ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка мобільного застосунку для підрахунку енергетичної та поживної цінності продуктів на Kotlin з використанням Clean Architecture та Coroutines»

(тема кваліфікаційної роботи українською мовою)

«Development of a mobile application for calculating the energy and nutritional value of products on Kotlin using Clean Architecture and Coroutines»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання

спеціальності 122 Комп'ютерні науки

(код, назва спеціальності)

Освітня програма Комп'ютерні науки

(назва)

Беркатюк Валентин Валентинович

(прізвище, ім'я, по-батькові здобувача)

Керівник асистент Молчанова А.Ю. (підпис)

(науковий ступінь, вчене звання, прізвище, ініціали)

Рецензент к.т.н., Домаскін О.М.

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .
№ 1 від 09 червня 2024 р.

Завідувачка кафедри

(підпис) КАЗАКОВА Надія .
(прізвище,ім'я)

Захищено на засіданні ЕК № 13 .
протокол № 10 від 20 червня 2024 р.

Оцінка добре / С / 75 .
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

(підпис) КОПИЧЕНКО Іван .
(прізвище,ім'я)

Одеса 2024

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних.

ОС – операційна система.

ПЗ – програмне забезпечення.

ADT – Android Development Tools – Інструменти розробки в ОС Android.

API – Application Programming Interface – програмний інтерфейс Програми.

IDE – Integrated Development Environment – інтегроване середовище розробки.

SDK – Software Development Kit – набір засобів розробки.

КБЖВ – калорії, білки, жири, вуглеводи.

БЖВ – білки, жири, вуглеводи

СКБД – Система керування базами даних

БД – База Даних

LLVM – Low Level Virtual Machine

LINQ – Language-Integrated Query

XML – EXtensible Markup Language

CLR – Common Language Runtime

JIT – динамічна компіляція

ARC – Автоматичний підрахунок посилань

ВСТУП

Кулінарія завжди мала сильний вплив на соціальний устрій суспільства. У наш час вона займає значущу позицію в житті, планах та графіку людини.

Розробка сучасних інформаційних та програмних систем, які забезпечують якісну роботу в сфері кулінарії – є досить сучасною та актуальною задачею, завдяки великій популярності цього напрямку. Представники обох статей проводять чималу кількість часу на кухні, не кажучи вже про час, який вони витрачають на вибір страви, яке готуватимуть. Найчастіше, люди готують з того, що є у них в холодильнику. В цьому і полягає вся проблема. Людина не може тримати в голові сотні рецептів з вмістом КБЖВ прабабусі, мами, порад з інтернету і готувати, покладаючись на них.

Для поліпшення загальних процесів автоматизації та створення єдиного інформаційного простору для зберігання та використання рецептів з вмістом КБЖВ та вказівок використовуються сучасні інформаційні технології, які дозволяють розробити сучасний програмний засіб, здатний виконати поставлене завдання за менший проміжок часу і забезпечити більш високу точність, ніж людина. А завдяки отриманим знанням з програмування, алгоритмізації та роботі з базами даних, розроблений програмний засіб може використовуватися з навчальною метою.

Розроблений застосунок може використовуватися в якості системи для взаємодії користувачів із базою кулінарних рецептів та вказівок, а також може виступати в якості помічника у справах з їжею. Головною особистістю такого застосунку є його гнучкість до запитів користувача та можливість адаптувати її під різні вимоги та погляди різних користувачів.

Результатом розробки є інформаційний простір для зберігання та використання рецептів та вказівок кулінарного характеру, яка дозволяє утворити взаємодію користувачів-клієнтів з інформаційною базою рецептів з вмістом КБЖВ.

Метою кваліфікаційної роботи є розробка мобільного застосунку «Calories Tracker» отримання користувачем актуальної інформації щодо КБЖВ. Для досягнення поставленої мети були сформульовані наступні завдання:

- провести аналіз предметної області щодо додатків кулінарії з КБЖВ та провести порівняльний аналіз існуючих програм-аналогів;
- обґрунтувати вибір програмних засобів розробки та технологій;
- провести проектування системи з використанням мови UML;
- виконати реалізацію застосунку, підготувати інструкцію користувача та виконати тестування застосунку.

Структура кваліфікаційної роботи бакалавра складається з вступу, 5 розділів, висновків, переліку посилань на 17 найменувань, 1 додатку. Повний обсяг проекту становить 51 сторінку, містить 19 рисунків і 4 таблиці.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ СИСТЕМ

1.1 Аналіз предметної області

Додатки для підрахунку калорій, білків, жирів та вуглеводів (КБЖВ) відіграють важливу роль у сучасному суспільстві. Вони допомагають людям контролювати своє харчування, слідкувати за здоров’ям та досягати цілей з контролю ваги.

Аналіз предметної області

Додатки для підрахунку КБЖВ зазвичай надають базу даних з інформацією про харчовий склад великої кількості продуктів та страв. Користувачі можуть вводити свої щоденні прийоми їжі, а застосунок автоматично розраховує кількість спожитих калорій, білків, жирів та вуглеводів. Деякі додатки також пропонують функції для відстеження фізичної активності та водного балансу.

Важливість та користь

Контроль ваги: Підрахунок КБЖВ допомагає людям контролювати свою вагу. Знаючи, скільки калорій вони споживають і витрачають, люди можуть більш ефективно планувати свою дієту та фізичну активність.

Здорове харчування: Додатки для підрахунку КБЖВ допомагають людям зрозуміти, які продукти є більш здоровими. Вони можуть дізнатися, які продукти містять більше білка або менше насичених жирів.

Досягнення спортивних цілей: Для людей, які займаються спортом або прагнуть набрати м’язову масу, підрахунок КБЖВ може допомогти визначити, скільки білка їм потрібно споживати.

Контроль за захворюваннями: Для людей з певними захворюваннями, такими як діабет, контроль за споживанням вуглеводів може бути важливою частиною контролю захворювання.

Однак слід зазначити, що підрахунок КБЖВ не рекомендується для всіх, оскільки це може призвести до порушень харчової поведінки. Завжди краще

звернутися до професіонала в галузі охорони здоров'я перед початком нової дієти або програми харчування.

1.2 Аналіз існуючих програмних систем

На даний час створюються різноманітні програми, що надають можливість відслідковувати КБЖВ, ми порівняємо три програми.

Програмний продукт «Добери рецепт»

Мобільний застосунок, в якому можна знайти прості рецепти і страви різних країн світу з фотографіями. Всі рецепти доступні без Інтернету. Основна функція програми - підбір страв за наявними вдома продуктам. При цьому користувач в будь-який момент може поміняти розміри виробу. Головне вікно програми представлене на рисунку 1.1

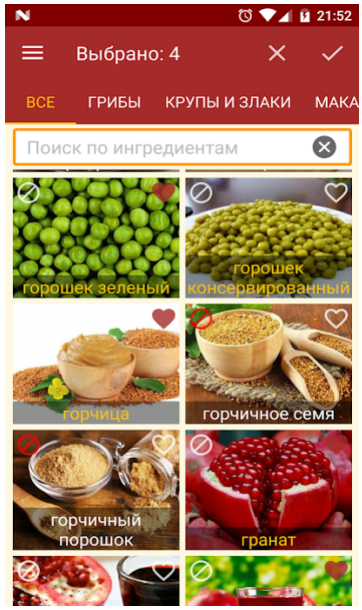


Рисунок 1.1 – Головне вікно програми Добери рецепт

Запускаючи застосунок користувач потрапляє до вікна з вибором інгредієнтів, де може вибрати продукти, додати їх до списку, переглянути їх та виконати пошук рецептів. Також у додатку можна обирати інгредієнти, які користувач не хоче бачити у списку інгредієнтів. Відкриваючи меню, можна

додати свої рецепти, переглянути усі рецепти, виконувати пошук по ним, додавати в обране. Існують також вкладки із таймером, таблицею вагів рецептів або інгредієнтів, посилання на групу у соціальних мережах, категорії рецептів та список продуктів, які користувач може хотіти купувати, якщо у рецепті який він знайшов немає інгредієнтів.

До переваг програмного продукту належать:

- Знайдені рецепти відсортовані на найбільшій кількості співпадаючих продуктів, а також за категоріями страв.
- Знайдені інгредієнти виділяються кольором всередині рецепта.
- Кожен рецепт містить фотографію, список інгредієнтів, калорійність, співвідношення БЖВ і докладний опис.
- У додатку можна позначити як вибраний не тільки сподобалися рецепти, а й інгредієнти.
- Також можна додати в стоп-лист ті інгредієнти, які не будуть використовуватися.
- Сподобалися рецептами можна поділитися в соціальних мережах, месенджерах, по електронній пошті і т.д.
- Всі доступні рецепти можна подивитися за категоріями або загальним списком, в яких присутній зручний пошук.
- Також можна використовувати додаткові функції: список покупок, таблиця мір і ваг, таблиця часу приготування.

До недоліків програмного продукту належіть:

- Не точний результат за обраними критеріями користувачів.
- Обмежена кількість інгредієнтів.

Програмний продукт «Що готуємо?»

Унікальний застосунок «Що готуємо?» легко підбере рецепти під наявні вдома продукти.

Основна функція програми - підбір рецептів страв, які можна приготувати прямо зараз. Вікно програми представлене на рисунках 1.2.

Рисунок 1.2 – Вікно програми Що готуємо?

Після запуску користувач знаходиться у головному вікні системи, де відображаються усі рецепти. Відкриваючи меню він може потрапити до вікна з вибором інгредієнтів, де може вибрати продукти, переглянути їх та виконати пошук рецептів. Також у додатку можна обирати інгредієнти, які користувач планує купувати та надрукувати список інгредієнтів. Переходячи на вкладку пошуку користувач може знайти рецепти за назвою або за категорією. Виконуючи пошук інгредієнтів користувачеві відображаються рецепти, в яких є і інші інгредієнти, застосунок пропонує одразу сформувати із них список.

До переваг програмного продукту належать:

- застосунок має рецептурну базу для більш ніж 850 страв;
- застосунок буде регулярно доповнюватися новими рецептами.

До недоліків програмного продукту належать:

- застосунок не відповідає описаному функціоналу;
- немає сумісності з більш старими пристроями;
- за обраними ингрдієнтами результат не відповідає запиту;
- немає зображення інгредієнтів;

- маленька кількість рецептів.

Програмний продукт: «Мені до смаку»

Інтернет ресурс, який містить базу кулінарних рецептів, пошук за обраними ингридієнтам, блог новин і цікавих рецептів. Головне вікно програми представлене на рисунку 1.3.

