МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Мобільний додаток «Розумний будильник»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-82 Паляниця Богдан Олегович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2019 р.

Науковий керівник _____ к.т.н., доц., Алексенко О. В.
(підпис)

Голова комісії _____ Шифрін Д. М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав секцією ІТП
_____В. В. Шендрик
«__» _____ 2019 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Паляниця Богдан Олегович
(прізвище, ім'я, по батькові)

1 Тема проекту *Мобільний додаток «Розумний будильник»*

затверджена наказом по університету від «19» листопада 2019 р. №2305-III

2 Термін здачі студентом закінченого проекту «_10__» ___грудня___ 2019 р.

3 Вхідні дані до проекту планування робіт, технічна документація операційної системи Android, технічна документація технологій створення мобільних додатків

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, моделювання мобільного додатку, реалізація мобільного додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність проекту, постановка задачі, задачі проекту, дослідження аналогів, порівняння додатків-аналогів, функціональні і нефункціональні вимоги, моделювання роботи додатку, діаграма варіантів використання, архітектура додатку, засоби реалізації, демонстрація мобільного додатку, апробація результатів роботи, висновки

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Аналіз предметної області	*Алексенко О.В.*		
Постановка задачі та методи дослідження	*Алексенко О.В.*		
Моделювання мобільного додатку	*Алексенко О.В.*		
Реалізація мобільного додатку	*Алексенко О.В.*		

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Постановка задачі і розробка вимог	09.09.19 – 16.09.19	
2	Проведення аналізу предметної області і додатків аналогів	16.09.19 – 21.09.19	
3	Моделювання процесу роботи додатку	22.09.19 – 03.10.19	
4	Проектування та розробка БД	05.10.19 – 07.10.19	
5	Розробка інтерфейсу додатку	07.10.19 – 17.10.19	
6	Розробка функціоналу додатку	18.10.19 – 15.11.19	
7	Тестування мобільного додатку	21.11.19 – 25.11.19	
8	Формування звітів по проекту	25.11.19 – 09.12.19	
9	Представлення роботи	10.12.19 – 19.12.19	

Магістрант _____ Паляниця Б.О.

Керівник роботи _____ к.т.н., доц. Алексенко О.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Мобільний додаток «Розумний будильник». Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 31 найменування, 3 додатків. Загальний обсяг роботи − 80 сторінок, у тому числі 55 сторінок основного тексту, 3 сторінки списку використаних джерел, 22 сторінки додатків.

Кваліфікаційну роботу магістра присвячено розробці мобільного додатку будильника для полегшення процесу планування розпорядку дня. Спочатку було визначено та обґрунтовано актуальність проекту та досліджено мобільні додатки-аналоги. Далі сформовано задачі проекту та проведено аналіз сучасного стану застосування мобільних додатків для вирішення поставлених задач.

Наступним етапом визначилась мета роботи та функціональні вимоги до мобільного додатку. Було проведено планування робіт із розробки додатку. Розроблене планування робіт наведено у додатку А. Інструментами розробки програмного продукту було обрано мову програмування Java та IDE Android Studio.

Після цього було змодельовано процес роботи мобільного додатку та його процес розробки. Проведено моделювання функціонування системи, розроблено структуру даних використаних у додатку та спроектовано інтерфейс мобільного додатку та його логотип.

Останній етап передбачає власне розробку самого додатку. Розроблено класи, методи та базу даних, які в сукупності складають робочий додаток. Після розробки проводилось тестування на можливі помилки, у разі їх виявлення додаток допрацьовувався. Результати розробки у вигляді лістингу представлені у додатку Б. Результатом проведеної роботи є готовий працездатний мобільний додаток будильнику для полегшення процесу планування розпорядку дня.

Ключові слова: мобільний додаток, програмний продукт, будильник, нагадування, сповіщення, користувач, Android Atudio, Java, база даних.

ЗМІСТ

ВСТУП

Інформаційні технології забезпечують вже всі види діяльності та інтересів людей, у т.ч. процеси, пов’язані із їхнім приватним життям. В першу чергу розширюється неба у мобільних додатках, які будуть допомагати людям організовувати свій розклад [1]. Щоб вирішити це питання, пропонується розумний будильник, який допоможе людині не тільки прокинутися вчасно, а може бути також використаний для нагадування про певне завдання, яке потрібно виконати або заплановану зустріч.

Об’єктом дослідження є інформаційні технології, які забезпечують процеси планування розпорядку дня. Предметом дослідження є мобільний додаток планування особистого розпорядку. Метою магістерської роботи є реалізація проекту, що передбачає розробку мобільного додатку, який користувач зможе застосувати як будильник, а також який дозволить швидко і легко встановити нагадування на важливі задачі чи події у повсякденному житті. Для досягнення даної мети були поставлені наступні задачі:

- аналіз предметної області, а саме: огляд основних можливостей операційної системи (ОС) Android, знайомство із розробкою мобільних додатків під дану ОС, знайомство з розміткою для створення дизайну мобільних додатків.
- проектування додатку, у т.ч. інтерфейсу та структури використовуваних даних;
- реалізація продукту як мобільного додатку;
- тестування програмного продукту (ПП).

Основними засобами розробки мобільного додатку буде обрано інтегроване середовище розробки (IDE) Android Studio, мову програмування Java та вбудовано у ПП бібліотеку SQLite, яка реалізує роботу з базами даних [2]. Практична

значущість роботи полягає в тому, що мобільний додаток може бути використаний усіма користувачами для поліпшення планування розкладу свого дня.

У першому розділі магістерської роботи проводиться аналіз сучасного стану застосування мобільних додатків для вирішення поставленої задачі та проводиться огляд літератури. Також проводиться огляд існуючих мобільних додатків, їх порівняння та виявлення недоліків, які сприяють розробці власного мобільного додатку.

Другий розділ передбачає визначення мети та задач проекту та мобільного додатку в цілому. А також проводиться аналіз та вибір засобів та методів реалізації мобільного додатку.

У третьому розділі проводиться моделювання розроблюваної системи. Зокрема створюється структура даних для подальшої побудови бази даних (БД). За допомогою графічної мови IDEF0 будується контекстна діаграма, потім будується декомпозиція першого рівня у нотації IDEF0. Для кращого розуміння які процеси буде виконувати актор при використанні додатку, буде створена діаграма варіантів використання за допомогою мови UML та визначено функціональні можливості кожного із виділених варіантів використання продукту. Також у цьому розділі розроблюється макет дизайну додатку та створюється логотип.

У висновку наводяться результати, досягнуті в ході розробки, тестування і впровадження розробленого додатка, а також перспективи розвитку даного продукту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз сучасного стану застосування мобільних додатків для вирішення поставлених задач

Будильники існують уже багато років і використовуються для багатьох інших цілей, ніж просто змусити когось прокинутися вчасно. Оскільки їх використовують для багатьох інших цілей, вони невпинно еволюціонували. Не лише апаратним забезпеченням, але й програмним забезпеченням (ПЗ) [3].

Деякі з існуючих систем будильника включають будильник, гучність якого поступово збільшується чи повторюється деякий час. Деякі з цих систем також надають користувачам додаткові переваги, такі як радіо, mp3 та багато іншого. Кожна з цих систем далека одна від одної і є корисною в тій чи іншій мірі [4].

Звичайний будильник - це найпоширеніший тип будильника і використовується для сповіщення людини про щось у визначений час. Він також використовує різні способи оповіщення, як спалах екрану, вібрації та ін. Цей тип будильників сьогодні можна знайти окремо або як програмне забезпечення у більшості мобільних телефонів. Цей вид будильника простий у використанні і насправді є найбільш використовуваним типом будильника.

Будильники, сигнал яких поступово збільшується, також базуються на загальноприйнятому принципі будильників. Замість того, щоб розбудити когось звичайним способом, він збільшує свою гучність щоразу, коли його відкладають. Також в комплекті є можливість встановити сигнал улюбленої музики.

1.2 Огляд існуючих мобільних додатків

Ринок мобільних додатків досить великий і з всіма необхідними ознайомитись неможливо, тому розглянемо і порівняємо можливості деяких мобільних додатків будильників для планування задач з магазину додатків.

Google Play – магазин додатків від компанії Google, що дозволяє власникам пристроїв з операційною системою Android купувати і встановлювати різні програми. У Google Play можна знайти багато корисних і різноманітних додатків. У магазині присутні як платні, так і безкоштовні програми [5].

Будильник прошивки MIUI

Користуючись вже тривалий час цим будильником було виділено наступні переваги цього мобільного додатку:

- мінімалістичний та інтуїтивно зрозумілий інтерфейс додатку;
- широкі можливості налаштування будильнику;
- велика база мелодій сигналу будильника.

Також є деякі недоліки, які представлені нижче:

- відсутня можливість створювати нагадування без сигналу;
- відсутня можливість персоналізувати нагадування.

Даний мобільний додаток призначений для операційної системи Android на прошивці MIUI. Приклади вікон даного додатку представлені на рис. 1.1.

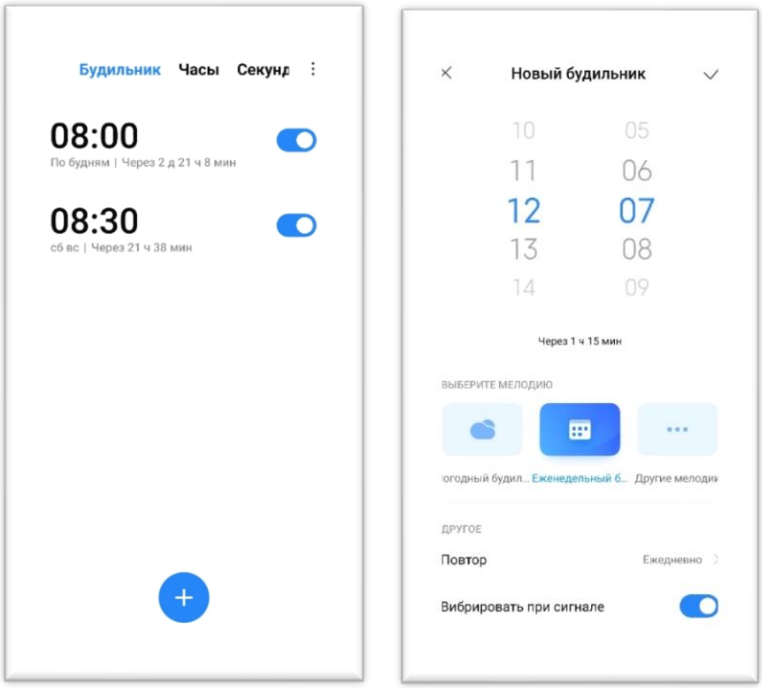


Рисунок 1.1 – Приклади функціоналу будильника прошивки MIUI

Ознайомимось з деякими мобільними додатками з магазину Play Market.

«Природный Будильник » [6]

Даний мобільний додаток орієнтований на функцію сигналу звуків природи. Розробником додатку є ddi.dev. Протестувавши мобільний додаток можна виділити такі плюси:

- можливість вибору звуковим сигналом наростаючі звуки природи;
- швидка і зручна можливість поставити спрацьовування будильнику на кілька хвилин пізніше;
- автоматична зміна сигналу сповіщення, у тому випадку коли користувача не будить один звук.

У противагу вищеперерахованим плюсам додатка можна виділити наступні мінуси додатку:

- виникає проблема з відключенням сигналу будильника;
- при максимальній гучності звуки дуже погано чутно;
- відсутня функція повтору кожні 5 днів;

- великий витрата заряду батареї смартфона.

Приклади функціоналу додатку представлені на рис. 1.2.

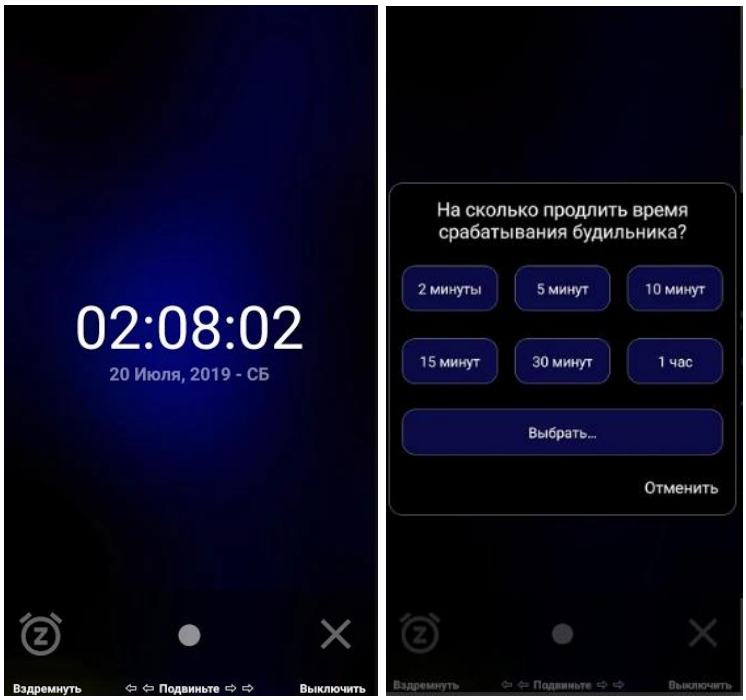


Рисунок 1.2 – Приклади функціоналу додатку «Природный Будильник»

Даний мобільний додаток призначений для операційної системи Android версій 4.1 та вище. Додаток доступний в магазині Google Play.

«Будильник - Умный будильник» [7]

Розробник описує свій продукт, як додаток для сигналізації з простим дизайном і інтелектуальними функціями. Розробником додатку є t440 soft. До особливостей мобільного додатку можна віднести наступні характеристики:

- простий інтерфейс та простота у використанні;
- вирішення математичних прикладів, щоб вимкнути будильник;
- встановлення тону для кожного сигналу;
- встановлення гучності будильника так, як ви хочете;
- повтор сигналу для кожного спеціального дня тижня або щодня.

Після тестування додатку та опрацювання відгуків на даний мобільний додаток можна сформулювати такі недоліки:

- для роботи з додатком вимагається доступ до контактів, відео та фото файлів на вашому пристрої;
- баги із часом спрацьовування будильнику;
- проблеми з працездатністю додатку;
- відсутність української мови;
- деякі інші менш значні недоліки.

Приклади функціоналу додатку представлені на рис. 1.3.

Рисунок 1.3 – Приклади функціоналу додатку «Будильник - Умный будильник»

Даний мобільний додаток призначений для операційної системи Android версій 4.2 та вище. Додаток доступний в магазині Google Play.

«Sleepytie - умный будильник» [8]

Додаток позиціонує себе як приємний, простий будильник із калькулятором часу сну. Розробником додатку є MaBo. До особливостей мобільного додатку можна віднести наступні характеристики:

- сигнал будильнику спрацьовує навіть, якщо смартфон стоїть на беззвучному режимі;
- автоматичний розрахунок часу пробудження після заведення будильнику;
- наростаючий сигнал.

Після тестування додатку та опрацювання відгуків на даний мобільний додаток можна виділити наступні мінуси додатку:

- складний функціонал і незрозумілий інтерфейс мобільного додатку;
- велика кількість дій для заведення будильнику;
- баги з зупинкою сигналу;
- відсутність української мови.

Приклади функціоналу додатку представлені на рис. 1.4.

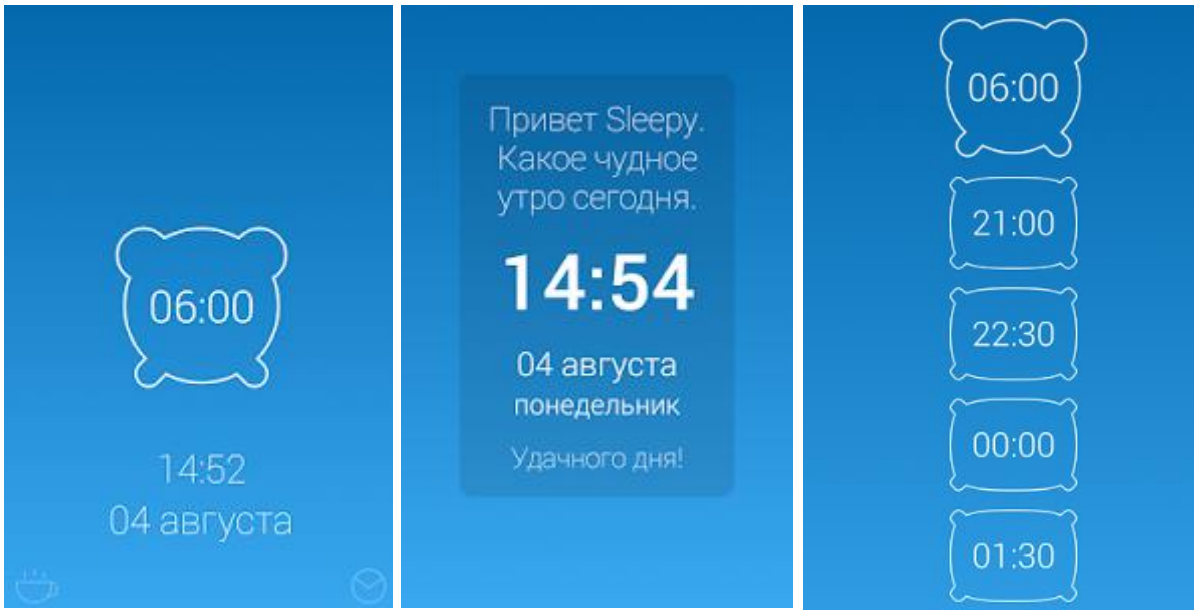


Рисунок 1.4 – Приклади функціоналу додатку «Sleepytie - умный будильник»

Даний мобільний додаток призначений для операційної системи Android версій 4.0.3 та вище. Додаток доступний в магазині Google Play.

Кожен з вищерозглянутих мобільних додатків-аналогів має як певні переваги так і деякі недоліки. Шляхом аналізу узагальнених даних була побудована наступна порівняльна таблиця:

Таблиця 1.1 Порівняльний аналіз існуючих аналогів

Функція	Назва мобільного додатку			
	Будильник прошивки MIUI	Природный Будильник	Умный будильник	Sleepytie
Функція регулярного повтору будильників	+	-	+	+
Можливість відкласти сповіщення будильнику	+	+	+	-
Перегляд всіх майбутніх і попередніх будильників	+	-	-	+
Можливість персоналізувати списки	-	-	-	-
Функція стійкого сповіщення	-	-	+	-
Функція створення заміток без будильнику	-	-	-	-
Інтуїтивно-зрозумілий інтерфейс	+	+	-	-
Підтримка української мови	+	-	-	-

Після опрацювання табл. 1.1 можна дійти висновку, що жоден з обраних для аналізу мобільних додатків не має всі обрані функції. Найбільш оптимальним рішенням у цьому випадку є розробка власного мобільного додатку, який буде володіти всіма виділеними функціями.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі

Після аналізу сучасного стану застосування інформаційних технологій для вирішення поставленої задачі, огляду доступної літератури та порівняння мобільних додатків-аналогів можна сформулювати мету магістерської роботи. Метою магістерської роботи є реалізація проекту, що передбачає розробку мобільного додатку, фунціонал якого дозволить користувачеві не тільки використовувати додаток як будильник, а й дозволить швидко і легко встановити нагадування на важливі задачі чи події у повсякденному житті.

Розроблюваний мобільний додаток надасть можливість використання наступного функціоналу:

- можливість перегляду активних будильників і тих, які вже завершились;
- функція створення нагадування без звукового сигналу;
- можливість переслати нагадування в месенджерах;
- можливість персоналізації будильника (вибір кольору, іконки, створення списків);
- функція вибору інтервалу повтору будильника (кожну годину, день, неділю, місяць, рік, окремі дні неділі, кожні декілька одиниці часу);
- функція стійкого сповіщення, без можливості його приховати;
- можливість вибору власної мелодії;
- можливість включення або вимкнення вібрації будильника або LED-індикатора.

До задач проекту можна віднести: визначення основних вимог до функціональності, поведінки і інтерфейсів додатку, вибір цільової мобільної платформи, проектування додатку і взаємодію окремих модулів та реалізація

додатку, що забезпечує виконання основного функціоналу. Потім необхідно наповнити додаток необхідною інформацією. Спроектований і реалізований продукт буде являти повноцінний мобільний додаток. Додаток повинен бути націлений на людей, які хочуть полегшити процес створення свого розпорядку дня та яким не достатньо настільного будильнику або стандартного додатку у смартфоні.

2.2 Вибір засобів реалізації мобільного додатку

За останніми даними порталу StatCounter Global Stats на сьогоднішній день найпопулярнішими мобільними ОС у світі є Android, частка якої складає 76,67%, iOS з часктою 22,09% та інші ОС, частка кожної з яких не перевищує 0,4% [9]. Статистика збиралась з жовтня 2017 року по жовтень 2019 року. Графік ринку мобільних операційних систем у всьому світі наведений на рис. 2.1.

