Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Куракіної Анастасії Станіславівни*
(ПІБ)

академічної групи *121-19ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення моделювання дієти на Python за допомогою Django*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	*Бердник М.Г.*			
розділів:				
спеціальний	*Бердник М.Г.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19ск-1* (група) *Куракіної А.С.* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення моделювання дієти на Python за допомогою Django*

затверджена наказом ректора НТУ «ДП» від *18 «травня» 2022 р.* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ (підпис) *проф. Бердник М.Г.* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Куракіна А.С.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 95 с., 57 рис., 5 табл., 3 дод., 31 джерело.

Об'єкт розробки: програмний додаток для моделювання дієти на Python за допомогою Django.

Мета кваліфікаційної роботи: розробка додатку для моделювання дієти з урахуванням особливостей харчування при цукровому діабеті другого типу. Додаток має надавати користувачам можливості введення даних для розрахунку щоденної потреби в енергії, задавати обмеження щодо загальної вартості продуктів та їх вживання, задавати термін, на який потрібно виконати моделювання дієти, корегувати вхідний список продуктів харчування, який буде виконано для розрахунків. За введеними користувачем даними додаток має виконувати розрахунок оптимального набору продуктів, що задовольняють обмеженням та мають потрібний при діабеті другого типу баланс білків, жирів та вуглеводів.

У вступі виконується аналіз сучасного стану проблеми, уточнюється постановка завдання, мета кваліфікаційної роботи та галузь її застосування, аналіз подібних розробок, обґрунтовується актуальність теми.

У першому розділі проводиться дослідження предметної галузі, визначається актуальність завдання та призначення розробки, розроблюється постановка завдання.

У другому розділі обирається платформа для розробки, виконується проектування програми і її розробка, наводиться опис алгоритму і структури функціонування системи, визначаються вхідні і вихідні дані, наводяться характеристики складу параметрів технічних засобів, описується інтерфейси та робота програми.

В економічному розділі визначається трудомісткість розробленого програмного продукту, проводиться підрахунок вартості роботи при створенні додатку та розраховується час на його створення.

Практичне значення полягає у створенні програмного додатку, що надає можливість отримувати список продуктів харчування, їх енергетичної цінності та вартості, робити розрахунок продуктового кошику за індивідуальними характеристиками користувача та виконувати оптимізацію представленого набору продуктів.

Актуальність інформаційної системи визначається попитом на подібні системи і відсутністю програмного забезпечення, що дозволяє поєднувати енергетичну цінність та актуальну вартість продуктів. Програма буде корисна для людей, що бажають оптимізувати власний раціон харчування з урахуванням оптимального енергетичного складу продуктів у раціоні.

Список ключових слів: WEB, СЕРВЕР, КЛІЄНТ, DJANGO, ANGULAR, SPA, ДІЄТА, ПРОДУКТОВИЙ КОШИК, ЕНЕРГЕТИЧНА ЦІННІСТЬ.

ABSTRACT

Explanatory note: 95 pp., 57 figs., 5 tabs., 3 apps, 31 sources.

Object of development: Python diet simulation software using Django.

Purpose of the qualification work: development of an application for modeling a diet taking into account the peculiarities of nutrition in type 2 diabetes. The application should provide users with the ability to enter data to calculate daily energy needs, set limits on the total cost of products and their consumption, specify the period for which you want to perform diet simulations, adjust the input list of foods to be performed for calculations. According to the data entered by the user, the application should calculate the optimal set of products that meet the restrictions and have the right balance of proteins, fats and carbohydrates in second type diabetes.

The introduction analyzes the current state of the problem, clarifies the problem, the purpose of the qualification work and the scope of its application, substantiates the relevance of the topic.

In the first section the research of the subject area and existing decisions is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed.

In the second section the platform for development is chosen, the program design and its development are carried out, the description of algorithm and structure of functioning of system is given, input and output data are defined, characteristics of structure of parameters of technical means are given, work of the program i described.

The economic section determines the complexity of the developed software product, calculates the cost of work to create an application and calculates the time to create it.

The practical significance is to create a software application that allows you to get a list of foods, their energy value and cost, to calculate the food basket according to individual user characteristics and to optimize the presented set of products.

The relevance of the information system is determined by the demand for such systems and the lack of software that combines energy value and actual cost of products. The program will be useful for people who want to optimize their diet based on the optimal energy composition of foods in the diet.

Keywords: WEB, SERVER, CLIENT, DJANGO, ANGULAR, SPA, DIET, GOODS BASKET, ENERGY VALUE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

ОС – операційна система;

ЕОМ – електронно-обчислювальна машина;

PIP – Python Package Index (з англ. Індексатор пакетів для Python);

МНК – метод найменших квадратів;

ІМТ – індекс маси тіла;

SQP – Sequential quadratic programming (з англ. метод послідовного квадратичного програмування);

SPA – Single Page Application (з англ. односторінковий додаток);

REST – Representational State Transfer (з англ. передача репрезентативного стану);

API – Application Programming Interface (з англ. інтерфейс для програмування застосунків);

UML – Unified Modeling Language (з англ. уніфікована мова моделювання);

JWT – JSON Web Token (з англ. JSON веб-токен).

ВСТУП

Питання здоров'я та харчування завжди тісно пов'язані між собою. Від щоденного раціону, якості, кількості та складу продуктів залежить чи буде отримувати організм людини належну кількість елементів для повноцінного функціонування.

Однією з багатьох хвороб, що потребує ретельного догляду за харчуванням та вибором продуктів для щоденного раціону, є цукровий діабет. Цукровий діабет є групою метаболічних захворювань, характеристикою яких є хронічна гіперглікемія. Вона виникає внаслідок порушень секреції інсуліну та /або дії інсуліну.

Дієта відіграє ключову роль у лікуванні діабету другого типу. Склад дієти включає підтримку балансу у споживанні вуглеводів, жирів та білків. Особливу увагу треба приділяти вуглеводам, що складають близько 55-60 відсотків раціону. Потрібно вводити обмеження у споживанні продуктів, що містять швидкі вуглеводи, такі як хлібобулочні та кондитерські вироби, цукерки, солодкі напої і т.п.

Досить часто стикаючись з необхідністю підтримки такого типу дієти, людини необхідно проводити окремі дослідження щодо складу та харчової цінності продуктів, вивчати які товари пропонують магазини та думати як на певну виділену суму придбати усі потрібні для раціону продукти.

Після дослідження предметної галузі та вивчення існуючих рішень для розв'язанні даного типу проблем, було виявлено нестачу програмних продуктів та калькуляторів для швидкого розрахунку рекомендованого списку продуктів з урахуванням індивідуальних характеристик користувача та пропозицій українського ринку.

Таким чином, було запущено процес проектування та реалізації програмного забезпечення, що буде виконувати вищеозначені функції, а саме: зберігати та своєчасно оновлювати базу продуктів, які представлені на українському ринку; надавати зручний інтерфейс вводу даних користувача та

можливість індивідуального налаштування списку продуктів та обмежень до них; виконувати моделювання продуктового кошику користувача за заданими обмеженнями та на задану суму з урахуванням рекомендованого балансу вуглеводів, білків та жирів.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Дієта – набір правил споживання їжі, що може включати в себе такі фактори як: кулінарна обробка їжі, фізичні властивості, хімічний склад, інтервал та час прийому їжі. Дієту часто призначають при різного роду захворюваннях, при цьому тип дієти та набір й состав продуктів, залежать від типу захворювання та індивідуальних особливостей людини.

Цукровий діабет є групою метаболічних захворювань, характеристикою яких є хронічна гіперглікемія. Вона виникає внаслідок порушень секреції інсуліну та /або дії інсуліну [1].

Одним з найбільш частих випадків діабету є цукровий діабет другого типу, який зустрічається найчастіше у дорослих. Його причиною є те, що організм не обробляє належним чином інсулін, що він виробляється. Одною з ключових ролей в лікуванні діабету другого типу є здорове харчування та підтримка належного рівня фізичної активності.

Рекомендованим розподілом щоденного раціону має бути підтримка наступного балансу: вуглеводи 55-60%, жири 20-25%, білки 15-20% та виключення з дієти рафінованих вуглеводів, що легко засвоюються [2-4].

Білки – є основної складовою клітин нашого організму, їхня основна функція – структурна. Вони бувають тваринного та рослинного походження. Тваринні білки містяться у яйцях, сирах, м’ясі, риби. Рослини білки містяться у горіхах, бобових. При окисленні одного грама білка виділяється енергія, що складає в середньому 4 кілокалорії.

Основною функцією жирів або ліпідів є запасна. Вони також поділяються на рослинні та тваринні. Тваринні жири знаходяться у кисломолочній продукції та у м’ясі. Рослинні жири зазвичай називають оліями: гарбузова, оливкова, лляна, кунжутна тощо. В одному грамі жиру міститься енергія, що складає в середньому 9 кілокалорій.

Основною функцією вуглеводів є енергетична. Вони є джерело харчування для клітин, так як саме під час окислення вуглеводів клітина отримує основну частину енергії. В одному грамі вуглеводу міститься енергія, що складає в середньому 4 кілокалорії [5].

Розрахунок енергетичної цінності добового раціону має робитись з урахуванням маси тіла, віку, статі, енергетичних витрат.

Добова потреба організму в енергії залежить від маси тіла [6]. Співвідношення денної потреби в енергії відносно індексу маси тіла, при стані спокою, представлено у таблиці 1.1.

Таблиця 1.1

Співвідношення денної потреби в енергії відносно індексу маси тіла, при стані спокою

Тип статури	Перевищення маси тіла, %	Денна потреба в енергії, ккал/кг
Худий	-5 -9	25
Нормальний	0	20
Ожиріння 1-2-го ступеня	10-49	17
Ожиріння 3-го ступеня	> 50	15

Індекс маси тіла розраховується за формулою [7]:

$$\text{ІМТ} = \frac{m}{h^2},$$

де m – маса тіла у кілограмах,

h – зріст людини у метрах.

Нормальним вважається ІМТ у межах 18,5 - 24,9. Потреба енергії у людини також залежать від рівню фізичної активності. Якщо прийняти потребу в

енергії людини в стані спокою за *A*, то потреба в енергії залежно від рівню фізичної активності буде приблизно така, як відображено у таблиці 1.2. [8].

Таблиця 1.2

Потреба в енергії в залежності від рівню фізичної активності людини

Рівень фізичної активності	Загальна кількість енергії, яка потрібна на добу
Немає фізичних навантажень, сидяча робота.	1,2**A*
Виконання невеликих пробіжок або легкої гімнастики 1-3 рази в тиждень.	1,375**A*
Заняття спортом із середніми навантаженнями 3-5 разів на тиждень.	1,55**A*
Повноцінні тренування 6-7 разів на тиждень.	1,725**A*
Робота пов'язана з фізичною працею. Тренування 2 рази в день з включання в програму тренувань силових вправ.	1,9**A*

При аналізі ринку програмного забезпечення за заданим напрямком було виявлено низьку калькуляторів енергетичної цінності продуктів з можливістю складання меню на їх основі. Прикладом такого калькулятору є web-додаток «Таблиця калорійності», що дозволяє дізнатися енергетичної цінності та склад продукту (рис. 1.1), скласти індивідуальне меню з підрахунком його загальної калорійності [9].

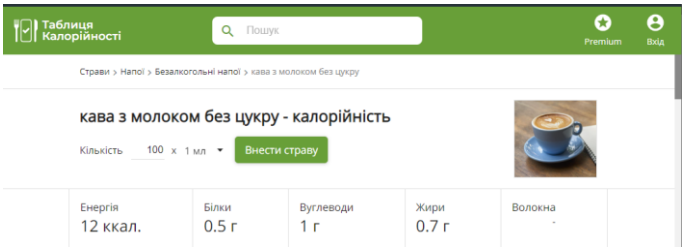
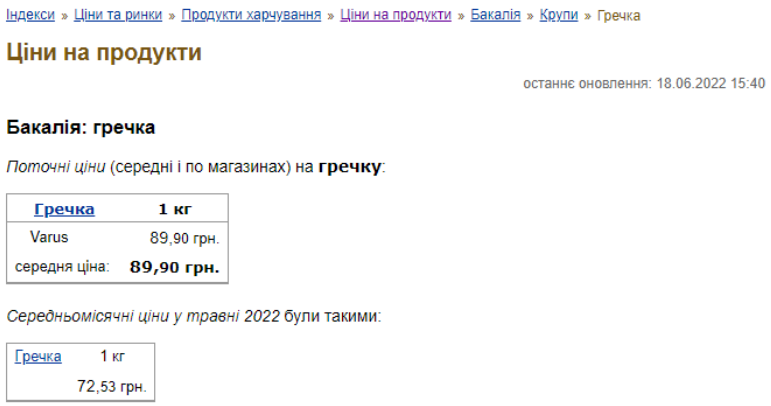


Рис. 1.1. Інтерфейс додатку «Таблиця калорійності»

Додаток є зручним інструментом для контролю продуктів споживання, але не надає можливості підраховувати вартість продуктового кошику та дізнаватися вартість продуктів.

Дізнатися середню вартість основних продуктів харчування з урахуванням українського ринку, дозволяє сайт міністерству фінансів, що містить актуальну інформацію щодо цін на першорядні продукти харчування, які представлені у торгових мережах Auchan, Fozzy, Furshet, Megamarket, Metro, Novus, Varus [10]. Для ознайомлення представлені як середньомісячні, так і поточні ціни продуктів (рис. 1.2).


Індекси » Ціни та ринки » Продукти харчування » Ціни на продукти » Бакалія » Крупи » Гречка

Ціни на продукти

останнє оновлення: 18.06.2022 15:40

Бакалія: гречка

Поточні ціни (середні і по магазинах) на **гречку**:

Гречка	1 кг
Varus	89,90 грн.
середня ціна:	**89,90 грн.**

Середньомісячні ціни у травні 2022 були такими:

Гречка	1 кг
	72,53 грн.

Рис. 1.2. Сторінка сайту міністерству фінансів з інформацією про ціни на продукти споживання

Таким чином особливістю програмного додатку даної кваліфікаційної роботи буде організація зв’язку між енергетичною цінністю продукту та його актуальною вартості, для формування оптимального продуктового кошику користувача.

1.2. Призначення розробки та галузі застосування

Метою даної роботи є спрощення процесу вибору товарів продуктового кошику для споживача з урахуванням особливостей дієтичного харчування для цукрового діабету другого типу.

Головним завданням при підборі продуктів для кошика споживача є максимізація кількості корисних калорій з підтриманням коректного

співвідношення вуглеводів, жирів та білків, з мінімізацією вартості цього кошика.

1.3. Підстава для розробки

Підставами для розробки та виконання кваліфікаційної роботи) є:

– освітня програма 121 «Інженерія програмного забезпечення»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18.05.2022 р;

– завдання на кваліфікаційну роботу на тему «Розробка програмного забезпечення моделювання дієти на Python за допомогою Django».

1.4. Постановка завдання

Розробити клієнт-серверний додаток для автоматизації процесу моделювання дієти при цукровому діабеті 2 типу мовою програмування Python за допомогою фреймворку Django.

Додаток має включати web-клієнт, що реалізовано за допомогою фреймворку JavaScript Angular. Програма повинна працювати у всіх сучасних браузерах.

1.4.1. Функціональні можливості:

1. Введення даних для розрахунків: термін, на який має рахуватися продуктовий кошик, максимальна сума на покупку, дані користувача.

2. Перегляд, редагування списку продуктів, що буде використано в процесі розрахунків.

3. Перегляд, редагування списку обмежень, що буде використано в процесі розрахунків.

4. Виконання моделювання дієти при ЦД 2 типу при заданих параметрах.

5. Перегляд результатів моделювання.

6. Завантаження результатів моделювання в одному зі зручних форматів: pdf, txt, doc, xls.

7. Друк результатів моделювання.

8. Адміністрування системи: перегляд / редагування списку продуктів для розрахунків за замовчуванням, запуск завантаження даних з зовнішніх джерел.

Опис функціоналу за ролями

Додаток має дві основні групи користувачів: користувач додатку та адміністратор додатку.

Користувач має можливість:

– вводити дані для розрахунку оптимальної кількості калорій на добу: стать, ріст, вагу, характер робітничої діяльності та рівень фізичного навантаження;

– вводити дані щодо терміну, на який буде виконано розрахунок продуктового кошику, та максимальної суми, на яку потрібно придбати продуктів;

– переглядати список продуктів для розрахунків;

– маніпулювати списком продуктів для розрахунків: редагування, додавання, видалення;

– перегляд списку обмежень до продуктів;

– додавати обмеження для продуктів: більше ніж, менше ніж, рівно певної кількості або масі;

– переглядати результати розрахунку продуктового кошику;

– друкувати результати розрахунку;

– завантажувати результати розрахунку у форматах: pdf, txt, doc, xls;

– зберігати кілька списків продуктів

– створюватися обліковий запис у системі, заповнивши дані щодо: імені користувача, адреси електронної скриньки скриньки, пароля користувача;

– авторизуватися у системі

Адміністратор має можливість:

– переглядати список продуктів для продуктового кошику та їх характеристик за замовчуванням, редагувати список продуктів для продуктового кошику та їх характеристик за замовчуванням;

– запускати підпрограму для завантаження даних вартості продуктів;

– переглядати список користувачів;

– видаляти користувачів.

Схематичне зображення можливостей користувачів представлено у діаграмі варіантів використання (рис. 1.3)

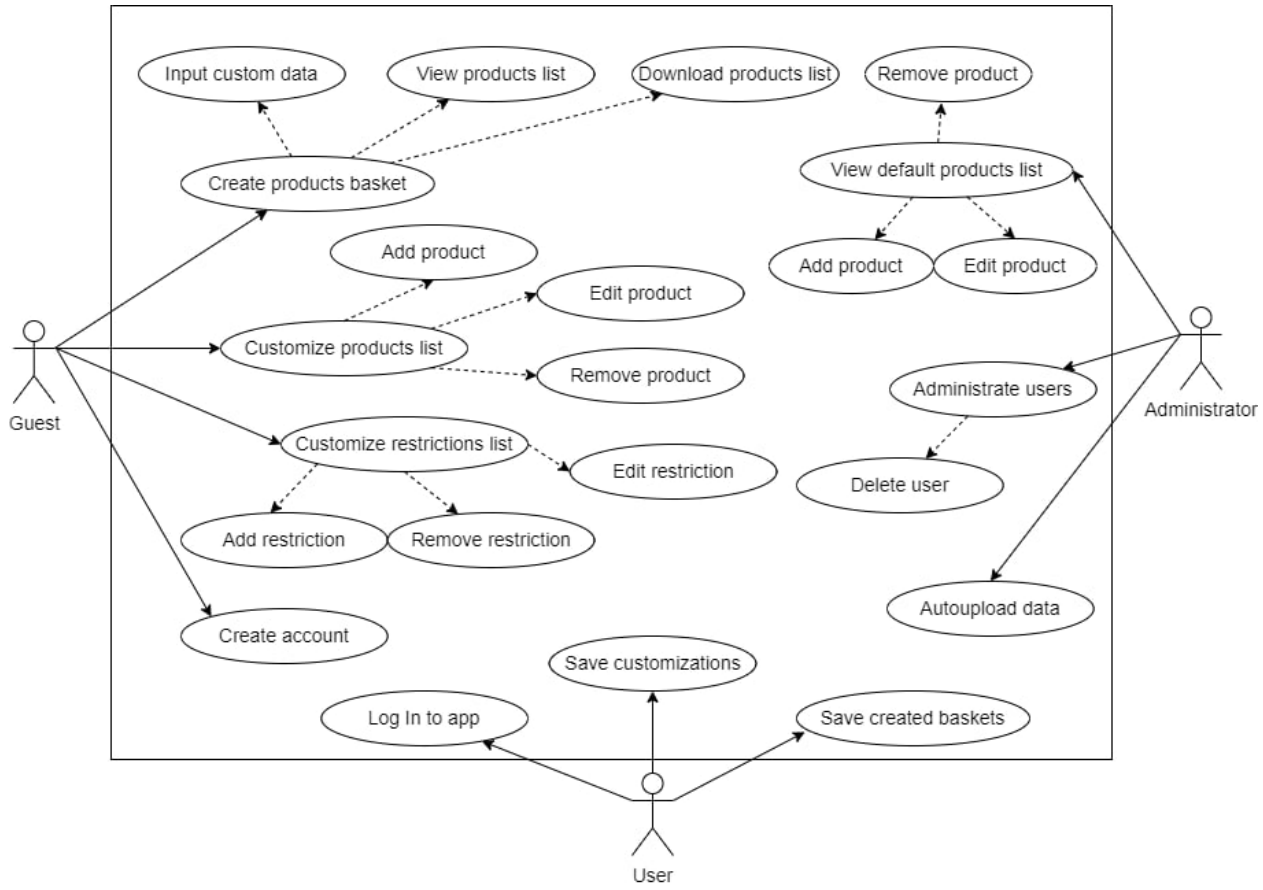


Рис. 1.3. – Діаграма варіантів використання

Опис основних об`єктів програмної системи

Опис основних об’єктів програмної системи представлено у таблиці 1.3.

Таблиця 1.3

Опис основних об`єктів програмної системи

Сутність	Опис атрибутів
Авторизований користувач	– email; – password – ім’я користувача.
Користувач	– ріст; – вага; – стать; – рівень фізичного навантаження.
Продукт	– назва; – категорія; – вартість в магазині; – одиниця вимірювання;
Продукт	– кількість енергії (кілокалорій) на одиницю вимірювання; – кількість білків, жирів та протеїнів на одиницю вимірювання; – стан продукту; – обмеження щодо харчування.
Обмеження	– продукт; – тип порівняння; – кількість продукту.
Продуктовий кошик	– дані користувача; – список продуктів; – термін розрахунку; – максимальна вартість кошику.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Система має бути реалізована як клієнт-серверний web-додаток. Web-клієнт має бути кросбраузерним та коректно працювати у всіх сучасних браузерах та кросплатформеним з адаптивністю до пристроїв різних розмірів.

Бізнес логіка та розрахунки мають відбуватися на боці серверу.

1.5.2. Вимоги до інформаційної безпеки

Вимоги до інформаційної безпеки включають:

– гарантування цілісність даних;

– захист та конфіденційність індивідуальних даних користувача та його налаштувань;

– розділення клієнту для виконання адміністрування додатку та клієнту для виконання розрахунків;

– захист від неавторизованого входу у систему адміністрування.

Для ідентифікації та автентифікації користувача використовувати унікальну пару email / password та використовувати токен автентифікації.

1.5.3. Вимоги до складу та параметрів технічних засобів

Рекомендовані вимоги до апаратного та програмного забезпечення:

Вимоги до серверу:

- об’єм оперативної пам’яті: 512МБ;
- операційна система: Ubunty Linux 16;
- підтримка протоколів: HTTP/ HTTPS та HTTP-сервер: Nginx, Gunicorn;
- підтримка бази даних: PostgresSQL версії 9.6 та вище;
- підтримка додаткових технологій: Django framework.

Рекомендованими вимогами до програмного забезпечення клієнта є:

- наявність пристрою з ОС 2007 року випуску та пізніше;

− наявність одного з перерахованих браузерів зі вказаною або вищою версією: Google Chrome 16, Opera 12.1, Mozilla Firefox 11, Apple Safari 5.

Персональний комп’ютер повинен мати доступ до мережі Інтернет.

1.5.4. Вимоги до інформаційної та програмної сумісності

Програмний продукт повинен бути розроблений з використанням мови програмування Python з використанням об’єктно-орієнтованої парадигми. Для виконання розрахунків необхідно застосовувати функції бібліотеки SciPy, що дозволяє виконувати оптимізацію за заданою цільовою функцією та обмеженнями.

Необхідно забезпечити функціонування розробленого додатку у низькі сучасних браузерів та підтримувати роботу у цих браузерах від мінімальних версій: Google Chrome 16, Opera 12.1, Mozilla Firefox 11, Internet Explorer 10, Apple Safari 5.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом кваліфікаційної роботи має бути клієнт-серверний додаток, що надає можливості введення даних для розрахунків, обчислення дієти, роботу з профілями користувачів.

Серверна частина додатку відповідає за роботу з даними, верифікацію користувачів, виконання обчислень та обслуговування клієнтів за допомогою REST API. Для зручності роботи з додатком було реалізовано два web-клієнта.

Web-клієнт, що відповідає за можливості адміністрування додатком та надає наступний функціонал:

– перегляд інформації щодо: користувачів системи, набору продуктів для обчислень, набору обмежень для вживання продуктів, розрахованих продуктових кошиків;

– керування профілями користувачів системи: додавання нового користувача, видалення користувача;

– керування списком продуктів за замовленням: додавання нового продукту, редагування існуючого продукту, видалення продукту;

– керування списком обмежень за замовленням: додавання нового обмеження, редагування існуючого обмеження, видалення обмеження;

– виконання запитів для автоматизації даних.

Web-клієнт, що відповідає за роботу з вводом даних користувача та відображення результатів обчислень, та надає наступний функціонал:

– реєстрація нового користувача з введенням наступних даних: ім’я користувача, адреса електронної скоринці, пароль. Під час реєстрації користувач має можливість відразу заповнити дані для виконання обчислень: стать, вік, ріст, вагу, рівень фізичної активності;

– авторизація користувача з введенням наступних даних: ім’ям користувача, пароль;

– введення даних для обчислення продуктового кошику: період, на який потрібно виконати обчислення (доба, тиждень, місяць), максимальна вартість продуктового кошику, дані користувача для обчислення кількості калорій на добу;

– перегляд списку продуктів, що будуть використані для обчислень;

– фільтрування списку продуктів за категорією продукту;

– пошук за назвою продукту;

– додавання, видалення та редагування продуктів;

– збереження зміненого списку продуктів для поточного користувача;

– перегляд списку обмежень на використання продуктів, що будуть використані для обчислень;

– пошук обмежень за назвою продукту;

– сортування обмежень за назвою продукту та типом обмеження;

– додавання, видалення та редагування обмежень для поточного користувача;

– збереження зміненого списку обмежень для поточного користувача;

– перегляд результатів обчислень та оптимізації, що включає: загальну вартість продуктів, загальну кількість калорій, співвідношення білків, жирів, вуглеводів у продуктовому кошику, список продуктів;

– збереження результатів обчислень під заданою назвою;

– перегляд списку власних продуктових кошиків користувача з інформацією щодо назви, вартості, енергетичної цінності, дати створення кошику;

– сортування продуктових кошиків з назвою, терміном, енергетичною цінністю, датою створення;

– пошук продуктових кошиків з назвою;

– видалення власних продуктових кошиків.

