource]. – Access mode: <https://www.oracle.com>.

ДОДАТОК А
Технічне завдання

A.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Розробка декстопного застосунку для створення нотаток» відповідно до наказу № 168 від 24 квітня 2024 року Національного університету «Запорізька політехніка».

A.2 Призначення розробки

Програма має дозволяти користувачам реєструватися та входити в акаунт для персоніфікованої роботи з нотатками, користувач зможе організовувати свої думки у вигляді цифрових нотаток зі зручним функціоналом всередині програми.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціоналу

Функціональні вимоги яким має відповідати програма:

- користувач повинен мати можливість зареєструватися;
- користувач повинен мати можливість увійти до застосунку після реєстрації;
- користувач повинен мати можливість поставити прапорець «запам'ятати мене» під час входу;
- користувач повинен мати можливість вийти зі свого акаунта;
- користувач повинен мати можливість створювати нотатки;
- користувач повинен мати можливість додавати заголовки нотатки;
- користувач повинен мати можливість змінювати атрибути тексту нотаток (змінення розміру шрифту, сімейства, кольору, оформлення у вигляді маркованих та нумерованих списків, вирівнювання);

- користувач повинен мати можливість бачити усі свої нотатки після створення і при повторних входах у свій акаунт;
- користувач повинен мати можливість додавати картинки до своїх нотаток;
- користувач повинен мати можливість обирати картинки які йому необхідно додати;
- користувач повинен мати можливість додати лише картинку в якості нотаток, без тексту;
- користувач повинен мати можливість бачити дату створення нотаток;
- користувач повинен мати можливість шукати нотатки за їх текстом, заголовком та за їх датою створення;
- користувач повинен мати можливість скасовувати пошук;
- користувач повинен мати можливість відсортувати нотатки за їх заголовком та за їх датою створення;
- користувач повинен мати можливість закріплювати та відкріплювати нотатки;
- користувач повинен мати можливість редагування тексту нотаток;
- користувач повинен мати можливість видалення нотатки;
- система повинна зберігати інформацію про усіх зареєстрованих користувачів;
- система повинна зберігати логін користувача якщо він натиснув прапорець «запам'ятати мене» під час входу;
- система повинна зберігати дані нотаток для конкретного користувача;
- система повинна завантажувати та відображати нотатки користувача для якого здійснено вхід;
- система повинна відображати результати пошуку;
- система повинна відображати повідомлення про помилки, якщо вони є.

А.3.2 Умови експлуатації

Під час роботи програми не бажано перезавантажувати ПК та від'єднувати живлення до нього. На комп'ютері повинно бути встановлено JDK версії 21 або новіше, також потрібне стабільне інтернет-з'єднання.

Експлуатація програми здійснюється відповідно до пояснення експлуатації відносно розробленої програми, яке міститься в розділі 4 пояснювальної записки, що містить інформацію необхідну для використання.

А.3.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програми, повинні виконуватися наступні умови до апаратної частини сервера:

- процесор AMD Ryzen 5 3600 і новіше;
- оперативна пам'ять 16 Гб і більше;
- постійна пам'ять 500 Гб і більше.

Вимоги для програмної частини клієнта:

- операційна система Windows 10;
- встановлена версія JDK-21 і новіше.

ДОДАТОК Б
Текст програми

Б.1 Текст файла User.java

```
public class User {  
    private int id;  
    private String username;  
    private String password;  
    public User() {}  
    public User(int id, String username, String password) {  
        this.id = id;  
        this.username = username;  
        this.password = password;  
    }  
    public User(String username, String password) {  
        this.username = username;  
        this.password = password;  
    }  
    public int getId() {  
        return id;  
    }  
    public void setId(int id) {  
        this.id = id;  
    }  
    public String getUsername() {  
        return username;  
    }  
    public void setUsername(String username) {  
        this.username = username;  
    }  
    public String getPassword() {  
        return password;  
    }  
    public void setPassword(String password) {  
        this.password = password;  
    }  
}
```

Б.2 Текст файла Note.java

```
import java.time.LocalDate;

public class Note {
    private int id;
    private String text;
    private LocalDate date;
    private boolean pinned;
    private int originalIndex;
    private String title;
    public Note () {};
    public Note(int id, String text, LocalDate date, boolean pinned, int originalIndex,
String title) {
        this.id = id;
        this.text = text;
        this.date = date;
        this.pinned = pinned;
        this.originalIndex = originalIndex;
        this.title = title;
    }
    public Note(String text, LocalDate date, boolean pinned, int originalIndex, String
title) {
        this(0, text, date, pinned, originalIndex, title);
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
```

```

    }
    public String getText() {
        return text;
    }
    public void setText(String text) {
        this.text = text;
    }
    public LocalDate getDate() {
        return date;
    }
    public void setDate(LocalDate date) {
        this.date = date;
    }
    public boolean isPinned() {
        return pinned;
    }
    public void setPinned(boolean pinned) {
        this.pinned = pinned;
    }
    public int getOriginalIndex() {
        return originalIndex;
    }
    public void setOriginalIndex(int originalIndex) {
        this.originalIndex = originalIndex;
    }
}

