МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програмного забезпечення

«Додаток ведення нотаток» для Android

Виконала: студентка групи 3157ст

_____ _____ Поліщук І.О.

(підпис, ПІБ)

Керівник роботи:

доцент каф. ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

_____ Макарова Л.М

(підпис, ПІБ)

Миколаїв – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Поліщук Ірині Олександрівні _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення «Додаток ведення нотаток» для Android _____

Керівник роботи Макарова Лідія Миколаївна _____

Затверджені наказом ректора №256-уч від «11» травня 2022 р.

2. Термін подання роботи: 10.06.2022 р. _____

3. Вихідні дані по роботі: _____
_____

4. Перелік питань, що належать до розробки (найменування розділів)_____
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Опис предметної галузі та постановка задачі; Проєкт програмного забезпечення (ескізний, технічний та робочий); Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів _____
Титульний слайд; Аналіз предметної галузі; Аналіз готових рішень; Постановка задачі; Діаграма варіантів використання; Концептуальна модель; Діаграма переходів станів; Інтерфейс майбутнього додатку; Логічна модель даних; Частина фізичної моделі БД; Діаграма класів; Засоби розробки; Результати розробки; Висновки; Заключний слайд._____
_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 11.05.2022 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу Вступ	24.03.2022	ПП*
2.	Опис та аналіз предметної галузі	28.03.2022	ПП*
3.	Постановка задачі	31.03.2022	ПП*
4.	Ескізний проєкт програмного забезпечення	27.04.2022	
5.	Технічний проєкт програмного забезпечення	06.05.2022	
6.	Робочий проєкт програмного забезпечення	13.05.2022	
7.	Підготовка розділу Результати розробки програмного забезпечення	17.05.2022	
8.	Підготовка розділу з охорони праці	18.05.2022	ПП*
9.	Підготовка розділу Висновки	19.05.2022	
10.	Оформлення списку використаних джерел та додатків	20.05.2022	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	10.06.2022	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	12.06.2022	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	14.06.2022	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	24.06.2022	

* - за результатами переддипломної практики (ПП), яка була з 21.03.2022 до 08.05.2022 р.

Студент _____ Поліщук І.О.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці програмного забезпечення «Додаток ведення нотаток» для Android», котре дозволить користувачу успішно вести особисті плани згідно раціонально-використаного часу.

Дана кваліфікаційна робота передбачає розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов.

У кваліфікаційній роботі міститься аналіз та опис предметної галузі за темою кваліфікаційної роботи, постановка задачі, проект програмного забезпечення, а також результати розробки та розділ з охорони праці.

Кваліфікаційна робота викладена на 87 сторінках друкованого тексту та містить 28 рисунків, 30 таблиць, 5 додатків та список використаних джерел з 18 найменувань.

ABSTRACT

Qualifying work is dedicated to the development of software "Additional note" for Android, which will allow the user to successfully manage personal plans according to the rational use of time.

This qualification work involves solving a practical problem of software engineering, which is characterized by complexity and uncertainty of conditions.

The qualification work contains an analysis and description of the subject area on the topic of qualification work, problem statement, software project, as well as the results of development and the section on labor protection.

The qualification work is presented on 87 pages of printed text and contains 28 figures, 30 tables, 5 appendices and a list of used sources of 18 titles.

ЗМІСТ

ВСТУП

Сучасна людина щодня виконує безліч справ, для ефективного використання свого часу необхідно заздалегідь розпланувати майбутні справи. Нотатник - незамінний атрибут успішної людини. Звичка записувати і планувати справи міцно зміцнилася в нашому житті, оскільки абсолютно кожна людина робить різні замітки і нагадування.

Ні для кого не секрет, що для успішного бізнесу вкрай важливо ефективно розподіляти свій час. Будь-яка успішна компанія приділяє особливу увагу здійсненню планування і організації роботи співробітників.

Для вирішення подібних завдань існують цілі системи, що дозволяють максимально оптимізувати роботу фірми. Це дуже зручно і ефективно.

Однак, навіть в повсякденних, простих справах часто необхідно планувати власний час – це дозволяє максимально раціонально розподілити час для вирішення поточних справ, завдяки чому залишається більше часу на відпочинок. Також різні нагадування допомагають не забути виконати заплановану справу.

Саме тому на сьогоднішній день так популярні різні сервіси для планування часу – web-сервіси, десктопні програми і мобільні додатки.

Підсумовуючи вищевикладене, нотатник є важливим завданням кожної людини. До того ж, необхідно в будь-який момент мати доступ до перегляду запланованих справ. Завдяки сучасним девайсам людина стала мобільною і безперешкодно справляється з довільними задачами, оскільки їй легко контролювати не лише своє переміщення, свою діяльність, а й контролювати власний час. В даний час людина знаходиться з смартфоном, тому нотатник у вигляді мобільного додатка є найкращим вибором для контролю справ і коректно розподіленого часу.

В кваліфікаційній роботі представлено опис автоматизації роботи нотатника для ОС Android. Дана робота передбачає аналіз та опис додатку, постановку задачі, технічне завдання до розробки програмного забезпечення, проект програмного забезпечення "Додаток ведення нотаток" для Android, результати розробки програмного забезпечення "Додаток ведення нотаток" для Android.

Дана кваліфікаційна робота передбачає розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробку програмного зебезпечення "Додаток ведення нотаток" для Android.

1 АНАЛІЗ РОБОТИ ДОДАТКА ВЕДЕННЯ НОТАТОК ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз та опис роботи додатку ведення нотаток

Предметною областю даного ПЗ є додаток для смартфону під ОС Android. Дане ПЗ – інструмент для успішного ведення особистих планів згідно раціонально-використаного часу.

Нотатник – це книга, зошит, пристрій, у який людина записує події, що відбуваються щодня і плани на майбутнє.

Додаток відноситься до прикладного програмного забезпечення і призначений для накопичення інформації користувача, а потім оперативного пошуку по ній, організації справ і контролю за їх виконанням, відслідковування визначених користувачем подій. Являється однією із форм персонального нотатнику.

Функції нотатника пов’язані із забезпеченням роботи наступних підрозділів:

- управління нотатками;
- управління контактами;
- управління папками;
- перегляд календаря;
- додання тексту і часу нагадування;
- управління кошиком;
- налаштування додатку (інтерфейс).

Сфери застосування і вимоги до нотатників досить різноманітні, тому існують різні версії, які можуть не містити деяких функцій, вказаних вище, проте мати певні специфічні доповнення, необхідні конкретній категорії користувачів.

Мобільний додаток являє собою ПЗ, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях. Вони, як правило, доступні через сервіси

розповсюдження, унікальні для кожного виробника під операційну систему, такі як Apple App Store, Google Play, Windows Phone Store тощо.

1.2 Аналіз існуючого аналогічного програмного забезпечення

TickTick є крос-платформенним додатком-планувальником, який допоможе зберігати всі поточні справи, а також синхронізовувати їх за допомогою хмарного сховища [1]. TickTick допомагає організувати своє життя в більш простий спосіб, завдяки зручності використання часу (див. рис. 1.1).

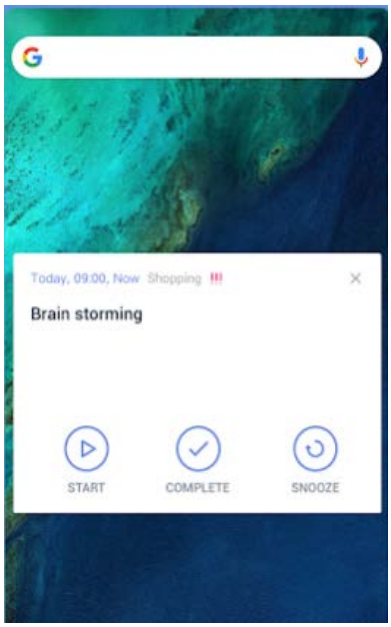

Рисунок 1.1 – Головне вікно додатку «TickTick» [1]

TickTick безкоштовний, але є можливість перейти на Професійну версію для повного доступу до функцій преміум-класу [1].

Особливості:

- синхронізація завдань між усіма пристроями;
- віджет для швидкого доступу;
- гнучні налаштування завдань які повторюються;

- створення списку;
- поділитись переліком завдань для співпраці;
- додати завдання за допомогою електронної пошти;
- додати вкладення до завдань;
- інтеграція з додатком календар;
- класифікація завдань за допомогою міток;

Більше функцій щоб допомогти тобі бути цілеспрямованим і більш продуктивним:

- часове назадування і нагадування за мізцезнаходженням;
- способи сортування (за послідовністю/датою/ім’ям/пріоритетом);
- додати нотатки/коментарії до завдань;
- групове редагування завдань.

Дизайн додатку представлений на рисунку 1.2.

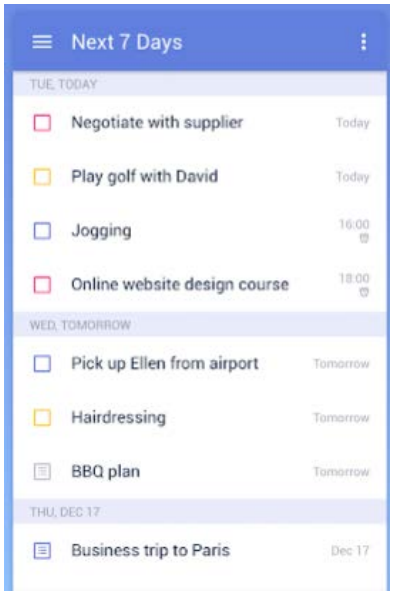

Рисунок 1.2 – Дизайн додатку «TickTick» [1]

Переваги:

- зручний в використанні;
- можливість змінювати колір дизайну;
- зрозумілий інтерфейс;

- змога сортувати списки;
- практичний;
- не містить рекламу.

Недоліки:

- не працює без створення свого акаунту;
- не використовує соцмережі.

Додаток Writeaday створено спеціально для особистих записів (див.рис. 1.3). Кожен день туди можна вносити текстові замітки з прийшли в голову думками або подіями. Каталогізувати написане можна по системі хештегів, які користувач може створювати сам в необмеженій кількості [2].

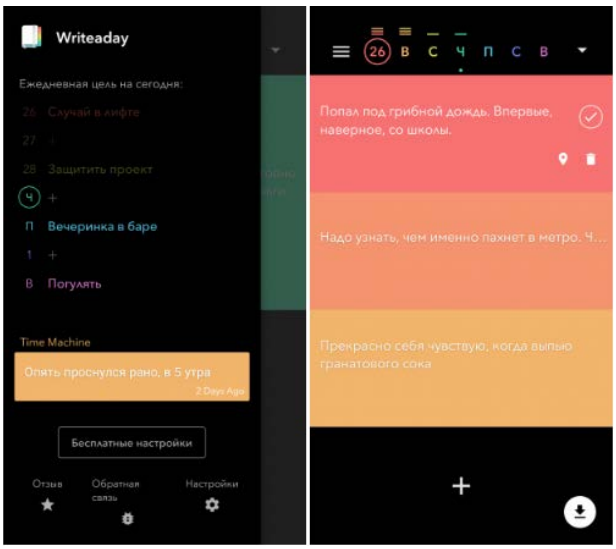

Рисунок 1.3 – Дизайн додатку «Writeaday» [2]

Додаток заохочує старанне ведення щоденника. Дні тижня в ньому пофарбовані в кольори спектру, а кожна додана запис робить їх яскравішими. До кінця тижня, зробивши 30-40 записів, можна побачити в плані на тиждень справжню веселку.

Кожному дню можна привласнити мета. Позначка буде служити «заголовком» дня і дозволить або сконцентруватися на будь-якої чіткої задачі, або запам'ятати день по якійсь ключовій події [2].

Функції:

- принципово унікальний спосіб швидко писати протягом дня.
- організуйте записи журналу, використовуючи #tags ('#' перед будь-яким словом);
- перегляд записів у організованому віджеті;
- резервне копіювання на власний персональний диск Google;
- показує запис із минулого;
- блокнот із замком (кодування чи відбиток пальця).

Переваги:

- зручний в використанні;
- кожен день тижня має власний колір;
- яскравий дизайн;
- зрозумілий інтерфейс;
- мість систему хештегів;
- практичний.

Недоліки:

- без можливості змінювати колір дизайну;
- якщо більше 10 заміток в один день – накладається один на одного текст;
- містить рекламу.

1.3 Постановка задачі для розробки програмного забезпечення "Додаток ведення нотаток" для Android

Виходячи з аналізу предметної області, необхідно розробити програмне забезпечення "Додаток ведення нотаток" для Android. Користувачем додатку буде власник, на мобільному телефоні якого встановлено даний додаток, який має повний доступ до всіх функцій. У ньому ставитиметься конкретне завдання чи нотаток,

підбиратиметься час нагадування, потрібна папка, контакт та ставити позначку «Важливі».

Функції:

- управління нотатками;
- управління контактами;
- управління папками;
- перегляд подій в місяці;
- додання тексту і часу нагадування;
- управління кошиком;
- налаштування додатку (інтерфейс).

Вхідні дані:

- заголовок нотатку;
- текст нотатку;
- дата і час нагадування нотатку;
- дані контактів;
- назва папки.

Вихідні дані:

- список нотатків;
- список контактів;
- список папок з нотатками;
- інформація про нотаток;
- дані з нагадуваннями про нотаток.

Розробка додатку виконується на основі технічного завдання, документа, в якому максимально докладно і точно вказані вимоги до додатку. Технічне завдання розробки нотатника наведено в додатку А.

2 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «ДОДАТОК ВЕДЕННЯ НОТАТОК» ДЛЯ ANDROID

Проектування програмного забезпечення – це процес вирішення задач та планування для створення програмного рішення. Для проектування програмної системи була обрана об’єктно-орієнтована парадигма, яка здійснюється за допомогою UML [3] , уніфікованої мови для розробки ПЗ.

Практика розробки програмного забезпечення передбачає для проектування такі типи діаграм:

- контекстна діаграма;
- діаграма варіантів використання;
- концептуальна модель БД;
- діаграма переходу станів;
- логічна модель;
- діаграма послідовності;
- діаграма пакетів;
- діаграма класів;
- фізична модель.

2.1 Ескізний проект програмного забезпечення

Проектування будь-якого програмного забезпечення починається з розробки ескізного проекту, у якому представляються результати зовнішнього проектування програмного забезпечення.

При проектуванні даного програмного забезпечення було обрано мову

моделювання UML.