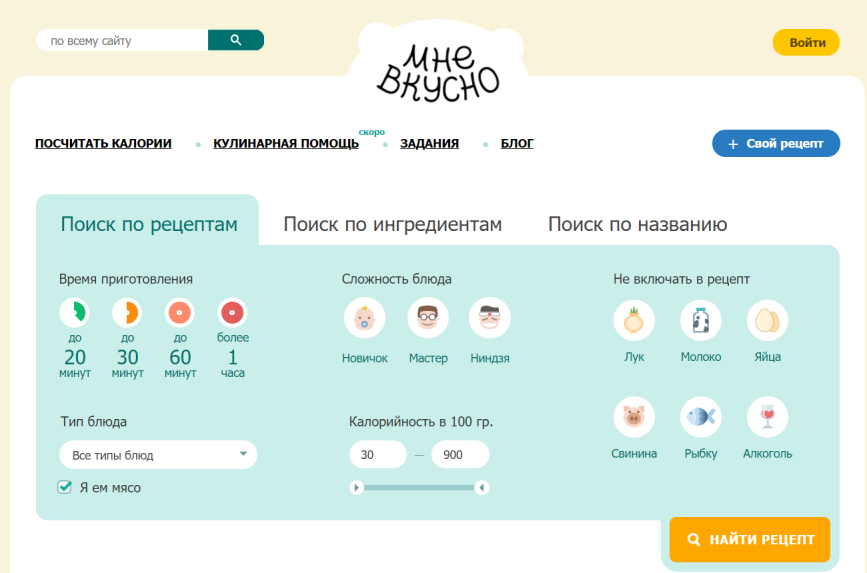


Рисунок 1.3 – Головне вікно програми Мені до смаку

Застосунок дозволяє користувачам шукати рецепти за важкістю приготування, за калоріями або за часом приготування, що дає змогу знаходити рецепти за більш гнучкішою системою фільтрації.

До переваг програмного продукту належать:

- динамічно оновлювана база даних з кулінарними рецептами;
- вбудована функція створення і друку списку відсутніх продуктів;
- блог з новинами у сфері кулінарії;
- калькулятор калорій;
- застосунок буде регулярно доповнюватися новими рецептами.

До недоліків програмного продукту належать:

- графічний дизайн,який вимагає доопрацювань;
- некоректно відображається регіональна реклама;

мобільна версія сайту не пристосована для комфортного використання користувачами на різних пристроях.

Після проведення аналізу існуючих рішень, можна зробити висновок, що проаналізовані рішення мають ряд особливостей, які потрібно реалізувати у проєкті. А саме:

- підтримка системи на старіших версіях операційної системи та старих пристроях
- правильне відображення системи на різних пристроях
- кількість інгредієнтів залежить від інгредієнтів, які є у рецептах.

Таким чином, під час виконання першого розділу бакалаврської роботи була обрана предметна область, в рамках якої розглянуті програмні аналоги і їх особливості. Переваги та недоліки розглянутих мобільних застосунків-аналогів були взяті до уваги при розробці програмного продукту.

2 ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ

2.1 Функціональні та нефункціональні вимоги

Перед початком розробки застосунку були визначені функціональні та нефункціональні вимоги до нього.

До функціональних вимог належать:

- розпізнавання картин за допомогою камери смартфону;
- відображення основної інформації про розпізнаний об’єкт;
- можливість зберігати об’єкт в «улюбленому» для подальшого перегляду.

До нефункціональних вимог належать:

- смартфон з ОС Android версією 7.0+;
- наявність камери на смартфоні;

підключення до Інтернету.

2.2 Обґрунтування вибору операційної системи

На рис. 2.1 представлена діаграма, що показує частки, які за даними на січень 2024 займають популярні мобільні ОС на ринку.

Згідно до статистики від компанії Statcounter, особистому досвіді та діаграмі на рис. 2.1 були виділені дві найбільш популярні ОС, які необхідно розглянути: iOS та Android.

iOS використовується тільки на пристроях Apple, таких, як iPhone. iOS не вимагає величезних обсягів оперативної пам'яті, оскільки вона може підтримувати загрузку і готовність більше десятка застосунків всього лише з 2 ГБ. І хоча роздільна здатність дисплея деяких моделей може здатися низькою, щільність пікселів залишається більш ніж достатньою. У той же час, більш низька роздільна здатність означає менше роботи для графічного процесора і, отже, меншу витрату батареї. Якість камери смартфона повністю залежить від виробника. У випадку з Apple, у iPhone завжди були відмінні камери.

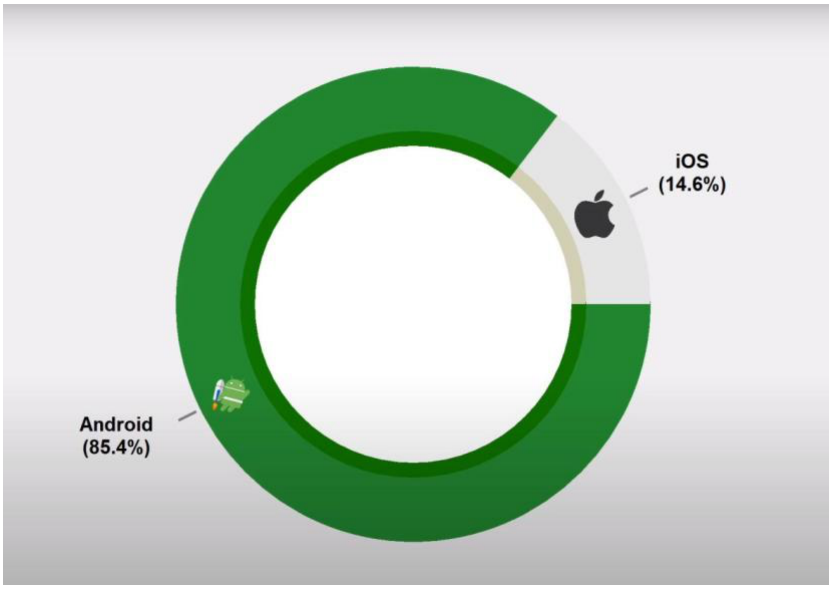


Рисунок 2.1 – Діаграма співвідношення мобільних ОС на ринку смартфонів від Pie Chart Pirate

В App Store багато платних застосунків, iOS не передбачає завантаження та встановлення застосунків з будь-яких інших джерел. З іншого боку, строгий контроль Apple над застосунками, доступними для їх платформи, гарантує, що вони функціонують належним чином, і що вони повністю безпечні і вільні від будь-якого шкідливого коду.

Пристрої Apple дорогі. Це стосується не тільки iPhone, але і всього іншого: MacBook, iMac, Apple Watch і практично всіх їхніх аксесуарів. Єдиний спосіб отримати доступний iPhone - це отримати модель більш старого покоління, потенційно навіть потриманий.

Таким чином, не дивлячись на високу продуктивність, високоякісні застосунки та довгострокові оновлення ОС, iOS не дуже приваблива для розробників. Для того, щоб опублікувати свій застосунок в App Store, необхідно мати аккаунт розробника Apple ($99 в рік) та фізичний комп'ютер з macOS для створення проектів в Xcode. До того, кількість користувачів iOS набагато менше за кількість користувачів Android, тож постає питання окупності.

В даний час Android є найпопулярнішою платформою для смартфонів у світі і використовується багатьма виробниками телефонів. Особливість

Android – відкритий вихідний код. Крім того, телефони Android дозволяють користувачам отримувати доступ до сторонніх магазинів застосунків. Телефони Android на відміну від iOS також дозволяють користувачеві завантажувати застосунки вручну, завантажуючи файли apk.

На 2020 рік кількість застосунків в магазині Google Play перевищує 2,8 млн. Згідно з даними, зібраними SafeBettingSites, в третьому кварталі 2020 року кількість завантажень з Google Play досягло 28,3 млрд, що в три рази перевищило показник App Store (8,2 млрд) [6].

Для того, щоб опублікувати свій застосунок в Google Play, необхідно одноразово сплатити збір $25, що значно дешевше у зрівнянні з Apple Store. До того, доступна безліч інструментів для розробки застосунків для Android.

З огляду на перераховані особливості операційних систем iOS та Android, перевага надається системі Android. Тож розробка застосунку буде виконуватися для пристроїв з ОС Android.

2.3 Вибір середовища розробки

Популярними середовищами розробки під ОС Android є Eclipse, Android Studio, Unity.

Eclipse – вільне інтегроване середовище розробки модульних кросплатформних застосунків. Розвивається і підтримується Eclipse Foundation. Для розробки Android- застосунків в Eclipse необхідно підключити плагін Android Development Tools. Однак плагін Eclipse ADT більше не підтримується, згідно з оголошенням у червні 2015 року. Цей плагін має багато відомих і потенційних помилок безпеки, які не будуть виправлені. Саме тому Google рекомендує переключитися на використання Android Studio.

Android Studio – офіційне інтегроване середовище розробки (IDE) для Android. Дане середовище орієнтоване на розробку застосунків саме під ОС Android, що виділяє його серед інших IDE.

Особливості Android Studio:

- середовище розробки підтримує роботу з декількома мовами програмування, до яких відносяться найпопулярніші - C / C ++, Java.
- редактор коду, з яким зручно працювати;
- дозволяє розробляти програми не тільки для смартфонів / планшетів, а й для портативних ПК, приставок для телевізорів Android TV, пристроїв Android Wear, мобільних пристроїв з незвичайним співвідношенням сторін екрану;
- тестування коректності роботи нових ігор, утиліт, їх продуктивності на тій чи іншій системі відбувається безпосередньо в емуляторі;
- рефакторинг вже готового коду;
- досить велика бібліотека з готовими шаблонами і компонентами для розробки ПЗ;
- розробка програм для Android N - останньої версії операційної систе-ми;
- попередня перевірка вже створеного застосунку на предмет помилок в ньому;
- великий набір засобів інструментів для тестування кожного елемента програми, ігри;
- для недосвідчених / початківців розробників спеціально створено посібник з використання Android Studio, розміщений на офіційному сайті.

Однак Android Studio не орієнтоване на розробку застосунків з використанням технології доповненої реальності. Багато платформ для роботи з AR (такі як Hololens, Oculus, Vuforia тощо) призначені для Unity.

Unity – крос-платформне середовище розробки застосунків, розроблена американською компанією Unity Technologies. Unity дозволяє створювати застосунки , що працюють на більш ніж 25 різних платформах, що включають

Windows, Mac, iOS, Android, PlayStation, Xbox, Nintendo Switch, а також провідні AR- і VR-платформи.

Unity не є повністю безкоштовним продуктом. Існує кілька видів підписок:

- Personal. Безкоштовна версія, яка містить всі основні функції движка. Має таке обмеження: дохід в рік або обсяг залучених коштів не повинен перевищувати 100 000 $.
- Plus. За 35 доларів на місяць надаються різні звіти і аналітика, а також можливість змінювати заставку, 20%-а знижка на покупки в Asset Store і різні дрібні переваги. Має таке обмеження: дохід за рік або обсяг залучених коштів не повинен перевищувати 200 000 $.
- Pro. За 125 доларів на місяць включає в себе всі переваги версії Plus і додатково професійний сервіс і преміум-підтримку. Без обмежень по обороту або обсягу коштів.
- Окремі версії для бізнесу (використовуються великими компаніями).