```

Б.3 Текст файла Controller.java

```

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;

```

```

import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.VBox;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.scene.web.HTMLEditor;
import java.io.File;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.*;
import java.time.LocalDate;
import java.util.*;

```

```

public class Controller {
    @FXML
    private TextField titleField;
    @FXML
    private HTMLEditor noteEditor;
    @FXML
    private TextField usernameField;
    private String currentUsername;
    @FXML
    private PasswordField passwordField;
    @FXML
    private PasswordField confirmPasswordField;
    @FXML
    private TableView<Note> notesTableView;
    private ObservableList<Note> notesList;
    @FXML

```

```

private CheckBox rememberMeCheckBox;
@FXML
private TextField searchField;
@FXML
private DatePicker datePicker;
@FXML
private void searchNotes() {
    String searchText = searchField.getText().trim();
    LocalDate selectedDate = datePicker.getValue();
    if (!searchText.isEmpty() || selectedDate != null) {
        ObservableList<Note> filteredNotes = FXCollections.observableArrayList();
        for (Note note : notesList) {
            String plainText = extractPlainText(note.getText());
            String title = note.getTitle();
            boolean matchesText = searchText.isEmpty() ||
plainText.contains(searchText) || title.contains(searchText);
            boolean matchesDate = selectedDate == null ||
note.getDate().equals(selectedDate);
            if (matchesText && matchesDate) {
                filteredNotes.add(note);
            }
        }
        notesTableView.setItems(filteredNotes);
    } else {
        notesTableView.setItems(notesList);
        notesTableView.refresh();
    }
}
@FXML
private void undoSearch() {
    notesTableView.setItems(notesList);
    searchField.clear();
    datePicker.setValue(null);
    notesTableView.refresh();
}

```

```

private void setupPinColumn() {
    TableColumn<Note, Void> pinColumn = new TableColumn<>("Pin");
    pinColumn.setPrefWidth(60);
    pinColumn.setCellFactory(col -> new TableCell<Note, Void>() {
        private final Button pinButton = new Button();
        {
            pinButton.setOnAction(event -> {
                Note note = getTableView().getItems().get(getIndex());
                notesList.remove(note);
                note.setPinned(!note.isPinned());
                if (note.isPinned()) {
                    notesList.add(0, note);
                } else {
                    int index = note.getOriginalIndex() >= notesList.size() ? notesList.size()
: note.getOriginalIndex();
                    notesList.add(index, note);
                }
                try {
                    updateNoteInDatabase(note, note.getId());
                } catch (SQLException e) {
                    e.printStackTrace();
                    showAlert("Error when updating the note status: " + e.getMessage());
                }
                sortAndUpdateView();
            });
        }
    });
    @Override
    protected void updateItem(Void item, boolean empty) {
        super.updateItem(item, empty);
        if (empty) {
            setGraphic(null);
        } else {
            Note note = getTableView().getItems().get(getIndex());
            pinButton.setText(note.isPinned() ? "Unpin" : "Pin");
            setGraphic(pinButton);}}});

```

```

        notesTableView.getColumns().add(0, pinColumn);
    }
    private void sortAndUpdateView() {
        Collections.sort(notesList, (n1, n2) -> {
            if (n1.isPinned() && !n2.isPinned()) {
                return -1;
            } else if (!n1.isPinned() && n2.isPinned()) {
                return 1;
            } else {
                return Integer.compare(n1.getOriginalIndex(), n2.getOriginalIndex());
            }
        });
        notesTableView.refresh();
    }
    public void initialize(String username) {
        this.currentUsername = username;
        notesList = FXCollections.observableArrayList();
        notesTableView.setItems(notesList);
        notesTableView.getColumns().clear();
        setupPinColumn();
        TableColumn<Note, String> textColumn = new TableColumn<>("Title");
        textColumn.setCellValueFactory(new PropertyValueFactory<>("title"));
        textColumn.setPrefWidth(745);
        textColumn.setCellFactory(col -> new TableCell<>() {
            private final WebView webView = new WebView();
            {
                webView.setPrefWidth(700);
                webView.setPrefHeight(50);
                webView.getEngine().documentProperty().addListener((obs,
newDoc) -> {
                    if (newDoc != null) {
                        webView.getEngine().executeScript(
                            "var style = document.createElement('style');"
                            + "style.type = 'text/css';"
                            + "style.innerHTML = 'body { overflow: hidden; }';"

