https://www.sworldjournal.com/index.php/swj/article/view/swj31-01-048
DOI: 10.30888/2663-5712.2025-31-01-048

УДК 004.42:; 378.147:; 004.94

DEVELOPMENT OF A MOBILE APPLICATION FOR REMINDER ABOUT TAKING MEDICINES

РОЗРОБКА МОБІЛЬНОГО ДОДАТКА ДЛЯ НАГАДУВАННЯ ПРО ПРИЙМАННЯ ЛІКІВ

Altukhova T. V. / Алтухова Т.В.
c.t.s. / к.т.н.
ORCID: 0000-0002-8255-5725
Donetsk National Technical University, Drohobych, Sambirska,76, 82111
Донецький національний технічний університет, м. Дрогобич, Самбірська, 76, 82111
Sergienko L. H. / Сергієнко Л. Г.
c.p.s., as.prof. / к.пед.н., доц.
ORCID: 0000-0001-8668-7736
Donetsk National Technical University, Drohobych, Sambirska, 76, 82111
Донецький національний технічний університет, м. Дрогобич, Самбірська, 76, 82111
Sergienko O. O. / Сергієнко О. О.
student / студент гр.ІПЗм-11,
ORCID: 0009-0005-6309-5814
Lutsk National Technical University, Lutsk, Lvivska, 75, 43018
Луцький національний технічний університет, м. Луцьк, Львівська, 75, 43018

Анотація. *В статті розглядається питання розробки зручного та функціонального мобільного застосунку, що забезпечує управління нагадуваннями про своєчасне приймання ліків. Розглянуто основні аспекти предметної області, обґрунтовано вибір технологій, реалізовано функціонал додатка та проведено його тестування. Це дозволяє не лише продемонструвати досягнутий результат, а й надати рекомендації для подальшого вдосконалення продукту. В процесі проєктування застосовувалися широко використовувані технології, зокрема, мова програмування Kotlin, фреймворк Jetpack Compose для створення інтерфейсу користувача з використанням Material Design 3, Jetpack Compose Navigation для організації навігації, а також ORM Room для роботи з базою даних. В результаті був запропонований та апробований мобільний додаток, що забезпечує зручний інтерфейс, функціональність для управління графіком та нагадуваннями, а також надійне збереження даних користувача.*

Ключові слова: *мобільний додаток, Kotlin, Jetpack Compose, Jetpack Compose Navigation, Material Design 3, ORM Room, MVVM, OkHTTP.*

Вступ.

Сучасний ритм життя ставить перед людиною багато завдань, серед яких важливе місце займає підтримання власного здоров'я. Для цього необхідно дотримуватись рекомендацій лікарів, якщо це необхідно, зокрема, приймати ліки за встановленим графіком. Проте через велику кількість справ і зобов'язань багато людей стикаються з труднощами у відстеженні часу приймання ліків [1].

Розвиток мобільних технологій дозволяє значно спростити цей процес. Завдяки спеціалізованим додаткам користувачі можуть зручно планувати свій

графік, отримувати нагадування та контролювати приймання медичних препаратів. Такі інструменти не лише підвищують дисциплінованість, а й сприяють покращенню якості життя [1].

Метою роботи є створення функціонального програмного продукту, що забезпечує зручність у користуванні, надійність і високий рівень інтерактивності. Результати розробки можуть бути корисними як для індивідуальних користувачів, так і для медичних установ, які прагнуть підвищити ефективність лікування пацієнтів будь якого віку.

У роботі розглянуто основні аспекти предметної області, обґрунтовано вибір технологій, реалізовано функціонал додатка та проведено його тестування. Це дозволяє не лише продемонструвати досягнутий результат, а й надати рекомендації для подальшого вдосконалення продукту.

Аналіз останніх досліджень і публікацій в області сучасних медичних додатків. Сучасні медичні додатки займають важливе місце у забезпеченні здорового способу життя та покращенні медичного обслуговування. Їх розвиток обумовлений зростанням попиту на зручні цифрові інструменти для управління станом здоров'я, зокрема, контролю за прийманням ліків, моніторингу фізичної активності, аналізу показників організму та підтримки комунікації з лікарями [1, 2].

Додатки для управління графіком приймання ліків мають на меті спростити взаємодію користувачів з власними медичними завданнями. Серед популярних продуктів такого типу можна виділити [1]: Medisafe, який дозволяє створювати детальні графіки приймання ліків, встановлювати нагадування та ділитися інформацією з близькими; MyTherapy, що забезпечує відстеження прийманням ліків, фізичної активності та інших показників здоров’я; Pill Reminder, який орієнтований на мінімалістичний інтерфейс і простоту використання. Основні функції, які характерні для цих додатків [1, 2]: користувачі можуть встановлювати час приймання ліків, періодичність та додавати примітки; додатки генерують сповіщення у вигляді звукових сигналів або push-сповіщень; можливість синхронізації із системними календарями для підвищення зручності;

розширений функціонал, який дозволяє записувати симптоми чи інші важливі дані; забезпечення конфіденційності користувацької інформації є ключовим аспектом у розробці медичних додатків.

