МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ

Кафедра Комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
Аліна САВЧЕНКО
«_____» _____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА

(ДИПЛОМНИЙ ПРОЄКТ, ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ **"БАКАЛАВР"**
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
"ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: " Android-додаток для організації та планування завдань **"**

Виконавець: студентка групи УС-412 Пашина Уляна Олексіївна

Керівник: к.т.н., доцент Єгоров Сергій Вікторович

Нормоконтролер: Олександр ШЕВЧЕНКО

Київ – 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *Комп'ютерних інформаційних технологій*

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«____»_____2023р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студентки

Пашиної Уляни Олексіївни
(прізвище, ім'я, по батькові)

1. Тема роботи: «Android-додаток для організації та планування завдань» затверджена наказом ректора від 01.05.2023р. №623/ст.

2. Термін виконання роботи: з 15.05.2023 р. По 25.06.2023р.

3. Вихідні данні до роботи: Android-додаток для організації та планування завдань, який допоможе користувачам раціонально розприділити потрібні задачі по категоріям, та не забути про важливі справи.

4. Зміст пояснювальної записки: вступ, класифікація мобільних додатків, мобільні операційні системи, основні поняття та види мов програмування, середовище розробки, основні етапи розробки мобільного додатку, сервіси для створення прототипів додатку, аналіз аналогічних додатків у Google Play, уточнення вимог до системи та дизайн додатку, створення прототипу додатку у сервісі Figma, створення піктограми для додатку, основний алгоритм написання програми для створення нотаток, архітектура програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу: слайди презентації Microsoft Power Point.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Отримання завдання на дипломну роботу, створення плану дипломної роботи та побудова плану-графіку виконання робіт.	15.05.2023–17.05.2023	
2.	Розроблення та затвердження календарного плану виконання дипломної роботи.	18.05.2023–20.05.2023	
3.	Проведення консультацій з науковим керівником.	21.05.2023–23.05.2023	
4.	Розробка Розділу 1	24.05.2023–02.06.2023	
5.	Розробка Розділу 2	03.06.2023–09.06.2023	
6.	Розробка Розділу 3 та Висновки	10.06.2023–12.06.2023	
7.	Оформлення та друк пояснювальної записки.	12.06.2023	
8.	Створення презентації, доповіді та підготовка до захисту дипломної роботи.	13.06.2023–15.06.2023	
9.	Підготовка матеріалів дипломної роботи для передачі секретарю ДЕК (папка, конверт, диск із файлом диплому, рецензія, відгук).	16.06.2023	

7. Дата видачі завдання: «15» травня 2023 р.

Керівник дипломної роботи Сергій ЄГОРОВ
(П.І.Б.)

Завдання прийняв до виконання Уляна ПАШИНА
(П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Android-додаток для організації та планування завдань» складається зі вступу, трьох розділів, висновку, списку бібліографічних посилань та одного додатку і містить 75 сторінок та 30 рисунків. Список бібліографічних посилань складається з 14 найменувань.

Об'єктом дослідження є процес аналізу та розробки робочого фунціоналу Android-додатка для організації та планування завдань.

Метою дипломної роботи розробка програми ведення нотаток для телефонів з операційною системою Android.

Методи дослідження включають у себе:

- методи тестування програмного забезпечення;
- методи створення бази даних;
- методи створення користувацького інтерфейсу.

Результати: результат полягає в тому, що на основі отриманих знань:

1) можна в короткий термін ознайомитися з необхідною теорією з розробки мобільних додатків;
2) знайдено та використано: найефективніше середовище розробки та сервіс для створення прототипів додатку;
3) розроблено Android-додаток для організації та планування завдань, алгоритм написання якого, можна використовувати при розробці подібних програм.

ANDROID-ДОДАТОК, МОБІЛЬНИЙ ДОДАТКИ, ОПЕРАЦІЙНА СИСТЕМА, МОВА ПРОГРАМУВАННЯ, СЕРЕДОВИЩЕ РОЗРОБКИ, РОЗРОБКА, ПРОТОТИП ДОДАТКУ, GOOGLE PLAY, FIGMA, ПІКТОГРАМА, АЛГОРИТМ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, АРХІТЕКТУРА, ТЕСТУВАННЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ПЗ	Програмне забезпечення
ОС	Операційна система
API	Application Programming Interface – це програмний інтерфейс, що дозволяє пов'язувати між собою різні програми. Створений для спрощення та прискорення розробки. Містить набори методів, класів, бібліотек та функцій
WPA3	WPA3 – це третя версія захищеного Wi-Fi. Об'єднання Wi-Fi Alliance випустило WPA3 у 2018 році.
SDK	Software Development Kit – комплект розробки програмного забезпечення – набір засобів розробки, що дозволяє фахівцям з програмного забезпечення створювати програми для певного пакету програм, програмного забезпечення базових засобів розробки, апаратної платформи
IDE	Інтегроване середовище розробки – комплекс програмних засобів, що використовується програмістами для розробки програмного забезпечення.
DAO	Data Access Object – абстрактний інтерфейс до якогось типу бази даних або механізму зберігання.
SQL	Structured Query Language – декларативна мова програмування, яка використовується для створення, модифікації та управління даними в реляційній базі даних, що керується відповідною системою управління базами даних.
JVM	Java Virtual Machine – віртуальна машина Java – основна частина виконуючої системи Java, так званої Java Runtime Environment.

ВСТУП

За останні роки смартфони змінили спосіб нашого життя, роботи та спілкування. Ці портативні пристрої, які надають широкий спектр можливостей і застосувань, стали невід'ємною частиною нашого повсякденного життя. Операційна система Android від Google стала значним учасником ринку смартфонів. Android змінив мобільні технології, надавши розробникам платформу для створення нових додатків завдяки своїй природі з відкритим кодом і великій базі користувачів.

Актуальність: смартфони мають значний вплив на суспільство. Вони зробили революцію в таких галузях, як розваги, електронна комерція та охорона здоров'я, надаючи швидкий доступ до інформації та сприяючи безперебійній співпраці. Поширення смартфонів змінило те, як люди взаємодіють із технологіями, зробивши мобільні додатки важливою складовою сучасного життя.

Статистика з сайту ms.detector.media показує що, 88 % користувачів смартфонів в Україні використовують андроїд-пристрої. Цікаво, що чим більший розмір екрана смартфона, тим більше часу люди проводять у соцмережах, більше дивиться відео, роблять більше покупок. Все, що раніше робили на десктопі, переводять у смартфон. 38 % користувачів зранку найперше беруть у руки смартфон, щобпрокинутися. Тож для багатьох людей ранок починається не з кави, а зі смартфона. Окрім того, як виявилося, без смартфона не можна снідати чи обідати, відпочивати ввечері. Якщо подивитися детальніше по віку молодих користувачів (16–24 роки), то виявиться: 66 % із них прокидаються, обідають і засинають зі смартфоном. Звертаючи увагу на цифри, більша частина людей проводить за смартфоном весь свій час, що напряму впливає на психічне здоров'я кожного.

Недивно, що інформаційне перевантаження свідомості стає все більш поширеною проблемою суспільства і вчені говорять про це як про цілком реальну загрозу для здоров'я. Інформаційне перевантаження негативно впливає на якість

життя, виникають проблеми з пам'яттю, людина не може запам'ятати та згадати про важливі зустрічі, невідкладні справи та т.п.

Основною метою виконання дипломної роботи є проектування та розробка android-додатоку для організації та планування задань, завдяки якому можна буде раціонально розподілити свій час, успішно спланувати свій день та пам'ятати про всі задачі та завдання.

Новизна прєкту полягає в тому, що для користувачів даний додаток буде узагальненим варіантом корисних функцій та можливостей, які будуть зібрані в одному застосунку.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

1.1. Класифікація мобільних додатків

Мобільні програми можна класифікувати на основі різноманітних характеристик, включаючи платформу, для якої вони призначені, функції, які вони виконують, і технологію, яку вони використовують. Ось декілька типових методів класифікації мобільних програм:Класифікація на основі платформи. Мобільні програми можна класифікувати на три основні категорії залежно від платформи, для якої вони розроблені:

1. Категоризація на основі платформи:
   - Програми для Android, програми розроблені спеціально для операційної системи Android.
   - Програми для iOS, програми розроблені спеціально для операційної системи iOS від Apple.
   - Кросплатформні програми, це програми, призначені для роботи в кількох операційних системах, наприклад React Native або Flutter.
2. Класифікація за призначенням/функціоналом:
   - Службові програми, це програми, які надають зручні інструменти та послуги, такі як калькулятори, календарі, прогнози погоди тощо.
   - Програми соціальних мереж, програми, які дозволяють користувачам підключатися та взаємодіяти один з одним, ділитися вмістом і спілкуватися один з одним.

- Програми для розваг, це програми, які пропонують розваги та дозвілля, такі як ігри, потокові послуги, музичні та відеоплеєри.
- Продуктивні програми, такі програми, як програми для створення нотаток, диспетчери завдань і редактори документів, які допомагають користувачам підвищити продуктивність.
- Додатки для способу життя, додатки, пов’язані з особистими інтересами, хобі, здоров’ям, фітнесом, подорожами та іншими сферами способу життя.
- Освітні програми, це програми, які пропонують освітню інформацію, уроки, вивчення мови та розвиток навичок.
- Програми для новин і медіа: це програми, які надають новинні статті, відео, подкасти та інші типи медіа.
- Програми для електронної комерції: програми, які спрощують онлайн-покупки та електронні транзакції.
- Фінансові програми, банківські програми, програми для управління особистими грошима, складання бюджету, інвестування та платіжних послуг.
- Програми для подорожей і навігації: програми, які допомагають у плануванні подорожей, навігації, картах і пошуку місцевих служб.
- Програми для здоров’я та оздоровлення, це програми, які зосереджені на моніторингу здоров’я, відстеженні вправ, медитації та загальному самопочутті.

3. Класифікація на основі моделі доходу:

- Платні програми, програми для завантаження та використання яких потрібна передоплата.
- Безкоштовні програми з покупками в програмі, програми, які можна завантажити безкоштовно, але пропонують покупки в програмі для додаткових функцій або вмісту.

- Програми з підтримкою реклами, програми, які можна безкоштовно завантажити та використовувати, але показують рекламу, щоб заробити гроші.
- Програми на основі підписки, програми які потребують регулярних платежів, щоб отримати доступ до функцій або вмісту програми.
- Програми Freemium, програми які надають безкоштовну базову версію, але стягують плату за додаткові функції або преміальний вміст.

1.2. Концепція мобільного додатку

Основні принципи та ідеї, які впливають на розробку та дизайн мобільних додатків, називають теорією концепції мобільних додатків. Це передбачає розуміння основних частин і проблем, які впливають на дизайн і планування успішної мобільної програми.

Нижче наведені деякі з найважливіших особливостей теорії концепції мобільних додатків:

- Вимоги користувача та вирішення проблем. Концепція мобільного додатку має відповідати конкретним вимогам або проблемам користувача. Важливо визначити цільову аудиторію та її проблемні точки, щоб розробити додаток, який додає цінності та ефективно вирішує їхні проблеми.
- Дослідження ринку. Ретельне дослідження ринку допомагає зрозуміти конкурентноспроможність і виявити існуючі ринкові рішення або прогалини. Це дослідження робить внесок у концепцію програми, виявляючи переваги користувачів, тенденції та тактику потенційного прибутку.
- Сильна унікальна торгова пропозиція відрізняє програму від конкурентів і окреслює її ціннісну пропозицію. Теорія ідеї повинна зосереджуватися на виявленні та просуванні відмінних рис, можливостей або переваг програми, які допомагають їй виділитися та залучити споживачів.

- Взаємодія з користувачем та інтерфейс користувача. Теорія концепції повинна враховувати взаємодію з користувачем та інтерфейс програми. Розробка інтуїтивно зрозумілих і зручних інтерфейсів, оптимізація потоків навігації та гарантування безперебійної та чудової взаємодії з користувачем є частиною цього.
- Технологічна життєздатність. Критично важливо оцінити технологічну життєздатність концепції програми. Це включає в себе роздуми про необхідний стек технологій, взаємодію з різними мобільними платформами (наприклад, Android, iOS), інтеграцію з іншими системами чи API та масштабованість для майбутнього зростання.
- Стратегія монетизації. Чітка стратегія монетизації програми повинна бути викладена в теорії концепції. Покупки в програмі, моделі підписки, реклама та співпраця – це всі можливості. Обрана стратегія має відповідати ціннісній пропозиції програми та цільовій аудиторії.
- Ітеративна розробка. Ітеративна природа розробки додатків визнається теорією ідей мобільних додатків. Створення прототипів, тестування та отримання інформації від користувачів є частиною процесу перед тим, як перейти до повномасштабної розробки.
- Конфіденційність і безпека даних. Важливо розглянути заходи щодо конфіденційності та безпеки даних, щоб захистити інформацію користувачів і розвинути довіру. Теорія ідеї повинна включати стратегії для гарантування безпечної обробки даних, дотримання нормативних вимог і впровадження відповідних заходів безпеки.
- Постійне вдосконалення та оновлення. Щоб програма залишалася актуальною та відповідала введеним користувачами, теорія концепції повинна включати ідею постійного вдосконалення та оновлень. Це включає планування майбутніх удосконалень, виправлення помилок і оновлення функцій, щоб забезпечити тривалий і мінливий досвід роботи програми.

Теорія концепції мобільних додатків служить основою для процесу розробки та підтримує прийняття рішень протягом життєвого циклу додатка. Щоб створити привабливу та ефективну мобільну програму, вона об’єднує дослідження ринку, дизайн, орієнтований на користувача, технічні проблеми та бізнес-стратегію.

1.3. Мобільні операційні системи

Зараз у світі існує величезна кількість мов програмування для розробки мобільних додатків. Це пов’язано з тим, що для різних мобільних пристроїв потрібні різні мови програмування. Зазвичай це визначається тим, що мобільні пристрої мають різні операційні системи:

- iOS — це операційна система Apple, яка працює на пристроях iPhone, iPad та iPod Touch. Його зручний інтерфейс, функції безпеки та безперебійне з’єднання з іншими пристроями та службами Apple роблять його популярним. У iOS є власний App Store, який пропонує широкий вибір програм, розроблених спеціально для пристроїв Apple.
- Windows 10 Mobile — це операційна система для смартфонів і невеликих планшетів, розроблена Microsoft. Вона заснований на ядрі Windows NT і пропонує єдиний досвід роботи на таких пристроях, як ПК і консолі Xbox. Однак Microsoft зазначила, що підтримка Windows 10 Mobile буде припинена, і для неї не буде створено багато нових функцій або обладнання.
- ОС BlackBerry, розроблена компанією BlackBerry Limited, це операційна система, яка використовувалася в ранніх телефонах BlackBerry. Вона мала потужні заходи безпеки, справжню клавіатуру та потужні можливості обміну повідомленнями. BlackBerry, з іншого боку, перемістила свою увагу на пристрої на базі Android і тепер використовує Android як основну операційну систему.
- Tizen — це операційна система з відкритим кодом, розроблена Linux Foundation і підтримується Samsung. Вона, в основному, використовується в смарт-

годинниках і телевізорах Samsung. Дана ОС прагне забезпечити узгоджену взаємодію з користувачами на всіх пристроях Samsung і полегшує створення веб-додатків і нативних програм.

– Android від Google є популярною мобільною операційною системою. Вона побудована на ядрі Linux і, особливо, призначена для пристроїв із сенсорним екраном, таких як смартфони, планшети, розумні годинники та смарт-телевізори. Android стала найпопулярнішою у світі операційною системою для смартфонів. Розглянемо Android ОС більш детальніше.

Основні функції операційної системи Android:

– Відкритий код. Оскільки Android є платформою з відкритим кодом, її вихідний код є у вільному доступі. Це дозволяє виробникам і розробникам налаштовувати та персоналізувати операційну систему відповідно до своїх індивідуальних вимог.
– Google Play Store містить велике різноманіття програм для Android. Користувачі можуть вибирати з мільйонів додатків, серед яких ігри, інструменти підвищення продуктивності, програми для соціальних мереж тощо.
– Багатозадачність. Android підтримує багатозадачність, що дозволяє користувачам легко переходити між кількома програмами. У неї є панель «Останні програми», де користувачі можуть отримувати доступ і перемикатися між своїми нещодавно використаними програмами.
– Інтеграція служб Google. Google Пошук, Gmail, Google Карти, Google Диск і Google Assistant тісно пов’язані з Android. Ці послуги пропонують клієнтам різноманітні можливості для спілкування, роботи та використання контенту.
– Сповіщення. Android має інтерактивну систему сповіщень, яку можна налаштовувати. Сповіщення з різних програм доставляються користувачам, для того щоб переглянути певну інформацію.
– Android підтримує широкий спектр пристроїв, від недорогих бюджетних смартфонів до високоякісних флагманських телефонів. У результаті виробники

можуть надати різноманітний асортимент смартфонів Android із різними технічними характеристиками та ціновими діапазонами.