UML може бути застосовано на всіх етапах розробки прикладних програм. Різні види діаграм, які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису програмних систем.

2.1.1 Контекстна діаграма

Першою при проектуванні програмної системи розробляється контекстна діаграма.

Контекстна діаграма представляє вимоги до системи на самому верхньому рівні – рівні взаємодії з оточенням.

Контекстна діаграма для програмної системи представлена на рисунку 2.1.

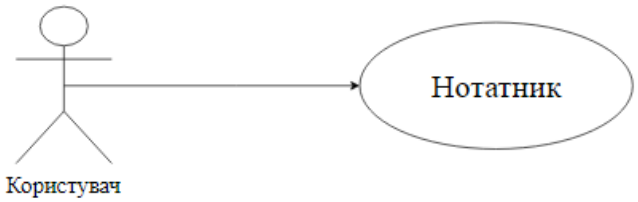

Рисунок 2.1 – Контекстна діаграма програмної системи

2.1.2 Діаграма варіантів використання

Діаграма варіантів використання – діаграма, яка відображає відношення між акторами та прецедентами в системі. Дана діаграма визначає функціональну поведінку системи [4].

Діаграма варіантів використання для програмної системи представлена на рисунку 2.2.

Рисунок 2.2 – Діаграма варіантів використання для програмної системи

Розглянемо більш детально кожний варіант використання, надаючи опис потоків подій. Специфікації варіантів використання представлено в таблицях 2.1 – 2.7.

Таблиця 2.1 – Специфікація прецеденту «Робота з нотатками»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу додавати нотатки
Передумови	Користувач повинен обрати потрібний календарний день
Основний потік подій	Користувач ініціює додавання нового елементу списку . Система відкриває форму елементу. Користувач вводить дані та підтверджує збереження, система зберігає дані. Якщо обов'язкові поля не були не заповнені відбувається альтернативний потік
Альтернативний потік подій	Система пропонує ввести поля або інформація не буде збережена
Послідуюче	Не визначені

Таблиця 2.2 – Специфікація прецеденту «Перегляд нотатків за місяцем»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу переглядати календар
Передумови	Користувач повинен увійти в систему
Основний потік подій	Користувач ініціює взаємодію з календарем. Користувач обирає поточну дату
Альтернативний потік подій	Не визначено
Послідуюче	Виконання інших варіантів використання

Таблиця 2.3 – Специфікація прецеденту «Управління папками»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу додавати нотаток в потрібну папку
Передумови	Користувач повинен створити нотаток
Основний потік подій	Користувач ініціює додавання нотатку до папки
Альтернативний потік подій	Не визначено
Послідуюче	Виконання інших варіантів використання

Таблиця 2.4 – Специфікація прецеденту «Управління контактами»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу обирати потрібний контакт
Передумови	Користувач повинен створити нотаток
Основний потік подій	Користувач ініціює додавання контакту до нотатку. Користувач обирає потрібний контакт
Альтернативний потік подій	Не визначено
Послідуюче	Виконання інших варіантів використання

Таблиця 2.5 – Специфікація прецеденту «Додавання нагадування»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу додавати нагадування для нотатку
Передумови	Користувач повинен створити нотаток
Основний потік подій	Користувач ініціює додавання нагадування до нотатку. Користувач обирає потрібний час
Альтернативний потік подій	Не визначено
Послідуюче	Виконання інших варіантів використання

Таблиця 2.6 – Специфікація прецеденту «Управління кошиком»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу видаляти непотрібний нотаток в кошик
Передумови	Користувач повинен створити нотаток
Основний потік подій	Користувач ініціює видалення нотатку. Нотаток потрапляє до кошику
Альтернативний потік подій	Не визначено
Послідуюче	Виконання інших варіантів використання

Таблиця 2.7 – Специфікація прецеденту «Налаштування додатку»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу виконувати налаштування для додатку
Передумови	Користувач повинен запустити додаток і виконати рекомендації
Основний потік подій	Користувач виконує потрібні йому налаштування в додатку
Альтернативний потік подій	Не визначено
Послідуюче	Виконання інших варіантів використання

2.1.3 Концептуальна модель

Концептуальна модель – це модель, представлена безліччю понять і зв'язків між ними, що визначають смислову структуру розглянутої предметної області або її конкретного об'єкта [5].

ER-діаграма БД для програмної системи представлена на рисунку 2.3.

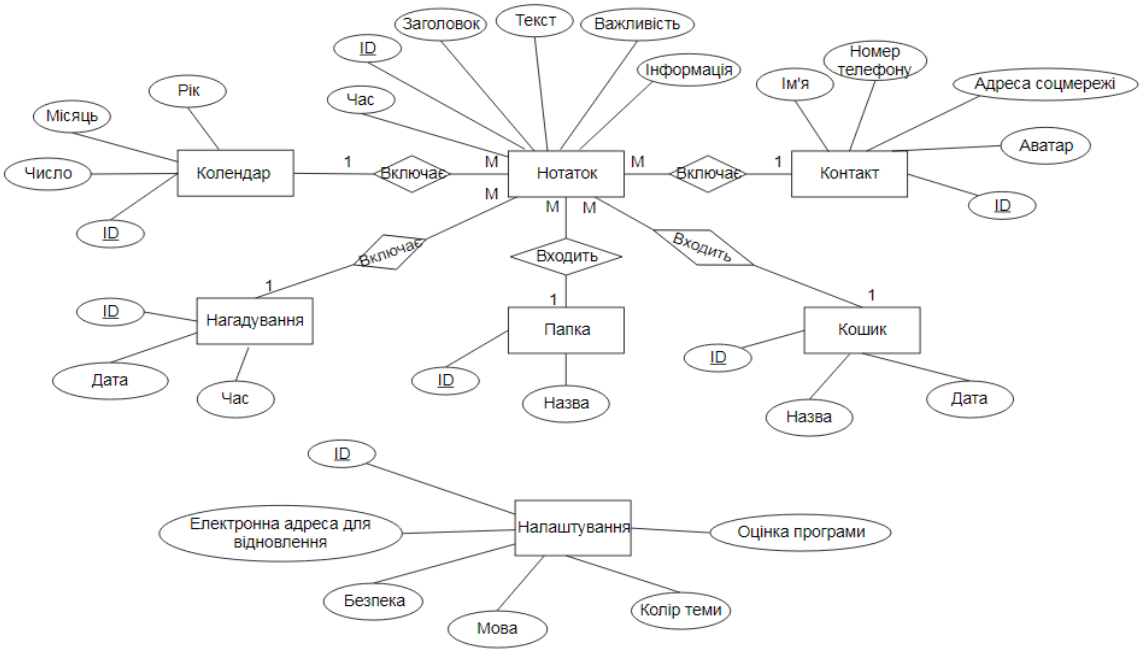

Рисунок 2.3 – ER-діаграма бази даних

Опис зв’язків між сутностями БД для програмної системи «Розробка додатку ведення нотаток для ОС Android» показаний у таблиці 2.8.

Таблиця 2.8 – Зв’язки між сутностями

Сутності	Тип зв’язку	Зміст зв’язку
1	2	3
Нотаток Календарний день	M:1	Декілька нотатків може бути заплановано в один календарний день
Контакт Нотаток	1:M	Один контакт може бути у декількох нотатках
Нагадування Нотаток	1:M	Одне нагадування може бути у декількох нотатках
Папка Нотаток	1:M	Одна папка може містити декілька нотатків
Кошик Нотаток	1:M	Один кошик може містити декілька нотатків

Таким чином, розроблено таблиці звязків між сутностями програмного забезпечення «Додаток ведення нотаток» для Android.

2.1.4 Діаграма переходів станів

Діаграма переходів станів демонструє поведінку розроблюваної програмної системи при отриманні управлінських дій [6]. Перехід між підсистемами здійснюватиметься через головне вікно, частина якого буде активна незалежно від обраної підсистеми і інших об’єктів конфігурації. Діаграма переходів станів для програмної системи представлена на рисунку 2.4.

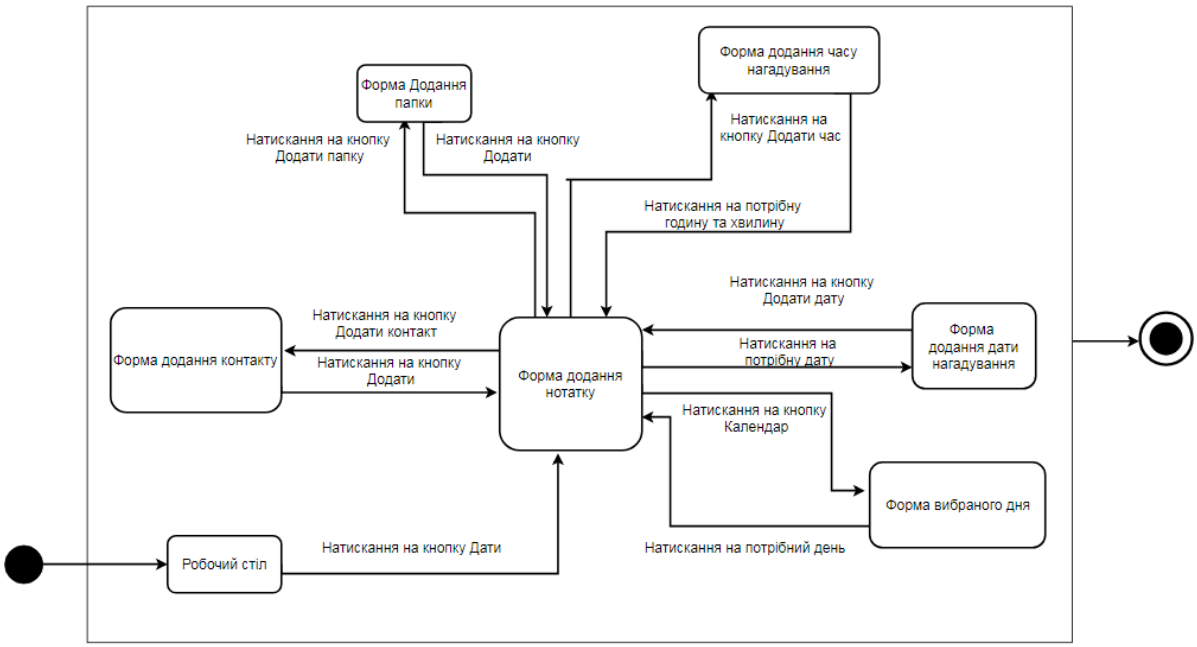

Рисунок 2.4 – Діаграма переходів станів для переходу між формами ПС

Таким чином, представлено діаграму переходів станів для переходу між формами програмного забезпечення «Додаток ведення нотаток» для Android

2.1.5 Проектування інтерфейсу

Інтерфейс користувача – засіб зручної взаємодії користувача з інформаційною системою; сукупність засобів для обробки та відображення інформації, максимально застосованих для зручності користувача [7].

Макети перегляду нотатку та список нотатків представлені на рисунках 2.5 – 2.9.

Користувач починає своє знайомство з графічним інтерфейсом головної сторінки програми. Шаблон форми головної сторінки зображений на рисунку 2.5.

Рисунок 2.5 – Форма головної сторінки для нотатку

Якщо натиснути кнопку «Додати», то користувач перейде на форму створення нотатку, яка зображена на рисунку 2.6. Ця форма дозволяє записати заголовок нотатку, текст нотатку, обрати папку в якій буде знаходитись нотаток контакт зі списку, дату для нагадування про нотаток, поставити позначку «Важливий нотаток», а також переглянути інформацію про нотаток.

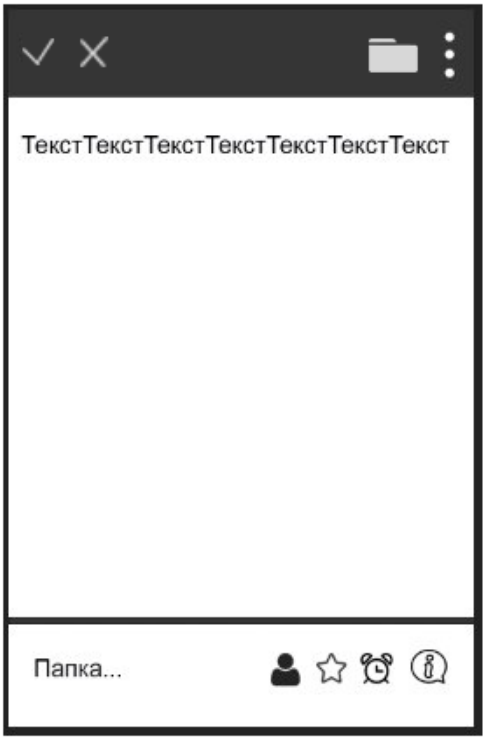

Рисунок 2.6 – Форма створення нотатку

Після натиску на кнопку «Створити нотаток» відразу відкривається форма «Перегляд нотатку», яка представлена на рисунку 2.7. В цій формі можна додавати папки, контакт, нагадування без переходу на форму «Редагувати нотаток».

Рисунок 2.7 – Форма перегляду нотатку

Форма «Меню» дозволяє переходити по вкладкам додатку, представлена на рисунку 2.8.

Рисунок 2.8 – Форма меню додатку

Форма меню з розгорнутою вкладкою «Папки» представлена на рисунку 2.9. В цій вкладці можна переглядати різні папки зі списками нотатків які були включені до них раніше та керувати папками. До керування папок відносяться додання нових та видалення раніше створених папок, а також перегляд кількості нотаток які знаходяться у різних папках.

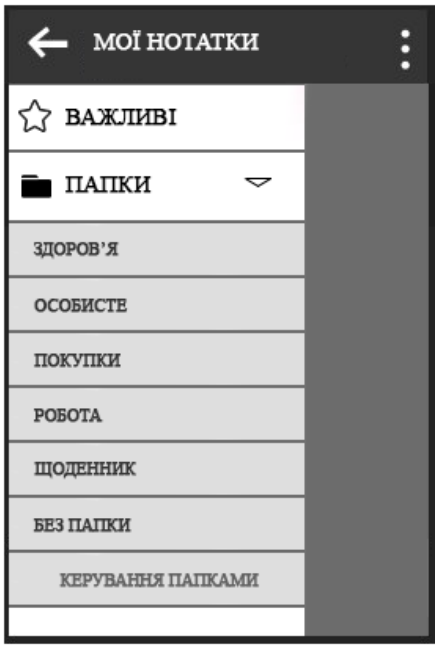

Рисунок 2.9 – Форма меню додатку з розгорнутою вкладкою «Папки»

Таким чином, представлено інтерфейс користувача програмного забезпечення «Додаток ведення нотаток» для Android.

