

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка
до дипломного проєкту (роботи)
бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему «РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ПІДРАХУНКУ
КАЛОРІЙ КОРИСТУВАЧА»

Виконав: студент 4 курсу, групи КНТ-512сп
спеціальності 123 Комп'ютерна інженерія

Освітня програма (спеціалізація)

Комп'ютерна інженерія

(код і назва спеціальності)

КРАПІВКІН К.Г.

(прізвище та ініціали)

Керівник ІЛЛЯШЕНКО М.Б.

(прізвище та ініціали)

Рецензент МАЛИЙ О.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти (освітній ступінь) бакалаврський
 Спеціальність 123 Комп'ютерна інженерія
 (код і назва)
 Освітня програма (спеціалізація) Комп'ютерна інженерія
 (назва)

ЗАТВЕРДЖУЮ

Зав. кафедри

Кудерметов Р.К.

“ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

КРАПІВКІН Кирило Геннадійович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Розробка мобільного додатку для підрахунку калорій користувача»

керівник проекту (роботи) ІЛ'ЯШЕНКО М.Б. к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “01” квітня 2025 року № 105

2. Строк подання студентом проекту (роботи) 01.06. 2025 року

3. Вихідні дані до проекту (роботи) операційна система Android,

мова програмування Java, платформа Android Studio

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____

1) Опис предметної області _____

2) Вибір засобів реалізації _____

3) Реалізація програмного забезпечення для системи _____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди презентацій

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ІЛЬЯШЕНКО М.Б. к.т.н.. доцент		
Нормоконтроль	ЩЕРБАК Н.В., ст. викладач		

7. Дата видачі завдання 01.04. 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Опис предметної області	05.04.2025	
2	Вибір засобів реалізації	10.05.2025	
3	Вибір програмного забезпечення	17.04.2025	
4	Вибір середовища розробки	23.04.2025	
5	Реалізація програмного забезпечення	18.05.2025	
6	Принцип роботи додатку	25.05.2025	
7	Оформлення отриманих результатів у ПЗ	30.05.2025	
8	Захист роботи		

Студент Кирило КРАПІВКІН
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) Матвій ІЛЬЯШЕНКО
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:

ПЗ: 58 с., 16 рис., 3 табл., 19 джерел.

ANDROID, ANDROID STUDIO, IDE UML, БАЗА ДАНИХ, ЗАСТОСУНОК, КАЛОРІЇ, ХАРЧУВАННЯ

У цій бакалаврській роботі було розроблено мобільний додаток для підтримки здорового харчування, реалізований за допомогою мови програмування Java та інтегрованого середовища розробки Android Studio. Додаток орієнтований на користувачів, які прагнуть покращити якість свого життя шляхом контролю харчування, що є особливо актуальним у сучасних умовах, коли більшість людей ведуть малорухомий, сидячий спосіб життя, мають низьку фізичну активність і стикаються з великою кількістю шкідливих харчових звичок.

Метою створення цього додатку є надання користувачеві зручного інструменту для формування правильного раціону, контролю спожитих калорій, білків, жирів та вуглеводів, а також ведення особистої статистики харчування.

Програмне забезпечення має інтуїтивно зрозумілий інтерфейс, що дозволяє легко додавати продукти, редагувати їх інформацію, зберігати улюблені позиції, переглядати детальну інформацію про харчову цінність та історію прийомів їжі. Додаток працює на основі формули Міффіна – Сан Жеора, яка використовується для розрахунку добової потреби в калоріях з урахуванням таких параметрів, як вік, вага, зріст, стать та рівень фізичної активності користувача.

На момент завершення роботи додаток знаходиться на етапі прототипу та проходить бета-тестування. Завантаження здійснюється у вигляді APK-файлу, який користувачі можуть встановити на свої пристрої вручну. Після завершення тестування, усунення виявлених помилок та врахування відгуків користувачів

планується випуск стабільної версії на платформі PlayMarket.

Таким чином, у результаті виконаної роботи створено мобільний додаток, який виконує всі поставлені функції, забезпечує користувача зручними інструментами для контролю харчування і має потенціал для подальшого розвитку та вдосконалення.

ABSTRACT

Explanatory note to the master's work: 58 p., 16 figures, 3 tables, 19 sources.

ANDROID, ANDROID STUDIO, IDE UML, DATABASE, APPLICATION,
CALORIES, NUTRITION

This bachelor's thesis developed a mobile application to support healthy eating, implemented using the Java programming language and the Android Studio integrated development environment. The application is aimed at users who want to improve their quality of life by controlling their diet, which is especially relevant in today's world, where most people lead a sedentary lifestyle, have low physical activity, and face a large number of unhealthy eating habits.

The purpose of this application is to provide users with a convenient tool for forming a proper diet, controlling calorie, protein, fat, and carbohydrate intake, and keeping personal nutrition statistics.

The software has an intuitive interface that allows you to easily add products, edit their information, save your favorite items, and view detailed information about nutritional value and meal history. The app is based on the Mifflin-St. Jeor formula, which is used to calculate daily calorie requirements based on parameters such as age, weight, height, gender, and the user's level of physical activity.

At the time of completion, the app is in the prototype stage and is undergoing beta testing. It is downloaded as an APK file, which users can install on their devices manually. After testing is complete, identified bugs are fixed, and user feedback is taken into account, a stable version is planned to be released on the PlayMarket platform.

Thus, as a result of the work performed, a mobile application has been created that performs all the required functions, provides the user with convenient tools for monitoring nutrition, and has the potential for further development and improvement.

ЗМІСТ

Вступ.....	10
1 Опис предметної області	11
1.1 Огляд існуючих мобільних додатків для підрахунку калорій	11
1.1.1 MyFitnessPal.....	11
1.1.2 Lifesum	11
1.1.3 FatSecret	13
1.1.4 Cronometer.....	13
1.1.5 YAZIO	14
2 Вибір засобів реалізації.....	16
2.1 Вибір програмного забезпечення	16
2.1.1 Опис мови програмування Java	16
2.1.2 Опис мови програмування Kotlin	23
2.1.3 Опис мови програмування Dart	25
2.1.4 Опис мови програмування C++	27
2.2 Вибір середовища розробки	30
2.2.1 Опис Andriod Studio	30
2.2.2 Опис IntelliJ IDEA як середовище розробки.....	33
2.2.3 Опис Eclipse як середовище розробки	34
2.4 Вибір бази даних	36
2.4.1 SQLite.....	36
2.4.2 Room (Android Jetpack)	37
2.4.3 Firebase Realtime Database	37
2.3 Операційна система Android.....	39
3 Реалізація програмного забезпечення для системи.....	47
3.1 Діаграми варіантів використання	47
3.2 Зберігання даних	49
3.3 Принцип роботи додатку	51

3.4 Вивинтаження додатку	54
Висновки.....	55
Перелік джерел посилання	56
Додаток А	58

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

ADT	Android Development Tools , Інструменти розробки для Android
AOT	Ahead-of-Time Compilation, Попереднє компілювання
ARM	Advanced RISC Machines , архітектура процесорів
iOS	iPhone Operating System , Операційна система iPhone
G1GC	Garbage-First Garbage Collector, Збірник сміття «Garbage-First»
GitHub	Git Repository Hosting Service , Сервіс для розміщення Git-репозиторіїв
JNI	Java Native Interface , Інтерфейс Java для взаємодії з нативним кодом
JIT	Just-In-Time Compilation , Динамічне компілювання
JVM	Java Virtual Machine, Віртуальна машина Java

ВСТУП

Сучасний світ безпосередньо залежить від науково-технічного прогресу: більшість сфер діяльності людини повністю інтегровані з технічними та програмними рішеннями, без яких існування сучасної людини було б значно ускладненим. Програмні рішення проникають у всі аспекти життя — від складних професійних завдань до рекомендацій щодо харчування. Оскільки більшість сучасного суспільства веде малорухомий спосіб життя і працює сидячи, здорове харчування стає одним із ключових методів підтримки здоров'я.

З огляду на це, розробка інформаційних систем для здорового харчування є надзвичайно актуальною.

Об'єктом дослідження є принципи і методи розробки мобільних додатків, а предметом — аналіз інструментів і методів створення мобільних додатків та їх застосування для розробки програми, що сприяє підтриманню здорового харчування.

Основна мета роботи — проектування і розробка мобільного додатку для здорового харчування.

Для досягнення цієї мети поставлено такі завдання:

- проаналізувати поняття здорового харчування,
- розглянути поняття мобільних додатків,
- дослідити методи розробки мобільних додатків;
- провести огляд мови програмування,
- оглянути середовище розробки,
- проаналізувати цільову операційну систему;
- створити діаграму варіантів використання,
- описати структуру програмного забезпечення,
- побудувати діаграму класів,
- розробити графічний інтерфейс користувача,
- підготувати інструкцію користувача.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих мобільних додатків для підрахунку калорій

На сьогоднішній день існує велика кількість мобільних додатків для підрахунку калорій, які допомагають користувачам контролювати свій раціон і досягати фітнес-цілей. Вони пропонують різноманітні функції, такі як ведення харчового щоденника, автоматичний розрахунок калорійності продуктів та моніторинг фізичної активності. Крім того, багато з цих додатків використовують машинне навчання та штучний інтелект для більш точного прогнозування потреб користувача [1]. У цьому розділі буде розглянуто найбільш популярні з них, їхні переваги та недоліки.

1.1.1 MyFitnessPal

MyFitnessPal – один з найпопулярніших додатків для підрахунку калорій. Основні можливості (рис.1.1):

- велика база продуктів (понад 11 мільйонів найменувань);
- сканування штрих-кодів для швидкого додавання продуктів;
- інтеграція з фітнес-трекерами (Fitbit, Apple Health, Google Fit);
- можливість налаштовувати індивідуальні цілі споживання калорій;
- автоматичне прогнозування ваги на основі щоденного споживання калорій.

Недоліки: безкоштовна версія містить рекламу, а деякі функції доступні лише в платній підписці. Крім того, база продуктів може містити неточності через додавання користувачами [2].

1.1.2 Lifesum

Lifesum – додаток, що комбінує підрахунок калорій з рекомендаціями щодо здорового харчування. Основні функції (рис.1.2):

- персоналізовані плани харчування;
- оцінка якості раціону;
- мотиваційні нагадування та рекомендації;

- вбудовані тести для визначення харчових звичок;
- аналіз водного балансу організму.

Недоліки: обмежені можливості в безкоштовній версії, відсутність повної інтеграції з українськими продуктами. Також користувачі скаржаться на обмежену базу локальних страв і відсутність можливості редагувати поживну цінність продуктів [3].

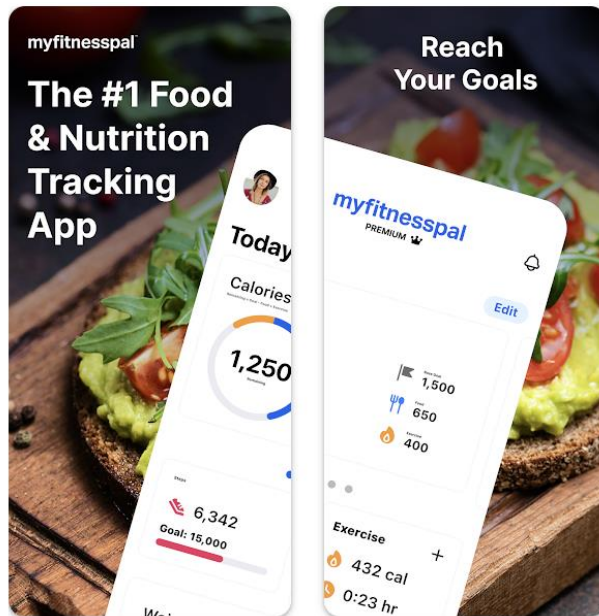


Рисунок 1.1 – Додаток MyFitnessPal

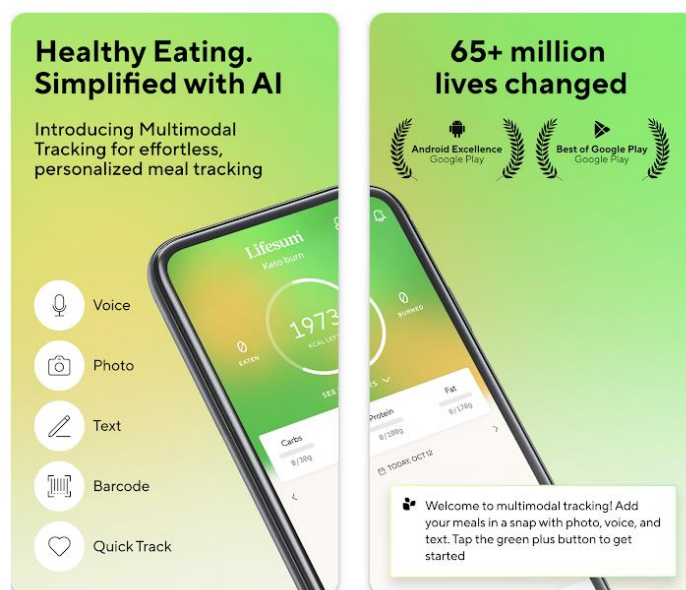


Рисунок 1.2 – Lifesum

1.1.3 FatSecret

FatSecret – додаток із фокусом на підрахунок калорій та соціальну взаємодію. Його особливості (рис.1.3):

- доступ до великої бази даних продуктів;
- форум для обміну досвідом з іншими користувачами;
- функція розпізнавання продуктів за фотографією;
- докладний звіт про спожиті калорії та фізичну активність;
- вбудований журнал для нотаток щодо харчування та тренувань.

