Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка мобільного додатку для підтримки здорового способу життя із використанням Kotlin

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Крвавич Д.П.
(підпис) (прізвище та ініціали)

Керівник _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Крвавичу Дмитру Петровичу
(прізвище, ім'я, по батькові)
1. Тема роботи Розробка мобільного додатку для підтримки здорового способу життя із використанням Kotlin

Керівник роботи Шимчук Григорій Валерійович, старший викладач кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444*
2. Термін подання студентом завершеної роботи 23 червня 2025 р.
3. Вихідні дані до роботи Літературні та інтернет джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання. 1.1 Огляд існуючих рішень для підтримки здорового способу життя. 1.2 Аналіз вибору мови програмування та середовища розробки, що використовується для створення мобільних додатків. 1.3 Постановка задачі для розробки мобільного додатку для підтримки здорового способу життя. 1.4 Висновок до першого розділу. 2. Реалізація та проектування мобільного додатку для підтримки здорового способу життя. 2.1 Архітектура мобільного додатку для підтримки здорового способу життя. 2.2 Пошук актантів та варіантів використання. 2.3 Проектування та реалізація екранів користувацького інтерфейсу мобільного додатку для підтримки здорового способу життя 2.4 Проектування бази даних мобільного додатку для підтримки здорового способу життя 2.5 Реалізація логіки додатку. 2.6 Висновки до другого розділу. 3. Практична частина. Демонстрація та тестування мобільного додатку. 3.1 Демонстрація роботи мобільного додатку для підтримки здорового способу життя. 3.2 Тестування та валідація мобільного додатку для підтримки здорового способу життя. 3.3 Висновки до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. 4.1 Долікарська допомога при ушкоджені м'яких тканин, суглобів і кісток. 4.2 Проведення інструктажів з охорони праці. Висновки. Перелік джерел. Додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., к.т.н., доцент		

7. Дата видачі завдання 07 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	07.05.2025	*Виконано*
2.	Підбір, переклад та опрацювання літературних джерел	08.05.2025-11.05.2025	*Виконано*
	по розробці мобільних додатків для підтримки		
	здорового способу життя.		
3.	Виконання дослідження щодо вибору програмних	11.05.2025-13.05.2025	*Виконано*
	засобів та архітектури.		
4.	Розроблення мобільного додатку для підтримки	14.05.2025-24.05.2025	*Виконано*
	здорового способу життя із використанням Kotlin		
5.	Оформлення розділу «Аналіз предметної області та	25.05.2025-03.06.2025	*Виконано*
	постановка завдання»		
6.	Оформлення розділу «Реалізація та проектування	25.05.2025-03.06.2025	*Виконано*
	мобільного додатку для підтримки здорового способу		
	життя»		
7.	Оформлення розділу «Практична частина.	25.05.2025-03.06.2025	*Виконано*
	Демонстрація та тестування мобільного додатку»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	04.06.2025-07.06.2025	*Виконано*
9.	Виконання завдання до підрозділу «Основи охорони	04.06.2025-07.06.2025	*Виконано*
	праці»		
10.	Оформлення кваліфікаційної роботи	08.06.2025-10.06.2025	*Виконано*
11.	Нормоконтроль	11.06.2025-13.06.2025	*Виконано*
12.	Перевірка на плагіат	16.06.2025	*Виконано*
13.	Попередній захист кваліфікаційної роботи	18.06.2025	*Виконано*
14.	Захист кваліфікаційної роботи	25.06.2025	

Студент _____ Крвавич Д.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка мобільного додатку для підтримки здорового способу життя із використанням Kotlin // Кваліфікаційна робота освітнього рівня «Бакалавр» // Крвавич Дмитро Петрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2025 // С. 60, рис. – 22, табл. – 0, кресл. – 0, додат. – 2, бібліогр. – 52.

Ключові слова: мобільний додаток, здоровий спосіб життя, kotlin, android, jetpack compose, room, mvvm, android studio.

Кваліфікаційна робота присвячена розробці мобільного додатку для підтримки здорового способу життя з використанням мови програмування Kotlin.

У першому розділі кваліфікаційної роботи проведено аналіз предметної області, зокрема огляд існуючих мобільних додатків, визначено їх переваги та недоліки, а також сформульовано вимоги до розроблюваного додатку. Проаналізовано вибір мови програмування та середовища розробки.

У другому розділі кваліфікаційної роботи описано проектування додатку, включаючи розробку архітектури за патерном MVVM, створення діаграма варіантів використання, а також використання сучасних бібліотек Jetpack Compose для інтерфейсу та Room для роботи з базою даних. Розглянуто методи реалізації основних функцій.

У третьому розділі кваліфікаційної роботи продемонстровано реалізацію та тестування додатку. Продемонстровані всі екрани мобільно додатку для підтримки здорового способу життя, проведено тестування за допомогою інструментів Android Studio, що підтвердило стабільність та функціональність додатку.

ANNOTATION

Development of a Mobile Application for a Healthy Lifestyle Using Kotlin // Qualification work of the educational level «Bachelor» // Krvavych Dmytro // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2025 // P.60, fig. – 22, tabl. – 0, chair. – 0, annexes. –2, references – 52.

Keywords: mobile application, healthy lifestyle, kotlin, android, jetpack compose, room, mvvm, android studio.

The qualification work is devoted to the development of a mobile application to support a healthy lifestyle using the Kotlin programming language.

The first chapter of the qualification work analyzes the subject area, including a review of existing mobile applications, identifies their advantages and disadvantages, and formulates the requirements for the application being developed. The choice of programming language and development environment is analyzed.

The second section of the qualification work describes the design of the application, including the development of the architecture according to the MVVM pattern, the creation of a use case diagram, and the use of modern libraries Jetpack Compose for the interface and Room for working with the database. The methods of implementing the main functions are considered.

The third section of the qualification work demonstrates the implementation and testing of the application. All screens of the mobile application for maintaining a healthy lifestyle are demonstrated, testing is carried out using Android Studio tools, which confirmed the stability and functionality of the application.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Android – власницька мобільна операційна система від Google.

API (англ. Application Programming Interface) – набір визначень і протоколів для взаємодії між програмними компонентами.

DAO (англ. Data Access Object) – шаблон проєктування для організації доступу до даних у базі.

IOS (англ. Iphone Operating System) – власницька мобільна операційна система від Apple.

MVVM – Model-View-ViewModel.

SQLite – компактна вбудована реляційна база даних з відкритим кодом.

UI (англ. User Interface) – частина програми, через яку користувач взаємодіє з додатком.

БД – база даних.

Користувач – особа, яка взаємодіє з мобільним додатком.

Мобільний додаток – програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі здоровий спосіб життя стає все більш актуальним. Зростання захворюваності на хронічні захворювання, зниження рівня рухової активності населення, широке розповсюдження шкідливих звичок зумовили необхідність пошуку ефективних засобів підтримки власного здоров'я. Одним із найбільш доступних рішень для цього є мобільні додатки, які дозволяють користувачам контролювати фізичну активність, харчування, рівень споживаної води та інші важливі показники. Зважаючи на те, що сьогодні майже кожен має мобільний пристрій, створення мобільного додатку для підтримки здорового способу життя є надзвичайно важливим.

Розробка мобільного додатку для підтримки здорового способу життя дозволяє автоматизувати рутинні процеси самоконтролю, надає зручний інтерфейс для ведення обліку харчування, фізичної активності та водного балансу. Саме з метою підтримки здорового способу життя було запропоновано розробити мобільний застосунок із використанням мови програмування Kotlin.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробка мобільного застосунку для підтримки здорового способу життя з використанням мови Kotlin. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- Дослідити сучасні мобільні застосунки, що спрямовані на підтримку здорового способу життя.
- Ознайомитись з доступними інструментами для розробки мобільних застосунків мовою Kotlin.
- Сформулювати постановку задачі – проаналізувати потреби користувачів, визначити вимоги до додатку та виділити основні функціональні можливості.
- Розробити архітектуру додатку з урахуванням принципів модульності та зручності у використанні.

- Реалізувати основні функції додатку, зокрема облік калорій, води, макронутрієнтів, активності та персоналізовані налаштування.
- Провести тестування розробленого мобільного додатку та оцінити його функціональність і зручність для користувача.

Практичне значення одержаних результатів. Розроблений мобільний додаток для підтримки здорового способу життя із використанням Kotlin має важливе практичне значення, оскільки сприяє покращенню якості життя користувачів шляхом автоматизації та систематизації процесів самоконтролю. Мобільний додаток дозволяє ефективно керувати основними аспектами здорового способу життя, такими як харчування, фізична активність та водний баланс, що особливо актуально в умовах дедалі більшого поширення сидячого способу життя та харчових звичок, що шкодять здоров’ю.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд існуючих рішень для підтримки здорового способу життя

Для визначення оптимального функціоналу та інтерфейсу розроблюваного мобільного додатку для підтримки здорового способу життя було проаналізовано три популярні додатки (по кількості завантаженню Google Play Market): лічильник калорій fatsecret, таблиця калорійності – калорії (Dine4Fit), YAZIO Food & Calorie Counter.

Почнемо з першого додатку fatsecret. Fatsecret – це безкоштовний лічильник калорій [1], який пропонує широкий набір функцій. Він містить харчовий щоденник, базу даних продуктів, колекцію корисних рецептів, журнал фізичних вправ та графік змін ваги. Також містить вбудований сканер штрих-кодів за допомогою якого можна відстежувати упаковані продукти. На головному екрані відображається загальна кількість спожитих калорій, а також співвідношення білків, жирів і вуглеводів – як за весь день, так і за окремі прийоми їжі. На рисунку 1.1 можем побачити дизайн додатку.

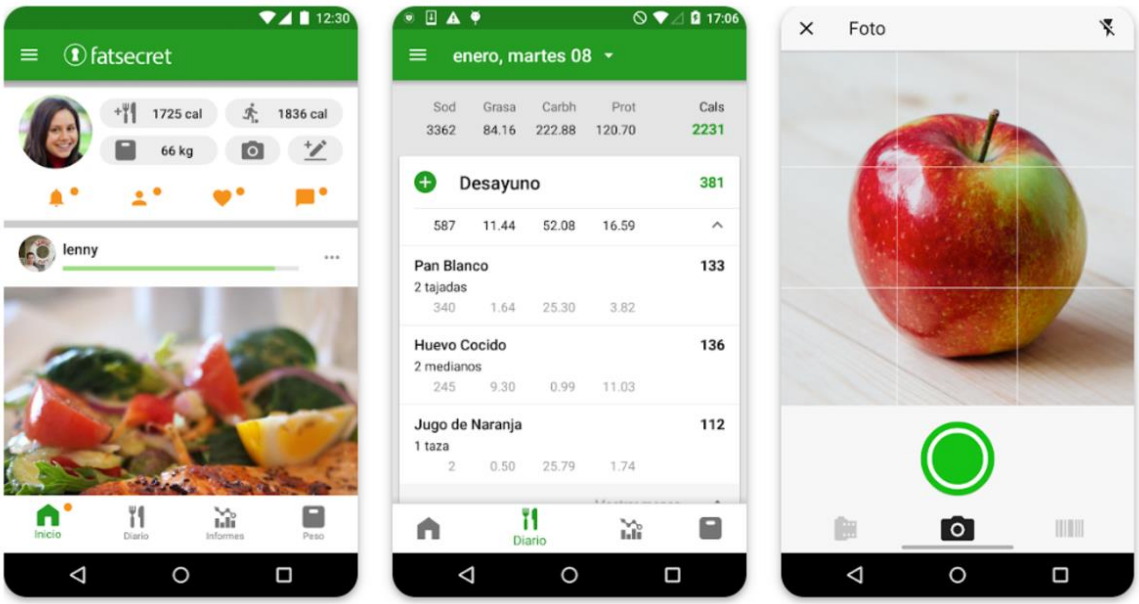

Рисунок 1.1 – Дизайн додатку fatsecret

Серед плюсів цього додатку виділяють: велику базу даних продуктів харчування; інтеграція з Google Fit, Samsung Health та Fitbit для відстеження вправ; доступність багатьох функцій безкоштовно [2]. Серед мінусів користувачі виділяють досить захаращений і заплутаний інтерфейс.

Далі перейдемо до наступного додатку таблиця калорійності – калорії. Таблиця калорійності позиціонує себе як додаток що допоможе схуднути без шкоди для здоров'я та розумно, або допоможе набрати м'язову масу безкоштовно та дуже ефективно [3]. На рисунку 1.2 можем побачити дизайн додатку.

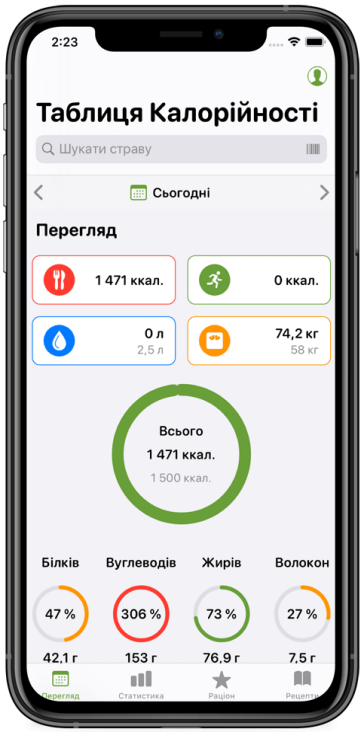

Рисунок 1.2 – Дизайн додатку таблиця калорійності

Серед плюсів таблиці калорійності виділяють [4]: зручний список окремих продуктів і готових страв з калоріями та фото; можна добавляти свою їжу; можливість залишати свої замітки; можна додавати продукти по штрих-коду з пакування.

З недоліків користувачі жаліються на часту рекламу, а також те що забрали деякий функціонал і перемістили його в преміум підписку.

Перейдемо до останнього третього додатку YAZIO Food & Calorie Counter. YAZIO рекомендує себе, як дуже ефективний додаток для схуднення здоровим способом без необхідності дотримуватися суворої дієти [5], із чудовими рейтингами в магазині Google Play 4,4 зірки з п’яти на основі 632 тисячі відгуків. На рисунку 1.3 можем побачити дизайн додатку YAZIO.

Рисунок 1.3 – Дизайн додатку YAZIO

До особливостей цього додатку належать [22]: система підказує, скільки потрібно вжити калорій, випити води та зробити кроків за день, виходячи з поставленої мети; можливість добавляння спортивних вправ, які будуть враховуватись при створенні індивідуального плану харчування; щоденні поради від програм; наявність сканеру штрих-коду.

Однак додаток також має і недоліки, він обмежений оскільки пропонує додаткову преміальну щомісячну оплату, яка дає доступ до покращених функцій, таких як спеціалізовані плани харчування або різні аналізи поживних речовин і вітамінів, серед інших покращень. Крім того, користувач скаржаться на рекламу ледь не після кожної дії і незручний калькулятор випитої води.