```

```

        + "document.head.appendChild(style);"}));
    setGraphic(webView);
}
@Override
protected void updateItem(String title, boolean empty) {
    super.updateItem(title, empty);
    if (empty || title == null) {
        webView.getEngine().loadContent("");
        setGraphic(null);
    } else {
        Note note = getTableView().getItems().get(getIndex());
        String displayContent = "<h2>" + title + "</h2>";
        webView.getEngine().loadContent(displayContent);
        setGraphic(webView);
        webView.setOnMouseClicked(event -> {
            HTMLEditor editorForWindow = new HTMLEditor();
            editorForWindow.setHtmlText(note.getText());
            Button chooseImageButton = new Button("Choose Image");
            chooseImageButton.setOnAction(e -> chooseImage(editorForWindow));
            TextField titleField = new TextField(note.getTitle());
            titleField.setStyle("-fx-font-size: 1.5em; -fx-font-weight: bold; -fx-text-
fill: #333;");

            VBox vbox = new VBox(new Label("Title:"), titleField,
editorForWindow, chooseImageButton);

            Scene scene = new Scene(vbox, 800, 600);
            Stage editorStage = new Stage();
            editorStage.setScene(scene);
            editorStage.setTitle("Edit Note");
            editorStage.show();
            editorStage.setOnHiding(e -> {
                String updatedTitle = titleField.getText().trim();
                if (updatedTitle.isEmpty()) {
                    updatedTitle = extractTitle(editorForWindow.getHtmlText());
                }
                note.setTitle(updatedTitle);
            });
        });
    }
}

```

```

        note.setText(editorForWindow.getHtmlText());
        try {
            updateNoteInDatabase(note, note.getId());
        } catch (SQLException ex) {
            ex.printStackTrace();
            showAlert("Error updating note in the database: " +
ex.getMessage());}

        notesTableView.refresh();});});});});});});
notesTableView.getColumns().add(textColumn);
TableColumn<Note, LocalDate> dateColumn = new TableColumn<>("Date");
dateColumn.setCellValueFactory(new PropertyValueFactory<>("date"));
notesTableView.getColumns().add(dateColumn);
for (TableColumn<?, ?> column : notesTableView.getColumns()) {
    column.sortTypeProperty().addListener((observable, oldValue, newValue) -> {
        notesTableView.refresh();
    });
}
try {
    int userId = getCurrentUserId();
    loadNotesFromDatabase(userId);
} catch (SQLException e) {
    e.printStackTrace();
    showAlert("Error loading notes: " + e.getMessage());
}
}

private String extractTitle(String text) {
    String plainText = text.replaceAll("<[^\>]*\>", "").trim();
    String[] words = plainText.split("\\s+");
    int wordCount = Math.min(words.length, 5);
    return String.join(" ", Arrays.copyOfRange(words, 0, wordCount));
}

public void openRegistrationWindow() throws IOException {
    Stage register = new Stage();
    FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("Registration.fxml"));
    FXXMLLoader(Main.class.getResource("Registration.fxml"));

```

```

        Scene scene = new Scene(fxmlLoader.load(), 800, 600);
        register.setTitle("Registration");
        register.setScene(scene);
        register.show();}

    public void openNotesWindow(Stage stageToClose) throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(Main.class.getResource("Notes.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 1280, 960);
        Stage notesStage = new Stage();
        notesStage.setTitle("Notes");
        notesStage.setScene(scene);
        Controller controller = fxmlLoader.getController();
        controller.currentUsername = this.currentUsername;
        controller.initialize(this.currentUsername);
        notesStage.show();
        if (stageToClose != null) {
            stageToClose.close();
        }
    }

    @FXML
    private void chooseImage(ActionEvent event) {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Choose Image");
        fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter("Image Files", "*.png", "*.jpg", "*.gif")
        );
        File selectedFile = fileChooser.showOpenDialog(new Stage());
        if (selectedFile != null) {
            String imagePath = selectedFile.toURI().toString();
            String imageHtml = "<img src='" + imagePath + "' width='100' height='100' />";
            insertImageAtCursor(noteEditor, imageHtml);
        }
    }
}

```

@FXML

```
private void chooseImage(HTMLoader editor) {
    FileChooser fileChooser = new FileChooser();
    fileChooser.setTitle("Choose Image");
    fileChooser.getExtensionFilters().addAll(
        new FileChooser.ExtensionFilter("Image Files", "*.png", "*.jpg", "*.gif")
    );
    File selectedFile = fileChooser.showOpenDialog(new Stage());
    if (selectedFile != null) {
        String imagePath = selectedFile.toURI().toString();
        String imageHtml = "<img src=\"" + imagePath + "\" width='100' height='100' />";
        insertImageAtCursor(editor, imageHtml);
    }
}

private void insertImageAtCursor(HTMLoader editor, String html) {
    WebView webView = (WebView) editor.lookup(".web-view");
    WebEngine webEngine = webView.getEngine();
    String escapedHtml = html.replace(""", "\\"");
    Platform.runLater(() -> {
        webEngine.executeScript(
            "document.execCommand('insertHTML', false, \"" + escapedHtml + "\");"
        );
    });
}

private boolean isEmpty(String text) {
    String textContent = text.replaceAll("<(?!img\\b)[^>]*>", "").trim();
    return textContent.isEmpty();
}
```