2.2 Технічний проект програмного забезпечення

2.2.1 Логічна модель

Для розробки програмного забезпечення було обрано реляційну модель [8]. Логічна модель даних представлена на рисунку 2.10.

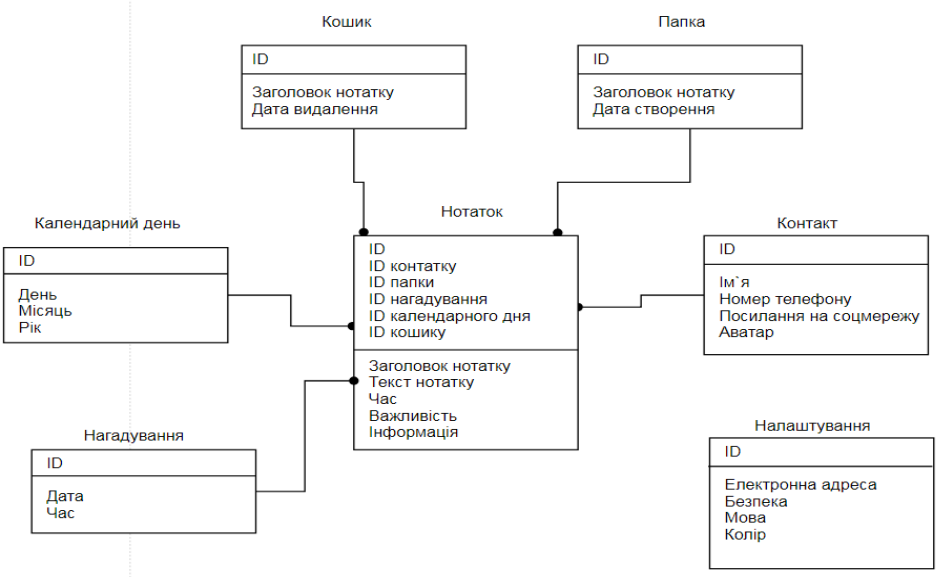

Рисунок 2.10 – Логічна модель даних

Таким чином, представлено логічну модель даних програмного забезпечення «Додаток ведення нотаток» для Android.

2.2.2 Діаграма послідовності

Діаграма послідовності в UML відображає взаємодії об'єктів, впорядкованих за часом [9].

Діаграма послідовності відображає взаємодію об'єктів впорядкованих за часом, зокрема, відображає задіяні об'єкти та послідовність відправлених повідомлень. Діаграма послідовності для основного потоку подій прецеденту.

Діаграма послідовності для основного потоку подій прецеденту «Додання нотатку» представлена на рисунку 2.11.

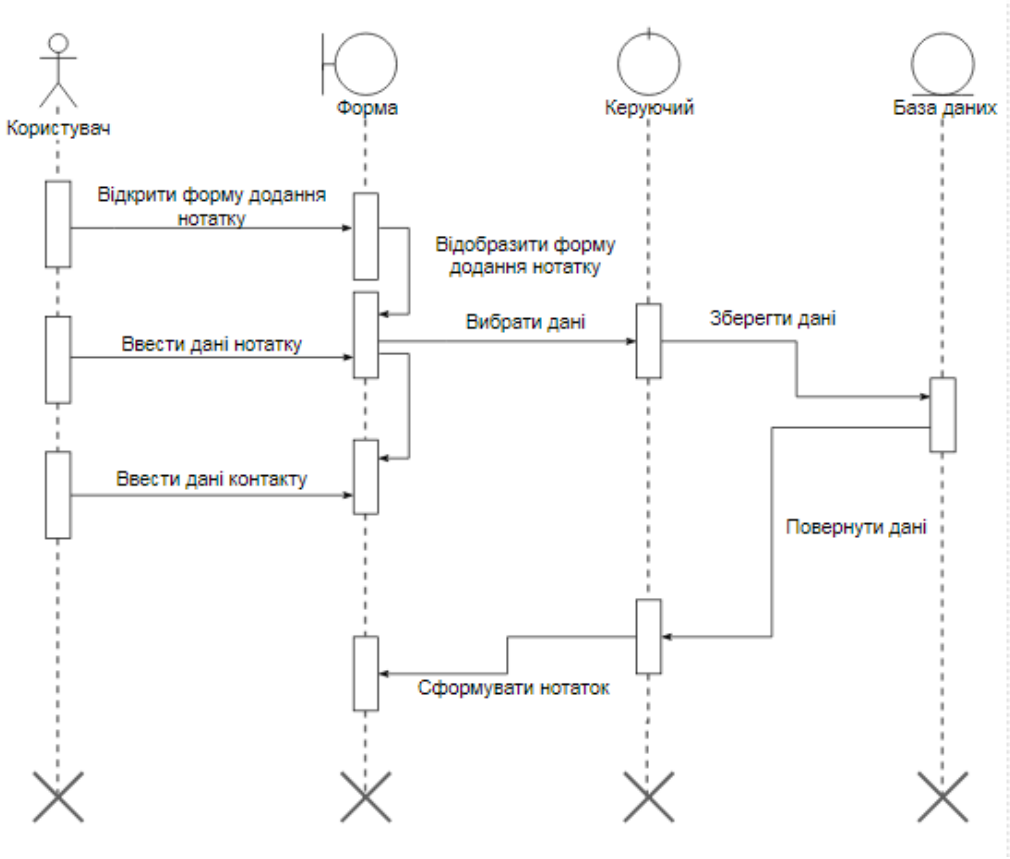

Рисунок 2.11 – Діаграма послідовності для основного потоку подій прецеденту «Додання нотатку»

Таким чином, розроблено діаграму послідовності програмного забезпечення «Додаток ведення нотаток» для Android.

2.2.2 Діаграма пакетів

Діаграма пакетів містить в собі структуру пакетів програми та залежності між ними [10]. Діаграма пакетів представлена на рисунку 2.12. Вона дозволяє наглядно відобразити залежності коду та допомагає правильно запланувати послідовність розробки модулів програми.

На даній діаграмі не відображається перегляд пакету view (представлення) так, як в андроїдній розробці цей пакет міститься в окремому каталозі та представлений за допомогою файлів *.xml

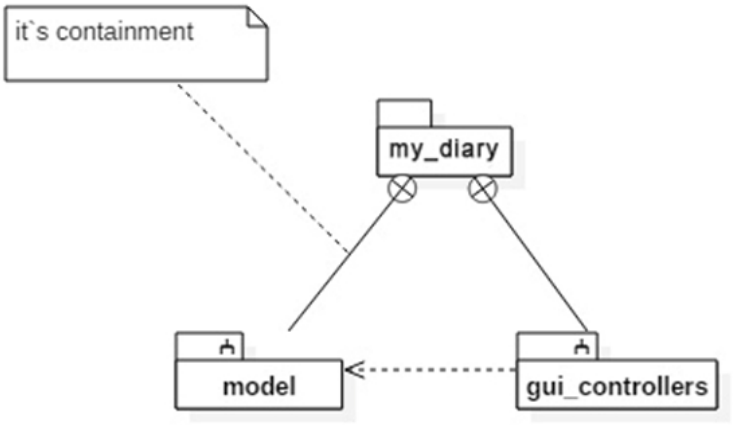

Рисунок 2.12 – Діаграма пакетів додатку

Таким чином, представлено діаграму пакетів програмного забезпечення «Додаток ведення нотаток» для Android.

2.2.3 Діаграма класів

Діаграма класів відноситься до статичних діаграм UML. Її призначення – представити класи, типи даних їх зміст та відношення [11]. Діаграма класів, яка показує взаємодію класів для пакету «Model» зображена на рисунку 2.13.

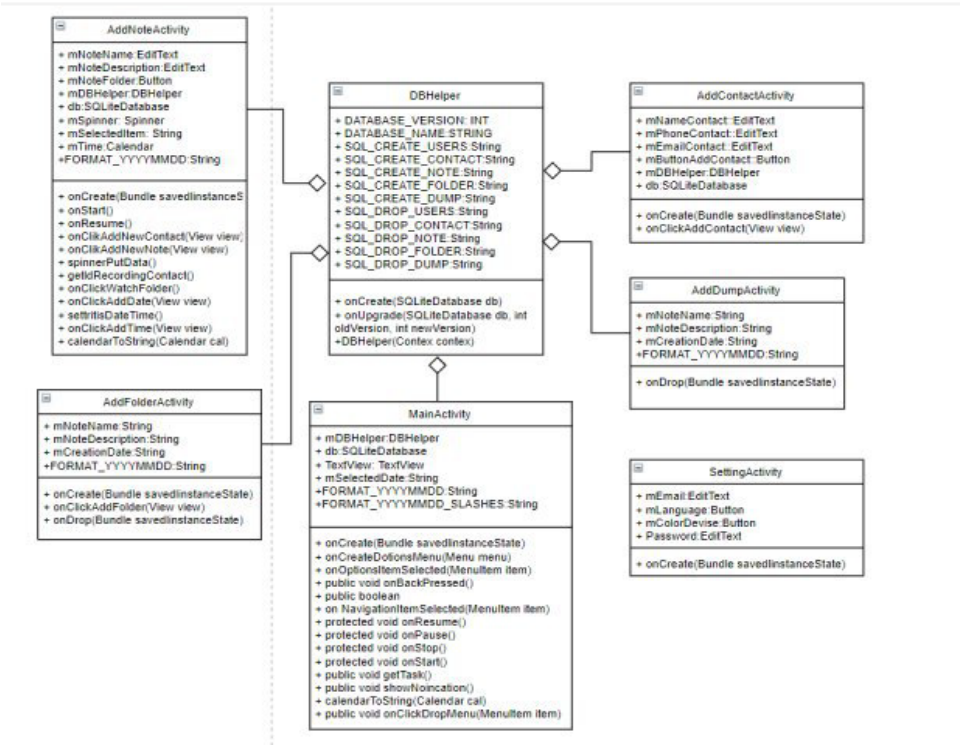

Рисунок 2.13 – Діаграма класів додатку «Мої нотатки»

Таким чином, представлено діаграму класів програмного забезпечення «Додаток ведення нотаток» для Android

2.2.4 Діаграми діяльності

Діаграма діяльності відображає взаємодію об'єктів, впорядкованих за часом [12]. Діаграма діяльності для запису даних до програмної системи представлена на рисунку 2.14.

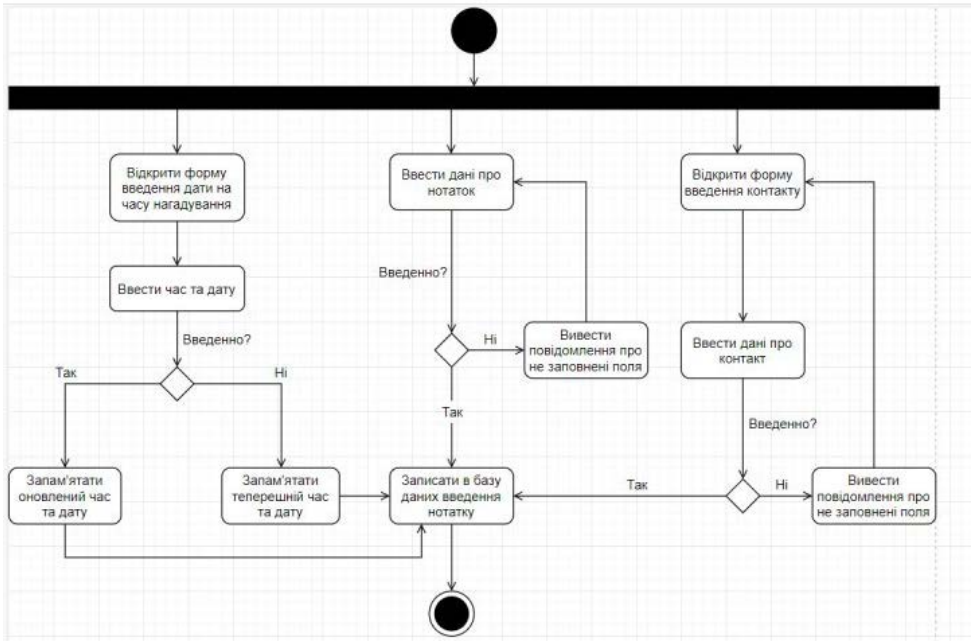

Рисунок 2.14 – Діаграма діяльності для ПС

Діаграма діяльності для запису даних нотатку представлена на рисунку 2.15.

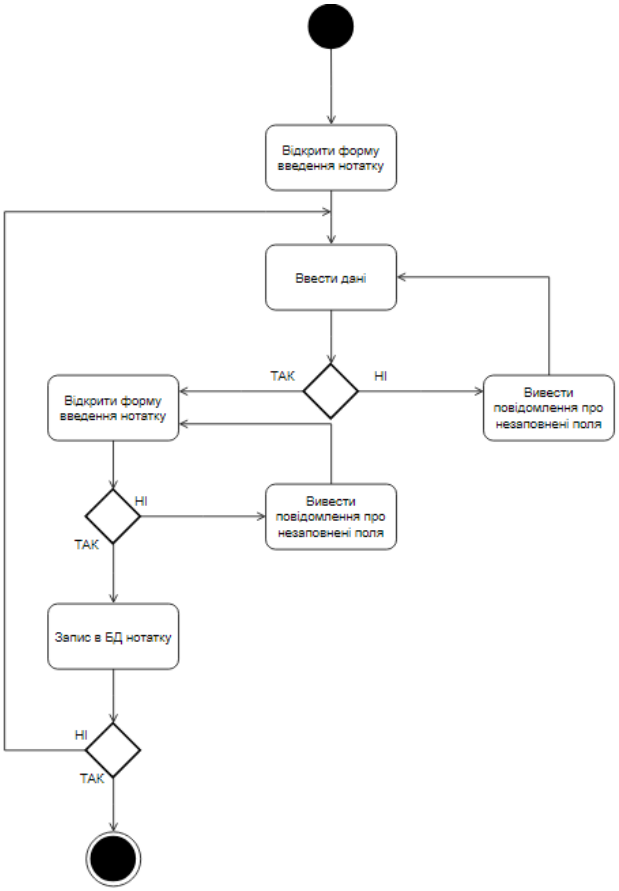

Рисунок 2.15 – Діаграма діяльності для введення даних нотатку

Таким чином, розроблено діаграму діяльності програмного забезпечення «Додаток ведення нотаток» для Android

2.3 Робочий проект програмного забезпечення

2.3.1 Обґрунтування вибору інструментарію

Критерії вибору середовища програмування є:

- можливість візуального програмування;
- використання системи контролю версіями Git;
- відкритий код;
- безкоштовність;
- зручність інтерфейсу.