Отже, незважаючи на наявність великої кількості подібних додатків, користувачі часто стикаються з надмірною складністю інтерфейсу, потребою в

платних підписках або відсутністю підтримки української мови. Тому актуальним є створення зручного, інтуїтивно зрозумілого мобільного додатку для підтримки здорового способу життя, який враховує локальні потреби користувача та є доступним без обов'язкових платних функцій.

1.2 Аналіз вибору мови програмування та середовища розробки, що використовується для створення мобільних додатків

Розробка мобільних додатків є однією з найактуальніших і найпоширеніших задач у сучасній IT-галузі. За даними різних джерел, користувачі витрачають від 50% до 70% свого «цифрового часу» саме на взаємодію з мобільними додатками. Сьогодні основними операційними системами для мобільних пристроїв залишаються Android, яка займає приблизно 70% ринку, та iOS з часткою близько 28%. Решта 2% припадають на інші операційні системи, такі як Windows Phone, KaiOS чи Samsung OS, які поступово втрачають популярність [6].

Розробка додатків для кожної операційної системи здійснюється з використанням відповідного середовища програмування та спеціалізованих мов програмування. Наприклад, створення програм для Android базується на використанні Android Studio, де застосовують мови Kotlin або Java. У випадку iOS розробка здійснюється в Xcode за допомогою мови Swift. Такий підхід, де застосовуються нативні (рідні) інструменти та мови для кожної платформи, отримав назву «рідна розробка» (Native).

Головними перевагами Native розробки є можливість створення додатків із необмеженими функціональними можливостями, за винятком обмежень самої платформи, а також максимальна швидкість роботи додатків на пристроях користувачів. Проте цей підхід має і певні недоліки. Серед них варто виділити необхідність розробки та підтримки двох окремих додатків – для Android та iOS – що ускладнює процес задоволення потреб більшості користувачів.

Для вирішення цих проблем були розроблені кросплатформенні інструменти, які дозволяють створювати додатки для різних операційних систем за допомогою єдиної кодової бази. Зокрема, у 2014 році компанія Google представила фреймворк Flutter, який використовує власну мову програмування Dart, а в 2015 році Facebook випустила React Native [23]. Обидва інструменти значно спрощують процес створення додатків для Android та iOS, даючи можливість розробникам працювати у єдиному середовищі, економлячи час і ресурси як для себе, так і для своїх клієнтів.

Перевага створення кросплатформних додатків цілком очевидна – завдяки єдиній кодовій базі програмне забезпечення легше як розробляти, так і підтримувати. Однак цей підхід має й суттєві недоліки. Найважливішим із них є те, що на розробку впливають обмеження не тільки самої платформи, але й фреймворку. Ще одним мінусом є те, що додатки, створені за допомогою кросплатформних технологій, часто демонструють нижчу швидкість роботи порівняно з нативними програмами, особливо якщо йдеться про фреймворк React Native.

Оскільки мобільний додаток для підтримки здорового способу життя повинен бути ефективним, тобто швидким і здатним максимально використовувати можливості операційної системи та апаратного забезпечення, а потреба в кросплатформенності наразі відсутня, доцільно розробляти додаток виключно для платформи Android, застосовуючи нативний підхід.

Для нативної розробки під Android доступні декілька мов програмування, проте ми розглянемо дві між якими у нас був вибір, тому необхідно проаналізувати їхні переваги та недоліки, щоб вибрати найбільш відповідний варіант.

Розглянемо першу мову програмування Java [30]. Класична мова програмування Java відіграла значну роль у розвитку Android-додатків. Вона стала першою офіційною мовою програмування для цієї платформи, адже Google свого часу обрав її, щоб зробити створення додатків для Android більш

доступним і популярним. Завдяки Java було розроблено величезну кількість додатків, які й досі успішно функціонують на мільйонах пристроїв.

Однією з ключових переваг Java у розробці для Android є її надійність. Додатки, створені за допомогою цієї мови, зазвичай відзначаються стабільністю в роботі та високим рівнем безпеки, що є важливим для користувачів і гарантує захист даних. Окрім того, Java має велетенську спільноту розробників. Це забезпечує доступ до численних ресурсів, підтримки, а також великої кількості бібліотек і фреймворків, які значно полегшують процес створення програмного забезпечення. Коли виникають труднощі, особливо цінно мати можливість легко знайти відповідь або рішення [7].

Тепер про складнощі. Однією з головних проблем під час використання Java у порівнянні зі сучаснішими мовами, такими як Kotlin, є те, що Java часто вимагає написання більшої кількості коду для досягнення тієї ж функціональності. Це може призвести до збільшення обсягу коду та ускладнення структури додатка, що, у свою чергу, робить розробку менш ефективною і ускладнює його підтримку. Додатково, синтаксис Java може бути важким, особливо якщо порівнювати з більш сучасними мовами, що подовжує час розробки та створює додаткові можливості для помилок. І останнє, Java часом відстає у впровадженні нових функцій порівняно з сучасними альтернативами.

Розглянемо тепер Kotlin [31]. Ця мова програмування справила значний вплив на Android-спільноту, завдяки своїй інноваційності та ефективності. У 2017 році Google проголосила Kotlin офіційною мовою розробки для Android. Розгляньмо докладніше, що саме робить Kotlin сучасним і перспективним вибором для створення Android-додатків.

Kotlin став офіційною мовою розробки для Android через декілька важливих переваг:

- Повна сумісність із Java: Завдяки цьому розробники можуть поступово інтегрувати Kotlin в існуючі проєкти без необхідності починати все спочатку.

- Типобезпека: Ця особливість дозволяє виявляти багато помилок ще на етапі компіляції, що значно знижує ймовірність появи багів у додатках.
- Сучасні можливості: Kotlin пропонує лямбди, функції вищого порядку, а також розширення, що допомагають створювати більш простий для читання, компактний і виразний код.
- Зменшення шаблонного коду: Це сприяє спрощенню підтримки додатків і прискорює процес розробки.

Головною ж перевагою Kotlin є його тісна інтеграція з Android Studio – офіційним середовищем розробки для Android від Google. Це створює максимально комфортні умови для роботи над додатками, забезпечуючи широку підтримку і зручність у використанні всіх інструментів платформи.

Розглянемо ще середовище розробки – Android Studio. Android Studio – це інтегроване середовище розробки (IDE), створене спеціально для роботи з платформою Android [32]. Його було офіційно представлено 16 травня 2013 року на конференції Google I/O. Початкова версія 0.1 стала доступною в тому ж році, у травні, а в червні 2014 року на етап бета-тестування перейшла версія 0.8. Перша стабільна версія 1.0 побачила світ у грудні 2014 року, після чого Google припинила підтримку плагіна Android Development Tools (ADT) для Eclipse. Заснована на основі популярного програмного забезпечення IntelliJ IDEA від компанії JetBrains, Android Studio є офіційним і рекомендованим засобом розробки додатків для Android.

Крім цього, Android Studio надає [24]:

- Інтеграцію з Gradle для автоматизованого збирання проєктів.
- Емулятори Android-пристроїв для тестування додатків.
- Інструменти профілювання продуктивності.
- Шаблони основних макетів на компонентів Android.
- Підтримку сучасних мов програмування.

Це середовище доступне для операційних систем Windows, macOS та Linux. Приклад інтерфейсу Android Studio зображено на рисунку 1.4.

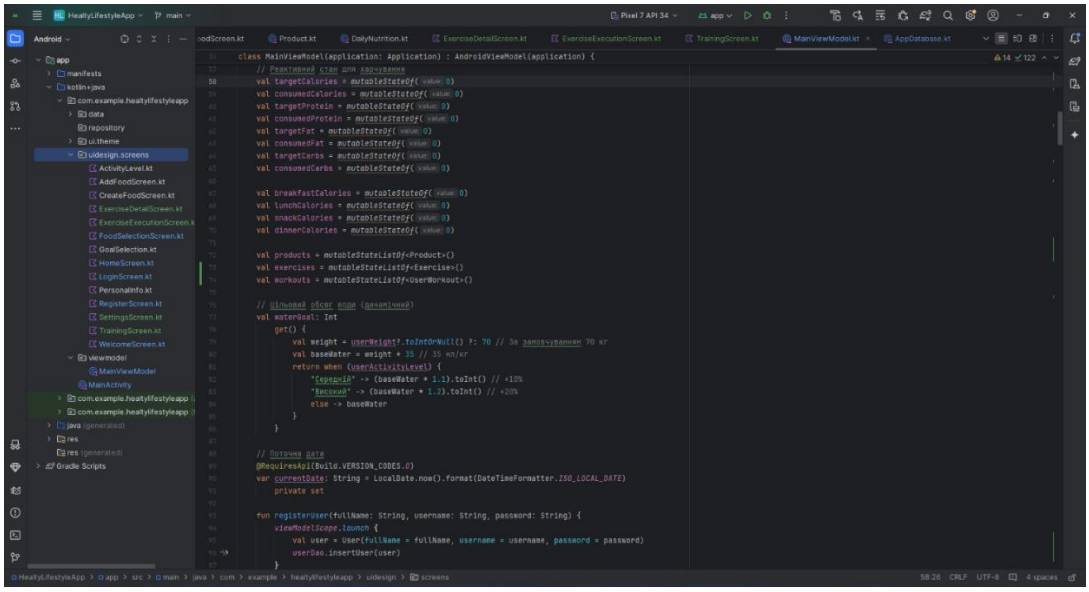

Рисунок 1.4 – Інтерфейс Android Studio

Таким чином, комбінація Kotlin + Android Studio є оптимальним вибором для розробки мобільного додатку для підтримки здорового способу життя під Android, який відповідає сучасним вимогам як з точки зору продуктивності, так і користувацького досвіду.

1.3 Постановка задачі для розробки мобільного додатку для підтримки здорового способу життя

Розробка мобільного додатку для підтримки здорового способу життя призначена для автоматизації процесів самоконтролю над споживанням калорій та макронутрієнтів, а також для ведення обліку тренувань і особистих фізіологічних параметрів. Підхід до визначення постановки задачі додатку в першу чергу повинен грунтуватися на принципах Human-centered design (HCD), тобто орієнтуватися на проблеми людини (користувача), і робити ставку на інтерфейс [25].

Головна мета застосунку – допомогти користувачеві досягти своєї фітнес-мети (набір ваги, скидання ваги або підтримка поточної ваги) шляхом зручного ведення харчового щоденника, відстеження активності, споживання води.

На етапі проектування можна виділити наступні сутності:

- Користувач (людина, яка використовує мобільний застосунок; характеризується ім’ям, логіном, паролем, вагою, рівнем активності, віком, висотою, статтю, ціллю).
- Продукт (одиниця їжі, що додається до прийому їжі; характеризується ім’ям продукту, грами на порцію, кількістю калорій, грамів білків, жирів і вуглеводів на порцію).
- Харчування (облік прийомів їжі, які поділяються на сніданок, обід, полуденок і вечерю; характеризується користувачем який добавляє, датою, вибором часу прийому їжі, продуктом, який добавляється і вагою яку спожив користувач в грамах).
- Вода (щоденний облік випитої води; характеризується користувачем який добавляє, датою і спожитою водою в мл).
- Щоденний раціон (облік калорій і макронутрієнтів; характеризується користувачем який добавляє, датою, цільовою кількістю калорій, білків, жирів і вуглеводів, кількістю вже спожитих калорій, білків, жирів і вуглеводів, кількістю спожитих калорій за сніданок, обід, полуденок і вечерю).
- Вправа (зберігає інформацію про вправу; характеризується ім’ям вправи, категорією, кількістю повторень і підходів, складністю, описом виконанням і анімацією).
- Історія тренувань (зберігає інформацію про історію тренувань користувача; характеризується користувачем який виконував, датою, вправою яку виконував і часом виконання в секунду).

Проаналізувавши предметну область і додатки, можна сформувати загальні вимоги до розробки додатку:

1. Функціональні вимоги:

- Реєстрація та авторизація користувачів (локально).
- Заповнення onboarding-форм при першому вході (мета, активність, особисті дані).
- Автоматичний розрахунок денних норм калорій і макроелементів.
- Облік харчування за прийомами їжі.

- Можливість додавати нові продукти.
- Облік випитої води.
- Перегляд і запуск тренувань.
- Зберігання історії тренувань.
- Можливість редагування особистих даних та виходу з профілю.

2. Нефункціональні вимоги:

- Зручний і інтуїтивно зрозумілий користувацький інтерфейс (UI), реалізований за допомогою Jetpack Compose.
- Локальне зберігання даних із використанням Room.
- Побудова додатку з урахуванням архітектурної моделі MVVM.
- Плавна навігація між екранами, мінімізація часу відгуку.
- Масштабованість проєкту з можливістю розширення функціоналу.

Таким чином, додаток має забезпечити користувача всіма необхідними інструментами для контролю над власним харчуванням і фізичною активністю, надаючи гнучкий функціонал та простоту у використанні.

1.4 Висновок до першого розділу

В першому розділі кваліфікаційної роботи був проведений огляд існуючих рішень у сфері мобільних додатків для підтримки здорового способу життя, на основі аналізу можна зробити висновок, що попри широкий функціонал, більшість з них мають спільні недоліки: складний або перевантажений інтерфейс, наявність реклами, обмеження у безкоштовній версії. Також був проведений аналіз вибору мови програмування та середовища розробки, що використовується для створення мобільних додатків. Вибір нативного підходу пояснюється потребою у високій продуктивності та глибокій інтеграції з платформою Android. Була сформована постановка задачі – були поставлені вимоги до розробки мобільного додатку та виділені основні сутності, що присутні в мобільному додатку.

РОЗДІЛ 2. РЕАЛІЗАЦІЯ ТА ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ПІДТРИМКИ ЗДОРОВОГО СПОСОБУ ЖИТТЯ

2.1 Архітектура мобільного додатку для підтримки здорового способу життя

Розробники завжди прагнуть досягти чистоти та структурованості коду в проектах. Використання шаблонів проектування сприяє легшій підтримці програмного забезпечення. Коли всі ключові логічні частини програми Android добре організовані, додавати чи видаляти функції стає значно простіше. До того ж, шаблони проектування забезпечують покриття коду модульним тестуванням, виключаючи втручання інших класів.

Одним із таких популярних архітектурних патернів є Model-View-ViewModel (MVVM). Цей підхід пропонує чіткий поділ між логікою представлення даних (View або UI) та бізнес-логікою додатка [8].

Шари MVVM

Модель: Відповідає за абстрагування джерел даних. Вона взаємодіє з ViewModel для отримання і збереження інформації, необхідної для роботи системи.

Представлення (View): Головна задача цього рівня – реагувати на дії користувача та повідомляти про них ViewModel. Представлення стежить за станом ViewModel, не містить прикладної логіки та відповідає лише за відображення.

ViewModel: Слугує посередником між Моделлю та Представленням. Він забезпечує потоки даних, актуальні для відображення, а також координує взаємодію між іншими шарами системи.

