НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

на тему: Веб-додаток для комплексної системи схуднення

Виконав (-ла): студент (-ка) __4__ курсу, групи ___ІП-74
(шифр групи)

Симончук Богдан Ярославович
(прізвище, ім’я, по батькові) (підпис)

Керівник асистент Іваніщев Б.В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) професор, доктор технічних наук Сімоненко В. П.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2021 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Симончука Богдана Ярославовича

1. Тема проєкту *Веб-додаток для комплексної системи схуднення*
керівник проєкту асистент Іваніщев Б. В.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від _____ 2021 року №_____
2. Термін здачі студентом закінченого проєкту *21 травня 2021 р.*
3. Вихідні дані до проєкту *технічна документація. клієнт-серверна система, база даних PostgreSQL, теоретичні та статистичні дані*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) *Аналіз предметної області, дослідження аналогів, дослідження методики побудови клієнт-серверних систем на принципі MVC, розробка системи схуднення, інструкція користувача*
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) *схема взаємодії, структурна схема, функціональна схема.*
6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В. П., професор		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	01.12.2020 - 14.12.2020	
2	Вивчення та аналіз завдання	15.12.2020 - 03.02.2021	
3	Огляд існуючих рішень	03.02.2021 - 15.02.2021	
4	Розробка архітектури та загальної структури системи	15.02.2021 - 02.03.2021	
5	Розробка структур окремих підсистем	02.03.2021 - 30.03.2020	
6	Програмна реалізація системи	30.03.2021 - 30.04.2021	
7	Виправлення помилок	1.05.2021 - 10.05.2021	
8	Оформлення пояснювальної записки	10.05.2021	
9	Передзахист	01.06.2021	
10	Захист	18.06.2021	

Студент Богдан СИМОНЧУК

Керівник Богдан ІВАНІЩЕВ

АНОТАЦІЯ

У даній бакалаврській роботі детально розглянуто проблему ожиріння в сучасному світі, проведений розбір концепцій, на яких базується процес схуднення, зроблено аналіз та порівняння існуючих веб-систем схуднення. Були проаналізовані їхні слабкі і сильні сторони. На основі цієї інформації було розроблено перелік вимог до системи схуднення, що лягли в основу розробленої клієнт-серверної систему, за допомогою якої може значно полегшитись процес схуднення та підрахунку обігу калорій в організмі.

ANNOTATION

In this bachelor's thesis the problem of obesity in the modern world is considered in detail, the concepts on which the weight loss process is based are analyzed, the analysis and comparison of existing weight loss web systems is made. Their strengths and weaknesses were analyzed. Based on this information, a list of requirements for the weight loss system was developed, which formed the basis of the developed client-server system, which can greatly facilitate the process of weight loss and calorie counting in the body.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Завдання на дипломний проєкт	2		
			Відомість дипломного проєкту	1		
	А4	***ІАЛЦ.467200.002 ТЗ***	Технічне завдання	3		
	А4	***ІАЛЦ.467200.003 ПЗ***	Пояснювальна записка	70		
	А4	***ІАЛЦ.467200.004 Д1***	Схема взаємодії	1		
	А4	***ІАЛЦ.467200.005 Д2***	Структурна схема	1		
	А4	***ІАЛЦ.467200.006 Д3***	Функціональна схема	1		
	А4	***ІАЛЦ.467200.007 Д4***	Лістинг програми	20		

					ІАЛЦ.467200.001 ОА	
Зм	*Апк.*	*№ докум.*	*Підп.*	*Дата*		
Розроб		Симончук Б.Я.			*Веб-додаток для комплексної системи схуднення* Опис альбому	***НТУУ “КПІ” ФІОТ ІП-74***
Перев		Іваніщев Б.В.				
Реценз.						
Н.Контр		Сімоненко В. П.				
Затверд.		Стіренко.С. Г.				

ТЕХНІЧНЕ ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ

на тему: «Веб-додаток для комплексної системи схуднення»

Київ – 2021

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування : «Веб-додаток для комплексної системи схуднення». Область застосування: альтернатива існуючим веб-додаткам, які пропонують сервіс для схуднення та обігу калорій в організмі

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврської дипломної роботи, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут» імені Ігоря Сікорського.

