Міністерство освіти і науки України

Луцький національний технічний університет

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ДОДАТКУ «MINDS KEEPER» ДЛЯ НОТАТОК НА IOS ЗА ДОПОМОГОЮ SWIFTUI

DEVELOPMENT OF "MINDS KEEPER" APPLICATION FOR NOTES ON IOS USING SWIFTUI

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-21
Кліменко Артем Вікторович
_____
(підпис)

Керівник:
к.т.н., доцент
Ящук Андрій Анатолійович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2024 р.
к.т.н., доцент
Гарант освітньої програми:
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2024 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Н. Лимина
«2» 02 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Кліменку Артему Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *розробка додатку «Minds Keeper» для нотаток на iOS за допомогою SwiftUI.*

Керівник роботи: *к.т.н., доцент, Яцук Андрій Анатолійович*
затверджені наказом закладу вищої освіти від «*30*» *грудня 2023 р.* № *477/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«5» червня 2024 р.*
3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.*

5. Перелік графічного матеріалу: *18 рисунків.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Яцук А. А.*		
Специфікації вимог до розробленої системи	*Яцук А. А.*		
Розробка об'єкта проєктування	*Яцук А. А.*		
Нормоконтроль	*Повстяна Ю. С.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	3,8 %		
Академічна доброчесність	*Яцук А. А.*		

7. Дата видачі завдання *«2» лютого 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*07.02.2024 р.*	вик
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2024 р.*	вик
3	*Обгрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*12.03.2024 р.*	вик
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2024 р.*	вик
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2024 р.*	вик
6	*Тестування та налагодження об'єкта проектування*	*01.06.2024 р.*	вик
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*19.06.2024 р.*	вик

Здобувач вищої освіти _____ (підпис) *Кліменко А. В.* (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) *Яцук А.А.* (прізвище, ініціали)

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти: Кліменко Артем Вікторович

Тема: Розробка додатку «Minds Keeper» для нотаток на iOS за допомогою SwiftUI

Коротка характеристика кваліфікаційної роботи: метою роботи є розробка функціонального додатку для нотаток, який використовує можливості SwiftUI для створення зручного і естетично привабливого інтерфейсу. Завдання включають аналіз вимог користувачів, проектування архітектури додатку, розробку інтерфейсу, реалізацію функціоналу та тестування готового продукту.

Самостійні розробки і пропозиції автора: розроблено додаток для нотаток з функціями приховування вмісту та блокування.

Практичне значення роботи: використання додатку для створення, збереження приховування та блокувння нотаток у зручному та швидкому інтерфейсі користувача.

Недоліки: Обмежена підтримка старих версій iOS: SwiftUI доступний лише на iOS 13 та новіших версіях, що обмежує користувацьку базу на старіших пристроях.Обмежені можливості кастомізації: Хоча SwiftUI спрощує створення інтерфейсу, він може бути менш гнучким у порівнянні з UIKit для складних кастомних компонентів. Відсутність деяких функціональних можливостей: SwiftUI все ще розвивається, і деякі можливості можуть бути недоступними або потребувати обхідних рішень.

Загальний висновок: Розробка додатку для нотаток "Minds Keeper" з використанням SwiftUI має значні переваги, включаючи швидкий та ефективний процес створення інтерфейсу, сучасний підхід до програмування та можливість інтеграції з іншими технологіями Apple для забезпечення безпеки та зручності користувачів. Однак, існують певні обмеження, такі як підтримка лише новіших версій iOS, обмежені можливості кастомізації та потенційні проблеми з продуктивністю.

Рецензент: Саварин Павло Вікторович
(прізвище, ім'я, по-батькові, посада)

Рецензент кваліфікаційної роботи _____
(підпис)

«7» червня 2024 р.

Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

ВІДГУК
КЕРІВНИКА НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Здобувач вищої освіти Кліменко Артем Вікторович
(прізвище, ім'я, по батькові)

група ІПЗс-21

Тема кваліфікаційної роботи: РОЗРОБКА ДОДАТКУ «MINDS KEEPER» ДЛЯ НОТАТОК НА IOS ЗА ДОПОМОГОЮ SWIFTUI.

Керівник к.т.н., доц. Ящук А. А.

Актуальність теми. у сьогоднішньому світі, де обсяги інформації зростають щодня, мобільні застосунки для ведення нотаток стають все більш важливими. Вони допомагають користувачам ефективно організовувати свої дані, легко знаходити необхідну інформацію та забезпечувати її збереження. В контексті стрімкого розвитку ринку мобільних технологій, створення застосунка для нотаток на мові Swift для iOS є актуальним і своєчасним завданням.

Об'єкт дослідження – програмне середовище Xcode для створення прикладних програм з використанням мови Swift та Ojective C.

Характеристика теоретичного рівня, наявності самостійних розробок і практичної значимості роботи. Було проведено аналіз предметної області, що включає в себе дослідження аналогів, аналіз ринку та потреб користувача. Також було вибрано сучасні технології для розробки: Xcode, Swift та Ojective C. Здійснено розробку додатку «Minds Keeper» для зберігання нотаток. Розроблений додаток дозволить створювати нотатки та приховувати їх за допомогою пароля або біометричних даних.

Загальний висновок

Роботу виконано у повному обсязі у відповідності до поставлених завдань, а здобувач Кліменко А. В. заслуговує на високу оцінку.

Керівник _____ **Ящук А. А.**
(підпис) (прізвище, ім'я, по батькові)

«7» червня 2024 р.

АНОТАЦІЯ

Кліменко А. В. Розробка додатку «Minds Keeper» для нотаток на iOS за допомогою SwiftUI. Рукопис.

Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків і пропозицій, списку використаних джерел та додатків.

У першому розділі здійснено аналіз сучасного стану проблеми розробки додатку нотатки і сформульовано завдання. В другому розділі визначено вимоги до розроблюваної системи, а також засоби, методи і алгоритми розробки. У третьому розділі розроблено додаток нотатки «Minds Keeper» мовою SwiftUI. У висновках узагальнено інформацію, відображену у попередніх частинах. Результати розробки демонструють можливості розвитку мобільних застосунків з функцією блокування та приховування важливих даних.

Ключові слова: SwiftUI, FaceID, TouchID, Xcode, iOS.

ABSTRACT

Klimenko A. V. Development of "Minds Keeper" application for notes on iOS using SwiftUI. Manuscript.

Bachelor's qualifying thesis of the educational program "Software Engineering". Lutsk National Technical University. Lutsk, 2024.

The bachelor's qualifying thesis consists of an introduction, three sections, conclusions and proposals, a list of used sources and annexes.

In the first chapter, an analysis of the current state of the problem of developing computer games was carried out and the task was formulated. The second section defines the requirements for the developed system, as well as means, methods and development algorithms. In the third chapter, application "Minds Keeper" was developed and tested. The conclusions summarize the information presented in the previous parts. The development results demonstrate the possibilities of development of creating mobile applications with security functions and hiding significant data.

Keywords: SwiftUI, FaceID, TouchID, Xcode, iOS.

ЗМІСТ

ВСТУП

Актуальність розробки. У сьогоднішньому світі, де обсяги інформації зростають щодня, мобільні застосунки для ведення нотаток стають все більш важливими. Вони допомагають користувачам ефективно організовувати свої дані, легко знаходити необхідну інформацію та забезпечувати її збереження. В контексті стрімкого розвитку ринку мобільних технологій, створення застосунка для нотаток на мові Swift для iOS є актуальним і своєчасним завданням.

У рамках роботи буде проведено аналіз наявних на ринку аналогів, встановлено ключові вимоги до функціонала майбутнього додатку, розроблено його архітектуру та дизайн, а також втілено в життя за допомогою інструментів Swift.

Метою кваліфікаційної роботи є створення додатку для нотаток «Minds Keeper» засобами об’єктно-орієнтованого середовища мови програмування SwiftUI та дослідження його складових компонентів.

Об’єкт кваліфікаційної роботи – програмне середовище Xcode для створення прикладних програм з використанням мови Swift та Ojective C.

Предметом кваліфікаційної роботи є дослідження особливостей, принципу та алгоритмів функціонала додатку для нотаток «Minds Keeper».

Завданнями кваліфікаційної роботи є:

- провести детальний опис предметної області;
- оглянути аналогічні програмні продукти;
- провести аналіз вимог до розроблювального продукту;
- визначити методи та засоби для створення проекту;
- здійснити розробку додатку «Minds Keeper» для зберігання нотаток;
- провести тестування функціональностей додатку;
- створення інсталяційного пакету.

Розроблений додаток дозволить створювати нотатки та приховувати їх за допомогою пароля або біометричних даних.

РОЗДІЛ 1
АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ДОДАТКУ МОВОЮ ПРОГРАМУВАННЯ SWIFTUI І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

iOS – це операційна система для мобільних пристроїв, розроблена компанією Apple Inc і представлена в [1]. Вона розроблена для використання з такими пристроями Apple, як iPhone, iPad та iPod Touch, зображена на рисунку 1.1.

Рисунок 1.1 – Інтерфейс iPhone

Переваги користування iOS:

– стійкість до шкідливого ПЗ, iOS підтримує закритість файлової системи, підтвердження цього факту стверджується, що iOS працює тільки з офіційними додатками App Store, в яких шкідливий код був попередньо перевірений;

– довго тривала підтримка, оновлення iOS випускаються регулярно, щоб остання версія iOS була доступна на всіх підтримуваних пристроях Apple, користувачі отримують нові функції та розширення, які усувають помилки та проблеми з системою;

– якість та ексклюзивність додатків, користувач має можливість завантажувати додатки з App Store, що містить значну кількість високоякісних та ексклюзивних програмних продуктів, недоступних в інших мобільних ОС;

– практичність вбудованого ПЗ, вбудовані додатки iOS (такі як: Календар, Нотатки, Камера і т. д.), що є найзручнішими і найпродуманішими серед всіх мобільних операційних систем;

– екосистема Apple, дозволяє користувачеві легко працювати в парі зі своїм Mac, iPad та іншими пристроями Apple, створювати та передавати файли через хмарний сервіс iCloud.

