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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-додатку трекера спожитих калорій із застосуванням технологій ASP.NET та React

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Одінцов К.І.
(підпис) (прізвище та ініціали)

Керівник _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

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

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2026

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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«___» _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Одінцов Костянтину Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-додатку трекера спожитих калорій із застосуванням технологій ASP.NET та React

Керівник роботи доктор технічних наук, професор кафедри КН Литвиненко Ярослав Володимирович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*14*» *травня* 20*26* року № *4/9-239*

2. Термін подання студентом завершеної роботи червня 2026 р.

3. Вихідні дані до роботи Літературні джерела які стосуються робзробки веб-додатку та використаних технологій ASP.NET та React

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області та обґрунтування вибору технологій. 1.1. Обґрунтування актуальності розробки. 1.2. Аналіз існуючих аналогів. 1.3. Визначення цільової аудиторії проєкту. 1.4. Функціональні вимоги до системи. 1.5. Проєктування архітектури варіантів використання системи. 1.6. Обґрунтування вибору технологій. 1.7. Розміщення вебдодатка в хмарному середовищі. 1.8. Висновок до розділу 1. Розділ 2 Проєктування та програмна реалізація веб-додатку трекера спожитих калорій. 2.1. Архітектурна побудова серверної частини за принципами Чистої архітектури. 2.2. Проєктування та розробка реляційної моделі даних у PostgreSQL. 2.3. Програмна реалізація бізнес-логіки із застосуванням патерну CQRS. 2.4. Програмна реалізація підсистеми безпеки та автентифікації. 2.5. Програмна реалізація клієнтської частини додатка на базі бібліотеки React. 2.6. Контейнеризація сервісів та налаштування конфігурації Docker. 2.7. Тестування працездатності компонентів системи. 2.8. Висновок до розділу 2. Розділ 3. Практична демонстрація веб-додатку трекера калорій . 3.1. Тестування процесу реєстрації та авторизації. 3.2. Функціонал сторінки «Журнал калорій» та додавання продукту. 3.3. Функціонал сторінки «Аналітика». 3.4. Функціонал сторінки «Профіль». 3.5. Мобільна версія. ПК. користувацького інтерфейсу. 3.6. Висновок до розділу 3. Розділ 4. Безпека життєдіяльності та охорона праці. 4.1 Долікарська допомога при харчових отруєннях. 4.2. Вимоги ергономіки до організації робочого місця оператора 4.3. Висновок до розділу 4. Висновки. Перелік джерел. Додатки

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання 07 травня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2026	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	08.05.2026-15.05.2026	*Виконано*
3.	Виконання дослідження щодо предметної області	08.05.2026-15.05.2026	*Виконано*
	Розроблення плану виконання та етапів створення	08.05.2026-15.05.2026	
	веб-додатку		
4.	Оформлення розділу «Аналіз предметної області та	10.05.2026–26.05.2026	*Виконано*
	обґрунтування вибору технологій»		
5.	Оформлення розділу «Проєктування та програмна	10.05.2026–26.05.2026	*Виконано*
	реалізація веб-додатку трекера спожитих калорій»		
6.	Оформлення розділу «Практична демонстрація веб-	10.05.2026–26.05.2026	*Виконано*
	додатку трекера калорій»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	10.05.2026–26.05.2026	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	10.05.2026–26.05.2026	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	27.05.2026–14.06.2026	*Виконано*
10.	Нормоконтроль		*Виконано*
11.	Перевірка на плагіат		*Виконано*
12.	Попередній захист кваліфікаційної роботи		*Виконано*
13.	Захист кваліфікаційної роботи		

Студент _____ Одінцов К.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-додатку трекера спожитих калорій із застосуванням технологій ASP.NET та React // Кваліфікаційна робота освітнього рівня «Бакалавр» // Одіцнов Костянтин Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2026 // С. 75 , рис. – 16 , табл. 2, кресл. – , додат. – 5, бібліогр. – .

Ключові слова: веб-додаток, трекер калорій, .NET, React, Postgresql, чиста архітектура, Docker, Azure.

Кваліфікаційна робота присвячена створенню веб-додатку трекера спожитих калорій для зручного відслідковування балансу спожитих калорій та нутрієнтів, для підтримання стану здоров'я.

У першому розділі кваліфікаційної роботи описано предметну область, проаналізовано конкурентне середовище, визначено цільову аудиторію та ключові вимоги до системи. Висвітлено підходи до вибору технологій, платформ та хостингу.

У другому розділі досліджено архітектуру системи, змодельовано базу даних, подано структуру клієнтської та серверної частини, описано користувацький інтерфейс та розміщення проекту на хмарному сервері.

У третьому розділі описано процес тестування вебсайту, перевірку його функціональності, а також ефективність обраних рішень.

Об'єкт дослідження: процес розробки вебдодатку для відслідковування спожитих калорій та балансу КБЖВ

ANNOTATION

Development of a Calorie Intake Tracker Web Application Using ASP.NET and React Technologies // Qualification work of the educational level «Bachelor» // Odintsov Kostiantyn // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-42 // Ternopil, 2025 // P.75, fig. –16, tabl. – 2, chair. – , annexes. – 5 , references – .

Keywords: web application, calorie tracker, .NET, React, Postgresql, clean architecture, Docker, Azure.

The qualification thesis is dedicated to the development of a web application for tracking nutrients and body metrics to ensure convenient monitoring of calorie balance and support a healthy lifestyle.

The first chapter of the qualification thesis investigates the subject domain, analyzes the competitive environment, defines the target audience, and establishes key system requirements. Methodological approaches to the selection of development tools, software platforms, and hosting infrastructure are substantiated.

The second chapter covers the architectural design of the backend and frontend sides, relational database modeling, a detailed description of the user interface, norm calculation algorithms, and the results of automated component testing.

The third chapter highlights the processes of software product containerization and the stages of its practical deployment in a cloud environment with continuous delivery configuration.

The object of research is the process of development and cloud deployment of a web application for tracking dietary caloric intake and macronutrient balance.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

.NET — кросплатформове програмне середовище від Microsoft для реалізації серверної частини системи.

API (англ. Application Programming Interface) — програмний інтерфейс для взаємодії між клієнтською та серверною частинами.

ASP.NET — вебфреймворк платформи .NET для створення серверного веб-API.

Azure — хмарна платформа Microsoft для розміщення та супроводу компонентів додатка.

CQRS (англ. Command Query Responsibility Segregation) — патерн розділення операцій запису та читання даних.

CSS (англ. Cascading Style Sheets) — мова стилізації для оформлення користувацького інтерфейсу вебсторінок.

Docker — платформа контейнеризації для ізольованого запуску сервісів додатка.

DTO (англ. Data Transfer Object) — об’єкт передачі даних між шарами застосунку та через API.

EF Core (англ. Entity Framework Core) — ORM-інструмент .NET для взаємодії з реляційною базою даних через моделі C#.

FluentValidation — бібліотека декларативної валідації вхідних команд і запитів у прикладному шарі.

HTTP (англ. HyperText Transfer Protocol) — мережевий протокол обміну запитами та відповідями між клієнтом і сервером.

JWT (англ. JSON Web Token) — формат токена доступу для безстанової автентифікації користувачів.

MediatR — бібліотека реалізації патерну посередника для CQRS-взаємодії команд і запитів.

Nginx — вебсервер, що використовується для роздачі production-збірки клієнтської SPA-частини.

OAuth 2.0 — протокол делегованої авторизації, що використовується для входу через Google.

OpenAPI — специфікація опису REST API для автоматичної генерації документації та тестування ендпоінтів.

PostgreSQL — реляційна система керування базами даних, що використовується як основне сховище даних проєкту.

React — JavaScript-бібліотека для побудови компонентного SPA-інтерфейсу користувача.

Recharts — бібліотека React-компонентів для візуалізації графіків і аналітичних даних.

Scalar UI — вебінтерфейс для перегляду та перевірки API на основі OpenAPI-специфікації.

SPA (англ. Single Page Application) — формат вебзастосунку з динамічною навігацією без повного перезавантаження сторінок.

Vite — інструмент збірки та запуску фронтенд-проєкту з швидким development-циклом.

Vitest — тестовий фреймворк для автоматизованої перевірки клієнтської частини на базі Vite.

КБЖВ — скорочення для показників калорій, білків, жирів і вуглеводів у раціоні користувача.

ЗМІСТ

ВСТУП 9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ 11
1.1 Обґрунтування актуальності розробки 11
1.2. Аналіз існуючих аналогів 12
1.3 Визначення цільової аудиторії проєкту 14
1.4. Функціональні вимоги до системи 15
1.5. Проєктування архітектури варіантів використання системи 17
1.6. Обґрунтування вибору технологій 19
1.7. Розміщення вебдодатка в хмарному середовищі 20
1.8 Висновок до розділу 1 21
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ТРЕКЕРА СПОЖИТИХ КАЛОРІЙ 23
2.1. Архітектурна побудова серверної частини за принципами Чистої архітектури 23
2.2. Проєктування та розробка реляційної моделі бази даних у PostgreSQL 24
2.3. Програмна реалізація бізнес-логіки із застосуванням патерну CQRS 26
2.4. Програмна реалізація підсистеми безпеки та автентифікації 28
2.5. Програмна реалізація клієнтської частини додатка на базі бібліотеки React 30
2.6 Контейнеризація сервісів та налаштування конфігурації Docker 31
2.7. Тестування працездатності компонентів системи 34
2.8 Висновок до розділу 2 36
РОЗДІЛ 3. ПРАКТИЧНА ДЕМОНСТРАЦІЯ ВЕБ-ДОДАТКУ ТРЕКЕРА КАЛОРІЙ 38
3.1 Тестування процесу реєстрації та авторизації 38
3.2. Функціонал сторінки «Журнал калорій» та додавання продукту 41
3.3 Функціонал сторінки «Аналітика» 42
3.4 Функціонал сторінки «Профіль» 44
3.5. Мобільна версія користувацького інтерфейсу 46
3.6 Висновок до розділу 3 48

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА 50
ОХОРОНА ПРАЦІ 50
4.1. Долікарська допомога при харчових отруєннях 50
4.2 Вимоги ергономіки до організації робочого місця оператора ПК 52
ВИСНОВКИ 56
ПЕРЕЛІК ДЖЕРЕЛ 58
ДОДАТКИ

ВСТУП

Актуальність теми. У сучасному світі питання раціонального харчування та контролю стану здоров’я набувають дедалі більшого значення через зростання темпу життя та поширення захворювань, пов'язаних із неправильним способом життя. Існуючі рішення часто є перевантаженими або складними для кінцевого користувача, що зумовлює потребу у розробці інтуїтивно зрозумілих та технологічно досконалих систем для моніторингу харчової поведінки. Проєкт Nutrio спрямований на вирішення цієї проблеми шляхом створення вебдодатка, що дозволяє користувачам не лише вести щоденник калорій, а й отримувати інтелектуальну підтримку через впровадження сучасних архітектурних рішень. Використання підходу чистої архітектури та патерну CQRS забезпечує високу масштабованість та надійність системи, що робить розробку актуальною у контексті створення сучасних хмарних сервісів.

Мета і задачі дослідження. Метою роботи є проєктування та розробка вебдодатка «Nutrio» для відстеження калрій, нутрієнтів та антропометричних показників користувача.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- Проаналізувати предметну область та існуючі аналоги в сфері цифрового контролю харчування;
- Реалізувати вебплатфому за допомогою сучасних технологій вебпрограмування а саме: .NET для серверної частини, React для клієнтської та PostgreSQL як системи управління базами даних;
- Спроектувати архітектуру системи на основі принципів розділення відповідальності та незалежності шарів;
- Розробити базу даних, що забезпечує зберігання інформації про продукти, прийоми їжі та динаміку метрик тіла;
- Реалізувати серверне API з використанням MediatR для розділення команд та запитів;

- Створити адаптивний інтерфейс користувача з візуалізацією прогресу за допомогою графіків та аналітичних панелей;
- Налаштувати процеси контейнеризації через Docker та підготувати систему до розгортання в хмарному середовищі Azure;

Практичне значення одержаних результатів. Результатом роботи є повністю функціональний вебдодаток «Nutrio», готовий до промислової експлуатації. Програмний продукт надає користувачам можливість автоматизованого розрахунку денних норм калорій залежно від мети (схуднення, підтримка, набір маси) та рівня активності. Практична цінність полягає в реалізації гнучкого механізму ведення журналу харчування з деталізацією за КБЖВ та клітковиною, а також у наявності системи аналітики, що дозволяє відстежувати ефективність досягнення цілей у довгостроковій перспективі. Технічна база проєкту дозволяє легко інтегрувати в майбутньому додаткові сервіси на основі штучного інтелекту для надання персоналізованих рекомендацій користувачам.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

1.1 Обґрунтування актуальності розробки

У сучасних умовах розвитку суспільства питання раціонального харчування та ведення здорового способу життя набувають особливої ваги. Згідно з даними Всесвітньої організації охорони здоров'я [1], значна частина неінфекційних захворювань, таких як цукровий діабет, серцево-судинні патолії та ожиріння, безпосередньо пов'язана з порушеннями енергетичного балансу та неконтрольованим споживанням нутрієнтів. Це зумовлює високий попит на ефективні інструменти самоконтролю, які дозволяють користувачам оперативно відстежувати свій раціон та антропометричні показники. Метою розробки веб-додатка «Nutrio» є створення комплексної інтелектуальної системи для моніторингу харчової активності та прогресу досягнення цілей щодо маси тіла.

Актуальність даної розробки зумовлена низкою вагомих факторів, серед яких першочергове місце посідає стрімка цифровізація сфери здоров’я та фітнесу. Сучасне суспільство потребує доступних сервісів, які дозволяють автоматизувати трудомісткий процес підрахунку калорій, білків, жирів та вуглеводів. При цьому аналіз ринку аналогічних систем свідчить про те, що більшість існуючих продуктів мають або надмірно складний інтерфейс, або пропонують критично важливі функції аналітики виключно на платній основі. Проєкт «Nutrio» покликаний розв'язати ці проблеми шляхом надання користувачам простого у використанні інструменту з гнучкою персоналізацією. Система забезпечує збір детальних даних під час первинного налаштування для автоматичного розрахунку індивідуальної норми енергоспоживання, що робить підхід до кожного користувача унікальним.

З технічної точки зору актуальність проєкту підтверджується використанням передових підходів до побудови програмного забезпечення. Реалізація системи на засадах Чистої архітектури [2] та патерну CQRS [3] із

застосуванням бібліотеки MediatR дозволяє створити масштабований та відмовостійкий продукт. Таке архітектурне рішення забезпечує стабільну роботу системи під значним навантаженням та її легку інтеграцію з хмарними середовищами, зокрема з платформою Azure. Таким чином, розробка «Nutrio» є актуальною відповіддю на сучасні виклики усфері цифрової підтримки здоров'я, поєднуючи в собі інженерну якість та орієнтованість на потреби кінцевого споживача.

1.2. Аналіз існуючих аналогів

На етапі проєктування інформаційної системи «Nutrio» критично важливим є проведення ґрунтовного аналізу існуючих програмних рішень, що представлені на ринку цифрових інструментів для контролю харчування. Найбільш поширеним та затребуваним аналогом серед користувачів в Україні є сервіс Таблиця калорійності, який пропонує широкі можливості для ведення харчових щоденників та моніторингу активності. Основною перевагою даного продукту є наявність масштабної бази даних продуктів місцевого виробництва, що значно спрощує процес внесення інформації для вітчизняного споживача. Програма дозволяє автоматично розраховувати споживання енергії та основних нутрієнтів, а також підтримує синхронізацію даних між клієнтською частиною вебдодатка та мобільними пристроями.

Попри значну популярність, детальний аналіз практичної реалізації користувацького інтерфейсу базового аналога дозволяє виявити суттєві недоліки, притаманні «Таблиці калорійності» [4] та іншим подібним фітнес-застосункам. Як показано на рисунку 1.1, головний робочий простір сервісу є надмірно перевантаженим великою кількістю рекламних банерів, сторонніми комерційними блоками та надлишковими інформаційними елементами, що значно ускладнює загальну навігацію, розсіює увагу та суттєво відволікає користувача від основної мети використання системи [5].