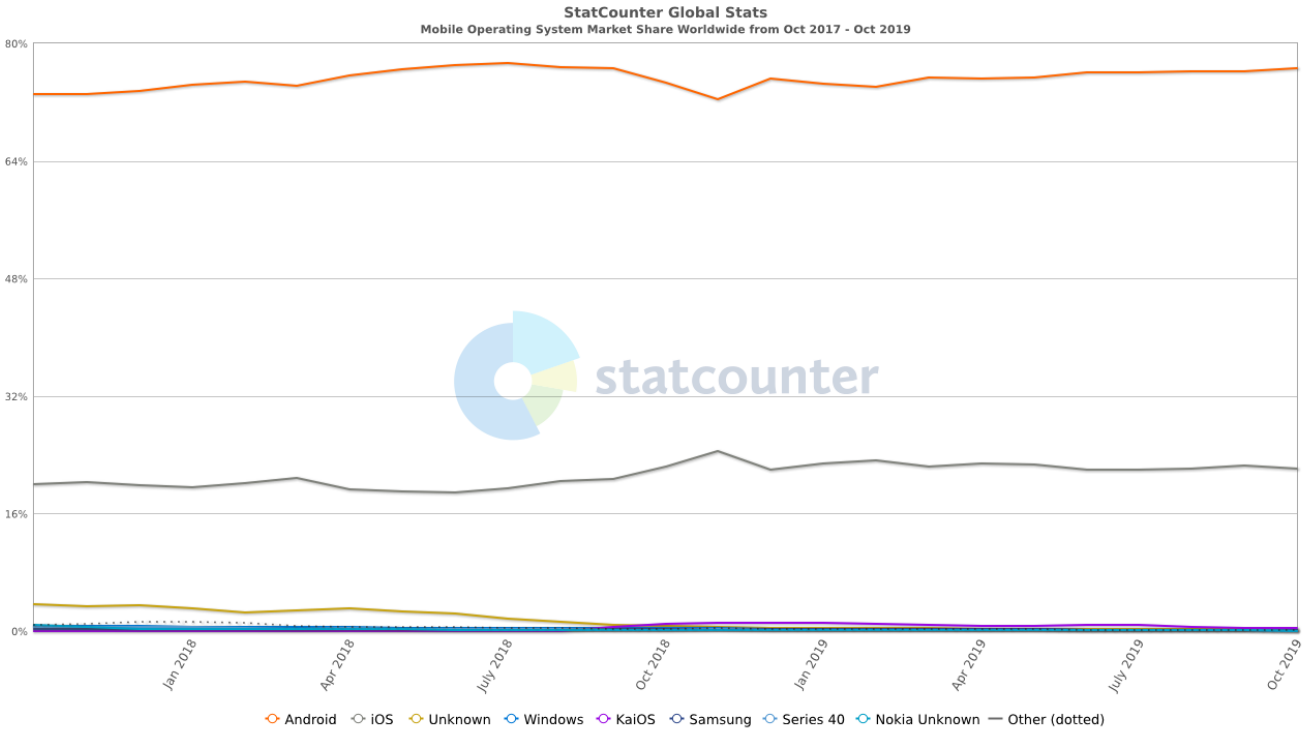


Рисунок 2.1 – Статистика мобільних операційних систем у світі

Кількість пристроїв з системою Android в Україні трішки різниться і складає 79,88%, а iOS складає 18,93% [10]. Більш докладні дані наведені на рис. 2.2.

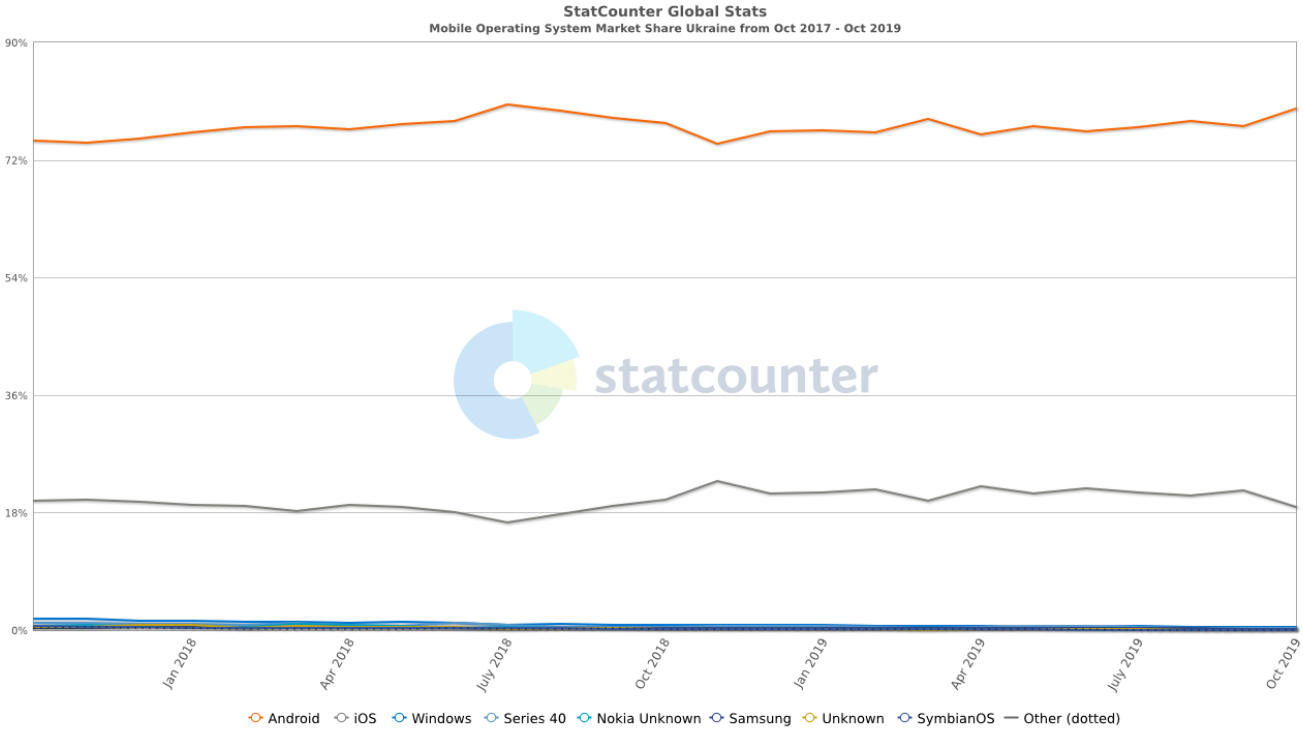


Рисунок 2.2 – Статистика мобільних операційних систем в Україні

Згідно з рис. 2.1 та рис. 2.2 найпоширенішою мобільною операційною системою є Android, а отже розробка мобільних додатків саме під цю ОС є найбільш актуальною.

Також була проаналізована статистика використання версій Android і найпопулярнішими версіями є версія 9.0 Pie (35,98%), версія 8.1 Oreo (14,64%) та версія 6.0 Marshmallow (13,43%). Найнижчою версією Android за останні три роки серед найбільш використовуваних є версія 4.4, а отже і розробка буде проводитися починаючи з цієї версії [11]. Детальний графік приведений на рис. 2.3.

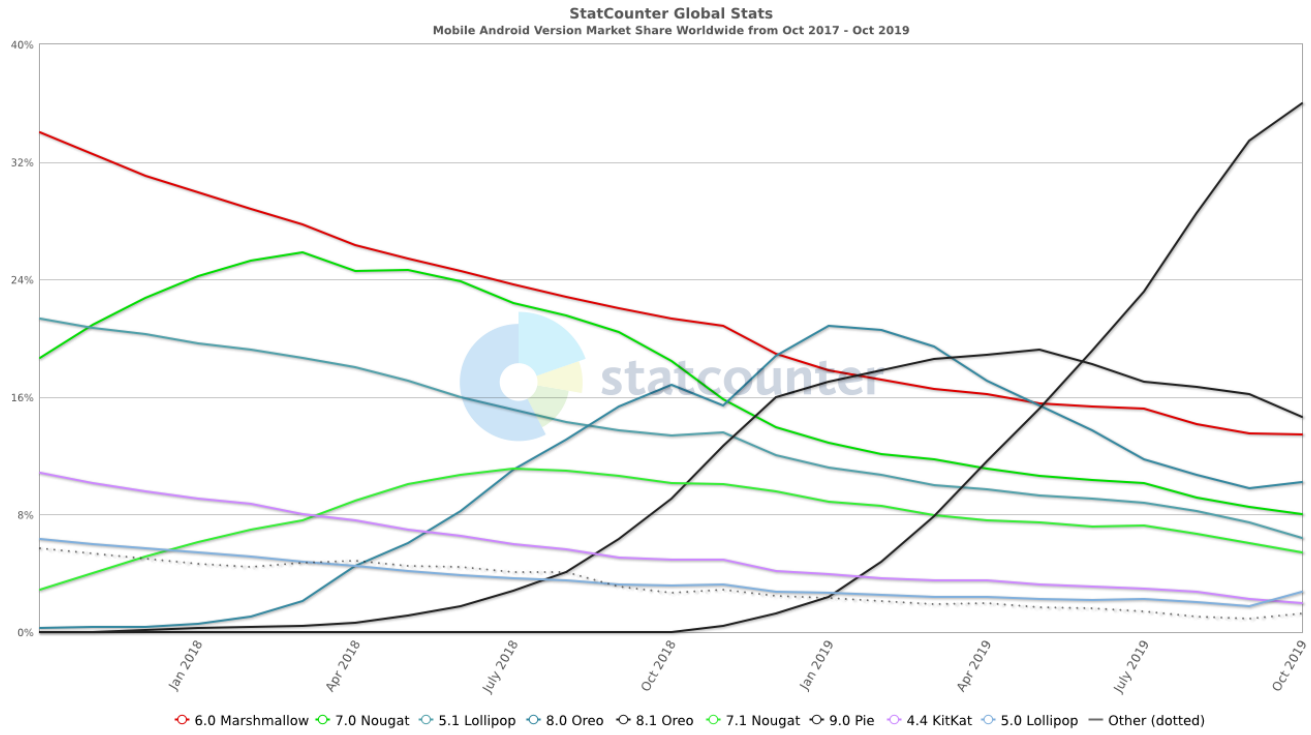


Рисунок 2.3 – Статистика актуальних версій Android

Так як для реалізації даного проекту була обрана ОС Android, розглянемо деякі доступні для розробки під цю платформу мови програмування. За останні роки можна виділити дві основні мови програмування: Java та Kotlin.

Kotlin це новітня статично-типізована мова програмування з відкритим кодом. Він може ефективно запускатися на віртуальній машині Java (JVM). Kotlin розроблений JetBrains і офіційно підтримується Google. Недавнє опитування порталу Jexenter помістив Kotlin на шосте місце серед технологічних трендів. В даний час Kotlin використовується для створення Android-додатків такими лідерами бізнесу як Pivotal, Atlassian, Pinterest, Evernote і Uber. Остання статистика App brain показує, що в сегменті топових додатків 2018 року Kotlin займає 25,3% ринку. При цьому 40,76% нових інсталяцій додатків також припадають на додатки, написані на Kotlin.

Щоб порівняти Kotlin з Java, було виділено переваги і недоліки цієї мови:

- код відкритий. Впровадження не вимагає матеріальних витрат;
- програми використовують фреймворки і бібліотеки Java;
- Kotlin компілюється в байткод JVM або в JavaScript;
- його легко вивчити;
- Null безпека Kotlin на високому рівні;
- виконання код-рев'ю не є проблемою.

Але окрім переваг мови також можна виділити і недоліки Kotlin:

- швидкість компіляції – розробники скаржаться на коливання швидкості компіляції коду на Kotlin [12]. У деяких випадках вона відбувається дуже швидко, а в інших помітно повільніше;
- менша підтримка – кількість ресурсів для вивчення цієї мови обмежена;
- у Kotlin в порівнянні з Java відсутні: статичні члени; примітивні типи, які не є класами; приватні поля; wildcard-типи; відмічені винятки.

Java вважається однією з кращих мов для розробки додатків. Однією з основних особливостей цієї мови є об'єктно-орієнтованість. І вона допомагає далеко не тільки в підтримці розробки Android-додатків. Недавній TIOBE index [13] (грудень 2019) показав, що Java лідирує серед топових мов програмування. Після опрацювання літератури можна виділити основні плюси мови:

- за допомогою віртуальної машини Java програми на цій мові можуть запускатися практично в будь-якій систем;
- кросплатформеність – Java прекрасно підходить і для розробки кросплатформених додатків;
- ресурси – ця мова має вже готові бібліотеки і SDK для полегшення процесу розробки.

Серед мінусів даної мови програмування є:

- швидкість операцій – Java вимагає більше пам'яті і, в порівнянні з іншими мовами (C++, php, perl), працює набагато повільніше [14];
- на Java важче писати код, він вимагає більше часу для написання, в ньому більше помилок і багів;
- у Java в порівнянні з Kotlin відсутні: шаблони рядків; сінглтони; функції розширення; розумні приведення типів (smart casts).

Порівнявши всі переваги та недоліки кожної із мов було обрано мову Java, так як окрім її переваг над мовою Kotlin також важливим фактором став більший досвід в розробці мобільних додатків під ОС Android.

Стосовно вибору IDE для розробки, то було обрано Android Studio, так як це офіційний IDE для Android, програмний пакет побудований Google, у якому є все, що допоможе розробити надійний додаток для Android. Він відомий як третій найвідоміший IDE у світі та здатний прискорити процес розробки, не приносячи шкоди якості. Деякі функції, які роблять Android Studio найкращим інтегрованим середовищем розробки, - це його гнучка система побудови, аналізатор у режимі реального часу, інтуїтивно зрозумілий макет та редактор смарт-коду [15].

3 МОДЕЛЮВАННЯ МОБІЛЬНОГО ДОДАТКУ

Моделювання розроблюваного мобільного додатку є одним із найважливіших етапів створення якісного програмного продукту. Під час даного процесу розробляються проектні рішення щодо вибору платформи, де буде функціонувати майбутній програмний продукт, визначаються вимоги до майбутнього призначеного для користувача інтерфейсу, з'ясовуються найбільш підходящі інструменти для досягнення поставленої мети.

3.1 Моделювання процесу роботи мобільного додатку

Комплексне визначення функціонального моделювання (IDEF0) - це метод моделювання функцій для бізнес-систем та інженерний підхід для аналізу потреб. Техніка IDEF0 використовує вікно для представлення функцій у процесі та показує стосунки до дочірньої та батьківської систем. Ця методика дає креслення для розуміння систем організації [16]. Можна виділити основні переваги побудови контекстної діаграми IDEF0:

- забезпечує вдосконалений системний аналіз;
- забезпечує вдосконалені методи спілкування;
- допомагає зрозуміти систему та її зв’язки.

Діаграма IDEF0 наведено на рис. 3.1. Вона демонструє процес роботи мобільного додатку. На вході ми отримуємо потребу користувача у нагадуванні, а на виході отримуємо спрацьований будильник. Управліннями розробки мобільного додатку є нормативна документація, а механізмами в свою чергу: користувач, технічне і програмне забезпечення та база даних (БД). Точку зору моделювання процесу роботи мобільного додатку представляє розробник програмного забезпечення.

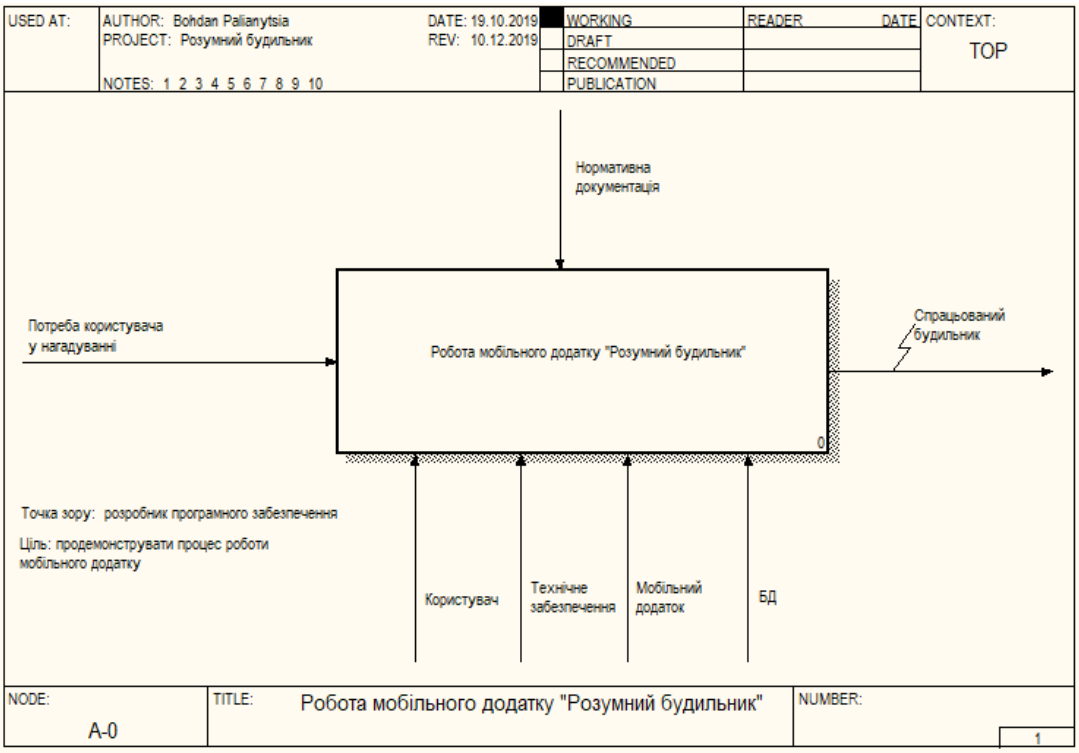


Рисунок 3.1 – Контекстна діаграма IDEF0 для роботи додатку

Декомпозиція першого рівня у нотації IDEF0 відображає взаємодію процесів роботи мобільного додатку (рис. 3.2).

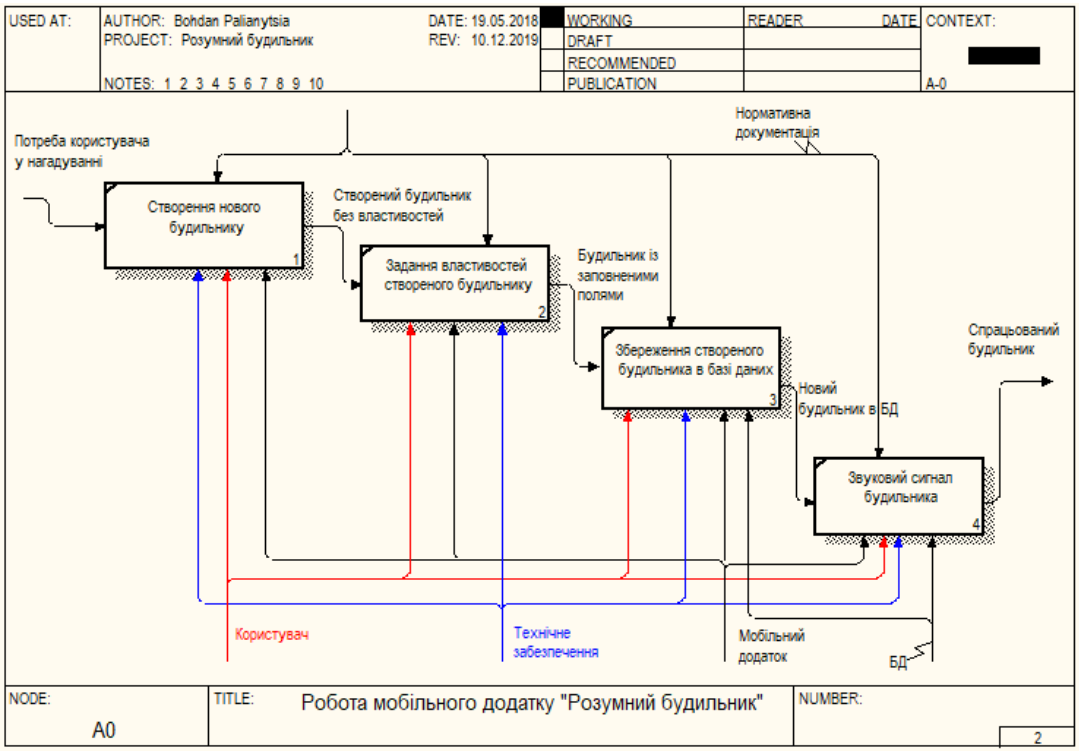


Рисунок 3.2 – Перший рівень декомпозиції IDEF0 роботи додатку

Процес роботи мобільного додатку поділяється на чотири основних блоки: створення нового будильнику, задання властивостей створеного будильнику, збереження створеного будильнику в базі даних та звуковий сигнал будильника. Результатом блоку «створення нового будильнику» є створений будильнику без властивостей, блоку «задання властивостей створеного будильнику» – будильник із заповненими полями, блоку «збереження створеного будильнику в базі даних» – новий запис в базі даних. На виході процесу роботи додатку отримуємо звукове сповіщення будильнику.