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є розробка системи для схуднення та обігу калорій в організмі людини.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література по комп’ютерним системам, публікації в періодичних виданнях та мережі Інтернет.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

- Проводити фіксацію та обрахунок обігу калорій в організмі з можливістю подальшого моніторингу;
- Можливість вибору продуктів та фізичних навантажень з бази даних;

- Коректність та актуальність інформації, яка надається користувачам;
- Надавати користувачам індивідуальний план схуднення.

5.2 Вимоги до програмного забезпечення

- Операційна система Linux, macOS, Windows;
- Доступ до мережі інтернет;
- Браузер Google Chrome, Firefox, Safari чи Microsoft Edge.

5.3 Вимоги до апаратної частини

- Джерело живлення 220В для локального комп’ютера;
- Доступ до мережі Інтернет;
- Локальний або хмарний комп’ютер.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	20.12.2020
Складання і узгодження технічного завдання	03.01.2021
Проектування програмного забезпечення	16.02.2021
Програмна реалізація продукту	30.04.2021
Налагодження та виправлення помилок	10.05.2021
Оформлення документації дипломної роботи	11.05.2021

Пояснювальна записка
до дипломного проєкту
на тему: «Веб-додаток для комплексної системи схуднення»

Київ – 2021 року

Зміст

ВСТУП

У нашому сучасному світі дедалі гострішою постає проблема ожиріння та зайвої ваги. Через стрімкий розвиток технологій та удосконалення встановлених виробничих потужностей і процесів, стало можливим швидко виробляти величезну кількість їжі. А це в свою чергу дало можливість доступу до дешевої та висококалорійної їжі з високим вмістом солі, цукру та жирів набагато більшій кількості людей порівняно з минулим. В поєднанні з поширеним малорухливим способом життя все це неодмінно призводить до ожиріння та проблем з зайвою вагою, а в гіршому випадку – ще й до різноманітних проблем зі здоров'м.

Згідно зі статистикою [13] Всесвітньої організації охорони здоров'я, рівень ожиріння у всьому світі виріс втричі з 1975 року. А в 2016 році більше, ніж 1,9 мільярда дорослих у віці 18 років і старше мали лишню вагу, з них більше 650 мільйонів страждало ожирінням.

Беручи до уваги всю цю інформацію, головною ціллю даної роботи є створення веб-додатку з функціоналом, який зможе допомогти його користувачам залишатись в формі, пропонуючи можливість контролю обігу калорій та таких компонентів їжі, як білки, жири, вуглеводи, а також практичні рекомендації, спрямовані на схуднення.

Практична цінність полягає в розробці зрозумілого для користування та універсального веб-додатку, доступного без обмежень, в той час як аналоги зазвичай пропонують такий сервіс платно чи з обмеженнями і реалізують подібний функціонал за допомогою мобільних додатків.

Даний веб-додаток можна використовувати як довідник з інструкціями по здоровому харчування і здоровому способу життя, як калькулятор обігу калорій, білків, жирів, вуглеводів, а також як сервіс для довготривалого схуднення

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ

1.1 Опис завдання

Сьогодні, популярність мережі інтернет та браузерів є надзвичайно високою і з кожним днем вона тільки збільшується. Найпопулярніші веб-браузери працюють практично на всіх типах комп'ютерів та операційних систем. На даний момент абсолютна більшість браузерів безкоштовна, вони легкі у використанні та не вимогливі, а головне – доступні кожному. В той час, як різноманітні десктопні чи мобільні програми повинні бути спочатку встановлені на пристрій перед використанням, веб-додатки надають можливість миттєвого їх використання. Програми вимагають хорошу конфігурацію пристрою користувача, веб-додатки можуть працювати на пристроях навіть з низькою конфігурацією. Також, при збоях в роботі вся програма виходить з ладу.

Враховуючи цю інформацію, а також той факт, що більшість додатків для схуднення розроблені під мобільні платформи [14], було вирішено створити веб-версію такого додатку, що також дасть можливість більш тонкого контролю над процесом.

Тож, завданням даної роботи є створення клієнт-серверного додатку, що стане в нагоді при схудненні і контролі обігу деяких поживних речовин і калорій.

Для створення такого додатку, слід проаналізувати, наскільки ті чи інші можливості будуть корисними у такій системи. Для цього слід зрозуміти, як це працює на біологічному рівні і виділити ключові умови успішного схуднення.

Різниця калорій, що надходять в організм і спалюються – проста та вічна формула для створенню дефіциту калорій в організмі. Звучить просто: слід спалити калорій більше тих, що надходять до організму. Це перевірена

часом тактика схуднення і згідно з дослідженнями [15], є більш ефективною, ніж покладатись тільки на фізичні вправи.