Критерії вибору сховища даних:

- безкоштовність;
- надійність.

Критерії обрання мови програмування:

- кроссплатформеність;
- власні знання мови.

В якості засобу розробки кваліфікаційної роботи було обрано середовище Android Studio. Це інтегроване середовище розробки (IDE) для роботи з платформою Android. IDEA перебувала у вільному доступі. Android Studio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, - офіційний засіб розробки Android-додатків. Дане середовище розробки доступне для Windows, OS X та Linux. Офіційними мовами програмування для платформи Android є Java [13], C ++ та Kotlin.

Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в IntelliJ IDEA, в AndroidStudio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції [14].

Для прискорення розробки застосунків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу застосунку (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання

шаблонів. У середовище вбудовані функції завантаження типових прикладів коду з GitHub [15].

До складу також включені пристосовані під особливості платформи Android розширені інструменти рефакторингу, перевірки сумісності з минулими випусками, виявлення проблем з продуктивністю, моніторингу споживання пам'яті та оцінки зручності використання. У редактор доданий режим швидкого внесення правок. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API. Інтегрована підтримка оптимізатора коду ProGuard. Вбудовані засоби генерації цифрових підписів. Надано інтерфейс для управління перекладами на інші мови [14].

Кожна програма для Android може створювати і використовувати БД SQLite для зберігання великих обсягів структурованих даних.

SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання (англ. public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg [16].

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу [16].

Android використовує файлову систему, яка схожа на дискові файлові систем інших платформ.

Внутрішня пам'ять:

- завжди доступна;
- файли, збережені в ній, є доступними лише, коли додаток за замовчуванням;
- коли користувач видаляє додаток, то система видаляє всі файли додатку з внутрішньої пам'яті.

Зовнішні сховища:

- не завжди доступні, тому що користувач може встановити зовнішній накопичувач USB як сховище, але в деяких випадках видалити його з пристрою;
- це читання всім світом, тому файли, збережені тут, можна прочитати поза контролем;
- коли користувач видаляє додаток, система видаляє файли додатку звідси тільки тоді, коли ви зберігаєте їх у каталозі з getExternalFilesDir().

Внутрішній каталог зберігання додатку вказаний в імені пакета додатку в спеціальному місці файлової системи Android. Технічно, інший додаток може читати внутрішні файли, якщо встановлено режим файлу, щоб він був читабельним. Тим не менше, інші додатки також повинні знати ім'я пакету вашого додатку та імена файлів. Інші додатки не можуть переглядати внутрішні каталоги і не мають доступ для читання або запису, якщо явно не встановили дозвіл для файлів на читання або запис.

2.3.2 Розробка фізичної моделі

Фізична модель даних (або проектування бази даних) – подання дизайну даних як реалізованого чи призначеного для реалізації у системі керування базами даних. Завершена фізична модель даних включатиме всі артефакти бази даних, необхідні для створення відношень між таблицями чи для досягнення мети продуктивності, як-от індексів, визначень обмежень, зв'язаних і секціонованих таблиць або кластерів. Фізична модель даних представлена на рисунку 2.18.

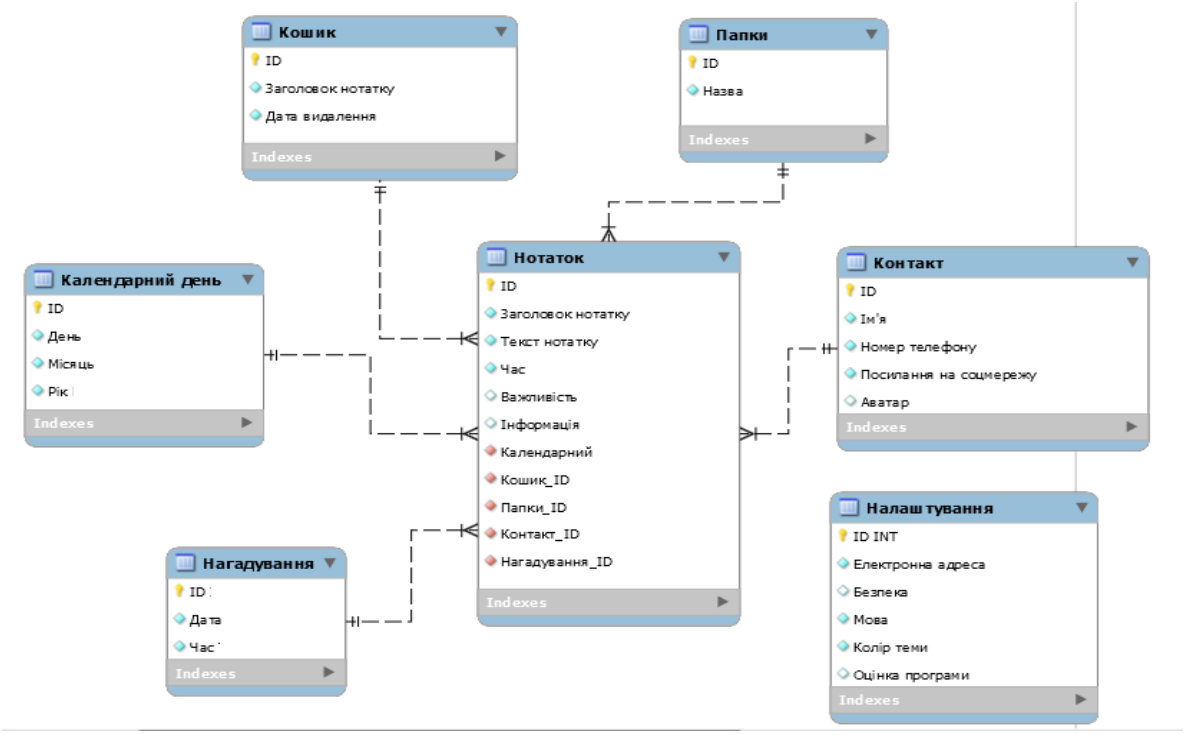

Рисунок 2.18 – Фізична модель даних ПС

На основі аналізу атрибутів сутностей бази даних кожному атрибуту був призначений відповідний тип. В таблицях 2.9 – 2.15 наведено опис структур таблиць бази даних.

Таблиця 2.9 – Структура фізичної моделі даних «Folder»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження	Примітка
1	2	3	4	5
ID	PK	INTEGER	NOT NULL	
Folder_name		VARCHAR(50)	NOT NULL	Назва папки

Таблиця 2.10 – Структура фізичної моделі даних «Dump»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження	Примітка
1	2	3	4	5
ID	PK	INTEGER	NOT NULL	
Title		VARCHAR(50)	NOT NULL	Заголовок нотатку
Delete_date		DATE	NOT NULL	Дата видалення

Таблиця 2.11 – Структура фізичної моделі даних «Contacts»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження	Примітка
1	2	3	4	5
ID	PK	INTEGER	NOT NULL	
Name		VARCHAR(45)	NOT NULL	Ім'я
Telephone		VARCHAR(13)	NOT NULL	Номер телефону
Link		VARCHAR(45)	NOT NULL	Посилання на соцмережу
Picture		BLOB		Аватар

Таблиця 2.12 – Структура фізичної моделі даних «TimeNote»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження	Примітка
1	2	3	4	5
ID	PK	INTEGER	NOT NULL	
Date		DATE	NOT NULL	Дата
Time		TIME	NOT NULL	Час

Таблиця 2.13 – Структура фізичної моделі даних «Calendar»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження	Примітка
1	2	3	4	5
ID	PK	INTEGER	NOT NULL	
Day		DATE	NOT NULL	День
Month		DATE	NOT NULL	Місяць
Year		DATE	NOT NULL	Рік

Таблиця 2.14 – Структура фізичної моделі даних «Note»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження	Примітка
1	2	3	4	5
ID	PK	INTEGER	NOT NULL	
Title		VARCHAR(50)	NOT NULL	Заголовок нотатку
Text		VARCHAR(500)	NOT NULL	Текст нотатку
Time		DATE	NOT NULL	Час створення
Importance		BOOLEN		Важливість
Information		VARCHAR(200)	NOT NULL	Інформація

Таблиця 2.15 – Структура фізичної моделі даних «Settings»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження	Примітка
1	2	3	4	5
Id	PK	INTEGER	NOT NULL	
Email		VARCHAR(50)	NOT NULL	Електронна адреса
Security		VARCHAR(100)		Безпека
Language		VARCHAR(100)	NOT NULL	Мова
App rating		TEXT		Оцінка програми

Таким чином, розроблено таблиці фізичної моделі даних програмного забезпечення «Додаток ведення нотаток» для Android

2.3.3 Реалізація та тестування основних класів еквівалентності програмного забезпечення

Реалізація розробки програмного забезпечення була виконана на мові Java в середовищі Android Studio, для створення бази даних було обрано СКБД SQLite.

Тестування функції «Створити новий контакт»

Тестування: для тестування функції будемо застосувати метод еквівалентного розбиття. Класи еквівалентності наведені в таблиці 2.16.

Таблиця 2.16 – Класи еквівалентності функції «Створити новий контакт»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Ім’я	1.Символи кирилиці або латиниці, та числа	2. Символи, окрім кирилиці, латиниці та чисел
Дата народження	3. Дата (рядок у форматі «дд.мм.рррр»)	4. Символи або числа у будь-якому форматі, окрім зазначеного
Номер телефону	5. Будь-які числа	6. Символи, окрім чисел
Пошта	7.Символи латиниці, числа, «@», «.», «-»,«_»	8. Символи, окрім латиниці, чисел, «@», «.», «-»,«_»
Аватар	9. Зображення формата JPG	10. Зображення інших форматів, окрім JPG

Тести з правильних класів еквівалентності наведено в таблиці 2.17.

Таблиця 2.17 – Тести з правильних класів еквівалентності

№ Тесту	№ покритого класу еквівалентності	Вхідні дані	Результат
1	1	Іванов	В результаті тестування з’являється напис «Новий контакт успішно створено»
	3	15.03.1997	
	5	380970825529	
	1	ivanov@gmail.com	
	9	icon.jpg	

Тести з неправильних класів еквівалентності наведено в таблиці 2.18.

Таблиця 2.18 – Тести з неправильних класів еквівалентності

№ Тесту	№ покритого класу еквівалентності	Вхідні дані	Результат
1	2	Іва,_N	В результаті тестування з’являється повідомлення «Помилка у форматі даних»
2	4	11219980	В результаті тестування з’являється повідомлення «Помилка у форматі даних»
3	5	84777	В результаті тестування з’являється повідомлення «Помилка у форматі даних»
4	6	ivangmail	В результаті тестування з’являється повідомлення «Помилка у форматі даних»
5	8	icon.png	В результаті тестування з’являється повідомлення «Помилка у форматі даних»

Тестування функції «Створити нотаток»

Тестування: для тестування функції будемо застосовувати метод еквівалентного розбиття. Класи еквівалентності наведені в таблиці 2.19.

Таблиця 2.19 – Класи еквівалентності функції «Створити нотаток»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Заголовок	1. Будь-які символи	
Основний текст	1. Будь-які символи	
Дата нагадування	2. Дата (рядок у форматі «дд.мм.рррр»)	3. Символи або числа у будь-якому форматі, окрім зазначеного
Контакт	4. Символи кирилиці або латиниці, та числа	5. Символи, окрім кирилиці, латиниці та чисел

Тести з правильних класів еквівалентності наведено в таблиці 2.20.

Таблиця 2.20 – Тести з правильних класів еквівалентності

№ Тесту	№ покритого класу еквівалентності	Вхідні дані	Результат
2	1	День 1-й	В результаті тестування з’являється напис «Нотаток успішно заплановано»
	1	Запланувати зустріч	
	2	20.07.2022	
	4	Іванов	

Тести з неправильних класів еквівалентності наведено в таблиці 2.21.

Таблиця 2.21 – Тести неправильних класів еквівалентності

№ Тесту	№ покритого класу еквівалентності	Вхідні дані	Результат
2	2	А6п	В результаті тестування з’являється напис «Дату введено в неправильному форматі,правильний формат:(дд.мм.рррр).»
	4	Іва,_N	В результаті тестування з’являється повідомлення «Помилка у форматі даних»

Таким чином, було проведено тестування функцій методом еквівалентного розбиття.

2.3.4 Випробування програмного забезпечення

Перевірка дотримання вимог програмної документації на програмний продукт виконується візуально представником служби, відповідальної за експлуатацію. У ході перевірки зіставляється склад і комплектність програмної документації, представленої розробником, з переліком програмної документації, наведеним нижче:

- Технічне завдання.
- Текст програми.
- Опис програми.
- Програма та методика випробувань.

- Інструкція користувача .

Перевірка вважається завершеною у випадку відповідності складу та комплектності програмної документації, представленої розробником, переліком програмної документації, наведеному у зазначеному вище переліку.

Перевірка працездатності програми виконується згідно з вимогами до функціональних характеристик ПЗ. Перевірка вважається завершеною у разі відповідності складу і послідовності виконаних дій вимогам до функціональних характеристик. В процесі тестування була перевірена функціональність за темою «Розробка програмного забезпечення «Додаток ведення нотаток» для Android». У табл. 2.22 вказано перелік випробувань основних функціональних можливостей.

Таблиця 2.22 – Методика випробування

№	Дія	Очікуваний результат	Результат перевірки	Зауваження
1	Перевірка правильності введення даних при авторизації	Програма повинна перевірити введені дані з записами у БД	Виконано	
2	Перевірка правильності введення даних при реєстрації	Програма повинна перевірити коректність введених даних	Виконано	
3	Перевірка правильності додавання нового нотатку в БД	Програма повинна перевірити коректність введених даних	Виконано	
4	Перевірка правильності введених даних при редагуванні нотатку в БД	Програма повинна перевірити коректність введених даних	Виконано	
5	Перевірка правильності видалення нотатку з БД	Програма повиннаперевірити введені дані з записами у БД	Виконано	
6	Перевірка правильності введених даних при пошуку нотатків в БД	Програма повинна перевірити введені дані з записами у БД	Виконано	
7	Перевірка правильності формування нотатків	Програма повинна перевірити введенідані з записами у БД	Виконано	

Таким чином, було проведено методику випробувань програмного забезпечення «Додаток ведення нотаток» для Android .