Недоліки: менш зручний інтерфейс у порівнянні з конкурентами, недостатня деталізація макронутрієнтів. Також система розпізнавання продуктів часто видає неточні результати [4].



Рисунок 1.3 – Додаток FatSecret

1.1.4 Cronometer

Cronometer – додаток для тих, хто детально стежить за своїм раціоном і здоров'ям (рис.1.4). Він пропонує:

- точний підрахунок мікроелементів (вітамінів, мінералів);

- підтримку кетогенної, вегетаріанської та інших спеціалізованих дієт;
- розширений аналіз нутрієнтів;
- моніторинг рівня фізичної активності;
- інтеграцію з розумними вагами та трекерами активності.

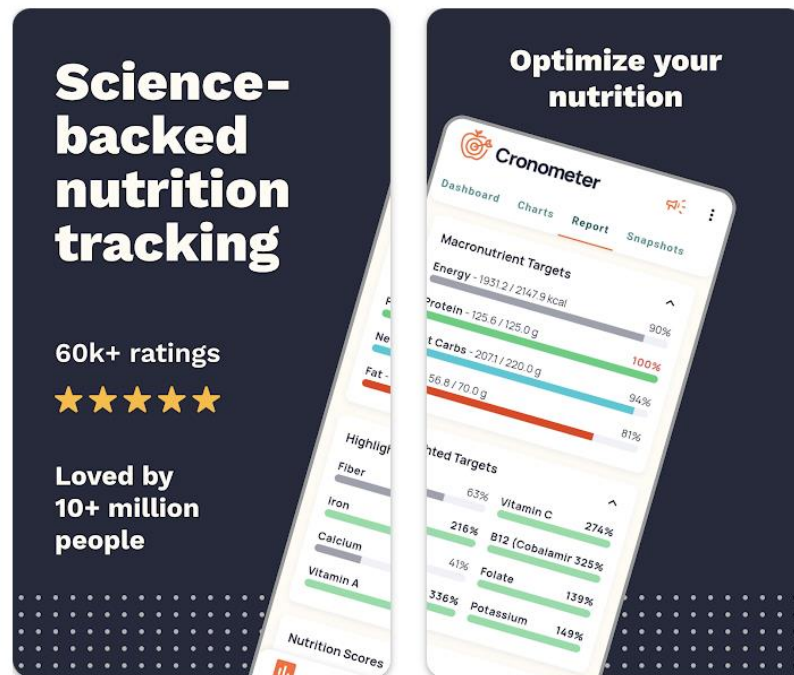


Рисунок 1.4 – Додаток Cronometer

Недоліки: складний інтерфейс для початківців, платний доступ до деяких функцій. Також додаток може вимагати багато часу для ручного введення продуктів через відсутність локалізованої бази.

1.1.5 YAZIO

YAZIO – ще один популярний додаток для контролю калорійності раціону, що пропонує (рис.1.5):

- персоналізовані плани харчування;
- підрахунок калорій і макроелементів;
- відстеження рівня водного балансу;
- підтримку періодичного голодування;
- візуалізацію досягнень у вигляді графіків.



Рисунок 1.5 – Додаток YAZIO

Недоліки: безкоштовна версія має значні обмеження, а платний доступ коштує дорожче, ніж у конкурентів[5].

Аналіз існуючих рішень показав, що більшість додатків пропонують схожий базовий функціонал – підрахунок калорій, моніторинг макронутрієнтів та інтеграцію з фітнес-трекерами. Проте кожен з них має свої особливості, які можуть впливати на вибір користувача. Деякі з додатків орієнтовані на соціальну взаємодію, інші – на персоналізовані дієтичні плани або точний підрахунок мікроелементів.

В рамках цієї роботи буде розроблено мобільний додаток, який поєднуватиме найкращі можливості існуючих рішень та враховуватиме їхні недоліки для покращення користувацького досвіду. Зокрема, планується створення інтуїтивного інтерфейсу, локалізованої бази продуктів та точного розрахунку енергетичної цінності страв.

На основі проведеного аналізу існуючих рішень та отриманих результатів можна сформулювати такі висновки:

- переважна більшість представлених на ринку рішень є дорогими у розробці та подальшій підтримці;

- найбільш зручними для користувачів виявилися ті рішення, які підтримуються на мобільних пристроях;
- розробка кросплатформенного застосунку, який не потребує доступу до Інтернету та здатний повноцінно виконувати всі необхідні функції, є складним завданням.

2 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Вибір програмного забезпечення

2.1.1 Опис мови програмування Java

Java — це об'єктно-орієнтована мова програмування (ООП), побудована на основі класів і спроектована таким чином, щоб мінімізувати залежність від конкретної реалізації (рис. 2.1). Завдяки своїй універсальності Java дозволяє розробникам створювати програми один раз і запускати їх будь-де (WORA). Це означає, що скомпільований код може виконуватися на будь-якій платформі, що підтримує Java, без необхідності його перекомпіляції [6].

Програми, написані на Java, зазвичай компілюються в байт-код, який може виконуватися на будь-якій віртуальній машині JVM, незалежно від базової комп'ютерної архітектури. Синтаксис Java подібний до C і C++, але має обмежені можливості програмування низького рівня в порівнянні з ними. Виконуване середовище Java підтримує динамічні можливості, такі як модифікація та розширення коду під час виконання, що зазвичай недоступне в інших скомпільованих мовах.

Станом на 2019 рік Java залишалася однією з найпопулярніших мов програмування за даними GitHub, особливо у сфері розробки клієнт-серверних веб-додатків. Загальна кількість розробників Java перевищувала 9 мільйонів .

Мову було створено Джеймсом Гослінгом у Sun Microsystems (згодом придбаною Oracle) та випущено в 1995 році як ключову технологію компанії.

Перші реалізації компіляторів, бібліотек класів і віртуальних машин Java були представлені Sun під власними ліцензіями. У травні 2007 року, відповідно до процесу Java Community Process, Sun відкрила більшу частину своєї технології Java під публічною ліцензією GNU.

Oracle підтримує власну віртуальну машину Java HotSpot, але офіційною довідковою реалізацією JVM є OpenJDK — безкоштовне ПЗ з відкритим кодом, яке використовується більшістю розробників і є стандартним для більшості дистрибутивів Linux.



Рисунок 2.1 - Логотип мови Java

На сьогодні актуальною версією є Java 16, а Java 11 має статус довгострокової підтримки. У 2019 році Oracle припинила публічні оновлення для Java 8 LTS у комерційних цілях, але продовжила надавати загальні оновлення для приватного використання. Інші постачальники продовжили випускати збірки OpenJDK 8 та 11 із підтримкою оновлень безпеки.

Oracle та інші компанії рекомендують видаляти застарілі версії Java через потенційні загрози безпеці. Оскільки Java 9, 10, 12, 13, 14 і 15 більше не підтримуються, користувачам радять оновлюватися до останньої версії (Java 16) або версії з LTS-підтримкою.

Проект Java був ініційований у червні 1991 року Джеймсом Гослінгом, Майком Шеріданом і Патріком Нотоном. Початково Java призначалася для інтерактивного телебачення, але виявилася надто складною для індустрії цифрового кабельного мовлення. Спочатку її називали "Oak" на честь дуба біля офісу Гослінга, потім проєкт отримав назву "Green", а згодом — "Java", на честь сорту індонезійської кави.

Гослінг розробив Java із синтаксисом у стилі C/C++, що робило її зручною для системних та прикладних програмістів. Перша публічна реалізація була представлена Sun Microsystems у 1996 році під назвою Java 1.0. Вона пропонувала концепцію "Write Once, Run Anywhere", що дозволяло запускати програми без перекомпіляції на різних платформах [7].

Java також забезпечувала високий рівень безпеки, дозволяючи обмежувати доступ до файлів і мережі. Основні веббраузери швидко почали підтримувати Java-аплети, що сприяло її популярності. Оригінальний компілятор Java 1.0 був модифікований Артуром ван Гоффом для суворої відповідності специфікації мови.

З випуском Java 2 з'явилися різні конфігурації для різних платформ. J2EE включала технології для корпоративних додатків, які працюють у серверному середовищі, J2ME була оптимізована для мобільних пристроїв, а J2SE використовувалася для настільних застосунків. У рамках маркетингової стратегії Sun перейменувала J2 версії у Java SE, Java ME та Java EE.

Sun Microsystems зверталася до ISO/IEC JTC 1 та Ecma International з пропозицією зробити Java офіційним стандартом, але згодом відмовилася від цього. Java залишається де-факто стандартом, контрольованим через Java Community Process. Sun надавала безкоштовний доступ до більшості реалізацій Java, заробляючи на продажу ліцензій для корпоративних продуктів, таких як Java Enterprise System.

Sun випустила значну частину своєї JVM як вільне програмне забезпечення за ліцензією GNU GPL. 8 травня 2007 року компанія завершила цей процес, відкривши весь основний код JVM, за винятком деяких компонентів, права на які

належали третім сторонам.

Віце-президент Sun, Річ Грін, наголошував, що компанія прагне відігравати роль "євангеліста" Java. Після придбання Sun Microsystems компанією Oracle у 2009–2010 роках, Oracle заявила про намір сприяти відкритості й розвитку спільноти Java. Однак це не завадило компанії подати судовий позов проти Google через використання Java в Android SDK.

2 квітня 2010 року Джеймс Гослінг покинув Oracle.

У січні 2016 року Oracle оголосила, що в JDK 9 буде припинено підтримку плагіна Java у браузерях.

Java використовується практично всюди: від ігрових консолей до наукових суперкомп'ютерів, від ноутбуків до центрів обробки даних.

Однією з ключових особливостей Java є її портативність: програми, створені для цієї платформи, працюють однаково на будь-якому апаратному забезпеченні та операційній системі, що підтримує Java. Це досягається шляхом компіляції вихідного коду в байт-код Java, який не залежить від конкретної архітектури. Байт-код Java виконується у віртуальній машині, спеціально розроблений для апаратної платформи хоста.

Ось перефразований текст зі збереженням змісту та структури:

Стандартний набір бібліотек забезпечує універсальний спосіб доступу до функцій хоста, зокрема для роботи з мережами, створення потоків і графіки. Перенесення коду можна спростити завдяки використанню універсального байт-коду. Однак через значні витрати на його інтерпретацію програми, що виконуються у такий спосіб, зазвичай працюють повільніше, ніж власні виконувані файли. Щоб вирішити цю проблему, ще на ранніх етапах розробки були впроваджені JIT-компілятори, які трансформують байт-код у машинний код під час виконання.

Java не залежить від платформи, але адаптується до конкретного середовища шляхом завантаження відповідної віртуальної машини JVM, яка інтерпретує байт-код у машинні команди.

Більшість програм, написаних на Java, працюють повільніше та споживають

більше пам'яті, ніж, наприклад, C++-програми. Незважаючи на це, швидкодія значно зросла завдяки впровадженню динамічної компіляції та вдосконалень у віртуальній машині. Зокрема, оптимізації JVM, такі як HotSpot (що стала стандартною JVM від Sun у 2000 році), а також додавання нових мовних конструкцій, зокрема класу `StringBuilder`, допомогли покращити продуктивність. Значним досягненням стало включення пакета `java.util.concurrent` у Java 1.6, який містить структури даних для багатопотокової обробки.

Деякі апаратні платформи мають пряму підтримку Java. Наприклад, існують мікроконтролери, що можуть працювати з байт-кодом Java на рівні заліза, без програмної JVM. Крім того, деякі процесори на основі ARM підтримували запуск байт-коду через технологію Jazelle, хоча в нових реалізаціях від неї відмовилися.

Java використовує автоматичний збирач сміття для управління пам'яттю під час життєвого циклу об'єктів. Розробник створює об'єкти, а звільнення зайнятої ними пам'яті виконується автоматично, коли об'єкти більше не використовуються. Якщо на об'єкт більше немає посилань, його пам'ять може бути повернена системі. Проте, якщо код містить посилання на непотрібні об'єкти (наприклад, у контейнерах, які продовжують використовуватись), може виникнути витік пам'яті. Спроба викликати методи об'єкта, що вже не існує, призведе до `NullPointerException`.

Основною ідеєю моделі автоматичного управління пам'яттю у Java є те, що розробникам більше не потрібно виконувати ручне керування пам'яттю, як у C або C++. У цих мовах пам'ять виділяється вручну та звільняється розробником, що може призвести до витоків або нестабільної роботи через доступ до вже звільнених областей пам'яті. Хоча інші мови містять інструменти для вирішення цих проблем, вони ускладнюють розробку. Однак навіть у Java збирач сміття не може запобігти логічним витокам пам'яті, коли об'єкт більше не використовується, але на нього все ще є посилання.

Збирання сміття може відбуватися в будь-який момент. В ідеалі це відбувається, коли програма неактивна, однак найчастіше процес запускається тоді, коли у купі закінчується вільна пам'ять, що може спричинити короткочасну

зупинку програми. У Java немає можливості явного керування пам'яттю.

На відміну від C і C++, Java не підтримує арифметичні операції з покажчиками, що унеможлиблює прямий доступ до адрес об'єктів. Як і в C++, змінні примітивних типів можуть зберігатися у стеку або в полях об'єктів, а не у купі, що було зроблено навмисно для підвищення продуктивності.

Java підтримує кілька типів збирачів сміття. Починаючи з Java 9, стандартним є Garbage First Garbage Collector, хоча доступні й інші варіанти для керування пам'яттю. У Java 8 за замовчуванням використовувався паралельний збирач сміття. Водночас автоматичне керування пам'яттю не звільняє розробників від необхідності правильно закривати інші ресурси, такі як з'єднання з базами даних, файлові дескриптори або мережеві підключення [8].