На рисунку 2.1 можем побачити візуальне представлення цієї архітектури.

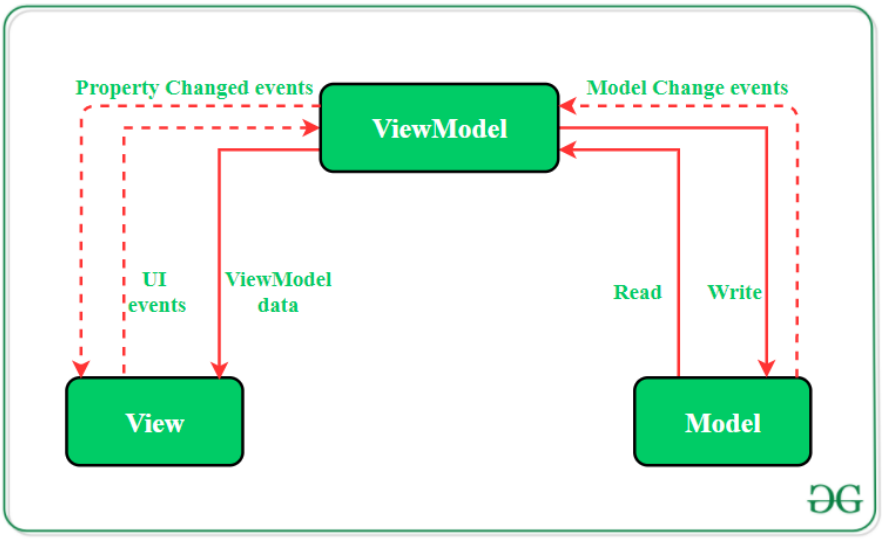

Рисунок 2.1 – Принцип роботи архітектури MVVM

У нашій розробці мобільного додатку для підтримки здорового способу життя ми використаємо спрощений варіант архітектури MVVM – без окремого шару Repository. Уся взаємодія з базою даних реалізується безпосередньо у ViewModel. Такий підхід є прийнятним для застосунків середньої складності та дозволяє зменшити кількість посередницьких шарів.

Основні компоненти архітектури:

- Model – відповідає за роботу з даними. У додатку ця частина реалізована у пакеті data за допомогою бібліотеки Room, яка забезпечує локальне збереження даних у базі SQLite. Вона включає: сутності (Entity) для опису таблиць, зокрема для акаунтів користувачів, даних про воду, калорії, макроелементи тощо; DAO (Data Access Objects) – інтерфейси для доступу до бази даних; AppDatabase – основний клас бази даних, який пов’язує DAO та об'єкти бази.
- ViewModel – проміжний шар між UI та даними. У додатку реалізований у вигляді одного файлу MainViewModel, де зосереджена основна логіка: збереження та обробка особистих даних користувача після проходження onboarding; обчислення денної норми калорій та макроелементів; відстеження прогресу у споживанні води та калорій; зберігання стану інтерфейсу та його зміна у відповідь на дії користувача.

- View – інтерфейс користувача, реалізований у вигляді окремих екранів з використанням Jetpack Compose. Завдяки декларативному підходу Compose, UI автоматично оновлюється при зміні стану у ViewModel.

Компоненти додатку взаємодіють наступним чином [26]: ViewModel безпосередньо звертається до DAO у Room для отримання або збереження даних. ViewModel обробляє отримані дані, обчислює необхідні значення (наприклад, залишок калорій, кількість випитої води) та формує стан (state). View підписується на зміни стану та оновлює UI відповідно до нових даних. У зворотному напрямку, дії користувача (наприклад, натискання кнопки додавання води) передаються до ViewModel для обробки.

Архітектура MVVM була обрана через її здатність забезпечувати модульність і підтримувати реактивний підхід до оновлення інтерфейсу. На рисунку 2.2 можем побачити структуру нашого мобільного додатку.

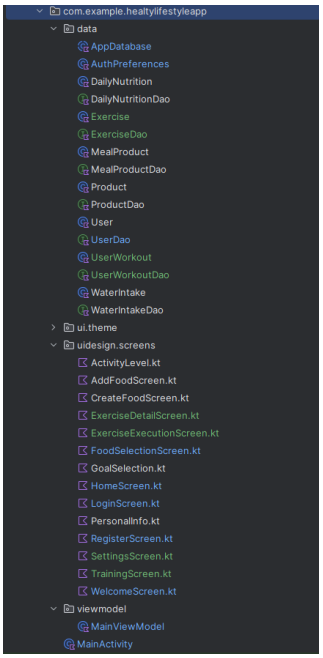

Рисунок 2.2 – Структура мобільного додатку для підтримки здорового способу життя

Обрана структура є достатньо простою та зручною для проєктів середнього масштабу, коли не потрібно впроваджувати складну багаторівневу архітектуру.

Відсутність окремого шару репозиторію компенсується чітким розділенням відповідальностей між папками data, viewmodel і screens.

2.2 Пошук актантів та варіантів використання

Під час розробки мобільного додатку для підтримки здорового способу життя важливим етапом стало визначення основних актантів системи та сценаріїв їхньої взаємодії з додатком. Це дозволяє чітко зрозуміти функціональні можливості продукту та побудувати логічну структуру його майбутньої реалізації [9].

Актантами (або користувачами системи) у контексті даного додатку є:

- Користувач – головний учасник взаємодії з додатком, який має доступ до повного функціоналу, зокрема ведення харчового щоденника, контролю споживання води тощо.
- Система – внутрішній механізм додатку, який забезпечує обчислення показників, збереження даних, оновлення стану інтерфейсу, перевірку введених даних тощо.

На основі виділених актантів було визначено основні варіанти використання, які охоплюють базову функціональність додатку:

1. Реєстрація користувача – введення логіна, пароля та імені для створення нового облікового запису.
2. Авторизація – вхід у додаток із використанням логіна і пароля.
3. Проходження onboarding – вибір цілей (скидання ваги, набір ваги, підтримка ваги), рівня активності, введення особистих даних (вік, стать, вага, зріст).
4. Відображення головного екрану – показ щоденного прогресу користувача: спожиті калорії, випита вода, макроелементи, показ по прийомах їжі.
5. Додавання прийому води – користувач натискає кнопку, щоб зафіксувати спожиту кількість води.

6. Додавання продукту – користувач додає продукти до одного з прийомів їжі (сніданок, обід, полуденок, вечеря).

7. Додавання власного продукту до списку продуктів – користувач, добавляє власний продукт до списку вказуючи назву продукту, розмір порції, кількість калорій, білків, жирів і вуглеводів.

8. Відображення екрану тренувань – показує історію тренувань за певну дату, а також можливість перейти до конкретної вправи.

9. Перегляд вправи – можливість прочитати опис вправи, як правильно її виконувати, а також почати її виконувати.

10. Виконання вправи – запуск секундоміра на час виконання вправи з певною кількістю відпочинків і підходів.

11. Відображення екрану налаштувань – можливість перейти до якихось конкретних налаштувань.

12. Зміна логін і паролю – користувач натискає кнопку, щоб змінити логін і пароль.

13. Зміна персональних даних – користувач натискає кнопку, щоб змінити персональні дані, такі як вага, вік, зріст.

14. Зміна рівня активності – користувач натискає кнопку, щоб змінити рівень активності.

15. Зміна цілі – користувач натискає кнопку, щоб змінити ціль.

16. Вихід з аккаунту – користувач натискає кнопку, щоб вийти з аккаунту.

Для наочного представлення взаємодії актантів із системою було побудовано діаграму варіантів використання [29] за допомогою додатку draw.io [27] – рисунок 2.3.

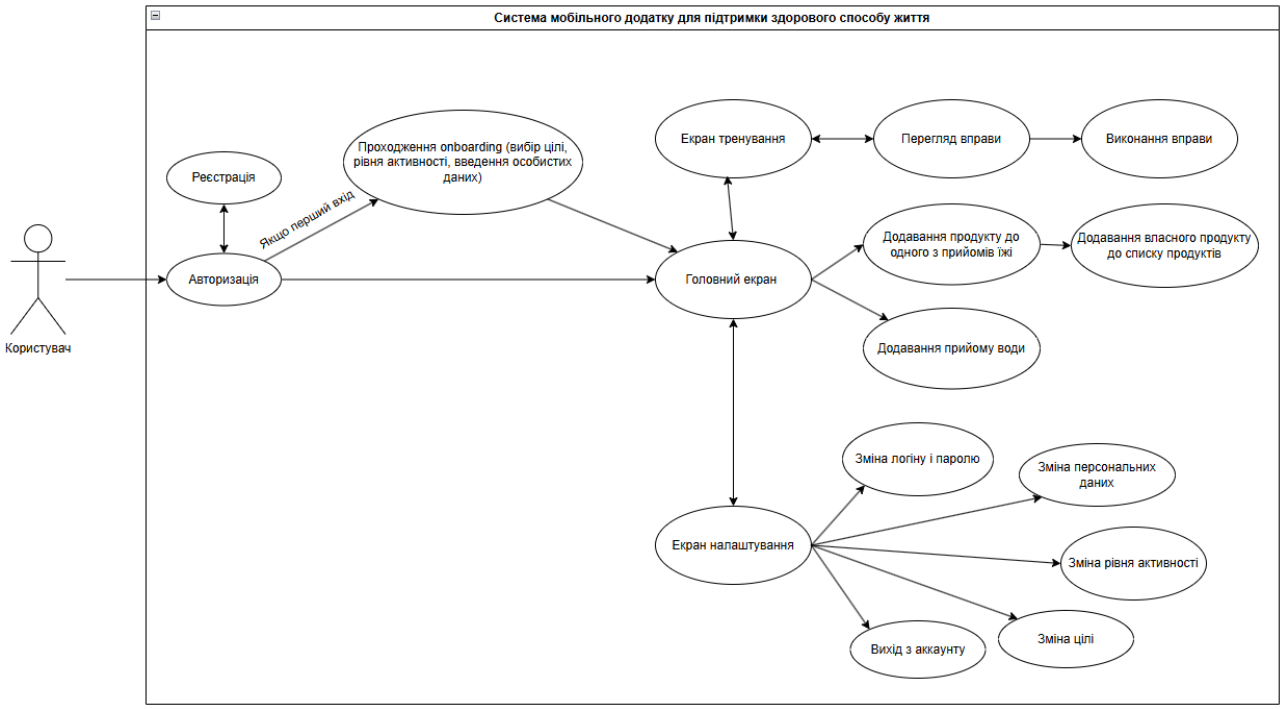

Рисунок 2.3 – Діаграма варіантів використання системи мобільного додатку для підтримки здорового способу життя

Ця діаграма демонструє, як користувач взаємодіє із системою через набір функціональних сценаріїв. Вона дозволяє побачити загальну архітектуру дій та зрозуміти, як реалізована логіка роботи додатку.

2.3 Проектування та реалізація екранів користувацького інтерфейсу мобільного додатку для підтримки здорового способу життя

Користувацький інтерфейс є ключовою складовою мобільного додатку, адже саме через нього відбувається взаємодія користувача з усіма функціональними можливостями системи. У межах реалізації додатку для підтримки здорового способу життя було створено низку екранів, які виконують різні ролі – від збору особистої інформації до щоденного моніторингу активності.

Інтерфейс розроблено з використанням технології Jetpack Compose [33], яка дозволяє будувати сучасні інтерфейси швидко і ефективно, набуває все

більше популярності серед Android-розробників. Використання Compose для створення інтерфейсу користувача є дуже ефективним та менш енергозатратним [10]. Були відокремлені наступні екрани:

- екран авторизації (LoginScreen), за допомогою якого користувач проходить авторизацію;
- екран реєстрації (RegisterScreen), за допомогою якого користувач проходить реєстрацію;
- екран вибору цілі (GoalSelection), при новому аккаунті і першому вході, користувач за допомогою цього екрану обирає ціль: набір, скидання, підтримка ваги;
- екран вибору рівня активності (ActivityLevel), при новому аккаунті і першому вході, користувач за допомогою цього екрану обирає свій рівень активності: низький, середній, високий;
- екран вказання персональних даних (PersonalInfo), при новому аккаунті і першому вході, користувач за допомогою цього екрану вказує свої персональні дані: вік, вага, зріст, стать;
- екран з привітання (WelcomeScreen), який нас зустрічає після заповнення всієї необхідної інформації;
- головний екран (HomeScreen), який виконує роль дашборду щоденної активності (показує кількість спожитих калорій порівняно з денною нормою, кількість спожитої води порівняно з денною нормою, і також показує прийоми їжі за день);
- екран з вибором продуктів (FoodSelectionScreen), який дозволяє добавити споживання продукту до прийому їжі;
- екран добавляння продукту (AddFoodScreen), за допомогою якого вказуємо порцію прийому їжі;
- екран створення нового продукту (CreateFoodScreen), за допомогою якого добавляємо новий продукт до списку продуктів, вказавши: назву продукта, розмір порції, кількість калорій, білків, жирів і вуглеводі;

- екран вкладки тренування (TrainingScreen), який дозволяє переглядати історію тренувань, а також переглянути різні вправи;
- екран з детальною інформацією про вправу (ExerciseDetailScreen), за допомогою якого можем переглянути як виконувати вправу, прочитати опис, а також запустити виконання вправи вказавши час на виконання і час на відпочинок;
- екран з виконання вправи (ExerciseExecutionScreen), за допомогою якого відбувається засікання часу на виконання вправи і часу на відпочинок;
- екран налаштувань (SettingsScreen), за допомогою якого можемо змінити логін і пароль, персональні дані, рівень активності, ціль, а також вийти з аккаунту.

Для переміщення між цими екранами використано систему навігації Jetpack Compose Navigation [11]. Вона дозволяє будувати декларативну навігацію між екранними компонентами, використовуючи єдиний NavHost, у якому визначені маршрути (routes) для кожного екрану. Навігація реалізується за допомогою NavController, що забезпечує перехід між екранами (наприклад, navController.navigate("HomeScreen")) та повернення назад (navController.popBackStack()). Це дозволяє гнучко керувати навігаційним стеком і зберігати стан інтерфейсу, забезпечуючи плавну і послідовну взаємодію користувача з додатком.

Домашнім (тобто першим при запуску) екраном встановлений екран авторизації (LoginScreen), або головний екран (HomeScreen), залежно від того чи користувач виходив з аккаунту, чи ні. Користувач також може зареєструватись перейшовши з екрану авторизації на екран реєстрації (RegisterScreen). Якщо користувач перший раз заходить в додаток, то він повинен пройти екрани з вибором цілі (GoalSelection), рівня активності (ActivityLevel), вказання персональних даних (PersonalInfo) і екран привітання (WelcomeScreen), після цього він вже тоді потрапить на головний екран.

