

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

**Пояснювальна записка**

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ КАЛЕНДАРЯ  
РОБОЧИХ ЗМІН  
DEVELOPMENT OF A MOBILE APPLICATION FOR WORK SHIFT  
CALENDAR

Виконав(ла): студент(ка) 4 курсу, групи КНТ-151  
Спеціальності 121 Інженерія програмного  
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЯХНО Д.А.

(ПРІЗВИЩЕ та ініціали)

Керівник ГЛАДКОВА О.М.

(ПРІЗВИЩЕ та ініціали)

Рецензент ДЬЯЧУК Т.С.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет КНТ  
Кафедра програмних засобів  
Ступінь вищої освіти бакалавр  
Спеціальність 121 Інженерія програмного забезпечення  
(код і найменування)  
Освітня програма (спеціалізація) Інженерія програмного забезпечення  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н, проф.  
Сергій СУББОТІН  
“      ”        2025 року

**З А В Д А Н Н Я**  
**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

ЯХНА Дмитра Анатолійовича  
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка мобільного застосунку календаря робочих змін  
Development of a Mobile Application for Work Shift Calendar

керівник проєкту (роботи) к.т.н., ГЛАДКОВА Ольга Миколаївна,  
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

| Розділ              | ПРИЗВИЩЕ, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|-------------------------------------------|----------------|---------------------------|
|                     |                                           | завдання видав | прийняв виконане завдання |
| 1-4 Основна частина | ГЛАДКОВА О.М., доцент                     |                |                           |
| Нормоконтроль       | БЄЛОВА А.В., асистент                     |                |                           |

7. Дата видачі завдання “ 15 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)                             | Строк виконання етапів проєкту ( роботи ) | Примітка     |
|-------|----------------------------------------------------------------------|-------------------------------------------|--------------|
| 1     | Постановка завдання роботи.                                          | 1 тиждень                                 | Завдання, ТЗ |
| 2     | Аналіз предметної області.                                           | 1 тиждень                                 | Розділ 1     |
| 3     | Вибір мови програмування та інших технологій розробки.               | 2 тиждень                                 | Розділ 2     |
| 4     | Розробка архітектури програми.                                       | 2 тиждень                                 | Розділ 3     |
| 5     | Розробка програми.                                                   | 3-4 тижні                                 | Розділ 3, 4  |
| 6     | Тестування та експериментальне дослідження програмного забезпечення. | 5 тиждень                                 | Розділ 4     |
| 7     | Оформлення пояснювальної записки та документів до неї.               | 6 тиждень                                 | Додатки      |
| 8     | Нормоконтроль та рецензування.                                       | 7 тиждень                                 |              |
| 9     | Захист роботи.                                                       | 8 тиждень                                 |              |

Студент(ка)

ПІДПИС

( підпис )

Дмитро ЯХНО

(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

( підпис )

Ольга ГЛАДКОВА

(Імя ПРИЗВИЩЕ)

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 80 с., 32 табл., 24 рис., 3 дод., 26 джерел.

ДОХІД, РОБОЧА ЗМІНА, СТАВКА, ДОДАТОК, JAVA, ANDROID STUDIO

Об'єкт дослідження – процес розроблення додатку для обліку робочого процесу працівника.

Предмет дослідження – програмні засоби для обліку робочого процесу.

Мета роботи – створення мобільного додатку для автоматизації обліку робочого часу та доходу працівника для особистого використання у телефоні.

Матеріали, методи та технічні засоби: система керування базами даних SQLite, мова програмування Java, розширювана мова розмітки XML, інтегроване середовище розробки Android Studio, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено Android-додаток, який дозволяє у календарному форматі представляти і редагувати інформацію про робочий графік користувача, автоматично створювати звіт про дохід на основі цих даних.

Висновки. Розроблено додаток для автоматизації обліку робочого процесу працівника.

Галузь використання – особисте використання особами, що працюють.

## **ABSTRACT**

Explanatory note to the diploma qualifying work of the bachelor: 80 pages, 32 tables, 12 figures, 3 appendixes, 26 sources.

INCOME, WORK SHIFT, WAGE, APPLICATION, JAVA, ANDROID STUDIO

The object of research is the process of developing an application for recording the employee's workflow.

The subject of research is software for recording the workflow.

The purpose of the work is to create a mobile application to automate the accounting of working hours and employee income for personal use on the phone.

Materials, methods and technical means: SQLite database management system, Java programming language, XML extensible markup language, AndroidStudio integrated development environment, personal computer with Intel Core i5 processor running Microsoft Windows 10 operating system.

Results. An Android application was created that allows you to present and edit information about the user's work schedule in a calendar format, and automatically create an income report based on this data.

Conclusions. An application has been developed to automate the accounting of an employee's workflow.

Field of use is a personal use by working individuals.

## ЗМІСТ

|                                                                           |    |
|---------------------------------------------------------------------------|----|
|                                                                           | С. |
| Перелік скорочень та умовних позначень .....                              | 8  |
| Вступ.....                                                                | 9  |
| 1 Аналіз проблеми та постановка завдань дослідження .....                 | 10 |
| 1.1 Формулювання вимог до програмного забезпечення .....                  | 10 |
| 1.2 Опис аналогів.....                                                    | 11 |
| 1.2.1 Google Календар.....                                                | 11 |
| 1.2.2 ShifterCalendar.....                                                | 12 |
| 2 Матеріали і методи.....                                                 | 14 |
| 2.1 Обґрунтування вибору операційної системи .....                        | 14 |
| 2.2 Обґрунтування вибору мови програмування.....                          | 15 |
| 2.3 Обґрунтування вибору IDE .....                                        | 18 |
| 2.4 Обґрунтування вибору СКБД.....                                        | 20 |
| 3 Опис програми.....                                                      | 21 |
| 3.1 Опис класів об'єктів.....                                             | 21 |
| 3.2 Опис структури БД .....                                               | 23 |
| 3.3 Опис інтерфейсу програми.....                                         | 24 |
| 3.4 Опис основних функціональних компонентів .....                        | 27 |
| 3.4.1 Головна сторінка.....                                               | 28 |
| 3.4.2 Редактор змін.....                                                  | 30 |
| 3.4.3 Створювач змін .....                                                | 31 |
| 3.4.4 Звіт з доходу.....                                                  | 32 |
| 3.4.5 Налаштування автоматичного заповнення .....                         | 34 |
| 3.5 Опис додаткових компонентів.....                                      | 35 |
| 3.5.1 Опис адаптерів списків об'єктів.....                                | 35 |
| 3.5.2 Опис класу взаємодії з БД.....                                      | 36 |
| 4 Експлуатація, тестування та експериментальне дослідження програми ..... | 38 |
| 4.1 Методика роботи користувача із застосунком.....                       | 38 |

|                                    |    |
|------------------------------------|----|
| 4.2 Функціональне тестування ..... | 51 |
| 4.3 Тестування інтерфейсу .....    | 52 |
| Висновки .....                     | 55 |
| Перелік джерел посилання .....     | 56 |
| Додаток А Технічне завдання .....  | 58 |
| Додаток Б Текст програми .....     | 62 |
| Додаток В Слайди презентації ..... | 73 |

**ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ**

|      |                                      |
|------|--------------------------------------|
| IDE  | - Integrated Development Environment |
| SQL  | - Structured Query Language          |
| XML  | - Extensible Markup Language         |
| БД   | - база даних                         |
| ОС   | - операційна система                 |
| ПЗ   | - програмне забезпечення             |
| СКБД | - система керування базами даних     |

## ВСТУП

Інформаційні технології вже давно впроваджуються суспільством для вирішення різноманітних проблем у різних сферах нашого життя. Програмне забезпечення має високу швидкість роботи, гнучкість у використанні і легко розповсюджується, що робить його чудовим інструментом для вирішення різноманітних задач.

Одним з варіантів програмної реалізації рішення поставленої проблеми є створення мобільного застосунку. Мобільні пристрої неймовірно широко розповсюджені і вже давно стали невід'ємною частиною нашого життя. Тому створення рішення до поставленої проблеми у вигляді мобільного застосунку сприятиме його широкому розповсюдженню, зручності і легкості використання.

Поставленою проблемою є облік робочого процесу найманого працівника. Володіння інформацією про робочий час і дохід є необхідним для розуміння фінансового положення працюючої особи, що є вкрай необхідним у сучасному світі. Створене ПЗ також допоможе уникати помилок і вирішувати конфлікти на тему заробітної плати, що суттєво підвищить якість роботи і мотивацію працівника. Враховуючи, що працюючі особи складають велику частину суспільства [1], рішення цієї проблеми має високий попит.

# 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

## 1.1 Формулювання вимог до програмного забезпечення

Робочий процес працівника можна охарактеризувати певним набором ознак, що підлягають аналізу і дають змогу оцінити його. Серед таких ознак можна виділити:

- тарифна ставка - годинний (денний або місячний) розмір грошової компенсації працівника за участь у робочому процесі [2];
- робоча зміна - період часу, в котрий працівник приймає участь у робочому процесі, робоча зміна має такі характеристики, як час початку і час завершення, на основі котрих вираховується загальна тривалість робочого часу, і тарифна ставка, на основі цих даних можна визначити розмір заробленої за відпрацювання зміни суми грошей;
- робочий день(робочий запис) - це інформація про те, як працівник приймав участь у робочому процесі у певний календарний день, робочий день характеризується фактом відпрацювання працівником певної робочої зміни або відсутністю участі у робочому процесі.

Збирання і аналіз цих даних традиційними методами є досить складною задачею. Тому існує попит на створення доступної, невибагливої реалізації рішення поставленої проблеми.

Відповідно до цього було сформульовано низку вимог до програмного забезпечення, що розробляється. Функціональні вимоги:

- можливість створювати і видаляти робочі зміни, переглядати їх характеристики;
- можливість додавати і видаляти робочі записи, шляхом присвоєння робочих змін;
- можливість переглядати робочі записи у структурованому форматі;
- можливість створювати звіти на основі доданих робочих записів;

- можливість налаштовувати автоматичне створення записів для полегшення роботи з ПЗ, що розробляється.

Вимоги до інтерфейсу:

- легкий, лаконічний інтерфейс;
- легкі засоби керування роботою програми;
- інформативні підказки до заповнення форм.

## 1.2 Опис аналогів

### 1.2.1 Google Календар

Google Календар - це один з найпопулярніших календарних додатків для мобільних пристроїв у світі. Він надає користувачам великий вибір інструментів для планування часу. У додатку можна створювати розпорядки, задачі, особисті і багатокористувацькі події. Календар підтримує синхронізацію з бронюваннями і надає змогу переглядати всі події одночасно [3]. Головну сторінку застосунку наведено на рисунку 1.1

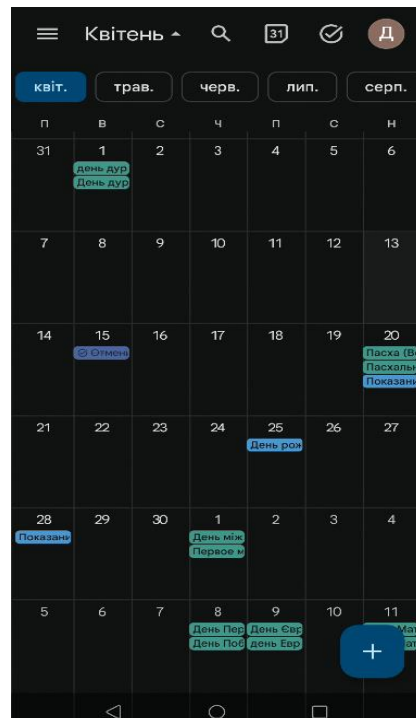


Рисунок 1.1 – Головна сторінка Google Календаря

Цей додаток не втілює поставлені функціональні вимоги, але має зразкову реалізацію інтерфейсу і інструменти для роботи з програмою.

### 1.2.2 ShifterCalendar

ShifterCalendar - розроблений компанією LRHSoft застосунок, що надає користувачам велику кількість можливостей для організації робочого процесу [4]. Застосунок володіє такими видатними особливостями:

- створення і керування робочими записами, що містять інформацію. про робочий час та дохід;
- створення нотатків і нагадувань;
- віджети робочого столу;
- інтеграція з іншими календарними застосунками для легкого перенесення даних.

Головну сторінку застосунку наведено на рисунку 1.2.



Рисунок 1.2 – Головна сторінка ShifterCalendar

Застосунок чудово відповідає сформованим функціональним вимогам, але має дещо застарілий і сильно перевантажений інтерфейс, який окрім того додатково дезорієнтує користувача наявністю реклами.

## **2 МАТЕРІАЛИ І МЕТОДИ**

### **2.1 Обґрунтування вибору операційної системи**

Вибір операційної системи для розроблення мобільного застосунку проводиться з двох варіантів, які покривають майже 100% всіх мобільних пристроїв у світі [5] - Android та IOS.

Android – створена у 2008 році компанією “Google” мобільна операційна система. Вона належить до класу UNIX-подібних ОС і базується на платформі Java Virtual Machine [6]. З 2012 року Android є найпопулярнішою мобільною ОС у світі [7]. До переваг Android можна віднести:

- відкритість – Android базується на відкритій ОС Linux, що дає виробникам мобільних пристроїв широкі можливості налаштувань під власні потреби;
- кастомізація – пристрої на базі Android надають користувачам широку можливість для налаштування як інтерфейсу, так і функціонування пристрою.

До недоліків ОС можна віднести безпеку. Через велику кількість змін, що вносяться виробниками пристроїв до базової ОС, не можна гарантувати, що конкретний Android-пристрій відповідає стандартам безпеки і працездатності.

IOS - створена 2007 року компанією “Apple”, це особиста мобільна операційна система для пристроїв компанії. З моменту виходу IOS домінувала на ринку мобільних пристроїв, доки Android не вирвалася уперед у 2011 році. До переваг ОС можна віднести:

- високий рівень безпеки – ОС ретельно обслуговується компанією-розробником, що є причиною високого рівня стабільності її роботи;
- загальна оновлюваність системи – вихід оновлень для ОС відбувається на всіх підтримуваних компанією пристроях;
- тривале обслуговування – загальна оновлюваність забезпечує довготривалу підтримку конкретної моделі пристроїв компанії.