- Google випускає щомісячні оновлення операційної системи Android, які включають виправлення помилок, патчі безпеки та нові функції. Однак фактична доступність і час цих оновлень можуть відрізнятися залежно від виробника пристрою та оператора.

Зсилаючись на останній пункт про оновлення, проведемо аналіз версій операційної системи Android. Порівняємо між собою API: 29 (Android 10 Quince Tart) та API: 24 (Android 7.0 Nougat) .

Особливості API: 24 (Android 7.0 Nougat):

- Дана версія містить нову опцію багатовіконності, яка дозволяє користувачам запускати дві програми одночасно.
- Nougat покращила безпеку за допомогою шифрування на основі файлів, які ізолюють та шифрують дані окремих користувачів. Також було представлено нову парадигму дозволів, що дозволяє користувачам надавати або забороняти дозволи під час виконання.
- Nougat покращила зв’язок Bluetooth, додавши підтримку Bluetooth 5.0, що забезпечує більш високу швидкість і більший діапазон. Ця версія також представила опцію Data Saver, яка допомагає клієнтам зменшити використання стільникових даних.
- Представлено низку покращень продуктивності, зокрема компілятор Just-in-Time для швидшого встановлення додатків і оновлень системи.
- Версія Nougat додала вдосконалення які орієнтовані на розробників, наприклад: API Vulkan для високопродуктивної 3D-графіки, підтримки кількох вікон і системою швидкиї сповіщень із додатковими корисними функціями.

Особливості API: 29 (Android 10 Quince Tart):

- API 29 зосереджена на навігації за допомогою жестів, забезпечуючи повністю жестову систему навігації, яка усуває потребу в кнопках навігації. Вона також включає загальносистемний темний режим для подовження терміну служби

батареї та забезпечення більш комфортного перегляду в умовах слабкого освітлення.

- Завдяки розширеним дозволам додатків Android 10 ще більше посилила захист конфіденційності та безпеки. Вона надала користувачам додатковий контроль над доступом до місцезнаходження, обмеживши його параметрами. Також було представлено обмежене сховище, яке обмежує доступ програми до даних користувача.
- В Android 10 було додано підтримку мережі 5G, що дозволяє підвищити швидкість завантаження та вивантаження. В цій версії було покращено продуктивність Wi-Fi завдяки підтримці шифрування WPA3 і була представлена функція «Wi-Fi Easy Connect», яка спрощує налаштування мережі Wi-Fi.
- Android 10 надає пріоритет оптимізації системних ресурсів і швидшому запуску програм.
- Були представлені нові інструменти та API для розробників, наприклад: біометричний API для стандартизованої біометричної аутентифікації, також була представлен Android Jetpack платформа, яка слугує для спрощення розробки та має розширену підтримку більш складних пристроїв.

1.4. Основні поняття та види мов програмування

Мова програмування — це набір правил і структур, які визначають, як виглядає комп’ютерна програма, якими якостями чи функціями вона володіє.

Програма — це код, написаний відповідно до парадигм і правил певної мови програмування.

Вихідний код є структурою програмного забезпечення.

Оскільки мови програмування є штучними, вони включають в себе такі поняття, як алфавіт, синтаксис і семантика.

Алфавіт – набір символів, дозволений до використання, за допомогою якого можуть бути утворені слова даної мови.

Семантика – система правил тлумачення всіх мовних конструкцій, що дозволяє виконувати процес обробки даних.

Синтаксис – система правил, що визначає допустимі конструкції мови програмування з символів алфавіту.

Всі мови програмування можна поділити на два основних види: мови низького та високого рівня, розглянемо ці поняття більш детальніше:

- Мови низького рівня жорстко орієнтовані на конкретний тип обладнання. Загалом, мови низького рівня – спосіб написання комп'ютерних інструкцій на апаратній мові, тобто, в машинних кодах.
- Мови високого рівня – це мови, що дозволяють записувати програми в зручній для людини формі. Вони орієнтовані не на систему інструкцій апаратного забезпечення, а на систему операторів (інструкцій), що характерна для написання певного класу алгоритмів.

Мови високого рівня більш прості у використанні, оскільки їх завдання – обслуговувати потреби програміста, а не визначати можливості комп'ютера. Програми, написані на цих мовах, повинні бути перекодовані, тобто переведені на машинну мову, щоб перед запуском програми комп'ютер міг їх зрозуміти. Тому системи програмування, наприклад на Java, включають в себе або інтерпретатор мови, або компілятор.

Мови низького рівня, більш близькі до машинної мови, дозволяють створювати програми, які працюють швидше і дозволяють більш ефективно використовувати ресурси комп'ютера.

На сьогоднішній день існує велика кількість різноманітних мов програмування, у кожної з них своя область застосування, проте задля проведення аналізу, з метою вибору найкращої мови програмування необхідно обрати декілька самих популярних мов програмування, щоб між порівняти кожну з них.

На рис. 1.1 відображено відсоток використання мов програмування за інформацією з сайту dou.ua в комерційних робочих проектах 2023 року.

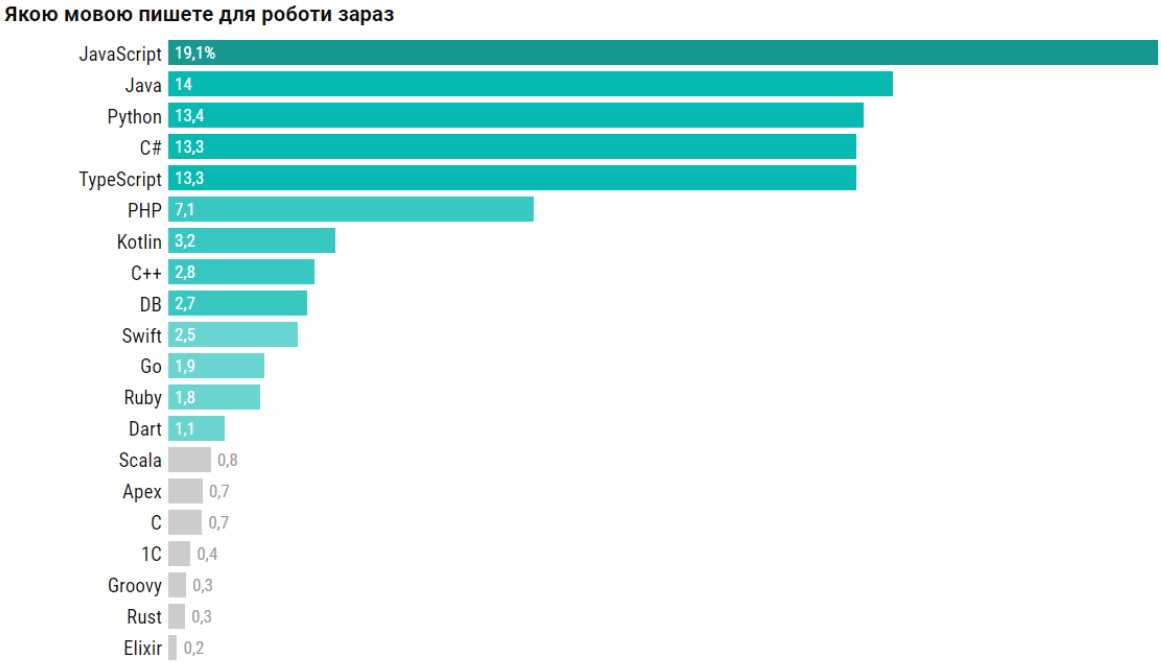


Рис. 1.1. Відсоток використання мов програмування в комерційних проєктах

Проаналізувавши дані з статистики за 2023, можна зробити висновок що: цього року найпопулярнішою мовою залишається JavaScript — 19% розробників пишуть нею комерційні проєкти. Далі — Java (14%) і Python (13,4%). Остання, вперше серед найпопулярніших. З мінімальним відривом за Python йдуть C# і TypeScript.

Близько 90% програм, представлених у сервісі Google Play, розроблені мовою програмування Java, як видно на Рис.1.1, Java займає 2 місце, тому для Android-додатків вона є найпопулярнішою. Загалом, Java використовується повсюдно, для створення різних продуктів. Так, іншим мовам поки що не світить перевищити її за частотою використання для Android. Але сьогодні все частіше починають використовуватися й інші.

Альтернативні мови для роботи з програмами Android:

- Kotlin – це друга базова мова програмування для створення програм Android. Kotlin більш легка як у вивченні, так і в роботі, що дозволяє їй стрімко набирати популярність.
- Python – використовується повсюдно і також є популярною мовою програмування на сьогоднішній день. Незважаючи на те, що її, в основному, використовують в інших сферах, для створення додатків Android вона також пристосована. Але її у цій сфері використовують дуже рідко, оскільки при написанні коду цією мовою, потрібно використати пару інструментів: бібліотеку Kivy та набір інструментів/бібліотек BeeWare.
- C/C++ – ці мови відмінно підійдуть для програм у будь-якому середовищі, у тому числі й Android. Завдяки своїй високій продуктивності, вони допоможуть зробити відмінний продукт. Але використовувати їх від початку та до кінця розробки навряд чи вдасться. Швидше за все, доведеться заручитися підтримкою Java.
- JavaScript – підходить для створення повного життєвого циклу мобільного додатка, але за умови відмінного володіння React Native та використання його у роботі.
- Dart – альтернатива JavaScript, яка створила свій додатковий інструмент, що дозволяє за його допомогою створювати продукти для Android. Його називають Flutter. Він представлений цілим комплексом пристроїв, який загалом підходить для роботи.
- Lua – мова також отримала свою платформу для розробки додатків під назвою Corona SDK. Вона також дозволить займатися реалізацією продуктів Android.

Для себе я обрала мову Java, так як ця мова довгий час була основною для ОС Android, тому в вільному доступі є досить багато літератури, що дозволить без проблемно створити додаток.

1.5. Середовище розробки

Для створення додатку нам потрібне середовище розробки. Вибір ідеального середовища – складне завдання, яке включає в себе: тестування, особисті уподобання і остаточне рішення. Перш ніж зробити вибір, проаналізуємо деякі з них.

1. Студія Xamarin – один з найкращих інструментів розробки додатків для Android. Xamarin розширює платформу розробників .NET за допомогою інструментів та бібліотек. Спеціально розробляючи програми для Android, iOS, tvOS, watchOS, macOS та Windows.

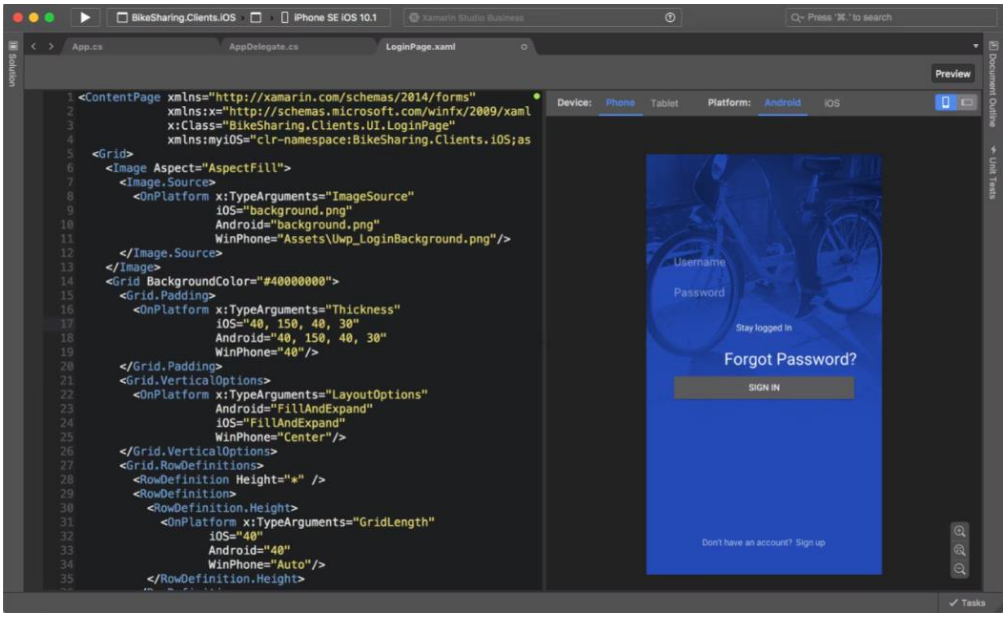


Рис. 1.2. Інтерфейс студії Xamarin

2. Microsoft Visual Code Studio – це інтегроване середовище розробки. Є продуктом від Microsoft. Отже, воно використовується для розробки комп'ютерного програмного забезпечення для Microsoft Windows та веб-сайтів. Можна використовувати і для розробки програм з ОС Android, веб-додатків та веб-сервісів. Visual Code Studio використовує платформу розробки

програмного забезпечення Microsoft. Наприклад, Windows API, Windows Forms, Windows Presentation Foundation і т.д.

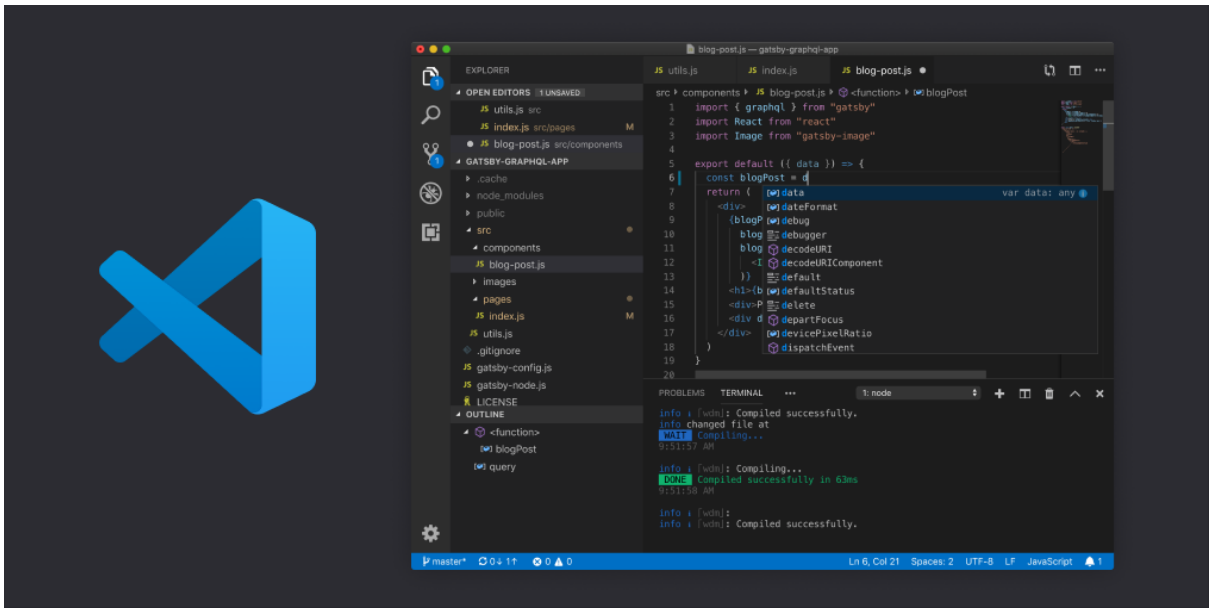


Рис. 1.3. Інтерфейс Visual Studio Code

3. Android Studio – це IDE (інтегроване середовище розробки), створене Google для розробки додатків Android. Але на відміну від інших IDE, Android Studio розроблено спеціально для Android і містить певний набір функцій та інструментів, які призначені саме для цієї мобільної операційної системи. Загалом, як офіційна IDE для розробки додатків Android, Android Studio зможе полегшити роботу розробників. Користування Android Studio є абсолютно безкоштовним, тож не доведеться вкладати додаткові фінансові ресурси, при використовуванні (крім того, що вже інвестується в розробку та тестування свого мобільного додатка, звичайно).

