Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мультимедійних систем

Курсова робота

освітній ступінь – бакалавр

на тему: «**РОЗРОБКА API ДЛЯ СИСТЕМИ ПЛАНУВАННЯ ХАРЧУВАННЯ ТА ЗАКУПІВЕЛЬ З ПЕРЕВІРКОЮ ВІДПОВІДНОСТІ ДІЄТИЧНИМ НОРМАМ**»

Виконав: студент 4-го року навчання,

Освітньої програми «Інженерія програмного забезпечення», 121

Єгоров Ігор Вікторович

Керівник Федючено М.І.,_____

ст. викл.

Рецензент _____

(прізвище та ініціали)

Кваліфікаційна робота захищена

з оцінкою _____

Секретар ЕК _____

«__9_» ____травня_____ 2025___ р.

Київ – 2025

ЗМІСТ

ВСТУП

Планування харчування є актуальною проблемою для багатьох людей, оскільки безпосередньо впливає на якість життя, самопочуття та загальний рівень здоров'я. У повсякденному ритмі сучасного життя люди часто стикаються з браком часу на приготування їжі та ухвалення рішень щодо щоденного раціону. Зі зростанням інтересу до здорового способу життя, збалансованого харчування та підвищеної обізнаності про вплив харчових звичок на організм, зростає попит на інструменти, які допомагають людям упорядкувати свій раціон, дотримуючись водночас індивідуальних потреб та науково обґрунтованих норм харчування.

Хоча на ринку існує багато рішень, серед яких онлайн-сервіси, мобільні додатки та дієтичні платформи, більшість із них мають певні недоліки або не адаптовані до потреб середньостатистичного користувача. Найбільш пересічною проблемою для багатьох додатків є некоректна агрегація продуктів при генерації списку покупок. Також бракує цілісних рішень, які б поєднували всі необхідні аспекти — управління рецептами, формування меню на день або тиждень, перевірку відповідності добовим нормам споживання нутрієнтів, а також створення повного та впорядкованого списку необхідних продуктів для закупівлі. Це створює потребу у комплексному рішенні, яке буде простим у використанні, інтуїтивно зрозумілим та адаптивним до особистих вподобань користувача.

Метою даної роботи є розробка зручного та практичного прикладного програмного інтерфейсу (API), який спрощує процес щоденного планування харчування, дозволяє легко контролювати дотримання рекомендованих норм

харчування і автоматично формує списки покупок на основі обраного меню. Такий API не надає кінцевому користувачу візуального інтерфейсу, однак є відкритим до розширення, легко інтегрується з іншими програмними рішеннями і може використовуватися у сторонніх або опен-сорс розробках. Він здатен стати основою для створення різних клієнтських застосунків — мобільних, веб або десктопних — що надає йому універсальності та гнучкості.

Для досягнення поставленої мети були визначені наступні завдання:

1. Провести аналіз сучасних цифрових інструментів для планування харчування, оцінити їх переваги та недоліки, а також визначити функціональні прогалини.
2. Розробити концепцію API, який поєднує можливості управління рецептами та формування списків покупок, без автоматизованого складання раціону, але з перевіркою його відповідності добовим нормам.
3. Реалізувати механізм перевірки плану харчування на відповідність добовим нормам споживання калорій, білків, жирів, вуглеводів та інших життєво важливих компонентів.
4. Інтегрувати систему генерації списків продуктів на основі складання меню з урахуванням об’єднання однакових інгредієнтів.
5. Провести тестування розробленого API на різних сценаріях використання для оцінки його зручності, ефективності та практичного застосування у повсякденному житті.

Запропоноване рішення дозволить суттєво полегшити завдання формування повноцінного та збалансованого меню для окремих осіб або цілих родин. API

забезпечить чітку відповідність раціону встановленим нормам харчування, зменшить час на планування меню та складання списків покупок, а також допоможе уникнути харчових дисбалансів або нераціональних витрат. Його відкритість та модульність дозволяє залучати зовнішніх розробників до створення нових рішень на основі API, що сприятиме формуванню усвідомлених харчових звичок, покращенню якості життя та підвищенню ефективності щоденного побуту користувачів.

Робота складається з трьох основних розділів:

1. Перший розділ присвячений загальному огляду предметної області з акцентом на дослідження актуальних цифрових рішень у сфері планування харчування. У ньому коротко розглядаються можливості, які надають сучасні мобільні додатки та сервіси для організації раціону, а також визначаються ключові функціональні особливості, що можуть бути враховані при розробці власного рішення. Окрему увагу приділено огляду наукових публікацій, які можуть бути використані як база для формування системи обрахунку добових норм споживання. Аналіз допоможе виділити головні переваги та недоліки існуючих підходів і визначити напрямки удосконалення для майбутньої реалізації.
2. Другий розділ містить опис архітектури програмного продукту, розподіл функціональних модулів, механізмів взаємодії між ними, а також обґрунтування вибору технологій, мов програмування та підходів до розробки.
3. У третьому розділі викладено етапи реалізації API, детально розглянуто алгоритми перевірки добових норм, логіку формування

списків покупок, а також подано результати тестування, приклади використання API та аналітичні висновки щодо переваг і можливих напрямів подальшого вдосконалення системи.

Розроблений API орієнтований на широке коло користувачів — від студентів і зайнятих офісних працівників до молодих батьків і людей літнього віку. Завдяки відкритому коду та розширеній функціональності, він може стати основою для створення клієнтських інтерфейсів, бути інтегрованим у вже існуючі рішення та широко використовуватись у практиці планування харчування.

РОЗДІЛ І

1.1. Проблематика планування харчування

Раціональне харчування є основою здорового способу життя. За даними Всесвітньої організації охорони здоров'я, спосіб життя людини визначає до 70% її здоров'я, при цьому харчування є одним з головних факторів впливу [1]. Воно безпосередньо впливає на рівень енергії, працездатність, профілактику хвороб та загальний стан організму.

У сучасному суспільстві харчові звички часто формуються під впливом зовнішніх факторів: реклами, соціальних мереж, ритму життя. Широке розповсюдження фаст-фуду, низький рівень фізичної активності та хронічний стрес призводять до незбалансованого харчування, що може спричинити ожиріння, серцево-судинні захворювання, діабет та інші хронічні хвороби [2].

Попри зростаючу увагу до здорового способу життя, багато людей стикаються з труднощами при самостійному плануванні раціону. Серед основних проблем можна виділити:

- недостатній рівень знань щодо калорійності, нутрієнтів, макро- і мікроелементів [3];
- складність ручного обліку енергетичної цінності продуктів і дотримання балансу БЖВ (білки, жири, вуглеводи);
- відсутність адаптації до індивідуальних особливостей (алергії, хронічні хвороби, цілі);
- нестача мотивації та зовнішнього контролю;

- недоступність або складність існуючих інструментів.