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «ДОДАТОК ВЕДЕННЯ НОТАТОК» ДЛЯ ANDROID

В даній кваліфікаційній роботі було розроблено додаток ведення нотаток для смартфонів з ОС Android. Для досягнення поставленої мети виконано аналіз предметної області, в якому було досліджено роботу інших нотатків .

В ескізному проекті побудовано контекстну діаграму, діаграму варіантів використання, концептуальну модель, діаграми переходів станів, розроблено ескізи користувацького інтерфейсу.

В технічному проекті побудовано логічну модель бази даних, розроблено діаграму послідовності, пакетів та класів для варіантів використання програмного забезпечення «Мої нотатки».

В робочому проекті виконано та аргументовано вибір інструментів, а також побудовано фізичну модель даної бази даних.

В результаті розроблено додаток, що має наступні функції (рисунки 3.1 – 3.3):

- управління нотатками;
- управління контактами;
- управління папками;
- перегляд подій в місяці;
- додання тексту і часу нагадування;
- управління кошиком;
- налаштування додатку (інтерфейс).

Детальніше результат роботи представлений в додатку Б – Інструкція користувача.

Рисунок 3.1 – Форма головної сторінки

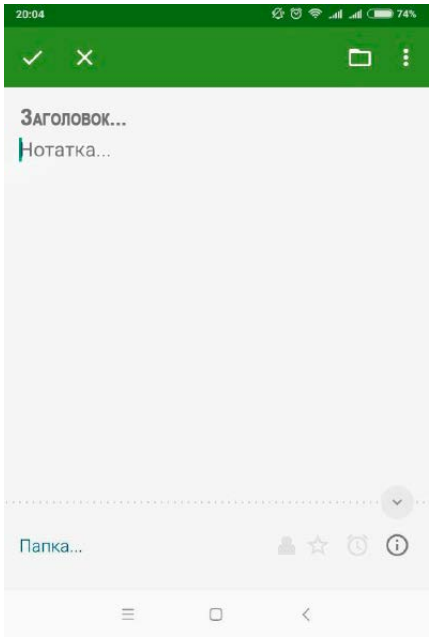

Рисунок 3.2 – Форма створення нотатку

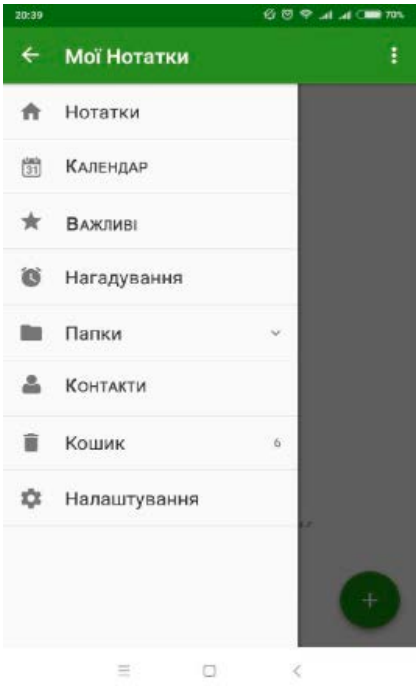

Рисунок 3.3 – Форма меню

Для розробки додатку було використано наступні інструменти:

- Java – об’єктно-орієнтована мова програмування.
- SQLite – вільна система для керування реляційними БД.
- AndroidStudio – це інтегроване середовище розробки (IDEA) для роботи з платформою Android.

В програмній документації було розроблено :

- Технічне завдання – Додаток А;
- Інструкцію користувача – Додаток Б;
- Опис програми – Додаток В;
- Текст програми – Додаток Г;
- Програму і методику випробувань – Додаток Д.

4 ОХОРОНА ПРАЦІ

4.1 Загальні положення охорони праці

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [17].

Державна політика в галузі охорони праці базується на принципах:

- пріоритету життя і здоров'я працівників, повної відповідальності роботодавця за створення належних, безпечних і здорових умов праці;
- підвищення рівня промислової безпеки шляхом забезпечення суцільного технічного контролю за станом виробництв, технологій та продукції, а також сприяння підприємствам у створенні безпечних та нешкідливих умов праці;
- комплексного розв'язання завдань охорони праці на основі загальнодержавних, галузевих, регіональних програм з цього питання та з урахуванням інших напрямів економічної і соціальної політики, досягнень в галузі науки і техніки та охорони довкілля;
- соціального захисту працівників, повного відшкодування шкоди особам, які потерпіли від нещасних випадків на виробництві та професійних захворювань;
- встановлення єдиних вимог з охорони праці для всіх підприємств та суб'єктів підприємницької діяльності незалежно від форм власності та видів діяльності;
- адаптації трудових процесів до можливостей працівника з урахуванням його здоров'я та психологічного стану;

використання економічних методів управління охороною праці, участі держави у фінансуванні заходів щодо охорони праці, залучення добровільних

внесків та інших надходжень на ці цілі, отримання яких не суперечить законодавству;

- інформування населення, проведення навчання, професійної підготовки і підвищення кваліфікації працівників з питань охорони праці [17];

- забезпечення координації діяльності органів державної влади, установ, організацій, об'єднань громадян, що розв'язують проблеми охорони здоров'я, гігієни та безпеки праці, а також співробітництва і проведення консультацій між роботодавцями та працівниками (їх представниками), між усіма соціальними групами під час прийняття рішень з охорони праці на місцевому та державному рівнях;

- використання світового досвіду організації роботи щодо поліпшення умов і підвищення безпеки праці на основі міжнародного співробітництва.

Основні законодавчі і нормативні документи, які регламентують охорону праці в Україні [17].

До основних документів, що регламентують охорону праці в Україні, відносяться:

- Конституція України;

- КЗпП;

- закони «Про охорону праці», «Про охорону здоров’я», «Про пожежну безпеку», «Про використання ядерної енергії та радіоактивний захист», «Про забезпечення санітарного та епідемічного благополуччя населення». Коло питань щодо охорони праці розглядається в підзаконних актах, указах Президента, постановах Верховної Ради та Кабміну, Цивільному, Кримінальному та Адміністративному кодексах України [18].

Кожній людині Конституція гарантує право на належні, безпечні і здорові умови праці. Конституцією гарантується захист від незаконного звільнення. Кожна людина має право на достатній життєвий рівень, на охорону здоров’я, медичну допомогу та медичне страхування.

В КЗпП охорона праці відображена в окремому розділі. В окремих статтях

цього розділу розглянуто створення здорових і безпечних умов праці; додержання вимог охорони праці.

До найважливіших актів з охорони праці належать:

- Закон України «Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності» (23.09.1999 р. № 1105);
- Порядок розслідування та обліку нещасних випадків, професійних захворювань і аварій на виробництві (25.08.2004 р. №1112);
- Порядок видачі дозволів Державним комітетом з нагляду за охороною праці та його територіальними органами (15.10.2003 р. № 1631);
- Порядок проведення атестації робочих місць за умовами праці (01.08.1992 р. №442).

Спеціальними законодавчими актами є міжгалузеві та галузеві акти про охорону праці: Державні стандарти, Системи стандартів безпеки праці, Санітарні норми, норми радіаційної безпеки тощо.

4.2 Вимоги до охорони праці користувачів ПК.

Основними нормативно-правовими актами з охорони праці користувачів персональних комп'ютерів є:

- «Правила охорони праці під час експлуатації ЕОМ» (ДНАОП 0.00-1.31-99);
- «Державні санітарні правила і норми роботи з відео дисплейними терміналами ЕОМ» (Дсан П і Н 3.3.2.007-98).

Ці нормативно-правові акти поширюються:

- на всі підприємства і організації будь якої форми власності і виду діяльності;

- на всіх фізичних осіб, що займаються підприємницькою діяльністю з правом найму робочої сили, і використовують персональні ЕОМ.

Правила не розповсюджуються на:

- комп'ютерні класи навчальних закладів;
- ЕОМ управління і контролю за роботою АЕС;
- бортові ЕОМ транспортних засобів і літаків, що переміщуються в процесі їх роботи;
- портативні ЕОМ;
- комп'ютерні гральні автомати;
- системи обробки даних громадського користування (інформаційні системи аеропортів, залізничних станцій, тощо).

У Правилах викладені гігієнічні та ергономічні вимоги до організації робочих місць, виробничих приміщень та параметрів мікроклімату, які забезпечують збереження стану здоров'я працівників при експлуатації ПК. Відповідальність за виконання цих Правил покладається на посадових і фізичних осіб, які використовують найману працю і застосування ПЕОМ у своїй діяльності.

Державний нагляд за дотриманням цих вимог покладається на державні санітарно-епідеміологічні органи МОЗ України, та відповідні відомчі санітарно-епідеміологічні служби МО України, МВС України, СБУ і Прикордонних військ України (ст.31 Закону України «Про забезпечення санітарного та епідеміологічного благополуччя населення»). Особи, винні у порушені вимог даних Правил, несуть дисциплінарну, адміністративну, матеріальну та кримінальну відповідальність [18].

4.3 Загальні вимоги до виробничих приміщень ЕОМ

Серед основних вимог виділяють:

- розміщення робочих місць у підвальних і цокольних приміщеннях заборонено;

- площа на одне робоче місце ПК повинна бути не менше 6 $м^2$, а об’єм – не менше 20 $м^3$;

- підлога всієї зони обслуговування ЕОМ має бути вкрита діелектричними килимами, з метою заниження електростатичного поля між оператором і ПК;

- усі металеві конструкції, що мають заземлення (батареї опалення, водопровідні труби, екрани електропроводки, тощо) повинні бути надійно захищені діелектричними щитками або сітками від випадкового дотику людини до них;

- в приміщеннях для роботи з ПК повинні бути кімнати для відпочинку і психологічного розвантаження працівників. У цих кімнатах повинно бути передбачена роздача тонізуючих напоїв, та місця для занять фізичною культурою.

Гігієнічні вимоги до приміщень та нормативи чинників, що створюються при роботі комп'ютерів, гігієнічні вимоги до експлуатації персональних комп'ютерів, що застосовуються в навчально-виховному процесі в навчальних закладах різних форм власності встановлюють Державні санітарні правила і норми «Влаштування і обладнання кабінетів комп'ютерної техніки в навчальних закладах та режим праці учнів на персональних комп'ютерах».

Дія та вплив електромагнітного випромінювання основані на концепції взаємодії зовнішніх полів з внутрішніми полями організму людини, на центральну нервову систему, очі, кровотворну систему, серцево-судинну систему, ендокринну та імунну системи і обмінні процеси.

Регулярна робота з комп'ютером без застосування відповідних захисних засобів приводить до зниження імунітету, захворювання органів зору, до хвороб серцево-судинної системи та шлунково-кишкового тракту. Робота з комп’ютером супроводжується нервовим напруженням, оскільки вимагає швидкої реакції. Короткочасна концентрація нервових процесів викликає у людини втому.

4.4 Санітарно-гігієнічні вимоги до виробничого середовища ЕОМ.

Санітарно-гігієнічні вимоги до виробничого середовища ЕОМ визначають:

- мікроклімат;
- степінь освітленості;
- рівень шуму і вібрації;
- іонізуюче та неіонізуюче випромінювання.

Мікроклімат

Мікроклімат на робочих місцях з ПК нормується ДЕСТ 12.1.005-88 і повинен мати:

- температуру оточуючого повітря – 21–240С;
- відносну вологість – 40–60 %;
- швидкість руху повітря – 0,1–0,2 м/с.

Освітлення

1. Робочі місця з ПК повинні мати природне і штучне освітлення;
2. Вікна приміщень з ПК повинні мати орієнтацію на північ або на північний схід, з КПО1,5%. Дозволяється експлуатація ЕОМ у приміщеннях без природного освітлення за узгодженням з органами Держнаглядохоронпраці та санітарно-епідеміологічними установами;
3. Рівень освітленості на робочому місці в зоні розміщення документів має бути 300–500лк;
4. Штучне освітлення виконується, як правило, люмінесцентними лампами. Допускається для місцевого освітлення використовувати лампи розжарювання;
5. Місцеве освітлення не повинно створювати блисків на екрані ПК, а його освітленість повинна не перевищувати 300 лк.

Рівні шуму

Рівні шуму на робочих місцях ЕОМ визначені ДсанПіН 3.3.2-007-98 в залежності від його тональності та виду трудової діяльності користувачів ЕОМ (програмісти, оператори ЕОМ, чи оператори комп'ютерного набору).Рівні звукового тиску описані у таблиці 4.1.

Таблиця 4.1 – Рівні звукового тиску в дБ в октавних смугах частот, Гц.

Вид трудової діяльності / Рівні звуку еквівалентні рівні звуку, дБА/дБАекв.	31.5	63	125	250	500	1000	2000	4000	8000
Програмісти ЕОМ	71	61	54	49	45	42	40	38	50
Оператори ЕОМ та оператори комп'ютерного набору	83	74	68	63	60	57	55	54	65
В приміщеннях, де розташовуються шумні агрегати ЕОМ	91	83	77	73	70	68	66	64	75

Для нормування шуму застосовуються шумопоглинальні засоби, визначені спеціальними інженерно-акустичними розрахунками.

Рівні вібрації

Рівні вібрації не повинні перевищувати допустимих норм, визначених ДсанПіН 3.3.2-007-98.

Рівні іонізуючого випромінювання

Рівні іонізуючого випромінювання повинні відповідати НРБУ–97. Потужність експозиційної дози рентгенівського випромінювання на відстані 5 см від екрану ПК повинна бути не більше 0,1 мбер/год (100 мкР/год).

Рівні неіонізуючого випромінювання

Рівні неіонізуючого випромінювання повинні відповідати вимогам ДЕСТ 12.1.006, СН №3206-85 «Гранично допустимі рівні магнітного поля частотою 50 Гц,

СН №4557-88 «Санітарні норми ультрафіолетового випромінювання у виробничих приміщеннях».

4.5 Ергономічні вимоги до робочого місця програміста

Робоче місце користувача ПК повинно відповідати вимогам ДЕСТ 12.2.032-78. Конструкція робочого місця має забезпечити оптимальну позу користувача ПК, а саме: ступні ніг – на підлозі або на підставці для ніг; стегна – в горизонтальному положенні; передпліччя – вертикально; лікті – під кутом 70–90 градусів до вертикальної площини; зап'ястя – зігнуті під кутом не більше 20 градусів відносно горизонту; нахил голови – 15–20 градусів відносно вертикальної площини.