Також було розроблено діаграми, які демонструють процес розробки мобільного додатку. На рис. 3.3 представлена діаграма IDEF0. На вході маємо потребу в програмному продукті, а на виході отримуємо готовий програмний продукт. Управліннями розробки мобільного додатку є технічна документація ОС Android, технологій створення мобільних додатків, а механізмами в свою чергу: розробник, замовник та технічне і програмне забезпечення. Точку зору моделювання процесу розробки мобільного додатку представляє розробник ПЗ.

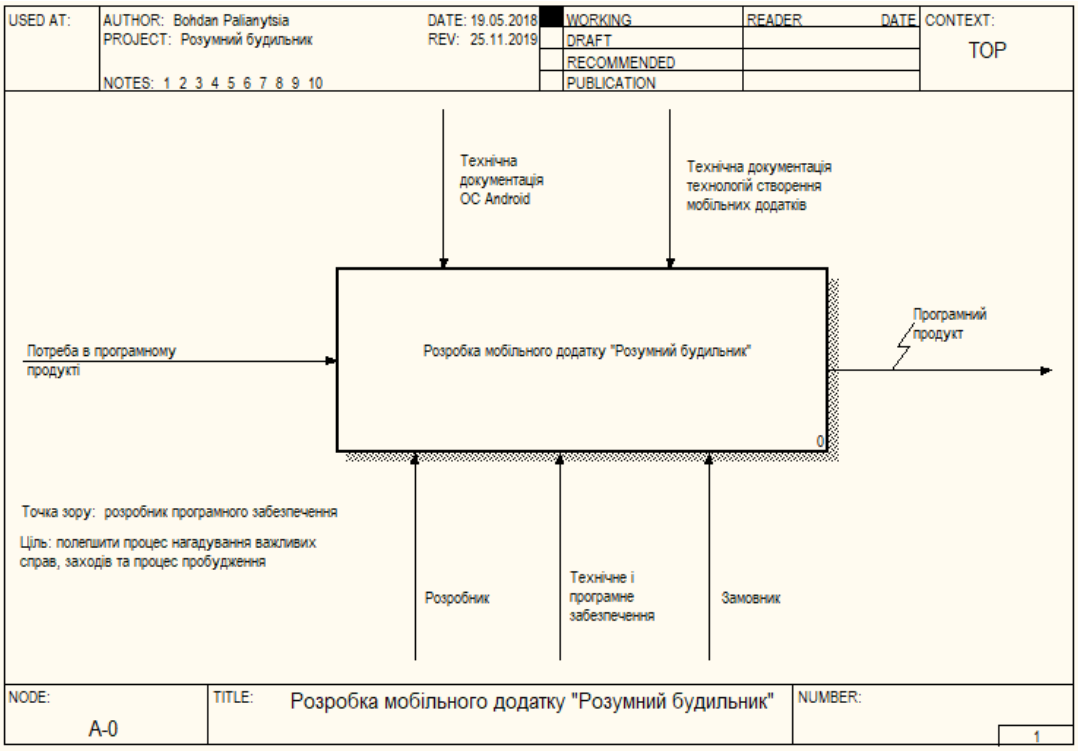


Рисунок 3.3 – Контекстна діаграма IDEF0 для розробки мобільного додатку

Декомпозиція першого рівня у нотації IDEF0 відображає взаємодію процесів розробки мобільного додатку (рис. 3.4).

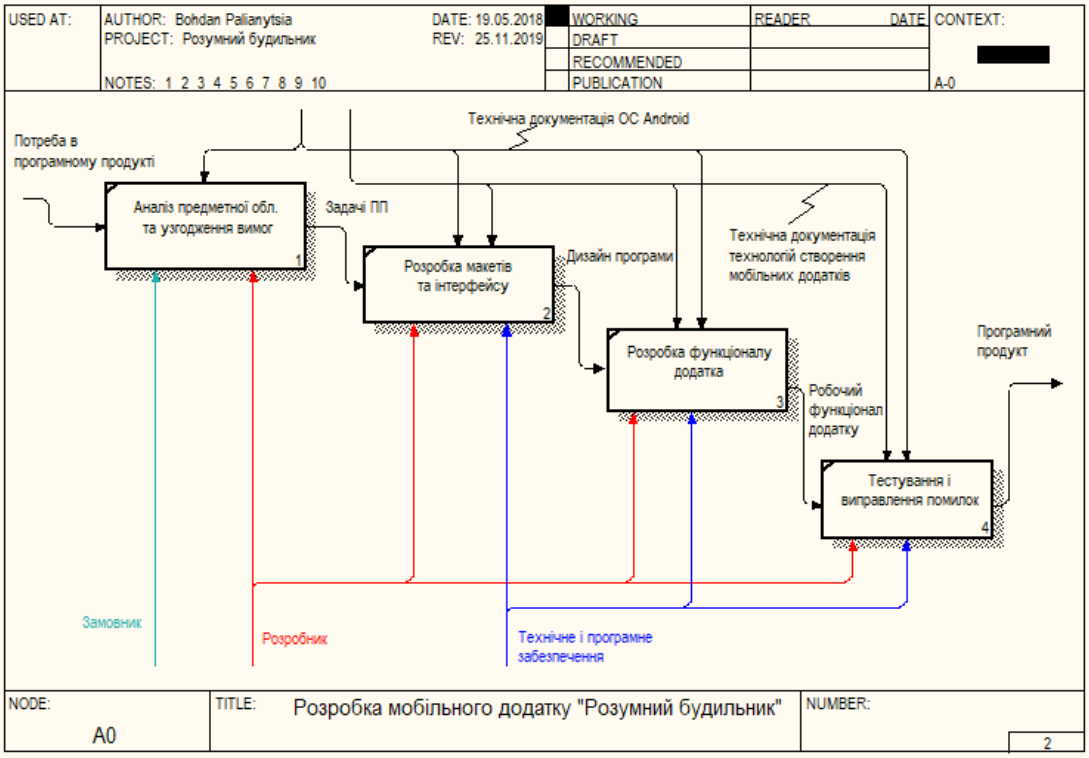


Рисунок 3.4 – Перший рівень декомпозиції IDEF0 розробки мобільного додатку

Процес розробки мобільного додатку поділяється на чотири основних блоки: аналіз предметної області та узгодження вимог, розробка макетів та інтерфейсу, розробка функціоналу додатку та тестування і виправлення помилок. Результатом блоку «аналіз предметної області і узгодження вимог» є задачі ПП, блоку «розробка макетів та інтерфейсу» – готовий дизайн програми, блоку «розробка функціоналу програми» – робочий функціонал додатку. На виході процес розробки отримуємо готовий програмний продукт.

3.2 Моделювання функціонування системи

Для того, щоб зрозуміти функціювання програмного продукту використовується модель варіантів використання, яка включає опис акторів, варіанти використання системи, взаємодію між ними у формі діаграми прецедентів.

Діаграма варіантів використання являє собою список дій, які зазвичай визначають взаємодію між роллю (відомої в уніфікованій мові моделювання як актор), і системи для досягнення мети [17]. Актором може бути людина чи інша зовнішня сутність. Діаграма варіантів використання може допомогти представити:

- сценарії, в яких система чи додаток взаємодіють з людьми, організаціями чи зовнішніми системами;
- цілі, які система чи додаток допомагає цим організаціям (відомим як актори);
- обсяг розроблюваної системи.

Можемо виділити два актори, які взаємодіють з розроблюваним додатком:

- користувач – людина, яка має змогу використовувати функціонал мобільного додатку;
- БД – система, що надає функціонал обробки сукупності даних, які використовуються мобільним додатком.

Також провівши детальний аналіз було виділено наступні основні варіанти використання даного мобільного додатку:

- ВВ 1 Перегляд створених будильників – ВВ дозволяє користувачеві переглядати активні будильники і ті, що вже спрацювали; ініціюється актором користувач;

  передумова: будильник створений; процес: користувач запускає мобільний

додаток і бачить активні будильники (рис. 4.7) чи спрацьовані (рис. 4.8); післяумова: всі будильники відображаються у списку будильників;

- ВВ 2 Створення нового будильнику – ВВ дозволяє користувачеві створити новий будильник визначивши всі необхідні його властивості; ініціюється актором користувач;

  передумова: додаток запущений; процес: користувач у списку будильників натискає кнопку створення нового будильнику (рис. 4.7). Відкривається форма визначення параметрів будильнику (рис. 4.15). Користувач обов'язково визначає Заголовок. Користувач може додати опис будильнику. Будильник має налаштування (рис. 4.12). Після настроювання параметрів Користувач повинен зберегти будильник (рис 4.16); післяумова: створений будильник показується у списку будильників (рис. 4.20);
- ВВ 3 Редагування будильнику – ВВ дозволяє користувачеві змінювати раніше створений будильник; ініціюється актором користувач;

  передумова: додаток запущений та створений будильник; процес: користувач у списку будильників натискає на створений будильник і кнопку редагувати (рис. 4.21). Відкривається форма з редагуванням будильнику (рис. 4.16). Користувач може змінити будь-які властивості будильнику. Після редагування Користувач повинен зберегти будильник; післяумова: змінений будильник показується у списку будильників;
- ВВ 4 Видалення будильнику – ВВ дозволяє користувачеві видалити раніше створений будильник; ініціюється актором користувач;

  передумова: додаток запущений та є будь-який будильник; процес: користувач у списку будильників натискає на будь-який будильник і кнопку видалити (рис. 4.21). Будильник видаляється;

  післяумова: видалений будильник зникає з бази даних;

- ВВ 5 Перегляд інформації про додаток – ВВ дозволяє користувачеві ознайомитися з детальною інформацією про додаток; ініціюється актором користувач;

  передумова: додаток запущений; процес: користувач у початковому вікні натискає кнопку з трьома точками (рис. 4.7). Відкривається форма про додаток (рис. 4.9). Користувач може переглядати детальну інформацію про додаток (рис. 4.11), форму з бібліотеками, контактну форму (рис. 4.10);
- ВВ 6 Відправка змісту сповіщення в месенджери – ВВ дозволяє користувачеві відправити створений будильник у месенджерах чи на пошту; ініціюється актором користувач;

  передумова: додаток запущений та створений будильник; процес: користувач обирає будь-який будильник. Відкривається вікно будильнику (рис. 4.21). Користувач може поділитися сповіщенням в месенджерах чи на пошту (рис. 4.22 та рис. 4.23);

  післяумова: відправлене сповіщення обраним каналом зв’язку;
- ВВ 7 Налаштування будильнику – ВВ дозволяє користувачеві налаштувати основні можливості будильнику; ініціюється актором користувач;

  передумова: додаток запущений та створений будильник; процес: користувач відкриває меню налаштувань (рис. 4.12) для зміни бажаних властивостей, наприклад зміним мелодії (рис. 4.13) чи частоти повтору будильнику (рис. 4.14). Після вмикання або вимкнення бажаної властивості – зміни вступають відразу;

  післяумова: змінені налаштування для окремого будильнику чи для загальні налаштування для всіх.

На основі виділених вище акторів та ВВ розроблена діаграма варіантів використання яка зображена на 3.5.

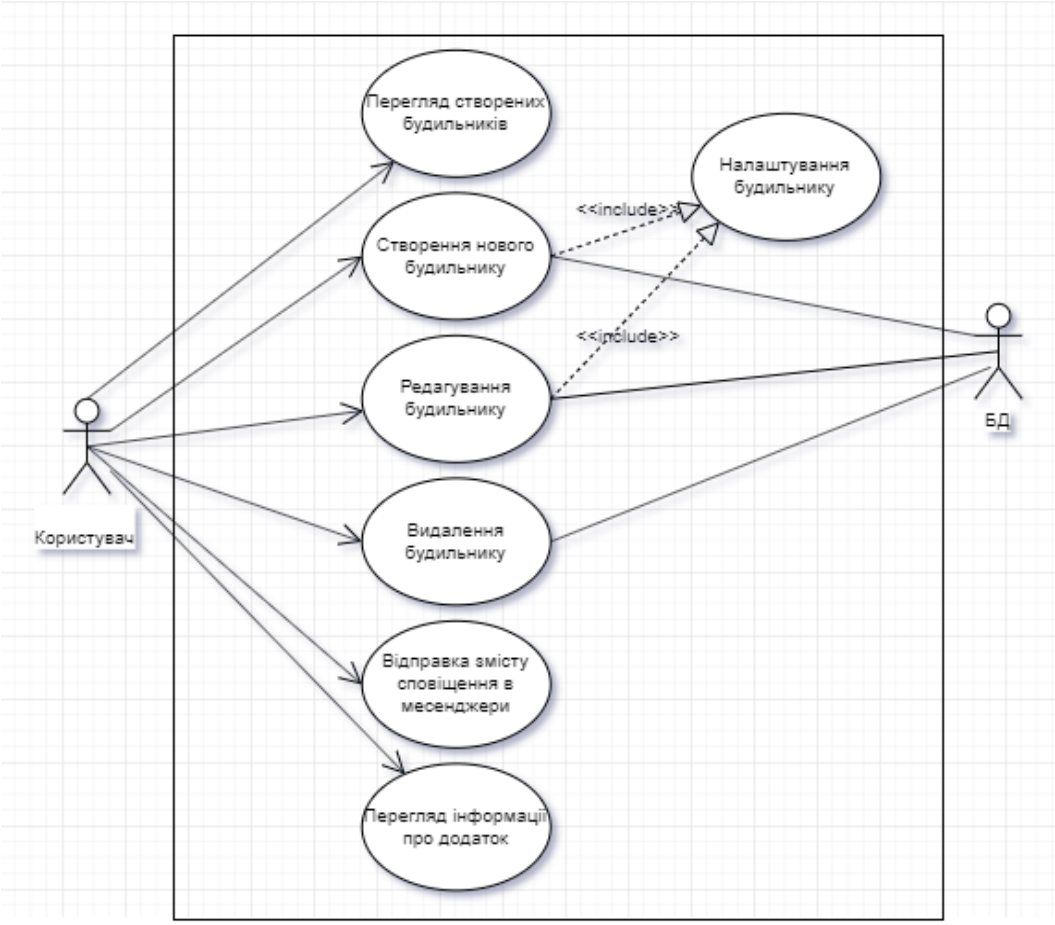


Рисунок 3.5 – Діаграма ВВ користувачем та БД

Варіанти використання «Створення нового будильнику» та «Редагування будильнику» включають варіант «Налаштування будильнику».

3.3 Структура даних використаних у додатку

Для коректного функціонування мобільного додатку і використання всіх можливостей програмного продукту було спроектовано базу даних (БД), де будуть зберігатися дані про будильник та його властивості.

Розглянемо спроектовані таблиці з їх полями. Таблиця DAYS_OF_WEEK містить у собі поле ID типу integer та поля днів неділі з типом text. Дана таблиця

призначення для зв’язку з таблицею NOTIFICATION для вибору повтору будильника в окремі дні неділі.

Таблиця бази даних NOTIFICATIONS зберігає інформацію про створений будильник. Поля даної таблиці з їх типами та описом перераховані у списку нижче:

- поле "id" типу integer – id будильника;
- поле "title" типу text – заголовок будильника;
- поле "content" типу text – опис будильника;
- поле "date_and_time" типу integer – дата та час спрацювання будильника;
- поле "repeat_type" типу integer – кількість повторів будильника;
- поле "forever" типу text – вмикач властивості повторювати завжди;
- поле "number_to_show" типу integer – кількість, яку повинен спрацювати будильник;
- поле "number_shown" типу integer – кількість вже спрацьованих будильників;
- поле "icon" типу text – іконка сповіщення;
- поле "colour" типу text – колір вмісту сповіщення;
- поле "interval" типу integer – інтервал повтору будильника.

Також спроектована таблиця ICONS для роботи з іконками для нагадувань. Таблиця має поля id типу integer – ключове поле, яке зв’язане з полем icon таблиці NOTIFICATION, поле name типу text для назви іконки та поле use_frequence типу integer для сортування іконок по частоті використання.

Таблиця PICKER_COLOURS призначена для підбору кольорів. Полями таблиці PICKER_COLOURS є colour типу integer для зберігання кольору та date_and_time типу integer для відображення тексту сповіщення заданим кольором коли він спрацює (зв’язаний з полем date_and_time таблиці NOTIFICATION).

На основі виділених таблиць та їх атрибутів можемо зробити діаграму типу «сутність-зв'язок». ER-діаграма мобільного додатка представлена на рис. 3.6.

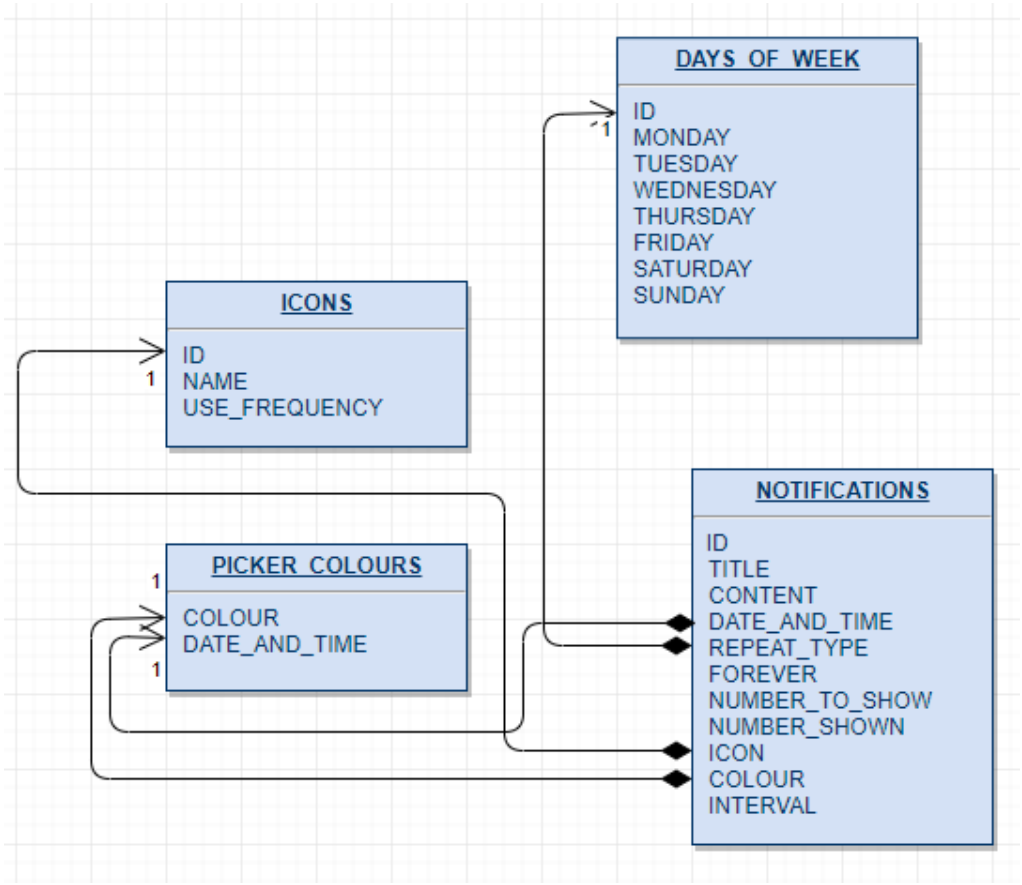


Рисунок 3.6 – ER-діаграма бази даних

Процес розробки БД наведений у розділі реалізації мобільного додатку.

3.4 Проектування інтерфейсу мобільного додатку

Згідно визначення, прототипування - це створення макета, моделі майбутнього мобільного додатка для того, щоб визначити правильність його структури, його функціональності та, в цілому, концепції програми. Макет має чудову властивість усувати непорозуміння між різними фахівцями (керівник, дизайнер, програміст, клієнт), залученими до проекту, структурувати думки і запобігати помилкам і виконання зайвої роботи ще на ранніх стадіях розробки. Можна тестувати майбутній додаток, використовуючи фокус-групу, це допоможе отримати корисну інформацію від майбутніх користувачів [18].

Для даного мобільного додатку було використано поняття «швидке прототипування» [19], яке допомогло швидко намітити основні вікна майбутнього програмного продукту. Макет було розроблено стандартними засобами Windows. Під час його розробки основними критеріями були наступні пункти:

- всі елементи повинні бути виконані в єдиному стилі;
- елементи вікон не повинні заважати один одному;
- продумати інтуїтивно-зрозумілу навігацію.

Створені макети дизайну представлені на рис. 3.7 – 3.9.