@FXML

```
private void addNote() {
    String title = titleField.getText().trim();
    String noteText = noteEditor.getHtmlText().trim();
    if (isEmpty(noteText)) {
        showAlert("Note content cannot be empty.");
        return;
    }
}
```

```

        if (title.isEmpty()) {
            title = extractTitle(noteText);
        }
        Note newNote = new Note();
        newNote.setOriginalIndex(notesList.size());
        newNote.setText(noteText); // сохраняем только текст заметки
        newNote.setTitle(title); // заголовок сохраняется отдельно
        newNote.setDate(LocalDate.now());
        notesList.add(newNote);
        titleField.setText("");
        noteEditor.setHtmlText("");
        try {
            int userId = getCurrentUserId();
            saveNoteToDatabase(newNote, userId);
        } catch (SQLException e) {
            e.printStackTrace();
            showAlert("Error saving note: " + e.getMessage());
        }
        notesTableView.refresh();
    }

    private List<Note> loadNotesFromDatabase(int userId) throws SQLException {
        String query = "SELECT id, text, date, pinned, original_index, title FROM notes
WHERE user_id = ?";

        List<Note> notes = new ArrayList<>();
        try (Connection conn = getDatabaseConnection();
            PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setInt(1, userId);
            try (ResultSet rs = stmt.executeQuery()) {
                while (rs.next()) {
                    Note note = new Note(
                        rs.getInt("id"),
                        rs.getString("text"),
                        rs.getDate("date").toLocalDate(),
                        rs.getBoolean("pinned"),
                        rs.getInt("original_index"),

```

```

        rs.getString("title"));
        notes.add(note);}}}
notes.sort((n1, n2) -> {
    if (n1.isPinned() && !n2.isPinned()) return -1;
    if (!n1.isPinned() && n2.isPinned()) return 1;
    return Integer.compare(n1.getOriginalIndex(), n2.getOriginalIndex());
});
notesList.setAll(notes);
return notes;
}
@FXML
private void saveNoteToDatabase(Note note, int userId) throws SQLException {
    String query = "INSERT INTO notes (user_id, text, date, pinned, original_index,
title) VALUES (?, ?, ?, ?, ?, ?)";
    try (Connection conn = getDatabaseConnection();
        PreparedStatement stmt = conn.prepareStatement(query,
Statement.RETURN_GENERATED_KEYS)) {
        stmt.setInt(1, userId);
        stmt.setString(2, note.getText());
        stmt.setDate(3, java.sql.Date.valueOf(note.getDate()));
        stmt.setBoolean(4, note.isPinned());
        stmt.setInt(5, note.getOriginalIndex());
        stmt.setString(6, note.getTitle());
        stmt.executeUpdate();
        try (ResultSet generatedKeys = stmt.getGeneratedKeys()) {
            if (generatedKeys.next()) {
                note.setId(generatedKeys.getInt(1));}}}
    private void updateNoteInDatabase(Note note, int noteId) throws SQLException {
        String query = "UPDATE notes SET text = ?, date = ?, pinned = ?, original_index
= ?, title = ? WHERE id = ?";
        try (Connection conn = getDatabaseConnection();
            PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, note.getText());
            stmt.setDate(2, java.sql.Date.valueOf(note.getDate()));
            stmt.setBoolean(3, note.isPinned());

```

```

        stmt.setInt(4, note.getOriginalIndex());
        stmt.setString(5, note.getTitle());
        stmt.setInt(6, noteId);
        stmt.executeUpdate();
    }
}

@FXML
private void deleteNote() {
    Note selectedNote = notesTableView.getSelectionModel().getSelectedItem();
    if (selectedNote != null) {
        Alert confirmAlert = new Alert(Alert.AlertType.CONFIRMATION);
        confirmAlert.setTitle("Delete Note");
        confirmAlert.setHeaderText(null);
        confirmAlert.setContentText("Are you sure you want to delete this note?");
        Optional<ButtonType> result = confirmAlert.showAndWait();
        if (result.isPresent() && result.get() == ButtonType.OK) {
            try {
                deleteNoteFromDatabase(selectedNote.getId());
                notesList.remove(selectedNote);
                notesTableView.refresh();
            } catch (SQLException e) {
                e.printStackTrace();
                showAlert("Error deleting note from the database: " + e.getMessage());
            }
        }
    } else {
        showAlert("No note selected for deletion");
    }
}

private void deleteNoteFromDatabase(int noteId) throws SQLException {
    String query = "DELETE FROM notes WHERE id = ?";
    try (Connection conn = getDatabaseConnection();
        PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setInt(1, noteId);
        stmt.executeUpdate();
    }
}

```

@FXML

```

public void registerUser() throws SQLException {
    String username = usernameField.getText();
    String password = passwordField.getText();
    String confirmPassword = confirmPasswordField.getText();
    if (username.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()) {
        showAlert("All fields must be filled");
        return;
    }
    if (username.length() > 24 || password.length() > 24) {
        showAlert("Username and password must be at most 24 characters long");
        return;
    }
    if (password.length() < 8) {
        showAlert("Password must be at least 8 characters long");
        return;
    }
    if (!isValidString(username) || !isValidString(password)) {
        showAlert("Username and password must contain only English letters, digits,
and special characters");
        return;
    }
    if (!password.equals(confirmPassword)) {
        showAlert("Passwords do not match");
        return;
    }
    if (isUsernameTaken(username)) {
        showAlert("User with this name already exists");
        return;
    }
    registerUser(username, password);
    showAlert("Registration successful");
    Stage stage = (Stage) usernameField.getScene().getWindow();
    stage.close();
}

