МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «УНІВЕРСИТЕТ «КРОК»

Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА

за темою

«Розробка та створення додатку "список покупок" на Android»

Студентка 4 курсу групи КН-20К

Ситник К.О

(прізвище, ім'я та по-батькові студента)

Керівник дипломної роботи

кандидат фіз-мат. наук

(посада керівника)

Кириченко В.В.

(прізвище, ім'я та по-батькові керівника)

До захисту

(резолюція «До захисту»)

(підпис студента)

10.06.2024

(дата)

(підпис викладача)

Київ, 2024 рік

Скорочення

MVVM (Model-View-View-Model) - шаблон проєктування, що застосовується під час проєктування архітектури застосунків (додатків).

API Application Programming Interface) - це спосіб завдяки якому дві або більше комп'ютерні програми можуть спілкуватися між собою.

UI (User Interface) - засіб зручної взаємодії користувача (людини) з інформаційною системою.

NDK (Native Development Kit) - необхідний набір інструментарію для розробки компонентів програмного забезпечення для платформи Android, який базується на C/C++ та інших мовах програмування.

XML (Extensible Markup Language) - це спрощений діалект мови SGML, призначений для опису ієрархічних структур даних в World Wide Web.

Зміст

Вступ

У світі, де швидко зростає популярність мобільних пристроїв та безперервно розвиваються технології, мобільні додатки стають невід'ємною частиною нашого повсякденного життя. Вони не лише спрощують багато аспектів нашої рутини, але й дозволяють нам бути більш організованими та продуктивними. Однією з найбільш актуальних задач, що стикаються багато з нас, є планування покупок.

Від продуктів харчування до предметів першої необхідності, список покупок є не тільки переліком товарів, які ми хочемо придбати, але й інструментом для ефективного управління нашими фінансами та часом. Чи плануємо ми свято або просто потрібно поповнити запаси в холодильнику, точне та організоване ведення списку покупок допомагає нам зберегти свій час та зосередитися на тому, що справді важливо.

Проте, у сучасному темпі життя, коли кожна мить має значення, паперові списки покупок втрачають свою актуальність та не забезпечують необхідної зручності. Створення та підтримка паперових списків може виявитися непростим завданням через необхідність постійного оновлення та переписування. Крім того, паперові списки не забезпечують можливості швидкої корекції або додавання нових елементів під час магазинних походів. Також вони вразливі до втрати або пошкодження, що може призвести до втрати важливої інформації.

Мобільні додатки для ведення списків покупок, натомість, пропонують безліч переваг. Вони забезпечують зручний інтерфейс для швидкого створення, оновлення та організації списків покупок. Можливість використання різних функцій, таких як нагадування, сортування та категоризація елементів, робить їх особливо корисними в повсякденному використанні. Крім того, мобільні додатки дозволяють з легкістю ділитися списками покупок з іншими користувачами, що робить

спільне планування покупок ще більш зручним.

Додатки для мобільних пристроїв, призначені для ведення списків покупок, часто надають користувачам додаткові функції, які сприяють полегшенню процесу планування та здійснення покупок. Наприклад, деякі з них мають вбудовані інструменти для сканування штрих-кодів, що дозволяє додавати товари до списків покупок швидко та безпомилково. Більшість таких додатків також пропонують можливість зберігання інформації про попередні покупки, що дає можливість легко повторити списки або використовувати минулі покупки як основу для нових. Ці додаткові функції значно підвищують зручність та продуктивність використання мобільних додатків для списків покупок порівняно з традиційними паперовими списками.

Отже, у світі, де мобільність та швидкість стають ключовими факторами, список покупок на смартфоні перетворюється на невід'ємну частину нашого щоденного життя. Він допомагає нам бути організованими, ефективними та економити наш час та зусилля в процесі здійснення покупок.

1. Огляд існуючих додатків для списків покупок на Android

1.1 Аналіз існуючих додатків для створення списків покупок на платформі Android.

У цьому розділі буде розглянуто та проаналізовано кілька популярних додатків для створення списків покупок на платформі Android. Вибір упав на п'ять найбільш відомих та часто завантажуваних додатків, що доступні в Google Play. Такий аналіз стане важливим кроком у процесі розробки власного додатку для списків покупок, оскільки дозволить краще зрозуміти потреби та вподобання користувачів, а також виявити сильні та слабкі сторони наявних рішень.

Перш ніж почати написання власного додатку, важливо здійснити глибокий аналіз конкурентів. Це дозволить виявити, що вже пропонує ринок та які можливості вже задовольняються. Аналізуючи додатки, які користуються популярністю, можна зрозуміти, що саме привертає користувачів, а також які аспекти можна поліпшити у власному продукті.

Одним із ключових завдань цього огляду є виявлення переваг та недоліків кожного додатку. Це дозволить зробити обґрунтований висновок про те, що працює добре в існуючих програмах та що можна покращити. Такий аналіз надасть важливі вказівки щодо того, які функції слід врахувати в новому додатку, а також які аспекти розробки потребують особливої уваги.

Крім того, аналіз конкурентів допоможе виявити нові можливості, які ще не були задіяні на ринку. Виявлення таких можливостей може стати ключовим фактором успіху власного додатку, який буде вирішувати раніше невирішені проблеми або задовольняти нові потреби користувачів.

Отже, цей аналіз є важливим кроком у процесі розробки додатку для списків покупок, який дозволить зрозуміти ринок, виявити конкурентні переваги та недоліки, а також визначити ключові функції для успішного впровадження нового продукту.

Вибір додатків

Вибір якісних додатків, яким надають перевагу користувачі, допоможе розробити приємний та зручний інтерфейс, який не заплутує користувача і забезпечить швидке та зрозуміле формування списку покупок.

В процесі вибору додатків для аналізу було враховано декілька ключових критеріїв. Перше, на що потрібно звернути увагу, це кількість завантажень кожного додатка. Популярність додатка свідчить про його ефективність та зручність використання, було вибрано ті додатки, які мали значну кількість завантажень.

Другим важливим критерієм були відгуки користувачів. Для обрання додатків у розгляд враховувалися лише ті, які мали позитивні відгуки. Це дозволить підвищити ймовірність того, що обрані додатки будуть відповідати вимогам та очікуванням користувачів.

Остаточний вибір додатків базувався на їх рейтингу. Віддавалася перевага додаткам з високим рейтингом, оскільки це може вказувати на їх якість та ефективність.

Такий підхід до вибору додатків забезпечує об'єктивність та стабільність аналізу, оскільки обрані додатки вже довели свою якість у спільноті користувачів платформи Android.

Listonic

Додаток Listonic не лише спрощує створення списку покупок, але й надає користувачам ряд корисних функцій для ефективного управління ними. Однією з цих функцій є можливість поділитися списками з друзями та родиною прямо з додатка. Це дозволяє спільно планувати покупки та розподіляти обов'язки між учасниками, що значно спрощує виконання щоденних справ вдома. Поділитися списками можна не лише для підготовки щоденних закупок, а й для планування спеціальних подій, таких як дні народження чи святкові заходи.

Крім того, Listonic надає можливість спільного доступу до списків покупок через будь-які пристрої. Наприклад, уявіть ситуацію, коли ви плануєте сімейний вечір або святкове заходи. Ви можете створити список покупок у додатку Listonic та поділитися ним з родиною чи друзями. Кожен учасник може додавати необхідні продукти, а ви миттєво побачите їхні зміни на своєму пристрої. Така можливість спільного доступу дозволяє всім учасникам бути в курсі поточного стану списку, що значно полегшує організацію та координацію закупівельних потреб.

У Listonic також реалізована інтелектуальна система підказок для швидкого створення списку. Додаток пропонує користувачам продукти, які вони часто покупають, забезпечуючи ефективніше та швидше складання списку. Також ви можете додати фотографії товарів, щоб точніше знати, які саме марки купувати, та вказати ціни для кожного елемента, контролюючи свій бюджет.

Окрім цього, Listonic підтримує голосовий ввід, що дозволяє додавати елементи в список голосом. Це особливо зручно, коли ви зайняті приготуванням їжі чи перебуваєте в дорозі. Додаток автоматично сортує елементи за категоріями, що полегшує збирання покупок у магазині та прискорює процес.

Завдяки функціоналу хмарного резервного копіювання користувач може не боятися втратити свій список. Не важливо, чи втратили телефон чи замінили його, списки покупок завжди залишаються в безпеці. Всі вони зберігаються в хмарі, що дозволяє користуватися ними на будь-якому пристрої. Це означає, що ви можете редагувати свій список покупок на смартфоні вдома, а потім переглянути його на своєму планшеті, коли ви виходите на покупки.

Крім того, Listonic використовує історію покупок, щоб створювати нові списки ще швидше. Понад 70% продуктів, які ми купуємо, залишаються однаковими час від часу. Тому додаток пропонує елементи, які ви раніше

купували, щоб полегшити процес створення списку. Наприклад, якщо ви часто додаєте в список молоко чи хліб, Listonic автоматично запропонує їх при наступному створенні списку, що зекономить ваш час та зусилля.

З цим додатком можна створювати будь-яку кількість списків для будь-яких подій. Додаток надає можливість створювати і розповсюджувати різні списки, включаючи списки для різних магазинів чи спеціальних заходів. Навіть можна створювати списки, не пов'язані з покупками, такі як контрольний список для новонароджених або список для переїзду.

Також список покупок завжди синхронізований між Android, iOS, ПК та іншими пристроями. Можна редагувати списки з будь-якого з них, включаючи веб-версію (Рис. 1). Крім того, додаток працює на більш як 40 мовах, що робить його доступним для користувачів по всьому світу.

З Listonic можна навіть користуватися списком покупок на смарт-годиннику. Додаток дозволяє створювати, редагувати та відмічати елементи безпосередньо на годиннику, що забезпечує максимальну зручність під час покупок.

Рис 1. *Можливості* додатку Listonic.

Завдяки корисному віджету завжди можна швидко та зручно отримати доступ до додатка Listonic, додаючи елементи прямо з головного екрана, включаючи голосовий ввід. Таким чином, Listonic стає вашим надійним помічником у плануванні покупок, який завжди залишається з вами та готовий допомогти у будь-який момент.

В цілому, Listonic - це інноваційний та корисний інструмент для

планування покупок, який допомагає економити час та гроші, забезпечуючи зручність і ефективність у щоденних справах.

SoftList

SoftList - це прогресивний додаток для організації покупок, що вирізняється простотою та зручністю використання, а також широким спектром корисних функцій. Він створений з метою полегшення повсякденних покупок користувачів та надання їм інструментів для ефективного управління списками товарів.

За допомогою SoftList користувачі можуть не лише створювати необмежену кількість списків покупок, а й легко редагувати їх, ділитися з іншими користувачами, що робить спільну роботу над покупками більш організованою та зручною для всіх учасників.

Однією з важливих функцій SoftList є можливість автоматичного розпізнавання товарів за допомогою сканування штрих-кодів. Це значно економить час користувачів та робить процес додавання товарів до списку максимально швидким та ефективним. Всі товари автоматично сортуються за категоріями, які можна налаштовувати. Крім того, є можливість створювати нові категорії і шаблони товарів.