Однак, незважаючи на широкий вибір існуючих додатків, багато з них стикаються з проблемами, такими, як перевантаження інтерфейсу, відсутність персоналізації або низька інтеграція з іншими сервісами. Розглядаючи аналоги, можна дійти висновку, що існує необхідність у створенні медичних додатків з інтуїтивно зрозумілим інтерфейсом, адаптованим під індивідуальні потреби користувачів. У цій роботі буде запропоновано рішення, що поєднує сучасний дизайн, зручність навігації та надійність зберігання даних.

Вимоги до додатків для управління нагадуваннями.

Додатки для управління нагадуваннями відіграють важливу роль у підтримці дисципліни та організації повсякденного життя користувачів. Особливо це актуально для медичних застосунків, що допомагають дотримуватися графіка приймання ліків. Вимоги до таких програм визначаються потребами користувачів у функціональності, зручності використання та надійності роботи [3].

Однією з основних вимог є інтуїтивно зрозумілий інтерфейс, який дозволяє легко створювати, редагувати та видаляти нагадування. Користувачі повинні мати можливість швидко налаштовувати параметри, такі як: час, частота та тривалість нагадувань, без необхідності витрачати багато часу на навчання або освоєння програми. Важливою вимогою є надійність сповіщень. Додаток повинен забезпечувати своєчасне нагадування у вигляді сповіщень, навіть у випадках, коли пристрій перебуває в режимі енергозбереження. Це гарантує той факт, що користувачі не пропустять важливі події чи час приймання ліків [4-5].

Захист конфіденційності даних користувачів є ще одним ключовим аспектом. Оскільки додатки для нагадувань часто містять чутливу інформацію, зокрема щодо медичних препаратів, вони повинні відповідати високим стандартам безпеки. Забезпечення локального шифрування даних або використання надійних серверів для їх зберігання є обов’язковим [4].

Для підвищення зручності ці додатки мають підтримувати інтеграцію з іншими системами, такими, як календарі чи хмарні сервіси. Це дозволяє користувачам синхронізувати нагадування з особистим розкладом і отримувати доступ до них на різних пристроях. Крім того, важливою є можливість адаптації додатка до індивідуальних потреб. Наприклад, програма може пропонувати персоналізовані рекомендації чи функції, засновані на аналізі поведінки користувача. Такий підхід сприяє підвищенню довіри до продукту та його популярності серед користувачів.

Загалом, високоякісний додаток для управління нагадуваннями має бути зручним, ефективним і надійним. Це дозволить забезпечити користувачам можливість легко керувати своїм розкладом і своєчасно виконувати важливі завдання, зокрема, пов’язані з власним здоров’ям.

Вибір технологій для реалізації додатка. Мова програмування Kotlin та її переваги. Kotlin – це сучасна мова програмування, створена для вирішення завдань розробки програмного забезпечення з акцентом на продуктивність, зручність використання та надійність. Вона є офіційно підтримуваною мовою для розробки Android-додатків і широко використовується завдяки своїм потужним можливостям та інтеграції з екосистемою Java [2]. Однією з основних переваг Kotlin є його лаконічність. Код, написаний на Kotlin, зазвичай коротший і зрозуміліший, ніж аналогічний код на Java, завдяки використанню сучасного синтаксису та зменшенню шаблонного коду. Це сприяє швидшій розробці та зменшує ризик помилок. Крім того, Kotlin підтримує сумісність із Java, що дозволяє використовувати існуючі бібліотеки та інтегрувати новий код із наявними Java-проєктами. Це робить його привабливим вибором для поступового переходу від Java до сучасніших підходів у програмуванні [6].

Безпека є ще однією важливою перевагою Kotlin. Мова має вбудовані механізми, що зменшують ймовірність виникнення помилок, таких як NullPointerException. Використання системи типів Kotlin дозволяє уникнути помилок, пов’язаних із роботою з null-значеннями, забезпечуючи більшу надійність додатків [6, 7].

Функціональні можливості Kotlin включають підтримку лямбда-виразів, корутин для асинхронного програмування, розширення функцій класів без їх зміни та багато інших сучасних інструментів. Це робить мову дуже гнучкою та придатною для широкого кола завдань [8]. Додатковою перевагою цих можливостей є активна підтримка розробників з боку спільноти та компанії JetBrains, яка розробляє Kotlin. Регулярні оновлення мови та інструментів, а також велика кількість документації та ресурсів забезпечують комфортну роботу з цією мовою [6].

Таким чином, Kotlin є потужним інструментом для розробки сучасних додатків. Завдяки своїй зручності, безпеці та гнучкості, мова дозволяє створювати ефективні, надійні та зручні програмні продукти, що відповідають високим стандартам сучасного програмування.

Jetpack Compose для створення інтерфейсу користувача. Jetpack Compose – це сучасний інструмент для розробки інтерфейсів користувача на платформі Android, створений для спрощення та прискорення роботи розробників. Він базується на декларативному підході, що дозволяє описувати інтерфейс у вигляді функцій, які формують структуру елементів без необхідності використання XML [9].

Однією з головних переваг Jetpack Compose є спрощення процесу розробки. Завдяки декларативному синтаксису, розробники можуть описувати компоненти інтерфейсу безпосередньо в коді, а також легко оновлювати їх під час зміни стану програми. Це дозволяє швидше створювати прототипи, тестувати функціональність та вносити зміни. Jetpack Compose забезпечує повну інтеграцію з іншими бібліотеками та компонентами екосистеми Android, такими як Material Design, Navigation і ViewModel. Зокрема, підтримка Material Design 3 дозволяє створювати сучасні, естетичні інтерфейси, які відповідають останнім стандартам дизайну [7, 9, 10].