Цифрові інструменти, зокрема мобільні додатки, можуть значно спростити процес планування раціону, проте більшість з них або надто спрощені, або складні у використанні. Водночас існує потреба у функціональному рішенні, яке б дозволяло не лише планувати раціон, але й:

- автоматично формувати списки покупок на основі рецептів;
- адаптувати кількість інгредієнтів відповідно до бажаної кількості порцій;
- синхронізувати план приготування з календарем;
- здійснювати підрахунок калорійності та БЖВ;
- сумувати однакові інгредієнти з різних страв;
- зручно змінювати планування під індивідуальні цілі користувача.

1.2. Огляд існуючих цифрових рішень

Ринок цифрових рішень для планування харчування стрімко розвивається. Користувачі мають доступ до десятків застосунків, які обіцяють покращити якість харчування, полегшити приготування страв і зменшити час на планування покупок. Проте функціональність і зручність таких сервісів суттєво відрізняються.

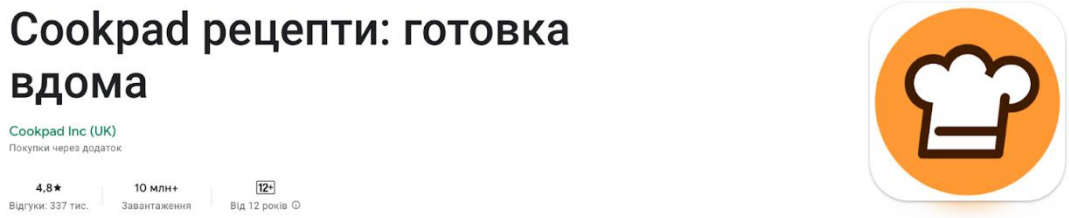
У цьому підрозділі здійснено аналіз найпоширеніших застосунків, доступних у App Store та Google Play, за рядом ключових критеріїв, серед яких:

- наявність вбудованої бази рецептів;

- можливість створювати власні рецепти;
- наявність планера харчування;
- гнучкість вибору порцій;
- підтримка сумування інгредієнтів у спільний список;
- відображення поживної цінності страв (калорій, білків, жирів, вуглеводів);
- генерація списку покупок.

Аналіз проводився з урахуванням як безкоштовного, так і платного функціоналу додатків, а також на основі фактичної перевірки роботи застосунків.

Нижче подано опис кожного з проаналізованих інструментів, а також узагальнена таблиця для швидкого зіставлення функціональних можливостей.



Доступний у App Store та Google Play.

Безкоштовна версія:

- велика база рецептів з детальними покроковими діями
- можливість групувати та організовувати тематичні колекції рецептів, створювати приватні папки за категоріями

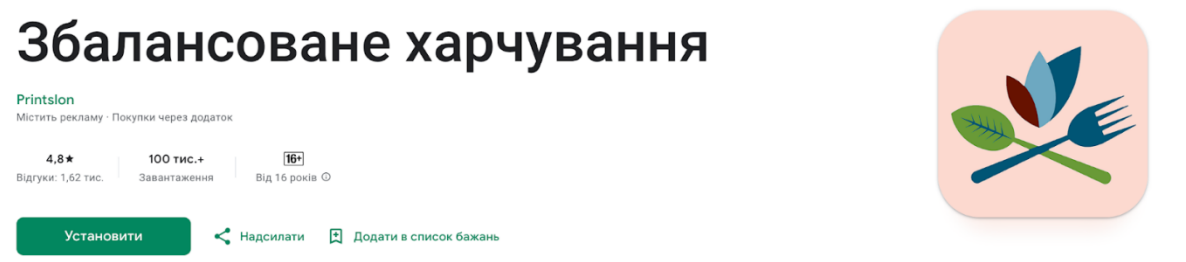
- публікація в мережі Cookpad власних рецептів (як приватно, так і публічно) з можливістю коментувати дописи користувачів
- можливість зберегти рецепт у вкладку “Готую сьогодні” та сформувати списки інгредієнтів для відповідних страв

Платна версія - 1,99 USD/місяць:

- припинення транслювання реклами
- поява фільтрів для зручною навігації (пошук за популярністю, інгредієнтами, дієтами, способами приготування)
- Premium плани для харчування
- Топ рецепти з фотовідгуками та переглядами

Недоліки:

- не групує усі інгредієнти в один список, що ускладнює купування продуктів якщо один і той же товар буде у 2-х списках
- не вказується кількість жирів, білків, вуглеводів та калорійність продуктів
- відсутній вибір кількості порцій



Доступний у App Store та Google Play.

Безкоштовна версія:

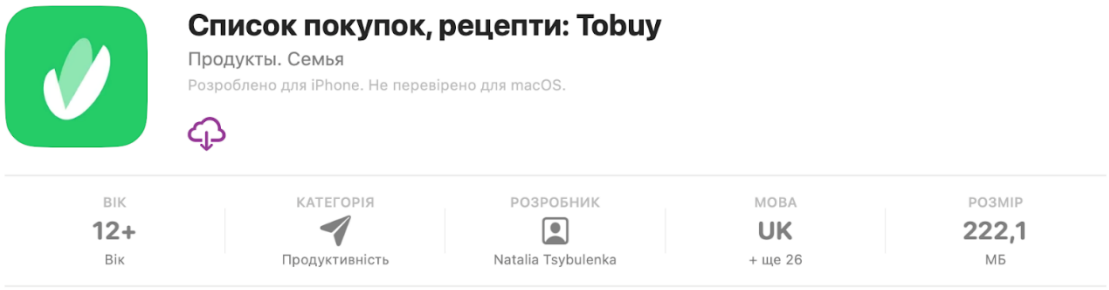
- велика база рецептів з детальними покроковими діями
- зручна фільтрація та навігація, можливість вибрати дієту або кухню
- інгредієнти, що можна додавати по одному з рецепту, так і усі разом до Списку покупок
- рецепти, інгредієнти яких повторюються сумуються у фінальному Списку покупок
- до кожного рецепту додається кількість жирів, білків, вуглеводів та калорійність продуктів

Платна версія - 5,99 USD - одноразова покупка:

- припинення транслювання реклами

Недоліки:

- відсутній вибір кількості порцій
- у безкоштовній версії реклама відпрацьовує на кожну третю ключову дію у додатку



Доступний лише у App Store.

Безкоштовна версія:

- можливість додавання власних рецептів з обмеженням у кількості (до 20 рецептів)
- можливість групувати та організовувати тематичні списки рецептів
- інгредієнти, що можна додавати по одному з рецепту, так і усі разом до Списку покупок
- рецепти, інгредієнти яких повторюються сумуються у фінальному Списку покупок
- функція перегляду та редагування покупок за допомогою Apple Watch
- можливість поділитися списком з користувачами та бачити їх динаміку покупок

Платна версія - 6,99 USD - покупка на рік:

- велика база рецептів з детальними покроковими діями
- додавання фото до власних продуктів
- необмежені списки шаблонів
- припинення транслювання реклами

Недоліки:

- відсутня база рецептів у безкоштовній версії (як приклад додано дві страви)
- не вказується кількість жирів, білків, вуглеводів та калорійність продуктів
- відсутній вибір кількості порцій

Доступний лише у App Store.

