МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

БЕВЗЮК АНАСТАСІЯ ЮРІЇВНА

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
д-р. техн. наук, професор
_____Наталія ВЕСЕЛОВСЬКА
«____»_____ 2026 р.

РОЗРОБКА МОБІЛЬНОГО ПОМІЧНИКА ДЛЯ ПЕРСОНАЛЬНОГО КОНТРОЛЮ ХАРЧУВАННЯ ТА ЗБАЛАНСОВАНОСТІ РАЦІОНУ З УРАХУВАННЯМ ІНДИВІДУАЛЬНИХ ОСОБЛИВОСТЕЙ КОРИСТУВАЧА

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Юрій АНТОНОВ, доцент кафедри
інформаційних технологій
к. фіз.-мат. наук, доцент,
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2026

АНОТАЦІЯ

Бевзюк А.Ю. Розробка мобільного помічника для персонального контролю харчування та збалансованості раціону з урахуванням індивідуальних особливостей користувача. Спеціальність 122 “Комп’ютерні науки”. Освітня програма “Комп’ютерні науки”. Донецький національний університет імені Василя Стуса, Вінниця, 2026.

У кваліфікаційній роботі представлено розробку мобільного помічника для персонального контролю харчування та забезпечення збалансованості раціону з урахуванням індивідуальних особливостей користувача. Проведено аналіз предметної області, сучасних рішень та підходів до створення мобільних застосунків.

У роботі спроєктовано архітектуру системи, розроблено структуру бази даних та реалізовано механізми розрахунку добових норм харчування, формування персоналізованих рекомендацій, обліку спожитої їжі, водного балансу та статистичного аналізу даних користувача.

Програмний застосунок реалізовано з використанням сучасних технологій мобільної розробки та локального збереження даних. Результатом роботи є мобільний помічник, який забезпечує автоматизацію контролю харчування, моніторинг харчових звичок і підтримку користувача у дотриманні збалансованого раціону.

Ключові слова: мобільний помічник, персоналізація, план харчування, збалансований раціон, рекомендаційна система.

66 с., 22 рис., 4 табл., 50 джерела.

ABSTRACT

Bevziuk A.Y. Development of a Mobile Assistant for Personal Nutrition Control and Diet Balance, Considering the Individual Characteristics of the User. Specialty 122 “Computer Science”. Educational Program “Computer Science”. Vasyl Stus Donetsk National University, Vinnytsia, 2026.

The qualification paper presents the development of a mobile assistant for personal dietary monitoring and ensuring a balanced diet tailored to the user's individual characteristics. An analysis of the subject area, current solutions, and approaches to mobile application development was conducted.

The project designs the system architecture, develops the database structure, and implements mechanisms for calculating daily nutritional requirements, generating personalized recommendations, tracking food intake, monitoring water balance, and performing statistical analysis of user data.

The application was implemented using modern mobile development technologies and local data storage. The result of this work is a mobile assistant that automates dietary control, monitors eating habits, and supports the user in maintaining a balanced diet.

Keywords: mobile assistant, personalization, meal plan, balanced diet, recommendation system.

66 p., 22 fig., 4 tab., 50 sources.

ЗМІСТ

ВСТУП

Актуальність теми дослідження: З огляду на стрімкий розвиток інформаційних технологій та зростання інтересу людей до здорового способу життя, особливої актуальності набуває питання раціонального харчування. Неправильні харчові звички, незбалансованість щоденного меню, відсутність контролю за споживанням їжі можуть погіршити стан здоров'я, спричинити хронічні захворювання і знизити якість життя. Тому необхідно розробити ефективні заходи, які допоможуть користувачам контролювати своє харчування.

Мобільні технології дають змогу створювати інтелектуальні системи підтримки прийняття рішень у сфері здорового способу життя [1]. Сучасні застосунки дозволяють не лише фіксувати все з'їдене, а й аналізувати раціон, враховувати індивідуальні параметри користувача (вік, стать, рівень фізичної активності) та отримувати персоналізовані поради щодо поліпшення власного раціону.

Мета дослідження: створення мобільного помічника для персонального контролю харчування та забезпечення збалансованості раціону з урахуванням індивідуальних особливостей користувача.

Завдання дослідження: Для досягнення поставленої мети необхідно вирішити такі завдання:

- аналіз предметної області та визначення основних проблем та потреб користувачів;
- дослідити сучасні підходи до розробки мобільних застосунків;
- визначити вимоги до системи та бази даних;
- спроєктувати архітектуру системи та інформаційну структуру даних;
- розробити алгоритмічну модель формування персоналізованих рекомендацій;
- реалізувати програмний застосунок та проаналізувати результати роботи.

Об’єкт дослідження: процес персонального контролю харчування та формування збалансованого раціону з урахуванням індивідуальних характеристик користувача.

Предмет дослідження: методи та алгоритми формування персоналізованих рекомендацій щодо харчування, підходи до обробки та аналізу даних користувача з метою забезпечення збалансованості раціону.

Практичне значення результатів роботи полягає у можливості використання розроблених підходів та програмного рішення у сфері підтримки здорового способу життя. У межах роботи створено мобільний застосунок на базі сучасної кросплатформної технології .NET MAUI, який забезпечує облік харчування, аналіз складу раціону та калорійність, а також формування персоналізованих рекомендацій для досягнення оптимального балансу поживних речовин. Отримані результати можуть бути використані як інструмент для підвищення обізнаності користувачів щодо власних харчових звичок. У підсумку це формує усвідомлене ставлення до харчування, сприяє закріпленню корисних звичок і може слугувати профілактикою різних захворювань.

Апробація результатів дослідження: Публікація тез на тему «Activity-based modeling of a personalized meal recommendation system in a mobile assistant» у матеріалах X Міжнародної науково-практичної конференції для студентів, аспірантів та молодих вчених «Topical issues of humanities, technical and natural sciences» [2].

Публікація тез на тему «Проєктування бази даних системи персонального контролю харчування» у матеріалах VII Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих вчених «Прикладні інформаційні технології» (ПІТ-2026) [3].

Структура кваліфікаційної роботи: Бакалаврська робота складається із вступу, трьох розділів та висновків до них, списку використаних джерел та додатків.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Особливості персонального контролю харчування.

Здоров’я людини є багатогранним поняттям, яке охоплює не лише фізичний стан, а й психічне, соціальне та емоційне благополуччя. Одним із ключових чинників, що визначають рівень здоров’я, є харчування. Воно впливає на функціонування організму, його розвиток та здатність до адаптації в умовах навколишнього середовища. Сьогодні дедалі більшого значення набуває персональний контроль харчування – як спосіб підтримувати хорошу якість життя.

Їжа відіграє провідну роль у формуванні здоров’я людини, забезпечуючи організм необхідною енергією та поживними речовинами. Харчування сприяє нормальному росту та розвитку організму, підтримує працездатність, впливає на активність людини та підвищує стійкість до несприятливих чинників довкілля. За різними оцінками, саме харчування має найбільший внесок у загальний стан здоров’я порівняно з іншими факторами, такими як умови довкілля, спадковість чи охорона здоров’я [4].

Недостатньо збалансоване або надмірне харчування може призводити до порушень у роботі організму. Подібні порушення зазвичай виникають не одразу, а поступово і є наслідком тривалого недотримання збалансованого раціону. Вони можуть проявлятися у зниженні стійкості організму, погіршенні самопочуття та появі захворювань, пов’язаних із дефіцитом або надлишком певних поживних речовин.

Саме тому важливим елементом персоналізованого контролю харчування є систематичне відстеження ключових складових раціону, зокрема білків, жирів, вуглеводів, загальної калорійності, а також мікроелементів. Важливим аспектом також є правильне співвідношення цих компонентів у раціоні, що визначає його збалансованість (рис.1.1).

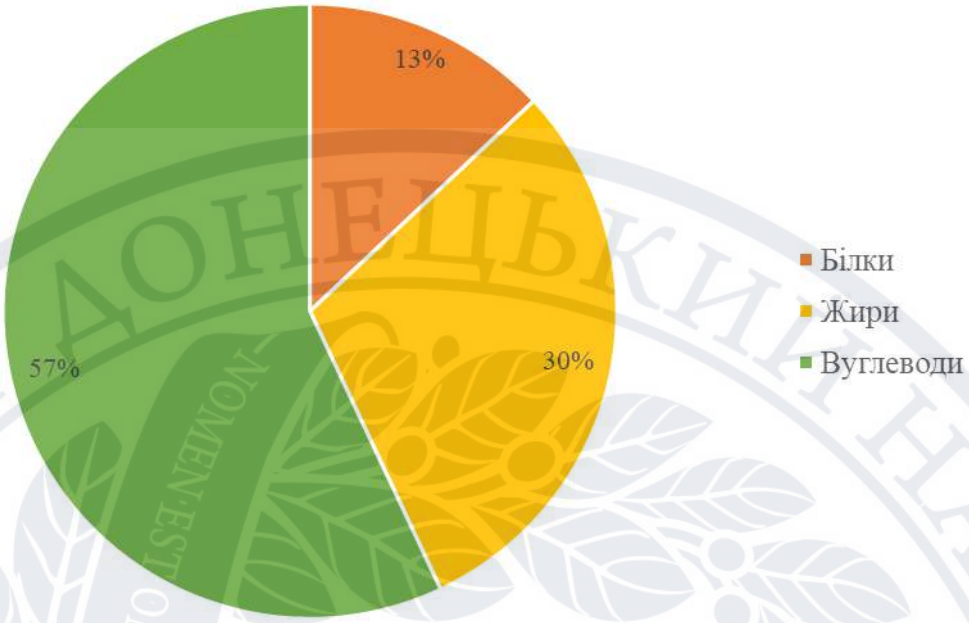

Рисунок 1.1 – Рекомендоване співвідношення макронутрієнтів у добовому раціоні

Як видно з рисунка, найбільшу частку в раціоні становлять вуглеводи, тоді як білки та жири займають меншу частку, що відповідає сучасним рекомендаціям щодо раціонального харчування [5]. Розглянемо детальніше роль кожного з макронутрієнтів у забезпеченні життєдіяльності організму.

Білки належать до ключових поживних речовин, необхідних для нормального функціонування організму. Вони виконують будівельну функцію – беруть участь у рості, регенерації клітин і тканин, а також у роботі м'язів і внутрішніх органів. Організм людини не накопичує білки, тому їх регулярне надходження з їжею є обов'язковим. Недостатнє споживання може призводити до зниження працездатності та ослаблення імунітету, тоді як надлишок створює додаткове навантаження [6].

Жири – це ще одна необхідна складова раціону. Вони забезпечують організм енергією та сприяють засвоєнню окремих вітамінів. Крім того, виконують захисну функцію, підтримують роботу нервової системи та впливають на загальний стан організму. Разом з тим важливо дотримуватися балансу, оскільки як дефіцит, так і надлишок можуть мати негативні наслідки [7].

Вуглеводи є основним джерелом енергії, необхідної для повсякденної

активності, зокрема для роботи мозку. Вони також беруть участь у підтримці нормального обміну речовин та допомагають організму ефективно використовувати інші поживні речовини. Але важливо звертати увагу не лише на кількість, а й на якість вуглеводів, віддаючи перевагу складним (наприклад, крупам), адже вони забезпечують більш рівномірне надходження енергії [6, 7].

Не менш важливими є вітаміни та мінерали, які хоча й потрібні в невеликих кількостях, але виконують критично важливі функції. Вони підтримують імунітет, беруть участь у роботі нервової системи, також впливають на стан шкіри, кісток та внутрішніх органів. Недостатнє надходження цих речовин може призвести до різних порушень, навіть за умов достатньої енергетичної цінності раціону [7].

Окрему увагу слід звернути на калорійність раціону, яка відображає загальну кількість енергії, отриманої з їжі. Для підтримки стабільної ваги необхідно, щоб кількість спожитих калорій відповідала енерговитратам організму. У разі перевищення енергетичного балансу відбувається набір маси тіла, тоді як його дефіцит призводить до її зниження. Саме тому розрахунок добової потреби в калоріях є важливою частиною персоналізованого підходу [8].

Водночас реакція організму на той самий раціон може суттєво варіюватися в різних людей. Це обумовлено індивідуальними особливостями обміну речовин, віком, рівнем фізичної активності та іншими чинниками. Таким чином, універсальні харчові рекомендації не завжди виявляються однаково ефективними для кожного, що підкреслює важливість застосування персоналізованого підходу [6, 8].

Актуальність такого підходу підтверджується сучасними епідеміологічними даними. Згідно з результатами національного дослідження STEPS, проведеного у 2019 році за підтримки World Health Organization, в Україні близько 59% дорослого населення мають надмірну масу тіла, а майже чверть – ожиріння. Крім того, поширеність діабету другого типу становить близько 6%. Отримані дані свідчать про значну поширеність факторів ризику неінфекційних захворювань, що підсилює необхідність системного контролю

харчування та впровадження індивідуальних підходів [9].

1.2 Аналіз сучасних підходів до розробки мобільних застосунків

Стрімке поширення мобільних пристроїв стимулює розвиток функціональності застосунків. Сьогодні вони стали основним інструментом взаємодії користувачів із цифровими сервісами [10, 11]. У зв'язку з цим особливої актуальності набуває питання створення ефективних програмних рішень, що відповідають сучасним стандартам продуктивності, безпеки та зручності використання (UX/UI).

У сучасній практиці розробки мобільного програмного забезпечення виділяють три основні підходи: нативний, кросплатформний та гібридний [12]. Кожен із них має власні переваги та недоліки, які впливають на вибір технічних рішень та інструментів реалізації. Для порівняння доцільно враховувати такі критерії, як продуктивність, швидкість розробки, вартість, зручність підтримки, доступ до функціональних можливостей пристрою та якість користувацького інтерфейсу. Розглянемо кожен із них окремо:

1. Нативна розробка – це підхід, який передбачає створення застосунків окремо для кожної операційної системи із використанням відповідних мов програмування та інструментів. Наприклад, для платформи iOS використовують Swift або Objective-C, а для Android – Kotlin або Java [13, 14]. Це забезпечує найвищий рівень продуктивності, кращу інтеграцію з операційною системою та повний доступ до апаратних можливостей пристрою. Водночас недоліком цього підходу залишається значна ресурсномісткість: необхідність паралельної розробки та підтримки окремих версій застосунку для кожної платформи суттєво збільшує як бюджет, так і терміни реалізації проєкту.
2. Кросплатформна розробка базується на створенні програмних продуктів, які працюють на кількох платформах із використанням

єдиної кодової бази. Серед найпоширеніших фреймворків можна виділити .NET MAUI [15], Flutter [16] та React Native. Основною перевагою цього підходу є зменшення витрат на реалізацію та підтримку, а також скорочення часу виходу продукту на ринок. Попри те, що сучасний інструментарій забезпечує продуктивність, близьку до нативної та доступ до більшості функцій пристрою, у деяких випадках можуть виникати обмеження щодо оптимізації або використання специфічних можливостей платформи.

3. Гібридна розробка поєднує елементи веб-технологій та нативні можливості [12]. В основі таких рішень лежить використання стеку HTML, CSS і JavaScript, що виконується всередині спеціальної оболонки (WebView), яка забезпечує доступ до функцій пристрою. Основною перевагою є спрощення процесу реалізації, а також можливість повторного використання веб-рішень, що мінімізує витрати. Водночас гібридний підхід здебільшого поступається нативним та кросплатформним рішенням за продуктивністю та якістю інтерфейсу користувача – особливо це помітно у складних проєктах.

Для більш обґрунтованого вибору технології реалізації мобільного застосунку доцільно провести порівняльний аналіз основних підходів розробки за ключовими критеріями ефективності результати якого наведено у табл. 1.1.

Таблиця 1.1 – Порівняльна таблиця основних підходів до мобільної розробки

Критерій	Нативна	Кросплатформна	Гібридна
Продуктивність	Висока	Висока (близька до нативної)	Середня
Швидкість реалізації	Низька	Висока	Висока
Вартість розробки	Висока	Середня	Низька
Доступ до API	Повний	Майже повний	Обмежений
Якість UI	Максимальна	Висока	Середня

Порівняльний аналіз характеристик свідчить про те, що кожен із розглянутих підходів має свою сферу застосування. Нативна розробка забезпечує максимальну продуктивність і якість інтерфейсу, однак вимагає значних витрат часу та ресурсів. Гібридний підхід є найбюджетнішим варіантом, проте поступається іншим за продуктивністю та якістю UI. Кросплатформна розробка займає збалансовану позицію: вона поєднує достатній рівень продуктивності, високу швидкість реалізації та економію витрат завдяки єдиній кодовій базі для різних операційних систем.