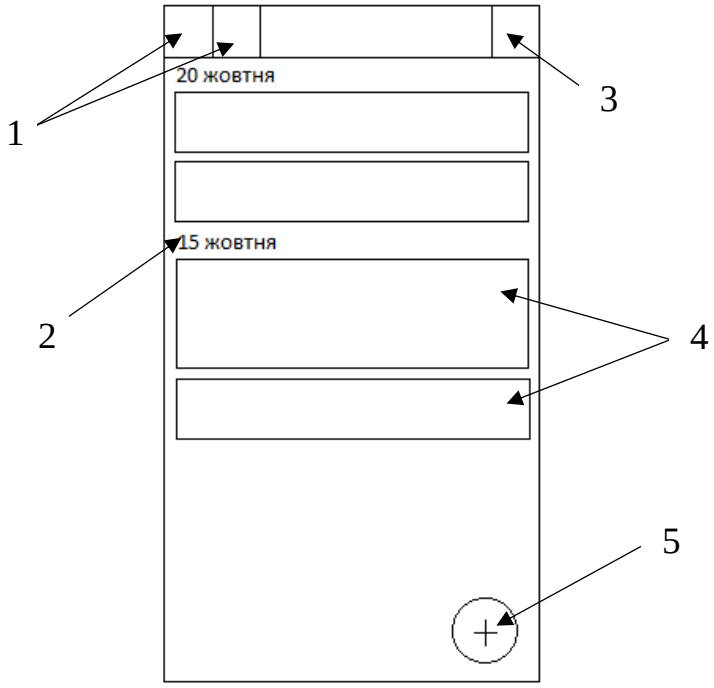


Рисунок 3.7 – Макет головного вікна додатку

Стрілками та позначеннями до них показані основні елементи екранів мобільного додатку. На рисунку, який представлений вище показано головне activity додатку. Роз’яснення до деяких елементів наведено нижче:

1. Кнопки переключення між активними будильниками і тими, що вже спрацювали.
2. Елемент, який відображає дату коли повинно бути нагадування.
3. Кнопка налаштувань для будильника.
4. Поля із створеними нагадуваннями.

5. Кнопка для створення нового будильника.

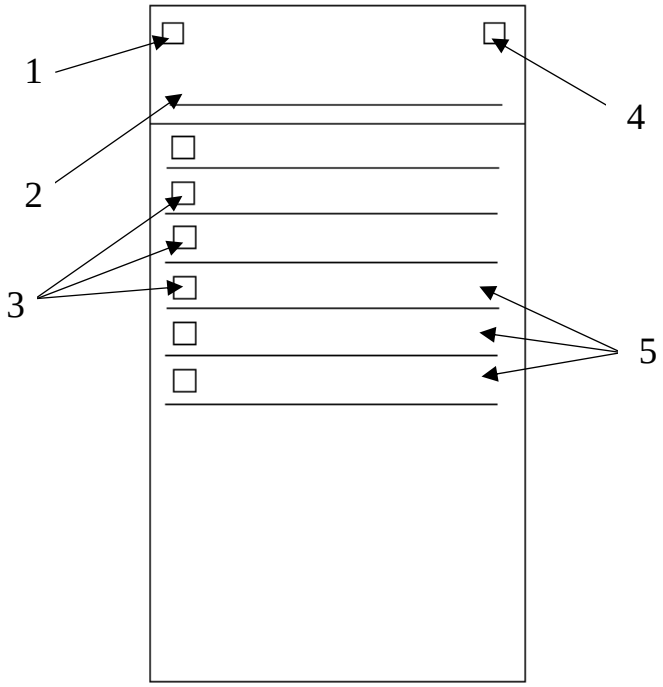


Рисунок 3.8 – Макет вікна створення нового будильнику

На даному рисунку показано activity із створення нового будильнику і задання його властивостей:

1. Кнопка відміни створення нового будильнику.
2. Поле задання заголовку для будильнику.
3. Іконки для властивостей будильника.
4. Кнопка збереження нового будильника.
5. Кнопки для задання властивостей будильника.

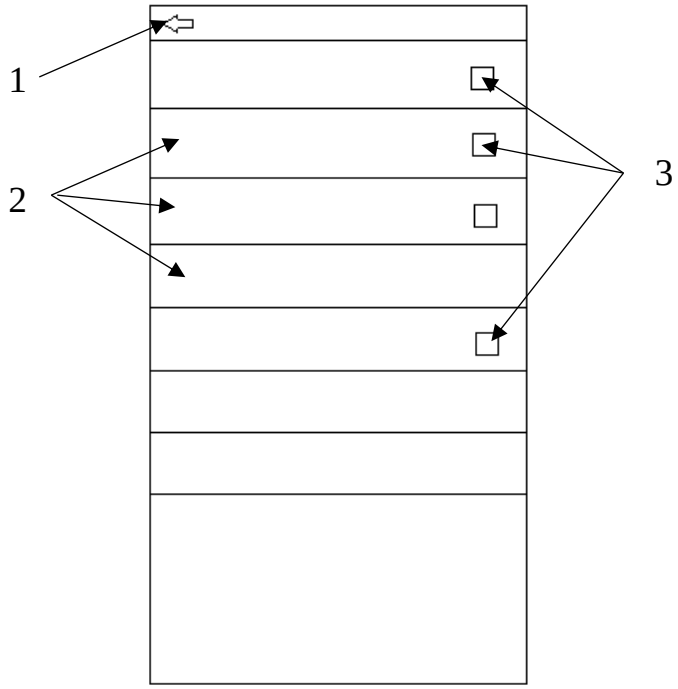


Рисунок 3.9 – головного вікна налаштування будильників

На рисунку, який представлений вище показано activity з налаштувань мобільного додатка:

1. Кнопка повернення до основного вікна додатку.
2. Поля з описом основних налаштувань додатка.
3. CheckBox для вибору чи відміни обраної функції додатка.

Для мобільного додатку «Розумний будильник» було створено логотип. Даний логотип відповідає наступним вимогам, які передбачені Google Play:

- підсумковий розмір: 512 x 512 пікселів;
- формат: 32-розрядний PNG;
- кольорова палітра: RGB;
- максимальний розмір файлу: 1024 КБ;
- форма: повний квадрат [20].

На рис. 3.10 показано загальний розмір об'єкта логотипу мобільного додатку.

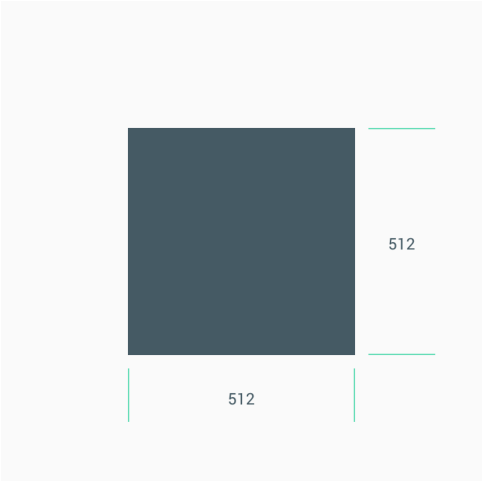


Рисунок 3.10 – Загальний розмір об'єкта

На рис. 3.11 показано контури об'єкта, який лежить всередині квадрата.

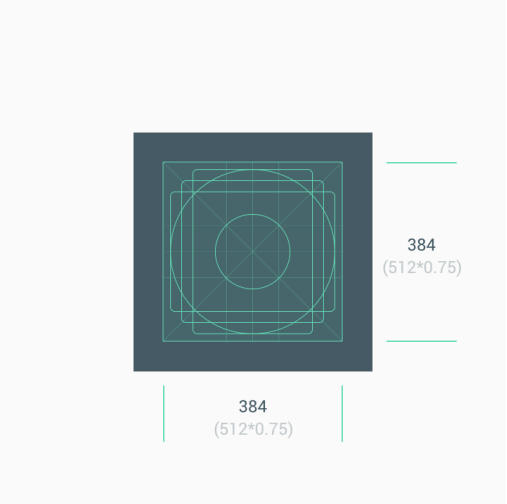


Рисунок 3.11 – Контури значка продукту

За основу логотипу було обрано мінімалістичний стиль з трьома основними кольорами: білий, сірий та блакитний. Логотип для мобільного додатку був розроблений за допомогою Adobe Illustrator – векторний графічний редактор, розроблений і поширюваний фірмою Adobe Systems [21]. Спочатку було нарисовано ескіз вручну на папері, потім було поетапно нарисовано логотип мобільного додатку, як показано на рис. 3.12.

Рисунок 3.12 – Етапи створення логотипу

Далі було завершено логотип і створено білий заокруглений фон з тінями. Логотип представлений на рис. 3.13 нижче.

Рисунок 3.13 – Логотип мобільного додатку

Кінцевий логотип відображає основні кольори додатку та його мінімалістичний інтерфейс.

4 РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ

Розробка мобільного додатку «Розумний будильник» відбувалася згідно розроблених макетів і створених діаграм, які розширено описані в третьому розділі «Моделювання мобільного додатку» даної магістерської роботи.

Було створено проект за допомогою IDE Android Studio. Діалогове вікно з налаштуваннями проекту представлені на рис. 4.1.

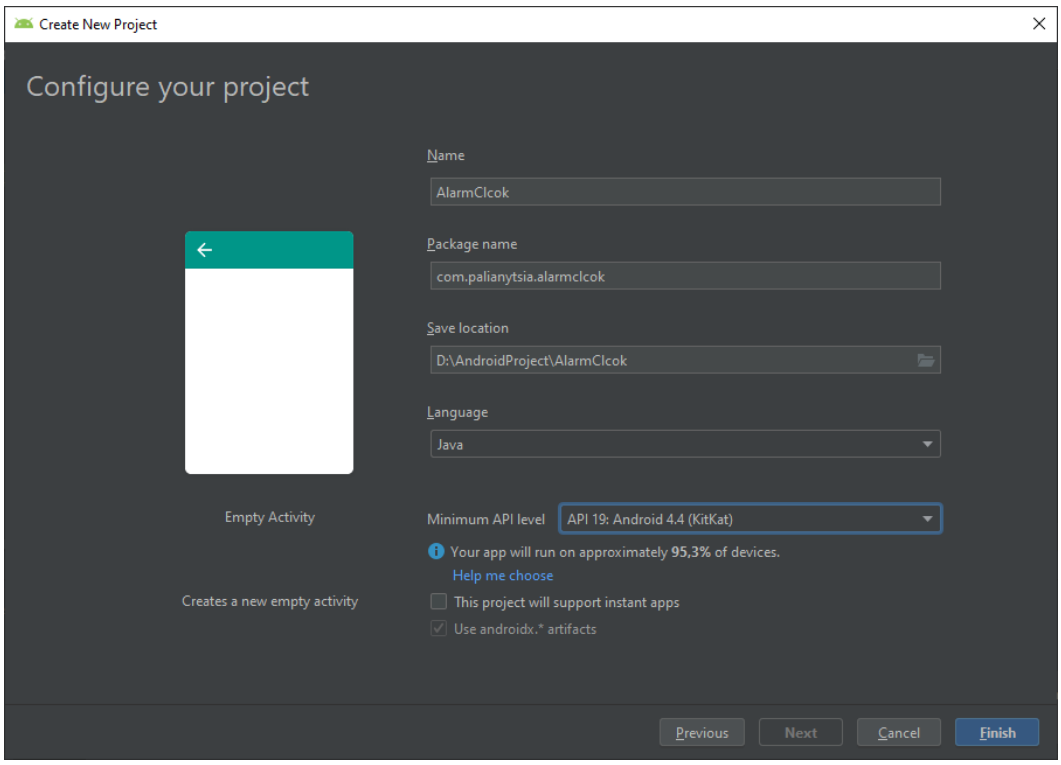


Рисунок 4.1 – Вікно створення нового проекту

З даного рисунка можемо бачити, що проект має назву AlarmClock, ім’я пакету com.palianytsia.alarmclock, мова програмування Java та мінімальна версія API – 19:Android 4.4 (KitKat).

Структура проекту представлена на рис. 4.2.

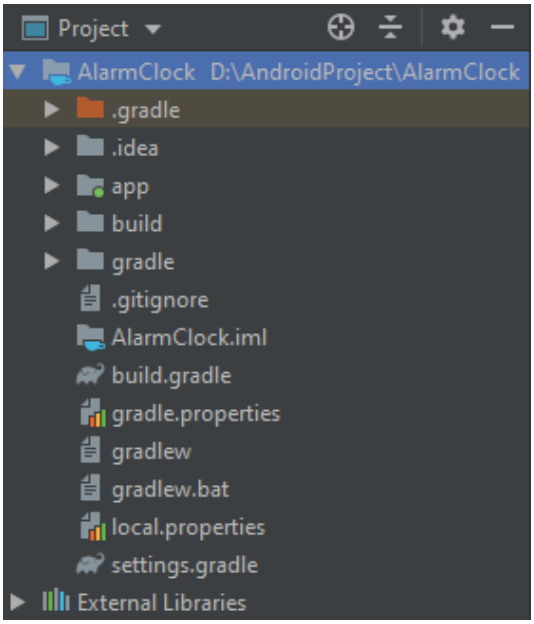


Рисунок 4.2 – Структура проекту

У цьому проекті використовувалися як стандартні бібліотеки так і сторонні бібліотеки, завантажені з веб-ресурсів. Деякі бібліотеки були завантажені з GitHub [22], для прикладу деякі з них:

- PagerSlidingTabStrip [23] (інтерактивний індикатор для навігації між різними сторінками ViewPager).
- Butter Knife [24] (прив'язка поля та методу для Android Views).
- Material Dialogs [25] (призначена для інтерфейсу діалогових вікон).

Дані бібліотеки використовуються в розробленому мобільному додатку і допомагають у реалізації всіх функціональних можливостей.

4.1 Створення бази даних SQLite

БД була розроблена за допомогою вбудованої СУБД SQLite. Основними пакетами для роботи з БД в IDE Android Studio є android.database і android.database.sqlite. На рис. 4.3 – рис. 4.6 представлені таблиці бази даних з їх полями. Розглянемо створені таблиці з їх полями. На рис. 4.3 представлена таблиця DAYS_OF_WEEK, яка містить дні тижня для вибору повтору будильника.

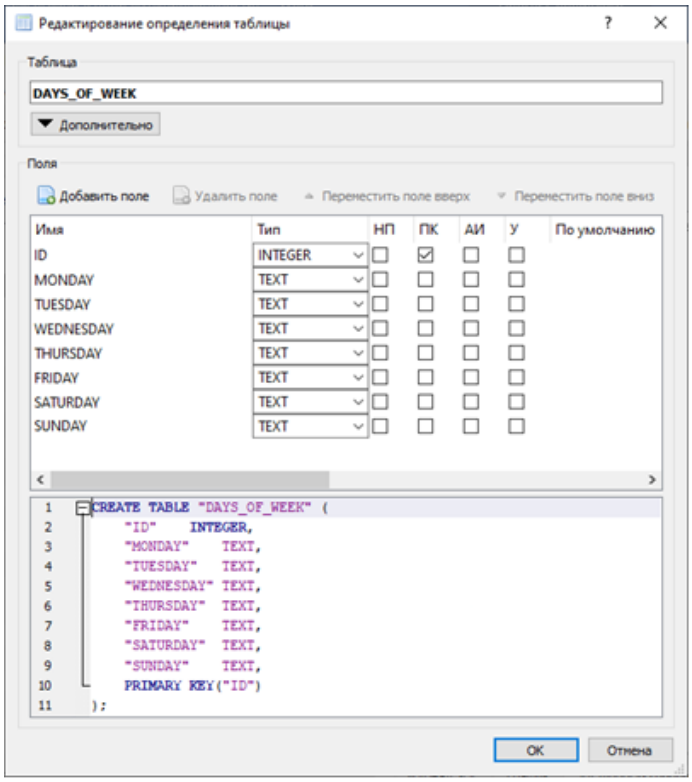


Рисунок 4.3 – Структура таблиці DAYS_OF_WEEK бази даних

На рис. 4.4 зображена таблиця NOTIFICATIONS, яка зберігає інформацію про створений будильник.

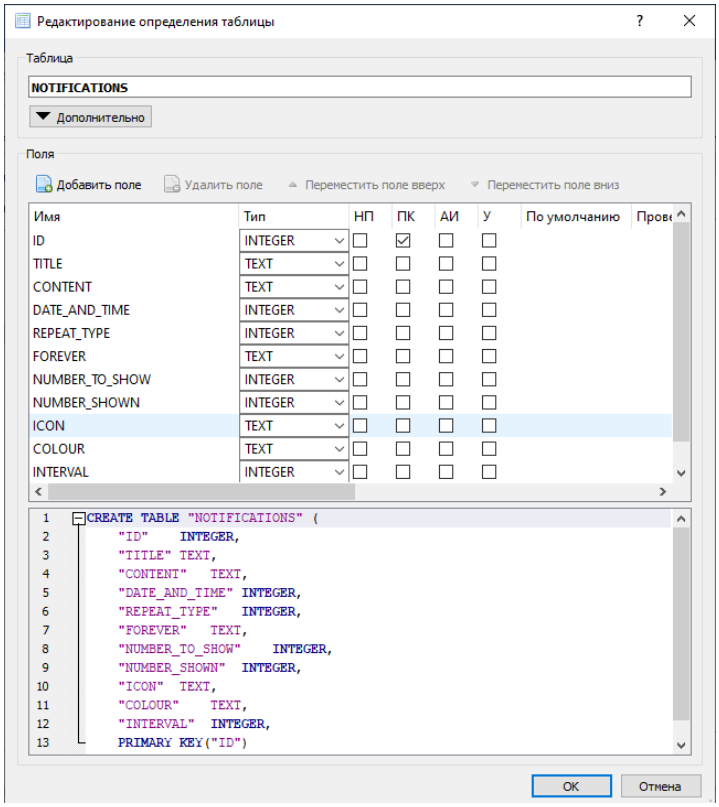


Рисунок 4.4 – Структура таблиці NOTIFICATIONS бази даних

Також створена таблиця ICONS для роботи з іконками для нагадувань, яка зображена на рис. 4.5.

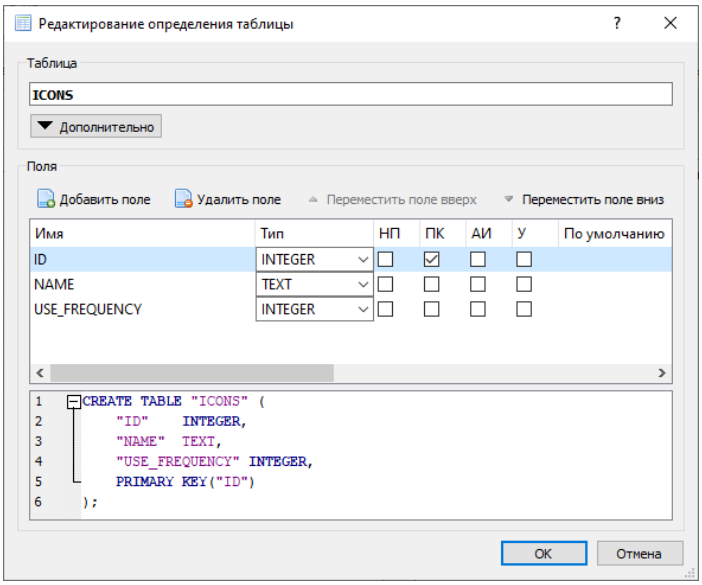


Рисунок 4.5 – Структура таблиці ICONS бази даних

Наостанок створено таблицю PICKER_COLOURS для підбирання кольорів. Таблиця показана нижче на рис. 4.6.

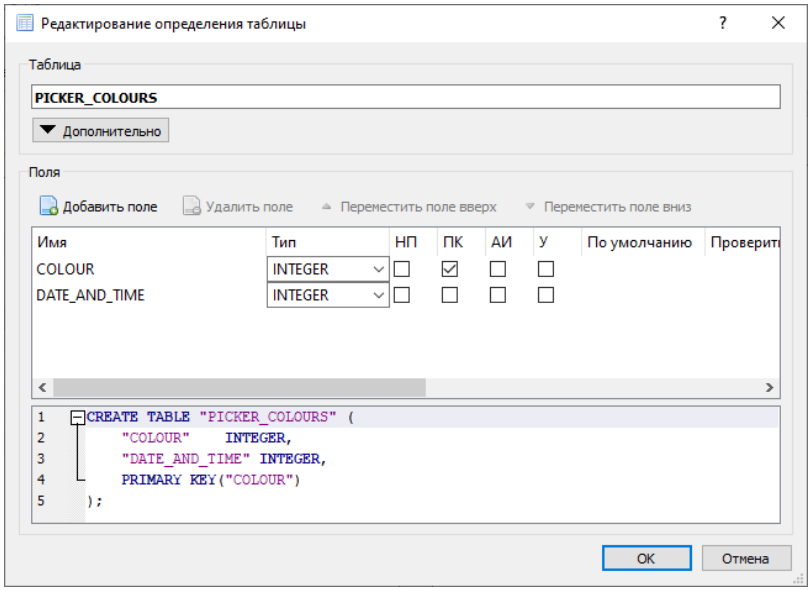


Рисунок 4.6 – Структура таблиці PICKER_COLOURS бази даних

Зв’язок таблиць у базі даних представлений у третьому розділі на рис. 3.6 – ER-діаграма бази даних.