Гнучкість є ще однією перевагою Jetpack Compose. Інструмент дозволяє легко налаштовувати вигляд і поведінку компонентів, створювати власні елементи інтерфейсу та динамічно змінювати їх залежно від контексту або даних

[10]. Compose також сприяє оптимізації продуктивності. Завдяки використанню ефективного механізму перебудування елементів, оновлюються тільки ті частини інтерфейсу, які зазнали змін. Це підвищує швидкодію додатків, особливо на пристроях із низькою продуктивністю. Ще одним важливим аспектом є простота тестування. Оскільки компоненти інтерфейсу в Compose представлені у вигляді функцій, їх можна легко ізолювати та тестувати окремо. Це знижує складність підтримки коду та дозволяє забезпечувати високу якість продукту [9].

Таким чином, Jetpack Compose є ефективним і сучасним інструментом для створення інтерфейсів користувача, який забезпечує гнучкість, швидкість розробки та відповідність сучасним стандартам дизайну. Його використання сприяє створенню зручних, естетичних і продуктивних додатків, що задовольняють потреби як розробників, так і користувачів.

Окрім того, для організації навігації застосовують бібліотеку Jetpack Compose Navigation, яка забезпечує зручний і сучасний підхід до реалізації навігації в додатках, розроблених із використанням Jetpack Compose. Вона дозволяє ефективно управляти переходами між екранами та створювати логічну структуру додатка, підтримуючи декларативний стиль програмування [7]. Однією з основних переваг Jetpack Compose Navigation є її інтеграція з Compose, що дозволяє створювати та налаштовувати маршрути безпосередньо в коді. Замість використання XML-файлів, які традиційно застосовувалися у фреймворку Android, розробники можуть визначати навігаційні графи у вигляді функцій, що робить процес більш гнучким і зручним [8-10]. Бібліотека підтримує роботу з параметрами маршруту, що дозволяє передавати дані між екранами. Наприклад, користувач може передати ID нагадування чи інформацію про приймання ліків із одного екрану на інший без зайвих складнощів. Це забезпечує динамічну поведінку додатка та гнучкість у роботі з даними.

Jetpack Compose Navigation також підтримує функцію back stack (стек повернень), яка автоматично управляє переходами назад. Це дозволяє користувачам легко повертатися до попередніх екранів, зберігаючи їхній стан. У

складніших сценаріях розробники можуть додатково налаштувати стек навігації для управління поведінкою додатка. Ще однією важливою можливістю є підтримка deep links (глибоких посилань). Це дозволяє користувачам переходити до конкретного екрану в додатку безпосередньо з посилання, наприклад, з електронної пошти або повідомлення. Такі функції особливо корисні для додатків, пов'язаних із нагадуваннями, де deep links можуть використовуватися для швидкого доступу до конкретного запису чи події. Jetpack Compose Navigation також спрощує організацію складних навігаційних структур, таких як вкладені навігаційні графи. Це дозволяє створювати модульні та масштабовані додатки, в яких кожна частина має свою ізольовану логіку навігації [8, 9].

У підсумку, Jetpack Compose Navigation є потужним інструментом для створення логічної та зручної навігації в додатках. Завдяки декларативному стилю програмування, гнучкості та підтримці сучасних функцій, вона дозволяє створювати інтуїтивно зрозумілі та функціональні продукти, які задовольняють потреби користувачів.

Користувацький дизайн для зручності користувачів можно створювати на основі Material Design 3 (MD3), яка є сучасною системою дизайну, розроблена компанією Google, яка спрямована на створення інтуїтивно зрозумілих, естетично привабливих і адаптивних інтерфейсів користувача. Вона забезпечує узгодженість зовнішнього вигляду додатків на різних платформах, сприяючи підвищенню зручності та задоволеності користувачів [5,7,8].

Однією з ключових переваг MD3 є підтримка персоналізації. Завдяки концепції Material You, користувачі можуть адаптувати вигляд додатка відповідно до своїх уподобань, включаючи вибір кольорової палітри, що генерується автоматично на основі шпалер пристрою. Це створює відчуття індивідуального підходу та покращує взаємодію з продуктом. MD3 орієнтований на доступність, забезпечуючи оптимальну читабельність текстів, контрастність елементів і зрозумілі візуальні підказки. Наприклад, кнопки, поля вводу та інші компоненти мають чітко визначені межі та зрозумілі стани (натиснута, вибрано, недоступно), що полегшує їх використання навіть для людей із вадами зору.

Особлива увага приділяється інтерактивності та анімації [7, 9]. Material Design 3 пропонує набір анімованих переходів і ефектів, які роблять взаємодію з додатком більш плавною та природною. Це допомагає користувачам краще орієнтуватися в інтерфейсі, спрощуючи виконання завдань. MD3 підтримує принципи адаптивного дизайну, що дозволяє інтерфейсам автоматично підлаштовуватися під різні розміри екранів. Це забезпечує зручне використання додатків як на мобільних пристроях, так і на планшетах або навіть настільних комп'ютерах [8]. Крім того, використання MD3 сприяє економії часу розробників завдяки готовому набору компонентів, таких як кнопки, картки, діалогові вікна та панелі навігації. Ці елементи вже оптимізовані для сучасних вимог і відповідають кращим практикам дизайну, що значно прискорює процес розробки.