До недоліків ОС можна віднести:

- закритість. IOS не підтримує встановлення сторонніх додатків, що обмежує можливості користувача при користуванні пристроями з цією ОС;
- обмежена кастомізація не дає користувачеві можливості тонко налаштовувати роботу пристрою під власні потреби.

Під час вибору операційної системи для розробки застосунку основним критерієм виступала доля операційної системи на ринку мобільних пристроїв у порівнянні з конкурентом. Порівняння описаних ОС наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння Android та IOS

| Операційна система | Доля на ринку мобільних пристроїв, % |
|--------------------|--------------------------------------|
| Android            | 70.93                                |
| IOS                | 29,07                                |

За результатами порівняння долі операційних система на ринку мобільних пристроїв було обрано ОС Android. Розробка застосунку саме для цієї ОС суттєво збільшить потенційне число користувачів застосунку.

## 2.2 Обґрунтування вибору мови програмування

Мобільна розробка є однією з найпопулярніших сфер програмування сьогодення. Величезне поширення мобільних пристроїв спонукало створення великої кількості засобів програмування їх, довгострокову підтримку і постійне покращення. Тому при виборі мови програмування для розробки мобільного додатку наявни широкий вибір варіантів, пристосованих для певних задач.

Найбільш популярними мовами програмування для мобільної розробки є мови наведені нижче.

Java. Випущена 1995 року компанією “Sun Microsystems”, з 2009 року підтримувана компанією “Oracle”, є об’єктно-орієнтованою мовою програмування, і однією з найбільш популярних мов у світі загалом. Однією з ключових особливостей Java, яка стала основною підставою для її створення, є кросплатформеність цієї мови. Вона базується на принципі “Write once - run everywhere” (“Напиши один раз - запусти всюди”), який реалізовано шляхом створення Java Virtual Machine [8]-[9].

Kotlin. Випущена 2010 року компанією “JetBrains”, є статично типізованою об’єктно-орієнтованою мовою програмування, створеною як альтернативу Java. Kotlin також працює на базі JVM. До особливостей мови можна віднести більш лаконічний синтаксис, що сприяє ефективному процесу розробки, і покращену безпеку у вигляді розділення типів на нульові і ненульові на рівні мови [10]-[11].

Dart (Flutter). Випущена 2011 року компанією “Google”, є об’єктно-орієнтованою мовою програмування, спрямованою на розробку кросплатформенних мобільних додатків. Dart працює у парі з фреймворком Flutter, також розробленим “Google” для нативно компільованих кросплатформенних додатків. Головною перевагою мови є її висока продуктивність. Однак Dart має значну меншу популярність за інші зазначені мови, а отже і меншу кількість користувацьких бібліотек і навчальних матеріалів, що може ускладнити розробку [12]-[13].

C# (.NET MAUI). Випущена 2001 року компанією “Microsoft”, є об’єктно-орієнтованою мовою програмування, створеною як альтернативу Java для розробки на ОС Windows. Мобільна розробка забезпечується використанням .NET MAUI - платформи для створення кросплатформенних мобільних додатків [14]-[15].

Порівняння описаних мов програмування за декількома важливими критеріями наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння мов програмування

| Критерії порівняння   | Мови програмування                               |                                                    |                                                  |                                                  |
|-----------------------|--------------------------------------------------|----------------------------------------------------|--------------------------------------------------|--------------------------------------------------|
|                       | Java                                             | Kotlin                                             | Dart                                             | C#                                               |
| Продуктивність        | Помірна (знижена через використання JVM)         | Помірна (знижена через використання JVM)           | Висока (нативна компіляція за допомогою Flutter) | Помірна (знижена через використання .NET MAUI)   |
| Синтаксис             | Базується на C/C++, популярний, середня важкість | Сильно базується на Java, трохи збільшена важкість | Базується на C/C++, популярний, середня важкість | Базується на C/C++, популярний, середня важкість |
| Підтримка спільнотою  | Дуже висока                                      | Висока (активно збільшується)                      | Помірна                                          | Висока                                           |
| Підтримка розробником | Висока                                           | Висока                                             | Висока                                           | Висока                                           |

За результатами порівняння можна зробити висновок, що всі описані мови програмування мають досить високі оцінки за обраними критеріями, чим й обумовлена їх популярність у сфері мобільної розробки.

За показником продуктивності найкращою є Dart, але інші порівнянні мови не демонструють поганих показників за цим критерієм. Це робить Dart не найкращим варіантом через низький у порівнянні з конкурентами рівень підтримки спільнотою. За цим критерієм виділяється Java, найстарша та найпопулярніша мова програмування з описаних. За іншими критеріями вона не уступає конкурентам, тому для розроблення застосунку було обрано саме її.

Високий рівень підтримки спільнотою надає широкий вибір інструментів для розробки.

## 2.3 Обґрунтування вибору IDE

Вибір інтегрованого середовища розробки є одним з найважливіших етапів планування розробки програмного забезпечення. Саме у IDE буде проходити основний процес розробки, тому наявність чи відсутність підтримки середовищем різноманітних інструментів, швидкодія середовища та інші критерії суттєво впливають на процес розробки ПО.

Найпопулярніші сучасні IDE наведені далі.

Microsoft Visual Studio - це випущене в 1997 році компанією “Microsoft” IDE, створене для підтримки розробки різними мовами програмування. Середовище має широкий набір інструментів для розробки як консольних програм, так і програм з інтерфейсом [16].

Visual Studio Code - це випущений у 2015 компанією “Microsoft” редактор коду, головною особливістю якого є висока швидкодія. Редактор має систему користувацьких розширень, що забезпечує можливість розробки різноманітних програм на багатьох мовах програмування [17].

Android Studio - це випущене у 2014 році компанією “Google” IDE, спрямоване саме на розробку мобільних застосунків на базі ОС Android. Середовище базується на системі складання Gradle і підтримує низку функцій спрямованих на підвищення ефективності розробки мобільних застосунків, так як от емуляція роботи застосунку на популярних Android-пристроях [18].

Порівняння описаних IDE наведено у таблиці 2.3

Таблиця 2.3 – Порівняння IDE

| Критерії оцінювання | IDE                     |                                                           |                |
|---------------------|-------------------------|-----------------------------------------------------------|----------------|
|                     | Microsoft Visual Studio | Visual Studio Code                                        | Android Studio |
| 1                   | 2                       | 3                                                         | 4              |
| Швидкодія           | Помірна                 | Висока (може бути значно змінена використанням розширень) | Помірна        |

Кінець таблиці 2.3

| 1                                 | 2                                                       | 3                                                                                  | 4                                                                     |
|-----------------------------------|---------------------------------------------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| Ресурсозатратність                | Висока                                                  | Низька (може бути значно змінена використанням розширень)                          | Висока                                                                |
| Підтримувані мови програмування   | Широкий список мов, підтримуваних платформою .NET       | Майже будь яка мова програмування (за рахунок розширень)                           | Java, Kotlin                                                          |
| Інтерфейс                         | Дещо застарілий                                         | Сучасний, зручний                                                                  | Сучасний, зручний                                                     |
| Можливості розширення функціоналу | Помірні                                                 | Дуже високі (IDE базується на роботі з розширеннями, в тому числі користувацькими) | Високі (Gradle дозволяє зручне використання користувацьких бібліотек) |
| Передвстановлені інструменти      | Широкий набір інструментів для обраної області розробки | Мінімальний                                                                        | Широкий набір інструментів для мобільної розробки                     |

За результатами порівняння IDE можна зробити висновок що Visual Studio Code випереджає конкурентів за показником продуктивності. Але це досягається шляхом мінімізації передвстановлених інструментів розробки, що змушує користувача власноруч встановлювати і налаштовувати необхідні компоненти. Це робить Visual Studio Code не найбільш оптимальним варіантом, оскільки об'єм ресурсів ПК, на якому буде проводитися розробка застосунку, дозволяє ефективно працювати з іншими описаними IDE попри їх нижчу продуктивність.

Для розробки застосунку було обрано IDE Android Studio оскільки, це середовище має велику кількість передвстановлених інструментів, спрямованих саме на мобільну розробку, деякі з котрих є відсутніми у конкурентів, як от емуляція Android-пристроїв.

## 2.4 Обґрунтування вибору СКБД

СКБД – це набір програмних засобів для керування пов'язаними даними. Використання СКБД дозволяє зберігати дані, необхідні для роботи ПЗ, у структурованому форматі, утворювати зв'язки між цими даними, що спрощує додавання, видалення, перетворення, аналіз й інші операції над цими даними.

Для розробки застосунку було використано СКБД SQLite. SQLite - це створена у 2000 році розробником Річардом Хіппом реляційна СКБД, втілена у вигляді бібліотеки, що зберігає БД як частину застосунку. SQLite є найбільш поширеною СКБД у світі і одним із найбільш поширених програмних засобів загалом у світі [19]-[20].

SQLite було обрано через низку переваг, які роблять цю СКБД ідеальним рішенням для застосунку, що розробляється:

- локальність СКБД жодним чином не обмежує можливості при використанні у оффлайн-застосунку;
- малий у порівнянні з конкурентами розмір СКБД, що досягається за рахунок відсутності серверної частини, позитивно впливає на продуктивність застосунку;
- ця СКБД є передвстановленою на всіх мобільних пристроях, що виробляються, що сприяє максимізації потенційних користувачів застосунку, що розробляється.

### **3 ОПИС ПРОГРАМИ**

#### **3.1 Опис класів об'єктів**

Клас становить основу об'єктно-орієнтованої парадигми програмування.

Клас в ООП - це структура, що складається з пов'язаних між собою змінних і функцій, і є програмною інтерпретацією певного об'єкту або явища.

Клас має такі властивості:

- змінні, що описують характеристики втіленого об'єкта, називаються полями класу;
- функції, що описують поведінку об'єкта, називаються методами класу.

Для розробки застосунку було розроблено класи для наступних об'єктів:

а) “Робоча зміна”, характеристики:

- 1) назва зміни;
- 2) час початку зміни;
- 3) час завершення зміни;
- 4) ставка;

б) “Робочий запис”, характеристики:

- 1) дата запису;
- 2) робоча зміна.

Властивості створених класів “ShiftType” і “ShiftRecord” наведено у таблицях 3.1 і 3.2.

Таблиця 3.1 – Властивості класу “ShiftType”

| Назва        | Вид   | Тип даних | Опис                                                                   |
|--------------|-------|-----------|------------------------------------------------------------------------|
| id           | Поле  | int       | Містить унікальний номер зміни для ідентифікації серед інших           |
| name         | Поле  | String    | Містить назву зміни                                                    |
| startTime    | Поле  | String    | Містить час початку зміни                                              |
| endTime      | Поле  | String    | Містить час завершення зміни                                           |
| color        | Поле  | String    | Містить HEX-код кольору зміни для графічного відображення у застосунку |
| wage         | Поле  | double    | Містить погодинну ставку зміни                                         |
| getId        | Метод | int       | Повертає id зміни                                                      |
| getName      | Метод | String    | Повертає назву зміни                                                   |
| getStartTime | Метод | String    | Повертає час початку зміни                                             |
| getEndTime   | Метод | String    | Повертає час завершення зміни                                          |
| getColor     | Метод | String    | Повертає колір зміни                                                   |
| getWage      | Метод | double    | Повертає ставку зміни                                                  |

Таблиця 3.2 - Властивості класу “ShiftRecord”

| Назва      | Вид   | Тип даних | Опис                                        |
|------------|-------|-----------|---------------------------------------------|
| date       | Поле  | String    | Містить дату робочого запису                |
| shiftId    | Поле  | int       | Містить id відповідної запису робочої зміни |
| getDate    | Метод | String    | Повертає дату запису                        |
| getShiftId | Метод | int       | Повертає id робочої зміни запису            |

### 3.2 Опис структури БД

Відповідно до сформульованих для роботи програми об'єктів було створено наступні таблиці БД:

- таблиця “shift\_types” відповідно об'єкту “Робоча зміна”;
- таблиця “shift\_records” відповідно об'єкту “Робочий запис”.

Опис полів таблиць наведено у таблиця 3.3. і 3.4.

Таблиця 3.3 – Властивості таблиці БД “shift\_types”

| Назва поля | Тип даних | Відповідник класу “ShiftType” |
|------------|-----------|-------------------------------|
| id         | INTEGER   | id                            |
| name       | TEXT      | name                          |
| start_time | TEXT      | startTime                     |
| end_time   | TEXT      | endTime                       |
| wage       | REAL      | wage                          |
| color      | TEXT      | color                         |

Таблиця 3.4 – Властивості таблиці БД “shift\_records”

| Назва поля    | Тип даних | Відповідник класу “ShiftRecord” |
|---------------|-----------|---------------------------------|
| date          | TEXT      | date                            |
| shift_type_id | INTEGER   | shiftId                         |

Структуру розробленої БД наведено на рисунку 3.1.