```

```

private boolean isValidString(String str) {
    return str.matches("^[a-zA-Z0-9!@#%&*?+-]+$");
}

private void showAlert(String message) {
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("Information");
    alert.setHeaderText(null);
    alert.setContentText(message);
    alert.showAndWait();
}

private void registerUser(String username, String password) throws SQLException {
    String hashedPassword = hashPassword(password);
    String query = "INSERT INTO users (username, password) VALUES (?, ?)";
    try (Connection conn = getDatabaseConnection();
        PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setString(1, username);
        stmt.setString(2, hashedPassword);
        stmt.executeUpdate();
    }
}

private boolean isUsernameTaken(String username) throws SQLException {
    String query = "SELECT COUNT(*) FROM users WHERE username = ?";
    try (Connection conn = getDatabaseConnection();
        PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setString(1, username);
        try (ResultSet rs = stmt.executeQuery()) {
            if (rs.next()) {
                return rs.getInt(1) > 0;
            }
        }
    }
    return false;
}

private User authenticateUser(String username, String hashedPassword) throws
SQLException {

```

```
String query = "SELECT id, username FROM users WHERE username = ? AND
password = ?";
```

```
try (Connection conn = getDatabaseConnection();
    PreparedStatement stmt = conn.prepareStatement(query)) {
    stmt.setString(1, username);
    stmt.setString(2, hashedPassword);
    try (ResultSet rs = stmt.executeQuery()) {
        if (rs.next()) {
            return new User(rs.getString("username"), hashedPassword);
        }
    }
}
return null;
}
```

@FXML

```
public void loginValidate(ActionEvent event) throws IOException, SQLException {
    String username = usernameField.getText();
    String password = passwordField.getText();
    if (username.isEmpty() || password.isEmpty()) {
        showAlert("All fields must be filled");
        return;
    }
    if (!isValidString(username) || !isValidString(password)) {
        showAlert("Username and password must contain only English letters, digits,
and special characters");
        return;
    }
    String hashedPassword = hashPassword(password);
    User authenticatedUser = authenticateUser(username, hashedPassword);
    if (authenticatedUser == null) {
        showAlert("Invalid username or password");
        return;
    }
    boolean rememberMe = rememberMeCheckBox.isSelected();
    updateLoginState(username, rememberMe);
    currentUsername = authenticatedUser.getUsername();
    openNotesWindow(((Stage) ((Node) event.getSource()).getScene().getWindow()));}
```

```

private String hashPassword(String password) {
    try {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest(password.getBytes(StandardCharsets.UTF_8));
        StringBuilder hexString = new StringBuilder();
        for (byte b : hash) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) hexString.append('0');
            hexString.append(hex);
        }
        return hexString.toString();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        showAlert("Error in hashing");
        return null;
    }
}

public boolean checkLoginState(Stage primaryStage) throws SQLException,
IOException {
    String query = "SELECT id, username, password FROM users WHERE
remember_me = true LIMIT 1";
    try (Connection conn = getDatabaseConnection();
        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet rs = stmt.executeQuery()) {
        if (rs.next()) {
            String username = rs.getString("username");
            String hashedPassword = rs.getString("password");
            User user = authenticateUser(username, hashedPassword);
            if (user != null) {
                currentUsername = username;
                System.out.println("Auto-login successful for user: " + username);
                openNotesWindow(primaryStage);
                return true;
            } else {
                System.out.println("Auto-login failed for user: " + username);
            }
        }
    }
}

```

```

        }
    }
}
return false;
}
@FXML
private void handleLogout(ActionEvent event) throws IOException, SQLException {
    updateLoginState(currentUsername, false);
    Stage notesStage = (Stage) ((Node) event.getSource()).getScene().getWindow();
    notesStage.close();
    openLoginWindow();
}

private void updateLoginState(String username, boolean rememberMe) throws
SQLException {
    String query = "UPDATE users SET remember_me = ? WHERE username = ?";
    try (Connection conn = getDatabaseConnection();
        PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setBoolean(1, rememberMe);
        stmt.setString(2, username);
        stmt.executeUpdate();
    }
}

public void openLoginWindow() throws IOException {
    Stage loginStage = new Stage();
    FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("Login.fxml"));
    Scene scene = new Scene(fxmlLoader.load(), 800, 600);
    loginStage.setTitle("Login");
    loginStage.setScene(scene);
    loginStage.show();
}

private Connection getDatabaseConnection() throws SQLException {
    String url = "jdbc:postgresql://localhost/notes";
    String username = "postgres";
    String password = "Prk270830";

```

```

        return DriverManager.getConnection(url, username, password);
    }
    private int getCurrentUserId() throws SQLException {
        String query = "SELECT id FROM users WHERE username = ?";
        try (Connection conn = getDatabaseConnection();
            PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setString(1, currentUsername);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    return rs.getInt("id");
                }
            }
        }
        return -1;
    }
}