2.2. Опис застосованих математичних методів

2.2.1. Математична модель

Вивчимо поведінку споживача, який має намір за певний період закупити продукти харчування в кількості: x_i, $i = 0,1, \dots n-1$, за ціною p_i. На закупку продуктів споживач виділяє z гривень. Вказані величини повинні бути зв'язані співвідношенням, яке відображено у формулі (2.1).

$$\sum_{i=0}^{n-1} p_i x_i = z, z > 0 \qquad (2.1)$$

На суму z товари можуть бути закуплені неоднозначно. Тому з усіх варіантів закупок споживач повинен обрати найкращий. Для цього покупець повинен обрати стратегію закупок.

При формування стратегії закупок потрібно врахувати, що для повноцінного функціонування організму споживач має отримувати певну кількість енергії з продуктів харчування. Кількість енергії E залежить від індивідуальних особливостей споживача та розраховується за даними представленими у розділі 1.1., таблицями 1.1-1.2. Згідно даному фактору стратегія закупівель продуктів харчування повинна задовольняти обмеження, описані у формулі (2.2.).

$$\begin{cases} \sum_{i=0}^{n-1} e_i x_i = E, E = const \\ z \le z_{max} \end{cases} \qquad (2.2)$$

де e_1 – кількість енергії на одиницю продукту,

E – затрати енергії на певний період,

z_{max} – максимальна допустима вартість продуктів.

Для оптимізації раціону харчування продукти споживання мають містити певну кількість компонентів: вуглеводів, жирів та білків. Для задоволення даної

потреби для стратегії закупівель було введено додаткові обмеження, задані у формулі (2.3).

$$\begin{cases} c_{min} \leq \sum_{i=0}^{n-1} x_i c_i \leq c_{max} \\ f_{min} \leq \sum_{i=0}^{n-1} x_i f_i \leq f_{max} \\ pr_{min} \leq \sum_{i=0}^{n-1} x_i pr_i \leq pr_{max} \end{cases} \quad (2.3)$$

де c_i – кількість вуглеводів на одиницю продукту,

c_{min} – мінімальна рекомендована кількість вуглеводів за певний період,

c_{max} – максимальна рекомендована кількість вуглеводів за певний період,

f_i – кількість жирів на одиницю продукту,

f_{min} – мінімальна рекомендована кількість жирів за певний період,

f_{max} – максимальна рекомендована кількість жирів за певний період,

pr_i – кількість білків на одиницю продукту,

pr_{min}– мінімальна рекомендована кількість білків за певний період,

pr_{max}– максимальна рекомендована кількість білків за певний період.

Кожен продукт харчування може мати максимальну та мінімальну межу рекомендованої кількості для споживання у певний період. Це надає додаткові обмеження, представлені у формулі (2.4)

$$x_{i\,min} \leq x_i \leq x_{i\,max} \quad (2.4)$$

де x_i – кількість продукту,

$x_{i\,min}$ – мінімальна рекомендована кількість продукту за певний період,

$x_{i\,max}$ – мінімальна рекомендована кількість продукту за певний період.

Таким чином стратегія закупівель полягає у мінімізації вартості продуктів харчування для закупівлі $z \rightarrow \min$ з врахуванням обмежень, представлених у формулах (2.2), (2.3) та (2.4).

2.2.2. Метод оптимізації

У якості методу оптимізації було обрано метод послідовного квадратичного програмування SQP. Він є одним з найбільш ефективних та поширених оптимізаційних алгоритмів загального призначення. Основною ідеєю методу є послідовне вирішення задач квадратичного програмування, що апроксимують дану задачу оптимізації [11].

SQP виконує мінімізацію функції кількох змінних з будь-якою комбінацією обмежень, рівності та нерівності.

Метод було обрано через низьку факторів:

– можливість рішення оптимізаційних завдань з обмеженнями нелінійного та лінійного типу у вигляді рівностей або нерівностей;

– можливість розв’язання великих систем рівнянь, використовуючи обчислювальні алгоритми, що враховують особливості реалізації даних у програмній системі.

Для виконання оптимізації SQP використовує лінійний метод найменших квадратів, що призначений для пошуку наближеного розв'язку надлишково-визначеної системи.

Процес підготовки параметрів для функції оптимізації представлено у блок-схемі на рис. 2.1.

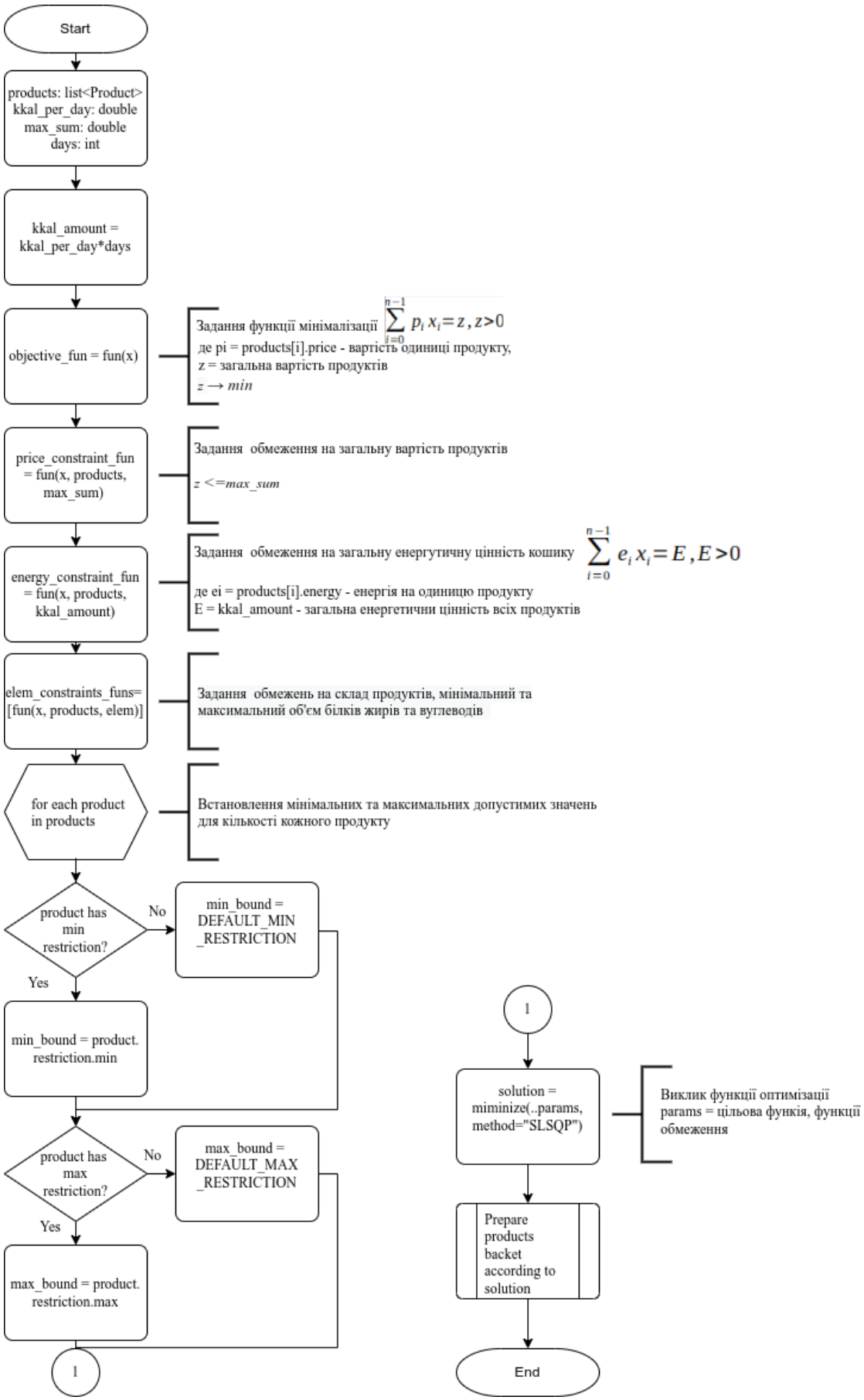


Рис. 2.1. – Блок-схема функції підготовки параметрів для процесу оптимізації та виклик функції оптимізації

2.3. Опис використаної архітектури та шаблонів проектування

Фреймворк Django реалізовує шаблон MTV (Model - Template - View) для організації роботи серверу. Таким чином проект розділено на три основни частини:

− M – Модель(Model), шар доступу до даних. Виконує отримання даних, їх валідацію, роботу з даними та зв'язок між ними.

− T – Шаблон (Template), шар представлення даних. Цей шар приймає рішення відносно представлення даних: як і що повинно відображатися на сторінці або в іншому типі документу.

− V – Представлення (View), шар бізнес-логіки. Цей шар містить логіку, як діставати доступ до моделей і застосовувати відповідний шаблон. Це «міст» між моделями та шаблонами [12].

Приклад взаємодії між частинами серверу представлено на рис. 2.2.

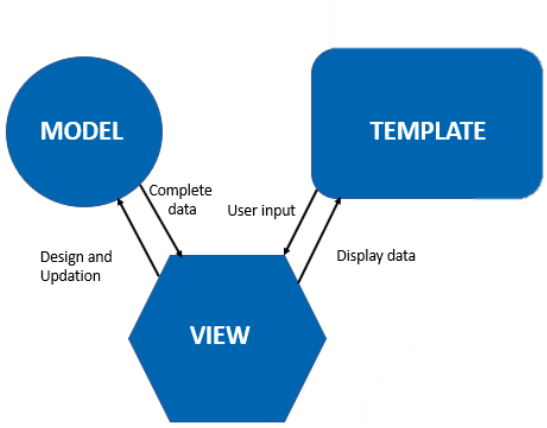


Рис. 2.2. Взаємодія компонентів серверу

Для організації роботи з базою даних було використано шаблон ORM.

ORM (Object-relational mapper) – механізм об'єктно-орієнтованого відображення, що дозволяє автоматизувати типові взаємодії з базою даних і використовує об'єктно-орієнтований підхід замість інструкцій SQL [13]. Таблиці бази даних представлені у вигляді класів, де властивості – це поля таблиці, а запис у таблиці – об’єкт класу.

Для організації взаємодії серверу та клієнту було використано принципи архітектури REST (англ. Representational State Transfer). Це підхід до реалізації архітектури взаємодії мережевих протоколів, що базується на можливостях функціонування глобальних мереж та протоколу HTTP.

REST пропонує передавати дані, що можуть бути представлені кількома стандартними форматами, такими як (JSON, XML, HTML). Підтримує кешування та не має мати залежність від особливостей мережевого прошарку. REST підхід не має зберігати інформацію про стан між запитом та відповіддю та надає можливість будувати масштабовані програмні системи [14].

2.4. Опис використаних технологій та мов програмування

Список використаних мов програмування та додаткових бібліотек, що було використано при розробці програмного продукту наведено у таблиці 2.1.

Таблиця 2.1

Список основних технологій та мов програмування

Сервер	
Фреймворк	Django framework version 14
Мова програмування	Python version 3.8
База даних	PostgreSQL
Бібліотека для обчислень	SciPy
Бібліотека для побудови API	Django Rest framework
Клієнт	
Фреймворк	Angular framework version 13
Мова програмування	TypeScript, JavaScript
Мова шаблонів	HTLM
Таблиця стилів, препроцесор	CSS, SCSS
Бібліотека для побудови інтерфейсу	Angular Material
Бібліотека для побудови графіків	Ng2 Charts

Для мови програмування для реалізації серверу було задано наступні критерії вибору: наявність математичних бібліотек, які підтримують методи оптимізації, що були розглянуті у пункті 2.2.2.; наявність фреймворку або бібліотеки для розгортання web-серверу; підтримка об’єктно-орієнтованої парадигми програмування; швидкість розробки.

Було розглянути такі популярні мови програмування: JavaScript, C#, Python. Кожна вищеозначених мов підтримує об’єктно-орієнтовану парадигму програмування, має сучасний фреймворк для реалізації взаємодії по HTTP-протоколу. За критерієм підтримки математичного апарату було виключено JavaScript, так як ця мова розробки не є орієнтованою на виконання точних математичних обчислень. C# та Python мають в наявності математичні бібліотеки для вирішення завдань оптимізації, але Python надає певні переваги щодо швидкості розробки web-серверів. Фреймворк Python Django надає можливості швидкого підключення та організації роботи з користувачами, пропонує вже готову екосистему для роботи з базою даних та обслуговування запитів. Тому в результатом досліджень був вибір мови програмування Python 3.

Python – це об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією, що дозволяє гнучко працювати з системою класів та об’єктів, створювати ієрархії класів і т.п.

Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється під вільною ліцензією Python Software Foundation License, що дозволяє використати його без обмежень у будь-яких проектах [15].

Python пропонує зручну систему модулів та пакетів модулів, що дозволяють розділити програмний код на логічно розмежовану структуру файлів та сприяє повторному використанню коду. Серед переваг мови можна також відмітити зручну структура, яка вимагає застосовувати відступи заданого розміру для виділення блоків коду, що позитивно відображається на чистоті коду.

Python включає велика низьку готових модулів та бібліотек, які підключаються до власного проекту за допомогою системи управління пакетами Python Package Index (pip) [16].

Для створення web-додатку було використано **Django** – високорівневий Python фреймворк для розробки web-додатків. Він є безкоштовним та має відкритий вихідний код. Django дозволяє швидко розробляти web-додатки будь-якого розміру та підтримує читку та прагматичну структуру проекту.

Django заохочує вільне зв'язування і строге розділення частин додатка. Завдяки цьому можна легко вносити зміни в одну конкретну частину додатка без збитку для інших частин.

Django було обрано через наступні фактори:

– підтримка концепції MTV (Model - Template - View). Django додаток дозволяє розмежувати бізнес-логіку проекту від логіки відображення та збереження даних (рис. 2.3);

– вбудований ORM (Object-relational mapper). Механізм об'єктно-орієнтованого відображення, що дозволяє автоматизувати типові взаємодії з базою даних;

– висока швидкість роботи.

Django-додаток може витримувати високе навантаження, а також має вбудовані можливості кешування і розподілу навантаження [17].

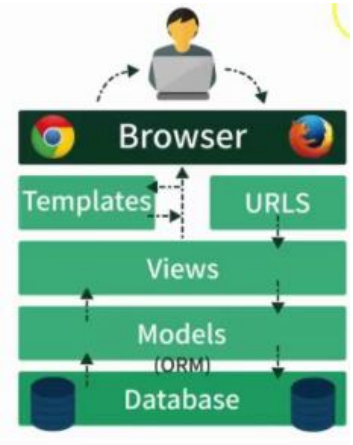


Рис. 2.3. Зв’язок між частинами Django проєкту

Для побудови взаємодії з клієнтом було використано бібліотеку Django Rest Framework, що надає можливості серіалізації та десеріалізації даних та приведення їх у формат JSON для відправлення та отримання даних з клієнта.

Для виконання обчислень було використано **SciPy** – відкриту бібліотеку, що включає високоякісні наукові інструменти для мови програмування Python. Вона включає в себе функції для виконання оптимізації та підтримує велику низьку методів оптимізації: COBYLA, SLSQP, trust-constr та інші [18].

Для реалізації процесу оптимізації було використано метод SLSQP, що приймає та оперує наступними параметрами:

– func – цільова функція;

– constraints – набір обмежень для виконання оптимізації;

– bounds – граничні значення для кожної змінної;

– maxiter – максимальна кількість ітерацій (за замовчуванням 100);

– ftol – допустима похибка обчислень (за замовчуванням 1.0E-6);

– eps – розмір кроку, що використовується для чисельної апроксимації Якобіана. За замовчуванням дорівнює кореню квадратному з ε (найменшому позитивному числу, що задовольняє нерівність 1 + eps > 1).

Для реалізації бази даних було обрано PostgresSQL – об'єктно-реляційну систему керування базами даних. Вона дозволяє створювати, зберігати і витягати складні структури даних. PostgreSQL підтримує uuid, грошовий, геометричний, бінарний типі, перерахування мережеві адреса, бітові рядки, текстовий пошук, xml, json, масиви, композитні типі та діапазони. Є можливість створювати власні типи даних.

PostgreSQL представляє такі функціональні можливості, як транзакції, вкладені запити, представлення, посилальна цілісність – зовнішні ключі, складні блокування, типи, визначувані користувачем, спадковість, правила, послідовності, контроль цілісності, реплікації, загальні табличні вирази й рекурсивні запити, аналітичні функції, перевірка сумісності версій, протокол поділюваних блокувань, вбудовану підтримку SSL і Kerberos [19].

Головною мовою для реалізації логіки на боці web-клієнту є JavaScript.

JavaScript – це скрипкова мова програмування з динамічною типізацією. Вона підтримує як функціональних, так і об'єктно-орієнтовний парадигми програмування. Є реалізацією стандарту ECMAScript [20, 21]. Надає можливості реагувати на дії користувача під час роботи з web-сторінкою, динамічно керувати вмістом структури сторінки, виконувати асинхронні операції, такі як відправлення запитів на сервер, встановлення таймерів і т.п.

Для роботи з великими масивами коду відсутність типізації викликати певні проблеми, серед яких складнощі у дослідженні структури даних та необхідність спостереження за небажаною конвертацію типів. Тому компанією Microsoft було розроблено надбудову над мовою JavaScript – TypeScript.

TypeScript розширює функціонал JavaScript, надає можливості типізації та підтримує використання класів за класичними принципами ООП [22]. Код на TypeScript після компіляції перетворюється на оптимізований JavaScript для підтримки сумісності з браузерами.

TypeScript має наступні переваги:

– типізація – можливість вказувати типи змінних, аргументів, повертаємих даних і т.п. TypeScript надає низьку базових типів: string, number, boolean і т.п. Типізація корисна для аналізу, прозорості, рефакторингу, оптимізації реалізуємого коду. Також типізація даних дозволяє зменшити кількість помилок при розробці;

– розширення роботи з ООП – можливість створення інтерфейсів, класів, реалізація інкапсуляції, наслідування, поліморфізму;

– підтримка підключення модулів.

Таким чином TypeScript є зручною мовою програмування для проектів середніх та великих розмірів.

HTML (HyperText Markup Language) – спеціальна мова розмітки гіпертексту. Є стандартом для побудови web-сторінок. Web-сторінка будується з певного набору HTML-тегів. Теги вкладуться один в одного за принципом «дерева» та складають DOM (Document Object Model) – об'єктну модель документа (рис. 2.4.).

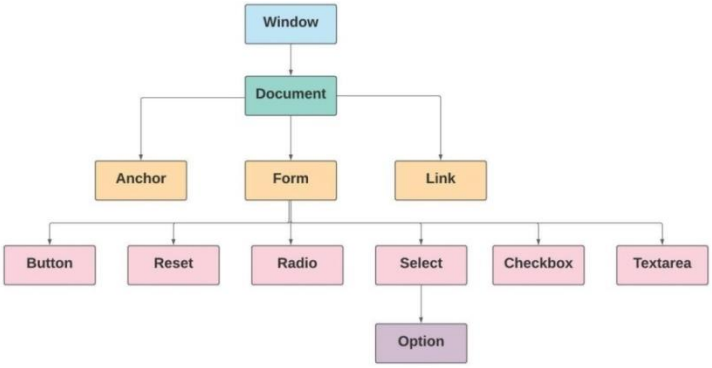


Рис. 2.4. Приклад DOM-дереву web-сторінки

Теги розділяються за типами та призначенням: для структуризації тексту, виводу мультимедійної інформації, побудови інтерактивних елементів форм, розділення сторінки на семантичні зони.

Для стилізації та збільшення зручності роботи з інтерфейсом web-сторінки використовується **CSS** (Cascading Style Sheets) – каскадні таблиці стилів, мова опису стилів документу. Дозволяє розділити зміст та оформлення сторінки, задавати позиції, відступи, кольори, границі елементів, стилізувати текст, задавати анімацію елементів і т.п.

Для пришвидшення та збільшення зручності розробки для існує низька препроцесорів CSS, що дозволяють більш гнучко задавати та працювати з описом стилів. Для сприйняття браузерами препроцесору компілюється у звичайний CSS [23].

Для опису стилів додатку було обрано препроцесор **SCSS**. Він дозводяє використовувати змінні, функції умови, цикли, надає більш зручний синтаксис взаємодії селекторів, підтримує вкладеність селекторів (рис. 2.5).

SCSS

```scss
section {
    height: 100px;
    width: 100px;

    .class-one {
        height: 50px;
        width: 50px;

        .button {
            color: #074e68;
        }
    }
}
```

CSS

```css
section {
    height: 100px;
    width: 100px;
}

section .class-one {
    height: 50px;
    width: 50px;
}

section .class-one .button {
    color: #074e68;
}
```

Рис. 2.5. Різниця між описом стилів на SCSS та CSS

Для реалізації клієнт частини web-додатку було обрано **Angular**. Це JavaScript фреймворк, що пропонує зручну систему для розробки односторінкових додатків. Перевагою даного типу web-додатків є завантаження основної структури додатку лише один раз при відкритті сторінки, після чого додаткова інформація завантажуються за потребою. При необхідності показати зміни даних, перемальовується лише потрібна частина інтерфейсу, що дозволяє оптимізувати проходження трафіка між клієнтом та сервером.

Angular використовує TypeScript для реалізації логіки, HTML – для створення розмітки та CSS або його препроцесори – для стилізації додатку [24].

Особливостями Angular є:

– модульно-компонентна архітектура. Додаток складається з модулів, кожен з яких містить певний набір компонентів. Компонент представляє елемент інтерфейсу, що включає структуру, стилі та логіку цього елементу. Компоненти вкладаються один в одного за принципом дерева, мають можливість взаємодіяти між собою за допомогою передачі параметрів у дочірні компоненти та прослуховування подій у компонентах верхнього рівня;

– можливість навігації між сторінками: опису який самий компонент відображається за певним маршрутом, створювати вкладену або паралельну навігацію, отримання у компонентах даних щодо поточного маршруту;

– робота з формами: реактивне зв’язування елементів форм з програмним кодом, динамічна реакція на зміни даних, перевірка на помилки та вивід повідомлень про некоректно введені дані, перевірка стану форму і т.п.;

– «ліниве» завантаження модулів при переході на певний маршрут для оптимізації роботи додатку [25].

Angular було обрано через велику базу можливостей роботи з формами вводу даних та зручність розбиття додатку на модулі. На рис. 2.6. представлено приклад структури Angular з описом взаємодії між навігацією (Routes) та компонентом (Component). Компонент може містити інші компоненти та

HTML-шаблон, а також код, який взаємодіє з шаблоном та зовнішніми сервісами.

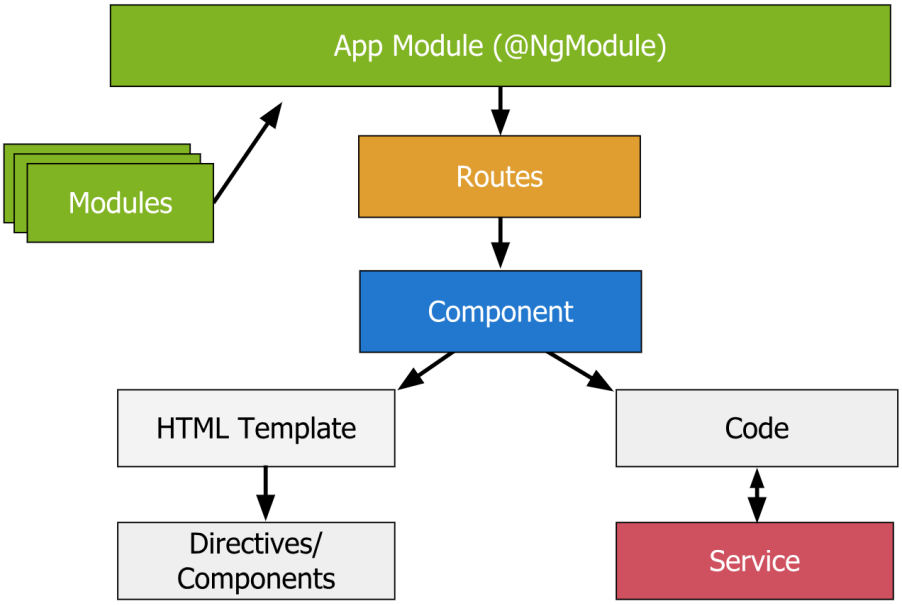


Рис. 2.6. Приклад структури Angular додатку

Для розробки інтерфейсу було використано бібліотеку Angular Material, що надає низьку готових компонентів, які можна налаштовувати та описувати взаємодію між ними та власними компонентами. Angular Material надає такі елементи як: таблиці, кнопки, діалогові вікна, кнопки, іконки, елементи форм, блоки з переходами і т.п (рис. 2.7).