Синтаксис Java схожий на C++ і C. Однак, на відміну від C++, що підтримує як структурне, так і об'єктно-орієнтоване програмування, Java є майже повністю об'єктно-орієнтованою мовою. Всі елементи програми знаходяться всередині класів, а всі дані представлені у вигляді об'єктів, за винятком примітивних типів, таких як цілі числа або числа з плаваючою комою.

Ось перефразований текст зі збереженням змісту та структури:

Усі вихідні файли повинні бути названі відповідно до загальнодоступного класу, який вони містять (рис. 2.2), із додаванням суфікса .java, наприклад, HelloWorldApp.java. Перед виконанням цей файл необхідно скомпілювати в байт-код за допомогою компілятора Java, у результаті чого буде створений файл із розширенням .class (у цьому випадку HelloWorldApp.class). Лише після цього програму можна запустити.

```
1 public class MyFirstJavaProgram {  
2     /*  
3      * Це моя перша java програма Вона буде друкувати 'Hello World'  
4      */  
5     public static void main(String[] args) {  
6         System.out.println("Hello World"); // друкує 'Hello World'  
7     }  
8 }
```

Рисунок 2. 2 Приклад програми на Java

Вихідний файл Java може містити лише один загальнодоступний клас, але допускається наявність кількох класів із модифікатором непублічного доступу, а також будь-якої кількості загальнодоступних вкладених класів. Якщо вихідний файл містить декілька класів, один із них має бути оголошений загальнодоступним (позначається ключовим словом `public`), а ім'я файлу повинно збігатися з ім'ям цього класу. Клас, який не є загальнодоступним, може знаходитися в будь-якому `.java` файлі. Під час компіляції буде створений `.class` файл для кожного класу, визначеного у вихідному файлі, причому його ім'я відповідатиме імені класу з доданим `.class`. Анонімні класи отримують автоматично згенеровані імена у форматі `<ім'я класу, що містить>$<номер>`.

Ключове слово `public` означає, що метод може бути викликаний іншими класами або що цей клас доступний для використання поза межами своєї ієрархії. Ієрархія класів також відображається у структурі каталогів, де розташовані `.java` файли. Це так званий модифікатор рівня доступу. Інші модифікатори включають `private` (обмежує доступ методами того ж класу) та `protected` (дозволяє доступ із того ж пакета). Якщо код намагається звернутися до `private` або `protected` методів без дозволу, JVM видасть виняток `SecurityException`.

Ключове слово `static` перед методом означає, що цей метод є статичним, тобто належить класу, а не його конкретному екземпляру. Викликати статичні методи можна без створення об'єкта класу. Водночас статичні методи не можуть отримувати доступ до нестатичних членів класу. Якщо метод не оголошений `static`, він є екземплярним і вимагає створення об'єкта класу для використання.

Ключове слово `void` вказує, що метод `main` не повертає жодного значення. Якщо програма має завершитися з певним кодом помилки, необхідно явно викликати `System.exit()`.

Назва методу `main` не є зарезервованим словом у Java, а лише стандартним іменем для точки входу в програму. Програми, що виконуються у спеціалізованих середовищах (наприклад, аплети або `Enterprise JavaBeans`), можуть не мати методу `main()`. Програма може містити кілька класів із методами `main`, тому JVM потрібно вказати, з якого саме класу починати виконання.

Основний метод отримує на вхід масив об'єктів String. Зазвичай його називають args, хоча можна використати будь-яку допустиму назву змінної. Починаючи з Java 5, main також може використовувати змінну кількість аргументів у вигляді `public static void main(String... args)`, що дозволяє викликати його з довільною кількістю аргументів. По суті, ця форма є еквівалентною масиву `String[] args`, але пропонує більш зручний синтаксис.

Запуск Java-програми відбувається шляхом завантаження вказаного класу (визначеного в командному рядку або як атрибут у JAR-файлі) і виконання його методу `public static void main(String[])`. Для самостійних програм цей метод обов'язковий. Масив args містить аргументи, передані через командний рядок.

Вивід на екран здійснюється за допомогою стандартної бібліотеки Java: клас System містить загальнодоступне статичне поле out, яке є екземпляром класу `PrintStream`. Він надає різні методи для друку інформації, зокрема `println(String)`, що додає новий рядок до переданого тексту.

З огляду на всі ці особливості, мову Java було обрано для реалізації цього проєкту та виконання поставлених завдань. Її функціональність найбільш підходить для цієї розробки завдяки зручній реалізації основних принципів об'єктно-орієнтованого програмування.

2.1.2 Опис мови програмування Kotlin

Kotlin — це сучасна, статично типізована мова програмування, яка була створена компанією JetBrains і вперше представлена у 2011 році. З моменту свого створення вона набула великої популярності, особливо в сфері розробки мобільних додатків під операційну систему Android. У 2017 році компанія Google офіційно оголосила про повну підтримку Kotlin як мови для розробки Android-додатків, а з 2019 року рекомендує її як основну мову для нових проєктів [9].

Kotlin розроблялася як повністю сумісна з Java мова, що означає, що вона може працювати з існуючим Java-кодом без будь-яких обмежень. Це дуже важливо для розробників, які хочуть поступово переходити з Java на Kotlin, не переписуючи повністю свої проєкти. Бібліотеки, фреймворки та навіть Android SDK, написані на Java, можуть використовуватись у Kotlin без проблем.

Однією з найбільших переваг Kotlin є її лаконічність. Завдяки компактному синтаксису, Kotlin дозволяє досягати того ж самого результату з меншим обсягом коду порівняно з Java. Наприклад, такі задачі як ініціалізація об'єктів, створення геттерів/сеттерів, обробка подій та багато інших виконуються набагато простіше.

Kotlin має вбудовану систему захисту від однієї з найбільш поширених помилок у Java — `NullPointerException`. У Kotlin змінні за замовчуванням не можуть містити `null`, і якщо розробнику потрібно допустити `null`, він повинен явно це зазначити. Такий підхід знижує кількість потенційних помилок та підвищує стабільність додатків.

Kotlin підтримує багато можливостей функціонального програмування, таких як лямбда-вирази, вищі функції, імутабельність та інші. Це дозволяє писати більш модульний, зручний для тестування та масштабований код.

Kotlin має приємний і логічний синтаксис, який наближений до сучасних мов програмування, таких як Swift або Python. Це дозволяє новачкам швидко освоїти мову, а досвідченим програмістам — писати чистий і зрозумілий код.

Для розробки сучасних Android-додатків важливо вміти ефективно працювати з асинхронними задачами, наприклад, при взаємодії з мережею або базою даних. Kotlin надає вбудовану підтримку корутин (`coroutines`) — легких потоків, які дозволяють писати асинхронний код так само просто, як і звичайний, без складних зворотних викликів (`callback'ів`) або ручного керування потоками.

Android Studio — офіційне середовище розробки для Android — має повну інтеграцію з Kotlin. Усі функції IDE, такі як автозаповнення коду, рефакторинг, перевірка помилок у режимі реального часу, підтримуються на рівні Java. До того ж Android Studio дозволяє автоматично конвертувати Java-код у Kotlin, що спрощує міграцію та вивчення мови.

Kotlin є основною мовою для бібліотеки Jetpack Compose — сучасного фреймворку для побудови UI на Android. Compose дозволяє будувати інтерфейси декларативно, використовуючи переваги Kotlin, зокрема функції вищого порядку та лямбда-вирази. Це дає можливість створювати адаптивні та анімовані інтерфейси швидше та з меншим кодом.

Недоліки Kotlin:

- вища вартість компіляції — на перших етапах Kotlin може компілюватися трохи повільніше, ніж Java;
- менш розвинена документація у порівнянні з Java — хоча це поступово змінюється, деякі розробники можуть відчувати брак інформації при пошуку специфічних рішень;
- крутіший вхідний поріг для новачків — особливо для тих, хто не знайомий із функціональним стилем програмування.

Мова програмування Kotlin є одним із найкращих інструментів для створення Android-додатків на сьогоднішній день. Вона поєднує в собі сучасний синтаксис, високу продуктивність, потужні інструменти для обробки помилок і чудову підтримку асинхронності. Kotlin дозволяє розробникам писати менше, але якіснішого коду, що сприяє швидкому створенню стабільних і ефективних мобільних застосунків. Саме тому Kotlin активно використовується у провідних ІТ-компаніях, а його знання є обов'язковим для більшості вакансій Android-розробників.

Таким чином, вибір Kotlin для розробки додатків під Android є виправданим, особливо для тих, хто прагне поєднати надійність, сучасність та ефективність у своїх проєктах.

2.1.3 Опис мови програмування Dart

Мова програмування Dart є сучасною, об'єктно-орієнтованою мовою, розробленою компанією Google у 2011 році. Найбільшого поширення вона набула завдяки фреймворку Flutter, який використовується для створення кросплатформених мобільних застосунків. Dart забезпечує розробникам можливість створювати зручні, швидкі та візуально привабливі додатки, які однаково добре працюють як на Android, так і на iOS [10].

Dart має багато переваг, які роблять його привабливим інструментом для розробки мобільних додатків. Насамперед, ця мова відзначається тісною інтеграцією з Flutter — фреймворком, що дозволяє створювати інтерфейси користувача будь-якої складності з єдиної кодової бази. Завдяки цьому значно

спрощується розробка додатків для кількох платформ одночасно, що дозволяє скоротити витрати часу та ресурсів.

Однією з ключових переваг Dart є його висока продуктивність. За допомогою компіляції AOT (Ahead-of-Time) додатки, написані на цій мові, можуть працювати з майже нативною швидкістю. У той же час під час розробки використовується Just-in-Time- компіляція, яка забезпечує так звану функцію «гарячого перезапуску» (hot reload). Це означає, що будь-які зміни в коді миттєво відображаються в інтерфейсі без потреби повного перезапуску програми, що значно підвищує ефективність і зручність розробки.

Синтаксис Dart є зрозумілим і логічним, особливо для розробників, які мають досвід роботи з такими мовами, як Java або JavaScript. Це полегшує його вивчення та впровадження в реальні проекти. Dart також підтримує строгий об'єктно-орієнтований підхід і статичну типізацію, що сприяє написанню структурованого, масштабованого та безпечного коду.

Крім того, екосистема Dart постійно розвивається. Існує велика кількість бібліотек і пакетів, доступних через офіційний репозиторій pub.dev, що дає можливість легко додавати у проекти необхідну функціональність: підключення до баз даних, роботу з API, інтеграцію з Firebase тощо.

Однак варто зазначити й деякі обмеження. Наприклад, поза межами Flutter Dart не набув великої популярності, що дещо обмежує його використання у веброзробці чи десктопних застосунках. Також у порівнянні з Java чи Kotlin екосистема Dart є молодшою, тому деякі вузькоспеціалізовані бібліотеки можуть бути недоступними або менш стабільними.

Незважаючи на це, Dart у поєднанні з Flutter залишається одним із найкращих рішень для швидкої, сучасної та ефективної розробки мобільних додатків для Android. Завдяки можливості створення адаптивного інтерфейсу, високій продуктивності та зручності розробки, Dart дозволяє створювати повноцінні мобільні рішення, що відповідають актуальним вимогам користувачів і бізнесу. Якщо метою є створення універсального додатку з єдиною базою коду для Android і, за потреби, iOS — Dart є доцільним і стратегічно вигідним вибором.

2.1.4 Опис мови програмування C++

Мова програмування C++ є однією з найстаріших і найпотужніших мов, яка активно використовується в розробці програмного забезпечення, зокрема і для мобільної платформи Android. Хоча ця мова не є основною для створення Android-додатків, як-от Java чи Kotlin, вона відіграє важливу роль у розробці високопродуктивних, системно наближених компонентів, а також додатків, що потребують складних обчислень або роботи з мультимедійним контентом [11].

C++ активно застосовується для створення Android-додатків через NDK — інструментарій, що дозволяє писати частину застосунку мовами C або C++ та компілювати цей код у нативну бібліотеку. Такий підхід використовується тоді, коли необхідно забезпечити максимальну продуктивність, мінімальне споживання ресурсів або при створенні додатків, що взаємодіють безпосередньо з апаратним забезпеченням пристрою.

Однією з основних причин використання C++ для Android є його висока швидкодія. Ця мова забезпечує доступ до низькорівневого управління пам'яттю, що дозволяє розробникам оптимізувати використання ресурсів і створювати надзвичайно ефективний код. Саме тому C++ часто застосовують у створенні ігор, графічних движків, аудіо- та відеообробки, систем машинного навчання, а також криптографічних алгоритмів.

Велика кількість переносимого коду також є перевагою C++. Розробники можуть використовувати один і той самий кодовий базис для кількох платформ, таких як Android, Windows, Linux чи iOS, що суттєво знижує витрати на розробку і супровід застосунків.

Ще однією перевагою є можливість інтеграції з іншими мовами, насамперед із Java, через JNI. Це дає змогу розробникам поєднувати нативні модулі C++ із традиційними Android-компонентами, написаними Java або Kotlin.

Однак слід також враховувати складність роботи з C++. У порівнянні з Java чи Kotlin, ця мова вимагає більш глибоких знань і досвіду, особливо щодо управління пам'яттю, вказівників і обробки помилок. Розробка на C++ зазвичай є більш трудомісткою, а налагодження коду — складнішим. Крім того, розробка

Android-додатків виключно на C++ ускладнює використання стандартних Android API, оскільки більшість із них розраховані на Java або Kotlin.