4.2 Розробка класів та методів мобільного додатку

У цьому підрозділі описано розроблені класи та методи. Основні розроблені класи та їх призначення приведено в табл. 4.1.

Таблиця 4.1 – Розроблені класи та їх призначення

Ім’я класу	Призначення класу
AboutActivity	призначений для роботи з вікном про додаток (зворотній зв’язок, відображення діалогових вікон з використаними бібліотеками, детальною інформацією)
CreateEditActivity	клас призначений для створення будильників, задання умов властивостей будильника, збереження нового будильнику тощо
MainActivity	призначений для відображення головного вікна додатку, переключення між активними і завершеними будильниками, відповідає за меню налаштувань
SnoozeDialogActivity	даний клас призначений для налаштування функції відкласти сповіщення
ViewActivity	клас призначений для роботи з activity відображення створеного будильнику, можливості поділитися нагадуванням, позначити спрацьованим будильник, показати зараз
IconsAdapter	призначений для роботи з іконками до будильнику, сортування за частотою вибору, відображенням вікна з вибором іконок
ViewPageAdapter	клас, який відповідає за відображення який будильник активний і вже спрацьований
DatabaseHelper	призначений для роботи з базою даних: підключення, оновлення даних, редагування та видалення будильників

Продовження таблиці 4.1 – Розроблені класи та їх призначення

AdvancedRepeatSelector	даний клас призначений для задання та налаштування повтору будильника
DaysOfWeekSelector	клас, який відповідає за вибір днів неділі, коли повинен спрацювати будильник
IconPicker	призначений для вибору іконки та присвоєнню її будильнику
PreferenceNagTimePicker	призначений для роботи з можливістю відкласти сповіщення
RepeatSelector	клас призначений для роботи з функцією повтору будильнику (кожен час, день, окремі дні тощо)
PreferenceFragment	клас, який налаштовує вибір частоти повтору будильнику
AlarmReceiver	призначений для відображення повідомлення сповіщення
BootReceiver	викликає метод onReceive() і передає об'єкт Intent,
SnoozeReceiver	налаштовує кнопку відложити будильник у сповіщені
AnimationUtil	відповідає за анімацію деяких компонент
DateAndTimeUtil	клас, який працює з датою та часом спрацювання будильнику
NotificationUtil	клас в якому створюється саме сповіщення, прив’язуються його властивості обрані в налаштуваннях
TextFormatUtil	клас, який працює з обробкою інформації

Кожен клас має свої власні методи, які відповідають виконання різних дій. У табл. 4.2 описані методи класу, який працює з базою даних DatabaseHelper.

Таблиця 4.2 – Методи класу DatabaseHelper

Метод	Опис
onCreate	реалізовується логіка створення всіх таблиць, додається база даних іконок, викликається при першому створенні бази даних
onUpgrade	проходить перевірка номеру версії бази даних, викликається при модифікації бази даних
addNotification	призначений для внесення в таблицю БД нового запису будильника
getLastNotificationId	для виведення об'єднання таблиць
getNotificationList	призначений для отримання всього списку будильників
getNotification	призначений для отримання окремого сповіщення
deleteNotification	для видалення запису з таблиці бази даних будильника
getDaysOfWeek	отримує дані вибрані користувачем про день спрацювання будильнику
addDaysOfWeek	перетворює тип Boolean в String
addAllIcons	додає масив іконок для будильників
getIconList	призначений для отримання списку іконок
getColoursArray	отримуємо масив кольорів
addAllColours	призначений для додавання всіх кольорів для персоналізації будильника

Далі опишемо інші основні методи та їх призначення, які використовувались у різних класах. Методи представлені в табл. 4.3.

Таблиця 4.3 – Основні методи інших класів

Метод	Клас	Призначення
launchEmail	AboutActivity	Налаштовує функцію відправки повідомлень на пошту
showContributorsDialog	AboutActivity	Відображає діалогове вікно з детальною інформацією

Продовження таблиці 4.3 – Основні методи інших класів

onOptionsItemSelected	AboutActivity	Метод вибору пунктів у activity інформації про додаток
timePicker	CreateEditActivity	Призначений для відображення годинника і задання часу будильника
datePicker	CreateEditActivity	Призначений для відображення календаря і задання дати будильника
iconSelector	CreateEditActivity	Відкриває вікно вибору іконок
saveNotification	CreateEditActivity	Для збереження будильнику
onCreateOptionsMenu	CreateEditActivity	Ініціює створення меню
onOptionsItemSelected	PreferenceActivity	Налаштовує вибір пункту меню налаштувань
setUpMinutePicker	SnoozeDialogActivity	Призначений для налаштування вибору хвилин
setUpHourPicker	SnoozeDialogActivity	Призначений для налаштування вибору годин
confirmDelete	ViewActivity	Викликає діалогове вікно з підтвердженням на видалення будильнику
actionShowNow	ViewActivity	Демонструє сповіщення зараз
actionDelete	ViewActivity	Працює з видаленням будильнику
updateReminder	ViewActivity	Призначений для оновлення будильнику в базі даних

Інші створені класи та їх методи мобільного додатку приведені у лістингу Додатку Б. Тепер розглянемо кожне вікно мобільного додатку та опишемо, які функціональні можливості мають. Запустивши мобільний додаток відразу відкривається вікно з активними будильниками, яке показано на рис. 4.7.

Рисунок 4.7 – Вікно відображення активних будильників

Наразі активних будильників немає. Натиснувши на кнопку закресленого дзвінка – перейдемо на вкладку будильників, які вже спрацювали. Дане вікно продемонстроване на рис. 4.8.

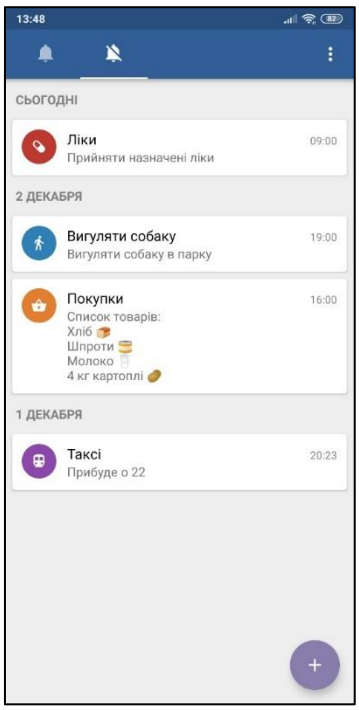


Рисунок 4.8 – Вікно відображення спрацьованих будильників

З рис. 4.8 бачимо будильник з їх описом, які раніше вже спрацювали. Будильники сортуються по даті та часу спрацювання – від новіших до більш давніх. Коли будильник спрацьовує, то він автоматично переноситься на дану вкладку з попередньої. Натиснувши на кнопку трьох точок відображається меню, де можна обрати «налаштування» чи «про додаток». На рис. 4.9 було обрано «про додаток» і показується скріншот цього вікна.

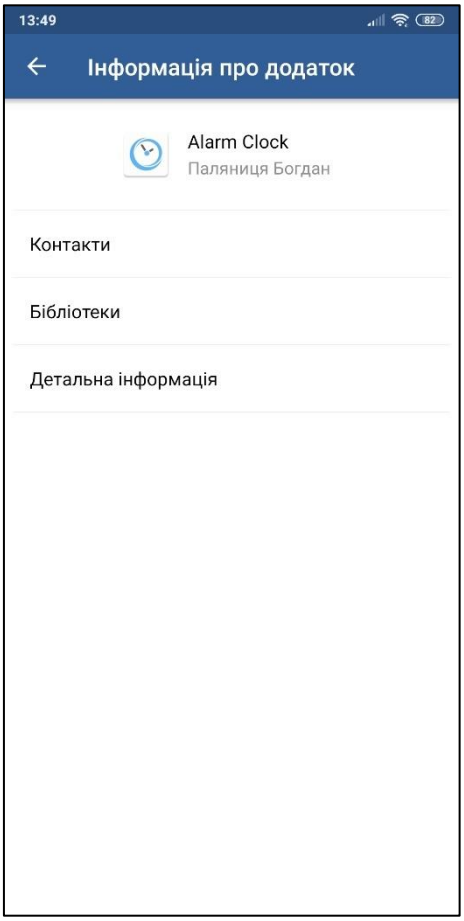


Рисунок 4.9 – Вікно з інформацією про додаток

Це вікно показує логотип та ініціали розробника мобільного додатку, а також кнопки «Контакти», «Детальна інформація» та «Бібліотеки», де наведені деякі використані при розробці бібліотеки, їх розробник та посилання на них у веб-ресурсах. На рис. 4.10 та рис 4.11 продемонстровано натискання на кнопку «Контакти» та вікно детальної інформації

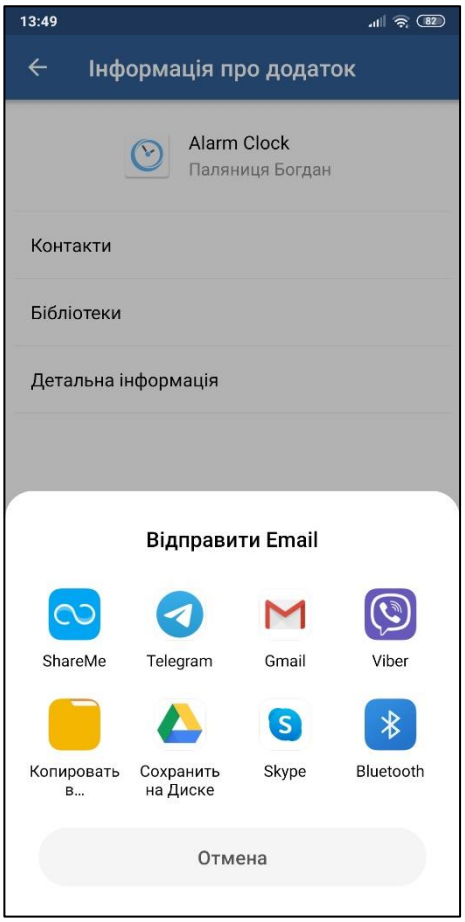


Рисунок 4.10 – Зворотній зв’язок

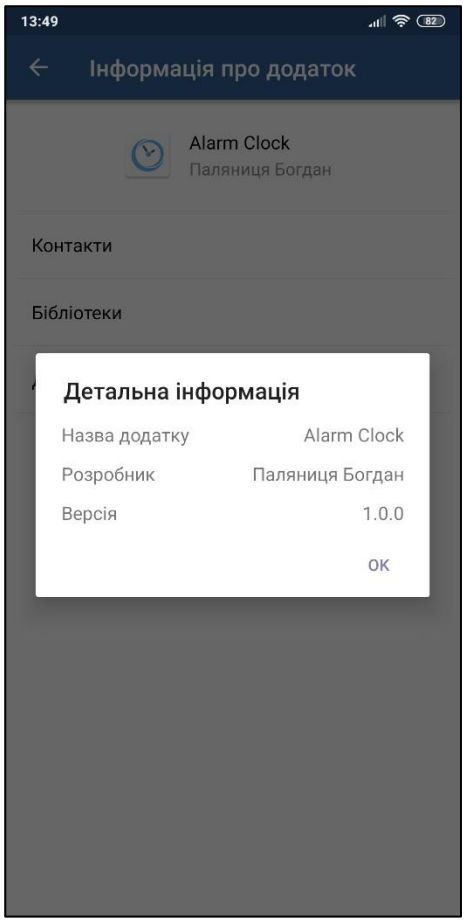


Рисунок 4.11 – Вікно з детальною інформацією

На рис. 4.10 натиснувши на Gmail можемо зв’язатися з розробником. На рис. 4.11 бачимо вікно з інформацією про автора, назву мобільного додатку та версію. Повертаючись до рис. 4.8, якщо буде натиснута кнопка в меню налаштування, то відкриється вікно з всіма налаштуваннями для будильнику. Зокрема на рис. 4.12 можемо бачити наступні налаштування:

- вібрація при сигналі;
- мелодія;
- LED-індикатор;
- стійке сповіщення;
- вимкнути;
- відкласти;

– повтор будильника;
– частота повтору будильника.

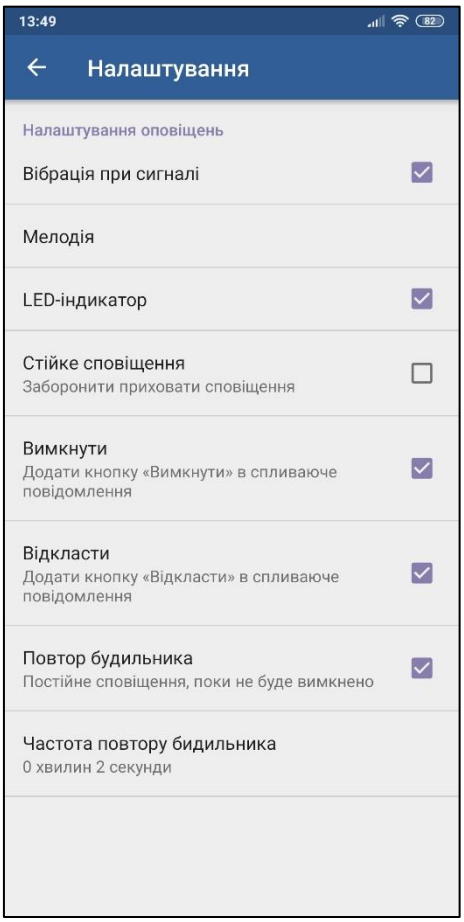


Рисунок 4.12 – Вікно з налаштуваннями будильника

До кожного варіанту налаштування є коротке пояснення. Включається налаштування натиснувши та поставивши галочку, окрім мелодії та частоти повтору будильника (відкривається вікно вибору мелодії та частоти повтору). Скріншоти вибору мелодії та частоти повтору будильника показані на рис. 4.13 та рис. 4.14 відповідно.

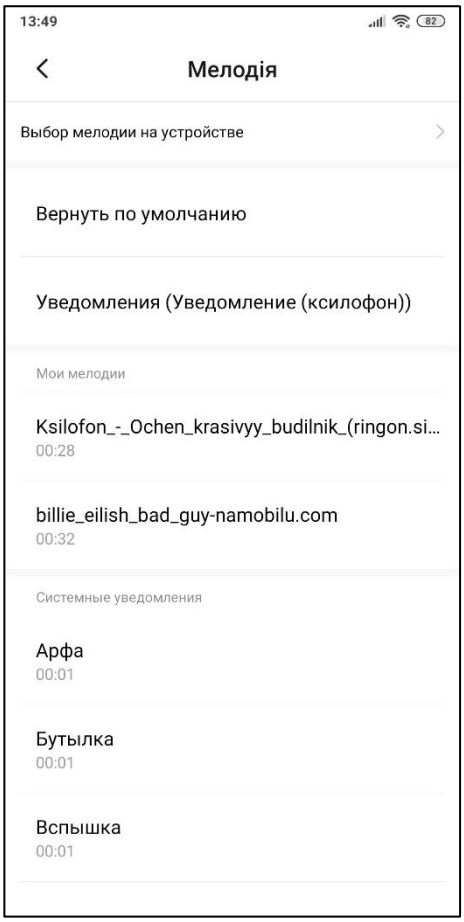


Рисунок 4.13 – Вікно вибору мелодії

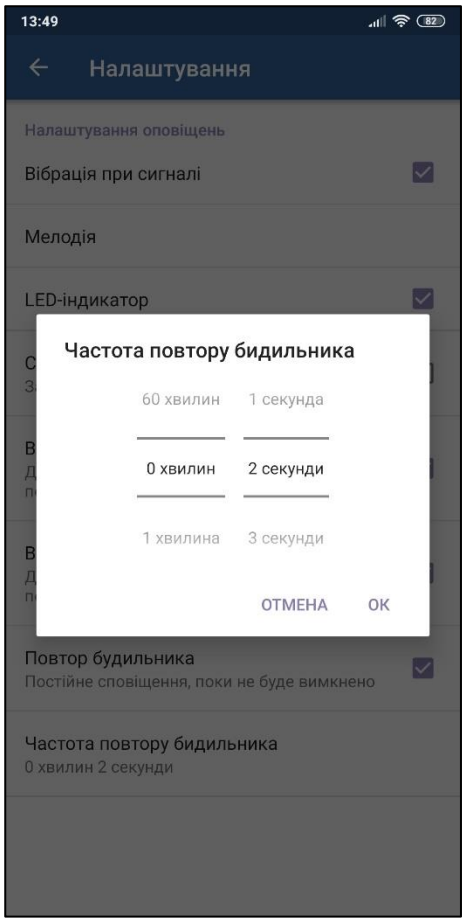


Рисунок 4.14 – Вікно задання частоти

Мелодію можна обрати як стандартну так свою завантажену. Частота повтору задається прокручуванням хвилин та секунд.

Натиснувши на головному вікні кнопку плюса – відкриється вікно із створенням нового будильнику. Заголовок будильнику є обов’язковим полем для заповнення. Всі інші можна не заповнювати та не змінювати, вони залишаться за замовчуванням. На рис. 4.15 показано дане вікно створення нового будильнику.

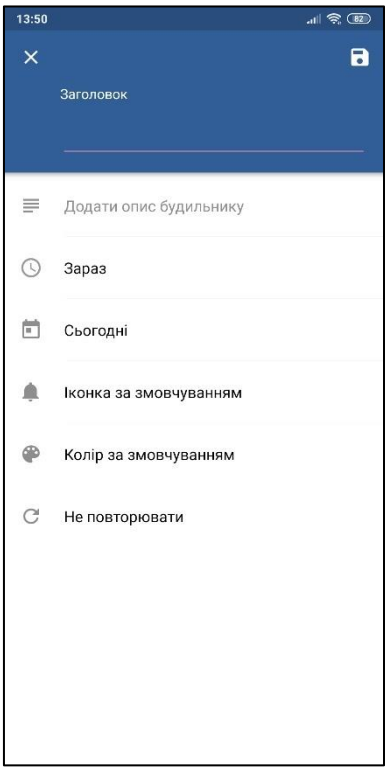


Рисунок 4.15 – Вікно створення будильнику

На рис. 4.16 – рис. 4.19 показані вікна налаштування створеного будильнику.