Для багатьох проектів буде достатнім використання безкоштовної підписки Personal.

Unity підтримує дві системи збірки Android: Gradle та внутрішню.

Кроки, пов’язані зі збіркою для Android:

- підготовка та побудова активів Unity Assets;
- компіляція скріптів C#;
- обробка плагінів;
- розділення ресурсів на частини, які переходять до файлів .apk та OBB, якщо вибрано Split Application Binary;
- створення ресурсів Android за допомогою утиліти AAPT (лише для внутрішньої збірки);
- створення маніфесту Android
- об’єднання маніфестів бібліотеки та маніфесту Android (лише для внутрішньої збірки).

- компіляція коду Java у формат Dalvik Executable (DEX) (лише для внутрішньої збірки.
- збірка бібліотеки IL2CPP, якщо вибрано IL2CPP Scripting Backend.

За підсумком аналізу існуючих середовищ розробки, було обрано середовище Android Studio, тому що воно є крос-платформним, найбільше підходить під потреби застосунку, що розроблюється, та має багато навчальної інформації і форум для підтримки.

3 ТЕХНОЛОГІЯ РОЗРОБКИ ДОДАТКУ

3.1 Технології мов програмування

Для вибору найбільш відповідної до вимог проекту мови програмування проводився аналіз трьох аналогів: Kotlin, C#, Swift.

Мова програмування Kotlin

Kotlin (Котлін) - статично типізований, об'єктно-орієнтована мова програмування, що працює поверх Java Virtual Machine і розробляється компанією JetBrains. Також компілюється в JavaScript і в виконуваний код ряду платформ через інфраструктуру LLVM. Мова названий на честь острова Котлін в Фінській затоці, на якому розташоване місто Кронштадт.

Автори ставили за мету створити мову більш лаконічний і тіпобезопасний, ніж Java, і більш простий, ніж Scala. Наслідком спрощення в порівнянні зі Scala стали також більш швидка компіляція і найкраща підтримка мови в IDE. Мова повністю сумісний з Java, що дозволяє java-розробникам поступово перейти до його використання; зокрема, в Android мову вбудовується за допомогою Gradle, що дозволяє для існуючого android-додатки впроваджувати нові функції на Kotlin без переписування програми цілком.

Переваги мови програмування Kotlin

У Kotlin інтуїтивно зрозумілий синтаксис, що значно підвищує продуктивність і швидкість роботи програмістів. У ньому прекрасно дотримується послідовність. Для написання програми тепер потрібно менше рядків і менше часу. Багато людей себе звуть Kotlin за стисліть, якість і читаність, які стали його конкурентною перевагою перед Java. Перед творцями Kotlin стояло завдання використовувати всі знання і досвід по максимуму. Так, програмістам доступні всі необхідні бібліотеки. Можна писати модулі на Kotlin, і вони будуть відмінно працювати з Java, адже обидві мови можна використовувати в одному проекті. Для компаній, з великою базою коду на Java, це стало великим плюсом. Код Kotlin відкритий для всіх програмістів, які хочуть з ним працювати. Його впровадження в проект також абсолютно безкоштовно.

Також вихідний код дає можливість легко знайти і позначити проблему в разі її виникнення, щоб повідомити про це розробникам мови. Правила створення коду допомагають розробникам знайти незначні помилки, які складно виявити до запуску програми. Найпоширенішим підводним каменем багатьох мов програмування є спроба зробити доступ до null-значенням. Це призводить до помилки. Kotlin покликаний виключити помилки подібного роду з вашого коду. Для повної зручності роботи з Kotlin творці зробили його сумісним з бібліотекою Java, так як багато програмісти перейшли з нього на Kotlin. Він може використовувати всі відомі Java фреймворки і бібліотеки, а також окремі модулі в поточних проектах.

Недоліки мови програмування Kotlin.

Початківцям розробникам доведеться нелегко, адже вирішувати проблеми доведеться поодинці. У Kotlin дуже невелика спільнота і мало ресурсів для вивчення. Залишається покладатися тільки на підтримку. Часто виникають проблеми зі швидкістю компіляції коду. Це не постійна недоробка, іноді вона відбувається дуже швидко, іноді помітно повільніше. Однак розробникам не дуже подобається такі моменти.

Мова програмування C#

C# - об'єктно-орієнтована мова програмування. Компілятор з C# входить в стандартну установку самої .NET, тому програми на ньому можна створювати і компілювати навіть без інструментальних засобів, на кшталт Visual Studio. C# відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів, в тому числі операторів явного і неявного приведення типу, делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML. Перейнявши багато від своїх попередників - мов C++, Java, Delphi, Модула і Smalltalk - C #, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так, C# не підтримує множинне

успадкування класів. C# розроблявся як мова програмування прикладного рівня для CLR і, як такий, залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#, яка відображає BCL. Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльований в відповідні конструкції CLR.

Переваги мови програмування C#

Дана мова використовує об'єктно-орієнтований підхід до програмування. Це означає, що необхрдімо буде описувати абстрактні конструкції на основі предметної області, а потім реалізовувати між ними взаємодія. Даний підхід користується великою популярністю, тому що дозволяє не тримати в голові всю інформацію, а працювати за принципом чорного ящика: подавати вхідні дані і результат.

Також в мові присутня велика кількість насиченного синтаксису, який робить важке життя програміста крапельку легше. Замість того, щоб писати мільон рядків коду, можна просто використовувати готову конструкцію, а компілятор зробить за тебе всю брудну роботу. Але деякі такі конструкції не є найоптимальнішими з точки зору продуктивності. Але все це перекривається за рахунок легкості читання коду і високою швидкістю розробки.

C# працює на базі .NET Framework. Написаний код на мові C# транслюється в проміжний мова, який в свою чергу вже перетворюється в машинний код на твоєму комп'ютері прямо під час виконання програми . Суть в тому, що з'являється можливість писати на різних мовах один і той же проект з іншими людьми і жодному не доведеться перевчатися. Но це ще не все. Так як остаточна компіляція з проміжного коду виконується в живу на конкретній машині, то можливе збільшення продуктивності за рахунок використання специфічних команд процесора.

Недоліки мови програмування C#

C# дуже легко дезасемблюється. Це означає, що з великою часткою ймовірності ваш код буде отримано і вивчений конкурентами. Звичайно ж, є

спеціальні інструменти, які можуть ускладнити цей процес, але на 100% захиститися від цього практично неможливо.

.NET використовує концепцію JIT-компіляції. Це означає, що програма буде скомпільована в машинні коди в міру необхідності прямо під час роботи програми. З одного боку, це звичайно добре, але при першому запуску можливі дуже серйозні гальма.

C# не є повсюдно поширеною мовою. Більшість програмістів зосереджені в комерційній Enterprise сфері, що накладає дуже серйозні обмеження на пошук роботи в невеликих містах. До того ж, як би там не було, C# в першу чергу асоціюється з Windows.

Мова програмування Swift

Swift - відкрита мультипарадигмальна компільована мова програмування загального призначення. Програми на Swift компілюються за допомогою LLVM, що входить в інтегровану середу розробки Xcode 6 і вище. Swift може використовувати рантайм Objective-C, що робить можливим використання обох мов в рамках однієї програми. Swift запозичив досить багато з Objective-C, проте він визначається не покажчиками, а типами змінних, які обробляє компілятор. За аналогічним принципом працюють багато скриптові мови. У той же час, він надає розробникам багато функцій, які раніше були доступні в C++ і Java, такі як визначаються найменування, узагальнення і перевантаження операторів.

Частина функцій мови виконується швидше в порівнянні з іншими мовами програмування. Наприклад, сортування комплексних об'єктів виконується в 3,9 разів швидше, ніж в Python, і майже в 1,5 рази швидше, ніж в Objective-C.

Код, написаний на Swift, може працювати разом з кодом, написаним на мовах програмування C і Objective-C в рамках одного і того ж проекту.

Коли команда Apple розробляла заміну Objective-C, у них було дві основні вимоги:

- мова повинена бути простою у вивченні;
- повинна сприяти прискоренню циклу розробки додатків.

У підсумку, Swift має всі атрибути сучасної мови програмування і виразно перевершує Objective-C по всіх фронтах. Основні особливості:

- немає невизначених або неініціалізованих змінних;
- немає помилок з размерностями масивів;
- немає помилок переповнення;
- явна обробка значень nil (null);
- автоматичне управління пам'яттю.

Таким чином витрачається більше часу на реалізацію ідей і менше - на занепокоєння з приводу можливих помилок, збоїв і конфліктів вашого коду. Крім того, мова поборов синтаксичну багатослівність в Objective-C, що спростило запис і читання. Результат - в рази менше часу на написання аналогічного коду в Swift.

Чистий і виразну мову зі спрощеним синтаксисом і граматикою, Swift легше читати і писати. Це дуже лаконічно, що означає, що для виконання тієї ж задачі потрібно менше коду, ніж в Objective-C. Автоматичний підрахунок посилань (ARC) виконує всю роботу з відстеження та управління використанням пам'яті додатком, тому розробникам не потрібно витрачати час і сили, роблячи це вручну. Відповідно, створення додатків для iOS за допомогою Swift зазвичай займає менше часу. На застосунок до скорочення часу розробки виходить продукт, який орієнтований на майбутнє і може бути доповнений новими функціями в міру необхідності. Таким чином, проекти Swift зазвичай легше масштабувати. Той факт, що Apple з більшою ймовірністю буде підтримувати Swift, ніж Objective-C, також повинен серйозно розглянути питання про довгострокові інвестиції.

Переваги мови програмування Swift.

Swift використовує автоматичний підрахунок пам'яті - технологію, призначену для додавання функції збирача сміття, яка раніше не була представлена в iOS. Такі мови, як Java, C# і Go, використовують збирачі сміття для видалення екземплярів класів, які більше не використовуються. Вони корисні для зменшення обсягу використовуваної пам'яті, але можуть збільшити

навантаження на процесор до 20 відсотків. До появи ARC розробникам iOS доводилося керувати пам'яттю вручну і постійно управляти рахунками зберігання кожного класу. ARC Swift визначає, які екземпляри більше не використовуються, і позбавляється від них від вашого імені. Це дозволяє збільшити продуктивність додатку, не відстаючи від пам'яті або процесора.

Недоліки мови програмування Swift.