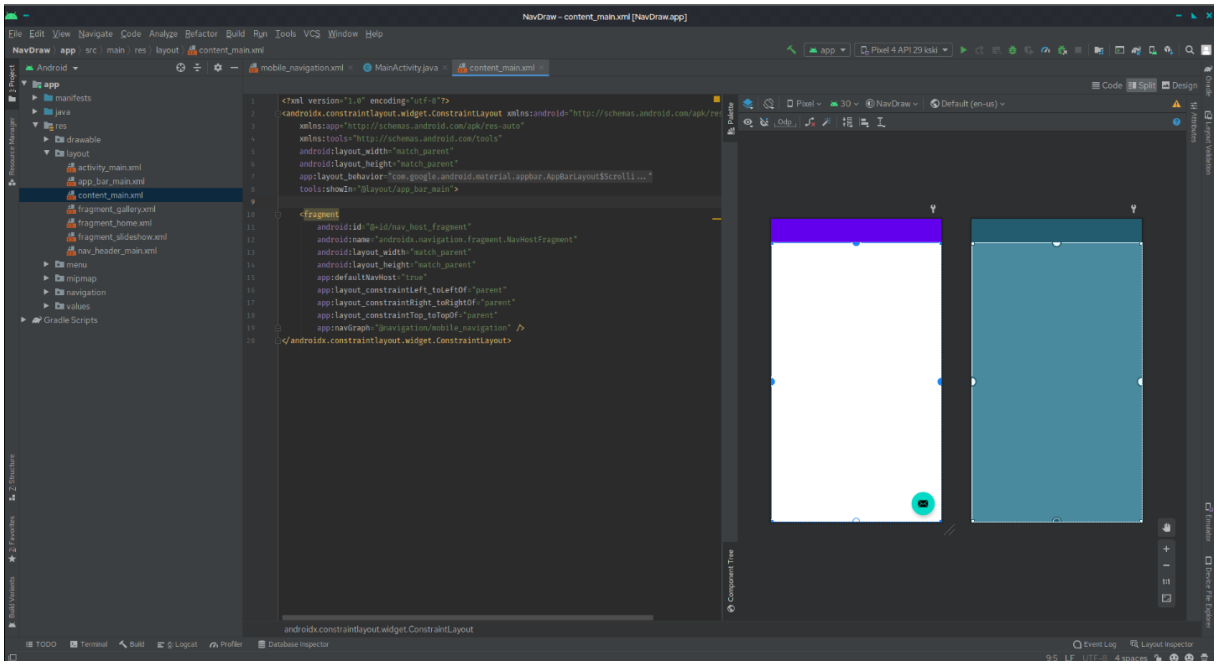


Рис. 1.4. Інтерфейс Android Studio

Android Studio є одним з найпопулярніших середовищ розробки, тому розглянемо переваги, недоліки та додаткові функції, більш детально.

Інструменти створення Android SDK:

– Android Debug Bridge – це інструмент командного рядка, який дозволяє розробникам мобільних додатків Android спілкуватися з комп'ютером і полегшує різні дії, такі як інсталяція або налагодження програми.
– Менеджер віртуальних пристроїв Android – це інструмент, який дозволяє розробникам створювати та керувати віртуальними пристроями Android, які в основному є емуляторами, за допомогою яких можна перевірити, як програми працюють на різних пристроях Android.
– Android Studio Dolphin – це інструмент, який після розробникам створює додатки макету та дизайну інтерфейсу користувача.
– Редактор макета – це інструмент перетягування, який разом швидко створює та редагує елементи інтерфейсу без необхідності писати код.

Переваги використання Android Studio:

– Швидке кодування;

Однією з головних переваг використання Android Studio є те, що він швидше кодує. Це пов’язано з такими функціями, як рендеринг у реальному часі, які дозволяють побачити зміни, які вносяться у код наглядно.

– Швидкий емулятор;

Новий емулятор в Android Studio 3.0 виявився дуже швидким, що є значною перевагою, особливо враховуючи, що повноцінність емулятора була з речами, на якій розробники найбільше скаржилися. Нова версія може запускатися менше ніж за 6 секунд, що дозволяє тестувати свою програму на більш широкому діапазоні різних пристроїв.

– Надійне тестування;

Під час створення мобільного додатка тестування є першим. На щастя, Android Studio надає розробникам усе необхідне для тестування своїх програм на різноманітних пристроях.

– Підтримка Firebase та інтегрована хмара;

Android Studio поставляється з підтримкою Firebase та інтегрованою хмарою для обміну повідомленнями. Це велика перевага, оскільки дозволяє розробникам легко додавати push-сповіщення та інші подібні функції до своїх програм.

– Багатофункціональне середовище;

Android Studio – це багатофункціональне середовище, яке надає розробникам усіх інструментів, необхідних для створення високоякісної мобільної програми Android.

Недоліки використання Android Studio:

– Використовує багато оперативної пам’яті;
– Має високі вимоги до обладнання;
– Повільне встановлення.

Зробимо висновок що, Android Studio – чудовий інструмент для розробки додатків Android. Його інтуїтивно зрозумілий інтерфейс і надійний функціонал роблять його ідеальним вибором як для початківців, так і для досвідчених

розробників, тому при розробці я буду використовувати саме це середовище розробки.

1.6. Основні етапи розробки мобільного додатку

Розробка мобільних додатків складається з кількох етапів. Розглянемо основні етапи:

- Планування та аналіз включає в себе визначення мети програми, визначення цільової аудиторії та окреслення функцій і можливостей, які надаватиме програма, дослідження ринку та техніко-економічні обґрунтування, для того щоб оцінити життєздатність програми.
- Дизайн включає в себе створення прототипу та дизайну інтерфейсу користувача для програми.
- Розробка цей етап складається з кодування програми та інтеграції всіх необхідних функцій.
- Тестування на цьому етапі розробники перевіряють програму на наявність помилок, проблем із зручністю використання та продуктивністю. Також проводяться користувацькі тести, щоб зібрати відгуки та вдосконалити дизайн і функціональність програми.
- Розгортання, коли додаток протестовано та готово, розробники публікують його в магазині для програм. Вони також оптимізують програму для пошукових систем, налаштовують аналітику програми та інтегрують необхідні стратегії монетизації програми.
- Технічне обслуговування та оновлення, після випуску програми розробники продовжують відстежувати продуктивність програми, виправляти будь-які помилки чи проблеми, що виникають, і надавати регулярні оновлення для додавання нових функцій або можливостей.

Загалом процес розробки мобільних додатків є повторюваним і включає різні етапи планування, проектування, розробки, тестування, розгортання та

обслуговування. Кожен етап потребує ретельного розгляду, планування та виконання, щоб гарантувати, що додаток є функціональним, простим у використанні та привабливим для цільової аудиторії.

1.7. Сервіси для створення прототипів додатку

Для більш легшої розробки додатку, слід спочатку спроектувати інтерфейс, щоб мати уявлення про оформлення та розташування потрібних елементів. Для цього можна використовувати багато сервісів, тому я розглянула та порівняла два популярних сервіси:

1. Figma;

Це хмарний інструмент для проектування та створення прототипів, який використовується для проектування інтерфейсу. Він дозволяє дизайнерам створювати та ділитися дизайнами з членами своєї команди в режимі реального часу. Figma пропонує ряд функцій, таких як інструменти для редагування векторів, сітки макетів та інструменти для створення прототипів, які допомагають дизайнерам створювати високоточні проекти, якими можна легко поділитися та протестувати. Figma також дозволяє дизайнерам створювати інтерактивні прототипи, які можна тестувати та перевіряти разом з користувачами, допомагаючи покращити взаємодію з продуктом або послугою. Figma підтримує різні формати дизайну, такі як файли Sketch, Adobe, що полегшує імпорт та експорт дизайнів. Він також пропонує широкий спектр плагінів та інтеграцій з іншими інструментами, такими як Jira, Slack і Trello, що робить його універсальним інструментом для команд дизайнерів.

2. InVision.

Це хмарна платформа, яка дозволяє дизайнерам створювати інтерактивні прототипи, співпрацювати з членами команди та отримувати відгуки від зацікавлених сторін. InVision надає ряд інструментів для команд проектувальників, включаючи керування системою проектування, тестування користувачами та співпрацю в дизайні. Прототипами можна легко поділитися із зацікавленими сторонами,

дозволяючи їм протестувати дизайн і надати відгук про нього. InVision також пропонує інструмент керування системою проектування, який дозволяє дизайнерам створювати та підтримувати бібліотеку компонентів дизайну та стилів, які можна легко повторно використовувати в кількох проектах. InVision підтримує різні формати дизайну, зокрема Sketch, Adobe, що полегшує імпорт та експорт дизайнів. Він також пропонує інтеграцію з іншими інструментами, такими як Jira, Slack і Trello, що дозволяє дизайнерам легше співпрацювати з членами команди. InVision також надає інструменти для тестування та дослідження користувачів, які дозволяють дизайнерам тестувати свої проекти з реальними користувачами та збирати відгуки. Ці інструменти можуть допомогти дизайнерам краще зрозуміти, як користувачі взаємодіють із їхніми проектами та приймають керовані даними дизайнерські рішення.

Отже, порівнявши ці дві платформи, я вирішила що буду використовувати сервіс Figma, оскільки він зручний, зрозумілий у використанні та досить популярний.

1.8. Висновок до розділу 1

В першому розділі було детально розглянуто теоретичні основи розробки мобільних додатків. Мною було проаналізовано класифікацію мобільних додатків, досліджено теорію концепції мобільних додатків, досліджено операційні системи, вивчено основні поняття та обрано мову програмування. Було розглянуто декілька середовищ розробки, проаналізовано інструменти створення Android SDK, розглянуто основні етапи розробки мобільного додатку. Також було розглянуто та проаналізовано сервіси для створення прототипів додатку.

Мною було прийнято рішення використовувати найефективніше середовище Android Studio і мову програмування Java. Для проектування прототипу мого додатку я буду використовувати сервіс Figma.

РОЗДІЛ 2
ВИМОГИ ДО СИСТЕМИ

2.1. Google Play та його затребуваність користувачами Android

Google Play — це офіційний магазин програм для пристроїв Android, який часто використовується користувачами Android з різних причин. Ось деякі з основних причин його популярності:

- У Google Play є величезний вибір програм, який включає все: від службових і продуктивних інструментів до програм для розваг, ігор і соціальних мереж. Користувачі мають доступ до мільйонів безкоштовних і платних програм у різноманітних категоріях.
- Google Play є офіційним магазином програм для Android, який пропонує Google. Користувачі довіряють програмам, які пропонуються в Google Play, оскільки вважають, що вони законні, безпечні та надійні.
- Запобіжні заходи: щоб захистити споживачів від потенційно шкідливих або зловмисних програм, Google встановив суворі запобіжні заходи в Google Play. Платформа використовує комплексне сканування безпеки та алгоритми виявлення зловмисного програмного забезпечення для виявлення та видалення небезпечних програм, гарантуючи безпечніший досвід користувачів.
- Google Play часто оновлюється та вдосконалюється, пропонуючи нові функції та зміни для покращення взаємодії з користувачем і вирішення будь-яких проблем безпеки.

Ці оновлення забезпечують споживачам доступ до найновіших функцій і виправлень помилок, що підвищує загальну ефективність магазину програм.

– Google Play плавно інтегрований в операційну систему Android, що дозволяє користувачам досліджувати та завантажувати програми прямо зі своїх телефонів. Його часто попередньо встановлюють на смартфонах і планшетах Android, що забезпечує легкість використання та доступність.
– Підтримка розробників: Google Play надає розробникам складну платформу для просування та розповсюдження своїх програм серед великої кількості користувачів. Магазин надає різноманітні ресурси, документацію та інструменти, які допоможуть розробникам ефективно створювати, тестувати та публікувати свої програми. Завдяки цій допомозі розробників платформа тепер має різноманітний вибір високоякісних програм.
– Відгуки та оцінки користувачів: Google Play дозволяє користувачам оцінювати та переглядати програми, які вони завантажили. Базуючись на досвіді та відгуках інших користувачів, ця функція допомагає користувачам приймати обґрунтовані рішення щодо встановлення програм. Він підтримує підхід, керований спільнотою, за якого користувачі можуть покладатися на колективну мудрість інших користувачів Android.
– Google Play надає розробникам зручну платформу для здійснення покупок у додатках і підписок, що дозволяє споживачам отримувати доступ до додаткових функцій або вмісту в їхніх улюблених додатках. Ця екосистема заохочує прибутковість для розробників, а також дозволяє користувачам налаштовувати свої додатки.

Загалом величезний асортимент програм, заходи безпеки, регулярні оновлення, легка інтеграція, допомога розробників, відгуки користувачів і можливості покупки в програмі – усе це сприяє популярності Google Play серед користувачів Android. Він став життєво важливою частиною екосистеми Android, надаючи широкий вибір додатків і забезпечуючи користувачам зручне та надійне завантаження додатків.

2.2. Аналіз аналогічних додатків у Google Play

У магазині мобільних додатків Google Play є багато додатків-планерів. Тому розглянемо найпопулярніші з них:

1. Лідером є додаток «Список задач» від розробника Splend Apps, у цього додатку більш ніж 10 млн завантажень.

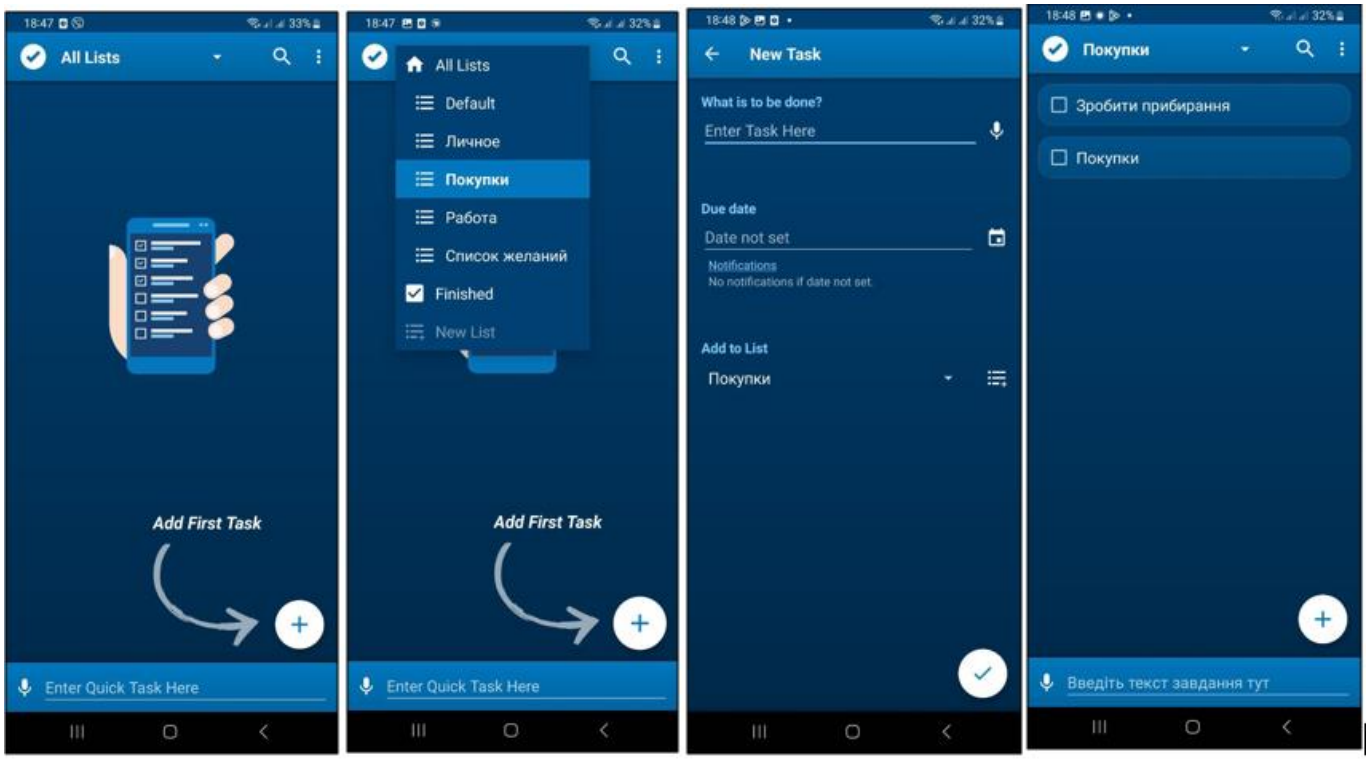


Рис. 1.5. Інтерфейс додатку «Splend Apps»

Проаналізувавши цей додаток, можу відмітити простоту в використанні. Задачі розподіляються по категоріям, можна швидко ввести задачу за допомогою голосового вводу, виконані задачі прибираються з головного списку, але їх можна проглянути у категорії «Finished».

Із недоліків: мова пристрою (в моєму випадку українська) не підтримується, тому все відображається некоректно, частина слів англійською а частина українською. Голосовий ввід працює некоректно, тому потреби в ньому, як такої, немає.

2. Наступним додатком буде «Блокнот, замітки та списки» від розробника KiteTech.