Таким чином, C++ не є основною мовою для повноцінної розробки Android-застосунків, але є незамінною у випадках, коли необхідно реалізувати високопродуктивні обчислення, обробку великих обсягів даних або працювати з ресурсомісткими мультимедійними задачами. Ця мова найкраще підходить для створення нативних бібліотек, які доповнюють основну логіку додатку, написаного Java або Kotlin.

У підсумку, C++ є ефективним вибором у спеціалізованих сценаріях розробки Android-додатків, зокрема в ігровій індустрії, обробці відео, 3D-графіки або будь-яких інших завданнях, де критичне значення мають продуктивність та оптимізація ресурсів.

Таблиця 2.1 – Порівняльний аналіз мов програмування для ОС Android

Критерій	Java	Kotlin	Dart (Flutter)	C++ (NDK)
Офіційна підтримка Google	Так	Так	Так (через Flutter SDK)	Обмежена (через NDK)
Простота вивчення	Середня	Вища (для тих, хто знає Java)	Середня (новий підхід)	Висока складність
Стабільність та сумісність	Висока (сумісна з більшістю пристроїв)	Висока	Обмежено сумісна з Android SDK	Висока (але складна інтеграція)
Продуктивність	Висока	Висока	Середня (через абстракції Flutter)	Дуже висока
Наявність бібліотек	Дуже велика кількість	Сумісна з Java-бібліотеками	Обмежена (тільки Flutter-пакети)	Обмежена для Android UI
Інструменти розробки	Android Studio, Gradle	Android Studio, Gradle	Flutter SDK, Android Studio	Специфічне середовище (NDK, CMake)

Продовження таблиці 2.1

Критерій	Java	Kotlin	Dart (Flutter)	C++ (NDK)
Підтримка старих пристроїв	Широка	Добра, але трохи обмежена	Потрібна підтримка Flutter Runtime	Можлива, але складна реалізація
Спільнота та підтримка	Дуже велика	Швидко зростає	Менша спільнота	Вузька спільнота
Гнучкість та контроль	Висока	Висока	Менше контролю над системою	Максимальний контроль

Після аналізу основних мов програмування, які використовуються для розробки мобільних додатків на платформі Android, можна зробити висновок, що хоча на сьогодні існує кілька сучасних та зручних рішень, таких як Kotlin чи Dart (Flutter), Java залишається актуальною, потужною та виправданою для створення стабільного, надійного Android-додатку, зокрема — для системи підрахунку калорій.

По-перше, Java є однією з основних офіційних мов Android-розробки, яка підтримується Google з моменту створення Android. Це забезпечує широку документацію, стабільну підтримку та численні приклади коду, що значно спрощує процес навчання, розробки, тестування та підтримки додатку. Завдяки цьому Java підходить як для початківців, так і для досвідчених розробників.

По-друге, Java дозволяє створювати максимально сумісні додатки, які працюватимуть на більшості Android-пристроїв, включаючи старіші версії ОС. Це особливо важливо для додатку з практичною ціллю, орієнтованого на широку аудиторію користувачів, зокрема на тих, хто використовує недорогі смартфони з попередніми версіями Android.

По-третє, Java має високу продуктивність та стабільність при роботі з нативними функціями Android, такими як керування життєвим циклом активностей, робота з базами даних, системними ресурсами (наприклад, датчиками або камерами) та інтерфейсами. У випадку з додатком для підрахунку калорій, де важлива точність обчислень, робота з локальним сховищем, взаємодія

з користувачем у реальному часі та можлива інтеграція зі сторонніми API (наприклад, для отримання інформації про продукти), Java надає розробнику повний контроль над логікою та архітектурою додатку.

Крім того, існуючі фреймворки та бібліотеки для Java — зокрема Room, Retrofit, Glide, Firebase SDK — повністю підтримуються в Android-екосистемі, що дозволяє пришвидшити процес розробки та покращити функціональність без значного ускладнення коду.

Не менш важливим є те, що велика спільнота розробників Java означає легкий доступ до підтримки, консультацій та готових рішень. Це значно знижує ризики на етапі впровадження проєкту, полегшує тестування та обслуговування додатку в майбутньому.

Отже, незважаючи на поширення нових мов, таких як Kotlin та Dart, вибір Java для розробки мобільного додатку для підрахунку калорій є цілком виправданим і стратегічно обґрунтованим. Java забезпечує високу стабільність, сумісність, доступ до великої кількості ресурсів і бібліотек, а також зручність підтримки, що є критично важливими факторами для ефективної реалізації функціонального та зручного для користувача мобільного додатку.

2.2 Вибір середовища розробки

2.2.1 Опис Android Studio

Android Studio — це офіційне інтегроване середовище розробки (IDE) для операційної системи Android, створене Google на основі програмного забезпечення IntelliJ IDEA від компанії JetBrains, з метою спрощення процесу розробки Android-додатків (рис. 2.3). Це середовище доступне для завантаження та встановлення на операційних системах Linux, macOS і Windows. Android Studio прийшло на зміну Eclipse з плагіном Android Development Tools, ставши основним інструментом для створення власних мобільних додатків під Android [12].

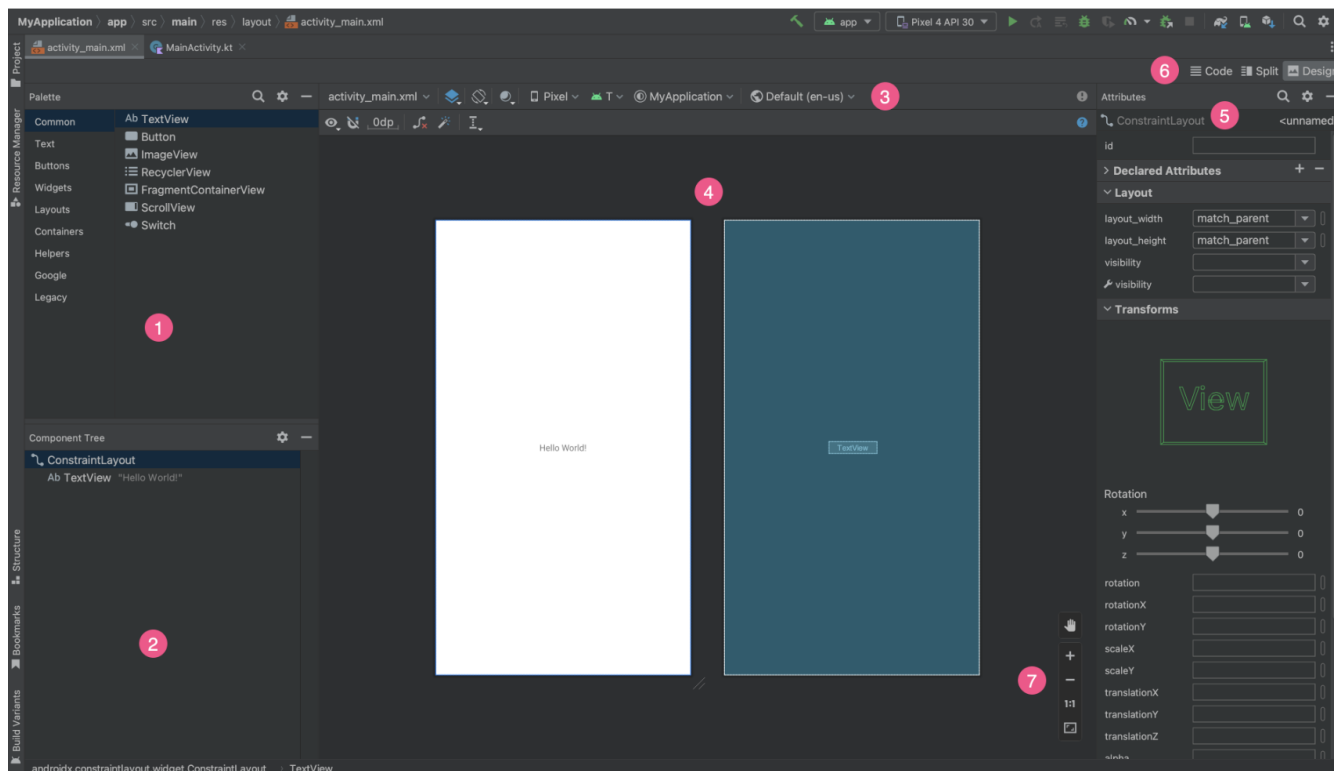


Рисунок 2.3 - Інтерфейс Android Studio

16 травня 2013 року на конференції Google I/O було представлено Android Studio. На той час середовище перебувало на етапі прототипу, і перша версія 0.1 була випущена у травні 2013 року. Згодом вийшла бета-версія, а вже в грудні 2014 року була опублікована перша стабільна версія під номером 1.0.

З часом Kotlin став офіційною мовою для розробки Android-додатків, замінивши Java як основну мову програмування. Проте Java і надалі підтримується, як і C++.

Серед функціональних можливостей останніх стабільних версій Android Studio:

- підтримка системи збірки на основі Gradle;
- рефакторинг та автоматичні виправлення, спеціально адаптовані для Android;
- інструмент Lint для виявлення проблем, пов'язаних із зручністю використання, сумісністю версій та іншими недоліками;
- інтеграція з інструментом ProGuard для обфускації та оптимізації коду;

- готові шаблони для створення стандартних компонентів Android-додатків;
- візуальний редактор макетів з функцією drag-and-drop, що дозволяє переглядати інтерфейс на різних типах пристроїв;
- підтримка розробки додатків для Android Wear;
- можливість інтеграції з Google Cloud Platform, зокрема Firebase Cloud Messaging та Google App Engine;
- емулятор для тестування, запуску та налагодження додатків.

Android Studio підтримує всі мови, доступні в IntelliJ IDEA, включаючи Java, C++ та Kotlin. Останні версії повноцінно працюють з Kotlin, підтримують усі функції Java 7 і частково — Java 8. Деякі зовнішні проєкти можуть використовувати функціональність Java 9, а згідно з документацією IntelliJ, заявлено підтримку й новіших версій Java, таких як Java 12. Водночас підтримка Java 8 і вище не є повною — деякі функції реалізовані частково.

Для роботи емулятора Android, крім основних системних вимог Android Studio, необхідно мати:

- інструмент SDK Tools версії 26.1.1 або новішої;
- 64-розрядний процесор;
- на Windows — процесор з підтримкою UG (Unrestricted Guest);
- Intel Hardware Accelerated Execution Manager (HAXM) для прискореної віртуалізації.

Апаратне прискорення має додаткові вимоги:

- на Windows або Linux потрібен процесор Intel;
- на Linux підтримується також процесор AMD з увімкненою віртуалізацією;
- на Windows — процесор AMD підтримується лише в нових версіях ОС.

Для тестування систем на Android 8.1 (API рівень 27) або вище, веб-камера, підключена до ПК, повинна підтримувати запис відео з роздільною здатністю не нижче 720р.

2.2.2 Опис IntelliJ IDEA як середовище розробки

IntelliJ IDEA — це сучасне інтегроване середовище розробки (IDE), розроблене компанією JetBrains, яке підтримує широкий спектр мов програмування, серед яких Java та Kotlin, що є основними для розробки Android-додатків. Android Studio, офіційне середовище розробки від Google, базується саме на IntelliJ IDEA, проте має додаткові плагіни та інструменти, спеціально адаптовані для розробки під платформу Android [13].

Використання IntelliJ IDEA для створення Android-додатків можливе завдяки підтримці основних компонентів Android SDK, системи збірки Gradle, а також налаштовуваним плагінам, що забезпечують роботу з Android-проектами. IntelliJ IDEA надає розробнику розвинений редактор коду з можливістю автодоповнення, рефакторингу, навігації та статичного аналізу коду, що суттєво підвищує ефективність розробки.

Переваги IntelliJ IDEA для Android-розробки. Серед ключових переваг IntelliJ IDEA можна виділити:

- потужний інструментарій розробки, що забезпечує зручне написання, редагування та підтримку коду на Java і Kotlin;
- підтримка Gradle як системи автоматизації збірки, що дозволяє гнучко налаштовувати процес компіляції та розгортання додатку;
- можливість інтеграції з Android SDK, що дає змогу запускати, відлагоджувати і тестувати додатки;
- велика кількість додаткових плагінів для розширення функціоналу IDE.

Незважаючи на потужний функціонал, IntelliJ IDEA має певні обмеження у порівнянні з Android Studio, зокрема:

- відсутність вбудованих візуальних редакторів інтерфейсу користувача (Layout Editor), що ускладнює розробку UI;
- необхідність ручного налаштування Android SDK та емуляторів, що підвищує складність початкового етапу роботи;

- відсутність інтегрованих інструментів для профілювання продуктивності та глибокого аналізу Android-додатків.

Використання IntelliJ IDEA доцільне у випадках, коли розробник вже має досвід роботи з цим IDE для інших видів програмування і бажає використовувати єдине середовище для різних проєктів. Також IntelliJ IDEA може бути оптимальним вибором для розробників, які прагнуть гнучкості у налаштуванні робочого процесу та підтримці кількох мов програмування в межах одного проєкту.

Таким чином, IntelliJ IDEA є потужним та універсальним інструментом для розробки Android-додатків, однак для повноцінного і зручного розроблення мобільних додатків рекомендується використовувати офіційне середовище Android Studio.