Swift може бути найшвидшим і потужним мовою в світі, але все ще занадто молодий. У цього є багато проблем, які повинні бути вирішені і "зростаючі болю". Проте, з цього випливає, що Swift часто вважається нестабільним через серйозних змін, що вносяться до кожен новий випуск. Однією з ключових проблем, про яку говорили багато розробників, є відсутність зворотної сумісності зі старими мовними версіями. Отже, розробники змушені повністю переписувати свої проекти, якщо вони хочуть перейти на останню версію Swift. Багато в чому через часті оновлення і відсутності зворотної сумісності, як уже згадувалося вище, часто важко знайти правильні інструменти для вирішення певних завдань.

3.2 Порівняння систем керування базами даних

Система керування базами даних (СКБД) - комплекс програмного забезпечення, що надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних. Порівнюються такі СКБД, як PostgreSQL, SQLite і MySQL.

Система керування базами даних MySQL

Мова структурованих запитів - Structured Query Language, SQL - найпоширеніша мова, призначений для запису, вилучення, оновлення та видалення інформації в системах керування базами даних. Реляційна означає, що база даних відповідає реляційної моделі, і відноситься до схеми і принципам зберігання даних.

MySQL - повнофункціональна вільно поширювана система керування базами даних. MySQL почали розробляти в 1990-х роках, оскільки потреба в розумному управлінні комп'ютерною інформацією постійно росла. Розробники Ядро MySQL намагався вирішити цю Проблему з допомогою Маленького і простий Бази даних Msql. Коли з'ясувалося, що Msql не справляється з усіма завданнями, які творцям хотілося на неї покласти, вони створили більш потужну базу даних, яка перетворилася в MySQL.

Переваги системи управління базами даних MySQL:

– простота у встановленні та використанні;

– підтримується необмежена кількість користувачів, що одночасно працюють із БД;

– кількість рядків у таблицях може досягати 50 млн;

– висока швидкість виконання команд;

– наявність простої і ефективної системи безпеки.

Недоліки системи управління базами даних MySQL:

– не реалізована підтримка трансакцій. Натомість пропонується використовувати LOCK/UNLOCK TABLE;

– відсутня підтримка зовнішніх (foreign) ключів;

– відсутня підтримка тригерів і збережених процедур;

– відсутня підтримка представлень (VIEW).

– простота у встановленні та використанні;

– підтримується необмежена кількість користувачів, що одночасно працюють із БД;

– кількість рядків у таблицях може досягати 50 млн.;

– висока швидкість виконання команд;

– наявність простої і ефективної системи безпеки.

Система керування базами даних SqLite

SqLite є бібліотекою, скомпонованою з додатком. Як і всі вбудовані бази даних, SQLite надає значний набір інструментів для роботи з різними наборами даних при зменшених витратах порівняно з базами даних, побудованих за клієнт-серверної моделі. Завдяки тому, що вся база даних зберігається в єдиному файлі, SQLite є дуже швидкою і ефективною СУРБД, яка використовується як у вбудованих системах, так і на виділеній машинах.

Переваги системи управління базами даних sqLite:

- Зручність. Вся база складається з єдиного файлу даних, що забезпечує найвищий рівень переносимості.

- Стандарти. Незважаючи на свою полегшену структуру, в основі SQLite лежить SQL і його стандарти.

- Сумісність з багатьма мовами. SQLite зручна для розробників додатків, оскільки складається всього з одного файлу, а її бібліотека написана на Сі і має безліч прив'язок до інших мов програмування.

- Розміри БД. Кількість баз даних, а так само таблиць в них, обмежено тільки вільним місцем, наявним на сайті. Максимально можливий обсяг однієї бази даних становить 2 Тб.

Недоліки системи керування базами даних SqLite:

- Відсутність розмежування прав доступу. Інші СУБД включають в своєму складі систему управління правами доступу користувачів. Зазвичай застосування цієї функції не так критично, так як ця СУБД використовується в невеликих додатках.
- Швідкість роботи. Оскільки движок бази і інтерфейс до неї реалізований як єдине ціле, величезна перевага SQLite є висока продуктивність - для більшості типових завдань програма, побудована на SQLite, працює швидше, ніж при використанні MySQL, в 2-3 разів і швидше PostgreSQL в 10-20 разів.

Система керування базами даних PostgreSQL

PostgreSQL вільно поширювана система керування базами даних, що максимально відповідає стандартам SQL. PostgreSQL або Postgres намагаються застосовувати ANSI/ISO SQL стандарти своєчасно з виходом нових версій.

Від інших СУБД PostgreSQL відрізняється підтримкою об'єктно-орієнтованого та реляційного підходу до баз даних. Наприклад, повна підтримка надійних транзакцій, тобто атомарність, послідовність, ізоляційність, міцність (Atomicity, Consistency, Isolation, Durability (ACID).) Завдяки потужним технологіям Postgre дуже продуктивна. Паралельність досягнута не за рахунок блокування операцій читання, а завдяки реалізації управління різноманітним паралелізмом (MVCC), що також забезпечує відповідність ACID. PostgreSQL дуже легко розширювати своїми процедурами, які називаються збережені процедури. Ці функції спрощують використання постійно повторюваних операцій.

Хоча PostgreSQL і не може похвалитися великою популярністю на відміну від PostgreSQL, існує досить велике число додатків, що полегшують роботу з PostgreSQL, незважаючи на всю потужність функціоналу. Зараз досить легко встановити цю СУБД використовуючи стандартні менеджери пакетів операційних систем.

Переваги системи управління базами даних PostgreSQL

- Відкрите ПЗ відповідне стандарту SQL - PostgreSQL - безкоштовне ПЗ з відкритим вихідним кодом. Ця СУБД є дуже потужною системою.
- Велика спільнота. Існує досить велика спільнота в якому ви запросто знайдете відповіді на свої запитання
- Велика кількість доповнень - незважаючи на величезну кількість вбудованих функцій, існує дуже багато доповнень, що дозволяють розробляти дані для цієї СУБД і керувати ними.
- Розширення - існує можливість розширення функціоналу за рахунок збереження своїх процедур.
- Об'єктність - PostrgreSQL це не тільки реляційна СУБД, але також і об'єктно-орієнтована з підтримкою спадкування і багато іншого

Недоліки системи управління базами даних PostgreSQL

- Продуктивність - при простих операціях читання PostgreSQL може значно уповільнити сервер і бути повільніше своїх конкурентів, таких як PostgreSQL
- Популярність - за своєю природою, популярністю ця СУБД похвалитися не може, хоча і є досить велика спільнота.
- Хостинг - в силу вище перерахованих факторів іноді досить складно знайти хостинг з підтримкою цієї СУБД.

Ключовими критеріями вибору мови програмування для написання дипломної роботи служили такі чинники: можливості, реалізація та використання баз даних в додатку, труднощі вивчення самої мови програмування і гнучкість його можливостей.

Виходячи з перерахованих вище характеристик даного набору мов був зроблений вибір на користь мови програмування під платформу Android – Kotlin. Дана мова програмування має всі необхідні якості для написання дипломної роботи, серед яких: доступна, проста в реалізації база даних, лаконічний синтаксис, висока надійність.

Оптимальним рішенням для розробки сучасної програмної системи є використання об’єктно-реляційної системи керування базами даних MySQL, котра поєднує можливості реляційної системи керування базами даних та об’єктно-орієнтованої системи керування базами даних. Також MySQL має багато особливостей, яких немає в інших систем керування базами даних. До них можна віднести модель даних, яка підтримує об’єкти, створені користувачем, та їх поведінку. MySQL підтримую велику кількість різноманітних типів даних, та підтримують можливість створення масивів для цих типів даних. MySQL може працювати з великим обсягом даних.

4 ПРОЕКТУВАННЯ ЗАСТОСУНКУ «CALORIES TRACKER»

Після проведення аналізу предметної області, програмних продуктів-аналогів та вибору засобів розробки, необхідним етапом є проектування інформаційної системи. На цьому етапі необхідно створити UML-діаграми варіантів використання (прецедентів), активності і послідовності та розробити макети інтерфейсу застосунку.

4.1 Створення UML-діаграми варіантів використання системи

Діаграма варіантів використання (прецедентів) відображає функціональне призначення проектованої програмної системи. Суть діаграми прецедентів полягає в тому, що систему представляють як групу акторів, які за допомогою варіантів використання взаємодіють із нею.

Актор – це сутність, що взаємодіє з системою для вирішення деяких завдань. Актором може бути людина, інша система, пристрій або програмний засіб.

Після визначення акторів системи, необхідно сформувати перелік усіх варіантів використання, з якими будуть взаємодіяти визначені актори.

За результатами сформованих варіантів використання та акторів системи було розроблено діаграму варіантів використання, яку наведено на рис. 4.1.

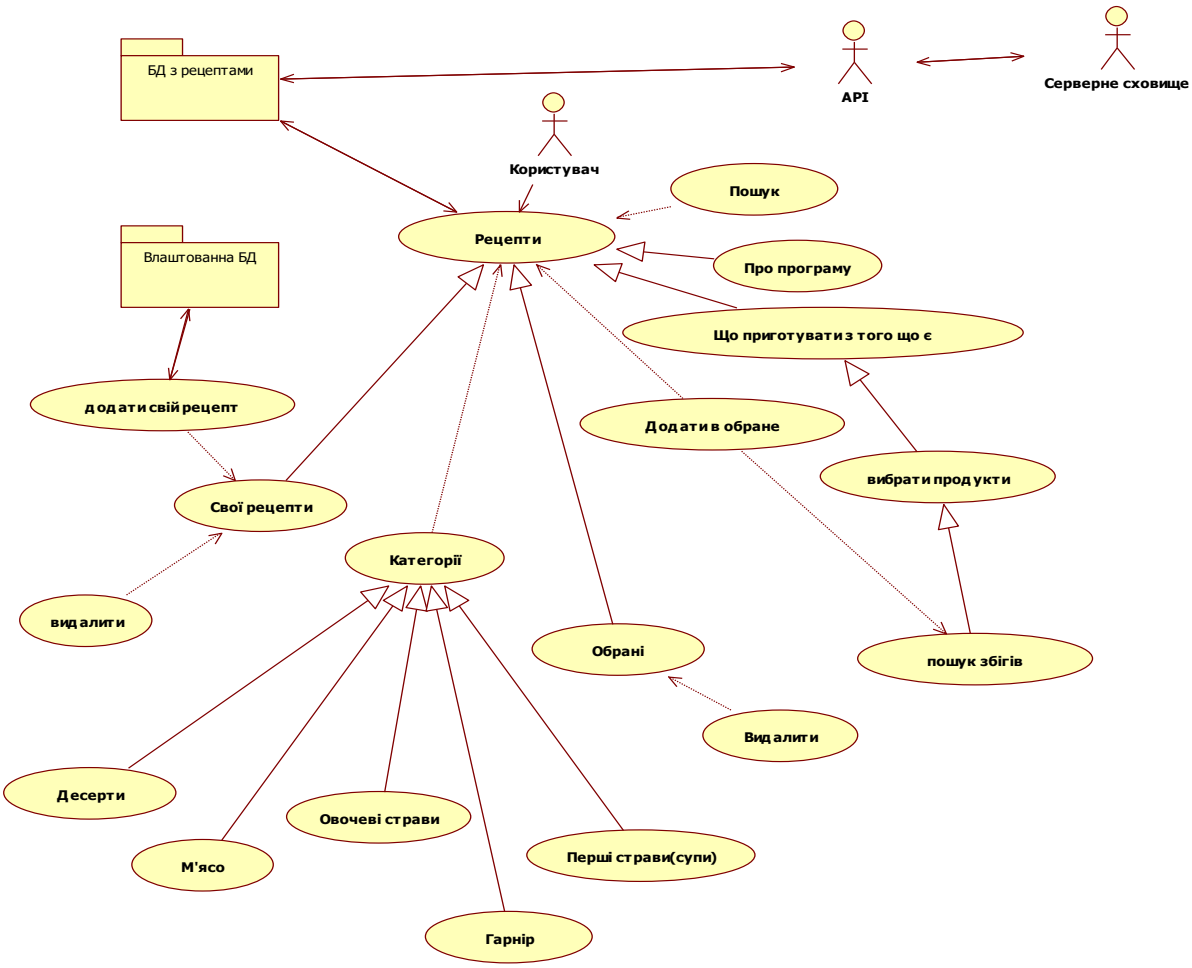


Рисунок 4.1 – Діаграма варіантів використання (прецедентів) системи

4.2 Створення діаграми послідовності

Також була створена діаграма послідовностей. Такі діаграми використовуються для уточнення діаграм прецедентів і більш детального опису логіки сценаріїв використання.