Безкоштовна версія:

- велика база рецептів з детальними покроковими діями
- можливість групувати та організовувати тематичні списки рецептів – Плани їжі на тиждень
- кожен інгредієнт, слід додавати окремо, або додавати увесь рецепт

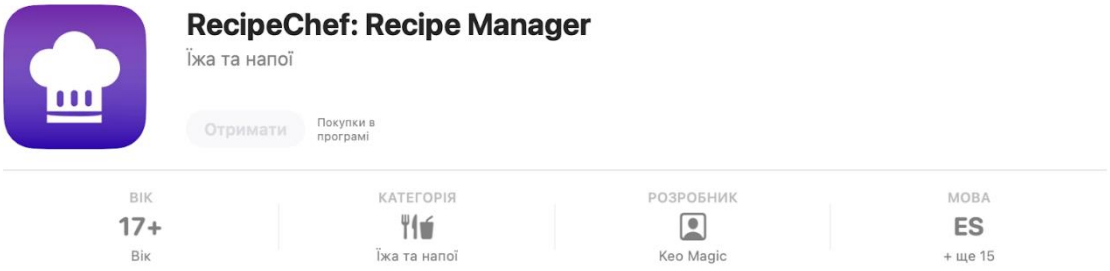
Платна версія - 2,99 USD USD/місяць:

- додаються ексклюзивні рецепти та набори меню
- можливість додавати власні рецепти

- до кожного рецепту додається кількість жирів, білків, вуглеводів та калорійність продуктів
- додані розширені фільтри для пошуку рецептів
- додана історія харчувальних планів

Недоліки:

- доступна лише на англійській, відсутність перекладів на інші мови
- рецепти, інгредієнти яких повторюються не сумуються у фінальному Списку покупок
- відсутній вибір кількості порцій



Доступний лише у App Store.

Безкоштовна версія:

- можливість додавання власних рецептів з обмеженням у кількості (до 20 рецептів)
- можливість групувати та організовувати тематичні списки рецептів, планувати покупки по дням тижня
- кожен інгредієнт, слід додавати окремо, або додавати увесь рецепт

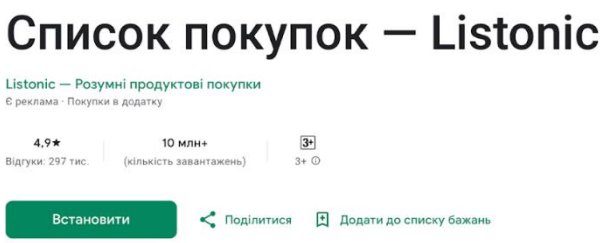
- можливість сканування тексту рецепту та додавання його до бази рецептів
- можливість змінити кількість порцій

Платна версія - 0,99 USD USD/місяць:

- можливість додавати власні рецепти до 10 000 штук
- можливість використання одного акаунту на різних девайсах без обмежень
- онлайн бекап

Недоліки:

- доступна лише на англійській, відсутність перекладів на інші мови
- рецепти, інгредієнти яких повторюються не сумуються у фінальному списку покупок
- не вказується кількість жирів, білків, вуглеводів та калорійність продуктів



Доступний у App Store та Google Play.

Безкоштовна версія:

- можливість групувати та організовувати тематичні списки товарів, створювати приватні папки за категоріями

Платна версія - 0,99 USD/місяць:

- припинення транслювання реклами
- поява фільтрів для зручною навігації (пошук за популярністю, інгредієнтами, дієтами, способами приготування)

Недоліки:

- відсутня база рецептів
- не вказується кількість жирів, білків, вуглеводів та калорійність продуктів
- відсутній вибір кількості порцій

Додаток	Вимоги до додатку							
	1		2		3	4	5	6	7
	Б	П	Б	П					
Cookpad	Так		Так		Ні	Ні	Ні	Ні	Так
Збалансоване харчування	Так		Ні		Так	Ні	Так	Так	Так
Tobuy	Ні	Так	Так		Ні	Ні	Так	Ні	Так
Mealime	Так		Ні	Так	Ні	Ні	Ні	Так	Так
RecipeChef	Так		Так		Ні	Так	Ні	Ні	Так
Listonic	Ні		Так		Ні	Ні	Так	Ні	Так

Таблиця 1. Порівняльний аналіз функціоналу додатків

1. Доступ до вбудованої бази рецептів (безкоштовна (Б) та платна (П) версії)
2. Можливість створювати власні рецепти
3. Планер харчування: планування приготування + харчування (сніданок, обід, вечеря ітд)
4. Вибір кількості порцій при додаванні рецепту у список покупок
5. Групування та сумування товарів з різних рецептів в один список
6. Підрахунок калорій, жирів, білків, вуглеводів
7. Генерація списку покупок

Проведений аналіз виявив, що жоден із розглянутих застосунків не забезпечує повного циклу планування харчування — від вибору рецепта до формування списку покупок з урахуванням кількості порцій, підрахунку поживної цінності та календарного планування. Це свідчить про наявність ніші для розробки інструменту, який би поєднував ці функції в одному рішенні.

Наукові дослідження підтверджують ефективність цифрових інструментів у підтримці здорового харчування. Зокрема, огляд, опублікований у журналі Nutrition Reviews, зазначає, що цифрові додатки можуть сприяти покращенню харчової поведінки та контролю ваги, особливо коли вони персоналізовані та інтегровані в повсякденне життя користувача [4].

Інше дослідження, опубліковане в Public Health Nutrition, підкреслює важливість функціоналу планування харчування та створення списків покупок у мобільних додатках для підтримки здорового харчування серед батьків з низьким доходом [5].

Ці висновки підкреслюють необхідність розробки комплексного цифрового інструменту, який би враховував індивідуальні потреби користувачів та забезпечував повний цикл планування харчування.

1.3. Теоретичні засади обрахунку добових норм харчування

Підрахунок калорійності та вмісту макронутрієнтів (білків, жирів, вуглеводів) є ключовим для формування збалансованого раціону. Це дозволяє користувачам контролювати енергетичний баланс, досягати цілей щодо ваги та підтримувати здоров'я.

Згідно з Dietary Guidelines for Americans 2020–2025, рекомендовані пропорції макронутрієнтів у добовому раціоні становлять:

- Вуглеводи: 45–65% від загальної калорійності
- Жири: 20–35%
- Білки: 10–35%

Ці пропорції можуть бути адаптовані залежно від індивідуальних потреб користувача, таких як рівень фізичної активності, цілі (схуднення, набір м'язової маси) та особливості здоров'я.

Для визначення добових потреб у макронутрієнтах необхідно враховувати:

- Загальну калорійність раціону: визначається на основі базального метаболізму та рівня фізичної активності.
- Бажані пропорції макронутрієнтів: встановлюються відповідно до індивідуальних цілей користувача.

Наприклад, для раціону в 2000 ккал з розподілом 50% вуглеводів, 30% жирів і 20% білків:

- Вуглеводи: 1000 ккал / 4 ккал/г = 250 г
- Жири: 600 ккал / 9 ккал/г ≈ 67 г
- Білки: 400 ккал / 4 ккал/г = 100 г

Ці розрахунки можуть бути автоматизовані в додатку для надання користувачам персоналізованих рекомендацій щодо споживання макронутрієнтів.