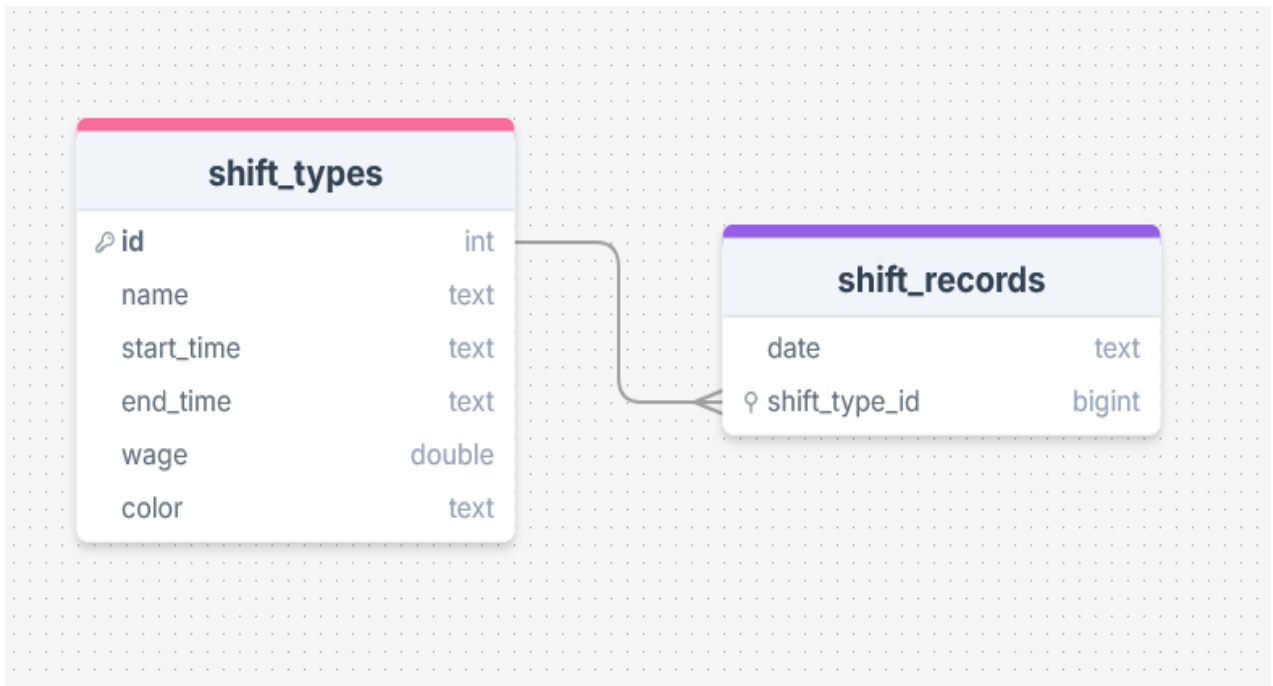


Рисунок 3.1 – Структура БД

Утворений між таблицями бази даних зв'язок забезпечує каскадне видалення об'єктів “Робочий запис” при видаленні відповідного об'єкту “Робоча зміна”

### 3.3 Опис інтерфейсу програми

Розробка інтерфейсу Android-застосунку відбувається шляхом створення макетів інтерфейсу користувача мовою XML.

Макет - це структура інтерфейсу певного компонента застосунку, що розробляється. Всі елементи макету представляють собою складові ієрархії класів “ViewGroup” і “View” [21].

Основні розроблені макети інтерфейсу визначають зовнішній вигляд сторінок застосунку. До основних розроблених макетів інтерфейсу належать:

- activity\_main;
- activity\_shift\_editor;
- activity\_shift\_creator;
- activity\_income\_report;

- activity\_autofill.

activity\_main – макет інтерфейсу початкової сторінки застосунку.

Містить такі елементи:

- текстове поле відображення поточного місяця;
- текстові поля зі скороченими назвами днів тижня;
- макет, що відображає календарні дні;
- макет, що відображає список змін;
- кнопки навігації між місяцями;
- кнопки навігації програми, що викликають інші основні

компоненти застосунку.

activity\_shift\_editor – макет інтерфейсу сторінки, на якій відображається інформація про зміни. Містить такі елементи:

- текстове поле з назвою сторінки;
- макет, що відображає список змін з інформацією про них;
- кнопки навігації програми, що викликають інші основні

компоненти застосунку.

activity\_shift\_creator – макет інтерфейсу сторінки, на якій надається можливість створити нову зміну. Містить такі елементи:

- текстове поле з назвою сторінки;
- поля вводу даних, що характеризують зміну;
- тестові поля-підказки для вводу даних;
- функціональна кнопка, що викликає спробу створення нової зміни

за введеними даними;

- кнопки навігації програми, що викликають інші основні

компоненти застосунку.

activity\_income\_report – макет сторінки отримання звітності за певний календарний період. Містить такі елементи:

- текстове поле з назвою сторінки;
- поля вводу даних про календарний період;
- тестові поля-підказки для вводу даних;

- текстові поля, що демонструють отримані дані;
- функціональна кнопка, що викликає спробу отримати дані за вказаним календарним періодом;
- кнопки навігації програми, що викликають інші основні компоненти застосунку.

`activity_autofill` – макет сторінки налаштування автоматичного заповнення календаря. Містить такі елементи:

- текстове поле з назвою сторінки;
- флаг, що відповідає за активацію і деактивацію функції автоматичного заповнення;
- поле вводу даних для налаштування функції;
- текстове поле-підказка до поля вводу даних;
- функціональна кнопка, що викликає спробу змінити налаштування функції;
- кнопки навігації програми, що викликають інші основні компоненти застосунку.

Додаткові макети інтерфейсу визначають зовнішній вигляд елементів макетів на основних сторінках. До розроблених додаткових макетів інтерфейсу належать:

- `calendar_cell`;
- `shift_cell`;
- `shift_editor_cell`.

`calendar_cell` – макет календарного дня для відображення на головній сторінці. Містить такі елементи:

- текстове поле, що відображає номер календарного дня;
- фон, що відображає факт відпрацювання зміни відповідного календарного дня.

`shift_cell` – макет зміни для відображення на головній сторінці застосунку. Містить такі елементи:

- текстове поле для відображення назви зміни;

- фон, що слугує ідентифікатором зміни серед інших.

`shift_editor_cell` – макет зміни для відображення на сторінці редагування змін. Містить такі елементи:

- текстове поле для відображення назви зміни;
- текстові поля для відображення характеристик зміни;
- функціональну кнопку, що видаляє відповідну зміну.

### **3.4 Опис основних функціональних компонентів**

Базова структура розробленого застосунку складається з п'яти основних компонентів, пов'язаних з основними макетами користувацького інтерфейсу.

При розробці Android-застосунку основним компонентом програми виступає `Activity`. `Activity` – це клас, що втілює пов'язаний з макетом інтерфейсу компонент застосунку, який фокусується на наданні користувачеві можливості виконати певний набір задач. Активність реалізує набір функцій, які відповідають етапам її життєвого циклу. При розробці застосунку виклик виконання поставлених функцій забезпечується у методі `onCreate`, що викликається при створенні `Activity` [22]. Розроблені компоненти забезпечують виконання поставлених до застосунку функціональних вимог.

Структуру основних компонентів застосунку зображено на рисунку 3.2.

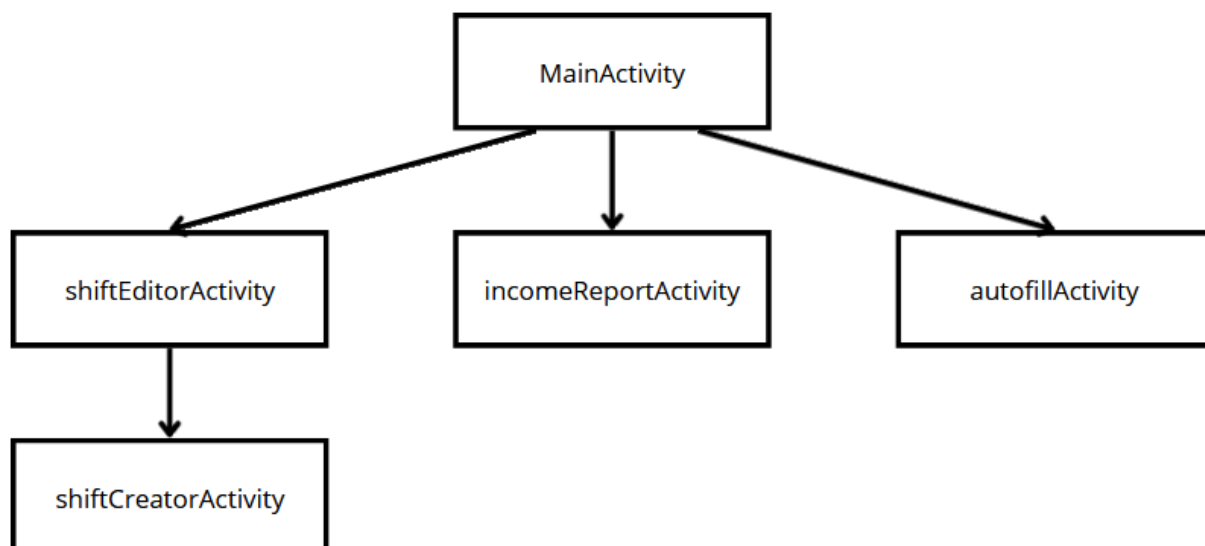


Рисунок 3.2 – Структура основних компонентів

### 3.4.1 Головна сторінка

MainActivity – це головний компонент застосунку, що створюється при його запуску. Цей компонент забезпечує виконання однієї з основних вимог до розробленого застосунку - відображення робочих записів у календарному форматі. Також, саме він реалізує створення нових записів і ручне видалення старих. Окрім того в MainActivity, як і в інших компонентах, реалізовано засоби навігації користувача програмою.

Опис полів цього класу наведено у таблиці 3.5.

Таблиця 3.5 – Опис полів MainActivity

| Назва поля  | Опис                                                                                |
|-------------|-------------------------------------------------------------------------------------|
| 1           | 2                                                                                   |
| calendar    | Посилання на елемент інтерфейсу, що відображає робочі записи у календарному форматі |
| monthYearTV | Посилання на текстове поле інтерфейсу, що відображає                                |

Кінець таблиці 3.5

| 1                | 2                                                                                    |
|------------------|--------------------------------------------------------------------------------------|
|                  | поточний обраний місяць                                                              |
| selectedDate     | Містить поточну обрану дату                                                          |
| shiftSelector    | Посилання на елемент, що відображає список створених робочих змін                    |
| db               | Об'єкт класу для роботи з базою даних                                                |
| selectedShift    | Містить поточну обрану зі списку зміну                                               |
| editorLauncher   | Елемент навігації, що реалізує очікування результату роботи створеної нової Activity |
| autofillLauncher |                                                                                      |

MainActivity містить такі методи, що забезпечують виконання поставлених функцій.

onCreate – метод базового класу Activity, що викликається при створенні активності. Проводить перевірку активності функції автоматичного заповнення календаря робочими записами, і залежно від результату перевірки і виконання умов спрацювання функції автозаповнення, проводить або не проводить його. Викликає функції оновлення вигляду календаря і списку змін при запуску програми.

setMonthView – метод оновлення вигляду календаря. Отримує дані про робочі записи поточного обраного місяця і заповнює календар на їх основі.

makeMonthArray – отримує з бази даних інформацію про робочі записи поточного обраного місяця для функції setMonthView.

setSelectorView – метод оновлення списку змін. Отримує з бази даних усі створені зміни і заповнює список на їх основі.

previousMonthAction, nextMonthAction – викликаються при натисканні кнопок зміни поточного місяця. Методи змінюють значення поточної дати і викликають функцію оновлення календаря.

onCalendarCellClick – викликається при натисканні на елемент

календаря. Цей метод реалізує можливість додавання і видалення робочих записів. Якщо, було обрано зміну зі списку, метод створює робочий запис для обраного дня. У разі, коли обраний день вже мав відповідний робочий запис, він попередньо видаляється. Якщо зміну зі списку не було обрано і обраний день містить відповідний робочий запис, він видаляється. Після внесення змін викликається метод оновлення вигляду календаря.

`openShiftEditor`, `onSettingsClick`, `onIncomeReportClick` – методи навігації, що створюють і запускають інші основні Activity.

### 3.4.2 Редактор змін

`shiftEditorActivity` – це компонент, що створюється з головної сторінки застосунку. Ця активність реалізує відображення списку створених змін, їх видалення і відображення їх характеристик. Компонент також реалізує навігацію програмою.

Опис полів цього класу наведено у таблиці 3.6.

Таблиця 3.6 – Опис полів `shiftEditorActivity`

| Назва поля               | Опис                                                                                 |
|--------------------------|--------------------------------------------------------------------------------------|
| <code>shiftEditor</code> | Посилання на елемент інтерфейсу, що відображає список створених змін                 |
| <code>db</code>          | Об'єкт класу для роботи з базою даних                                                |
| <code>launcher</code>    | Елемент навігації, що реалізує очікування результату роботи створеної нової Activity |

`shiftEditorActivity` містить такі методи, що забезпечують виконання поставлених функцій.

`onCreate` – метод базового класу Activity, що викликається при створенні активності. Викликає метод оновлення списку змін при створенні цієї активності.

`updateShiftEditor` – метод оновлення вигляду списку змін. метод оновлення списку змін. Отримує з бази даних усі створені зміни і заповнює список на їх основі.

`onBackButtonClick`, `openShiftCreator` – методи навігації, що створюють і запускають інші активності, або припиняють роботу поточної і повертаються на попередню.

### 3.4.3 Створювач змін

`shiftCreatorActivity` – це компонент, що створюється зі сторінки редактору змін. Ця активність реалізує функцію додавання робочих змін. Компонент також реалізує навігацію програмою.

Опис полів цього класу наведено у таблиці 3.7.

Таблиця 3.7 – Опис полів `shiftEditorActivity`

| Назва поля                   | Опис                                                                 |
|------------------------------|----------------------------------------------------------------------|
| <code>startTimePicker</code> | Посилання на об'єкт для вводу даних про робочу зміну, що створюється |
| <code>endTimePicker</code>   |                                                                      |
| <code>colorPicker</code>     |                                                                      |
| <code>wagePicker</code>      |                                                                      |
| <code>namePicker</code>      |                                                                      |
| <code>db</code>              | Об'єкт класу для роботи з базою даних                                |

`shiftCreatorActivity` містить такі методи, що забезпечують виконання поставлених функцій.

`onCreate` – метод базового класу `Activity`, що викликається при створенні активності. За посиланнями ініціалізує об'єкти інтерфейсу, потрібні

для роботи методів.

`isColorValid`, `isWageValid`, `isTimeValid`, `isNameValid` – перевіряють валідність введених користувачем даних. У разі невідповідності введених даних вимогам, виводять повідомлення, що описує ці вимоги.

`setTimeFor` – викликає діалогове вікно вводу часових даних для визначення часу початку та завершення робочої зміни, що додається.

`onStartTimePickerClick`, `onEndTimePickerClick` – викликаються при натисканні на поля вводу часових даних, викликають функцію вводу даних для поля, що їм відповідає.

`onColorPickerClick` – викликається, при натисканні на поле вводу кольору, викликає діалогове вікно вибору кольору.

`onCreateButtonClick` – викликається при натисканні на кнопку створення робочої зміни. Викликає методи перевірки валідності даних, у разі відповідності даних вимогам, додає до бази даних новий запис про робочу зміну і інформує користувача про успішне додавання.

`onBackButtonClick` – метод навігації, що завершує роботу поточної активності і повертає користувача до попередньої.

#### **3.4.4 Звіт з доходу**

`incomeReportActivity` – це компонент, що створюється з головної сторінки застосунку. Ця активність реалізує функцію отримання звітності з доходу за вказаний календарний період. Компонент також реалізує навігацію програмою.