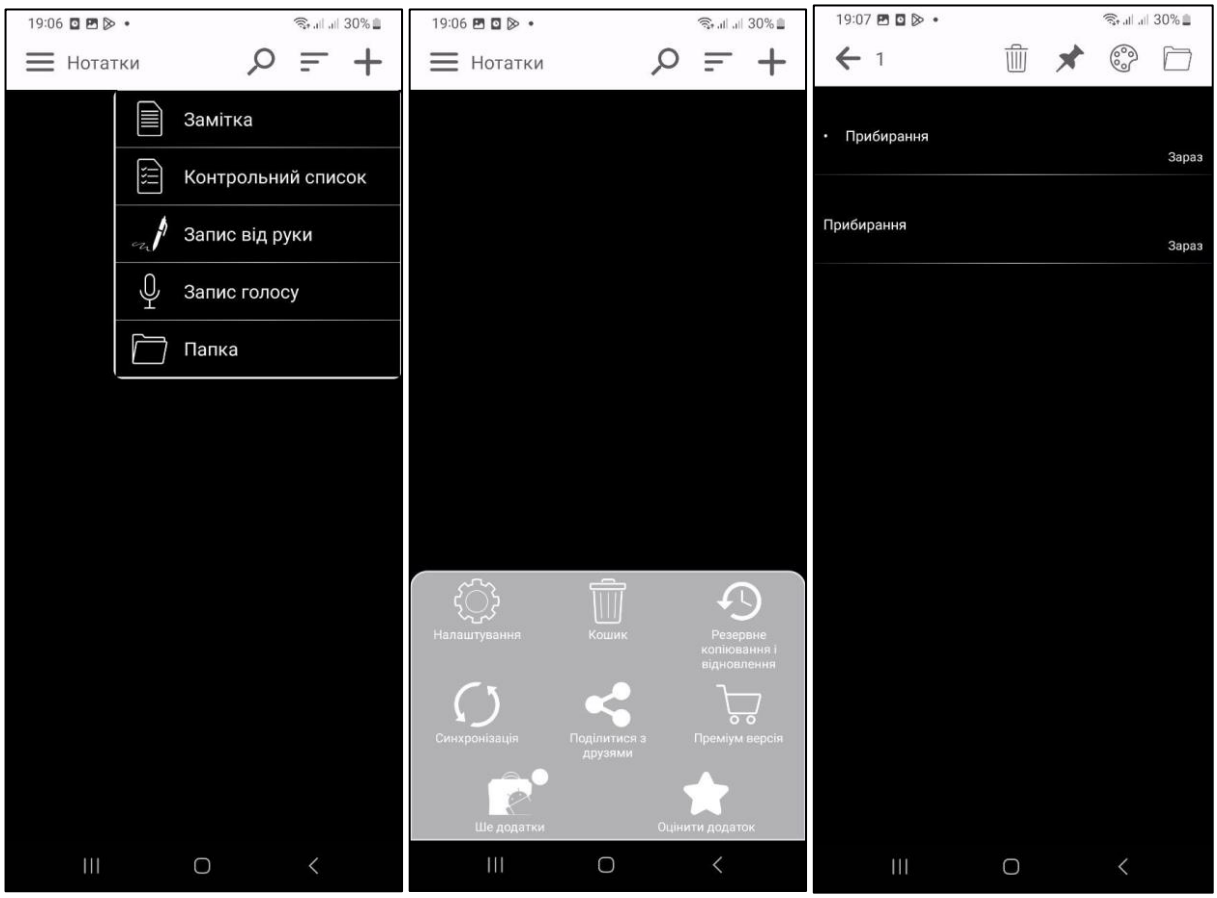


Рис. 1.6. Інтерфейс додатку «Блокнот, замітки та списки»

Проаналізувавши даний додаток, можу зазначити, додаток хоч і дуже примітивний, але незрозумілий та незручний. Я помітила проблеми з видаленням справ, також список справ і нотатки відображаються в одному полі, що є досить незручним для користувача. А ще є проблеми з рекламою, яка висвічується кожні 2 хвилини, і після її закриття додаток перестає коректно працювати.

3. Останнім додатком, який я обрала для аналізу, буде додаток «Список завдань і планувальник», від розробника Innim Mobile Exp. Цей додаток має середню оцінку 4,9 з 5, і багато позитивних відгуків.

Рис. 1.7. Інтерфейс додатку «Список завдань і планувальник»

Проаналізувавши даний додаток, можу відмітити інтерактив з користувачем, можна обрати асистента, можна зареєструватись і синхронізувати облікові записи. Також можна додати корисну звичку, щоб про неї не забути. Задачі можна розсортувати за терміном виконання, є календар де можна все переглянути. Задачі можна розсортувати за категоріями: робота, додому, покупки та інше, це досить зручно. Із недоліків, на мій погляд, багато різних категорій, що відволікає. Впершу чергу, в очі кидається категорії, а не самі задачі.

2.3. Уточнення вимог до системи та дизайн додатку

В рамках моєї дипломної роботи представлена інформаційна система, що надає можливість записувати задачі, як поля з заголовком та основним текстом. Проаналізувавши аналогічні додатки у пункті 2.2, для себе я обрала наступні вимоги:

Функціонал:

– Створення нотатки;

– Редагування нотатки;

– Закріплення нотатки;
– Відкріплення нотатки;
– Видалення нотатки.

Вимоги до оформлення інтерфейсу додатку:

– Нотатки повинні бути розділені по категоріям окркмими блоками;
– Слід використати для оформлення нейтральні, пастельні кольори;
– Кожен блок повинен мати різний колір;
– У кожної категорії нонатки, має бути строка про дату та час створення.
– Технічні вимоги:
– Мова програмування: java;
– Середовище розробки: Android Studio, з інструментом Android SDK;
– Операційна система та версія, для якої має бути розроблений додаток: від API:24 Android 7.0 (Nougat) до API:29 Android 10 (Quince Tart).

2.4. Створення прототипу додатку у сервісі Figma

Для більшого розуміння кінцевого результату, створимо прототип додатку, та підготуємо іконку додатку.

У сервісі Figma можна створити дизайн додатку для будь якого пристрою, виконавши такий алгоритм:

1. Створимо новий файл, та встановимо розмір полотна відповідно до розмірів бажаного мобільного пристрою, в моєму випадку це 5.0, як і в емуляторі Android Studio.
2. Розробимо макет мобільного додатка, використовуючи інструмент рамки Figma для створення окремих екранів, а інструмент форми для створення елементів інтерфейсу користувача, таких як кнопка «додати».
3. Використаємо інструмент для роботи з текстом, щоб додати тостові заголовки та основний тестовий текст до макету. За потреби також можна

імпортувати шрифти із зовнішніх джерел, але треба враховувати, які з них можна застосувати в Android Studio.

4. Найголовнішим етапом у роботі в Figma, буде визначити кольорове оформлення. Для цього створимо палітру кольорів, з існуючими Rgb кодами кольорів, для подальшого застосування їх у Android Studio, та застосовуємо її до елементів інтерфейсу макету.
5. Фінальний етап – створимо інтерактивний прототипи, додаючи зв'язки між кадрами та додаючи анімацію та переходи.

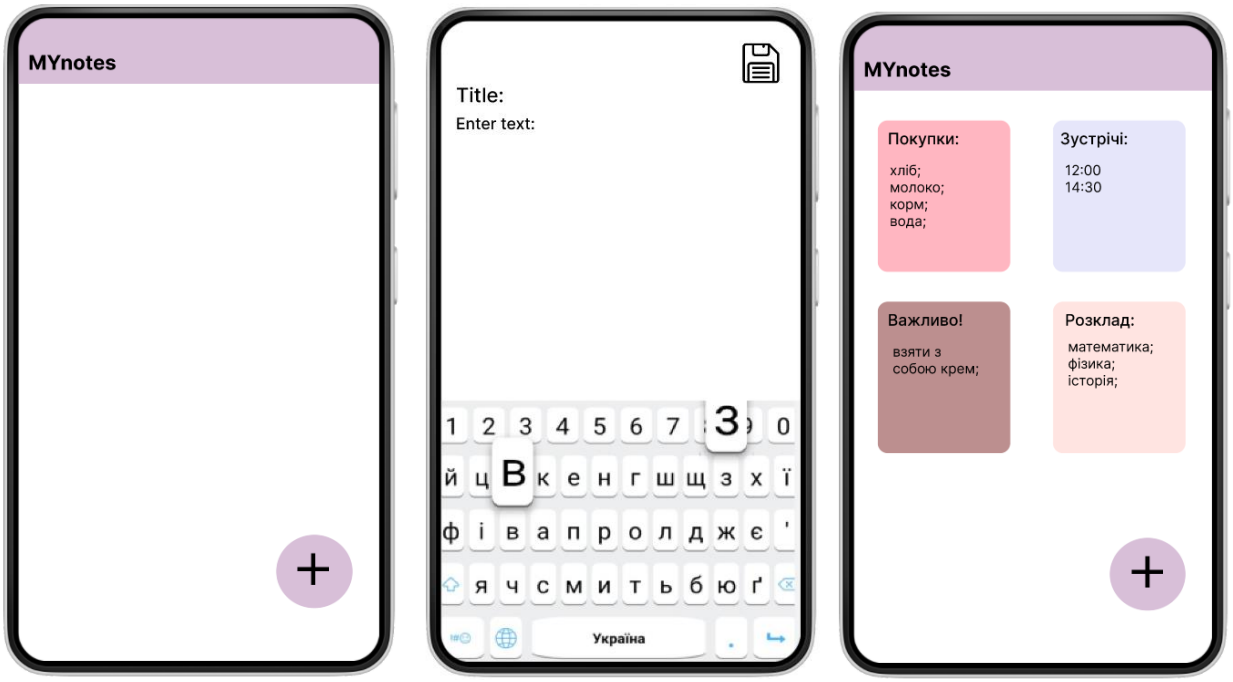


Рис.1.8. Прототип додатку для планування задач

Отже, у сервісі Figma я оформила 3 сторінки:

– Початковий екран без задач з кнопкою «Додати»;
– Активність «Додати нотатку»;
– Відображення списку задач, які зберігаються у окремих блоках.

Додати затачу можна буде натиснувши на кнопку «+». Задачі будуть розташовуватись у спеціальних полях-блоках. Зверху поля буде розташовуватись заголовок, в якому можна буде вказати групу, до якої задачі відносяться, наприклад: робота, продукти, домашні справи і тд. Під заголовком

розташовуватимуться самі пункти задач, які можна буде видалити разом з полем-блоком.

2.5. Створення піктограми для додатку

Під час створення піктограми для програми Android дуже важливо розробити візуально привабливий і ідентифікований символ, який представлятиме мету або сам бренд програми. Виділимо основні побажання до дизайну піктограми:

- Простота, значки мають бути чіткими та простими для сприйняття з першого погляду. Слід уникати перевантаження свого дизайну занадто великою кількістю деталей або складних візерунків.
- Важливо підтримувати узгодженість із мовою дизайну платформи Android. Потрібно проаналізувати та дізнатись про принципи матеріального дизайну, які включають пропозиції щодо дизайну піктограм, форм і розмірів.
- Унікальність, піктограма додатку має виділятися серед інших. Ідеальною є унікальна форма або колірна гамма, яка виділяється.
- Контраст, важливо щоб піетограма мала достатній контраст, щоб його було видно навіть на невеликих розмірах або на пристроях із низькою роздільною здатністю.
- Піктограми розроблено як візуальні ресурси, тому варто уникати додавання тексту до піктограми. Натомість, можна використати графіку або фігури, щоб висловити мету програми.

Отже, проаналізувавши побажання, піктограма додатку MYnotes матиме вигляд:

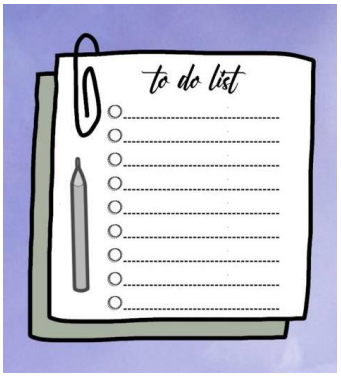


Рис.1.9. Піктограма додатку MYnotes

2.6. Висновок до розділу 2

В другому розділі було досліджено сервіс Google Play та його затребуваність користувачами Android, було проаналіовано аналогічні додатки у сервісі Google Play. Було уточнено вимоги до системи та дизайн додатку. Я виділила для себе такі вимоги до функціоналу: створення нотатки, редагування нотатки, закріплення нотатки, відкріплення нотатки та видалення нотатки. Було виділено такі вимоги до інтерфайсу: нотатки повинні бути розділені по категоріям окркмими блоками, слід використати для оформлення нейтральні, пастельні кольори, кожен блок повинен мати різний колір, у кожної категорії нонатки, має бути строка про дату та час створення. Технічною вимогою є версія додатку: від API:24 Android 7.0 (Nougat) до API:29 Android 10 (Quince Tart). Було створено прототип та піктограму додатку.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Основний алгоритм написання програми для створення нотаток

1. Створити новий проєкт Android.
2. Оформлення дизайну інтерфейса користувача відбувається у файлі «activity_main.xml», який відповідає активності за замовчуванням. Потрібно використати RecyclerView, щоб показати список нотаток, EditText для назв і тексту нотаток, кнопки для таких дій: як збереження, додавання, видалення та закріплення нотатки, і EditText для дій з нотатками.
3. Щоб представити нотатку, потрібно створити клас під назвою «Note». Який включить у себе будь-які додаткові поля, які потрібні, а також такі атрибути, як заголовок і вміст. Щоб отримати доступ і змінити ці атрибути, потрібно згенерувати геттери та сеттери.
4. Створення адаптера RecyclerView: цей адаптер відповідає за додавання даних приміток до RecyclerView. Щоб розгорнути макет для кожного елемента нотатки та прив’язати дані нотатки до перегляду, використовуються відповідні методи, такі як getItemCount(), onCreateViewHolder() і onBindViewHolder().
5. Для того щоб створити нотатку слід додати відповідний код, щоб керувати створенням нотаток до класу діяльності. Потрібно створити новий екземпляр класу Note і взяти дані користувача з полів EditText, слід помістити нотатку в базу даних або ArrayList як структуру даних.

6. Щоб реалізувати редагування та видалення приміток потрібно розширити логіку для редагування та видалення приміток. Користувач зможе змінити властивості нотатки після отримання вибраної нотатки зі структури даних, після чого структура даних оновиться.
7. Підключення елементів інтерфейсу користувача до коду вимагає пошуку необхідних елементів інтерфейсу користувача у класі активності за допомогою findViewById() і призначення їм змінних. Для кнопок або будь-яких інших інтерактивних елементів потрібно налаштувати Click, longClick. Слід створити логіку для виконання таких операцій, як додавання, редагування або видалення нотаток під час кожного контакту з цими компонентами.
8. Екземпляр RecyclerView у класі активності та метод findViewById(), використовують для того, щоб приєднати його до відповідного елемента макета, щоб нотатки могли відображатися в RecyclerView. Застосуємо адаптер, який створили раніше, і менеджер макета в RecyclerView та надамо адаптеру список приміток для відображення в RecyclerView.
9. Запуск додатка.

3.2. Архітектура програмного забезпечення

Архітектура додатку MYnotes представлена на рисунку 1.10.1 та 1.10.2:

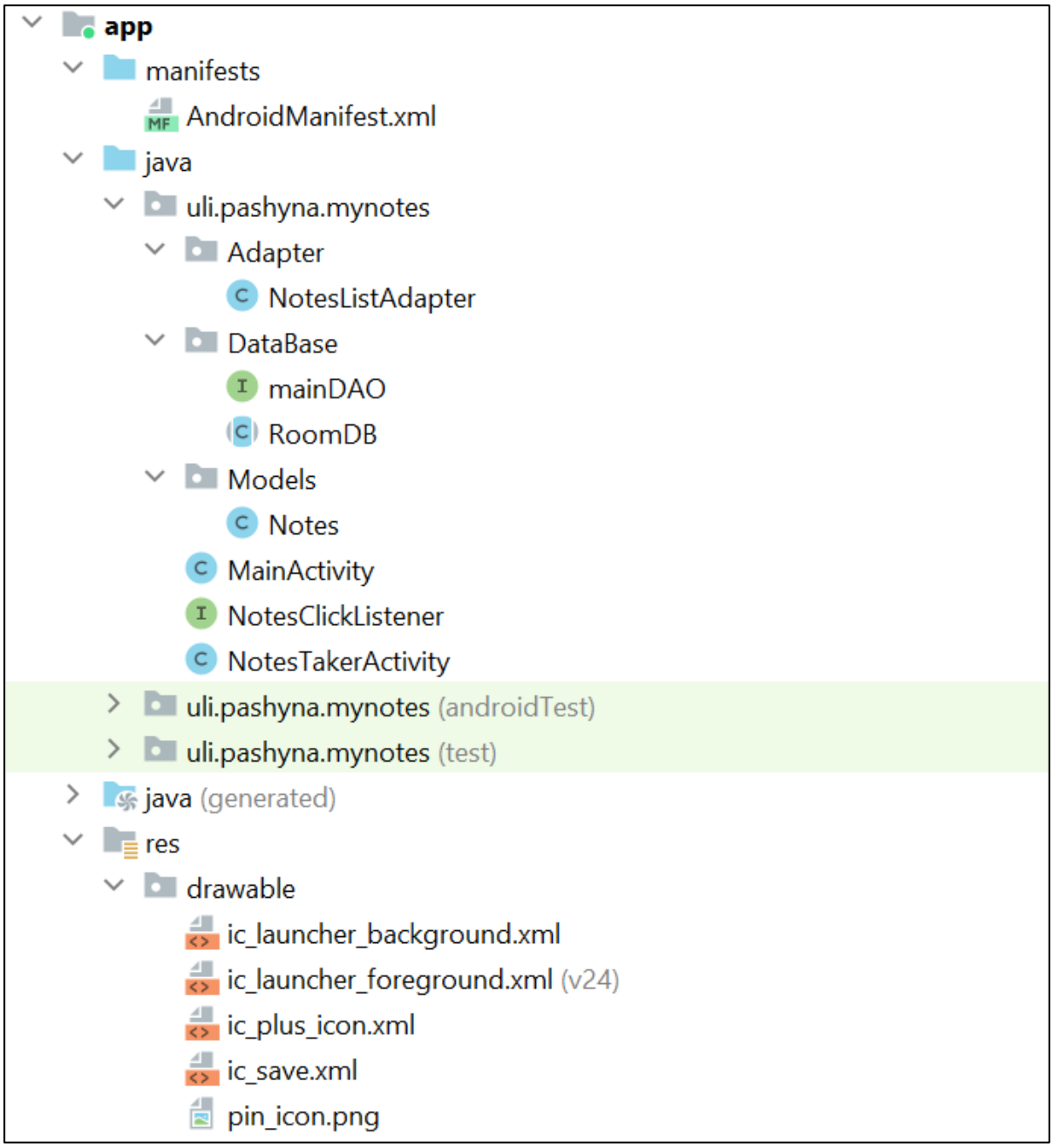


Рис. 1.10.1. Архітектура ПЗ.