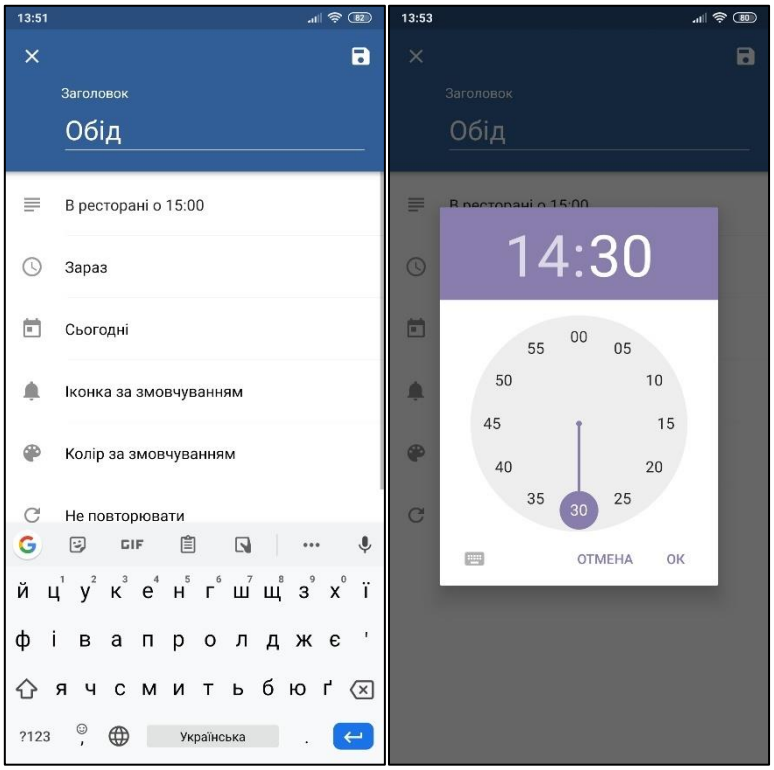


Рисунок 4.16 – Додавання опису будильнику та вибір часу спрацювання

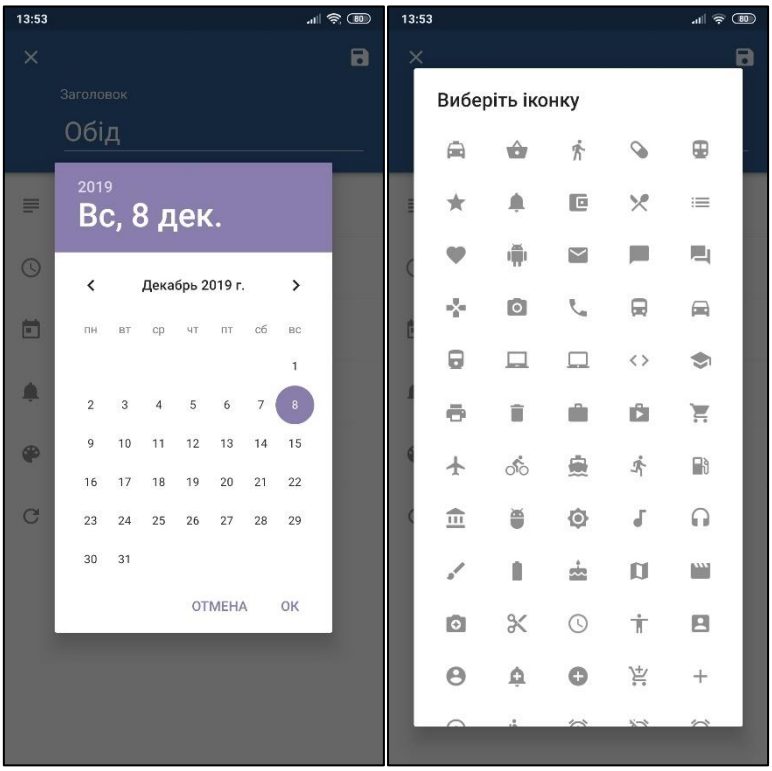


Рисунок 4.17 – Вікно вибору дати та іконки будильника

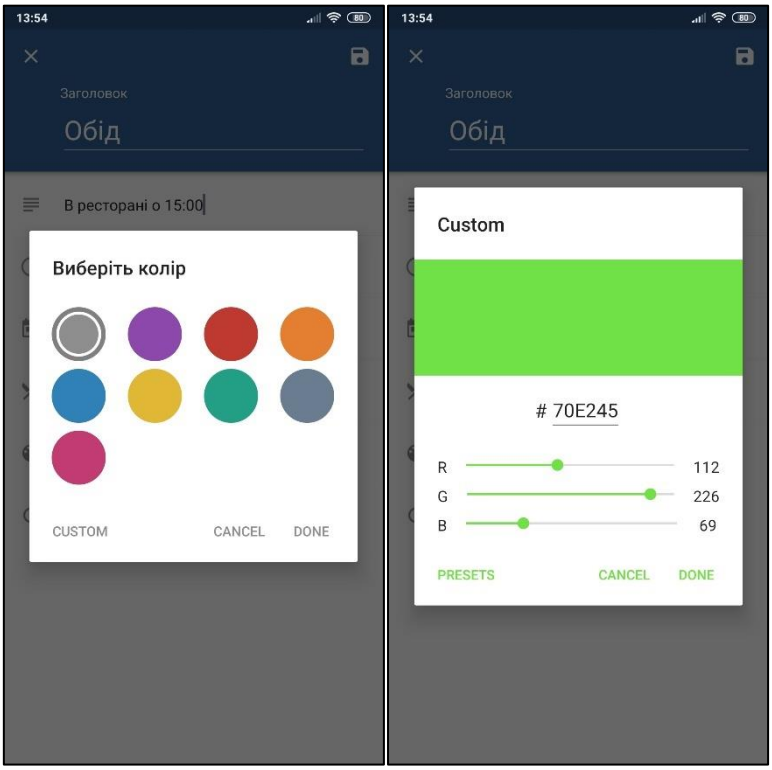


Рисунок 4.18 – Вікна вибору стандартного кольору та його налаштування

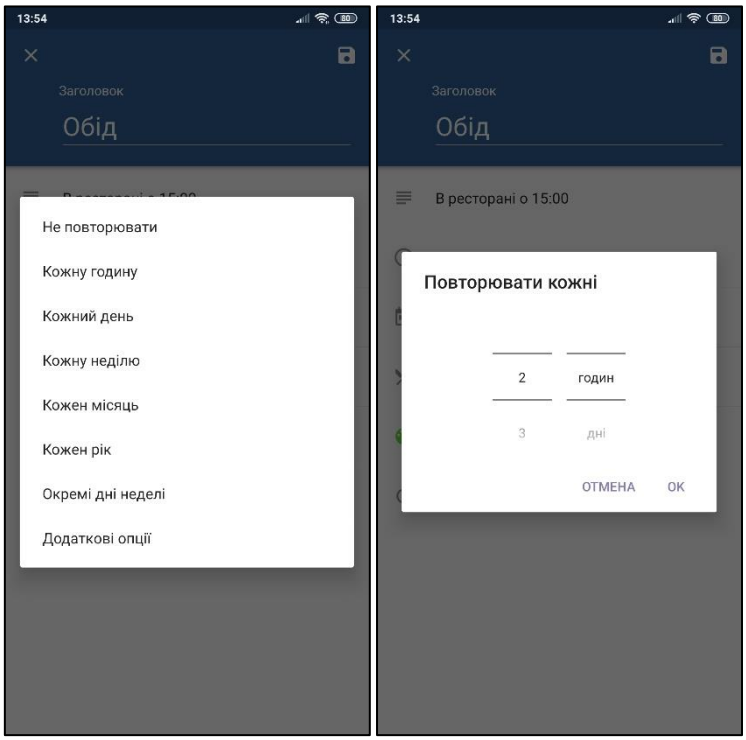


Рисунок 4.19 – Вікно вибору повторення та його налаштування

Після задання всіх необхідних полів натискаємо на кнопку збереження і щойно створений будильник додається до БД та відображається в активних будильниках. Результат показано на рис. 4.20.

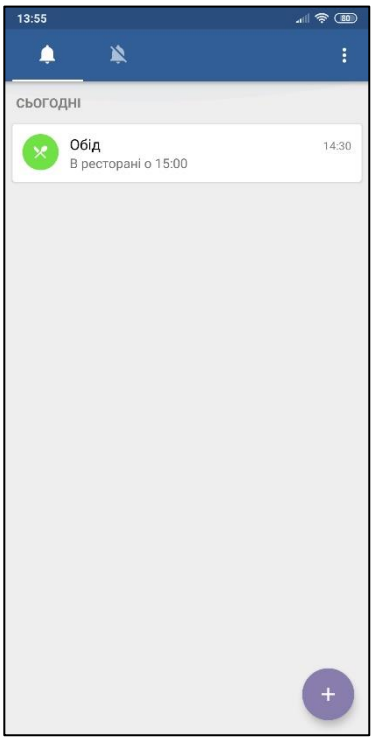


Рисунок 4.20 – Створений будильник

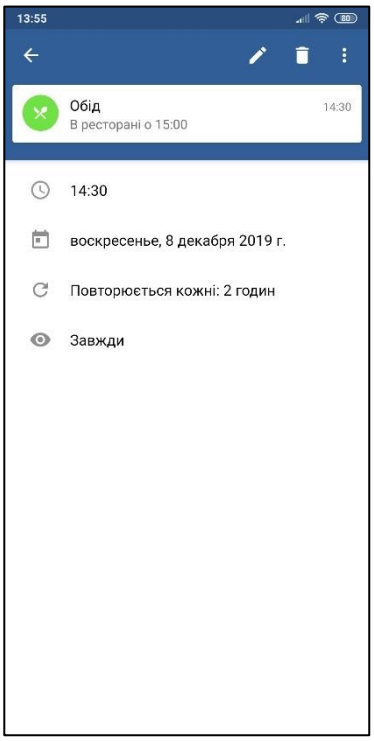


Рисунок 4.21 – Вікно перегляду

На рис. 4.21 показано вікно перегляду будильника. Воно дозволяє натиснувши на відповідну кнопку редагувати та видалити будильник, а також повернутися назад та відкрити додаткове меню, що містить пункти поділитися, показати зараз та позначити як спрацьований. Рис. 4.22 та рис. 4.23 демонструє відправку змісту будильника на пошту. Також є змога відправити сповіщення в месенджерах.

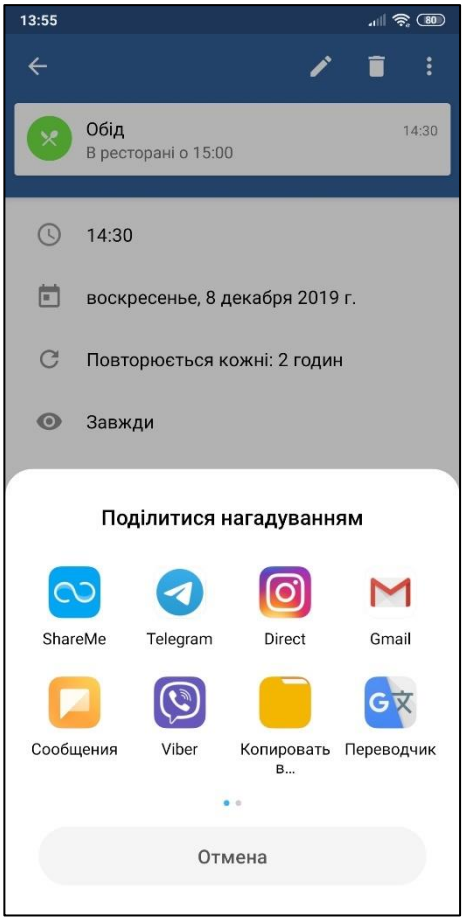


Рисунок 4.22 – Вибір способу відправки сповіщення

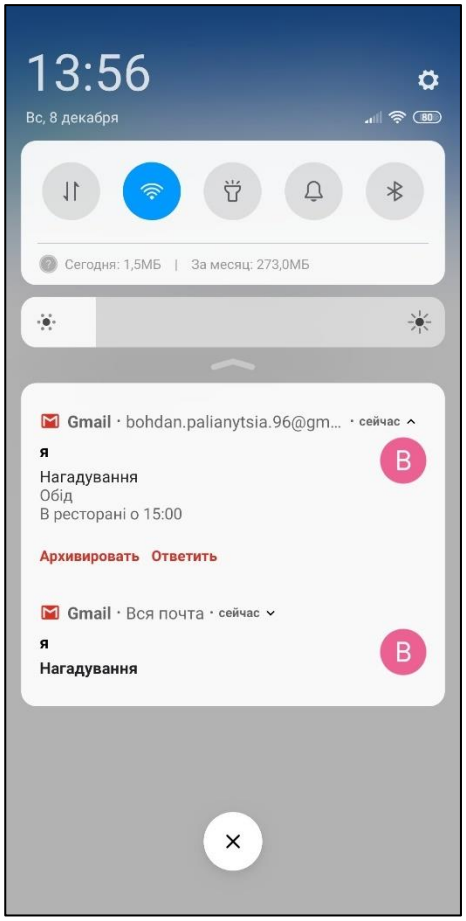


Рисунок 4.23 – Результат відправки на пошту

На рис. 4.24 та рис. 4.25 показано повідомлення будильника у призначений час та дату.

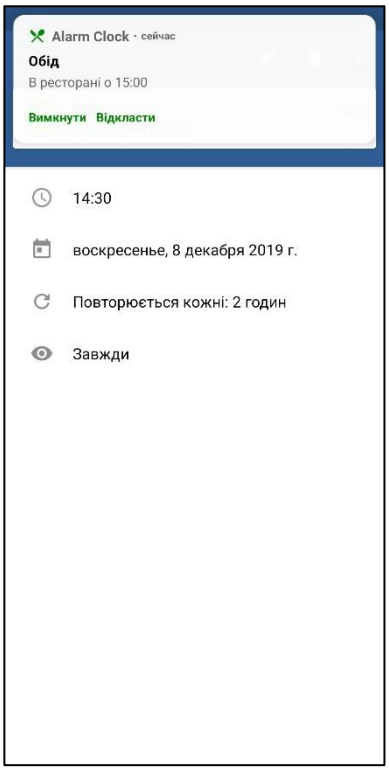


Рисунок 4.24 – Спливаюче сповіщення

Рисунок 4.25 – Сповіщення на заблокованому екрані

Для того, щоб не можна було замахом сповіщення виключити його, у налаштуваннях потрібно включити функцію стійке сповіщення. У цьому випадку його можна відключити лише зайшовши у мобільний додаток.

4.3 Тестування мобільного додатку

Тестування мобільних додатків – це такий процес, за допомогою якого програмне забезпечення, розроблене для мобільних пристроїв, тестується на його функціональність, зручність використання та послідовність [26]. Тестування допомагає:

- показати замовнику та власне самому розробнику, що програмний ПП відповідає поставленим вимогам та правильно працюючий додаток;
- знайти такі ситуації, в яких поведінка ПП є некоректною, небажаною або не відповідає специфікації.

У даному магістерському проекті для розробленого мобільного додатку – застосовувалось ручне тестування ПП. В табл. 4.4 продемонстровані базові Test Cases з необхідними кроками, очікуваним результатом та статусом роботи (пройдено чи провалено). Test Cases – це сукупність умов чи змінних, за яких тестер визначатиме, чи відповідає система, яка тестується, вимогам чи працює правильно ПП [27]. Початковими для тестування діями для даного мобільного додатку є його запуск.

Таблиця 4.4 – Test Cases для програмного продукту

№	Test Case	Послідовність кроків	Очікуваний результат	Статус (пройдено/ провалено)
1	Перевірка можливості задання для будильнику дати / часу, який вже минув	1. Відкрити вікно для створення нового будильнику 2. Задати дату / час спрацювання будильнику, який вже минув	1. Повідомлення про помилку 2. Нова спроба задати дату / час	Пройдено
2	Перевірка функції появи звукового сигналу та сповіщення на заблокованому екрані	1. Задати будильник з його властивостями 2. Заблокувати екран і дочекатися заданий час його спрацювання	1. У заданий час заблокований екран стає активним 2. Поява звукового сигналу із змістом сповіщення	Пройдено
3	Перевірка можливості залишити поле заголовку при створенні будильника пустим	1. Відкрити вікно для створення нового будильнику 2. Залишити поле заголовку будильника пустим	1. Повідомлення про помилку 2. Нова спроба задати заголовок будильнику	Пройдено

Було проведено перевірку реалізації усього функціоналу мобільного додатку. Так для більшого розуміння було створено тест-кейси, які наведені у табл. 4.4 для перевірки функціоналу мобільного додатку. Дані тест-кейси пройдені успішно, а це значить, що розробка відповідає функціональним вимогам.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було вирішено актуальну проблему – створення мобільного додатку «Розумний будильник» для полегшення процесу планування розпорядку дня. У ході роботи проведено аналіз сучасного стану застосування інформаційних технологій для вирішення поставлених задач та огляд літератури, сформульовано мету та задачі проекту, проаналізовано додатки-аналоги.

Також проаналізовані основні процеси будильників та проведений їх аналіз, який показав необхідність створення мобільного додатку. Після цього були визначені цілі, поставлені задачі, сплановано роботу, прораховані ризики та обрані методи проектування нового мобільного додатку.

Після формування функціональних вимог було проаналізовано методи створення мобільних додатків та обраний інструментарій для виконання поставлених задач. Були обрані технології Java, IDE Android Studio як сучасні та продуктивні засоби для реалізації мобільних додатків. У результаті огляду предметної області був розроблений мобільний додаток. Додаток було протестовано, що підтвердило його працездатність.

За час розробки мобільного додатку у великому обсязі використовувалися матеріали з програмування, що допомогло закріпленню напрацьованих навичок і нових вмінь у цих областях знань.

Результати кваліфікаційної роботи будуть апробовані в науковій конференції IMA 2020. Детально з тезисами магістерської роботи можна ознайомитися на рис. В.1 у додатку В.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dr. T. Sreenivasulu, Sanif Himani, Rishika Goud, Sharan Kumar IoT based Smart Alarm Clock. International Journal of Innovations & Advancement in Computer Science. 2018. ISSN 2347 – 8616. С. 68–71.
2. About SQLite: веб-сайт. URL: https://www.sqlite.org/about.html (дата звернення: 21.11.2019).
3. Kareen and Miranda Alarm Clock Evolution : веб-сайт. URL: https://www.sutori.com/story/alarm-clock-evolution--aTe4Wgds8B9UhYEN9srCX8dT (дата звернення: 09.12.2019).
4. Rosie Osmun The Role of Alarm Clocks in Getting Better Sleep : веб-сайт. URL: https://www.huffpost.com/entry/the-role-of-alarm-clocks-in-getting-better-sleep_b_9676870 (дата звернення: 09.12.2019).
5. Google Play Wikipedia: веб-сайт. URL: https://en.wikipedia.org/wiki/Google_Play (дата звернення: 19.11.2019).
6. Природный Будильник. Google Play: веб-сайт. URL: https://play.google.com/store/apps/details?id=ddidev94.NatureAlarmClock&hl=fr (дата звернення: 21.11.2019).
7. Будильник - Умный будильник. Google Play: веб-сайт. URL: https://play.google.com/store/apps/details?id=com.sma.smartalarm (дата звернення: 21.11.2019).
8. Sleepytie - умный будильник. Google Play: веб-сайт. URL: https://play.google.com/store/apps/details?id=com.mabo.sleepyti (дата звернення: 21.11.2019).
9. Mobile Operating System Market Share Worldwide Oct 2017 - Oct 2019 StatCounter Global Stats: веб-сайт. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата звернення: 22.11.2019).

10. Mobile Operating System Market Share Ukraine Oct 2017 - Oct 2019 StatCounter Global Stats: веб-сайт. URL: https://gs.statcounter.com/os-market-share/mobile/ukraine (дата звернення: 22.11.2019).
11. Mobile Android Version Market Share World Oct 2017 - Oct 2019 StatCounter Global Stats: веб-сайт. URL: https://gs.statcounter.com/os-version-market-share/android/mobile/world (дата звернення: 23.11.2019).
12. AJ Alt Kotlin vs Java: Compilation speed Sep 9, 2016: веб-сайт. URL: https://medium.com/keepsafe-engineering/kotlin-vs-java-compilation-speed-e6c174b39b5d (дата звернення: 07.12.2019).
13. TIOBE Index for December 2019: веб-сайт. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 09.12.2019).
14. Результаты теста скорости языков C++, Java, PHP, Ocaml, Perl, Python, Ruby…: веб-сайт. URL: https://www.linux.org.ru/forum/development/4147618 (дата звернення: 09.12.2019).
15. 13 Best Mobile IDEs for Android [Fall 2019 Update] Intellectsoft: веб-сайт. URL: https://www.intellectsoft.net/blog/5-of-the-best-mobile-ides-for-android (дата звернення: 23.11.2019).
16. IDEF0 (Integrated Definition for Function Modeling) Service Quality Division The Global Voice of Quality: веб-сайт. URL: http://asqservicequality.org/glossary/idef0-integrated-definition-for-function-modeling/ (дата звернення: 24.11.2019).
17. UML Use Case Diagram LucidChart: веб-сайт. URL: https://www.lucidchart.com/pages/uml-use-case-diagram (дата звернення: 24.11.2019).
18. Создаем быстрый прототип мобильного приложения Habr: веб-сайт. URL: https://habr.com/ru/post/189524/ (дата звернення: 25.11.2019).
19. Быстрое прототипирование Wikipedia: веб-сайт. URL: https://ru.wikipedia.org/wiki/Быстрое_прототипирование (дата звернення: 25.11.2019).

20. Спецификации дизайна значков для Google Play: веб-сайт. URL: https://developer.android.com/google-play/resources/icon-design-specifications?hl=ru (дата звернення: 26.11.2019).
21. Adobe Illustrator Wikipedia: веб-сайт. URL: https://en.wikipedia.org/wiki/Adobe_Illustrator#cite_ref-_6feaca787ce222de_1-0 (дата звернення: 26.11.2019).
22. GitHub: веб-сайт. URL: https://github.com/ (дата звернення: 27.11.2019).
23. Andreas Stuetz PagerSlidingTabStrip: веб-сайт. URL: https://github.com/jpardogo/pagerslidingtabstrip (дата звернення: 03.10.2019).
24. Jake Wharton Butter Knife: веб-сайт. URL: http://jakewharton.github.io/butterknife/ (дата звернення: 07.10.2019).
25. Aidan Follestad Material Dialogs: веб-сайт. URL: https://github.com/afollestad/material-dialogs (дата звернення: 20.10.2019).
26. Mobile application testing Wikipedia: веб-сайт. URL: https://en.wikipedia.org/wiki/Mobile_application_testing (дата звернення: 29.11.2019).
27. Test Case Software testing fundamentals: веб-сайт. URL: http://softwaretestingfundamentals.com/test-case/ (дата звернення: 3.12.2019).
28. What is Work Breakdown Structure? – [Електронний ресурс]. – Режим доступу: https://www.visual-paradigm.com/guide/project-management/what-is-work-breakdown-structure/ (дата звернення: 10.11.2019).
29. Ronda Bowen What is an Organizational Breakdown Structure (OBS) : веб-сайт. URL: https://www.brighthubpm.com/project-planning/11639-how-and-when-to-use-an-obs/ (дата звернення: 12.11.2019).
30. Jeremy Bradley Organizational Structure: веб-сайт. URL: https://smallbusiness.chron.com/organizational-breakdown-structure-obs-72523.html (дата звернення: 12.11.2019).
31. Gantt chart: веб-сайт. URL: https://www.apm.org.uk/resources/find-a-resource/gantt-chart/ (дата звернення: 13.11.2019).