Хоча може здатися, що істотне скорочення споживання калорій автоматично пришвидшує втрату жиру, це насправді працює проти людини у довгостроковій перспективі, бо уповільнює метаболізм і призводить до того, що організм не захоче спалювати запаси жиру, оскільки вважає, що перебуває в так званому "стані голоду". Крім того, важко підтримувати сильний дефіцит калорій. Коли тіло не отримує енергії, необхідної для функціонування, підвищується рівень гормонів голоду. Рекомендується почати з 15-20% дефіциту калорій. Головне у створенні дефіциту калорій - спалити трохи більше (або з'їсти трохи менше), ніж вимагає ваше тіло для підтримки ваги. Калорії, що спалюються під час фізичних вправ + активність без фізичних вправ + рівень метаболізму, повинні бути більшими, ніж калорії, споживані з їжею, щоб досягти втрати ваги. В загальному випадку, потрібно створити дефіцит у 250–500 калорій на день, щоб втрачати від 200 до 400 грам на тиждень [11].

Оскільки на метаболізм (калорії, які спалюються в спокої) припадає 60–70% калорій, спалених протягом дня, важливо розрахувати цей показник як вихідну точку для створення дефіциту калорій. Те, скільки калорій тіло спалює в стані спокою, залежить від багатьох змінних, таких як, вік, стать, висота, вага та м'язова маса. Звичайно, на це також впливає і безліч інших показників, таких як генетика, м'язова маса, рівень жиру в тілі, гормони, стан здоров'я та структура тіла. Оскільки за звичайних умов точно і правильно визначити всі показники не є можливим, застосовується формула Міффліна-Сан-Жеора [9][10], в якій застосовуються основні та доступні показники

Отож, для контролю власної ваги необхідно знати, скільки калорій, білків, жирів, вуглеводів споживається і спалюється впродовж дня. Для контролю ведення таких підрахунків надзвичайно корисною буде

можливість зберігати і обробляти таку інформацію, щоб розуміти, як виконується план і чи потрібно збільшити/зменшити споживання тих чи інших продуктів.

Іншим параметром системи буде можливість обрати ціль – бажану вагу, до якої треба схуднути. На основі цих даних обирається програма схуднення, що являє собою графік фізичних вправ, а також по можливості оптимальну дієту або норму калорій, білків, жирів, вуглеводів, які повинні надходити до організму впродовж курсу схуднення.

Ще одною особливістю такого застосунку може бути наявність різноманітних графіків прогресу схуднення, наприклад, графіки зміни ваги кожного дня, графіки обігу калорій, жирів, білків, вуглеводів та за можливості інші корисні графіки.

Також непоганим рішенням було б реалізувати можливість використання додатку, як довідник з інструкціями по здоровому харчування і здоровому способу життя

З метою корегування та покращення функціональності додатку, було вирішено проаналізувати поточний ринок аналогів та існуючих рішень.

1.2 Огляд існуючих рішень

1.2.1 MyFitnessPal

Рис. 1.1. MyFitnessPal Logo

MyFitnessPal (рис. 1.1) - додаток для підрахунку калорій, який містить свою масивну базу даних про їжу. Наприкінці кожного дня додаток надає інформацію, що може бути використана для корегування власного раціону харчування і кращого розуміння, як досягти своїх цілей зі зменшенням ваги. MyFitnessPal пропонує як безкоштовний рівень обслуговування, так і платний преміум рівень. Преміум коштує 10 доларів на місяць або 50 доларів на рік.

За допомогою безкоштовного облікового запису можна відстежувати продукти харчування та калорії, активність, вагу та кілька інших показників. Безкоштовна версія MyFitnessPal підходить для того, щоб допомогти комусь підтримувати свою вагу після програми схуднення, а також щоб допомогти схуднути за допомогою підрахунку калорій, коли не потрібна допомога програми схуднення для досягнення успіху.

Далі розглядаються основні компоненти додатку.