Недоліки iOS:

– обмежена кастомізація, брак можливостей настроювання різних аспектів системи, таких як теми, шрифтів, розмір іконок і т. д.;

– висока ціна, пристрої Apple відомі своєю високою ціною порівняно з іншими виробниками.

Вибраною мовою програмування для даного проекту є SwiftUI.

SwiftUI – це декларативний фреймворк для створення інтерфейсів користувача для додатків на платформах iOS, macOS, watchOS та tvOS наведено у [2, 3].

Ось деякі ключові характеристики мови SwiftUI:

– декларативний підхід, можливість описувати інтерфейс користувача у вигляді декларативних структур, що визначають, як має виглядати UI в різних станах та за різних умов. Цей підхід спрощує розробку та робить код більш зрозумілим;

– реактивність, змога SwiftUI використовувати парадигму реактивного програмування, де зміни в даних автоматично відображаються в

UI, а зміни в UI можуть відразу ж впливати на дані. Це дозволяє створювати динамічні та взаємодійні додатки;

– універсальність, підтримка різних платформ Apple, таких як: iOS, macOS, watchOS та tvOS, що дозволяє розробникам писати спільний код для додатків для різних пристроїв;

– інтуїтивне API, здатність SwiftUI надавати простий та інтуїтивно зрозумілий API, який дозволяє швидко створювати складні інтерфейси користувача за допомогою мінімальної кількості коду;

– попередній перегляд у реальному часі, можливість переглядати зміни у реальному часі прямо в Xcode без необхідності перезапуску додатка. Це дозволяє розробникам швидше прототипувати та тестувати зміни у інтерфейсі;

– інтеграція з іншими технологіями, взаємодія з іншими технологіями та фреймворками Apple, такими як: Combine (для роботи з асинхронними подіями), Core Data (для роботи з базами даних), та іншими.

SwiftUI змінив підхід до створення інтерфейсів користувача для платформ Apple, забезпечуючи простоту, ефективність та потужність для розробників.

Принцип будування додатку для iOS за допомогою SwiftUI. Створення нового проекту у Xcode для платформи iOS з використанням мови програмування Swift. Використовуючи дану мову програмування, розробники описують інтерфейс користувача за допомогою декларативного підходу. Вони визначають різні елементи UI, такі як: тексти, кнопки, списки тощо, та їх взаєморозміщення. Відповідно, визначають структуру додатка, включаючи різні екрани, їх навігацію та взаємодію між ними. Встановлення дій та обробників подій для використання взаємодії елементів таких як кнопки та тест та задання певних дій. Після цього використовуються структури даних та властивості для управління даними в додатку. Вони можуть використовувати спеціальні класи для завантаження та збереження даних на пристрої, (рис. 1.2).

Рисунок 1.2 – Приклад використання SwiftUI

Додаток тестується на різних пристроях та в різних умовах, а також налаштовується для виявлення та виправлення помилок.

Розгортання на пристроях, після успішного тестування додаток готовий до розгортання на реальних пристроях користувачів або публікації в App Store.

В сучасному світі мобільних технологій iOS визнається однією з найпопулярніших та широко використовуваних платформ для розробки мобільних додатків. У той же час, мова програмування Swift вирізняється своєю ефективністю та потужністю, що робить її ідеальним вибором для створення програмного забезпечення для цієї платформи. Основні особливості створення додатків для iOS мовою Swift:

– простота вивчення та використання, легкий спосіб використання Swift, що сприяє швидкому вивченню для початківців. Її синтаксис досить зрозумілий і схожий на англійську мову, що полегшує розробку та збільшує продуктивність;

– висока продуктивність, здатність Swift надати високу продуктивністю, що дозволяє створювати швидкі та ефективні додатки для iOS. Вона має оптимізовану систему керування пам’яттю та різноманітні оптимізації, які допомагають покращити продуктивність програм;

– інтеграція зі сховищами даних, взаємодія з різноманітними сховищами даних, такими як: Core Data, Realm або Firebase. Це дозволяє легко зберігати та управляти даними в додатках для iOS;

– велика спільнота та підтримка, активна спільнота розробників, що постійно розвивається та надає численні джерела, бібліотек та інструментів для полегшення розробки додатків для iOS;

– платформа для майбутнього, розвивиток Swift та платформи iOS, випускаючи нові версії мови та оновлення для операційної системи. Це робить Swift ідеальним вибором для розробки додатків, оскільки вона постійно адаптується до оновлень платформи.

Використання UIKit та SwiftUI, дані фреймворки є двома основними інструментами для створення інтерфейсів користувача в додатках для iOS. UIKit є класичним інтерфейсним фреймворком, тоді як SwiftUI є новітнім декларативним підходом до побудови інтерфейсів. Swift підтримує обидва фреймворки, що дає розробникам вибір у залежності від їх потреб та вимог проекту.

Для створення додатків для платформи iOS головними мовами програмування є Swift та Objective-C. Для цього необхідне спеціальне програмне забезпечення, яке можна завантажити з офіційного сервісу App Store для iOS та macOS. Одним з ключових інструментів у цьому програмному комплексі є IDE Xcode, розроблений компанією Apple.

Щоб розробити корисний і зручний мобільний додаток для операційної системи iOS, необхідно провести детальний аналіз популярних застосунків на ринку. Такий аналіз допоможе визначити ключові функції, які потрібно включити, а також визначити сильні та слабкі сторони існуючих додатків. Це дозволить сформувати чітке уявлення про важливі аспекти, на які слід звернути увагу в процесі розробки.

«Notability: Notes, PDF» є додатком для створення нотаток, який використовується студентами та професіоналами для запису інформації та організації їх робочого процесу наведено у [4]. Цей додаток доступний на

платформах iOS та macOS і відомий своєю гнучкістю та широким набором функцій представлений на рисунку 1.3. Основні функції представлені нижче.

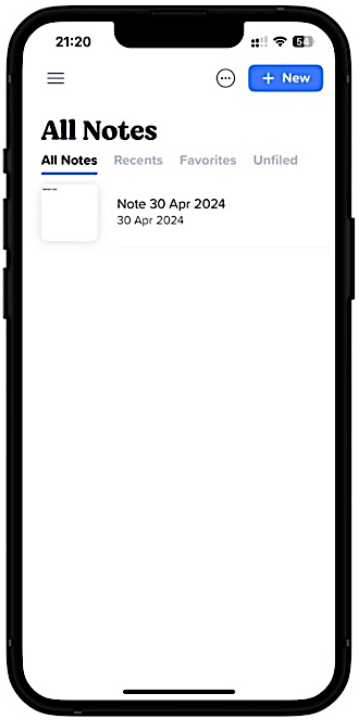

Рисунок 1.3 – Інтерфейс головного екрану Notability

Створення нотаток, запис тексту за допомогою клавіатури або рукописний ввід, можливість додавання малюнків та нотаток різними кольорами та стилями, вставка фотографій та інших медіафайлів.

PDF анотації, імпорт PDF файлів для анотування, можливість підкреслення, виділення тексту, додавання коментарів у PDF, експорт анотованих PDF файлів для подальшого використання або розділення.

Аудіо записи, запис аудіо одночасно з роботою над нотатками, синхронізація аудіо з текстом, що дозволяє легко повертатися до конкретних моментів лекцій або зустрічей.

Організація і співпраця, організація нотаток у папки та розділи, поділ нотаток з іншими через різні формати, такі як: PDF, DOCX або через безпосереднє спілкування в додатку, синхронізація нотаток з iCloud, Google Drive, Dropbox та іншими популярними хмарними сервісами.

Перевагами використання Notability стане можливість використовувати на різних пристроях з синхронізацією усіх даних, універсальністю, що підходить для студентів, вчителів, дослідників, бізнес-професіоналів та підтримка рукописного вводу оптимізовано для використання зі стилусами, як Apple Pencil.

Недоліками Notability може бути те що додаток платний, хоча одноразова вартість зазвичай не висока, може бути бар'єром для деяких користувачів, обмежена підтримка платформ, головним чином фокусується на користувачах Apple, що може бути незручно для користувачів інших систем.

«Craft: Write docs, AI editing» для покращення процесу написання та редагування створення записів в додаток інтегровано штучний інтелект. Даний додаток містить розширений функціонал в якому можна створити швидку нотатку та документ, ключовою особливість є наявність готових шаблонів з актуальними оформленнями на будь-які випадки, представлений екран додавання нотатків на рисунку 1.4 та наведено у роботі [5].

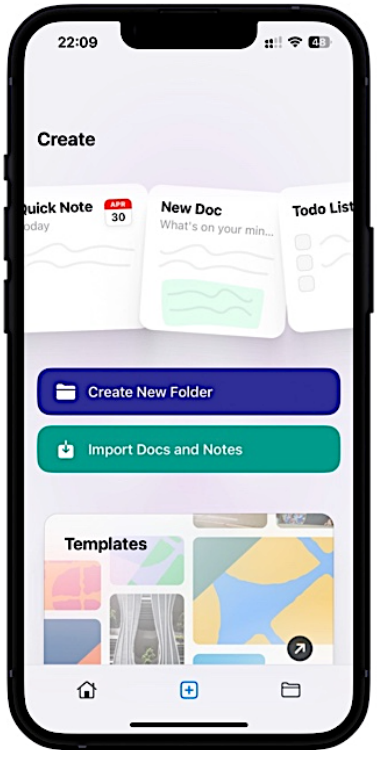

Рисунок 1.4 – Екран створення записів Craft

Основні функції цього додатку представлені нижче.