Головна послідовність сценарію розпізнавання зображення складається з взаємодій, які відображені на діаграмі на рис. 4.3: користувач може оттримати рецепти з сервера та відправити їх на сервер.

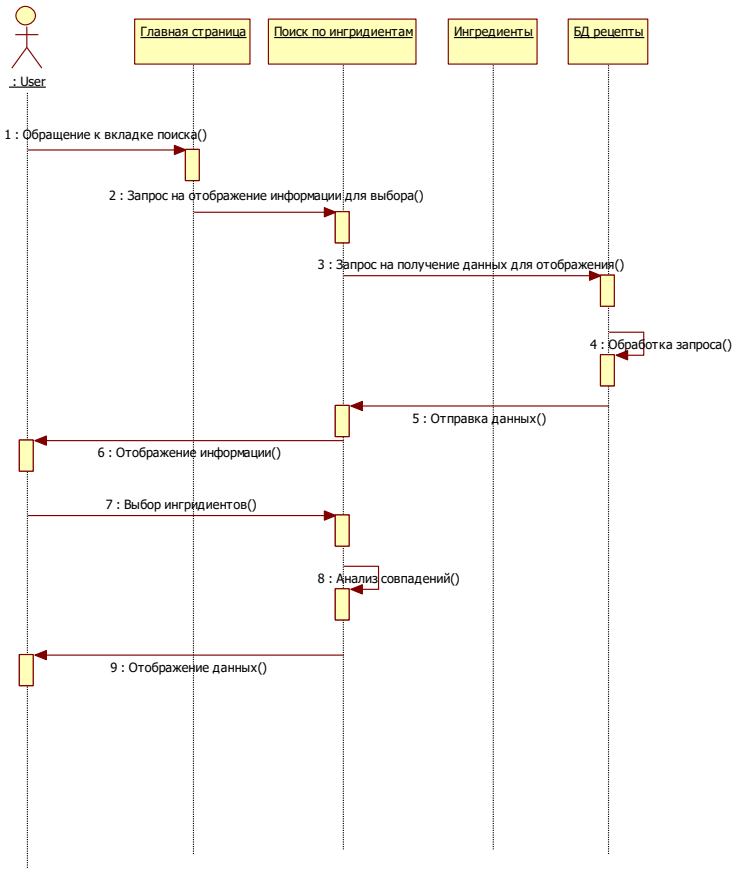


Рисунок 4.2 – Діаграма послідовностей

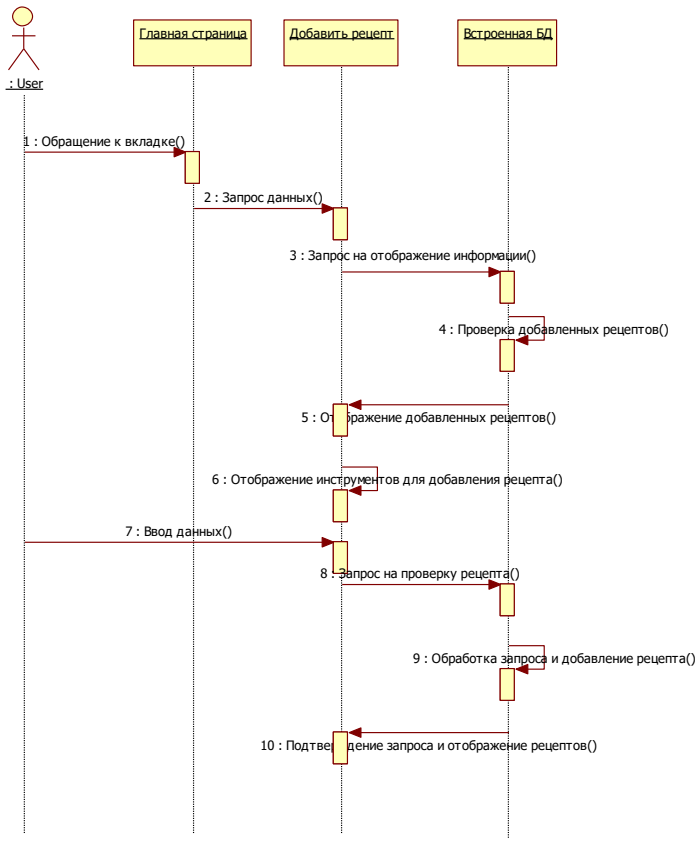
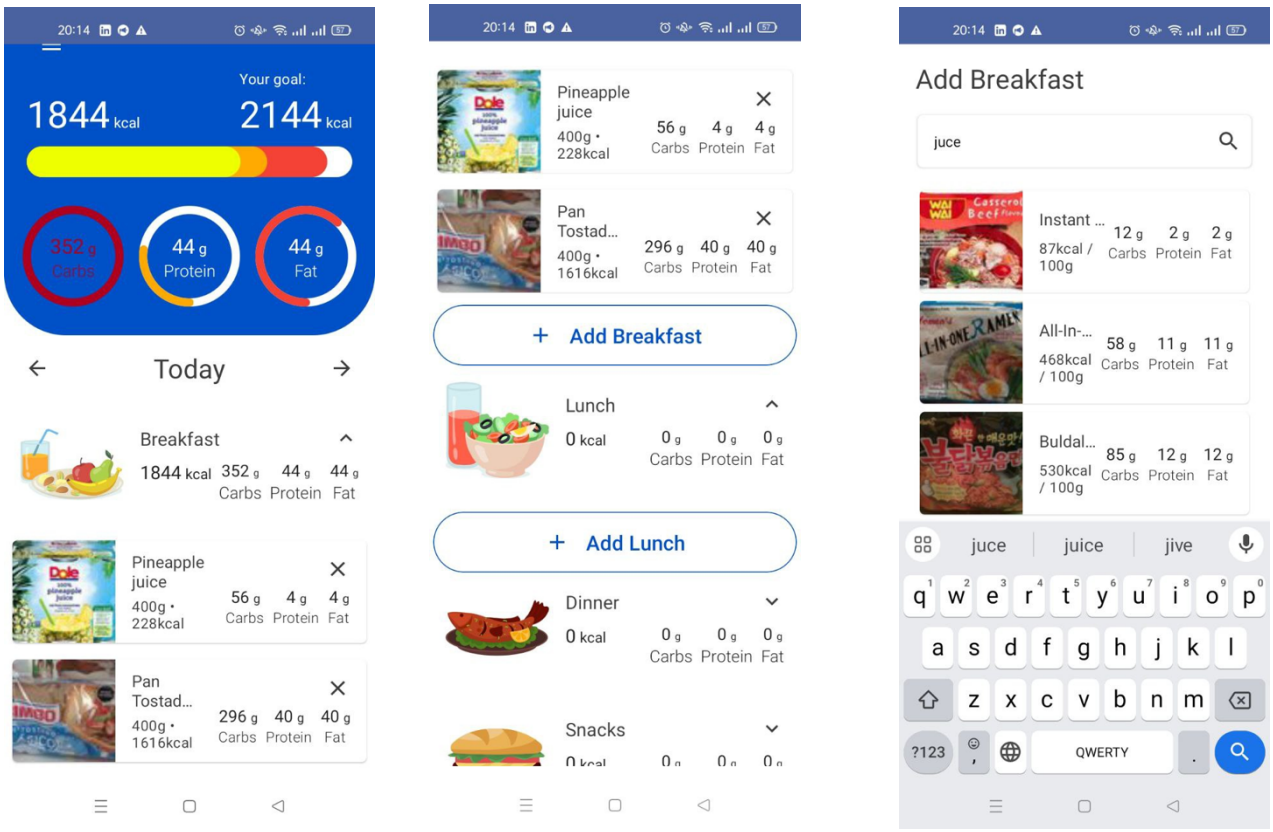


Рисунок 4.3 – Діаграма послідовності додавання нового рецепту.

4.3 Проектування макетів інтерфейсу застосунку

Після цього були створені макети інтерфейсу за допомогою Adobe XD. Було вирішено розробити 8 сторінок у застосунку та зробити дизайн легким та простим. Прототипи інтерфейсу наведені на рис. 4.4.



а) б) в)

Рисунок 4.4 – Макети інтерфейсу застосунку – а) головна сторінка б) сторінка з доданою їжею в) пошук рецепту

Таким чином, на етапі проектування системи були створені UML-діаграми варіантів використання (прецедентів), активності і послідовності та розроблені макети інтерфейсу застосунку. Це допомогло зрозуміти основні варіанти використання системи та логіку роботи застосунку.

Готовий застосунок містить таку функціональність:

1.реєстрація характеристик та побажань користувача, таких як вага, рост, вік, стать, бажання по раціону, відсотки білків, жирів та вуглеводів в раціоні приклад вказаний на рисунках 4.6, 4.7, 4.8

2. головний екран, зміна теми додатку зі світлої на темну.

3. екран додавання рецепту

4. екран пошуку рецепту
5. запити в бд та збереження обранних рецептів в локальну базу даних.
6. анімації кількості білків, жирів, вуглеводів та КБЖВ.