У підсумку, Material Design 3 пропонує потужний інструментарій для створення зручних, функціональних і стильних інтерфейсів, які відповідають очікуванням користувачів. Завдяки адаптивності, персоналізації та орієнтації на доступність, MD3 є невід'ємною складовою сучасних додатків, які прагнуть забезпечити високий рівень користувацького досвіду.

Для зберігання даних застосовується бібліотека ORM Room, яка призначена для роботи з базами даних у додатках, що використовують платформу Android. Вона є частиною Android Jetpack і призначена для спрощення доступу до SQLite-бази даних, дозволяючи зберігати та отримувати дані за допомогою об'єктів, що представляють таблиці бази даних. Room автоматизує багато аспектів роботи з базами даних, забезпечуючи більшу зручність і надійність порівняно з прямим використанням SQLite. Однією з основних переваг Room є його інтеграція з Kotlin та Jetpack Compose. Це дозволяє розробникам використовувати сучасні можливості мови та фреймворку для створення ефективних додатків, що працюють із базами даних. Room використовує анотації для визначення сутностей бази даних (таблиць), що робить код чистим і зрозумілим, а також дозволяє автоматично генерувати SQL-запити [2].

Бібліотека Room включає кілька важливих компонентів, а саме Entity – анотація, що визначає клас як сутність бази даних (таблицю); Dao (Data Access

Object) – інтерфейс, який містить методи для роботи з базою даних. Room автоматично генерує імплементацію цього інтерфейсу; Database – анотація для створення бази даних, яка визначає всі сутності та DAO, що використовуються в додатку. Цей підхід дозволяє зменшити кількість шаблонного коду та уникнути помилок, пов'язаних із прямим написанням SQL-запитів. Room підтримує складні операції, такі як асинхронні запити, транзакції, індекси та зовнішні ключі, що робить роботу з базою даних більш гнучкою та безпечною. Room також дозволяє зберігати складні структури даних, такі як списки або об'єкти, і працювати з ними, підтримуючи різні типи відносин між сутностями (один до одного, один до багатьох, багато до багатьох). Ця гнучкість робить його ідеальним рішенням для зберігання даних у додатках, які потребують складної логіки збереження. Нарешті, Room забезпечує високий рівень продуктивності, використовуючи індексацію та оптимізуючи запити, щоб зменшити навантаження на систему, навіть за великих обсягів даних. Room працює на основі SQLite, який є легким та швидким механізмом збереження даних, що дозволяє використовувати його в мобільних додатках з обмеженими ресурсами [7-10].

У підсумку, ORM Room є потужним інструментом для зберігання та управління даними в додатках Android. Завдяки своїй простоті використання, безпеці та інтеграції з сучасними технологіями, Room дозволяє розробникам створювати надійні, продуктивні та зручні додатки для зберігання даних.

Для HTTP-запитів використовується бібліотека OkHTTP, яка є потужною засобом для здійснення HTTP-запитів та широко використовується в Android-розробці для обміну даними між клієнтом та сервером. Вона пропонує простий, ефективний та гнучкий інтерфейс для роботи з мережевими запитами, що дозволяє розробникам зосередитися на логіці додатка, а не на деталях взаємодії з мережею. OkHTTP підтримує різні типи HTTP-запитів, включаючи GET, POST, PUT, DELETE та інші, а також забезпечує високий рівень кастомізації. Вона дозволяє легко налаштовувати заголовки, параметри запитів, а також працювати з тілами запитів та відповідей у різних форматах, таких як JSON, XML, чи навіть

мультимедійні файли. Бібліотека ефективно працює з великими обсягами даних, використовуючи потоки для обробки великих файлів і мінімізації витрат пам'яті [7].

Однією з ключових переваг OkHTTP є підтримка асинхронного виконання запитів. Це дозволяє виконувати мережеві операції у фоновому режимі, не блокуючи основний потік програми, що критично для мобільних додатків, де важлива швидкість відгуку та зручність використання. Бібліотека також забезпечує зручну обробку помилок і має вбудовані механізми для управління повторними спробами запитів, що значно покращує стабільність додатків [8].

OkHTTP підтримує додаткові функціональності, такі як кешування відповідей, що дозволяє зменшити навантаження на сервери і зменшити затримки у виконанні запитів, особливо в умовах обмеженої пропускної здатності мережі. Бібліотека також має інтеграцію з іншими інструментами, такими як Retrofit, для ще зручнішої обробки API-запитів. Використання OkHTTP забезпечує надійну та ефективну роботу з мережею в Android-додатках. Завдяки своїй простоті, гнучкості та широкому набору функцій, OkHTTP є однією з найпопулярніших бібліотек для роботи з HTTP-запитами на платформі Android. OkHTTP має низку механізмів безпеки, таких як підтримка SSL/TLS для захищених з'єднань, що дозволяє забезпечити безпечний обмін даними між клієнтом і сервером. Вона також дозволяє налаштовувати власні параметри безпеки, зокрема управління сертифікатами або перевіркою сертифікатів під час підключень [8].