Допомога при написанні на основі штучного інтелекту, ця функція, щоб допомогти користувачам генерувати текст на основі наданих підказок або тем. Штучний інтелект може допомогти у написанні есе, звітів, статей і навіть творчих історій. Він пропонує пропозиції щодо покращення структури речень, словникового запасу та загальної зв’язності, що робить процес написання плавнішим та ефективнішим.

Виправлення граматики та стилю у режимі реального часу, коли користувач друкує, програма надає відгуки про граматичні, пунктуаційні та стилістичні помилки. Ця функція виходить за рамки простої перевірки орфографії, аналізуючи контекст кожного речення, щоб запропонувати виправлення та пропозиції, які покращують читабельність і професіоналізм.

Контекстний тезаурус і розширення словникового запасу, який пропонує синоніми та альтернативні вирази на основі контексту написаного. Це допомагає користувачам уникнути повторень і збагатити їхню мову.

«Writer – Notes, Lists, Editor» додаток призначений для спрощення процесу створення, редагування та організації нотаток, наведено у роботі [6]. Цей додаток особливо корисний для користувачів які люблять мінімалізм (рис. 1.5).

Рисунок 1.5 – Написання та редагування нотатки Writer

Зручний інтерфейс, додаток має чистий інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко почати робити нотатки відразу. Навігація проста, що дозволяє користувачам швидко знаходити та впорядковувати свої нотатки.

Редагування тексту, користувачі можуть користуватися редактором форматованого тексту, додати його у новостворену категорію, створити закладку або ж додати у категорію «Вподобані».

У налаштуваннях користувачі можуть налаштувати зовнішній вигляд піктограм програм на свій смак, зокрема змінити теми, шрифти та макети.

«Noted – Record, Transcribe» дозволяє користувачам записувати аудіо, а потім транскрибувати його в текст, наведено у роботі [7]. Це може бути корисним для різних цілей, таких як створення нотаток під час зустрічей, лекцій чи інтерв'ю, де запис аудіо та його текстової транскрипції може допомогти в документації, перегляді та доступності представлено на рисунку 1.6.

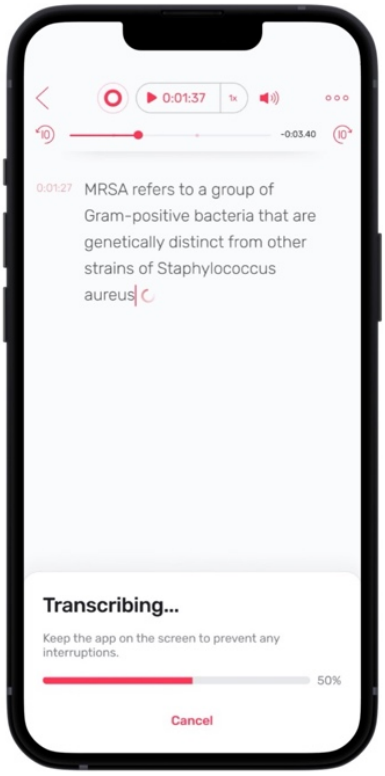

Рисунок 1.6 – Використання штучного інтелекту, транскрибування записаного тексту Noted

Запис аудіо, користувачі можуть записувати аудіо безпосередньо в додатку за допомогою мікрофона смартфона або периферійних пристроїв.

Жива транскрипція, під час запису аудіо, технологія перетворення мови в текст перетворює вимовлені слова в письмовий текст у режимі реального часу. Ця функція буде корисною для користувачів, яким потрібні негайні письмові записи розмов або виступів.

Редагування запису, користувачі можуть мати можливість редагувати як аудіо, так і транскрипцію. Це може включати вирізання небажаних частин аудіо або виправлення будь-яких помилок у транскрипції.

Функція пошуку, дає можливість знайти в транскрипціях конкретних слів або фраз для полегшення пошуку важливої інформації.

Багатомовна підтримка, додаток підтримує різні мови, розширюючи його можливості використання в різних лінгвістичних демографічних групах.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Темою кваліфікаційної роботи – створення додатку для нотатки «Minds Keeper» для операційної системи iOS та iPadOS мовою програмування SwiftUI в середовищі розробки Xcode Apple.co.

Поставлені задачі до виконання, а саме:

– проведення огляду аналогічних програмних продуктів, їхніх функцій та ключових функцій;

– проведення детального аналізу технологій розробки додатків мовою SwiftUI, опис предметної області розроблюваної системи;

– визначення вимог до розроблюваного проекту, створення функціональної схеми;

– реалізація додатку з використаням даної мови програмування та інстроментів для розробки;

– тестування продукту, проведення детального тестування використовуючи локальні інстументи;

– створення інсталяційного пакету, виконання всіх вимог до архівування новоствореного продукту.

Технічними вимогами є мова програмування Swift та тип збереження даних UserDefaults, даний додаток міститиме бібліотеку LocalAuthentication, що даватиме змогу блокувати його паролем та біометричними даними (FaceID і TouchID).

Проектування інтерфейсу буде створенням головного екрану, що міститиме список усіх нотаток з заголовком і частиною тексту, кнопка для додавання нової нотатки, пошуковий рядок для фільтрації нотаток, сортування ноток за критерієм.

Висновки до розділу 1

Проаналізувавши технологію створення додатків на мові програмування SwiftUI та поставивши завдання для кваліфікаційної роботи, змогли оглянути можливості цієї технології та її придатність для розробки сучасних мобільних додатків.

SwiftUI, набір інструментів Apple, пропонує розробникам значні переваги, зокрема декларативний синтаксис, ефективну маніпуляцію даними через зв’язування даних і тісну інтеграцію в екосистему Apple.

Під час дослідження було визначено важливі аспекти та функції SwiftUI, які можна використовувати для покращення процесу розробки та оптимізації взаємодії з користувачем.

Поставлені завдання кваліфікаційної роботи включають розробку прототипу додатку, аналіз його інтерфейсу та взаємодії з користувачами, завдяки чому, тепер можлива практичне використання отриманих теоретичних знань.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Розробка програми для створення нотаток вимагає глибокого розуміння потреб користувачів і поточних технічних можливостей представлено в [8].

Важливо розглянути аналогічні продукти на ринку, такі як: Evernote, Google Keep і Apple Notes, щоб визначити, які функції та інтерфейс зроблять нову програму більш конкурентоспроможною.

Слід проаналізувати такі аспекти:

– потреби користувачів, збір відгуків та аналіз використання існуючих рішень для виявлення їхніх переваг та недоліків;

– технологічні тенденції, вибір сучасних фреймворків та платформ для розробки, враховуючи мультиплатформеність, безпеку та швидкість роботи;

– конкурентний аналіз, визначення ключових особливостей конкурентів та можливостей для диференціації.

На основі аналізу формулюються функціональні та нефункціональні вимоги до додатку.

Функціональні вимоги:

– можливість створення, редагування та видалення нотаток;

– підтримка текстового форматування (жирний шрифт, курсив, підкреслення);

– можливість додавання зображень та файлів до нотаток;

– організація нотаток за допомогою папок або тегів;

– функції пошуку по тексту нотаток;

– захист нотаток паролем або біометричними даними.

Нефункціональні вимоги:

– продуктивність, швидка реакція додатку на користувацькі команди;

– доступність, інтуїтивно зрозумілий інтерфейс, адаптований під різні пристрої та розміри екранів;

– безпека, надійне шифрування даних та захист від несанкціонованого доступу;

– скальованість, можливість легко масштабувати систему для обслуговування зростаючої кількості користувачів.

Схема додатку представлена на рисунку 2.1.

Рисунок 2.1 – Схема додатку

На етапі проектування розробляється архітектура додатку, інтерфейс користувача та базова модель даних.

Архітектура визначатиме тип архітектури між монолітною та мікросервісною архітектурою в залежності від очікуваного навантаження та масштабів проекту.

Інтерфейс користувача включатиме розробку прототипів екранів додатку, забезпечення зручності та логічності навігації.

Модель даних буде використовуватись для структурування бази даних для зберігання нотаток, зображень та користувацьких налаштувань.

Функціональний алгоритм додатку представлено на UML діаграмі (рис. 2.2).