1.2.1 Порівняння архітектурних патернів проєктування

Ефективність сучасного програмного забезпечення безпосередньо залежить від фундаментальних архітектурних рішень, закладених на етапі проєктування. Архітектурні підходи не лише слугують основою структури системи, а й визначають логіку взаємодії між її компонентами. Раціональний розподіл обов’язків між складовими застосунку впливає на якість реалізації бізнес-логіки, а також на такі критичні характеристики, як масштабованість, надійність і зручність технічного обслуговування [17].

На сьогодні існує велика кількість архітектурних підходів, проте найбільш поширеними для створення клієнтських та мобільних застосунків є патерни MVC, MVP та MVVM [18]. Кожен з них пропонує власний спосіб розподілу відповідальності між інтерфейсом користувача, логікою обробки даних та механізмом роботи з моделлю даних.

З огляду на різноманітність підходів, кожен з яких має свої переваги та обмеження, важливим етапом стає їх порівняльний аналіз. Він дає змогу об’єктивно оцінити кожне рішення з урахуванням конкретних технічних вимог. Це особливо актуально у сфері розробки мобільних застосунків, де динамічність інтерфейсів і необхідність адаптивності створюють додаткові виклики. Таким чином, обґрунтований вибір архітектурного підходу є не просто технічним кроком, а основою для створення якісного програмного продукту.

Серед зазначених архітектурних патернів одним із найперших і найвідоміших є MVC (Model-View-Controller). Цей підхід передбачає поділ застосунку на три взаємопов’язані складові: модель, представлення та контролер [19]. Модель відповідає за роботу з даними та бізнес-логікою, представлення – за інтерфейс користувача, а контролер керує взаємодією між цими компонентами та обробляє дії користувача. Такий поділ дозволяє відокремити логіку обробки даних від інтерфейсу, що спрощує підтримку системи та повторне використання окремих компонентів. Однак MVC має певні недоліки, особливо у випадку складних клієнтських або мобільних застосунків. Зі зростанням функціональності контролер нерідко бере на себе забагато обов’язків, що призводить до надмірної складності структури коду та зниження його читабельності [19]. Крім того, сильна залежність між компонентами інтерфейсу та контролером ускладнює тестування і масштабування [20].

Альтернативним рішенням став патерн MVP (Model-View-Presenter), розроблений для підвищення рівня ізоляції логіки представлення від графічного інтерфейсу. У цій архітектурі роль посередника між моделлю та відображенням бере на себе Presenter, який приймає дії користувача й оновлює інтерфейс через чітко визначені механізми [21]. На відміну від MVC, представлення в MVP є більш пасивним і не містить значної логіки обробки даних. Це сприяє покращенню тестованості і забезпечує ефективніший контроль над взаємодією компонентів. Разом із тим, використання MVP може призводити до збільшення кількості допоміжного коду, особливо у великих проєктах із численними екранами. Presenter поступово накопичує значний обсяг логіки, що ускладнює його підтримку, а необхідність окремих інтерфейсів для зв’язку між View і Presenter підвищує загальну складність архітектури [18].

Подальшою еволюцією клієнтської архітектури став патерн MVVM (Model-View-ViewModel), який набув широкого поширення у розробці мобільних та кросплатформних застосунків. Його ключовою особливістю є використання компонента ViewModel, що забезпечує зв’язок між моделлю та інтерфейсом через механізми прив’язки даних [22]. Головними перевагами

MVVM є покращення модульності та підтримуваності програмного продукту. ViewModel не залежить від конкретної реалізації інтерфейсу, що позитивно впливає на тестованість і повторне використання логіки [23]. Проте надмірне використання прив’язок або складних механізмів відстеження стану може ускладнити налагодження та підвищити вимоги до організації коду.

Для наочного представлення особливостей розглянутих архітектурних патернів доцільно виконати їх порівняння за ключовими характеристиками: рівнем зв’язності компонентів, тестованістю, складністю реалізації, підтримуваністю та придатністю для мобільної розробки (табл.1.2) [18].

Таблиця 1.2 – Порівняльна характеристика архітектурних патернів

Критерій	MVC	MVP	MVVM
Зв’язність компонентів	Висока	Середня	Низька
Тестованість	Низька	Висока	Висока
Складність реалізації	Низька	Середня	Середня
Підтримуваність	Середня	Середня	Висока
Придатність для мобільної розробки	Обмежена	Помірна	Відмінна

Як видно з таблиці, жоден із розглянутих підходів не є універсальним. Кожен із них має власні сильні та слабкі сторони, що визначають його доцільність у конкретних умовах. Тому під час архітектурного проєктування системи важливим є обґрунтований вибір патерну з урахуванням вимог до масштабованості, тестованості та підтримуваності програмного продукту.

1.3 Огляд існуючих рішень

Для визначення ключових функцій і вимог до мобільного помічника проведено порівняльний аналіз наявних рішень у сфері контролю харчування та моніторингу раціону. Дослідження наявних систем дає змогу проаналізувати

сучасні підходи до персоналізованого харчування, оцінити переваги та недоліки, які доцільно врахувати під час розробки нової системи.

FatSecret є одним із найпоширеніших додатків для підрахунку калорій і ведення харчового щоденника. Приклад інтерфейсу подано на рисунку 1.2.

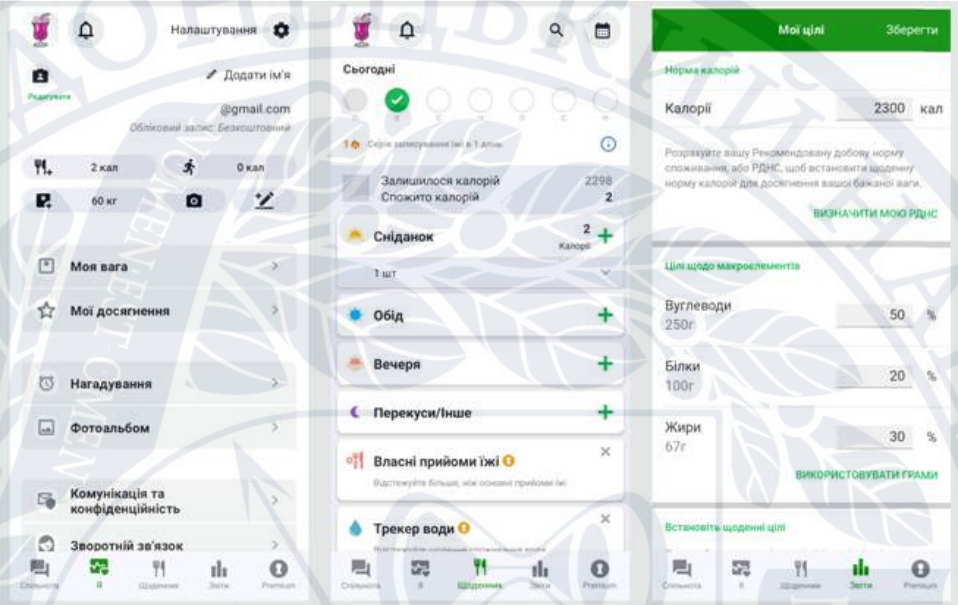

Рисунок 1.2 – Інтерфейс додатку FatSecret

Основними перевагами є наявність великої бази продуктів та можливість відстеження спожитих калорій і динаміки ваги. Водночас у процесі використання користувачі відзначають низку суттєвих недоліків. Платні функції не завжди виправдовують очікування, оскільки значна частина розширеного функціоналу не забезпечує помітного покращення зручності використання. Також є складність із додаванням власних рецептів через велику кількість кроків і часові витрати на заповнення даних [24].

LifeSum орієнтований на формування здорових харчових звичок. Його перевагами є сучасний дизайн, зручний інтерфейс (рис. 1.3), наявність функцій планування раціону, інтеграція з іншими сервісами та сканування штрихкодів. Проте значна частина функціоналу доступна лише у платній версії, що обмежує можливості безкоштовного використання. Важливим недоліком є також відсутність підтримки української мови, що підтверджується відгуками користувачів [25].

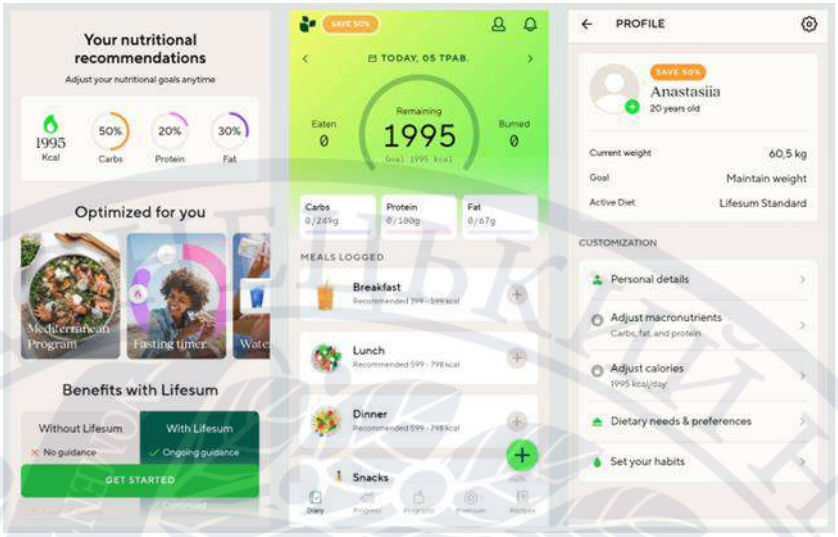

Рисунок 1.3 – Інтерфейс додатку LifeSum

MyFitnessPal – найбільш функціональний додаток, який поєднує можливості підрахунку калорій, аналізу раціону та відстеження фізичної активності (рис. 1.4.). Додаток містить велику базу продуктів та підтримує введення даних вручну або за допомогою сканування штрихкодів. До його переваг належать широкий функціонал, гнучкі налаштування індивідуальних цілей та інтеграція з іншими сервісами для моніторингу здоров'я.

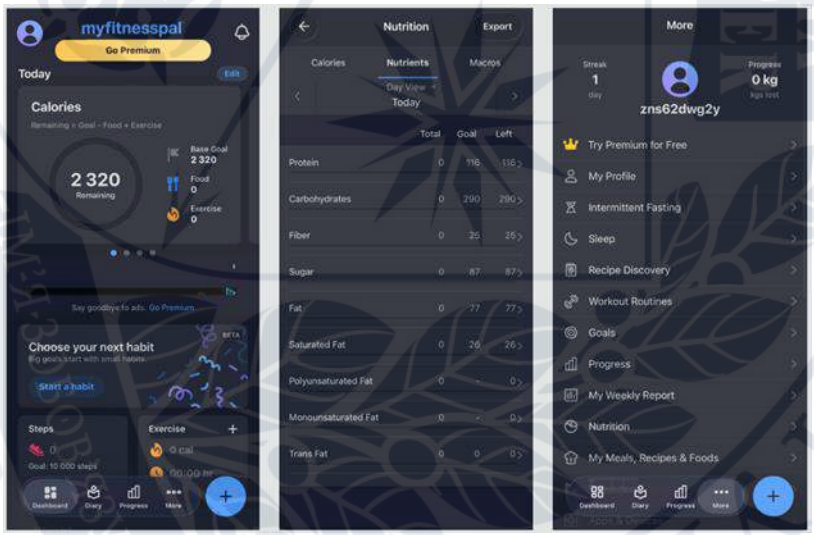

Рисунок 1.4 – Інтерфейс додатку MyFitnessPal

Водночас користувачі відзначають складність інтерфейсу, а також обмежений доступ до частини функцій у безкоштовній версії. Окремим недоліком є відсутність повноцінної підтримки української мови, що ускладнює використання для частини користувачів [26].

На основі здійсненого огляду розглянутих рішень доцільно узагальнити

їхні ключові характеристики та відмінності. З цією метою сформовано порівняльну таблицю 1.3, яка дозволяє наочно оцінити функціональні можливості, переваги та обмеження кожного із застосунків.

Таблиця 1.3 – Порівняльний аналіз мобільних застосунків для контролю харчування

Критерій порівняння	FatSecret	LifeSum	MyFitnessPal
Велика база продуктів	+	+	+
Підрахунок калорій та БЖВ	+	+	+
Відстеження ваги	+	+	+
Відстеження фізичної активності	+	+	+
Зручність інтерфейсу	-	+	-
Простота додавання власних рецептів	-	+	+
Гнучке налаштування цілей	+	+	+
Підтримка української мови	-	-	-
Сканер штрихкодів	+	+	+
Доступність функцій у безкоштовній версії	+	-	-

Отримані результати свідчать про те, що сучасні мобільні застосунки для контролю харчування мають подібний базовий функціонал, який включає підрахунок калорій, ведення харчового щоденника, аналіз раціону та використання бази продуктів. Водночас більшість існуючих рішень має спільні недоліки, серед яких переважають обмежений функціонал безкоштовних версій, недостатній рівень локалізації, зокрема відсутність підтримки української мови, а також складність або перевантаженість інтерфейсу.

Висновок до першого розділу

Проведений у розділі аналіз дозволив розглянути як предметну область дослідження, так і технологічні аспекти створення мобільних застосунків.

Встановлено, що харчування є одним із визначальних чинників, що впливають на стан здоров’я людини та якість її життя. Аналіз основних складових раціону, зокрема макро- та мікронутрієнтів, а також калорійності, підтверджує необхідність їх систематичного контролю, що особливо актуально в умовах сучасного ритму життя.

З урахуванням індивідуальних особливостей користувачів актуалізується потреба у застосуванні персоналізованих інформаційних систем, що забезпечують ефективний облік і аналіз даних про харчування. Результати аналізу існуючих рішень свідчать про доцільність розробки програмного продукту, який поєднуватиме базовий функціонал, зручність використання, інтуїтивний інтерфейс та повну адаптацію до потреб користувачів, зокрема підтримку локалізації та мінімізацію ручного введення даних.

Аналіз сучасних підходів до розробки мобільних застосунків і архітектурних патернів дав змогу окреслити основні тенденції та вимоги до розробки сучасного мобільного програмного забезпечення. Встановлено, що вибір технологічного підходу та архітектурної моделі суттєво впливає на продуктивність, масштабованість, підтримуваність і зручність використання застосунку.

Дослідження архітектурних патернів MVC, MVP та MVVM засвідчило їхні характерні особливості та сферу застосування, а також підкреслило важливість забезпечення низького рівня зв'язності компонентів, високої тестованості та можливості масштабування у сучасних мобільних застосунках.

Огляд наявних систем для контролю харчування дозволив визначити як найбільш поширені функціональні можливості, так і типові обмеження існуючих рішень, серед яких – обмежений безкоштовний функціонал, недостатня локалізація та перевантаженість інтерфейсу. Узагальнені результати формують теоретичну основу для подальшого проєктування мобільних систем у цій сфері з урахуванням вимог до зручності, персоналізації та ефективної організації функціональних можливостей.

РОЗДІЛ 2.
МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПЕРСОНАЛЬНОГО ПОМІЧНИКА З ХАРЧУВАННЯ

2.1 Постановка задачі та визначення вимог

Результати аналізу, проведеного у першому розділі, підтвердили доцільність розробки мобільного застосунку для персонального контролю харчування. Сучасні підходи до організації раціону потребують використанням цифрових засобів, які дозволяють не лише фіксувати спожиті продукти, а й здійснювати аналіз харчової цінності, контролювати дотримання добових норм та формувати індивідуальні рекомендації відповідно до фізіологічних особливостей користувача.

Основним завданням розроблюваної системи є створення мобільного застосунку, який забезпечуватиме комплексний облік харчування, аналіз показників раціону та підтримку користувача у процесі досягнення поставлених цілей, пов’язаних зі здоров’ям і способом життя. Для реалізації зазначеної мети необхідно визначити, формалізувати та систематизувати вимоги до програмного продукту, що дозволить чітко окреслити його функціональні можливості та якісні характеристики.

Вимоги до системи поділяються на функціональні та нефункціональні. Функціональні вимоги описують основні можливості застосунку та логіку його роботи, тоді як нефункціональні визначають характеристики якості, продуктивності, надійності та зручності використання системи.