Реєстрація (рис.1.2). Розпочинається зі сторінки, на якій вводяться електронна пошта користувача та пароль для входу в профіль

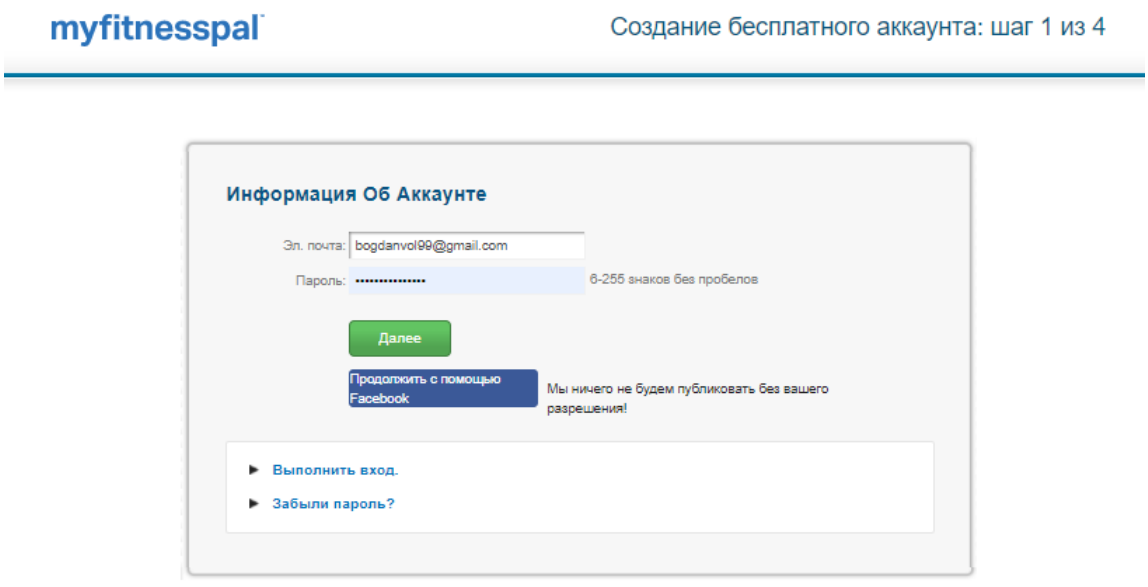


Рис. 1.2. Сторінка реєстрації MyFitnessPal

Далі необхідно ввести інформацію про свою вагу, зріст, стать, дату народження, спосіб життя, як часто планується займатись фізичними навантаженнями і яка ціль схуднення переслідується (рис. 1.3)