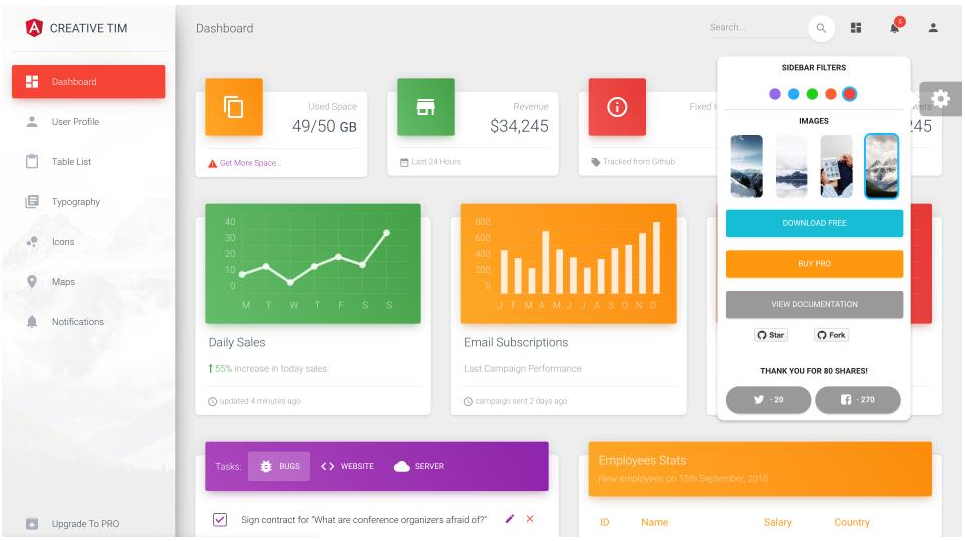


Рис. 2.7. Приклад інтерфейсу, побудованого за допомогою компонентів Angular Material

Для відображення графічної інформації було використано бібліотеку **Ng2 Charts**. Вона надає низьку діаграм та графіків різних типів, з можливістю налаштування, відображення та взаємодії з даними (рис. 2.8).

Рис. 2.8. Види діаграм та графіків у бібліотеці Ng2 Charts

Бібліотеку було обрано через зручність використання у Angular додатку та гнучкість налаштування.

2.5. Опис структури програми та алгоритмів її функціонування

Приклад структури каталогів web-серверу представлено на рис. 2.9.

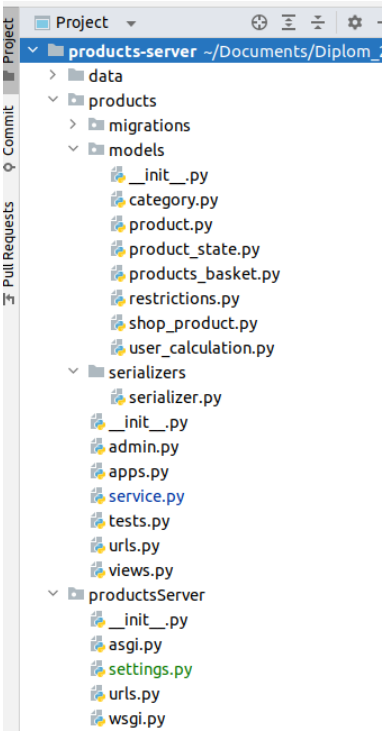


Рис. 2.9. Структура проекту web-серверу

Web-сервер складається з двох основних модулів: productsServer та products.

Модуль productsServer містить файли з загальних налаштувань проекту та файл з глобальною маршрутизацією:

– settings – загальні налаштування системи, опис зв’язку з базою даних, мова інтерфейсу, підключення зовнішніх модулів, параметри перевірки доступів;

– urls – файл з глобальною маршрутизацією. Прив’язка модулів до маршрутів, за якими буде виконуватися їх обробка.

Модуль products складається з наступних частин:

– models – моделі системи. Опис таблиць у базі даних через класи програмної системи;

– serializers – класи серіалізації моделей. Опис того як сутності системи мають бути переведені у JSON формат;

– admin – файл з налаштуваннями того, які сутності системи мають бути представлені у інтерфейсі адміністратора;

– service – файл з функціями для виконання розрахунків;

– urls – файл з прив’язкою маршрутів до класів, що обобляють ці маршртути;

– views – файл з класами, що виконують обробку запитів, що було надіслано від клієнтів, виконують обробку вхідних параметрів, підготовку даних та відправку даних до клієнта.

Діаграма головних класів модуля для роботи з продуктами та виконання оптимізації представлена на рис. 2.10.

Рис. 2.10 – Діаграма класів модуля оптимізації

Приклад структури каталогів web-клієнту представлено на рис. 2.11.

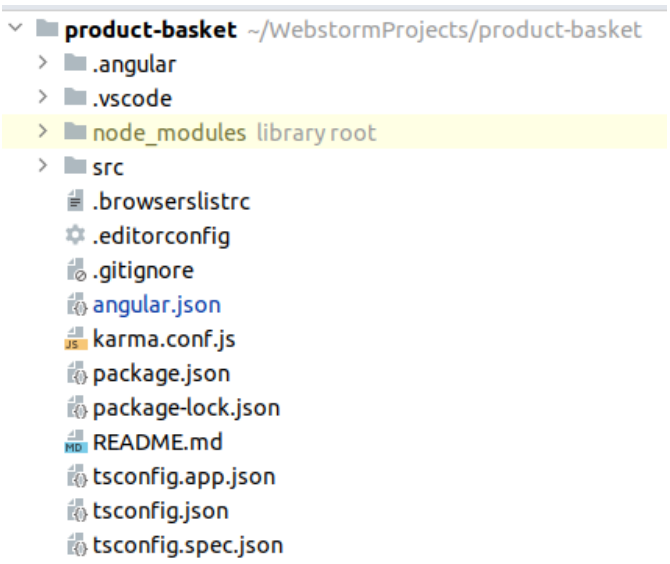


Рис. 2.11. Структура проекту web-клієнту

Основні елементи проекту web-клієнту:

– src – тека, де розміщується основний програмний код системи;

– node_modules – тека з залежностями та зовнішніми бібліотеками, необхідними для функціонування проекту;

– angular.json – файл з описом загальних налаштувань проекту;

– package.json – файл з описом залежностей проекту.

Приклад організації структури каталогів теки з програмним кодом представлено на рис. 2.12.

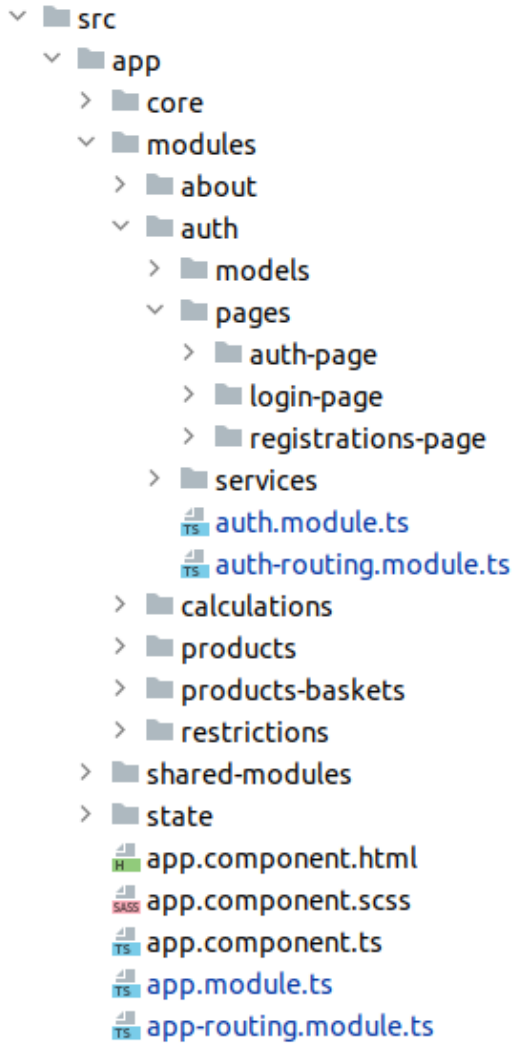


Рис. 2.12. Структура модулів web-клієнту

Основні елементи теки app:

– modules – тека, що містить тематичні модулі програмної системи: модуль авторизації, модуль для роботи з продуктами, модуль для роботи з

обмеженнями, модуль обчислень та модуль розрахованих продуктових кошиків;

– core – тека з загальними елементами системи: компонентом, що описує головне меню та заголовок додатку, сервісами для відправлення запитів та сервер, класами переривань для виконання додаткових дій перед та після відправлення запитів на сервер;

– shared-modules – тека з модулями, що містять компоненти, які використовуються у різних модулях системи;.

– state – тека з описом глобальних сховищ даних для різних модулів;

– app.component – кореневий компонент системи.

– Кожен тематичний модуль у теці modules включає наступні елементи:

– <module-name>.module – файл, що клас з загальним описом модуля та його залежностей;

– <module-name>-routing.module – файл, що клас з описом маршрутів даного модуля та їх прив’язку до компонентів відображення;

– pages – тека з головними компонентами модуля, що прив’язані до певних маршрутів;.

– components – тека з допоміжними компонентами модуля, що можуть використовуватися у цьому модулі;

– services – тека з сервісами для роботи з сервісами для цього модуля.

У базі даних зберігається дані щодо продуктів (назва, вартість, кількість, одиниця виміру, енергетичний склад), обмежень (продукт та його максимальні / мінімальні норми вживання на добу), користувачів (дані для авторизації, дані для виконання обчислень) та продуктових кошиків користувачів (загальна вартість та енергетична цінність кошику, список продуктів у кошику).

Для реалізації оптимальної структури бази даних було досліджено взаємозв’язки між цими сутностями та виконано нормалізацію бази даних до третьої нормальної форми: для представлення сутностей було виділено окремі таблиці, задано унікальні ідентифікатори сутностей, тобто первинні ключі. Для значень, що можуть відноситися до кількох записів було виділено окремі

таблиці та зв’язано за допомогою первинних ключів, виконано розрив відношень типу «багато-до-багатьох» через створення додаткових таблиць. Далі більш детально описано процес розбиття сутностей та організації зв’язків між ними.

Кожен продукт відноситься до певної категорії та одна категорія може відноситися до кількох продуктів. Таким чином сутності «Категорія» та «Продукт» пов’язані між собою відношенням один до багатьох.

Так як у магазинах може бути представлено кілька варіацій одного продукту, що можуть мати різну вартість, було виділено окрему сутність «Продукт у магазині», сутності «Продукт» та «Продукт у магазині» пов’язані між собою відношенням один до багатьох.

Кожен продукт має певну енергетичну цінність та кількість білків, жирів, та вуглеводів. Енергетична цінність продукту залежить від типу його приготування. Через це було виділено окрему сутність «Стан продукту». Так як різні варіації одного продукту в магазині можуть мати різну вартість, але однаковий енергетичний склад, сутності «Продукт у магазині» та «Стан продукту» пов’язані відношенням типу «багато-до-багатьох». Для розриву цього зв’язку було створено окрему таблицю, що містить посилання на продукт у магазині та стан продукту.

Так як продукт може мати певні обмеження у вживанні було виділено сутність «Обмеження».

На рис. 2.13 представлено основні сутності програмної системи та зв’язки між ними.

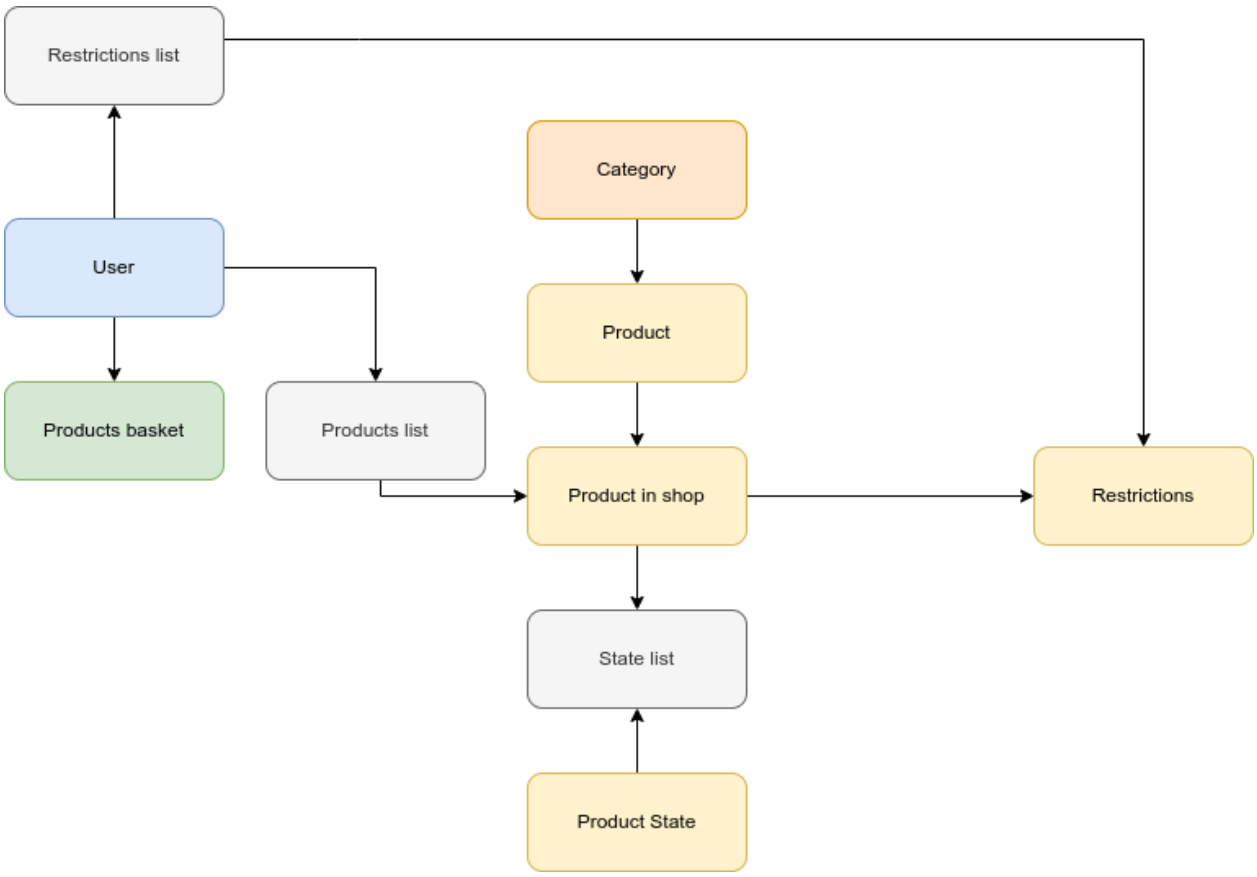


Рис. 2.13. Опис зв’язків між основними сутностями системи

Користувач може налаштовувати власні списки продуктів та обмежень.

Для збереження зв’язка між користувачем та продуктами, обмеженнями було створено сутності «Список продуктів» та «Список обмежень».

Користувач може мати кілька продуктових кошиків. Таким чином сутність «Користувач» та «Продуктовий кошик» пов’язані зв’язком один до багатьох.

Сутність «Користувач» було додатково розділено на дві сутності за логічним призначенням:

– сутність «Користувач» – містить дані для ідентифікації та авторизації користувача,

– сутність «Характеристика користувача для розрахунків» – містить дані для розрахунку добової норми калорій для користувача.

У таблиці 2.2 представлено опис сутностей користувача.

Таблиця 2.2

Опис сутностей користувача

Назва сутності	Атрибут	Тип	Опис
Користувач (User)	id	integer	ID користувача
	username	varchar(150)	Ім’я користувача
	password	varchar(128)	пароль
	email	varchar(254)	поштова скринька
	is_superuser	boolean	Позначка адміністратора
Характеристика користувача для розрахунків (User Calculations)	user id	integer	ID користувача
	height	double	Зріст користувача у сантимет-рах
	weight	double	Вага користувача у кілограмах
	years	integer	Вік користувача
	sex	varchar(2)	Стать користувача
User Calculations	activity level	varchar(2)	Рівень фізичної активності
	id	integer	ID Категорії
Продукти користувача (User Products)	user id	integer	ID користувача
	shop product	integer	ID продукту в магазині
Обмеження користувача (User Restriction)	user id	integer	ID користувача
	restriction id	integer	ID обмеження

У таблиці 2.3 представлено опис сутностей продуктів.

Таблиця 2.3

Опис сутностей продуктів

Назва сутності	Атрибут	Тип	Опис
Категорія (Category)	name	varchar(250)	Назва категорії
	id	integer	Назва категорії
Продукт (Product)	category id	integer	ID Категорії
	name	varchar(250)	Назва продукту
Продукт у магазині (Shop Product)	id	integer	ID продукту в магазині
	name	varchar(250)	Назва продукту в магазині
	price	double	Вартість продукту
	amount	double	Кількість продукту
	unit	varchar(2)	Одиниця вимірю
Стан продукту (State)	id	integer	ID стану
	state	varchar(5)	Стан продукту
	dish name	varchar(250)	Назва продукту у даному стані
	energy	double	Кількість енергії на 100 грам
	carbohydrates	double	Кількість вуглеводів на 100 грам
	proteins	double	Кількість білків на 100 грам
	fats	double	Кількість жирів на 100 грам

У таблиці 2.4 представлено опис сутностей продуктового кошику.

Таблиця 2.4

Опис сутностей продуктового кошику

Назва сутності	Атрибут	Тип	Опис
Стан продукту (Product state)	product state id	integer	ID стану
	product id	integer	ID продукту
Обмеження (Restriction)	id	integer	ID обмеження
	product id	integer	ID продукту
	comparator	varchar(2)	Порівняння (більше, менше, рівно)
	amount	double	Кількість продуктів
	unit	varchar(2)	Одиниця вимірю
Продуктовий кошик (Product brackets)	user id	integer	ID користувача
	name	varchar(250)	Назва кошику
	creation date	date-time	Дата створення
	period	integer	Кількість днів, на який розраховується кошик
	max sum	double	Максимальна вартість кошику
	products	jsonb	Результати оптимізації

ER-діаграма бази даних представлена на рис. 2.14.

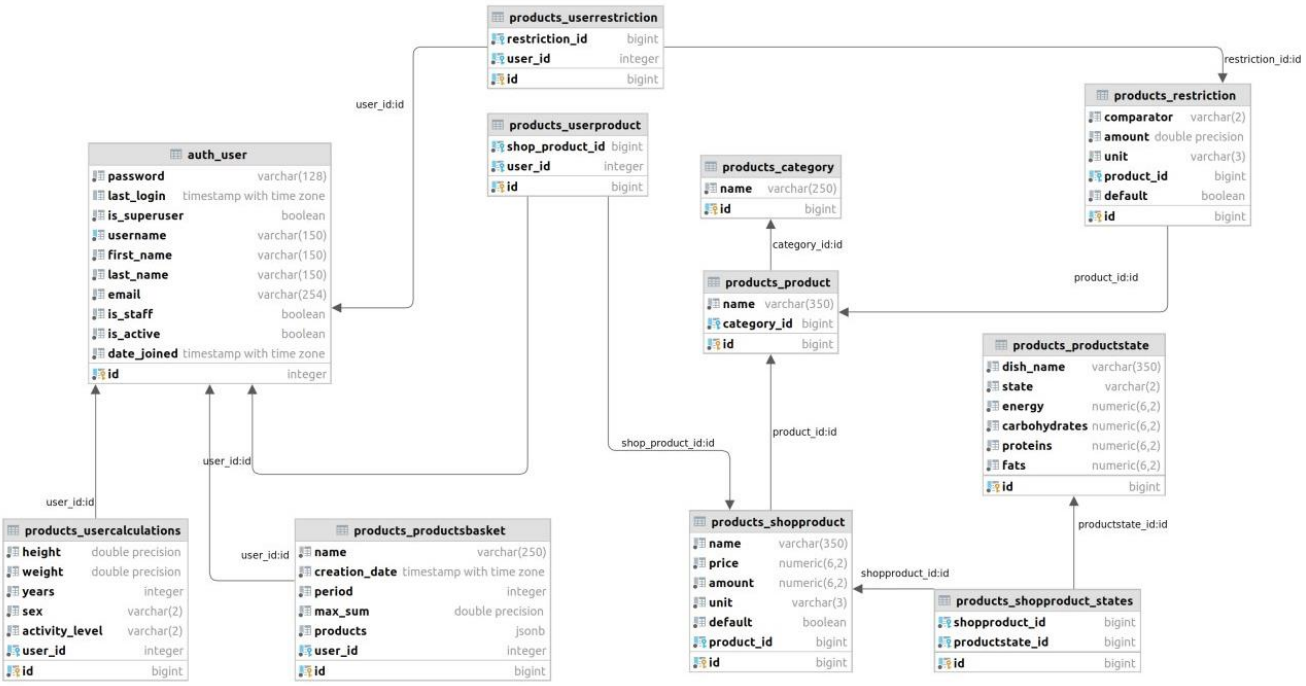


Рис. 2.14 – ER-діаграма бази даних

Процес обміну даних між web-клієнтом та web-сервером відбувається за наступним алгоритмом (рис. 2.15).

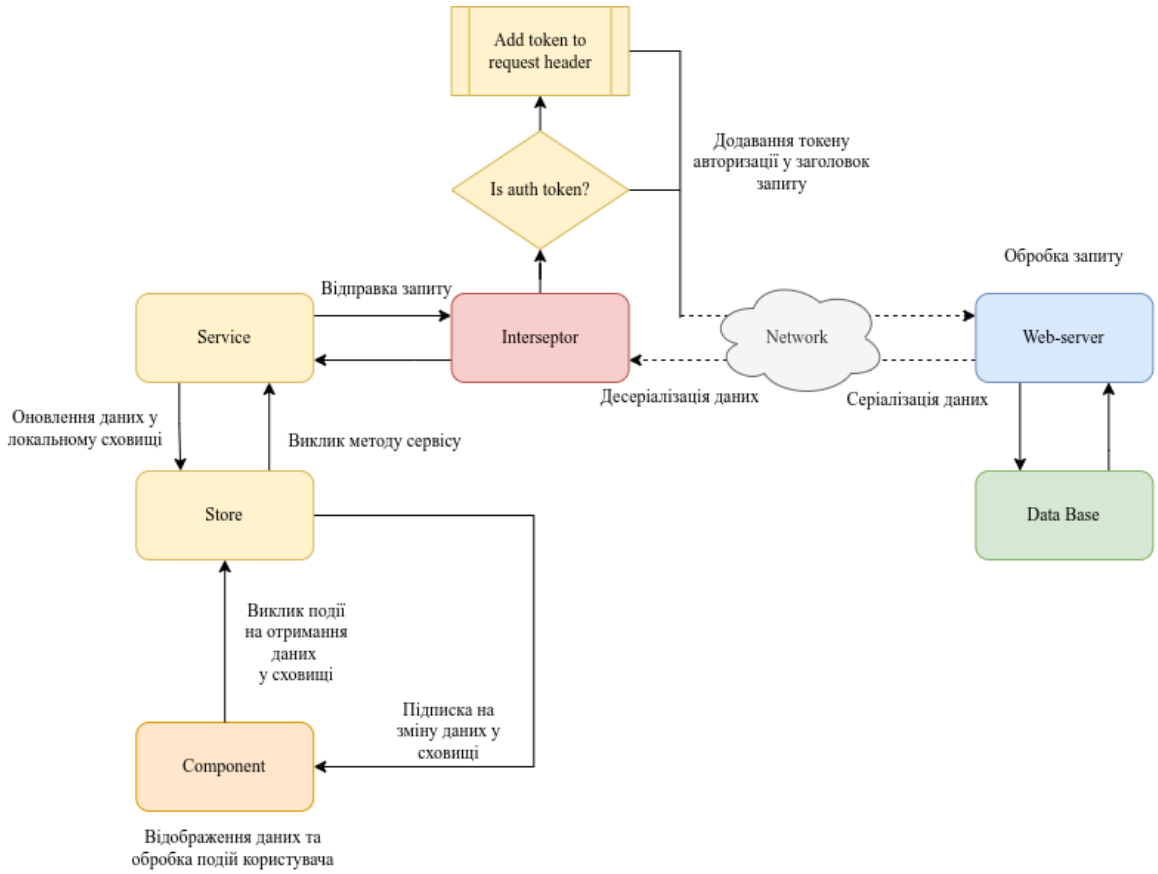


Рис. 2.15. Процес обміну даними між клієнтом та сервером

Процес введення даних для розрахунків та отримання результатів обчислень представлено у діаграмі активності на рис. 2.16.