Функціональні вимоги охоплюють такі можливості:

- введення, редагування та збереження персональних даних користувача (вік, стать, зріст, вага, рівень фізичної активності);
- налаштування профілю користувача із зазначенням цілей харчування, алергенів та дієтичних обмежень;
- розрахунок індивідуальної добової потреби в калоріях та макронутрієнтах (БЖВ) на основі параметрів користувача;

- ведення щоденника харчування з можливістю фіксації спожитих продуктів і страв;
- пошук, вибір та додавання продуктів і страв до раціону з автоматичним оновленням показників;
- врахування індивідуальних обмежень та алергенів під час формування рекомендацій та аналізу раціону;
- аналіз збалансованості добового раціону та визначення відхилень від рекомендованих норм;
- формування персоналізованих рекомендацій корекції раціону;
- відображення статистики харчування та прогресу у вигляді графіків та числових показників;
- збереження історії харчування для подальшого аналізу та порівняння результатів;
- фіксація споживання води протягом дня з відображенням виконання добової норми.

Нефункціональні вимоги визначають загальні характеристики якості системи та умови її ефективної експлуатації:

- інтуїтивно зрозумілий та зручний інтерфейс, що забезпечує мінімальний час на освоєння застосунку;
- висока продуктивність і швидкодія застосунку під час обробки щоденних харчових даних;
- коректна робота на мобільних пристроях із різними розмірами екранів та версіями операційної системи;
- забезпечення надійного збереження даних користувача та їх цілісності;
- можливість подальшого розширення функціональності та масштабування системи.

Таким чином, сформульовані вимоги визначають основні принципи функціонування мобільного застосунку та створюють основу для подальшого

проєктування архітектури системи, структури бази даних і механізмів формування персоналізованих рекомендацій.

2.1.2 Аналіз вимог до бази даних та інформаційного наповнення системи

Для забезпечення коректного функціонування мобільного помічника необхідно визначити перелік даних, які підлягають збереженню, аналізу та обробці. На основі аналізу предметної області було виокремлено ключові інформаційні сутності, що відображають основні процеси системи, зокрема облік харчування, моніторинг стану користувача та формування рекомендацій.

Кожен із визначених об’єктів характеризується набором властивостей, що забезпечують можливість їх подальшого збереження, обробки та використання в аналітичних механізмах системи. Узагальнені результати ідентифікації сутностей та їх властивостей наведено у таблиці 2.1.

Як показано в таблиці, інформаційна модель системи охоплює ключові аспекти контролю харчування користувача, включаючи персональні дані, параметри раціону, облік спожитих продуктів та моніторинг прогресу.

Центральною сутністю системи є «Користувач», який містить персональні дані, необхідні для розрахунку індивідуальних параметрів харчування. Із нею безпосередньо пов’язані «Ціль користувача» та «Денна норма»: перша визначає бажаний результат (зниження, підтримка або набір маси), тоді як друга відображає розраховані на її основі добові показники калорійності, макронутрієнтів і споживання води.

Компонент системи «Прогрес користувача» фіксує динаміку змін ваги у часі, що дозволяє відстежувати відповідність реального результату поставленій цілі. Обмеження та алергени враховуються при формуванні рекомендацій і фільтрації продуктів, що забезпечує безпечність та персоналізованість раціону.

Інші об’єкти, такі як «Продукти», «Страви», «Прийоми їжі» та «Вода», забезпечують безпосередній облік споживання та формують інформаційну

основу для аналізу раціону користувача.

Таблиця 2.1 - Інформаційні сутності системи та їх властивості [3]

Об'єкт (процес)	**Властивості**
Користувач	Ім'я, дата народження, стать, зріст, рівень фізичної активності, початкова вага
Прогрес користувача	Вага, дата вимірювання
Ціль користувача	Тип (підтримка ваги/набір маси/схуднення), цільова вага
Обмеження та алергени	Назва, тип обмеження (алергія/дієта/хвороба), рівень критичності
Продукти	Назва, категорія, калорії (ккал на 100г / 100мл), БЖВ, цукор, вітаміни(A, B, C...), клітковина, алергени, одиниця вимірювання (г, мл, шт)
Страви	Назва, опис, розмір порції, категорія, БЖВ, калорії
Склад страви	Продукт, кількість, одиниця вимірювання (г, мл, шт)
Денна норма	Дата встановлення, калорії, БЖВ, вітаміни, клітковина, норма води
Прийоми їжі	Тип, дата та час, список страв / продуктів, фактична вага
Вода	Дата, кількість (мл)
Рекомендації	Тип, опис, рекомендовані продукти/страви

Між визначеними сутностями існують логічні зв’язки, що відображають взаємодію об’єктів у системі. Зокрема, користувач пов’язаний із записами про прогрес, встановленими цілями, обмеженнями та історією харчування.

Більшість зв’язків між сутностями мають тип «один-до-багатьох» (наприклад, користувач – прийоми їжі), тоді як взаємозв’язок між сутностями «Страви» та «Продукти» реалізується як «багато-до-багатьох» із використанням проміжної сутності «Склад страви». Визначені зв’язки будуть використані при побудові ER-діаграми бази даних.

Виділені об’єкти та їх властивості становлять основу для проєктування структури бази даних та визначають необхідні зв’язки між сутностями, що забезпечують цілісність та узгодженість інформації. На основі сформованої інформаційної моделі здійснюється подальше проєктування логічної на фізичної структури бази даних.

2.2 Моделювання структури даних системи

Проєктування інформаційної структури системи здійснюється на основі модельно-орієнтованого підходу, який передбачає визначення ключових сутностей предметної області, їхніх атрибутів та взаємозв’язків [27]. Побудова такої інформаційної моделі забезпечує цілісне представлення даних, що слугує базисом для подальшого створення бази даних, програмної логіки додатків і реалізації основних бізнес-процесів [28].

У рамках реалізації системи було вибрано реляційну систему управління базами даних SQLite. Вибір цієї СУБД зумовлений її компактністю, низькими вимогами до ресурсів та ефективною інтеграцією з мобільними застосунками. Використання SQLite дозволяє забезпечити локальне збереження даних користувача без необхідності постійного підключення до зовнішнього сервера, що є доцільним для мобільних інформаційних систем персонального користування. Крім того, SQLite підтримує реляційну модель даних, механізми забезпечення цілісності через первинні та зовнішні ключі, а також виконання SQL-запитів для обробки й аналізу інформації [29].

На основі визначених у попередньому підрозділі інформаційних сутностей було розроблено логічну структуру бази даних, яка складається з 29 взаємопов'язаних таблиць, об'єднаних системою первинних та зовнішніх ключів. База даних призначена для збереження інформації про користувачів, параметри фізичної активності, цілі харчування, обмеження, продукти, страви, прийоми їжі, показники споживання води та персоналізовані рекомендації. Кожна таблиця містить набір атрибутів, необхідних для запису та подальшого використання відповідної інформації.

Центральним об’єктом структури слугує таблиця «Users», яка містить основну інформацію про користувачів і виконує роль зв’язувальної ланки між іншими сутностями системи. Для відстеження фізичних показників користувачів створено таблицю «UserMeasurements», що містить дані про результати вимірювань ваги та інших параметрів із зазначенням часу їх фіксації. Типи

вимірювань класифіковано в окремій таблиці «MeasurementTypes», яка забезпечує гнучкість і масштабованість системи з огляду на можливість додавання нових типів показників.

З метою реалізації механізму персоналізації раціону в базі даних передбачено таблиці «UserGoals», «DailyNorms» та «DailyNormNutrients». Вони забезпечують збереження даних про індивідуальні цілі користувачів, добові норми калорійності та оптимальний склад нутрієнтів. Структуру взаємозв'язків між сутностями, що забезпечують збереження персональних даних користувача, його фізичних параметрів, цілей та добових норм, наведено на рисунку 2.1.

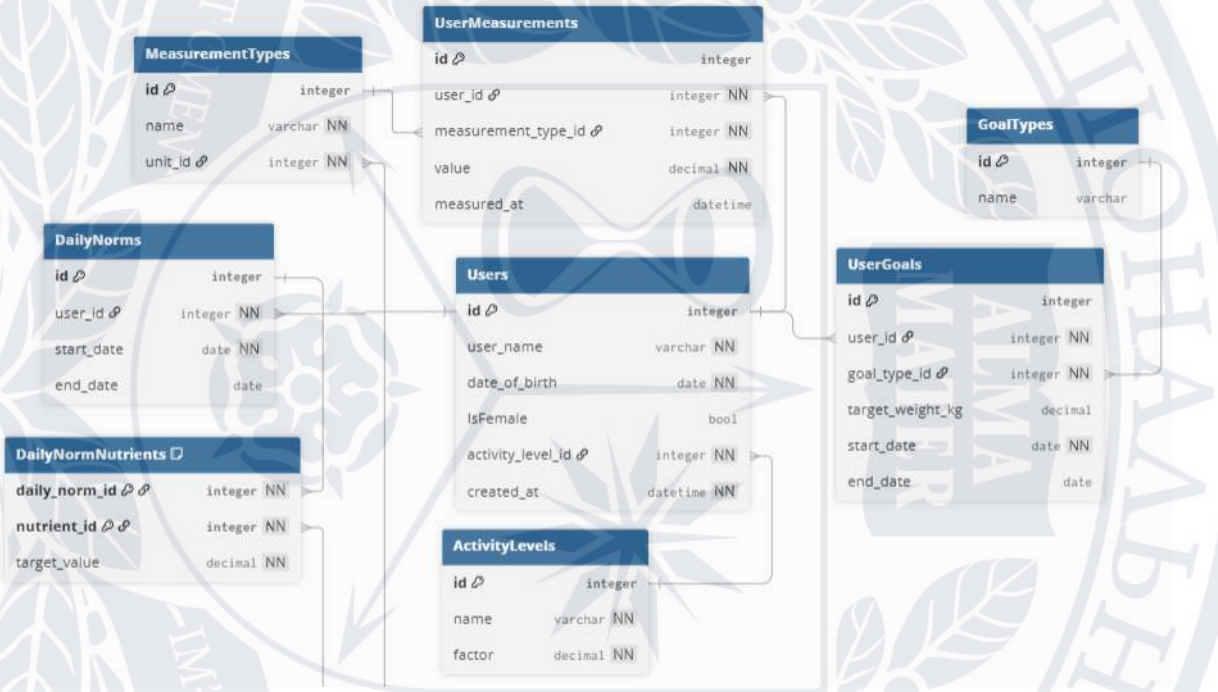

Рисунок 2.1 – Фрагмент структури бази даних модуля користувача та моніторингу фізичних показників

Для опису харчових характеристик продуктів передбачено таблицю «Nutrients», а їхній зв'язок із продуктами організовано через проміжну таблицю «ProductNutrients», яка дозволяє реалізувати відносини типу «багато-до-багатьох».

Облік харчування реалізовано через комплекс таблиць (рис. 2.2), зокрема «Products», «Dishes», «DishIngredients», «Meals» та «MealItem». Зокрема, зв'язок між стравами та продуктами реалізовано через таблицю «DishIngredients», що дозволяє зберігати склад кожної страви із зазначенням кількості інгредієнтів та

одиниць вимірювання. Таблиця «MealItem» забезпечує можливість збереження як окремих продуктів, так і готових страв у складі прийому їжі користувача.

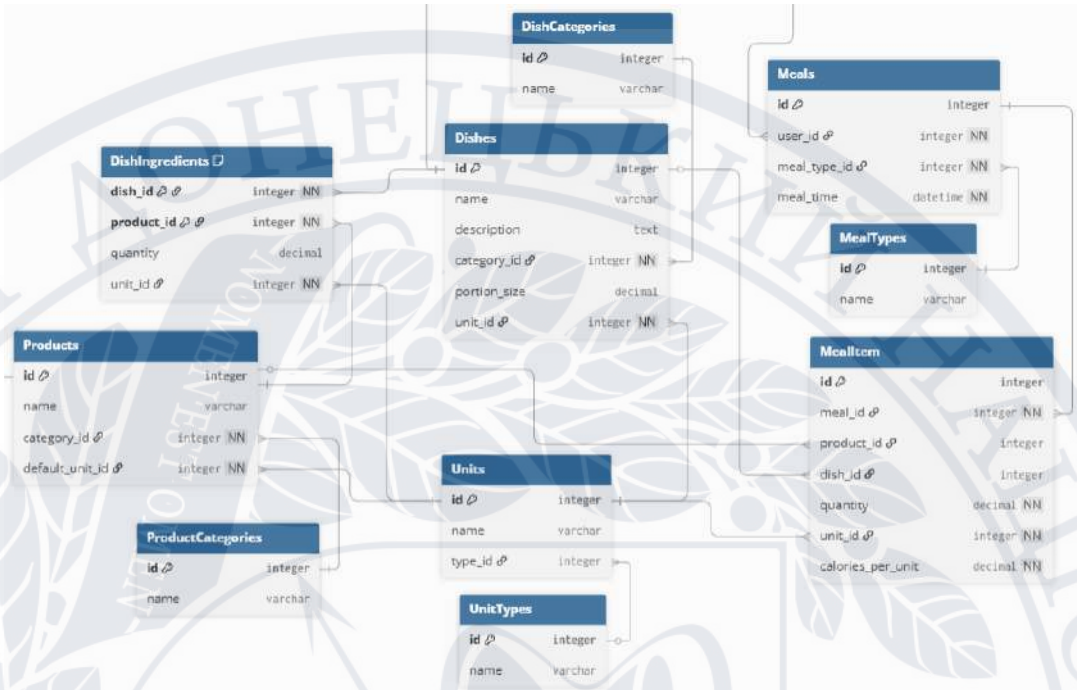

Рисунок 2.2 – Фрагмент схеми бази даних підсистеми обліку харчування

Враховуючи необхідність адаптації під індивідуальні потреби користувачів, зокрема можливі алергії чи медичні протипоказання, до структури бази даних внесено таблиці «Restrictions», «RestrictionTypes» та «UserRestrictions». Вони дозволяють гнучко описувати різноманітні типи обмежень і визначати ступінь їх значущості. Схему організації даних наведено на рисунку 2.3.

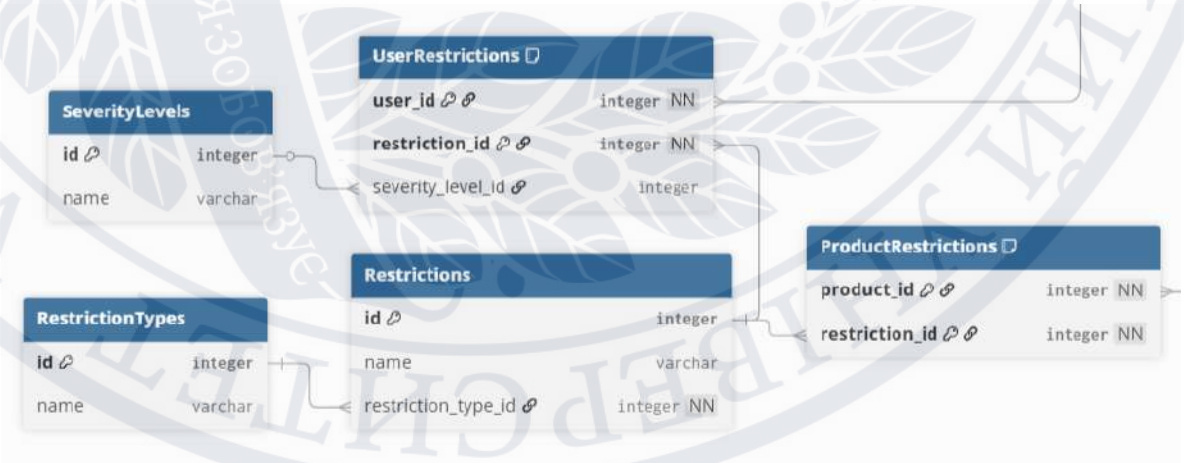

Рисунок 2.3 – Фрагмент схеми бази даних підсистеми обмежень та харчових алергенів

Для підтримки цілісності та узгодженості даних у структурі бази було застосовано механізми первинних та зовнішніх ключів, а також проміжні таблиці, необхідні для організації зв’язків типу «багато-до-багатьох» [27]. Використання такого підходу дозволяє мінімізувати дублювання інформації, сприяє нормалізації структури даних та підвищує ефективність виконання SQL-запитів.