При розробці додатку важливо враховувати індивідуальні особливості користувачів, такі як:

- Рівень фізичної активності: впливає на загальні енергетичні потреби.
- Мета користувача: схуднення, підтримка ваги або набір м'язової маси.
- Особливості здоров'я: наявність хронічних захворювань, алергій тощо.

Це дозволить забезпечити більш точні та релевантні рекомендації щодо харчування.

1.4. Висновки до розділу

Проведене дослідження підтверджує актуальність тематики раціонального харчування в умовах сучасного способу життя та необхідність використання цифрових інструментів для його підтримки. Аналіз існуючих застосунків показав, що хоча багато з них вирішують окремі задачі (наприклад,

формування списку покупок або збереження рецептів), жоден не забезпечує повноцінного комплексного підходу.

Окремо було розглянуто теоретичні засади обрахунку добових потреб у поживних речовинах. Наукові джерела підтверджують доцільність використання підрахунку калорій, білків, жирів і вуглеводів як дієвого інструменту у формуванні збалансованого раціону. Визначено рекомендовані пропорції макронутрієнтів, наведено приклади розрахунків добової потреби, які можуть бути реалізовані в програмному забезпеченні.

Здобуті висновки створюють підґрунтя для формування функціональних вимог до системи. Особливу увагу буде приділено реалізації механізмів автоматичного обчислення поживної цінності страв, адаптації під індивідуальні цілі користувача, а також інтеграції з логікою календарного планування й формування списку покупок.

РОЗДІЛ ІІ

2.1. Цільова аудиторія та сценарії використання

Розроблюване API орієнтоване на широке коло розробників, команд і ентузіастів, які прагнуть створювати або інтегрувати рішення у сфері планування харчування без необхідності самостійно реалізовувати логіку перевірки нутрієнтів, роботи з рецептами та генерації списків продуктів. Основною метою є надати універсальний програмний інтерфейс, що дозволяє інтегрувати можливості планування раціону у будь-який клієнтський застосунок – веб, мобільний або десктопний. API має бути зрозумілим навіть для тих розробників, які не мають глибоких знань у дієтології, та забезпечити повноцінний інструментарій для побудови рішень, орієнтованих на користувача.

Серед основних типів користувачів API можна виділити:

- Розробники мобільних застосунків, які прагнуть швидко інтегрувати перевірку добових норм, формування меню та списків покупок без створення складної бекенд-архітектури з нуля.
- Команди стартапів, які будують продукти у сфері здоров’я, фітнесу чи нутріціології і хочуть використати готову логіку як базу для своїх клієнтських інтерфейсів.
- Науковці, дослідники або ентузіасти, які створюють експериментальні продукти чи дослідження, де потрібне точне формування раціону, обчислення нутрієнтів або симуляції харчових моделей.

- Освітні проєкти, що прагнуть продемонструвати студентам чи користувачам принципи раціонального харчування через інтерактивні додатки, не витрачаючи ресурси на складну бізнес-логіку.

Очікувані сценарії використання API включають:

- Додавання та фільтрацію рецептів із можливістю обрахунку їх харчової цінності на основі інгредієнтів.
- Формування та збереження індивідуальних планів харчування з розподілом страв по днях і прийомах їжі.
- Автоматичний аналіз відповідності денного раціону рекомендованим добовим нормам калорій, білків, жирів і вуглеводів.
- Побудова зведених списків покупок з агрегацією інгредієнтів з урахуванням кількості порцій та уникненням повторів.
- Отримання рекомендацій або повідомлень про виявлені порушення балансу нутрієнтів.

Таким чином, функціонал API підпорядкований реальним запитам розробників і дозволяє їм будувати корисні для кінцевого користувача продукти без потреби у створенні складної логіки з нуля. API орієнтований на масштабоване, багаторазове використання та відкритий до подальшого розвитку або інтеграції з іншими системами у сфері здорового способу життя.

2.2. Функціональні вимоги

Функціональні вимоги визначають основні можливості, які повинен реалізовувати API для задоволення потреб цільової аудиторії. Вони охоплюють усі ключові дії, які інші застосунки можуть реалізовувати за допомогою викликів до API. Інтерфейс має забезпечити повноцінну взаємодію з усіма основними сутностями системи, надаючи стандартизовані кінцеві точки у форматі REST.

1. **Управління рецептами:**
   - Додавання нових рецептів за допомогою POST-запитів з вказанням інгредієнтів, способу приготування, порційності та часу приготування.
   - Редагування або видалення існуючих рецептів через PUT/PATCH/DELETE-запити.
   - Отримання списку рецептів із фільтрами за категоріями, тривалістю приготування, складом або дієтичними обмеженнями (GET-запити).
2. **Формування плану харчування:**
   - Створення нового плану харчування, в якому можна задавати набір страв на день або тиждень із розподілом по прийомах їжі.
   - Редагування окремих прийомів їжі: додавання, оновлення або видалення рецептів у межах запланованих днів.
3. **Перевірка відповідності плану добовим нормам:**
   - Автоматичне порівняння запланованого споживання калорій, білків, жирів та вуглеводів із рекомендованими добовими нормами, сформованими на основі профілю користувача.

- Генерація попереджень про потенційні порушення (дефіцит чи надлишок окремих нутрієнтів).

4. **Генерація списку покупок:**
   - Формування зведеного списку інгредієнтів на основі одного або кількох планів харчування.
   - Агрегація однакових інгредієнтів з різних рецептів з урахуванням порційності.
   - Позначення кожного пункту як «придбано» або «очікує» для відображення в клієнтських застосунках.

Функціональність API має бути реалізована з використанням асинхронних REST-запитів з повною валідацією вхідних даних, описом моделей запитів і відповідей, а також автоматично згенерованою документацією (наприклад, Swagger UI). Такий підхід дозволить інтегрувати API в широке коло рішень: від мобільних застосунків до складних вебсервісів у сфері нутриціології, фітнесу та охорони здоров'я.

2.3. Нефункціональні вимоги

Окрім функціональних можливостей, API має відповідати певним нефункціональним вимогам, які визначають його якість, надійність, стабільність роботи, безпеку та потенціал до інтеграції й масштабування. У цьому підрозділі описуються вимоги до продуктивності, безпеки, документації, масштабованості, доступності та підтримуваності API, які критично важливі для розробників та інтеграторів.