Опис полів цього класу наведено у таблиці 3.8.

Таблиця 3.8 – Опис полів incomeReportActivity

| Назва поля     | Опис                                                                            |
|----------------|---------------------------------------------------------------------------------|
| startDayPicker | Посилання на об'єкт для вводу календарного періоду для отримання бажаного звіту |
| endDayPicker   |                                                                                 |
| hoursTV        | Посилання на текстове поле для відображення отриманої інформації                |
| incomeTV       |                                                                                 |
| db             | Об'єкт класу для роботи з базою даних                                           |

shiftCreatorActivity містить такі методи, що забезпечують виконання поставлених функцій.

onCreate – метод базового класу Activity, що викликається при створенні активності. За посиланнями ініціалізує об'єкти інтерфейсу, потрібні для роботи методів.

setDateFor – викликає діалогове вікно вводу дати для визначення першого та останнього дня календарного періоду, для якого потрібно створити звітність.

onStartDayPickerClick, onEndDayPickerClick – викликаються при натисканні на поля вводу дати, викликають функцію вводу даних для поля, що їм відповідає.

isDatesValid – перевіряють валідність введених користувачем дат. У разі невідповідності введених даних вимогам, виводить повідомлення, що описує ці вимоги.

onGetDataButtonClick – викликається при натисканні на кнопку отримання звітності. Викликає методи перевірки валідності даних, у разі

відповідності даних вимогам, виводить на текстові поля отримані дані про робочий час і дохід за вказаний період.

`onBackButtonClick` – метод навігації, що завершує роботу поточної активності і повертає користувача до попередньої.

### 3.4.5 Налаштування автоматичного заповнення

`autofillActivity` – це компонент, що створюється з головної сторінки застосунку. Ця активність реалізує функцію автоматичного створення робочих записів. Компонент також реалізує навігацію програмою.

Опис полів цього класу наведено у таблиці 3.9.

Таблиця 3.9 – Опис полів `autofillActivity`

| Назва поля                 | Опис                                                                                                        |
|----------------------------|-------------------------------------------------------------------------------------------------------------|
| <code>shiftSpinner</code>  | Посилання на випадаючий список створених робочих змін                                                       |
| <code>shift</code>         | Містить обрану з випадаючого списку робочу зміну                                                            |
| <code>prefs</code>         | Об'єкт класу для запису і читання налаштувань застосунку                                                    |
| <code>autofillCheck</code> | Посилання на флаг, що відповідає за активацію і деактивацію функції автоматичного створення робочих записів |
| <code>db</code>            | Об'єкт класу для роботи з базою даних                                                                       |

`shiftCreatorActivity` містить такі методи, що забезпечують виконання поставлених функцій.

`onCreate` – метод базового класу `Activity`, що викликається при створенні активності. Отримує з бази даних список усіх робочих змін.

`setAdapter` – заповнює випадаючий список робочими змінами зі списку, отриманого в `onCreate`.

`onSaveSettingsClick` – викликається при натисканні на кнопку

збереження налаштувань функції автоматичного створення робочих записів. У разі відповідності введених даних вимогам, зберігає встановлені налаштування.

`onBackButtonClick` – метод навігації, що завершує роботу поточної активності і повертає користувача до попередньої.

### **3.5 Опис додаткових компонентів**

#### **3.5.1 Опис адаптерів списків об'єктів**

Розроблені класи `calendarAdapter`, `shiftSelectorAdapter` і `shiftEditorAdapter` наслідуються від вбудованого класу `RecyclerView.Adapter`. Їх функція полягає у заповненні списків об'єктів на сторінках застосунку об'єктами за створеним макетом інтерфейсу. Кожен адаптер містить клас-нащадок класу `RecyclerView.ViewHolder`, який містить посилання на об'єкти макету інтерфейсу, що представляє елемент у списку [23].

Для адаптерів створено інтерфейси функцій, що обробляють викликають подію при натисканні на елемент списку. Опис події відбувається у `Activity`, котра містить список.

Адаптери мають такі поля:

- список даних, які будуть відображатися на об'єктах макету інтерфейсу;

- об'єкт розробленого інтерфейсу обробки натискання на елемент.

Адаптери містять такі методи, успадковані від `RecyclerView.Adapter`:

- `onCreateViewHolder` - встановлює параметри макету інтерфейсу елементу списку;

- `onBindViewHolder` - передає об'єктам макету інтерфейсу дані зі списку;

- `getItemCount` - повертає розмір списку даних.

### 3.5.2 Опис класу взаємодії з БД

Для забезпечення взаємодії компонентів застосунку зі створеною базою даних, було розроблено клас DatabaseHelper. Він успадковується від вбудованого в Java класу SQLiteOpenHelper для керування базою даних [24].

Опис полів класу наведено у таблиці 3.10.

Таблиця 3.10 – Опис полів DatabaseHelper

| Назва поля | Опис              |
|------------|-------------------|
| DB_NAME    | Назва бази даних  |
| DB_VERSION | Версія бази даних |

Для класу розроблено такі методи, що реалізують взаємодію з базою даних:

- onCreate – наслідуваний з класу-нащадка метод, що викликається при створенні бази даних, метод створює таблиці БД;
- onUpgrade – наслідуваний з класу-нащадка метод, що викликається при оновленні бази даних;
- onConfigure – наслідуваний з класу-нащадка метод, що викликається коли відбувається налаштування зв'язків між таблицями БД, метод активує підтримку сторонніх ключів;
- insertShiftType – додає новий запис до таблиці робочих змін;
- insertShiftRecord – додає новий запис до таблиці робочих записів;
- getAllShiftTypes – повертає список усіх записів таблиці робочих змін;
- deleteShiftRecordByDate – видаляє запис з таблиці робочих записів за вказаною датою;
- deleteShiftTypeByName – видаляє запис з таблиці робочих змін за назвою зміни;

- `getShiftRecord` – повертає запис з таблиці робочих записів за датою;
- `getShiftTypeById` – повертає запис з таблиці робочих змін за `id` зміни;
- `getShiftTypeByName` – повертає запис з таблиці робочих змін за назвою зміни.

# 4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

## 4.1 Методика роботи користувача із застосунком

Схему роботи користувача з розробленим застосунком у вигляді діаграми прецедентів зображено на рисунку 4.1.

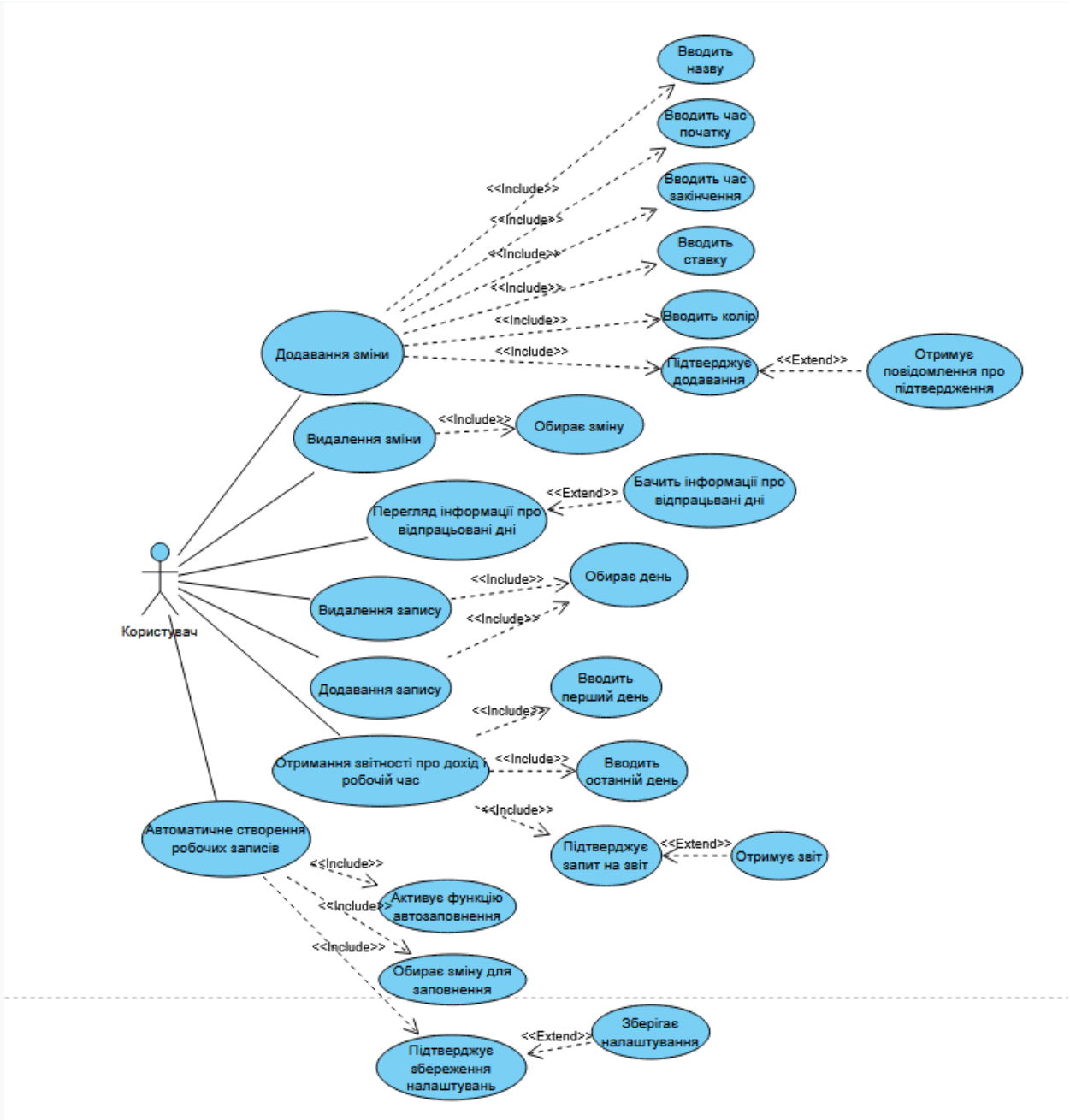


Рисунок 4.1 – Діаграма прецедентів користувача

Застосунок не є вибагливим до ресурсів пристрою, на якому запускається. Мінімальною версією ОС для роботи застосунку є Android 8 “Oreo” [25].

При запуску застосунку користувач потрапляє на головну сторінку, що містить календар з робочими записами і список робочих змін. При першому запуску календар і список змін є порожніми. Зовнішній вигляд головної сторінки зображено на рисунку 4.2.

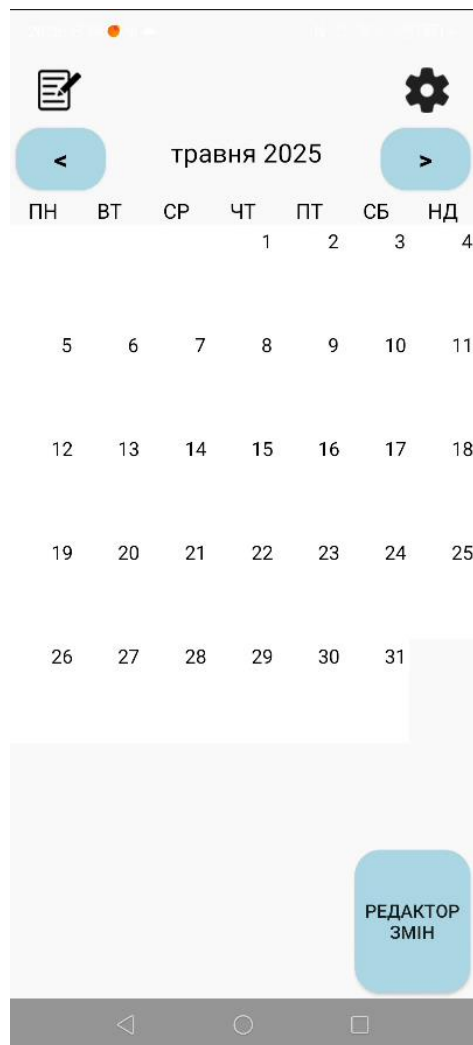


Рисунок 4.2 – Головна сторінка

Для подальшої роботи користувачу потрібно створити нову робочу зміну. Для цього потрібно перейти до редактора змін (рис. 4.3), звідки перейти до створювача змін (рис. 4.4).

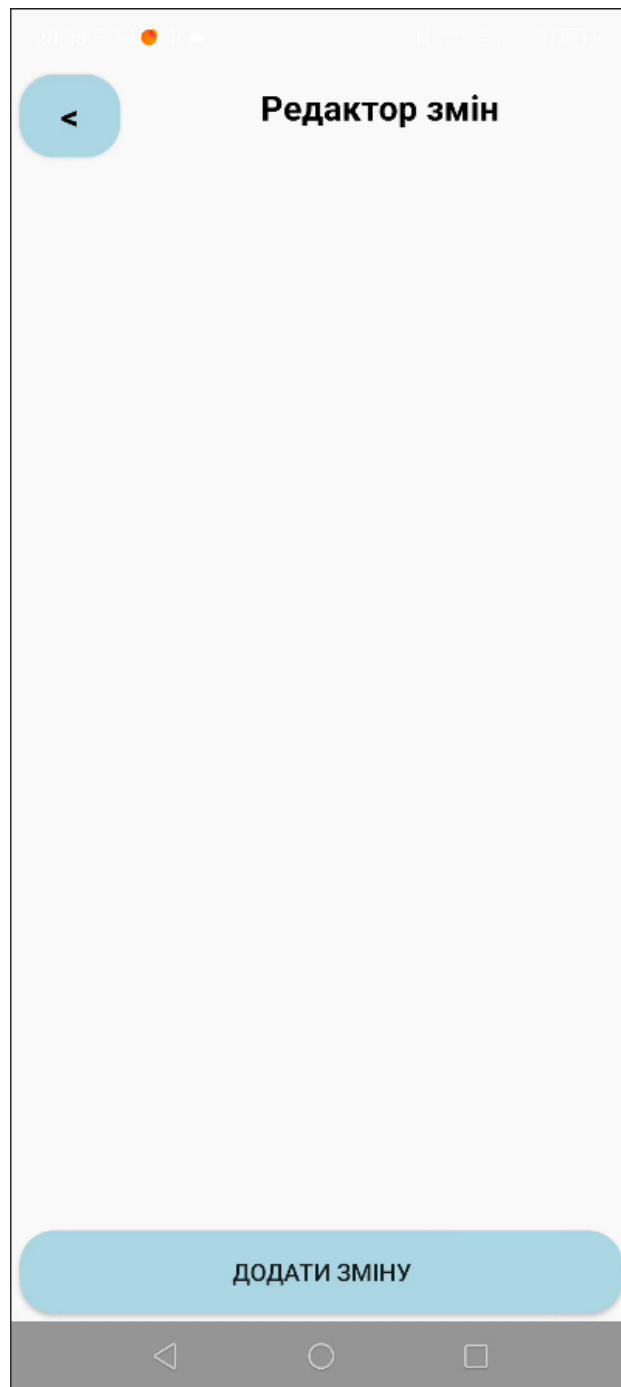


Рисунок 4.3 – Редактор змін

Створення зміни

Назва назва

Час початку час початку

Час закінчення час закінчення

Ставка ставка

Колір колір

ЗБЕРЕГТИ

Рисунок 4.4 – Створювач змін

Для створення зміни користувачу потрібно заповнити форму валідними даними про зміну і натиснути кнопку збереження. Введення часу початку і закінчення проводиться за допомогою діалогового вікна вибору часу (рис. 4.5). Введення кольору проводиться за допомогою діалогового вікна вибору кольору (рис. 4.6). При заповненні форми невалідними даними користувач отримає повідомлення, що описує помилку у заповненні. Приклад повідомлення про помилку зображено на рисунку 4.7.