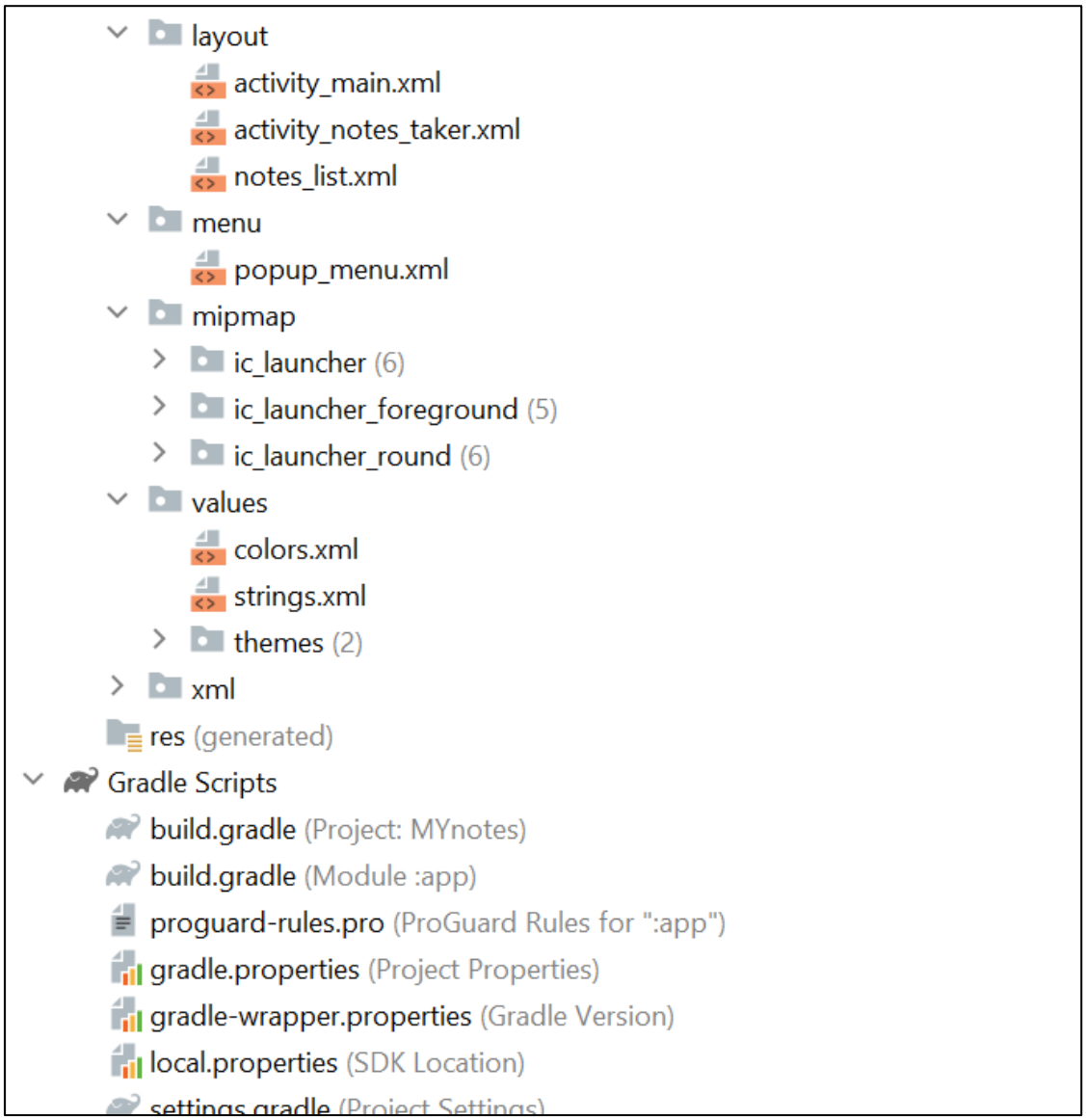


Рис. 1.10.2. Архітектура ПЗ (продовження)

Як представлено на рисунку 1.10.1 та 1.10.2 , додаток містить такі елементи:

1. AndroidManifest.xml – XML-файл, який надає основну інформацію про програму системи. В цьому файлі можна налаштувати такі можливості, як: ім'я Java-пакета програми, яка є унікальним ідентифікатором; Описати компоненти програми; вказати список необхідних дозволів для звернення до захищених частин API та взаємодії з іншими програмами; вказати мінімальний та максимальний рівень API Android, необхідний для роботи програми.

Структура файлу AndroidManifest.xml представлена на рисунку 1.11:

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="MYnotes"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MYnotes"
        tools:targetApi="31">
        <activity
            android:name=".NotesTakerActivity"
            android:theme="@style/Theme.MYnotes.NoActionBar"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Рис. 1.11. Структура файлу AndroidManifest.xml

Тег <manifest> – є головним тегом, в який вкладена вся конфігурація проєкту. А параметр xmlns:android визначає місце імен Android. Тег <application> – дозволяє встановити дозвіл на використання ресурсів системи або навпаки, заборонити використання компонентів програми.

Цей тег включає в себе такі основні атрибути як:

- android:allowBackup – що відповідає за створення резервної копії;
- android:dataExtractionRules – даний атрибут вказує правила, що визначають, які файли та каталоги можна копіювати з пристрою, в рамках операцій резервного копіювання;
- android:fullBackupContent - вказує на файл XML, який містить повні правила резервного копіювання для автоматичного резервного копіювання . Ці правила визначають, для яких файлів буде створено резервну копію;
- android:icon – піктограма для програми;
- android:label – ім’я додатку, яке відображається користувачеві;
- android:roundIcon – адаптивний значок;
- android:supportsRtl – вказує, чи можуть використовуатись спеціальні API для роботи з правосторонньою орієнтацією тексту. Наприклад, для таких мов як арабська;
- android:theme – встановлює тему додатку;
- tools:targetApi – даний атрибут вказує мінімальну варсію API, яка буде використана для ресурсів додатку.

Вкладені activity визначають всі елементи, що використовуються в додатку activity. На рисунку 1.11 видно, що в додатку є основна activity – MainActivity, та діяльність NotesTakerActivity, в якій вказана тема NoActionBar, яка призначена тільки для цієї activity.

Тег <intent-filter> – це фільтр наміру, який визначає тип намірів, які він приймає, на основі дії даних і категорій. Система передає неявний намір до компонента програми, лише якщо намір може пройти через один із фільтрів. В свою чергу, вкладений тег <category> додає назву категорії до фільтра намірів.

1. В пакеті Adapter міститься клас NotesListAdapter, код якого є реалізацією RecyclerView.Adapter для відображення списку нотаток. Цей адаптер використовується для додавання списку нотаток до RecyclerView разом із

заголовком, вмістом, датою та додатковою pin image. Через інтерфейс NotesClickListener він додатково пропонує можливість «клацання» та «тривалого клацання».

Розглянемо клас NotesListAdapter та методи цього класу більш детальніше:

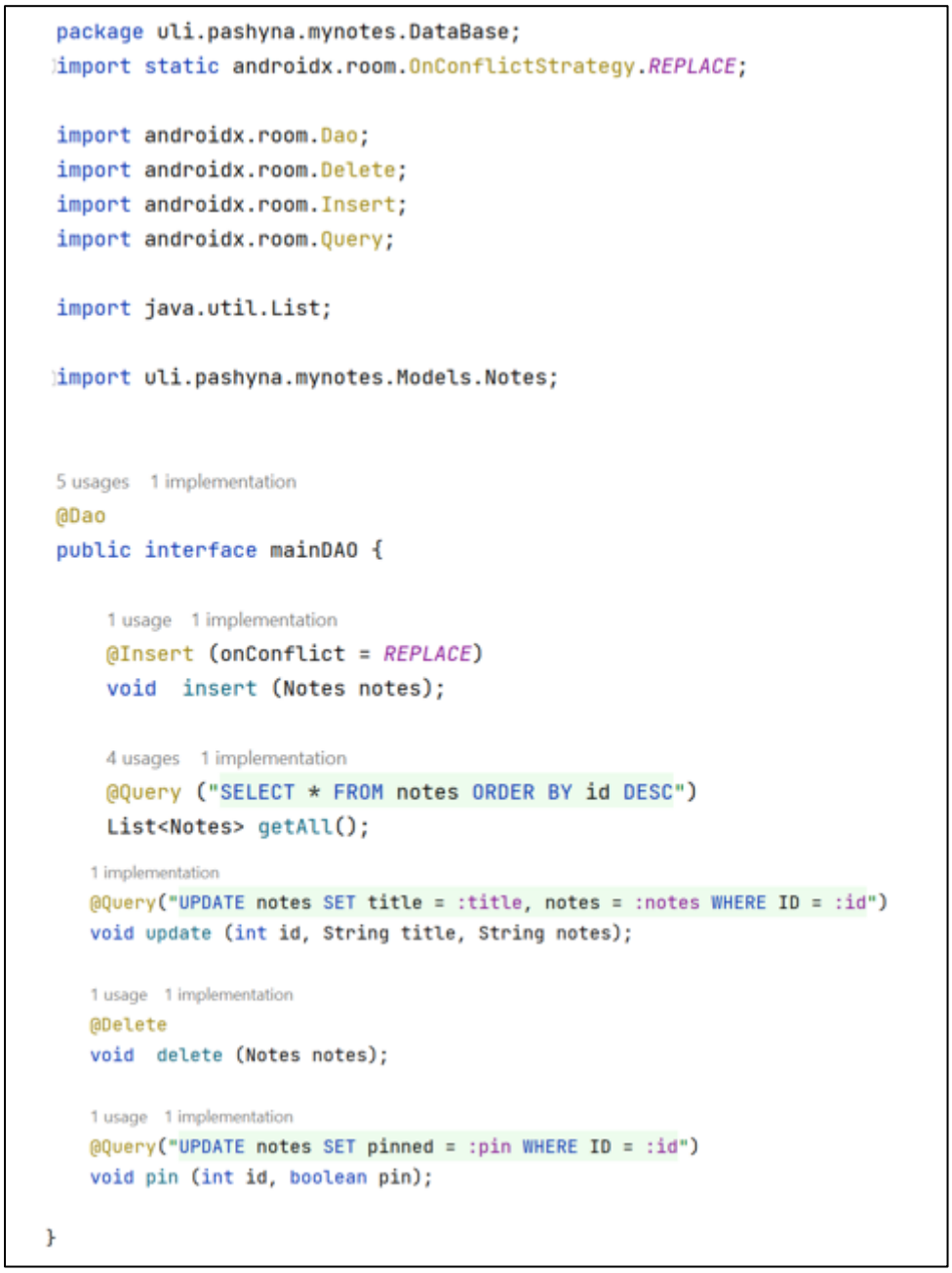
- Адаптер визначає параметр типу NotesViewHolder, унікальний клас ViewHolder, визначений у адаптері, і наслідує клас RecyclerView.Adapter.
- Клас «NotesListAdapter» має три змінні-члени: «context» (типу «Context»), «list» (типу «List<Notes>») і «listener» (типу «NotesClickListener»).
- Макет для кожного елемента в RecyclerView розповсюджується функцією onCreateViewHolder. Використовуючи LayoutInflater, він надуває файл макета під назвою notes_list.xml.
- Метод onBindViewHolder RecyclerView відповідає за прив’язку даних до відображення кожного елемента. Відповідні дані встановлюються для представлень після того, як вони вилучають об’єкт Notes зі списку у вказаному місці.
- Усередині «onBindViewHolder» дані з об’єкта «Notes» у поточній позиції витягуються та встановлюються для відповідних переглядів у «NotesViewHolder».
- Метод getRandomColor використовується всередині методу onBindViewHolder для створення випадкового кольору фону для кожного елемента нотатки. Колір було встановлено як колір тла notes_conteiner CardView.
- Щоб обробляти події «клацання» елемента, у notes_container налаштовано слухачі(listeners) «клацань» і «тривалих клацань». Методи інтерфейсу NotesClickListener викликаються слухачами(listeners), які також передають відповідний об’єкт Notes слухачу.
- Функція getRandomColor надає код кольору, після випадкового вибору кольору з колекції попередньо визначених колірних ресурсів.

Код класу NotesListAdapter знаходиться у додатках.

2. В пакеті DataBase знаходиться інтерфейс mainDAO та клас RoomDB.

А) Інтерфейс mainDAO: код є прикладом інтерфейсу Data Access Object (DAO) для бази даних Room в Android. Він визначає кілька методів для виконання операцій бази даних над таблицею під назвою "notes" за допомогою бібліотеки Room. Розглянемо код інтерфейсу mainDAO більш детально:

- Анотація «@Dao» позначає інтерфейс як об’єкт доступу до даних;
- Анотація «@Insert» вказує, що цей метод використовується для вставки даних у таблицю «notes»;
- «onConflict = REPLACE» вказує на те, що якщо виникає конфлікт (наприклад, нотатка з таким же ідентифікатором уже існує), існуючу нотатку буде замінено новою;
- Анотація «@Query» дозволяє користувальницькі запити SQL. Цей метод отримує всі нотатки з таблиці «notes» і повертає їх у вигляді списку, упорядкованого за стовпцем «id» у порядку спадання;
- Метод update використовує інструкцію SQL UPDATE для зміни певної нотатки, визначеної її ідентифікатором. Він оновлює стовпці «заголовок» і «нотатки» наданими значеннями;
- Анотація «@Delete» вказує на те, що метод delete використовується для видалення нотатки з таблиці «notes»;
- Метод delete очікує об’єкт «Notes» як параметр, і він видалить примітку, яка відповідає даному об’єкту на основі його первинного ключа;
- Метод pin оновлює «закріплений» стовпець певної нотатки, визначеної її ідентифікатором. Він встановлює «закріплене» значення на надане логічне значення.



```java
package uli.pashyna.mynotes.DataBase;
import static androidx.room.OnConflictStrategy.REPLACE;

import androidx.room.Dao;
import androidx.room.Delete;
import androidx.room.Insert;
import androidx.room.Query;

import java.util.List;

import uli.pashyna.mynotes.Models.Notes;


5 usages  1 implementation
@Dao
public interface mainDAO {

    1 usage  1 implementation
    @Insert (onConflict = REPLACE)
    void  insert (Notes notes);

    4 usages  1 implementation
    @Query ("SELECT * FROM notes ORDER BY id DESC")
    List<Notes> getAll();

    1 implementation
    @Query("UPDATE notes SET title = :title, notes = :notes WHERE ID = :id")
    void update (int id, String title, String notes);

    1 usage  1 implementation
    @Delete
    void  delete (Notes notes);

    1 usage  1 implementation
    @Query("UPDATE notes SET pinned = :pin WHERE ID = :id")
    void pin (int id, boolean pin);

}
```

Рис. 1.12. Структура інтерфейсу mainDAO

Отже, ці методи визначають основні операції з базою даних (вставлення, оновлення, видалення) і користувальницькі запити, необхідні для взаємодії з таблицею «notes» у базі даних Room.

Б) Клас RoomDB: код представляє конфігурацію бази даних Room для програми Android. База даних використовується для зберігання та отримання екземплярів класу «Notes».