SoftList пропонує широкий набір інтуїтивно зрозумілих функцій, які гарантують швидкість та зручність використання. Він дозволяє користувачам створювати списки покупок швидко та без зайвих ускладнень, а також додавати до них різноманітні деталі, такі як ціна, одиниця виміру, категорія товару, замітки та фотографії. Користувачі можуть легко вести контроль за своїми покупками, додавши до списків ціни товарів і дозволяючи додатку SoftList автоматично розраховувати загальну вартість покупки.

Крім цього, SoftList дозволяє зберігати історію покупок, що дозволяє користувачам відстежувати свої покупки в різні моменти часу та в різних магазинах. Це створює можливість для аналізу витрат та покращення

управління бюджетом.

Однією з ключових переваг додатку є можливість контролювати свої витрати за допомогою звітів і діаграм, які надаються для аналізу розходів. Такий підхід дозволяє користувачам детально вивчати інформацію про свої витрати та визначати, на які товари або категорії товарів витрачено найбільше коштів. Додаток також надає можливість використовувати хмарні сервіси для синхронізації даних між різними пристроями, поділу списків з іншими користувачами та автоматичного резервного копіювання даних. Це забезпечує зручність та безпеку використання додатку для користувачів.

Bring!

Це додаток для складання списків покупок, що переповнений не лише зручністю у формуванні списків, але й рядом інноваційних функцій, що роблять процес планування покупок ще приємнішим та ефективнішим. Однією з найцікавіших особливостей є вбудована колекція рецептів, що постійно оновлюється, що дозволяє користувачам зберегти свої улюблені рецепти та навіть вибрати необхідні інгредієнти безпосередньо з додатку. Наприклад, коли ви вирішуєте приготувати піцу, достатньо зайти в розділ рецептів, обрати потрібний рецепт, і весь список інгредієнтів автоматично додасться до вашого списку покупок. Це дуже зручно, оскільки не потрібно шукати папірці з рецептом або записувати інгредієнти окремо, все відразу під рукою та у зручному форматі.

Не менш важливою є можливість створювати декілька списків одночасно, розділяючи їх за категоріями, такими як продукти або ліки. Кожен список має привабливе візуальне оформлення, а також можна вибрати із понад 350 наочних значків продуктів, що робить списки ще більш зрозумілими та зручними для використання.

Завдяки цим функціям користувачі можуть легко організовувати свої покупки за категоріями, що спрощує процес планування та забезпечує

більшу структурованість у формуванні списків. Наприклад, можна створити окремий список для продуктів, які потрібно купити для приготування обіду, та інший список для необхідних медикаментів чи побутових товарів. Кожен список буде мати своє унікальне візуальне оформлення та можливість вибору символу, що робить їх легко відрізняються один від одного.

Завдяки системі профілів з загальним доступом та синхронізацією з хмарним сховищем, користувачі можуть отримати доступ до списків покупок з будь-якого пристрою та в будь-який час. Це означає, що навіть якщо ви змінили свій пристрій або хочете переглянути список зі свого комп'ютера, ви завжди можете зручно переглянути або оновити свої покупки.

Крім цього, додаток регулярно нагадує про необхідність відвідати магазин, і має вбудовану систему профілів з загальним доступом та синхронізацією з хмарним сховищем. Це дозволяє користувачам завжди залишатися організованими та в курсі своїх покупок, навіть якщо вони змінюються або оновлюються.

У додатку доступно багато налаштувань персоналізації, таких як темна та світла теми, різні варіанти вигляду списку (плитка або список з місцем для опису). Bring! дозволяє адаптувати додаток до власних уподобань та потреб користувача, надаючи можливість вибору оптимальних параметрів, які відповідають вашому стилю та представленню інформації.

Купи батон!

Призначений для планування покупок, додаток "Купи Батон!" для Android створений з урахуванням потреб користувачів у зручності та ефективності. Він пропонує відмінну альтернативу традиційним паперовим спискам, дозволяючи організувати та керувати списками покупок безпосередньо з мобільного пристрою. Немає потреби хвилюватися про те, що забудете або загубите папір зі списком, оскільки весь перелік товарів знаходиться в телефоні.

Один обліковий запис дозволяє користувачам спільно працювати над списками разом з родиною або друзями, забезпечуючи швидку та автоматичну синхронізацію всіх змін між пристроями. Це означає, що ви можете оновлювати список покупок на своєму смартфоні, а ваш партнер чи члени сім'ї будуть бачити ці зміни миттєво на своїх пристроях, що спрощує спільне планування і покупки.

Крім того, додаток надає можливість керувати списками покупок через веб-інтерфейс, що дозволяє зручно редагувати списки на більшому екрані комп'ютера. Це особливо корисно, коли потрібно внести багато змін або коли користувачу простіше працювати зі списками на комп'ютері.

Автоматична синхронізація забезпечує, що будь-які зміни, зроблені через мобільний додаток або веб-інтерфейс, відображатимуться на всіх пристроях. Це забезпечує єдність даних і спрощує спільне планування покупок, уникнення пропускання елементів або дублювання інформації. Наприклад, якщо один учасник додав новий елемент до списку покупок через мобільний додаток, інші учасники зможуть побачити цю зміну миттєво через веб-інтерфейс, що дозволяє всій групі бути завжди в курсі оновлень та координації дій.

Користувачі також мають зручну можливість ділитися списками покупок з різними групами людей, надаючи доступ до всіх списків або окремих списків лише обраним користувачам. Наприклад, якщо ви плануєте організувати вечірку або родинний захід, ви можете легко поділитися списком необхідних продуктів з усіма учасниками, забезпечуючи гнучкість та зручність у спільній організації закупівель. Це створює можливість ефективного контролю за тим, хто має доступ до якої інформації, і дозволяє керувати покупками з урахуванням потреб і переваг кожного учасника. Такий підхід сприяє зручності та організації спільних заходів, а також сприяє ефективній координації всіх процесів підготовки.

Твій список покупок

Додаток "Твій список покупок" - це інструмент, який робить планування покупок швидким та зручним. З його допомогою можна створювати, редагувати та ділитися списками покупок з легкістю. Основні можливості додатку включають стабільну роботу в режимі офлайн, велику базу товарів з розширеним набором продуктів і візуальне відображення категорій за допомогою кольорів і зображень. Це особливо зручно, коли ви перебуваєте в магазині і потрібно швидко знайти потрібний товар серед великої кількості елементів.

Зручний пошук продуктів дозволяє швидко знаходити необхідні товари та вказувати їх кількість. Голосове введення не лише спрощує процес формування списків, але й економить час. Користувач може додавати продукти голосом, навіть якщо не має можливості писати, наприклад, коли користувач зайнятий готуванням або перебуває в дорозі. Такий метод вводу дозволяє ефективно використовувати час і зробити процес планування покупок ще більш зручним і приємним.

Для швидкості і зручності використання програма запам'ятовує ваші часті покупки, які можна легко використовувати в майбутньому. Це особливо зручно, коли користувач має свої улюблені товари або регулярно купує одні й ті ж продукти.

Додаток також дозволяє створювати книгу рецептів, додавати рецепти з Інтернету або вводити їх вручну. Також додаток дозволяє редагувати існуючі категорії продуктів та додавати свої власні. Чистка, видалення та копіювання списків здійснюється всього одним натисканням. Налаштування персоналізації дозволяють вам адаптувати інтерфейс додатку до вашого смаку та потреб. Ви можете налаштовувати колірну схему, шрифти та розташування елементів так, як зручно користувачу (Рис. 1.1).

Рис. 1.1 Налаштування інтерфейсу

Найкраще в усьому цьому - відсутність постійної реклами та обмежень у доступі до всіх функцій безкоштовно. Користувачу більше не доведеться перейматися нав'язливою рекламою, яка часто відволікає від користування додатком, чи обмеженнями, які заважають повністю використовувати його потенціал. Користувач може насолоджуватися всіма перевагами програми без будь-яких перешкод, що дозволяє вам ефективно та комфортно використовувати додаток для планування покупок та інших завдань.

1.2 Архітектурні підходи у додатках списку покупок

Операційні системи, які зараз використовуються в смартфонах, представляють різноманітність інтерфейсів та функціоналу, але найбільшою популярністю користуються Android та iOS.

Android, розроблений компанією Google, є найпоширенішою операційною системою у світі. Він використовується в широкому спектрі пристроїв, від доступних моделей до флагманських смартфонів, а також в планшетах, телевізорах, автомобільних системах та інших пристроях. Android відкритий для розробників і користувачів, що дозволяє налаштовувати пристрої та використовувати різноманітні додатки з магазину Google Play.

iOS - операційна система, розроблена компанією Apple, яка виключно використовується в їхніх пристроях, зокрема iPhone, iPad та iPod Touch. iOS відрізняється від Android більш закритою екосистемою, де кожен аспект пристрою та додатку контролюється Apple. Це забезпечує високий рівень безпеки та стабільності, але обмежує можливості користувачів та розробників щодо налаштувань та вибору додатків.

Крім того, є інші операційні системи, такі як KaiOS, яка використовується в деяких дешевих смартфонах та кнопкових телефонах, а також Windows Phone, яка була припинена компанією Microsoft у 2019 році через низьку популярність та відсутність конкурентоспроможності.

Щодо вибору Android для розробки додатку, на це вплинуло кілька факторів. По-перше, Android є найпопулярнішою операційною системою для смартфонів, займаючи значну частку ринку. Це означає, що додаток буде доступний для широкого кола користувачів, що збільшує його потенційну аудиторію.

Крім того, Android має велику спільноту розробників та різноманітні пристрої, що робить його привабливим для розробки додатків. Операційна система Android також надає більшу свободу для розробників у виборі

інструментів та рішень для розробки, що може сприяти швидкому та ефективному процесу створення додатків.

У виборі архітектурного підходу для розробки Android-додатків, такого як MVVM (Model-View-ViewModel) в поєднанні з Clean Architecture, Android надає розробникам значну гнучкість та можливості для побудови добре структурованих та легко супроводжуваних додатків. MVVM сприяє розділенню логіки додатку на рівні моделі, представлення та моделі представлення, що спрощує розробку та тестування окремих компонентів додатку.

Застосування Clean Architecture додає до цього підходу принципи розділення додатку на різні рівні абстракції (або шари), такі як представлення, домен та джерела даних. Це дозволяє створювати додатки, що є незалежними від конкретних технологій та легко модифікувати.

Використання таких архітектурних підходів робить процес розробки ефективним і допомагає зберігати код організованим. Це дозволяє розробникам швидше реагувати на зміни в вимогах та підтримувати якість коду на високому рівні протягом усього життєвого циклу додатку. Крім того, чітко визначені межі між компонентами сприяють більшій розширюваності та перевикористанню коду, що дозволяє ефективно вести розробку навіть у великих проектах (додаток D., рис. 1.2.).