2.2.3 Опис Eclipse як середовище розробки

Eclipse — це потужне інтегроване середовище розробки з відкритим кодом, яке спочатку було розроблене для програмування на Java. На початку розвитку Android-платформи Eclipse з плагіном Android Development Tools (ADT) був основним IDE для розробки мобільних додатків під Android. Завдяки широкому функціоналу та великій спільноті розробників, Eclipse надавав розробникам інструменти для створення, компіляції, відлагодження і тестування Android-додатків [14].

Для створення Android-додатків в Eclipse необхідно було встановити плагін Android Development Tools, який інтегрував Android SDK та надавав спеціалізовані інструменти для роботи з мобільними додатками. ADT забезпечував такі функції як:

- розробка та редагування Java-коду;
- візуальний редактор макетів інтерфейсу користувача;
- управління збіркою проєктів;
- запуск і відлагодження додатків на емуляторах або реальних пристроях.

Переваги Eclipse для Android-розробки:

- відкритий код і безкоштовність — Eclipse доступний усім і має велику кількість плагінів;
- гнучкість і розширюваність — середовище підтримує різноманітні мови і технології, що дозволяє використовувати його для багатьох типів розробки;
- зручний візуальний редактор для створення інтерфейсів користувача.

Недоліки та обмеження Eclipse у сучасній Android-розробці. З виходом Android Studio, яке базується на IntelliJ IDEA та пропонує більш глибоку інтеграцію з Android SDK, Eclipse поступово втратило актуальність у сфері мобільної розробки. Основні недоліки Eclipse:

- відсутність регулярних оновлень та підтримки плагіну ADT від Google;
- менша інтеграція з сучасними інструментами Android, такими як Gradle;
- складнощі з налаштуванням і менш інтуїтивний інтерфейс у порівнянні з Android Studio;
- відсутність підтримки нових можливостей Android-платформи та мовних фіч Kotlin.

Eclipse із плагіном ADT був популярним середовищем для розробки Android-додатків на ранніх етапах розвитку платформи. Проте на сьогодні його використання не рекомендується через обмежену підтримку та відсутність актуальних інструментів (табл..2.2).

Таблиця 2.2 – Порівняльні характеристики середовищ розробки для системи

Критерій	Android Studio	IntelliJ IDEA	Eclipse + ADT
Офіційна підтримка Google	Так	Ні	Ні
Мова програмування	Kotlin, Java, C++	Kotlin, Java	Java
Інтегровані Android-інструменти	Повна (UI редактор емулятор, профілювання,	Обмежена (треба додатково налаштовувати)	Часткова (ADT плагін)

Продовження таблиці 2.2

Критерій	Android Studio	IntelliJ IDEA	Eclipse + ADT
Підтримка кросплатформеності	Ні	Ні	Ні
Продуктивність IDE	Високі вимоги до ресурсів	Менш вимогливе	Помірні
Зручність UI-дизайнера	Вбудований візуальний редактор	Відсутній	Візуальний редактор (ADT)
Автоматизація збірки	Gradle	Gradle	Ant / Gradle (обмежено)
Підтримка Kotlin	Повна	Повна	Ні
Спільнота та ресурси	Дуже велика	Велика	Зменшується
Простота початку роботи	Відносно складна (потрібен потужний ПК)	Середня	Складна (потрібне налаштування)
Підтримка нативних API Android	Повна	Повна	Часткова

Сучасна розробка Android-додатків значною мірою орієнтована на Android Studio, що забезпечує більш зручний і функціональний інструментарій.

2.4 Вибір бази даних

2.4.1 SQLite

SQLite — це вбудована реляційна база даних, яка постачається разом з Android. Вона є найпоширенішим вибором для зберігання локальних структурованих даних у мобільних додатках. SQLite дозволяє створювати таблиці, виконувати SQL-запити, індексувати дані та підтримує транзакції [15].

Переваги:

- не потребує додаткового серверу, працює безпосередньо на пристрої;
- легка і швидка, ідеально підходить для локального зберігання даних;
- підтримує стандартні SQL-запити.

Недоліки:

- порівняно з сучасними базами має обмежений функціонал;
- робота із запитамі іноді складна для початківців.

2.4.2 Room (Android Jetpack)

Room — це бібліотека, яка є абстракцією над SQLite і входить до складу Android Jetpack. Вона значно спрощує роботу з базою даних, дозволяючи розробнику працювати з об'єктами, а не з сирими SQL-запитами. Room забезпечує типобезпечний доступ до даних через DAO (Data Access Objects), підтримує асинхронні операції, інтегрується з LiveData і Kotlin Coroutines [16].

Переваги:

- спрощує розробку, автоматично генерує SQL-код;
- підтримує асинхронність і реактивне програмування;
- полегшує керування міграціями бази даних.

Недоліки:

- потребує додаткової обробки для складних запитів;
- додаткова абстракція може трохи впливати на продуктивність.

2.4.3 Firebase Realtime Database

Firebase Realtime Database — це хмарна NoSQL база даних від Google, орієнтована на зберігання і синхронізацію даних у реальному часі. Вона дозволяє створювати додатки з функціоналом миттєвого оновлення інформації на всіх підключених пристроях. Firebase також автоматично підтримує офлайн-режим і синхронізує дані при відновленні інтернет-з'єднання [17].

Переваги:

- синхронізація даних у реальному часі між користувачами;
- підтримка офлайн-режиму;
- легка інтеграція з іншими сервісами Firebase (аутентифікація, аналітика).

Недоліки:

- обмеження на складні запити.

- дані зберігаються у форматі JSON, що не завжди зручно для реляційних структур.
- залежність від інтернет-з'єднання для повної функціональності.

Таблиця 2.3 – Порівняльна характеристика баз даних для Android

Критерій	SQLite	Room	Firestore Database
Тип бази даних	Локальна реляційна	Абстракція над SQLite	Хмарна NoSQL
Платформа	Android (локально на пристрої)	Android (локально на пристрої)	Хмарна (підключення до інтернету)
Модель даних	Таблиці, реляційна	Таблиці, реляційна	JSON-подібна (дерево ключ-значення)
Підтримка SQL	Повна	Повна (через генерацію коду)	Обмежена
Простота використання	Середня	Вища, завдяки абстракції	Висока (API простий для інтеграції)
Асинхронність та реактивність	Потрібна власна реалізація	Вбудована (LiveData, Coroutines)	Вбудована
Підтримка офлайн-режиму	Так	Так	Автоматично
Масштабованість	Локальна	Локальна	Висока, завдяки хмарній архітектурі
Синхронізація даних між пристроями	Відсутня	Відсутня	Так
Підтримка міграцій	Власна реалізація	Вбудована	Немає
Інтеграція з іншими сервісами	Обмежена	Обмежена	Висока (Firebase Authentication, Analytics)
Підходить для	Простих і середніх за обсягом локальних даних	Складніших проєктів з Kotlin/Java, зручний доступ до бази	Додатків із потребою реального часу і синхронізації
Вимоги до ресурсів	Низькі	Низькі	Залежить від мережі

2.3 Операційна система Android

Android — це мобільна операційна система, побудована на модифікованому ядрі Linux та іншому програмному забезпеченні з відкритим кодом, яка в основному призначена для сенсорних пристроїв, таких як смартфони та планшети. Її розробкою займається консорціум розробників Open Handset Alliance за фінансової підтримки Google. Офіційна презентація Android відбулася у листопаді 2007 року, а перший пристрій на базі цієї ОС вийшов у вересні 2008 року. Операційна система є безкоштовною і має відкритий код, доступний у межах Android Open Source Project, який поширюється за ліцензією Apache. Проте більшість пристроїв на Android постачаються з попередньо встановленим закритим програмним забезпеченням, наприклад Google Mobile Services, що включає Google Play, Google Chrome та інші сервіси. Близько 70% усіх Android-пристроїв функціонують у межах екосистеми Google, хоча існують і альтернативи — LineageOS, Fire OS та інші.



Рисунок 2.4 – Логотип ОС Android

Попри відкритий характер системи, бренд “Android” і логотип є зареєстрованими торговими марками Google, тому компанія встановлює вимоги до сертифікації пристроїв, що використовують її назву. Код Android також використовується в інших пристроях — медіаплеєрах, ігрових приставках, ПК, цифрових камерах, тощо. Окремі адаптації Android, такі як Android TV або Wear OS, орієнтовані на телевізори й смарт-годинники відповідно. Мобільні застосунки для Android поширюються у форматі APK через офіційні та альтернативні магазини: Google Play Store, Samsung Galaxy Store, Huawei AppGallery, F-Droid, Aptoide тощо [18].

Android лідирує на світовому ринку смартфонів за обсягами продажу з 2011 року. Станом на травень 2017 року Android мав понад 2 мільярди активних користувачів щомісяця, що зробило його операційною системою з найбільшою встановленою базою у світі. Магазин додатків Google Play нараховує понад 3 мільйони застосунків. Останньою стабільною версією ОС є Android 11, випущена 8 вересня 2020 року.

Компанія Android Inc. була заснована у жовтні 2003 року в Пало-Альто (Каліфорнія) Енді Рубіном, Річом Майнером, Ніком Сірсом і Крісом Уайтом. Ідея полягала у створенні "розумних мобільних пристроїв", які знали б про місце розташування та уподобання користувача. Спочатку планувалося створення ОС для цифрових камер, але через обмежений потенціал цього ринку компанія швидко переорієнтувалась на створення мобільної платформи, яка б конкурувала із Symbian та Windows Mobile.

У перші роки компанія мала фінансові труднощі, і один із друзів Рубіна, Стів Перлман, допоміг грошима, не вимагаючи частки у бізнесі. У липні 2005 року Google придбав Android Inc. щонайменше за 50 мільйонів доларів. Рубін та ключові члени команди перейшли працювати до Google, де почалася розробка мобільної платформи на базі ядра Linux. Компанія запропонувала її виробникам мобільних телефонів та мобільним операторам як гнучку й сучасну альтернативу існуючим платформам.

Перші витoki про плани Google щодо мобільного ринку з'явилися у грудні

2006 року. Перші прототипи Android нагадували пристрої BlackBerry — без сенсорного екрана, із фізичною клавіатурою. Проте вихід iPhone у 2007 році змусив команду змінити підхід, і в подальших версіях була запроваджена повна підтримка сенсорних екранів. У 2008 році й інші великі гравці — Nokia і BlackBerry - представили свої сенсорні смартфони, а Android остаточно перейшов на цей формат. Першим комерційним смартфоном на базі Android став HTC Dream (відомий також як T-Mobile G1), представлений 23 вересня 2008 року.

5 листопада 2007 року було оголошено про створення Open Handset Alliance - консорціуму провідних технологічних компаній, серед яких Google, HTC, Motorola, Samsung, Sprint, T-Mobile, Qualcomm, Texas Instruments. Їх метою була розробка першої відкритої, повнофункціональної платформи для мобільних пристроїв. У подальшому Android зіткнувся з конкуренцією з боку інших ініціатив з відкритим кодом — Symbian Foundation та LiMo Foundation, які також працювали над Linux-орієнтованими платформами. Наприкінці 2007 року було виявлено, що Google активно подає заявки на патенти в сфері мобільного зв'язку, що свідчило про серйозні амбіції компанії в цій галузі.

Починаючи з 2008 року, Android зазнав численних оновлень, які поступово вдосконалювали систему, додаючи нові функції та виправляючи помилки попередніх версій. Кожен основний випуск традиційно називався на честь десерту в алфавітному порядку — перші версії мали назви "Cupcake", "Donut", "Eclair", "Froyo" тощо. Під час презентації Android KitKat у 2013 році Google пояснив, що назви відображають солодкий внесок цих пристроїв у життя користувачів, хоча один із представників компанії зазначив, що ця традиція — внутрішня ініціатива.

У 2010 році Google запустив серію пристроїв Nexus у співпраці з виробниками техніки, щоб демонструвати нові версії Android та встановлювати програмні й апаратні стандарти. Ці пристрої вирізнялись "чистим" Android без надбудов і своєчасними оновленнями. У травні 2013 року Google оголосив про випуск спеціальної версії Samsung Galaxy S4 із "стоковим" Android, яка стала частиною програми Google Play Edition. У межах програми з'явилися також HTC One Google Play Edition та Moto G Google Play Edition. У 2015 році всі пристрої

цієї серії були зняті з продажу, що фактично означало завершення програми.

Уго Барра представляв Android на заходах та конференціях із 2008 по 2013 рік, після чого перейшов до компанії Xiaomi. За пів року до цього генеральний директор Google Ларрі Пейдж повідомив, що Енді Рубін залишає проєкт Android, щоб зайнятися іншими ініціативами в Google, а Сундар Пічай очолить Android. У серпні 2015 року Пічай став генеральним директором Google після утворення холдингу Alphabet, а керівництво Android перейшло до Хіросі Локгаймера.

У червні 2014 року Google представив програму Android One — набір еталонних апаратних рішень для створення недорогих, але якісних смартфонів, орієнтованих на ринки, що розвиваються. У вересні того ж року стартували перші пристрої Android One в Індії. Однак у 2015 році проєкт визнали невдалим через низький інтерес з боку споживачів і партнерів. Пізніше Google заявив про намір відновити Android One — спочатку в Африці (серпень 2015), а згодом і в США (2017), хоча при цьому не планував самостійно виготовляти пристрої.

У жовтні 2016 року Google представив смартфони Pixel і Pixel XL — перші моделі, які компанія позиціонувала як власні. Вони містили ексклюзивні функції, зокрема Google Assistant, і стали прямими наступниками серії Nexus. У жовтні 2017 року Google випустив друге покоління Pixel, завершивши перехід від Nexus до нової лінійки.