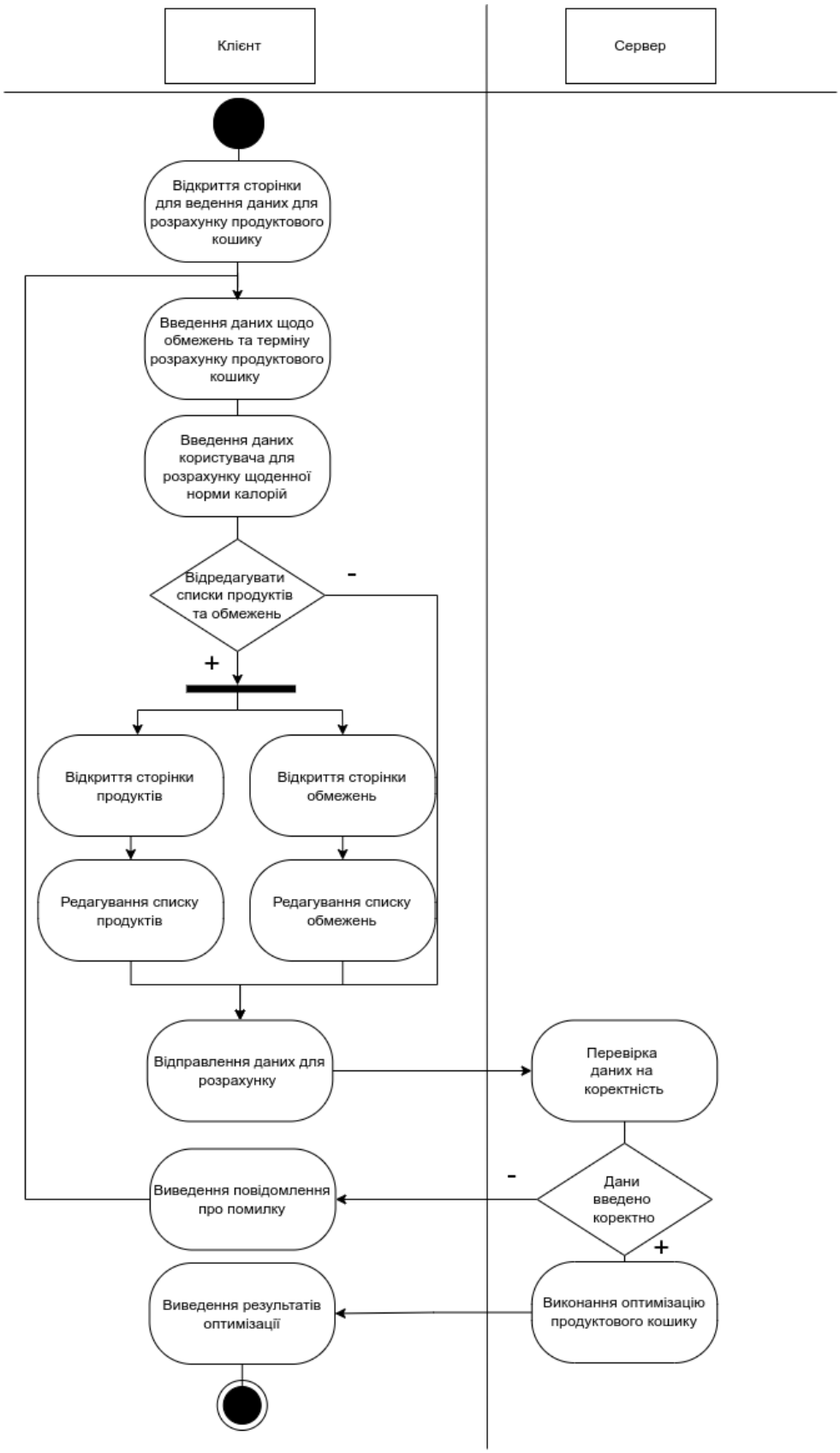


Рис. 2.16. Діаграма активності процесу введення даних для розрахунків

При відкритті сторінки чи обробці дії користувача компонент web-клієнту викликає подію для зміни даних у локальному сховищі. Сховище обробляє вхідну подію та викликає метод відповідного сервісу для запиту даних з серверу.

Перед відправкою запита на сервер, відбувається перевірка чи існує локально збережений токен авторизації користувача. Якщо так, токен додається у заголовок запиту, для того щоб сервер мав можливість ідентифікувати поточного користувача.

На сервері запит обробляється, за необхідність відбувається звернення до бази даних. Результат запиту відправляється на клієнт у форматі JSON, дані серіалізуються перед відправкою та десериалізується при отриманні. Локальне сховище отримує дані та оновлює відповідну частину сховища.

У компоненті задано підписку на зміну даних у локальному сховищі, таким чином компонент отримує та відображає результати запиту.

Для ідентифікації авторизованого користувача використовується обмін JSON web-токеном (JWT) між клієнтом і сервером. При авторизації користувача у системі сервер генерує JWT токен та відправляє його клієнту.

Приклад запиту з відправкою токену на сервер представлено на рис. 2.17.

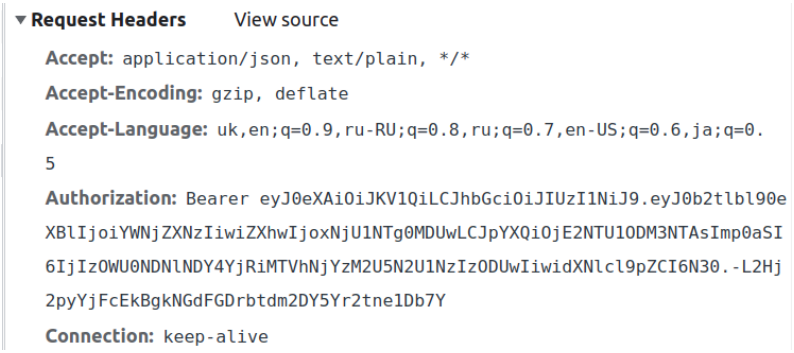


Рис. 2.17. Відправка JWT токену у заголовку запиту

Токен складається з трьох частин: заголовку, де закодовано об'єкт, що описує токен та алгоритм його хешування; корисного навантаження, що формує основу токену та є пропорційним збереженої в ньому кількості даних;

підпису, що генерується на основі корисного навантаження та заголовку, за допомогою цієї частини відбувається перевірка коректності JWT токену [26].

Клієнт отримує токен та зберігає у локальному сховищі браузеру. При подальших запитах до серверу токен додається до заголовку Authorization.

Програмний код головних модулів та класів web-сервера та web-клієнта представлено у додатку А.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

При обміні даних між сервером та клієнтом дані передаються у форматі JSON. Це текстовий формат даних, зручний для опису структурованої інформації, списків, об’єктів. Дані серіалізуються при відправці через web-мережу та відповідно десериалізується при отриманні на сервері та клієнті.

Користувач має можливість введення наступних даних:

– дані для реєстрації (ім’я користувача, email, пароль);

– дані для створення та редагування розрахунку щоденної норми калорій (стать, вік, вага, зріст, рівень фізичної активності);

– дані для авторизації (ім’я користувача, пароль);

– дані для створення та редагування продукту (назва продукту, категорія, вартість, кількість, одиниця виміру, стани продукту);

– дані для створення та редагування обмеження (продукт, тип обмеження, кількість);

– дані для розрахунку продуктового кошику (термін розрахунку, максимальна вартість).

При введенні даних користувачем відбуваються перевірки на коректність вхідних даних: відповідність потрібному типу, максимальні, мінімальні значення, перевірка на відповідність формату. Аналогічні перевірки відбуваються на боці серверу перед збереженням даних у базі.

2.7. Опис розробленого програмного продукту

Розроблений програмний додаток являє собою комбінацію web-серверу та web-клієнту, комунікація яких відбувається через протокол HTTP/S. Додаток надає користувачам можливість введення даних для розрахунку щоденної потреби в енергії, задавати обмеження щодо загальної вартості продуктів та їх вживання, задавати термін, на який потрібно виконати моделювання дієти, корегувати вхідний список продуктів харчування, який буде виконано для розрахунків.

За введеними користувачем даними додаток виконує розрахунок оптимального набору продуктів, що задовольняють обмеженням та мають потрібний при діабеті другого типу баланс білків, жирів та вуглеводів.

2.7.1. Використані технічні засоби

Для повноцінного функціонування додатку потрібно виділити ЕОМ для розгортання серверів Django та Angular додатків. Сервера можуть знаходитися як на одному пристрої, так і на різних.

Мінімальні технічні вимоги для записку серверу:

- накопичувач [HDD]: 500 МБ;
- кількість ядер процесору: 4;
- оперативна пам’ять [RAM]: 512 МБ.

Розробка та тестування програмного додатку було виконано на ЕОМ, що має наступні технічні характеристики:

- центральний процесор: AMD Ryzen 5 2600.
- накопичувач [SDD]: 240 ГБ
- оперативна пам'ять [RAM]: 16 ГБ.

2.7.2. Використані програмні засоби

При розробці продукту були використані програмні засобі представлені у таблиці 2.5.

Таблиця 2.5

Опис використаних програмних засобів

Назва	Призначення
WebStorm, PyCharm	Інтегровані середовища розробки
Git	Управління версіями
GitHub	Хмарне сховище для збереження програмного коду
Figma	Розробка прототипу та дизайну продукту
Docker	Розгортання та запуск серверу

PyCharm – потужний і функціональний редактор коду з підсвічуванням синтаксису, автоформатуванням і автовідступами для підтримуваних мов. PyCharm пропонує просту і потужну навігацію, допомагає при написанні коду, підтримуючи такі функції: автодоповнення, авто-імпорт, шаблони коду, перевірка на сумісність версії інтерпретатора мови і т.д.

PyCharm містить панель інструментів з командами для запуску, зупинки програми, запуску відгадчику, збереження змін проекту і т.д. Зліва екрана знаходиться дерево проекту, зручне для навігації та пошуку потрібних тек та файлів.

Внизу екрану знаходиться набір вкладок для відкриття консолі для Python, Django, SSH, відладчика та баз даних [27]. Є можливість створювати декілька терміналів для запуску команд та маніпулювання проектом. Приклад інтерфейсу IDE PyCharm представлено на рис. 2.18.

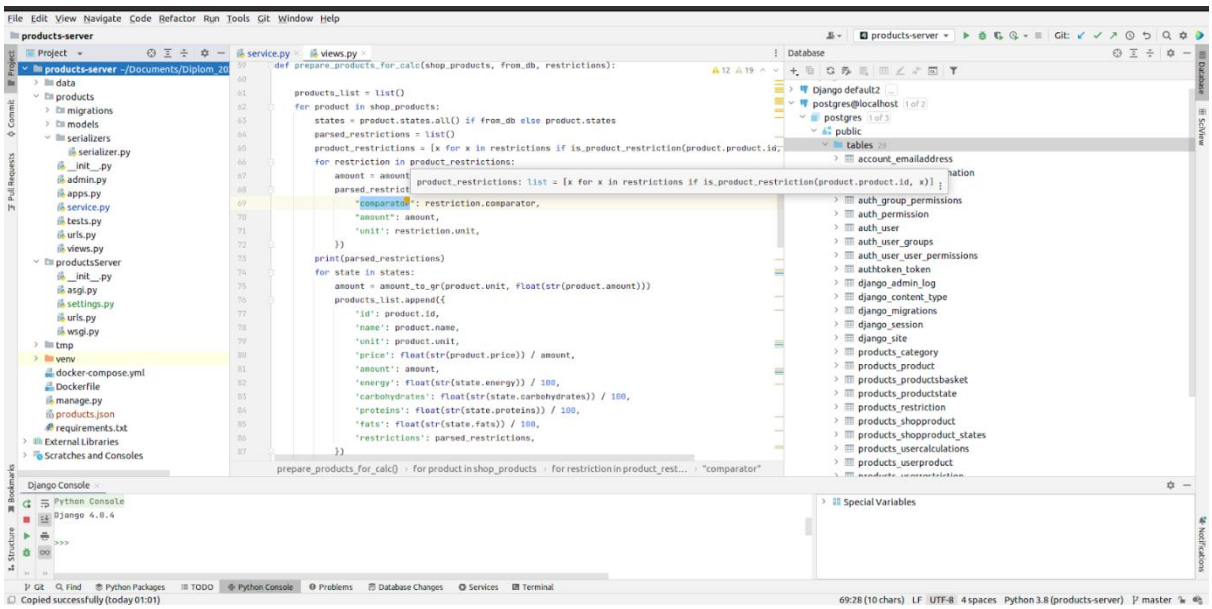


Рис. 2.18. Інтерфейс IDE PyCharm

Для перегляду поточної бази даних існує вкладка в правому кінці екрану. Вона дозволяє переглядати та відкривати таблиці БД, будувати ER-діаграми, тощо.

IDE включає потужний рефакторинг коду, який надає широкі можливості по виконанню швидких глобальних змін в проекті.

PyCharm включає редактори Javascript, Coffescript, HTML/CSS, SASS, LESS, HAML та дозволяє підключати проект к системі контролю версій. За допомогою PyCharm можна будувати UML діаграми класів та діаграми моделей Django.

WebStorm – інтегроване середовище розробки від компанії JetBrains, що підтримує мови програмування JavaScript, HTML, CSS, а також фреймворки JavaScript та різноманітні препроцесори. Пропонує повноцінний редактор коду, з можливістю рефакторингу, перегляду історії змін, пошуку використання програмних сутностей і таке інше. Пропонує набір додаткових інструментів, консолей, функціонал побудови діаграм.

Figma – хмарний сервіс для розробки дизайну та прототипів проектів. Надає гнучкий та зручний інтерфейс розробки елементів дизайну, підтримує вставлення зображень, модульну графіку, завантаження шрифтів, роботу з

анімацією та переходами між блоками. Підтримує як Web, так і Desktop версії для роботи.

При розробці програмного продукту Figma була використана для створення прототипів головних сторінок додатку. Приклади прототипів форми введення даних для розрахунків та сторінки результатів обчислень, що було розроблено за допомогою Figma, наведено на рис. 2.19. – рис. 2.21.

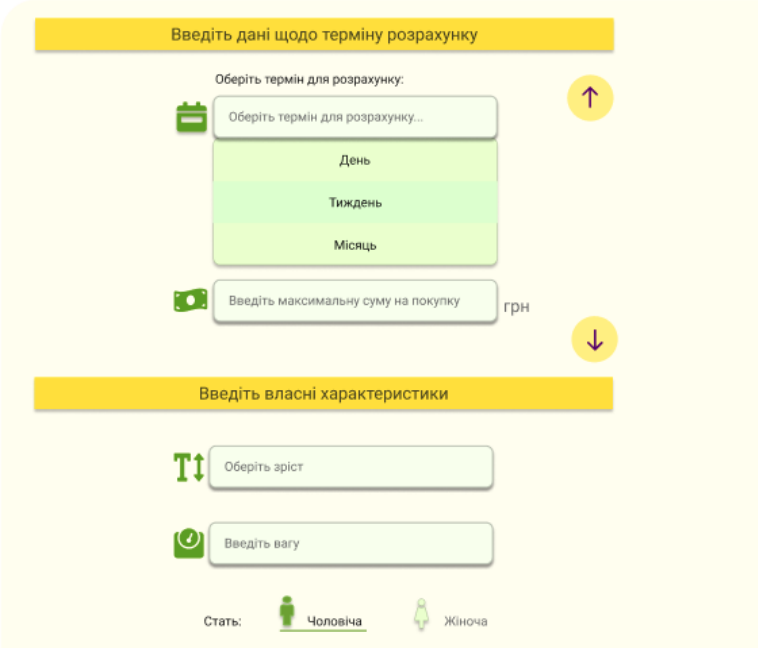


Рис. 2.19. Прототип форми введення даних

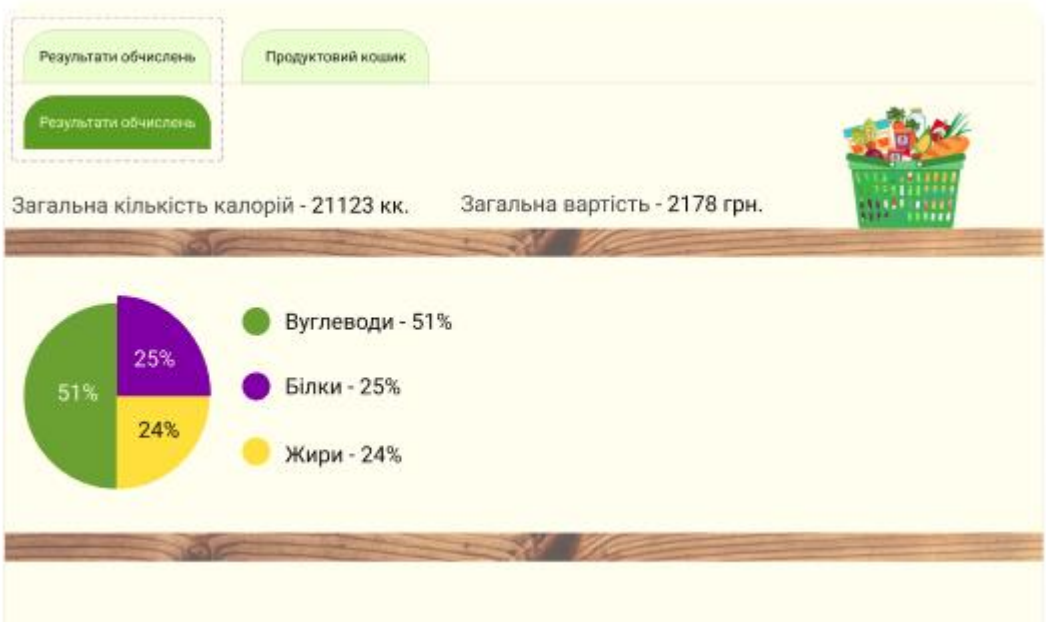


Рис. 2.20. Прототип сторінки результатів обчислень

Рис. 2.21. Прототип сторінки зі списком продуктів, які є результатом розрахунків

Git – розподілена система контролю версій, розроблена Лінусом Торвальдсом. Система пропонує гнучкий метод зберігання різних версій проекту, відстеження змін між версіями, допомагає знаходити конфлікти у файлу під час розробці, повертатися до більш старих версій і т.п.

Система надає можливість більш ефективної роботи над спільними проектами, керувати та відкачувати програмний код до більш стабільної версії, якщо є така необхідність. Також Git дозволяє дивитися історію змін у файлах, порівнювати стан файлу у різних версіях, що зручно при необхідності точної відміни змін чи аналізу коду у пошуках помилок [29, с. 12].

Git дозволяє реалізувати процес паралельної розробці та роботи з різними версіями файлу (рис. 2.22).

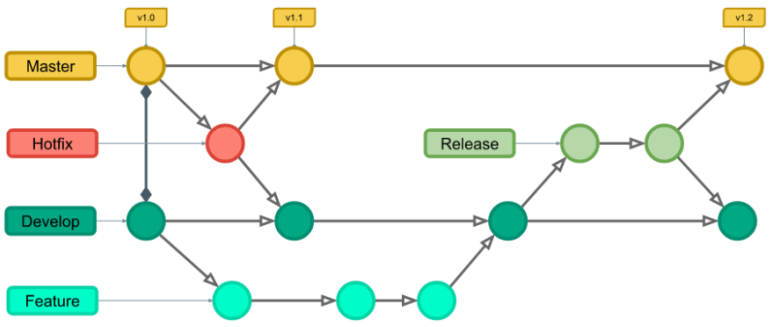


Рис. 2.22. Процес збереження версій у системі Git

Git було використано при розробці програмного продукту для фіксації результатів роботи та збереження змін у програмному коді на web-хостингу.

GitHub – web-хостінг, розроблений компанією GitHub, Inc, що пропонує рішення для зберігання та спільної розробки програмних продуктів. В його основі знаходиться підтримка системи керування версіями Git.

Сервіс підтримує проекти з відкритим вихідним кодом, надає широкі можливості користувачам, такі як: створення публічних та приватних репозиторіїв, перегляд вмісту репозиторіїв, навігація та пошук між файлами, аналіз програмного коду, налаштування безпечного з'єднання через SSL протокол і т.п.

GitHub було використано при розробці програмного продукту для збереження та публікації коду web-клієнту та web-серверу.

Docker - популярне програмне забезпечення та платформа для здійснення управління контейнерами. Дозволяє реалізувати незалежність програмного забезпечення від системи, через створення спеціальних контейнерів, які виділяються як окрема частина ядра операційної системи та запускаються, як окремі процеси [30]. Docker допомагає пришвидшити та оптимізувати процес розгортання програмних продуктів, полегшує їх масштабування та гарантує стабільність при запускі контейнерів у інших середовищах.

Docker було використано при розробці програмного продукту для розгортання, ініціалізації та запуску web-серверу та серверу бази даних.

2.7.3. Виклик та завантаження програми

Для запуску додатку потрібно виконати збірку та запуск серверів сервера та клієнта. Для початку роботи web-серверу потрібно у теці проекту виконати команду «docker-compose up», що виконає запуск web-серверу та серверу бази даних. Для початку роботи клієнту потрібно виконати команду «ng serve» у теці проекту web-клієнту. Після запуску потрібно перейти на сторінку браузеру за відповідним адресом для роботи з додатком. Локальна адреса для роботи з клієнтом за замовчуванням є http://localhost:4200. Адреса для відкриття web-

інтерфейсу для адмініструванням додатком за замовчування є http://0.0.0.0:3000/admin.

2.7.4. Опис інтерфейсу користувача

Робота з інтерфейсом користувача включає три головних сценарії: робота з додатком без реєстрації, робота авторизованого користувача, робота адміністратора додатку.

Опис інтерфейсу неавторизованого користувача

При відкритті додатку користувачу відкривається вітальне вікно, що пропонує перейти до процесу виконання розрахунків (рис. 2.23).

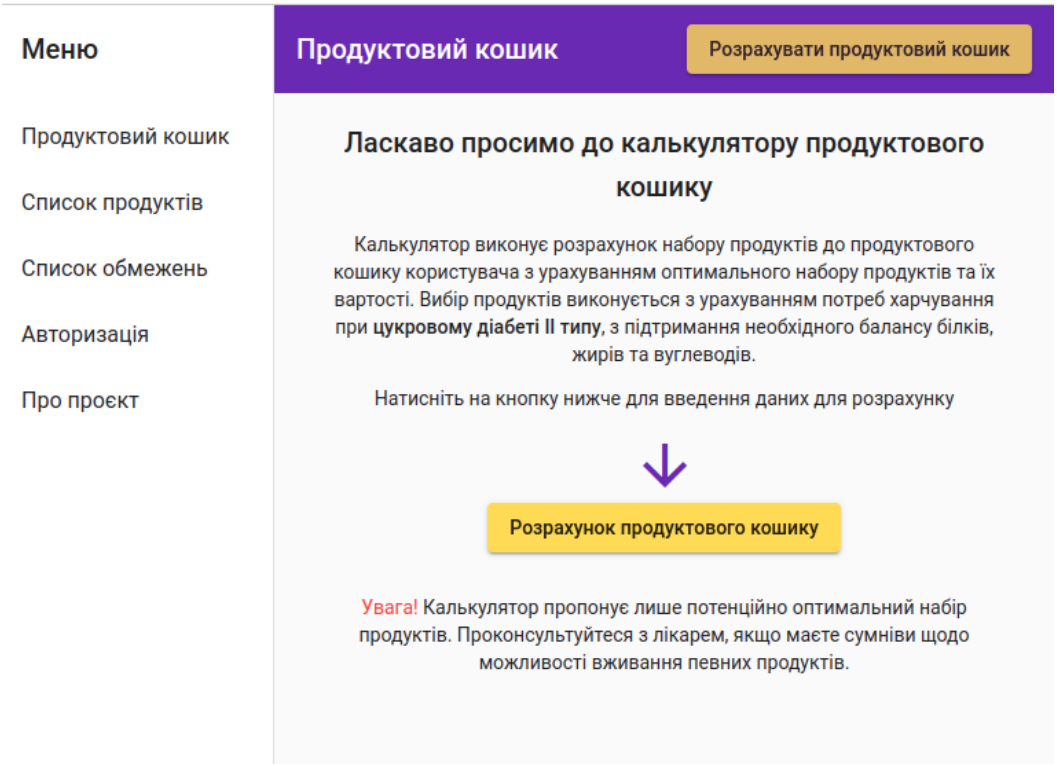


Рис. 2.23. Вітальна сторінка додатку

При натисканні на кнопку «Оптимізувати продуктовий кошик» користувача буде направлено на сторінку з формою введення інформації щодо характеристик продуктового кошику та користувача для розрахунку оптимальної кількості калорій на добу (рис. 2.24 – рис. 2.26).