MVVM дозволяє відокремити представлення (Activities та Fragments) від бізнес-логіки, що сприяє збереженню чистоти коду та зручності в його розробці.

Бізнес-логіка - це сукупність правил, процесів і обчислень, які визначають основні функціональні вимоги та ділові процеси програмного забезпечення. Вона відображає логіку і алгоритми, які використовуються для обробки даних, взаємодії з користувачем та виконання бізнес-операцій. Бізнес-логіка визначає, які дії потрібно виконати, щоб досягти поставлених цілей та вирішити бізнес-задачі.

У контексті програмної архітектури, бізнес-логіка включає в себе усі правила, процеси та умови, які визначають функціональність додатку. Вона часто відокремлюється від рівнів презентації та даних, щоб забезпечити чітку структуру та підтримувати принципи чистої архітектури.

Починаючи з невеликого проекту, реалізація лише MVVM може бути достатньою. Однак при зростанні проекту стає складніше управляти View Model. View Model або viewpoints framework у системній інженерії, розробці програмного забезпечення та корпоративній інженерії — це структура, яка визначає узгоджений набір уявлень, які будуть використовуватися при створенні архітектури системи, архітектури програмного забезпечення або архітектури підприємства (додаток D., рис. 1.3.).

І в даному випадку на допомогу приходить Clean Architecture.

Clean Architecture дозволяє розділити код на шари з одним правилом залежності: внутрішні шари не повинні залежати від зовнішніх. Це дозволяє створювати більш розширюваний та легко змінюваний код. Приклад використання Clean Architecture, де внутрішні шари не залежать від зовнішніх, можна побачити в додатку для управління списками покупок.

У Clean Architecture основні компоненти - Presentation Layer, Domain Layer і Data Layer. Presentation Layer відповідає за відображення інформації користувачу та обробку його взаємодії з додатком. Domain Layer включає бізнес-логіку, яка не залежить від конкретної реалізації взаємодії з даними або інтерфейсу користувача. Data Layer відповідає за доступ до даних і

реалізує способи їх зберігання та отримання.

У відповідності з принципами Clean Architecture, кожен з цих шарів має свої внутрішні абстракції та не залежить від деталей реалізації інших шарів. Наприклад, Presentation Layer може використовувати інтерфейси або моделі з Domain Layer для відображення даних користувачу, не залежно від того, які конкретні дані отримані з Data Layer. Domain Layer може містити визначення різних виконавчих викликів (use cases), які використовуються Presentation Layer, і вони будуть незалежними від того, як саме дані будуть отримані або збережені.

Застосування Clean Architecture є критично важливим для додатків списку покупок, оскільки вони зазвичай потребують постійного розширення функціоналу та взаємодії з різними даними. Clean Architecture допомагає зберігати код організованим і розширюваним, оскільки шари архітектури не залежать один від одного.

Шарована структура Clean Architecture розділяє додаток на незалежні компоненти, такі як представлення, домен та джерела даних. Це означає, що логіка додатку залишається чистою та розділеною, що дозволяє змінювати або додавати функціональність без необхідності модифікувати всю систему. Наприклад, якщо ви хочете додати нову функцію до списку покупок, вам не потрібно змінювати логіку взаємодії з базою даних або інші частини додатку.

Це дозволяє розробникам легко тестувати окремі компоненти додатку і змінювати їх, не переймаючись впливом на інші частини програми. Такий підхід допомагає зберігати код додатку організованим і спрощує процес розробки, особливо в умовах постійних змін вимог і швидкого розвитку програмного забезпечення.

Використання MVVM з Clean Architecture у проектуванні інтерфейсу додатків списку покупок дозволяє зосередитися на важливих аспектах розробки, таких як дизайн та функціональність, та забезпечує зручність та

швидкість впровадження нових змін.

Архітектурний підхід Clean Architecture, як і будь-яка інша система, має свої позитивні та негативні аспекти, які варто ретельно враховувати при розробці програмного забезпечення. Для визначення його придатності у конкретному проекті важливо оцінити, які саме переваги та обмеження він може принести (Рис. 1.4.).

Переваги	Недоліки
Код розділений. Бізнес-логіка повністю відокремлена від інтерфейсу користувача форми.	Потрібен час, щоб звикнути. Ознайомлення з усіма різними шарами на початковому етапі може виявитися складним для деяких користувачів.
Тестування коду стає легким завдяки цій архітектурі.	
Навігація по структурі пакетів дуже проста.	Вимагає додавання додаткових класів, тому, якщо проект невеликого масштабу з невеликою складністю, вибір цієї архітектури не є рекомендованим.
Проект буде простим у супроводі.	
Додавання нових функцій до проекту буде простим.	

Рис. 1.4. Таблиця аналізу архітектури

Рівень презентації

Рівень презентації у Clean Architecture - це шар програмного забезпечення, який відповідає за відображення інформації користувачу та обробку його взаємодії з додатком. У цьому шарі зосереджені дії, фрагменти та моделі представлення. Цей шар відіграє ключову роль у забезпеченні користувачеві зручного та ефективного інтерфейсу, що дозволяє здійснювати різноманітні дії в додатку. Важливо відзначити, що в діях не міститься ніякої частки бізнес-логіки, їх основна мета - це взаємодія з користувачем та відображення даних у зручному форматі.

Програма переходить від дії до моделі представлення та від моделі представлення до рівня домену, щоб виконати необхідні дії. Цей підхід дозволяє чітко розділити рівні програми та забезпечити їх незалежність один

від одного. Така структура полегшує розробку, тестування та модифікацію додатку в подальшому.

Для кращого розуміння можна розглянути приклад, де передаються два варіанти використання до ViewModel. Використання (UseCase) - це, в основному, дія, що визначає, як відбувається зв'язок між ViewModel та даними. Це дозволяє відокремити презентаційний шар від рівня даних і домену, забезпечуючи чистоту та розширюваність коду. (Рис. 1.5).

```
class PostListViewModel(
        val useCaseHandler: UseCaseHandler,
        val getPosts: GetPosts,
        val savePost: SavePost): ViewModel() {
    fun getAllPosts(userId: Int, callback:
PostDataSource.LoadPostsCallback) {
        val requestValue = GetPosts.RequestValues(userId)
        useCaseHandler.execute(getPosts, requestValue, object :
        UseCase.UseCaseCallback<GetPosts.ResponseValue> {
            override fun onSuccess(response: GetPosts.ResponseValue)
{
                callback.onPostsLoaded(response.posts)
            }
            override fun onError(t: Throwable) {
                callback.onError(t)
            }
        })
    }
    fun savePost(post: Post, callback:
PostDataSource.SaveTaskCallback) {
        val requestValues = SavePost.RequestValues(post)
        useCaseHandler.execute(savePost, requestValues, object :
        UseCase.UseCaseCallback<SavePost.ResponseValue> {
            override fun onSuccess(response: SavePost.ResponseValue)
{
                callback.onSaveSuccess()
            }
            override fun onError(t: Throwable) {
                callback.onError(t)
            }
        }
    }
}
```

Рис. 1.5. Код варіанти використання ViewModel

Рівень домену

Рівень домену в Clean Architecture - це шар програмного забезпечення, який містить усі варіанти використання проекту. Основною метою цього шару є визначення бізнес-логіки та правил, які стосуються додатку.

Для кращого розуміння можна розглянути приклад абстрактного класу - UseCase. Використання (UseCase) - це абстрактний клас або інтерфейс, який визначає конкретні дії або операції, які можуть бути виконані в додатку. Наприклад, використання може включати створення нового користувача, отримання списку покупок або розрахунок вартості покупки.

Рівень домену дозволяє відокремити бізнес-логіку від презентаційного шару та рівня даних, що спрощує тестування, розширення та зміну програмного забезпечення. Він є ключовим компонентом Clean Architecture, який дозволяє підтримувати чистоту та структурованість коду проекту. (Рис. 1.6)

```
abstract class UseCase<Q : UseCase.RequestValues, P :
UseCase.ResponseValue> {
    var requestValues: Q? = null
    var useCaseCallback: UseCaseCallback<P>? = null
    internal fun run() {
        executeUseCase(requestValues)
    }
    protected abstract fun executeUseCase(requestValues: Q?)
    /**
     * Data passed to a request.
     */
    interface RequestValues
    /**
     * Data received from a request.
     */
    interface ResponseValue
    interface UseCaseCallback<R> {
        fun onSuccess(response: R)
        fun onError(t: Throwable)
    }
}
```

Рис. 1.6. Код абстрактний клас UseCase

Використання (UseCase) - це місце, де приймаються рішення щодо виконання конкретного варіанту використання у фоновому потоці та отримання відповіді у основному потоці. У цьому шарі відбувається виклик методів, які визначені у варіанті використання, та обробка результатів їх виконання.

Отже, UseCase діє як посередник між рівнями Presentation та Data. Він відділяє бізнес-логіку від деталей реалізації взаємодії з даними і представлення даних користувачу. Якщо потрібно додати або видалити функції під час розвитку проекту, можна просто додати або видалити відповідні варіанти використання. Це не вплине на інші частини системи, що дозволяє легко керувати кодом і зручно відлагоджувати його.

Таким чином, кожен варіант використання для кожної функції допомагає зберігати код організованим і розширюваним, оскільки вони є незалежними один від одного і можуть бути змінені або розширені без

впливу на інші частини додатка.

Рівень даних

Рівень даних - це компонент архітектури, який містить усі репозиторії, що можуть використовуватися доменним рівнем додатка. В основному, цей рівень виступає як інтерфейс для зовнішніх класів, які надають доступ до даних з різних джерел.

У рівні даних визначаються методи для отримання, збереження та оновлення даних. Ці методи викликаються доменним рівнем для взаємодії з різними джерелами даних, такими як локальна база даних, віддалений сервер або інші зовнішні джерела.

Репозиторії у рівні даних відповідають за управління доступом до даних і надають інтерфейс для роботи з ними. Вони ізолюють доменний рівень від конкретних деталей реалізації джерел даних, що дозволяє змінювати або розширювати ці джерела без впливу на решту системи.

Отже, рівень даних виконує важливу роль у забезпеченні незалежності доменного рівня від деталей реалізації джерел даних і сприяє створенню розширюваних та легко змінюваних додатків. (додаток D., рис. 1.7.).

2. Проектні і технічні рішення

2.1. Архітектура мобільних додатків

У загальній архітектурі мобільних додатків використовуються три ключові рівні: Презентація, Бізнес та Дані. Кожен з цих рівнів виконує важливу роль у структурі та функціонуванні додатка. Рівень Презентації відповідає за інтерфейс користувача (UI), який взаємодіє з користувачем та відображає необхідну інформацію. Він включає в себе елементи, такі як екрани, кнопки, текстові поля та інші візуальні компоненти, що створюють користувацький досвід.

Рівень Бізнесу в архітектурі додатків відповідає за управління бізнес-логікою та обробку даних. Цей рівень виконує ключову роль у визначенні та реалізації основних функцій додатка, що відповідають за бізнес-процеси та логіку застосунку.