ДОДАТОК А

ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART

Продуктом кваліфікаційної роботи має бути готовий до використання мобільний додаток «Розумний будильник» із всіма необхідними функціями. Ідентифікація мети в рамках проекту методом SMART представлена в табл. А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Робота над проектом передбачає створення мобільного додатку, аналогів якому наразі досить мало, для виконання поставлених задач, тобто реалізація унікальної ідеї.
Measurable (вимірювана)	Підтверджуючим фактом, що то мета методом SMART виявилася досягнута є сто процентне виконання всіх поставлених вимог
Achievable (досяжна)	Кожна програма-аналог має свої переваги та недоліки. У нашому випадку, маючи необхідні знання та навички буде розроблений мобільний додаток, який буде володіти кращими функціоналом та властивостями
Relevant (актуальна)	Реалізація даного проекту можлива з урахуванням поточної кількості людей, при чому досягти результату можна за допомогою програмного забезпечення (Android Studio та мова програмування Java) і наявних ресурсів
Time-bound (обмежена у часі)	Терміни проведення розробки проекту: вересень 2019 року – грудень 2019 року

Проаналізувавши дані з табл. А.1, можна визначити кінцеву мету проекту. Отже мета проекту методом SMART полягає у розробці мобільного додатку «Розумний будильник» терміном до 20 грудня 2019 року, який би виконував поставлені задачі у сто процентному обсязі.

Планування змісту структури робіт IT-проекту (WBS)

Work Breakdown Structure (WBS) - це орієнтована на результати ієрархічна декомпозиція роботи, яка повинна бути виконана виконавцем проекту для досягнення цілей проекту та створення необхідних результатів. WBS є основою ефективного планування, розробки, контролю, моніторингу та звітності. Вся робота, що міститься в WBS, повинна бути визначена, оцінена та запланована [28].

WBS розроблена для встановлення єдиного розуміння обсягу проекту. Це ієрархічний опис роботи, яку необхідно виконати для завершення результатів проекту. Кожен низхідний рівень у WBS являє все більш детальний опис результатів проекту. Перші два рівні WBS визначають набір запланованих результатів, які в сукупності та виключно представляють 100% обсягу проекту.

Добре розроблена WBS описує заплановані результати замість запланованих дій. Результати - це бажані цілі проекту, такі як продукт чи послуга, і їх можна точно передбачити. Добре розроблена WBS повинна мати такі характеристики:

- визначена - може бути описана та легко зрозуміла учасниками проекту;
- керована - змістовна одиниця роботи, де конкретна відповідальність та повноваження можуть бути покладені на відповідальну особу;
- приблизна - тривалість може бути оцінена за час, необхідний для завершення, а вартість може бути оцінена в ресурсах, необхідних для їх завершення;
- незалежна - мінімальний інтерфейс або залежність від інших елементі;
- вимірювальна - може використовуватися для вимірювання прогресу; має дати початку та завершення та вимірювані проміжні етапи;
- адаптована - досить гнучка, щоб додавання / усунення обсягу роботи можна було легко розмістити в рамках WBS.

Була побудована діаграма WBS, яка представлена на рис. А.1.

Рисунок А.1 – WBS діаграма розробки мобільного додатку

Організаційна структура проекту

Organizational Breakdown Structure (OBS) можна визначити як модель, структуровану таким чином, щоб визначити, які працівники будуть відповідати за конкретні частини проекту. OBS відображає організаційні відносини, а потім використовує їх для віднесення роботи до ресурсів у проекті.

Як і WBS, OBS дозволяє розбити складні проекти, забезпечуючи більш організоване представлення роботи, яку слід завершити. У той час як WBS використовується для визначення проекту на ранніх етапах його циклу, OBS забезпечує організаційну структуру проекту по мірі його завершення. Ієрархічний характер OBS дозволяє зазначити обов'язки всіх членів проекту. Організаційна структура розподілу використовується в складних проектах та спільно з WBS [29].

OBS працює добре в поєднанні з графіком робочого процесу. Це документ, який детально описує, що повинна робити кожна особа, перелічена в OBS, для виконання цілей проекту. В ідеалі люди покладаються на завдання, які відповідають їх інтересам та їх сильним сторонам. Це виховує почуття власності у проекті та передбачає, що люди візьмуть на себе відповідальність за розробку та виконання своїх завдань. У менших організаціях учасники проекту можуть бути закликані виконувати декілька ролей. У цих випадках OBS та графік робочого процесу є ще важливішими для того, щоб утримати людей на меті, допомагати їм ставити цілі та повідомляти деталі щодо їхніх прямих зобов'язань перед проектом [30]. Побудована діаграма OBS представлена на рис. А.2.

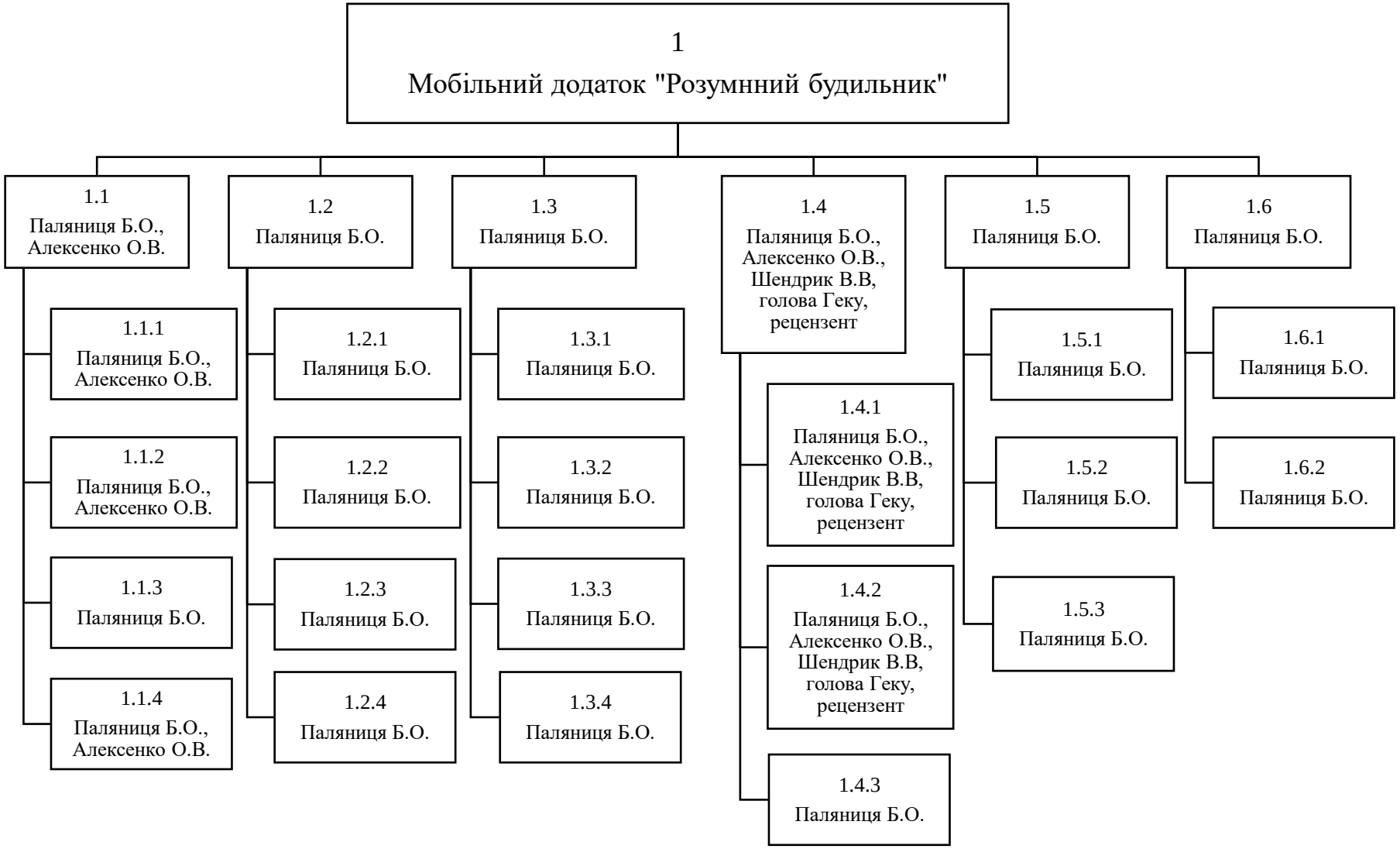


Рисунок А.2 – OBS діаграма розробки мобільного додатку

Побудова календарного графіка виконання ІТ-проекту

Діаграма Ганта - це інструмент управління проектами, що допомагає планувати проекти всіх розмірів, хоча вони особливо корисні для спрощення складних проектів. Терміни та завдання управління проектами перетворюються на горизонтальну діаграму із зазначенням дати початку та закінчення, а також залежностей, планування та строків, у тому числі, скільки завдань виконано на етапі та хто є виконавець завдання. Це корисно для відстеження завдань, коли існує велика команда та декілька зацікавлених сторін, коли область застосування змінюється [31].

За допомогою діаграми Ганта можна побачити:

- візуальний показ всього проекту;
- терміни виконання всіх завдань;
- зв’язки та залежності між різними видами діяльності;
- етапи проекту.

Рішення для управління проектами, що інтегрують діаграми Ганта, надають менеджерам видимість навантаження в команді, а також поточну та майбутню доступність, що дозволяє більш точно планувати. За допомогою програми Microsoft Project Professional побудовано діаграму Ганта (рис. А.3)

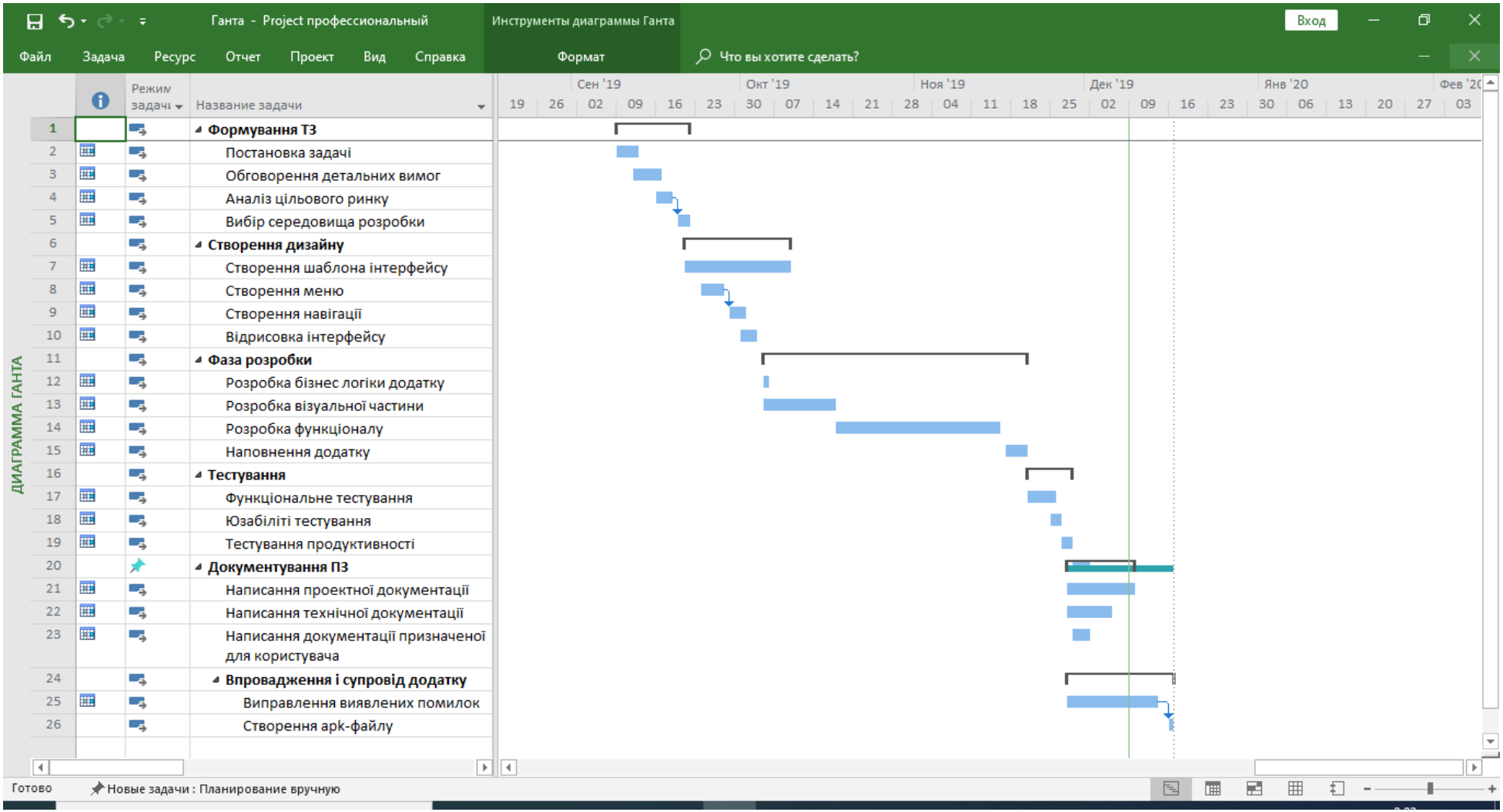


Рисунок А.3 – Діаграма Ганта розробки мобільного додатку

Управління ризиками

Вивчення ризиків, з якими може зустрітися проект із розробки мобільного додатку, показав, що можна виділити кілька десятків, але до основних ризиків розробки мобільного додатку і проекту загалом можна віднести наступні:

- збільшення вимог чи зміна цілей у ході реалізації проекту;
- низька продуктивність;
- внутрішні складнощі календарного планування;
- збільшення навантаження під час реалізації проекту.

Повний список виділених у проекті ризиків представлений у табл. А.2. Також там наведені оцінки вірогідності їх настання та очікуваний їх вплив на проект.

Таблиця А.2 – Вірогідність виникнення і величина ризику

№	**Ризики**	**Вірогідність виникнення**	**Втрати**
R1	Відсутність досвіду та необхідних навичок	2	2
R2	Збільшення навантаження під час реалізації проекту	3	2
R3	Низька продуктивність	3	4
R4	Проблеми з апаратним чи програмним забезпеченням комп’ютера	2	3
R5	Внутрішні складнощі календарного планування	2	3
R6	Збільшення вимог чи зміна цілей у ході реалізації проекту	4	4
R7	Перевищення виділеного на проект бюджету	2	4
R8	Ризик управління проектом	4	3
R9	Повна чи часткова втрата даних	2	5

На рис. А.4 представлена матриця «Вірогідність – Втрати»

Вірогідність							
	-	**5**	5	10	15	20	25
	R6, R8	**4**	4	8	12	16	20
	R2, R3	**3**	3	6	9	12	15
	R1, R4, R5, R7, R9	**2**	2	4	6	8	10
	-	**1**	1	2	3	4	5
			1	**2**	**3**	**4**	**5**
			-	R1 R2	R4 R5 R8	R3 R6 R7	R9

Втрати

Рисунок А.4 – Матриця «Вірогідність – Втрати»

Аналізуючи ризики за ймовірністю їх виникнення, можемо їх розділити на:

- ігноровані:
    1) відсутні;
- незначні:
    1) проблеми з апаратним чи програмним забезпеченням комп’ютера;
    2) відсутність досвіду та необхідних навичок;
    3) внутрішні складнощі календарного планування;
    4) перевищення виділеного на проект бюджету;
    5) повна чи часткова втрата даних;
- помірні:
    1) збільшення навантаження під час реалізації проекту;
    2) низька продуктивність;
- істотні:

1) збільшення вимог чи зміна цілей у ході реалізації проекту;
2) ризик управління проектом;

- критичні:
  1) відсутні.

Також було класифіковано ризики за рівнем впливу:

- прийнятні:
  1) відсутні;
- виправдані:
  1) відсутність досвіду та необхідних навичок;
  2) збільшення навантаження під час реалізації проекту;
  3) низька продуктивність;
  4) проблеми з апаратним чи програмним забезпеченням комп’ютера;
  5) внутрішні складнощі календарного планування;
  6) перевищення виділеного на проект бюджету;
  7) ризик управління проектом;
  8) повна чи часткова втрата даних;
- неприпустимі:

  збільшення вимог чи зміна цілей у ході реалізації проекту.

ДОДАТОК Б

ЛІСТИНГ

AboutActivity.java

```
package com.palianytsia.alarmclock.activities;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.palianytsia.alarmclock.BuildConfig;
import com.palianytsia.alarmclock.R;

import butterknife.BindView;
import butterknife.ButterKnife;

public class AboutActivity extends AppCompatActivity {

    //@BindView(R.id.version) TextView versionText;
    @BindView(R.id.toolbar) Toolbar toolbar;
    @BindView(R.id.root) LinearLayout linearLayout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about);
        ButterKnife.bind(this);
        setSupportActionBar(toolbar);
        toolbar.setNavigationIcon(R.drawable.ic_arrow_back_white_24dp);
        if (getActionBar() != null) getActionBar().setDisplayHomeAsUpEnabled(true);

        //versionText.setText(BuildConfig.VERSION_NAME);
    }

    public void launchEmail(View view) {
        Intent intent = new Intent(android.content.Intent.ACTION_SEND);
        intent.setType("plain/text");
        intent.putExtra(android.content.Intent.EXTRA_EMAIL, new String[]{getString(R.string.email)});
        startActivity(Intent.createChooser(intent, getString(R.string.send_email)));
    }

    public void showLibrariesDialog(View view) {
        LayoutInflater inflater = getLayoutInflater();
        View dialogView = inflater.inflate(R.layout.view_dialog_libraries, linearLayout, false);
```

```
        AlertDialog.Builder builder = new AlertDialog.Builder(this, R.style.Dialog);
        builder.setTitle(R.string.libraries);
        builder.setView(dialogView);
        builder.setPositiveButton(R.string.ok, null);
        builder.show();

        dialogView.findViewById(R.id.tab_link).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_VIEW);
                intent.setData(Uri.parse(getString(R.string.tab_link)));
                startActivity(intent);
            }
        });

        dialogView.findViewById(R.id.butter_knife_link).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_VIEW);
                intent.setData(Uri.parse(getString(R.string.butter_knife_link)));
                startActivity(intent);
            }
        });

        dialogView.findViewById(R.id.material_dialogs_link).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_VIEW);
                intent.setData(Uri.parse(getString(R.string.material_dialogs_link)));
                startActivity(intent);
            }
        });
    }

    public void showContributorsDialog(View view) {
        LayoutInflater inflater = getLayoutInflater();
        View dialogView = inflater.inflate(R.layout.view_dialog_contributors, linearLayout, false);

        AlertDialog.Builder builder = new AlertDialog.Builder(this, R.style.Dialog);
        builder.setTitle(R.string.detail_information);
        builder.setView(dialogView);
        builder.setPositiveButton(R.string.ok, null);
        builder.show();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                onBackPressed();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

CreateEditActivity

```
package com.palianytsia.alarmclock.activities;

import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
```

```
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.support.annotation.ColorInt;
import android.support.annotation.NonNull;
import android.support.design.widget.CoordinatorLayout;
import android.support.design.widget.Snackbar;
import android.support.v4.app.DialogFragment;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.SwitchCompat;
import android.support.v7.widget.Toolbar;
import android.text.format.DateFormat;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.TimePicker;

import com.afollestad.materialdialogs.color.ColorChooserDialog;
import com.palianytsia.alarmclock.database.DatabaseHelper;
import com.palianytsia.alarmclock.dialogs.AdvancedRepeatSelector;
import com.palianytsia.alarmclock.dialogs.DaysOfWeekSelector;
import com.palianytsia.alarmclock.dialogs.IconPicker;
import com.palianytsia.alarmclock.dialogs.RepeatSelector;
import com.palianytsia.alarmclock.models.Colour;
import com.palianytsia.alarmclock.models.Reminder;
import com.palianytsia.alarmclock.R;
import com.palianytsia.alarmclock.receivers.AlarmReceiver;
import com.palianytsia.alarmclock.utils.AlarmUtil;
import com.palianytsia.alarmclock.utils.AnimationUtil;
import com.palianytsia.alarmclock.utils.DateAndTimeUtil;
import com.palianytsia.alarmclock.utils.TextFormatUtil;

import java.util.Calendar;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class CreateEditActivity extends AppCompatActivity implements ColorChooserDialog.ColorCallback,
        IconPicker.IconSelectionListener, AdvancedRepeatSelector.AdvancedRepeatSelectionListener,
        DaysOfWeekSelector.DaysOfWeekSelectionListener, RepeatSelector.RepeatSelectionListener {