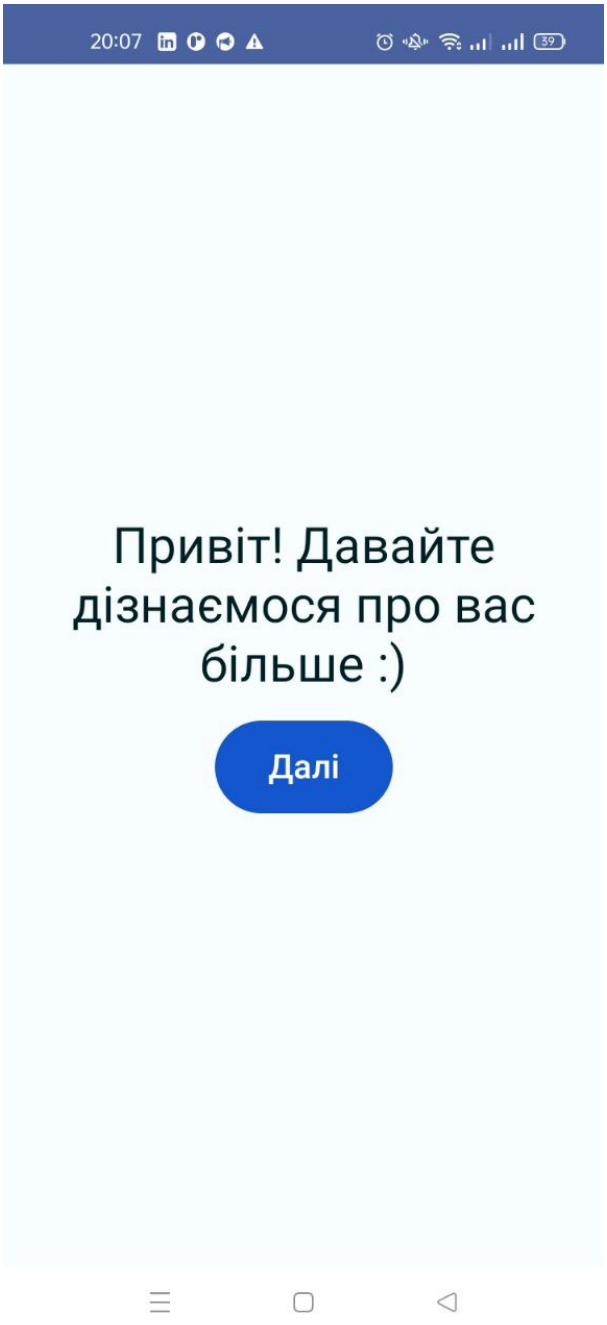
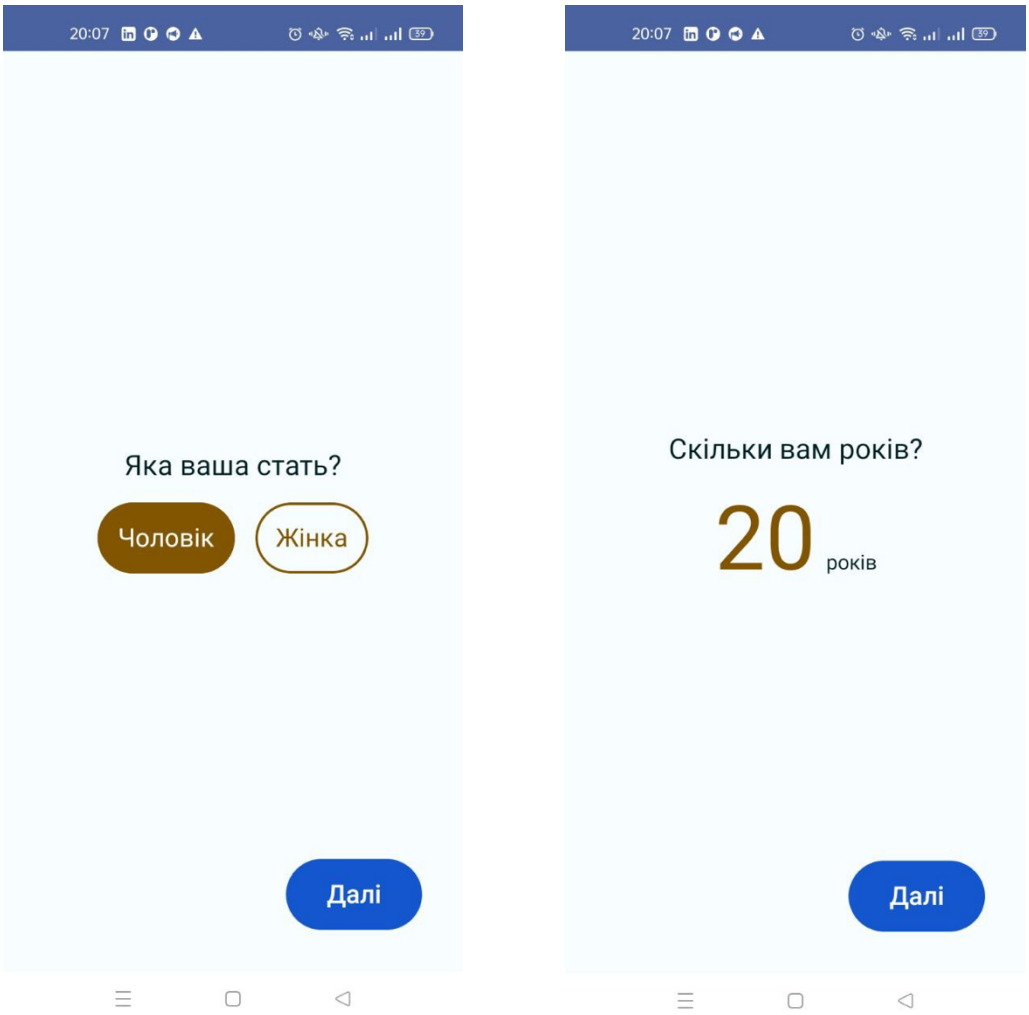


Рисунок 4.6 – інтерфейс готового застосунку, екран привітання користувача



а) б)

Рисунок 4.7 – інтерфейс готового застосунку, приклади екранів а) вибору статі б) віку

Рисунок 4.8 – інтерфейс готового застосунку, приклад екрану побажань щодо поживних речовин