На рівні Бізнесу розташовані бізнес-компоненти, які відповідають за обробку даних, виконання бізнес-правил і логіку додатка. Ці компоненти взаємодіють з іншими рівнями архітектури для забезпечення правильного функціонування програмного забезпечення.

Рівень Бізнесу включає в себе такі компоненти, як сервіси, інтерактори або менеджери, які виконують обробку даних, виконують бізнес-логіку та забезпечують правильну реакцію на користувацькі дії. Вони реалізують основні функції додатка, включаючи обробку вхідних даних, взаємодію з базою даних, валідацію та обчислення.

Цей рівень є ключовим для правильного функціонування додатка, оскільки він визначає та реалізує основні бізнес-процеси та функціональність. Його правильна організація та реалізація допомагають забезпечити ефективність, надійність та масштабованість додатка.Нарешті, рівень Дані відповідає за управління та обробку даних в додатку. Він включає в себе роботу з базами даних, зберігання та витягування інформації,

а також забезпечує доступ до даних з різних джерел. Цей рівень є важливим для забезпечення ефективної роботи додатка та збереження цілісності даних.

Трьохрівнева архітектура дозволяє забезпечити масштабованість та гнучкість додатків у випадку потреби змін. Крім того, вона є вартісно-ефективною у плані оновлень, а також надійною та швидкою в експлуатації. Нижче наведено діаграму архітектури мобільного додатка, яка ілюструє розподіл функцій між цими рівнями.. (Рис. 2.6).

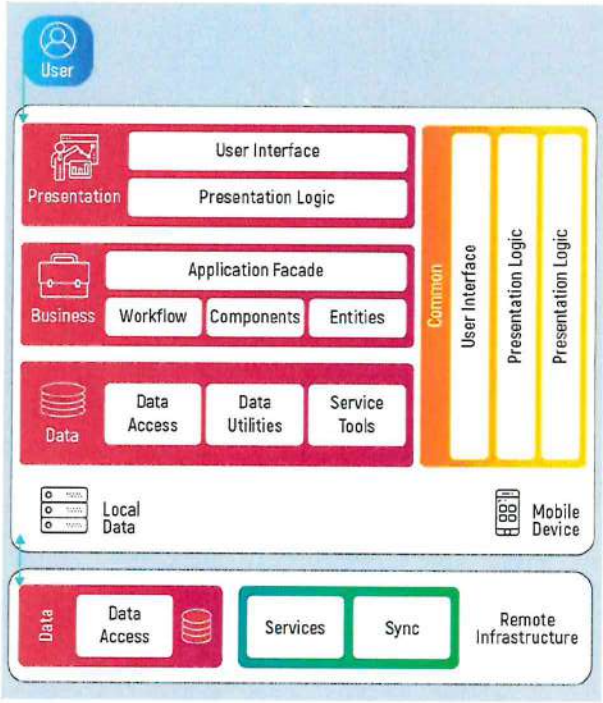


Рис. 2.4. Діаграма архітектури

Ця архітектура є ключовою для успішного розроблення та підтримки мобільних додатків, оскільки вона дозволяє ефективно розділити функціональність та забезпечити чітку структуру та розподіл обов'язків між різними компонентами додатка. Вона сприяє швидкому розвитку, простоті управління та підтримці, а також забезпечує високу якість та надійність мобільних додатків.

Архітектура мобільних додатків для платформи Android є складною системою, яка включає специфічні компоненти, такі як activities, fragments, services, content providers та broadcast receivers. Ці компоненти розроблені для оптимізації взаємодії додатка з операційною системою та апаратним

забезпеченням пристрою. Наприклад, класи activities відповідають за повну інтеграцію користувача з додатком, дозволяючи швидко реагувати на події, такі як вхідні дзвінки або сповіщення електронної пошти, навіть при відкритому додатку.

Фрагменти є ще одним важливим компонентом Android-додатків, які дозволяють розробникам побудувати гнучкий та адаптивний інтерфейс для різних пристроїв та умов. Вони працюють в межах activity і мають визначену функціональність, що дозволяє їх легко перевикористовувати в різних частинах додатку. Це дозволяє створювати додатки, які ефективно працюють як на смартфонах, так і на планшетах, а також пристосовуються до змін у розмірах та орієнтації екрану.

Зазвичай мобільні додатки для Android включають шар відображення з activity та fragments для реалізації інтерфейсу користувача (UI) та шар даних з API та cacheProvider для доступу до даних та їх зберігання. Ця архітектура дозволяє ефективно організувати роботу додатка та забезпечити зручний та швидкий доступ до інформації для користувачів (Рис. 2.4).

Таким чином, застосування цих компонентів дозволяє розробникам створювати потужні та ефективні мобільні додатки для платформи Android, які забезпечують зручну взаємодію з користувачем та надійну роботу в різних умовах та на різних пристроях.

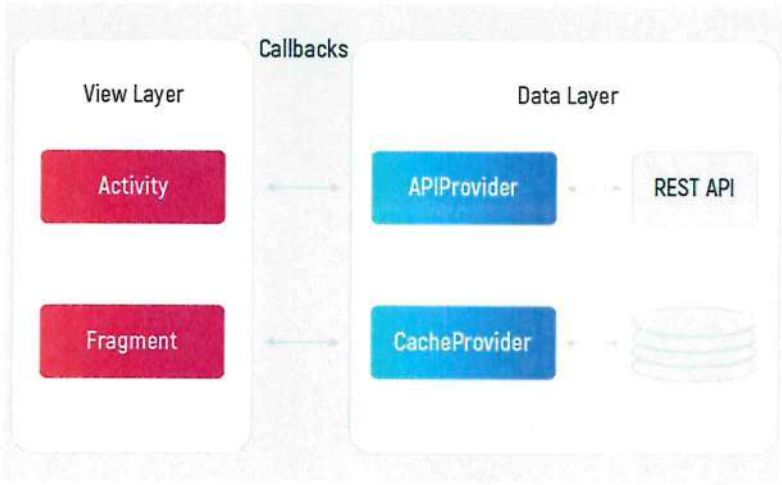


Рис. 2.4. Структура мобільних додатків для Android

Однією з ключових концепцій архітектури в Android є файл маніфесту додатка (App manifest), що визначає, як саме пристрій взаємодіє з додатком. Цей документ дозволяє налаштовувати параметри, визначаючи поведінку додатка на пристроїв. Основна мета маніфесту полягає у забезпеченні правильної інтеграції додатка з операційною системою Android та його коректної роботи в середовищі пристрою.

Під час використання додатка на пристрої очікується, що він буде взаємодіяти з ним зручним та швидким способом. Наприклад, за одну хвилину можуть трапитися різноманітні події: відкриття фото-редактора для редагування зображень, створення фотографії з особливим ефектом, завантаження зображення з редактора та поділ цього зображення у трьох соціальних мережах. Додаток спілкується з камерою пристрою, здійснює доступ до файлового сховища для зберігання зображень та взаємодіє з API соціальних мереж для публікації контенту.

Одним з ключових аспектів цієї взаємодії є безперервна робота додатка та його здатність забезпечити швидке та безперешкодне виконання операцій, таких як публікація зображень у соціальних мережах, без необхідності виходу користувача зі свого облікового запису. Тому важливо, щоб додаток був добре налаштований та міг ефективно взаємодіяти з різними ресурсами пристрою та зовнішніми сервісами для забезпечення комфортного

користування.

Помилки в архітектурі Android-додатків можуть значно вплинути на їх ефективність та надійність. Некоректне встановлення або використання компонентів архітектури може призвести до проблем у роботі додатка. Наприклад, неправильне використання активностей, служб та отримувачів трансляцій як джерел даних може породити проблеми з обробкою та передачею інформації між різними частинами додатка.

Додатково, руйнування меж між модулями та порушення їх відповідальностей може спричинити заплутаність та непрозорість у коді, що ускладнює розуміння та супровід програмного забезпечення.

Крім того, важливо враховувати особливості роботи додатка в офлайн-режимі на етапі планування. Деякі додатки можуть вимагати доступу до мережі для повноцінної роботи, тоді як інші повинні бути здатні працювати без Інтернет-з'єднання. Це особливо важливо для додатків, які пропонують функціонал, що потребує обміну даними з сервером. Вирішення цих питань на етапі планування дозволяє забезпечити максимальну доступність та зручність для користувачів, незалежно від умов їх використання.

Усе це підкреслює значення створення Android-додатків з урахуванням тестовості, що передбачає винос логіки в окремий модуль та організацію мережевої взаємодії та інших аспектів для полегшення тестування. Такий підхід дозволить розробникам створювати надійні та ефективні додатки, які задовільнять потреби користувачів у зручній та продуктивній роботі.

2.2 Технології реалізації функціональності.

Технології, використовувані при реалізації функціоналу мобільних додатків для платформи Android, відіграють важливу роль у створенні високоякісних та ефективних програмних продуктів. Цей процес включає в себе використання різноманітних інструментів, мов програмування та архітектурних підходів, які дозволяють розробникам ефективно створювати, тестувати та розгортати додатки.

Реалізація функціоналу мобільних додатків вимагає від розробників глибокого розуміння платформи Android та навичок у використанні різних інструментів. Важливо враховувати особливості мобільного середовища, такі як обмежені ресурси, різноманітність пристроїв та розмір екранів, підвищені вимоги до безпеки та ефективності роботи додатків у реальному часі.

У цьому контексті розробники використовують різноманітні інструменти, які дозволяють їм створювати додатки з високою продуктивністю та надійністю. Мови програмування, такі як Java та Kotlin, надають розробникам потужний інструментарій для створення додатків різного рівня складності. Використання Android SDK разом з Android Studio дозволяє швидко розробляти, тестувати та розгортати додатки, забезпечуючи при цьому високу якість коду та продуктивність розробки.

Окрім того, архітектурні підходи, такі як MVVM (Model-View-ViewModel) та Clean Architecture, дозволяють структурувати додатки таким чином, щоб вони були легко супроводжуваними та розширюваними. Це особливо важливо для реалізації функціоналу додатків, таких як список покупок, які часто потребують змін та доповнень з плином часу.

Java та Kotlin

Java була визнаною стандартною мовою програмування для розробки Android-додатків протягом багатьох років. Її довга історія використання та широкий спектр ресурсів і підтримки забезпечили високий рівень надійності та стабільності для розробників. З Java розробники отримували доступ до всіх можливостей Android SDK і API, що сприяло створенню різноманітних додатків для різних потреб користувачів.

Однак з появою Kotlin, ситуація змінилася. Kotlin, який є сучасною мовою програмування, розроблений компанією JetBrains, набуває все більшої популярності серед розробників Android-додатків. Він пропонує простіший синтаксис, більшу безпеку типів, підтримку функціонального програмування та інші продуктивні можливості, які полегшують розробку.

Однією з головних переваг Kotlin є його інтероперабельність з Java. Це означає, що розробники можуть поступово переходити до Kotlin, додаючи новий код Kotlin до існуючого проекту на Java, або навпаки. Це дозволяє плавно оновлювати додатки і використовувати переваги обох мов.