Завдяки своїй гнучкості, швидкості та простоті використання, OkHTTP є однією з найпопулярніших бібліотек для роботи з HTTP-запитами в екосистемі Android і Java. Її можливості з кешування, асинхронної обробки та налаштування з'єднань роблять її ідеальним вибором для побудови стабільних, масштабованих і високопродуктивних додатків.

Розробка та побудова архітектури програмного продукту.

Архітектура програмного продукту визначає структуру та взаємодію всіх компонентів додатка, що впливає на його ефективність, масштабованість та

зручність для користувача [3]. Одним із найбільш популярних підходів до побудови архітектури сучасних Android-додатків є модель MVVM (Model-View-ViewModel), яка забезпечує чітке розділення логіки додатка, інтерфейсу користувача та збереження даних. У контексті додатка для управління графіком приймання ліків і нагадуваннями, архітектура MVVM забезпечує гнучкість і зручність у роботі з даними, управлінні взаємодією з користувачем та обробці різних подій у додатку [4]. Модель MVVM складається з трьох основних компонентів: Model, View і ViewModel (рис. 1).

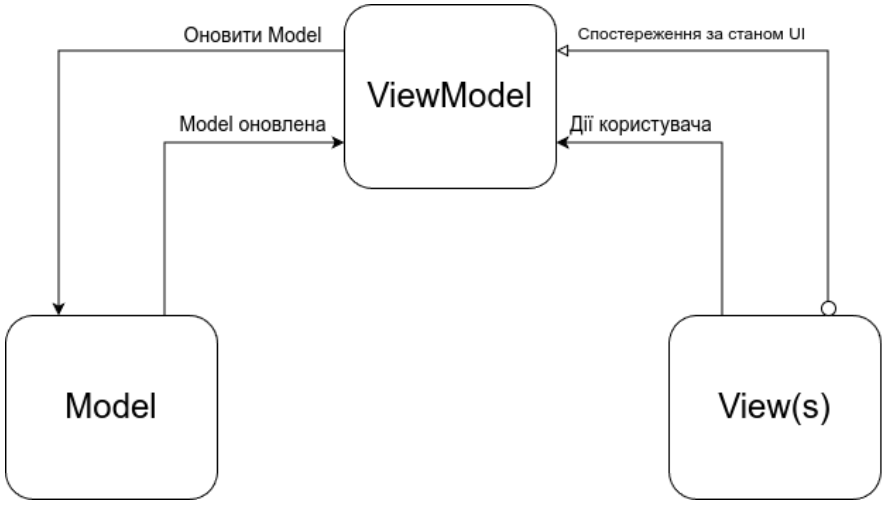

Рисунок 1 – Діаграма моделі MVVM

Model є складовою, яка відповідає за бізнес-логіку додатка та обробку даних. Вона включає в себе компоненти, які зберігають і обробляють інформацію, таку як дані про ліки, час приймання, нагадування і стан користувача. У додатку це може бути база даних, якою керує ORM Room, або зовнішні джерела даних, такі як API для синхронізації з іншими сервісами. Модель не повинна знати нічого про інтерфейс користувача та не має прямого доступу до елементів UI. View є безпосередньо інтерфейсом користувача, через який користувач взаємодіє з додатком. Це, наприклад, екрани для додавання нових ліків, перегляду розкладу, налаштування тощо. View відповідає за виведення інформації на екран і реагує на введення користувача, наприклад, натискання кнопок чи вибір елементів списку. Вона отримує всі необхідні дані від ViewModel і відображає їх, не виконуючи ніякої складної логіки. Це дає змогу

відокремити відображення даних від їхнього оброблення, що спрощує підтримку коду. ViewModel є проміжним шаром між View і Model. Його основна роль полягає в управлінні станом UI та обробці бізнес-логіки. ViewModel отримує дані з Model і підготовлює їх у вигляді, який зручно відображати в інтерфейсі користувача. Завдяки цьому ViewModel дозволяє підтримувати чистоту інтерфейсу користувача, не навантажуючи його логікою, що призводить до більш гнучкого та тестованого коду. Використання MVVM в архітектурі додатка дозволяє досягти кількох ключових переваг. По-перше, воно забезпечує чітке розділення відповідальностей, що робить додаток більш модульним і зручним для масштабування. По-друге, архітектура сприяє кращій тестованості, оскільки логіка додатка розділена на окремі частини, що дозволяє тестувати кожну з них незалежно. І, нарешті, MVVM дозволяє легко керувати станом UI та реакцією на зміни даних, що є критично важливим для створення додатків із динамічним інтерфейсом, таких як додатки для управління нагадуваннями [5].

Таким чином, архітектура MVVM для створення програмного продукту не лише покращує організацію коду, але й підвищує ефективність взаємодії з користувачем, робить додаток більш зручним і стабільним.

Реалізація функціоналу запису лікарських засобів.

Функціонал запису лікарських засобів у додатку реалізується через використання архітектури MVVM та ORM Room для зберігання даних. Цей процес забезпечує зручне введення, збереження та подальше використання інформації про ліки, які приймає користувач.