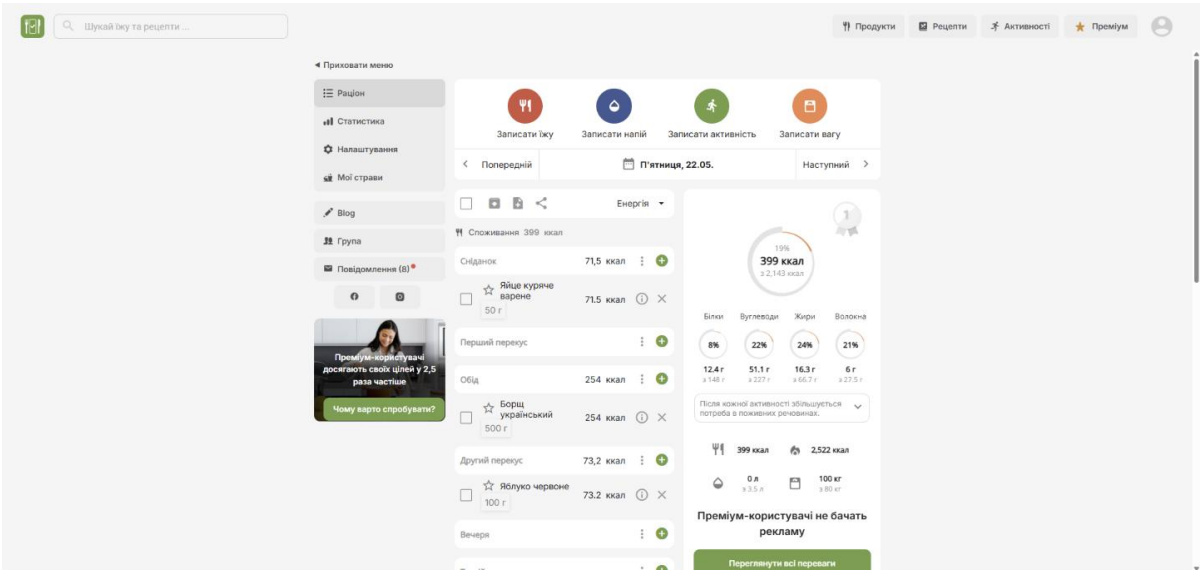

Рисунок 1.1 - Користувацький інтерфейс головного вікна сервісу «Таблиця калорійності»

Візуальне представлення щоденного підсумку споживання калорій, білків, жирів та вуглеводів у аналізованому сервісі виконано у вигляді дрібних текстових блоків, які змішуються із кнопками швидкого доступу та преміальними пропозиціями, створюючи додаткове когнітивне навантаження. Багато критично важливих функцій, таких як розширена аналітика, детальні графічні звіти про довгострокову динаміку антропометричних показників та глибокий аналіз раціону, заблоковані комерційними обмеженнями та доступні користувачам виключно на платній основі. Крім того, складність первинного налаштування та велика кількість надлишкових функцій можуть створювати бар'єри для людей, які шукають лаконічний та швидкий інструмент для щоденного моніторингу.

Проєкт «Nutrio» покликаний розв'язати виявлені проблеми шляхом надання користувачам інтуїтивно зрозумілого інтерфейсу, повністю зосередженого на ключових функціях контролю. На відміну від існуючих аналогів, розроблювана система пропонує збалансований підхід до візуалізації даних, де основний акцент зроблено на лаконічному відстеженні історії змін тіла та чіткому дотриманні норм білків, жирів та вуглеводів без будь-якого зайвого візуального чи рекламного навантаження. Використання передових

архітектурних підходів, таких як Чиста архітектура та патерн CQRS, забезпечує високу швидкість роботи системи та її стабільність. Це дозволяє ефективно конкурувати з відомими рішеннями за рахунок вищої якості користувацького досвіду, повної відсутності нав'язливої реклами та доступності розширених аналітичних інструментів для всіх зареєстрованих користувачів. Таким чином, розробка власної системи є обґрунтованою відповіддю на потребу ринку у функціонально потужному, але простому у використанні засобі персонального контролю здоров'я.

1.3 Визначення цільової аудиторії проєкту

Ефективна розробка веб-додатку «Nutrio» потребує чіткого розуміння потреб потенційних користувачів. З цією метою було застосовано метод моделювання архетипів цільової аудиторії [5], що дозволяє трансформувати запити споживачів у конкретні функціональні вимоги до системи. У результаті дослідження було виділено три ключові архетипи користувачів, які відображають основні сегменти ринку цифрових засобів контролю раціону.

Першим архетипом є офісний працівник, головною метою якого є зниження маси тіла та подолання наслідків малорухливого способу життя. Через високу щоденну зайнятість такий користувач потребує інструменту для максимально швидкої фіксації спожитих продуктів. Головним бар'єром в існуючих аналогах для цього сегмента є надмірне перевантаження інтерфейсу комерційною рекламою, що розсіює увагу. Для задоволення його потреб у «Nutrio» реалізовано лаконічний журнал харчування, автоматизований розрахунок індивідуальної норми енергоспоживання під час первинного налаштування та функцію фіксації антропометричних змін.

Другим архетипом є спортсмен-аматор, який веде активний спосіб життя і прагне чітко контролювати добовий баланс макронутрієнтів та клітковини для досягнення конкретних фізичних результатів, таких як набір м'язової маси або підвищення витривалості. Основна проблема цієї категорії користувачів при

роботі з існуючими сервісами — обмеженість безкоштовних інструментів аналітики. Для цього архетипу критично важливою є наявність розвиненої підсистеми візуалізації, яка надає детальні довгострокові звіти про споживання нутрієнтів та динаміку ваги.

Третім архетипом є людина з проблемами зі здоров'ям або схильністю до хронічних захворювань, безпосередньо пов'язаних із порушеннями енергетичного балансу та неконтрольованим споживанням нутрієнтів. Головною мотивацією такого користувача є систематичний моніторинг якості раціону та збалансованості нутрієнтів з метою підтримки стабільного стану організму та профілактики ускладнень. Для цього сегмента першочергове значення має наявність зручного механізму часових міток для відстеження історії антропометричних змін у поєднанні з відсутністю зайвого інформаційного тиску.

Синтез вимог усіх трьох виділених архетипів обґрунтовує доцільність реалізації проєкту саме у форматі вебдодатка, що забезпечує високу кросплатформову доступність системи з будь-якого сучасного пристрою без необхідності встановлення стороннього програмного забезпечення. Повна безкоштовність усього розширеного аналітичного інструментарію повністю усуває комерційні бар'єри для користувачів, а максимальна простота та лаконічність інтерфейсу дозволяють мінімізувати когнітивне навантаження, забезпечуючи швидкий, інтуїтивно зрозумілий та комфортний щоденний самоконтроль для кожного учасника системи.

1.4. Функціональні вимоги до системи

Для забезпечення повноцінної роботи вебдодатка «Nutrio» та задоволення потреб цільової аудиторії визначено комплекс функціональних вимог, що охоплюють усі етапи взаємодії користувача з інформаційною системою. Першочерговою вимогою є реалізація захищеної підсистеми реєстрації та автентифікації, яка дозволяє створювати персональні облікові записи. Система

має підтримувати як традиційний вхід за допомогою електронної пошти та пароля, так і спрощений механізм авторизації через інтеграцію зі сторонніми сервісами єдиного входу. Після успішної авторизації програма повинна забезпечувати процес первинного налаштування, під час якого здійснюється збір ключових персональних даних: статі, віку, зросту, а також поточної та цільової маси тіла користувача. Важливою функцією на цьому етапі є автоматизований розрахунок індивідуальної денної норми калорій та балансу нутрієнтів [6] залежно від обраної мети та рівня фізичної активності, що дозволяє персоналізувати підхід до кожного споживача .

Центральним елементом системи є журнал харчування, який надає користувачу можливість вести детальний щоденний облік спожитих продуктів із логічним розподілом за прийомами їжі, такими як сніданок, обід, вечеря та перекуси. Програма має забезпечувати зручний пошук харчових продуктів у базі даних та автоматично обчислювати вміст білків, жирів, вуглеводів та клітковини для кожної обраної порції. Окрім безпосереднього контролю раціону, система повинна підтримувати функцію фіксації антропометричних змін за допомогою механізму часових міток метрик тіла, що дозволяє надійно зберігати повну історію вимірювань і проводити порівняльний аналіз стану організму в різні періоди часу.

Важливою вимогою до системи є наявність розвиненої підсистеми аналітики, яка візуалізує накопичені дані для підвищення мотивації користувача та наочності його прогресу. Програма має формувати інтерактивні графіки динаміки маси тіла, відображати добовий дефіцит або профіцит енергії, а також надавати статистику щодо дотримання норм макронутрієнтів. Додатково система повинна містити інструменти моніторингу активності ведення журналу у вигляді теплових карт, що дозволяє об'єктивно оцінювати регулярність самоконтролю. Усі перелічені функції повинні реалізовуватися на базі стабільної серверної частини з використанням Чистої архітектури та патерну CQRS, що гарантує цілісність даних та високу швидкість обробки запитів навіть за умови значного масштабування інформаційної системи.

1.5. Проєктування архітектури варіантів використання системи

На етапі проєктування веб-додатку системи «Nutrio» визначено архітектуру варіантів використання, що деталізує всі можливі сценарії взаємодії користувачів із вебдодатком. Графічне представлення цієї структури наведено на рисунку 1.2.

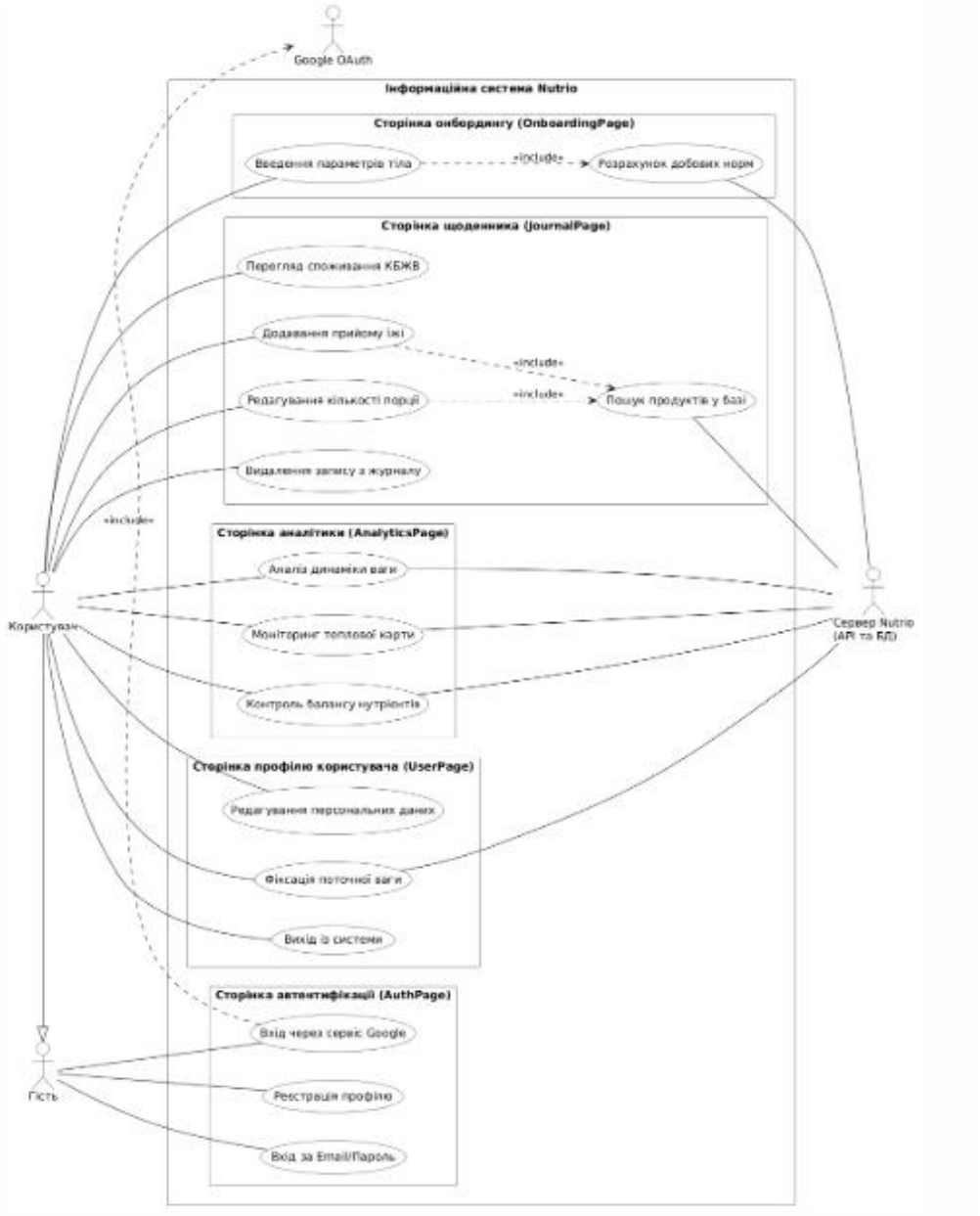

Рисунок 1.2 - Діаграма варіантів використання інформаційної системи «Nutrio»

Суб'єктами взаємодії в системі виступають первинні учасники — неавторизований гість та зареєстрований користувач, а також вторинні обчислювальні модулі — зовнішній сервіс авторизації Google OAuth та серверна частина системи.

Для неавторизованого гостя передбачено базові прецеденти взаємодії з підсистемою безпеки на сторінці автентифікації. Вони включають процедури реєстрації нового облікового запису та входу в систему, зокрема із застосуванням делегованого механізму єдиного входу через зовнішній сервіс.

Після успішної авторизації користувач отримує доступ до розширеного функціонала, що починається з процесу первинного налаштування профілю. Під час цього етапу здійснюється збір персональних параметрів та автоматичний розрахунок індивідуальних добових норм енергоспоживання через обов’язкове звернення до серверного ядра.

Ключовими сценаріями щоденної взаємодії є управління журналом харчування та моніторинг антропометричних змін. Зареєстрований користувач здійснює перегляд, додавання, редагування та видалення записів про прийоми їжі, що реалізується з безпосереднім залученням серверної підсистеми пошуку продуктів.

Для відстеження довгострокового прогресу передбачено роботу з аналітичним модулем та профілем користувача. Система надає інструменти для фіксації поточних метрик тіла та формує детальні графічні звіти щодо динаміки ваги, балансу макронутрієнтів і загальної активності ведення журналу на основі накопичених часових міток.

Усі визначені варіанти використання тісно інтегровані між клієнтською частиною та серверною архітектурою. Це забезпечує високу швидкість обробки інформації, цілісність даних та зручність роботи з інтерфейсом системи.

1.6. Обґрунтування вибору технологій

Вибір технологічного стеку для реалізації інформаційної системи «Nutrio» зумовлений потребою у створенні високоефективного, масштабованого та безпечного продукту, здатного стабільно функціонувати в умовах хмарного розгортання. Для розробки серверної частини обрано платформу ASP.NET Core [7], що пояснюється її високою продуктивністю, наявністю потужних засобів типізації та розвиненою екосистемою бібліотек для реалізації складних бізнес-сценаріїв.

В архітектурі бекенду використано низку критично важливих інструментів: MediatR [8] для реалізації патерну CQRS, що розділяє команди та запити; FluentValidation [9] для автоматизованої перевірки вхідних даних; EF Core [10] як ефективний засіб об’єктно-реляційного відображення; а також бібліотеки JWT Bearer [11] та Google.Apis.Auth [12] для забезпечення захищеної автентифікації користувачів.

Клієнтська частина системи побудована на базі бібліотеки React 19 [13] та інструменту збірки Vite [14], що гарантує швидке оновлення інтерфейсу та високу продуктивність. Серед зовнішніх бібліотек фронтенду варто виділити Recharts [15] для побудови інтерактивних графіків, React Router [16] для навігації між сторінками застосунку, а також власний клієнтський модуль на базі Fetch API [17] для стандартизованої асинхронної взаємодії з серверним API. Такий стек дозволяє створювати складні аналітичні панелі, зберігаючи модульність та легкість підтримки коду.