Обидві мови, Java і Kotlin, надають розробникам широкі можливості для створення ефективних і функціональних Android-додатків. Вибір між ними зазвичай залежить від індивідуальних уподобань, досвіду розробника та особливостей конкретного проекту.

Android SDK (Software Development Kit)

Android SDK - це комплекс інструментів, які використовуються для розробки мобільних додатків для платформи Android. Він складається з різноманітних компонентів, серед яких API (Application Programming Interface), інструменти розробки та документація.

API - це набір програмних інтерфейсів, які надають доступ до функцій операційної системи Android. За допомогою цих інтерфейсів розробники можуть взаємодіяти з різними складовими системи, такими як графіка, мережа, бази даних тощо.

Інструменти розробки включають в себе Android Studio - офіційне інтегроване середовище розробки для Android, SDK Manager - інструмент для управління версіями SDK та залежностями, а також різноманітні додаткові утиліти для створення, тестування та оптимізації додатків.

Документація, яка надається разом з Android SDK, містить інформацію про різні аспекти розробки додатків для Android, включаючи опис API, приклади коду, рекомендації щодо дизайну тощо. Ця документація допомагає розробникам краще зрозуміти особливості платформи та ефективно використовувати її можливості у своїх проектах.

Загалом, Android SDK є ключовим інструментом для розробки мобільних додатків для платформи Android і забезпечує необхідні ресурси та інструменти для успішної розробки, тестування та впровадження додатків.

Android Studio

Android Studio є основним інструментом для розробки мобільних додатків для платформи Android і представляє собою інтегроване середовище розробки (IDE), спеціально розроблене компанією Google. Це потужне інструментарій, призначений для створення, налагодження та тестування програмного забезпечення для мобільних пристроїв, що працюють під управлінням операційної системи Android.

Однією з основних переваг Android Studio є його зручний інтерфейс, який дозволяє розробникам швидко і ефективно писати код. Інтегроване середовище надає широкий набір інструментів, включаючи розумний редактор коду з автодоповненням, функцію перегляду коду, можливість рефакторингу, а також підтримку мови Kotlin, яка є додатковим варіантом мови програмування поряд з Java.

Крім того, Android Studio включає в себе інтегровану систему збірки та розгортання додатків, що дозволяє розробникам легко створювати APK-файли для розповсюдження на різних пристроях. Інструмент також надає можливості тестування додатків на вбудованих емуляторах або реальних пристроях, що дозволяє виявляти та виправляти помилки перед релізом.

Однією з важливих особливостей Android Studio є його постійне оновлення та підтримка з боку Google. Компанія постійно вдосконалює інструмент, додаючи нові функції та виправляючи помилки, що дозволяє розробникам тримати крок з останніми тенденціями розробки мобільних додатків.

XML (Extensible Markup Language)

XML є стандартною мовою розмітки, яка використовується для створення макетів інтерфейсу користувача (UI) в Android-додатках. Вона забезпечує простий та зручний спосіб описувати розташування та вигляд різних елементів інтерфейсу, що відображається на екрані пристрою.

Основна перевага використання XML для розробки макетів полягає в його декларативному підході. Розробники можуть описати елементи інтерфейсу та їх властивості за допомогою простого текстового файлу, вказуючи розміщення, розміри, кольори, шрифти та інші атрибути. Це дозволяє швидко створювати складні макети без необхідності програмування кожного елемента окремо.

Крім того, використання XML дозволяє відокремлювати логіку

програми від її представлення. Розмітка інтерфейсу окремо від коду логіки додатка дозволяє розробникам працювати паралельно та забезпечує більшу зрозумілість та підтримку коду.

XML також підтримує різні типи елементів і властивостей, що дозволяє створювати різноманітні інтерфейси, від простих списків і кнопок до складних макетів з вкладеними елементами, кастомними анімаціями та іншими ефектами. Використання XML спрощує розробку, підтримку та масштабування Android-додатків, забезпечуючи зручність та ефективність у створенні користувацького інтерфейсу.

Gradle

Gradle є потужним інструментом збирання проектів, який знаходить широке застосування в розробці Android-додатків. Він використовується для автоматизації процесу збирання, тестування та розгортання програмного забезпечення, забезпечуючи зручний та ефективний спосіб керування проектами (Рис. D. 2.5).

Однією з ключових переваг Gradle є його гнучкість і конфігуруємість. Він дозволяє розробникам налаштовувати різні аспекти збірки проекту, включаючи завантаження залежностей, налаштування середовища розробки та налаштування параметрів збірки. Це робить Gradle відмінним інструментом для вирішення різних завдань розробки, від простих проектів до складних і великих програм.

Крім того, Gradle інтегрується з різними іншими інструментами розробки, такими як Android Studio, що робить його особливо зручним для розробників Android-додатків. Він підтримує автоматичне визначення та завантаження залежностей з репозиторіїв Maven або інших джерел, а також дозволяє легко налаштовувати та налагоджувати різні етапи збирання проекту.

Узагальнюючи, Gradle є надійним інструментом для збирання та

керування проектами, який дозволяє розробникам зосередитися на написанні коду та вдосконаленні їх додатків, забезпечуючи при цьому швидкий та ефективний процес розробки.

Android Architecture Components

Android Architecture Components - це ключовий набір бібліотек, розроблений компанією Google, який пропонує рекомендовані підходи для побудови архітектури Android-додатків. Цей набір інструментів включає в себе різноманітні компоненти, такі як Lifecycle, ViewModel, LiveData та інші, які спрощують і поліпшують розробку додатків на платформі Android.

Однією з ключових переваг Architecture Components є їх спрямованість на покращення архітектурного дизайну додатків. Наприклад, Lifecycle компонент надає можливості для управління життєвим циклом додатку та його компонентів, що дозволяє розробникам ефективно керувати ресурсами та пам'яттю. ViewModel дозволяє зберігати та управляти даними, пов'язаними з інтерфейсом користувача, забезпечуючи чистоту коду та уникнення проблем, пов'язаних з втратою даних під час перевороту екрана. LiveData, з свого боку, забезпечує реактивну підтримку для сповіщення про зміни даних та автоматичну оновлення інтерфейсу користувача при їх зміні.

Крім того, використання Architecture Components сприяє розподілу функціональності в додатку, роблячи його більш модульним та легко змінюваним. Ці компоненти допомагають забезпечити чистоту коду, що полегшує розуміння та підтримку додатку в майбутньому. Вони спрощують процес розробки, роблячи код більш структурованим і керованим, що в свою чергу покращує продуктивність розробника та якість додатку.

SQLite та Room Persistence Library

Бази даних SQLite є одним з найпоширеніших інструментів для збереження даних у Android-додатках. SQLite - легка вбудована реляційна система керування базами даних, яка забезпечує ефективну та надійну

роботу з даними без значного використання ресурсів пристрою. Вона підтримує багато зручних функцій, таких як транзакції, індексація та обмеження цілісності даних.

Однак для спрощення взаємодії з базами даних і покращення безпеки та зручності розробки, використовуються бібліотеки, такі як Room Persistence Library. Room - це частина Android Architecture Components, яка надає вищий рівень абстракції для роботи з базами даних SQLite. Вона дозволяє розробникам працювати з базами даних на рівні об'єктів та запитів, а не прямо з SQL-запитами, що робить розробку більш зручною та безпечною.

Room дозволяє визначати структуру бази даних за допомогою анотацій у Java або Kotlin, а також автоматично генерує потрібний код для роботи з базою даних, включаючи об'єкти доступу до даних (DAO), базові класи та інші необхідні компоненти. Використання Room дозволяє забезпечити більшу стабільність та безпеку додатку, а також спрощує його розробку і підтримку.

3. Програма

3.1 Визначення платформи для створення додатка.

У сучасному світі мобільні телефони перетворилися на невід'ємну складову нашого повсякденного життя, відіграючи ключову роль у забезпеченні зв'язку, доступу до інформації та виконанні різноманітних завдань. Завдяки їм ми можемо бути завжди на зв'язку зі світом, спілкуватися з друзями та колегами, отримувати оновлення та повідомлення про події, а також використовувати різноманітні корисні додатки.

У цьому контексті мобільні додатки відіграють важливу роль, забезпечуючи максимальний комфорт і зручність для користувачів. Вони дозволяють нам ефективно керувати нашим часом, планувати справи, ведення списку покупок є одним з таких завдань, яке може значно полегшити

життя користувача. Враховуючи популярність мобільних пристроїв та потребу користувачів у зручних інструментах, розробники мобільних додатків створюють все більше і більше програм, щоб задовольнити ці потреби.

Серед різних видів мобільних додатків, список покупок виокремлюється як один з найбільш корисних і популярних. Незалежно від того, чи йдеться про продуктові покупки, покупки для дому або будь-які інші, мати зручний додаток, який завжди під рукою, дозволяє більш ефективно планувати та здійснювати покупки, не забуваючи ні про один товар. Тому розгляд мобільних платформ для створення додатку списку покупок, є логічним кроком для забезпечення його максимальної доступності для користувачів.

Розгляд можливості створення додатку для платформ Android і iOS обґрунтовується декількома ключовими аспектами. По-перше, обидві ці платформи мають велику кількість користувачів по всьому світу, що робить їх надзвичайно привабливими для розробки додатків. Android та iOS є двома найпоширенішими мобільними операційними системами, які використовуються мільйонами людей щодня. Це означає, що розробка додатків для цих платформ відкриває широкі можливості для досягнення великої аудиторії та успіху на ринку.

По-друге, Android і iOS відрізняються своєю глибокою інтеграцією з екосистемою виробника, що включає в себе пристрої, програмне забезпечення та послуги. Наприклад, Android є продуктом Google, тому він має тісну інтеграцію з різними сервісами Google, такими як Gmail, Google Drive та Google Maps. З іншого боку, iOS розробляється компанією Apple, що забезпечує його високу сумісність з іншими пристроями та сервісами Apple, такими як iCloud, iMessage та Siri. Це створює унікальні можливості для розробки додатків, які взаємодіють з існуючими технологіями та сервісами виробника.

Крім того, Android і iOS відрізняються своїм функціоналом, і це важливо врахувати при виборі платформи для розробки додатку. Android відомий своєю відкритістю та гнучкістю, що дозволяє розробникам створювати різноманітні додатки з різними функціями та можливостями. З іншого боку, iOS славиться своєю стабільністю та безпекою, що робить його привабливим вибором для розробки додатків з високими стандартами безпеки та стабільності.

На відміну від Android, iOS відомий своєю більш закритою системою, що може стати перешкодою для деяких розробників. Ця закритість включає обмеження доступу до певних системних ресурсів та інструментів, які можуть бути корисними для реалізації специфічних функцій або інноваційних рішень. Проте, варто відзначити, що ця закритість дозволяє забезпечити високий рівень безпеки та стабільності в додатках, розроблених для iOS. Це особливо важливо для додатків, які мають доступ до конфіденційної інформації або обробляють фінансові транзакції, де забезпечення приватності та безпеки є пріоритетним завданням.