Структура розробленої бази даних характеризується високим рівнем масштабованості та забезпечує можливість подальшого розширення функціональних можливостей системи. Зокрема, структура дозволяє додавати нові типи рекомендацій, нутрієнтів, категорії продуктів, а також реалізовувати інтеграцію із зовнішніми сервісами без суттєвого порушення існуючих зв’язків між сутностями. Крім того, така організація даних створює основу для впровадження додаткових аналітичних інструментів, механізмів автоматизованого формування рекомендацій та синхронізації даних із зовнішніми інформаційними системами [30].

Повна схема бази даних наведена у додатку А.

2.3 Архітектурне проєктування системи

2.3.1 Вибір технологічного стеку та архітектурного патерну

У підрозділах 1.2 та 1.2.2 було проведено аналіз сучасних підходів до мобільної розробки, а також виконано порівняння поширених архітектурних патернів. Результати аналізу показали, що кросплатформний підхід є найбільш доцільним для реалізації мобільного застосунку персонального контролю харчування, оскільки забезпечує оптимальне співвідношення між продуктивністю, швидкістю розробки та витратами на підтримку системи. Крім того, використання єдиної кодової бази дозволяє спростити процес супроводу, тестування та оновлення програмного забезпечення для різних мобільних платформ.

З урахуванням визначених вимог до системи як основний інструмент

розробки було обрано фреймворк .NET MAUI [31]. Він забезпечує створення кросплатформних застосунків для Android, iOS, Windows та інших платформ із використанням спільної логіки бізнес-процесів. Серед ключових переваг .NET MAUI можна виділити інтеграцію з екосистемою .NET, підтримку сучасних механізмів побудови інтерфейсу, доступ до нативних API пристроїв, а також вбудована підтримка Dependency Injection, Data Binding та архітектурних шаблонів проєктування.

Для організації внутрішньої структури програмного забезпечення обрано архітектурний патерн MVVM (Model-View-ViewModel). Такий вибір обумовлений органічною інтеграцією MVVM із механізмами прив’язки даних (Data Binding) платформи .NET MAUI [22]. Патерн дозволяє відокремити логіку представлення від графічного інтерфейсу, зменшити рівень зв’язності між компонентами системи та підвищити модульність застосунку. Структурну взаємодію основних компонентів архітектури MVVM наведено на рисунку 2.4.

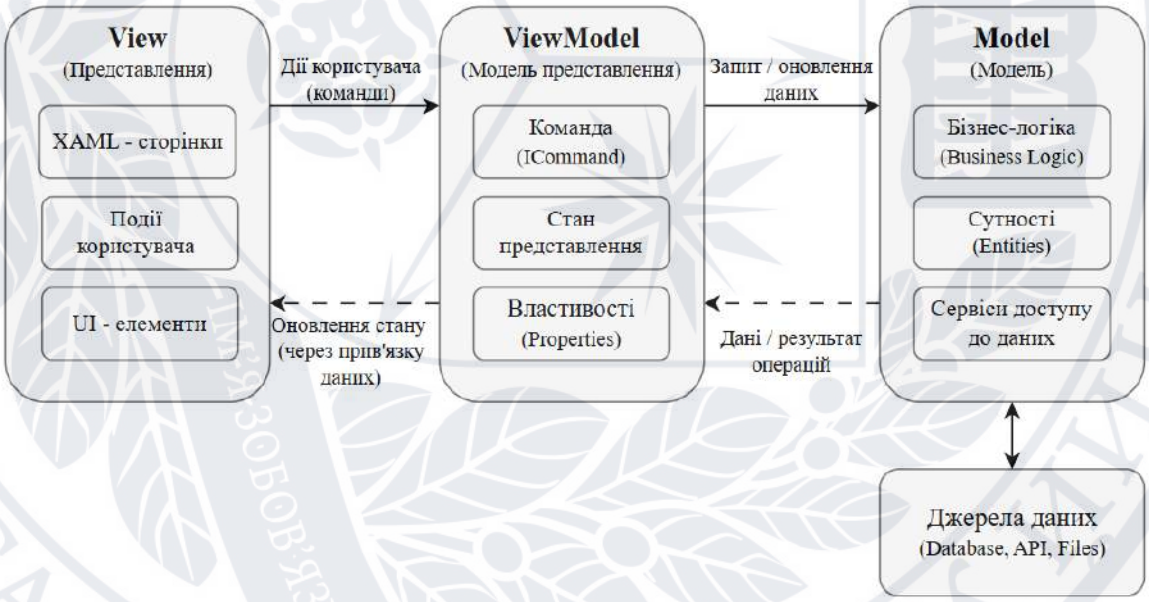

Рисунок 2.4 – Структурна схема взаємодії компонентів MVVM

Як показано на рисунку, архітектура MVVM передбачає чіткий розподіл відповідальності: компонент View відповідає за відображення користувацького інтерфейсу, ViewModel – за обробку команд і логіку представлення, тоді як Model забезпечує роботу з даними та реалізацію основних бізнес-процесів системи [32]. Взаємодія між компонентами здійснюється за допомогою

механізмів прив’язки даних і команд, що забезпечує слабку зв’язність, спрощує тестування та полегшує супровід.

2.3.2 Структура програмного забезпечення

Архітектура застосунку побудована відповідно до принципів модульності та розподілу відповідальності між функціональними рівнями. Структура програмного забезпечення поділяється на чотири основні рівні: представлення (Presentation Layer), бізнес-логіки (Business Logic Layer), доступу до даних (Data Access Layer) та керування станом (State Management Layer).

Рівень представлення включає сторінки інтерфейсу та відповідні ViewModel-компоненти, реалізовані відповідно до патерну MVVM. Він відповідає за взаємодію користувача із системою, обробку команд, відображення інформації та організацію навігації між модулями.

Рівень бізнес-логіки реалізовано у вигляді спеціалізованих сервісів, відповідальних за обробку даних, розрахунок нутрієнтів і добових норм, керування споживанням води та формування персоналізованих рекомендацій. Винесення бізнес-логіки в окремий рівень ізолює її від інтерфейсних компонентів і забезпечує повторне використання модулів у різних частинах системи.

Рівень доступу до даних забезпечує взаємодію з локальною базою даних SQLite через контекст Entity Framework Core та спеціалізовані сервіси [33]. Це централізує механізми роботи з даними, спрощує підтримку структури бази даних і забезпечує контроль цілісності інформації.

Для керування залежностями між компонентами системи використовується механізм Dependency Injection [34]. У контейнері залежностей реєструються сторінки інтерфейсу, ViewModel-компоненти, навігаційні сервіси та модулі бізнес-логіки, що забезпечує централізоване керування їх життєвим циклом і підвищує гнучкість архітектури.

2.3.3 Проєктування сценаріїв взаємодії користувача

Практична реалізація обраних архітектурних рішень передбачає структурований поділ функціональності застосунку відповідно до основних сценаріїв взаємодії користувача із системою. Такий підхід дозволяє впорядкувати обробку даних, спростити навігацію між модулями та забезпечити послідовність роботи застосунку.

Архітектура системи базується на розмежуванні двох ключових сценаріїв: процесу первинного налаштування користувача (onboarding) та основного режиму функціонування. Це дає змогу відокремити етап збору початкових даних від подальшої роботи із вже сформованим профілем, що підвищує структурованість системи та спрощує її масштабування.

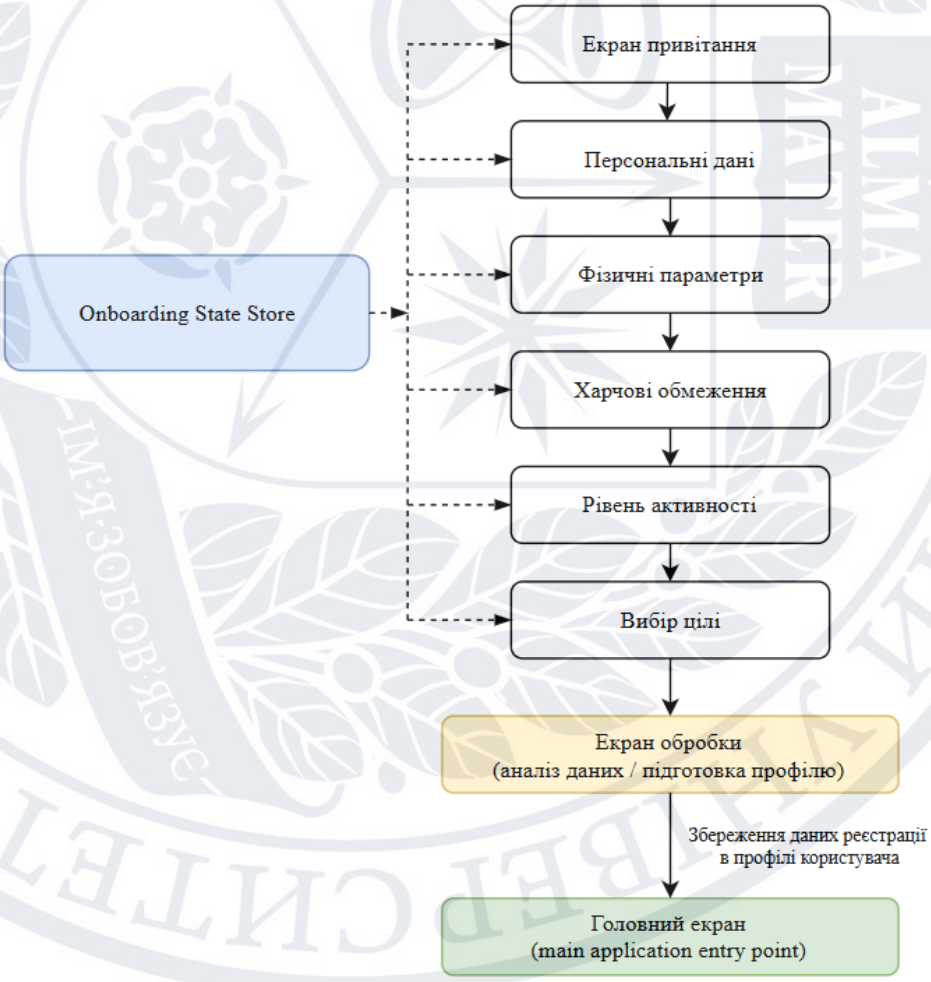

Рисунок 2.5 – Схема процесу первинного налаштування користувача (Onboarding flow)

Процес onboarding реалізовано як послідовний багатокроковий сценарій, у межах якого користувач поетапно вводить персональні дані, фізичні параметри та цілі. Навігація між етапами є лінійною, що унеможливлює пропуск обов'язкових кроків та зменшує кількість помилок введення. Завершальним етапом є обробка зібраних даних і формування профілю користувача, після чого система переходить до основного режиму роботи. Структуру процесу первинного налаштування наведено на рисунку 2.5.

У межах onboarding-процесу реалізовано централізоване тимчасове сховище стану, яке використовується для накопичення та узгодження даних, введених користувачем на різних етапах первинного налаштування. У ньому зберігаються персональні дані користувача, параметри фізичної активності, цілі та обмеження до моменту завершення процесу. Такий підхід дозволяє відокремити збір інформації від основної моделі зберігання, унеможливлює часткове збереження даних і забезпечує цілісне формування користувацького профілю (рис. 2.6).

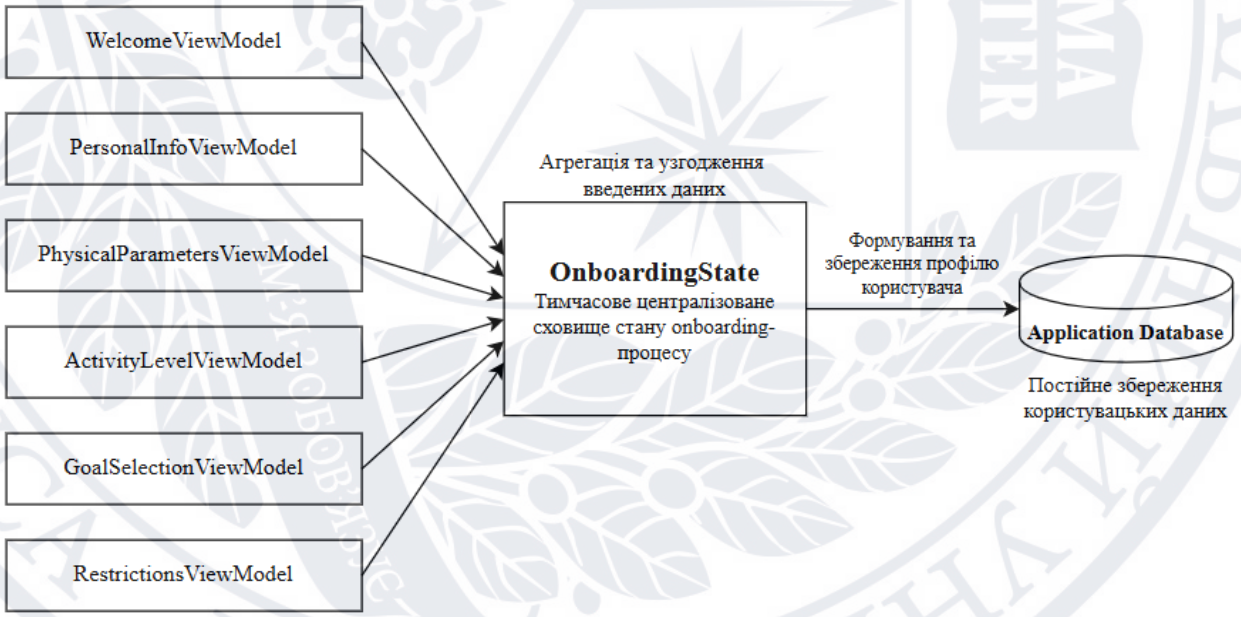

Рисунок 2.6 – Модель керування станом onboarding-процесу

Кожен етап onboarding реалізовано як окремий функціональний модуль із власною сторінкою (View), моделлю представлення (ViewModel) та взаємодією із централізованим сховищем стану – відповідно до принципів MVVM.

Навігацію в застосунку реалізовано на основі Shell-архітектури платформи

.NET MAUI із централізованою маршрутизацією [35]. Логіку переходів між екранами винесено в окремий навігаційний компонент, що ізолює її від інтерфейсного рівня, забезпечує послідовність проходження onboarding-сценарію та чітке розділення між процесом первинного налаштування й основним режимом роботи. Навігаційна структура передбачає окремі маршрути для кожного етапу onboarding та основних модулів застосунку – головного екрану, профілю користувача, плану харчування і моніторингу прогресу.

Після завершення onboarding-процесу система переходить до основного режиму роботи, у якому сформований користувацький профіль використовується як основа для подальшого функціонування застосунку без потреби повторного проходження етапу первинного налаштування (рис. 2.7).

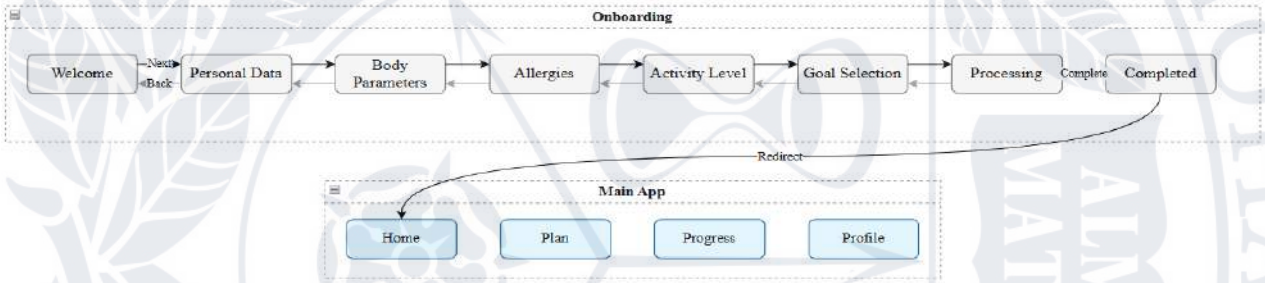

Рисунок 2.7 – Схема навігації

Таким чином, спроєктована архітектура забезпечує чітке розділення відповідальностей між компонентами системи, підтримує незалежний розвиток окремих модулів і формує надійну основу для реалізації функціональних можливостей застосунку персонального контролю харчування.

2.2.4 Алгоритмічна та математична модель формування персоналізованих рекомендацій

Формування персоналізованих порад у мобільному застосунку базується на застосуванні математичної моделі, яка бере до уваги фізіологічні ознаки користувача, особливості харчування та поставлені цілі. Основне призначення моделі полягає у формуванні адаптивного способу планування раціону, що дає змогу врахувати індивідуальні енергетичні потреби організму, підтримувати

оптимальний баланс макро- і мікронутрієнтів, а також враховувати можливі дієтичні обмеження.