Після успішної авторизації користувач потрапляє на головний екран (HomeScreen), який виконує роль центрального навігаційного вузла. З нього

доступні переходи до всіх інших функціональних частин додатку. З головного екрану користувач може перейти до екрану з вибором продуктів (FoodSelectionScreen), з якого також можна перейти до екрану добавляння продукту (AddFoodScreen), або до екрану створення нового продукту (CreateFoodScreen). Також, з головного екрану користувач може перейти до екрану тренування (TrainingScreen), або екрану налаштувань (SettingsScreen) через нижню навігацію, присутньою на всіх трьох екранах.

На екрані тренувань користувач може перейти до екрану з детальною інформацією про вправу (ExerciseDetailScreen), з якого може перейти до екрану виконання вправи (ExerciseExecutionScreen).

2.4 Проектування бази даних мобільного додатку для підтримки здорового способу життя

У розробці мобільного застосунку для підтримки здорового способу життя було використано Room – офіційну бібліотеку від Google, що надає обгортку над SQLite та забезпечує типобезпечний доступ до локальної бази даних [34].

Room – це бібліотека, яка значно спрощує взаємодію з базою даних Android SQLite, надаючи більш високий рівень абстракції. Використання цієї бібліотеки дозволяє зменшити кількість коду, зробити його лаконічнішим і полегшити роботу розробника.

Завдяки Room вам не потрібно витрачати час на опис базових операцій, таких як SQL-запити, всередині програми. Достатньо інтегрувати потрібні компоненти цієї бібліотеки – і ви отримаєте готові інструменти, з якими розробка стане швидшою та зручнішою.

Єдина ситуація, коли варто утриматися від використання Room, – це робота з великим існуючим проєктом. У таких випадках витрати на міграцію можуть перевищувати вигоди від переходу, тому краще залишитися з поточною реалізацією.

Без Room пряме використання SQLite займає значно більше часу як на написання коду, так і на пошук помилок. Ця розробка Google вважається одним із найкращих підходів у роботі з базами даних на Android і настійливо рекомендується для більшості проєктів [12].

На рисунку 2.4 наведено принцип роботи бібліотеки Room [28].

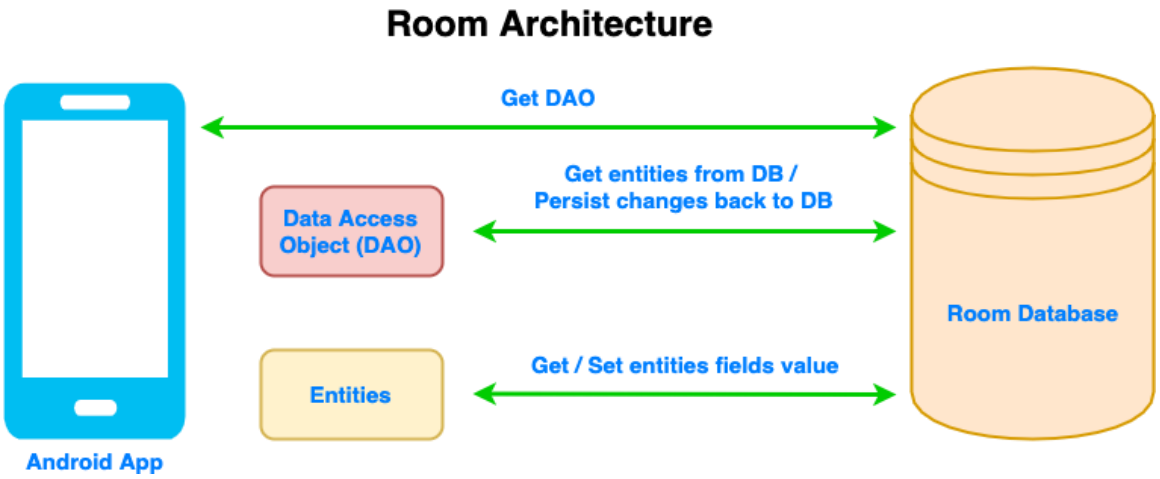

Рисунок 2.4 – Принцип роботи бібліотеки Room

У нашому застосунку реалізовано такі сутності (таблиці):

- User – зберігає інформацію про користувача.
- WaterIntake – облік спожитої води.
- DailyNutrition – щоденна статистика спожитих калорій та макроелементів.
- Product – база продуктів харчування.
- MealProduct – зв’язок між продуктами та прийомами їжі.
- Exercise – вправа з описом, повтореннями, категорією.
- UserWorkout – збережені тренування користувача.

Схему бази даних визначено в класі AppDatabase, який успадковується від RoomDatabase (див. додаток А).

При першому запуску застосунку в onCreate() додаються вправи до таблиці Exercise (див. додаток А). Це дозволяє відразу надати користувачу готовий список фізичних вправ без потреби підключення до інтернету.

Одним із ключових елементів застосунку є облік спожитих калорій, білків, жирів та вуглеводів. Для цього реалізовано сутність DailyNutrition, яка містить цільові та фактичні показники на день (див. додаток А).

Кожен запис відповідає окремому дню користувача та дозволяє вести облік усіх необхідних макронутрієнтів.

Кожна сутність має відповідний DAO-інтерфейс (наприклад, DailyNutritionDao), який містить функції для додавання, оновлення та отримання даних (див. додаток А).

Для зберігання облікових даних (логін, пароль, ID користувача) використовується клас AuthPreferences, який працює з SharedPreferences. Це дозволяє забезпечити швидкий доступ до сесії без необхідності запитувати логін щоразу при запуску.

Загалом, ознайомитись з сутностями бази даних можна на рисунку 2.5, на якому можна побачити ER-діаграму БД створену за допомого сервісу Mermaid [13].

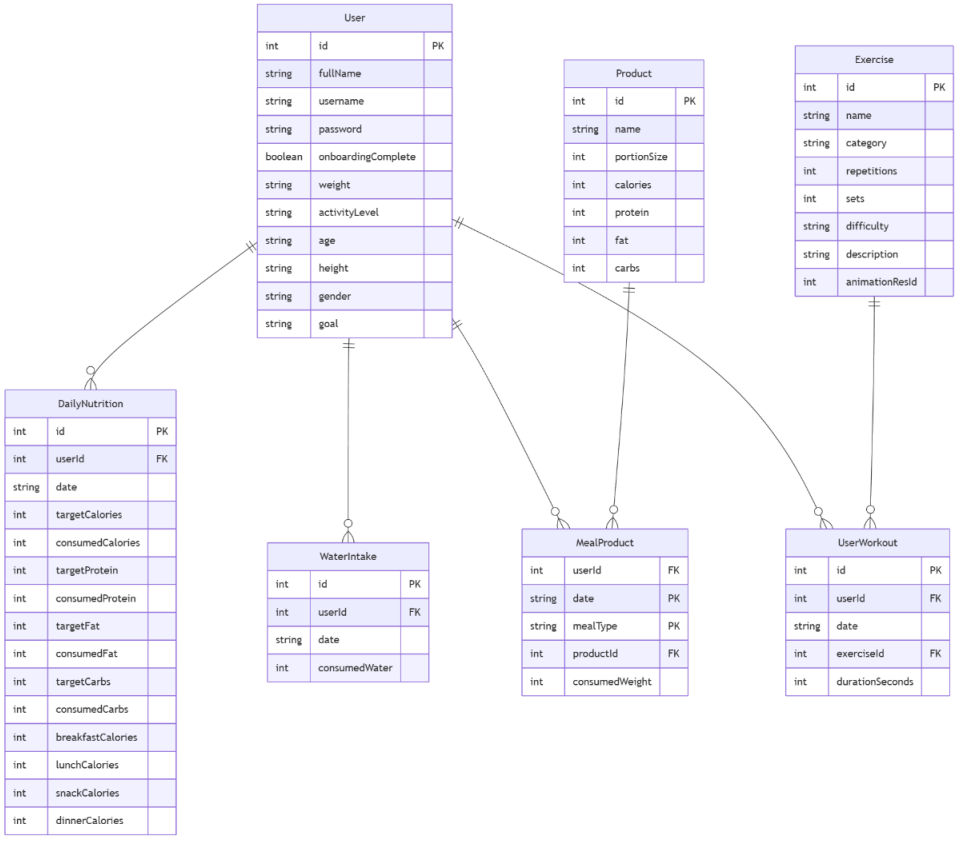

Рисунок 2.5 – ER-діаграма бази даних

Зв’язки між сутностями:

- User → DailyNutrition: Один до багатьох (1:N). Один користувач може мати багато записів про харчування (userId у DailyNutrition посилається на User.id).
- User → WaterIntake: Один до багатьох (1:N). Один користувач може мати багато записів про споживання води (userId у WaterIntake посилається на User.id).
- User → MealProduct: Один до багатьох (1:N). Один користувач може мати багато записів про продукти в прийомах їжі (userId у MealProduct посилається на User.id).
- Product → MealProduct: Один до багатьох (1:N). Один продукт може бути пов’язаний з багатьма записами в прийомах їжі (productId у MealProduct посилається на Product.id з каскадним видаленням).
- User → UserWorkout: Один до багатьох (1:N). Один користувач може мати багато записів про тренування (userId у UserWorkout посилається на User.id).
- Exercise → UserWorkout: Один до багатьох (1:N). Одна вправа може бути використана в багатьох тренуваннях (exerciseId у UserWorkout посилається на Exercise.id).

Загалом, використання Room у мобільному додатку для підтримки здорового способу життя дозволило створити стабільну, зручну та розширювану локальну базу даних. Така реалізація забезпечує ефективне зберігання персональних даних, харчових показників, тренувань , обліку води, тощо, що є ключовими компонентами функціональності застосунку для підтримки здорового способу життя.

2.5 Реалізація логіки додатку

Як вже згадувалося раніше, у рамках реалізації мобільного додатку було обрано архітектурний підхід MVVM (Model-View-ViewModel).

У межах цього архітектурного підходу вся логіка додатку зосереджена у класі MainViewModel, який є підкласом AndroidViewModel. Він виступає у ролі єдиного центру обробки подій, зберігання станів та комунікації з базою даних. На відміну від класичної реалізації MVVM, де часто використовується окремий шар Repository, то у нашому мобільному додатку доступ до DAO-інтерфейсів Room здійснюється безпосередньо з ViewModel. Такий підхід дозволив спростити структуру додатку та скоротити обсяг коду на ранніх етапах розробки.

Для зберігання змінних стану у ViewModel активно використовуються обгортки mutableStateOf та mutableStateListOf, які є частиною API бібліотеки Jetpack Compose [14]. Вони дозволяють реактивно оновлювати інтерфейс користувача при зміні даних.

- mutableStateOf(value) – обгортка для простих змінних типу String, Int, Boolean, які автоматично викликають перекомпозицію в Compose, якщо їх значення змінюється.
- mutableStateListOf<T>() – обгортка для списків, яка забезпечує відстеження змін (додавання, видалення, оновлення елементів) у реактивний спосіб.

Таким чином, будь-яка зміна значення у ViewModel миттєво відображається в UI без необхідності ручного оновлення.

Перейдем до основних компонентів логіки. Одним із перших етапів взаємодії користувача з додатком є реєстрація та вхід. Методи registerUser() та loginUser(.) (див. додаток Б) відповідають за додавання нового користувача та перевірку його облікових даних. Під час входу відбувається ініціалізація змінних стану користувача, включаючи його ціль, рівень активності, фізичні характеристики тощо. У випадку, якщо користувач уже пройшов онбординг, з бази завантажуються актуальні дані щодо його денних норм харчування та тренувань.

У процесі початкового налаштування користувач вводить свою мету, рівень активності, вік, вагу, зріст і стать. Всі ці параметри зберігаються у базу даних через відповідні методи setUserGoal() (див. додаток Б),

setUserActivityLevel() та setUserPersonalInfo(). Після завершення онбордингу викликається метод setOnboardingCompleted(), що оновлює статус користувача і запускає логіку розрахунку добових цілей.

Метод calculateNutritionTargets() (див. додаток Б) розраховує базовий метаболізм (BMR) за формулою Міффліна-Сан Жеора [15], враховуючи стать користувача (для чоловіків див. формулу 2.1, для жінок див. формулу 2.2) . На основі рівня активності застосовується відповідний множник, а цільові калорії коригуються залежно від мети користувача (скидання, набір або підтримка ваги). Макронутрієнти розподіляються у відсотковому співвідношенні, яке залежить від цілі, а результати зберігаються у реактивних станах для відображення в інтерфейсі.

$$\text{БРМ} = 10\ W\,\frac{\text{вага}}{\text{кг}} + 6{,}25\ H\,\frac{\text{ріст}}{\text{см}} - 5\ \text{А}\,\frac{\text{вік}}{\text{роки}} + 5 \tag{2.1}$$

$$\text{БРМ} = 10\ W\,\frac{\text{вага}}{\text{кг}} + 6{,}25\ H\,\frac{\text{ріст}}{\text{см}} - 5\ \text{А}\,\frac{\text{вік}}{\text{роки}} - 161 \tag{2.2}$$

W – вага тіла у кілограмах, Н – ріст у сантиметрах, А – вік.

Облік води реалізовано через loadWaterIntakeForDate() та addWaterIntake(amount: Int) (див. додаток Б). Цільовий обсяг води розраховується на основі ваги користувача (35 мл/кг) з урахуванням рівня активності. Наприклад, для середнього рівня активності додається 10% до базового обсягу, а для високого – 20%. Дані зберігаються у таблиці WaterIntake, а значення кількості води оновлюється в інтерфейсі користувача в режимі реального часу.

Зміна дати здійснюється через метод setDate() (див. додаток Б). Це дозволяє користувачу переглядати історію прийомів їжі, води та тренувань за минулі дні. При зміні дати повторно викликаються методи для завантаження актуальних даних з бази.

Для управління харчуванням метод addMealProduct дозволяє додавати продукти до прийомів їжі, обчислюючи калорії та макронутрієнти на основі спожитої ваги продукту. Тренування обробляються через методи loadExercisesForCategory та addUserWorkout, які дозволяють завантажувати вправи за категоріями та зберігати дані про тренування користувача.

У результаті, завдяки використанню архітектури MVVM та реактивного підходу Jetpack Compose, логіка додатку була реалізована у структурованій, масштабованій та зручній для обслуговування формі. Централізація логіки у MainViewModel дозволила ефективно координувати взаємодію між UI, базою даних та бізнес-логікою, забезпечуючи при цьому плавну та інтуїтивну взаємодію користувача з інтерфейсом. Такий підхід також створює хорошу основу для подальшого розвитку додатку, розширення функціональності та інтеграції нових сервісів.

2.6 Висновки до другого розділу

У другому розділі кваліфікаційної роботи було детально розглянуто процес проєктування та реалізації мобільного додатку для підтримки здорового способу життя.

У результаті аналізу архітектурних підходів для Android-додатків було обґрунтовано вибір шаблону MVVM. У реалізації проєкту застосовано спрощену версію MVVM без шару репозиторію, що дозволило зменшити складність структури, зберігши водночас основні переваги архітектури.