В якості системи управління базами даних обрано PostgreSQL 18 [18], що характеризується надійністю, підтримкою стандартів ACID та здатністю ефективно працювати зі складними структурами даних. Впровадження технології контейнеризації Docker [19] є ключовим елементом стратегії розгортання «Nutrio». Використання конфігурації docker-compose [20] дозволяє об’єднати серверні компоненти, клієнтську частину, та базу даних у єдине

ізольоване середовище, що гарантує ідентичність роботи додатка на локальних машинах розробників та у хмарній інфраструктурі Azure [21]. Така архітектура забезпечує легке масштабування окремих сервісів, гарантуючи високу доступність та відмовостійкість додатка під навантаженням.

1.7. Розміщення вебдодатка в хмарному середовищі

Вибір стратегії розгортання веб-додатку«Nutrio» є критично важливим етапом, що безпосередньо впливає на стабільність, доступність та масштабованість програмного продукту. На відміну від традиційного віртуального хостингу, який часто обмежує можливості налаштування системного оточення та висуває жорсткі вимоги до підтримки сучасних інструментів контейнеризації, хмарна інфраструктура надає повну свободу конфігурування та автоматизації.

Переваги використання хмарних серверів над звичайним хостингом полягають у наданні високого рівня відмовостійкості, можливості динамічного виділення обчислювальних ресурсів залежно від поточного навантаження та підтримці процесів безперервної доставки програмного забезпечення [22]. Саме тому для розміщення системи було обрано платформу Microsoft Azure. Даний вибір обумовлений глибокою нативною інтеграцією технологій платформи .NET із хмарними сервісами Azure, що забезпечує безшовну синхронізацію інструментарію розробки з виробничим середовищем, спрощує моніторинг стану додатків та мінімізує час на обслуговування інфраструктури.

Інженерна архітектура розгортання «Nutrio» передбачає виділення трьох автономних ресурсів, що забезпечує логічне розділення компонентів системи: два окремі вебдодатки та одна база даних. Перший вебдодаток відповідає за функціонування серверної частини, тоді як другий забезпечує хостинг клієнтської частини, що гарантує незалежне масштабування та вищий рівень безпеки. База даних функціонує як окремий виділений ресурс, що дозволяє оптимізувати зберігання даних та забезпечити цілісність інформації.

Такий підхід, реалізований із застосуванням контейнеризації Docker, дозволяє об’єднати всі компоненти у єдине ізольоване середовище, що гарантує ідентичність роботи додатка на локальних машинах розробників та у хмарі. Завдяки цьому забезпечується висока доступність системи та її готовність до промислової експлуатації під змінними навантаженнями, що вигідно вирізняє «Nutrio» на фоні типових рішень, розміщених на статичних хостинг-платформах.

1.8 Висновок до розділу 1

У першому розділі кваліфікаційної роботи виконано комплексний аналіз предметної області та існуючих аналогів у сфері цифрового контролю харчування. Проведене дослідження підтвердило високу актуальність розробки інформаційної системи «Nutrio», що зумовлено стрімким зростанням попиту на доступні інструменти автоматизації моніторингу раціону.

Завдяки застосуванню інженерного методу моделювання архетипів цільової аудиторії чітко визначено пріоритетні сегменти користувачів та їхні специфічні запити до системи. На основі отриманих даних сформовано комплекс функціональних вимог та спроєктовано оптимізовану архітектуру варіантів використання за компонентною структурою інтерфейсу вебдодатка.

Науково обґрунтовано вибір сучасного технологічного стеку, який базується на платформі ASP.NET Core для серверної частини та бібліотеці React для клієнтського інтерфейсу. Реалізація системи на засадах концепції Чистої архітектури та патерну CQRS у поєднанні з реляційним сховищем даних PostgreSQL 18 закладає надійний фундамент для створення масштабованого та відмовостійкого продукту.

Визначено ефективну стратегію розгортання та супроводу системи із застосуванням інструментів контейнеризації Docker . Спроєктовано розподілену інфраструктуру в хмарному середовищі Microsoft Azure з логічним виділенням трьох автономних ресурсів, що забезпечує нативну сумісність інструментарію

розробки та повну готовність додатка до стабільного функціонування під навантаженням.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ТРЕКЕРА СПОЖИТИХ КАЛОРІЙ

2.1. Архітектурна побудова серверної частини за принципами Чистої архітектури

Проєктування серверної частини вебдодатка «Nutrio» базується на концепції Чистої архітектури, яка забезпечує високий рівень модульності, зменшення зв’язаності бізнес-правил із зовнішніми фреймворками та гнучкість супроводу програмного продукту. Основна інженерна перевага такого підходу полягає в дотриманні принципу інверсії залежностей, де технічні деталі реалізації, включаючи бази даних, вебсервери та сторонні сервіси автентифікації, підключаються через контракти прикладного та доменного рівнів. Серверний стек розділений на чотири логічні шари, кожен з яких виконує визначені функції та має власну зону відповідальності.

Серцем і найвищим рівнем абстракції системи є доменний шар, представлений проєктом доменних моделей та контрактів. Цей рівень не містить залежностей від вебфреймворку та інфраструктурної конфігурації застосунку. Він містить опис базових сутностей, таких як користувач, продукт, добовий запис у журналі харчування та мітка антропометричних показників, а також інтерфейси репозиторіїв, що визначають правила доступу до даних. Такий підхід забезпечує стабільність ключових бізнес-правил при еволюції технічних компонентів системи.

Прикладний шар, також відомий як шар додатку або шар бізнес-логіки, реалізує конкретні бізнес-сценарії інформаційної системи. Взаємодія компонентів на цьому рівні побудована за допомогою патерну розділення відповідальності команд та запитів CQRS [23] із залученням бібліотеки MediatR. Шар застосунку посилається на доменний рівень і містить обробники команд на модифікацію даних, обробники запитів на читання інформації, об’єкти передачі даних та компоненти конвеєрної валідації запитів.

Шар інфраструктури відповідає за взаємодію додатка із зовнішнім середовищем. Тут зосереджена конкретна реалізація репозиторіїв доступу до бази даних PostgreSQL за допомогою EF Core, налаштування контексту даних, а також адаптери безпеки, що забезпечують генерацію маркерів доступу JWT та перевірку токенів авторизації через сервіси Google. Інфраструктурний шар використовує сутності домену та сценарії застосунку, оскільки його головне завдання — надати технічне забезпечення для інтерфейсів, оголошених на вищих рівнях системи.

Рівень інтерфейсу програмування застосунків виступає безпосереднім хостом та вхідною точкою для зовнішніх HTTP-запитів із боку клієнтської частини додатка. Цей шар відповідає за маршрутизацію, конфігурацію контейнера впровадження залежностей та побудову конвеєра обробки запитів за допомогою middleware. Саме на рівні веб-API інтегровано модулі єдиної обробки виняткових ситуацій, сервіси перевірки CORS-політик для взаємодії з клієнтською частиною, а також засоби автоматичної генерації документації OpenAPI [24] та Scalar UI [25], що забезпечує зручну інтеграцію та тестування API-контрактів.

2.2. **Проєктування та розробка реляційної моделі бази даних у PostgreSQL**

Проєктування реляційної моделі даних для веб-додатку системи «Nutrio» здійснюється в середовищі системи управління базами даних PostgreSQL 18. Створення структури таблиць та зв’язків між ними підпорядковане вимогам надійного, безпечного та оптимізованого зберігання даних користувачів, каталогів продуктів, щоденних записів журналу харчування та часових міток антропометричних показників організму.

Робота зі сховищем даних у проєкті реалізована за допомогою технології об’єктно-реляційного відображення EF Core [26], що дозволяє абстрагувати взаємодію з базою даних на рівні об’єктних моделей. При цьому архітектурне

проєктування серверної частини розмежовує оголошення абстрактних контрактів доменного шару та їх технічну реалізацію в інфраструктурі. Інтерфейси репозиторіїв доступу до даних оголошуються у доменному ядрі системи, що спрощує заміну технічних реалізацій та підтримку коду.

Для точного відображення бізнес-правил у реляційну схему було спроєктовано чотири базові доменні сутності.

Сутність користувача, яку відображає таблиця Users, є центральним елементом системи, що зберігає облікові дані для автентифікації та параметри для персоналізації.

Сутність харчового продукту, за яку відповідає таблиця Products, виступає довідником, що містить назву та комплексний об’єкт-значення нутрієнтів на 100 грам продукту. Базове наповнення каталогу продуктів сформовано на основі попередньо підготовленої вибірки даних Open Food Facts [27], після чого використовується у внутрішньому довіднику системи.

Сполучною ланкою є сутність запису журналу харчування, яку відображає таблиця FoodEntries, яка фіксує факт споживання конкретного продукту користувачем із зазначенням маси порції, календарної дати та типу прийому їжі.

Для довгострокового моніторингу прогресу використовується сутність історичної мітки стану тіла, яка зв’язана на таблиця BodyMetricStamps, що дозволяє зберігати хронологічну послідовність змін ваги та інших антропометричних параметрів.

Типи зв'язків між таблицями реляційної моделі даних наведено у таблиці 2.1.

Таблиця 2.1 - Реляційна модель сутностей та зв'язків

Назва таблиці	**Тип зв’язку до якої сутності**
User	1:N FoodEntries
User	1:N BodyMetricStamp
Products	1:N FoodEntries

Продовження таблиці 2.1

FoodEntries	N:1 User
FoodEntries	N:1 Products
BodyMetricStamp	N:1 User

Реєстрація підсистеми бази даних та пов’язаних із нею репозиторіїв в інформаційній системі забезпечується за допомогою механізму впровадження залежностей DI. Для цього в шарі інфраструктури реалізовано метод розширення AddInfrastructure, у межах якого через контейнер сервісів .NET реєструється контекст бази даних із налаштуванням підключення до PostgreSQL через рядок конфігурації. У цьому ж контейнері реєструються конкретні реалізації репозиторіїв із життєвим циклом Scoped, що забезпечує коректне надання екземплярів прикладним обробникам у межах кожного HTTP-запиту.

Управління схемою бази даних та її еволюція автоматизовані за допомогою інструментарію міграцій EF Core [28]. Створені міграції зберігають історію змін реляційної моделі та застосовуються під час запуску застосунку у підтримуваних середовищах виконання (зокрема локально та в тестових сценаріях). Це зменшує обсяг ручних операцій зі схемою, підтримує узгодженість структури БД із кодом і спрощує супровід системи при подальшому розвитку.

2.3. Програмна реалізація бізнес-логіки із застосуванням патерну CQRS

Програмна реалізація основної бізнес-логіки інформаційної системи «Nutrio» зосереджена в прикладному шарі Nutrio.Application та побудована на засадах архітектурного патерну CQRS. Головна інженерна суть цього підходу полягає у чіткому розділенні операцій модифікації даних, так званих команд, та операцій читання інформації, так званих запитів. Для забезпечення слабкої зв’язаності компонентів системи та уникнення прямих залежностей між точками

веб-API й бізнес-обробниками інтегровано бібліотеку MediatR, яка реалізує патерн посередника.

Завдяки впровадженню архітектури CQRS контролери рівня веб-API залишаються спрощеними та не містять доменної логіки. Основне завдання контролерів полягає у прийомі HTTP-запитів, трансформації в об’єкти команд або запитів та подальшій передачі в конвеєр медіатора через асинхронний метод Send. Посередник маршрутизує отриманий об’єкт до відповідного зареєстрованого обробника, який виконує конкретний варіант використання системи.

Операції, що призводять до зміни стану системи, оформлені у вигляді команд, які інкапсулюють вхідні параметри сценарію. До таких операцій належать, зокрема, реєстрація облікового запису користувача, внесення початкових антропометричних даних під час онбордингу, створення нових записів у журналі харчування та фіксація поточних параметрів профілю. Натомість запити відповідають за отримання інформації без зміни її стану: формування щоденного підсумку споживання КБЖВ, пошук продуктів у базі та побудову аналітичних представлень для клієнтського інтерфейсу.

Важливим інженерним рішенням у межах реалізації бізнес-логіки є впровадження автоматизованої конвеєрної перевірки вхідних даних. Завдяки підключенню поведінки медіатора ValidationBehavior кожна команда або запит перед потраплянням до обробника проходить через шар валідації. Перевірка виконується на основі декларативних правил FluentValidation, що підвищує передбачуваність обробки запитів і зменшує кількість помилок у бізнес-обробниках.

Таким чином, використання CQRS у поєднанні з MediatR та FluentValidation забезпечує структуровану реалізацію бізнес-логіки, спрощує підтримку коду та створює стабільну основу для подальшого розширення функціоналу системи.

2.4. Програмна реалізація підсистеми безпеки та автентифікації

Програмна реалізація підсистеми безпеки та автентифікації у веб-додатку «Nutrio» спроєктована з урахуванням сучасних підходів до захисту персональних даних і контролю доступу у веб-додатках. З огляду на розподілену архітектуру системи та формат взаємодії між клієнтською і серверною частинами, базовим механізмом захисту кінцевих точок веб-API обрано технологію маркерів доступу JWT. Це дозволяє реалізувати безстанову автентифікацію та підтримувати масштабовану модель роботи сервера.

У межах базового сценарію доступу система забезпечує перевірку облікових даних користувача, які складаються з адреси електронної пошти та пароля. Під час реєстрації та авторизації паролі користувачів проходять криптографічне хешування [29], що виключає збереження чутливих даних у відкритому вигляді в базі даних. Після успішної перевірки автентифікаційних даних серверна частина генерує маркер доступу JWT, який передається клієнтському додатку для авторизації наступних запитів.

Важливим інженерним рішенням у конфігурації сервера є валідація параметрів JWT Bearer [30]. Конвеєр обробки запитів налаштований на перевірку видавця, аудиторії, терміну дії токена [31] та криптографічного підпису на основі секретного ключа. Така перевірка обмежує можливість використання невалідних або підроблених токенів і забезпечує доступ до захищених ресурсів лише для авторизованих користувачів.

Практична реалізація інтерфейсу користувача під час проходження процедури автентифікації в інформаційній системі представлена на рисунку 2.1.

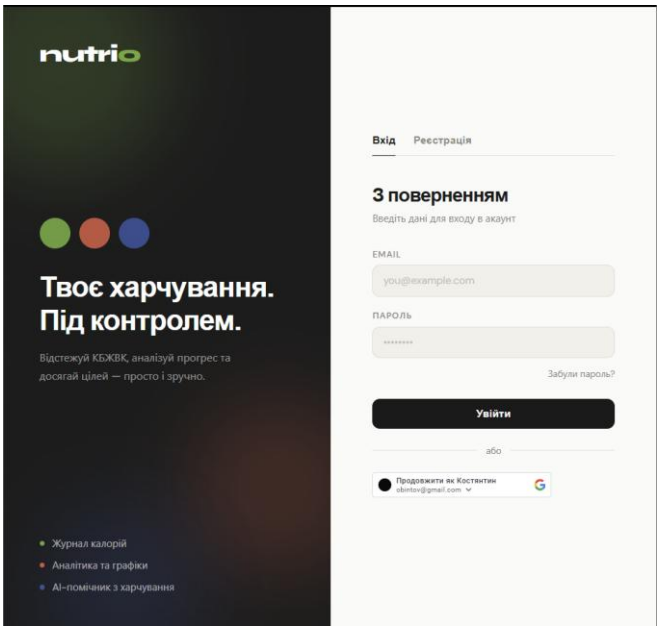

Рисунок 2.1 – Авторизація в системі «Nutrio»

Окрім базової автентифікації, у підсистемі безпеки реалізовано механізм делегованого входу через Google OAuth 2.0. У цьому сценарії клієнтська частина отримує Google ID token і передає його на серверне веб-API. Серверна частина з використанням бібліотеки Google.Apis.Auth [32] виконує перевірку токена, а після успішної валідації формує стандартний JWT системи «Nutrio» для подальшої роботи користувача в захищеній зоні застосунку.