Запропонований підхід належить до класу детермінованих rule-based систем підтримки прийняття рішень, у яких результати формуються на основі заздалегідь визначених математичних моделей та логічних правил обробки вхідних фізіологічних параметрів користувача, без використання ймовірнісних або стохастичних методів. Це забезпечує прозорість алгоритмічної роботи, відтворюваність отриманих результатів та полегшує інтерпретацію процесу формування рекомендацій.

Для формалізованого опису алгоритмічної моделі введемо множини вхідних та вихідних даних. Вхідні дані моделі можна представити у вигляді вектора:

$$X = \{age, gender, weight, height, activityLevel, goal, restrictions\} \quad (2.1)$$

де параметри визначені наступним чином:

- age – вік користувача;
- $gender$ – стать;
- $weight$ – маса тіла;
- $height$ – зріст;
- $activityLevel$ – рівень фізичної активності;
- $goal$ – ціль (зниження, підтримка або набір маси);
- $restrictions$ – харчові обмеження та алергени.

Вихідні дані моделі формуються у вигляді множини:

$$Y = \{calories, protein, fat, carbs, water, recommendations\} \quad (2.2)$$

де:

- $calories$ – добова енергетична потреба;
- $protein, fat, carbs$ – рекомендовані значення макронутрієнтів;
- $water$ – добова потреба у споживанні води;
- $recommendations$ – набір персоналізованих рекомендацій;

Таким чином модель можна представити як відображення:

$$Y = f(X) \tag{2.3}$$

де f – це детермінована функція, що реалізує алгоритм розрахунку та формування рекомендацій.

Модель функціонує на основі поетапного розрахунку ключових фізіологічних параметрів, зокрема індексу маси тіла (BMI), базового метаболічного обміну (BMR) та загальної добової енергетичної потреби (TDEE).

Індекс маси тіла (Body Mass Index, BMI) використовується для оцінювання співвідношення маси тіла та зросту користувача. Отримане значення дає змогу оцінити відповідність маси тіла фізіологічним нормам та використовується для подальшого коригування нутріційних параметрів, зокрема рекомендованих норм споживання білків та жирів.

Розрахунок BMI здійснюється за формулою [36]:

$$BMI = \frac{weight}{height^2} \tag{2.4}$$

де $weight$ – маса тіла користувача у кілограмах, а $height^2$ – зріст у метрах.

Наступним етапом є визначення базового метаболічного обміну (Basal Metabolic Rate, BMR). Показник BMR характеризує мінімальну кількість енергії, необхідної організму для підтримання основних життєвих функцій у стані спокою, зокрема роботи серцево-судинної системи, дихання, терморегуляції та функціонування внутрішніх органів.

Розрахунок базового метаболічного обміну здійснюється за формулою Mifflin-St Jeor [37]:

$$BMR = 10 \times weight + 6.25 \times height - 5 \times age + s \tag{2.5}$$

де:

- $weight$ – маса тіла (кг);
- $height$ – зріст (см);
- age – вік (роки);
- s – коефіцієнт, що залежить від статі (для чоловіків $s = 5$, для жінок $s = -161$).

Після визначення BMR система виконує розрахунок загальної добової

енергетичної потреби (TDEE) [38]:

$$TDEE = BMR \times k_{activity} \quad (2.6)$$

Показник TDEE характеризує загальні добові витрати енергії організму з урахуванням рівня фізичної активності користувача. Для розрахунку використовується коефіцієнт фізичної активності $k_{activity}$, значення якого визначається відповідно до способу життя та інтенсивності фізичних навантажень. Значення коефіцієнтів наведено в таблиці 2.2:

Таблиця 2.2 – Коефіцієнти фізичної активності

Рівень фізичної активності	Опис	k
Малорухливий спосіб життя	Відсутність або мінімальна фізична активність	1,2
Низька активність	Легкі навантаження 1-3 рази на тиждень	1,375
Помірна активність	Тренування 3-5 разів на тиждень	1,55
Висока активність	Інтенсивні навантаження 6-7 разів на тиждень	1,725

Для врахування поставленої цілі користувача застосовуються коригувальний коефіцієнт [37]:

$$Calories = TDEE \times k_{goal} \quad (2.7)$$

де :

- k_{goal} = 0,85 – для зниження маси тіла;
- k_{goal} = 1,00 – для підтримання маси;
- k_{goal} = 1,10 – для набору маси тіла.

Отримане значення визначає рекомендовану добову калорійність раціону користувача. Для забезпечення безпечного рівня харчування модель передбачає мінімально допустимі значення калорійності: 1200 ккал для жінок та 1500 ккал для чоловіків.

Після визначення енергетичної потреби система здійснює розрахунок

рекомендованої кількості макронутрієнтів. Кількість білків та жирів визначається у грамах на кілограм маси тіла з урахуванням цілі користувача та показника BMI [6]:

$$Protein = weight \times p \tag{2.8}$$

$$Fat = weight \times f \tag{2.9}$$

де p – рекомендована норма білка, f – рекомендована норма жирів.

Кількість вуглеводів визначається залишковим методом [7]:

$$Carbs = \frac{Calories - (Protein \times 4 + Fat \times 9)}{4} \tag{2.10}$$

У наведеній формулі враховано енергетичну цінність макронутрієнтів: 1 г білків і вуглеводів відповідає 4 ккал, а 1 г жирів – 9 ккал. Додатково, алгоритм контролює мінімально допустимий рівень споживання вуглеводів залежно від рівня фізичної активності користувача.

Окремим етапом виконується розрахунок добової потреби у воді (мл) [37]:

$$Water = weight \times 35 \tag{2.11}$$

де добова потреба у воді визначається з розрахунку 35 мл на 1 кг маси тіла користувача. У випадку підвищеної фізичної активності отримане значення додатково коригується.

Окрім макронутрієнтів, алгоритмічна модель також передбачає розрахунок рекомендованих норм окремих мікронутрієнтів, зокрема харчових волокон, цукрів, вітамінів та мінеральних речовин. Розрахунок виконується на основі добової калорійності раціону, статі користувача та рівня фізичної активності.

Наприклад, добова рекомендована кількість харчових волокон визначається пропорційно енергетичній потребі користувача [36]:

$$Fiber = 14 \times \frac{Calories}{1000} \tag{2.12}$$

де $Calories$ – добова калорійність раціону користувача; 14 – рекомендована кількість грамів клітковини на 1000 ккал.

Розраховані показники адаптуються з урахуванням рівня фізичної активності користувача, фізіологічних характеристик (вік, стать, вага, зріст), а

також поставлених цілей – зниження, підтримання або збільшення маси тіла. Після розрахунку нормативних величин система порівнює отримані показники з фактичними даними про харчування користувача, зокрема добовою калорійністю, частотою прийомів їжі та складом раціону. Це дозволяє виявити відхилення у загальному енергетичному балансі та структурі харчування.

На основі проведеного аналізу система формує персоналізовані рекомендації для корекції харчової поведінки та оптимізації нутріційного статусу користувача. Рекомендації охоплюють як загальний енергетичний баланс, так і пропорції макронутрієнтів – білків, жирів і вуглеводів – з конкретними орієнтирами у відсотках від добової потреби або у грамах на кілограм маси тіла.

Крім того, алгоритм враховує додаткові чинники, зокрема наявність алергенів, харчових обмежень (вегетаріанство, пескетаріанство, медичні дієти), особливості способу життя, рівень споживання рідини й гідратаційні потреби. Прикладом практичної рекомендації може бути збільшення частки білків у раціоні для користувача, який прагне набору м’язової маси, або зменшення кількості простих вуглеводів та збільшення споживання клітковини для підтримання стабільного рівня глюкози в крові.

Система також формує рекомендації щодо режиму харчування, планування прийомів їжі та поступової зміни харчової поведінки. Це забезпечує комплексний підхід до формування персоналізованих нутріційних рекомендацій та підвищує ефективність адаптації раціону відповідно до індивідуальних потреб користувача.

На основі розробленої математичної моделі було побудовано діаграму активностей, яка відображає послідовність взаємодії користувача із мобільним застосунком та основні етапи формування персоналізованих рекомендацій (рис. 2.8). Діаграма демонструє процес введення та перевірки персональних даних користувача, врахування харчових обмежень, вибору цілі харчування, розрахунку енергетичних потреб та генерації індивідуального плану харчування.

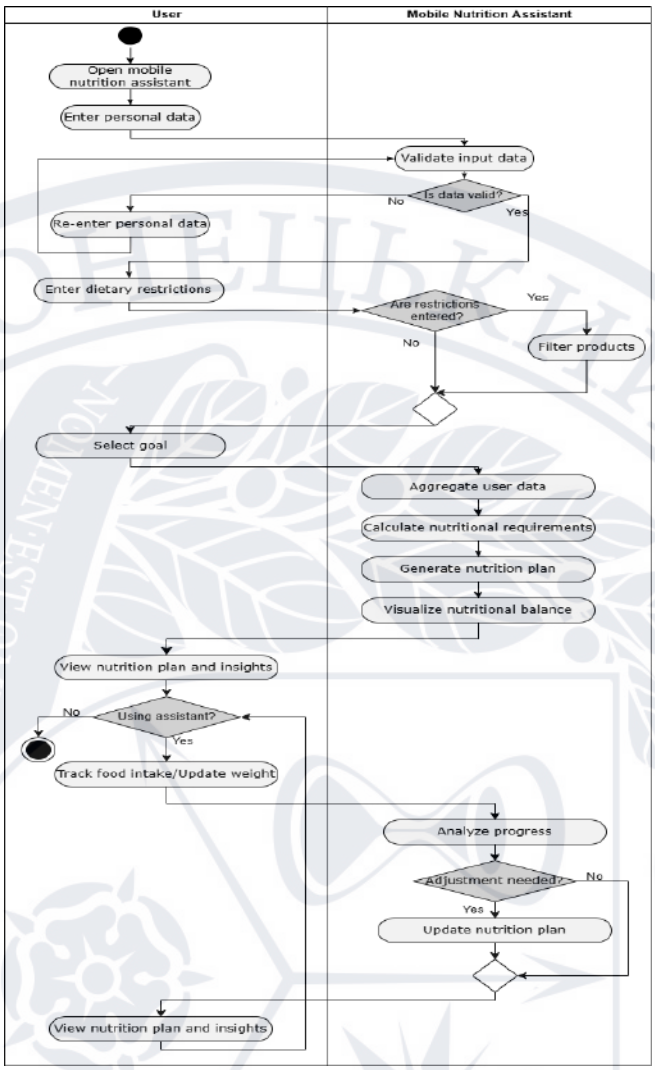

Рисунок 2.8 – Діаграма активностей процесу формування персоналізованих нутріційних рекомендацій [2]

Крім того, модель передбачає можливість подальшого відстеження прогресу користувача, аналізу змін показників та автоматичного коригування рекомендацій відповідно до оновлених даних.

Представлена діаграма активностей формалізує основні етапи роботи системи та відображає логіку функціонування алгоритму персоналізованого формування нутріційних рекомендацій. Використання такого підходу забезпечує структурованість процесу взаємодії користувача із системою та спрощує подальшу реалізацію програмної логіки мобільного застосунку.

Висновок до другого розділу

У даному розділі здійснено проєктування мобільної системи персонального контролю харчування, що дозволило сформувати її

концептуальну та архітектурну основу. Визначено функціональні та нефункціональні вимоги до системи, які охоплюють облік харчування користувача, розрахунок добових енергетичних потреб і макронутрієнтів, моніторинг фізіологічних показників, а також формування персоналізованих рекомендацій щодо корекції раціону та харчових звичок.

На основі аналізу предметної області було виділено ключові інформаційні сутності та встановлено зв'язки між ними, що стало основою для проєктування структури реляційної бази даних. Обґрунтовано вибір СУБД SQLite як оптимального рішення для мобільного середовища, що забезпечує локальне зберігання даних, цілісність інформації та достатню продуктивність для задач застосунку.

У межах архітектурного проєктування обрано кросплатформний підхід із використанням .NET MAUI та патерну MVVM, що забезпечує розділення логіки, представлення та даних, підвищує модульність системи та спрощує подальше масштабування і підтримку програмного забезпечення. Окремо спроєктовано процес первинної конфігурації користувача (onboarding), який забезпечує структуроване формування профілю та введення необхідних вихідних даних для подальших розрахунків.

Ключовим результатом розділу є розробка детермінованої математичної моделі формування персоналізованих нутриційних рекомендацій. Модель базується на поетапному обчисленні BMI, BMR та TDEE з урахуванням фізіологічних параметрів користувача, рівня фізичної активності, цілей харчування та індивідуальних обмежень. Це забезпечує формування узгоджених рекомендацій щодо добової калорійності, співвідношення макронутрієнтів, режиму харчування та водного балансу.

Отже, результати проєктування формують цілісну основу для подальшої реалізації програмного забезпечення та забезпечують передумови для створення функціональної, масштабованої та адаптивної мобільної системи персоналізованого контролю харчування.

РОЗДІЛ 3.
ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ПОМІЧНИКА

3.1 Побудова інтерфейсу користувача

Інтерфейс користувача є одним із ключових компонентів мобільного застосунку, оскільки саме він забезпечує взаємодію із системою та безпосередньо впливає на ефективність і зручність її використання [39]. Тому його якісне проєктування є вирішальним фактором у забезпеченні позитивного досвіду. Сучасний інтерфейс має бути інтуїтивно зрозумілим, функціональним, привабливим з візуальної точки зору та відповідати актуальним стандартам користувацького досвіду [40]. Особливого значення це набуває під час розробки мобільних застосунків, де важливими є зручність введення даних, адаптивність інтерфейсу та швидкий доступ до функцій.

Одним із важливих елементів візуальної складової інтерфейсу є фірмовий стиль застосунку, зокрема його логотип, який формує перше враження користувача та передає основну ідею продукту. Для мобільного помічника персонального контролю харчування “EatSmart Balance” було розроблено логотип, який поєднує елементи здорового способу життя, гармонії та турботи про власне здоров’я (рис.3.1). Центральним елементом логотипу виступає стилізована фігура людини з піднятими руками – символ активності, життєвої енергії, прагнення до підтримки здорового способу життя.

Рисунок 3.1 – Логотип та назва застосунку

Основний елемент розміщено в межах абстрактної форми, що поєднує

риси листка та плавних округлих форм. Така композиція викликає асоціації з природністю і внутрішньою гармонією, що безпосередньо відображає концепцію балансу, закладену в назву продукту.

Кольорова гама базується на відтінках зеленого, які традиційно асоціюються зі здоров'ям, спокоєм та свіжістю. Темно-зелений колір передає відчуття стабільності та надійності, тоді як світло-зелений акцентує увагу на розвитку, оновленні та позитивних змінах.

Завдяки простоті та мінімалістичності логотип добре масштабується та зберігає впізнаваність на різних екранах мобільних пристроїв. Графічний елемент може використовуватися автономно як іконка продукту, тоді як повна комбінація з текстовою частиною слугує основним фірмовим знаком для його ідентифікації. Це забезпечує ефективність його використання як складової загальної концепції інтерфейсу EatSmart Balance.

Окрім візуальної складової, важливу роль у формуванні користувацького досвіду відіграє функціональна організація користувацького середовища. Одним із ключових елементів такої організації є процес онбордингу, який забезпечує початкове налаштування застосунку та збір базової інформації про користувача. Завдяки цьому підходу система може адаптуватися до індивідуальних особливостей, цілей і потреб кожного користувача, надаючи персоналізований досвід взаємодії.

Візуальне оформлення процесу онбордингу, представлене на рисунку 3.2, виконане у мінімалістичному стилі з використанням фірмової кольорової палітри. Вітальний екран зосереджує увагу користувача на основних елементах – логотипі та основній кнопці, що формує чітку та зрозумілу точку входу в систему. Подальші етапи онбордингу демонструють використання інтуїтивно зрозумілих елементів взаємодії, зокрема текстових полів, а також інтерактивних карток.

Застосування індикатора процесу, розміщеного у верхній частині екрана, забезпечує можливість візуального контролю за проходженням етапів збору інформації, що, відповідно до принципів UX-дизайну, сприяє підвищенню рівня

завершення сценаріїв користувача [41].