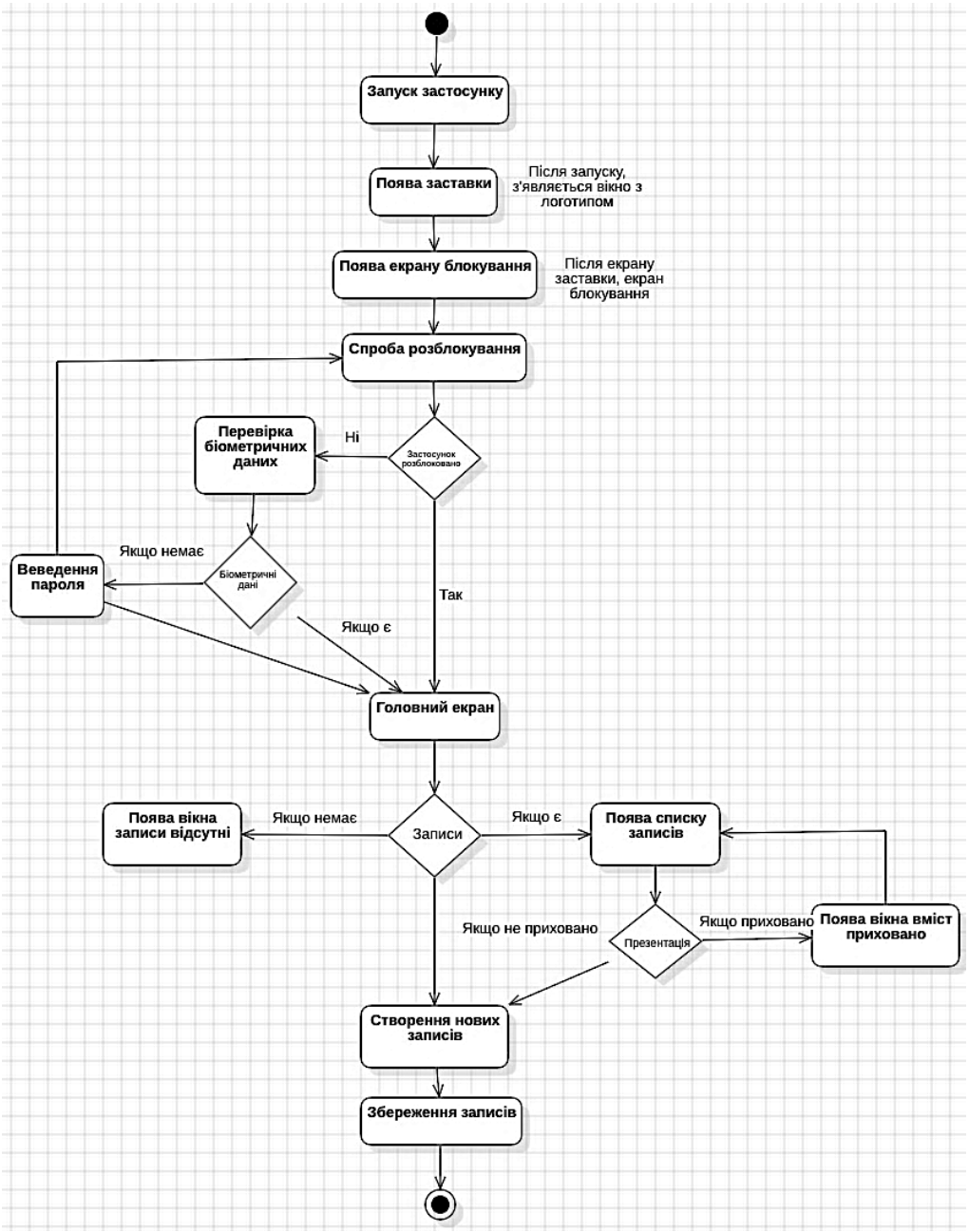

Рисунок 2.2 – UML діаграма, функціональна схема додатку

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Xcode – це набір інструментів, створений Apple для створення програм на Mac OS X і iPhone наведено у [9]. Остання доступна версія – Xcode 15.3, яка включається безкоштовно на дистрибутивному диску Mac OS X Install DVD, що постачається разом з операційною системою Mac OS X, але не встановлюється за умовчанням. Хоча третя версія не підтримується на старих версіях Mac OS, Xcode доступний для безкоштовного завантаження через Apple Developer Connection. Оновлення можна безкоштовно отримати на офіційному веб-сайті підтримки. На сьогодні Xcode є єдиним засобом для написання «універсальних» (Universal Binary) програм для Mac OS X і має тісну інтеграцію з фреймворком Cocoa.

Xcode IDE має усе необхідне для розробки: від професійних редакторів з автозаповненням коду та вбудованого рефакторінгу Cocoa до можливостей налаштування компілятора з відкритим кодом. Це інтегроване середовище розробки було створено з нуля для повного використання всіх можливостей Cocoa та останніх технологій від Apple (рис. 2.3).

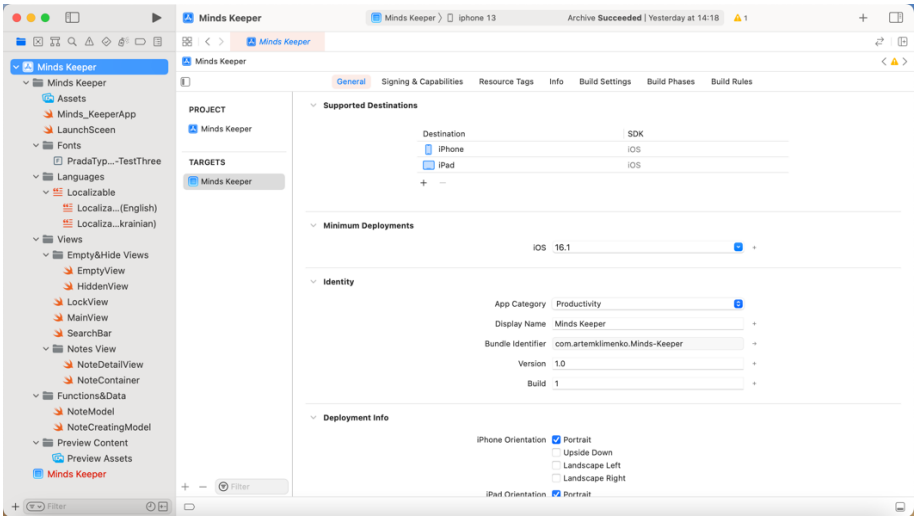

Рисунок 2.3 – Головний екран Xcode IDE

Щоб полегшити розробку додатків для iOS, Xcode включає багато функцій, таких як:

– систему керування проектом для розпізнавання програмних продуктів;

– середовище для редагування коду, яке має можливості підсвічування синтаксису, автозаповнення коду та індексацію символів;

– просунуту систему компіляції проекту з перевіркою залежностей і перевіркою правил складання;

– збірки коду GCC, що підтримують мови C, C++, Objective-C, Objective-C++ та інші;

– підтримка LLVM і Clang для мов C, C++ і Objective-C;

– вбудоване налаштування вихідного коду з використанням GDB, розподілені обчислення, що дозволяють поширювати великі проекти через кілька мережевих пристроїв;

– інтелектуальна збірка проекту, яка прискорює час компіляції одного файлу;

– просунуті засоби налагодження, що дозволяють виконувати глобальні зміни в коді без зміни його поведінки;

– підтримка знімків проекту, які надають легше управління вихідним кодом;

– підтримка AppleScript для автоматизації процесу складання.

Для додатку нотатки важливо вибрати правильну архітектуру, що підтримує читабельність, масштабованість та легке управління.

MVC (Model-View-Controller) – традиційна для iOS архітектура, яка дозволяє розділити дані (Model), інтерфейс (View) та логіку управління (Controller).

MVVM (Model-View-ViewModel) – покращує тестування і розділення коду за рахунок введення компоненту ViewModel, який спрощує біндінг даних.

Інструментами та технологіями буде мова програмування SwiftUI яка є для розробки інтерфейсу. Swift UI пропонує декларативний підхід до створення UI. UserDefaults – компонент у системі iOS, який забезпечує простий механізм для зберігання легких даних, таких як налаштування користувача чи короткі стани додатків.

AppState для зберігання даних та стану, які потрібні через багато або всі UI компоненти додатку. Використання спільного стану допомагає уникнути дублювання даних і забезпечує консистентність додатку.

Основні функції додатку:

- створення, редагування та видалення заміток;
- можливість додавання тегів або категорій;
- пошук по замітках;
- сортування заміток за датою або за пріоритетом.

Компоненти для створення інтерфейсу слугуватимуть, List і ForEach в SwiftUI для відображення списку заміток, Buttons, та TextFields для введення та редагування даних та Navigation Views для організації навігації в додатку. Для розміщення об’єктів та функціональних елементів VStack і HStack, вертикально і горизонтально.

Інтеграцією в додаток WidgetKit для створення віджетів на головний екран.

Тестування і розгортання додатку за допомогою, Unit tests для логіки додатку, UI tests для інтерфейсу та Integration tests для перевірки взаємодії компонентів.

Ресурси для навчання, офіційна документація Swift і SwiftUI та використання тестування мови програмування в Playgrounds.

В розробці використано бібліотеки SwiftUI, Foundation, Combine та LocalAuthentication.

Бібліотека SwiftUI – це набір інтерфейсних компонентів та інструментів, що дозволяють створювати інтерфейси користувача для програм на платформах Apple, таких як iOS, iPadOS, macOS, watchOS та tvOS. SwiftUI автоматично

оновлює відображення інтерфейсу користувача при зміні стану даних. Також містить велику бібліотеку стандартних інтерфейсних компонентів, таких як кнопки, тексти, списки, зображення та інших.

Бібліотека Foundation – це фреймворк у мові програмування Swift, який надає базові класи та утиліти для роботи з даними, мережами, файловою системою, рядками, датами, серіалізацією та іншими базовими операціями надано у [10]. Типи даних, фреймворк Foundation містить класи для роботи з основними типами даних, такими як рядки (String), числа (NSNumber), масиви (NSArray), словники (NSDictionary), набори (NSSet) та інші. Фреймворк надає можливості роботи з мовними ресурсами, локалізацією текстових рядків та форматуванням даних відповідно до локалі користувача.

Бібліотека Combine надає декларативну мову програмування, щоб працювати з асинхронними потоками даних і подій, такими як зміни значень властивостей, мережеві запити, обробка даних і т. д. надано у [11]. Combine базується на концепції реактивного програмування і дозволяє легко збирати, перетворювати та реагувати на потоки подій. Operator (Оператор) представляє функції, які можна застосувати до потоку даних, щоб перетворити, фільтрувати, комбінувати або обробляти його.