Опис реалізації цього функціоналу охоплює кілька ключових етапів. На першому етапі, Model в архітектурі MVVM визначає сутність лікарського засобу. Використовуючи Room, кожен запис про лікарський засіб стає окремим об'єктом, що містить основні властивості, такі як назва ліків, дозування, періодичність приймання, час та інші деталі, що можуть бути корисними для користувача. Room дозволяє ефективно зберігати ці дані в локальній базі даних, автоматично генеруючи необхідні SQL-запити та забезпечуючи доступ до них через DAO (Data Access Object). Для збереження даних у базі, необхідно

створити відповідну таблицю, де кожен запис буде мати унікальний ідентифікатор, а також інші атрибути, які дозволять користувачеві налаштувати нагадування для кожного лікарського засобу.

Наступним етапом є ViewModel, який займається управлінням бізнес-логікою і забезпечує зв'язок між View та Model. ViewModel отримує дані про лікарські засоби з Room, обробляє їх і надає в доступному для інтерфейсу вигляді. Наприклад, ViewModel може отримувати список всіх збережених лікарських засобів або окремі записи, фільтруючи їх за певними критеріями, такими як дата приймання або ім'я препарату. У процесі додавання нового лікарського засобу ViewModel обробляє введену користувачем інформацію, перевіряє коректність введених даних (наприклад, дозування, частота приймання) та передає ці дані до Model для збереження в базі.

ViewModel також може передавати необхідні команди для виклику додаткових функцій, таких як налаштування нагадувань. View, як частина архітектури MVVM, відповідає за відображення даних на екрані та введення користувача. Це може бути форма для введення даних лікарських засобів, де користувач заповнює необхідні поля, такі як назва ліків, дозування, час приймання тощо. Після заповнення форми та підтвердження введених даних View передає їх у ViewModel для подальшої обробки та збереження. Завдяки використанню MVVM, логіка взаємодії з користувачем та збереження даних є чітко розділеною. ViewModel обробляє всю бізнес-логіку, а View відповідає лише за виведення інформації та даних від користувача. Це дозволяє значно зменшити складність коду та покращити підтримку і тестування додатка. Room, у свою чергу, забезпечує ефективне зберігання і отримання даних про лікарські засоби, дозволяючи додатку працювати швидко і стабільно.

Таким чином, використання MVVM і ORM Room для реалізації функціоналу запису лікарських засобів дозволяє побудувати ефективну, масштабовану та зручну систему управління прийманням ліків, що забезпечує користувачеві можливість легко додавати, редагувати та переглядати інформацію про ліки.

Реалізація нагадувань про приймання ліків.

Функціонал нагадувань про приймання ліків у додатку реалізується через використання AlarmManager – інструмента Android для планування виконання задач у певний час або через певні інтервали. Це дозволяє створювати нагадування, які будуть активуватися навіть якщо додаток не працює в цей момент.

Основний принцип реалізації цього функціоналу полягає в плануванні повторюваних або одноразових будильників для нагадувань про приймання ліків, використовуючи введені користувачем дані про час приймання ліків. Користувач має можливість задати конкретний час для приймання кожного лікарського засобу, а також налаштувати частоту повторення нагадувань – наприклад, щодня або кілька разів на день. Після введення цих даних, додаток повинен зберегти ці налаштування в локальній базі даних (за допомогою ORM Room), а також налаштувати відповідні будильники для кожного приймання ліків.

AlarmManager дозволяє створювати будильники, які будуть виконуватися в заданий час. Кожен будильник пов'язаний з певною дією, наприклад, сповіщенням користувача про необхідність приймання ліків. Для цього можна використовувати PendingIntent – об'єкт, який вказує на дію, яку потрібно виконати, коли будильник спрацює. У випадку нагадування це може бути створення сповіщення, яке з’являтиметься на екрані користувача у вигляді push-повідомлення або системного сповіщення. При створенні нагадувань додаток має враховувати точність часу. Коли користувач додає нові ліки або редагує наявні записи, система повинна оновлювати відповідні будильники, замінюючи старі на нові, якщо час або інтервал змінилися. Це дозволяє забезпечити точність у нагадуваннях і уникнути дублювання нагадувань для одного і того ж часу.

Одним з важливих аспектів є можливість скасування або повторного планування будильників. Наприклад, якщо користувач скасовує приймання ліків або змінює частоту нагадувань, додаток повинен видалити попередньо налаштовані будильники для цього препарату і створити нові, з урахуванням

змін. У випадку, коли додаток знаходиться на фоні або неактивний, AlarmManager все одно буде забезпечувати своєчасне виконання нагадувань, що робить функціонал нагадувань надзвичайно важливим для ефективного використання. Це дозволяє користувачеві не залежати від того, чи працює додаток в даний момент, і гарантує отримання повідомлення у потрібний момент.

Завдяки використанню AlarmManager додаток для нагадувань про приймання ліків може функціонувати безперервно, навіть коли користувач не взаємодіє з додатком, забезпечуючи максимальну зручність і підтримку здоров'я.

Розробка бази даних для збереження інформації.

Для збереження інформації про лікарські засоби та нагадування про їх приймання в додатку використовується локальна база даних, що забезпечує ефективне збереження та доступ до даних навіть без підключення до Інтернету. Для реалізації цієї бази даних застосовується ORM-бібліотека Room, яка надає зручний спосіб взаємодії з SQLite. База даних складається з двох основних сутностей: Medicine і NotificationCard, кожна з яких відповідає за збереження певної інформації.