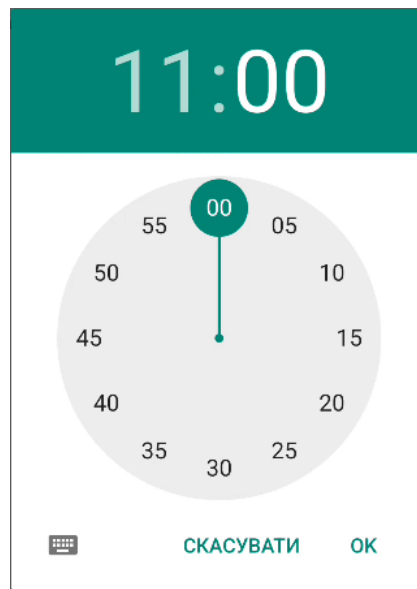


Рисунок 4.5 – Діалогове вікно вибору часу

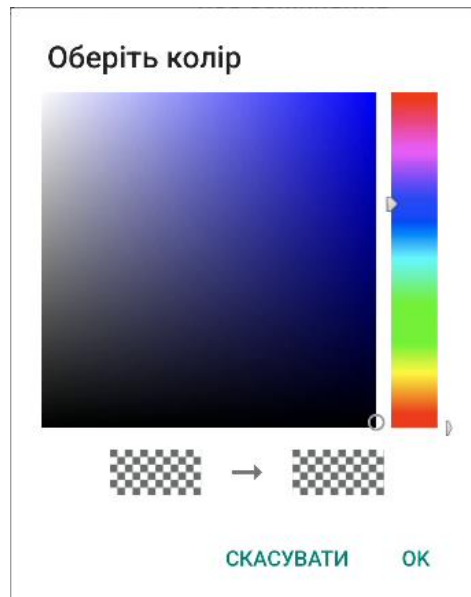
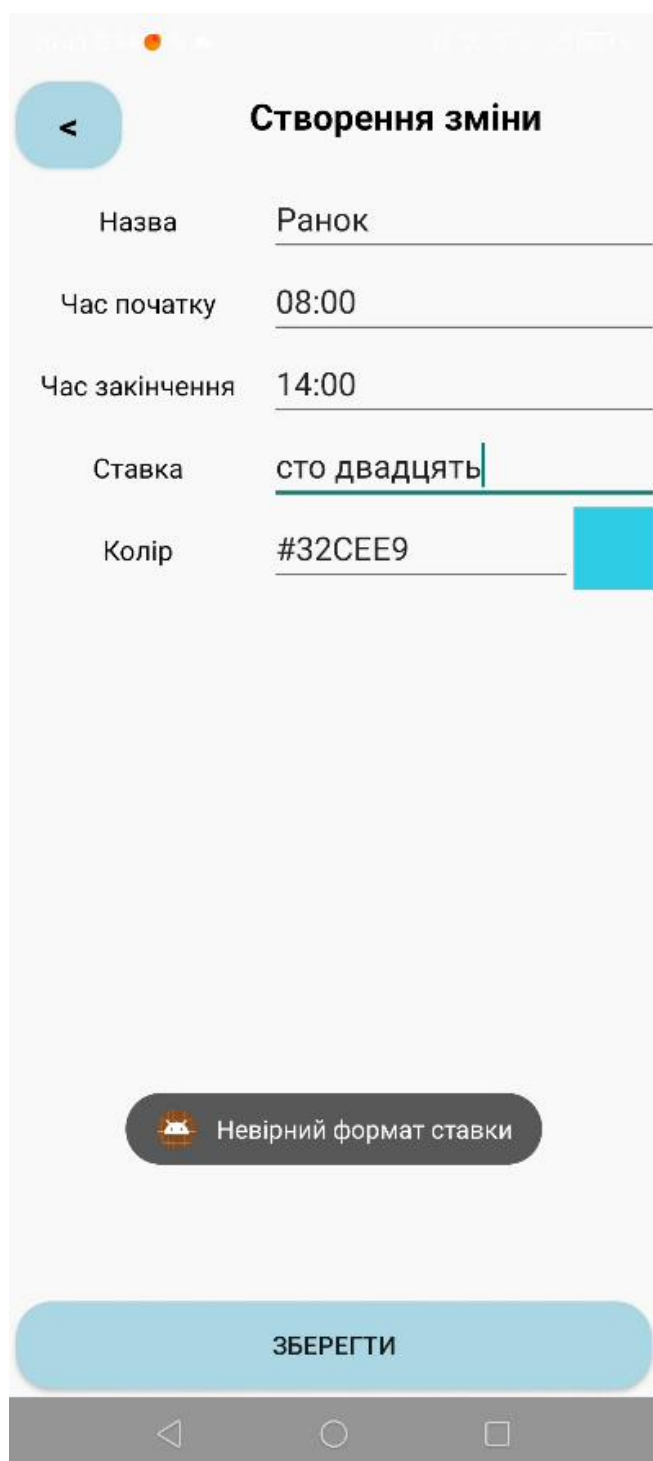


Рисунок 4.6 – Діалогове вікно вибору кольору



Створення зміни

|                |              |
|----------------|--------------|
| Назва          | Ранок        |
| Час початку    | 08:00        |
| Час закінчення | 14:00        |
| Ставка         | сто двадцять |
| Колір          | #32CEE9      |

Невірний формат ставки

ЗБЕРЕГТИ

Рисунок 4.7 – Повідомлення про невірний формат ставки

Після заповнення форми валідними даними і натискання кнопки збереження користувач отримає повідомлення про успішне збереження робочої зміни. Результат успішного додавання зображено на рисунку 4.8.

|                |         |
|----------------|---------|
| Назва          | Ранок   |
| Час початку    | 08:00   |
| Час закінчення | 14:00   |
| Ставка         | 120     |
| Колір          | #32CEE9 |

Зміну додано

ЗБЕРЕГТИ

Рисунок 4.8 – Успішне додавання зміни

Після додавання зміни і повернення на сторінку редактора шляхом натискання кнопки “Назад” або кнопки навігації на сторінці, додана зміна і інформація про неї відобразиться у списку змін редактора (рис. 4.9). Також зміна відобразиться у списку змін головної сторінки після повернення до неї засобами навігації (рис. 4.10).

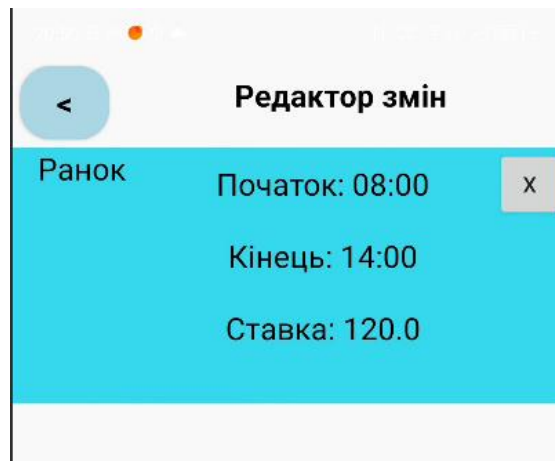


Рисунок 4.9 – Додана зміна у редакторі

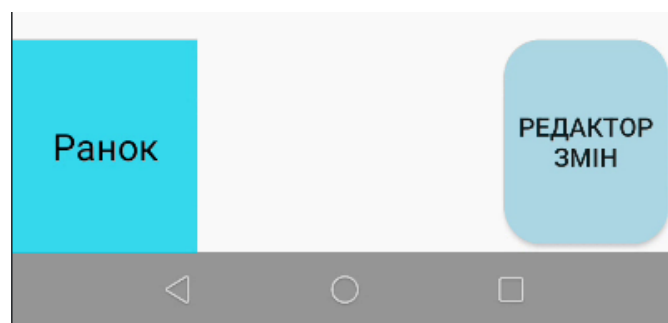


Рисунок 4.10 – Додана зміну у списку на головній сторінці

На сторінці редактора змін користувач має можливість видалити зміну шляхом натискання на кнопку видалення на елементі відображення зміни. Після видалення зміна відразу зникне зі списку редактора і зникне зі списку головної сторінки після повернення до неї.

Якщо було додано хоча б одну зміну, користувач має можливість створювати робочі записи. Для цього на головній сторінці потрібно обрати зміну зі списку, натиснувши на неї, і натиснути на комірку календаря того дня, про який потрібно зробити запис. Після цього вигляд комірки обраного дня оновиться і матиме колір обраної робочої зміни. Після натиснення на комірку, котрій вже було присвоєно запис, її запис зміниться на відповідний обраній робочій зміні. Результат створення робочого запису наведено на рисунку 4.11.



Рисунок 4.11 – Доданий робочій запис у календарі

Для видалення робочого запису потрібно впевнитись, що жодної зміни не обрано. Зміна не обрано за замовчуванням після запуску застосунку, також вибір зміни можна скинути повторним натисканням на обрану зміну. Після цього при натисканні на комірку з робочім записом, запис видалиться, і колір комірки зміниться на білий.

Користувач має можливість активувати автоматичне створення робочих записів. Для цього потрібно перейти до сторінки налаштувань автозаповнення. Зовнішній вигляд сторінки зображено на рисунку 4.12.

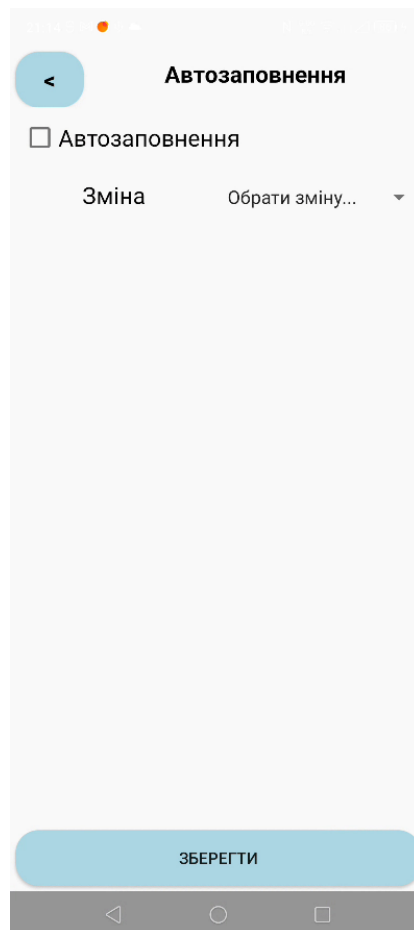


Рисунок 4.12 – Сторінка налаштування автозаповнення

Для активації функції потрібно поставити флажок автозаповнення обрати робочу зміну для заповнення з випадаючого списку (рис. 4.13) і натиснути кнопку збереження налаштувань. При спробі активувати функцію без вибору робочої зміни користувач отримає повідомлення про помилку (рис 4.14).

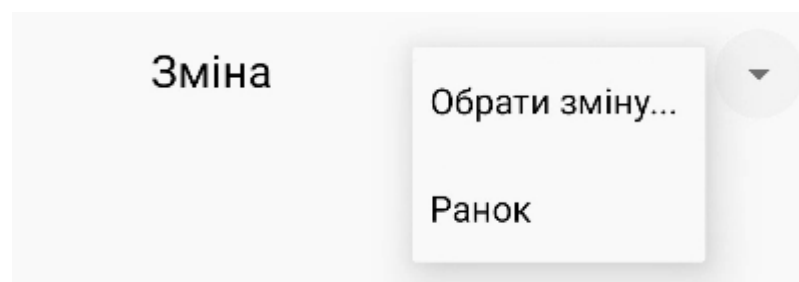


Рисунок 4.13 – Випадаючий список змін

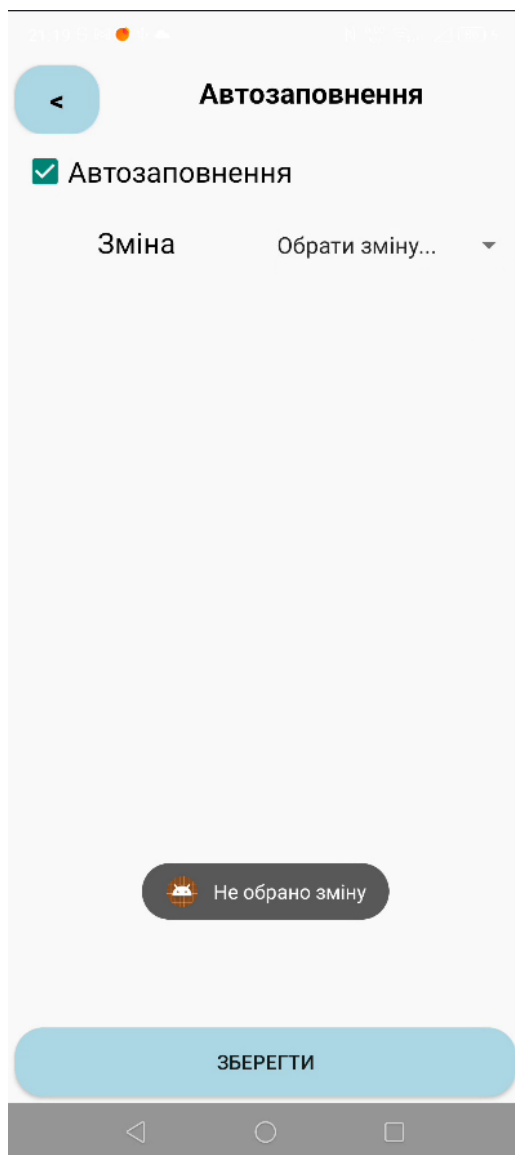


Рисунок 4.14 – Повідомлення про необрану зміну

Після активації функції, при запуску застосунку у понеділок, якщо запису про цей день немає, відповідно до стандартного п'ятиденного робочого графіку, про цей і чотири наступних дні буде створено записи з обраною у налаштуваннях робочою зміною.