Під час етапу проєктування були визначені основні актанти системи та варіанти використання, що стало основою для формування функціональних вимог до додатку. Ці вимоги були реалізовані у вигляді логіки, зосередженої у ViewModel, що координує роботу інтерфейсу з даними.

Користувацький інтерфейс розроблено з використанням сучасної технології Jetpack Compose. Було створено серію екранів, що охоплюють усі

основні сценарії взаємодії. Завдяки використанню Compose, інтерфейс є гнучким, адаптивним та швидко реагує на зміни у стані ViewModel.

Для зберігання та обробки локальних даних використано бібліотеку Room, яка надала ефективні засоби для роботи з базою даних на основі SQLite. Структура бази була реалізована з використанням DAO-інтерфейсів та центрального класу AppDatabase, що забезпечило об’єднання усіх компонентів бази даних, забезпечуючи їх ефективну взаємодію.

Загалом, другий розділ демонструє цілісну реалізацію мобільного додатку – від вибору архітектури до реалізації функціональної логіки та інтуїтивного інтерфейсу. У результаті створено зручний, функціональний і ефективний додаток, який відповідає потребам користувачів у підтримці здорового способу життя та має потенціал для подальшого масштабування та вдосконалення.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА. ДЕМОНСТРАЦІЯ ТА ТЕСТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Демонстрація роботи мобільного додатку для підтримки здорового способу життя

Розроблений мобільний додаток для підтримки здорового способу життя реалізовано для платформи Android з використанням мови програмування Kotlin, середовища Android Studio та бібліотеки Jetpack Compose для створення інтерфейсу. Для демонстрації роботи додатку розглянемо ключові функціональні можливості через послідовність взаємодії користувача з системою. Демонстрація проведена на Android Emulator(Pixel 7, API 34). Емулятор Android дозволяє відтворювати роботу пристроїв Android на вашому комп'ютері, що дає змогу тестувати власні додатки на різних моделях пристроїв і версіях API Android, не маючи потреби в реальному обладнанні для кожного варіанту [16].

При запуску додатку користувач потрапляє на екран авторизації (LoginScreen). Якщо обліковий запис відсутній, користувач може перейти на екран реєстрації (RegisterScreen), де вводить ім’я, логін та пароль. Після успішної реєстрації дані зберігаються локально в базі даних за допомогою Room та SharedPreferences (для збереження сесії). На рисунку 3.1 зображено екран авторизації та реєстрації.

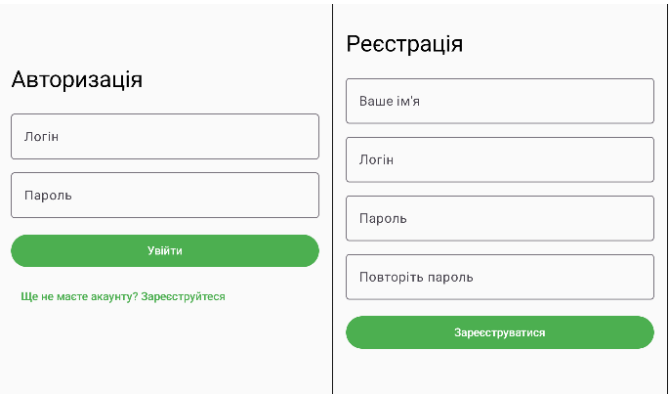

Рисунок 3.1 – Екрани авторизації та реєєстрації

Для нових користувачів після реєстрації запускається процес onboarding, який включає:

- Вибір мети (GoalSelection): скидання ваги, набір ваги або підтримка ваги.
- Вибір рівня активності (ActivityLevel): низький, середній, високий.
- Введення особистих даних (PersonalInfo): вік, вага, зріст, стать.

Після завершення onboarding користувач потрапляє на екран привітання (WelcomeScreen), а введені дані використовуються для автоматичного розрахунку денних норм калорій та макронутрієнтів за формулою Міффліна-Сан Жеора. Onboarding це свого роду зустріч нового користувача додатку [17]. На рисунку 3.2 показано екрани вибору мети, рівня активності, введення особистих даних та екрану привітання.

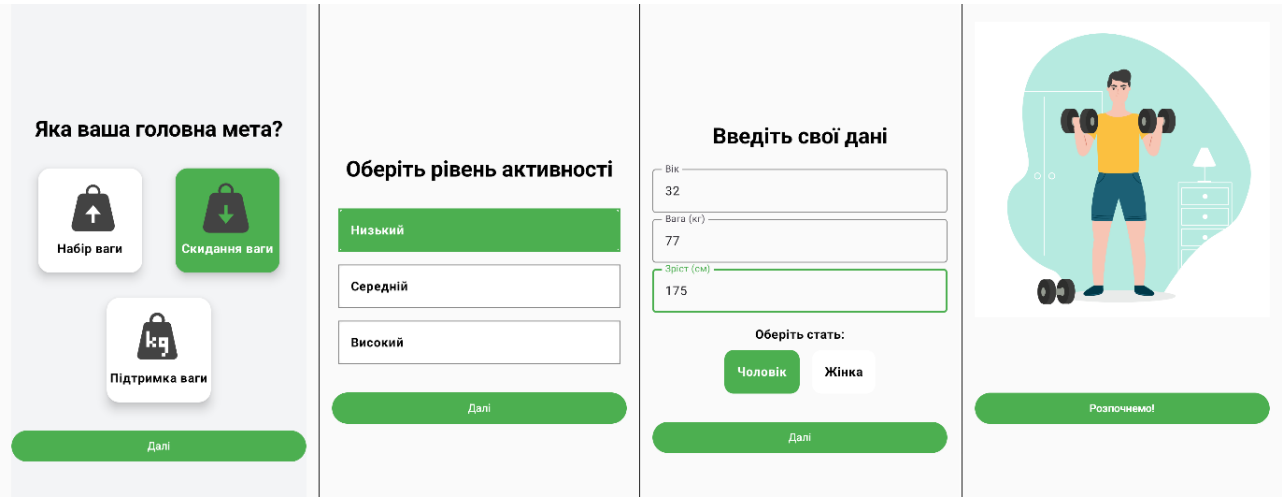

Рисунок 3.2 – Екрани вибору мети, рівня активності, введення особистих даних та привітання

Після завершення onboarding користувач потрапляє на головний екран додатку. Центральний екран додатку містить інформацію про щоденну активність користувача:

- Кільце калорій: візуальний елемент у формі кільця, що поступово заповнюється в залежності від кількості спожитих калорій. У центрі кільця вказано денну норму калорій.

- Статистика макронутрієнтів: під кільцем розміщено інформацію про залишкову кількість білків, жирів та вуглеводів, яку потрібно спожити протягом дня.
- Водний баланс: реалізована можливість фіксації кількості випитої води порівняно з нормою.
- Прийоми їжі: користувач може додавати їжу до кожного з чотирьох прийомів їжі: сніданок, обід, полуденок і вечеря. Для цього передбачено кнопку "+", яка відкриває вибір типу прийому їжі та переадресовує на екран додавання продуктів (FoodSelectionScreen).
- Нижня навігація: забезпечує швидкий перехід між головним екраном, тренуванням та налаштуваннями.

На рисунку 3.3 зображено головний екран.

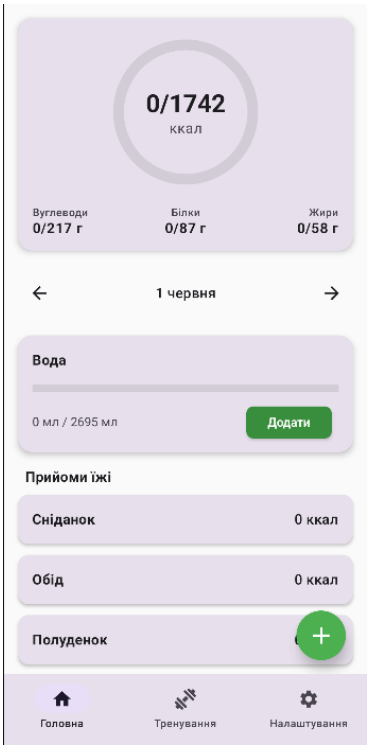

Рисунок 3.3 – Головний екран мобільного додатку

На екрані FoodSelectionScreen користувач обирає продукти з бази даних або додає власний продукт через CreateFoodScreen, вказуючи назву, розмір порції, калорії та макронутрієнти. Після вибору продукту на екрані

AddFoodScreen користувач вказує вагу порції, а система автоматично розраховує калорії та макронутрієнти, додаючи їх до відповідного прийому їжі. Дані оновлюються в таблиці DailyNutrition, а інтерфейс реагує миттєво завдяки реактивному підходу Jetpack Compose. На рисунку 3.4 показано екрани FoodSelectionScreen, CreateFoodScreen, AddFoodScreen.

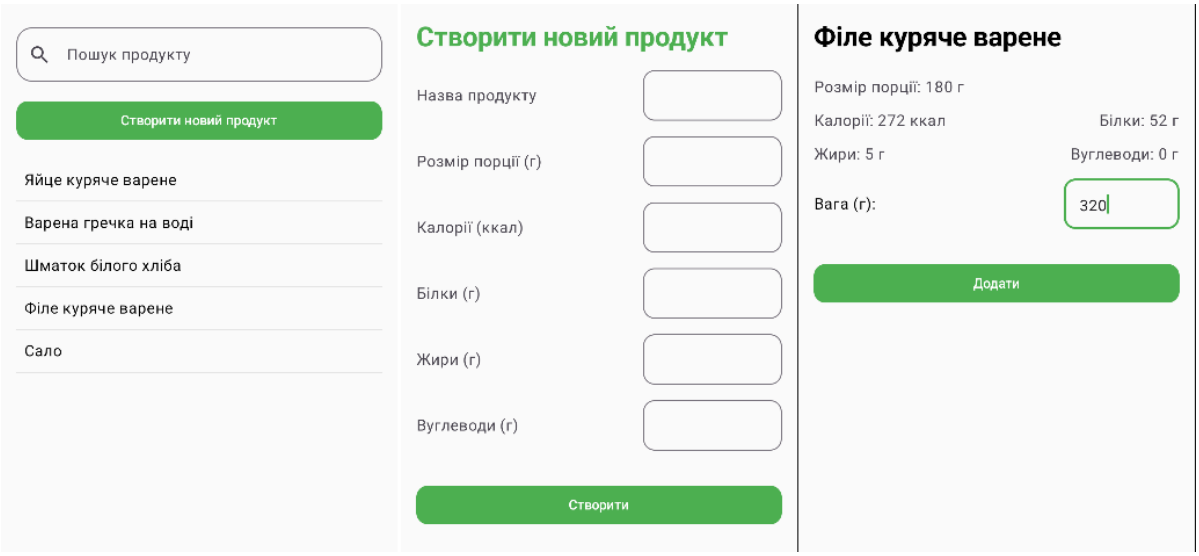

Рисунок 3.4 – Екрани FoodSelectionScreen, CreateFoodScreen та AddFoodScreen

Також, на рисунку 3.5 можем побачити процес добавляння прийому води.

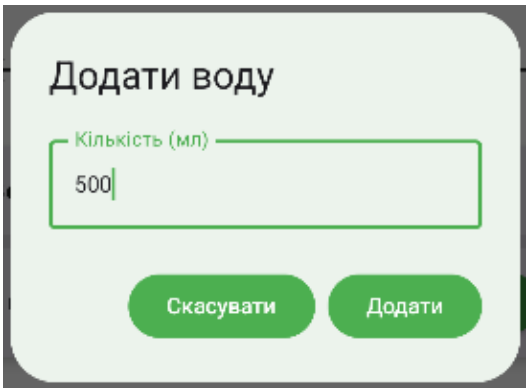

Рисунок 3.5 – Процес добавляння прийому води

Після внесення всіх цих змін, на рисунку 3.6 можем побачити як змінився наш головний екран.

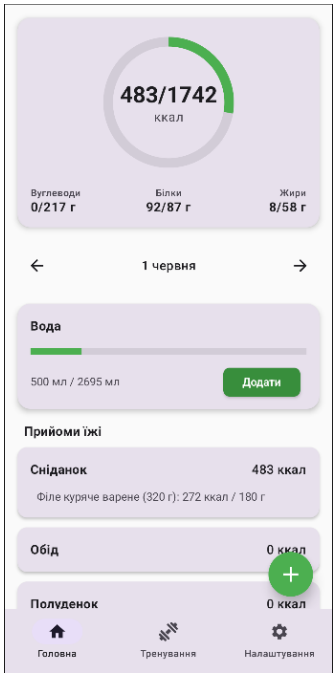

Рисунок 3.6 – Результат зміни головного екрану після добавлення прийому їжі і води

На екрані SettingsScreen користувач може змінювати логін, пароль, особисті дані, рівень активності, ціль або вийти з облікового запису. Зміни миттєво синхронізуються з базою даних через методи ViewModel. На рисунку 3.7 можна побачити екран налаштувань і всі варіанти зміни налаштувань.

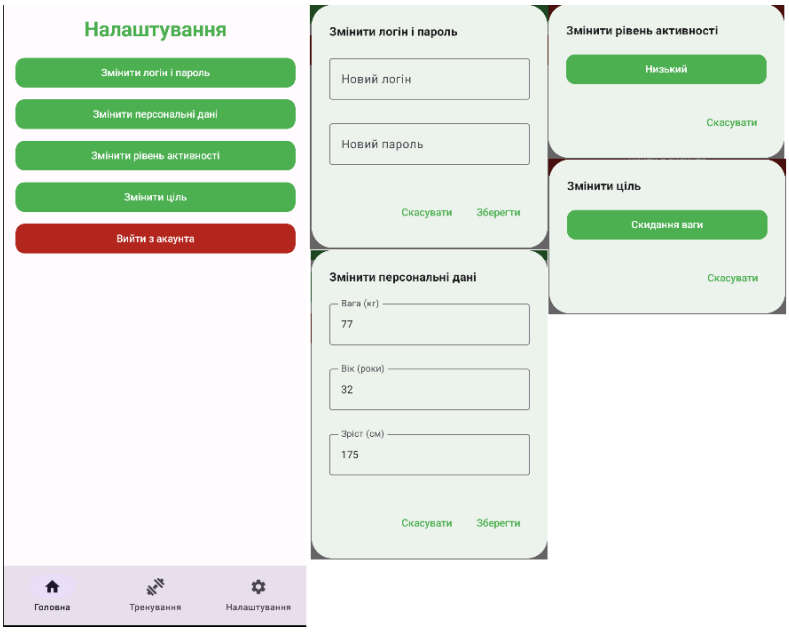

Рисунок 3.7 – Екран налаштувань та всі варіанти зміни налаштувань

Екран TrainingScreen дозволяє переглядати історію тренувань та доступні вправи. Користувач може вибрати вправу, переглянути її опис та анімацію на ExerciseDetailScreen, а також запустити виконання вправи через ExerciseExecutionScreen, де відображається таймер для виконання та відпочинку. Дані про тренування зберігаються в таблиці UserWorkout. На рисунку 3.8 зображено екрани TrainingScreen, ExerciseDetailScreen та ExerciseExecutionScreen.