Сутність Medicine зберігає основну інформацію про лікарські засоби, яку користувач вводить або редагує. Вона містить такі поля, як назва ліків, тип препарату, час і умови приймання, а також додаткові нотатки, які можуть бути корисними для користувача. Поле imageUri зберігає посилання на зображення препарату, яке може бути корисним для візуальної ідентифікації ліків, хоча воно є необов'язковим. Всі ці дані зберігаються в таблиці medicines, що дає змогу додаткам зберігати та відновлювати їх для подальшого використання. Сутність NotificationCard відповідає за збереження інформації про нагадування для кожного лікарського засобу. Вона містить дані про час приймання ліків в форматі HH:mm і список днів тижня, коли потрібно приймати ліки. Дні тижня зберігаються як серіалізований список, що дозволяє гнучко налаштовувати нагадування на різні комбінації днів. Кожен запис у таблиці notification_cards

пов'язаний із конкретним препаратом, зберігаючи його назву та час приймання, що дає можливість створювати нагадування для кожного з лікарських засобів на відповідні дні.

Використання Room забезпечує зручне зберігання цих даних у локальній базі даних, а також ефективне управління ними через DAO (Data Access Object). DAO використовуються для доступу до таблиць, виконання запитів для додавання, оновлення та видалення записів.

Таким чином, база даних дозволяє зберігати всю необхідну інформацію про лікарські засоби та нагадування про їх приймання, що забезпечує зручне та швидке управління даними, а також підтримує їхню цілісність і ефективність у рамках роботи додатка. Всі ці можливості дають змогу користувачеві мати доступ до актуальної інформації в будь-який час без необхідності підключення до Інтернету

Розробка інструкції користувача.

Почнемо з того, що користувачу необхідно завантажити та встановити додаток. Відразу після запуску додатка відкривається початковий екран з якої можна почати використовувати «Додаток» та всі його функції. Це зображено на рис. 2.

Рисунок 2 – Початковий екран

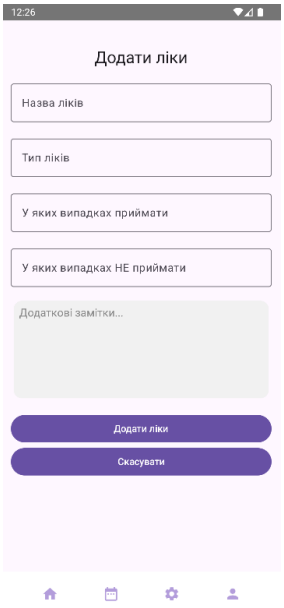

Рисунок 3 - Форма додавання ліків

Натиснувши на кнопку додати – користувачу буде запропоноване заповнити форму та додати новий лікарський засіб (рис. 3).

Всі додані лікарські засоби будуть додані на головний екран у вигляді списку з можливістю їх видалити, якщо це буде потрібно (рис. 4). Для створення нагадування про приймання ліків – користувач може перейти на наступний екран (рис. 5) та натиснути на кнопку «Додати», що у свої чергу відкриє форму для додавання нагадування (рис. 6).

Рисунок 4 – Список ліків на головному екрану

Рисунок 5 – Сторінка нагадувань

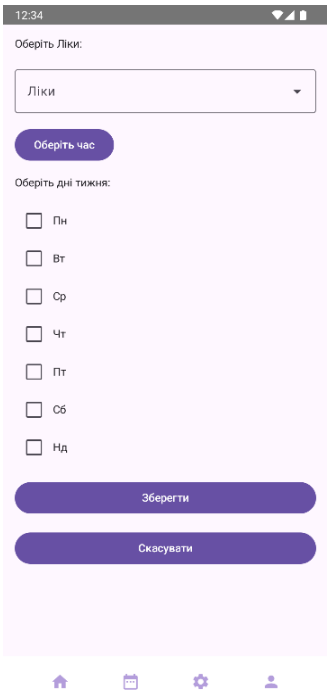

Рисунок 6 – Форма створення нагадування

Всі додані нагадування будуть відображатися на екрані у вигляді списку з можливістю їх відмітити та видалити, якщо це буде потрібно користувачу (рис. 7). Для синхронізації даних та зберігання інформації на віддаленому сервері – користувач може зареєструватись та увійти у свій обліковий запис (рис. 8). Якщо дані вірні – користувач побачить екран (рис. 9), що проінформує користувача, що він успішно зайшов у свій обліковий запис з можливістю виходу з нього.

Користувачі можуть реєструватися, входити в систему, переглядати

лікарські засоби, додавати інші засоби, або змінювати їх; виконувати вправи, які рекомендують реабілітологи, перевіряти свій прогрес та виходити з системи.

Висновки. Проведено аналіз предметної області та вимог до мобільних додатків для управління графіком і нагадуваннями про приймання ліків. На основі цього було визначено ключові функціональні вимоги до розробки програмного продукту. Було обґрунтовано вибір технологій, включаючи мову програмування Kotlin, Jetpack Compose для створення інтерфейсу користувача, Material Design 3 для забезпечення сучасного дизайну, Jetpack Compose Navigation для навігації, а також ORM Room для збереження даних. Розроблено архітектуру мобільного додатка, яка включає функції створення та управління налаштуваннями нагадувань і надійного збереження даних. Виконана інтеграція всіх компонентів та протестована функціональність програми, включаючи її інтерфейс та зручність використання користувачами.