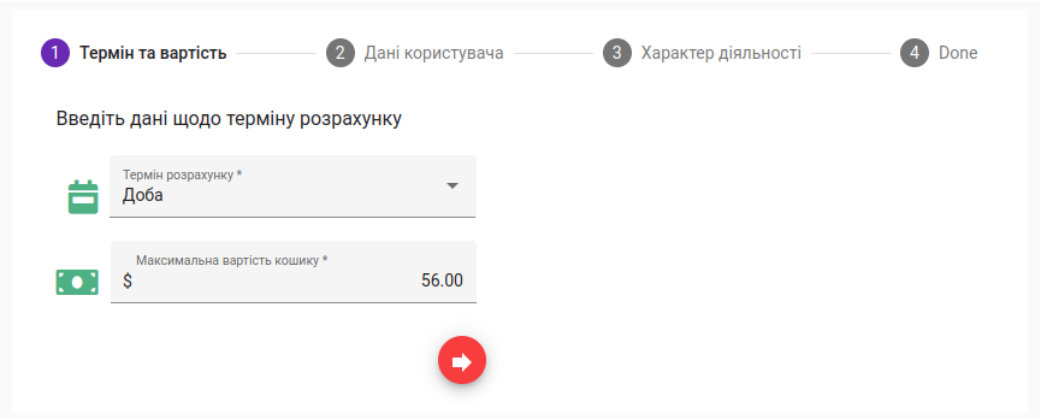


Рис. 2.24. Форма введення даних щодо термінів розрахунку та максимальної вартості продуктового кошику

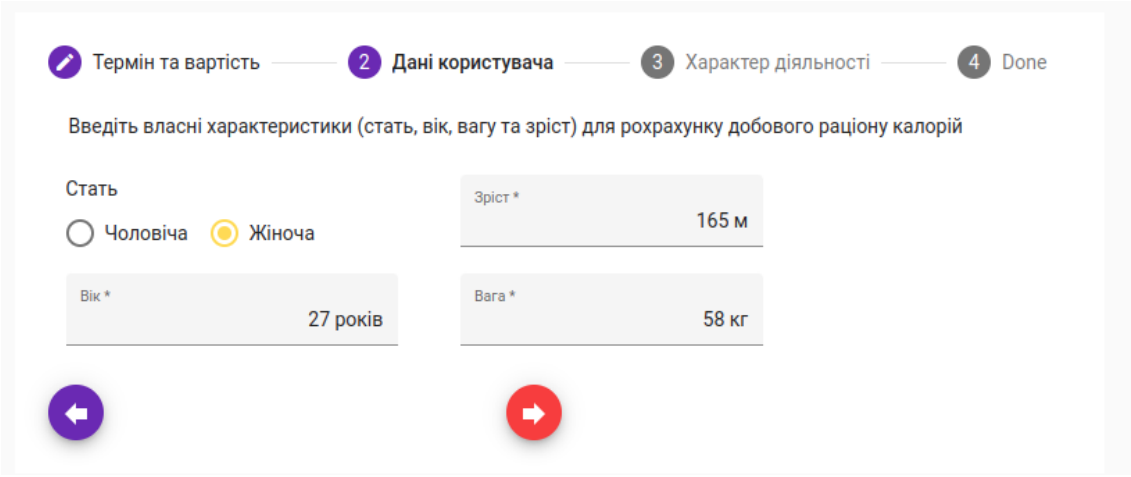


Рис. 2.25. Форма введення даних користувача для обчислення норми калорій на добу

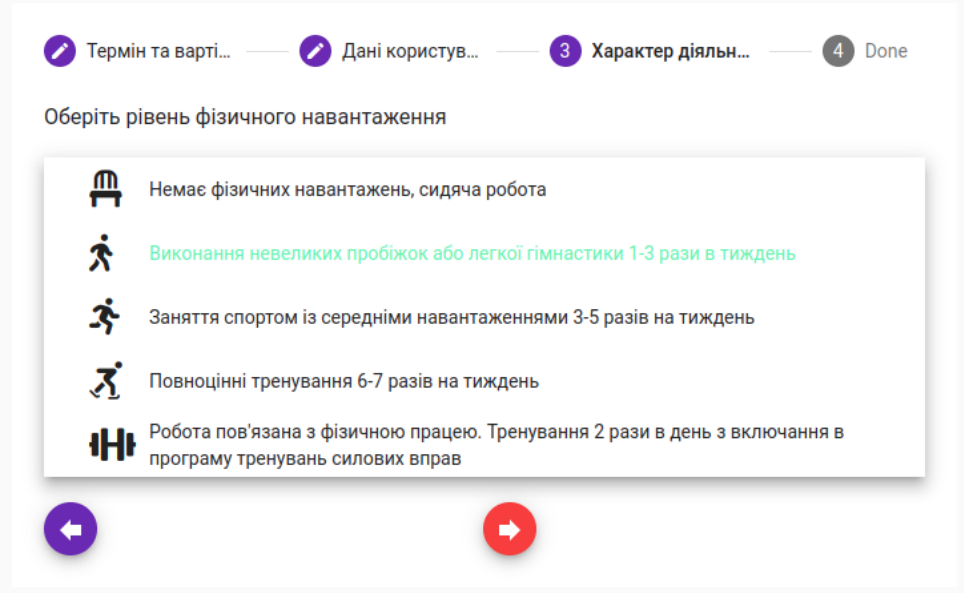


Рис. 2.26. Вибір рівню фізичної активності

Після введення даних користувача буде направлено на сторінку, де представлено загальну інформацію згідно якої будуть виконані розрахунки (рис. 2.27).

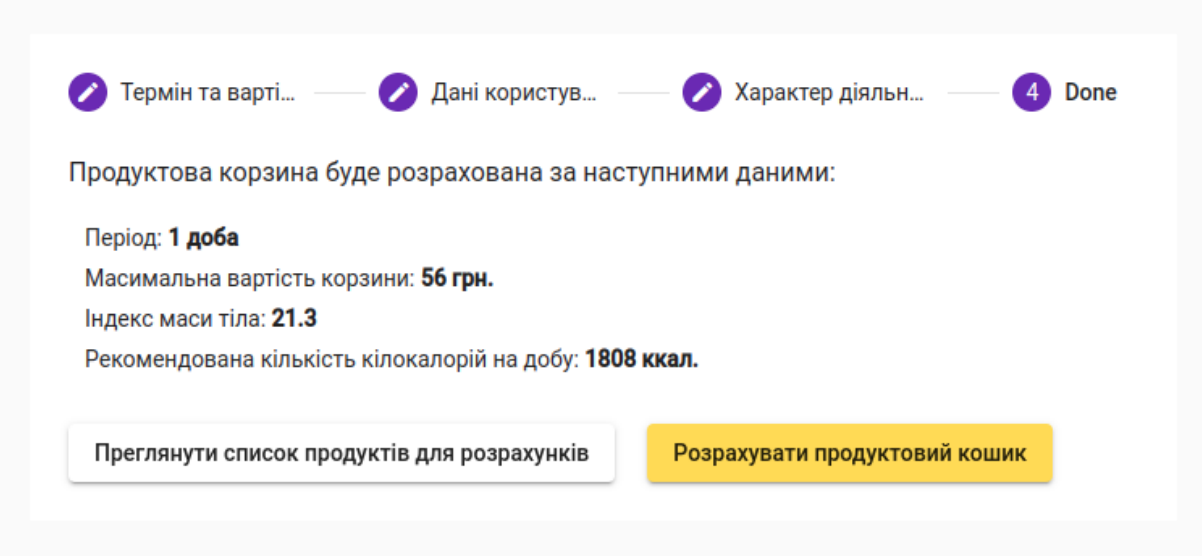


Рис. 2.27. Вивід загальної інформації для майбутніх розрахунків

Користувач може відразу перейти на обчислення продуктового кошику, а може переглянути та відкоригувати інформацію про продукти та обмеження, що будуть використано у процесі розрахунків (рис. 2.28). Для відкриття сторінки зі списком продуктів потрібно натиснути кнопку «Переглянути список продуктів для розрахунків» або натиснути на пункт у головному меню

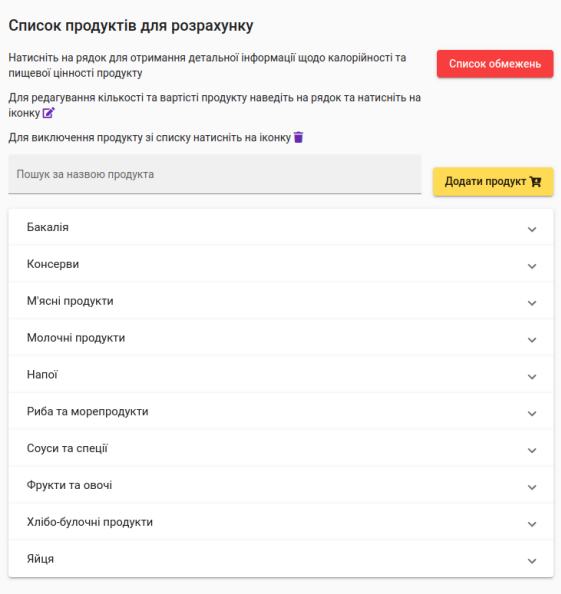


Рис. 2.28. Вивід сторінки зі списком продуктів

Сторінка містить набор продуктів, згрупованих за категоріями. При відкритті категорії буде відображено список продуктів цієї категорії. При натисканні на продукт можна також дізнатися його енергетичні характеристики на 100 грам для різних типів приготування продукту (рис. 2.29).

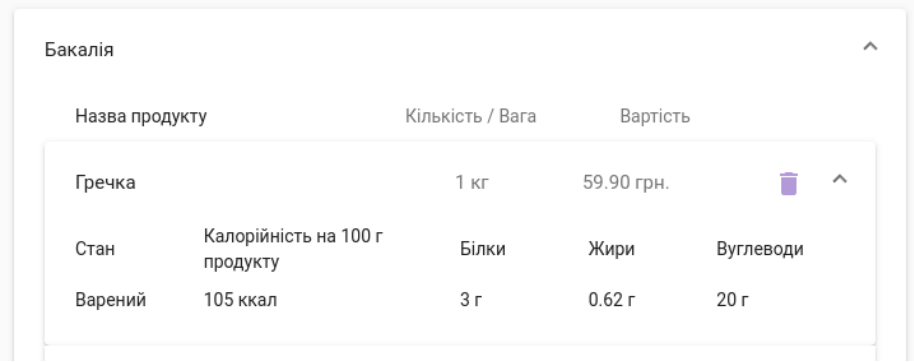


Рис. 2.29. Перегляд інформації щодо енергетичної цінності продукту

Користувач має можливість додати власний продукт, заповнивши інформацію щодо категорії, назви продукту, його ваги, вартості та енергетичних характеристик (рис. 2.30).

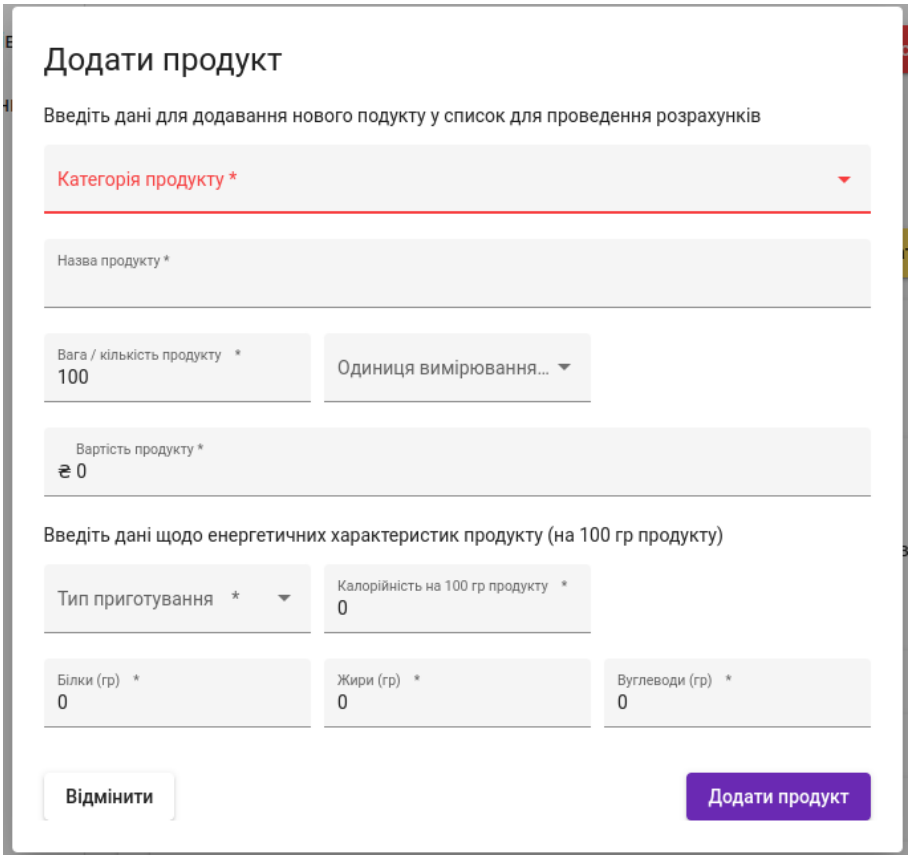


Рис. 2.30. Форма додавання продукту

На приклад заповненої форми додавання продукту зображено на рис. 2.31.

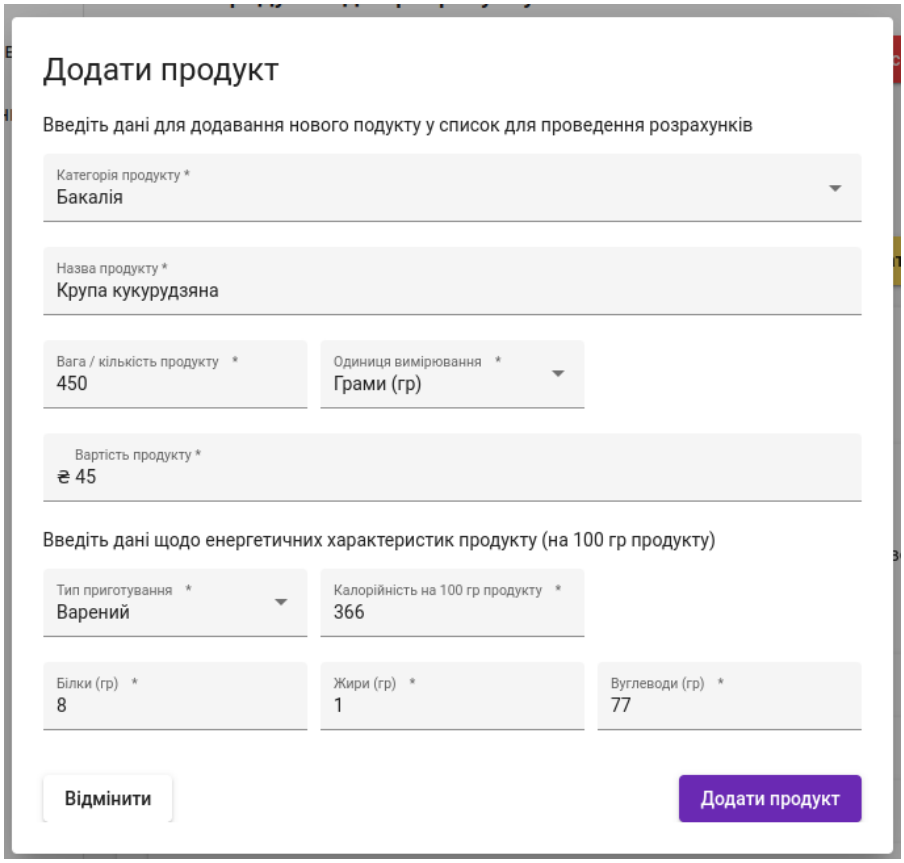


Рис. 2.31. Заповнена форма додавання продукту

Користувач має можливість видалити продукти, які не бажає включати в список проведення розрахунків. Перед видаленням продукту, необхідно підтвердити свої наміри у відповідному вікні (рис. 2.32).

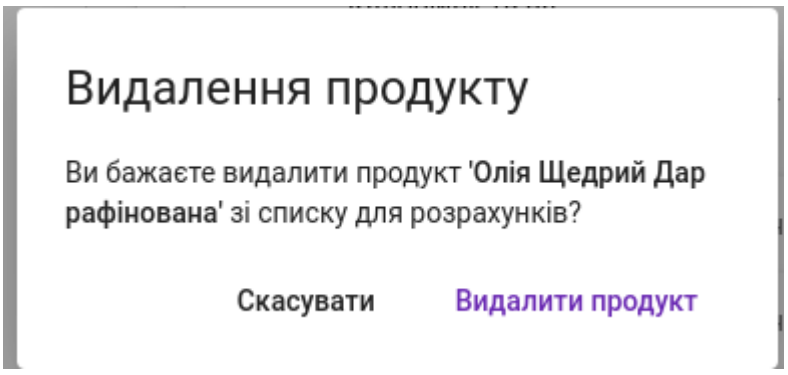


Рис. 2.32. Підтвердження видалення продукту

Також користувач може переглянути та відкоригувати список обмежень на вживання продукту. Відкрити список обмежень можна натиснувши на кнопку

«Список обмежень» на сторінці продуктів або відкривши відповідний пункт меню. Сторінка продуктів мстить форму для додавання або редагування обмеження та список існуючих обмежень (рис. 2.33).

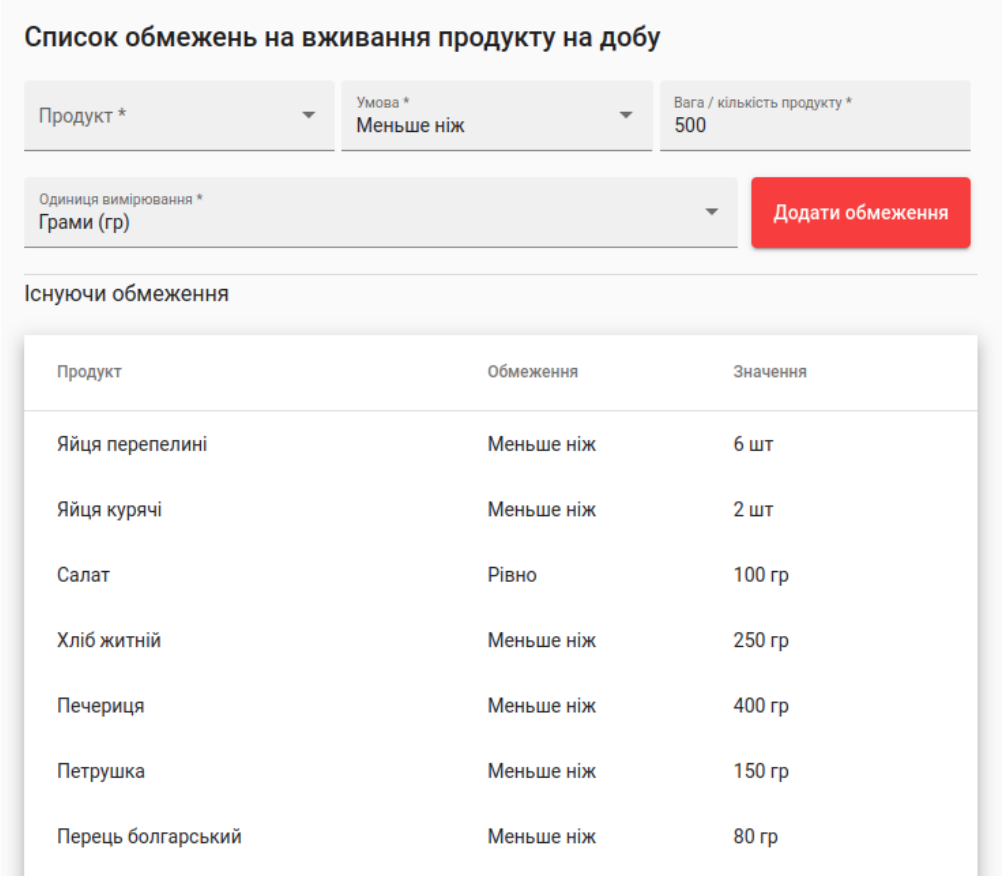


Рис. 2.33. Вивід сторінки зі списком обмежень на вживання продукту

Форма додавання обмеження пропонує обрати продукт, умову та кількість продукту (рис. 2.34). Обмеження задається на добу.

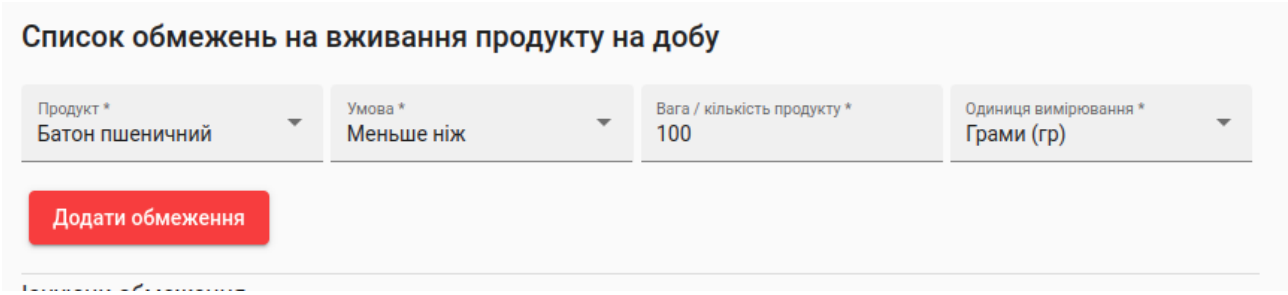


Рис. 2.34. Форма додавання обмеження

Користувач має можливість непотрібні обмеження, які не бажає включати в список проведення розрахунків. Перед видаленням обмеження, необхідно підтвердити свої наміри у відповідному вікні (рис. 2.35).

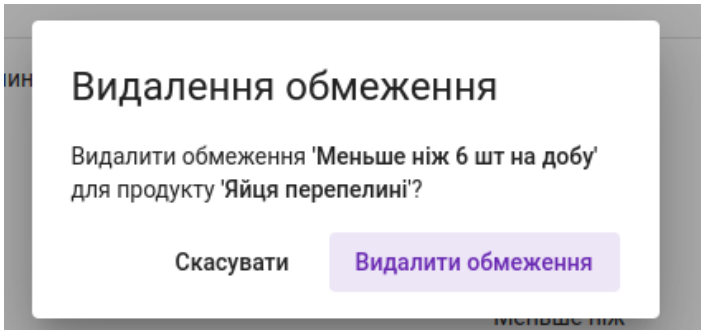


Рис. 2.35. Підтвердження видалення обмеження

Для редагування обмеження потрібно двічі клацнути на рядок обмеження та підтвердити наміри редагування у відповідному вікні (рис. 2.36).

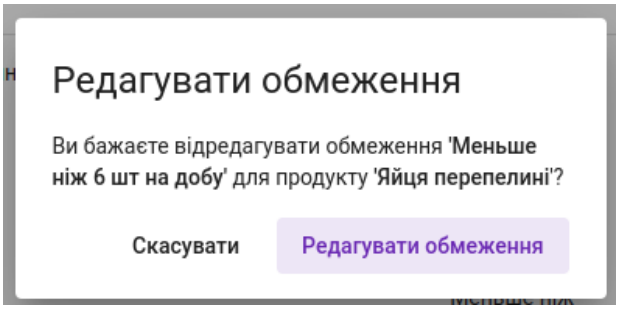
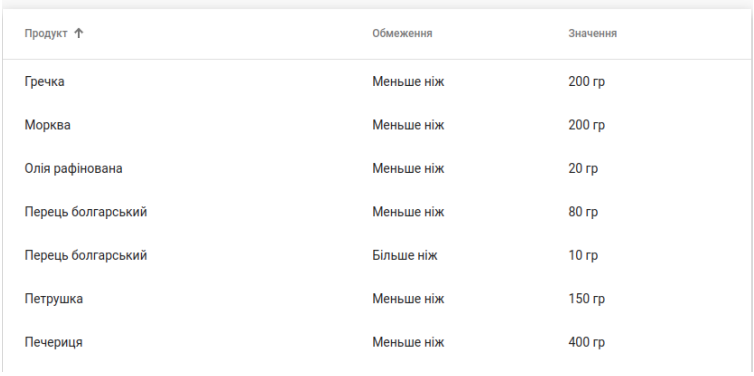


Рис. 2.36. Підтвердження редагування обмеження

Також користувач може виконувати сортування обмежень за назвою або типом умови (рис. 2.37).



Продукт ↑	Обмеження	Значення
Гречка	Меньше ніж	200 гр
Морква	Меньше ніж	200 гр
Олія рафінована	Меньше ніж	20 гр
Перець болгарський	Меньше ніж	80 гр
Перець болгарський	Більше ніж	10 гр
Петрушка	Меньше ніж	150 гр
Печериця	Меньше ніж	400 гр

Рис. 2.37. Сортування обмежень за назвою

Після проведення необхідних налаштувань користувач має натиснути на кнопку «Розрахувати продуктовий кошик» у верхньому правому куті додатку. Після цього буде проведено необхідні оптимізації та відображено результати розрахунків. Під час розрахунку відображається сторінка з процесом розрахунків (2.38).

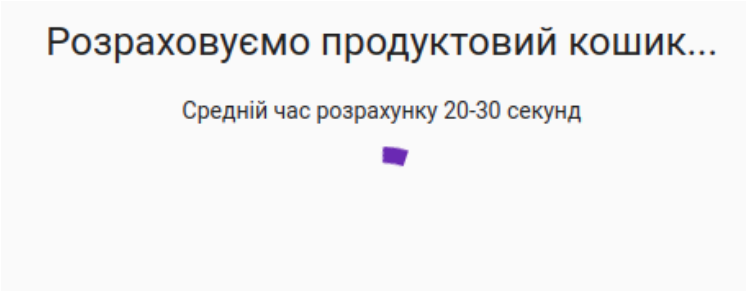


Рис. 2.38. Відображення процесу розрахунків

Результати розподілені на дві сторінки, перша з яких містить загальну інформацію щодо результатів розрахунків: загальну кількість калорій, вартість, баланс білків, житів та вуглеводів (рис. 2.39).

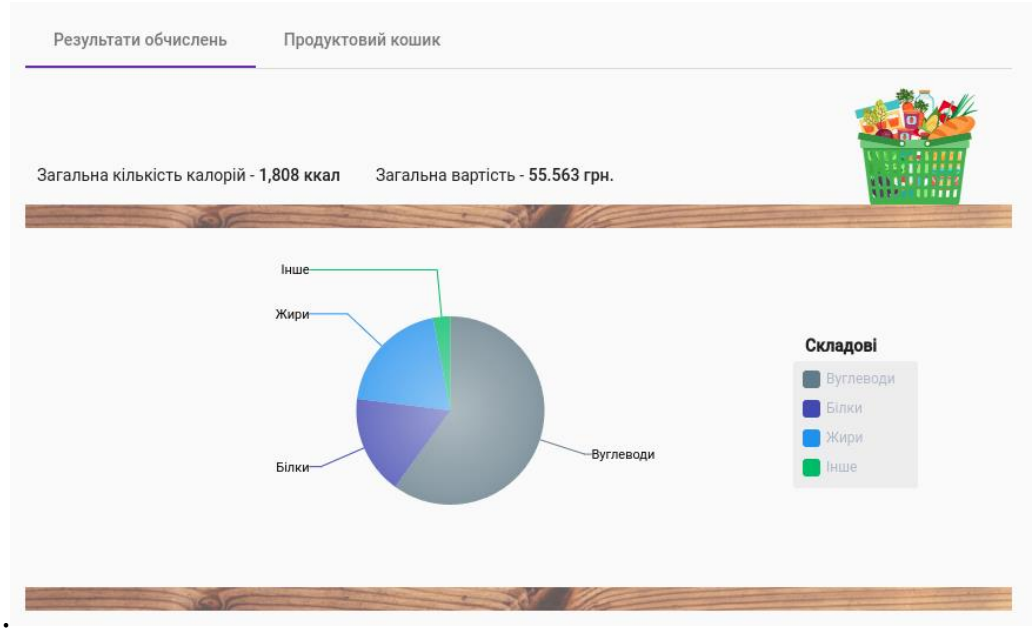


Рис. 2.39. Вивід загальних результатів оптимізації

Друга сторінка містить список продуктів з інформацією щодо їх енергетичної цінності та вартості. Також тут запропоновані можливості для друку сторінки або завантаження у вигляді файлу різного формату (рис. 2.40).