Бібліотека LocalAuthentication – надає функції блокування/розблокування біометричної аутентифікації та аутентифікації за допомогою пін-коду на пристроях iOS та macOS надано у [12]. Вона дозволяє розробникам використовувати вбудовані механізми аутентифікації на пристроях Apple для захисту конфіденційних даних або доступу до певних функцій додатка. Touch ID та Face ID за допомогою цієї бібліотеки можна легко використовувати біометричні дані користувача, якщо пристрій має апаратну сумісність.

Висновки до розділу 2

Аналіз включає вивчення потреб користувачів і конкурентного середовища, оцінку технічних можливостей і обмежень платформи, а також визначення основних функцій і характеристик програми.

Визначення вимог – це специфікація функціональних і нефункціональних вимог програми, що точно визначає, що програма повинна робити і яким вимогам вона повинна відповідати, щоб досягти цієї мети. Це може включати опис інтерфейсу користувача, функціональність, робочий процес, вимоги безпеки та інші аспекти

Розробка програмного забезпечення з використанням мови SwiftUI передбачає розробку архітектур додатків, вибір шаблонів проектування та структур коду, створення інтерфейсів користувача за допомогою SwiftUI та інтеграцію з іншими компонентами системи та зовнішніми службами.

Загальний висновок полягає в тому, що аналіз, визначення вимог і проектування програмного забезпечення є критичними етапами, які визначатимуть успіх майбутньої розробки додатків SwiftUI вони допомагають визначити цілі та потреби користувачів, створювати ефективні та функціональні проекти та забезпечувати якість та ефективність програмного забезпечення.

РОЗДІЛ 3
РОЗРОБКА ДОДАТКУ «MINDS KEEPER» ДЛЯ НОТАТОК ЗА ДОПОМОГОЮ SWIFTUI

3.1 Практична реалізація об’єкта проєктування

Початком створення будь-якого додатку та коректної роботи є вікно запуску що включає в себе назву, слоган та графічні елементи [13-16]. Код програми наведений у додатку А та представлений вигляд на рисунку 3.1.

Рисунок 3.1 – Вікно запуску

Задля безпеки записів у додатку міститься вікно блокування, функцію розблокування по біометричним даним (FaceID і TouchID) та за паролем користувача. Дане вікно містить текст стану та кнопку розблокування, коли не доступні біометричні дані. Функцію розблокування представлено в лістингу 3.1. Основний код вигляду вікна блокування представлений у додатку Б.

Лістинг 3.1 – Функція розблокування

```
func authenticate() {
        // Зміст вікна розблокування
        let context = LAContext()
        // Вікно в разі помилки розблокування
        var error: NSError?
```

```
            // Перевірка чи взагалі доступне розблокування по біометричним
даним (FaceID&TouchID)
          if context.canEvaluatePolicy(.deviceOwnerAuthentication, error:
&error) {
// Вікно розблокування
              let stat = NSLocalizedString("statReason", comment: "")
              context.evaluatePolicy(.deviceOwnerAuthentication,
localizedReason: stat)
                  { success, _ in
                  // Розблокування успішне
                  if success {
                      unlocked = true
                  }
                  // Помилка розблокування спроба знову або пароль
                  else {
                     status_text = NSLocalizedString("unabil", comment: "")
                  }
              }
          }
          // Розблокування не можливе
          else {
              status_text = NSLocalizedString("nobio", comment: "")
         }
    }
```

Кінець лістингу 3.1

Зовнішній вигляд вікна блокування представлений на рисунку 3.2.

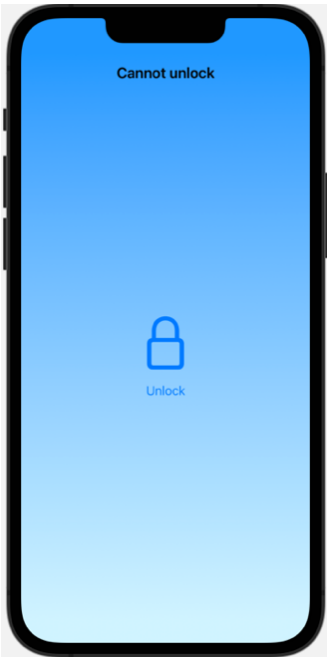

Рисунок 3.2 – Вікно блокування

Відображення елементів керування, логіки та даних слугуватиме головне вікно. В даному вікні містяться кнопки «Додати» та «Приховати», пошуковий рядок та сортування. Зовнішній вигляд вікна представлено на рисунку 3.3 та програмний код у додатку В.

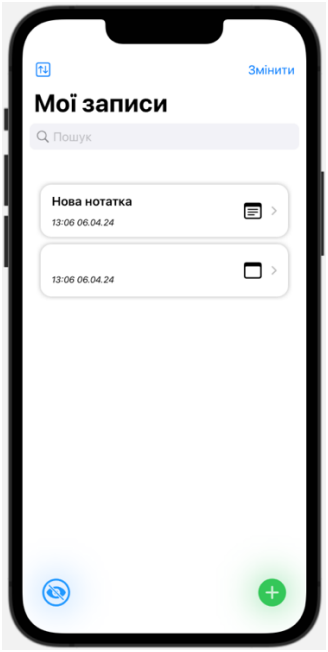

Рисунок 3.3 – Головне вікно

Принцип створення нотатки в додатку включає етапи, які наведені далі.

Дані нотатки яка міститиме ідентифікатор, вміст, дату створення та спосіб сортування, представлено в лістингу 3.2.

Лістинг 3.2 – Модель нотатки, функція сортування

```
import SwiftUI
// Модель нотатки
struct Note: Identifiable, Codable, Hashable, Comparable {
//  Ідентифікатор
    var id: UUID = UUID()
//  Вміст
    var content: String
//  Дата створення
    var date: Date
// Реалізація Comparable(порівняння) для сортування | По даті | По
ідентифікатору
     static func < (lhs: Note, rhs: Note) -> Bool {
            lhs.date < rhs.date
```

```
        }
      static func == (lhs: Note, rhs: Note) -> Bool {
            lhs.id == rhs.id
        }
      }

//    Вибір сортування | від нових, від старих, від А до Я
    enum SortOption {
            case byDateAscending, byDateDescending, byContent
       }
//    Тип сортування
    func sortNotes() {
            switch currentSort {
            case .byDateAscending:
                notes.sort(by: { $0.date < $1.date })
            case .byDateDescending:
                notes.sort(by: { $0.date > $1.date })
            case .byContent:
    notes.sort(by: { $0.content.lowercased() < $1.content.lowercased() })
            }
        }
```

Кінець лістингу 3.2

Натискання кнопки «Додати нотатку» ініціює процес створення нової нотатки, натиснувши відповідну кнопку або значок, код програми (ліст. 3.3).

Лістинг 3.3– Функція створення нотатки

```
func add(){
    let newNote = Note(
           content: NSLocalizedString("new_note", comment: ""),
           date: Date())
    notes.append(newNote)}
```

Кінець лістингу 3.3

Створюється новий екземпляр класу Note, параметр content встановлюється на локалізовану рядок «new_note», це рядок, що вказує на те, що це нова нотатка. Параметр date встановлюється на поточну дату та час створення за допомогою Date().

Введення вмісту та редагування, текст примітки відображається у відповідному полі, код програми (ліст. 3.4).

Лістинг 3.4 – Функція редагування нотатки

```
func update(note: Note, with newContent: String){
     if let index = notes.firstIndex(where: { $0.id == note.id }) {
          notes[index].content = newContent
          notes[index].date = Date()}}
```

Кінець лістингу 3.4

Функція firstIndex(where:) шукає перший елемент у масиві, який задовольняє умову, вказану в замиканні. Умова $0.id = = note.id порівнює id нотатки з id переданої нотатки для редагування, id є унікальним ідентифікатором для кожної нотатки. Якщо нотатка знайдена у списку, то її вміст (content) замінюється на новий вміст, переданий як параметр newContent. Дата нотатки також оновлюється на поточну дату та час за допомогою Date().

Збереження нотатки відбуватиметься, коли користувач створить запис, а потім натисне кнопку «Зберегти», код програми (ліст. 3.5).

Лістинг 3.5 – Функція збереження нотатки

```
func save(){
    if let encoded = try? JSONEncoder()
      .encode(notes) {
          UserDefaults.standard.set(encoded, forKey: "notes")}}
```

Кінець лістингу 3.5

Використовується JSONEncoder, який дозволяє перетворити об’єкти Swift в формат JSON, .encode(notes) – кодує список notes у JSON формат. Закодовані дані (encoded) зберігаються у локальному сховищі UserDefaults за ключем «notes».

Даний додаток міститиме дві мови: англійську та українську, тому потребуватиме локалізацію. Локазізація відбувається зі створеням локалізованих рядків у сам проект, додаванням потрібних мов у проект та створеням ключів з готовим перекладом (рис. 3.4).