Для забезпечення такої пози комп'ютерний стіл повинен мати розміри: висота – 725 мм; ширина – 600–1400 мм; глибина – 800–1000 мм. Розміри підставки для ніг: ширина – не менше 300 мм; глибина – не менше 400 мм; висота – регульована до 150 мм; кут нахилу – в межах 200.

Робочі місця з ПК повинні розташовуватися таким чином, щоб природне освітлення з вікон падало на них переважно з лівого боку, а їх взаємо-розташування становило: від стін з вікнами – не менше 1 м; по ширині між сусідніми ПК – не менше 1,2 м: по глибині від переднього до заднього ПК – не менше 2,5 м; прохід між рядами ПК – не менше 1 м.

Такі вимоги взаємного розташування ПК стосуються і для тих ПК, які знаходяться в сусідніх приміщеннях, відгороджені між собою стінкою або перегородкою. Відстань від екрану до очей користувача ПК повинна бути не меншою 600 мм. Клавіатура ПК повинна розташовуватися на відстані 100–300 мм від краю стола на опорному пристрої, який має високий коефіцієнт тертя, та має можливість змінювати кут нахилу клавіатури в межах 5–150.

Для захисту працівника від шкідливих випромінювань ПК, необхідно застосовувати при екранні фільтри, локальні світлофільтри, та засоби індивідуального захисту, які пройшли необхідне випробовування, і мають

відповідний гігієнічний сертифікат.

4.6 Розрахунок точковим методом освітлення приміщення

За правилами спроектоване освітлення забезпечить високий рівень працездатності та сприятиме підвищенню продуктивності праці.

Розрахунок освітлення будемо виконувати методом коефіцієнта використання світлового потоку.

Розрахуємо потрібну кількість світильників для приміщення з комп'ютером при загальному рівномірному освітленні. Маємо наступні дані: довжина приміщення А = 5 м, ширина приміщення В = 4 м, висота приміщення Н=3 м. Висота робочої поверхні h_p=1 м.

Стеля обладнається світильниками з люмінесцентними лампами денного світла ЛПО–01. Коефіцієнт відбиття стелі $S_п$=70%, стін $S_с$=50%, робочої поверхні $S_р$=10%. Напруга мережі 220В.

Відстань від стелі до робочої поверхні:

$$H_0 = H - hp = 3 - 1 = 2 \text{ м}; \quad (4.1)$$

Відстань від стелі до світильника:

$$h_c = 0{,}2 * H_0 = 0{,}2 * 2 = 0{,}4 \text{ м}; \quad (4.2)$$

Висота підвісу світильника над освітлюваною поверхнею:

$$h = H_0 - h_c = 2 - 0{,}4 = 1{,}6 \text{ м}; \quad (4.3)$$

Висота підвісу світильника над підлогою:

$$H_p = h + h_p = 1{,}6 + 1 = 2{,}6 \text{ м}; \quad (4.4)$$

Для досягнення найбільшої рівномірності освітлення приймаємо відношення L/h=1,5. Тоді відстань між центром і світильником:

$$L = 1{,}5 * h = 1{,}5 * 1{,}6 = 2{,}4 \text{ м}; \quad (4.5)$$

Потрібна кількість світильників:

$$N = \frac{S}{L} = \frac{(5 * 4)}{2,4} = 8,33 \text{ світильників;} \quad (4.6)$$

Отже, потрібна кількість світильників дорівнює восьми (8 світильників (2 ряди по 4 світильники));

Індекс приміщення:

$$i = \frac{(A * B)}{h * (A + B)} = \frac{(5 * 4)}{1,6(5 + 4)} = 1,39; \quad (4.7)$$

По таблиці коефіцієнтів використання світлового потоку при i=1,25, Sп=70%, Sc=50%, Sp=10% для світильника типу «ЛПО–01» коефіцієнт використання світлового потоку η=0,51.

Світловий потік однієї лампи:

$$Ф = \frac{(E_{min} * S * K_з * Z)}{(N * \eta)} = \frac{(100 * 20 * 1,5 * 1,1)}{(8 * 0,51)} = 808 \text{ лм;} \quad (4.8)$$

де $K_з$ – коефіцієнт запасу, який враховує запилення світильників і знос джерел світла в процесі експлуатації; для приміщень, освітлюваних люмінесцентними лампами $K_з$=1,5 за умови чищення світильників не рідше двох разів на рік;

Z – коефіцієнт нерівномірності освітлення (Z =1,1);

По знайденому значенню світлового потоку кожної лампи з таблиці, при напрузі мережі 220В, визначаємо її потужність, що має світловий потік 850 лм, найбільш близькій до розрахункового. При цьому фактична освітленість $E_ф$ буде дорівнювати:

$$E_ф = E_{min}(\frac{Фл}{Фр}) = 100 * (\frac{850}{808}) = 105 \text{ лк;} \quad (4.9)$$

Загальна потужність світильників:

$$P_{заг} = P_л * N = 20 * 8 = 160 \text{ Вт.} \quad (4.10)$$

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено програмне забезпечення «Додаток ведення нотаток» для Android, який дозволяє користувачу планувати свій графік і складати список завдань на день.

Дана кваліфікаційна робота передбачає розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, саме розробку програмного забезпечення «Додаток ведення нотаток».

Під час роботи над кваліфікаційною роботою було пройдено етапи:

- детального дослідження предметної галузі, аналізу готових рішень та постановки задачі, де була аргументована актуальність створення додатку;
- проектування (цей етап складався з ескізного, технічного та робочого проектів. Для проектування була обрана об’єктно-орієнтована методологія);
- дослідження розділу з охорони праці;

Самостійно опрацьовано основні компоненти додатків під OS Android, життєві принципи роботи, створення екрану (в редакторі), робота з файлами, зберігання даних (SQLite) та динамічне відтворення даних на екрані додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. TickTick – додаток для Android. URL: https://uk.phoneky.com/android/?id=d1d117174#gsc.tab=0 (дата звернення 12.02.2022).

2. Writeaday – додаток для Android. URL: https://apkpure.com/writeaday-journal-mood/com.compscieddy.writeaday (дата звернення 15.02.2022).

3. Опис мови UML. URL: https://www.docs.kde.org/stable4/uk/kdesdk/umbrello/uml-basics.html (дата звернення 19.06.2022).

4. Розробка uml діаграми варіантів використання. URL: https://studwood.net/1874589/informatika/diagrama_variantiv_vikoristannya (дата звернення 11.05.2022).

5. Концептуальна модель. URL: https://helpiks.org/7-49575.html (дата звернення 10.05.2022).

6. Діаграма переходів станів. URL: http://iwanoff.inf.ua/oop_kn/LabTraining05.html (дата звернення 18.05.2022).

7. Проектування інтерфейсу. URL: http://ekmair.ukma.edu.ua/bitstream/handle/123456789/22417/Kryvosheienko_Bakalavrska_robota.pdf?sequence=1&isAllowed=y (дата звернення 17.05.2022).

8. Побудова логічної моделі даних. URL: https://helpiks.org/2-9488.html (дата звернення 17.05.2022).

9. UML Діаграма послідовності. URL: https://sites.google.com/site/flashukrainian/cxefpagxo/uml/diagrama-poslidovnosti?overridemobile=true (дата звернення 22.05.2022).

10. Діаграма пакетів. URL: http://iwanoff.inf.ua/oop_kn/LabTraining05.html (дата звернення 22.05.2022).

11. Розробка діаграм класів. URL: https://infopedia.su/6x11d3.html (дата звернення 17.05.2022).

12. Діаграма діяльності. URL: https://www.wiki.uk-ua.nina.az/Діаграма діяльності.html (дата звернення 17.05.2022).

13. Хорстманн К.С., Корнелл Г. Бібліотека професіонала. Java 2. Том 1. Харків: Видавницький дім «Вільямс», 2003. 848 с.

14. Автоматизація побудови із Gradle. URL: http://eprints.library.odeku.edu.ua/1504/1/Andrus_Rozrob_mob_B_2018.pdf (дата звернення 03.05.2022).

15. Android Studio. URL: https://uk.wikipedia.org/wiki/Android_Studio (дата звернення 09.05.2022).

16. Система керування базами даних SQLite. URL: ttps://uk.wikipedia.org/wiki/SQLite (дата звернення 11.05.2022).

17. Жидецький В.Ц., Джигирей В.С., Мельников О.В. Основи охорони праці/ за ред. В.Ц. Жидецький. Київ, 2010. 325 с.

18. Про охорону праці : Закон України від 21.11.2002 р. № 229-IV. Дата оновлення: 04.02.2021. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text. (дата звернення 15.02.2022).

ДОДАТОК А – Технічне завдання

ВСТУП

Найменування програмного забезпечення, що розробляється – «Додаток ведення нотаток». Область застосування – смартфон користувачів.

1 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Розробка програмного забезпечення виконується на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Експлуатаційне призначення

Експлуатаційним призначенням програмного забезпечення «Додаток ведення нотаток» для Android є автоматизація планування графіку користувача та складання списоку завдань на день, в якому ставитиметься конкретне завдання чи нотаток, підбиратиметься час нагадування, потрібна папка.

2.2 Функціональне призначення

Функціональним призначенням програмного забезпечення «Додаток ведення нотаток» для Android є автоматизація процесів:

- управління нотатками;
- управління контактами;

- управління папками;
- перегляд подій в місяці;
- додання тексту і часу нагадування;
- управління кошиком;
- налаштування додатку (інтерфейс).

3 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вимоги до функціональних характеристик

Програмний додаток повинен виконувати всі нижчеописані функції.

Додавання нового нотатку

Для створення нотатку потрібно натиснути кнопку «Додати нотаток» та ввести потрібну інформацію, після чого натиснути галочку «Зберегти».

Перегляд нотатку

Для перегляду нотатку слід обрати в списку всіх нотатків потрібний для перегляду нотаток.

Пошук необхідного нотатку

Для пошуку потрібних нотатку необхідно обрати таблицю з БД системи та ввести ключові слова для пошуку.

Редагування запису

Для редагування запису потрібно обрати таблицю та відповідний запис з БД системи та замінити інформацію обраного нотатку.

Видалення запису

Для видалення запису потрібно обрати таблицю та відповідний запис з БД системи.

3.2 Вимоги до організації вхідних та вихідних даних

Дані зберігаються в файлі БД з довільним доступом типу .myd.

Вхідні дані: введення даних здійснюється за допомогою пристроїв введення

даних (смартфон, планшет).

Дані до нотатку або вводяться вручну, або здійснюються голосовим вводом.

Вихідні дані: виведення даних здійснюється на дисплей смартфону (сповіщення або нагадування).

3.3 Вимоги до надійності

При проектуванні повинні бути використані типові проектні рішення там, де це можливо. На стадії технічного проектування повинні прийматися рішення, що зменшують трудомісткість експлуатації системи.

3.4 Вимоги до умов експлуатації

Необхідний рівень підготовки користувачів: мінімальні навики в користуванні смартфону.

3.5 Вимоги до технічного забезпечення

Система повинна задовольняти вказаним вимогам смартфону наступної мінімальної комплекції:

- ОС Android від версії 5.0 і вище;
- 300 MB вільного місця.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація повинна містити:

- технічне завдання (ТЗ);
- текст програми;
- опис програми;
- інструкцію користувача;

− програму і методику випробувань.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки представлені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи робіт	Термін виконання	
1 Технічне завдання	1.1Обгрунтування необхідності розробки програми	08.02.22	01.03.22
	1.2 Розробка технічного завдання	02.03.22	19.03.22
	1.3 Затвердження технічного завдання	20.03.22	25.03.22
2 Ескізний проект	2.1 Розробка ескізного проекту	26.03.22	10.04.22
	2.2 Затвердження ескізного проекту	11.04.22	14.04.22
3 Технічний проект	3.1 Розробка технічного проекту	15.04.22	30.04.22
	3.2 Затвердження технічного проекту	01.05.22	05.05.22
4 Робочий проект	4.1 Розробка програми	06.05.22	15.05.22
	4.2 Розробка програмної документації	16.05.22	17.05.22
	4.3 Випробування програми	18.05.22	20.05.22

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення відбувається на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Приймання проводиться відповідно до програми і методики випробувань і документується за допомогою протоколу проведення випробувань. У разі знаходження помилок під час приймання програмного забезпечення, складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджуються керівником організації–замовника та організації–розробника. Розробник повинен, на протязі не більше ніж 2 тижні, виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б – Інструкція користувача

Для інсталяції розробленого ПЗ на смартфон з операційною системою Android повинен бути встановлений даний додаток «Мої нотатки». Користувачем системи, буде власник сматрфону на якому встановлений даний додаток.

Запуск програми здійснюється за допомогою ярлика на робочому столі смартону.

Після запуску додатку перед користувачем з’являється головна сторінка, яка зображена на рисунку Б.1.

Рисунок Б.1 – Форма головної сторінки

Після натискання кнопки «Додати» здійснюється перехід на форму «Створення нотатку», що представлено на рисунку Б.2.

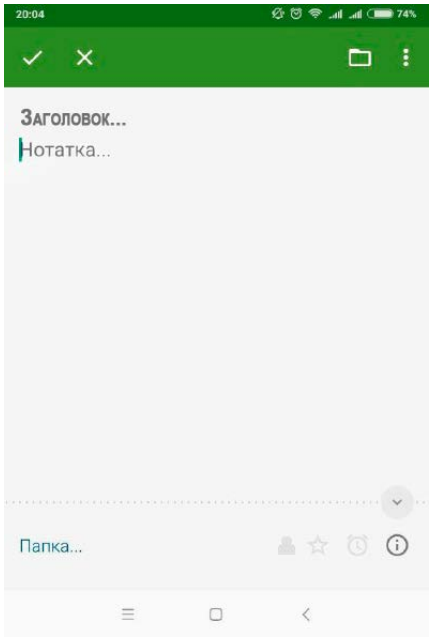

Рисунок Б.2 – Форма «Створення нотатку»

Після натиску на кнопку «Створити нотаток» відразу відкривається форма «Перегляд нотатку» що представлено на рисунку Б.3. В цій формі можна додавати папки, контакт, нагадування без переходу на форму «Редагувати нотаток».