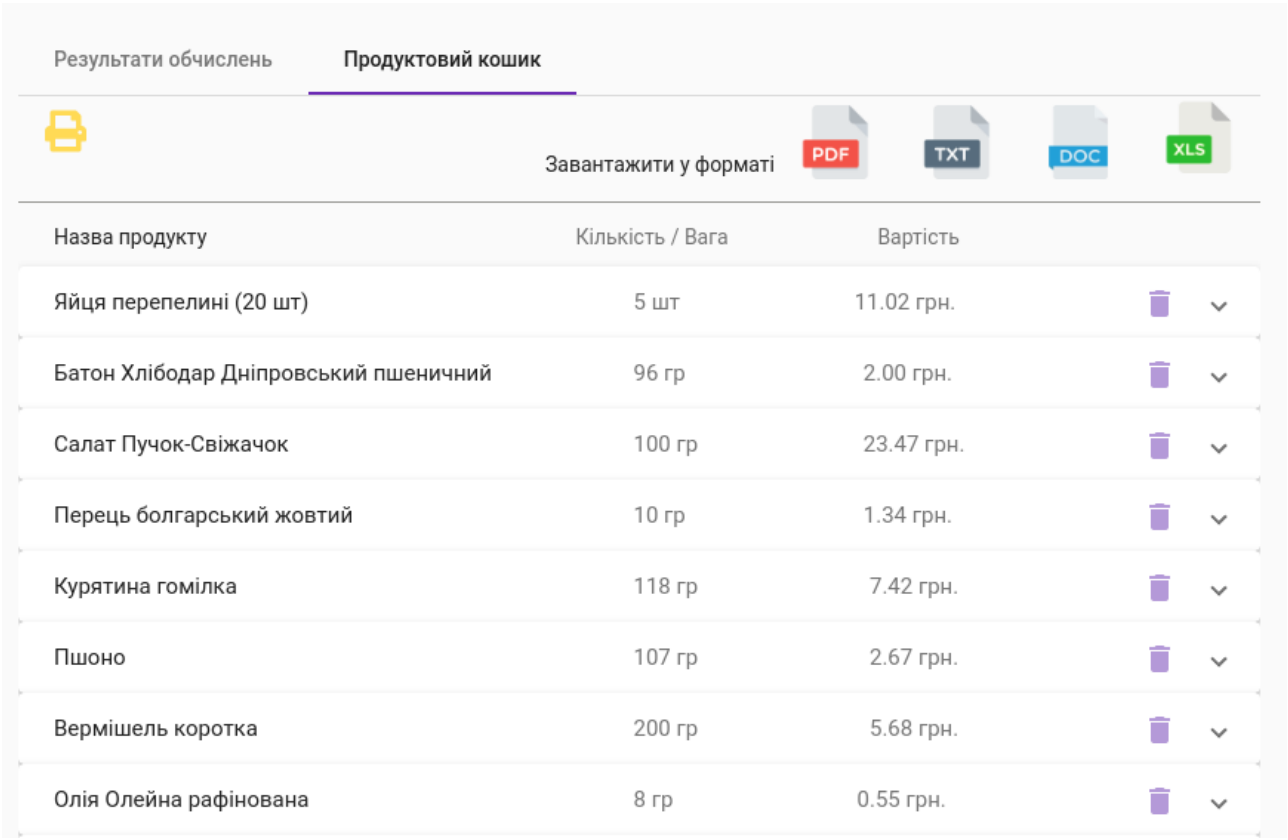


Рис. 2.40. Вивід списку оптимізованого продуктів для раціону на добу

При натисканні на рядок можна отримати інформацією щодо його харчової цінності (рис. 2.41).

Рис. 2.41. Перегляд інформації про продукт

При натисканні на символу друку у верхньому лівому рядку сторінки користувачу буде відкрито вікно для налаштування параметрів друку (рис. 2.42).

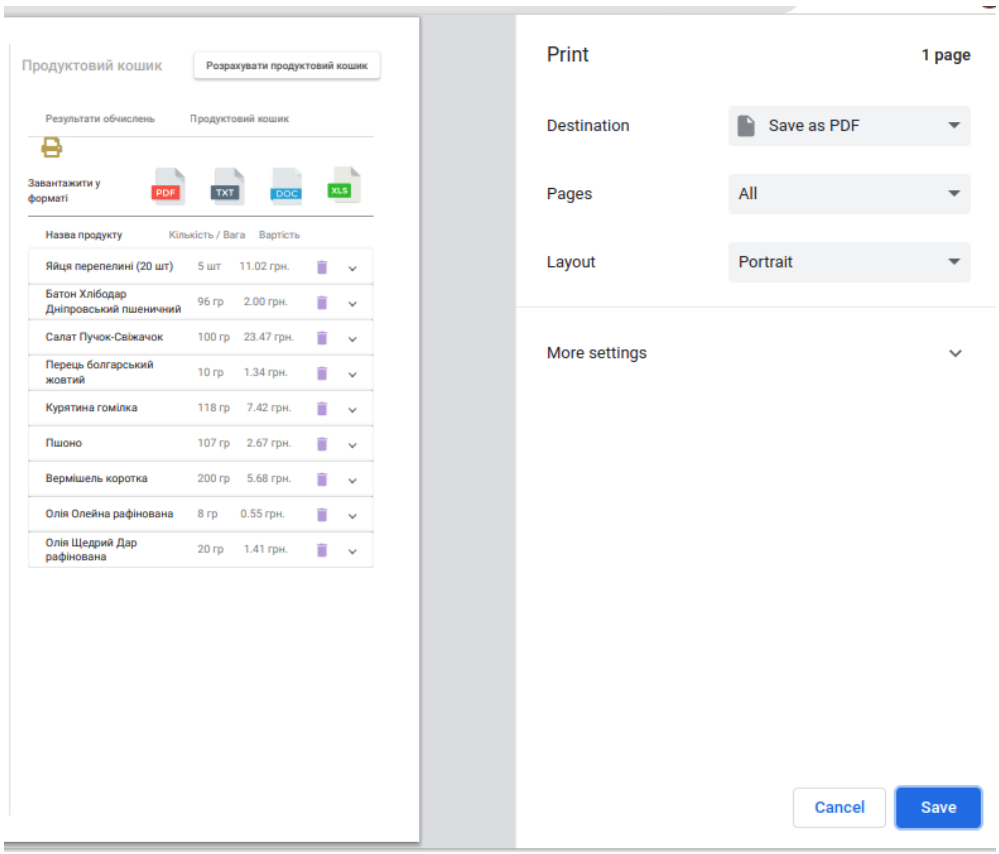


Рис. 2.42. Друк результатів розрахунків

Приклад обчислень на тиждень з максимальною вартістю кошику 670 грн. для чоловіка, 32 років, з вагою 70 кг, зростом 180 см та великим рівнем фізичного навантаження. Результати розрахунків представлено на рис. 2.43 – рис. 2.44.

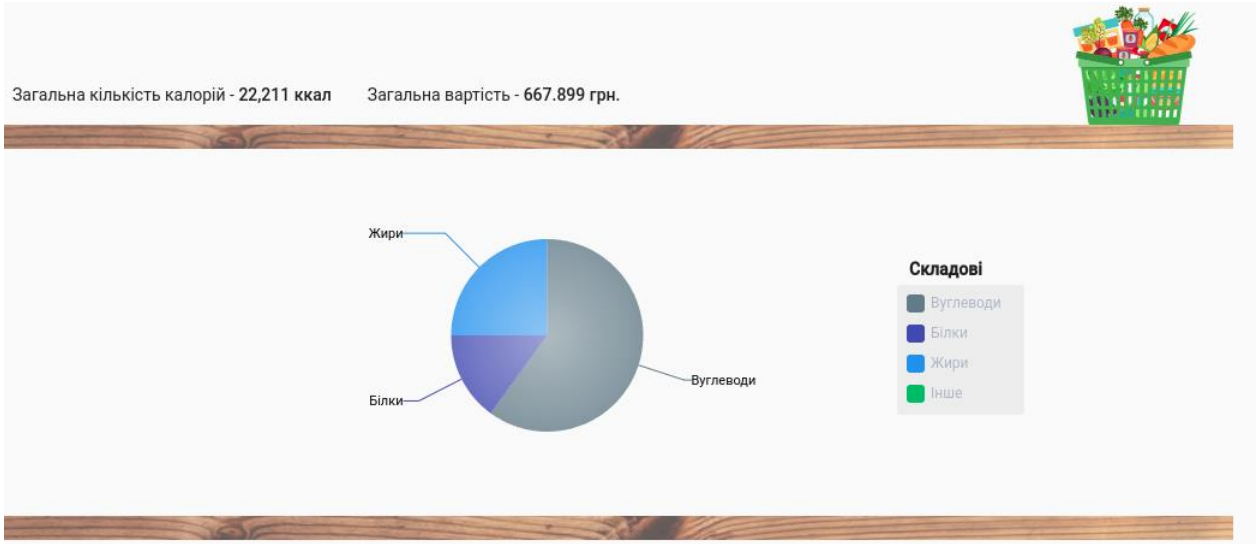


Рис. 2.43. Загальні результати розрахунків

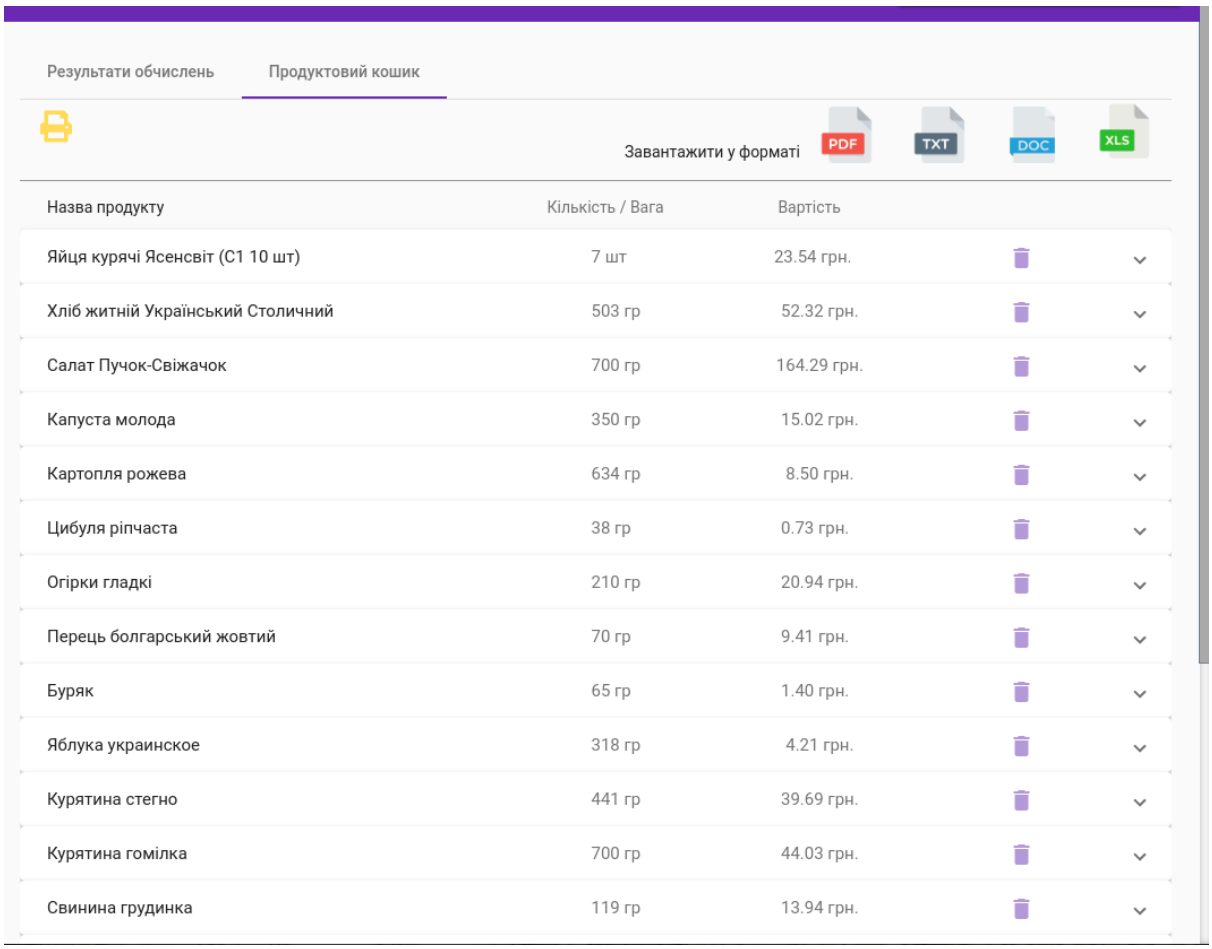


Рис. 2.44. Список продуктів, що є результатом розрахунків

Опис інтерфейсів реєстрації та авторизації

Для реєстрації у системі потрібно перейти до сторінки авторизації, відкривши відповідний пункт головного меню програми. На цій сторінці представлено дві вкладки: «Авторизація» та «Створити аккаунт». При відкритті останньої вкладки буде зображена форма введення даних для реєстрації у системі (рис. 2.45)

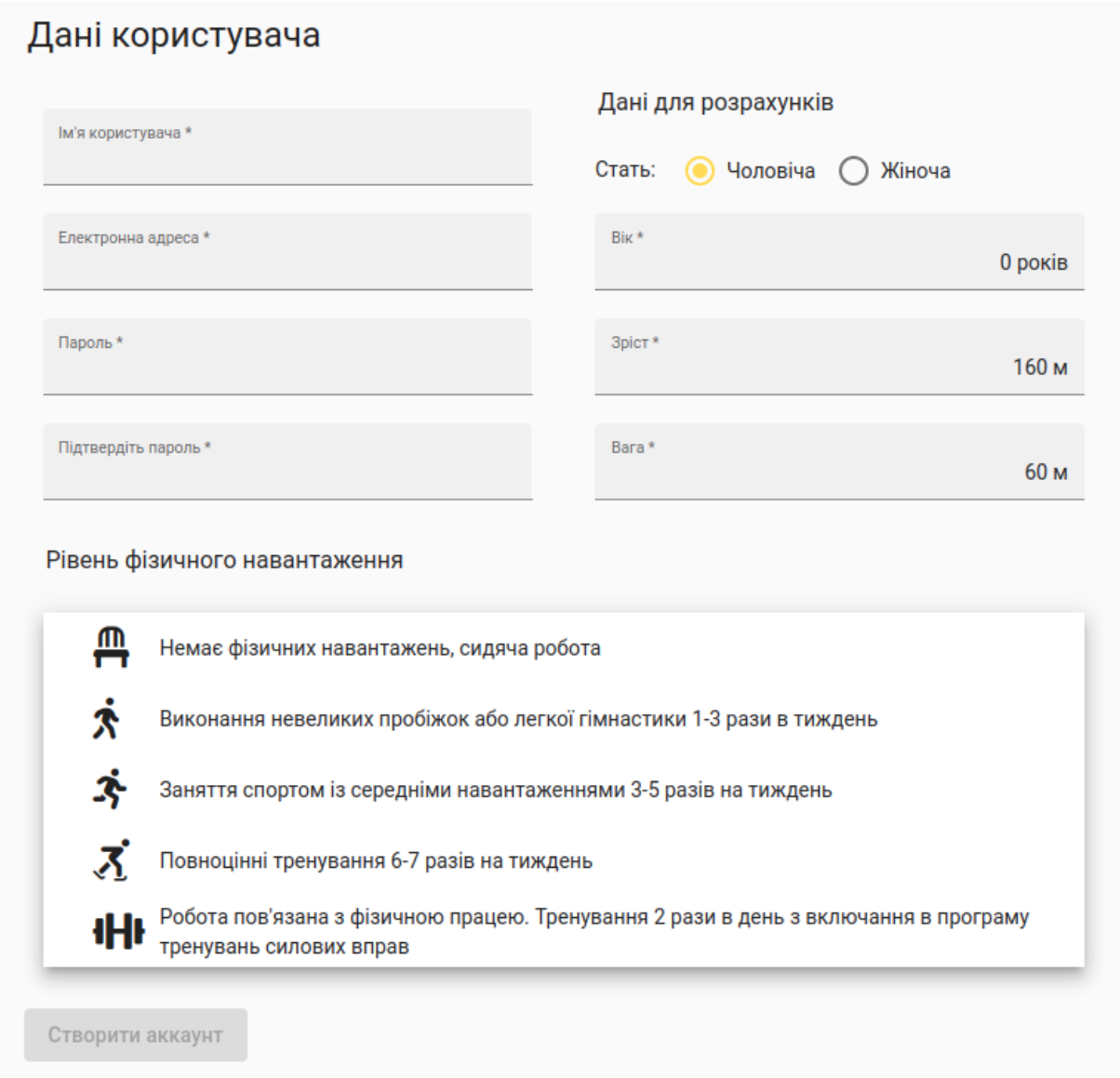


Рис. 2.45. Форма реєстрації у системі

У формі реєстрації потрібно ввести дані щодо ім'я користувача, адресу поштової скриньки та пароль до системи. Також заповнити дані для розрахунків добової норми калорій. Якщо користувач раніше вже заповняв поля даних для розрахунків, їх буде завантажено у поля форми автоматично (рис. 2.46.).

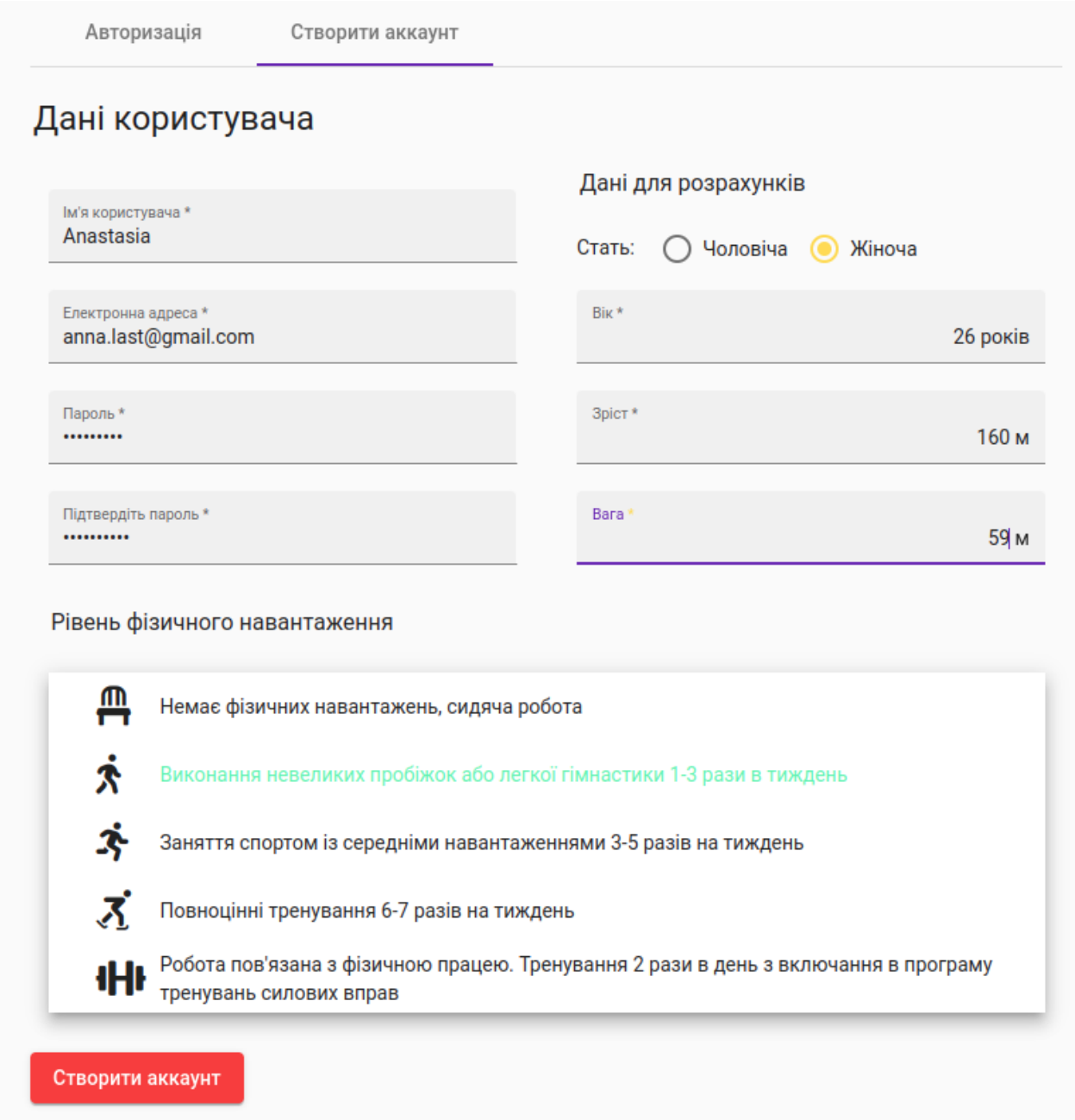


Рис. 2.46. Заповнена форма реєстрації у системі

Якщо користувач вже має існуючий обліковий запис, йому потрібно ввести дані щодо імені користувача та паролю у форму на вкладці «Авторизаці» (рис. 2.47). При введенні помилкових даних авторизації буде виведено відповідну помилку (рис. 2.48).

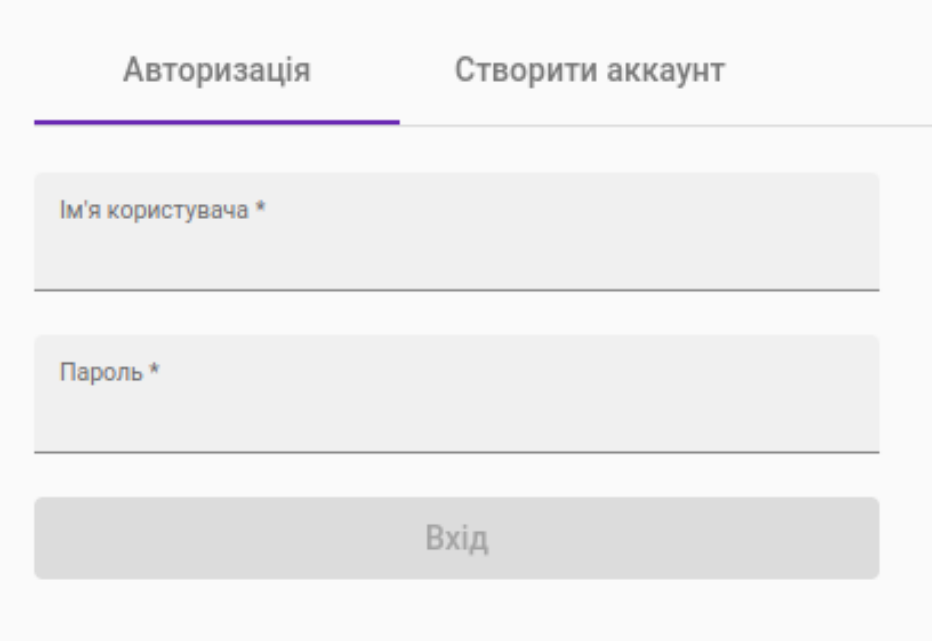


Рис. 2.47. Форма авторизації

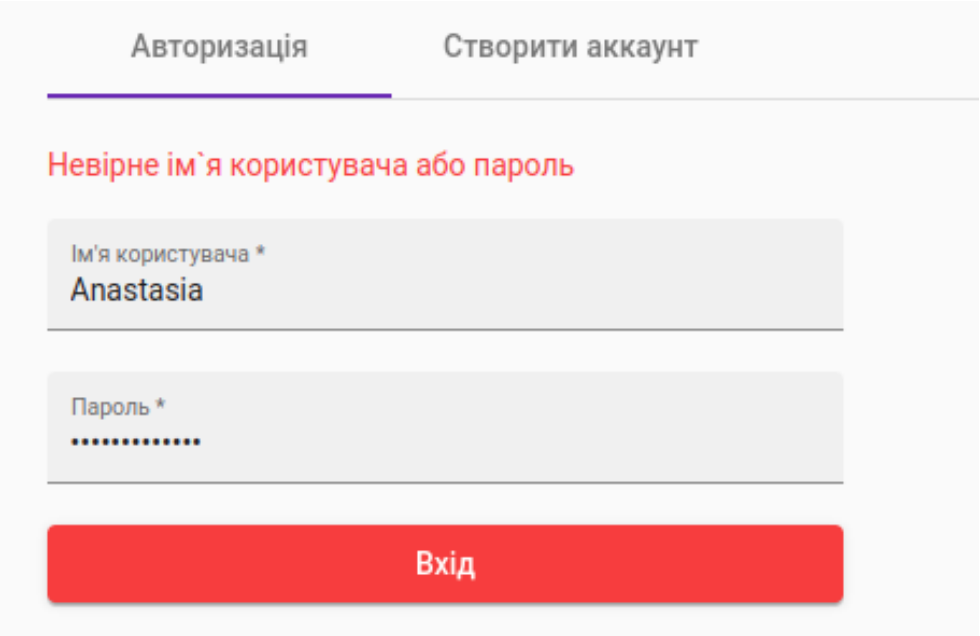


Рис. 2.48. Виведення помилки при введенні помилкових даних авторизації

Після авторизації, користувача буде направлено у його приватний кабінет, що містить інформацію щодо характеристик користувача та список розрахованих продуктових кошиків. Приклад сторінки кабінету користувача представлено на рис. 2.49.