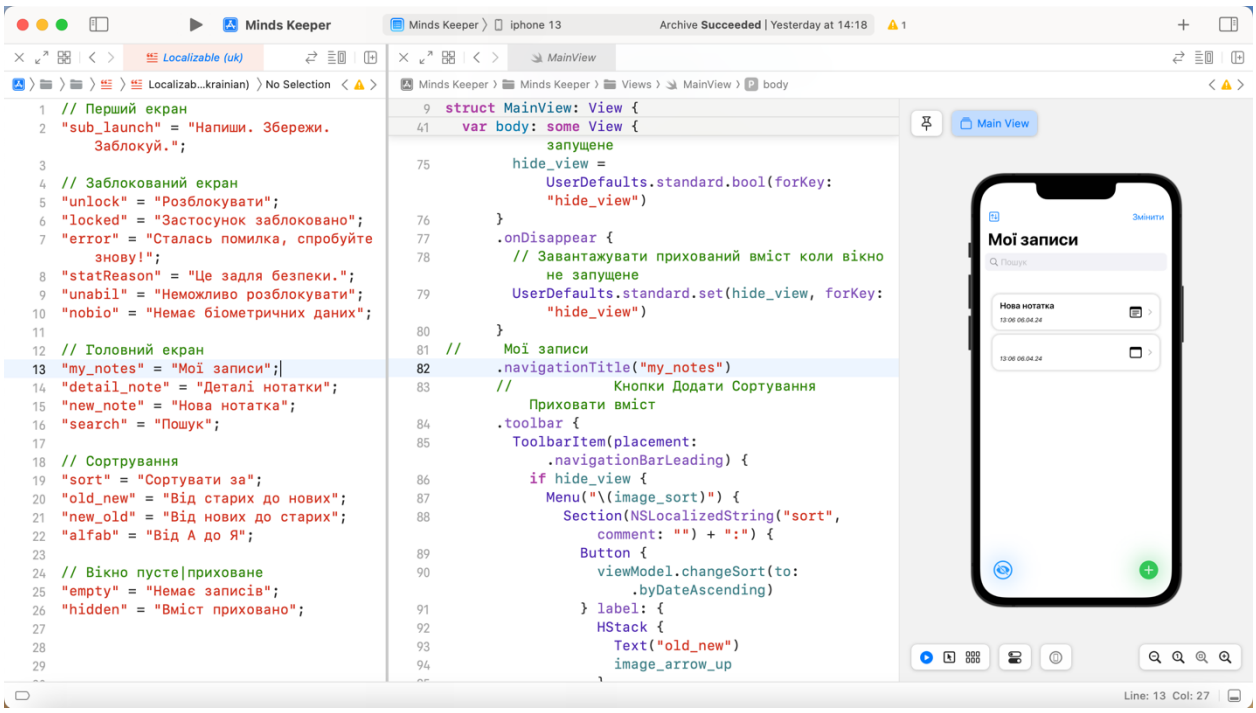

Рисунок 3.4 – Локазізація додатку

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування додатку написаним SwiftUI це визначення цільової системи iOS в даному випадку iOS 16.1, створенням архіву тестування Debug та запуску на віртуальному пристрої це може бути так як і iPhone та і iPad наведено у [17]. На рисунку 3.5 представено ієрархію запуску додатку, використання пам’яті центрального процесора та пристрою, використання енергії та інтернет з’єднаня.

Основні дії при тестуванні:

– очищення кешу та папки збірки;

– встановлення потрібної версії iOS;

– архівація проекту для тестування;

– запуск додатку на віртуальний або фізичний пристрій;

– перевірка всіх функцій та локалізації.

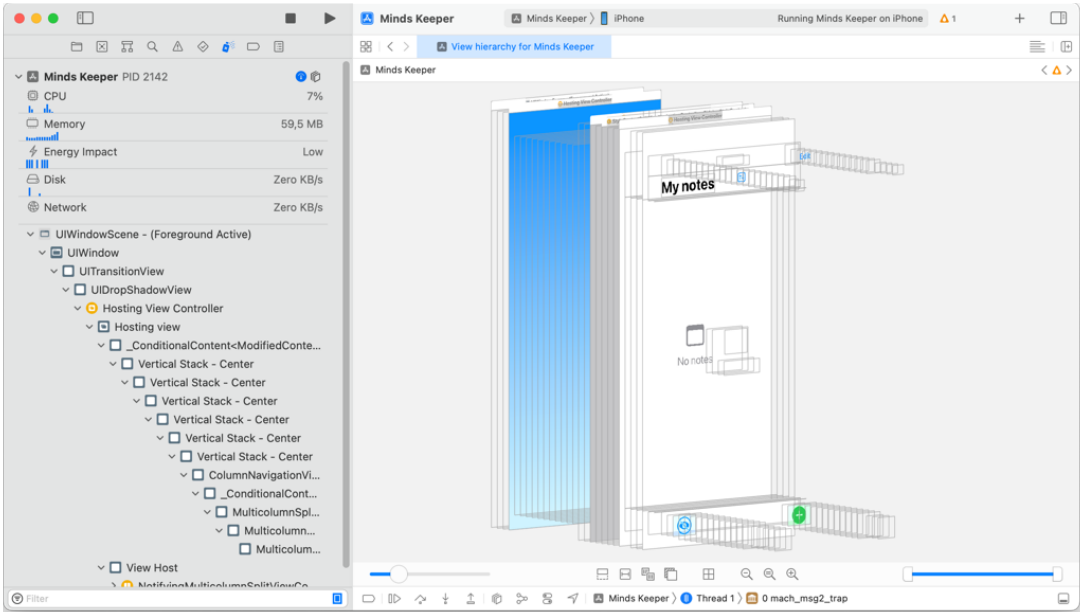

Рисунок 3.5 – Тестування додатку

Unit-тести – це стандартні інструменти для написання unit-тестів для коду, вони використовую фреймворки, XCTest та тести для окремих компонентів і їх коректність в ізоляції.

UI-тести використовуються для перевірки коректності роботи інтерфейсу.

Previews – це перегляд інтерфейсу користувача в реальному часі прямо в коді. Використання функціоналу для швидкої перевірки зовнішнього вигляду та поведінки інтерфейсу без запуску всього додатку.

Debugging використовуються як вбудовані інструменти налагодження Xcode, для виявлення та виправлення помилок у коді SwiftUI. Xcode надає інструменти, консоль відладки, точки зупинки, інспектори.

Performance Profiling надає інструменти профілювання продуктивності Xcode, для виявлення та усунення проблем з продуктивністю коду SwiftUI, якщо додаток працює повільно або неефективно.

3.3 Розробка інсталяційного пакета

Створення інсталяційного пакету є важливим адже для встановлення на пристрій проект мусить бути у певному форматі, щоб система розуміла і

розгорнула та запустила, в даному випадку розширенням буде (.ipa) для платформи iOS та iPadOS надано у [18].

Info.plist – це файл у форматі XML, містить інформацію про конфігурацію додатка, таку як ідентифікатор додатка, версія, мінімальна версія iOS або macOS, необхідні дозволи та права на приватні дані, а також іншу метаінформацію представлено на рисунку 3.6.

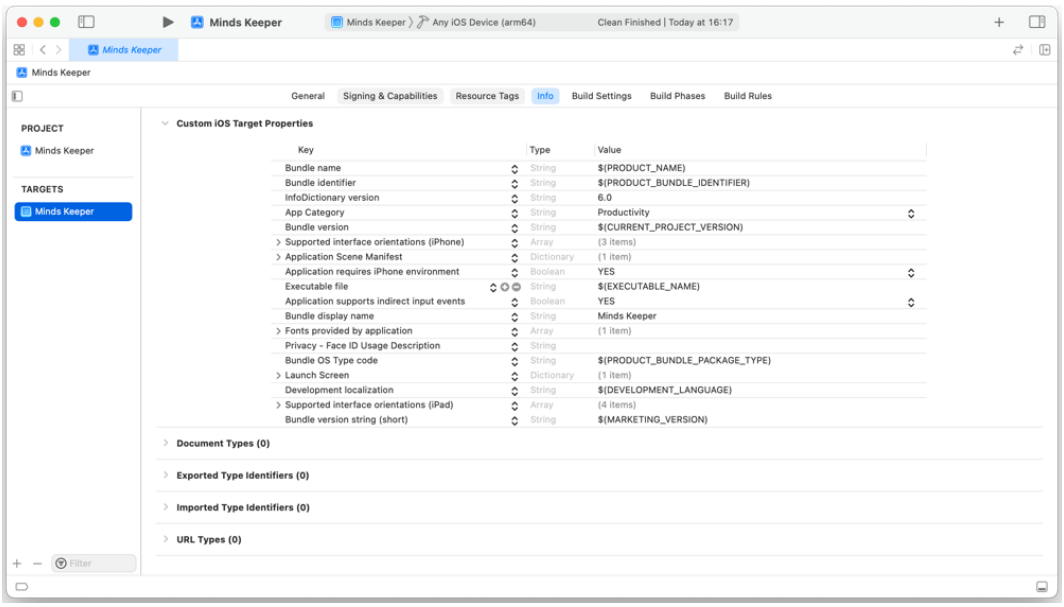

Рисунок 3.6 – Інформаційний лист

Для функціонування FaceID додається опис «Privacy – Face ID Usage Description». Інформація щодо назви додатку, орієнтації та локалізації додається автоматично.

Процедура створення інсталяційного пакету починається з налаштуванням вище сказаного інформаційного листа та створенням архіву. Так в Xcode не доступно створити файли з розширенням (.ipa) не завантажуючи проект в AppStore, можливо створити вручну.

Етапи створення:

– архівування всього проекту;
– перевірка цілісності всіх файлів;

– архівування файлу з розширенням (.app) та перейменовування його у Payload;

– зміна розширення (.zip) у (.ipa).

Цей метод не є офіційним але єдний, задля розповсюдження додатку на будь-який пристрій. Список файлів які міститиме архів представлений на рисунку 3.7.

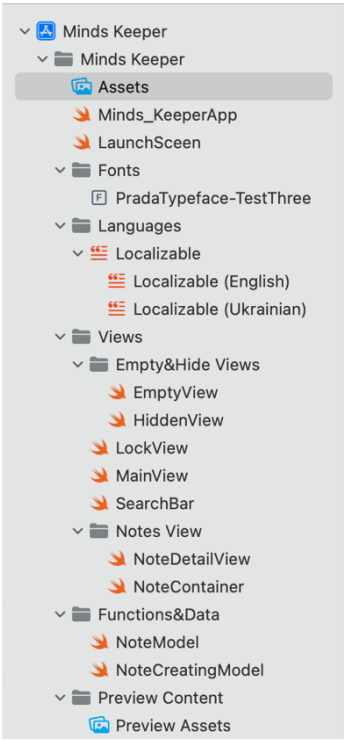

Рисунок 3.7 – Ресурси додатку

Метод встановлення застосунків на iPhone та iPad проводенься через основний магазин застосунків AppStore. Компанія Apple забороняє завантажувати та встановлювати їх через не офіційні джерела. Одже, додаток містить інформацію про ствоврення, налаштування, вміст пакетів/бібліотек та найголовніше – сертифікати.