```

Б.4 Текст файла Main.java

```

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.stage.Stage;
import java.io.IOException;
import java.sql.SQLException;
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) throws IOException, SQLException {
        FXMLLoader fxmlLoader = new
FXMLLoader(Main.class.getResource("Login.fxml"));
        Parent root = fxmlLoader.load();
        Controller controller = fxmlLoader.getController();
        if (controller.checkLoginState(primaryStage)) {
            return;
        }
        controller.openLoginWindow();
    }
    public static void main(String[] args) {
        launch();
    }
}

```

Б.5 Текст файлу Login.fxml

```

<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>

<VBox alignment="CENTER" prefHeight="600.0" prefWidth="800.0" spacing="20.0"
xmlns="http://javafx.com/javafx/17.0.2-ea"                xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.example.notes.Controller">
    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />
    </padding>
    <Label graphicTextGap="0.0" prefHeight="75.0" prefWidth="190.0" text="N O T E
S" translateY="-50.0">
        <font>
            <Font name="Gabriola" size="62.0" />
        </font>
    </Label>
    <VBox alignment="TOP_CENTER" prefHeight="123.0" prefWidth="760.0"
spacing="15.0">
        <children>
            <Label text="Login">
                <font>
                    <Font size="18.0" />
                </font>
            </Label>
            <TextField fx:id="usernameField" maxWidth="200.0" prefHeight="30.0"
prefWidth="200.0" />
            <Label text="Password">
                <font>
                    <Font size="18.0" />
                </font>
            </Label>

```

```

        <PasswordField fx:id="passwordField" maxWidth="200.0" prefHeight="30.0"
prefWidth="750.0" />
        <CheckBox fx:id="rememberMeCheckBox" text="Remember Me"
layoutX="100" layoutY="200" />
    </children></VBox>
    <VBox alignment="CENTER" prefHeight="140.0" prefWidth="760.0"
spacing="10.0">
        <children>
            <Button alignment="CENTER" focusTraversable="false"
mnemonicParsing="false" onAction="#loginValidate" prefHeight="40.0" prefWidth="100.0"
text="Log In">
                <font>
                    <Font size="18.0" />
                </font></Button>
            <Label text="No account yet?" translateX="-120.0" translateY="40.0">
                <font>
                    <Font name="System Bold" size="15.0" />
                </font>
            </Label>
            <Button alignment="CENTER" focusTraversable="false"
onAction="#openRegistrationWindow" prefHeight="40.0" prefWidth="100.0" text="Sign Up">
                <font>
                    <Font size="18.0" />
                </font></Button>
        </children>
    </VBox>
</VBox>

```

Б.6 Текст файла Registration.fxml

```

<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>

```

```

    <VBox alignment="CENTER" prefHeight="600.0" prefWidth="800.0" spacing="20.0"
xmlns="http://javafx.com/javafx/17.0.2-ea"                xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.example.notes.Controller">
    <padding>
        <Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />
    </padding>
    <Label graphicTextGap="0.0" prefHeight="75.0" prefWidth="190.0" text="N O T E
S" translateY="-20.0">
        <font>
            <Font name="Gabriola" size="62.0" />
        </font>
    </Label>
    <VBox alignment="TOP_CENTER" prefHeight="123.0" prefWidth="760.0"
spacing="15.0">
        <children>
            <Label text="Login">
                <font>
                    <Font size="18.0" />
                </font>
            </Label>
            <TextField fx:id="usernameField" maxWidth="200.0" prefHeight="30.0"
prefWidth="200.0" />
            <Label text="Password">
                <font>
                    <Font size="18.0" />
                </font>
            </Label>
            <PasswordField fx:id="passwordField" maxWidth="200.0" prefHeight="30.0"
prefWidth="750.0" />
            <Label text="Confirm password">
                <font>
                    <Font size="18.0" />
                </font>
            </Label>

```

```

        <PasswordField fx:id="confirmPasswordField" maxWidth="200.0" />
    </children></VBox>

    <VBox alignment="TOP_CENTER" prefHeight="116.0" prefWidth="760.0"
spacing="10.0">
        <children>
            <Button alignment="CENTER" focusTraversable="false"
mnemonicParsing="false" onAction="#registerUser" prefHeight="40.0" prefWidth="181.0"
text="Create an account" textAlignment="CENTER" translateY="10.0">
                <font>
                    <Font size="18.0" />
                </font></Button>
        </children>
    </VBox>
</VBox>

```

Б.7 Текст файла Notes.fxml

```

<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.control.cell.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.web.*?>

<VBox alignment="CENTER" prefHeight="960.0" prefWidth="1280.0"
spacing="20.0" xmlns="http://javafx.com/javafx/17.0.2-ea" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="com.example.notes.Controller">
    <HBox alignment="BASELINE_LEFT" spacing="20.0" translateY="70.0">
        <Button fx:id="logoutButton" focusTraversable="false" mnemonicParsing="false"
onAction="#handleLogout" prefHeight="35.0" prefWidth="75.0" text="Log Out">
            <font>
                <Font size="14.0" />
            </font>
        </Button>
    </HBox>