Особисто я обираю платформу Android через її відкритість та простоту програмування. Відкрита система Android надає розробникам значну свободу у розробці додатків, дозволяючи використовувати різноманітні інструменти та ресурси для створення високоякісних програм. Це відкриває широкі можливості для інновацій та розвитку додатків, оскільки розробники можуть ефективно використовувати різноманітні технології та інтегрувати їх у свої проекти. Такий підхід сприяє більшій гнучкості та інноваціям у процесі розробки, що дозволяє створювати додатки, які відповідають унікальним потребам користувачів і відзначаються високою якістю.

Поруч з цим, слід зазначити, що на сьогоднішній день платформа Android є однією з найбільш актуальних та популярних серед користувачів (Рис. 3). За даними статистики, кількість користувачів Android значно перевищує кількість користувачів інших мобільних платформ, що робить

його привабливим вибором для розробки додатків. Враховуючи це, обрання Android для розробки додатку дозволить досягти більшої аудиторії та забезпечити йому широке охоплення серед користувачів.

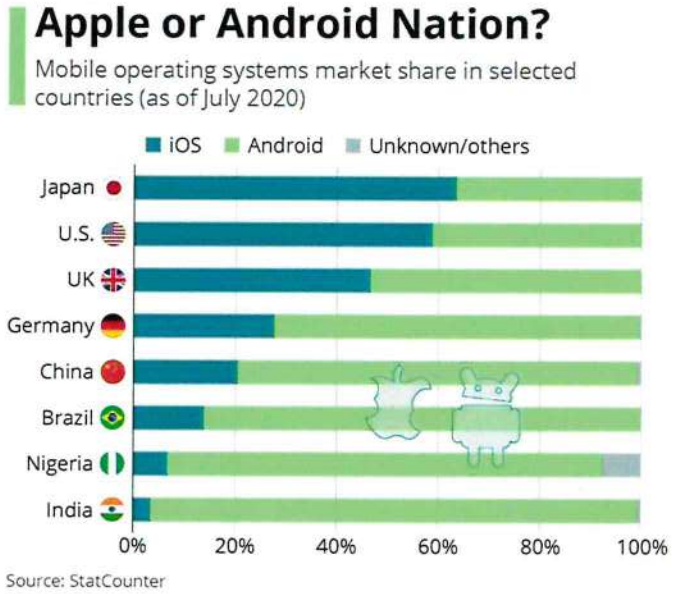


Рис. 3. Частка ринку мобільних операційних систем

3.2 Визначення мови програмування.

У сфері розробки додатків для платформи Android існує вражаюча різноманітність мов програмування, які розробники можуть використовувати для реалізації своїх ідей. Починаючи від традиційних та добре відомих мов, таких як Java, і до новітніх і прогресивних, наприклад, Kotlin, це надає широкий спектр вибору для розробників. Кожна мова має свої унікальні переваги та особливості, що робить процес розробки додатків ще цікавішим і захопливішим.

Основними мовами програмування для розробки додатків на Android є Java та Kotlin. Java, яка вже довгий час є стандартом для розробки Android-додатків, відома своєю надійністю та великою кількістю ресурсів та підтримки спільноти розробників. У той же час Kotlin, яка була офіційною мовою програмування для Android з 2017 року, привертає увагу своєю сучасністю, простотою та безпекою, що дозволяє створювати додатки більш швидко та ефективно.

Крім того, варто згадати про інші мови програмування, такі як C++, які можуть бути використані для розробки мобільних додатків за допомогою NDK. Використання C++ і NDK відкриває можливості для створення високопродуктивних частин додатків або використання специфічних функцій, які вимагають оптимізованої роботи з ресурсами пристрою. Наприклад, це може бути корисно для розробки графічно важких ігор або додатків, що вимагають високої швидкодії та продуктивності.

Використання C++ також дозволяє розробникам використовувати бібліотеки та інструменти, які є стандартом у сферах комп'ютерної графіки, обробки зображень та інших областях, що вимагають низькорівневого програмування. Такий підхід дозволяє оптимізувати продуктивність додатків та отримувати максимальний рівень контролю над їхнім функціоналом.

Вибір мови програмування для створення Android-додатків є важливим етапом у процесі розробки, і він часто залежить від індивідуальних уподобань та потреб розробника. Розглянувши переваги та недоліки мов програмування, таких як Java та Kotlin, можна зробити обґрунтований вибір на користь однієї з них.

Java відома своєю широкою популярністю та довірою спільноти розробників. Ця мова є однією з основних мов програмування для розробки Android-додатків і має досить великий стек інструментів та ресурсів для підтримки розробки. Однією з переваг Java є її стабільність та надійність. Розробники можуть розраховувати на те, що програми, написані на Java, працюватимуть стабільно та ефективно на різних пристроях. Крім того, Java є добре документованою мовою з великою кількістю відкритих бібліотек та інструментів, що спрощує розробку додатків.

Незважаючи на ці переваги, Java має свої обмеження та недоліки. Наприклад, вона може бути більш "тяжкою" мовою порівняно з Kotlin, що може призвести до більшого обсягу коду для досягнення тих самих результатів. Також, Java не має всіх новітніх функцій та можливостей, які

присутні у сучасних мовах програмування.

Насправді, вибір мови програмування залежить від конкретних потреб проекту та власних уподобань розробника. Однак, у багатьох випадках обирають Java через її широку підтримку, великий екосистему і надійність.

Java

Переваги

- Широке сприйняття: Java - це традиційна мова програмування для Android, що має велику кількість ресурсів та підтримку спільноти. Розробники, які володіють Java, можуть використовувати ці ресурси для вирішення різноманітних завдань.

- Надійність: Java відома своєю надійністю та стабільністю, що робить її популярним вибором для великих та складних проектів.

- Широкий досвід: Багато розробників мають досвід роботи з Java, що полегшує роботу в команді та обмін знаннями.

Недоліки:

- Обмежена продуктивність: У порівнянні з деякими іншими мовами, наприклад, Kotlin, Java може вимагати більшого обсягу коду для досягнення тих самих результатів.

- Відсутність сучасних функцій: Java може вважатися менш сучасною мовою порівняно з Kotlin через відсутність деяких сучасних функцій та синтаксичних полегшень.

Kotlin

Переваги:

- Короткий код: Kotlin відомий своєю простотою та зручністю, що дозволяє писати менше коду, щоб досягти тих самих результатів, що й за допомогою Java.

- Сучасність: Kotlin пропонує багато сучасних функцій та полегшень, що роблять процес розробки більш приємним та продуктивним.

- Сумісність з Java: Kotlin є повністю сумісним з Java, що дозволяє поступово переходити на цю мову без необхідності повного переписування вже наявного коду.

Недоліки:

- Велике навантаження на систему: Використання Kotlin може призвести до додаткового навантаження на систему та збільшення розміру додатку.

- Недостатній досвід користувачів: Оскільки Kotlin є новішою мовою програмування, ніж Java, деякі розробники можуть не мати достатнього досвіду в її використанні.

Обравши Java для розробки Android-додатків, я врахувала декілька ключових факторів, що переконали мене в її виборі. По-перше, важливо відзначити широке поширення цієї мови програмування в середовищі розробки мобільних додатків. Java вже давно є основною мовою для розробки Android-додатків, що створює додатковий рівень довіри серед розробників та користувачів.

Далі, важливим аспектом була надійність Java та її довірчий статус у розробницькому середовищі. Завдяки своїй стабільності та добре пророблені архітектурі, додатки, розроблені на Java, можуть працювати ефективно та надійно на різних пристроях і версіях операційної системи Android.

Крім того, важливим був широкий досвід та підтримка спільноти розробників. Java має значну кількість ресурсів, від відкритих бібліотек до документації та онлайн-курсів, що значно полегшує процес розробки та допомагає розробникам знайти відповіді на свої питання.

Усі ці фактори переконали мене в обранні Java як основної мови програмування для реалізації моїх проектів Android-додатків. Її поширення, надійність та широкий спектр ресурсів і підтримки роблять її оптимальним вибором для успішної розробки високоякісних додатків для мобільних платформ.

3.3 Проектування додатку.

Для ефективного проектування додатку "Список покупок" важливо ретельно розглянути основні вимоги та необхідну функціональність. Перед тим, як переходити до фази розробки, необхідно чітко визначити, які саме можливості будуть доступні користувачам. Належне розуміння цих аспектів дозволить створити додаток, який буде відповідати потребам користувачів та забезпечувати їм комфорт та зручність використання.

У додатку "Список покупок" користувачам буде доступно кілька основних можливостей, які роблять його корисним і зручним інструментом для планування покупок та управління списками. Серед цих можливостей:

- Додавання елементів до списку: Користувачі можуть додавати нові товари до списку покупок. Це може бути здійснено шляхом введення назви товару в текстове поле або за допомогою голосового вводу.
- Видалення елементів зі списку: Користувачі можуть видаляти товари зі списку, які вже були куплені або які більше не потрібні. Це можна зробити шляхом вибору елемента списку та використання відповідного функціоналу для його видалення.
- Копіювання елементів: Користувачі можуть скопіювати існуючі елементи списку для швидкого додавання їх до інших частин списку або для створення копій товарів, які їм потрібно придбати у кількості. Це дозволяє ефективно управляти списком покупок та зменшує час на його редагування.

Забезпечення цих функціональних можливостей дозволить створити додаток "Список покупок", який буде ефективним та зручним у використанні для користувачів.

Для реалізації функціональності додавання, видалення, редагування та копіювання елементів у списку покупок можна використати підхід з використанням ListView та адаптера в Android. ListView - це компонент інтерфейсу користувача, який відображає список елементів у вертикальному списку, дозволяючи користувачеві прокручувати його вгору чи вниз.

Адаптер використовується для зв'язку даних, які потрібно відобразити, з ListView. Він відповідає за створення кожного елемента списку та його відображення на екрані. При використанні адаптера можна легко взаємодіяти з даними, змінювати їх, додавати нові елементи чи видаляти існуючі.

Застосування ListView та адаптера в додатку "Список покупок" дозволить нам ефективно керувати списком елементів, забезпечуючи зручний та інтуїтивно зрозумілий інтерфейс для користувачів. Крім того, цей

підхід дозволяє легко реалізувати функцію швидкого пошуку, видаляти та редагувати елементи списку, а також створювати їх копії.

При розробці інтерфейсу користувача для додатку "Список покупок" враховується кілька ключових аспектів, що визначають його зручність та ефективність використання.

Крім простоти та зрозумілості, важливо забезпечити мінімалізм елементів управління, щоб уникнути перевантаження екрану зайвими деталями. Це дозволить користувачам зосередитися на головних функціях додатку і не відволікатися від їх головної мети - створення списку покупок.

Ще одним важливим аспектом є інтуїтивність інтерфейсу, що передбачає легке розуміння та використання функцій додатку. Користувачі повинні негайно розуміти, як додавати, видаляти та редагувати елементи списку.