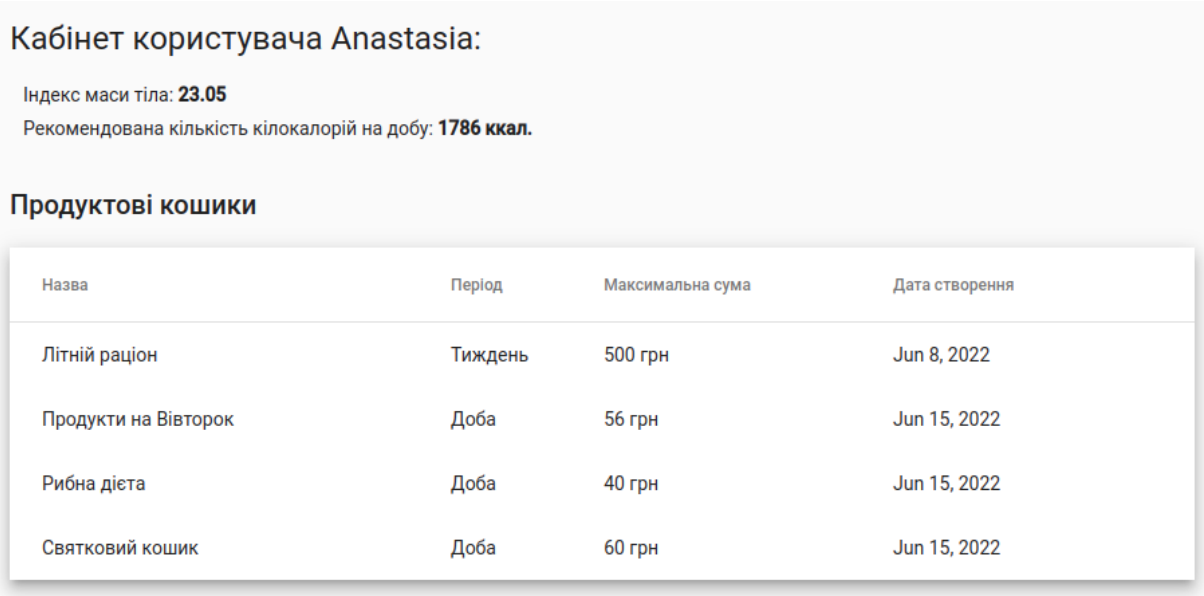


Рис. 2.49. Кабінет користувача

Опис інтерфейсів адміністратора

Для входу у кабінет адміністратора потрібно перейти за посиланням 0.0.0.0:8000/admin та ввести дані для авторизації (рис. 2.50).

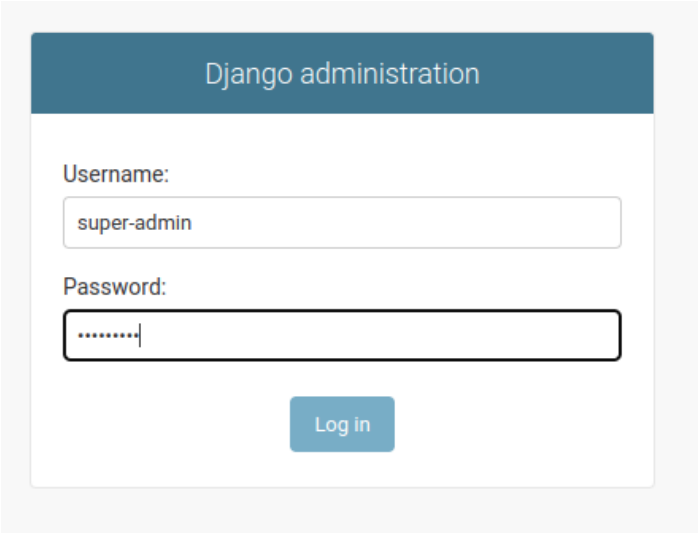


Рис. 2.50. Форма авторизації адміністратора

У кабінеті адміністратора представлено таблиці з даними системи з можливістю їх перегляду та керування. Також представлено список останніх дій користувача (рис. 2.51.).

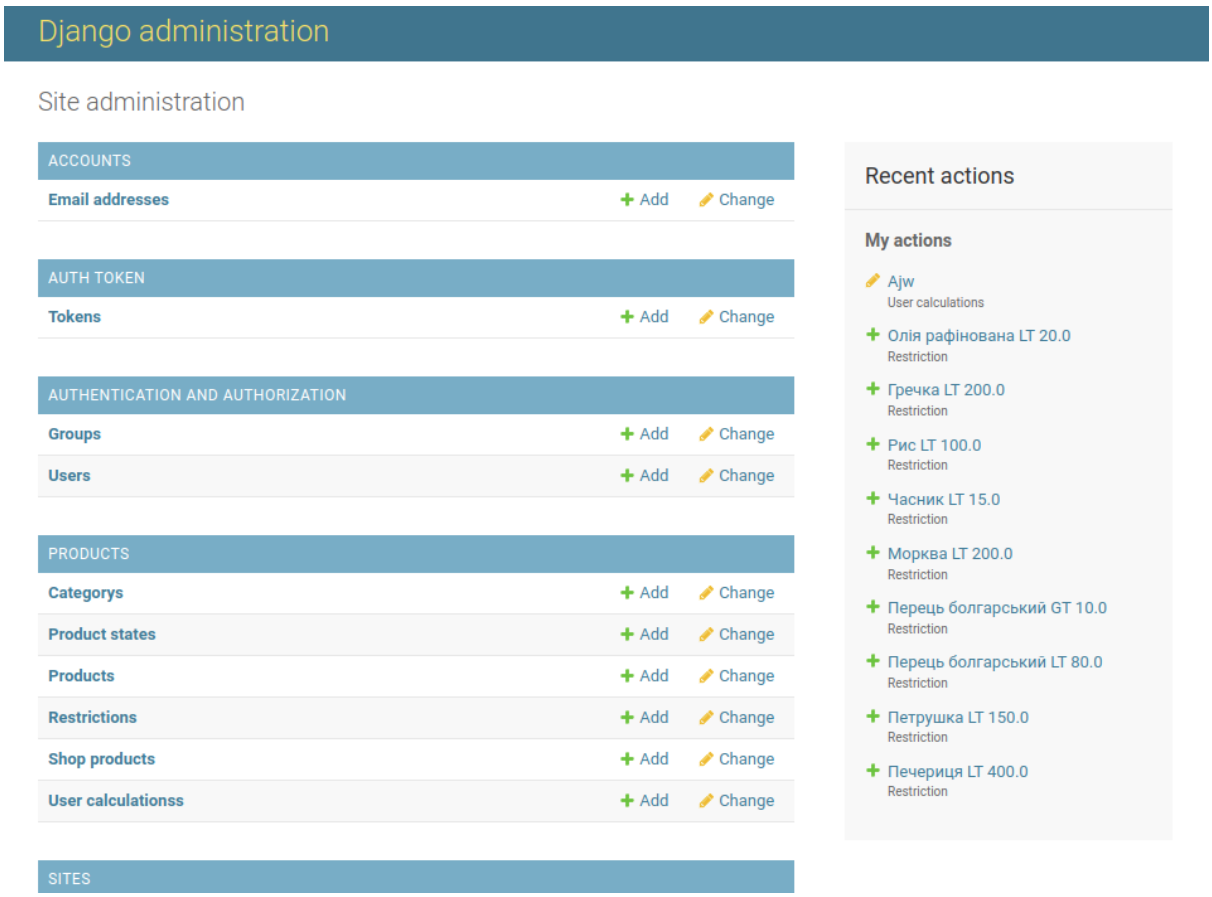


Рис. 2.51. Сторінка адміністрування

При відкритті списку продуктів адміністратор має можливість переглядати дані щодо окремого продукту, редагувати їх, додавати новий продукт або видаляти не потрібні продукти (рис. 2.52). На рис. 4.53 представлено форму додавання нового продукту.

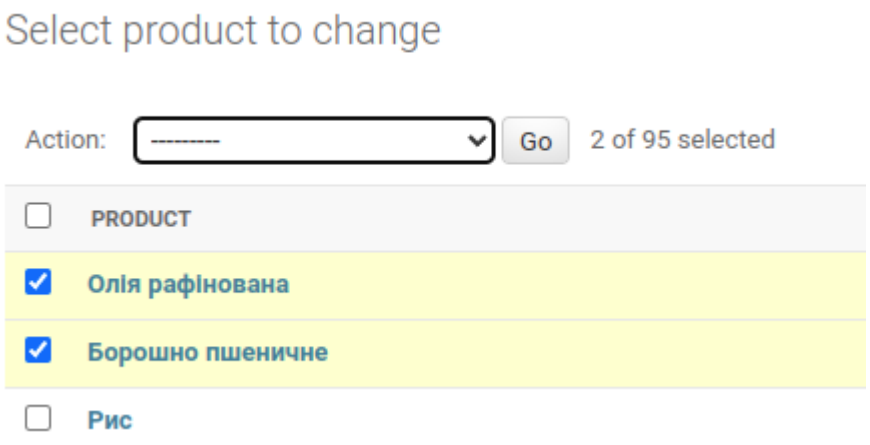


Рис. 2.52. Перегляд списку продуктів

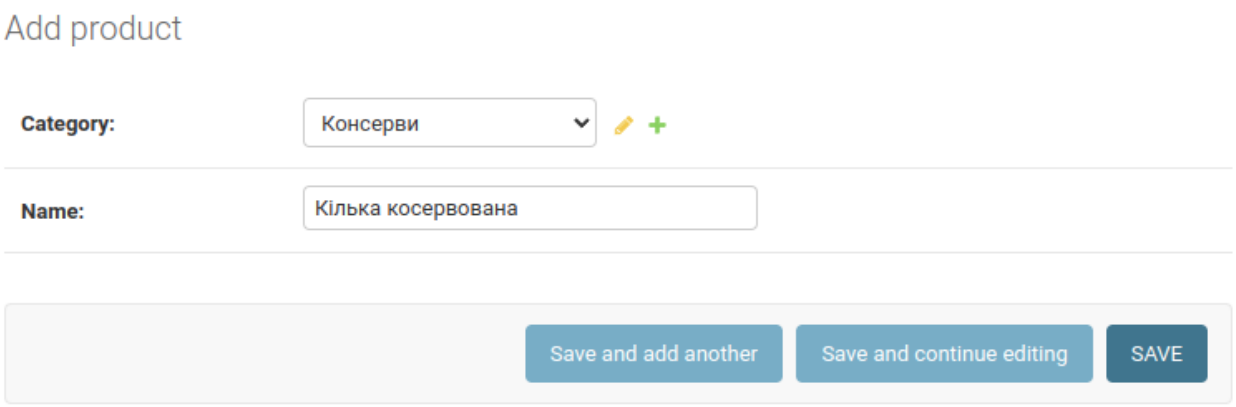


Рис. 2.53. Додавання нового продукту

Адміністратор має можливість переглядати список поточних користувачів системи (рис. 2.54), фільтрувати їх за статусом, деактивувати користувачів.

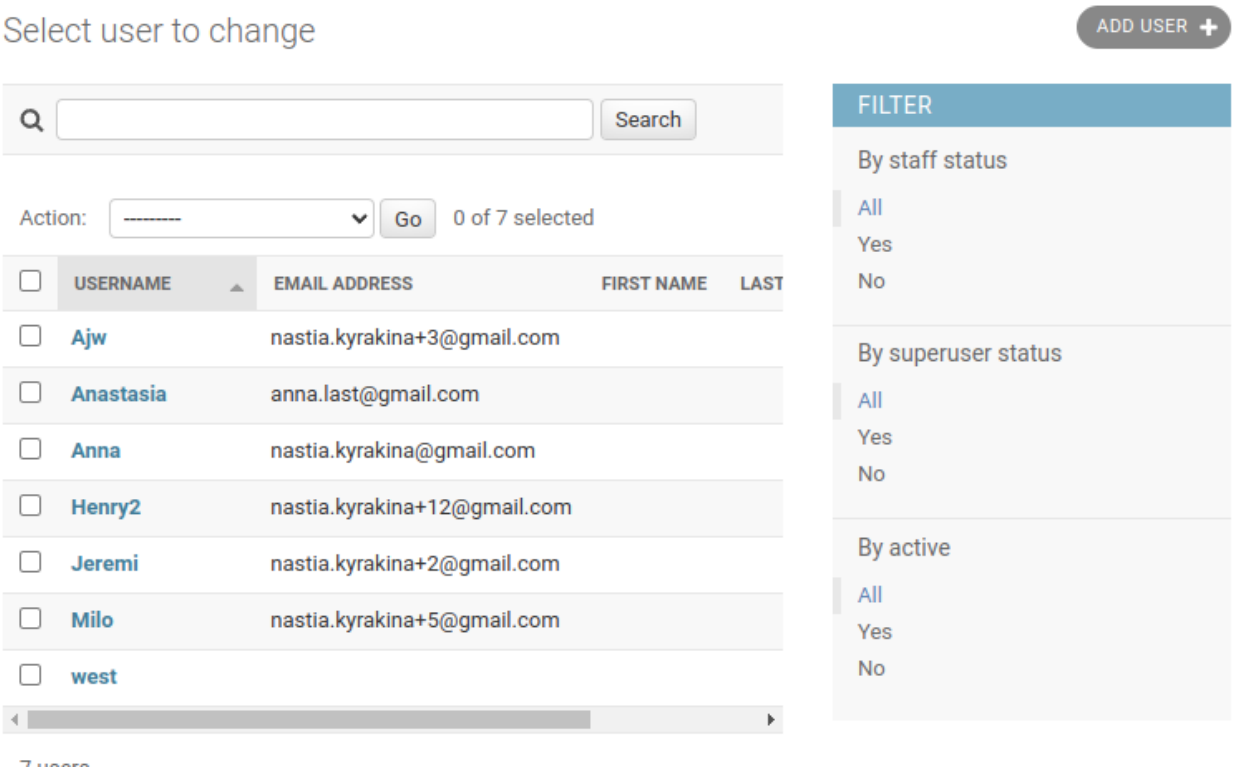


Рис. 2.54. Керування користувачами системи

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Обчислення трудомісткості розробки програмного забезпечення

Вхідні дані:

1. q – передбачуване число операторів – 1857.

2. C – коефіцієнт складності програми – 1,7.

3. p – коефіцієнт корекції програми в ході її розробки – 0,09.

4. B – коефіцієнт збільшення витрат – 1,3.

5. k – коефіцієнт кваліфікації програміста – 1.

6. $С_{ПР}$ – середня годинна заробітна плата програміста, грн/год. Згідно зі статистичними даними з сайту DOU.ua, для програміста кваліфікації Junior Software Engineer заробітна плата при розробці на Django Framework складає в середньому 700$ на місяць [31]. З урахуванням 40 годинного робочого тижня та при чинному курсі Національного банку України 1 долар = 29,46 гривень, отримаємо значення 117 грн/год.

7. B_k– число виконавців – 1.

8. F_p– місячний фонд робочого часу (при 40 годинному робочому тижні $F_p = 176$ годин).

9. $С_{Мч}$– вартість машино-години ЕОМ, грн/год – 4,8.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою (3.1):

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50),

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

t_n – витрати праці на програмування по готовій блок-схемі,

$t_{\text{отл}}$ – витрати праці на налагодження програми на ЕОМ,

$t_{\text{д}}$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється. Умовне число операторів (підпрограм) визначається за формулою (3.2):

$$Q = q * C * (1 + p), \tag{3.2}$$

де q – передбачуване число операторів,

C – коефіцієнт складності програми,

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1700 * 1{,}5 * (1 + 0{,}08) = 3\,441{,}02$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста представлені в формулі (3.3):

$$t_u = \frac{Q * B}{(75..85) * k}, \text{людино} - \text{годин}, \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

$$t_u = \frac{3\,441{,}02 * 1{,}3}{82 * 1} = 54{,}55, \text{людино} - \text{годин}.$$

Витрати праці на розробку алгоритму рішення задачі розраховуються за формулою (3.4):

$$t_a = \frac{Q}{(20..25) * k}, \text{людино} - \text{годин.}$$
$$t_a = \frac{3\ 441{,}02}{25 * 1} = 137{,}64, \text{людино} - \text{годин.} \quad (3.4)$$

Витрати на складання програми по готовій блок-схем розраховуються за формулою (3.5):

$$t_n = \frac{Q}{(20..25) * k}, \text{людино} - \text{годин.}$$
$$t_n = \frac{3\ 441{,}02}{21 * 1} = 163{,}86, \text{людино} - \text{годин.} \quad (3.5)$$

Витрати праці на налагодження програми на ЕОМ розраховуються за формулою (3.6):

$$t_{\text{отл}} = \frac{Q}{(4..5) * k}, \text{людино} - \text{годин.}$$
$$t_{\text{отл}} = \frac{3\ 441{,}02}{5 * 1} = 688{,}20, \text{людино} - \text{годин.} \quad (3.6)$$

Витрати праці на підготовку документації представлені у формулі (3.7):

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \quad (3.7)$$

де $t_{\text{др}}$ – трудомісткість підготовки матеріалів і рукопису, що розраховується за формулою 3.8.

$$t_{\text{др}} = \frac{Q}{(15..20) * k}, \text{людино} - \text{годин,}$$
$$t_{\text{др}} = \frac{3\ 441{,}02}{16 * 1} = 215{,}06, \text{людино} - \text{годин.} \quad (3.8)$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації, що розраховується за формулою 3.9.

$$t_{до} = 0{,}75 * t_{др}, \text{людино} - \text{годин}. \quad (3.9)$$

$$t_{до} = 0{,}75 * 215{,}06 = 161{,}30, \text{людино} - \text{годин}.$$

$$t_{д} = 215{,}06 + 161{,}30 = 376{,}36, \text{людино} - \text{годин}.$$

$$t = 50 + 54{,}55 + 137{,}64 + 163{,}86 + 688{,}20 + 376{,}36$$
$$= 1470{,}62, \text{людино} - \text{годин}$$

3.2. Обчислення витрат на створення програмного забезпечення

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ. Розраховується $К_{ПО}$ за формулою (3.10)

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{грн}. \quad (3.10)$$

Заробітна плата виконавців визначається за формулою (3.11):

$$З_{ЗП} = t * С_{ПР}, \text{грн}, \quad (3.11)$$

де: t – загальна трудомісткість, людино-годин,

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 1470{,}62 * 117 = 172\,062{,}26, \text{грн}.$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ визначається за формулою (3.12):

$$З_{МВ} = t_{отл} * С_{мч}, \text{грн}, \quad (3.12)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год,

$С_{мч}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 688{,}20 * 4{,}8 = 3\ 303{,}38, \text{грн.}$$

$$К_{ПО} = 172\ 062{,}26 + 3\ 303{,}38 = 175\ 365{,}64, \text{грн.}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ визначається за формулою (3.13):

$$T = \frac{t}{B_k * F_p}, \text{міс,} \tag{3.13}$$

де B_k - число виконавців,

F_p - місячний фонд робочого часу.

$$T = \frac{1470{,}62}{1 * 176} = 8{,}36, \text{міс.}$$

Висновки: обчислено трудомісткість розробленого додатку (1470,62 людино годин), проведено обчислення вартості роботи по створенню програми (175 365,64 грн.) та визначено час на її створення (8,36 місяців).

ВИСНОВКИ

В даній кваліфікаційній роботі був розроблений програмний додаток для моделювання дієти на Python за допомогою Djnago.

Це web-додаток, що надає можливість виконувати оптимізацію продуктового раціону шляхом розрахунку добової норми калорій згідно індивідуальних характеристик користувача, вставленню оптимального співвідношення білків, жирів та вуглеводів у обраних продуктах та мінімізації вартості продуктового кошику.

У результаті розробки програмного додатку було реалізовано наступний функціонал:

– реєстрація та авторизація користувачів;

– web-інтерфейси для адміністрування та введення даних для розрахунків;

– перегляд та корегування даних щодо продуктів, їх вартості та енергетичної цінності;

– перегляд та корегування даних щодо обмежень на вживання продуктів;

– виконання моделювання збалансованого набору продуктів харчування згідно заданих користувачем параметрів та обмежень;

– перегляд інформації щодо вже розрахованих продуктових кошиків.

Програмний додаток може буди розширений та :

– надання можливості користувачам вказувати власне співвідношення енергетичних речовин у змодельованому продуктовому кошику;

– надання можливості користувачам вказувати обмеження на різні періоди часу: тиждень, місяць і т.п.;

– надання можливості користувачам публікувати та ділитися розрахованими моделями дієт.

Під час виконання даної кваліфікаційної роботи проекту були виконані наступні задачі:

– проведено аналіз предметної галузі;

– проведено пошук та збір інформації щодо продуктів та їх характеристик;

– обрано раціональну структуру бази даних та web-додатків;
– написано програмний код web-сервісу та web-клієнту;
– налаштовано зв’язок між web-сервісом та web-клієнтом;
– проведено аналіз для подальшого розвитку проекту.

Web-сервер було реалізовано за допомогою мови програмування Python з використанням фреймворку Djnago. Web-клієнт було реалізовано за допомогою мови TypeScript з використанням фреймворку Angular. Для виконання обчислень було використано бібліотеку SciPy. Для реалізації бази даних СУБД PostgreSQL.

У кваліфікаційній роботі також було визначено трудомісткість створеного програмного продукту (1470,62 людино-годин), проведено підрахунок вартості роботи по розробці програми (175 365,64 грн) та підраховано час на її створення (8,36 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Паньків В.І. Цукровий діабет, переддіабет і серцево-судинні захворювання // Практична ангіологія. – 2007. – № 1(6). – с. 2-4

2. Швець О.В. Дієта при цукровому діабеті 2-го типу. Рекомендації для пацієнтів. Кафедра внутрішньої медицини №3 та ендокринології ХНМУ. URL: http://vnmed3.kharkiv.ua/wp-content/uploads/2014/05/diabetes2_diet.pdf (дата звернення: 13.05.2022).

3. Калмикова Ю.С. Особливості лікувального харчування при цукровому діабеті. Педагогіка, психологія та медико-біологічні проблеми фізичного виховання і спорту. 2013. Т. 17, № 1. С. 30–31.

4. Кравчун Н.О. Сучасні підходи до лікування дисліпопротеїдемій у хворих на цукровий діабет: [методичні рекомендації] / Кравчун Н.О., Козаков О.В., Пилипенко І.І. – Київ-Харків: Інститут проблем ендокринної патології, 2005. – 24 с.

5. Методика визначення хімічного складу та енергетичної цінності продуктів харчування. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/z0146-00 (дата звернення: 15.05.2022).

6. Сучасне лікування метаболічного синдрому: [методичні рекомендації] / І.І. Єрмакович, В.А. Чернишов, С.В. Белозерова. – Харків: ХНІІ терапії АМН України, 2004. – 24 с.

7. Як визначити оптимальну вагу: формула індексу маси тіла. Міністерство охорони здоров’я України. URL: https://moz.gov.ua/article/health/jak-viznachti-optimalnu-vagu-formula-indeksu-masi-tila (дата звернення: 15.05.2022).

8. Формула Міффліна-Сан-Жеора для розрахунку калорій для чоловіків і жінок. Жіночий Світ. URL: https://w2w.com.ua/formyla-mifflina-san-jeora-dlia-rozrahynky-kalorii-dlia-cholovikiv-i-jinok/ (дата звернення: 18.05.2022).

9. Tablycjakalorijnosti.com.ua - калорійність продуктів, ккал., калорії. Таблиця Калорійності. URL: https://www.tablycjakalorijnosti.com.ua/ (дата звернення: 20.05.2022).

10. Мінфін. Ціни на продукти. Ставки, индексы, тарифы. URL: https://index.minfin.com.ua/ua/markets/wares/prods/ (дата звернення: 27.05.2022).

11. Hoppe P. D. Robert. H. Optimization Theory : навчальний посібник. New York City : HarperCollins Publishers, 2015. 56 p. URL: https://www.math.uh.edu/~rohop/fall_06/Chapter4.pdf (дата звернення: 02.06.2022).

12. Django overview | Django. The web framework for perfectionists with deadlines | Django. URL: https://www.djangoproject.com/start/overview/ (дата звернення: 05.06.2022).

13. Bengisu B. What is Object-Relational Mapping (ORM)?. Medium. URL: https://blog.devgenius.io/what-is-object-relational-mapping-orm-177b8ab54d85 (дата звернення: 05.06.2022).

14. Subramanian H., Raj P. Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs. Packt Publishing, 2019. 378 p.

15. Python Software Foundation. 3.10.5 Documentation. URL: https://docs.python.org/3/ (дата звернення: 08.06.2022).

16. Лутц М. Программирование на Python, том II, 4-е издание: пер. з англ./ М. Лутц. – СПб. : Символ-Плюс, 2011. – 992 с.

17. Ayman H. Learning Website Development with Django. – Packt Publishing Ltd.: Birmingham, 2008. – 261с.

18. SciPy documentation – SciPy v1.8.1 Manual. Numpy and Scipy Documentation – Numpy and Scipy documentation. URL: https://docs.scipy.org/doc/scipy (дата звернення: 07.06.2022).

19. Васильев, А. Ю. Работа с PostgreSQL: настройка и масштабирование. – Creative Commons Attribution-Noncommercial 4.0 International, 2017. – 288 с.

20. Фрэйн, Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств. 2-е изд. – СПб.: Питер, 2017. – 272 с.

21. Флэнаган, Д. JavaScript. Подробное руководство, 6-е издание. – Пер. с англ. – СПб: Символ- Плюс, 2012. – 1080 с., ил.