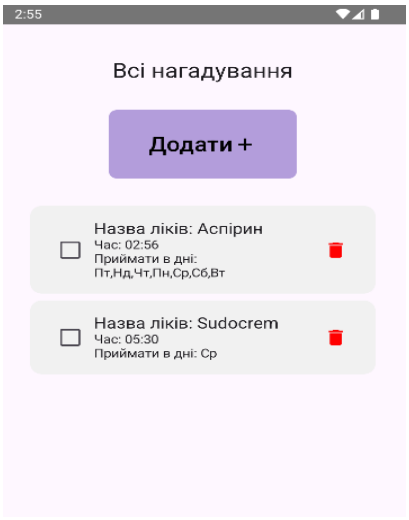

Рисунок 7 – Список нагадувань на сторінці нагадувань

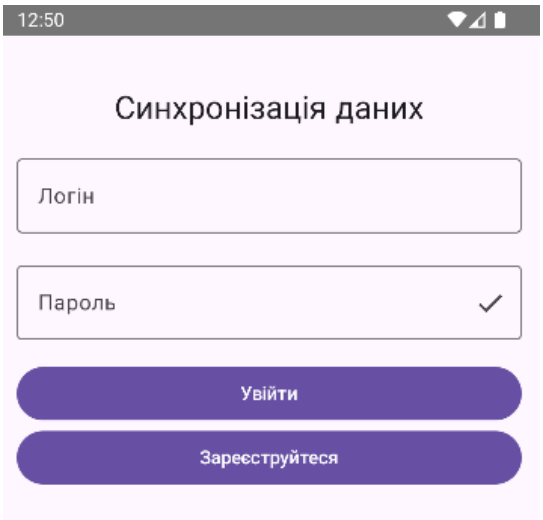

Рисунок 8 – Форма авторизації

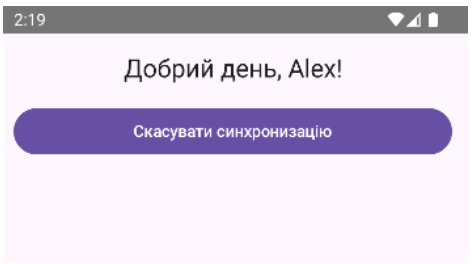

Рисунок 9 – Функціонал розділу «Exercises»

В якості резюме можна додати, що результатом роботи став мобільний додаток, що забезпечує зручний користувацький досвід, дозволяє користувачам ефективно керувати своїм графіком та отримувати своєчасні нагадування про приймання ліків. Отримані результати демонструють можливість створення високоякісного програмного продукту з використанням сучасних технологій і можуть слугувати основою для подальшого вдосконалення додатка не тільки для нагадування про приймання ліків, а й для інших задач (технічних, педагогічних, медичних, навчальних, облікових та інших завдань).

Література:

1. Як відстежувати прийом ліків за допомогою iOS та Android: 7 простих програм [Електронний ресурс]. – Режим доступу: https://mezha.media/articles/vidstezhyty-pryyom-likiv/

2. Android Basics with Compose [Електронний ресурс]. – Режим доступу: https://developer.android.com/courses/android-basics-compose/course

3. Роберт Мартін: Чиста архітектура: мистецтво розробки програмного забезпечення: Фабула, 2019, 416 с.

4. Paul Clements, Rick Kazman: Software Architecture in Practice: Addison-Wesley Professional, 2012, 322 p.

5. Моделювання та аналіз програмного забезпечення: методичні вказівки до практичних та лабораторних занять. / Укладач: Л.В. Глазунова - Одеса:ДУІТЗ, 2021., с. 92.

6. Create and use functions in Kotlin | Android Developers [Електронний ресурс]. – Режим доступу: https://developer.android.com/codelabs/basic-android-kotlin-compose-functions

7. Diana MacDonald, Practical UI Patterns for Design Systems: Fast-Track Interaction Design for a Seamless User Experience – Apress; 1st ed. edition (June 27, 2019) – 315 pages.

8. Jenifer Tidwell, Charles Brewer, Aynne Valencia, Designing Interfaces: Patterns for Effective Interaction, 3rd Edition – O'Reilly Media, (February 18, 2020) – 599

pages.

9. Martin Kleppmann, Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, 1st Edition – O'Reilly Media, (May 2, 2017) – 611 pages.

10. SOLID (об'єктно-орієнтоване програмування) – Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/SOLID_(об'єктно-орієнтоване програмування)

Abstract. *The article deals with the issue of developing a convenient and functional mobile application that provides management of reminders for timely medication. The main aspects of the subject area are considered, the choice of technologies is substantiated, the application functionality is implemented and its testing is carried out. This allows us not only to demonstrate the achieved result, but also to provide recommendations for further product improvement. In the design process, we used widely used technologies, including the Kotlin programming language, the Jetpack Compose framework for creating a user interface using Material Design 3, Jetpack Compose Navigation for organising navigation, and ORM Room for working with the database. As a result, a mobile application was proposed and tested that provides a user-friendly interface, functionality for managing schedules and reminders, and reliable storage of user data.*

Key words*: mobile app, Kotlin, Jetpack Compose, Jetpack Compose Navigation, Material Design 3, ORM Room, MVVM, OkHTTP*

Статтю надіслано: 20.05.2025 р.