Для спрощення ручної перевірки захищених маршрутів підсистему Bearer-автентифікації інтегровано із засобами документації OpenAPI та Scalar UI. Це дозволяє тестувати запити до захищених ендпоінтів через інтерфейс документації API та швидко перевіряти коректність авторизаційного конвеєра під час розробки і налагодження.

Таким чином, у системі реалізовано практичну багаторівневу модель автентифікації, що поєднує класичний email/password-вхід, делеговану авторизацію через Google і захист API на основі JWT Bearer. Це забезпечує достатній рівень захисту для робочих сценаріїв веб-додатка на поточному етапі розвитку проєкту.

2.5. Програмна реалізація клієнтської частини додатка на базі бібліотеки React

Програмна реалізація клієнтської частини інформаційної системи «Nutrio» побудована на базі бібліотеки React 19 та інструменту збірки Vite 8. Вебдодаток спроєктовано за принципом односторінкового застосунку SPA, що забезпечує динамічність інтерфейсу, швидкий відгук на дії користувача та відсутність повних перезавантажень сторінок під час навігації між внутрішніми розділами системи. Архітектура маршрутизації застосунку реалізована за допомогою React Router v7, який декларативно керує станом навігації та забезпечує захист приватних зон додатка через компонент ProtectedRoute. Кореневий маршрут системи перенаправляє неавторизованих користувачів на сторінку автентифікації, а внутрішня область додатка й етап онбордингу доступні лише після успішної перевірки стану авторизації.

Для оптимізації рендерингу та уникнення надлишкових обчислень у системі застосовано розмежування контекстів керування станом [33]. Глобальний провайдер авторизації AuthProvider охоплює кореневий рівень застосунку, тоді як спеціалізовані контексти такі як JournalDateProvider, AddProductModalProvider, підключаються локально в захищеній робочій зоні. Це дозволяє зменшити кількість зайвих перерендерів у несуміжних частинах інтерфейсу.

Взаємодія з серверною частиною абстрагована через власний клієнтський модуль на базі Fetch API [34], який реалізує централізовану обробку HTTP-запитів і авторизаційних маркерів. Модуль зчитує базову адресу API зі змінних оточення, додає JWT-токен із локального сховища браузера [35] до заголовків запиту та обробляє помилки на єдиному рівні. При отриманні відповіді 401 Unauthorized виконується очищення локальної сесії та ініціюється повернення користувача на сторінку входу.

Інтерфейс головного вікна журналу харчування відображає індикатори прогресу дотримання добових норм калорійності, білків, жирів, вуглеводів та

клітковини, що обчислюються на сервері. На сторінці аналітики інтегровано компоненти бібліотеки Recharts для візуалізації динаміки калорій, розподілу нутрієнтів за прийомами їжі, зміни маси тіла та активності ведення журналу. Такий підхід забезпечує наочне подання даних без перенесення важких обчислень у клієнтський шар.

Для стабільної роботи форм додавання продуктів застосовано механізм примусового скидання стану модальних компонентів через оновлення ключів рендеру. Це дозволяє уникати перенесення тимчасових значень між сесіями відкриття модального вікна та підтримує передбачувану поведінку інтерфейсу.

Отже, клієнтська частина «Nutrio» реалізована як модульний SPA-застосунок із захищеною маршрутизацією, контекстним керуванням станом, централізованим HTTP-доступом до API та інтерактивною аналітичною візуалізацією, що відповідає поточним функціональним вимогам проєкту.

2.6 Контейнеризація сервісів та налаштування конфігурації Docker

Впровадження технологій контейнеризації є важливою частиною забезпечення стабільного розгортання, переносимості та передбачуваності роботи веб-додатку «Nutrio». Використання платформи Docker дозволяє ізолювати архітектурні компоненти веб-додатка в окремі контейнери та мінімізувати залежність від відмінностей локальних середовищ розробки.

Для серверної частини системи на ASP.NET Core реалізовано багатоетапну збірку [36]. Такий підхід розділяє етапи відновлення залежностей, компіляції коду та формування фінального runtime-образу. У результаті кінцевий контейнер містить лише необхідні компоненти виконання та опублікований застосунок, що зменшує розмір образу та спрощує доставку.

Контейнеризація клієнтської частини на React передбачає побудову production-збірки через Vite з подальшим розміщенням статичних файлів у контейнері на базі Nginx. У поточній конфігурації Nginx виконує роль вебсервера для роздачі SPA-ресурсів та маршрутизації клієнтських шляхів

на index.html. Взаємодія клієнтської та серверної частин здійснюється напряму через налаштовану змінну середовища VITE_API_BASE_URL.

Оркестрація та одночасний запуск компонентів системи в локальному середовищі автоматизовані за допомогою docker-compose [37]. У межах конфігурації запускаються три ключові служби: клієнтський застосунок Nutrio-app-web, серверний API Nutrio-app та база даних PostgreSQL Nutrio-app-db. Стан і взаємодію контейнерів у процесі локального запуску з Docker Desktop показано в рисунку 2.4.

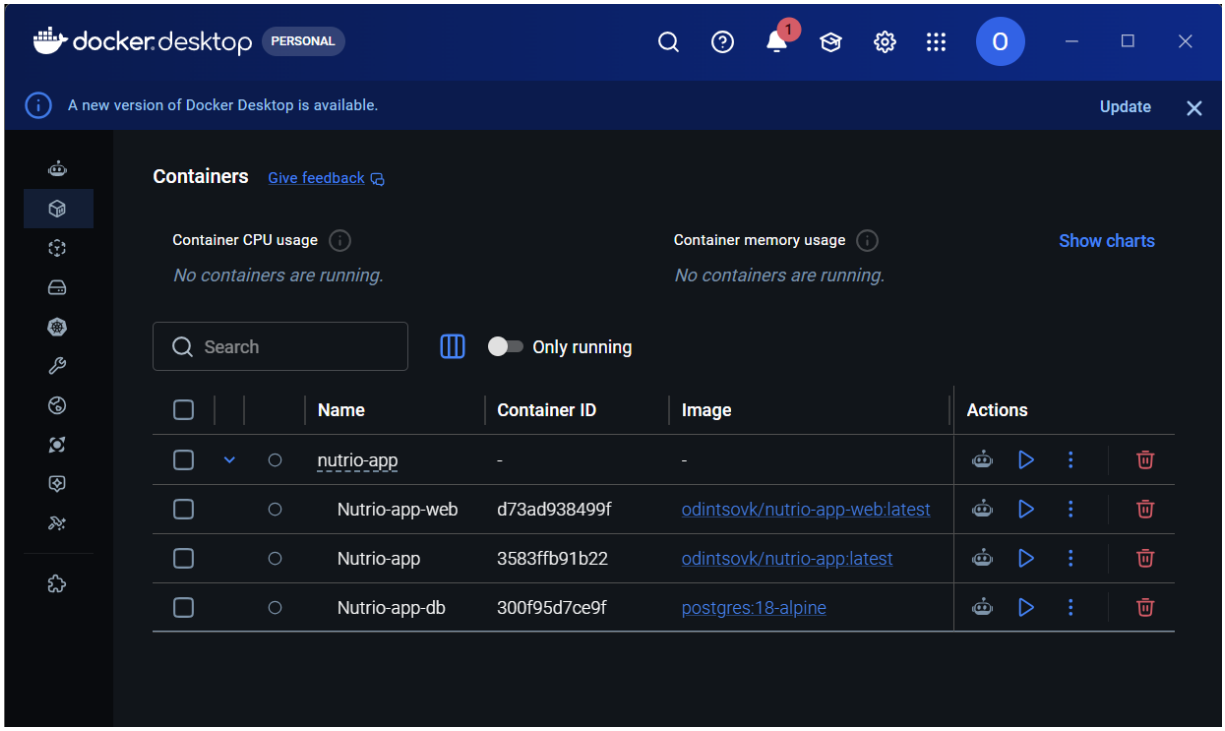

Рисунок 2.4 — Статус та конфігурація активних контейнерів системи «Nutrio» у Docker Desktop

Конфігурація контейнерів передбачає розмежування ролей сервісів і передавання параметрів через змінні середовища, такі як рядки підключення, параметри JWT, адреси API. Для локального середовища порт бази даних може бути опублікований на хост-машину для налагодження та адміністрування, тоді як у продуктивному середовищі доступ до БД має обмежуватися мережевими правилами платформи розгортання.

Контейнеризована модель локального запуску узгоджується з цільовим хмарним розміщенням. Фінальне розгортання реалізується шляхом використання підготовлених контейнерних образів для окремих компонентів системи у середовищі Azure. У цій конфігурації ресурси клієнтської частини, серверної частини та бази даних розділені логічно, що спрощує масштабування та супровід. Розділення на ресурси у середовищі Azure показано на рисунку 2.5.

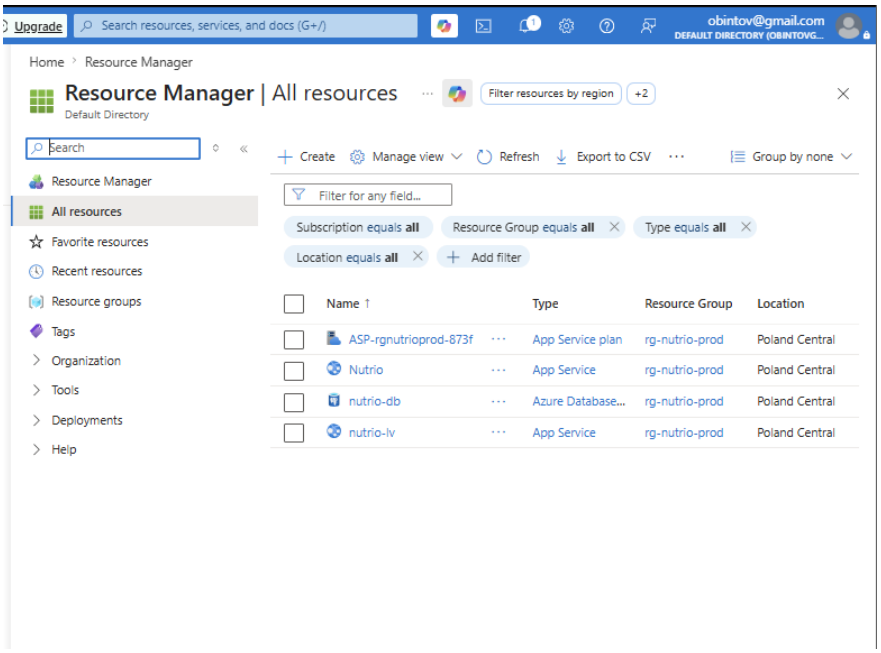

Рисунок 2.5 — Відображення виділених хмарних ресурсів інформаційної системи «Nutrio» в Microsoft Azure

Відповідно до обраної архітектурної стратегії, у хмарному середовищі використовуються автономні ресурси для вебчастин застосунку та окремий керований ресурс бази даних PostgreSQL [38]. Такий підхід забезпечує керованість інфраструктури, відокремлення відповідальностей між компонентами та практичну готовність системи до експлуатації з подальшим розвитком.

2.7. Тестування працездатності компонентів системи

Заключним етапом практичної реалізації інформаційної системи «Nutrio» є проведення тестування її компонентів. Метою цього етапу є підтвердження працездатності основних модулів, перевірка коректності взаємодії між клієнтською та серверною частинами, а також контроль стабільності ключових бізнес-сценаріїв у робочих умовах.

Тестування серверної частини виконувалося із застосуванням автоматизованих тестів у проєкті Nutrio.Tests [39]. Набір перевірок охоплює декілька рівнів:

- Модульні тести доменного шару. Тут відбувається валідація value objects і базових інваріантів.
- Інтеграційні тести API-сценаріїв [40]. Тут відбувається реєстрація, авторизація, доступ до захищених ендпоінтів, робота журналу та аналітики.
- Інтеграційні перевірки з використанням реальної PostgreSQL через Testcontainers [41].

Під час перевірки захищених маршрутів підтверджено коректну поведінку конвеєра безпеки: запити без валідного JWT-маркера повертають 401 Unauthorized, тоді як авторизовані запити успішно отримують доступ до ізольованих ресурсів API. Окремо перевірено сценарії помилок валідації та некоректних облікових даних, що дозволило підтвердити передбачувану поведінку обробки винятків.

Результати запуску автоматизованих тестів серверної частини наведено на рисунку 2.6.

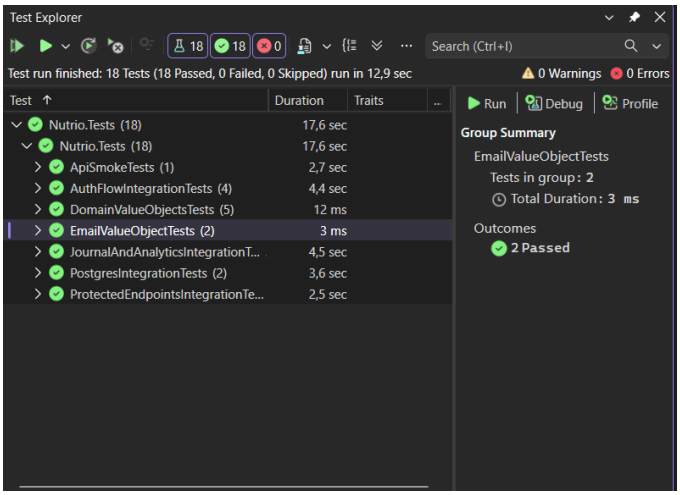

Рисунок 2.6 — Результати виконання тестів серверної частини в Test Explorer

Функціональне тестування клієнтської частини виконувалося засобами Vitest [42]. Набір фронтенд-перевірок включає тести обчислювальної логіки, API-клієнта, захисту маршрутів і відображення окремих UI-компонентів. Це дозволило підтвердити коректність базових сценаріїв роботи SPA-інтерфейсу та стабільність обробки авторизаційного стану.

Результати запуску автоматизованих тестів клієнтською частини наведено на рисунку 2.7.

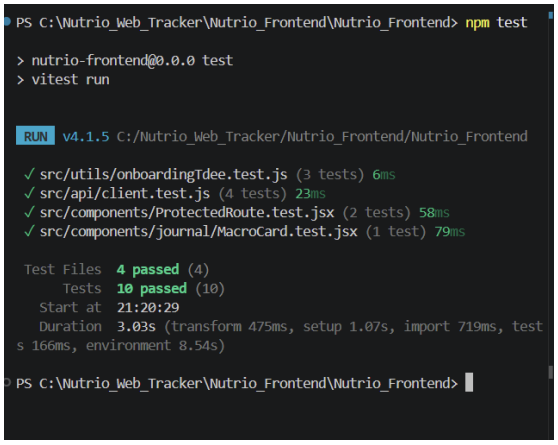

Рисунок 2.7 — Результати виконання тестів клієнтської частини

Отримані результати тестування підтвердили працездатність ключових компонентів інформаційної системи «Nutrio», коректність обробки основних

бізнес-сценаріїв та узгоджену взаємодію між клієнтською, серверною частинами і реляційним сховищем даних.

2.8 Висновок до розділу 2

Під час виконання другого розділу було реалізовано основні компоненти веб-додатку«Nutrio» та перевірено їхню працездатність у зв’язці клієнтської й серверної частин. Серверна архітектура побудована за підходом Чистої архітектури із розділенням на доменний, прикладний, інфраструктурний та API-рівні, а прикладна логіка реалізована з використанням патерну CQRS на базі MediatR.

Реляційна модель даних реалізована у PostgreSQL через EF Core та підтримується механізмом міграцій. Ключові сутності предметної області інтегровані в єдиний контур обробки даних із реєстрацією репозиторіїв через DI-контейнер. Каталог продуктів сформовано на основі попередньо підготовленої вибірки даних Open Food Facts, адаптованої для практичного використання в системі.

Підсистема безпеки реалізована на базі JWT Bearer із перевіркою основних параметрів токена, а також доповнена сценарієм делегованого входу через Google token validation. Це забезпечило контроль доступу до захищених ендпоінтів і передбачувану поведінку API в авторизованих та неавторизованих сценаріях.