У травні 2019 року Android опинився в центрі торгового конфлікту між США та Китаєм, що зачепив компанію Huawei, яка, як і багато інших технологічних фірм, залежала від доступу до платформи Android. У відповідь Huawei влітку 2019 року заявила про розробку власної альтернативної операційної системи — Harmony OS, а також подала заявки на реєстрацію інтелектуальної власності в ряді країн. Проте компанія повідомила, що найближчим часом не планує повністю відмовлятися від Android, оскільки Harmony OS орієнтована переважно на пристрої Інтернету речей, а не смартфони.

22 серпня 2019 року стало відомо, що Android "Q" офіційно називатиметься Android 10, припинивши традицію використання назв десертів для основних версій. Google пояснив це тим, що подібні назви не є універсально зрозумілими

для глобальної аудиторії. У той же день стало відомо, що компанія встановила статую з цифрою "10" у новому офісі розробників Android. Android 10 був випущений 3 вересня 2019 року для пристроїв серії Google Pixel.

Інтерфейс. Стандартний користувацький інтерфейс Android базується на прямій взаємодії із сенсорним екраном. Взаємодія реалізується через жести — натискання, проведення, стискання — а також через віртуальну клавіатуру. Також підтримуються геймпади та повнорозмірні клавіатури, які можна підключити через USB або Bluetooth. Введення користувача обробляється негайно, забезпечуючи плавну взаємодію з системою; для зворотного зв'язку часто використовується вібрація.

Датчики, такі як гіроскоп, акселерометр і датчик наближення, використовуються додатками для додаткових функцій, як-от автоматичне обертання екрана або управління ігровими персонажами шляхом нахилу пристрою.

Додатки в Android розміщуються на головному екрані — аналогові робочого столу на ПК. Головний екран містить ярлики додатків і віджети: ярлики відкривають програми, а віджети автоматично оновлюють контент, наприклад, новини, електронну пошту або погоду. Головний екран може мати кілька сторінок, між якими можна гортати вручну. У магазині Google Play доступні сторонні лаунчери, які можуть змінювати зовнішній вигляд головного екрана, включно з імітацією інших ОС, як-от Windows Phone.

Більшість виробників пристроїв адаптують інтерфейс Android під власні потреби, створюючи унікальний зовнішній вигляд. У верхній частині екрана знаходиться панель стану, що показує інформацію про пристрій — рівень заряду, зв'язок тощо. Проведення вниз по цій панелі відкриває екран сповіщень, який містить оновлення та повідомлення від додатків. Сповіщення — це коротка інформація від програм, що неактивні в даний момент. Натискання на сповіщення відкриває відповідний додаток. Починаючи з Android 4.1 Jelly Bean, сповіщення можна розгортати, щоб переглядати більше інформації або взаємодіяти з додатком безпосередньо. На панелі також відображаються як активні, так і

беззвучні повідомлення.

У вікні "Усі програми" відображаються всі встановлені додатки з можливістю перетягування будь-якого з них на головний екран. Вікно "Останні" дає змогу швидко перемикатися між нещодавно використаними програмами.

Програми. Більшість Android-пристроїв постачаються з попередньо встановленими додатками Google: Google, YouTube, Gmail, Google Chrome, Карти, Google Play тощо. Мобільні додатки, які розширюють функціональність пристрою, мають бути 64-розрядними та створюються з використанням Android SDK. Основною мовою розробки з травня 2019 року стала Kotlin (анонсована у 2017), яка замінила Java. Java, як і C++, досі підтримується; можливе також комбінування Kotlin, Java, мов JVM із C/C++ залежно від вибору середовища розробки. Мова Go також підтримується, хоча й має обмежену кількість API.

Android SDK містить усе необхідне для розробки: налагоджувач, емулятор на базі QEMU, бібліотеки, приклади коду, документацію та навчальні матеріали. Спершу як офіційне IDE використовувалося Eclipse із плагіном Android Development Tools, однак пізніше Google зробив основною платформою Android Studio, створену на основі IntelliJ IDEA. Доступні також інші засоби розробки, зокрема NDK для C/C++ та візуальні середовища для початківців. У 2014 році Google представив фреймворк на основі Apache Cordova для адаптації вебдодатків HTML5 до Android у вигляді нативних програм.

Того ж року Google придбав Firebase — платформу, що надає інструменти для мобільної та веброзробки.

Android підтримує встановлення додатків через APK-файли або за допомогою спеціального додатку для керування програмами. Основним джерелом додатків є Google Play Store — магазин, доступний на пристроях із ліцензією Google Mobile Services, що відповідають вимогам сумісності. Через Google Play користувачі можуть завантажувати, оновлювати й переглядати додатки, серед яких понад 3 мільйони. У 2013 році кількість завантажень перевищила 50 мільярдів. Деякі оператори дозволяють оплачувати покупки в Google Play через мобільний рахунок. Станом на травень 2017 року понад мільярд користувачів

щомісяця активно користувалися Android, Gmail, Google Play, Картами та Chrome.

Через відкритість Android існують альтернативні магазини додатків — наприклад, SlideMe, GetJar, F-Droid, Amazon Appstore. Вони використовуються на пристроях без доступу до Google Play або для поширення додатків, що не відповідають політикам Google. Часто ці магазини підтримують лише безкоштовне й відкрите ПЗ. У жовтні 2020 року Google видалив низку додатків, зокрема Number Coloring, Princess Salon та Cats & Cosplay (20 млн завантажень), через порушення політики збору даних, про що повідомила Міжнародна рада з цифрової відповідальності.

Оскільки Android-пристрої працюють від акумулятора, система оптимізована для мінімального енергоспоживання. Невикористовувані додатки автоматично переходять у призупинений режим, що дозволяє швидко їх відновити без витрати ресурсів. Коли пам'ять пристрою заповнена, Android самостійно завершує фонові процеси, починаючи з тих, що не використовувались найдовше. У 2011 році Lifehacker попередив, що сторонні "вбивці задач" можуть шкодити продуктивності пристрою.

Апаратне забезпечення. Основною апаратною платформою Android є ARM. Підтримка архітектур x86 і x86-64 була додана пізніше. Проєкт Android-x86 забезпечував неофіційну підтримку x86 до того, як її офіційно впровадили. Раніше також підтримувались ARMv5TE і MIPS32/64, але згодом їх було виключено. Пристрої на процесорах Intel почали з'являтися з 2012 року. З моменту впровадження 64-розрядної архітектури Android (спочатку на ARM64) система, починаючи з версії 5.0 Lollipop, підтримує всі основні апаратні платформи [19].

Мінімальні вимоги до оперативної пам'яті для пристроїв з Android 7.1 залежать від типу пристрою: 2 ГБ — для високопродуктивних конфігурацій, 1 ГБ — для пристроїв із типовим екраном, і мінімум 512 МБ — для пристроїв зі специфікаціями базового рівня на 32-бітній архітектурі. Для Android 4.4 рекомендовано мати щонайменше 512 МБ оперативної пам'яті, а пристрої з "низькою оперативною пам'яттю" потребують мінімум 340 МБ, не враховуючи

обсяг пам'яті, зарезервований для роботи певних API, зокрема для базової смуги процесора.

Android 4.4 потребує 32-бітного процесора типу MIPS, ARMv7 або x86, а також графічного процесора з підтримкою OpenGL ES 2.0. Платформа підтримує графічні API OpenGL ES 1.1, 2.0, 3.0, 3.1, 3.2, а з Android 7.0 — також Vulkan (версія 1.1 доступна на деяких пристроях). Деякі додатки потребують певних версій OpenGL ES, тому для їх запуску необхідна відповідна підтримка з боку GPU.

Android-пристрої можуть мати широкий набір апаратних модулів: камери, GPS, датчики орієнтації, елементи керування для ігор, барометри, гіроскопи, акселерометри, датчики тиску, магнітометри, термометри, датчики наближення та сенсорні екрани. Існують API для необов'язкових апаратних засобів, які стали стандартом для певних типів пристроїв, зокрема смартфонів. Початково Android вимагав наявності таких компонентів, як мікрофон чи камера з автофокусом, однак із часом ці вимоги були спрощені або усунені. Наприклад, підтримка камер стала необов'язковою після впровадження Android у медіаприставках.

Окрім смартфонів і планшетів, Android використовується і на традиційному ПК-обладнанні — з клавіатурами, мишами тощо. Деякі постачальники налаштовують Android під стандартне апаратне забезпечення комп'ютерів. Такі версії Android, сумісні з ПК, зокрема на базі Android 4.4, доступні завдяки проєкту Android-x86. Android також можна запускати на архітектурі x86 через емулятор, що входить до складу Android SDK, або за допомогою сторонніх емуляторів.

Китайські компанії розробляють власні ПК та мобільні ОС на базі Android, позиціонуючи їх як конкурентів Microsoft Windows і Google Android. Після введення заборони на використання Windows 8 у державних установах, понад десять китайських компаній адаптували Android для використання на урядовому обладнанні.

Після аналізу доступних інструментів для реалізації поставленого завдання було затверджено наступний технологічний стек і сформульовано такі висновки:

- обрано мову програмування Java, вивчено її синтаксис і методи

розробки.

- визначено Android OS як цільову платформу для створення застосунку.
- вибрано інтегроване середовище розробки Android Studio для ефективного й зручного створення інтерфейсу мобільного додатку.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СИСТЕМИ

3.1 Діаграми варіантів використання

Під час першого запуску додатку відкривається вікно реєстрації, де користувач вводить свої персональні дані. Після завершення реєстрації йому пропонується додати продукти до холодильника для початку використання додатку.

На рисунку 3.1 представлена діаграма, що ілюструє доступні дії користувача зі списком продуктів у додатку. У початковій версії доступно 50 продуктів, проте їх кількість може змінюватися, оскільки передбачено можливість додавання нових позицій.

Після додавання продуктів до холодильника користувач переходить на головний екран додатку, де представлена інформація про рекомендований раціон. Діаграма на рисунку 3.2 демонструє дії, доступні користувачеві у вікні профілю.

Структурна діаграма інформаційної системи демонструє її внутрішню модульну організацію та взаємодію між окремими модулями. На рисунку 3.3 представлено відповідну структурну діаграму системи.