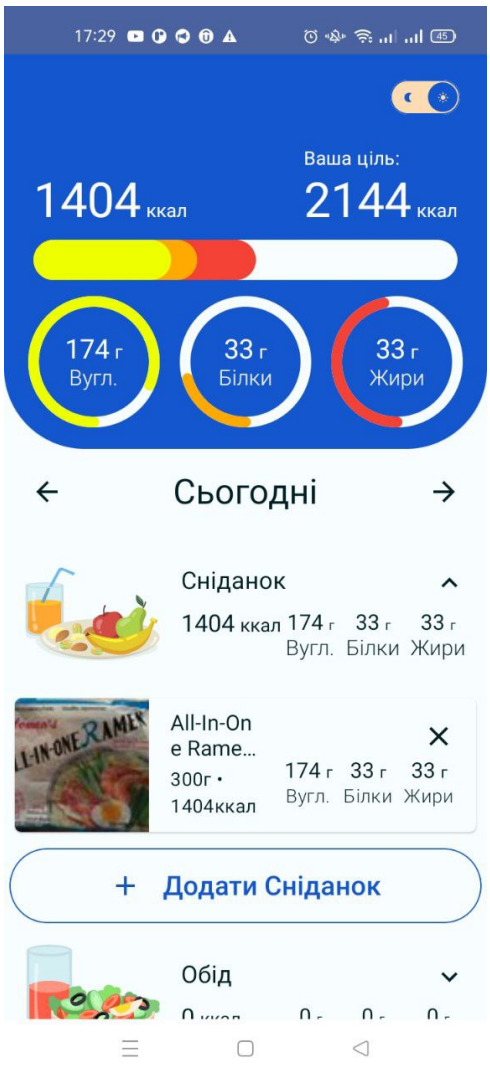


Рисунок 4.9 – інтерфейс готового застосунку, головний екран

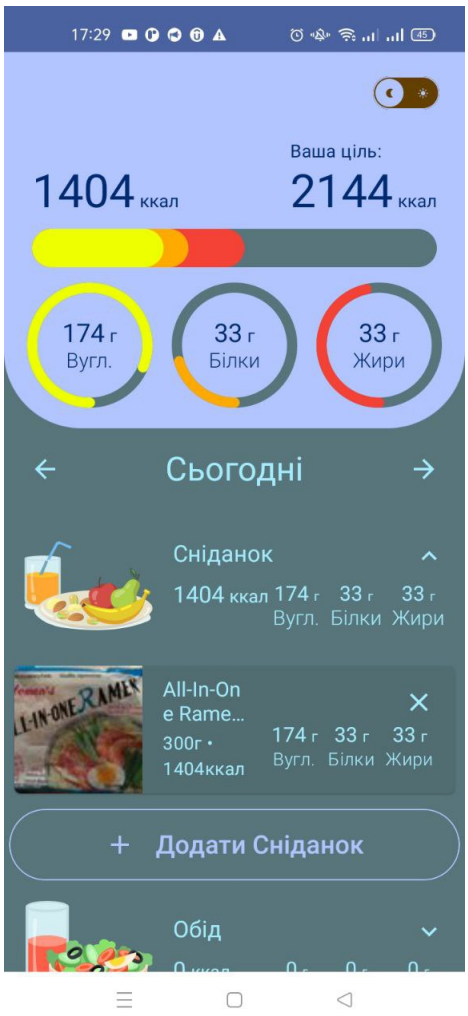


Рисунок 4.10 – інтерфейс готового застосунку, головний екран з темною темою

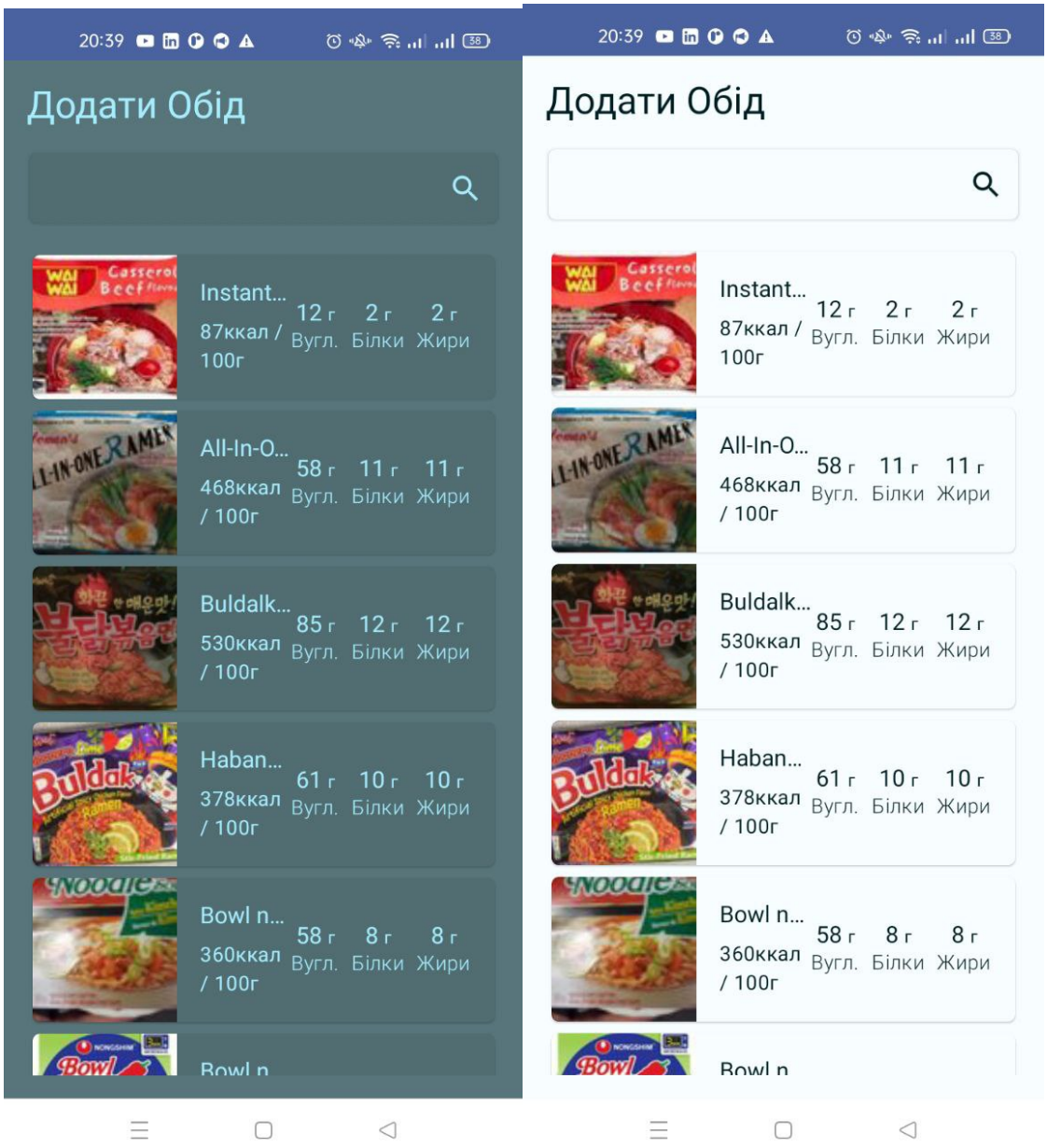


Рисунок 4.11 – інтерфейс готового застосунку, приклади екранів

а) додавання рецепту – темна тема б) додавання рецепту – світла тема

5. РЕАЛІЗАЦІЯ ЗАСТОСУНКУ З РОЗРАХУНКУ КБЖВ

Для створення мобільного додатку для пошуку рецепту страви виходячи з інгредієнтів, використовувалася мова програмування Kotlin, технологія Realm Framework для збереження данних користувача, API ,яке було створено за допомогою PHP для виведення інформації з бази даних та надсилання її у застосунок, а також система керування базами даних MySQL.

5.1. Кросплатформна мобільна база даних Realm

Realm - це крос-платформна база даних, доступна для Swift, Objective-C, Java, JavaScript, Xamarin, яка створювалася спеціально для мобільних пристроїв з оглядкою на недоліки Core Data і SQLite.

Realm - перша база даних, яка розроблялася спеціально для мобільних пристроїв і для роботи саме на цих пристроях.

Realm краще і швидше за свої аналоги, при цьому він простий у використанні і за допомогою кількох рядків коду ви можете, як покласти об'єкт в базу, так і викликати його з бази.

Завдяки використанню власного двигуна, що забезпечує високу швидкість роботи і простоту в застосуванні, мобільна база даних Realm виконує запити і синхронізує об'єкти значно швидше, ніж Core Data і SQLite, і здійснює паралельний доступ до даних без проблем. Це означає, що кілька джерел можуть отримати доступ до одного і того ж об'єкту без необхідності керувати блокуванням або будь-яких проблем з неузгодженістю даних. Дана особливість є серйозною конкурентною перевагою. Практика показує, що в більшості випадків Realm значно перевершує в швидкості не тільки SQLite, але й інші альтернативні ORM для Android, такі як ORMLite і Greendao.

Розвиток Realm розпочався наприкінці 2010 року Олександром Стигсеном разом з Бьярном Крістіансеном під назвою TightDB. Компанія розпочала свою діяльність у 2011 році в Combinator. Його рекламували як

NoSQL із настроюваною довговічністю та можливістю обміну однаковими групами даних у кількох процесах, а також навіть у декількох пристроях та кластерах.

Realm не підтримує спадкування, що є серйозною проблемою. Будь-Realm об'єкт повинен або успадковуватися від RealmObject або реалізовувати інтерфейс маркер RealmModel і бути поміченим анотацією @RealmClass. Успадковуватися від існуючих Realm об'єктів не можна. Рекомендується використовувати композицію замість наслідування.

Оскільки Realm є сховищем об'єктів, його типізовані для мови API відображають набрані об'єкти безпосередньо у файл Realm - тому класи використовуються як визначення схеми.

Realm відмінно підходить для MV * архітектур, коли вся реалізація ховається за інтерфейсом бази даних. Всі звернення і вибірки відбуваються в модулі бази даних. При записи об'єктів відкриваємо Realm, записуємо дані і закриваємо його, на вхід подається тільки об'єкт для збереження. Використання Realm (без методу .copyFromRealm ()) накладає серйозні обмеження на використання підходу Clean Architecture. Використовувати різні моделі даних для різних верств не вийде, пропадає весь сенс live об'єктів. Також труднощі виникають при створенні незалежних шарів і відкриття / закриття Realm, так як ця операція прив'язана до життєвого циклу Activity / Fragment. Частковим вирішенням цієї проблеми буде створення ізольованого шару отримання даних, перетворення об'єктів і збереження їх в базі даних.

Realm дуже зручний при побудові offline-first додатків, коли всі дані для відображення ми отримуємо з бази даних .

Зв'язок між об'єктами дозволяється через "посилання". Кожне "посилання" створює "зворотне посилання" як зворотне відношення до будь-яких об'єктів, які посилаються на поточний об'єкт.

Результати запиту, що повертаються Realm, - це локальні подання потоків до поточної "версії бази даних", і ці подання "автоматично оновлюються", коли транзакція фіксується з будь-якого потоку , поки Realm

здатний оновити свою версію примірника (що можливо для потоків, які можуть отримувати сповіщення про зміни). Коли це трапляється, Realm викликає прослуховувачі змін, які додаються до результатів його запиту (якщо вони змінилися).

Кожне локальне подання потоку повертає проксі-об'єкти, які читають із бази даних та записують її лише при виклику методу доступу, тобто весь доступ до бази даних ледаче завантажується. Записи дозволяються лише під час транзакції запису.

Оскільки кожен результат запиту та кожен об’єкт проксі є видом на базові дані, будь-які зміни, внесені до бази даних, відображаються у всіх об’єктах, які вказують на ті самі дані. Realm зазвичай називає цю поведінку "архітектурою з нульовим копіюванням" (разом із раніше згаданим доступом до завантажених даних).

Основні класи фреймворку.

Realm: екземпляри Realm - це серце фреймворка, це ваша точка доступу до основної бази даних, аналогічно керованого контексту об'єкта в Core Data. Ви будете створювати екземпляри за допомогою ініціалізатор Realm ().

Object (Об'єкт): Це ваша модель Realm. Процес створення моделі визначає схему бази даних. Для того, щоб створити модель ви просто створюєте підклас Object і визначаєте fields (поля), які ви хочете зберегти як властивості.

Relationships (Зв'язки або відносини): Ви створюєте зв'язок one-to-many (один до багатьох) між об'єктами, просто оголошуючи властивість типу Object, на яке ви хочете посилатися.

Write Transactions (Запис операцій): Будь-які операції в базі даних, такі як створення, редагування або видалення об'єктів, повинні виконуватися в рамках операцій writes, які відбуваються за допомогою виклику write (_ :) у примірників Realm.

Queries (Запити): Для вилучення об'єктів з бази даних вам потрібно використовувати запити. Найпростіша форма запиту - це виклик objects () у

примірника Realm, передаючи клас Object, який ви шукаєте. Якщо ваші пошукові запити є більш складними, ви можете використовувати предикати, вибудовуючи запити, і також організовуючи результати пошуку.

Results (Результати): Результати - це автоматично оновлюваний тип контейнера, який ви отримуєте у відповідь на об'єктні запити. У них багато спільного зі звичайними масивами, в тому числі синтаксис сабскріпта для захоплення елемента в індексі.

При розробці ця технологія використовується для збереження рецептів, які користувач записує у застосунок.

5.2 API для отримання інформації з бази даних

Використовуючи можливості мови PHP було розроблено API яке використовується для виведення інформації з бази даних та надсилання її у застосунок. API - інтерфейс взаємодії з будь-яким об'єктом (програмою, додатком), що включає в себе набір правил, які дозволяють одному з додатком спілкуватися з іншим. Ці «правила» можуть включати в себе операції створення, читання, оновлення та видалення. Прикладом API може служити всім відома бібліотека jQuery.

Хостинг, на якому знаходяться API та сама База данних, дозволяє звертатися до API за посиланням, та отримувати данні у форматі JSON.

Треба зауважити, що данне API являє собою реалізацію яка використовує концепію Rest. Rest - це концепція для організації взаємодії між незалежними об'єктами за допомогою протоколу HTTP. Включає в себе набір принципів взаємодії клієнт-серверних додатків. Зазвичай він представлений в форматі JSON.

У багатьох додатках REST API необхідний, тому що це найлегший спосіб створення, читання, оновлення або видалення інформації між різними додатками через Інтернет або протокол HTTP. Ця інформація може надаватися

користувачеві в одну мить, особливо якщо ви використовуєте JavaScript для відображення даних на веб-сторінці.

Типовий процес роботи з СУБД в PHP-сценарії складається з декількох кроків:

- Установити підключення до сервера СУБД, передавши необхідні параметри: адреса, логін, пароль.
- Переконатися, що підключення пройшло успішно: сервер СУБД доступний, логін і пароль вірні і так далі.
- Сформувати правильний SQL запит (наприклад, на читання даних з таблиці).
- Переконатися, що запит був виконаний успішно. Отримати результат від СУБД у вигляді масиву з записів.
- Використовувати отримані записи в своєму сценарії (наприклад, показати їх у вигляді таблиці).