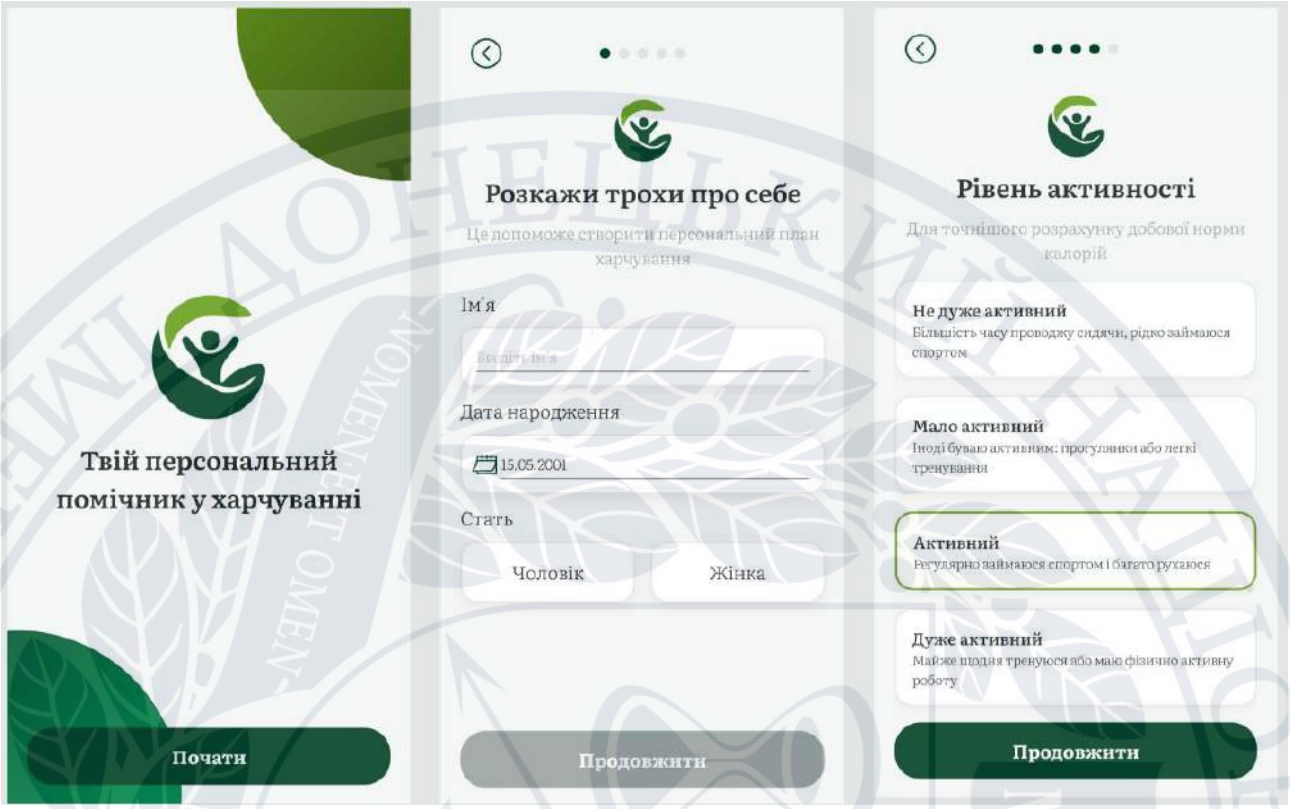

Рисунок 3.2 – Приклад онбордингу в мобільному застосунку EatSmart Balance

Ергономічність інтерфейсу досягається завдяки використанню великих елементів керування, зокрема кнопок навігації, а також раціональному застосуванню вільного простору. Такий підхід дозволяє уникнути перевантаження інтерфейсу та забезпечує комфортну взаємодію користувача з системою.

3.2 Впровадження ключових функцій системи

Після завершення етапу проєктування архітектури, структури бази даних та алгоритмічної моделі було здійснено програмну реалізацію основних функціональних можливостей мобільного помічника для персонального контролю харчування EatSmart Balance. Практична реалізація системи виконувалася відповідно до визначених функціональних і нефункціональних вимог, а також із дотриманням обраного архітектурного підходу на основі патерну MVVM.

У межах реалізації програмного забезпечення було розроблено механізми локального збереження та обробки даних, процес первинного налаштування користувача, модулі обліку харчування, формування персоналізованих рекомендацій, моніторингу фізіологічних показників і статистичного аналізу. Особливу увагу приділено забезпеченню модульності системи, розподілу відповідальностей між компонентами.

3.2.1 Організація локальної бази даних та механізму збереження даних

Для забезпечення стабільної роботи мобільного застосунку була впроваджена локальна система збереження даних на основі реляційної бази даних SQLite [42]. Використання цієї технології дає змогу забезпечити автономне функціонування застосунку без необхідності постійного підключення до зовнішнього сервера, що є важливим для мобільних систем персонального користування. Крім того, зберігання даних локально підвищує швидкодію застосунку та гарантує захист особистої інформації користувача завдяки її зберіганню безпосередньо на пристрої [43].

Для взаємодії із базою даних використовується технологія Entity Framework Core, яка впроваджує ORM-підхід (Object-Relational Mapping) [44]. Це забезпечує зручну взаємодію з реляційними структурами даних через об'єктні моделі, притаманні мові програмування C#. Такий метод дозволяє спростити реалізацію CRUD-операцій, централізувати доступ до даних та мінімізувати кількість низькорівневих SQL-запитів у коді додатка.

Ключовим елементом підсистеми доступу до даних є контекст бази даних AppDbContext. Він успадковується від класу DbContext і включає набори сутностей (DbSet) [45], які використовуються для роботи з головними об’єктами системи: користувачами, продуктами, стравами, прийомами їжі, нутрієнтами, обмеженнями та рекомендаціями. У лістингу 3.1 можемо бачити приклад реалізації контексту бази даних.

Лістинг 3.1 – Фрагмент реалізації AppDbContext

```
public class AppDbContext : DbContext
{
    public DbSet<User> Users => Set<User>();
    public DbSet<Product> Products => Set<Product>();
    public DbSet<Meal> Meals => Set<Meal>();
    public DbSet<MealItem> MealItems => Set<MealItem>();
    public DbSet<DailyNorm> DailyNorms => Set<DailyNorm>();

    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.ApplyConfigurationsFromAssembly(
            typeof(AppDbContext).Assembly);
    }
}
```

Використання окремих моделей сутностей дозволило забезпечити структуроване представлення даних предметної області у програмному середовищі. Кожна сутність містить набір властивостей та навігаційних зв’язків, необхідних для реалізації відношень між таблицями бази даних. Зокрема, сутність користувача містить зв’язки із записами про прийоми їжі, добові норми, фізіологічні показники, рекомендації та харчові обмеження (лістинг 3.2).

Лістинг 3.2 – Приклад реалізації моделі користувача

```
public class User
{
    public int Id { get; set; }
    public string Name { get; set; } = string.Empty;
    public DateTime DateOfBirth { get; set; }
    public bool IsFemale { get; set; }
    public int ActivityLevelId { get; set; }
    public ActivityLevel ActivityLevel { get; set; } = null!;
    public DateTime CreatedAt { get; set; }
}
```

Для забезпечення гнучкості структури бази даних та ефективного управління складними зв’язками між сутностями було впроваджено проміжні таблиці, які реалізують відношення типу «багато-до-багатьох». Одним із прикладів є зв’язок між продуктами та нутрієнтами, організований за допомогою сутності ProductNutrient. Цей підхід дозволяє зберігати дані про склад продукту та кількість окремих нутрієнтів, що містяться у 100 грамах продукту. Такий підхід ілюструється через приклад реалізації проміжної сутності (лістинг 3.3):

Лістинг 3.3 – Приклад реалізації сутності ProductNutrient

```
public class ProductNutrient
{
    public int ProductId { get; set; }
    public Product Product { get; set; } = null!;
    public int NutrientId { get; set;}
    public Nutrient Nutrient { get; set; } = null!;
    public decimal AmountPer100g { get; set; }
}
```

Конфігурація структури таблиць, обмежень та зв’язків між сутностями реалізована за допомогою Fluent API [46]. Для кожної сутності створено окремий конфігураційний клас, у якому визначаються назви таблиць, первинні ключі, індекси, типи зв’язків та початкові дані. Такий підхід дозволяє централізувати налаштування структури бази даних і спрощує подальшу підтримку програмного забезпечення. Приклад конфігурації сутності UnitType продемонстровано у лістингу 3.4:

Лістинг 3.4 – Конфігурація сутності UnitType

```
builder.ToTable("UnitTypes");
builder.HasKey(x => x.Id);
builder.Property(x => x.Name)
  .IsRequired()
  .HasMaxLength(60);
builder.HasIndex(x => x.Name)
  .IsUnique();
builder.HasData(
            new UnitType { Id = 1, Name = "Вага" },
            new UnitType { Id = 2, Name = "Об’єм" });
```

Для створення та оновлення структури бази даних використовувався механізм міграцій Entity Framework Core [47]. Міграції дозволяють автоматизувати процес формування таблиць, оновлення структури бази даних та внесення змін до схеми без необхідності ручного редагування SQL-скриптів. Такий підхід забезпечує узгодженість між програмними моделями та фізичною структурою бази даних.

Для інтеграції підсистеми доступу до даних у загальну архітектуру мобільного застосунку було використано механізм Dependency Injection, вбудований у платформу .NET MAUI. Реєстрація контексту бази даних здійснюється під час ініціалізації застосунку із передачею шляху до локального

файлу бази даних SQLite, що розміщується у внутрішньому сховищі мобільного пристрою (лістинг 3.5).

Лістинг 3.5 – Фрагмент реєстрації контексту бази даних

```
string dbPath = Path.Combine(
      FileSystem.AppDataDirectory,"eatsmart.db");
builder.Services.AddDbContext<AppDbContext>(options =>
{
    options.UseSqlite($"Data Source={dbPath}");
});
```

Під час запуску додатка автоматично перевіряється актуальність структури бази даних і здійснюється застосування доступних міграцій. Такий підхід дозволяє підтримувати синхронність між фізичною структурою бази даних і актуальними програмними моделями. Це також забезпечує автоматизацію процесу оновлення схеми даних, усуваючи потребу в ручному втручанні. Реалізація механізму автоматичного застосування міграцій передбачає:

Лістинг 3.6 – Автоматичне застосування міграцій

```
using (var scope = app.Services.CreateScope())
{
   var db = scope.ServiceProvider
      .GetRequiredService<AppDbContext>();

   db.Database.Migrate();
}
```

Для ізоляції бізнес-логіки від рівня представлення у застосунку реалізовано окремі сервісні компоненти, відповідальні за обробку та збереження даних користувача. Сервіси взаємодіють із контекстом бази даних через механізм Dependency Injection та інкапсулюють логіку роботи із сутностями системи.

Одним із ключових сервісів є UserProfileService, який забезпечує створення користувацького профілю на основі даних, отриманих у процесі onboarding. Сервіс виконує перевірку повноти введених даних, формування основного запису користувача, а також збереження пов’язаних сутностей, зокрема цілей користувача, фізіологічних показників та харчових обмежень. Для забезпечення цілісності даних використовується механізм транзакцій, який гарантує узгоджене збереження всіх взаємопов’язаних записів у межах єдиної

операції (лістинг 3.7).

Лістинг 3.7 – Фрагмент реєстрації сервісу

```
public async Task<int> CreateUserAsync(OnboardingState state)
{
    var user = new User
    {
      Name = state.Name,
      DateOfBirth = state.BirthDate.Value,
      ActivityLevelId = state.ActivityLevelId.Value
    };

    _db.Users.Add(user);
    await _db.SaveChangesAsync();
    return user.Id;
}
```

У межах реалізації сервісу використовується асинхронна взаємодія з базою даних через метод SaveChangesAsync(), що дозволяє уникнути блокування основного потоку мобільного застосунку під час виконання операцій збереження. Використання транзакційного механізму Entity Framework Core забезпечує атомарність операцій створення профілю користувача та запобігає частковому збереженню даних у випадку виникнення помилок. Такий підхід підвищує надійність системи та забезпечує узгодженість інформації між взаємопов'язаними таблицями бази даних [48].

3.2.2 Побудова механізму первинного налаштування

Однією з ключових функціональних можливостей мобільного помічника є механізм первинного налаштування користувача (onboarding), який забезпечує збір базових персональних даних, необхідних для подальшого формування персоналізованих рекомендацій. Реалізація onboarding-процесу дозволяє адаптувати роботу системи до фізіологічних параметрів користувача, його рівня активності, цілей харчування та індивідуальних обмежень.

Відповідно до архітектурних рішень, описаних у підрозділі 2.3.3, онбординг реалізовано як послідовність взаємопов'язаних екранів із використанням патерну MVVM. Кожен етап представлений окремою сторінкою інтерфейсу (View) та відповідною моделлю представлення (ViewModel), що

забезпечує розділення логіки інтерфейсу й обробки даних.

Для накопичення даних, введених користувачем на різних етапах, у застосунку реалізовано централізоване сховище стану OnboardingState. Воно використовується як тимчасова модель даних до моменту завершення первинного налаштування та створення користувацького профілю.

Використання окремого об’єкта стану дозволяє забезпечити цілісність введених даних, уникнути часткового збереження інформації та спростити взаємодію між окремими етапами. Приклад реалізації моделі стану наведено у лістингу 3.8.

Лістинг 3.8 – Фрагмент реалізації OnboardingState

```
public class OnboardingState
{
    public string? Name { get; set; }
    public DateTime? BirthDate { get; set; }
    public bool? IsFemale { get; set; }
    public decimal? Weight { get; set; }
    public double? Height { get; set; }
    public int? ActivityLevelId { get; set; }
    public int? GoalId { get; set; }
    public List<int> Restrictions { get; set; } = new();
}
```

У процесі проходження онбордингу окремі ViewModel-компоненти отримують доступ до спільного об’єкта стану через механізм Dependency Injection та оновлюють відповідні властивості після введення користувачем нових даних.

Приклад реалізації взаємодії між ViewModel та централізованим станом наведено у лістингу 3.9. У представленому фрагменті ActivityLevelViewModel забезпечує збереження вибраного рівня фізичної активності користувача у спільному об’єкті OnboardingState.

Лістинг 3.9 – Взаємодія ViewModel із OnboardingState

```
partial void OnSelectedActivityLevelIdChanged(int? value)
   { State.ActivityLevelId = value; }
```

Використання механізму прив’язки даних (Data Binding) у поєднанні з патерном MVVM дозволяє автоматично синхронізувати зміни інтерфейсу із внутрішнім станом застосунку без прямої взаємодії між елементами

представлення та бізнес-логікою.

Для уніфікації логіки навігації та повторного використання функціональності у застосунку реалізовано базовий клас BaseOnboardingViewModel, який інкапсулює спільні механізми переходу між етапами onboarding, повернення до попереднього кроку та відображення поточного прогресу проходження первинного налаштування.

Окрему роль у реалізації відіграє компонент OnboardingNavigator, який відповідає за централізоване керування послідовністю екранів та маршрутизацією між ними. Навігатор забезпечує перехід між етапами відповідно до поточного стану користувача та контролює завершення процесу первинного налаштування.

Завершальний етап реалізовано за допомогою окремої сторінки обробки даних (ProcessingPage), яка забезпечує фінальну обробку отриманої інформації, розрахунок добових норм та створення профілю користувача. Для координації взаємодії між сервісами розрахунку нутрієнтів, генерації добових норм і збереження профілю використовується ProcessingViewModel.

Під час ініціалізації завершального етапу система послідовно аналізує введені користувачем параметри, обчислює базові фізіологічні показники та формує персоналізовані добові норми. Для покращення користувацького досвіду передбачено відображення індикатора виконання та текстового статусу поточної операції (рис. 3.3) [49].

У межах ProcessingViewModel реалізовано механізм координації взаємодії між окремими сервісами застосунку. Після отримання даних із централізованого об’єкта стану здійснюється розрахунок за допомогою сервісу NutritionEngineService, формується профіль користувача та виконується збереження результатів у локальній базі даних. На завершальному етапі система створює початкові записи добових норм нутрієнтів, що надалі використовуються для генерації рекомендацій та аналітики харчування. Зокрема, логіка розрахунку нутрієнтів, збереження користувача та формування добових норм реалізована незалежно від інтерфейсного рівня, що спрощує подальшу підтримку та

масштабування застосунку.