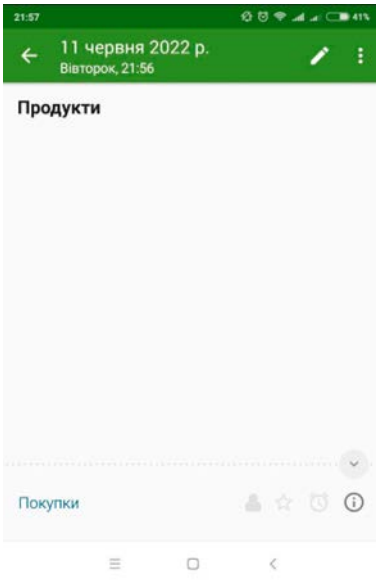

Рисунок Б.3 – Форма перегляду нотатку

Форма перегляду календаря представлена на рисунку Б.4.

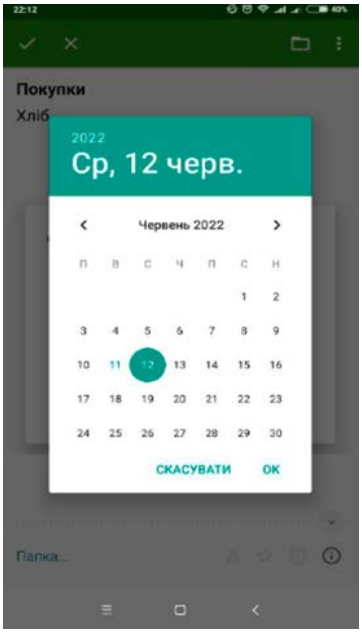

Рисунок Б.4 – Форма перегляду календаря

Форма меню представлена на рисунку Б.5.

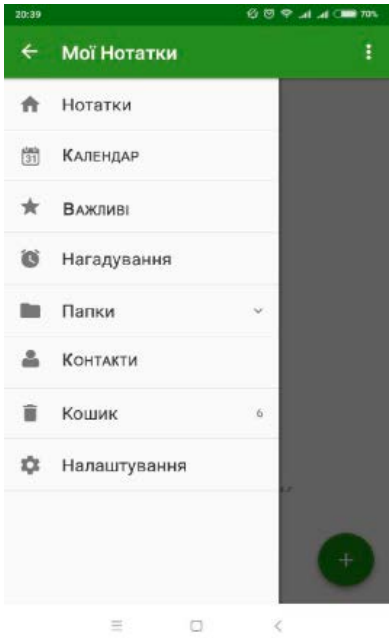

Рисунок Б.5 – Форма меню

Сповіщення з нагадуванням про нотаток представлено на рисунку Б.6.

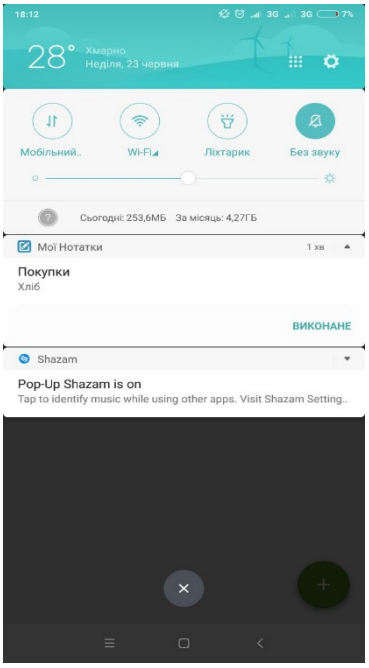

Рисунок Б.6 – Сповіщення з нагадуванням про нотаток

Після натиску на кнопку «Виконане» в сповіщенні про нагадування, автоматично перекреслюється дата та час нагадування в нотатку (рисунок Б.7).

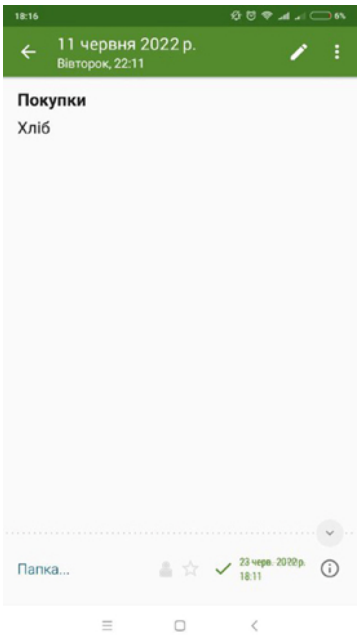

Рисунок Б.7 – Форма нотатку з виконаним нагадуванням

Після натиску на кнопку «Видалити» нотаток потрапляє до кошику, де його

можна буде переглянути, відновити або видалити назавжди. Форму кошику представлено на рисунку Б.8.

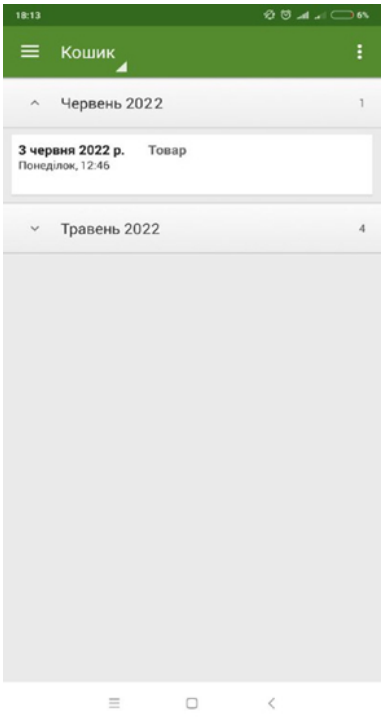

Рисунок Б.8– Форма кошику з видаленими нотатками

В формі «Керування папками» можна додавати, видаляти та переглядати кількість нотаток в різних папках (рисунок Б.9).

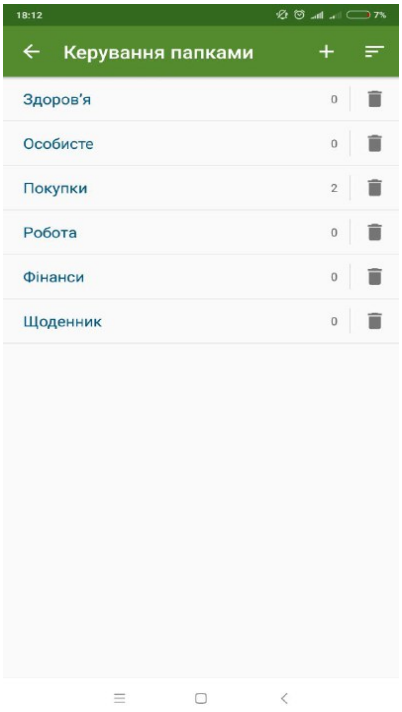

Рисунок Б.9 – Форма керування папок

В формі створення контакту можна додати ім’я контакту, номер телефону, електронну пошту та аватар. Форма «Контакт» представлена на рисунку Б.10

Рисунок Б.10 – Форма «Контакт»

Форма налаштування додатку представлена на рисунку Б.11. В ній додається

електронна адреса для відновлення, обирається потрібна мова, змінюється колір теми та ставиться оцінка додатку.

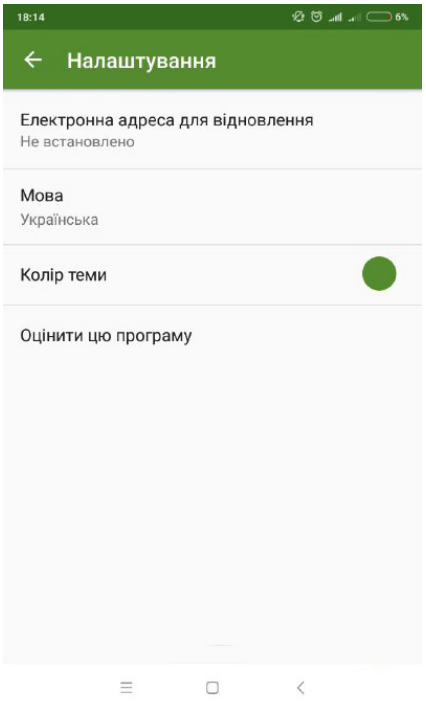

Рисунок Б.11 – Форма «Налаштування»

Додаток В – Опис програми

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування програмного забезпечення – «Додаток ведення нотаток» для Android», що розробляється на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова. Позначення програми – «Ведення нотаток».

1.2 Програмне забезпечення, необхідне для функціонування програми

Система повинна задовольняти вказаним вимогам на смартфон наступної мінімальної комплекції:

– Процессор: 8 ядер, 2 ГГц, 64 бита;

– Оперативная память: 3/4 ГБ;

– 300 МВ вільного місця.

Для повноцінного функціонування системи необхідно використовувати ОС Android від версії 5.0 і вище.

1.3 Мови програмування

Для розробки програмного забезпечення обрали бібліотеку для розробки програм з графічним інтерфейсом на мові Java та СУБД SQLite. Модулі, реалізовані на інших мовах програмування не використовуються.

2 Функціональне призначення

2.1 Класи розв’язуваних завдань та призначення програми Функціональним призначенням програмного забезпечення «Додаток ведення нотаток» для Android» є автоматизація наступних процесів:

– зберігання нотатку;

– перегляд нотатку;

– редагування нотатку;

– нагадування про нотаток;

– видалення нотатку.

3 Опис логічної структури

3.1 Структура програми

Програмне забезпечення «Додаток ведення нотаток» для Android» має дві складові частини: клієнтська частина – це графічний інтерфейс та база даних - програмне забезпечення, що забезпечує зберігання даних та їх видачу в момент запиту.

Логічна структура програми представлена взаємодією основних модулів програми: модулями програми можна представити варіанти використання на діаграмі варіантів використання, тому логічною моделлю програми можна представити діаграму варіантів використання.

3.2 Алгоритм програми

Робота програмного забезпечення «Додаток ведення нотаток» для Android» представлена взаємодією між клієнтською частиною та базою даних. Алгоритм роботи програми є наступним:

1. Користувач взаємодіє з програмним додатком.

2. Програмний додаток звертається до бази даних (якщо це потрібно) та відсилає результат.

3. Програмний додаток відображає отриманий результат.

3.3 Використовувані методи

Програмне забезпечення – «Додаток ведення нотаток» для Android» розроблено мовою програмування Java, що є об’єктно-орієнтованою.

4 Технічні засоби, що використовуються

Програмне забезпечення – «Додаток ведення нотаток» для Android» потребує від смартфону, на якому він буде встановлений, наступних характеристик, які треба розглядати як мінімальні:

– Процессор: 8 ядер, 2 ГГц, 64 бита;

– Оперативная память: 3/4 ГБ;

– 300 МВ вільного місця.

Для повноцінного функціонування системи необхідно використовувати ОС Android від версії 5.0 і вище.

5 Виклик та завантаження

Виклик та завантаження програмного забезпечення «Додаток ведення нотаток» для Android» відбувається запуском відповідно названого виконуваного файлу – vedennya_notatok.jar. Запуск програмного додатку «Додаток ведення нотаток» іншими способами неможливий.

6 Вхідні дані

Вхідні дані: введення даних здійснюється за допомогою пристроїв введення даних (смартфон, планшет). Дані до нотатку або вводяться вручну, або здійснюються голосовим вводом.

7 Вихідні дані

Вихідні дані: виведення даних здійснюється на дисплей смартфону (сповіщення або нагадування).

ДОДАТОК Г – Текст програми

В Android Studio весь код заходиться у класах, які розташовані в певному місці проекту.

Лістинг В.1 – Клас «AddContactActivity»

```
package com.example.pannatuck.materialtest210618;

import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class AddContactActivity extends AppCompatActivity {

    EditText mNameContact;
    EditText mPhoneContact;

    Button mButtonAddContact;

    DBHelper mDBHelper;
    SQLiteDatabase db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_contact);

        mNameContact = findViewById(R.id.editTextNameContact);
        mPhoneContact = findViewById(R.id.editTextPhoneContact);
        mButtonAddContact = findViewById(R.id.buttonAddContact);

        mDBHelper = new DBHelper(this);
        db = mDBHelper.getWritableDatabase();
    }

    public void onClickAddContact(View view) {
        ContentValues contentValues = new ContentValues();
        if (mNameContact.getText().toString().isEmpty()
                || mPhoneContact.getText().toString().isEmpty())
        {
            Toast.makeText(getApplicationContext(),"Пусті поля",
Toast.LENGTH_SHORT).show();
            finish();
        }
        else {
            contentValues.put(Contact.COLUMN_NAME_NAME,
mNameContact.getText().toString());
            contentValues.put(Contact.COLUMN_NAME_PHONE,
mPhoneContact.getText().toString());
```

```
            db.insert(Contact.TABLE_NAME,null,contentValues);
            Toast.makeText(getApplicationContext(),"Додано",
Toast.LENGTH_SHORT).show();
            db.close();
            finish();
        }

    }
}
```