Зразок коду, який використовується для звертання до API із мобільного додатку, використовуючи стабільне та надійне підключення до інтернету зображен нижче:

```
let url:URL = URL(string: "https://api. hurriyet. com.tr /v1
/articles...deleted...")!
let session = URLSession.shared
var request = URLRequest(url: url)
request.httpMethod = "GET"
request.cachePolicy = NSURLRequest.CachePolicy.reloadIgnoringCacheData
URLCache.shared = URLCache(memoryCapacity: 0, diskCapacity: 0, diskPath: nil)
let task = session.dataTask(with: request as URLRequest)
{
    (data, response, error) in zguard let data = data, let _:URLResponse =
response, error == nil else
{
        print("error")
        return
}
let dataString =  String(data: data, encoding: String.Encoding.utf8)
```

5.3 База даних MySQL

Для збереження данних була обрана MySQL, але для звернення та отримання данних потрібне API. MySQL потребує розгорнення окремого

серверу для роботи та збереження інформації. Саме тому мобільні додати використовують такі Бази данних як Core data, Realm та інші. До цих данних можно звертатися запитами, на відміну від випадка із MySQL, та швидкість читання та збереження данних переважає у СКБД для мобільних додатків.

MySQL — компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Застосунок працює сувмістно із API та Базою данних за такою схемою, зображенною на рисунку 3.1:

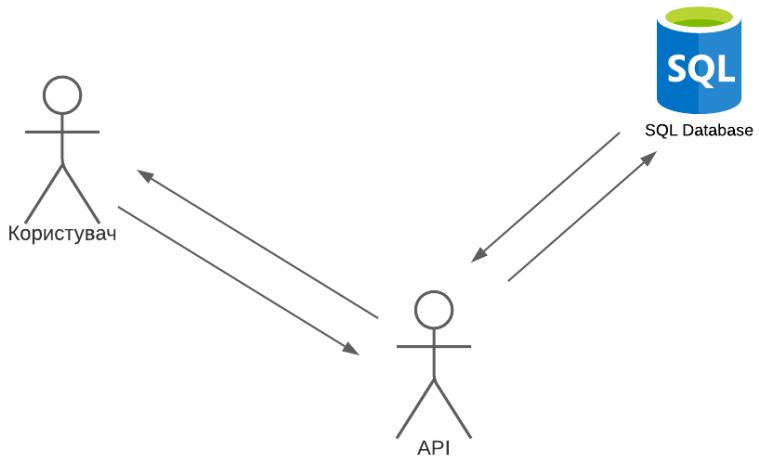


Рисунок 3.1 – Схема зв’язку додатку із Базою данних

Під час запуску застосунок робить запит до API через посилання, яке надає хост, а у свою чергу API звертається до Бази данних, роблячи GET запит та переносячи результат запиту до JSON масиву, який вже отримує застосунок, та далі із ним працює. Для отримання данних застосунок повинен мати підключення до інтернету.

5.4 Інструменти для розробки програмного забезпечення з вирахування КБЖВ

Для розробки додатку були використані такі бібілотеки та залежності, як Kotlin, Coroutines & Kotlin Flow, Dagger2, Room, Android Jetpack, Jetpack Compose, RxJava, Jupiter, MockK.

Kotlin: Kotlin - це сучасна мова програмування, розроблена JetBrains. Вона повністю сумісна з Java, але пропонує більш стислий синтаксис і низку поліпшень, таких як безпека від null-покажчиків і корутини для асинхронного програмування. У вашому проєкті Kotlin використовувався для написання всього коду програми. Ви використовували його для створення класів, функцій та інтерфейсів, які забезпечують функціональність вашої програми.

Clean Architecture: Це концепція, запропонована Робертом Мартіном (Uncle Bob), яка ставить бізнес-логіку в центр уваги, а деталі реалізації (як-от бази даних та інтерфейси) знаходяться на периферії. У вашому проєкті Clean Architecture використовувалася для забезпечення чистоти та гнучкості коду, що дало змогу легко модифікувати й тестувати різні компоненти системи. Ви розділили ваш застосунок на модулі, кожен з яких має свою власну сферу відповідальності.

Coroutines & Kotlin Flow: Coroutines - це функції, які можна призупинити та відновити, що дає змогу писати асинхронний код так само просто, як і синхронний. Kotlin Flow - це тип корутин, який може послідовно видавати кілька значень. У вашому проєкті вони використовувалися для обробки асинхронних операцій, таких як запити до бази даних або мережеві запити. Ви

використовували корутини для обробки довгих операцій, таких як завантаження даних з інтернету або читання з бази даних, без блокування основного потоку.

Dagger2 - це фреймворк для впровадження залежностей, який генерує код впровадження залежностей на етапі компіляції. Це забезпечує ефективність і передбачуваність порівняно з іншими фреймворками, які аналізують залежності під час виконання. У вашому проєкті Dagger2 використовувався для управління залежностями між різними класами та модулями. Ви використовували Dagger2 для створення та надання залежностей, як-от об'єкти бази даних або мережеві клієнти, які потім впроваджували в потрібні місця у вашому коді.

Room - це бібліотека для роботи з базами даних SQLite на Android. Вона надає абстракцію на рівні об'єктів над SQLite і дає змогу зручніше працювати з базами даних. У вашому проєкті Room використовувався для зберігання інформації про продукти та їхню енергетичну і поживну цінність. Ви використовували Room для створення таблиць бази даних, які зберігають інформацію про продукти, їхню енергетичну та поживну цінність, а також споживання продуктів користувачами.

Java – була офіційною мовою програмування на Android і залишається найпоширенішою, навіть після появи Kotlin, Dart та інших мов для цієї ОС1. Android Studio і Java використовуються для написання Android-додатків, включно з інтерактивними кнопками та відображенням другого екрана при натисканні кнопки.

Model-View-ViewModel (MVVM) значно покращує розробку Android-додатків, просуваючи чистий, керований код і ефективне управління даними3. MVVM встановлює новий стандарт в Android-розробці завдяки своєму акценту на поділ обов'язків, поліпшеній тестованості та масштабованості4. Реалізація шаблону репозиторію в шарі моделі допомагає в ефективному управлінні даними та масштабованості, абстрагуючи доступ до джерела даних.

MVP (Model-View-Presenter) - це один із найпопулярніших шаблонів архітектури, що є дійсним під час організації проєкту5. MVP надає можливість виділити весь той нудний низькорівневий Android-код, що потрібен для відображення нашого інтерфейсу та взаємодії з ним, у Подання.

Android Jetpack - це набір бібліотек, які допомагають розробникам слідувати кращим практикам, зменшувати шаблонний код і писати код, який працює послідовно на всіх версіях і пристроях Android. Jetpack Compose - це сучасний інструментарій Android для створення нативного користувацького інтерфейсу.

Jetpack Compose - це сучасний інструментарій Android для створення нативного користувацького інтерфейсу. Він спрощує і прискорює розробку користувацького інтерфейсу на Android.

Retrofit і OkHttp - це дві ключові бібліотеки, що використовуються в Android для роботи з мережевими запитами. Retrofit - це типобезпечний HTTP-клієнт для Android і Java, який дає змогу Android-додаткам легко підключатися й отримувати дані з веб-сервісів10. OkHttp - це HTTP-клієнт із відкритим вихідним кодом для Java і Kotlin, розроблений Square, який також створив Retrofit.

RxJava - це бібліотека Java, яка дозволяє функціональному реактивному програмуванню в Android-розробці12. Він піднімає рівень абстракції навколо потоків для спрощення реалізації складної паралельної поведінки13.

Unit-тестування в Android-розробці здебільшого фокусується на тестуванні невеликих, окремих частин коду, таких як окремі методи або функції, незалежно від операційної системи Android і користувацького інтерфейсу14. Unit-тести або малі тести перевіряють тільки дуже невелику частину програми, таку як метод або клас15.

Jupiter, наскільки мені відомо, не є поширеним інструментом в Android-розробці. Можливо, ви мали на увазі JUnit, фреймворк для написання і запуску unit-тестів у Java та Android16.

MockK - це потужна бібліотека для мови Kotlin, яка надає безліч корисних функцій17. Вона підтримує звичайні unit-тести, Android інструментальні тести за допомогою успадкування (< Android P) та Android інструментальні тести за допомогою вбудовування (≥ Android P)18.

ВИСНОВКИ

У даному дипломному проекті розроблена застосунок з управління базою кулінарних рецептів. Вона забезпечує користувача можливістю швидко підбирати рецепт, що зберігає його час та економить простір, завдяки тому що всі рецепти можна зберігати у додатку.

При розробці дипломної роботи були вирішені такі задачі:

– проведено огляд інформаційних систем аналогів та виявлення функціональних можливостей таких систем;

– зроблено вибір технології розробки мобільного додатку, який розробляється;

– розроблене початкове функціональне проектування програмної системи;

– виконаний опис функціональних можливостей мов програмування, використаних при розробці програми;

– розроблений програмний засіб згідно вимог та технічного завдання;

– описані функціональні складові програмного засобу з графічними матеріалами;

Мобільний застосунок з підрахунку КБЖВ можливо використовувати будь якому користувачу без необхідності мати спеціальні знання комп’ютерів, планшетів або телефону.

Список використаної літератури

1. ДСТУ 3008 - 2015. Звіти у сфері науки і техніки. Структура та правила оформлення.
2. Марк Д. Android 32 SDK. Розробка додатків для Android на Kotlin. Наттінг Д., Ламарш Д., Олссон Ф. СПб.:Вільямс, 2013. - 672 с.
3. Гамма Е. Прийоми об'єктно-орієнтованого проектування, патерни проектування; [пер. з англ.: А. Слінкін; наук. ред.: М. Шалаєв]. Хелм Р., Джонсон Р., Вліссідес Д. СПб: Пітер, 2014. - 366 с
4. Арлоу Д., Нейштадт А. UML 2 і Уніфікований процес: Практичний об'єктно-орієнтований аналіз і проектування (пер. з англ. Шатохіної Н.). - 2-е изд. - М.: Символ Плюс, 2008. - 622 с.
5. Tarasewich P. Designing Mobile Commerce Applications. // Communications of the acm. Грудень 2003. Vol. 46. Issue 12. - P. 57 - 60.
6. Головач В.В. Дизайн користувацького інтерфейсу v1.2. URL: http://uibook2.usethics.ru/
7. Документація Android. URL: https://developer.android.com/library/android/documentation(дата звернення:19.04.2024).
8. Офіційна документація платформи Realm. [Електронний ресурс] URL: https://realm.io/docs/objc/latest/ (дата звернення: 20.04.2024).
9. Застосування спецій у кулінарії. [Електронний ресурс] URL:https://vsadu.ru/post/kakie-specii-ispolzovat-v-prigotovlenii.html (дата звернення: 20.04.2024).
10. Android Databases. [Електронний ресурс] URL:https://rollout.io/blog/ios-databases-sqllite-core-data-realm/ (дата звернення:20.04.2024).
11. Interface Builder. [Електронний ресурс] URL: https://developer.apple.com/xcode/interface-builder/ (дата звернення:20.04.2024).