Розглянемо клас RoomDB більш детально:

- Основним класом для взаємодії з базою даних Room є RoomDatabase, який наслідується класом RoomDB.
- Об’єкти бази даних і версія вказуються за допомогою анотації @Database. Сутністю в цьому екземплярі є клас Notes, а для версії бази даних встановлено значення 1. Схему не буде експортовано до папки, оскільки для аргументу exportSchema встановлено значення false.
- Оскільки неможливо відразу створити екземпляр класу RoomDB, він оголошується як абстрактний. Будучи єдиним екземпляром класу RoomDB, змінна бази даних забезпечує створення лише одного екземпляра бази даних.
- Щоб отримати екземпляр класу RoomDB, використовується метод getInstance(). Він дотримується шаблону singleton для створення екземпляра бази даних, якщо він не існує. Він використовує метод Room.databaseBuilder() для створення нового екземпляра бази даних.
- Метод context.getApplicationContext() використовується для отримання контексту програми.
- Метод Room.databaseBuilder() приймає три аргументи: середовище програми, клас бази даних (RoomDB.class) і назву бази даних (DATABASE_NAME).
- allowMainThreadQueries() використовується, щоб дозволити операції з базою даних у головному потоці.
- Міграцією бази даних керує fallbackToDestructiveMigration(). Якщо розширити версію схеми, буде реконструйовано базу даних і видалено всі поточні дані.
- Примірник бази даних Room створюється шляхом виклику методу construct().

- Об’єкт доступу до даних (DAO) для роботи з базою даних оголошується абстрактною функцією mainDAO mainDao(). Інтерфейс mainDAO необхідно визначати окремо.

```
package uli.pashyna.mynotes.DataBase;
import android.content.Context;
import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;
import uli.pashyna.mynotes.Models.Notes;
7 usages  1 inheritor
@Database(entities = {Notes.class},version = 1,exportSchema = false)
public abstract class RoomDB extends RoomDatabase {
    3 usages
    private static RoomDB database;
    1 usage
    private static String DATABASE_NAME = "NoteApp";

    1 usage
    public synchronized static RoomDB getInstance(Context context){
        if (database == null) {
            database = Room.databaseBuilder(context.getApplicationContext(),
                            RoomDB.class, DATABASE_NAME)
                    .allowMainThreadQueries()
                    .fallbackToDestructiveMigration()
                    .build();
        }
        return database;
        8 usages  1 implementation
        public abstract mainDAO mainDao();
    }
```

Рис. 1.13. Структура класу RoomDB

Загалом, цей код налаштовує конфігурацію бази даних Room, надає екземпляр Singleton бази даних і визначає абстрактний метод доступу до об’єкта доступу до даних (DAO) для виконання операцій бази даних над сутністю «Notes».

3. Пакет Models містить у собі клас Notes, клас MainActivity, інтерфейс NotesClickListener, та клас NotesTakerActivity.

А) Клас Notes: представляє об’єкт «Notes» з полями для ідентифікатора, заголовка, вмісту, дати та доданого прапорця. Він призначений для використання в поєднанні з бібліотекою Room для виконання операцій з базою даних SQLite.

Розглянемо Клас Notes більш детально:

- Анотація «@Entity» бібліотеки збереження Room вказує, що цей клас представляє сутність у таблиці бази даних SQLite;
- Таблиця бази даних, де зберігатимуться екземпляри цього класу, позначається «tableName = "notes"»;
- «implements Serializable» означає, що екземпляри класу Notes можна перетворити на потоки байтів для зберігання або передачі, вказуючи, що клас серіалізується;
- Анотація з Room «@PrimaryKey» визначає поле ID як первинний ключ таблиці;
- Значення первинного ключа будуть створені базою даних автоматично, якщо вказано «autoGenerate = true». База даних надасть кожному новому об’єкту Notes спеціальний ідентифікатор, коли він додасться до таблиці;
- «int ID = 0» Оголошує та ініціалізує ідентифікатор цілочисельного поля зі значенням 0. Представляє первинний ключ таблиці;
- Анотація з Room під назвою «@ColumnInfo» описує особливості стовпця таблиці, основного тексту та дати;
- До приватних полів класу Notes можна отримати доступ і змінити їх поза межами класу за допомогою методів getter і setter(getID,setID; getTitle, setTitle, getNotes, setNotes; getDate, setDate; setPinned, isPinned), які можна згенерувати автоматично. Вони пропонують контрольований доступ до полів і захищають цілісність даних, дотримуючись стандартної норми Java для інкапсуляції.

Б) клас mainActivity**:** загалом, цей клас - основна діяльність, яка є початковим екраном, який з’являється, коли користувач запускає програму. Код представляє клас активності Android MainActivity, що наслідує AppCompatActivity, який також реалізує PopupMenu.OnMenuItemClickListener.

- Для керування даними нотаток код визначає низку змінних-членів, зокрема RecyclerView, FloatingActionButton, NotesListAdapter, RoomDB, ListNotes> і об’єкти Notes.
- Встановлено макет, налаштовано перегляди та зроблено запит до бази даних, щоб отримати нотатки в методі onCreate. Нотатки, які було отримано, також оновлюються в RecyclerView.
- Відповідно до пошукового запиту метод фільтра призначений для фільтрації нотаток.
- Результати, пов’язані із запуском NotesTakerActivity, обробляються функцією onActivityResult. Якщо результат RESULT_OK, RecyclerView змінюється, а до бази даних додається нова нотатка.
- Функція updateRecycler додає NotesListAdapter і налаштовує RecyclerView за допомогою StaggeredGridLayoutManager.
- Для обробки подій «клацання» та «тривалого клацання» в нотатках реалізовано клас NotesClickListener, а його методи за замовчуванням змінено.
- Коли нотатку довго клацають, метод showPopup створює та відображає PopupMenu. Popup_menu.xml використовується для розширення елементів меню.
- Вибір пунктів меню зі PopupMenu обробляється функцією onMenuItemClick. RecyclerView оновлюється, а закріплений стан бази даних для нотатки змінюється, якщо вибраний елемент є шпилькою. Якщо вибраний елемент видаляється, RecyclerView оновлюється, а примітка видаляється з бази даних.

  Код класу MainActivity знаходиться у додатках.

В) інтерфейс NotesClickListener: код цього інтерфейсу представляє собою публічний інтерфейс NotesClickListener. Цей інтерфейс визначає три методи:

- Коли подія «клацання» відбувається на об’єкті Notes, викликається метод onClick(Notes notes). Реалізація цього методу повинна обробляти подію «клацання» та виконувати потрібні дії.

- Коли подія «тривалого клацання» відбувається на об’єкті Notes разом із пов’язаним CardView, який виступає замість елемента інтерфейсу користувача, викликається метод onLongClick(Notes notes, CardView cardView). Подія «тривалого клацання» має бути оброблена реалізацією методу, і необхідно виконати певні дії.
- Використання notes_container як імені аргументу замість cardView, onLongClick(Notes notes, CardView notes_container) є альтернативним оголошенням методу onLongClick.

```
package uli.pashyna.mynotes;

import androidx.cardview.widget.CardView;

import uli.pashyna.mynotes.Models.Notes;

5 usages   1 implementation
public interface NotesClickListener {

    1 usage   1 implementation
    void onClick (Notes notes);

    1 usage   1 implementation
    void onLongCLick(Notes notes, CardView notes_conteiner);

}
```

Рис.1.14. Структура інтерфейсу NotesClickListener

Загалом, цей інтерфейс визначає зворотні виклики для обробки подій «клацання» та «тривалого клацання» на об’єктах Notes, а також пов’язаних елементів інтерфейсу користувача, представлених CardView.

Г) клас NotesTakerActivity: код є класом активності Android під назвою NotesTakerActivity. Він дозволяє користувачам робити нотатки, вводячи заголовок і опис, а потім зберігати нотатку.

Розглянемо код цього класу крок за кроком:

1) Клас наслідує AppCompatActivity, базовий клас для дій, які використовують функції панелі дій бібліотеки підтримки;
2) Клас визначає низку змінних-членів:
   - включаючи перегляди EditText editText_title та editText_notes, які використовуються для введення заголовка та приміток відповідно;
   - ImageView використовується як кнопка збереження та називається imageView_save;
   - Якщо дія оновлює наявну нотатку або генерує нову, логічне значення встановлюється на isOldNote.
3) XML-файл макета activity_notes_taker.xml розповсюджується у функції onCreate, щоб налаштувати макет для активності;
4) За допомогою відповідних ідентифікаторів подання editText_title і editText_notes ініціалізуються шляхом пошуку відповідних подання в макеті;
5) Подання imageView_save також ініціалізується шляхом пошуку відповідного йому подання макета за допомогою його ідентифікатора;
6) Код намагається отримати серіалізований об’єкт Notes із наміру за допомогою ключа «old_note». У разі успіху він встановлює заголовок і примітки в полях editText_title і editText_notes відповідно, а isOldNote встановлює значення true. Якщо пошук не вдається, виняток перехоплюється та друкується;
7) Представлення imageView_save має OnClickListener, налаштований для обробки подій «клацання»;
8) Рядки заголовка та опису отримуються з відповідних переглядів EditText і розміщуються всередині функції onClick;
9) Щоб обробити подію клацання, OnClickListener встановлюється в поданні imageView_save;

10) Рядки заголовка та опису беруться з пов’язаних переглядів EditText у функції onClick;
11) Якщо опис порожній, перед завершенням процедури відображається повідомлення, що сповіщає користувача написати опис;
12) Поточні дата й час форматуються за допомогою SimpleDateFormat;
13) Поточні дата й час отримуються шляхом створення об’єкта Date;
14) Якщо isOldNote має значення false і нотатка не є старою, створюється новий об’єкт Notes;
15) За допомогою коду результату RESULT_OK генерується намір зберегти об’єкт примітки, а потім повертається до дії виклику за допомогою setResult;
16) Повернення до попередньої дії після завершення NotesTakerActivity.

Код клас NotesTakerActivity знаходиться у додатках.

Отже, користувач може ввести назву та опис нотатки, зберегти її, а потім повернути об’єкт нотатки до виклику діяльності.

4. Пакет uli.pashyna.mynotes(androidTest) містить у собі клас ExampleInstrumentedTest.

Загалом, інструментальні тести використовують на реальних або віртуальних пристроях Android. Вони можуть скористатися перевагами API фреймворку Android. Тому, незважаючи на те, що вони працюють значно повільніше, ніж локальне тестування, інструментальні тести пропонують більшу точність. Їх слід використовувати лише в ситуаціях, коли необхідно перевірити поведінку справжнього пристрою. За потреби AndroidX Test пропонує низку бібліотек, які спрощують створення інструментальних тестів.

```
import ...

/**
 * Instrumented test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test
    public void useAppContext() {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
        assertEquals( expected: "uli.pashyna.mynotes", appContext.getPackageName());
    }
}
```

Рис. 1.15. Код класу ExampleInstrumentedTest.

Цей тест перевіряє, чи програма, що тестується, має очікувану назву пакета повністю. Тест буде невдалим, якщо назву пакета буде змінено, показуючи, що контекст не відповідає очікуванням.

5. Пакет uli.pashyna.mynotes(androidtest) містить у собі клас ExampleUnitTest.

На відміну від використання пристрою або емулятора Android, локальний тест виконується на самому комп'ютері. У результаті він запускає тести на локальній віртуальній машині Java (JVM) комп'ютера, а не на пристрої Android. Можна швидше оцінити логіку своєї програми завдяки локальному тестуванню, але типи тестів, які можна запускати, обмежені, оскільки неможливо взаємодіяти з інфраструктурою Android.

Ці обидва класи я не використовувала, при розробці свого додатку, тому не вважаю доцільним аналізувати ці класи більш детально. Оскільки можна проаналізувати логіку програми за допомогою локальних модульних тестів, це буде швидше.

6. Пакет res – містить структуру папок ресурсів програми. Містить такі папки:

А) папка drawable;

Для зображень. Зображення розраховані на відповідні роздільну здатність екрана мобільного пристрою.

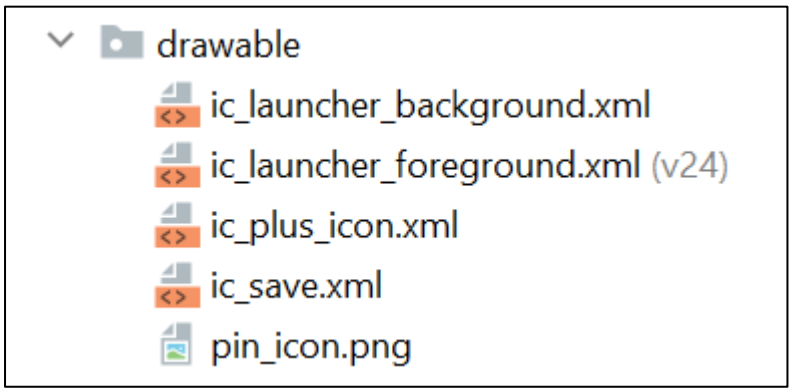


Рис. 1.16. Вміст папки drawable.

Як показано на рисунку 1.16, в даній папці зберігаються такі файли:

- ic_launcher_background.xml – цей xml-файл відповідає за фонове зображення головної іконки додатку;
- ic_launcher_foreground.xml – цей xml-файл відповідає за векторне зображення головної іконки додатку;
- ic_plus_icn.xml – векторне зображення на кнопці «додати», яка знаходиться на головному екрані додатку;
- ic_save.xml – векторне зображення кнопки збереження;
- pin_icon.png – зображення головної іконки додатку.

Б) папка layout;

Для XML-файлів компонування (компонування графічних елементів керування/розмітка).

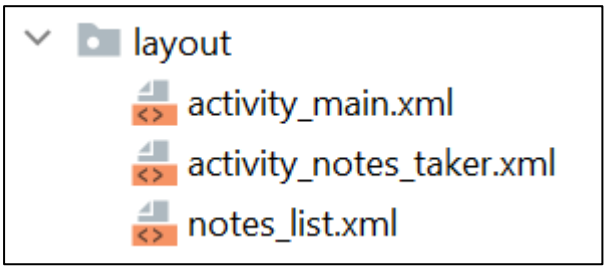


Рис. 1.17. Вміст папки layout.

Як показано на рисунку 1.17, ця папка містить такі XML-файли:

- activity_main.xml – що відповідає за компонування графічних елементів управління головною активності.
- activity_notes_taker.xml – компонування графічних елементів управління активності додавання нотатки;
- notes_list.xml – компонування графічних елементів керування елемента у списку Text View, який описаний у головній активності.

В) папка menu: містить у собі файл popup_menu.xml, що являє собою частину файлу ресурсів меню.

Рис. 1.18. Події у файлі popup_menu.xml.

Г) папка mipmap: зберігає у собі графічні файли, які використовуються для іконок програми під різні роздільні здатності екранів.

Д) папака values: призначена для зберігання ресурсів (констант) різних типів:

- colors.xml – містить у собі кольори, які використані у коді;
- strings.xml – зберігає всі строкові ресурси;
- styles.xml – зберігає стилі програми.

7. Gradle Scripts: Gradle – це потужна система збірки в Android Studio, яка використовується для автоматизації створення, тестування та упаковки програм Android. Файли конфігурації для проекту Android називаються файлами build.gradle, які також називаються сценаріями Gradle. Проект Android зазвичай містить два різних типи сценаріїв Gradle: build.gradle на рівні проекту та build.gradle на рівні модуля.

Build.gradle для проекту: цей файл, який можна знайти в кореневому каталозі проекту, використовується для налаштування параметрів проекту. Він визначає, як повинен бути побудований кожен модуль у вашому проекті.

Конструкція на рівні модуля. Gradle: цей файл, який використовується для налаштування параметрів, що стосуються кожного модуля (наприклад, модуля програми), знаходиться в каталозі кожного модуля. Він визначає параметри компіляції, залежності та інші параметри, що стосуються модуля.

```
implementation "androidx.room:room-runtime:2.5.1"
annotationProcessor "androidx.room:room-compiler:2.5.1"
```

Рис. 1.19. Що було додано у build.gradle(:app)

Як показано на рисунку 1.19, до файлу build.gradle(:app) було додано реалізацію «androidx.room:room-runtime:2.5.1», за допомогою цього рядка до проєкту додається бібліотека середовища виконання Room. Вона містить класи, необхідні для взаємодії з Room, включаючи міграції, сутності, DAO та базу даних.

«androidx.room:room-compiler:2.5.1», що додає до проєкту процесор анотацій Room. На основі анотацій, які застосовуються у класах, зв’язаних із Room, наприклад @Database, @Entity та @Dao, процесор анотацій створює необхідний код. Реалізація інтерфейсів DAO і класи, пов’язані з базою даних, які включені до створеного коду.

3.3. Алгоритм роботи усієї програми

Отже, для підведення підсумків про роботу програми, створимо загальну блок-схему використання додатку.