22. Documentation - TypeScript for JavaScript Programmers. TypeScript: JavaScript With Syntax For Types. URL: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html (дата звернення: 08.06.2022).

23. Sass: Syntactically Awesome Style Sheets. Sass: Syntactically Awesome Style Sheets. URL: https://sass-lang.com/ (дата звернення: 08.06.2022).

24. Angular. Angular Overview. URL: https://angular.io/ (дата звернення: 10.06.2022).

25. Grabovskiy P. Angular vs Vue.js: що обрати. ДОУ. URL: https://dou.ua/lenta/columns/angular-vs-vuejs/ (дата звернення: 19.06.2022).

26. Як використовувати JSON Web Tokens (JWT) для автентифікації. Codeguida. URL: https://codeguida.com/post/1567 (дата звернення: 12.06.2022).

27. PyCharm: the Python IDE for Professional Developers by JetBrains. JetBrains. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 13.06.2022).

28. Hoppe P. D. Robert. H. Optimization Theory : навчальний посібник. New York City : HarperCollins Publishers, 2015. 56 p. URL: https://www.math.uh.edu/~rohop/fall_06/Chapter4.pdf (дата звернення: 02.06.2022).

29. Chacon S., Straub B. Pro Git book. 2nd ed. New York : Apress, 2014. 440 p.

30. Що таке Docker і навіщо він?. QualityAssuranceGroup. URL: https://qagroup.com.ua/publications/shcho-take-docker-i-navishcho-vin/ (дата звернення: 12.06.2022)

31. Статистика зарплат програмістів, тестувальників і PM в Україні | DOU URL: https://jobs.dou.ua/salaries/?period=2021-12&position=Junior%20SE&technology=Python&education=3&english=2-3 (дата звернення 07.06.2022).

ДОДАТОК А

КОД ПРОГРАМИ

У додатку наведені приклади коду класів типу модель, в'ю, сервіс, компонент. У додатку наведено приклади 14 файлів, загальна кількість файлів – 208, вони додаються на оптичному носії разом з кваліфікаційною роботою.

Приклад ORM класів серверу

Category model

```
from django.db import models

# клас для опису категорії
class Category(models.Model):
    # назва категорії
    name = models.CharField(max_length=250)

    def __str__(self):
        return self.name
```

Product model

```
from django.db import models
from .category import Category

# клас для опису продуктів
class Product(models.Model):
    # опис зв'язку з моделлю Category
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    # назва продукту
    name = models.CharField(max_length=350)

    def __str__(self):
        return self.name
```

ShopProduct model

```
from django.db import models
from .product import Product
from .product_state import ProductState

# клас для опису продуктів в магазині
class ShopProduct(models.Model):
    # зв'язок з сутністю Продукт
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    # назва продукту в магазині
    name = models.CharField(max_length=350)
    # вартість продукту
    price = models.DecimalField(max_digits=6, decimal_places=2)
    # кількість продукту
    amount = models.DecimalField(max_digits=6, decimal_places=2)
    # одиниця виміру
    unit = models.CharField(max_length=3)
```

```
    # список станів продукта
    states = models.ManyToManyField(ProductState)
    # позначка чи є цей продукт за замовчуванням чи створений користувачем
    default = models.BooleanField(default=True)

    def __str__(self):
        return self.name
```

ProductState model

```
from django.db import models

# доступні стани продукту
PRODUCT_STATE_CHOICES = [
    ('AI', 'As is'),
    ('BL', 'Boiled'),
    ('BK', 'Baked'),
]
# клас для опису продуктів в магазині
class ProductState(models.Model):
     # назва блюда
    dish_name = models.CharField(max_length=350)
     # тип приготування
    state = models.CharField(max_length=2, choices=PRODUCT_STATE_CHOICES)
     # кількість кілокалорій на 100 грам
    energy = models.DecimalField(max_digits=6, decimal_places=2)
     # кількість вуглеводів на 100 грам
    carbohydrates = models.DecimalField(max_digits=6, decimal_places=2)
     # кількість білків на 100 грам
    proteins = models.DecimalField(max_digits=6, decimal_places=2)
     # кількість жирів на 100 грам
    fats = models.DecimalField(max_digits=6, decimal_places=2)

    def __str__(self):
        return self.state + ' ' + self.dish_name
```

Restriction model

```
from django.db import models

from . import Product
from .category import Category

COMPARATORS = [
    ('EQ', 'equal'),
    ('GT', 'Great than'),
    ('LT', 'Less than'),
]

# клас опису сутності обмежень
class Restriction(models.Model):
    # посилання на сутність продукту
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    # умова порівняння
    comparator = models.CharField(max_length=2, choices=COMPARATORS)
    # кількість продукту
    amount = models.FloatField()
    # одиниця виміру
    unit = models.CharField(max_length=3)
    default = models.BooleanField(default=True)

    def __str__(self):
```

```
        return self.product.name + ' ' + self.comparator + ' ' + str(self.amount)
```

UserCalculations model

```
from django.contrib.auth.models import User
from django.core.validators import MinValueValidator, MaxValueValidator
from django.db import models

from products.models import ShopProduct, Restriction

SEX_CHOICES = [
    ('M', 'Male'),
    ('F', 'Female'),
]
ACTIVITY_CHOICES = [
    ('1', 'lowest'),
    ('2', 'low'),
    ('3', 'medium'),
    ('4', 'high'),
    ('5', 'highest'),
]

# клас опису даніх для розрахунків користувача
class UserCalculations(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    height = models.FloatField()
    weight = models.FloatField()
    years = models.IntegerField()
    sex = models.CharField(max_length=2, choices=SEX_CHOICES)
    activity_level = models.CharField(max_length=2, choices=ACTIVITY_CHOICES)

    def __str__(self):
        return self.user.username

# клас опису зв’язку користувача та продукту
class UserProduct(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    shop_product = models.ForeignKey(ShopProduct, on_delete=models.CASCADE)

# клас опису зв’язку користувача та обмеження
class UserRestriction(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    restriction = models.ForeignKey(Restriction, on_delete=models.CASCADE)
```

ProductsBasket model

```
from django.contrib.auth.models import User
from django.core.validators import MinValueValidator, MaxValueValidator
from django.db import models

# клас опису розрахованого продуктового кошику
class ProductsBasket(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=250)
    creation_date = models.DateTimeField(auto_now_add=True)
    period = models.IntegerField(validators=[MinValueValidator(0), MaxValueValidator(366)])
    max_sum = models.FloatField()
    # результати розрахунків
    products = models.JSONField()

    def __str__(self):
        return self.name
```

Класи обробки запитів серверу

ProductCalcApiView model

```
# клас для обробки запитів на отримання та розрахунку продуктових кошиків
class ProductCalcApiView(APIView):
    # add permission to check if user is authenticated
    # permission_classes = [permissions.IsAuthenticated]

    # Отримати список продуктових кошиків
    def get(self, request, *args, **kwargs):
        baskets = ProductsBasket.objects.all()
        serializer = ProductsBasketSerializer(baskets, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)

    def post(self, request, *args, **kwargs):
        #отримання даних з запиту
        products = request.data.get('products')
        energy_per_day = request.data.get('energyAmount')
        max_sum = request.data.get('maxSum')
        term = int(request.data.get('term'))
        # отримрання спису продуктів в магазині з бази даних
        shop_products = products if len(products) else ShopProduct.objects.prefetch_related('states').all()
        # отримрання спису обмежень з бази даних
        restrictions = Restriction.objects.prefetch_related('product').all()
        # підготовка списку продуктів до разрахунків
        products_list = prepare_products_for_calc(shop_products, len(products) == 0, restrictions)
        # виконання оптимізації
        sol = optimize_products_bucket(products_list, [], max_sum, energy_per_day, term)
        # збереження даних для авторизованого користувача
        if request.user.is_authenticated:
            sol_json = json.dumps(sol, default=lambda o: o.__dict__, ensure_ascii=False, sort_keys=True, indent=4)
            basket = dict()
            basket['name'] = 'test basket'
            basket['period'] = 1
            basket['max_sum'] = max_sum
            basket['user'] = request.user
            basket['products'] = sol_json
            ProductsBasket.objects.create(**basket)
        return Response({'optimization': sol, 'products': products_list}, status=status.HTTP_200_OK)
```

UserCalculationsApiView model

```
# клас для обробки запитів на отриманні та збереження даних користувача
class UserCalculationsApiView(APIView):
    def get(self, request, *args, **kwargs):
        print(request.user.pk)
        print(request.user.is_authenticated)
        userCalculations = User.objects.get(id=request.user.pk).usercalculations
        serializer = UserCalculationsSerializer(userCalculations)
        return Response(serializer.data, status=status.HTTP_200_OK)
    def post(self, request, *args, **kwargs):
        user = request.user
        data = request.data.copy()
        data['user'] = user
        user_calculations = UserCalculations.objects.create(**data)
        serializer = UserCalculationsSerializer(user_calculations)
        return Response(serializer.data, status=status.HTTP_201_CREATED)
```

Класи обробки запитів серверу

ProductCalcApiView model

```
import math
from functools import reduce

import numpy as np
from scipy.optimize import minimize

# мінімальні та максимальні обмеження на вживання елементів продукту
energy_restrictions = dict({
        'carbohydrates': (0.55, 0.6),
        'proteins': (0.15, 0.20),
        'fats': (0.2, 0.25),
    })
# енергія на одиницю елемента
energy_per_components = dict({
        'carbohydrates': 4,
        'proteins': 4,
        'fats': 9,
    })

# отримання обмеження для елемента
def get_constraint_fun(name, constr_type, kkal_amount):
    def min_constraint(x: list, products):
        pr = products
        sum_energy = 0
        for index, xi in enumerate(x):
            sum_energy = sum_energy + xi*pr[index].get(name)
        return sum_energy*energy_per_components.get(name) - kkal_amount*energy_restrictions.get(name)[0]

    def max_constraint(x: list, products):
        pr = products
        sum_energy = 0
        for index, xi in enumerate(x):
            sum_energy = sum_energy + xi*pr[index].get(name)
        return kkal_amount*energy_restrictions.get(name)[1] - sum_energy*energy_per_components.get(name)
    return min_constraint if constr_type == 'min' else max_constraint

# функція підготовки та виконання оптимізації продуктового кошику
def optimize_products_bucket(products, constr, max_sum, kkal_per_day, days = 1):
    kkal_amount = kkal_per_day*days
   # цільова функція
    def objective(x: list):
        pr = products
        sum_prices = 0
        for index, xi in enumerate(x):
            sum_prices = sum_prices + xi*pr[index].get('price')
        return sum_prices
# обмеження
    def price_constraint(x: list, pr):
        sum_prices = 0
        for index, xi in enumerate(x):
            sum_prices = sum_prices + xi*pr[index].get('price')
        return max_sum - sum_prices

    def energy_constraint(x: list, pr):
        sum_energy = 0
        for index, xi in enumerate(x):
            sum_energy = sum_energy + xi*pr[index].get('energy')
```

```
        return sum_energy - kkal_amount
    def get_init_guess(x):
        guess_list = list()
        for index, xi in enumerate(x):
            guess_list.append(1)
        return guess_list
# допустові граничні значення для кожного з продуктів
    def get_bounds():
        pr = products
        bounds = list()
        for index, xi in enumerate(pr):
            equal_bound = [restriction for restriction in xi.get('restrictions') if restriction.get('comparator') == 'EQ']
            if len(equal_bound):
                bounds.append((equal_bound[0].get('amount')*days, equal_bound[0].get('amount')*days))
                continue
            min_restriction = [restriction for restriction in xi.get('restrictions') if restriction.get('comparator') == 'GT']
            max_restriction = [restriction for restriction in xi.get('restrictions') if restriction.get('comparator') == 'LT']
            min_bound = min_restriction[0].get('amount') if len(min_restriction) else 0
            max_bound = max_restriction[0].get('amount') if len(max_restriction) else 200

            bound = (min_bound*days, max_bound*days)
            bounds.append(bound)
        return bounds
 # створення параметрів обмежень
    con1 = {'type': 'ineq', 'fun': price_constraint, 'args': [products]}
    con2 = {'type': 'eq', 'fun': energy_constraint, 'args': [products]}
    con3 = {'type': 'ineq', 'fun': get_constraint_fun('carbohydrates', 'min', kkal_amount), 'args': [products]}
    con4 = {'type': 'ineq', 'fun': get_constraint_fun('carbohydrates', 'max', kkal_amount), 'args': [products]}
    con5 = {'type': 'ineq', 'fun': get_constraint_fun('fats', 'min', kkal_amount), 'args': [products]}
    con6 = {'type': 'ineq', 'fun': get_constraint_fun('fats', 'max', kkal_amount), 'args': [products]}
    con7 = {'type': 'ineq', 'fun': get_constraint_fun('proteins', 'min', kkal_amount), 'args': [products]}
    con8 = {'type': 'ineq', 'fun': get_constraint_fun('proteins', 'max', kkal_amount), 'args': [products]}

    constr = [con1, con2, con3, con4, con5, con6, con7, con8]
    bounds = get_bounds()
 # виклик фуннкці оптимізації
    sol = minimize(objective, get_init_guess(products), method='SLSQP', bounds=bounds, constraints=constr)
    amounts = sol.x
    res = list()
    general = {
        'energy': 0,
        'price': 0,
        'carbohydrates': 0,
        'proteins': 0,
        'fats': 0,
    }
 # обробка результатів оптимізації
    for index, product in enumerate(products):
        general['energy'] = general['energy'] + amounts[index] * product.get('energy')
        general['price'] = general['price'] + amounts[index] * product.get('price')
        general['carbohydrates'] = general['carbohydrates'] + amounts[index] * product.get('carbohydrates')
        general['proteins'] = general['proteins'] + amounts[index] * product.get('proteins')
        general['fats'] = general['fats'] + amounts[index] * product.get('fats')

        if round(amounts[index], 4) != 0:
            product_res = {
                'id': product.get('id'),
                'name': product.get('name'),
                'amount': round(amounts[index]),
                'unit': product.get('unit') if product.get('unit') == 'шт' else 'гр',
                'price': round(amounts[index] * product.get('price'), 4),
                'states': [{
                    'state': product.get('state'),
```

```
                'energy': round(amounts[index] * product.get('energy'), 4),
                'carbohydrates': round(amounts[index] * product.get('carbohydrates'), 4),
                'proteins': round(amounts[index] * product.get('proteins'), 4),
                'fats': round(amounts[index] * product.get('fats'), 4),
            }]
        }
        res.append(product_res)
    return {'product_bucket': res, 'general': general}
```

Компоненти клієнту

InputForms Component TS file

```
// компонент з формами для проведення розрахунків
@Component({
  selector: 'app-input-forms',
  templateUrl: './input-forms.component.html',
  styleUrls: ['./input-forms.component.scss']
})
export class InputFormsComponent implements OnInit, OnDestroy {

  @Select(CalculationsState.calculationsForm) calculationsForm$!: Observable<ICalculationsState>;
  firstFormGroup: FormGroup;
  calculationsGroup: FormGroup;
  basketForm: FormGroup;
  userForm: FormGroup;
  userWorkForm: FormGroup;
  isEditable = true;
  optimizationInProgress = false;

  energyPerDay!: number;
  userMIT!: number;

  private unsubscribe$: Subject<void> = new Subject<void>();

  constructor(
    private fb: FormBuilder,
    private store: Store,
    private router: Router,
    private calculationsService: CalculationsService,
  ) {}

// ініціалізація компоненту та створення форм
  ngOnInit() {
    this.firstFormGroup = this.fb.group({
      firstCtrl: ['', Validators.required],
    });
    this.basketForm = this.getBasketForm();
    this.userForm = this.getUserForm();
    this.userWorkForm = this.getUserWorkForm();
    this.calculationsGroup = this.fb.group({
      basket: this.basketForm,
      user: this.userForm,
      userWork: this.userWorkForm,
    });

    this.calculationsForm$
      .pipe(takeUntil(this.unsubscribe$))
      .subscribe(calculationsForm => {
        this.energyPerDay = calculationsForm.energyAmount;
        this.userMIT = calculationsForm.MIT;
```

```
    });
  }

  getBasketForm(): FormGroup {
    return this.fb.group({
      term: ['week', [Validators.required]],
      maxSum: [1000, [Validators.required, Validators.min(0), Validators.max(20000)]],
    });
  }

  getUserForm(): FormGroup {
    return this.fb.group({
      height: [160, [Validators.required]],
      weight: [50, [Validators.required]],
      years: [30, [Validators.required]],
      sex: [Sex.Male, [Validators.required]],
    });
  }

  getUserWorkForm(): FormGroup {
    return this.fb.group({
      activityLevel: ['', [Validators.required]],
    });
  }
  optimizeProductBasket(): void {
    this.optimizationInProgress = true;
    this.store.dispatch(new OptimizeBasketAndSetAsCurrent({}))
      .pipe(
        finalize(() => this.optimizationInProgress = false)
      )
      .subscribe(
        () => this.router.navigateByUrl('baskets/new')
      );
  }

  goBack(stepper: MatStepper) {
    stepper.previous();
  }
  goForward(stepper: MatStepper) {
    this.onDoneInteract();
    stepper.next();
  }
// запит на розрахунки
  onDoneInteract(): void {
    this.store.dispatch(new SetBasketFormData(this.getBasketFormData()));
  }
  getBasketFormData(): ICalculations {
    return {
      user: {
        ...this.userWorkForm.value,
        ...this.userForm.value,
      },
      ...this.basketForm.value,
    }
  }
  ngOnDestroy(): void {
    this.unsubscribe$.next();
    this.unsubscribe$.complete();
  }
}
```

InputForms Component HTML file

```
<form *ngIf="!optimizationInProgress; else optimizationLoader" [formGroup]="firstFormGroup">
 <mat-stepper linear #stepper>
  <mat-step [stepControl]="basketForm" [editable]="isEditable">
   <ng-template matStepLabel>Термін та вартість</ng-template>
   <app-basket-info [basketForm]="basketForm"></app-basket-info>
   <div class="stepper-buttons">
    <button matStepperNext
            mat-mini-fab
            color="warn"
            aria-label="Forward icon"
            [disabled]="basketForm.invalid">
     <mat-icon>forward</mat-icon>
    </button>
   </div>
  </mat-step>
  <mat-step [stepControl]="userForm" [editable]="isEditable">
   <ng-template matStepLabel>Дані користувача</ng-template>
   <app-user-private-info [userForm]="userForm"></app-user-private-info>
   <div class="stepper-buttons">
    <button matStepperNext
            mat-mini-fab
            color="warn"
            aria-label="Forward icon"
            [disabled]="userForm.invalid">
     <mat-icon>forward</mat-icon>
    </button>
    <button matStepperPrevious
            mat-mini-fab
            color="primary"
            aria-label="Back icon">
     <mat-icon class="back-forward">forward</mat-icon>
    </button>
   </div>
  </mat-step>
  <mat-step [stepControl]="userWorkForm" [editable]="isEditable">
   <form [formGroup]="calculationsGroup">
    <ng-template matStepLabel>Характер діяльності</ng-template>
    <app-user-work-info [userForm]="userWorkForm"></app-user-work-info>
    <div class="stepper-buttons">
     <button mat-mini-fab
             color="warn"
             aria-label="Forward icon"
             [disabled]="userWorkForm.invalid"
             (click)="goForward(stepper)">
      <mat-icon>forward</mat-icon>
     </button>
     <button matStepperPrevious
             mat-mini-fab
             color="primary"
             aria-label="Back icon">
      <mat-icon class="back-forward">forward</mat-icon>
     </button>
    </div>
   </form>
  </mat-step>
  <mat-step (interacted)="onDoneInteract()">
   <ng-template matStepLabel>Done</ng-template>
   <app-last-step [energyPerDay]="energyPerDay"
                  [userMIT]="userMIT"
                  [basketForm]="basketForm"
                  (optimizeProductsClick)="optimizeProductBasket()">
```

```
      </app-last-step>
    </mat-step>
  </mat-stepper>
</form>
<ng-template #optimizationLoader>
  <div class="loader-cover">
    <h3 class="mat-headline">Розраховуємо продуктовий кошик...</h3>
    <p>Средній час розрахунку 20-30 секунд</p>
    <app-optimization-process-loader></app-optimization-process-loader>
  </div>
</ng-template>
```

Calculations Service

```
import { Injectable } from '@angular/core';
import { ICalculationsUser, PhysicalActivityLevel, Sex } from '../models/calculations';

const ENERGY_GENDER = {
  [Sex.Male]: 5,
  [Sex.Female]: -161,
}

const Activity = {
  [PhysicalActivityLevel.lowest]: 1.2,
  [PhysicalActivityLevel.low]: 1.375,
  [PhysicalActivityLevel.medium]: 1.55,
  [PhysicalActivityLevel.high]: 1.725,
  [PhysicalActivityLevel.highest]: 1.9,
}

// сервіс для розрахунки добової норми енергії
@Injectable({
  providedIn: 'root'
})
export class CalculationsService {

  constructor() { }

  // формула Міффлін-Сан Жеора
  getDailyCCalAmount(userData: ICalculationsUser): number {
    let energyPerDay =
      (10 * userData.weight) + (6.25 * userData.height)
      - (5 * userData.years)
      + ENERGY_GENDER[userData.sex];

    return Math.round(this.getActivityDependency(energyPerDay, userData.activityLevel));
  }

  //IMT = m:h2  індекс маси тіла
  getMIT(userData: ICalculationsUser): number {
    return +(userData.weight / Math.pow(userData.height / 100, 2)).toFixed(2);
  }


  private getActivityDependency(energyPerDay: number, activityLevel: PhysicalActivityLevel): number {
    return energyPerDay * Activity[activityLevel];
  }
}
```

Calculations Service

```
import { Injectable } from '@angular/core';
import { ICalculationsUser, PhysicalActivityLevel, Sex } from '../models/calculations';

const ENERGY_GENDER = {
  [Sex.Male]: 5,
  [Sex.Female]: -161,
}
const Activity = {
  [PhysicalActivityLevel.lowest]: 1.2,
  [PhysicalActivityLevel.low]: 1.375,
  [PhysicalActivityLevel.medium]: 1.55,
  [PhysicalActivityLevel.high]: 1.725,
```

Calculations State

```
import { IShopProduct } from '../../../../models/products';
import { stateNames } from '../../consts/state-names';
import { Injectable } from '@angular/core';
import { Action, Selector, State, StateContext } from '@ngxs/store';
import { OptimizationService } from '../../../core/servers/optimization.service';
import { LocalStorageService } from '../../../core/servers/local-storage.service';
import { ICalculationsUser } from '../../../modules/calculations/models/calculations';
import { InitBasketState, ResetFormData, SetBasketFormData, SetUserCalcData } from './calculations.actions';
import { CalculationsService } from '../../../modules/calculations/servers/calculations.service';

// опис інтерфесу сховища
export interface ICalculationsState {
  user: ICalculationsUser | null;
  term: number;
  maxSum: number;
  energyAmount: number;
  MIT: number;
  products?: IShopProduct[];
  isActive: boolean;
}

// значення за замовченням
const DefaultCalculationsState: ICalculationsState = {
  user: null,
  term: 0,
  maxSum: 0,
  energyAmount: 0,
  MIT: 0,
  products: [],
  isActive: false,
};

// клас сховаща локальних даних розрахунків
@State<ICalculationsState>({
  name: stateNames.calculations,
  defaults: DefaultCalculationsState
})
@Injectable()
export class CalculationsState {

  readonly STORE_KEY = 'basket-form';

// селектори даних
  @Selector()
```

```
  static calculationsForm(state: ICalculationsState): ICalculationsState {
    return state;
  }

  @Selector()
  static user(state: ICalculationsState): ICalculationsUser | null {
    return state.user;
  }

  @Selector()
  static isActive(state: ICalculationsState): boolean {
    return state.isActive;
  }

  constructor(
    private optimizationService: OptimizationService,
    private calculationsService: CalculationsService,
    private localStorageService: LocalStorageService,
  ) {}

// Обробники подій

// Ініціалізація стейта
  @Action(InitBasketState)
  initBasketState(ctx: StateContext<ICalculationsState>) {
    const state = this.localStorageService.get<ICalculationsState>(this.STORE_KEY);
    if (state && state.isActive) {
      ctx.patchState(state)
    }
  }

// Встановлення даних поточного продуктового кошику
  @Action(SetBasketFormData)
  setAsCurrentBasket(ctx: StateContext<ICalculationsState>, action: SetBasketFormData) {
    ctx.patchState({
      ...action.payload,
      energyAmount: this.calculationsService.getDailyCCalAmount(action.payload.user),
      MIT: this.calculationsService.getMIT(action.payload.user),
      isActive: true,
    });
    this.localStorageService.set(this.STORE_KEY, ctx.getState());
  }

// Запит на виконання розрахунків
  @Action(SetUserCalcData)
  setUserCalcData(ctx: StateContext<ICalculationsState>, action: SetUserCalcData) {
    ctx.patchState({
      user: action.payload,
      energyAmount: this.calculationsService.getDailyCCalAmount(action.payload),
      MIT: this.calculationsService.getMIT(action.payload),
    });
    this.localStorageService.set(this.STORE_KEY, ctx.getState());
  }


// Зброс даних
  @Action(ResetFormData)
  resetFormData(ctx: StateContext<ICalculationsState>) {
    ctx.patchState(DefaultCalculationsState);
    this.localStorageService.set(this.STORE_KEY, ctx.getState());
  }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота_ Kurakina_A_S.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота_ Kurakina_A_S.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Kurakina_Products_server.rar	Архів. Містить коди програми серверу та її залежості
Kurakina_Products_client.rar	Архів. Містить коди програми клієнту та її залежості
Презентація	
Kurakina_A_S.ppt	Презентація кваліфікаційної роботи