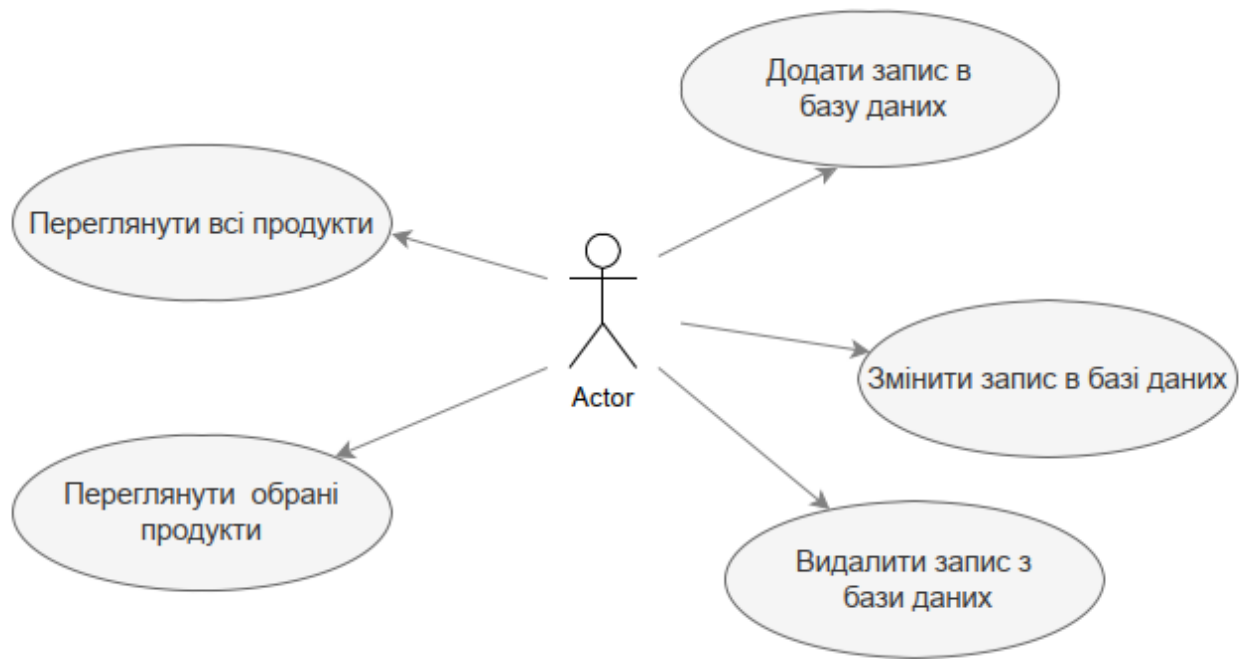


Рисунок 3.1 - Діаграма доступних дій користувача

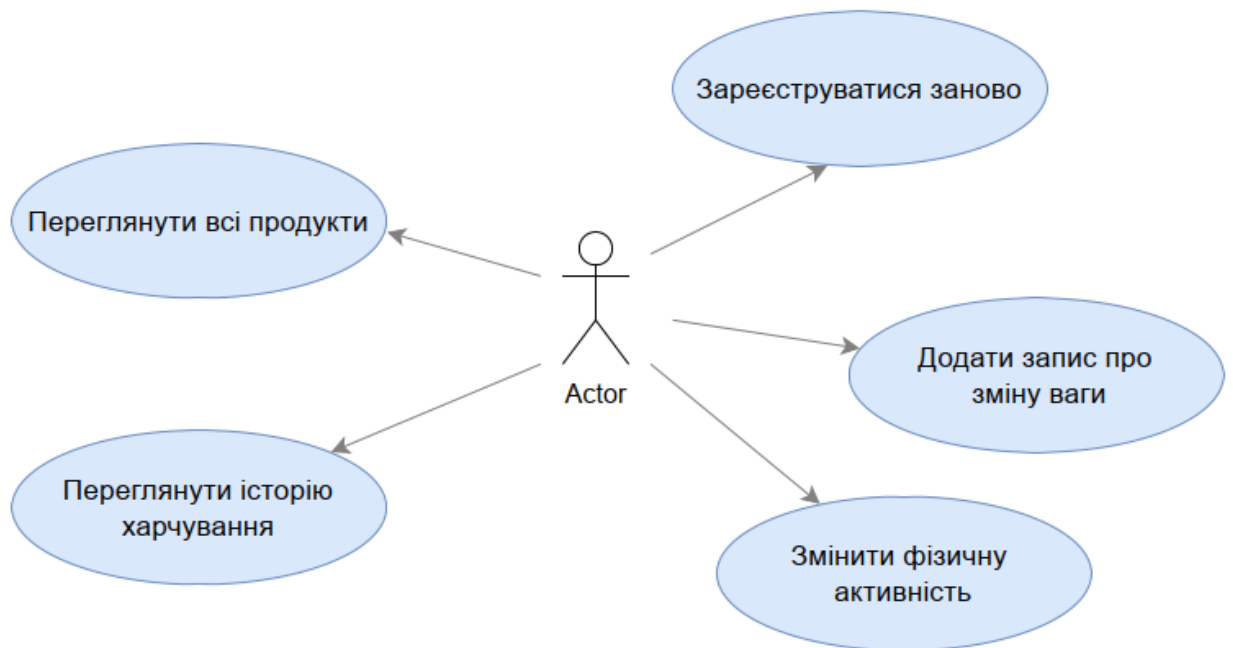


Рисунок 3.2 - Діаграма доступних дій користувача при вході у профіль

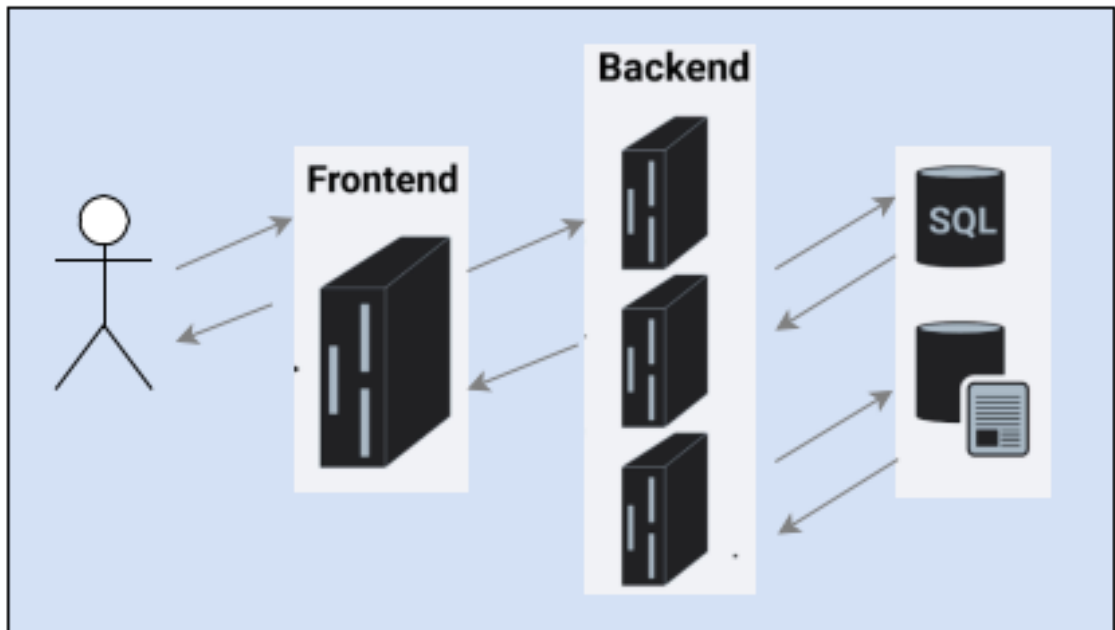


Рисунок 3.3 — Структурна діаграма ІС

3.2 Зберігання даних

База даних включає дві таблиці:

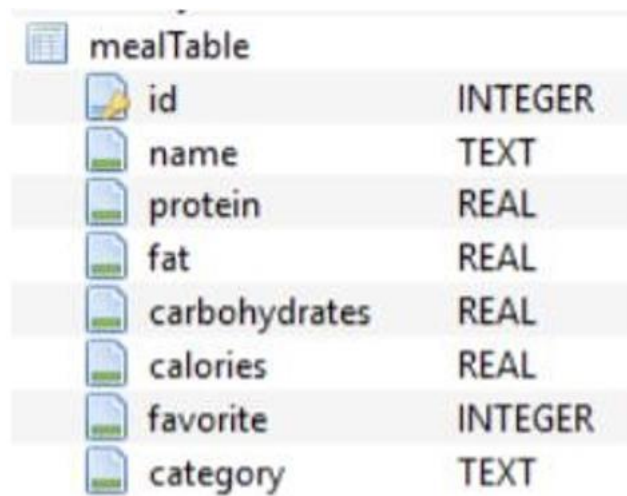
- Meal_table;
- History_meal.

Таблиці не пов'язані між собою та використовуються виключно для зберігання інформації. Таблиця Meal_table містить перелік харчових продуктів, з якими працює додаток (рис3.4).

Стовпці таблиці Meal_table:

- id – унікальний ідентифікатор;
- name – назва продукту;
- protein – вміст білків;
- fat – вміст жирів;
- carbohydrates – вміст вуглеводів;
- calories – калорійність продукту;

- `favorite` – ознака обраного продукту;
- `category` – категорія продукту.



mealTable	
id	INTEGER
name	TEXT
protein	REAL
fat	REAL
carbohydrates	REAL
calories	REAL
favorite	INTEGER
category	TEXT

Рисунок 3.4 - Таблиця MealTable

Таблиця `history_meal` зберігає дані про історію прийомів їжі користувача. Її структура включає такі стовпці:

- `id` – ідентифікатор;
- `date` – дата прийому їжі;
- `breakfast` – продукти та їх вага на сніданок;
- `breakfastTwo` – продукти та їх вага на другий сніданок;
- `lunch` – продукти та їх вага на обід;
- `dinner` – продукти та їх вага на вечерю;
- `dinnerTwo` – продукти та їх вага на другу вечерю;
- `vegetable` – кількість овочів у грамах.

На рисунку 3.5 зображено цю таблицю.

Дані користувача, зокрема ім'я, зріст, вага, вік, ціль і спосіб життя, зберігаються у файлі типу `Preferences`. Цей тип файлів використовується для зберігання системних налаштувань, подібно до `INI`-файлів у `Windows`.

historyTable		
	id	integer
	date	text
	breakfast	text
	breakfastTwo	text
	lunch	text
	dinner	text
	dinnerTwo	text
	vegetable	text

Рисунок 3.5 - Таблиця historyTable

3.3 Принцип роботи додатку

У додатку реалізовано такий функціонал:

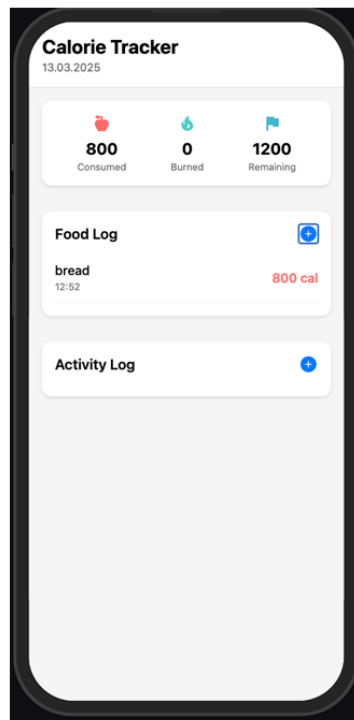
- додавання продукту до обраних;
- перегляд детальної інформації про продукт;
- редагування даних продукту;
- додавання та видалення продуктів із бази даних;
- перегляд інформації про користувача та його прогрес.

Робота додатку базується на розрахунку добової потреби в калоріях за формулою Міффіна — Сан Жеора (рис.3.6).

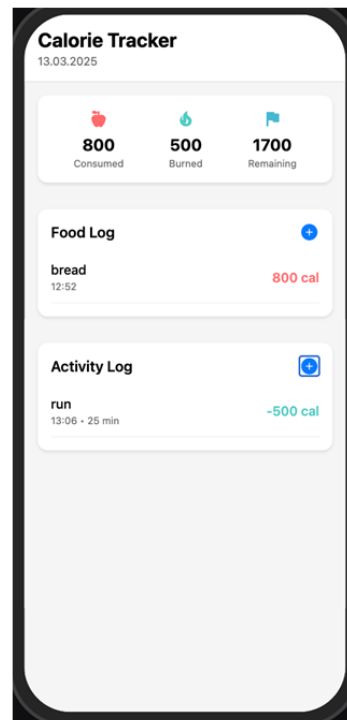
На сторінці історії (рис.3.6, б) можна зазначити які саме продукти були додані, та зберегається історія.

На рисунку 3.7 наведена сторінка функціоналу додатка.

Також існує можливість додання фізичної активності користувача, та скільки калорій було витрачено (рис.3.8).



а)



б)

Рисунок 3.6 – Екран додатку: а) початковий екран; б) історія доданих продуктів

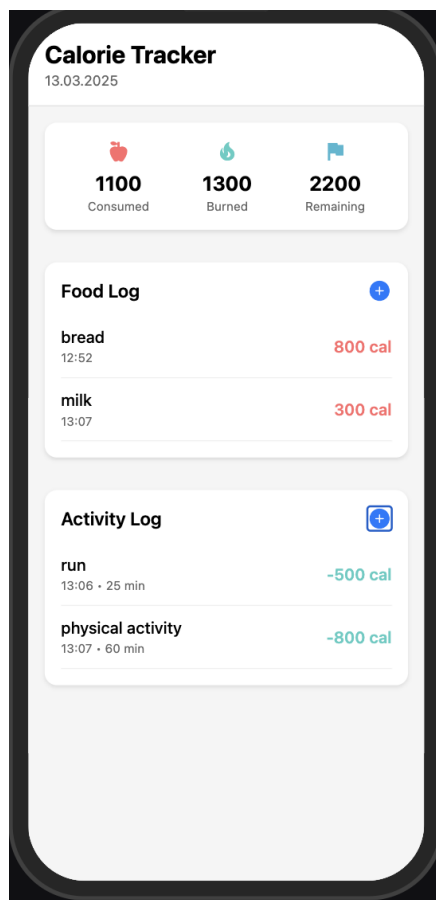


Рисунок 3.7 – Функціонал додатка

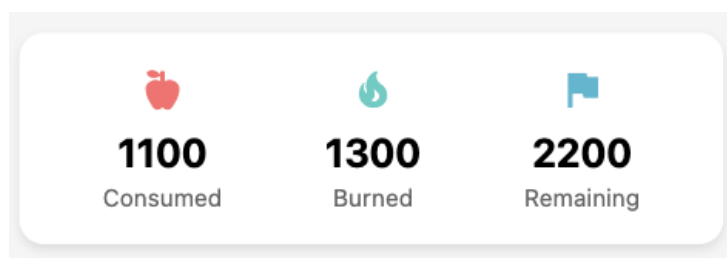


Рисунок 3.8 – Сумарна кількість спожитих та витрачених калорій

На рисунку 3.19 та 3.10 зображено підрахунок калорій у категоріях «Їжа» та «Активність».

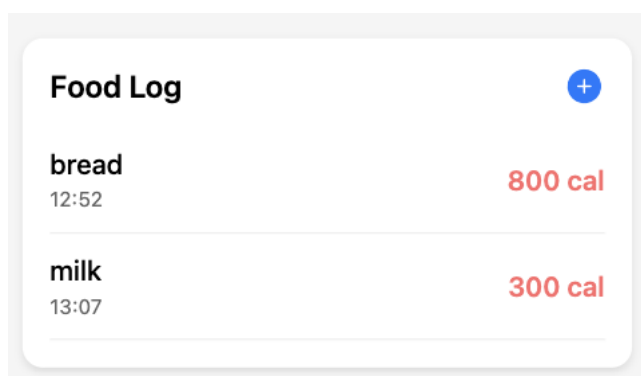


Рисунок 3.9 – Перелік калорій у категорії «Їжа»

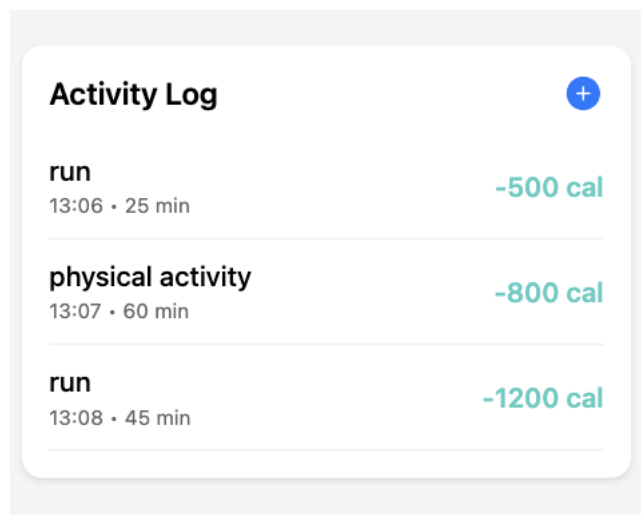


Рисунок 3.10- Перелік калорій витрачених у категорії «Активність»

В результаті функціонування додатку має покращити життя багатьох користувачів.