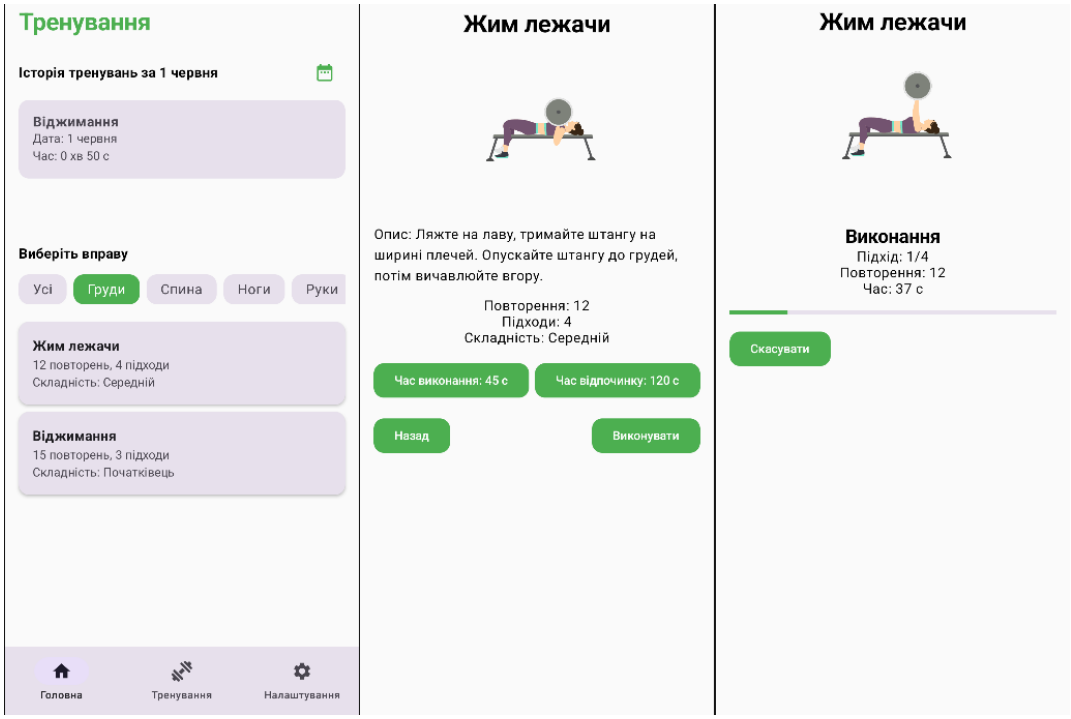

Рисунок 3.8 – Екрани TrainingScreen, ExerciseDetailScreen та ExerciseExecutionScreen

Демонстрація роботи додатку на емуляторі показала стабільну роботу всіх функцій, швидке оновлення інтерфейсу та коректне збереження даних.

3.2 Тестування та валідація мобільного додатку для підтримки здорового способу життя

Для перевірки коду мобільного додатка для підтримки здорового способу життя, було використано інтегроване середовище розробки Android Studio, яке

рекомендовано для створення мобільних застосунків і було зазначене у першому розділі. Процес валідації проводився за допомогою утиліти Code Inspection [18], як представлено на рисунку 3.9.

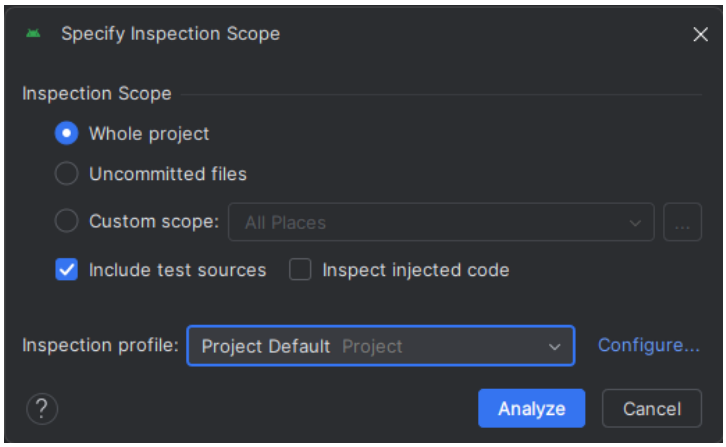

Рисунок 3.9 – Запуск утиліти Code Inspection

Усього було виявлено помилки, які можна класифікувати за п’ятьма категоріями: Android, JVM (Java Virtual Machine), Java, Kotlin і Proofreading. Остання категорія містить інформацію про типографічні помилки, як показано на рисунку 3.10, майже всі вони пов’язані з українськими словами, яких немає в англійській мові, що призводить до відповідних попереджень про незрозумілі назви змінних. Тому цю категорію можна пропустити.

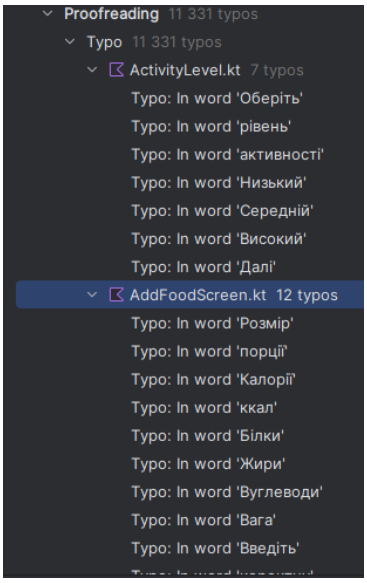

Рисунок 3.10 – Категорія поперджен Proofreading

Перша група, наведена на рисунку 3.11, включає попередження для стандартних Android-файлів, таких як конфігурації та ресурси.

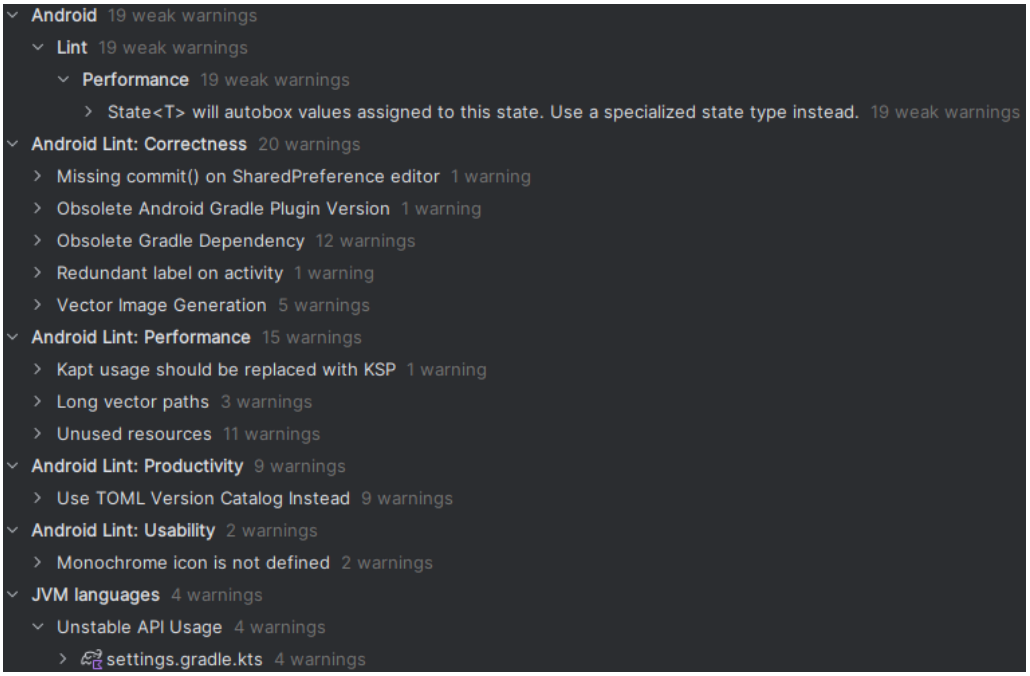

Рисунок 3.11 – Категорія попереджень Android

Був проведений детальний розгляд попереджень:

- State <T> will autobox values assigned to this state. Use a specialized type instead – Попередження вказує на автобоксинг значень у State<t>, що може знизити продуктивність. Попередження було пропущено, через те що автобоксинг не впливає на продуктивність у поточному контексті через низький обсяг даних.
- Missing commit() on SharedPreference editor – Відсутній виклик commit() для збереження даних у SharedPreferences, що може призвести до втрати даних. Попередження було пропущено, через те що використовується apply() для асинхронного збереження, що є достатнім.
- Obsolete Android Gradle Plugin Version – Версія плагіну Android Gradle застаріла. Попередження було пропущено, через те що поточна версія забезпечує стабільність і сумісність.

- Obsolete Gradle Dependency – Застосовуються застарілі залежності Gradle. Попередження було пропущено, через те що залежності відповідають вимогам проекту.
- Redundant label on activity – Надлишкові мітки в маніфесті для активностей. Попередження було пропущено, через те що мітки не впливають на функціональність.
- Vector Image Generation – Проблеми з генерацією векторних зображень, що можуть збільшувати розмір APK. Попередження було пропущено, через те що зображення відповідають вимогам дизайну.
- Kapt usage should be replaced with KSP – Kapt повільніший за KSP для обробки анотацій. Попередження було пропущено, через те що Kapt достатній для поточних потреб проекту.
- Long vector paths – Довгі векторні шляхи у графіці можуть сповільнити рендеринг. Попередження було пропущено, через те що векторні зображення оптимізовані для дизайну.
- Unused resources – Наявність невикористовуваних ресурсів у проекті. Було виправлено і позбулися невикористовуваних ресурсів.
- Use TOML Version Catalog Instead – Рекомендується використовувати TOML для управління залежностями. Попередження було пропущено, через те що поточне управління залежностями ефективне.
- Monochrome icon is not defined – Відсутня монохромна іконка для адаптивних іконок Android. Попередження було пропущено, через те що адаптивні іконки не потрібні для цільової версії.

Наступні дві групи, які розглянули стосуються Java та JVM, показані на рисунку 3.12.

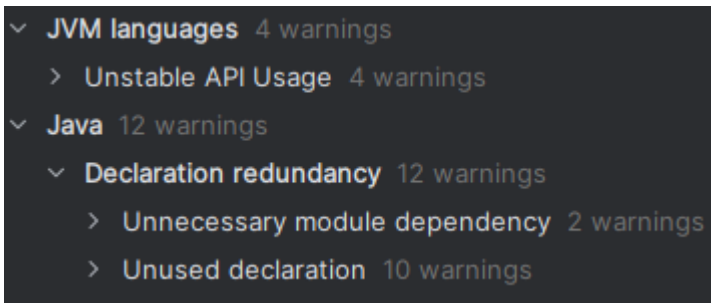

Рисунок 3.12 – Категорії поперджень для Java та JVM

Були розглянуті попередження для Java та JVM:

- Unstable API Usage – Використання нестабільних API, які можуть змінитися. Попередження було пропущено, через те що API необхідні для реалізації ключової функціональності.
- Unnecessary module dependency – Надлишкові залежності модулів у проекті. Попередження було пропущено, через те що залежності залишені для майбутньої розширюваності.
- Unused declaration – Наявність невикористовуваних декларацій у коді. Попередження було пропущено, через те що декларації залишені для потенційного використання.

І останню групу, яку було розглянуто стосується мови програмування Kotlin. Попередження щодо цієї групи можна побачити на рисунку 3.13.

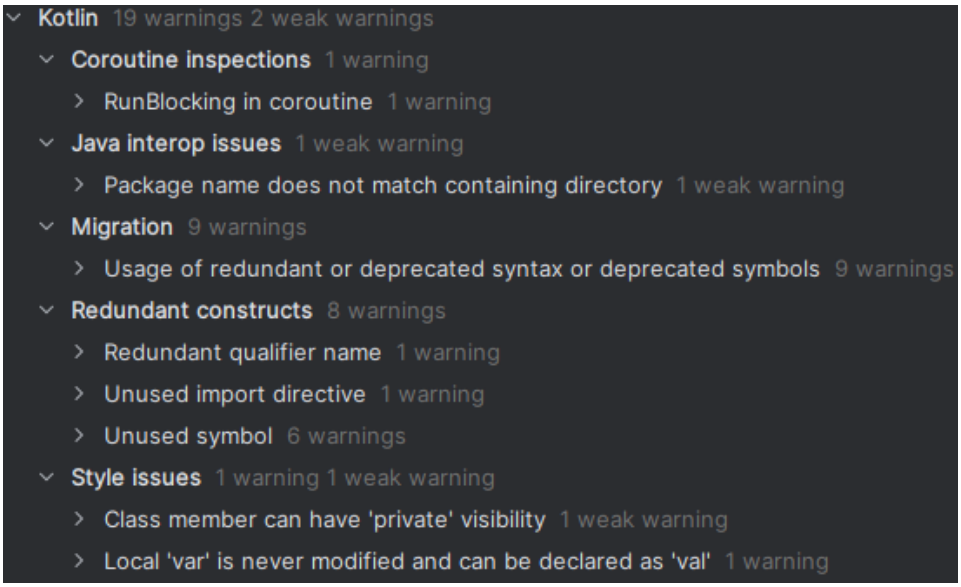

Рисунок 3.13 – Категорія попереджень для Kotlin

Були розглянуті попередження для Kotlin:

- RunBlocking in coroutine – Використання runBlocking у корутинах може блокувати основний потік. Попередження було пропущено, через те що runBlocking застосовується в специфічних сценаріях тестування.
- Package name does not match containing directory – Назва пакету не відповідає структурі директорій. Попередження було пропущено, через те що структура пакету відповідає логіці проекту.
- Usage of redundant or deprecated syntax or deprecated symbols – Використання застарілого або надлишкового синтаксису. Попередження було пропущено, через те що синтаксис не впливає на функціональність.
- Redundant qualifier name – Надлишкові кваліфікатори в коді. Попередження було пропущено, через те що надлишковість не критична для зрозумілості.
- Unused import directive – Наявність невикористовуваних імпортів. Попередження було пропущено, через те що імпорти не впливають на продуктивність.
- Unused symbol – Наявність невикористовуваних символів у коді. Попередження було пропущено, через те що символи залишені для майбутньої розширюваності.
- Class member can have 'private' visibility – Члени класів можуть бути приватними. Попередження було пропущено, через те що ширша видимість потрібна для інтеграції з іншими компонентами.
- Local 'var' is never modified and can be declared as 'val' – Локальні змінні var не змінюються і можуть бути val – виправлено.