1. **Юзабіліті та документація (Usability & Documentation):**
    - API повинен мати чітку та структуровану документацію (Swagger/OpenAPI), яка охоплює всі ендпоінти, параметри, формати відповідей та приклади використання.
    - Назви маршрутів та моделей повинні бути інтуїтивно зрозумілими та відповідати REST-принципам.
    - Повідомлення про помилки повинні бути зрозумілими, з відповідними кодами стану HTTP та описом причин.
2. **Продуктивність (Performance):**
    - API має обробляти запити із затримкою не більше 300–500 мс у більшості сценаріїв.
    - Необхідна підтримка пагінації, сортування та фільтрації для великих наборів даних.
3. **Безпека (Security):**
    - Підтримка авторизації та аутентифікації (наприклад, OAuth2, JWT) для захисту приватних даних користувачів.
    - Захист від типових атак, зокрема SQL-ін'єкцій, XSS, CSRF, brute-force тощо.
4. **Масштабованість (Scalability):**
    - API має бути готовим до розгортання у хмарному середовищі з можливістю горизонтального масштабування.
    - Архітектура повинна дозволяти легке додавання нових адрес без порушення вже існуючої логіки (principle of backward compatibility).

Таким чином, нефункціональні вимоги є основою для створення не лише функціонального, а й стабільного, безпечного, масштабованого та зручного для розробників API, який можна легко інтегрувати у різні проєкти та розширювати відповідно до нових вимог.

2.4. Архітектура системи

Архітектура API ґрунтується на клієнт-серверній моделі з акцентом на модульність, масштабованість та легку інтеграцію у зовнішні клієнтські застосунки. Основна увага зосереджена на проєктуванні надійного, продуктивного та добре задокументованого бекенду, який може бути використаний у веб-, мобільних чи десктопних рішеннях.

Основні компоненти системи:

1. **API-сервер:**
    - Відповідає за обробку HTTP-запитів за REST-принципами та забезпечення повного CRUD-функціоналу для основних сутностей: рецепти, продукти, плани харчування, списки покупок, профілі користувачів.
    - Реалізує всю бізнес-логіку: перевірку добових норм, агрегацію інгредієнтів, розрахунок калорійності, формування списку покупок тощо.
    - Забезпечує авторизацію користувачів, валідацію запитів, обробку помилок та форматування відповідей у структурованому форматі JSON.

2. **База даних:**
    - Використовується для зберігання структурованої інформації про користувачів, рецепти, продукти, плани харчування, нутрієнти тощо.
    - Забезпечує цілісність даних, підтримку зв'язків між сутностями та ефективний пошук і фільтрацію.

Логіка взаємодії:

- Клієнтські застосунки (мобільні, веб тощо) надсилають запити до API через HTTPS.
- API повертає відповіді з відповідними кодами HTTP та структурованими повідомленнями, включаючи результати обчислень, перевірки норм і згенеровані списки продуктів.
- Вся обробка логіки відбувається централізовано в межах API-сервера.

Варіанти розгортання:

- Локальний режим: запуск API-сервера на локальному середовищі для тестування або розробки.
- Хмарне середовище: розгортання API на обраному хостингу з публічним доступом та захистом даних користувача.
- Інтеграція в більші системи: можливість підключення API як частини комплексного сервісу, орієнтованого на здоров'я, дієтологію чи фітнес.

Така архітектура дозволяє зосередитись на якості логіки API, відкриває широкі можливості для повторного використання і масштабування, а також

створює гнучку основу для подальшого вибору конкретних технологій у наступних етапах проєктування.

2.5. Моделі даних

У цьому підрозділі представлено актуальну модель даних, що використовується у системі API. Описані сутності є основою для роботи всіх ключових функціональних модулів: управління рецептами, складання планів харчування, обрахунку нутрієнтів, формування списків покупок та автентифікації користувачів. Побудова логічно зв'язаної схеми забезпечує структуроване, надійне та розширюване зберігання інформації.

Користувач (User):

- id — унікальний ідентифікатор користувача.
- username — ім'я користувача.
- email — електронна пошта користувача.
- password_hash— захешований пароль.
- is_active — позначка активності облікового запису.
- is_superuser — позначка адміністративних прав.
- recipes — список рецептів, створених користувачем.
- meal_plans — список планів харчування користувача.
- shopping_lists — список покупок користувача.

Продукт (Product):

- id — унікальний ідентифікатор продукту.

- name — назва продукту.
- calories_per_unit — кількість калорій в одиниці виміру.
- proteins_per_unit — кількість білків в одиниці виміру.
- fats_per_unit — кількість жирів в одиниці виміру.
- carbohydrates_per_unit — кількість вуглеводів в одиниці виміру.
- unit_conversion_factor — коефіцієнт перерахунку в базову одиницю.
- original_unit — вихідна одиниця виміру продукту.

Одиниці виміру (Unit):

- GRAM, KILOGRAM, LITER, MILLILITER, PIECE — підтримувані одиниці виміру, використовуються у продуктах, інгредієнтах та списках покупок.

Рецепт (Recipe):

- id — унікальний ідентифікатор рецепта.
- name — назва рецепта.
- description — опис рецепта.
- photo — посилання на зображення страви.
- servings — кількість порцій.
- preparation_time — час приготування в хвилинах.
- instructions — текст інструкції з приготування.
- owner_id — ідентифікатор користувача-автора.
- owner — об’єкт користувача-автора.
- is_public— позначка, чи є рецепт публічним.
- ingredients — список інгредієнтів рецепта.

Інгредієнт (Ingredient):

- id — унікальний ідентифікатор інгредієнта.
- product_id — ідентифікатор продукту.
- product — об’єкт продукту.
- quantity — кількість інгредієнта.
- unit — одиниця виміру інгредієнта.
- calories — калорійність інгредієнта.
- proteins — вміст білків.
- fats — вміст жирів.
- carbohydrates — вміст вуглеводів.
- recipe_id — ідентифікатор рецепта.
- recipe — об’єкт рецепта.

План харчування (MealPlan):

- id — унікальний ідентифікатор плану.
- name — назва плану.
- owner_id — ідентифікатор власника.
- owner — об’єкт користувача-власника.
- entries — записи до плану харчування.

Запис у плані (MealPlanEntry):

- id — унікальний ідентифікатор запису.
- meal_plan_id — ідентифікатор плану харчування.
- date — дата споживання або приготування.

- recipe_id — ідентифікатор рецепта.
- recipe — об’єкт рецепта.
- meal_type — тип прийому їжі (сніданок, обід, вечеря, перекус).
- category — категорія (готування або споживання).
- servings — кількість порцій.
- meal_plan — об’єкт плану харчування.

Список покупок (ShoppingList):

- id — унікальний ідентифікатор списку.
- name — назва списку.
- owner_id — ідентифікатор власника.
- owner — об’єкт користувача.
- entries — список записів покупок.

Позиція у списку покупок (ShoppingEntry):

- id — унікальний ідентифікатор позиції.
- shopping_list_id — ідентифікатор списку покупок.
- shopping_list — об’єкт списку покупок.
- product_id — ідентифікатор продукту.
- product — об’єкт продукту.
- quantity — кількість продукту.
- unit — одиниця виміру.
- checked — позначка, чи було придбано товар.