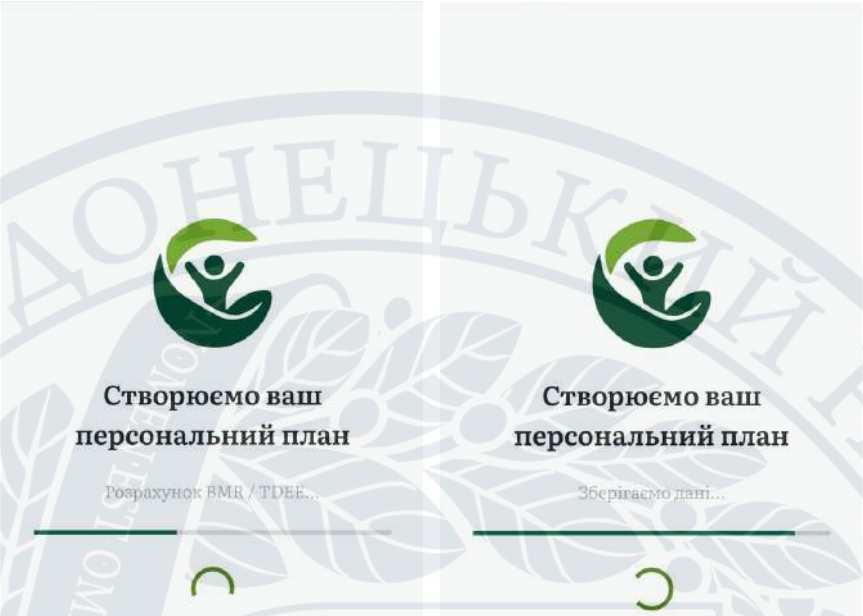

Рисунок 3.3 – Інтерфейс завершального етапу первинного налаштування застосунку

Інтерфейс екранів реалізовано за допомогою декларативної розмітки XAML із використанням механізму прив’язки даних [50]. Це дозволило забезпечити автоматичне оновлення стану інтерфейсу при зміні властивостей ViewModel. Для введення даних використовуються стандартні елементи керування .NET MAUI, зокрема текстові поля, вибір дати, інтерактивні картки та кнопки навігації.

Розроблений механізм забезпечує поетапний збір персональних даних користувача, автоматизовану перевірку коректності введених значень, централізоване накопичення стану та подальше формування персоналізованого профілю. Завдяки використанню патерну MVVM, механізмів Data Binding та сервісно-орієнтованого підходу вдалося забезпечити розширюваність, підтримуваність та модульність реалізованої підсистеми первинного налаштування.

3.2.3 Формування добових норм та персоналізованих рекомендацій

Однією з основних функціональних підсистем є механізм розрахунку добових норм харчування та формування персоналізованих рекомендацій. Його реалізація забезпечує адаптацію рекомендацій відповідно до фізіологічних параметрів користувача, рівня фізичної активності, поставлених цілей та індивідуальних особливостей профілю.

Відповідно до архітектурних рішень, описаних у попередніх розділах, розрахунок добових норм реалізовано у вигляді окремого сервісного компонента NutritionEngineService, який інкапсулює бізнес-логіку визначення енергетичних потреб користувача та формування індивідуальних цільових показників харчування.

На першому етапі система обчислює базові фізіологічні показники користувача, зокрема вік та індекс маси тіла (BMI). Для визначення базового рівня енергетичних витрат організму (BMR) у сервісі NutritionEngineService реалізовано програмну інтерпретацію формули Mifflin–St Jeor. Фрагмент реалізації відповідного методу наведено у лістингу 3.10:

Лістинг 3.10 – Фрагмент реалізації розрахунку BMR

```
private double CalculateBmr(
    double weight,
    double height,
    int age,
    bool isFemale)
{
    double baseValue =
        10 * weight +
        6.25 * height -
        5 * age;

    return isFemale
        ? baseValue - 161
        : baseValue + 5;
}
```

Після визначення базового обміну речовин система виконує розрахунок загальних добових енергетичних витрат (TDEE) з урахуванням коефіцієнта фізичної активності. Для цього використовується набір коефіцієнтів, що відповідають різним рівням активності – від малорухомого способу життя до

високих фізичних навантажень.

Після розрахунку загальної калорійності визначаються цільові показники нутрієнтів. Розрахунок здійснюється з урахуванням індивідуальних показників. При цьому використовуються окремі коефіцієнти для кожного нутрієнта, що дозволяє формувати більш адаптивні рекомендації (рис. 3.4).

Рисунок 3.4 – Відображення добових норм та прогресу їх виконання на головному екрані застосунку

У реалізованій підсистемі також передбачено обчислення рекомендованого об’єму споживання води залежно від маси тіла та рівня фізичної активності користувача. Отримані результати округлюються та об’єднуються у структуру NutritionResult, яка використовується іншими компонентами системи.

Для збереження розрахованих добових норм у базі даних реалізовано окремий сервіс DailyNormService, який забезпечує створення нового запису добової норми та пов’язаних нутрієнтних показників користувача. У разі наявності попередньої активної норми система автоматично завершує її дію шляхом встановлення кінцевої дати актуальності.

Окрім формування добових норм, у застосунку реалізовано процес

розрахунку нутрієнтного складу окремих продуктів та страв. Для цього використовується інформація про харчову цінність продуктів у розрахунку на 100 грамів. Під час обчислення враховується вага інгредієнтів, а показники автоматично масштабуються відповідно до фактичної порції.

Також було розроблено формування персоналізованих рекомендацій харчування. Генерація рекомендацій здійснюється сервісом PlanService, який підбирає страви відповідно до встановлених добових показників, типу прийому їжі та індивідуальних харчових обмежень користувача (рис. 3.5).

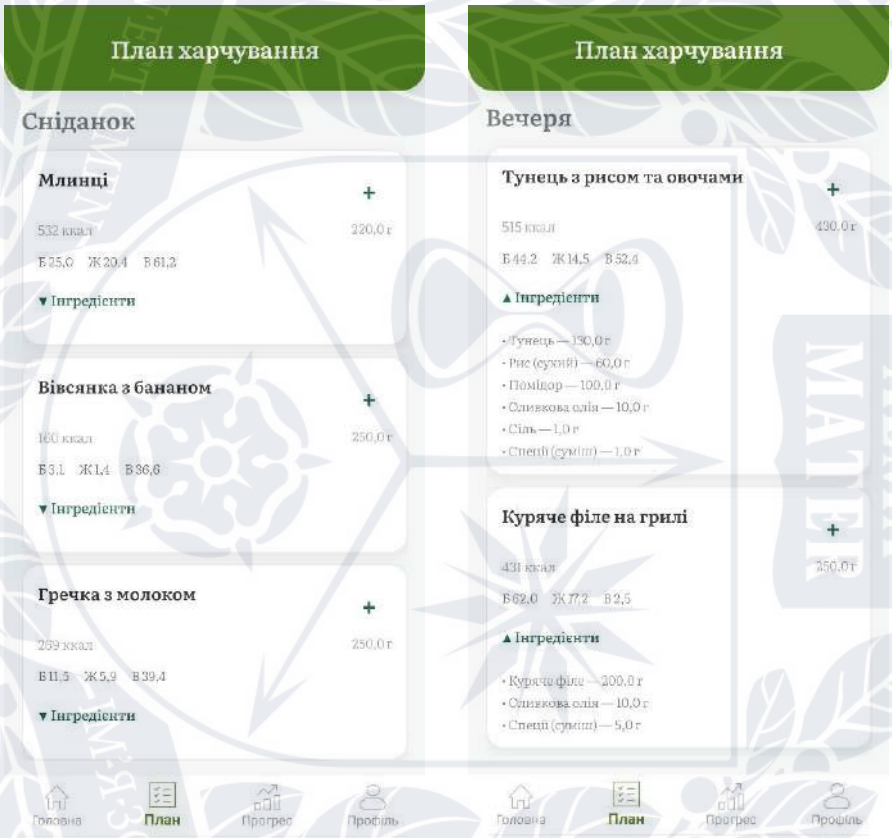

Рисунок 3.5 – Приклад згенерованого персоналізованого плану харчування на день

Для забезпечення гнучкості користувацького вибору формується декілька альтернативних варіантів страв для кожного основного прийому їжі, зокрема сніданку, обіду, вечері та перекусу.

Для оцінювання відповідності страв цільовим показникам використовується ранжування, яке враховує калорійність та співвідношення основних нутрієнтів. Кожна страва отримує інтегральну оцінку відповідності, після чого система формує набір найбільш релевантних рекомендацій для

конкретного прийому їжі.

Додатково у застосунку створено процес фільтрації страв відповідно до харчових обмежень користувача. Під час формування рекомендацій аналізується склад інгредієнтів та автоматично виключаються страви, які містять несумісні продукти або суперечать обраним обмеженням.

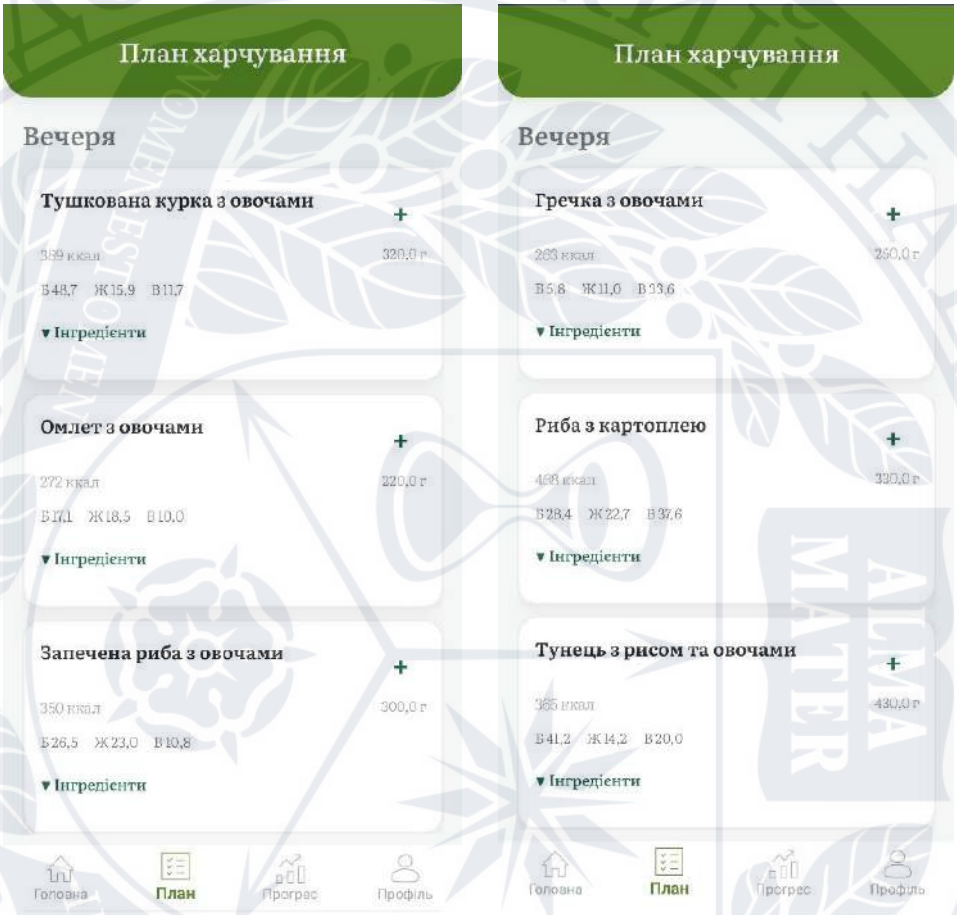

Рисунок 3.6 – Порівняння планів харчування для користувачів із різними харчовими обмеженнями

Для демонстрації роботи персоналізації було показано два варіанти рекомендацій: для користувача без харчових обмежень та для користувача, який дотримується пескетаріанського типу харчування (рис.3.6). У другому випадку система виконує фільтрацію страв, що містять м’ясні продукти, та формує альтернативний план харчування відповідно до заданих обмежень.

Такий підхід дозволяє забезпечити персоналізацію рекомендацій, автоматизацію процесу формування раціону та адаптацію харчового плану відповідно до індивідуальних потреб користувача.

3.2.4 Облік харчування та моніторинг водного балансу

Однією з ключових функціональних можливостей мобільного помічника є механізм щоденного обліку спожитої їжі та контролю водного балансу. Реалізація цієї підсистеми забезпечує можливість накопичення інформації про харчування користувача, автоматичний розрахунок фактично спожитих нутрієнтів, а також моніторинг виконання персональних добових норм.

Функціональність обліку харчування побудована відповідно до архітектурного підходу MVVM із розподілом відповідальностей між сервісним рівнем, моделями представлення та компонентами інтерфейсу. Для збереження інформації про прийоми їжі використовується сутність, яка містить записи про тип прийому їжі, дату споживання та перелік доданих продуктів або страв (рис.3.7).

Рисунок 3.7 – Інтерфейс журналу харчування з можливістю додавання продуктів та страв

Основна логіка додавання продуктів та страв реалізована у сервісі MealLogService. Сервіс забезпечує створення нового запису прийому їжі або використання вже існуючого запису за поточну дату. Після цього до відповідного прийому їжі додаються окремі елементи, що містять інформацію про продукт, його кількість та харчову цінність.

Перед збереженням даних система автоматично виконує розрахунок калорійності та макронутрієнтів на основі кількості продукту та значень нутрієнтів, що зберігаються у базі даних для кожного продукту. Такий підхід дозволяє мінімізувати повторні обчислення під час подальшого аналізу статистики та формування звітів.

Для забезпечення гнучкості користувацького сценарію реалізовано можливість додавання як окремих продуктів, так і готових страв. При роботі зі стравами система автоматично використовує дані про інгредієнти та їх харчову цінність для розрахунку сумарних показників страви.

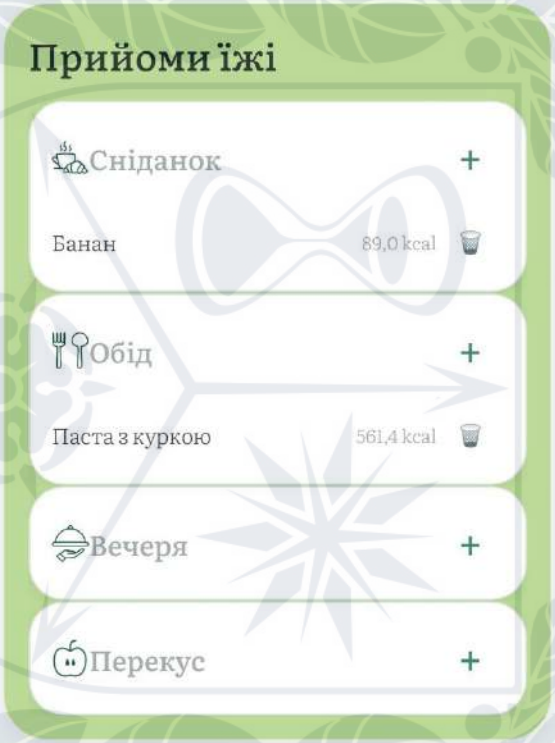

Рисунок 3.8 – Приклад редагування журналу харчування та видалення записів

Додатково реалізовано механізм редагування журналу харчування, який дозволяє користувачу видаляти окремі елементи прийому їжі (рис.3.8). Це забезпечує можливість коригування помилково доданих записів та підтримання актуальності статистики спожитих нутрієнтів.

Взаємодія між інтерфейсом користувача та сервісами обліку реалізована через модель представлення MealCardViewModel, яка відповідає за відображення окремого прийому їжі та списку доданих елементів. ViewModel

забезпечує взаємодію із сервісом вибору продуктів, обробку дій користувача та оновлення інтерфейсу після додавання нових записів.

Центральним компонентом відображення статистики є HomeViewModel, який виконує агрегацію даних про всі прийоми їжі користувача за поточний день. Під час оновлення даних система обчислює сумарне споживання нутрієнтів шляхом підсумовування значень усіх елементів

Отримані результати використовуються для формування індикаторів прогресу виконання добових норм та подальшого відображення у користувацькому інтерфейсі. Для кожного макронутрієнта система зберігає як цільове значення, так і поточний рівень споживання, що дозволяє реалізувати динамічний моніторинг харчування в режимі реального часу.

Окрім механізму обліку харчування, у застосунку реалізовано підсистему контролю водного балансу. Вона забезпечує можливість фіксації спожитої води, автоматичного визначення добової цілі та візуального контролю прогресу.