Користувач має можливість отримати звіт з доходу і робочого часу за певний календарний період. Для цього потрібно перейти на сторінку звіту з доходу. Зовнішній вигляд сторінки зображено на рисунку 4.15.

Звіт з доходу

Від: startDay

До: startDay

Робочий час      Дохід

ОТРИМАТИ ДАНІ

Рисунок 4.15 – Сторінка звіту з доходу

Для отримання звіту потрібно ввести перший і останній день бажаного календарного періоду. Вибір днів проводиться за допомогою діалогового вікна вибору дати (рис. 4.16). При введенні періоду у нуль днів або введені першого дня, пізнішого за останній, користувач отримає повідомлення про помилку. Приклад повідомлення наведено на рисунку 4.16.

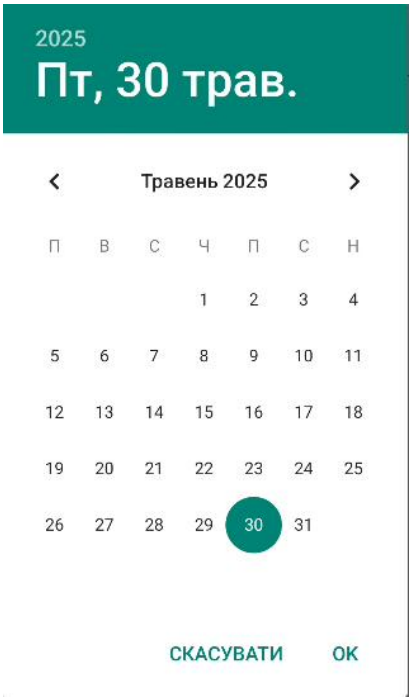


Рисунок 4.15 – Діалогове вікно вибору дати

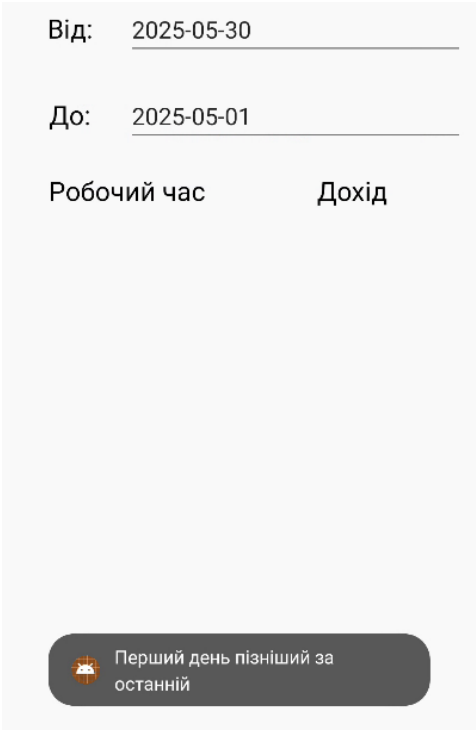


Рисунок 4.16 – Повідомлення про помилку

При введенні валідних даних, користувач отримає звіт з сумарного робочого часу і доходу за вказаний період.

Приклад звіту зображено на рисунку 4.17.

|             |                   |
|-------------|-------------------|
| Від:        | <u>2025-05-01</u> |
| До:         | <u>2025-05-31</u> |
| Робочий час | Дохід             |
| 6,00 год.   | 720,00 грн.       |

Рисунок 4.17 – Звіт з доходу і робочого часу

## 4.2 Функціональне тестування

Для того, щоб гарантувати здатність застосунку виконувати поставлені задачі, необхідно впевнитися у правильності роботи розроблених функцій шляхом проведення тестування. Функціональне тестування застосунку проводилося за допомогою інструменту відлагоджувача.

В Android Studio відлагоджувач надає низку можливостей керування розробленою програмою [26]:

- дозволяє обирати пристрій для проведення відлагодження;
- підтримує точки зупинки програми;
- дозволяє відстежувати стан змінних і роботу методів прямо у процесі роботи програми.

За допомогою інструментів відладки було проведено такі перевірки:

- перевірка правильності логіки умовних виразів;
- перевірка правильності логіки математичних обчислень;
- перевірка правильності обробки критичних ситуацій.

Помилки роботи розробленого застосунку, виявлені під час тестування, було усунено і проведено повторні перевірки. Таким чином вдалося впевнитися, що застосунок коректно виконує поставлені задачі.

### 4.3 Тестування інтерфейсу

Android Studio надає широкий ряд можливостей для тестування розробленого застосунку. Це досягається завдяки вбудованому інструменту емуляції Android-пристроїв. Таким чином можна перевірити роботи застосунку на великій кількості пристроїв з різноманітними характеристиками.

Тестування інтерфейсу проводилося на шляхом перевірки роботи застосунку на різних пристроях, а саме:

а) фізичні пристрої:

- 1) смартфон ZTE Blade V40 під керуванням Android 12;
- 2) смартфон Xiaomi Redmi 9A під керуванням Android 10;

б) емульований пристрій Pixel 7 Pro.

Приклад роботи розробленого застосунку на фізичних пристроях наведено на рисунках 4.18 і 4.19.

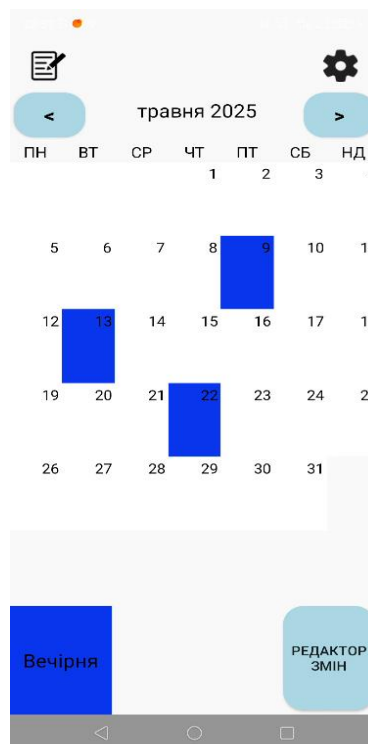


Рисунок 4.18 – Робота застосунку на ZTE Blade V40

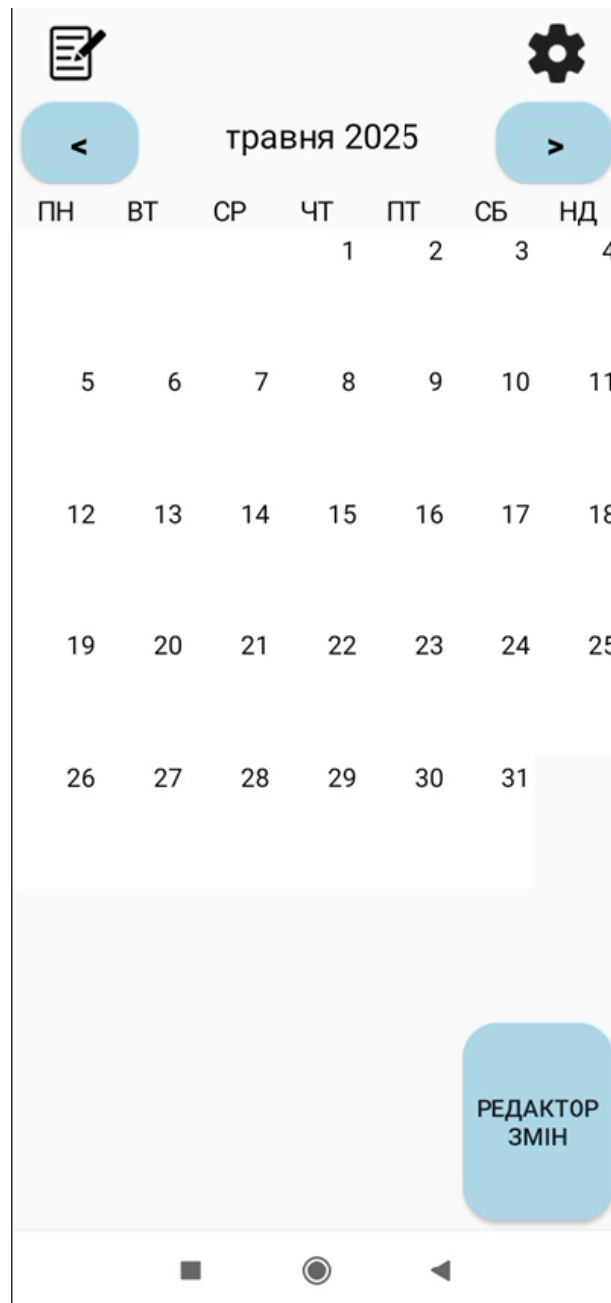


Рисунок 4.19 – Робота застосунку на Xiaomi Redmi 9A

Для більш результативного тестування в якості емульованого пристрою було модель, що має відмінну від використаних фізичних пристроїв роздільну здатність екрану. Результати роботи програми на емульованому пристрої наведено на рисунку 4.20.

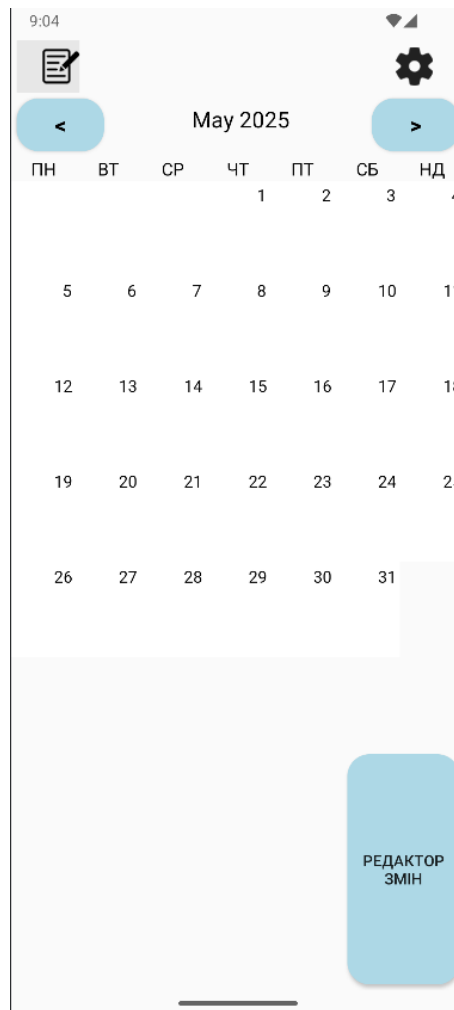


Рисунок 4.20 – Робота застосунку на Pixel 7 Pro

Як видно з результатів тестування, застосунок зберігає свою працездатність попри зміну роздільної здатності. Отже, можна зробити висновок, що розроблений інтерфейс забезпечує можливість користувача проводити роботу із застосунком.

## ВИСНОВКИ

У процесі виконання роботи було розроблено мобільний застосунок “Календар робочих змін”, що дозволяє користувачеві зберігати і аналізувати дані про робочій процес, і керувати ними зручними та інтуїтивними методами.

Було проведено аналіз предметної області, в результаті якого було сформульовано основні поняття і проблеми області, вимоги до програмного забезпечення, було описано і проаналізовано аналогічні програмні продукти.

Аналіз і порівняння доступних програмних засобів дозволило прийняти рішення розробити Android-застосунок мовою Java у IDE Android Studio з використанням системи керування базами даних SQLite. Вибір цих засобів позитивно вплинув на процес розробки.

Було детально описано розроблені програмні компоненти, вичерпно змодельовано методику роботи користувача з програмою і проведено тестування працездатності. Тестування продемонструвало, що розроблений застосунок справно виконує очікувані функції.