Лістинг В.2 – Клас «AddNoteActivity»

```
package com.example.pannatuck.materialtest210618;


import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;

public class AddNoteActivity extends AppCompatActivity {

    EditText mNoteName;
    EditText mNoteDescription;
    EditText mNoteFolder;

    Spinner mSpinner;
    String mSelectedItem;

    DBHelper mDBHelper;
    SQLiteDatabase db;


    String mSelectedDate;

    @Override
    protected void onStart() {
        super.onStart();

        spinnerPutData();
    }
    @Override
    protected void onResume(){
        super.onResume();

        spinnerPutData();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_add_task);

        mSpinner = findViewById(R.id.spinnerContact);

        mTaskName = findViewById(R.id.editTextNameTask);
        mTaskDescription = findViewById(R.id.editTextDescriptionTask);
        mTaskMap = findViewById(R.id.editTextMapTask);

        Intent intent = getIntent();
        mSelectedDate = intent.getStringExtra("Date");

        mDBHelper = new DBHelper(this);
        db = mDBHelper.getWritableDatabase();
        spinnerPutData();


    }

    public void onClickAddNewContact(View view) {
        startActivity(new Intent(AddTaskActivity.this, AddContactActivity.class));
    }

    public void onClickAddNewNote (View view) {
        ContentValues contentValues = new ContentValues();
        if (mTaskName.getText().toString().isEmpty()
                || mTaskDescription.getText().toString().isEmpty()
                || mTaskMap.getText().toString().isEmpty())
        {
            Toast.makeText(getApplicationContext(),"Пусті поля",
Toast.LENGTH_SHORT).show();
            finish();
        }
        else {
            contentValues.put(Task.COLUMN_NAME_NOTE_NAME,
mTaskName.getText().toString());
            contentValues.put(Task.COLUMN_NAME_NOTE_DESCRIPTION,
mTaskDescription.getText().toString());
            contentValues.put(Task.COLUMN_NAME_NOTE_POINT,
mTaskMap.getText().toString());
            contentValues.put(Task.COLUMN_NAME_ID_CONTACT, getIdRecordingContact());
            contentValues.put(Task.COLUMN_NAME_DATE, mSelectedDate);
            //contentValues.put(Task.COLUMN_NAME_COUNT_TASK, ++counter);


            db.insert(Note.TABLE_NAME,null,contentValues);
            Toast.makeText(getApplicationContext(),"Додано",
Toast.LENGTH_SHORT).show();
            db.close();
            finish();
        }

    }

    private void spinnerPutData(){
        String[] columns = new String[] {Contact.COLUMN_NAME_NAME};
        Cursor cursor = db.query(Contact.TABLE_NAME, columns,
                null, null, null, null, null);

        List<String> list = new ArrayList<>();

        if(cursor.moveToFirst()) {
            String string;
```

```
            do {
                string = "";
                for (String cn : cursor.getColumnNames()) {
                    string = cursor.getString(cursor.getColumnIndexOrThrow(cn));
                }
                list.add(string);
            } while (cursor.moveToNext());
        }
        cursor.close();

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_item, list);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

        mSpinner.setAdapter(adapter);
        mSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
                                       int position, long id) {
                mSelectedItem = mSpinner.getSelectedItem().toString();
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
                Toast.makeText(getApplicationContext(),"On nothing selected",
Toast.LENGTH_SHORT).show();
            }
        });
    }


    private int getIdRecordingContact(){
        String query = "SELECT " + Contact._ID +
                " FROM " + Contact.TABLE_NAME +
                " WHERE " + Contact.COLUMN_NAME_NAME + " = ?";

        String[] selectionArgs = new String[]{mSelectedItem};

        Cursor testCursor = db.rawQuery(query, selectionArgs);

        int IdRecording = 0;
        if (testCursor != null) {
            if (testCursor.moveToFirst()) {
                String str;
                do {
                    str = "";
                    for (String cn : testCursor.getColumnNames()) {
                        str = testCursor.getString(testCursor.getColumnIndex(cn));
                    }
                    IdRecording = Integer.parseInt(str);
                } while (testCursor.moveToNext());

                testCursor.close();
            }
        }
        return IdRecording;
    }

    public void onClickWatchMap(View view) {
        String geoURI = "geo:46.971403, 32.005866?z=2";
        Uri geo = Uri.parse(geoURI);
        Intent geoIntent = new Intent(Intent.ACTION_VIEW, geo);
```

```
        if (geoIntent.resolveActivity(getPackageManager()) != null) {
            startActivity(geoIntent);
        }
    }
}
```

Лістинг В.3 – Клас «DBHelper»

```
package com.example.pannatuck.materialtest210618;


import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.provider.BaseColumns;

public class DBHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "AppDB";

    private static final String SQL_CREATE_FOLDER =
            "CREATE TABLE " +Folder.TABLE_NAME + " (" +
                    Users._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    Users.COLUMN_NAME_FOLDER + " TEXT, " +
                    Users.COLUMN_DATE + " TEXT);";

    private static final String SQL_CREATE_CONTACT =
            "CREATE TABLE " + Contact.TABLE_NAME + " (" +
                    Contact._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    Contact.COLUMN_NAME_NAME + " TEXT, " +
                    Contact.COLUMN_NAME_PHONE + " TEXT);";

    private static final String SQL_DROP_FOLDER =
            "DROP TABLE IF EXISTS " + Folder.TABLE_NAME;

    private static final String SQL_DROP_CONTACT =
            "DROP TABLE IF EXISTS " + Contact.TABLE_NAME;

    private static final String SQL_CREATE_TASK =
            "CREATE TABLE " + Task.TABLE_NAME + " (" +
                    Task._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    Task.COLUMN_NAME_ID_CONTACT + " INTEGER, " +
                    Task.COLUMN_NAME_MEETING_POINT + " TEXT, " +
                    Task.COLUMN_NAME_NOTE_DESCRIPTION + " TEXT, " +
                    Task.COLUMN_NAME_DATE + " TEXT, " +
                    Task.COLUMN_NAME_NOTE_NAME + " TEXT);";

    private static final String SQL_DROP_NOTE =
            "DROP TABLE IF EXISTS " + Note.TABLE_NAME;

    public DBHelper(Context context) { super(context, DATABASE_NAME, null,
DATABASE_VERSION); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_NOTE);
        db.execSQL(SQL_CREATE_CONTACT);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

```
        db.execSQL(SQL_DROP_TASK);
        db.execSQL(SQL_DROP_USERS);
        db.execSQL(SQL_DROP_CONTACT);

        onCreate(db);
    }
}

class Note implements BaseColumns
{
    public static final String TABLE_NAME = "note";
    public static final String COLUMN_NAME_MEETING_POINT = "meeting_point";
    public static final String COLUMN_NAME_ID_CONTACT = "_id_contact";
    public static final String COLUMN_NAME_TASK_NAME = "note_name";
    public static final String COLUMN_NAME_DATE = "date";
    public static final String COLUMN_NAME_TASK_DESCRIPTION = "description";
}

class Contact implements BaseColumns
{
    public static final String TABLE_NAME = "contact";
    public static final String COLUMN_NAME_NAME = "name";
    public static final String COLUMN_NAME_PHONE = "phone";
    public static final String COLUMN_NAME_PHONE = "email";

}
```

Лістинг В.4 – Клас «MainActivity»

```
package com.example.pannatuck.materialtest210618;


import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.text.format.DateUtils;
import android.view.View;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.List;

import devs.mulham.horizontalcalendar.HorizontalCalendar;
import devs.mulham.horizontalcalendar.utils.HorizontalCalendarListener;
```

```
public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    public static final String FORMAT_YYYYMMDD = "yyyy-MM-dd";
    public static final String FORMAT_YYYYMMDD_SLASHES = "yyyy/MM/dd";

    DBHelper mDBHelper;
    SQLiteDatabase db;
    List<String> list = new ArrayList<>();

    TextView mTextView;


    String mSelectedDate;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mTextView = findViewById(R.id.textViewTask);


        mDBHelper = new DBHelper(this);
        db = mDBHelper.getWritableDatabase();


        mSelectedDate = calendarToString(Calendar.getInstance(), "yyyy-MM-dd");
        getTask();

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.floatingButtonAddTask);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intentAddTask = new Intent(MainActivity.this,
AddTaskActivity.class);
                intentAddTask.putExtra("Date", mSelectedDate);
                startActivity(intentAddTask);
            }
        });

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        /*Календарь в верхней части экрана*/
        Calendar startDate = Calendar.getInstance();
        startDate.add(Calendar.MONTH,-1);

        Calendar endDate = Calendar.getInstance();
        endDate.add(Calendar.MONTH, 1);
```

```
        final HorizontalCalendar horizontalCalendar = new
HorizontalCalendar.Builder(this, R.id.calendarView)
                .range(startDate,endDate)
                .datesNumberOnScreen(5)
                .build();


        horizontalCalendar.setCalendarListener(new HorizontalCalendarListener() {
            @Override
            public void onDateSelected(Calendar date, int position) {
                mTextView.setText("");
                mSelectedDate = calendarToString(date, "yyyy-MM-dd");

                getTask();

//Toast.makeText(getApplicationContext(),mSelectedDate,Toast.LENGTH_SHORT).show();
            }

            @Override
            public boolean onDateLongClicked(Calendar date, int position) {
                return super.onDateLongClicked(date, position);
            }
        });


    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
```

```
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();

        if (id == R.id.nav_aboutProgram) {
            startActivity(new Intent(MainActivity.this, AboutProgramActivity.class));
        }

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    @Override
    protected void onResume() {
        super.onResume();

        getTask();
    }

    @Override
    protected void onPause() {
        super.onPause();

        mTextView.setText("");
    }

    @Override
    protected void onStop() {
        super.onStop();

        mTextView.setText("");
    }

    @Override
    protected void onStart() {
        super.onStart();

        mTextView.setText("");
    }

    public void getTask(){
        String[] projection = {
                Task.COLUMN_NAME_TASK_NAME,
                //Task.COLUMN_NAME_ID_CONTACT,
                //Task.COLUMN_NAME_DATE,
                Task.COLUMN_NAME_MEETING_POINT,
                Task.COLUMN_NAME_TASK_DESCRIPTION
        };

        String selection = Task.COLUMN_NAME_DATE + " = ?";
        String[] selectionArgs = {mSelectedDate};

        if (checkDate(mSelectedDate)){
            Cursor cursor = db.query(Task.TABLE_NAME, projection, selection,
selectionArgs,
                    null, null, null,null);

            while (cursor.moveToNext())
            {
                String name =
cursor.getString(cursor.getColumnIndex(Task.COLUMN_NAME_TASK_NAME));
```

```
//            String date =
cursor.getString(cursor.getColumnIndex(Task.COLUMN_NAME_DATE));
                String map =
cursor.getString(cursor.getColumnIndex(Task.COLUMN_NAME_MEETING_POINT));
                String description =
cursor.getString(cursor.getColumnIndex(Task.COLUMN_NAME_TASK_DESCRIPTION));


                mTextView.setTextSize(20);
                mTextView.append(("\n-----------------------------------------------
------------" +
                        "\nЗаголовок нотатку: " + name +
                        "\nТекст нотатку: " + description +
                        "\nПапка: " + map));

            }
            cursor.close();
        } else{
            Toast.makeText(getApplicationContext(), "Нет задач на этот день",
Toast.LENGTH_SHORT).show();
        }

    }

    public boolean checkDate(String date){
        String query = "SELECT " + Task.COLUMN_NAME_DATE + " FROM " +
Task.TABLE_NAME;

        Cursor testCursor = db.rawQuery(query, null);
        if (testCursor != null) {
            if (testCursor.moveToFirst()) {
                String str;
                do {
                    str = "";
                    for (String cn : testCursor.getColumnNames()) {
                        str = testCursor.getString(testCursor.getColumnIndex(cn));
                    }
                    if (str.equals(date)){
                        return true;
                    }
                } while (testCursor.moveToNext());

                testCursor.close();
            }
        }
        return false;
    }

    public static final String calendarToString(Calendar cal, String dateformat) {
        StringBuffer ret = new StringBuffer();
        if(dateformat.equals(FORMAT_YYYYMMDD) ) {
            ret.append(cal.get(Calendar.YEAR));
            ret.append("-");

            String month = null;
            int mo = cal.get(Calendar.MONTH) + 1; /* Calendar month is zero indexed,
string months are not */
            if(mo < 10) {
                month = "0" + mo;
            }
            else {
                month = "" + mo;
```

```
            }
            ret.append(month);

            ret.append("-");

            String date = null;
            int dt = cal.get(Calendar.DATE);
            if(dt < 10) {
                date = "0" + dt;
            }
            else {
                date = "" + dt;
            }
            ret.append(date);
        }
        return ret.toString();
    }
    public void onClickDropMenu(MenuItem item) {
        String geoURI = "geo:46.971403, 32.005866?z=2";
        Uri geo = Uri.parse(geoURI);
        Intent geoIntent = new Intent(Intent.ACTION_VIEW, geo);
        if (geoIntent.resolveActivity(getPackageManager()) != null) {
            startActivity(geoIntent);
        }
    }
}
```

Лістинг В.5 – Клас «SettingsActivity»

```
package com.example.pannatuck.materialtest210618;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class AboutProgramActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about_program);    }
```

Додаток Д - Програма та методика випробувань програмного забезпечення

1 Об’єкт випробування

Об’єктом випробування є програмне забезпечення ««Додаток ведення нотаток» для Android», що розробляється на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

2 Мета випробування

Метою проведення випробування є перевірка працездатності даного програмного забезпечення, перевірка відповідності характеристик та вимог розробленої програми, викладених у технічному завдані, приведення прикладу роботи та отримання відповідних навичок.

3 Вимоги до додатка

При проведенні випробувань функціональні характеристики (можливості) програми підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

4 Вимоги до програмної документації

Програмна документація повинна включати в себе наступні документи:

1. Текст програми.
2. Опис програми.
3. Програма та методика випробувань.
4. Інструкція користувача.

5 Програма та методика випробувань програмного забезпечення

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувалися під час випробувань:

– Процессор: 8 ядер, 2 ГГц, 64 бита;

– Оперативная память: 3/4 ГБ;

– 300 МВ вільного місця.

5.2 Програмні засоби, що використовуються під час випробувань

Склад програмних засобів, що використовувалися під час випробувань:

- операційна система
- ОС Android версії 5 і вище;
- Java Runtime Environment (JRE 1.8).

5.3 Порядок проведення випробувань

Порядок проведення випробувань складається з перевірки вимог до функціональних характеристик програмного продукту та перевірки вимог до програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Перевірка дотримання вимог програмної документації на програмний продукт виконується візуально представником служби, відповідальної за експлуатацію. У ході перевірки зіставляється склад і комплектність програмної документації, представленої розробником, з переліком програмної документації, наведеним у пункті «Склад програмної документації, пропонованої на випробування» цього документа. Перевірка вважається завершеною у випадку відповідності складу та

комплектності програмної документації, представленої розробником, переліком програмної документації, наведеному у зазначеному вище пункті.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності програми виконується згідно з пунктом «Вимоги до функціональних характеристик» Додатку А – Технічне завдання. Перевірка вважається завершеною у разі відповідності складу і послідовності виконаних дій пункту «Вимоги до функціональних характеристик». В процесі тестування була перевірена функціональність за темою ««Додаток ведення нотаток» для Android». У табл. Д.1, що наведена нижче, вказано перелік випробувань основних функціональних можливостей

Таблиця Д.1 - Методика випробування

№	Дія	Очікуваний результат	Результат перевірки	Зауваження
1	2	3	4	5
1	Перевірка правильності введення даних при авторизації	Програма повинна перевірити введені дані з записами у БД	Виконано	
2	Перевірка правильності введення даних при реєстрації	Програма повинна перевірити коректність введених даних	Виконано	
3	Перевірка правильності додавання нового запису в БД	Програма повинна перевірити коректність введених даних	Виконано	

Продовження табл. Д.1

1	2	3	4	5
4	Перевірка правильності введених даних при редагуванні записів в БД	Програма повинна перевірити коректність введених даних	Виконано	
5	Перевірка правильності видалення запису з БД	Програма повинна перевірити введені дані з записами у БД	Виконано	
6	Перевірка правильності введених даних при пошуку записів в БД	Програма повинна перевірити введені дані з записами у БД	Виконано	
7	Перевірка правильності формування звітів	Програма повинна перевірити введені дані з записами у БД	Виконано	