```

```

    <Label alignment="CENTER" contentDisplay="CENTER" graphicTextGap="0.0"
prefHeight="69.0" prefWidth="190.0" text="N O T E S" textAlignment="CENTER">
        <font>
            <Font name="Gabriola" size="62.0" />
        </font>
    </Label>
    <VBox alignment="CENTER" maxWidth="900" prefHeight="500.0"
prefWidth="900.0" spacing="15.0">
        <TextField fx:id="titleField" focusTraversable="false" minHeight="40.0"
promptText="Enter title here (optional)" style="-fx-font-size: 1.5em; -fx-font-weight: bold; -fx-
text-fill: #333;" />
        <HTMLEditor fx:id="noteEditor" maxWidth="900" minHeight="350"
prefHeight="350.0" prefWidth="900.0" />
    </VBox>
    <Button alignment="CENTER" mnemonicParsing="false"
onAction="#chooseImage" prefHeight="35.0" prefWidth="100.0" text="Choose Image"
translateX="510.0" translateY="-400.0" />
    <Button alignment="CENTER" mnemonicParsing="false" onAction="#addNote"
prefHeight="35.0" prefWidth="100.0" text="Note" textAlignment="CENTER" translateY="-
50.0">
        <font>
            <Font size="18.0" />
        </font>
    </Button>
    <HBox alignment="CENTER" minHeight="-Infinity" prefHeight="300.0"
translateY="-60.0">
        <TableView fx:id="notesTableView" editable="true" focusTraversable="false"
minHeight="-Infinity" prefHeight="300.0" prefWidth="900.0">
            <columns>
                <TableColumn prefWidth="600.0" text="Title">
                    <cellValueFactory>
                        <PropertyValueFactory property="title" />
                    </cellValueFactory>
                </TableColumn>
                <TableColumn prefWidth="200.0" text="Date">