Між цими сутностями встановлено зв’язки:

- Один користувач (User) → багато рецептів (Recipe).
- Один користувач (User) → багато планів харчування (MealPlan).
- Один користувач (User) → багато списків покупок (ShoppingList).
- Один рецепт (Recipe) → багато інгредієнтів (Ingredient).
- Один інгредієнт (Ingredient) → належить одному продукту (Product).
- Один план харчування (MealPlan) → багато записів (MealPlanEntry).
- Один запис у плані (MealPlanEntry) → пов'язаний з одним рецептом (Recipe).
- Один список покупок (ShoppingList) → багато позицій (ShoppingEntry).
- Одна позиція у списку покупок (ShoppingEntry) → пов'язана з одним продуктом (Product).

Фіксація продукту як окремої сутності (Product) дозволяє забезпечити контроль над агрегацією при створенні списків покупок — система розпізнає однакові продукти з різних рецептів та правильно підсумовує їх кількість.

Це також створює базу для точного розрахунку харчової цінності кожного інгредієнта, а отже й усього рецепта, добового плану харчування та його відповідності до встановлених норм. Кожен продукт має прив’язані значення калорій, білків, жирів і вуглеводів, що дозволяє динамічно підраховувати ці показники при зміні порцій чи страв.

Така структура дозволяє ефективно реалізувати ключові функції API: формування плану харчування, розрахунок добових норм, перевірку балансу

нутрієнтів, а також генерацію інтерактивного списку покупок. Модель підтримує масштабування, розширення та подальшу інтеграцію нових функцій системи.

2.6. Вибір технологій

Для реалізації програмного API обрано сучасний стек технологій, який забезпечує високу продуктивність, масштабованість та зручність розробки серверної частини. Основні компоненти включають:

Мова програмування:

- Python — універсальна мова з широкою підтримкою у сфері веб-розробки, зокрема для побудови асинхронних REST API.

Бекенд:

- FastAPI — сучасний фреймворк для створення високопродуктивних REST API. Підтримує асинхронне виконання запитів, автоматичну генерацію документації (Swagger/OpenAPI), типізацію, аутентифікацію та інтеграцію з базами даних.
- Pydantic — бібліотека для валідації та серіалізації вхідних/вихідних даних, яка тісно інтегрується з FastAPI.

База даних:

- PostgreSQL — надійна реляційна СУБД, що підтримує складні запити, транзакції та масштабоване зберігання даних.

- SQLAlchemy — ORM-бібліотека, яка забезпечує зручну роботу з базою даних та дозволяє описувати структуру моделей у вигляді Python-класів.

Інфраструктура та додаткові інструменти:

- Docker — для контейнеризації API, що спрощує його розгортання, тестування та розповсюдження.
- Git — система контролю версій, яка використовується для ведення історії змін, командної роботи та безпечного зберігання коду.
- Uvicorn — ASGI-сервер, що слугує точкою запуску FastAPI-програми в продакшн-середовищі.
- Alembic — інструмент для управління міграціями бази даних, що дозволяє відслідковувати зміни у схемі та застосовувати їх поетапно.

На відміну від повноцінного клієнт-серверного застосунку з інтерфейсом, цей проєкт зосереджений виключно на бекенді. Фронтенд реалізація не входить до складу поточної роботи, однак API є повноцінною серверною основою для майбутньої інтеграції з веб, мобільними чи іншими клієнтськими застосунками.

Запропонований стек технологій орієнтований на ефективну розробку API: усі компоненти мають активну спільноту, добру документацію та відмінну інтеграцію між собою. Це дозволяє будувати стабільний, швидкий і масштабований серверний інтерфейс з підтримкою ключових функцій проєкту.

2.7. Висновки

У цьому розділі було здійснено детальний опис архітектури, функціональних і нефункціональних вимог, моделей даних та обґрунтовано вибір технологій для реалізації API-сервісу. Проєкт спрямований на створення відкритого, розширюваного та багаторазово використовуваного бекенду для подальшої інтеграції з веб, мобільними чи іншими клієнтськими застосунками.

Архітектурне рішення базується на REST-підході та модульному поділі обов'язків, що дозволяє забезпечити гнучку обробку запитів, перевірку планів харчування на відповідність добовим нормам, керування рецептами, інгредієнтами, списками покупок та планами харчування. Усі ці компоненти були описані відповідно до сучасних вимог до API-сервісів.

Особливу увагу приділено моделі даних: її логічна структура підтримує всі функціональні сценарії, включаючи точний розрахунок нутрієнтів та агрегацію інгредієнтів у списках покупок. Виділення продукту як окремої сутності дозволяє централізовано зберігати харчову інформацію та багаторазово її використовувати в різних контекстах.

Обрані технології — FastAPI, PostgreSQL, SQLAlchemy, Pydantic, Alembic, Uvicorn — забезпечують високу швидкодію, безпеку, зручність підтримки та розширення. Контейнеризація через Docker та контроль версій за допомогою Git роблять API готовим до масштабування та розгортання в різних середовищах.

Таким чином, створено основу для якісного, сучасного API, який можна використовувати в освітніх, комерційних чи дослідницьких проєктах.

Подальші етапи включають реалізацію серверної логіки, написання тестів і перевірку системи в умовах реального використання з подальшим удосконаленням.

РОЗДІЛ ІІІ

3.1. Загальна структура проєкту

Проєкт реалізовано як модульний API-сервіс, заснований на принципах розділення обов'язків та чистої архітектури. Основні компоненти зберігаються в кореневій директорії, яка включає як конфігураційні файли, так і вихідний код системи. Загальна структура каталогу виглядає так:

Коренева директорія:

- main.py — головна точка входу в додаток; створює екземпляр FastAPI, підключає маршрути та ініціалізує API.
- models.py — містить ORM-моделі бази даних.
- schemas.py — містить Pydantic-схеми для валідації даних.
- database.py — налаштовує підключення до бази даних, створює сесії, визначає базовий клас моделей.
- dependencies.py — реалізує логіку автентифікації та отримання поточного користувача.

Конфігураційні файли:

- .gitignore, .env, alembic.ini, docker-compose.yml, Dockerfile, requirements.txt — забезпечують контроль версій, налаштування середовища та інструменти розгортання.

Директорії:

- routers/ — містить окремі модулі маршрутизації для функціональних блоків API (наприклад, recipe, mealplan, shoppinglist, auth, product). Кожен з них визначає відповідні ендпоінти.
- alembic/ — директорія з файлами для керування міграціями бази даних.

Така структура дозволяє зберігати чіткий розподіл між логікою взаємодії з даними, схемами валідації, маршрутизацією запитів та конфігураційними компонентами. Вона також забезпечує зручність у тестуванні, масштабуванні та супроводі проєкту. У наступних підрозділах буде докладно розглянуто реалізацію кожного з компонентів системи.

3.2. Реалізація моделей даних

У файлі models.py реалізовано основні ORM-моделі, які відповідають за структуру таблиць у базі даних. Для опису моделей використано SQLAlchemy — потужну бібліотеку об'єктно-реляційного відображення (ORM), яка дозволяє працювати з базою даних через Python-класи.

Моделі оголошено у декларативному стилі з використанням базового класу Base, створеного за допомогою declarative_base(). Типи стовпців визначено з модуля sqlalchemy, включаючи Integer, String, Float, Boolean, Date, ForeignKey та Enum. Зв’язки між сутностями реалізовано через relationship().