Для встановлення будь-якого програмного продукту (.ipa) використано затосунок «Sideloadly» зобнажено на рисунку 3.8. Перш за все – додавання активного акаунту AppleID. Наступним етапом стане вибір пристрою на який потрібно встановити. Заключним етепом вибір додатку формату (.ipa). Після

натискання клавіші «Start» відбудеться зв’язок з сервером, що в свою чергу створить безкоштовний та тимчасовий профіль/сертифікат розробника терміном на 7 днів, представлено на рисунку 3.8.

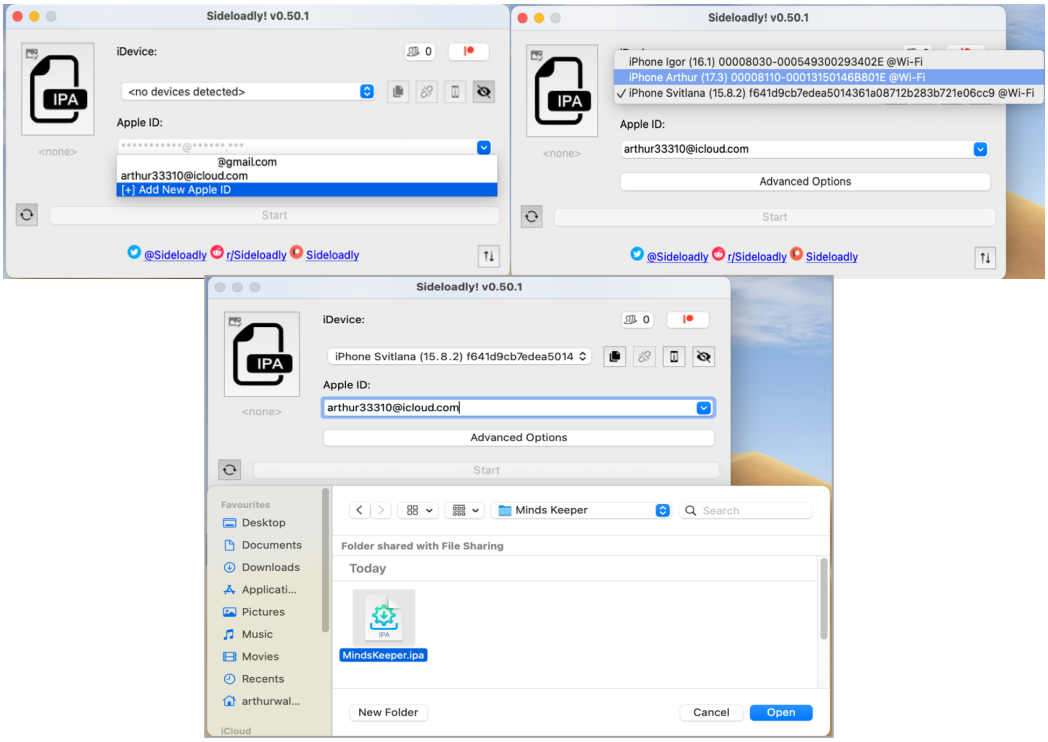

Рисунок 3.8 – Встановлення додатку формату (.ipa) Sideloadly

Після процесу встановлення, іконка додатку з’явиться на робочому столі iOS, однак не буде доступною для запуску через невідомого розробника. Відповідно до цього потрібно перейти в налаштування пристрою у розділ «Загальні – VPN і керування пристроєм» в якому відображатиметься підрозділ «Програми розробника». Обираємо розробника та натискаємо клавішу «Довіряти» представлено на рисунку 3.9, додаток доступний 7 днів з моменту підтвердження та встановлення.

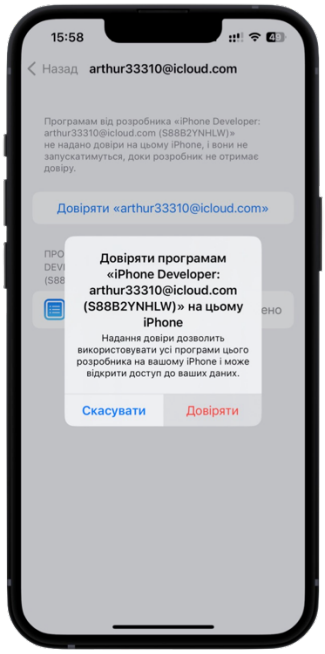

Рисунок 3.9 – Підтвердження для запуску додатку

Висновоки до розділу 3

У практичній реалізації об’єкта проєктування для додатку нотаток на мові SwiftUI було досягнуто важливої гнучкості та простоти управління. Модель даних була чітко визначена, що дозволило ефективно керувати станом нотаток та їх атрибутами. Використання властивостей об’єктів узгодження дозволило забезпечити консистентність даних та спростило процес взаємодії з користувачем. Крім того, додаток був розроблений з урахуванням принципів інтерфейсу користувача SwiftUI, що забезпечило високу швидкість розробки та стабільність функціонування. У підсумку, реалізація об’єкта проєктування у додатку нотаток мовою SwiftUI відкрила шлях до створення додатків, які поєднують в собі зручність у використанні з високою функціональністю.

Висновок щодо розробки інсталяційного пакета для SwiftUI полягає в тому, що цей процес виявляється ключовим для забезпечення ефективного розгортання та використання SwiftUI проєктів. Розробка інсталяційного пакета дозволяє забезпечити зручний та стандартизований спосіб розповсюдження програмного забезпечення серед користувачів.

Під час розробки інсталяційного пакета для SwiftUI важливо враховувати особливості цієї технології, такі як її інтеграція з іншими платформами Apple, такими як iOS, macOS, watchOS та tvOS. Крім того, потрібно враховувати вимоги безпеки та ефективності під час розгортання програм.

Основними кроками при розробці інсталяційного пакета є підготовка документації, встановлення відповідних залежностей та налаштування параметрів проєкту. Також важливо забезпечити механізми автоматизації тестування та випробування пакета перед його розповсюдженням.

Розробка інсталяційного пакета для SwiftUI є важливим етапом у процесі створення програмного забезпечення для платформ Apple. Цей процес вимагає уважності до деталей та використання кращих практик розробки програмного забезпечення для забезпечення надійності, безпеки та ефективності програми.

ВИСНОВКИ

Розробка додатку «Minds Keeper» була успішною завдяки використанню мови програмування Swift та фреймворку SwiftUI, які забезпечили швидку та ефективну розробку інтуїтивного і користувацького інтерфейсу. Використання SwiftUI дозволило легко налаштувати різноманітні візуальні елементи, щоб забезпечити зручність та естетичний вигляд додатку.

Здійснено аналіз аналогічних програмних продуктів, їхньої функціональності, відгуків користувачів та магазину додатків. Зокрема розглянуто такі аналоги як: «Notability: Notes», «PDF, Craft: Write docs», «AI editing, Writer – Notes», «Lists, Editor», «Noted – Record, Transcribe».

Проведено аналіз технологій створення додатків мовою SwiftUI використовуючи офіційні посібники та інструкції.

Було визначено вимоги до розроблюваного продукту, зокрема можливість додавання зображень та файлів, функції пошуку, захист паролем, висока продуктивність. Розроблено функціональну схему додатку.

Методами та засобами для розробки стали стандартні інструменти компанії Apple Xcode.

Реалізація програмного продукту була проведена за допомогою інструментів Xcode, бібліотек SwiftUI, Foundation, Combine та LocalAuthentication та методів (append()), (update()) та (UserDefaults).

Проведення тестування відбувалось виконанням тестів стандартною програмою Xcode та фізичним користуванням додатку, задля точності виконання функцій.

Було створено інсталяційний пакет. Архівування проекту відбувався зі створенням звичайного архіву з розширенням (.xcodearchive).

Під час розробки було звернуто увагу на дотримання найкращих практик програмування та забезпечення якості коду. Такий підхід дозволив створити стабільний та надійний додаток, який задовольняє потреби користувачів.

У результаті розробки «Minds Keeper» став унікальним інструментом для зберігання та організації нотаток на iOS, забезпечуючи зручний інтерфейс, надійність та розширені можливості для користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Apple iOS: що це таке? URL: http://surl.li/tikny (дата звернення: 30.04.2024).

2. Підручник SwiftUI: створення програм за допомогою нової платформи інтерфейсу користувача від Apple. URL: http://surl.li/ugzhb (дата звернення: 30.04.2024).

3. Документація SwiftU. URL: http://surl.li/ugzhw (дата звернення: 30.04.2024).

4. Notability: Notes, PDF. URL: https://notability.com (дата звернення: 30.04.2024).

5. Craft: Write docs, AI editing. URL: https://www.craft.do (дата звернення: 30.04.2024).

6. Writer – Notes, Lists, Editor. URL: https://shorturl.at/3mfWl (дата звернення: 30.04.2024).

7. Noted – Record, Transcribe. URL: https://www.notedapp.io (дата звернення: 30.04.2024).

8. Основні вимоги до Junior iOS Developer. URL: http://surl.li/ugzis (дата звернення: 05.05.2024).