```

```

        <cellValueFactory>
            <PropertyValueFactory property="date" />
        </cellValueFactory>
    </TableColumn>
</columns>
</TableView>
</HBox>
<HBox alignment="CENTER" spacing="10.0" translateY="-70.0">
    <Button mnemonicParsing="false" onAction="#deleteNote" prefHeight="35.0"
prefWidth="100.0" text="Delete">
        <font>
            <Font size="18.0" />
        </font>
    </Button>
</HBox>
<HBox alignment="CENTER" spacing="10.0" translateY="-70.0">
    <TextField fx:id="searchField" prefWidth="300.0" promptText="Search notes..."
/>

    <Button mnemonicParsing="false" onAction="#searchNotes" prefHeight="35.0"
prefWidth="75.0" text="Search" />
    <Button mnemonicParsing="false" onAction="#undoSearch" prefHeight="35.0"
prefWidth="100.0" text="Undo Search" />
    <DatePicker fx:id="datePicker" onAction="#searchNotes" promptText="Search by
date..." />
</HBox>
<padding>
    <Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />
</padding>
</VBox>

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра

Розробка десктопного застосунку для створення нотаток
Desktop Software Development for Note Taking

Виконав
студент групи КНТ-210

Олександр КОВЕРСУН

Керівник
к.т.н., доцент

Тетяна ФЕДОРОНЧАК

Рисунок В.1 – Вступний слайд

Об'єкт дослідження – програмні засоби для створення нотаток.

Предмет дослідження – десктопні застосунки для створення нотаток.

Мета роботи – розробка десктопного застосунку, що дозволяє користувачам вести роботу з нотатками.

Завдання роботи:

- аналіз предметної області;
- вибір і обґрунтування структури системи;
- основні рішення реалізації компонентів системи;
- експлуатація та тестування.

Рисунок В.2 – Слайд завдань, мети роботи та об'єкту, предмету дослідження

Порівняння програмних продуктів

Критерій порівняння	Microsoft OneNote	Google Keep	Evernote
Принципи організації	Блокноти, секції блокнотів, групи, секцій, теги	Теги, кольори	Блокноти, теги, стеки
Наявність синхронізації	Так	Так	Так
Наявність вебкліпінгу	Так	Лише посилання та заголовки вебсторінок	Так
Розпізнавання ручного вводу	Так	Ні	Так
Наявність пошуку	Так	Так	Так
Перевірка правопису	Так	Так	Так
Можливість друку	Так	Ні	Так
Можливість писати «чорнилами»	Так	Ні	Не можна поєднувати з текстом
Вартість	Безкоштовне, інтеграція з Office потребує підписки	Безкоштовне	Безкоштовне, з обмеженнями, які прибираються з платними тарифами

3

Рисунок В.3 – Слайд порівняння програмних продуктів

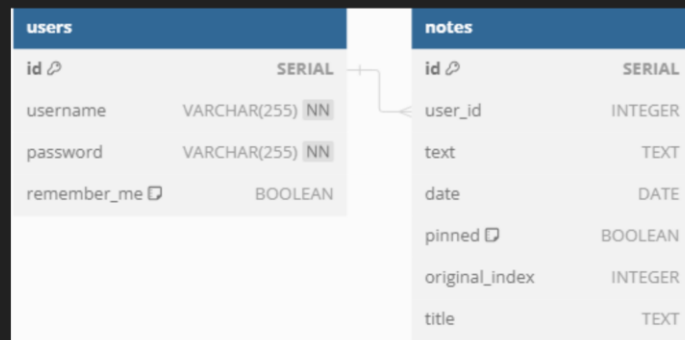
Порівняння мов програмування

Критерій порівняння	Java	C#	C++	Python
Швидкість виконання коду	Висока	Висока	Висока, потребує більше часу на розробку через ручне керування пам'яттю	Середня, через інтерпретованість
Зручність розробки	Висока	Висока	Середня	Висока
Garbage Collector	+	+	-	+
Підтримка Unicode	Так	Так	Так, вимагає використання окремих типів, напр. wstring	Так
Доступність бібліотек, фреймворків	Велика різноманітність	Велика різноманітність	Велика різноманітність, частина може бути на C та призвести до проблем	Велика різноманітність
Безпекова складова	Висока	Висока	Висока, потребує уважності через ручне керування та покажчики	Висока
Кросплатформність	Повна	Часткова	Часткова	Повна
Підтримка розробниками	Висока	Висока	Висока	Висока

4

Рисунок В.4 – Слайд порівняння мов програмування

Схема бази даних



7

Рисунок В.7 – Слайд схеми бази даних

Головне вікно програми та вікно реєстрації

NOTES

Login

Password

☐ Remember Me

Log In

No account yet? Sign Up

NOTES

Login

Password

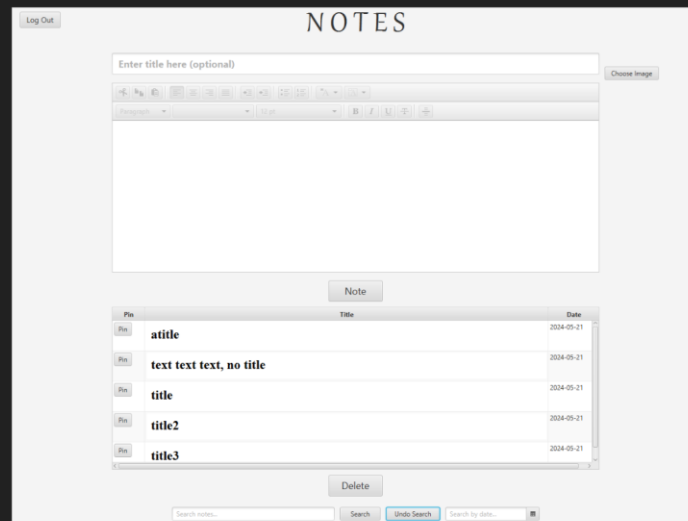
Confirm password

Create an account

8

Рисунок В.8 – Слайд головного вікна та вікна реєстрації

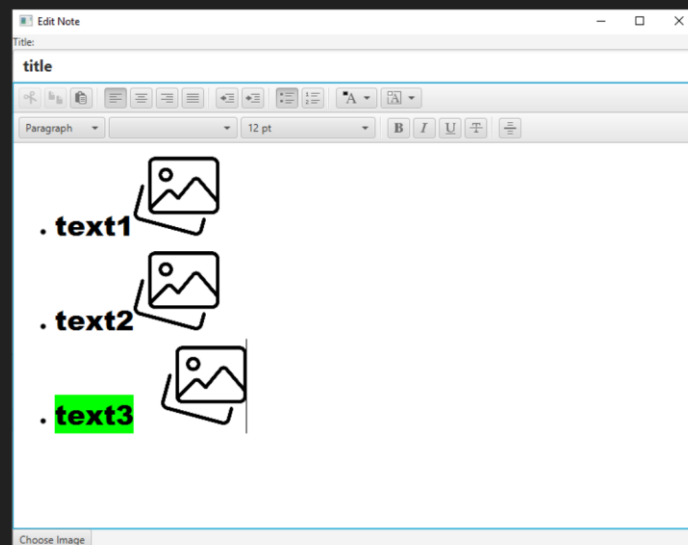
Вікно з нотатками



9

Рисунок В.9 – Слайд вікна з нотатками

Перегляд вмісту та редагування



10

Рисунок В.10 – Слайд з переглядом вмісту нотатки

Висновки

Розроблено структурну схему проєкту, визначено функціональні вимоги до ПЗ, спроектовано схему взаємодії між користувачем та програмою. Розроблено алгоритм функціонування програми та його схему.

Розроблене ПЗ створено з використанням мови програмування Java та фреймворку JavaFX це дозволило зручно та якісно створити бекенд частину проєкту та оформити зручний та швидко реагуючий інтерфейс. СКБД PostgreSQL перевагами якої визначено надійність та стійкість даних, допомогла зберігати, обробляти та завантажувати дані зареєстрованих користувачів та дані написаних ними нотаток.

Відносно розробленого ПЗ проведено експлуатацію та тестування які показали що програма коректно функціонує та дозволяє користувачам працювати з нотатками, виконуючи необхідні функції створення нотаток, додавання заголовків, зміни атрибутів шрифту, редагування тексту нотаток, додавання картинок, видалення нотаток, здійснення пошуку за текстом, заголовком та датою, закріплення та відкріплення нотаток, сортування за заголовком та датою. Розроблене ПЗ має потенціал для розвитку та підвищення кількісного фактору у функціоналі.

11

Рисунок В.11 – Слайд з висновками