Таким чином, було здійснено ретельну перевірку як технічної складової, так і стилістичних аспектів реалізації. Основні зауваження були виправлені або проаналізовані з відповідним обґрунтуванням. Це дозволило підвищити стабільність, підтримуваність і відповідність проєкту сучасним стандартам розробки для Android.

3.3 Висновки до третього розділу

У цьому розділі було продемонстровано роботу розробленого мобільного додатку для підтримки здорового способу життя. Було показано ключові екрани та функціональні можливості, зокрема процес реєстрації та авторизації користувача, проходження onboarding, розрахунок денних норм споживання калорій і макронутрієнтів, ведення обліку прийомів їжі та води, а також перегляд і виконання вправ. Інтерфейс додатку реалізовано за допомогою Jetpack Compose, що забезпечило сучасний і динамічний вигляд, а збереження даних організовано через Room та SharedPreferences.

Крім демонстрації функціональності, було проведено тестування додатку за допомогою інструментів Android Studio. Результати аналізу коду показали відсутність критичних помилок, що підтверджує стабільну роботу додатку, коректне збереження даних і швидке реагування інтерфейсу на дії користувача. Отже, мобільний додаток успішно реалізує поставлені функціональні вимоги та готовий до використання на платформі Android.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при ушкоджені м’яких тканин, суглобів і кісток

Щодня росія здійснює терористичні дії обстрілюючи українські міста ракетами та артилерією, навмисно знищуючи місцеве населення та критичну інфраструктуру. Кількість жертв серед мирного українського населення через терористичні дії російських військ продовжує зростати. Тому сьогодні вкрай важливо оволодіти елементарними навичками надання першої медичної допомоги для порятунку життя [19].

У сучасних умовах війни, питання надання долікарської допомоги при ушкодженнях м’яких тканин, суглобів і кісток набуває особливого значення. Травми можуть виникати внаслідок обстрілів, вибухів, завалів будівель і бойових дій. Долікарська допомога при ушкодженні м’яких тканин, суглобів і кісток полягає в розпізнаванні характерних ознак та наданні першої допомоги. Серед основних симптомів – припухлість у місці крововиливу, яка швидко прогресує, болючість при пальпації та рухах, обмеження рухів і зниження функцій організму. На місце гематоми накладають холод, тугу пов’язку та здійснюють транспортну іммобілізацію.

Вивихи – це ушкодження, за яких відбувається зміщення суглобних кінців кісток у суглобах.

Розтягнення і розрив зв'язок проявляються припухлістю і рухом у неприродному для суглоба напрямку.

Стискання виникає внаслідок тиску великої ваги на організм людини, що часто виникає під завалами чи уламками будівель, що призводить до розчавлення м’язів, підшкірної жирової клітковини, судин і нервів.

Ушкодження хребта трапляються при вибухах, надмірному згинанні або розгинанні, падінні з висоти, пірнанні в мілку воду, автомобільних катастрофах і обвалах. Переломи можуть бути відкритими і закритими, з ушкодженням або

без ушкодження спинного мозку. Вони супроводжуються припухлістю, локальним болем, можливо – деформацією хребта. При ушкодженні спинного мозку настає часткове або повне переривання його провідності, що викликає втрату чутливості нижче місця ураження, паралічі та розлади функцій тазових органів. Потерпілого транспортують дуже обережно, на щиті, в положенні лежачи на спині, підкладаючи м'які валики під плечі та голову. Для підняття постраждалого необхідна участь щонайменше трьох осіб [20].

Ушкодження таза (часто виникають при здавленні уламками будівель чи вибухах) включають травми м'яких тканин, переломи тазових кісток та ушкодження внутрішніх органів – сечового міхура, уретри, прямої кишки. Переломи виникають переважно при падінні з великої висоти, обвалах, аваріях. Вони бувають неускладнені (без ушкодження органів) та ускладнені. Ізольовані переломи часто є наслідком прямого удару. У мирний час переважають закриті переломи, які можуть супроводжуватись руйнуванням вертлюжної западини або центральним вивихом стегна. Подвійні переломи Мальгеня характеризуються переломами спереду (обох лобкових кісток) і ззаду (кубової кістки). Вогнепальні переломи належать до важких, часто супроводжуються крововтратою, травмами органів, множинними уламками. Діагноз встановлюють за анамнезом, механізмом травми та обстеженням. Ознаки включають локальний біль, гематому, деформацію, характерне положення потерпілого ("жаба"): ноги розведені і зігнуті, самостійно підняти їх він не може (симптом прилиплої п'ятки).

Струс головного мозку, поширений серед постраждалих від вибухових хвиль, відбувається при травмах черепа і супроводжується втратою пам'яті, запамороченням, нудотою, блювотою, блідістю, холодною шкірою, сповільненими диханням і пульсом. При важкому струсі може виникнути шок, тривала втрата пам'яті, розлади дихання та серцевої діяльності. Допомога передбачає укладання потерпілого на ноші з головою набік, контроль за диханням та блювотою, обережне транспортування. Здавлювання мозку – це тяжка черепна травма, спричинена крововиливом, що супроводжується

симптомами струсу, головним болем, порушенням свідомості, іноді – комою, судомами. Допомога така ж, як при струсі.

У разі пошкодження грудної клітки, які є частими під час обстрілів і обвалах, спостерігаються забої, струс і стиснення. Вони супроводжуються шоком, утрудненим диханням. При переломах ребер можливі розриви легенів, що призводять до пневмотораксу, емфіземи. Ознаки – сильний біль, поверхневе дихання, задуха, синюшність. Допомога включає тугу пов'язку на видиху, при потребі – пункцію плеври, транспортування в напівсидячому положенні. Проникаючі поранення грудної клітки супроводжуються болем, утрудненим диханням, кровохарканням, синюшністю, іноді – звуком засмоктування повітря через рану. Накладають герметичну пов'язку, транспортують терміново в напівсидячому положенні. У разі поранення серця спостерігається сильний біль, задуха, серцебиття, страх смерті, блідість, набухання шийних вен. Ранячий предмет не витягують, накладають асептичну пов'язку, потерпілого госпіталізують у лежачому положенні [20].

Удари супроводжуються припухлістю, болем, гематомами, помірним обмеженням руху. Можливі небезпечні внутрішні ушкодження (мозок, печінка, нирки). Потрібно забезпечити спокій, прикласти лід, накласти пов'язку.

При розтягненні виникає різкий біль, швидке утворення набряку, порушення функцій суглоба. Показані туга пов'язка, фіксація, холод, анальгетики, виклик лікаря.

Синдром здавлювання розвивається при тривалому стисненні тканин – найчастіше кінцівок. Він супроводжується шоком, інтоксикацією продуктами розпаду тканин. Ознаки: біль, порушення чутливості, набряк, блідість, твердість кінцівки, пухирі. Необхідно звільнити постраждалого, забезпечити дихання, накласти стерильні пов'язки, зафіксувати кінцівку бинтом і шиною, транспортувати на носилках, виводити з шоку, зігріти, дати чай, каву, алкоголь, ввести знеболювальні та серцеві засоби.

Вивих – це стійке зміщення кісток у суглобі, що супроводжується болем, деформацією, фіксацією кінцівки в неприродному положенні. Надається

допомога – холод, знеболення, вправлення виконує лише медперсонал, важливо не сплутати з переломом.

Утрата свідомості в умовах війни може бути наслідком стресу, болю, крововтрати, ураження мозку. Перша допомога: потерпілого вкладають дещо піднявши ноги і опустивши голову; розстібають комір, послаблюють ремінь. Дати понюхати нашатирний спирт, при потребі роблять штучне дихання. Триває недовго [20].

4.2 Проведення інструктажів з охорони праці

Відповідно до Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Державного комітету України з нагляду за охороною праці від 26.01.2005 № 15 (НПАОП 0.00-4.12-05), інструктажі з питань охорони праці є обов'язковою складовою забезпечення безпеки працівників на підприємстві. Вони спрямовані на ознайомлення працівників із вимогами нормативно-правових актів з охорони праці, надання домедичної допомоги потерпілим від нещасних випадків, а також правил поведінки у разі виникнення аварійних ситуацій, пожеж чи стихійних лих. Проведення інструктажів регулюється главою 6 Типового положення.

Інструктажі поділяються за характером і часом проведення на кілька видів. Вступний інструктаж проводиться до початку роботи для всіх новоприйнятих працівників, відряджених, учнів, студентів, екскурсантів. Первинний інструктаж здійснюється на робочому місці перед початком роботи для новоприйнятих працівників, тих, хто переводиться на іншу роботу, відряджених, а також учнів перед виконанням навчальних завдань. Повторний інструктаж проводиться періодично для закріплення знань з охорони праці: для робіт підвищеної небезпеки – раз на 3 місяці, для інших робіт – раз на 6 місяців. Позаплановий інструктаж проводиться у випадках введення нових нормативних актів, змін технологічного процесу, порушень вимог охорони праці, перерви в роботі (понад 30 днів для робіт підвищеної небезпеки, понад 60 днів для інших робіт) або за

вимогою наглядових органів. Цільовий інструктаж проводиться перед виконанням разових робіт, ліквідацією аварій, роботами за нарядом- допуском [35].

Організація та проведення інструктажів мають свої особливості. Вступний інструктаж проводить спеціаліст служби охорони праці або інша уповноважена особа, яка пройшла відповідне навчання, у кабінеті охорони праці або спеціально обладнаному приміщенні з використанням технічних засобів і наочних посібників. Програма та тривалість вступного інструктажу затверджуються роботодавцем, а запис про його проведення фіксується в журналі реєстрації вступного інструктажу та в наказі про прийняття на роботу. Первинний, повторний, позаплановий і цільовий інструктажі проводяться безпосереднім керівником робіт (начальником підрозділу, майстром) або фізичною особою, яка використовує найману працю. Первинний і повторний інструктажі проводяться індивідуально або з групою працівників одного фаху за інструкціями з охорони праці. Позаплановий інструктаж залежить від причин його проведення, а цільовий – від виду робіт. Усі ці інструктажі завершуються перевіркою знань шляхом усного опитування або за допомогою технічних засобів, а також набуттям навичок безпечної роботи.

Перевірка знань і облік інструктажів також чітко регламентовані. Після первинного, повторного чи позапланового інструктажу працівники, які показали незадовільні результати, проходять повторний інструктаж протягом 10 днів. Після цільового інструктажу при незадовільних результатах допуск до роботи не надається, а повторна перевірка не дозволяється. Записи про проведення інструктажів вносяться до журналу реєстрації інструктажів на робочому місці, який має бути пронумерованим, прошнурованим і скріпленим печаткою (за наявності). Для цільового інструктажу, пов’язаного з нарядом-допуском, запис може фіксуватися лише в наряді-допуску.

Особливістю є можливість звільнення від повторного інструктажу. Роботодавець може затвердити перелік посад і професій, працівники яких звільняються від повторного інструктажу, якщо їхня робота не пов’язана з

обслуговуванням обладнання чи застосуванням інструментів. Відповідальність за організацію та проведення інструктажів покладається на роботодавця, який забезпечує їх належне виконання та облік.

Документація відіграє важливу роль у процесі. Журнал реєстрації вступного інструктажу зберігається службою охорони праці або уповноваженим працівником. Журнал реєстрації інструктажів на робочому місці містить інформацію про вид інструктажу, причини позапланового чи цільового інструктажу, а також дані про стажування чи дублювання. Проведення інструктажів є ключовим елементом системи управління охороною праці, що сприяє зниженню ризиків виробничого травматизму та підвищенню безпеки на робочому місці [21].

ВИСНОВКИ

У процесі виконання бакалаврської роботи була проведена розробка мобільного додатку для підтримки здорового способу життя із використанням Kotlin.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» аналіз предметної області показав, що сучасні мобільні додатки для підтримки здорового способу життя, такі як FatSecret, Dine4Fit та YAZIO, мають широкий функціонал, але часто страждають від складних інтерфейсів, надмірної реклами та обмежень у безкоштовних версіях. Це підкреслює необхідність створення зручного, інтуїтивно зрозумілого додатку. Вибір технологій для розробки мобільного додатку був обґрунтованим. Використання мови програмування Kotlin та середовища Android Studio забезпечило високу продуктивність, сумісність із платформою Android та зручність розробки. Застосування нативного підходу дозволило максимально використати можливості операційної системи.

В другому розділі кваліфікаційної роботи було розглянуто архітектуру додатку, побудовану на основі спрощеної моделі MVVM, яка забезпечила модульність, масштабованість та реактивність інтерфейсу. Використання бібліотек Jetpack Compose і Room спростили створення сучасного інтерфейсу та забезпечили ефективне управління даними. Централізація логіки дозволила оптимізувати взаємодію між інтерфейсом, базою даних і бізнес-логікою, що сприяло стабільності та зручності підтримки коду. Функціональні можливості додатку, включаючи облік калорій, макронутрієнтів, водного балансу, тренувань, а також персоналізовані налаштування, були успішно реалізовані.

В третьому розділі кваліфікаційної роботи було продемонстровано роботу мобільного додатку для підтримки здорового способу життя. Тестування та валідація додатку, проведені за допомогою інструментів Android Studio, показали відсутність критичних помилок та високу стабільність роботи. Виявлені незначні попередження були проаналізовані та, за необхідності,

виправлені, що забезпечило відповідність проєкту сучасним стандартам розробки для Android.

В розділі «Безпека життєдіяльності, основи охорони праці» підкреслюється важливість знань з надання долікарської допомоги при ушкоджені м’яких тканин, суглобів і кісток в умовах війни, а також значення інструктажів з охорони праці для забезпечення безпеки працівників.

У цілому, розроблений мобільний додаток відповідає поставленій меті та завданням, демонструючи високий рівень функціональності, зручності та потенціалу для подальшого розвитку.

ПЕРЕЛІК ДЖЕРЕЛ

1. 5 кращих додатків для підрахунку калорій [Електронний ресурс]. Режим доступу до ресурсу: https://ingeniusua.org/articles/5-krashchykh-dodatkiv-dlya-pidrakhunku-kaloriy (дата звернення: 08.05.2025).

2. Найкращі програми, щоб навчитись рахувати калорії [Електронний ресурс]. Режим доступу до ресурсу: https://uk.androidayuda.com/застосування/рекомендується-а/додаток-для-підрахунку-калорій (дата звернення: 08.05.2025).

3. Таблиця калорійності – калорії [Електронний ресурс]. Режим доступу до ресурсу: https://apps.apple.com/ua/app/таблиця-калорійності-калорії/id477039928?l=uk (дата звернення: 08.05.2025).

4. Таблиця калорійності – калорії [Електронний ресурс]. Режим доступу до ресурсу: https://www.tablycjakalorijnosti.com.ua/ (дата звернення: 08.05.2025).

5. Кращі додатки для підрахунку калорій: топ 10 [Електронний ресурс]. Режим доступу до ресурсу: https://www.moyo.ua/ua/news/luchshie_prilozheniya_dlya_podscheta_kaloriy_top-10.html (дата звернення: 08.05.2025).