Розроблений застосунок надає користувачеві можливості створювати і видаляти робочі записи і зміни, представляти робочі записи у зручному структурованому форматі, отримувати звіти про робочий час і дохід, налаштувати автоматичне створення робочих записів, має зручний і лаконічний інтерфейс. Застосунок відповідає сформульованим вимогам, що робить його конкурентоспроможним варіантом вирішення поставленої проблеми.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Рівень безробіття в Україні [Електронний ресурс].— Режим доступу: <https://index.minfin.com.ua/ua/labour/unemploy/>
2. Тарифна ставка [Електронний ресурс].— Режим доступу: [https://uk.wikipedia.org/wiki/%D0%A2%D0%B0%D1%80%D0%B8%D1%84%D0%BD%D0%B0\\_%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0](https://uk.wikipedia.org/wiki/%D0%A2%D0%B0%D1%80%D0%B8%D1%84%D0%BD%D0%B0_%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0)
3. Google Календар [Електронний ресурс].— Режим доступу: <https://play.google.com/store/apps/details?id=com.google.android.calendar>
4. ShifterCalendar [Electronic resource]. – Access mode: <https://play.google.com/store/apps/details?id=com.lrhsoft.shiftercalendar>
5. Mobile Operating System Market Share Worldwide [Electronic resource]. – Access mode: <https://gs.statcounter.com/os-market-share/mobile/worldwide>
6. Android [Електронний ресурс].— Режим доступу: <https://uk.wikipedia.org/wiki/Android>
7. Android vs iOS Market Share: Most Popular Mobile OS in 2024 [Electronic resource]. – Access mode: <https://www.mobiloud.com/blog/android-vs-ios-market-share>
8. Java [Електронний ресурс].— Режим доступу: <https://uk.wikipedia.org/wiki/Java>
9. Віртуальна машина Java [Електронний ресурс].— Режим доступу: [https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0\\_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0\\_Java](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_Java)
10. Kotlin [Електронний ресурс].— Режим доступу: <https://uk.wikipedia.org/wiki/Kotlin>
11. Java: Pros and Cons in Mobile App Development [Electronic resource]. – Access mode: <https://amela.tech/java-pros-and-cons-in-mobile-app-development/>
12. Dart overview [Electronic resource]. – Access mode:

<https://dart.dev/overview>

13. Flutter [Electronic resource]. – Access mode: <https://flutter.dev/>
14. C# [Электронный ресурс].— Режим доступа: [https://uk.wikipedia.org/wiki/C\\_Sharp](https://uk.wikipedia.org/wiki/C_Sharp)
15. .NET Multi-platform App UI [Electronic resource]. – Access mode: <https://dotnet.microsoft.com/apps/maui>
16. Microsoft Visual Studio [Электронный ресурс].— Режим доступа: [https://uk.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio)
17. Microsoft Visual Studio [Electronic resource]. – Access mode: <https://visualstudio.microsoft.com/>
18. Android Studio [Electronic resource]. – Access mode: <https://developer.android.com/studio>
19. SQLite [Electronic resource]. – Access mode: <https://uk.wikipedia.org/wiki/SQLite>
20. Most Widely Deployed and Used Database Engine [Electronic resource]. – Access mode: <https://www.sqlite.org/mostdeployed.html>
21. Layouts in views [Electronic resource]. – Access mode: <https://developer.android.com/develop/ui/views/layout/declaring-layout>
22. Activity [Electronic resource]. – Access mode: <https://developer.android.com/reference/android/app/Activity>
23. RecyclerView.Adapter [Electronic resource]. – Access mode: <https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.Adapter>
24. SQLiteOpenHelper [Electronic resource]. – Access mode: <https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper>
25. Android Oreo [Electronic resource]. – Access mode: [https://uk.wikipedia.org/wiki/Android\\_Oreo](https://uk.wikipedia.org/wiki/Android_Oreo)
26. Debug your app [Electronic resource]. – Access mode: <https://developer.android.com/studio/debug>

**ДОДАТОК А**  
**Технічне завдання**

## **Вступ**

Розробка мобільного застосунку “Календар робочих змін” зумовлена актуальною проблемою важкості обліку робочого процесу, що є важливим процесом для значної частини населення.

Метою розробки є створення програмного забезпечення, що надасть інструменти для запису і аналізу характеристик робочого процесу, будучи при цьому простим для розуміння і керування.

### **A.1 Підстави для розробки**

Мобільний застосунок “Календар робочих змін” розробляється на підставі завдання на дипломну кваліфікаційну роботу бакалавра, затвердженого наказом № 209 Національного Університету “Запорізька Політехніка” від 28 квітня 2025 року за темою «Розробка мобільного застосунку календаря робочих змін».

### **A.2 Призначення розробки**

Розроблений мобільний застосунок “Календар робочих змін” повинен надавати користувачеві методи обліку даних про робочій процес.

### **A.3 Основні вимоги до програми**

#### **A.3.1 Вимоги до функціональних характеристик**

Для успішного вирішення поставленої проблеми мобільний застосунок “Календар робочих змін” повинен відповідати наступним функціональним вимогам:

- можливість створювати і видаляти робочі зміни, переглядати їх характеристики;

- можливість додавати і видаляти робочі записи, шляхом присвоєння робочих змін;
- можливість переглядати робочі записи у структурованому форматі;
- можливість створювати звіти на основі доданих робочих записів;
- можливість налаштовувати автоматичне створення записів для полегшення роботи з ПЗ, що розробляється.

### **А.3.2 Умови експлуатації**

Для успішної експлуатації мобільного застосунку “Календар робочих змін” необхідний мобільний пристрій під керуванням операційних системи Android не менше версії 8 “Oreo”.

### **А.3.3 Вимоги до надійності**

Мобільний застосунок “Календар робочих змін” повинен справно обробляти критичні ситуації для уникнення порушень роботи користувача.

### **А.3.4 Вимоги до інтерфейсу**

Мобільний застосунок “Календар робочих змін” повинен мати легкий, лаконічний інтерфейс і надавати користувачеві способи легкої взаємодії з застосунком.

### **А.3.5 Вимоги до складу та параметрів технічних засобів**

Для коректної роботи застосунку система, на якій проводиться експлуатація, повинна мати не менше ніж 1Гб оперативної пам’яті.

#### **А.4 Вхідні дані**

Для мобільного застосунку “Календар робочих змін” вхідними даними є властивості робочих змін, вихідними даними є звіт про робочий час і дохід.

**ДОДАТОК Б**  
**Текст програми**

```

package com.example.myapplication;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import java.time.LocalDate;
import java.time.YearMonth;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity implements
CalendarAdapter.OnCalendarCellListener {

    RecyclerView calendar;

    TextView monthYearTV;

    LocalDate selectedDate;

    RecyclerView shiftSelector;
    DatabaseHelper db;
    String selectedShift="";

    ActivityResultLauncher<Intent> editorLauncher;
    ActivityResultLauncher<Intent> autofillLauncher;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

```

```

        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        calendar = findViewById(R.id.calendarRecyclerView);
        monthYearTV = findViewById(R.id.monthYearTV);
        selectedDate = LocalDate.now();
        int dayOfWeek = selectedDate.getDayOfWeek().getValue();
        if(dayOfWeek==2)
        {
            SharedPreferences prefs = getSharedPreferences("my_settings",
            MODE_PRIVATE);
            if(prefs.getBoolean("autofill", false))
            {
                db = new DatabaseHelper(this);
                DateTimeFormatter format = DateTimeFormatter.ofPattern("yyyy-MM-dd");
                ShiftRecord record = db.getShiftRecord(selectedDate.format(format));
                if(record==null)
                {
                    LocalDate current = selectedDate;
                    for(int i=0;i<5;i++)
                    {
                        db.insertShiftRecord(current.format(format),
                        db.getShiftTypeByName(prefs.getString("shift","")).getId());
                        current = current.plusDays(1);
                    }
                }
            }
            setMonthView();
            editorLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),

```

```

        result -> {
            if (result.getResultCode() == Activity.RESULT_OK) {
                setSelectorView();
                setMonthView();
                setResult(RESULT_CANCELED);
            }
        }
    );
    autofillLauncher = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        result -> {
            if (result.getResultCode() == Activity.RESULT_OK) {
                setMonthView();
                setResult(RESULT_CANCELED);
            }
        }
    );
    setSelectorView();
}

private void setMonthView()
{
    monthYearTV.setText(monthYearFromDate(selectedDate));
    ArrayList<String> daysInMonth = new ArrayList<>();
    ArrayList<String> colors = new ArrayList<>();
    makeMonthArray(selectedDate, daysInMonth, colors);
    CalendarAdapter calendarAdapter = new
CalendarAdapter(daysInMonth, colors, this);
    RecyclerView.LayoutManager layoutManager = new
GridLayoutManager(getApplicationContext(), 7);
    calendar.setLayoutManager(layoutManager);
    calendar.setAdapter(calendarAdapter);
}

private String monthYearFromDate(LocalDate date)
{
    DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("MMMM yyyy");
    return date.format(formatter);
}
};

```

```

private void makeMonthArray(LocalDate Date, ArrayList<String>
daysInMonthArray, ArrayList<String> shiftColors)
{
    YearMonth yearMonth = YearMonth.from(Date);
    int daysInMonth = yearMonth.lengthOfMonth();
    LocalDate firstOfMonth = selectedDate.withDayOfMonth(1);
    int dayOfWeek = firstOfMonth.getDayOfWeek().getValue();
    for(int i =0; i<=daysInMonth+dayOfWeek-2;i++)
    {

        if(i<dayOfWeek-1)
        {
            daysInMonthArray.add("");
            shiftColors.add("#ffffff");
        }
        else
        {
            String currentDate = String.valueOf(YearMonth.from(Date))+'-';
            if(i-dayOfWeek+2<10) currentDate += '0';
            currentDate+=String.valueOf(i-dayOfWeek+2);
            db = new DatabaseHelper(this);
            ShiftRecord currentRecord = db.getShiftRecord(currentDate);
            int shiftId=0;
            if (currentRecord!=null) shiftId = currentRecord.getShiftId();

            if(shiftId==0)
            {
                shiftColors.add("#ffffff");
            }
            else
            {
                String shiftColor = db.getShiftTypeById(shiftId).getColor();
                shiftColors.add(shiftColor);
            }
            daysInMonthArray.add(String.valueOf(i-dayOfWeek+2));
        }
    }
};

public void previousMonthAction(View view)
{
    selectedDate = selectedDate.minusMonths(1);
    setMonthView();
}

```

```

public void nextMonthAction(View view) {
    selectedDate = selectedDate.plusMonths(1);
    setMonthView();
}

@Override
public void onCalendarCellClick(int position, String dayText)
{
    if(!dayText.isEmpty())
    {
        db = new DatabaseHelper(this);
        String                currentDate                =
String.valueOf(YearMonth.from(selectedDate))+'-';
        if(dayText.length()<2) currentDate+='0';
        currentDate+=dayText;

        int shiftRecordShiftId = 0;

        ShiftRecord currentRecord = db.getShiftRecord(currentDate);
        if(currentRecord!=null)                shiftRecordShiftId                =
currentRecord.getShiftId();

        if (!selectedShift.isEmpty())
        {

            int shiftId = db.getShiftTypeByName(selectedShift).getId();
            if(shiftRecordShiftId==0)
            {
                db.insertShiftRecord(currentDate, shiftId);
            }
            else if(shiftRecordShiftId!=shiftId)
            {
                db.deleteShiftRecordByDate(currentDate);
                db.insertShiftRecord(currentDate,shiftId);
            }

        }
        else
        {
            if(shiftRecordShiftId!=0)
                db.deleteShiftRecordByDate(currentDate);
        }
    }
    setMonthView();
}

```

```

    }

    public void openShiftEditor(View view)
    {

        Intent intent = new Intent(MainActivity.this, shiftEditor.class);
        editorLauncher.launch(intent);
    }

    private void setSelectorView() {
        db = new DatabaseHelper(this);
        shiftSelector = findViewById(R.id.shiftSelector);
        LinearLayoutManager layoutManager = new
LinearLayoutManager(getApplicationContext(),
LinearLayoutManager.HORIZONTAL, false);
        shiftSelector.setLayoutManager(layoutManager);
        ArrayList<String> shiftNames = new ArrayList<>();
        ArrayList<String> shiftColors = new ArrayList<>();
        ArrayList<ShiftType> shifts = db.getAllShiftTypes();

        for(ShiftType s: shifts)
        {
            shiftNames.add(s.getName());
            shiftColors.add(s.getColor());
        }

        shiftSelectorAdapter adapter = new shiftSelectorAdapter(shiftNames,
shiftColors, new shiftSelectorAdapter.OnShiftCellListener() {
            @Override
            public void onShiftCellClick(String shiftName) {
                db = new DatabaseHelper(MainActivity.this);
                ShiftType foundShift = db.getShiftTypeByName(shiftName);
                if(selectedShift.equals(foundShift.getName()))
                {
                    selectedShift="";
                }
                else
                {
                    selectedShift = foundShift.getName();
                }
            }
        });
        shiftSelector.setAdapter(adapter);
    }

```

```

    }

    public void onSettingsClick(View view) {
        Intent intent = new Intent(MainActivity.this, autofillActivity.class);
        autofillLauncher.launch(intent);
    }

    public void onIncomeReportClick(View view) {
        Intent intent = new Intent(MainActivity.this,
incomeReportActivity.class);
        startActivity(intent);
    }
}

package com.example.myapplication;

import android.app.DatePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.google.android.material.textfield.TextInputEditText;

import java.time.LocalDate;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Locale;

public class incomeReportActivity extends AppCompatActivity {

    TextInputEditText startDayPicker;
    TextInputEditText endDayPicker;
    TextView hoursTV;
    TextView incomeTV;
    DatabaseHelper db;
    @Override

```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_income_report);

    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars =
        insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right,
            systemBars.bottom);
        return insets;
    });

    startDayPicker = findViewById(R.id.incomeReportStartDayPicker);
    endDayPicker = findViewById(R.id.incomeReportEndDayPicker);
}

private boolean isDatesValid(TextInputEditText startDayPicker,
    TextInputEditText endDayPicker)
{
    if (startDayPicker.getText().toString().isEmpty())
    {
        Toast.makeText(this, "Перший день порожній",
            Toast.LENGTH_LONG).show();
        return false;
    }
    if (endDayPicker.getText().toString().isEmpty())
    {
        Toast.makeText(this, "Останній день порожній",
            Toast.LENGTH_LONG).show();
        return false;
    }
    DateTimeFormatter format = DateTimeFormatter.ofPattern("yyyy-MM-dd");
    LocalDate startDay = LocalDate.parse(startDayPicker.getText(),
        format);
    endDay = LocalDate.parse(endDayPicker.getText(), format);
    if (startDay.isAfter(endDay)) {
        Toast.makeText(this, "Перший день пізніший за останній",
            Toast.LENGTH_LONG).show();
        return false;
    }
    else if(startDay.equals(endDay))

```

```

        {
            Toast.makeText(this, "Порожній період",
Toast.LENGTH_LONG).show();
            return false;
        }
        return true;
    }

    private void setDateFor(TextInputEditText dayPicker)
    {
        Calendar calendar = Calendar.getInstance();
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePickerDialog = new
DatePickerDialog(incomeReportActivity.this,
        (view, selectedYear, selectedMonth, selectedDay) -> {
            String selectedDate = String.format(Locale.getDefault(), "%d-
%02d-%02d",
                selectedYear, selectedMonth + 1, selectedDay );
            dayPicker.setText(selectedDate);
        }, year, month, day);

        datePickerDialog.show();
    }

    public void onBackButtonClick(View view)
    {
        finish();
    }

    public void onStartDayPickerClick(View view)
    {
        setDateFor(startDayPicker);
    }

    public void onEndDayPickerClick(View view)
    {
        setDateFor(endDayPicker);
    }

    public void onGetDataButtonClick(View view)
    {
        if(isDatesValid(startDayPicker, endDayPicker))

```