Клієнтська частина реалізована як SPA на React 19 і Vite 8, із маршрутизацією через React Router v7, локальними контекстами стану та аналітичною візуалізацією через Recharts. Взаємодія з бекендом організована через власний HTTP-модуль на базі Fetch API із централізованою обробкою токена та помилок авторизації.

Локальне розгортання системи автоматизовано через Docker і docker-compose із трьома сервісами. Контейнер Nginx у поточній конфігурації використовується як вебсервер для роздачі статичних ресурсів клієнтської

частини. Хмарне розміщення реалізовано через підготовлені контейнерні образи та розділення ресурсів за ролями компонентів.

Працездатність системи підтверджено набором автоматизованих тестів: модульними, інтеграційними API-тестами та перевірками з реальною PostgreSQL через Testcontainers, а також тестами клієнтської частини. Зафіксовані результати запуску показали успішне проходження тестових сценаріїв без критичних збоїв, що підтверджує технічну цілісність реалізованого рішення на поточному етапі розвитку проєкту.

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ОПИС ВЕБ-ДОДАТКУ ТРЕКЕРА КАЛОРІЙ

3.1 Тестування процесу реєстрації та авторизації

Розробка інтерфейсу користувача для сторінок автентифікації здійснювалася з дотриманням принципів сучасного мінімалізму та ергономіки, що мало на меті забезпечити швидкий доступ до робочого простору системи та мінімізувати час на проходження формальних процедур перевірки.

Процес побудови сторінки автентифікації передбачав паралельну реалізацію двох альтернативних інженерних сценаріїв ідентифікації. Перший сценарій базувався на класичній формі введення адреси електронної пошти та пароля користувача. Під час проектування форми особлива увага приділялася інтерактивному зворотному зв'язку: поля введення оснащувалися динамічними масками та індикаторами стану.

Візуальне представлення розробленого користувацького інтерфейсу побудованої екранної форми автентифікації наведено на рисунку 3.1.

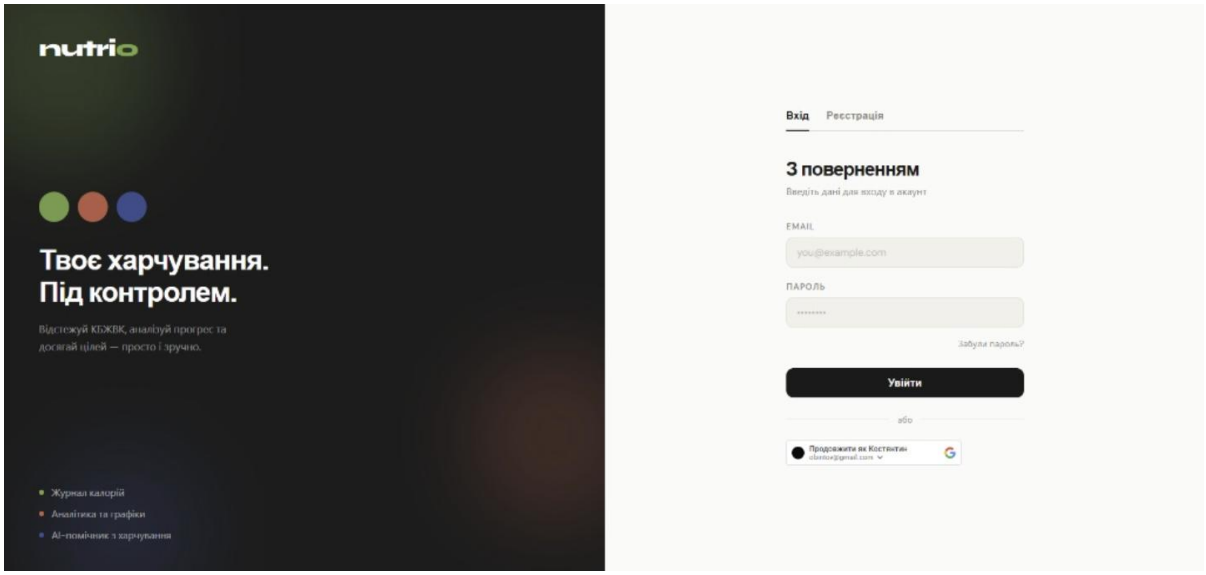

Рисунок 3.1 - Користувацький інтерфейс сторінки автентифікації системи «Nutrio»

Другий сценарій реалізовувався шляхом інтеграції окремого компонента для взаємодії з хмарною службою єдиного входу Google OAuth 2.0 [43]. Це дозволило спростити клієнтський шлях до одного натискання кнопки, забезпечуючи автоматичне отримання сертифікованого маркера безпеки від стороннього провайдера.

Розробка форми реєстрації нових користувачів супроводжувалася впровадженням жорстких механізмів клієнтської валідації вхідних потоків даних у реальному часі [44]. Перевірка коректності синтаксису, унікальності поштової адреси та відповідності встановленим інженерним правилам складності паролів здійснювалася безпосередньо на стороні React-додатка перед надсиланням результуючого об'єкта команди на серверне веб-API, що дозволило значно розвантажити обчислювальні ресурси бекенду, скоротити кількість зайвих HTTP-запитів до СКБД PostgreSQL та миттєво інформувати клієнта про помилки введення.

Графічне представлення екранної форми створення нового облікового запису з інтегрованими полями первинних даних та компонентами інтерактивної валідації наведено на рисунку 3.2.

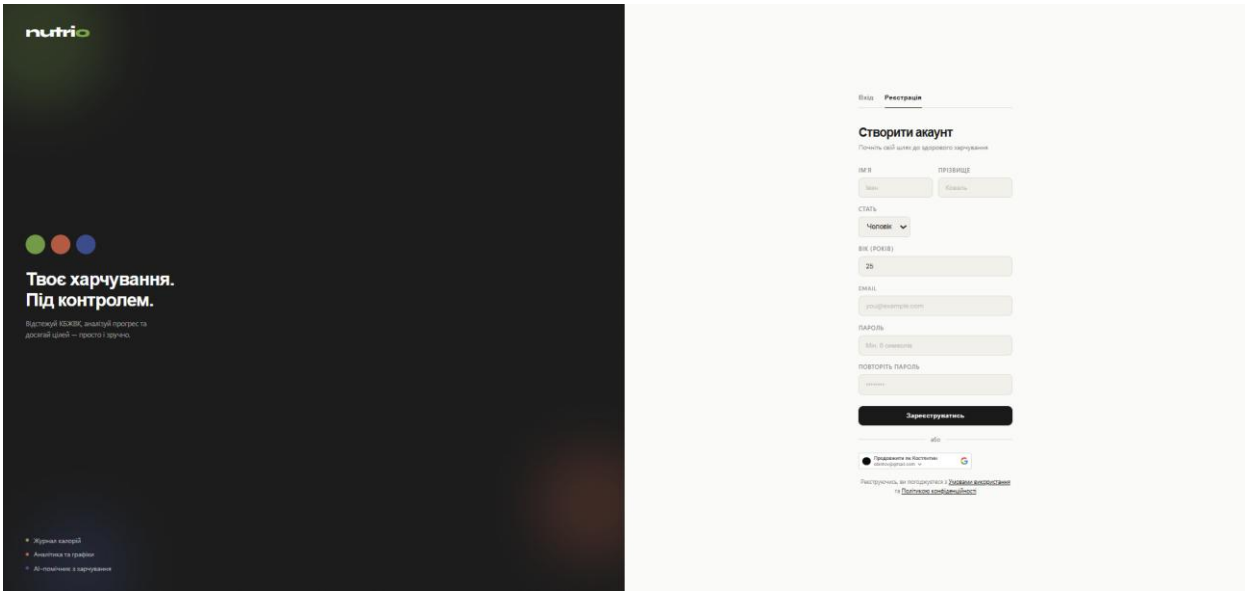

Рисунок 3.2 — Користувацький інтерфейс сторінки реєстрації системи «Nutrio»

Окремим інженерним етапом після успішного підтвердження реєстраційних даних стало впровадження обов'язкового процесу первинного анкетування для збору базових антропометричних та метаболічних показників користувача. Розроблений покроковий інтерфейс забезпечує послідовне введення статі, віку, поточної маси тіла, зросту, а також вибір рівня щоденної фізичної активності та глобальної фітнес-цілі. Отримані структуровані дані автоматично трансформуються клієнтською частиною у формат DTO-об'єкта та надсилаються на сервер, де бізнес-логіка застосунку за допомогою вбудованих математичних алгоритмів розраховує індивідуальну добову норму калорійності та оптимальний баланс КБЖВ, що є базовим фундаментом для подальшого функціонування щоденника.

Візуальне представлення екранних форм інтерфейсу первинного онбордінгу та введення антропометричних параметрів наведено на рисунку 3.3.

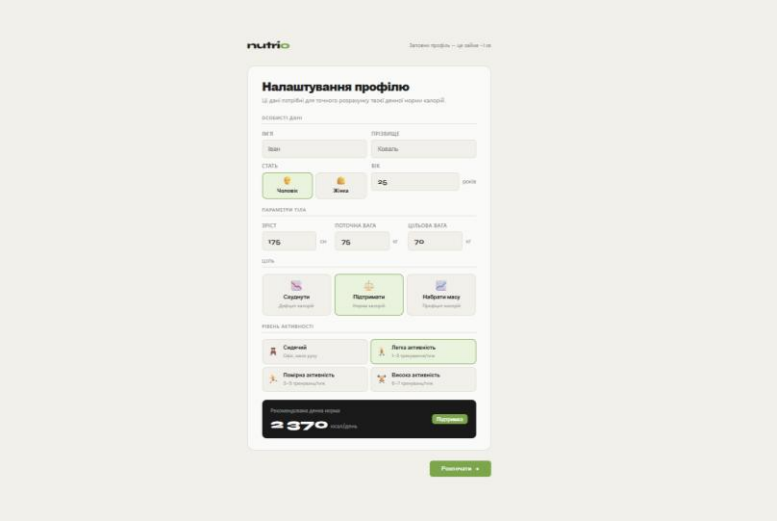

Рисунок 3.3 — Інтерфейс екранних форм покрокового онбордінгу користувача в «Nutrio»

Контроль за проходженням сесій користувачів після завершення етапу автентифікації виконувався через моніторинг конвеєра проміжних програмних модулів. У разі успішного збігу криптографічних хешів паролів або

підтвердження токена від сервісів Google, інфраструктурний шар бекенду здійснював формування результуючого JWT-маркера. Клієнтська частина додатка забезпечувавала перехоплення цього маркера, його надійне збереження в локальному ізольованому сховищі браузера та ініціювала автоматичний перенаправлення користувача на внутрішні захищені сторінки вебдодатка.

3.2. Функціонал сторінки «Журнал калорій» та додавання продукту

Центральним ядром практичної реалізації клієнтської частини веб-додатку є модуль щоденного журналу харчування. Робочий простір сторінки структурно розділено на дві основні логічні зони. Верхня частина відведена під інтерактивну панель прогресу з комплексними круговими індикаторами, що відображають поточний стан дотримання індивідуальної добової норми калорій та макронутрієнтів. Нижча частина містить вертикальну архітектуру групування спожитих страв за класичними прийомами їжі з деталізацією енергетичної цінності кожної порції.

Графічне представлення головного вікна щоденника наведено на рисунку 3.4.

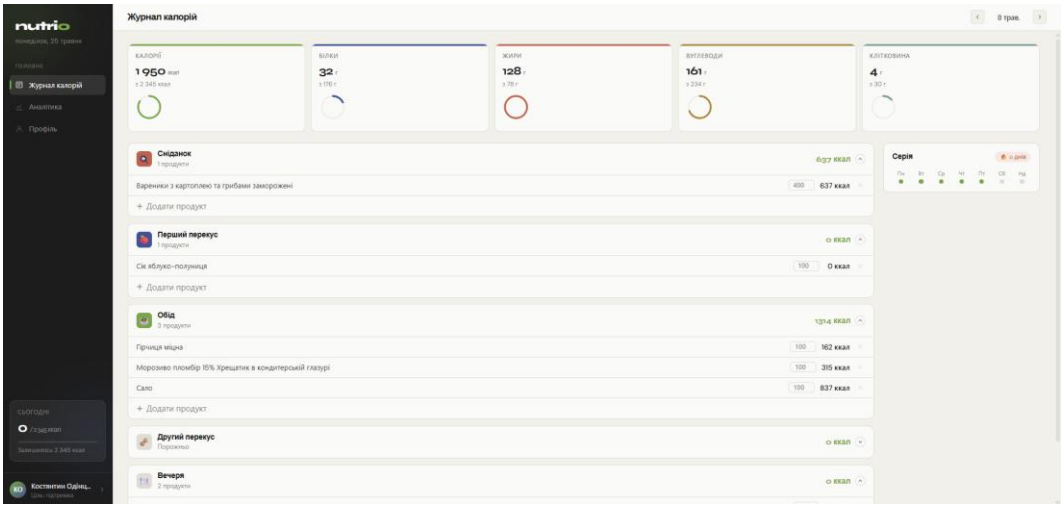

Рисунок 3.4 — Користувацький інтерфейс сторінки щоденника харчування системи «Nutrio»

Процес внесення нових даних оптимізовано завдяки використанню спеціалізованого модального вікна пошуку та додавання продуктів. Під час взаємодії користувача з формою система здійснює повнотекстовий пошук у базі даних та дозволяє швидко обрати потрібний продукт. Завдяки реактивним можливостям фреймворку React та оптимізованим HTTP-запитам до бекенду, будь-які зміни, такі як додавання, модифікація кількості або видалення, ініціюють миттєвий перерахунок загальної суми КБЖВ та плавний рендеринг індикаторів без повного перезавантаження сторінки. Візуалізація вікна додавання продукту представлена на рисунку 3.5.

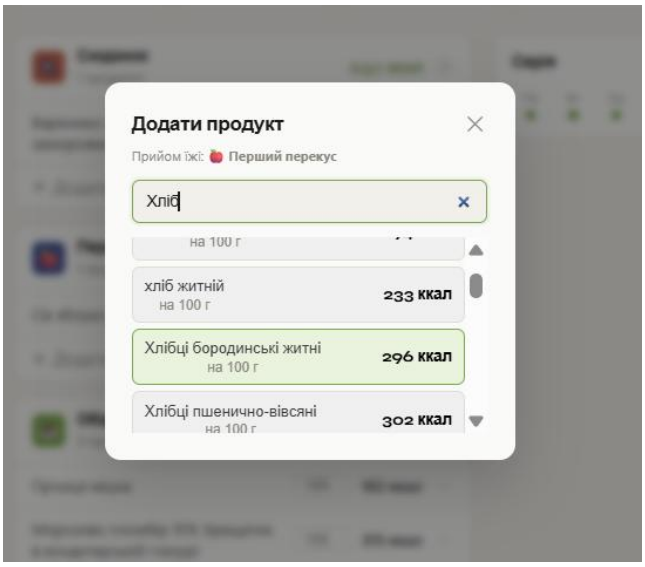

Рисунок 3.5 — Модальне вікно пошуку та додавання продукту до щоденника

Такий ергономічний підхід до реалізації журналу харчування максимально спрощує щоденний контроль раціону. Це мінімізує когнітивне навантаження на користувача та підвищує загальну ефективність використання вебдодатка.

3.3 Функціонал сторінки «Аналітика»

Важливим етапом перевірки експлуатаційної придатності системи «Nutrio» став аналіз підсистеми збору та відображення глибокої аналітики.

Розробка цього модуля мала на меті надати користувачеві наочні інструменти для довгострокового контролю динаміки маси тіла та оцінки збалансованості щоденного раціону.

Графічне представлення розробленої сторінки аналітики наведено на рисунку 3.6.