Ключові сутності:

- User — описує користувача з автентифікаційними полями та зв’язками до рецептів, планів і списків покупок.
- Recipe — містить назву, інструкції, час приготування, порційність, зображення, та зв’язується з користувачем та інгредієнтами.
- Ingredient — поєднує конкретний продукт (Product) з рецептом, зберігає нутрієнтну інформацію в контексті рецепта.
- Product — централізовано містить дані про калорійність, БЖВ на одиницю виміру, одиниці та конверсійний коефіцієнт.
- MealPlan і MealPlanEntry — реалізують зберігання плану харчування з деталізацією по днях і стравах.
- ShoppingList та ShoppingEntry — дозволяють створювати інтерактивні списки покупок із прив'язкою до продуктів.

У моделі також інтегруються перерахування enum.Enum, зокрема Unit (одиниці виміру), MealType (тип прийому їжі) та MealCategory (категорія дії: приготування або споживання).

Усі ці моделі формують чітку, нормалізовану структуру даних і безпосередньо підтримують функціонал API: перевірку добових норм, агрегацію інгредієнтів, формування меню та списків покупок. Реалізація через SQLAlchemy гарантує гнучкість, підтримку зв’язків, сумісність з PostgreSQL

та зручну інтеграцію з іншими частинами проєкту, включаючи Pydantic-схеми, міграції (Alembic) і маршрути.

3.3. Реалізація основної логіки API

Основна логіка API реалізована у вигляді модульних роутерів, згрупованих у директорії routers/ відповідно до функціональних областей: автентифікація, рецепти, продукти, плани харчування та списки покупок. Кожен роутер — окремий Python-модуль, що імпортується у main.py і реєструється в головному об'єкті FastAPI.

Ключові аспекти реалізації:

1. **Авторизація та безпека:**
   - Для автентифікації використовується стандарт OAuth2 з підтримкою JWT-токенів (jose.jwt).
   - Захист ендпоінтів реалізовано через залежності (Depends) з функцією get_current_user, яка перевіряє дійсність токена.
   - Паролі хешуються з використанням passlib і схеми bcrypt.
2. **Маршрутизація та REST:**
   - Для кожної сутності реалізовано повний набір CRUD-операцій: GET, POST, PUT, DELETE.
   - URL-параметри використовуються для ідентифікації об’єктів (наприклад, /recipes/{recipe_id}).

- Дані приймаються та повертаються у вигляді Pydantic-схем (schemas.py).

3. **Плани харчування та перевірка нутрієнтів:**
   - Ендпоінт /mealplans/{mealplan_id}/verify реалізує логіку перевірки плану харчування на відповідність добовим нормам (калорії, білки, жири, вуглеводи).
   - Норми масштабуються відповідно до кількості днів, на які складено план, і перевірка виконується по кожному макронутрієнту окремо.
4. **Інгредієнти та нутрієнтні обчислення:**
   - При додаванні інгредієнта до рецепта виконується автоматичний розрахунок його харчової цінності на основі вибраного продукту, враховуючи одиниці виміру та коефіцієнти перерахунку.
   - Це дозволяє точно акумулювати калорійність рецепта та подальші підрахунки в рамках плану харчування.
5. **Генерація списку покупок:**
   - API дозволяє створити список покупок на основі плану харчування (/shoppinglists/from-mealplan/{mealplan_id}), агрегуючи всі продукти з рецептів у плані.
   - Підтримується перерахунок одиниць виміру для правильного об’єднання однакових інгредієнтів.

Ця реалізація демонструє переваги використання FastAPI як платформи для побудови ефективних API-сервісів. Висока швидкодія, автоматична генерація документації, зручна система залежностей (Depends), а також інтеграція з Pydantic і SQLAlchemy роблять процес програмування інтуїтивно зрозумілим та продуктивним навіть для складних функціональних вимог.

3.4. Побудова маршрутизації

У проєкті маршрутизація побудована на базі механізму APIRouter з FastAPI. Це дозволяє створити окремі модулі для кожної логічної частини системи: користувачі (автентифікація), рецепти, продукти, плани харчування та списки покупок. Кожен з цих модулів має свій власний роутер, який реєструється у головному файлі main.py.

Кожен файл у директорії routers/ відповідає за окрему доменну область (наприклад, recipe.py, product.py).

Маршрути групуються за префіксами URL:

- /recipes/ — для створення, перегляду, редагування та видалення рецептів.
- /products/ — для роботи з базою продуктів.
- /mealplans/ — для планування харчування та перевірки раціонів.
- /shoppinglists/ — для генерації та редагування списків покупок.
- /signup, /login — для автентифікації користувачів.

Кожен роутер створюється за допомогою APIRouter() та підключається до FastAPI() через include_router().

Використовується Depends() для впровадження залежностей (авторизація, сесія БД).

Параметри запитів типізуються, а результати описуються через response_model, що дозволяє FastAPI автоматично формувати документацію.

Переваги такого підходу:

- Чіткий поділ функціоналу за модулями спрощує навігацію по коду.
- Можливість повторного використання залежностей і обробників помилок.
- Масштабованість: легко додавати нові функціональні блоки без порушення існуючої структури.
- Підтримка документації «з коробки» завдяки узгодженості між схемами та маршрутами.

Таким чином, побудова маршрутизації на основі APIRouter забезпечує ефективну організацію логіки API та сприяє легкому супроводу, модифікації й тестуванню системи.

3.5. Тестування API

Було проведено повне ручне тестування API з використанням Postman. Тестування охоплювало всі реалізовані ендпоінти для кожної з основних сутностей системи: користувачів, рецептів, продуктів, планів харчування та списків покупок.

Для кожного ендпоінта створено окремий набір запитів у Postman із параметрами, заголовками, тілом запиту та токенами автентифікації. У ході перевірки було протестовано типові сценарії використання API.

Всі відповіді сервера перевірялись на відповідність очікуваній структурі, правильність HTTP-статусів, коректність логіки фільтрації, створення, оновлення та видалення даних. Це дозволяє стверджувати про надійність API та його відповідність функціональним вимогам.

3.6. Розгортання та запуск API

Для забезпечення стабільного запуску та зручного розгортання API використано контейнеризацію за допомогою Docker та Docker Compose. Це дозволяє швидко налаштувати середовище для локальної розробки, тестування або розміщення на сервері.

У кореневій директорії розташований файл Dockerfile, який описує базове середовище виконання:

- використовується образ python:3.9-slim;
- встановлюються залежності з requirements.txt;
- відкривається порт 8000;
- запуск API здійснюється через uvicorn з вказаним головним файлом main:app.

Для одночасного запуску API та бази даних використано docker-compose. Конфігурація включає два сервіси:

- web: будується з поточного каталогу, чекає на запуск PostgreSQL, встановлює змінну DATABASE_URL, запускає міграції (alembic upgrade head) і сервер (uvicorn).
- db: використовує образ postgres:13, створює базу з логіном user та паролем password, зберігає дані у volume pgdata.