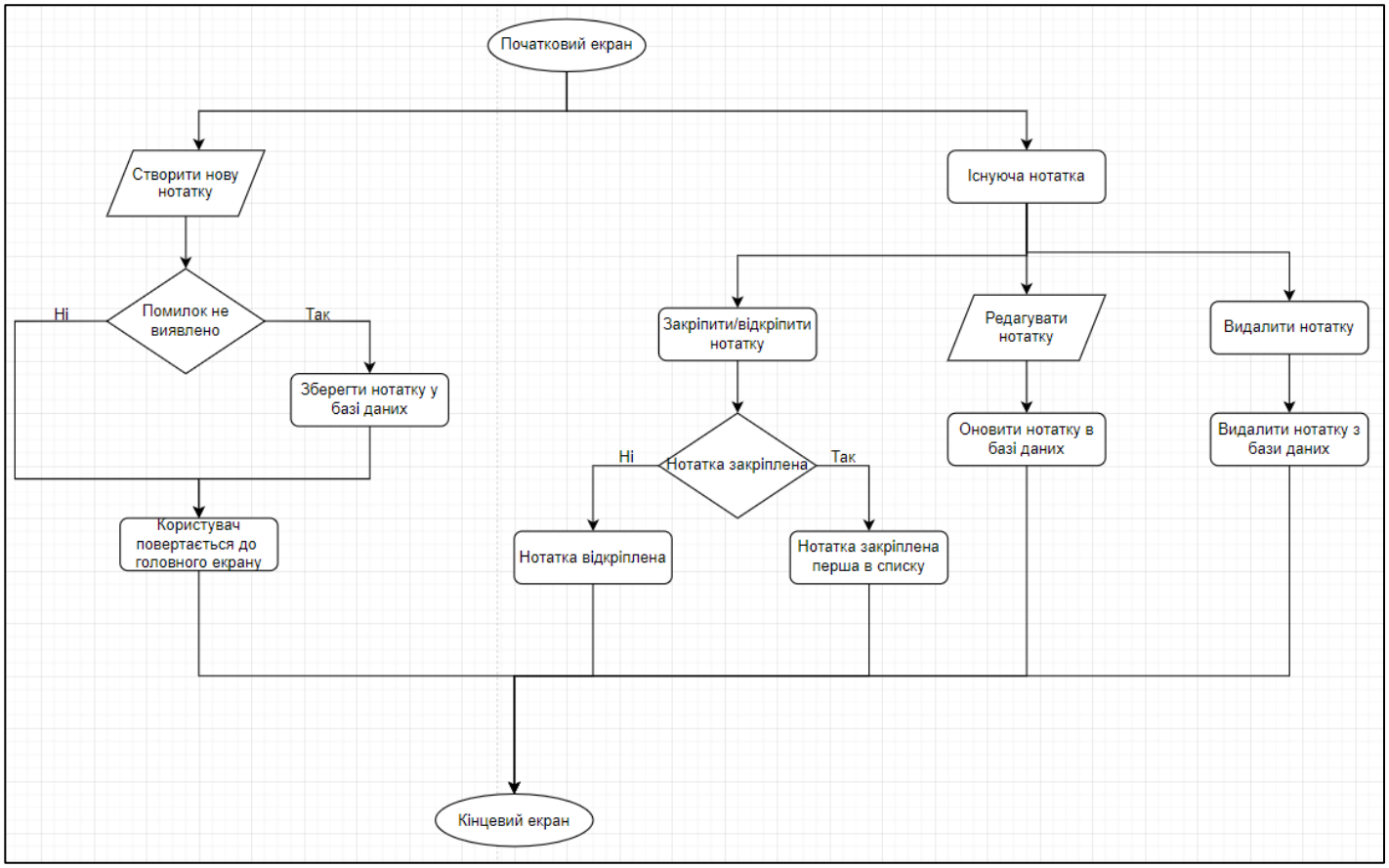


Рис. 1.20. Алгоритм роботи усієї програми

Як показано на рисунку 1.20, представлені наступні кроки роботи:

- На «Початковому екрані» користувач може вибирати з двох варіантів, зокрема додавати нову нотатку або вибирати наявну;
- Екран «Створити нову нотатку» відображається після того, як користувач вибере «Нова нотатка», де він може ввести вміст нотатки. Після завершення нотатка зберігається в базі даних;
- Якщо користувач вибирає «Існуючу нататку», йому буде показано список доступних приміток. Нотатку можна закріпити/відкріпити та видалити;

- Користувач може змінити текст нотатки, яка потім зберігається назад до бази даних. Нотатка видаляється з бази даних, або нотатка стає першою/останньою в списку.

3.4. Тестування програмного забезпечення

При першому завантаженні програми з'являється головна активність із порожнім списком нотаток. Ця активність представлена малюнку 1.21:

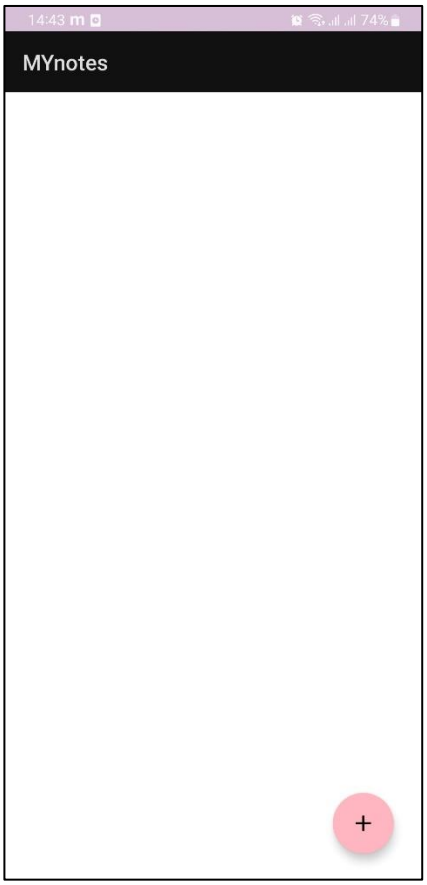


Рис. 1.21. Додаток при першому запуску

Щоб додати нотатку, потрібно натиснути на кнопку «+». Відбувається перехід до активності редактора нотатки:

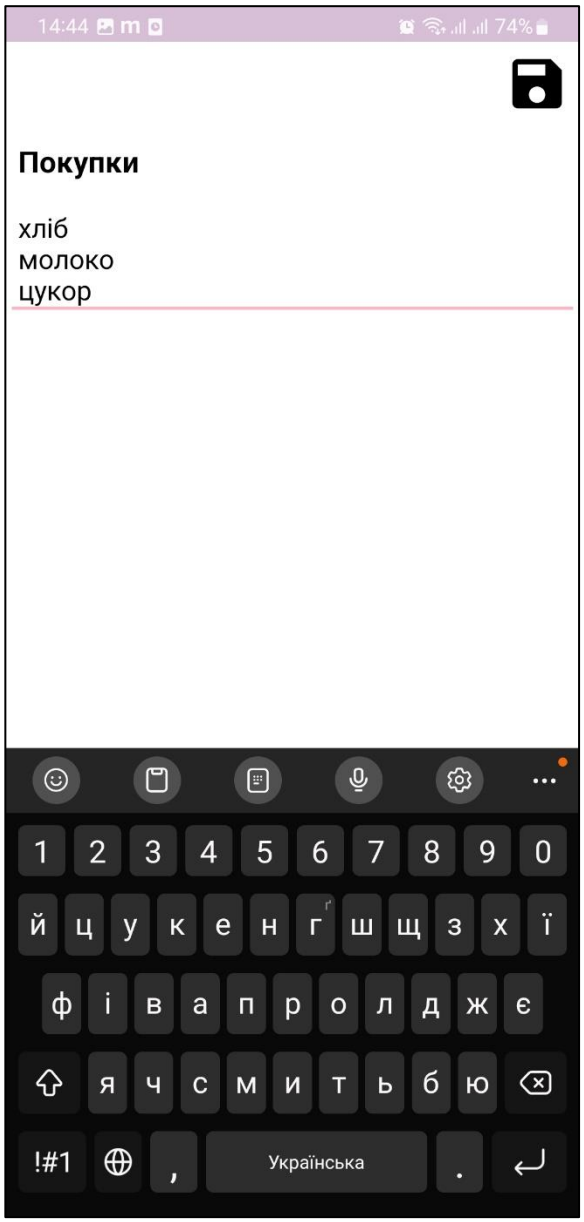


Рис. 1.22. Активність редактора нотатки

Якщо поля заповнені, то нотатка зберігається у основній активності. Якщо обдва поля не заповнені, то висвічується підсказка «Please, enter description».

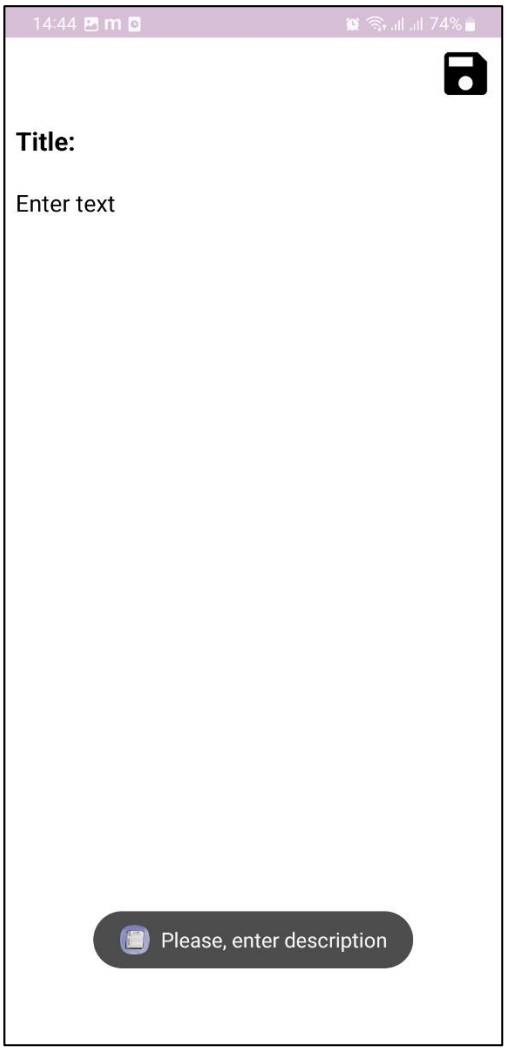


Рис. 1.23. Підсказка «Please, enter description»

Після успішного створення, всі нонатки висвічуються у головній активності, як показано на рисунку 1.24.

Дії з готовими нотаткам: нотатку можна повторно редагувати утримаючи потрібне поле декілька секунд, показано на рисунку 1.25; нотатку можна закріпити, натиснувши на поле, якщо все вірно то висвітиться підсказка «Pinned»; нотатку можна видалити обравши в меню delete, як показано на рисунку 1.27