3.4 Вивинтаження додатку

Додаток перебуває на стадії прототипу та проходить тестування. Його встановлення здійснюється шляхом завантаження APK-файлу до пам'яті мобільного пристрою за допомогою файлового менеджера. Після завершення бета-тестування, усунення виявлених недоліків і врахування відгуків користувачів буде випущено першу стабільну версію додатку, яку можна буде встановити через PlayMarket.

У цьому розділі було розглянуто реалізацію мобільного додатку, описано логіку його роботи та наведено use-case діаграми взаємодії користувача з додатком. Описано метод зберігання даних та структуру бази даних. Проведено ознайомлення з початковими можливостями додатку, розробленого для підтримки здорового харчування.

Функціонал додатку включає:

- додавання продукту до обраного;
- перегляд детальної інформації про продукт;
- редагування даних про продукт;
- додавання та видалення продуктів з бази даних.

Представлено інтерфейс додатку, який надалі буде вдосконалено. Додаток сумісний з різними версіями Android і підтримує широкий діапазон розширень екрана — від 800×480 до 4K.

ВИСНОВКИ

На сьогодні багатьом людям важко підтримувати самодисципліну, особливо в умовах великого вибору шкідливої їжі, згубних звичок і малорухомого способу життя. Контроль за харчуванням є мінімально необхідним кроком для збереження здоров'я. Така ситуація стимулює розробку цифрових систем, орієнтованих на здорове харчування.

У процесі роботи над проєктом було виконано такі завдання: проаналізовано поняття здорового харчування, розглянути поняття мобільних додатків, проведено дослід методів розробки мобільних додатків; проведено огляд мов програмування, середовища розробки, проаналізовано цільову операційну систему. Було створено діаграму варіантів використання, описати структуру програмного забезпечення, розробити графічний інтерфейс користувача. Реалізовано інформаційну систему - створено базу даних, розроблено мобільний додаток, що виконує поставлені задачі.

У результаті створено просту систему для підтримки здорового харчування, яка виконує основний функціонал і може бути розширена в подальшому.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методи та інструменти для швидкої розробки мобільних додатків.
URL: [https://brander.ua/blog/metody-ta-instrumenty-dlya-shvydkoyi-rozrobky-mobilnykh -dodatkov](https://brander.ua/blog/metody-ta-instrumenty-dlya-shvydkoyi-rozrobky-mobilnykh-dodatkov) (дата звернення 14.04.2025).
2. Огляд MyFitnessPal. Чи варто завантажити додаток MyFitnessPal.
URL: <https://icoola.ua/blog/my-fitness-pal-app?srsltid=AfmBOoqWJupRj4zbHGutvLWOc-37zLnLiyNGDIvayWMdpf2Ye6ufpe2X> (дата звернення 23.04.2025).
3. Lifesum: AI Calorie Tracker. URL: <https://play.google.com/store/apps/details?id=com.sillens.shapeupclub&hl=uk> (Last accessed: 14.04.2025).
4. FatSecret. URL: <https://ingeniusua.org/articles/5-krashchykh-dodatkov-dlya-pidrakhunku-kaloriy> (Last accessed: 18.04.2025).
5. Кращі додатки для підрахунку калорій: YAZIO. URL: https://www.moyo.ua/ua/news/luchshie_prilozheniya_dlya_podscheta_kaloriy_top-10.html (дата звернення 02.05.2025).
6. Ivanov O., Petrenko Y. Розробка мобільних додатків на платформі Android: від Java до Kotlin. Київ: Наукова думка; 2020. 256 с.
7. Сьєрра К., Бейтс Б. Head First. Java. Легкий для сприйняття Видавництво «Фабула». 2022. 720с.
8. Васильєв О. Програмування мовою Java URL: <https://bohdan-books.com/catalog/book/133954/>. (дата звернення 22.04.2025).
9. МакКензі, С. Kotlin для початківців / С. МакКензі. – К.: Видавнича група BHV, 2021. – 320 с.
10. Dart Programming Language. URL: <https://dart.dev/> (Last accessed: 28.04.2025).
11. Android NDK Guides URL: <https://developer.android.com/ndk> (Last accessed: 13.05.2025).
12. Chen L., Wang M. Android Studio: best practices for efficient app development. Software: Practice & Experience, 2019. P. 1568–1585.

13. Програмування в IntelliJ IDEA / Кол. авт. – К.: Видавництво «Фенікс», 2020. – 250 с.
14. Eclipse IDE URL: <https://www.eclipse.org/ide/> (Last accessed: 28.05.2025).
15. Браун, Дж. SQLite для Android. Зберігання даних у мобільних застосунках / Дж. Браун. – К.: Діалектика, 2021. – 240 с.
16. Room Persistence Library URL: <https://developer.android.com/jetpack/androidx/releases/room> (Last accessed: 21.05.2025).
17. Нільсен, М. Firebase. Повний посібник з розробки хмарних застосунків / М. Нільсен. – К.: Нове знання, 2022. – 280 с.
18. Garcia R., Singh A. Hardware considerations in Android devices: performance and energy efficiency. IEEE Transactions on Mobile Computing, 2021.P. 512–526.
19. Zhang Y., Liu X. Integrating wearables with Android platform: sensors, connectivity, and data handling. Sensors. 2022. P. 2501.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

Лістинг А.1 – Файл name.html

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Geolocation</h2>
<p>Click the button to get your coordinates.</p>
<button onclick="getLocation()">Try It</button>
<p id="demo"></p>

<script>
const x = document.getElementById("demo");

function getLocation() {
  try {
    navigator.geolocation.getCurrentPosition(showPosition);
  } catch {
    x.innerHTML = err;
  }
}

function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>

</body>
</html>
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Дипломний проєкт бакалавра
На тему: РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ПІДРАХУНКУ КАЛОРІЙ
КОРИСТУВАЧА

Ст. групи КНТ-512сп Кирило КРАПІВКІН
Керівник М.Б. ІЛ'ЯШЕНКО

1

Мета роботи — проектування і розробка мобільного додатку для здорового харчування.

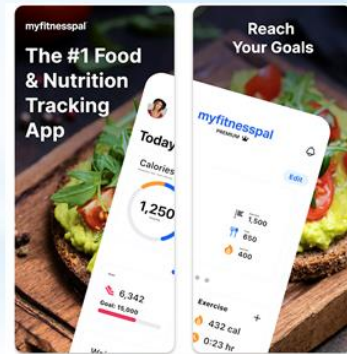
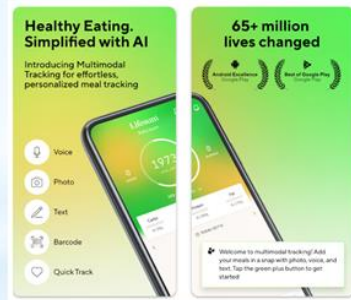
Для досягнення цієї мети поставлено такі завдання:

- проаналізувати поняття здорового харчування,
- розглянути поняття мобільних додатків,
- дослідити методи розробки мобільних додатків;
- провести огляд мови програмування,
- оглянути середовище розробки,
- проаналізувати цільову операційну систему;
- створити діаграму варіантів використання,
- описати структуру програмного забезпечення,
- побудувати діаграму класів,
- розробити графічний інтерфейс користувача,
- підготувати інструкцію користувача.

2

* Огляд існуючих мобільних додатків:

Lifsum
FatSecret
Cronometer
YAZIO



3

* Вибір програмного забезпечення

Мови програмування:

- Java
- Kotlin
- Dart
- C++

Середовища розробки:

- Andriod Studio
- IntelliJ IDEA
- Eclipse

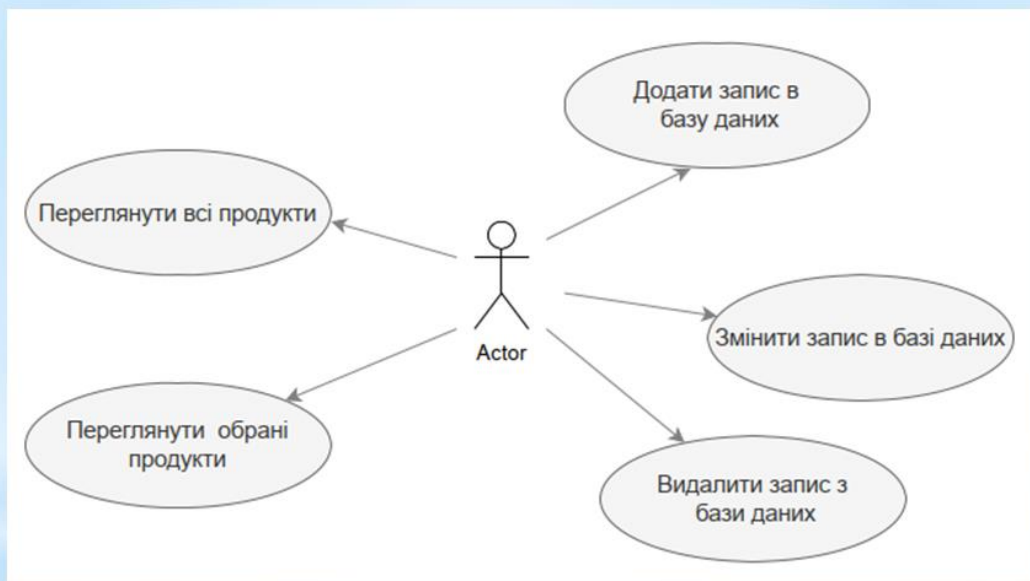
Бази даних:

- SQLite
- Room
- Firebase Realtime Database

Операційна система: Android

4

* РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СИСТЕМИ



Діаграма доступних дій користувача

5



Діаграма доступних дій користувача при вході у профіль

6

mealTable		
id	INTEGER	
name	TEXT	
protein	REAL	
fat	REAL	
carbohydrates	REAL	
calories	REAL	
favorite	INTEGER	
category	TEXT	

- ✓ id – унікальний ідентифікатор;
- ✓ name – назва продукту;
- ✓ protein – вміст білків;
- ✓ fat – вміст жирів;
- ✓ carbohydrates – вміст вуглеводів;
- ✓ calories – калорійність продукту;
- ✓ favorite – ознака обраного продукту;
- ✓ category – категорія продукту.

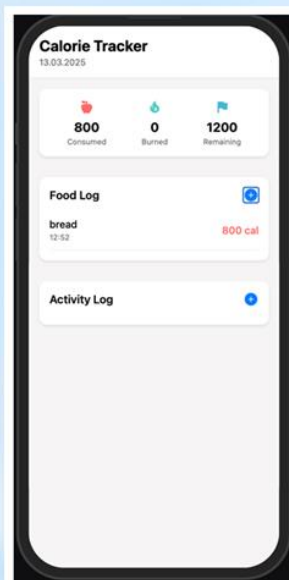
Перелік харчових продуктів

historyTable		
id	integer	
date	text	
breakfast	text	
breakfastTwo	text	
lunch	text	
dinner	text	
dinnerTwo	text	
vegetable	text	

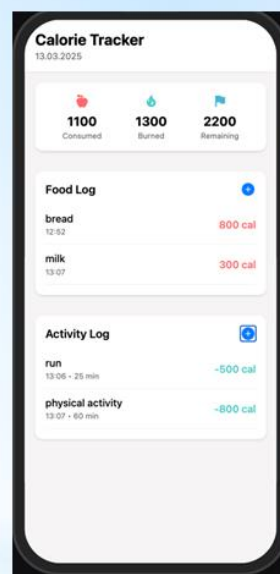
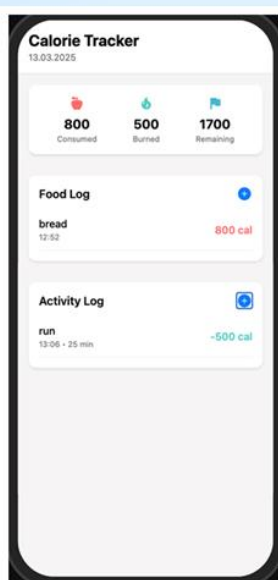
- ✓ id – ідентифікатор;
- ✓ date – дата прийому їжі;
- ✓ breakfast – продукти та їх вага на сніданок;
- ✓ breakfastTwo – продукти та їх вага на другий сніданок;
- ✓ lunch – продукти та їх вага на обід;
- ✓ dinner – продукти та їх вага на вечерю;
- ✓ dinnerTwo – продукти та їх вага на другу вечерю;
- ✓ vegetable – кількість овочів у грамах.

7

Дані про історію прийомів їжі користувача



Екран додатку



Функціонал додатка

* ВИСНОВОК

У процесі роботи над проєктом було виконано такі завдання:
проаналізовано поняття здорового харчування,

- ✓ розглянути поняття мобільних додатків,
- ✓ проведено дослід методів розробки мобільних додатків;
- ✓ проведено огляд мов програмування,
- ✓ середовища розробки,
- ✓ проаналізовано цільову операційну систему.
- ✓ створено діаграму варіантів використання,
- ✓ описати структуру програмного забезпечення,
- ✓ розробити графічний інтерфейс користувача.
- ✓ створено базу даних,
- ✓ розроблено мобільний додаток.

9

* Доклад завершено, дякую за увагу!

10