Також дуже важливою є можливість швидкого доступу до функцій додатку. Користувачам слід мати можливість швидко додавати нові елементи до списку, видаляти та редагувати існуючі, а також швидко знаходити необхідні товари серед існуючих у списку.

Дотримання загальноприйнятих стандартів дизайну для платформи Android є ключовим аспектом розробки мобільних додатків, оскільки це допомагає забезпечити єдність інтерфейсу та покращити зручність використання для користувачів. Загальноприйняті стандарти дизайну для Android включають такі концепції, як Material Design, який рекомендується компанією Google для створення зручних та привабливих інтерфейсів.

Material Design встановлює зразки та рекомендації для використання різних елементів інтерфейсу, таких як кнопки, панелі інструментів, картки тощо, що допомагає створювати консистентний та естетично привабливий вигляд додатків. Він також надає рекомендації щодо використання анімації, кольорів та інших важливих аспектів дизайну, які сприяють покращенню

взаємодії користувачів з додатками.

Дотримання цих стандартів дозволяє забезпечити єдність в інтерфейсі між різними додатками на платформі Android та підвищити розпізнаваність бренда, оскільки користувачі звикають до певних елементів та поведінки. Крім того, це допомагає покращити враження користувачів від додатку, що може призвести до збільшення активності користувачів та зниження відсотка відмов.

Щоб забезпечити надійну та зручну роботу додатку, необхідно провести тестування на різних пристроях та в різних умовах використання. Це допоможе виявити та усунути потенційні проблеми та помилки.

Крім того, важливо забезпечити регулярне оновлення та підтримку додатку, щоб користувачі могли отримувати нові функції та виправлення помилок.

Загалом, проектування додатку "Список покупок" включає в себе розгляд вимог, розробку інтерфейсу користувача, реалізацію функціональності та тестування для забезпечення його ефективної та зручної роботи.

3.4 Принцип роботи програми.

Програма "Список покупок" створена для забезпечення користувачів зручним і ефективним інструментом управління списком покупок. Дозволяючи швидко додавати, видаляти та редагувати елементи, вона спрощує процес планування покупок. Розглянемо, як саме цей додаток працює і які можливості він надає користувачам.

При запуску додатку, у методі onCreate, спочатку створюється ListView, що відповідає за відображення списку покупок. Цей список ініціалізується дефолтними елементами за допомогою створення об'єкту ArrayList<String> items та додавання до нього деяких значень (Рис. 3.1).

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

    // Ініціалізація ListView та інших елементів
    listView = findViewById(R.id.list);
    input = findViewById(R.id.input);
    enter = findViewById(R.id.add);
    context = getApplicationContext();

    // Ініціалізація списку покупок з дефолтними елементами
    items = new ArrayList<>();
    items.add("Цукор");
    items.add("Яблуко");
    items.add("Сік");
    items.add("Хліб");

    // Налаштування адаптера для ListView
    adapter = new ListViewAdapter(this, items);
    listView.setAdapter(adapter);

    // Налаштування слухачів подій та іншого функціоналу
    // ...
}
```

Рис. 3.1 фрагмент коду MainActivity (ListView)

Адаптер ListViewAdapter відповідає за відображення елементів списку та налаштування їх макету. Кожен рядок списку містить назву товару та кнопки для видалення та копіювання. Під час створення кожного рядка, адаптер отримує дані про товари та розміщує їх у відповідних полях макету (Рис. 3.2).

Наприклад, при створенні нового рядка списку, адаптер заповнює поле з назвою товару даними про цей товар та встановлює номер рядка. Крім того, адаптер налаштовує кнопки для видалення та копіювання, дозволяючи користувачеві взаємодіяти з кожним елементом списку. Такий підхід забезпечує зручність управління списком покупок та підвищує його функціональність для користувача.

```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup pare
    if (convertView == null) {
        LayoutInflater mInflater = (LayoutInflater) context.getSystemService(Activity
        convertView = mInflater.inflate(R.layout.list_row, null);
        TextView name = convertView.findViewById(R.id.name);
        ImageView remove = convertView.findViewById(R.id.remove);
        ImageView copy = convertView.findViewById(R.id.copy);
        TextView number = convertView.findViewById(R.id.number);

        // Встановлення даних для назви товару та номеру рядка
        number.setText(position + 1 + ".");
        name.setText(list.get(position));

        // Налаштування слухачів подій кнопок
        remove.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                MainActivity.removeItem(position);
            }
        });
        copy.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                MainActivity.addItem(list.get(position));
            }
        });
    }
    return convertView;
}
```

Рис. 3.2 фрагмент коду ListViewAdapter

Під час натискання кнопки "Видалити", відповідний елемент списку покупок видаляється. Це досягається шляхом звернення до методу `removeItem()` у класі `MainActivity`, який приймає позицію вибраного елемента та видаляє його зі списку покупок. Після цього адаптер списку оновлюється, щоб відобразити зміни.

При натисканні кнопки "Копіювати" створюється копія вибраного елемента. Це досягається за допомогою методу `addItem()` у класі `MainActivity`, який додає новий елемент до списку покупок, використовуючи дані вибраного елемента. Після цього адаптер списку оновлюється, щоб відобразити зміни.

Ця функціональність дозволяє користувачу легко керувати своїм списком покупок, додавати нові елементи або видаляти зайві, забезпечуючи зручність та ефективність використання додатку.

При додаванні нових елементів користувач має можливість використовувати текстове поле, куди він вводить назву нового товару, який

бажає додати до списку покупок. Після введення назви товару користувач натискає кнопку "Додати". Текст, який був введений у текстовому полі, автоматично додається до списку покупок. Це відбувається за допомогою методу addItem() у класі MainActivity, який додає новий елемент до списку покупок на основі введеного користувачем тексту. Після додавання елемента адаптер списку оновлюється для відображення змін.

Під час закриття програми, дані списку покупок автоматично зберігаються у файлі "list.txt". Це досягається за допомогою методу `onDestroy()` у класі `MainActivity`, де список покупок перетворюється у рядок та записується у вказаний файл. При наступному запуску програми дані з файлу "list.txt" завантажуються назад у список покупок (Рис. 3.3). Ця функціональність дозволяє користувачеві зберігати свій список покупок між різними сеансами використання додатку, забезпечуючи зручність в роботі.

```
@Override
protected void onDestroy() {
    File path = getApplicationContext().getFilesDir();
    try {
        FileOutputStream writer = new FileOutputStream(new File(path, "list.txt"));
        writer.write(items.toString().getBytes());
        writer.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    super.onDestroy();
}

public void loadContent() {
    File path = getApplicationContext().getFilesDir();
    File readFrom = new File(path, "list.txt");
    byte[] content = new byte[(int) readFrom.length()];

    FileInputStream stream = null;
    try {
        stream = new FileInputStream(readFrom);
        stream.read(content);

        String s = new String(content);
        s = s.substring(1, s.length() - 1);
        String split[] = s.split(", ");

        if (split.length == 1 && split[0].isEmpty())
            items = new ArrayList<>();
        else items = new ArrayList<>(Arrays.asList(split));

        adapter = new ListViewAdapter(this, items);
        listView.setAdapter(adapter);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Рис. 3.3 фрагмент коду MainActivity

Висновки

Розробка та впровадження додатка "список покупок" для платформи Android відображає важливість використання сучасних технологій для оптимізації щоденних процесів та підвищення ефективності. Цей додаток не лише створює зручний інструмент для планування покупок, а й допомагає користувачу в організації свого часу та фінансів.

Однією з ключових переваг додатка є його актуальність в умовах сучасного ритму життя, де швидкість та ефективність стають невід'ємною складовою кожного дня. Замість традиційного паперового списку покупок, який легко загубити чи забути, користувачі можуть скористатися мобільним додатком для створення, збереження та оновлення своїх списків просто за декілька кліків. Це значно зручніше та надійніше, адже додаток доступний на смартфоні завжди з користувачем, навіть під час походу в магазин.

Зокрема, додаток надає можливість створювати не лише основні списки продуктів, але й додаткові категорії для кращої організації та систематизації покупок. Наприклад, користувач може створити список для продуктів харчування, окремий список для домашніх потреб, а також список для подарунків або покупок у різні свята та заходи. Такий рівень деталізації дозволяє зробити планування покупок більш систематичним та зручним.

Додаток "список покупок" для платформи Android також відзначається своєю гнучкістю та можливістю швидко реагувати на зміни. Однією з його ключових переваг є легкість управління списками покупок. Користувачі можуть легко додавати нові продукти до списку, видаляти або редагувати існуючі елементи, змінювати кількість, а також переносити продукти між різними категоріями. Це дозволяє швидко адаптувати список до зміни потреб або пріоритетів.

Також є можливість додавання певних продуктів до списку на основі раніше зроблених покупок, можливість встановлення нагадувань про заплановані покупки або автоматичне оновлення списків за допомогою онлайн бази даних продуктів.

Крім того, додаток надає можливість надсилати список покупок іншим користувачам. Наприклад, якщо у користувача є спільні покупки з кимось іншим, він може легко надіслати їм список, щоб вони також могли додавати або видаляти елементи. Це особливо зручно для сімей або груп друзів, які планують спільні покупки або вечірку.

Отже, додаток "Список покупок" для платформи Android є не лише зручним інструментом для планування покупок, а й важливим компонентом сучасного способу життя, який дозволяє оптимізувати час та ресурси та забезпечує більш ефективне управління фінансами та бюджетом.

Література

1. Програми для списку покупок в Android [Електронний ресурс]. – Режим доступу до ресурсу: https://daad.org.ua/3126-dodatki-dlya-spisku-pokupok-na-androyid.html
2. MVVM + Clean Architecture-Android [Електронний ресурс]. – Режим доступу до ресурсу: https://medium.com/@binsdreams/mvvm-clean-architecture-android-3a3899fed973
3. Everything You Need to Know About Mobile App Architecture [Електронний ресурс]. – Режим доступу до ресурсу: https://inoxoft.com/blog/everything-you-need-to-know-about-mobile-app-architecture/
4. Прикладний програмний інтерфейс [Електронний ресурс]. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Прикладний_програмний_інтерфейс
5. View model [Електронний ресурс]. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/View_model
6. Clean Architecture [Електронний ресурс]. – Режим доступу до ресурсу: https://habr.com/ru/companies/otus/articles/732178/
7. КУПИ БАТОН! [Електронний ресурс]. – Режим доступу до ресурсу: https://buymeapie.com/ru
8. Bring! Список продуктів [Електронний ресурс]. – Режим доступу до ресурсу: https://bring.ru.aptoide.com/app
9. Список покупок – SoftList [Електронний ресурс]. – Режим доступу до ресурсу: https://softlist.ru.aptoide.com/app
10. Listonic [Електронний ресурс]. – Режим доступу до ресурсу: https://listonic.com/ru/features/
11. Java [Електронний ресурс]. – Режим доступу до ресурсу:

https://www.java.com/en/download/help/whatis_java.html

12. Kotlin [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.android.com/kotlin/overview#:~:text=Kotlin%20is%20an%20open-source,and%20Scala%2C%20among%20many%20others.
13. Android SDK [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.android.com/tools/releases/platform-tools, https://developer.android.com/tools/releases/platforms
14. Android Studio [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.android.com/studio/intro
15. XML (Extensible Markup Language) [Електронний ресурс]. – Режим доступу до ресурсу: https://www.sciencedirect.com/topics/computer-science/extensible-markup-language#:~:text=Extensible%20Markup%20Language%20(XML)%20is%20a%20markup%20language%20designed%20as,well%20as%20many%20other%20uses.
16. Gradle Basics [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.gradle.org/current/userguide/gradle_basics.html
17. Architecture Components [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.android.com/jetpack/androidx/releases/archive/arch
18. Room Persistence Library [Електронний ресурс]. – Режим доступу до ресурсу: https://developer.android.com/training/data-storage/room
19. SQLite [Електронний ресурс]. – Режим доступу до ресурсу: https://www.sqlite.org/about.html