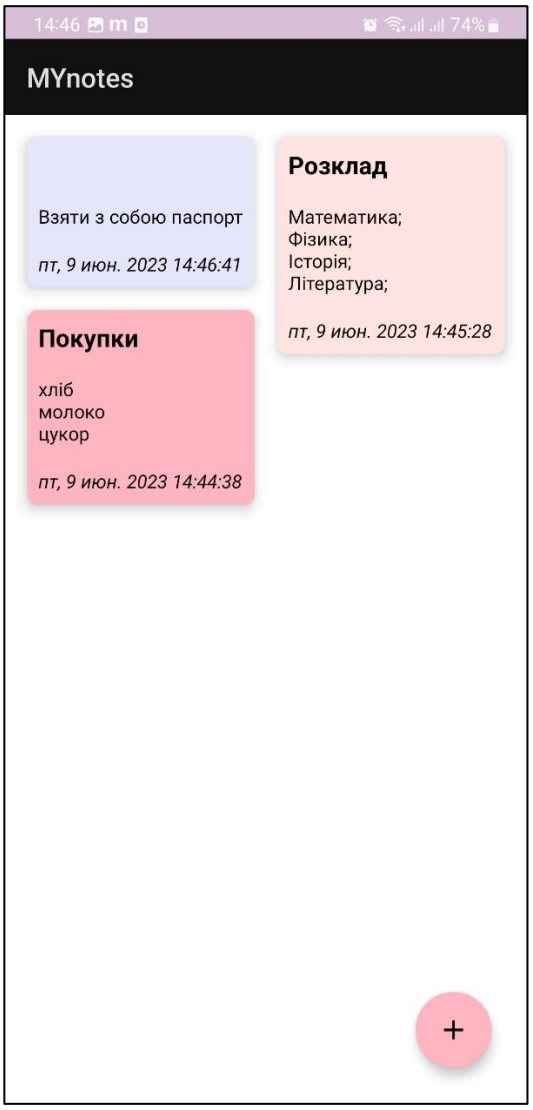


Рис. 1.24. Список нотаток після створення нової нотатки

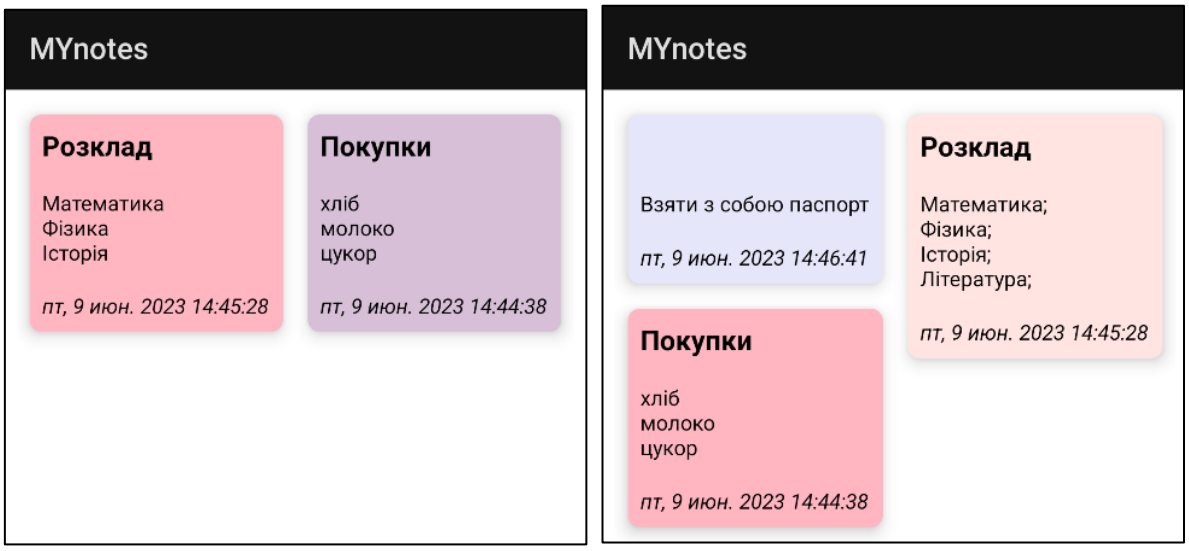


Рис. 1.25. Редагування нотатки: до та після

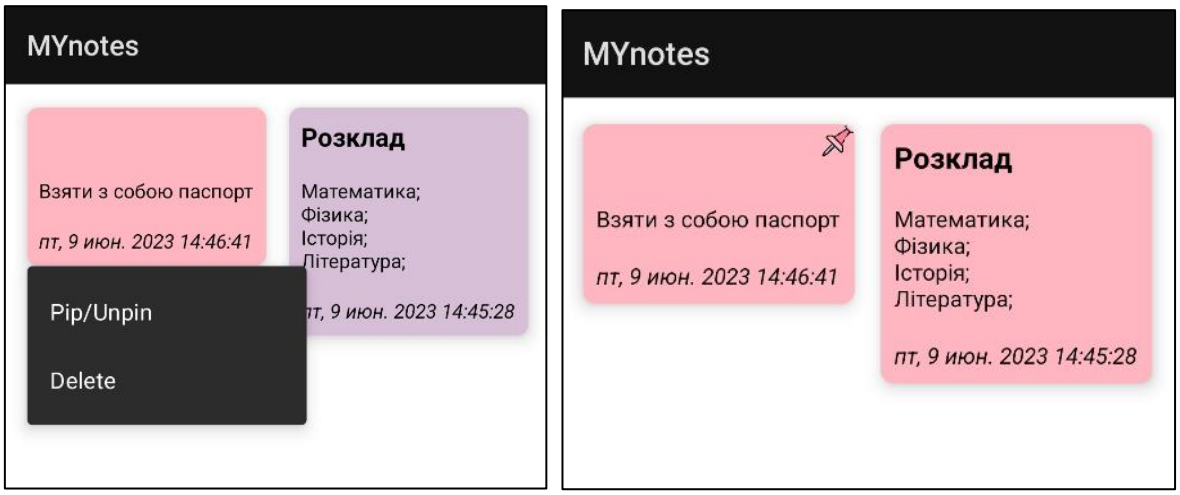


Рис. 1.26. Закріплення нотатки

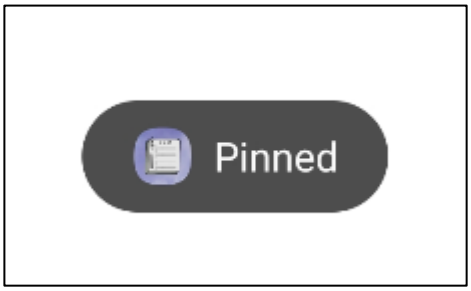


Рис. 1.26.1. Підсказка при успішному закріпленні

Рис. 1.26.2. Підсказка при відкріпленні

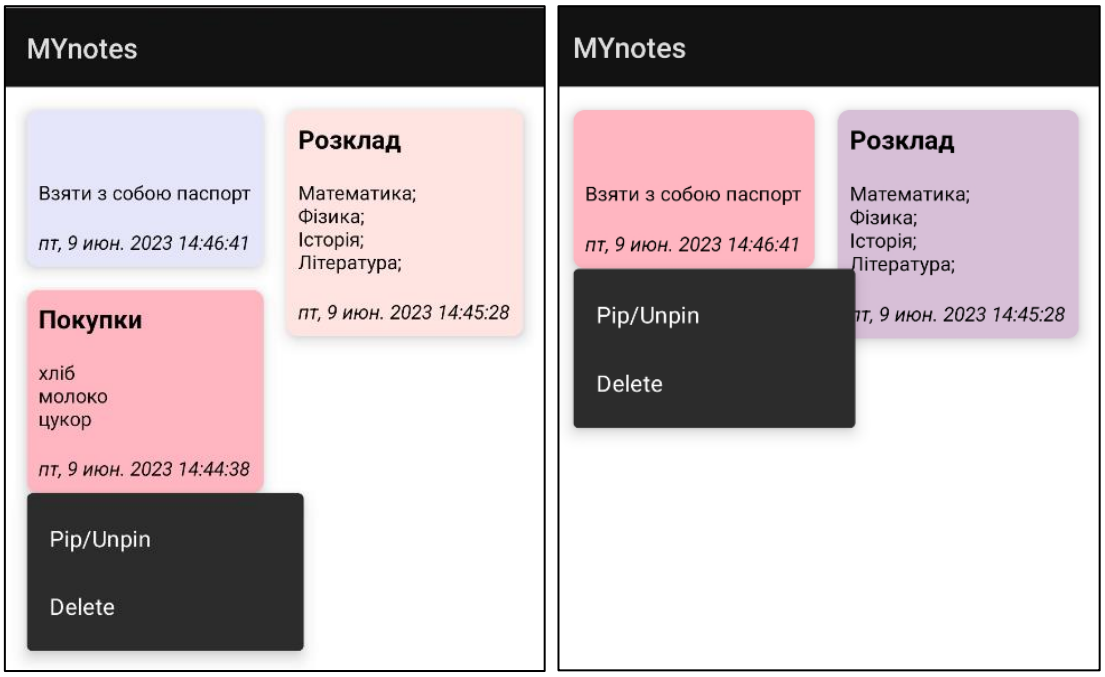


Рис. 1.27.1 Видалення нотатки: До та після

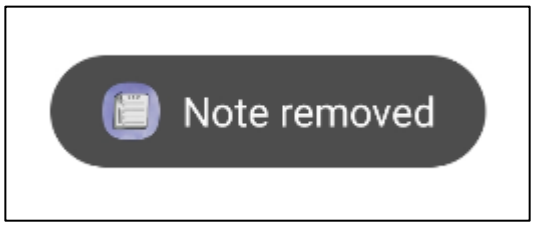


Рис. 1.27.2 Підсказка при видаленні

3.5. Висновок до розділу 3

Отже, колір полів змінюється при кожній взаємодії з цим полем, клацання та утримання працює, підсказки висвітлюються без затримки. Дата та час синхронізується з датою та часом пристрою, все коректно відображається. Нотатка видаляється безпроблемно, підсказка висвітлюється також без затримки. Тестування додатку MYnotes пройшло успішно.

ВИСНОВКИ

У ході розробки даної дипломної роботи було проведено детальний аналіз предметної області, а саме розробки додатку для планування та організації завдань на платформі Android.

Для досягнення поставленої мети необхідно було вирішити такі завдання:

- розглянути класифікацію мобільних додатків;
- розглянути мобільні операційні системи;
- вивчити основні поняття та розглянути види мов програмування;
- дослідити середовище розробки;
- проаналізувати основні етапи розробки мобільного додатку;
- розглянути сервіси для створення прототипів додатку;
- проаналізувати аналогічні додатки у сервісі Google Play;
- уточнити вимоги до системи та дизайн додатку;
- створити прототип додатку у сервісі Figma;
- створити піктограму для додатку;
- розробити основний алгоритм написання програми для створення нотаток;
- висвітлити архітектуру програмного забезпечення;
- створити алгоритм роботи усієї програми;
- протестувати готове програмного забезпечення.

Для створення програмного забезпечення використовувалися науково-технічна література та матеріали з сайтів, присвячених програмуванню для мобільних телефонів з операційною системою Android.

Було досліджено ключові аспекти створення мобільних додатків, включаючи класифікацію, аналіз операційних систем, мови програмувааня, зокрема Java, дослідження середовищ розробки та сервісів для створення прототипів, етапи розробки мобільного додатку. Цей аналіз дав змогу зрозуміти основні принципи та технології, що використовуються при розробці планерів на даній платформі.

Було проаналізовано аналогічні додатки у сервісі Google Play, та сформовано список функціональних критеріїв, та вимог до оформлення.

У результаті проведеного дослідження було встановлено, що розробка додатків для організації та планування завдань вимагає розуміння стандартних архітектурних компонентів, таких як Room Persistence, який є більш простим та сучасним способом управління. За допомогою компонентів Entity, Data Access Object та Database було створено таблицю, інтерфейс який описує методи доступу до БД та точку доступу до з'єднання з базою даних, відповідно.

Налаштування взаємодії користувача з елементами інтерфейсу є ще одним важливим аспектом, тому було використано такі «слухачі подій» як onClick та onLongClick, що дозволяє взаємодіяти з полями самої нотатки редагуючи її, видаляючи чи закріплюючи.

Для розробки програмного забезпечення використовувалося інтегроване середовище розробки Android Studio. Програмне забезпечення працює на апаратних засобах з операційною системою Android із версією від 7.0 до 10.0.

В цілому, розроблений Android-додаток для організації та планування завдань демонструє успішну реалізацію задуманої функціональності та архітектури. Він відповідає вимогам, поставленим перед розробкою, та забезпечує зручний функціонал та гарне естетичне оформлення для майбутніх користувачів.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення».

2. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс] – Режим доступу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/. (Дата звернення: 17.05.2023) – Назва з екрана.

3. Типи мобільних додатків [Електронний ресурс] – Режим доступу: https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types. (Дата звернення: 17.05.2023) – Назва з екрана.

4. Основи програмування на Java [Електронний ресурс] – Режим доступу: https://promoter.net.ua/articles/osnovi-programuvannya-na-java.html. (Дата звернення: 17.05.2023) – Назва з екрана.

5. Копитко М. Ф. Основи програмування мовою Java / М. Ф. Копитко, К. С. Іванків; Нац. ун-т «ЛНУ ім. Івана Франка». − Львів: Вид-во Нац. ун-ту «ЛНУ ім. Івана Франка», 2016.− 83 с.

6. Рейтинг мов програмування 2023. [Електронний ресурс] – Режим доступу: https://dou.ua/lenta/articles/language-rating-2023/. (Дата звернення: 17.05.2023) – Назва з екрана.

7. Інструменти і середовища розробки мобільних додатків [Електронний ресурс] – Режим доступу: https://ppt-online.org/341525. (Дата звернення: 18.05.2023) – Назва з екрана.

8. Розробка мобільних додатків [Електронний ресурс] – Режим доступу: https://webcase.com.ua/uk/razrabotka-mobilnyh-prilozhenij/ (Дата звернення: 18.05.2023) – Назва з екрана.

9. Figma з нуля [Електронний ресурс] – Режим доступу: https://ux.pub/designis/figma-z-nulia-vchimos-pratsiuvati-u-fighma-55pl/. (Дата звернення: 18.05.2023) – Назва з екрана.

10. Dao [Електронний ресурс] – Режим доступу: https://developer.android.com/reference/android/arch/persistence/room/Dao. (Дата звернення: 01.06.2023) – Назва з екрана.

11. Database [Електронний ресурс] – Режим доступу: https://developer.android.com/reference/android/arh/persistence/room/Database. (Дата звернення: 01.06.2023) – Назва з екрана.

12. Delete [Електронний ресурс] – Режим доступу: https://developer.android.com/reference/android/arh/persistence/room/Database. (Дата звернення: 01.06.2023) – Назва з екрана.

13. Insert [Електронний ресурс] – Режим доступу: https://developer.android.com/reference/android/arch/persistence/room/Insert. (Дата звернення: 01.06.2023) – Назва з екрана.

14. Пашина У. О. Фреймворк Flutter та його особливості / Єгоров С. В. Пашина У. О. / / Інформаційні технології в приладобудуванні та машинобудуванні: зб. наук. праць. : НАУ, 2023. – 261-262с.

ДОДАТКИ

Додаток А

Class NotesListAdapter

```
package uli.pashyna.mynotes.Adapter;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import uli.pashyna.mynotes.Models.Notes;
import uli.pashyna.mynotes.NotesClickListener;
import uli.pashyna.mynotes.R;

public class NotesListAdapter extends RecyclerView.Adapter
<NotesListAdapter.NotesViewHolder> {

    Context context;
    List<Notes> list;

    NotesClickListener listener;

    public NotesListAdapter(Context context, List<Notes> list, NotesClickListener listener)
{
        this.context = context;
        this.list = list;
        this.listener = listener;
    }
```

```
   @NonNull
    @Override
    public NotesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int

viewType) {
        return new
NotesViewHolder(LayoutInflater.from(context).inflate(R.layout.notes_list, parent, false));
    }

    @Override
    public void onBindViewHolder(@NonNull NotesViewHolder holder, int position) {

        holder.textView_title.setText(list.get(position).getTitle());
        holder.textView_title.setSelected(true);

        holder.textView_notes.setText(list.get(position).getNotes());

        holder.textView_date.setText(list.get(position).getDate());
        holder.textView_date.setSelected(true);


        if (list.get(position).isPinned()) {
            holder.imageView_pin.setImageResource(R.drawable.pin_icon);
        } else {
            holder.imageView_pin.setImageResource(0);
        }
        int color_code = getRandomColor();

holder.notes_conteiner.setCardBackgroundColor(holder.itemView.getResources().getColo
r(color_code, null));

        holder.notes_conteiner.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                listener.onClick(list.get(holder.getAdapterPosition()));
            }
        });
        holder.notes_conteiner.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {
                listener.onLongCLick(list.get(holder.getAdapterPosition()),
```

```
holder.notes_conteiner);
                return true;
            }
        });
    }
 private int getRandomColor() {
        List<Integer> colorCode = new ArrayList<>();
        colorCode.add(R.color.light_pink);
        colorCode.add(R.color.thistle);
        colorCode.add(R.color.misty_rose);
        colorCode.add(R.color.lavander);
        colorCode.add(R.color.light_pink);

        Random random = new Random();
        int random_color = random.nextInt(colorCode.size());
        return colorCode.get(random_color);
    }

    @Override
    public int getItemCount() {
        return list.size();
    }


    class NotesViewHolder extends RecyclerView.ViewHolder {

        CardView notes_conteiner;
        TextView textView_title, textView_notes, textView_date;
        ImageView imageView_pin;

        public NotesViewHolder(@NonNull View itemView) {
            super(itemView);

            notes_conteiner = itemView.findViewById(R.id.notes_conteiner);
            textView_title = itemView.findViewById(R.id.textView_title);
            textView_notes = itemView.findViewById(R.id.textView_notes);
            textView_date = itemView.findViewById(R.id.textView_date);
            imageView_pin = itemView.findViewById(R.id.imageView_pin);
        }
    }
}
```

Class NotesTakerActivity

```
package uli.pashyna.mynotes;

import androidx.appcompat.app.AppCompatActivity…

public class NotesTakerActivity extends AppCompatActivity {

    EditText editText_title, editText_notes;
    ImageView imageView_save;
    Notes notes;
    boolean isOldNote = false;

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

        editText_title = findViewById(R.id.editText_title);
        editText_notes = findViewById(R.id.editText_notes);

        imageView_save = findViewById(R.id.imageView_save);

        notes = new Notes();
        try {
            notes = (Notes) getIntent().getSerializableExtra("old_note");
            editText_title.setText(notes.getTitle());
            editText_notes.setText(notes.getNotes());
            isOldNote = true;
        }catch (Exception e){
            e.printStackTrace();
        }


        imageView_save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String title = editText_title.getText().toString();
                String description = editText_notes.getText().toString();
```

```
            if(description.isEmpty()) {
                Toast.makeText(NotesTakerActivity.this, "Please, enter description",
Toast.LENGTH_SHORT).show();
                return;
            }
            SimpleDateFormat formatter = new SimpleDateFormat("EEE, d MMM yyyy
HH:mm:ss");
            Date date = new Date();

            if (!isOldNote) {
                notes = new Notes();
            }

            notes.setTitle(title);
            notes.setNotes(description);
            notes.setDate(formatter.format(date));

            Intent intent = new Intent();
            intent.putExtra("note", notes);
            setResult(Activity.RESULT_OK, intent);
            finish();
        }
    });
  }
}
```

Class MainActivity

```
package uli.pashyna.mynotes;

import androidx.activity.result.contract.ActivityResultContracts…

public class MainActivity extends AppCompatActivity implements
PopupMenu.OnMenuItemClickListener {

  RecyclerView recyclerView;
  FloatingActionButton fab_add;
  NotesListAdapter notesListAdapter;
  RoomDB database;
  List<Notes> notes = new ArrayList<>();
```

```
Notes selectedNote;

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

    recyclerView = findViewById(R.id.recycler_home);
    fab_add = findViewById(R.id.fab_add);


    database = RoomDB.getInstance(this);
    notes = database.mainDao().getAll();


    updateRecyclre(notes);

    fab_add.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(MainActivity.this, NotesTakerActivity.class);
            startActivityForResult(intent, 101);
        }
    });


}

private void filter(String newText) {
    List<Notes> filteredList = new ArrayList<>();
    for (Notes singleNote : notes) {
        if (singleNote.getTitle().toLowerCase().contains(newText.toLowerCase())
                || singleNote.getNotes().toLowerCase().contains(newText.toLowerCase())) {
            filteredList.add(singleNote);

        }
    }
    updateRecyclre(filteredList);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)
```

```
            {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == 101) {
            if (resultCode == Activity.RESULT_OK) {
                Notes new_notes = (Notes) data.getSerializableExtra("note");
                database.mainDao().insert(new_notes);
                notes.clear();
                notes.addAll(database.mainDao().getAll());
                notesListAdapter.notifyDataSetChanged();
            }

        }
        if (requestCode == 102) {
            if (resultCode == Activity.RESULT_OK) {
                Notes new_notes = (Notes) data.getSerializableExtra("note");
                database.mainDao().update(new_notes.getID(), new_notes.getTitle(),
new_notes.getNotes());
                notes.clear();
                notes.addAll(database.mainDao().getAll());
                notesListAdapter.notifyDataSetChanged();
            }
        }
    }

    private void updateRecyclre(List<Notes> notes) {
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new StaggeredGridLayoutManager(2,
LinearLayoutManager.VERTICAL));
        notesListAdapter = new NotesListAdapter(MainActivity.this, notes,
notesClickListener);
        recyclerView.setAdapter(notesListAdapter);

    }
    private final NotesClickListener notesClickListener = new NotesClickListener() {
        @Override
        public void onClick(Notes notes) {
            Intent intent = new Intent(MainActivity.this, NotesTakerActivity.class);
            intent.putExtra("old_note", notes);
            startActivityForResult(intent, 102);
        }

        @Override
```

```
        public void onLongCLick(Notes notes, CardView cardView) {

            selectedNote = new Notes();
            selectedNote = notes;
            showPopup (cardView);
        }
    };

    private void showPopup(CardView cardView) {

        PopupMenu popupMenu = new PopupMenu(this, cardView);
        popupMenu.setOnMenuItemClickListener(this);
        popupMenu.inflate(R.menu.popup_menu);
        popupMenu.show();
    }

    @Override
    public boolean onMenuItemClick(MenuItem item) {
        int itemId = item.getItemId();

        if (itemId == R.id.pin) {
            boolean isPinned = selectedNote.isPinned();
            database.mainDao().pin(selectedNote.getID(), !isPinned);
            String toastMessage = isPinned ? "Unpinned" : "Pinned";
            Toast.makeText(MainActivity.this, toastMessage,
Toast.LENGTH_SHORT).show();
            notes.clear();
            notes.addAll(database.mainDao().getAll());
            notesListAdapter.notifyDataSetChanged();
            return true;
        } else if (itemId == R.id.delete) {
            database.mainDao().delete(selectedNote);
            notes.remove(selectedNote);
            notesListAdapter.notifyDataSetChanged();
            Toast.makeText(MainActivity.this, "Note removed",
Toast.LENGTH_SHORT).show();
            return true;
        }

        return false;
    }
}
```