Для збереження інформації про спожиту воду використовується сутність WaterLog, яка містить обсяг спожитої рідини та час відповідного запису. Операції додавання та видалення записів реалізовані через сервіс WaterCommandService.

3.2.5 Формування статистики та аналітики харчування

Для забезпечення ефективного контролю харчування та підтримки користувача у процесі досягнення поставлених цілей реалізовано підсистему статистики та аналітики харчування. Основним призначенням цієї підсистеми є візуалізація поточного прогресу користувача, аналіз спожитих нутрієнтів, моніторинг водного балансу та відображення динаміки зміни фізіологічних показників.

Підсистема аналітики побудована відповідно до архітектурного патерну MVVM, що забезпечує відокремлення логіки обробки даних від компонентів інтерфейсу користувача. Центральним компонентом механізму виступає модель

представлення ProgressViewModel, яка відповідає за завантаження, агрегацію та підготовку статистичних даних для подальшого відображення у графічному інтерфейсі застосунку.

Однією з основних функцій підсистеми є відображення прогресу досягнення вагової цілі користувача. Для цього система зберігає початкову вагу, поточне значення та цільову вагу, після чого автоматично обчислює відсоток виконання поставленої цілі.

Використання автоматичного перерахунку властивостей дозволяє забезпечити динамічне оновлення інтерфейсу після зміни вагових показників без необхідності ручного перезавантаження сторінки. Такий підхід підвищує інтерактивність системи та забезпечує користувача актуальною інформацією в режимі реального часу.

Механізм візуального контролю водного балансу реалізовано через набір інтерактивних елементів, які відображають умовні порції води. Кожен елемент відповідає певному об’єму рідини, а його стан змінюється після взаємодії користувача з інтерфейсом (рис. 3.9).

Рисунок 3.9 – Відображення стану виконання добової норми води

При зміні стану елемента система автоматично створює або видаляє запис WaterLog у базі даних, після чого оновлює показники прогресу виконання добової норми води (рис. 3.10).

Таким чином, реалізований механізм обліку харчування та водного балансу забезпечує централізоване накопичення даних про щоденне харчування користувача, автоматичний розрахунок спожитих нутрієнтів і візуальний контроль виконання персональних рекомендацій.

Рисунок 3.10 – Графік статистики споживання води

Використання сервісно-орієнтованого підходу, локального збереження даних та реактивного оновлення інтерфейсу дозволило забезпечити модульність системи, стабільність роботи та зручність користувацької взаємодії.

3.3 Результати роботи мобільного помічника

У результаті розроблено мобільний помічник для персонального контролю харчування EatSmart Balance, який забезпечує повний цикл роботи з користувачем – від первинного налаштування профілю до щоденного моніторингу харчування та аналізу досягнутих результатів.

Система реалізує повний цикл обробки персональних даних: їх збір під час онбордингу, обробку на рівні бізнес-логіки, збереження у локальній базі даних та подальше використання для формування персоналізованих рекомендацій. Це дозволяє адаптувати функціональність застосунку до фізіологічних параметрів, рівня активності та цілей користувача.

Ключовим результатом є реалізація механізму розрахунку добових норм харчування, який визначає енергетичні потреби організму та відповідні значення макро- і мікронутрієнтів. На його основі формуються персоналізовані плани харчування з урахуванням індивідуальних показників та харчових обмежень.

Реалізовано також підсистему обліку харчування та водного балансу, яка дозволяє фіксувати спожиті продукти, страви та обсяг води з подальшою

автоматичною агрегацією даних і оновленням статистики виконання добових норм.

Підсистема аналітики забезпечує візуалізацію харчових звичок користувача, прогресу досягнення цілей та динаміки фізіологічних показників, що дозволяє оцінювати ефективність обраного режиму харчування.

З технічної точки зору застосунок побудовано на основі архітектурного патерну MVVM, що забезпечує розділення відповідальностей між рівнями системи та підвищує її модульність і підтримуваність. Використання SQLite у поєднанні з Entity Framework Core забезпечує автономність роботи застосунку та ефективне управління локальними даними.

Висновок до третього розділу

У третьому розділі висвітлено процес практичної реалізації мобільного помічника EatSmart Balance та його основних функціональних підсистем.

Розроблено фірмовий стиль та інтерфейс користувача у мінімалістичному стилі з використанням фірмової кольорової палітри, що забезпечує інтуїтивну взаємодію та відповідає сучасним стандартам UX/UI-дизайну.

Реалізовано підсистему локального збереження даних на основі SQLite та Entity Framework Core із застосуванням механізмів міграцій, Fluent API та Dependency Injection, що забезпечує автономність роботи та цілісність даних.

Реалізовано підсистему розрахунку добових норм та генерації персоналізованих планів харчування з урахуванням харчових обмежень, а також механізм щоденного обліку харчування та водного балансу з автоматичною агрегацією показників.

Розроблено підсистему аналітики та статистики, що забезпечує візуалізацію прогресу досягнення цілей та динаміки фізіологічних показників користувача.

Отримані результати підтверджують відповідність реалізованого застосунку визначеним функціональним і нефункціональним вимогам та засвідчують досягнення мети роботи.

ВИСНОВКИ

У межах дипломної роботи було розроблено мобільного помічника для персонального контролю харчування, що дозволяє користувачам ефективно аналізувати власний раціон, формувати здорові харчові звички та досягати індивідуальних фізіологічних цілей.

У процесі виконання роботи:

- проаналізовано предметну область та сучасні підходи до контролю харчування, а також визначено недоліки існуючих програмних рішень;
- сформовано функціональні вимоги та побудовано інформаційну модель системи;
- спроєктовано структуру реляційної бази даних для збереження даних про користувача, його цілі, обмеження, продукти, страви та історію харчування;
- реалізовано архітектуру мобільного застосунку з використанням кросплатформного підходу (.NET MAUI) та патерну MVVM, що забезпечує модульність і зручність супроводу;
- розроблено сценарії взаємодії користувача із системою, включаючи процес первинного налаштування;
- створено математичну модель формування персоналізованих рекомендацій на основі розрахунку BMI, BMR та TDEE з урахуванням індивідуальних параметрів користувача;
- реалізовано механізм розрахунку добової калорійності, балансу макро та мікронутрієнтів та інших показників харчування.

Таким чином, виконана робота не лише вирішує поставлену практичну задачу, але й демонструє застосування сучасних підходів до розробки мобільних інформаційних систем у сфері здорового харчування та має перспективи подальшого розвитку і використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Римар П.В., Ліваковський В. К. Розробка платформи для управління харчуванням з інтеграцією штучного інтелекту. Вісник Східноукраїнського національного університету імені Володимира Даля. 2025. № 4 (290). С. 21–26. URL: https://doi.org/10.33216/1998-7927-2025-290-4-21-26.
2. Bevziuk A., Antonov Y.S., Lozynska L.F. Activity-based modeling of a personalized meal recommendation system in a mobile assistant. X International Scientific Conference for Bachelor, Master, Graduate Students, and Young Researchers «Topical issues of humanities, technical and natural sciences», April 30, 2026. Vinnytsia: Vasyl' Stus Donetsk National University, 2026.
3. Бевзюк А. Ю., Антонов Ю. С. Проєктування бази даних системи персонального контролю харчування. Прикладні інформаційні технології: Збірник тез доповідей, м. Вінниця, 15 травня 2026 р. Вінниця: ДонНУ імені Василя Стуса, 2026.
4. Зубар Н. Основи фізіології та гігієни харчування. Київ : «Кондор», 2018. 444 с.
5. Про затвердження Норм фізіологічних потреб населення України в основних харчових речовинах і енергії. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/z1206-17#Text (дата звернення: 01.04.2026).
6. Gropper S. S., Smith J. L., Carr T. P. Advanced nutrition and human metabolism. 8th ed. Cengage Learning, 2021. 608 p.
7. Sizer F., Whitney E. Nutrition: Concepts and Controversies. 16th ed. Cengage Learning, 2022. 880 p.
8. Mann J., Truswell S., Hodson L. Essentials of human nutrition. 6th ed. Oxford : Oxford University Press, 2023. 744 p.
9. Фактори ризику неінфекційних захворювань в Україні у 2019 році. Центр громадського здоров’я України, МОЗ. URL: https://phc.org.ua/sites/default/files/users/user90/2019_STEPS_summary_ukr.

pdf (дата звернення: 03.04.2026).

10. Резнік Р. Ю., Антонов Ю. С. Розробка додатків під платформи ANDROID та IOS. Прикладні інформаційні технології : Зб. тез доп. - 2020 р., м. Вінниця. Вінниця, 2020. С. 132–135. URL: https://jait.donnu.edu.ua/article/view/8940/0 (дата звернення: 30.05.2026).
11. Сидорчук Р. В., Потапова Н. А. Сучасні технології розробки мобільних застосунків. Матеріали VII Всеукр. науково-практ. конф. здобувачів вищ. освіти та молодих вчен. «Прикладні інформаційні технології», м. Вінниця, 2024.
12. Mobile App Development with Ionic, Revised Edition: Cross-Platform Apps with Ionic, Angular, and Cordova. O'Reilly Media, 2017. 290 p.
13. Swift Apple Developer Documentation. Apple Developer Documentation. URL: https://developer.apple.com/documentation/Swift (date of access: 06.04.2026).
14. Android Programming: The Big Nerd Ranch Guide / C. Stewart et al. 5th ed. Addison-Wesley Professional, 2022. 688 p.
15. .NET MAUI documentation. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-10.0 (date of access: 06.04.2026).
16. Flutter documentation. Google. URL: https://docs.flutter.dev/ (date of access: 06.04.2026).
17. Guide to app architecture. Android Developers. URL: https://developer.android.com/topic/architecture (date of access: 27.03.2026).
18. Ibraimi A., Memeti A., Idrizi F. MVC VS MVVM VS MVP: Which architectural pattern suits modern applications best? JNSM - Journal of Natural Sciences and Mathematics of UT. 2025. Vol. 10, no. 19-20. P. 306–311. URL: https://doi.org/10.62792/ut.jnsm.v10.i19-20.p3196 (date of access: 07.04.2026).
19. Overview of ASP.NET Core MVC. Microsoft Learn. URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-10.0 (date of access: 07.04.2026).

20. Späth P. About MVC: Model, View, Controller. Beginning Java MVC 1.0. Berkeley, CA, 2020. P. 1–18. URL: https://doi.org/10.1007/978-1-4842-6280-1_1 (date of access: 08.04.2026).
21. García R. F. MVP: Model–View–Presenter. iOS Architecture Patterns. Berkeley, CA, 2023. P. 107–144. URL: https://doi.org/10.1007/978-1-4842-9069-9_3 (date of access: 06.04.2026).
22. Stonis M. Model-View-ViewModel (MVVM). Enterprise Application Patterns Using .NET MAUI. Redmond, 2022. P. 9–21. URL: https://dotnet.microsoft.com/en-us/download/e-book/maui/pdf (date of access: 15.04.2026).
23. ViewModel overview. Android Developers. URL: https://developer.android.com/topic/libraries/architecture/viewmodel (date of access: 18.05.2026).
24. FatSecret. Free Calorie Counter. URL: https://www.fatsecret.com (date of access: 07.04.2026).
25. Lifesum App: Nutrition & Meal Planner. URL: https://www.lifesum.com (date of access: 07.04.2026).
26. Inc. MyFitnessPal: Calorie Counter. Android Apps on Google Play. URL: https://play.google.com/store/apps/details?id=com.myfitnesspal.android&hl=uk (date of access: 24.03.2026).
27. Hernandez M. J. Database Design for Mere Mortals: 25th Anniversary Edition. Pearson Education, Limited, 2020.
28. Павловський В. І., Петрашенко А. В. Бази даних та засоби управління: підручник. Київ : КПІ ім. Ігоря Сікорського, 2024. 293 с. URL: https://scs.kpi.ua/wpcontent/uploads/2025/09/bazy_danyh_ta_zasoby_upravlinnya.pdf (дата звернення: 01.06.2026).
29. SQLite Documentation. URL: https://www.sqlite.org/docs.html (date of access: 27.04.2026).
30. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, Incorporated,

2017.

31. Lawrence S. Introducing .NET MAUI. Berkeley, CA : Apress, 2025. URL: https://doi.org/10.1007/979-8-8688-1189-0 (date of access: 27.04.2026).
32. Чернишенко Я. А., Зелінська О. В. Застосування шаблону MVVM у реалізації UI-логіки десктопних додатків. Прикладні інформаційні технології : Збірник тез доповідей - 2025 р., м. Вінниця, 22 трав. 2025 р. С. 79–81. URL: https://jait.donnu.edu.ua/article/view/19308 (дата звернення: 01.06.2026).
33. Smith J. P. Entity Framework Core in Action, Second Edition. Manning Publications Co. LLC, 2021. 624 p.
34. Stonis M. Enterprise Application Patterns Using .NET MAUI. Redmond, 2022. 106 p. URL: https://learn.microsoft.com/en-us/dotnet/architecture/maui/ (date of access: 30.04.2026).
35. .NET MAUI Shell overview - .NET MAUI. Microsoft Learn. URL: https://learn.microsoft.com/uk-ua/dotnet/maui/fundamentals/shell/?view=net-maui-10.0 (date of access: 30.04.2026).
36. Modern nutrition in health and disease / A. Catharine Ross et al. 11th ed. Philadelphia : Lippincott Williams & Wilkins, 2020. 1616 p.
37. Morrow K., Raymond J. L. Krause and Mahan's Food and the Nutrition Care Process. 16th ed. Elsevier - Health Sciences Division, 2022.
38. Whitney E., Rolfes S. R. Understanding Nutrition. 16th ed. Cengage Learning, 2021. 816 p.
39. Weichbroth P. Usability Testing of Mobile Applications: A Methodological Framework. Applied Sciences. 2024. Vol. 14, no. 5. P. 1792. URL: https://doi.org/10.3390/app14051792 (date of access: 01.06.2026).
40. Lu G., Qu S., Chen Y. Understanding user experience for mobile applications: a systematic literature review. Discover Applied Sciences. 2025. Vol. 7, no. 6. URL: https://doi.org/10.1007/s42452-025-07170-3 (date of access: 02.06.2026).
41. Silva L. F. d., Junior P. A. P., Freire A. P. Mobile User Interaction Design Patterns: A Systematic Mapping Study. Information. 2022. Vol. 13, no. 5. P.

236. URL: https://doi.org/10.3390/info13050236 (date of access: 02.06.2026).
42. Feiler J. Introducing SQLite for Mobile Developers. Apress, 2015. 168 p.
43. Pothineni S. H. Offline-First Mobile Architecture: Enhancing Usability and Resilience in Mobile Systems. Journal of Artificial Intelligence General science (JAIGS) ISSN:3006-4023. 2024. Vol. 7, no. 01. P. 320–326. URL: https://doi.org/10.60087/jaigs.v7i01.387 (date of access: 02.06.2026).
44. Hule K., Ranawat R. Analysis of Different ORM Tools for Data Access Object Tier Generation: A Brief Study. International Journal of Membrane Science and Technology. 2023. Vol. 10, no. 1. P. 1277–1291. URL: https://doi.org/10.15379/ijmst.v10i1.2842 (date of access: 02.06.2026).
45. Miller R., Lerman J. Programming Entity Framework: Dbcontext. O'Reilly Media, Incorporated, 2012. 256 p.
46. Creating and Configuring a Model - EF Core. Microsoft Learn. URL: https://learn.microsoft.com/en-us/ef/core/modeling/ (date of access: 02.06.2026).
47. Entity Framework Core Migrations. Learn Entity Framework Core. URL: https://www.learnentityframeworkcore.com/migrations (date of access: 02.06.2026).
48. Gorman B. Practical Entity Framework Core 10. Berkeley, CA : Apress, 2025. URL: https://doi.org/10.1007/979-8-8688-2123-3 (date of access: 02.06.2026).
49. Brdnik S., Heričko T., Šumak B. Intelligent User Interfaces and Their Evaluation: A Systematic Mapping Study. Sensors. 2022. Vol. 22, no. 15. P. 5830. URL: https://doi.org/10.3390/s22155830 (date of access: 02.06.2026).
50. Ye R., .NET MAUI Cross-Platform Application Development: Leverage a First-Class Cross-platform UI Framework to Build Native Apps on Multiple Platforms. Packt Publishing, Limited, 2023. 400 p.

ДОДАТОК А

Схема взаємозв’язків таблиць бази даних