```

        {
            DateTimeFormatter format = DateTimeFormatter.ofPattern("yyyy-
MM-dd");

            LocalDate startDay = LocalDate.parse(startDayPicker.getText(),
format),
                endDay = LocalDate.parse(endDayPicker.getText(), format);
            ArrayList<ShiftType> shifts = new ArrayList<>();
            LocalDate currentDay = startDay;
            db = new DatabaseHelper(this);
            while(!currentDay.equals(endDay))
            {
                ShiftRecord currentRecord =
db.getShiftRecord(currentDay.format(format));
                if(currentRecord!=null)
                {
                    shifts.add(db.getShiftTypeById(currentRecord.getShiftId()));
                }

                currentDay = currentDay.plusDays(1);
            }

            double workTime=0;
            double income =0;

            for(ShiftType s: shifts)
            {
                LocalTime startTime = LocalTime.parse(s.getStartTime()),
                    endTime = LocalTime.parse(s.getEndTime());
                double hours = endTime.getHour() - startTime.getHour();
                double minutes = (endTime.getMinute() -
startTime.getMinute())/60.0;
                double shiftWorkTime = hours+minutes;
                workTime = workTime + shiftWorkTime;
                double wage = s.getWage();
                income=income+shiftWorkTime*wage;

            }
            hoursTV = findViewById(R.id.incomeReportHoursTV);
            incomeTV = findViewById(R.id.incomeReportWageTV);
            hoursTV.setText(String.format("%.2f", workTime)+" год.");
            incomeTV.setText(String.format("%.2f", income) +" грн.");
        }
    }
}

```

**ДОДАТОК В**  
**Слайди презентації**



Рисунок В.1 – Титульний слайд

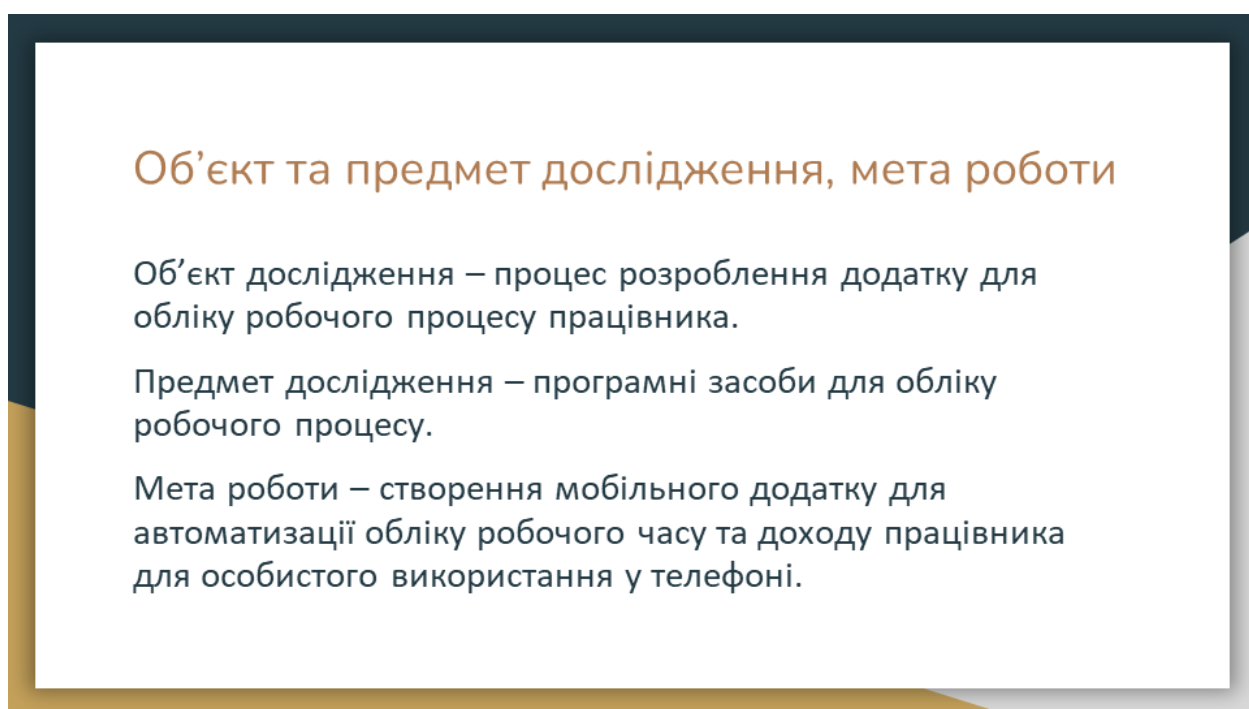


Рисунок В.2 – Об'єкт та предмет дослідження, мета роботи

## Вибір ОС

| Операційна система | Доля на ринку мобільних пристроїв, % |
|--------------------|--------------------------------------|
| Android            | 70.93                                |
| IOS                | 29,07                                |

Рисунок В.3 – Порівняння ОС

## Вибір мови програмування

| Критерії порівняння   | Мови програмування                               |                                                    |                                                  |                                                  |
|-----------------------|--------------------------------------------------|----------------------------------------------------|--------------------------------------------------|--------------------------------------------------|
|                       | Java                                             | Kotlin                                             | Dart                                             | C#                                               |
| Продуктивність        | Помірна (знижена через використання JVM)         | Помірна (знижена через використання JVM)           | Висока (нативна компіляція за допомогою Flutter) | Помірна (знижена через використання .NET MAUI)   |
| Синтаксис             | Базується на C/C++, популярний, середня важкість | Сильно базується на Java, трохи збільшена важкість | Базується на C/C++, популярний, середня важкість | Базується на C/C++, популярний, середня важкість |
| Підтримка спільнотою  | Дуже висока                                      | Висока (активно збільшується)                      | Помірна                                          | Висока                                           |
| Підтримка розробником | Висока                                           | Висока                                             | Висока                                           | Висока                                           |

Рисунок В.4 – Порівняння мов програмування

## Вибір IDE

| Критерії оцінювання               | IDE                                                     |                                                                                    |                                                                       |
|-----------------------------------|---------------------------------------------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
|                                   | Microsoft Visual Studio                                 | Visual Studio Code                                                                 | Android Studio                                                        |
| Швидкість                         | Помірна                                                 | Висока (може бути значно змінена використанням розширень)                          | Помірна                                                               |
| Ресурсозатратність                | Висока                                                  | Низька (може бути значно змінена використанням розширень)                          | Висока                                                                |
| Підтримувані мови програмування   | Широкий список мов, підтримуваних платформою .NET       | Майже будь яка мова програмування (за рахунок розширень)                           | Java, Kotlin                                                          |
| Інтерфейс                         | Дешо застарілий                                         | Сучасний, зручний                                                                  | Сучасний, зручний                                                     |
| Можливості розширення функціоналу | Помірні                                                 | Дуже високі (IDE базується на роботі з розширеннями, в тому числі користувацькими) | Високі (Gradle дозволяє зручне використання користувацьких бібліотек) |
| Передбачені інструменти           | Широкий набір інструментів для обраної області розробки | Мінімальний                                                                        | Широкий набір інструментів для мобільної розробки                     |

Рисунок В.5 – Порівняння IDE

## Вибір СКБД

| Критерій                     | СКБД      |              |              |
|------------------------------|-----------|--------------|--------------|
|                              | SQLite    | MySQL        | PostgreSQL   |
| Наявність серверної частини  | Відсутня  | Присутня     | Присутня     |
| Розмір                       | <1 Мб     | <600 МБ      | >1 Гб        |
| Підтримка мобільних додатків | Підтримує | Не підтримує | Не підтримує |

Рисунок В.6 – Вибір СКБД

## Структура БД

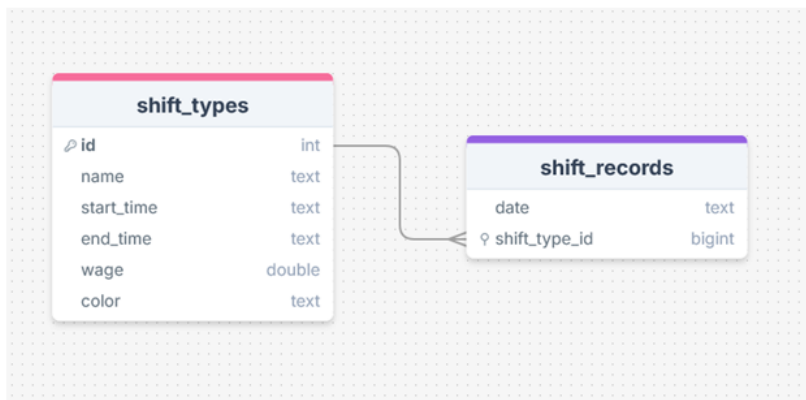


Рисунок В.7 – Структура БД

## Структура базових компонентів

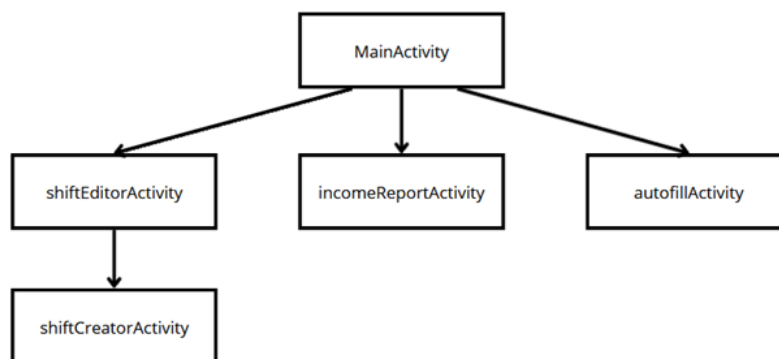


Рисунок В.8 – Структура базових компонентів

## Діаграма прецедентів користувача

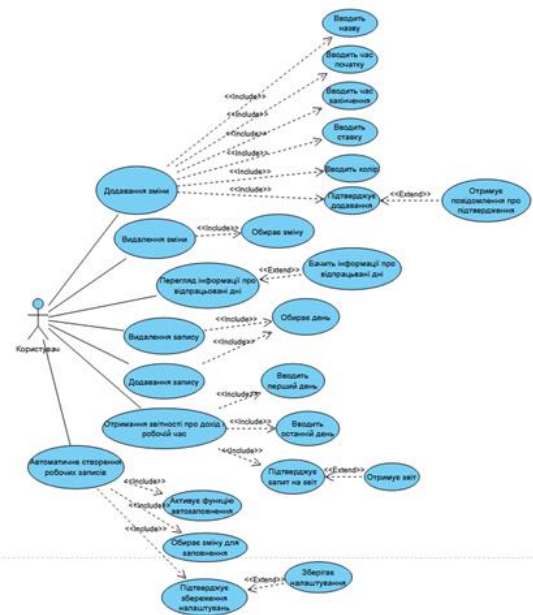


Рисунок В.9 – Діаграма прецедентів

## Головна сторінка, редактор змін, створювач змін

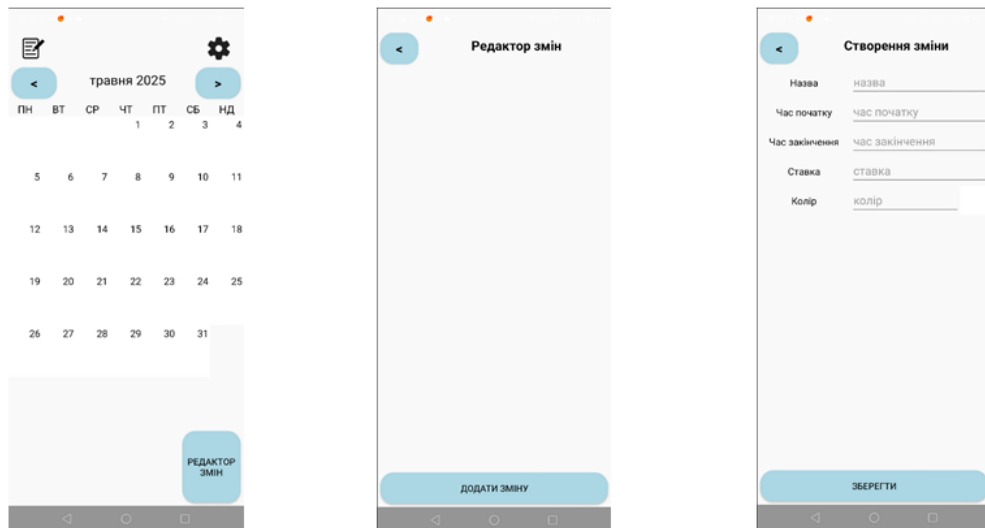


Рисунок В.10 – Макети інтерфейсу

## Автозаповнення, звіт

**Автозаповнення**

☐ Автозаповнення

Зміна    Обрати зміну...

ЗБЕРЕГТИ

**Звіт з доходу**

Від: startDay

До: startDay

Робочий час    Дохід

ОТРИМАТИ ДАНІ

Рисунок В.11 – Макети Інтерфейсу

## Тестування

**травня 2025**

ПН ВТ СР ЧТ ПТ СБ НД

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Вечірня

РЕДАКТОР ЗМІН

**травня 2025**

ПН ВТ СР ЧТ ПТ СБ НД

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

РЕДАКТОР ЗМІН

**May 2025**

ПН ВТ СР ЧТ ПТ СБ НД

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

РЕДАКТОР ЗМІН

Рисунок В.12 – Тестування

## Висновки

У процесі виконання роботи було розроблено мобільний застосунок “Календар робочих змін”, що дозволяє користувачеві зберігати і аналізувати дані про робочій процес, і керувати ними зручними та інтуїтивними методами.

Було проведено аналіз предметної області, в результаті якого було сформульовано основні поняття і проблеми області, вимоги до програмного забезпечення, було описано і проаналізовано аналогічні програмні продукти.

Аналіз і порівняння доступних програмних засобів дозволило прийняти рішення розробити Android-застосунок мовою Java у IDE Android Studio з використанням системи керування базами даних SQLite. Вибір цих засобів позитивно вплинув на процес розробки.

Було детально описано розроблені програмні компоненти, вичерпно змодельовано методику роботи користувача з програмою і проведено тестування працездатності. Тестування продемонструвало, що розроблений застосунок справно виконує очікувані функції.

Рисунок В.13 – Висновки