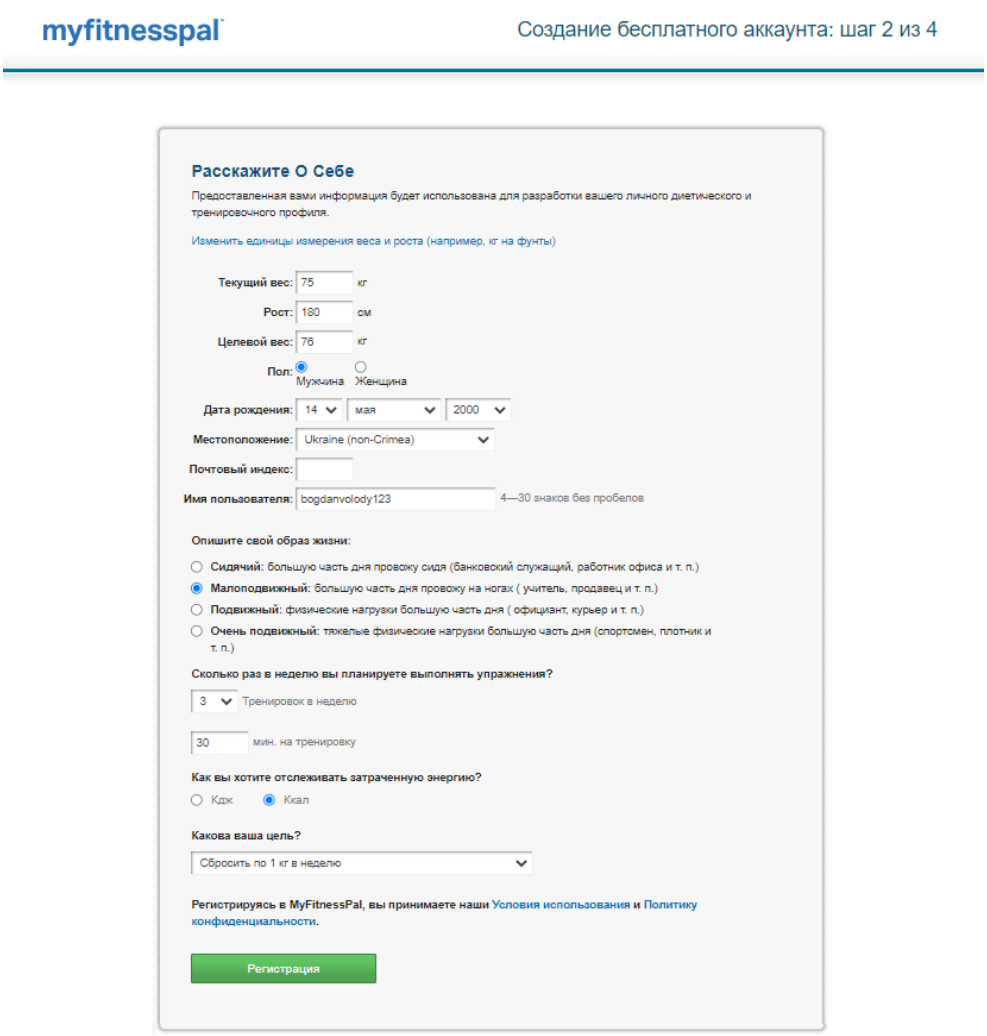


Рис. 1.3. Друга сторінка реєстрації

Після цього створюється особиста сторінка користувача і виводиться рекомендований план по отриманню калорій і прогнозований план схуднення (рис. 1.4)

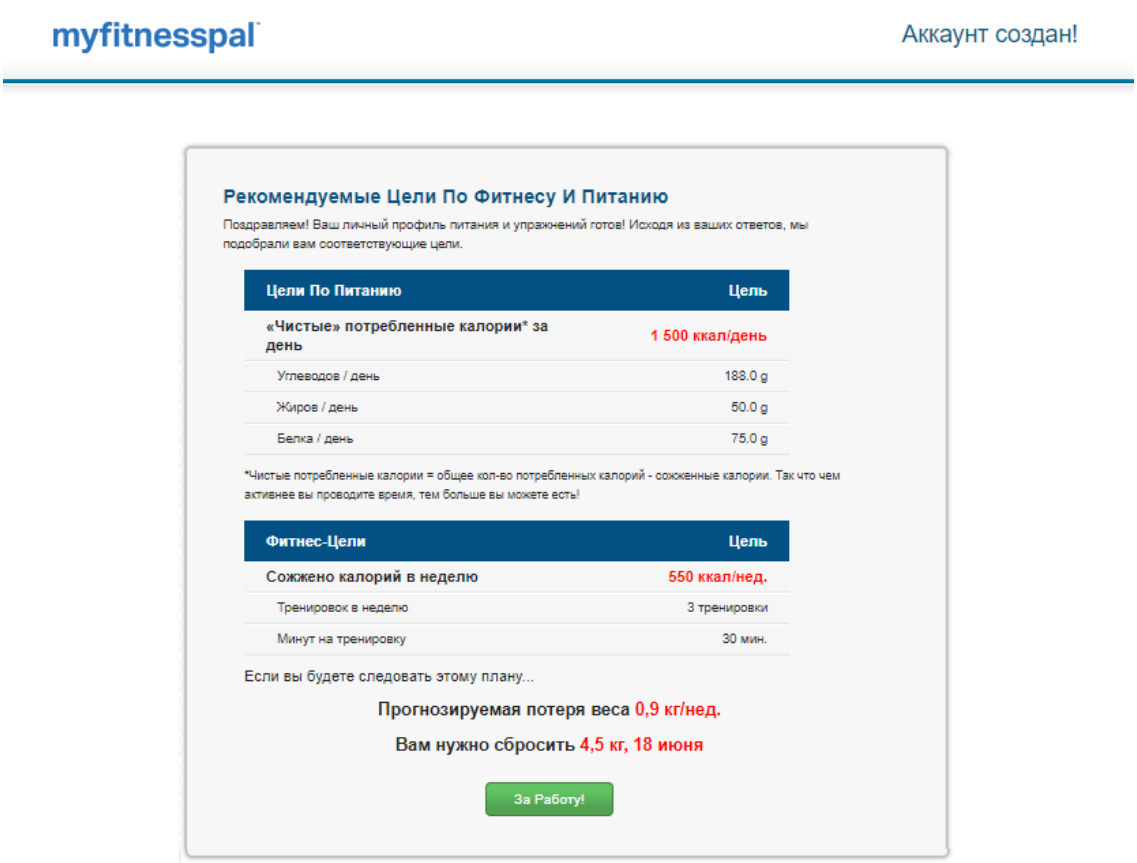


Рис. 1.4. Рекомендований план схуднення

Головне меню (рис. 1.5). Це головна сторінка, на якій виводиться денний прогрес отримання калорій з кнопками додавання вправ, що виконуються впродовж дня і страв, що з'їдаються впродовж дня. Також тут міститься нагадування про підтвердження своєї електронної адреси і корисні посилання на статті з форумів