9. Про програмне забезпечення XCODE. URL: http://surl.li/ugzif (дата звернення: 05.05.2024).

10. Бібліотека Foundation. URL: http://surl.li/ugzke (дата звернення: 05.05.2024).

11. Бібліотека Combine. URL: http://surl.li/ugzjv (дата звернення: 05.05.2024).

12. Бібліотека Local Authentication. URL: http://surl.li/ugzjk (дата звернення: 05.05.2024).

13. Комплексний посібник зі створення дизайну інтерфейсу користувача для iOS додатків. URL: http://surl.li/ugziy (дата звернення: 10.05.2024).

14. Ключові складові функціоналу мобільного додатка. URL: http://surl.li/ugzjd (дата звернення: 11.05.2024).

15. Створюємо додаток: завдання, етапи та терміни виконання. URL: http://surl.li/ugzki (дата звернення: 11.05.2024).

16. Розробка мобільних додатків для IOS. URL: http://surl.li/ugzku (дата звернення: 12.05.2024).

17. Основні етапи тестування мобільних додатків. URL: http://surl.li/ugzla (дата звернення: 12.05.2024).

18. Creating Real Device .ipa Files for Appium and XCUITest. URL: http://surl.li/ugzlf (дата звернення: 12.05.2024).

ДОДАТКИ

Додаток А

Вікно запуску

```
import SwiftUI
struct LaunchScreen: View {
  //     Світла та темна тема
  @Environment(\.colorScheme) var colorScheme
  //     Назва в залежності з назвою додатку
  public var launch_title =
(Bundle.main.infoDictionary?["CFBundleDisplayName"] as? String)
  //     Розмір заголовка
  var size_title: CGFloat = 35
  //    Розмір підзаголовка
  var size_subtitle: CGFloat = 15
  //     Кольори і Градієнти
  static let blue_dark = Color("blue_dark")
  static let blue_light = Color("blue_light")
  public var grad_light = Gradient(colors: [blue_light, .white,
blue_light])
  public var grad_dark = Gradient(colors: [blue_dark, .black, blue_dark])
  //    Зображення
  var image_wow: Image = Image("Wow")
  var image_expressed: Image = Image("Expressed")
  var image_note: Image = Image("NoteBook")
  //    Розмір зображень
  var img_size: CGFloat = 250
  var img_size_note: CGFloat = 50
  var body: some View {
    VStack {
      VStack {
        image_note
          .resizable()
          .frame(width: img_size_note, height: img_size_note)
          .scaledToFill()
        // Заголовок
        Text(launch_title!)
          .font(.custom("PradaTypeface-TestThree", size: size_title))
          .foregroundColor(colorScheme == .light ? Color("blue_dark") :
Color("blue_light"))
          .bold()
          .shadow(color: .cyan, radius: 15)
          .padding(10)
        // Підзаголовок
        Text("sub_launch")
          .font(.custom("PradaTypeface-TestThree", size: size_subtitle))
          .font(.footnote)
          .bold()
      }
      .overlay(
        VStack {
          image_wow
```

```
            .resizable()
            .frame(maxWidth: img_size, maxHeight: img_size)
            .scaledToFill()
            .position(x: 180, y: -200)
            .rotationEffect(.degrees(-20))

          image_expressed
            .resizable()
            .frame(maxWidth: img_size, maxHeight: img_size)
            .scaledToFill()
            .position(x: 300, y: 250)
            .rotationEffect(.degrees(10))
        })
    }
    .frame(maxWidth: .infinity, maxHeight: .infinity)
    .ignoresSafeArea()
    .background(colorScheme == .light ? grad_light : grad_dark)
}
```

Додаток Б

Функція розблокування

```
// Бібліотека використання аунтентифікації
import LocalAuthentication
import SwiftUI
struct LockView: View {
  @State private var unlocked = false
  @State private var status_text = NSLocalizedString("locked", comment:
"")
  @State private var status_img = Image(systemName: "lock")
  // Розміри зображень
  var size_img: CGFloat = 75
  var size_txt: CGFloat = 20
  var min_width_stat: CGFloat = 100
  //    Колір і градієнт
  static let color1 = Color("blue_dark")
  static let color2 = Color("blue_light")
  let gradient = Gradient(colors: [color1, color2])
  var body: some View {
    VStack {
      VStack {
        // Якщо додаток розблокований
        if unlocked {
          VStack {
            // Показує головний екран
            MainView()
          }
        }
        // Якщо додаток не розблокований
        else {
          VStack {
            VStack {
              // Статус
              Text(status_text)
                .font(.system(size: size_txt))
                .bold()
                .padding(15)
            }
            Spacer()
            VStack {
              // Кнопка розблокування
              Button(action: { authenticate() }) {
                VStack {
                  // Іконка замка
                  status_img
                    .font(.system(size: size_img))
                    .padding(.bottom, 5)

                  Text("unlock")
                }
```

```
                }
            }
            Spacer()
        }
      }
    }
    // Спроба розблокування щойно запущеного додатку
    .onAppear(perform: authenticate)
  }
  .frame(maxWidth: .infinity, maxHeight: .infinity)
  .background(gradient)
}
```

Додаток В

Головне вікно

```
import SwiftUI
struct MainView: View {
  //    Основний клас
  @StateObject private var viewModel = NoteCreatingModel()
  //    Вміст нотатки
  @State private var newNoteContent = ""
  //    Вибір елементу в списку
  @State private var selectionSet = Set<UUID>()
  //    Функція якщо вибраний елемент
  @State var isSelecting = false
  //    Пошук за вмістом
  @State private var searchText = ""
  //    Збереження вікна "приховано" після закриття додатку
  @AppStorage("hide_view") private var hide_view = false
  //    Зображення
  var image_search: Image = Image(systemName: "magnifyingglass.circle")
  var image_dismiss: Image = Image(systemName: "multiply.circle.fill")
  var image_add: Image = Image(systemName: "plus.circle.fill")
  var image_remove: Image = Image(systemName: "trash.circle.fill")
  var image_hide: Image = Image(systemName: "eye.slash.circle")
  var image_show: Image = Image(systemName: "eye.circle.fill")
  var image_sort: Image = Image(systemName: "arrow.up.arrow.down.square")
  var image_arrow_up: Image = Image(systemName: "arrow.up")
  var image_arrow_down: Image = Image(systemName: "arrow.down")
  //    Розміри та інші налаштування
  var size_image_top: CGFloat = 20
  var size_image_bottom: CGFloat = 30
  var blur: CGFloat = 4.0
  var shadow: CGFloat = 25
  var body: some View {
    NavigationView {
      ZStack {
        //                Вікно прихований вміст
        if !hide_view {
          HiddenView()
            .zIndex(1)
        }
        VStack {
          //                    Пошукова строка
          if !viewModel.notes.isEmpty {
            SearchBar(text: $searchText)
          }
          if !viewModel.notes.isEmpty {
            //                        Перегляд списку нотаткок
            NoteContainer(
              viewModel: viewModel,
              selectionSet: $selectionSet,
              isSelecting: $isSelecting,
```

```
            searchText: searchText)
          } else {
            //                          Якщо немає нотаток, вікно немає
нотаток
            EmptyView()
          }
        }
        .blur(radius: hide_view ? 0 : blur)
        .disabled(!hide_view)
      }
      .onAppear {
        // Завантажувати прихований вміст коли вікно запущене
        hide_view = UserDefaults.standard.bool(forKey: "hide_view")
      }
      .onDisappear {
        // Завантажувати прихований вміст коли вікно не запущене
        UserDefaults.standard.set(hide_view, forKey: "hide_view")
      }
      .navigationTitle("my_notes")
      //            Кнопки Додати Сортування Приховати вміст
      .toolbar {
        ToolbarItem(placement: .navigationBarLeading) {
          if hide_view {
            Menu("\(image_sort)") {
              Section(NSLocalizedString("sort", comment: "") + ":") {
                Button {
                  viewModel.changeSort(to: .byDateAscending)
                } label: {
                  HStack {
                    Text("old_new")
                    image_arrow_up
                  }
                }
                Button {
                  viewModel.changeSort(to: .byDateDescending)
                } label: {
                  HStack {
                    Text("new_old")
                    image_arrow_down
                  }
                }
                Button("alfab") {
                  viewModel.changeSort(to: .byContent)}
              }
            }
          }
        }
        ToolbarItem(placement: .navigationBarTrailing) {
          if !selectionSet.isEmpty && hide_view {
            Button {
              deleteSelectedNotes()
```

```
            } label: {
              image_remove
                .font(.system(size: size_image_top))
                .foregroundColor(Color.red)
            }
          }
        }
        ToolbarItem(placement: .navigationBarTrailing) {
          if hide_view {
            EditButton()
          }
        }
        ToolbarItem(placement: .bottomBar) {
          Button {
            self.hide_view.toggle()
          } label: {
            VStack {
              hide_view
                ? image_hide
                  .font(.system(size: size_image_bottom))
                  .foregroundColor(Color("blue_dark"))
                : image_show
                  .font(.system(size: size_image_bottom))
                  .foregroundColor(Color("blue_dark"))
            }
          }
          .shadow(color: .blue, radius: shadow)
        }

        ToolbarItem(placement: .bottomBar) {
          if hide_view {
            Button {
              viewModel.add()
            }

            label: {
              image_add
                .font(.system(size: size_image_bottom))
                .foregroundColor(Color.green)
            }
            .shadow(color: .green, radius: shadow)
          }
        }
      }
    }
  }
  // Функція видалити всі вибрані зі списку нотатки
  func deleteSelectedNotes() {
    viewModel.removeSelectedNotes(selection: selectionSet)
    selectionSet.removeAll()
    viewModel.save()}}
```