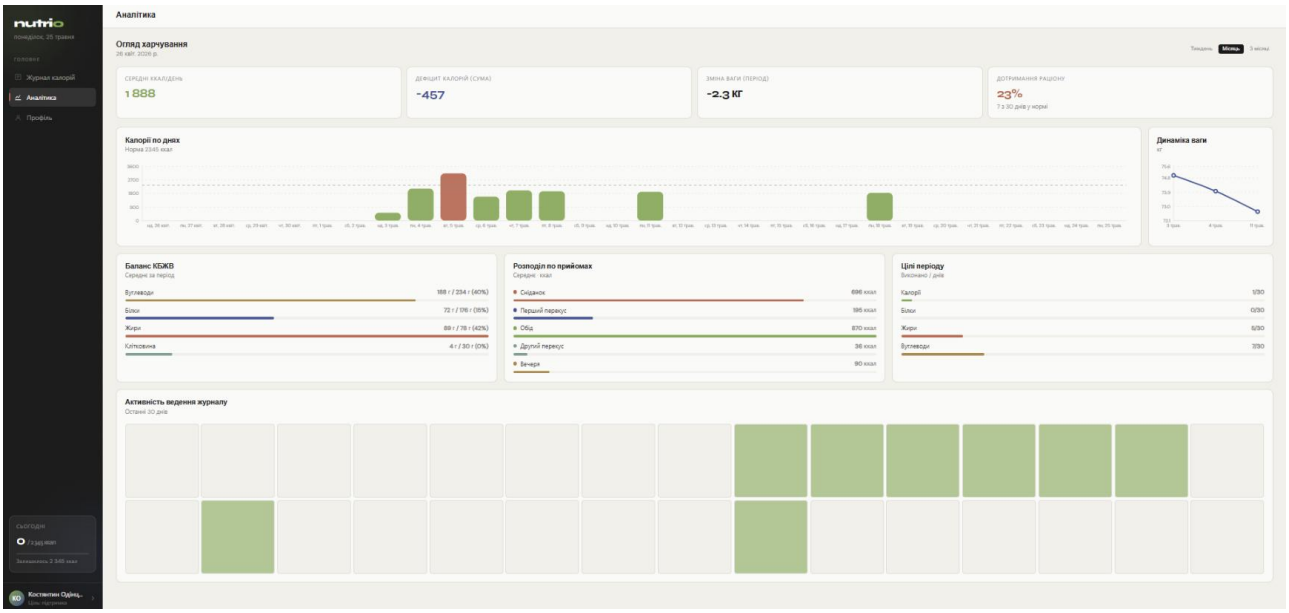

Рисунок 3.6 - Візуалізація аналітичних звітів та теплової карти активності в «Nutrio»

Програмна реалізація інтерфейсної зони аналітики здійснювалася шляхом інтеграції спеціалізованої бібліотеки Recharts. Використання декларативних компонентів Recharts дозволило створити складні інтерактивні графіки, які легко масштабуються та швидко рендеряться. Детальніше про її використання на цій сторінці:

1. Стовпчикові діаграми BarChart: Застосовано у блоці «Калорії по днях». Бібліотека Recharts автоматично вибудовує осі координат та масштабує стовпчики залежно від переданого масиву даних. Також на графік накладено лінію-орієнтир, яка візуально позначає індивідуальну денну норму калорій. Це дозволяє користувачу з одного погляду оцінити дні профіциту, які позначені червоним кольором, та дефіциту, відповідно позначеним зеленим кольором.

2. Лінійні графіки LineChart: Використано у блоці «Динаміка ваги». Графік з'єднує історичні мітки ваги плавними лініями, візуалізуючи загальний тренд схуднення.

3. Лінійні індикатори прогресу: У блоках «Баланс КБЖВ» та «Розподіл по прийомах» реалізовано горизонтальні смуги прогресу, які розраховують відсоткове співвідношення спожитих нутрієнтів до денної цілі, забезпечуючи швидку оцінку збалансованості раціону.

Окремим інженерним рішенням у межах аналітичної панелі стало впровадження координатної сітки активності користувача у формі теплової карти (Heatmap) («Активність ведення журналу»). Цей компонент динамічно зафарбовує часові клітинки матриці зеленим кольором залежно від наявності записів у щоденнику. Така візуалізація виконує роль гейміфікаційного фактора, мотивуючи користувача не пропускати дні.

Висока продуктивність сторінки аналітики досягається завдяки тому, що клієнтська частина не займається важкими обчисленнями. Патерн CQRS на бекенді дозволяє серверу самостійно згрупувати дані через оптимізовані SQL-запити та повернути на фронтенд готові легкі масиви у форматі DTO, які Recharts миттєво перетворює на графіки.

3.4 Функціонал сторінки «Профіль»

Кінцевим компонентом клієнтського інтерфейсу інформаційної системи «Nutrio» є модуль персонального профілю користувача. Цей підрозділ виступає в ролі керуючого центру, де акумулюються базові антропометричні параметри, здійснюється керування обліковим записом та відображаються розраховані метаболічні коефіцієнти. Інтерфейс розроблено із застосуванням модульних карток, що дозволяє структурувати велику кількість персональних полів без візуального перевантаження користувача.

Графічне представлення сторінки налаштувань профілю наведено на рисунку 3.7.

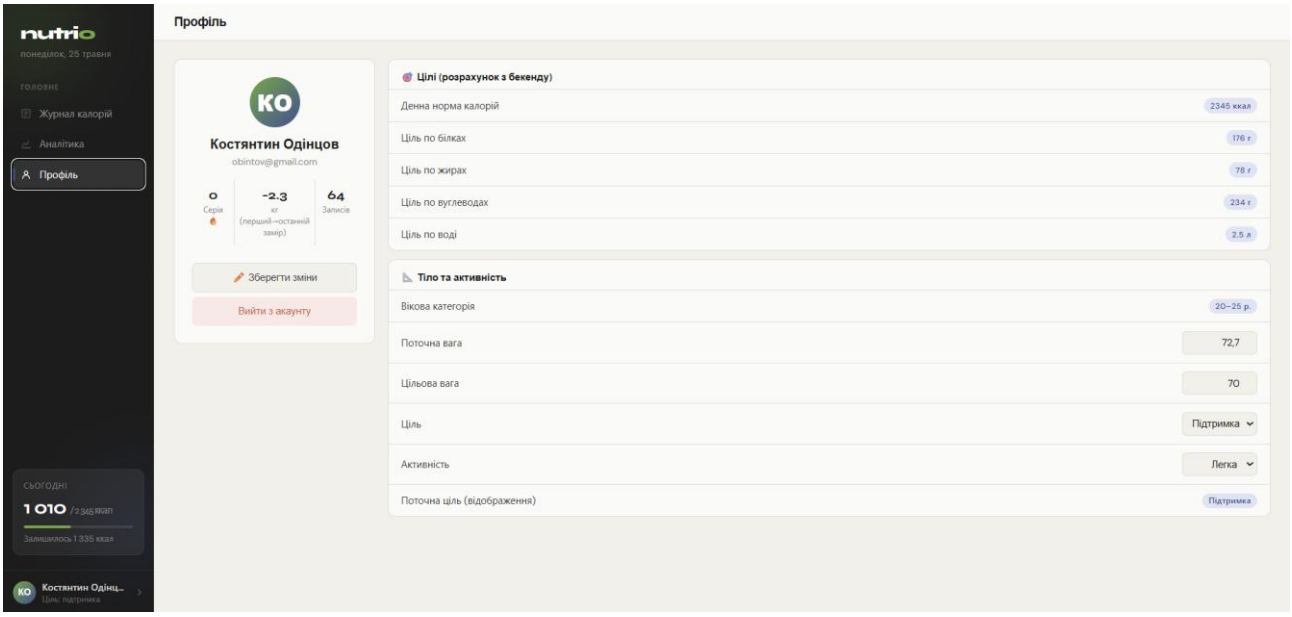

Рисунок 3.7 - Користувацький інтерфейс сторінки профілю системи «Nutrio»

Візуальна архітектура сторінки розділена на кілька логічних блоків. Перший блок містить базову інформацію про обліковий запис: ім'я користувача, адресу електронної пошти та обрану глобальну фітнес-ціль. Наступний блок забезпечує відображення та оперативне редагування поточних фізичних показників, таких як стать, вік, зріст, вага, а також коефіцієнт повсякденної фізичної активності. Окреме місце відведено під інформаційну картку «Ваші норми КБЖВ», яка наочно демонструє цільові добові ліміти калорійності, білків, жирів та вуглеводів, що є орієнтиром для роботи щоденника харчування.

З технічної точки зору взаємодія з даною формою побудована за принципом реактивного збереження станів. Користувач має можливість динамічно змінювати будь-який антропометричний показник. При натисканні кнопки збереження змін клієнтська частина ініціює HTTP-запит типу PUT на серверне веб-API. Обробник команд на стороні бекенду вносить нові координати до БД в PostgreSQL, автоматично перераховує індивідуальну норму КБЖВ за вбудованими математичними формулами та повертає оновлений об'єкт на фронтенд, забезпечуючи миттєву адаптацію інтерфейсу додатка під нові цілі користувача.

3.5. Мобільна версія користувацького інтерфейсу

Важливим інженерним критерієм експлуатаційної якості та практичної придатності інформаційної системи «Nutrio» є забезпечення доступності її функціонала з будь-яких типів пристроїв. Оскільки значна частина кінцевих користувачів взаємодіє з щоденниками харчування динамічно протягом дня безпосередньо через мобільні телефони, під час програмної реалізації клієнтської частини на React особлива увага приділялася побудові чуйного та адаптивного користувацького інтерфейсу [45]. Процес адаптації здійснювався за допомогою методів верстки на базі Flexbox [46] та CSS Grid, що дозволило повністю відмовитися від розробки громіздких окремих мобільних версій або нативних застосунків, динамічно підлаштовуючи структуру сторінок під поточні параметри розширення екрана . Візуальне відображення адаптованої сторінки налаштувань користувача наведено на рисунку 3.8.

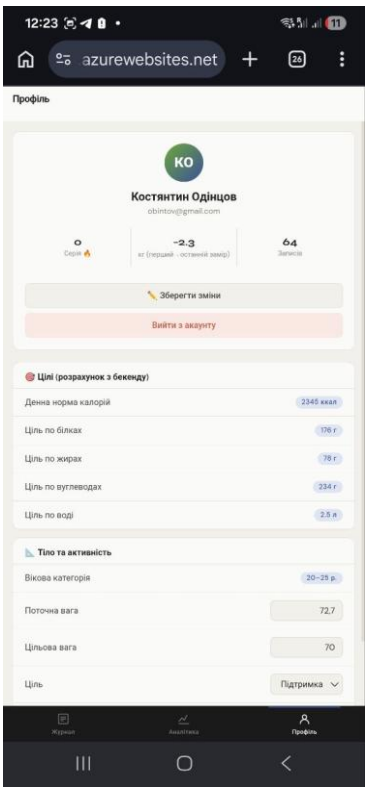

Рисунок 3.8 — Мобільна версія користувацького інтерфейсу сторінки профілю в системі «Nutrio»

Під час проектування мобільного відображення профілю головною метою було уникнення горизонтального прокручування сторінки та збереження читабельності текстових лімітів КБЖВ. Для цього всі інформаційні картки, які в десктопній версії розташовані горизонтально пліч-о-пліч , на мобільних екранах автоматично перешикувалися за вертикальною віссю. При цьому розмір шрифтів та полів введення антропометричних даних динамічно масштабувався, забезпечуючи зручність сенсорного керування та натискання кнопок пальцями однієї руки.

Окрім блоку налаштувань, значних адаптивних змін зазнав інтерфейс щоденного моніторингу нутрієнтів та історичних трендів. Візуальне представлення адаптованого під екрани смартфонів аналітичного модуля системи наведено на рисунку 3.9.

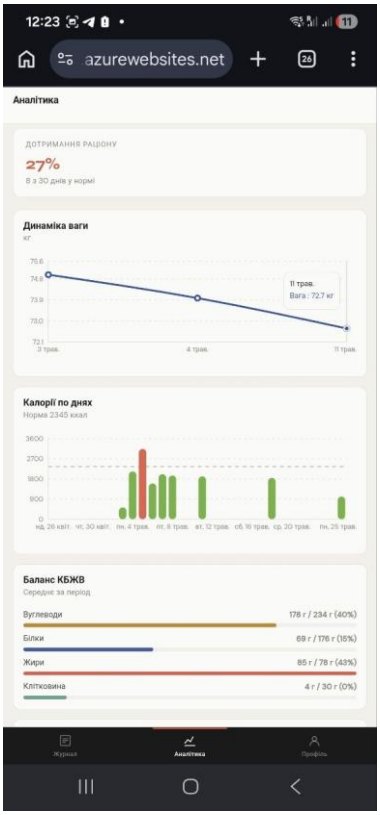

Рисунок 3.9 — Мобільна версія користувацького інтерфейсу сторінки аналітики в системі «Nutrio»

Особливістю розробки цієї сторінки стало забезпечення гнучкості інтегрованих графіків бібліотеки Recharts. Компоненти стовпчикових діаграм «Калорії по днях» та лінійних графіків «Динаміка ваги» були загорнуті у спеціалізовані адаптивні контейнери, що змусило їх автоматично стискатися до ширини мобільного дисплея без спотворення осей координат та текстових позначок. З метою економії екранного простору на смартфонах, великі картки огляду показників трансформувалися у компактні вертикальні списки, а бічна навігаційна панель десктопного інтерфейсу була повністю перенесена у нижню частину екрана у вигляді зафіксованого меню швидкого перемикання між щоденником, аналітикою та профілем. Такий підхід дозволив зберегти 100% функціональних можливостей системи на мобільних пристроях, гарантуючи комфортний і швидкий користувацький досвід.

3.6 Висновок до розділу 3

У розділі 3 кваліфікаційної роботи на тему розробки кросплатформової інформаційної системи для автоматизації контролю раціону харчування було проведено комплексну практичну демонстрацію та всебічну верифікацію працездатності розгорнутого користувацького інтерфейсу, що повністю підтвердило коректність реалізації основних бізнес-модулів системи «Nutrio». Експлуатація розгорнутого веб-додатка дозволила наочно оцінити ергономічність взаємодії користувача із екранними формами модулів автентифікації та реєстрації, покроковою сторінкою первинного онбордингу, інтерактивним щоденним журналом харчування та аналітичною панеллю. Оцінка поведінки реактивних клієнтських компонентів у реальному часі підтвердила високу якість користувацького досвіду, миттєву зміну стану кругових індикаторів добових лімітів КБЖВ при маніпуляціях із порціями, а також високу швидкість асинхронного відгуку SPA-інтерфейсу на базі React 19 без необхідності повних перезавантажень сторінок додатка.

Аналіз поведінки розроблених інтерфейсів на мобільних пристроях підтвердив бездоганне адаптивне відображення та збереження 100% функціональних можливостей хмарного застосунку на екранах смартфонів і планшетів. Завдяки методам гнучкої верстки Flexbox та CSS Grid, складні бізнес-компоненти, форми прийомів їжі та інтегровані аналітичні графіки бібліотеки Recharts автоматично трансформуються та перешиковуються за вертикальною віссю. Це дозволило повністю уникнути горизонтального прокручування екрана, оптимізувати простір за рахунок нижнього фіксованого меню навігації та забезпечити високу читабельність числових показників добових норм нутрієнтів і клітковини. Практична демонстрація процесів створення профілю, повнотекстового пошуку продуктів у базі даних та фіксації антропометричних міток повністю підтвердила логічну цілісність системи, стабільність збереження інформації у реляційному сховищі PostgreSQL та надійний захист даних за рахунок наскрізної валідації і безстанового сесійного контролю JWT Bearer.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОХОРОНА ПРАЦІ

4.1. Долікарська допомога при харчових отруєннях

Організація безпечного життєвого середовища та правова регламентація санітарно-епідеміологічного благополуччя в Україні базується на чинній нормативно-законодавчій базі. Фундаментальним документом у цій сфері є Закон України «Про забезпечення санітарного та епідемічного благополуччя населення» [48], який встановлює обов'язкові інженерно-гігієнічні вимоги до обробки, зберігання та реалізації харчових продуктів з метою запобігання масовим отруєнням.

Окрім цього, загальні правові засади захисту життя та здоров'я громадян у процесі будь-якої діяльності визначаються Законом України «Про охорону праці» [49]. Гігієнічні критерії та класифікація шкідливих факторів, що можуть впливати на стан організму при порушенні режимів харчування під час трудового процесу, узгоджуються з Гігієнічним нормативом «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу» [50]. Впровадження інтелектуальних автоматизованих модулів для відстеження термінів придатності та температурних режимів зберігання інгредієнтів дозволяє значно знизити ризик деструкції ліпідів і білків, що запобігає утворенню небезпечних бактеріальних токсинів.