Додаток А

Перша частина коду програми на мові Java

ListViewAdapter

```
package com.example.grocery_list_app;

import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import java.util.ArrayList;

class ListViewAdapter extends ArrayAdapter<String> {
    ArrayList<String> list;
    Context context;

    public ListViewAdapter(Context context, ArrayList<String> items) {
        super(context, R.layout.list_row, items);
        this.context = context;
        list = items;
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        if (convertView == null) {
            LayoutInflater mInflater = (LayoutInflater)
context.getSystemService(Activity.LAYOUT_INFLATER_SERVICE);
            convertView = mInflater.inflate(R.layout.list_row, null);
            TextView name = convertView.findViewById(R.id.name);
            ImageView remove = convertView.findViewById(R.id.remove);
```

```
        ImageView copy = convertView.findViewById(R.id.copy);
        TextView number = convertView.findViewById(R.id.number);

        number.setText(position + 1 + ".");
        name.setText(list.get(position));

        remove.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                MainActivity.removeItem(position);
            }
        });
        copy.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                MainActivity.addItem(list.get(position));
            }
        });
    }
    return convertView;
  }

}
```

Додаток В

Друга частина коду програми на мові Java

MainActivity

```
package com.example.grocery_list_app;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.gesture.Gesture;
import android.media.Image;
import android.os.Bundle;
import android.util.Log;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.Toast;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.Arrays;

public class MainActivity extends AppCompatActivity {

    static ListView listView;
    EditText input;
    ImageView enter;
    static ListViewAdapter adapter;
    static ArrayList<String> items;
    static Context context;
```

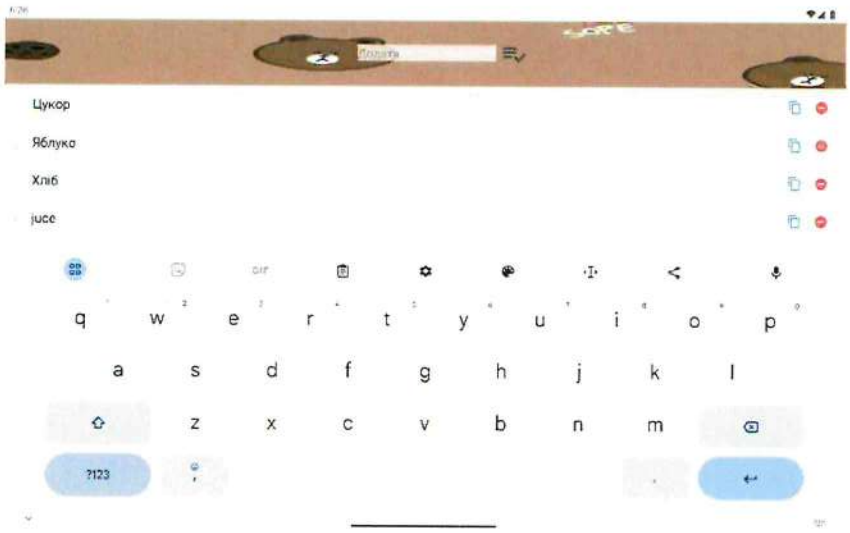
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    listView = findViewById(R.id.list);
    input = findViewById(R.id.input);
    enter = findViewById(R.id.add);
    context = getApplicationContext();

    items = new ArrayList<>();
    items.add("Цукор");
    items.add("Яблуко");
    items.add("Сік");
    items.add("Хліб");

    listView.setLongClickable(true);
    adapter = new ListViewAdapter(this, items);
    listView.setAdapter(adapter);

    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            String clickedItem = (String) listView.getItemAtPosition(position);
            Toast.makeText(MainActivity.this, clickedItem, Toast.LENGTH_SHORT).show();
        }
    });

    listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
        @Override
        public boolean onItemLongClick(AdapterView<?> adapterView, View view, int i, long l) {
            removeItem(i);
            return false;
        }
    });

    enter.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String text = input.getText().toString();
            if (text == null || text.length() == 0) {
                makeToast("Додати...");
            } else {
```

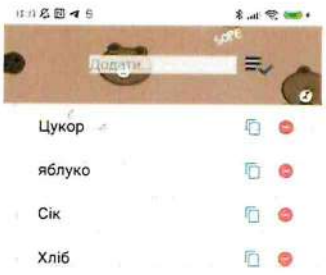
```
                addItem(text);
                input.setText("");
                makeToast("Додано " + text);
            }
        }
    });
    loadContent();
}

public void loadContent() {
    File path = getApplicationContext().getFilesDir();
    File readFrom = new File(path, "list.txt");
    byte[] content = new byte[(int) readFrom.length()];

    FileInputStream stream = null;
    try {
        stream = new FileInputStream(readFrom);
        stream.read(content);

        String s = new String(content);
        s = s.substring(1, s.length() - 1);
        String split[] = s.split(", ");

        if (split.length == 1 && split[0].isEmpty())
            items = new ArrayList<>();
        else items = new ArrayList<>(Arrays.asList(split));

        adapter = new ListViewAdapter(this, items);
        listView.setAdapter(adapter);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@Override
protected void onDestroy() {
    File path = getApplicationContext().getFilesDir();
    try {
        FileOutputStream writer = new FileOutputStream(new File(path, "list.txt"));
        writer.write(items.toString().getBytes());
        writer.close();
    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        super.onDestroy();
    }

    public static void removeItem(int i) {
        makeToast("Видалено: " + items.get(i));
        items.remove(i);
        listView.setAdapter(adapter);
    }

    public static void addItem(String item) {
        items.add(item);
        listView.setAdapter(adapter);
    }

    static Toast t;

    private static void makeToast(String s) {
        if (t != null) t.cancel();
        t = Toast.makeText(context, s, Toast.LENGTH_SHORT);
        t.show();
    }
}
```

Додаток С

Скріншоти створеного додатку.





Додаток D

Рисунки використані у дипломній роботі.

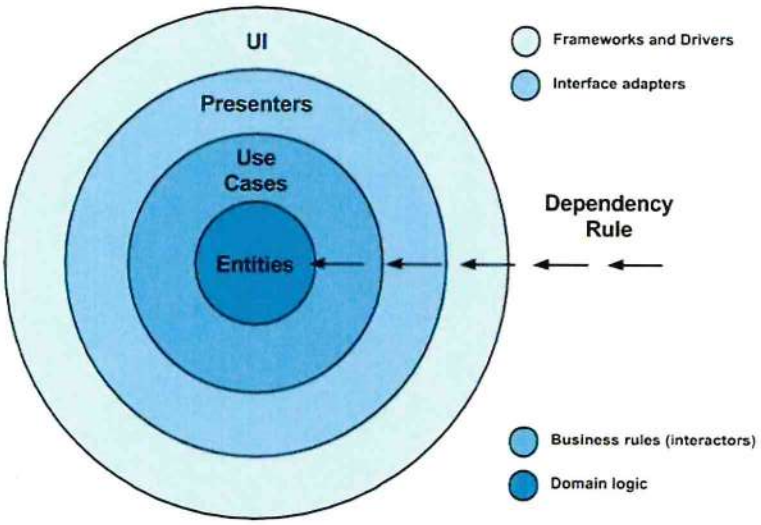


Рис. 1.2. Схема Clean Architecture

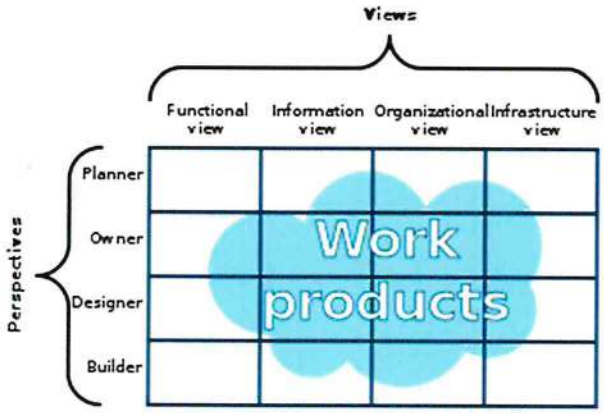


Рис. 1.3. Матриця поглядів і перспектив TEAF

```
interface PostDataSource {
    interface LoadPostsCallback {
        fun onPostsLoaded(posts: List<Post>)
        fun onError(t: Throwable)
    }
    interface SaveTaskCallback {
        fun onSaveSuccess()
        fun onError(t: Throwable)
    }
    fun getPosts(userId: Int, callback: LoadPostsCallback)
    fun savePost(post: Post)
}
```

Рис. 1.7. Код варіанти використання ViewModel

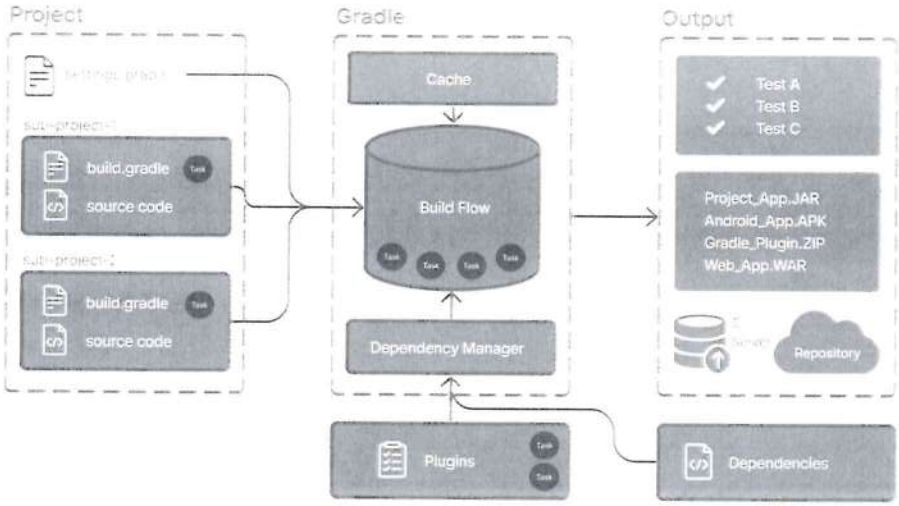


Рис. 2.5 Gradle схема