    @BindView(R.id.create_coordinator) CoordinatorLayout coordinatorLayout;
    @BindView(R.id.notification_title) EditText titleEditText;
    @BindView(R.id.notification_content) EditText contentEditText;
    @BindView(R.id.time) TextView timeText;
    @BindView(R.id.date) TextView dateText;
    @BindView(R.id.repeat_day) TextView repeatText;
    @BindView(R.id.switch_toggle) SwitchCompat foreverSwitch;
    @BindView(R.id.show_times_number) EditText timesEditText;
    @BindView(R.id.forever_row) LinearLayout foreverRow;
    @BindView(R.id.bottom_row) LinearLayout bottomRow;
    @BindView(R.id.bottom_view) View bottomView;
    @BindView(R.id.show) TextView showText;
    @BindView(R.id.times) TextView timesText;
    @BindView(R.id.select_icon_text) TextView iconText;
```

```
@BindView(R.id.select_colour_text) TextView colourText;
@BindView(R.id.colour_icon) ImageView imageColourSelect;
@BindView(R.id.selected_icon) ImageView imageIconSelect;
@BindView(R.id.error_time) ImageView imageWarningTime;
@BindView(R.id.error_date) ImageView imageWarningDate;
@BindView(R.id.error_show) ImageView imageWarningShow;
@BindView(R.id.toolbar) Toolbar toolbar;

private String icon;
private String colour;
private Calendar calendar;
private boolean[] daysOfWeek = new boolean[7];
private int timesShown = 0;
private int timesToShow = 1;
private int repeatType;
private int id;
private int interval = 1;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_create);
   ButterKnife.bind(this);

   setSupportActionBar(toolbar);
   toolbar.setNavigationIcon(R.drawable.ic_close_white_24dp);
   if (getActionBar() != null) getActionBar().setDisplayHomeAsUpEnabled(true);
   if (getSupportActionBar() != null) getSupportActionBar().setTitle(null);

   calendar = Calendar.getInstance();
   icon = getString(R.string.default_icon_value);
   colour = getString(R.string.default_colour_value);
   repeatType = Reminder.DOES_NOT_REPEAT;
   id = getIntent().getIntExtra("NOTIFICATION_ID", 0);

   // Check whether to edit or create a new notification
   if (id == 0) {
      DatabaseHelper database = DatabaseHelper.getInstance(this);
      id = database.getLastNotificationId() + 1;
      database.close();
   } else {
      assignReminderValues();
   }
}

public void assignReminderValues() {
   // Prevent keyboard from opening automatically
   getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

   DatabaseHelper database = DatabaseHelper.getInstance(this);
   Reminder reminder = database.getNotification(id);
   database.close();

   timesShown = reminder.getNumberShown();
   repeatType = reminder.getRepeatType();
   interval = reminder.getInterval();
   icon = reminder.getIcon();
   colour = reminder.getColour();

   calendar = DateAndTimeUtil.parseDateAndTime(reminder.getDateAndTime());

   showText.setText(getString(R.string.times_shown_edit, reminder.getNumberShown()));
```

```
        titleEditText.setText(reminder.getTitle());
        contentEditText.setText(reminder.getContent());
        dateText.setText(DateAndTimeUtil.toStringReadableDate(calendar));
        timeText.setText(DateAndTimeUtil.toStringReadableTime(calendar, this));
        timesEditText.setText(String.valueOf(reminder.getNumberToShow()));
        colourText.setText(colour);
        imageColourSelect.setColorFilter(Color.parseColor(colour));
        timesText.setVisibility(View.VISIBLE);

        if (!getString(R.string.default_icon).equals(icon)) {
            imageIconSelect.setImageResource(getResources().getIdentifier(reminder.getIcon(), "drawable",
getPackageName()));
            iconText.setText(R.string.custom_icon);
        }

        if (reminder.getRepeatType() != Reminder.DOES_NOT_REPEAT) {
            if (reminder.getInterval() > 1) {
                repeatText.setText(TextFormatUtil.formatAdvancedRepeatText(this, repeatType, interval));
            } else {
                repeatText.setText(getResources().getStringArray(R.array.repeat_array)[reminder.getRepeatType()]);
            }
            showFrequency(true);
        }

        if (reminder.getRepeatType() == Reminder.SPECIFIC_DAYS) {
            daysOfWeek = reminder.getDaysOfWeek();
            repeatText.setText(TextFormatUtil.formatDaysOfWeekText(this, daysOfWeek));
        }

        if (Boolean.parseBoolean(reminder.getForeverState())) {
            foreverSwitch.setChecked(true);
            bottomRow.setVisibility(View.GONE);
        }
    }

    public void showFrequency(boolean show) {
        if (show) {
            foreverRow.setVisibility(View.VISIBLE);
            bottomRow.setVisibility(View.VISIBLE);
            bottomView.setVisibility(View.VISIBLE);
        } else {
            foreverSwitch.setChecked(false);
            foreverRow.setVisibility(View.GONE);
            bottomRow.setVisibility(View.GONE);
            bottomView.setVisibility(View.GONE);
        }
    }

    @OnClick(R.id.time_row)
    public void timePicker() {
        TimePickerDialog TimePicker = new TimePickerDialog(CreateEditActivity.this, new
TimePickerDialog.OnTimeSetListener() {
            @Override
            public void onTimeSet(TimePicker timePicker, int hour, int minute) {
                calendar.set(Calendar.HOUR_OF_DAY, hour);
                calendar.set(Calendar.MINUTE, minute);
                timeText.setText(DateAndTimeUtil.toStringReadableTime(calendar, getApplicationContext()));
            }
        }, calendar.get(Calendar.HOUR_OF_DAY), calendar.get(Calendar.MINUTE), DateFormat.is24HourFormat(this));
        TimePicker.show();
    }
```

```
    @OnClick(R.id.date_row)
    public void datePicker(View view) {
        DatePickerDialog DatePicker = new DatePickerDialog(CreateEditActivity.this, new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker DatePicker, int year, int month, int dayOfMonth) {
                calendar.set(Calendar.YEAR, year);
                calendar.set(Calendar.MONTH, month);
                calendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
                dateText.setText(DateAndTimeUtil.toStringReadableDate(calendar));
            }
        }, calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH), calendar.get(Calendar.DAY_OF_MONTH));
        DatePicker.show();
    }

    @OnClick(R.id.icon_select)
    public void iconSelector() {
        DialogFragment dialog = new IconPicker();
        dialog.show(getSupportFragmentManager(), "IconPicker");
    }

    @Override
    public void onIconSelection(DialogFragment dialog, String iconName, String iconType, int iconResId) {
        icon = iconName;
        iconText.setText(iconType);
        imageIconSelect.setImageResource(iconResId);
        dialog.dismiss();
    }

    @OnClick(R.id.colour_select)
    public void colourSelector() {
        DatabaseHelper database = DatabaseHelper.getInstance(this);
        int[] colours = database.getColoursArray();
        database.close();

        new ColorChooserDialog.Builder(this, R.string.select_colour)
                .allowUserColorInputAlpha(false)
                .customColors(colours, null)
                .preselect(Color.parseColor(colour))
                .show();
    }

    @Override
    public void onColorSelection(@NonNull ColorChooserDialog dialog, @ColorInt int selectedColour) {
        colour = String.format("#%06X", (0xFFFFFF & selectedColour));
        imageColourSelect.setColorFilter(selectedColour);
        colourText.setText(colour);
        DatabaseHelper database = DatabaseHelper.getInstance(this);
        database.addColour(new Colour(selectedColour,
DateAndTimeUtil.toStringDateTimeWithSeconds(Calendar.getInstance())));
        database.close();
    }

    @OnClick(R.id.repeat_row)
    public void repeatSelector() {
        DialogFragment dialog = new RepeatSelector();
        dialog.show(getSupportFragmentManager(), "RepeatSelector");
    }

    @Override
    public void onRepeatSelection(DialogFragment dialog, int which, String repeatText) {
        interval = 1;
```

```
    repeatType = which;
    this.repeatText.setText(repeatText);
    if (which == Reminder.DOES_NOT_REPEAT) {
        showFrequency(false);
    } else {
        showFrequency(true);
    }
}

@Override
public void onDaysOfWeekSelected(boolean[] days) {
    repeatText.setText(TextFormatUtil.formatDaysOfWeekText(this, days));
    daysOfWeek = days;
    repeatType = Reminder.SPECIFIC_DAYS;
    showFrequency(true);
}

@Override
public void onAdvancedRepeatSelection(int type, int interval, String repeatText) {
    repeatType = type;
    this.interval = interval;
    this.repeatText.setText(repeatText);
    showFrequency(true);
}

public void saveNotification() {
    DatabaseHelper database = DatabaseHelper.getInstance(this);
    Reminder reminder = new Reminder()
            .setId(id)
            .setTitle(titleEditText.getText().toString())
            .setContent(contentEditText.getText().toString())
            .setDateAndTime(DateAndTimeUtil.toStringDateAndTime(calendar))
            .setRepeatType(repeatType)
            .setForeverState(Boolean.toString(foreverSwitch.isChecked()))
            .setNumberToShow(timesToShow)
            .setNumberShown(timesShown)
            .setIcon(icon)
            .setColour(colour)
            .setInterval(interval);

    database.addNotification(reminder);

    if (repeatType == Reminder.SPECIFIC_DAYS) {
        reminder.setDaysOfWeek(daysOfWeek);
        database.addDaysOfWeek(reminder);
    }

    database.close();
    Intent alarmIntent = new Intent(this, AlarmReceiver.class);
    calendar.set(Calendar.SECOND, 0);
    AlarmUtil.setAlarm(this, alarmIntent, reminder.getId(), calendar);
    finish();
}

@OnClick(R.id.forever_row)
public void toggleSwitch() {
    foreverSwitch.toggle();
    if (foreverSwitch.isChecked()) {
        bottomRow.setVisibility(View.GONE);
    } else {
        bottomRow.setVisibility(View.VISIBLE);
    }
```

```
}

@OnClick(R.id.switch_toggle)
public void switchClicked() {
    if (foreverSwitch.isChecked()) {
        bottomRow.setVisibility(View.GONE);
    } else {
        bottomRow.setVisibility(View.VISIBLE);
    }
}

public void validateInput() {
    imageWarningShow.setVisibility(View.GONE);
    imageWarningTime.setVisibility(View.GONE);
    imageWarningDate.setVisibility(View.GONE);
    Calendar nowCalendar = Calendar.getInstance();

    if (timeText.getText().equals(getString(R.string.time_now))) {
        calendar.set(Calendar.HOUR_OF_DAY, nowCalendar.get(Calendar.HOUR_OF_DAY));
        calendar.set(Calendar.MINUTE, nowCalendar.get(Calendar.MINUTE));
    }

    if (dateText.getText().equals(getString(R.string.date_today))) {
        calendar.set(Calendar.YEAR, nowCalendar.get(Calendar.YEAR));
        calendar.set(Calendar.MONTH, nowCalendar.get(Calendar.MONTH));
        calendar.set(Calendar.DAY_OF_MONTH, nowCalendar.get(Calendar.DAY_OF_MONTH));
    }

    // Check if the number of times to show notification is empty
    if (timesEditText.getText().toString().isEmpty()) {
        timesEditText.setText("1");
    }

    timesToShow = Integer.parseInt(timesEditText.getText().toString());
    if (repeatType == Reminder.DOES_NOT_REPEAT) {
        timesToShow = timesShown + 1;
    }

    // Check if selected date is before today's date
    if (DateAndTimeUtil.toLongDateAndTime(calendar) < DateAndTimeUtil.toLongDateAndTime(nowCalendar)) {
        Snackbar.make(coordinatorLayout, R.string.toast_past_date, Snackbar.LENGTH_SHORT).show();
        imageWarningTime.setVisibility(View.VISIBLE);
        imageWarningDate.setVisibility(View.VISIBLE);

        // Check if title is empty
    } else if (titleEditText.getText().toString().trim().isEmpty()) {
        Snackbar.make(coordinatorLayout, R.string.toast_title_empty, Snackbar.LENGTH_SHORT).show();
        AnimationUtil.shakeView(titleEditText, this);

        // Check if times to show notification is too low
    } else if (timesToShow <= timesShown && !foreverSwitch.isChecked()) {
        Snackbar.make(coordinatorLayout, R.string.toast_higher_number, Snackbar.LENGTH_SHORT).show();
        imageWarningShow.setVisibility(View.VISIBLE);
    } else {
        saveNotification();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_create, menu);
    return true;
```

```
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                onBackPressed();
                return true;
            case R.id.action_save:
                validateInput();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

MainActivity.java

```
package com.palianytsia.alarmclock.activities;

import android.content.Intent;
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.TypedValue;
import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;

import com.astuetz.PagerSlidingTabStrip;
import com.palianytsia.alarmclock.R;
import com.palianytsia.alarmclock.adapters.ReminderAdapter;
import com.palianytsia.alarmclock.adapters.ViewPageAdapter;

import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

public class MainActivity extends AppCompatActivity implements ReminderAdapter.RecyclerListener {

    @BindView(R.id.tabs) PagerSlidingTabStrip pagerSlidingTabStrip;
    @BindView(R.id.toolbar) Toolbar toolbar;
    @BindView(R.id.viewpager) ViewPager viewPager;
    @BindView(R.id.fab_button) FloatingActionButton floatingActionButton;

    private boolean fabIsHidden = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);

        setSupportActionBar(toolbar);
        if (getSupportActionBar() != null) {
            getSupportActionBar().setTitle(null);
        }

        ViewPageAdapter adapter = new ViewPageAdapter(getSupportFragmentManager());
```

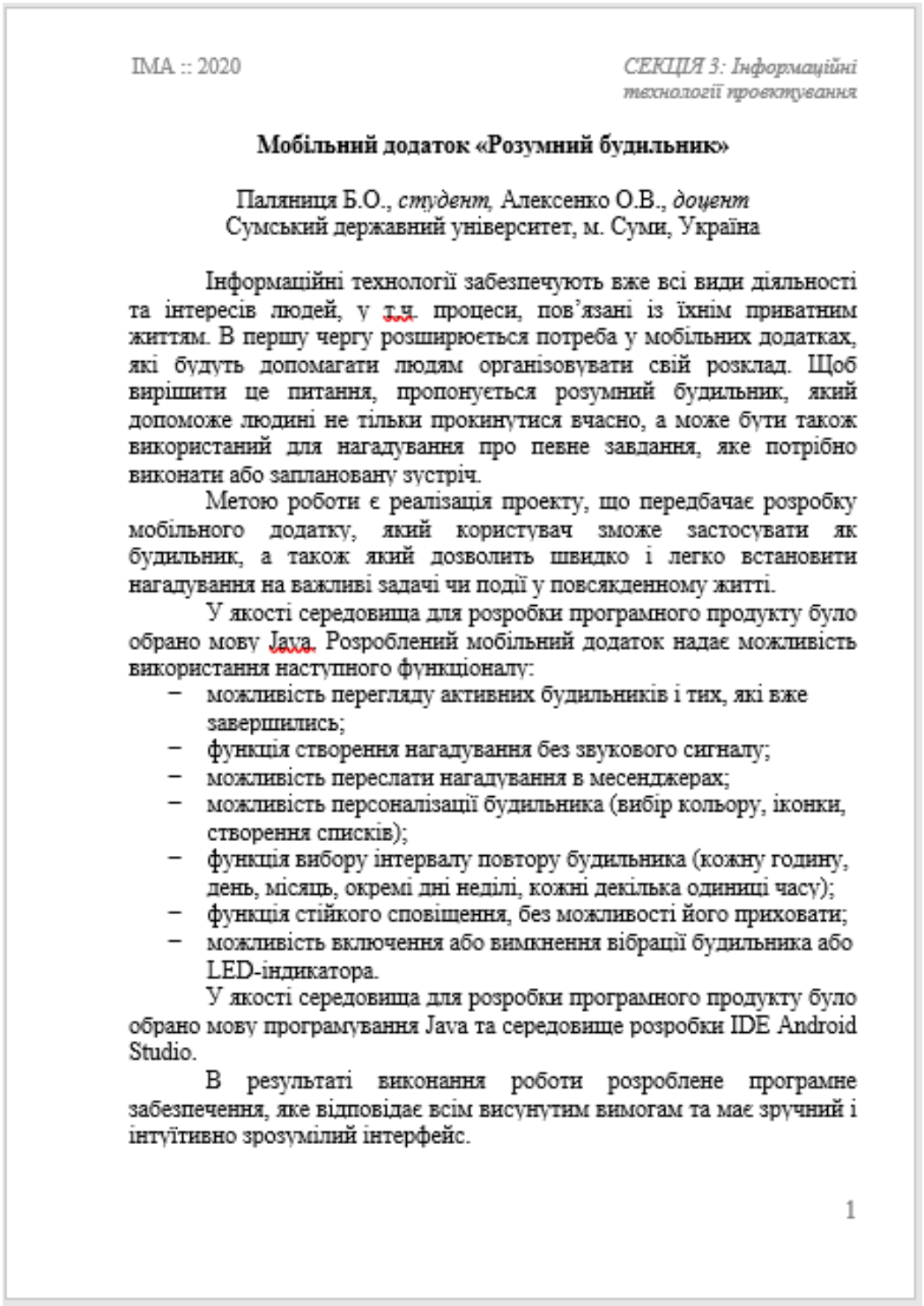
```
        viewPager.setAdapter(adapter);

        pagerSlidingTabStrip.setViewPager(viewPager);
        int pageMargin = (int) TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, 4,
getResources().getDisplayMetrics());
        viewPager.setPageMargin(pageMargin);
    }

    @OnClick(R.id.fab_button)
    public void fabClicked() {
        Intent intent = new Intent(this, CreateEditActivity.class);
        startActivity(intent);
    }

    @Override
    public void hideFab() {
        floatingActionButton.hide();
        fabIsHidden = true;
    }

    @Override
    protected void onResume() {
        super.onResume();
        if (fabIsHidden) {
            floatingActionButton.show();
            fabIsHidden = false;
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_settings:
                Intent preferenceIntent = new Intent(this, PreferenceActivity.class);
                startActivity(preferenceIntent);
                return true;
            case R.id.action_about:
                Intent aboutIntent = new Intent(this, AboutActivity.class);
                startActivity(aboutIntent);
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

PreferenceActivity.java

```
package com.palianytsia.alarmclock.activities;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.palianytsia.alarmclock.R;
import com.palianytsia.alarmclock.fragments.PreferenceFragment;
```

```
public class PreferenceActivity extends AppCompatActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.pref_with_toolbar);

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        if (toolbar != null) toolbar.setNavigationIcon(R.drawable.ic_arrow_back_white_24dp);
        if (getActionBar() != null) getActionBar().setDisplayHomeAsUpEnabled(true);

        getFragmentManager().beginTransaction().replace(R.id.content_frame, new PreferenceFragment()).commit();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                onBackPressed();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

ДОДАТОК В



IMA :: 2020 *СЕКЦІЯ 3: Інформаційні технології проектування*

Мобільний додаток «Розумний будильник»

Паляниця Б.О., *студент*, Алексенко О.В., *доцент*
Сумський державний університет, м. Суми, Україна

Інформаційні технології забезпечують вже всі види діяльності та інтересів людей, у т.ч. процеси, пов'язані із їхнім приватним життям. В першу чергу розширюється потреба у мобільних додатках, які будуть допомагати людям організовувати свій розклад. Щоб вирішити це питання, пропонується розумний будильник, який допоможе людині не тільки прокинутися вчасно, а може бути також використаний для нагадування про певне завдання, яке потрібно виконати або заплановану зустріч.

Метою роботи є реалізація проекту, що передбачає розробку мобільного додатку, який користувач зможе застосувати як будильник, а також який дозволить швидко і легко встановити нагадування на важливі задачі чи події у повсякденному житті.

У якості середовища для розробки програмного продукту було обрано мову Java. Розроблений мобільний додаток надає можливість використання наступного функціоналу:

- можливість перегляду активних будильників і тих, які вже завершились;
- функція створення нагадування без звукового сигналу;
- можливість переслати нагадування в месенджерах;
- можливість персоналізації будильника (вибір кольору, іконки, створення списків);
- функція вибору інтервалу повтору будильника (кожну годину, день, місяць, окремі дні неділі, кожні декілька одиниці часу);
- функція стійкого сповіщення, без можливості його приховати;
- можливість включення або вимкнення вібрації будильника або LED-індикатора.

У якості середовища для розробки програмного продукту було обрано мову програмування Java та середовище розробки IDE Android Studio.

В результаті виконання роботи розроблене програмне забезпечення, яке відповідає всім висунутим вимогам та має зручний і інтуїтивно зрозумілий інтерфейс.

1

Рисунок В.1 – Тези доповіді для конференції IMA 2020