Приклад запуску:

- Побудова образів: docker-compose build
- Запуск контейнерів: docker-compose up
- Зупинка контейнерів: docker-compose down

API стає доступним за адресою http://localhost:8000. Документація Swagger доступна за http://localhost:8000/docs, а Redoc — за http://localhost:8000/redoc.

Такий підхід дозволяє швидко розгортати API в будь-якому середовищі без залежності від локальних налаштувань, забезпечуючи однакові умови виконання для розробників і серверного хостингу.

Можливі варіанти розгортання:

- Локальний запуск через Docker Compose — підходить для розробки та тестування. Всі сервіси запускаються однією командою на локальній машині.
- Розгортання на VPS або хмарному сервері — можна перенести зібрані образи на сервер (через SCP або CI/CD) та запустити за

допомогою Docker Compose. Для цього слід додатково налаштувати доменне ім'я, HTTPS (наприклад, через nginx + certbot).

- Heroku / Render / Railway — сервіси, що підтримують розгортання FastAPI-додатків із Docker. Потрібна інтеграція з GitHub або CI/CD pipeline. Базу даних PostgreSQL можна також підключити як керовану службу.
- Хмарні платформи (AWS, GCP, Azure) — можливе розгортання API як контейнера (через ECS, Cloud Run або App Service). Це дає гнучкість, але потребує додаткової конфігурації та знань DevOps.

Вибір платформи залежить від цілей проєкту, бюджету та навантаження. Завдяки Docker API залишається незалежним від середовища й легко переноситься між рішеннями.

3.7. Висновки

У цьому розділі було реалізовано повнофункціональний REST API для системи планування харчування з перевіркою на відповідність добовим нормам. На основі сучасного стеку технологій — FastAPI, PostgreSQL, Docker, SQLAlchemy, Pydantic — вдалося створити гнучкий, зручний у використанні та масштабований серверний застосунок.

Усі основні компоненти реалізовані та протестовані: користувацька автентифікація, управління рецептами, робота з продуктами, планування раціону та генерація списку покупок. Автоматичні обчислення харчової цінності та перевірка плану харчування демонструють глибоку інтеграцію прикладної логіки з даними.

FastAPI виявився ефективним інструментом для створення сучасного API: чітка типізація, автоматична генерація документації, підтримка асинхронності та залежностей значно спростили розробку. Весь функціонал було протестовано вручну через Postman, що дозволило підтвердити коректність реалізації.

Проєкт розгортається за допомогою Docker, що забезпечує універсальність у виборі середовища — від локальної машини до хмарної інфраструктури. Структура проєкту дозволяє легко масштабувати або адаптувати систему до нових вимог, а відкритість API дає змогу використовувати його як бекенд-основу для майбутніх веб- або мобільних клієнтів.

ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

У процесі виконання курсової роботи було здійснено повний цикл дослідження, проєктування та реалізації API-сервісу для підтримки здорового харчування. Робота охоплює теоретичне обґрунтування проблеми, формалізацію вимог, технічне проєктування та практичну реалізацію системи, а також її тестування і підготовку до розгортання.

У першому розділі було обґрунтовано актуальність тематики та розглянуто проблематику сучасного підходу до раціонального харчування. Зокрема, було проаналізовано наявні цифрові рішення, виявлено їх обмеження та визначено напрями вдосконалення. Особлива увага була приділена науковим джерелам, які обґрунтовують важливість обліку калорійності та складу поживних речовин у щоденному раціоні.

У другому розділі було спроєктовано архітектуру системи, визначено функціональні та нефункціональні вимоги, побудовано моделі даних. Архітектура орієнтована на відкритий REST API, що може бути використаний у різних клієнтських застосунках. Модель даних оптимізована для підтримки основних бізнес-сценаріїв — управління рецептами, планування харчування, перевірка на відповідність нормам, генерація списків покупок. Обраний стек технологій забезпечив високу ефективність, масштабованість і зручність розгортання.

У третьому розділі реалізовано повнофункціональний API з використанням FastAPI. Створено роутери для кожного з ключових модулів: користувачі, рецепти, продукти, плани харчування, списки покупок. Особливістю реалізації

стало використання залежностей (Depends), JWT-автентифікації, валідації через Pydantic і ORM-моделей на базі SQLAlchemy. Було реалізовано механізм перевірки раціону на відповідність добовим нормам і систему агрегації інгредієнтів у списках покупок. Усі ендпоінти пройшли ручне тестування через Postman, що підтвердило стабільність роботи API.

Для розгортання системи застосовано Docker та Docker Compose, що дозволяє легко запускати сервіс як локально, так і в хмарі. Описано можливості хостингу на VPS, через сервіси Heroku, Render або Railway.

Розроблений API є гнучким та практичним інструментом, готовим до інтеграції з будь-яким типом клієнтських застосунків. Система має чітку структуру, легко масштабується та забезпечує автоматизовану підтримку здорового харчування користувача.

Рекомендації щодо подальшого розвитку:

- Реалізувати клієнтський інтерфейс (веб або мобільний) для взаємодії з API.
- Додати персоналізацію нутрієнтних норм за параметрами користувача.
- Розширити API системою автоматичного підбору страв.
- Інтегрувати зовнішні джерела даних (наприклад, бази продуктів, сканування штрихкодів).
- Запровадити автоматизоване тестування (unit, integration).
- Додати логування, моніторинг та систему збору статистики

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Здорове харчування — важливий чинник здоров'я [Електронний ресурс] / Буковинський державний медичний університет. — Режим доступу: https://www.bsmu.edu.ua/blog/2090-zdorove-harchujanja

2. Раціональне харчування — запорука здоров'я [Електронний ресурс] / Ужгородський національний університет. — Режим доступу: https://www.uzhnu.edu.ua/uk/news/charchuvrac.htm

3. Поняття правильного харчування [Електронний ресурс] / Освітній портал Socrates. — Режим доступу: https://socrates.vsau.org/b04213/html/cards/getfile.php/14554.pdf

4. Digital applications for diet monitoring, planning, and precision nutrition: a critical review [Електронний ресурс] // Nutrition Reviews. — 2024. — Т. 83, № 2. — e574. — Режим доступу: https://academic.oup.com/nutritionreviews/article/83/2/e574/7667651

5. Cooking Matters Mobile Application: a meal planning and preparation tool for low-income parents [Електронний ресурс] // Public Health Nutrition. — Режим доступу: https://www.cambridge.org/core/journals/public-health-nutrition/article/9389D118534A62C85EAFEC133026DD76

6. Dietary Guidelines for Americans, 2020–2025 [Електронний ресурс] / U.S. Department of Agriculture, U.S. Department of Health and Human Services. — Режим доступу:

https://www.dietaryguidelines.gov/sites/default/files/2020-12/Dietary_Guidelines_for_Americans_2020-2025.pdf

7. How to Count Macros: A Step-By-Step Guide [Електронний ресурс] / Healthline. — Режим доступу: https://www.healthline.com/nutrition/how-to-count-macros