6. Які технології використовуються для розробки мобільних додатків [Електронний ресурс]. Режим доступу до ресурсу: https://smart-solutions.com.ua/archives/632 (дата звернення: 08.05.2025).

7. Про мови програмування для Android [Електронний ресурс]. Режим доступу до ресурсу: https://foxminded.ua/movy-prohramuvannia-dlia-android/ (дата звернення: 08.05.2025).

8. MVVM (Model-View-ViewModel) Architecture Pattern in Android [Електронний ресурс]. Режим доступу до ресурсу: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/ (дата звернення: 08.05.2025).

9. Що таке діаграма варіантів використання UML [Електронний ресурс]. Режим доступу до ресурсу: https://www.mindonmap.com/uk/blog/what-is-a-uml-use-case-diagram/ (дата звернення: 08.05.2025).

10. Використання Jetpack Compose у сучасній Android-розробці [Електронний ресурс]. Режим доступу до ресурсу: https://careers.epam.ua/blog/jetpack-compose-in-android-development (дата звернення: 08.05.2025).

11. Navigation with Compose [Електронний ресурс]. Режим доступу до ресурсу: https://developer.android.com/develop/ui/compose/navigation (дата звернення: 08.05.2025).

12. Room [Електронний ресурс]. Режим доступу до ресурсу: https://brander.ua/technologies/room (дата звернення: 08.05.2025).

13. Mermaid Live Editor [Електронний ресурс]. Режим доступу до ресурсу: https://mermaid.live/ (дата звернення: 09.05.2025).

14. State and Jetpack Compose [Електронний ресурс]. Режим доступу до ресурсу: https://developer.android.com/develop/ui/compose/state (дата звернення: 09.05.2025).

15. Таблиця калорійності продуктів [Електронний ресурс]. Режим доступу до ресурсу: https://calc.tablycjakalorijnosti.com.ua/explanation (дата звернення: 09.05.2025).

16. Run apps on the Android Emulator [Електронний ресурс]. Режим доступу до ресурсу: https://developer.android.com/studio/run/emulator (дата звернення: 09.05.2025).

17. Mobile app onboarding: How-to, best practices, and examples [Електронний ресурс]. Режим доступу до ресурсу: https://adapty.io/blog/mobile-app-onboarding/ (дата звернення: 09.05.2025).

18. Improve your code with lint checks [Електронний ресурс]. Режим доступу до ресурсу: https://developer.android.com/studio/write/lint/ (дата звернення: 09.05.2025).

19. Домедична допомога в умовах воєнного стану: що потрібно робити [Електронний ресурс]. Режим доступу до ресурсу: https://armyinform.com.ua/2022/10/22/domedychna-dopomoga-v-umovah-voyennogo-stanu-shho-potribno-robyty/ (дата звернення: 09.05.2025).

20. Долікарська допомога при ушкодженні м’яких тканин, суглобів і кісток [Електронний ресурс]. Режим доступу до ресурсу: https://dl.tntu.edu.ua/content.php?cid=299863 (дата звернення: 09.05.2025).

21. Про затвердження Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці (НПАОП 0.00-4.12-05) [Електронний ресурс]. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0231-05#Text (дата звернення: 09.05.2025).

22. YAZIO: Calorie Counter & Diet [Електронний ресурс]. Режим доступу до ресурсу: https://www.yazio.com/en (дата звернення: 09.05.2025).

23. Кросплатформова розробка додатків [Електронний ресурс]. Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/krossplatformennaya-razrabotka-prilozhenij (дата звернення: 09.05.2025).

24. Android Studio: переваги та особливості [Електронний ресурс]. Режим доступу до ресурсу: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/ (дата звернення: 09.05.2025).

25. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс]. Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/#i-8 (дата звернення: 10.05.2025).

26. MVVM in Android [Електронний ресурс]. Режим доступу до ресурсу: https://medium.com/@zorbeytorunoglu/mvvm-in-android-059e9aae84c1 (дата звернення: 10.05.2025).

27. Draw.io [Електронний ресурс]. Режим доступу до ресурсу: https://www.drawio.com/ (дата звернення: 10.05.2025).

28. Android Room Tutorial: Simplifying How You Work with App Data [Електронний ресурс]. Режим доступу до ресурсу: https://gorillalogic.com/blog-and-resources/android-room-tutorial-simplifying-how-you-work-with-app-data (дата звернення: 10.05.2025).

29. Діаграма прецедентів [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Діаграма_прецедентів (дата звернення: 10.05.2025).

30. Java [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java (дата звернення: 10.05.2025).

31. Kotlin [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Kotlin (дата звернення: 10.05.2025).

32. Android Studio [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Android_Studio (дата звернення: 10.05.2025).

33. Jetpack Compose [Електронний ресурс]. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Jetpack_Compose (дата звернення: 10.05.2025).

34. Save data in a local database using Room [Електронний ресурс]. Режим доступу до ресурсу: https://developer.android.com/training/data-storage/room (дата звернення: 10.05.2025).

35. Зеркалов Д. В. Безпека життєдіяльності. Навч.посіб. / – Київ: Основа, 2016. – 267 с. – ISBN 978-966-699-866-1.

36. Leshchyshyn, Y., Scherbak, L., Nazarevych, O., Gotovych, V., Tymkiv, P., & Shymchuk, G. (2019, May). Multicomponent Model of the Heart Rate Variability Change-point. In 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) (pp. 110-113). IEEE.

37. Lytvynenko, I., Lupenko, S., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, September). Mathematical model of gas consumption process in the form of cyclic random process. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 232-235). IEEE.

38. Kozlovskyi, V., Balanyuk, Y., Martyniuk, H., Nazarevych, O., Scherbak, L., & Shymchuk, G. (2022, April). Information Technology for Estimating City Gas

Consumption During the Year. In 2022 International Conference on Smart Information Systems and Technologies (SIST) (pp. 1-4). IEEE.

39. Lytvynenko, I., Lupenko, S., Kunanets, N., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, November). Simulation of gas consumption process based on the mathematical model in the form of cyclic random process considering the scale factors. In 1st International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2021.

40. Kunanets, N., Pasichnyk, V., Bodnarchuk, I., Martsenko, S., Matsiuk, O., Matsiuk, A., ... & Shymchuk, H. (2019). Information system for visual analyzer disease diagnostics. In CEUR Workshop Proceedings (pp. 43-56).

41. Lupenko, S., Lytvynenko, I., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, December). Approach to gas consumption process forecasting on the basis of a mathematical model in the form of a random cyclic process. In Proceedings of the International Conference „Advanced applied energy and information technologies 2021", 2021 (pp. 213-219). TNTU, Zhytomyr «Publishing house „Book-Druk "» LLC.

42. Lytvynenko, I., Lupenko, S., Nazarevych, O., Shymchuk, H., & Hotovych, V. (2022). Additive mathematical model of gas consumption process. Вісник Тернопільського національного технічного університету, 104(4), 87-97.

43. Nazarevych, O., Leshchyshyn, Y., Lupenko, S., Hotovych, V., Shymchuk, G., & Shabliy, N. (2020, September). Method of Gas Consumption Change-point Detection Based on Seasonally Multicomponent Model. In 2020 10th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 152-155). IEEE.

44. Palianytsia, Y., Lytvynenko, I., Menoub, A., Shymchuk, H., & Dubchak, A. (2024). Development of an algorithm for identification of damage types on the surface of sheet metal.

45. Nazarevych, O., Gotovych, V., & Shymchuk, G. (2016). Information Technology for Monitoring of Municipal Gas Consumption, Based on Additive Model

and Correlated for Weather Factors. Journal of Information and Computing Science, 11(3), 180-187.

46. Shymchuk, G., Lytvynenko, I., Hromyak, R., Lytvynenko, S., & Hotovych, V. (2023). Gas Consumption Forecasting Using Machine Learning Methods and Taking Into Account Climatic Indicators. In CITI (pp. 156-163).

47. Leschyshyn, Y. Z., Nazarevych, O. B., Shymchuk, G. V., Revutskyi, E. A., & Shcherbak, L. M. (2016, September). The Methods of Change Point Detection and Statistical Estimating of Dynamic of the Noise Stochastic Signals Characteristics. In THE SEVENTH WORLD CONGRESS "AVIATION IN THE XXI-st CENTURY" Safety in Aviation and Space Technologies September 19-21, NATIONAL AVIATION UNIVERSITY. Kyiv: NAU.

48. Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Конспект лекцій з дисципліни Комп'ютерна графіка для студентів освітнього рівня «бакалавр» спеціальності 125 «Кібербезпека».

49. Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Конспект лекцій з дисципліни «Розподілені системи моніторингу та керування».

50. Шимчук, Г. В., Маєвський, О. В., Назаревич, О. Б., & Стадник, М. А. (2016). Конспект лекцій з дисципліни «Грід-системи та технології хмарних обчислень» для студентів освітніх рівнів «спеціаліст»,«магістр» 122 «Комп'ютерні науки та інформаційні технології».

51. Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Методичні вказівки до самостійної робіти студентів та модульного контролю знань з дисципліни «Розподілені системи моніторингу та керування» для студентів освітнього рівня «бакалавр» спеціальності 125–«Кібербезпека».

52. ШИМЧУК, Г., ШЕВЧЕНКО, Н., ШВИРЛО, К., & ГАРМАТЮК, Н. (2025). СИСТЕМА ВІДНОВЛЕННЯ ДАНИХ У БЕЗДРОТОВИХ СЕНСОРНИХ МЕРЕЖАХ НА ОСНОВІ МАШИННОГО НАВЧАННЯ. Herald of Khmelnytskyi National University. Technical sciences, 353(3.2), 246-250.

ДОДАТКИ

Додаток А

Код пов’язаний з базою даних

Лістинг конфігурації бази даних

```
@Database(
    entities = [User::class, WaterIntake::class,
DailyNutrition::class, Product::class,
        MealProduct::class, Exercise::class, UserWorkout::class],
    version = 8
)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
    abstract fun waterIntakeDao(): WaterIntakeDao
    abstract fun dailyNutritionDao(): DailyNutritionDao
    abstract fun productDao(): ProductDao
    abstract fun mealProductDao(): MealProductDao
    abstract fun exerciseDao(): ExerciseDao
    abstract fun userWorkoutDao(): UserWorkoutDao
    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null
        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "app_database"
                )
                .addCallback(object : RoomDatabase.Callback() {
                    override fun onCreate(db:
SupportSQLiteDatabase) {
                        // Додавання вправ під час створення бази
                    }
                })
                .build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

Лістинг додавання вправ до бази при створенні

```
CoroutineScope(Dispatchers.IO).launch {
    val exerciseDao = getDatabase(context).exerciseDao()
    exerciseDao.insertExercise(
        Exercise(
            name = "Жим лежачи",
```

```
            category = "Груди",
            repetitions = 12,
            sets = 4,
            difficulty = "Середній",
            description = "...",
            animationResId = R.raw.bench_press
        )
    )
    // Додаються інші вправи
}
```

Лістинг моделі DailyNutrition

```
@Entity(tableName = "daily_nutrition",
    indices = [Index(value = ["userId", "date"], unique = true)]
)
data class DailyNutrition(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val userId: Int,
    val date: String,
    val targetCalories: Int,
    val consumedCalories: Int,
    val targetProtein: Int,
    val consumedProtein: Int,
    val targetFat: Int,
    val consumedFat: Int,
    val targetCarbs: Int,
    val consumedCarbs: Int,
    val breakfastCalories: Int = 0,
    val lunchCalories: Int = 0,
    val snackCalories: Int = 0,
    val dinnerCalories: Int = 0
)
```

Лістинг DailyNutritionDao

```
@Dao
interface DailyNutritionDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertOrUpdateDailyNutrition(nutrition:
DailyNutrition)

    @Query("SELECT * FROM daily_nutrition WHERE userId = :userId AND
date = :date")
    suspend fun getNutritionForDate(userId: Int, date: String):
DailyNutrition?
}
```

Додаток Б

Код пов’язаний з логікою додатку

Лістинг методу loginUser()

```
@RequiresApi(Build.VERSION_CODES.O)
fun loginUser(username: String, password: String): Boolean {
    return runBlocking {
        val user = userDao.getUser(username, password)
        if (user != null) {
            userId = user.id
            isOnboardingComplete = user.onboardingComplete
            userGoal = user.goal
            userActivityLevel = user.activityLevel
            userAge = user.age
            userWeight = user.weight
            userHeight = user.height
            userGender = user.gender
            if (isOnboardingComplete) {
                calculateNutritionTargets()
                loadWorkoutsForDate()
            }
            true
        } else {false
        }
    }
}
```

Лістинг методу setUserGoal()

```
fun setUserGoal(goal: String) {
    userGoal = goal
    viewModelScope.launch {
        userDao.updateUserOnboardingData(
            userId = userId,
            weight = userWeight,
            activityLevel = userActivityLevel,
            age = userAge,
            height = userHeight,
            gender = userGender,
            goal = goal
        )
    }
}
```

Лістинг методу calculateNutritionTargets ()

```
private fun calculateNutritionTargets() {
    val weight = userWeight?.toIntOrNull() ?: 70
    val height = userHeight?.toIntOrNull() ?: 170
```

```
    val age = userAge?.toIntOrNull() ?: 30
    val isMale = userGender == "Чоловік"
    val bmr = if (isMale) {
        10 * weight + 6.25 * height - 5 * age + 5
    } else {
        10 * weight + 6.25 * height - 5 * age - 161
    }
    val activityMultiplier = when (userActivityLevel) {
        "Середній" -> 1.375
        "Високий" -> 1.55
        else -> 1.2
    }
    var totalCalories = (bmr * activityMultiplier).toInt()
    totalCalories = when (userGoal) {
        "Скидання ваги" -> (totalCalories * 0.85).toInt()
        "Набір ваги" -> (totalCalories * 1.15).toInt()
        else -> totalCalories
    }
    ...
}
```

Лістинг методу addWaterIntake(amount: Int)

```
fun addWaterIntake(amount: Int) {
    viewModelScope.launch {
        val newConsumedWater = consumedWater.value + amount
        consumedWater.value = newConsumedWater
        val                    existingIntake                    =
waterIntakeDao.getWaterIntakeForDate(userId, currentDate)
        if (existingIntake != null) {
            waterIntakeDao.updateWaterIntake(userId,   currentDate,
newConsumedWater)
        } else {
            val waterIntake = WaterIntake(
                userId = userId,
                date = currentDate,
                consumedWater = newConsumedWater
            )
            waterIntakeDao.insertWaterIntake(waterIntake)
        }
    }
}
```

Лістинг методу setDate()

```
@RequiresApi(Build.VERSION_CODES.O)
fun setDate(date: LocalDate) {
    currentDate = date.format(DateTimeFormatter.ISO_LOCAL_DATE)
    loadWaterIntakeForDate()
    loadDailyNutritionForDate()
    loadWorkoutsForDate()
}
```