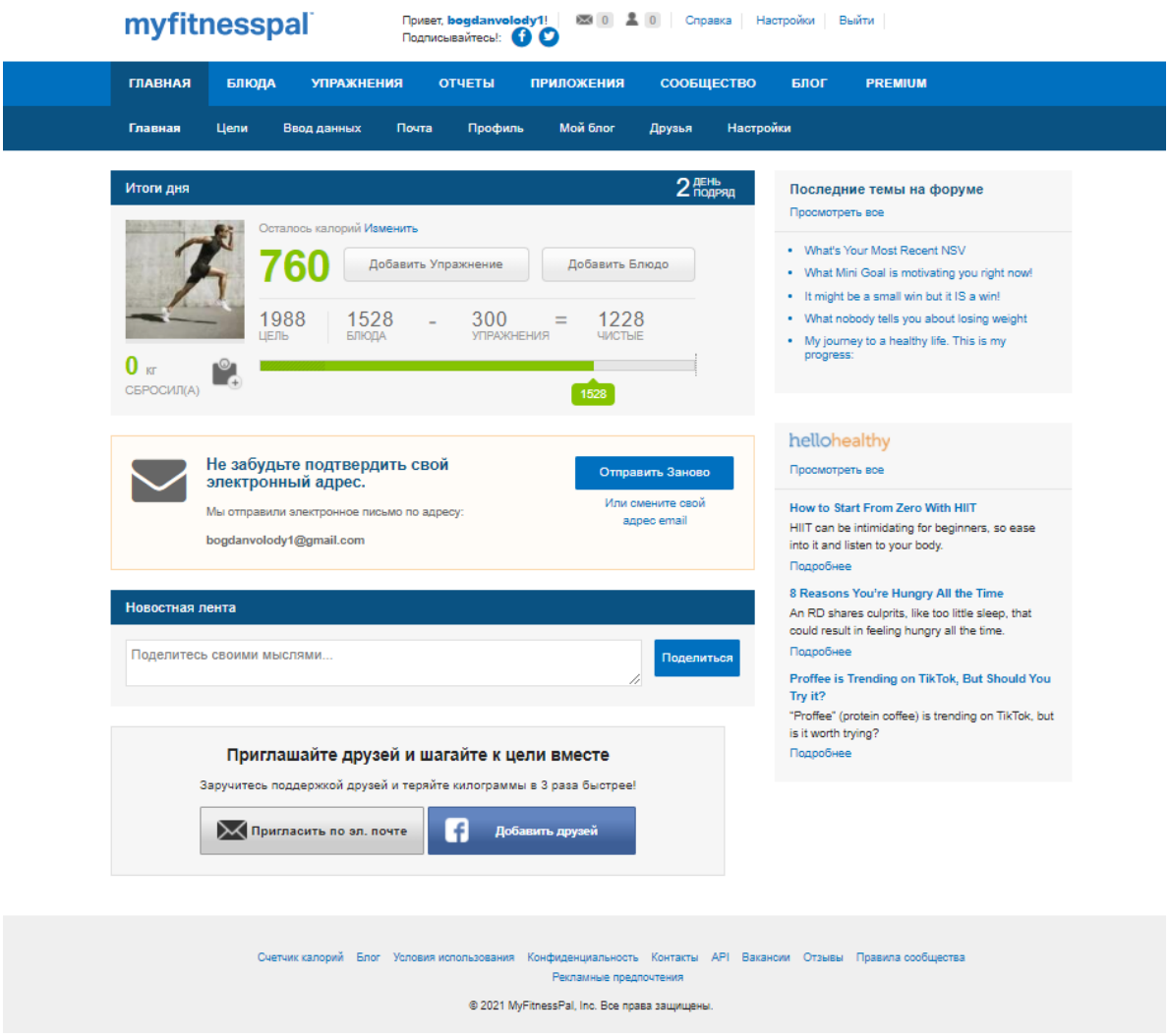


Рис. 1.5. Головне меню

При натисканні на кнопку додавання страви відкривається сторінка (рис. 1.6) , на якій пропонується зручний сервіс додавання страв на кожен прийом їжі впродовж дня. Страви можна вибрати з локальної бази даних страв, а також додати свою унікальну з вказанням різноманітних характеристик страви. Також на цій сторінці відображається таблиця, що ілюструє порівняльні показники компонентів їжі, що надійшли до користувача впродовж дня, а також показники, необхідні для дотримання плану, і інша корисна інфографіка .

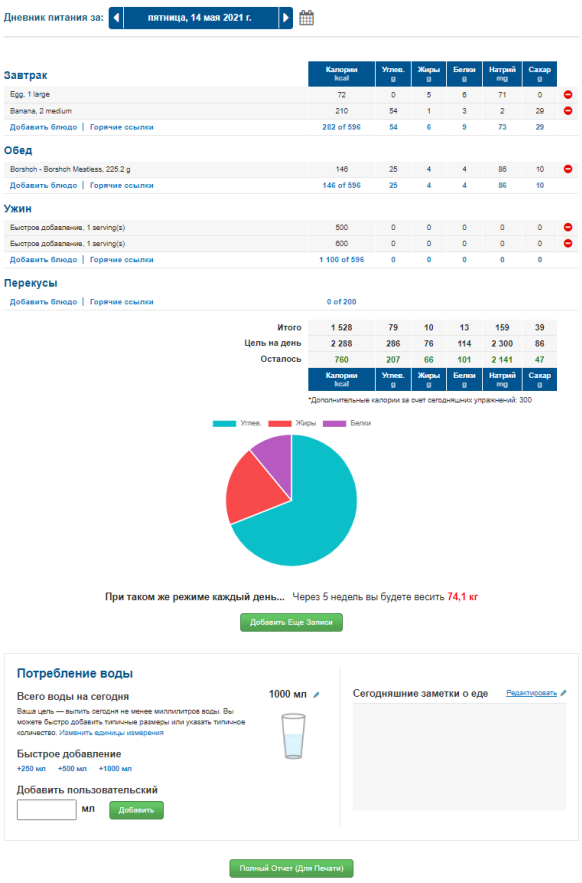


Рис. 1.6. Денна статистика

Додавання страви з бази даних (рис. 1.7)