Своєчасне розпізнавання клінічної картини гострої інтоксикації є ключовим етапом для успішного проведення лікувально-профілактичних заходів. Основними симптомами харчового отруєння, які зазвичай проявляються у проміжку від 2 до 6 годин після вживання неякісної їжі, є нудота, багаторазове блювання, спазматичний біль в епігастральній ділянці шлунка, діарея, загальна слабкість, а також субфебрильна або фебрильна температура тіла. У разі

виникнення подібних станів критично важливою є перша долікарська допомога, яка спрямована на якнайшвидше виведення токсинів з організму до моменту їх повного всмоктування у системний кровотік. Першим інженерно-медичним кроком є детоксикація шлунково-кишкового тракту шляхом промивання шлунка. Потерпілому необхідно дати випити від 1 до 1,5 літра чистої теплої води або слабкого 0,1% розчину перманганату калію чи розчину харчової соди, після чого штучно викликати блювання шляхом подразнення кореня язика; цю маніпуляцію повторюють до появи абсолютно чистої промивної води.

Наступним обов'язковим етапом долікарської допомоги є зв'язування та нейтралізація залишків патогенів у кишківнику за допомогою ентеросорбентів. Потерпілому вводять активоване вугілля, із розрахунку 1 грам на 10 кілограмів маси тіла або сучасні високодисперсні силікатні та гелеві сорбенти, які мають високу питому поглинальну здатність.

Оскільки інтенсивне блювання та діарея призводять до стрімкого зневоднення організму та вимивання життєво важливих електролітів, третім етапом є відновлення водно-сольового балансу. Для цього застосовують спеціалізовані аптечні сольові розчини або здійснюють часте дробове пиття невеликими порціями теплої мінеральної води без газу чи несолодкого чаю. При цьому категорично забороняється приймати знеболювальні або антидіарейні препарати до огляду лікарем, оскільки вони можуть замаскувати симптоми гострої хірургічної патології або затримати токсини всередині організму.

Після надання первинної допомоги потерпілому забезпечують повний фізичний спокій, зігрівають кінцівки за допомогою грілок та здійснюють обов'язковий виклик бригади екстреної медичної допомоги для запобігання розвитку ускладнень, таких як інфекційно-токсичний шок або гостра ниркова недостатність.

4.2 Вимоги ергономіки до організації робочого місця оператора ПК

Ефективність робочої діяльності, якість виконання поставлених задач та тривалість збереження високої працездатності фахівців безпосередньо залежать від рівня ергономічної організації їхнього безпосереднього робочого простору. Сучасне робоче місце оператора персонального комп'ютера проектується як складна біотехнічна система, в якій параметри обладнання повинні максимально відповідати антропометричним, фізіологічним та психофізіологічним характеристикам людського організму. Порушення цих пропорцій призводить до швидкої втоми, виникнення стійких статичних перенапружень опорно-рухового апарату та зниження концентрації уваги.

Організація такого простору регламентується суворими інженерними та санітарними стандартами, впровадження яких забезпечує комфортні та безпечні умови праці протягом усього робочого дня.

Геометричні параметри елементів робочого місця є базовим чинником зниження статичного навантаження. Висота робочої поверхні столу для дорослого користувача встановлюється фіксованою на рівні 725 міліметрів або конфігурується за допомогою регульованих механізмів у діапазоні від 680 до 800 міліметрів, що дозволяє індивідуально адаптувати меблі під антропометричні дані оператора. Простір для ніг під столом повинен мати глибину на рівні колін не менше 400 міліметрів, а на рівні витягнутих ніг — не менше 700 міліметрів при загальній ширині простору не менше 500 міліметрів, що забезпечує вільну зміну положення тіла. Поверхня робочого столу повинна мати матове або напівматове покриття з коефіцієнтом відбиття світла в межах від 0,3 до 0,5, що повністю унеможливлює виникнення дзеркальних або світлових відблисків, які викликають швидку утому зорових аналізаторів.

Конструкція робочого крісла оператора ПК проектується підйомно-поворотною для підтримки раціональної робочої пози і рівномірного розподілу ваги тіла. Обов'язковим є наявність механізмів регулювання висоти сидіння в

межах від 400 до 550 міліметрів та кута нахилу спинки відносно вертикалі у діапазоні від 90 до 110 градусів. Поверхня сидіння повинна мати ширину і глибину не менше 400 міліметрів із закругленим переднім краєм для запобігання здавлюванню судин підколінних западин. Опорна спинка крісла конструюється з урахуванням анатомічних вигинів хребта і обладнується поперековим валиком, який стабілізує хребетний стовп у фізіологічно правильному положенні, мінімізуючи ризики розвитку остеохондрозу та застійних явищ у малому тазі. Поверхня крісла повинна бути напівм'якою, з нековзним, повітропроникним покриттям, яке легко піддається санітарній обробці.

Взаємне розташування засобів відображення інформації та пристроїв введення даних підпорядковується правилам оптимізації моторного поля і зорового сприйняття. Монітор встановлюється безпосередньо по центру перед оператором на відстані від 600 до 700 міліметрів від очей користувача, при цьому верхня кромка екрана повинна знаходитися на рівні очей або трохи нижче, щоб напрямок погляду був орієнтований зверху вниз під кутом від 15 до 20 градусів. Клавіатура та маніпулятор типу «миша» розміщуються на поверхні столу на відстані від 100 до 300 міліметрів від переднього краю, що створює надійну опору для передпліч та кистей рук і запобігає виникненню тривалого статичного напруження м'язів плечового поясу і розвитку хронічного тунельного синдрому у лучезап'ястковому суглобі.

Нормативно-правове регулювання ергономічних вимог та санітарного контролю робочих місць обчислювальних центрів в Україні спирається на профільне законодавство. Основним актом є Закон України «Про охорону праці», який гарантує працівникам створення безпечних умов на кожному робочому місці. Деталізовані інженерно-гігієнічні параметри щодо геометрії меблів, рівнів освітленості та захисту від випромінювань визначаються Державними санітарними правилами і нормами «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» (ДСанПіН 3.3.2.007-98) [51]. Регламент використання екранних пристроїв та загальні вимоги до проектування робочих зон з урахуванням захисту здоров'я

персоналу підпорядковуються «Вимогам щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», затвердженим Наказом Міністерства соціальної політики України № 61. Суворе впровадження цих нормативних положень дозволяє підтримувати високий інженерний потенціал працівників і мінімізувати вплив негативних техногенних факторів.

4.3 Висновок до четвертого розділу 4

У розділі 4 кваліфікаційної роботи проведено ґрунтовний аналіз питань безпеки життєдіяльності та охорони праці, дотримання яких є невід’ємною умовою як на етапі безпосередньої інженерної розробки програмного комплексу, так і під час його подальшої повсякденної експлуатації кінцевими користувачами. Забезпечення комфортних і безпечних умов праці дозволяє мінімізувати вплив негативних техногенних і виробничих факторів, зберегти здоров’я фахівців та підтримувати високий рівень працездатності.

У межах дослідження організаційно-технічних заходів безпеки детально розглянуто алгоритм надання першої долікарської допомоги при гострих харчових отруєннях та інтоксикаціях, що мають прямий логічний зв'язок із предметною областю розробленої системи. Визначено нормативно-правове регулювання санітарно-епідеміологічного контролю в Україні, яке спирається на базові акти, зокрема Закони України «Про забезпечення санітарного та епідемічного благополуччя населення» та «Про охорону праці». Сформовано чіткий поетапний медико-інженерний регламент детоксикації організму, що охоплює первинне промивання шлунка об'ємом 1–1,5 літра води , нейтралізацію патогенів ентеросорбентами з розрахунку 1 грам на 10 кілограмів маси тіла та відновлення водно-електролітного балансу , що дає змогу запобігти розвитку важких системних ускладнень до прибуття екстреної медичної допомоги.

Аналіз інженерно-ергономічних вимог до організації робочого місця оператора персонального комп'ютера дозволив визначити оптимальну просторову конфігурацію елементів робочої зони як складної біотехнічної

системи. На основі Державних санітарних правил і норм ДСанПіН 3.3.2.007-98 та нормативних вимог Наказу Міністерства соціальної політики України № 61 обґрунтовано геометричні параметри меблів та обладнання, включаючи фіксовану висоту столу на рівні 725 мм , регулювання підйомно-поворотного крісла в межах 400–550 мм , розміщення відеотермінала на відстані 600–700 мм від очей оператора з кутом погляду 15–20 градусів , а також позиціонування пристроїв введення на відстані 100–300 мм від краю столу. Суворе впровадження цих параметрів у поєднанні з раціональним режимом праці й відпочинку та контролем параметрів мікроклімату і освітлення дозволяє повністю усунути перенапруження опорно-рухового апарату, запобігти розвитку хронічного тунельного синдрому і зорової втоми, забезпечуючи високу надійність та безпеку праці користувачів.

ВИСНОВКИ

У кваліфікаційній роботі здійснено успішне розв'язання актуального інженерно-практичного завдання, що полягає в комплексному аналізі предметної області, проєктуванні, програмній реалізації та хмарному розгортанні кросплатформової інформаційної системи «Nutrio». На основі отриманих результатів розробки сформовано такі висновки:

- На основі дослідження фітнес-ринку та компаративного аналізу сервісу «Таблиця калорійності» виявлено ключові інтерфейсні недоліки існуючих рішень та за допомогою методу моделювання архетипів цільової аудиторії обґрунтовано доцільність створення лаконічного кросплатформового вебдодатка без комерційних обмежень.
- Розроблено декомпозиційну схему варіантів використання системи, де всі атомарні операції взаємодії користувача чітко розподілено відповідно до сторінок користувацького інтерфейсу Single Page Application та взаємодії з вторинними акторами.
- Розроблено серверну частину веб-додатку, успішно реалізовано на базі передової програмної платформи ASP.NET Core із суворим дотриманням концепції Чистої архітектури та патерну CQRS за допомогою бібліотеки MediatR, що забезпечило повну ізоляцію доменного ядра від технічних інфраструктурних фреймворків.
- У середовищі СКБД PostgreSQL 18 розроблено реляційну модель бази даних на основі чотирьох доменних сутностей, взаємодію з якими абстраговано через Entity Framework Core та автоматизовано за допомогою механізму Dependency Injection з життєвим циклом Scoped.
- Інтегровано надійну підсистему безстанової автентифікації на основі криптографічних маркерів JWT Bearer і Google OAuth 2.0, захищену від SQL-ін'єкцій параметризацією запитів і наскрізною конвеєрною перевіркою вхідних даних на базі FluentValidation.

- Реалізовано клієнтську частину програмного комплексу побудовано за принципом Single Page Application на базі React 19 та Vite 8 із використанням декларативних компонентів бібліотеки Recharts для наочної інтерактивної візуалізації аналітичних звітів динаміки маси тіла та КБЖВ.
- Виконано автономне функціонування та оркестрацію всіх сервісів автоматизовано за допомогою Docker та docker-compose із зворотним проксіюванням через Nginx, а фінальне розгортання інфраструктури виконано у хмарне середовище Microsoft Azure з виділенням автономних ресурсів App Service та Flexible Server.
- Показано комплексну експлуатацію готового застосунку підтвердила логічну цілісність бізнес-модулів та 100% адаптивність інтерфейсу на мобільних пристроях завдяки методам Flexbox та CSS Grid, а контроль через сервіси W3C підтвердив повну відповідність коду стандартам HTML5/CSS3.
- Визначено параметри ергономічної організації робочого місця оператора ПК згідно з вимогами ДСанПіН 3.3.2.007-98 та Наказу № 61, сформовано медико-інженерний регламент надання долікарської допомоги при отруєннях та за рахунок мінімалістичного дизайну інтерфейсу усунено когнітивний тиск на користувача.

Практичне значення одержаних результатів повністю підтверджується високою технічною зрілістю, відмовоустійкістю та надійним захистом даних створеного програмного продукту, який повністю готовий до реальної промислової експлуатації та подальшого інженерного супроводу.

ПЕРЕЛІК ДЖЕРЕЛ

1. Healthy diet [Електронний ресурс] // World Health Organization. – Режим доступу до ресурсу: https://www.who.int/news-room/fact-sheets/detail/healthy-diet (дата звернення: 25.05.2026).

2. Common web application architectures: Clean Architecture [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures#clean-architecture (дата звернення: 25.05.2026).

3. CQRS pattern [Електронний ресурс] // Microsoft Learn (Azure Architecture Center). – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs (дата звернення: 25.05.2026).

4. Таблиця калорійності: трекер їжі та щоденник харчування [Електронний ресурс] // Офіційний сайт Dine4Fit. – Режим доступу до ресурсу: https://www.tablicakalorijnosti.com.ua (дата звернення: 25.05.2026).

5. User Experience Basics [Електронний ресурс] // Nielsen Norman Group. – Режим доступу до ресурсу: https://www.nngroup.com/articles/definition-user-experience/ (дата звернення: 25.05.2026).

6. Body Weight Planner [Електронний ресурс] // National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK). – Режим доступу до ресурсу: https://www.niddk.nih.gov/bwp (дата звернення: 25.05.2026).

7. ASP.NET Core documentation [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core (дата звернення: 25.05.2026).

8. MediatR [Електронний ресурс] // GitHub repository. – Режим доступу до ресурсу: https://github.com/jbogard/MediatR (дата звернення: 25.05.2026).

9. FluentValidation documentation [Електронний ресурс] // FluentValidation Docs. – Режим доступу до ресурсу: https://docs.fluentvalidation.net (дата звернення: 25.05.2026).

10. EF Core documentation [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/ef/core (дата звернення: 25.05.2026).

11. JWT Bearer authentication in ASP.NET Core [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/configure-jwt-bearer-authentication (дата звернення: 25.05.2026).

12. Google APIs Authentication Client Library for .NET [Електронний ресурс] // Google APIs for .NET. – Режим доступу до ресурсу: https://googleapis.dev/dotnet/Google.Apis.Auth/latest/ (дата звернення: 25.05.2026).

13. React documentation [Електронний ресурс] // React Official Website. – Режим доступу до ресурсу: https://react.dev (дата звернення: 25.05.2026).

14. Vite documentation [Електронний ресурс] // Vite Official Website. – Режим доступу до ресурсу: https://vite.dev/guide/ (дата звернення: 25.05.2026).

15. Recharts documentation [Електронний ресурс] // Recharts Official Docs. – Режим доступу до ресурсу: https://recharts.org/en-US/guide (дата звернення: 25.05.2026).

16. React Router documentation [Електронний ресурс] // React Router Official Docs. – Режим доступу до ресурсу: https://reactrouter.com/home (дата звернення: 25.05.2026).

17. Fetch API [Електронний ресурс] // MDN Web Docs. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (дата звернення: 25.05.2026).

18. PostgreSQL documentation [Електронний ресурс] // PostgreSQL Global Development Group. – Режим доступу до ресурсу: https://www.postgresql.org/docs/ (дата звернення: 25.05.2026).

19. Docker documentation [Електронний ресурс] // Docker Docs. – Режим доступу до ресурсу: https://docs.docker.com (дата звернення: 25.05.2026).

20. Docker Compose documentation [Електронний ресурс] // Docker Docs. – Режим доступу до ресурсу: https://docs.docker.com/compose/ (дата звернення: 25.05.2026).

21. Microsoft Azure documentation [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure (дата звернення: 25.05.2026).

22. What is CI/CD? [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/devops/what-is-devops#what-is-cicd (дата звернення: 25.05.2026).

23. Apply CQRS and CQS approaches in a .NET microservice [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/apply-cqrs-command-query-responsibility-segregation-approaches (дата звернення: 25.05.2026).

24. OpenAPI Specification [Електронний ресурс] // OpenAPI Initiative. – Режим доступу до ресурсу: https://www.openapis.org (дата звернення: 25.05.2026).

25. Scalar Documentation [Електронний ресурс] // Scalar Docs. – Режим доступу до ресурсу: https://docs.scalar.com (дата звернення: 25.05.2026).

26. Entity Framework Core Documentation [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 25.05.2026).

27. Open Food Facts API Documentation [Електронний ресурс] // Open Food Facts. – Режим доступу до ресурсу: https://openfoodfacts.github.io/api-documentation/ (дата звернення: 25.05.2026).

28. EF Core Migrations Overview [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/ (дата звернення: 25.05.2026).

29. Hash passwords in ASP.NET Core [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-

us/aspnet/core/security/data-protection/consumer-apis/password-hashing (дата звернення: 25.05.2026).

30. Configure JWT bearer authentication in ASP.NET Core [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/configure-jwt-bearer-authentication (дата звернення: 25.05.2026).

31. TokenValidationParameters Class [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/api/microsoft.identitymodel.tokens.tokenvalidationparameters (дата звернення: 25.05.2026).

32. Google APIs Authentication Client Library for .NET [Електронний ресурс] // Google APIs for .NET. – Режим доступу до ресурсу: https://googleapis.dev/dotnet/Google.Apis.Auth/latest/ (дата звернення: 25.05.2026).

33. Passing Data Deeply with Context [Електронний ресурс] // React Docs. – Режим доступу до ресурсу: https://react.dev/learn/passing-data-deeply-with-context (дата звернення: 25.05.2026).

34. Fetch API [Електронний ресурс] // MDN Web Docs. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (дата звернення: 25.05.2026).

35. Web Storage API [Електронний ресурс] // MDN Web Docs. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API (дата звернення: 25.05.2026).

36. Multi-stage builds [Електронний ресурс] // Docker Docs. – Режим доступу до ресурсу: https://docs.docker.com/build/building/multi-stage/ (дата звернення: 25.05.2026).

37. Docker Compose overview [Електронний ресурс] // Docker Docs. – Режим доступу до ресурсу: https://docs.docker.com/compose/ (дата звернення: 25.05.2026).

38. Azure Database for PostgreSQL – Flexible Server [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/postgresql/flexible-server/ (дата звернення: 25.05.2026).

39. xUnit.net [Електронний ресурс] // xUnit Official Website. – Режим доступу до ресурсу: https://xunit.net (дата звернення: 25.05.2026).

40. Integration tests in ASP.NET Core [Електронний ресурс] // Microsoft Learn. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/test/integration-tests (дата звернення: 25.05.2026).

41. Testcontainers for .NET [Електронний ресурс] // Testcontainers Documentation. – Режим доступу до ресурсу: https://dotnet.testcontainers.org (дата звернення: 25.05.2026).

42. Vitest [Електронний ресурс] // Vitest Official Website. – Режим доступу до ресурсу: https://vitest.dev (дата звернення: 25.05.2026).

43. Google Identity Services for Web [Електронний ресурс] // Google Developers. – Режим доступу до ресурсу: https://developers.google.com/identity/gsi/web (дата звернення: 25.05.2026).

44. Client-side form validation [Електронний ресурс] // MDN Web Docs. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Forms/Form_validation (дата звернення: 25.05.2026).

45. Responsive design [Електронний ресурс] // MDN Web Docs. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Responsive_Design (дата звернення: 25.05.2026).

46. Basic concepts of flexbox [Електронний ресурс] // MDN Web Docs. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox (дата звернення: 25.05.2026).

47. Web Content Accessibility Guidelines (WCAG) 2.2 [Електронний ресурс] // W3C. – Режим доступу до ресурсу: https://www.w3.org/TR/WCAG22/

(дата звернеро забезпечення санітарного та епідемічного благополуччя населення: Закон України [Електронний ресурс] // Офіційний вебпортал парламенту України. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/4004-12 (дата звернення: 25.05.2026).

48. Про охорону праці: Закон України [Електронний ресурс] // Офіційний вебпортал парламенту України. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 25.05.2026).

49. Про затвердження Державних санітарних норм та правил «Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу»: Наказ Міністерства охорони здоров'я України № 248 [Електронний ресурс] // Офіційний вебпортал парламенту України. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/go/z0472-14 (дата звернення: 25.05.2026).

50. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [Електронний ресурс] // Офіційний вебпортал парламенту України. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення: 25.05.2026).

ДОДАТКИ

Додаток А

Лістинг точки входу та конфігурації API “Program.cs”

```
using System.Text;
using FluentValidation;
using MediatR;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.HttpOverrides;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi;
using Nutrio.Application.Behaviors;
using Nutrio.Application.Commands.Auth.Register;
using Nutrio.Infrastructure;
using Nutrio.Infrastructure.Persistence;
using Nutrio.Middleware;
using Scalar.AspNetCore;

namespace Nutrio;

public class Program
{
    public static async Task Main(string[] args)
    {
        var builder = WebApplication.CreateBuilder(args);

        builder.Services.AddInfrastructure(builder.Configuration);
        builder.Services.AddControllers();

        builder.Services.AddAuthentication(JwtBearerDefaults.Authe
nticationScheme)
            .AddJwtBearer(options =>
            {
                options.TokenValidationParameters = new
TokenValidationParameters
                {
                    ValidateIssuer = true,
                    ValidateAudience = true,
                    ValidateLifetime = true,
                    ValidateIssuerSigningKey = true,
                    ValidIssuer =
builder.Configuration["JwtSettings:Issuer"],
                    ValidAudience =
builder.Configuration["JwtSettings:Audience"],
                    IssuerSigningKey = new SymmetricSecurityKey(
                        Encoding.UTF8.GetBytes(builder.Configurati
on["JwtSettings:Secret"]!)),
                };
            });
```

```
        builder.Services.AddAuthorization();

        builder.Services.AddCors(options =>
        {
            options.AddDefaultPolicy(policy =>
            {
                if (builder.Environment.IsDevelopment() ||
builder.Environment.IsEnvironment("Testing"))
                {
                    // Vite може зайняти 5173/5174/5175… —
дозволяємо будь-який localhost (HTTP/HTTPS).
                    policy.SetIsOriginAllowed(static origin =>
                    {
                        if (string.IsNullOrEmpty(origin)) return
false;
                        if (!Uri.TryCreate(origin,
UriKind.Absolute, out var uri)) return false;
                        return uri.IsLoopback
                               && (uri.Scheme == Uri.UriSchemeHttp
|| uri.Scheme == Uri.UriSchemeHttps);
                    });
                }
                else
                {
                    policy.WithOrigins(
                        "http://localhost:5173",
                        "https://localhost:5173",
                        "https://nutrio-lv-
erajenbjcfb9ewaf.polandcentral-01.azurewebsites.net");
                }

                policy.AllowAnyHeader().AllowAnyMethod();
            });
        });

        builder.Services.AddOpenApi(options =>
        {
            options.AddDocumentTransformer((document, context,
cancellationToken) =>
            {
                document.Components ??= new OpenApiComponents();
                document.Components.SecuritySchemes ??= new
Dictionary<string, IOpenApiSecurityScheme>();

                document.Components.SecuritySchemes.Add("Bearer",
new OpenApiSecurityScheme
                {
                    Type = SecuritySchemeType.Http,
                    Scheme = "bearer",
                    BearerFormat = "JWT",
                    Description = "Введіть свій JWT токен сюди
(слово Bearer писати НЕ треба)",
```

```
                });

                document.Security ??= new
List<OpenApiSecurityRequirement>();
                document.Security.Add(new
OpenApiSecurityRequirement
                {
                    [new OpenApiSecuritySchemeReference("Bearer",
document)] = [],
                });

                return Task.CompletedTask;
            });
        });

        builder.Services.AddValidatorsFromAssemblyContaining<Regis
terUserCommandValidator>();
        builder.Services.AddTransient(typeof(IPipelineBehavior<,>)
, typeof(ValidationBehavior<,>));

        builder.Services.AddMediatR(cfg =>
            cfg.RegisterServicesFromAssembly(typeof(RegisterUserCo
mmand).Assembly));

        builder.Services.Configure<ForwardedHeadersOptions>(option
s =>
        {
            options.ForwardedHeaders =
ForwardedHeaders.XForwardedFor | ForwardedHeaders.XForwardedProto;
            options.KnownNetworks.Clear();
            options.KnownProxies.Clear();
        });

        var app = builder.Build();

        app.UseForwardedHeaders();

        var isDevOrTest = app.Environment.IsDevelopment() ||
app.Environment.IsEnvironment("Testing");
        if (isDevOrTest)
        {
            app.MapOpenApi();
            app.MapScalarApiReference(options =>
            {
                options.WithTitle("Nutrio API")
                    .WithTheme(ScalarTheme.DeepSpace)
                    .WithDefaultHttpClient(ScalarTarget.JavaScript
, ScalarClient.Axios);
            });

            await using var scope =
app.Services.CreateAsyncScope();
```

```
            var db =
scope.ServiceProvider.GetRequiredService<NutrioDbContext>();
            var loggerFactory =
scope.ServiceProvider.GetRequiredService<ILoggerFactory>();
            var seederLogger =
loggerFactory.CreateLogger("DevelopmentDataSeeder");
            await db.Database.MigrateAsync();
            await
DevelopmentDataSeeder.SeedProductsIfEmptyAsync(db, seederLogger);
        }

        if (!isDevOrTest)
            app.UseHttpsRedirection();

        app.UseCors();
        app.UseMiddleware<ExceptionHandlingMiddleware>();

        app.UseAuthentication();
        app.UseAuthorization();

        app.MapGet("/health", () => Results.Ok());
        app.MapGet("/", () => Results.Text("Nutrio API",
"text/plain"));

        app.MapControllers();

        await app.RunAsync();
    }
}
```

Додаток Б

Лістинг реєстрації інфраструктурного шару «DependencyInjection.cs»

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Diagnostics;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Nutrio.Application.Commands.Auth.Google;
using Nutrio.Application.Interfaces;
using Nutrio.Domain.Interfaces;
using Nutrio.Infrastructure.Authentication;
using Nutrio.Infrastructure.Persistence;
using Nutrio.Infrastructure.Persistence.Repositories;
using Nutrio.Infrastructure.Security;

namespace Nutrio.Infrastructure;
public static class DependencyInjection
{
    public static IServiceCollection AddInfrastructure(
        this IServiceCollection services,
        IConfiguration configuration)
    {
        services.AddDbContext<NutrioDbContext>(options =>
        {if (string.Equals(configuration["UseInMemoryDatabase"],
"true", StringComparison.OrdinalIgnoreCase))
       { var name = configuration["InMemoryDatabaseName"] ??
"NutrioTests";
                options.UseInMemoryDatabase(name);
            }
            else
            {
                options.UseNpgsql(configuration.GetConnectionStrin
g("DefaultConnection"));
                options.ConfigureWarnings(w =>
w.Ignore(RelationalEventId.PendingModelChangesWarning));}});

        services.AddScoped<IUserRepository, UserRepository>();
        services.AddScoped<IProductRepository,
ProductRepository>();
        services.AddScoped<IFoodEntryRepository,
FoodEntryRepository>();
        services.AddScoped<IBodyMetricRepository,
BodyMetricRepository>();
        services.AddSingleton<IPasswordHasher, PasswordHasher>();
        services.AddScoped<IJwtTokenGenerator,
JwtTokenGenerator>(); /
        services.AddScoped<IGoogleTokenValidator,
GoogleTokenValidator>();
        return services; }}
```

Додаток В

Лістинг контексту бази даних і сутностей реаліаційної моделі «NutrioDbContext.cs»

```
using Microsoft.EntityFrameworkCore;
using Nutrio.Domain.Entities;
using System.Reflection;

namespace Nutrio.Infrastructure.Persistence;

public class NutrioDbContext : DbContext
{
    public NutrioDbContext(DbContextOptions<NutrioDbContext>
options) : base(options)
    {
    }

    public DbSet<User> Users { get; set; }
    public DbSet<Product> Products { get; set; }
    public DbSet<FoodEntry> FoodEntries { get; set; }
    public DbSet<BodyMetricStamp> BodyMetricStamps { get; set; }

    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
        modelBuilder.ApplyConfigurationsFromAssembly(Assembly.GetE
xecutingAssembly());
    }
}
```

Додаток Г

Лістинг клієнтської маршрутизації «App.jsx»

```
import { BrowserRouter, Navigate, Route, Routes } from 'react-
router-dom'
import { ProtectedRoute } from './components/ProtectedRoute.jsx'
import { AddProductModal } from
'./components/modals/AddProductModal.jsx'
import { AddProductModalProvider } from
'./context/AddProductModalProvider.jsx'
import { useAddProductModal } from './hooks/useAddProductModal.js'
import { AuthProvider } from './context/AuthContext.jsx'
import { JournalDateProvider } from
'./context/JournalDateContext.jsx'
import { AppShell } from './layouts/AppShell.jsx'
import AuthPage from './pages/AuthPage.jsx'
import OnboardingPage from './pages/OnboardingPage.jsx'
import AnalyticsPage from './pages/AnalyticsPage.jsx'
import JournalPage from './pages/JournalPage.jsx'
import UserPage from './pages/UserPage.jsx'

function AddProductModalKeyed() {
  const { modalKey } = useAddProductModal()
  return <AddProductModal key={modalKey} />
}

export default function App() {
  return (
    <BrowserRouter>
      <AuthProvider>
        <Routes>
          <Route path="/" element={<Navigate to="/auth" replace
/>} />
          <Route path="/auth" element={<AuthPage />} />
          <Route
            path="/onboarding"
            element={
              <ProtectedRoute>
                <OnboardingPage />
              </ProtectedRoute>
            }
          />
          <Route
            path="/app"
            element={
              <ProtectedRoute>
                <AddProductModalProvider>
                  <JournalDateProvider>
                    <AppShell />
```

```
                    <AddProductModalKeyed />
                  </JournalDateProvider>
                </AddProductModalProvider>
              </ProtectedRoute>
            }
          >
            <Route index element={<Navigate to="journal" replace
/>} />
            <Route path="journal" element={<JournalPage />} />
            <Route path="analytics" element={<AnalyticsPage />} />
            <Route path="user" element={<UserPage />} />
          </Route>
          <Route path="*" element={<Navigate to="/auth" replace
/>} />
        </Routes>
      </AuthProvider>
    </BrowserRouter>
  )
}
```

Додаток Г

Лістинг файлу оркестрації розгортання контенеризації через Docker «docker-compose.yml»

```
name: nutrio-app

services:
  db:
    image: postgres:18-alpine
    container_name: Nutrio-app-db
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: "****"
      POSTGRES_DB: NutrioDb
    ports:
      - "5433:5432"
    volumes:
      - nutrio_pgdata:/var/lib/postgresql
      - ./docker/postgres/init:/docker-entrypoint-initdb.d:ro
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres -d NutrioDb"]
      interval: 5s
      timeout: 5s
      retries: 30
      start_period: 120s

  api:
    image: odintsovk/nutrio-app:latest
    build:
      context: ./Nutrio_Backend
      dockerfile: Dockerfile
    container_name: Nutrio-app
    ports:
      - "5250:8080"
    environment:
      ASPNETCORE_URLS: http://+:8080
      ASPNETCORE_ENVIRONMENT: Development
      ConnectionStrings__DefaultConnection:
"Host=db;Port=5432;Database=NutrioDb;Username=postgres;Password=12
34"
      JwtSettings__Secret:
"NutrioSuperSecretKeyThatIsAtLeast32CharactersLong2024!"
      JwtSettings__Issuer: NutrioBackend
      JwtSettings__Audience: NutrioFrontend
      JwtSettings__ExpiryMinutes: "60"
      GoogleAuth__ClientId: ${GOOGLE_CLIENT_ID:- **** }
    depends_on:
      db:
```

```
        condition: service_healthy

  web:
    image: odintsovk/nutrio-app-web:latest
    build:
      context: ./Nutrio_Frontend/Nutrio_Frontend
      dockerfile: Dockerfile
      args:
        VITE_API_BASE_URL: ${VITE_API_BASE_URL:-
http://localhost:5250}
        VITE_GOOGLE_CLIENT_ID: ${GOOGLE_CLIENT_ID:-**** }
    container_name: Nutrio-app-web
    ports:
      - "9080:80"
    depends_on:
      - api

volumes:
  nutrio_pgdata:
```