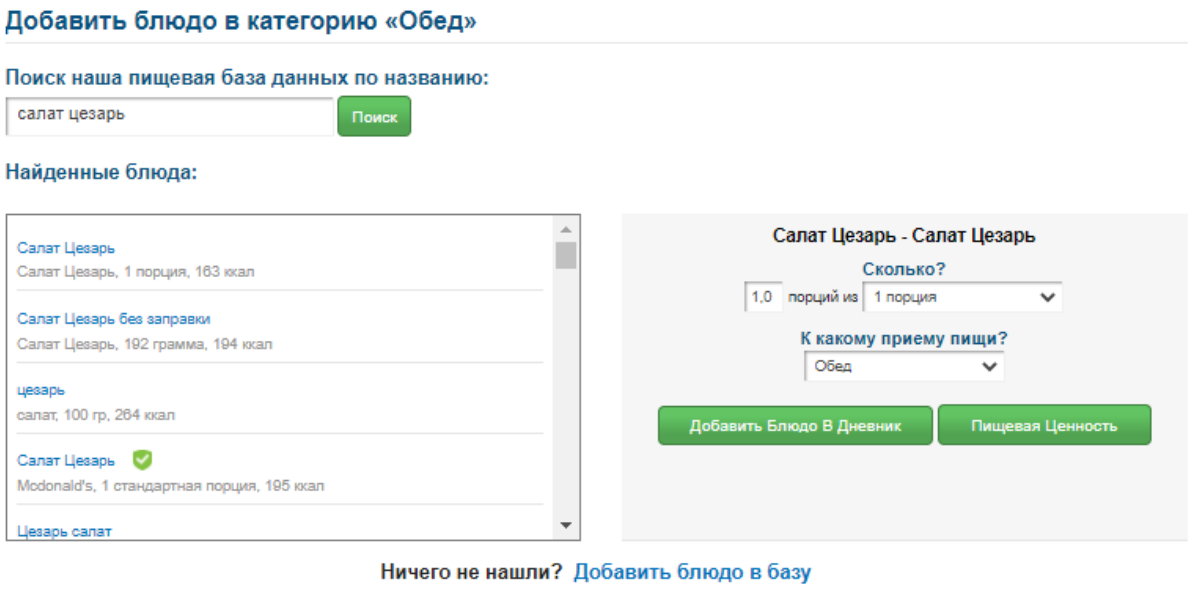


Рис. 1.7. Додавання страви з бази даних

Також передбачена можливість додати власну унікальну страву (рис. 1.8) в разі, якщо необхідної не було знайдено в базі даних. Для цього користувачу слід обов'язково назвати страву, визначити розмір порції та енергетичну цінність у кілокалоріях. Всі інші показники – необов'язкові та можуть бути опущені в разі необхідності

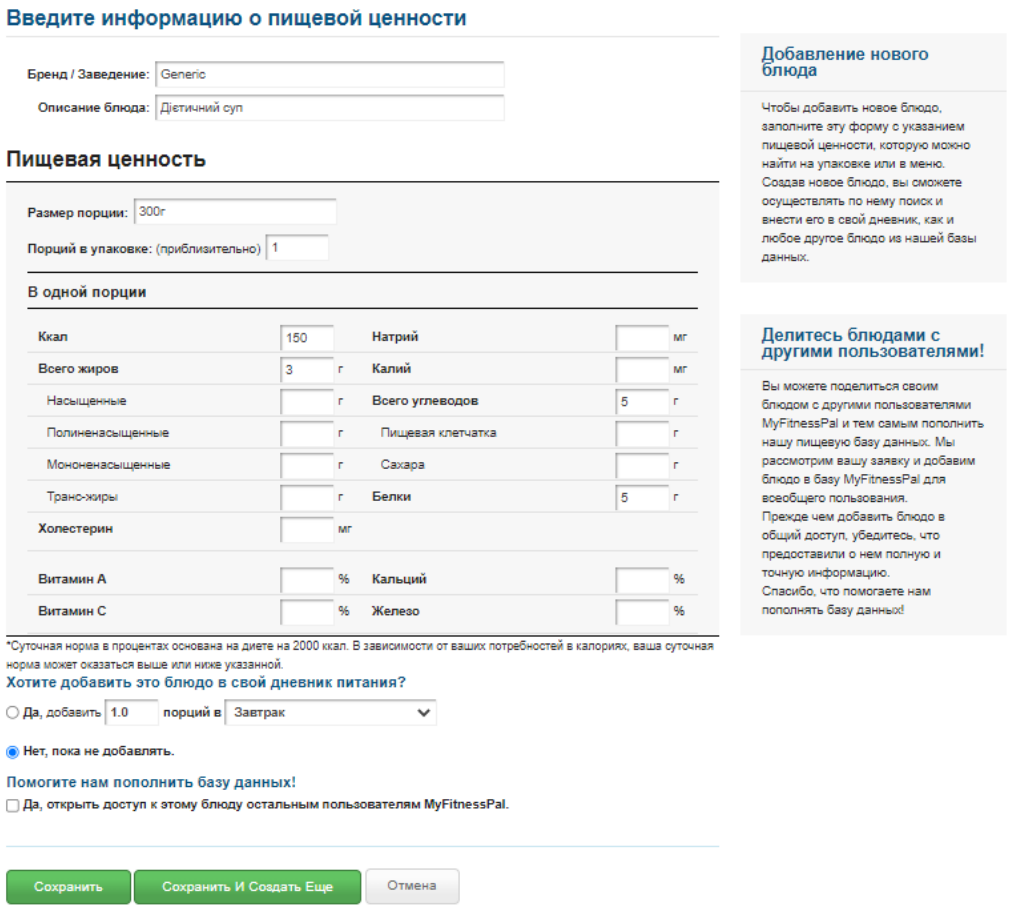


Рис. 1.8. Додавання власної страви

Аналогічний сервіс пропонується для добавляння фізичних вправ (рис. 1.9) : користувач може обрати вправу з локального сховища (бази даних) вправ, а також визначити свою унікальну вправу з вказанням ключових показників, таких як протяжність у хвилинах, кількість підходів, кількість повторень і підходах, а також кількість енергії , що спалюється в результаті успішного здійснення такої вправи.

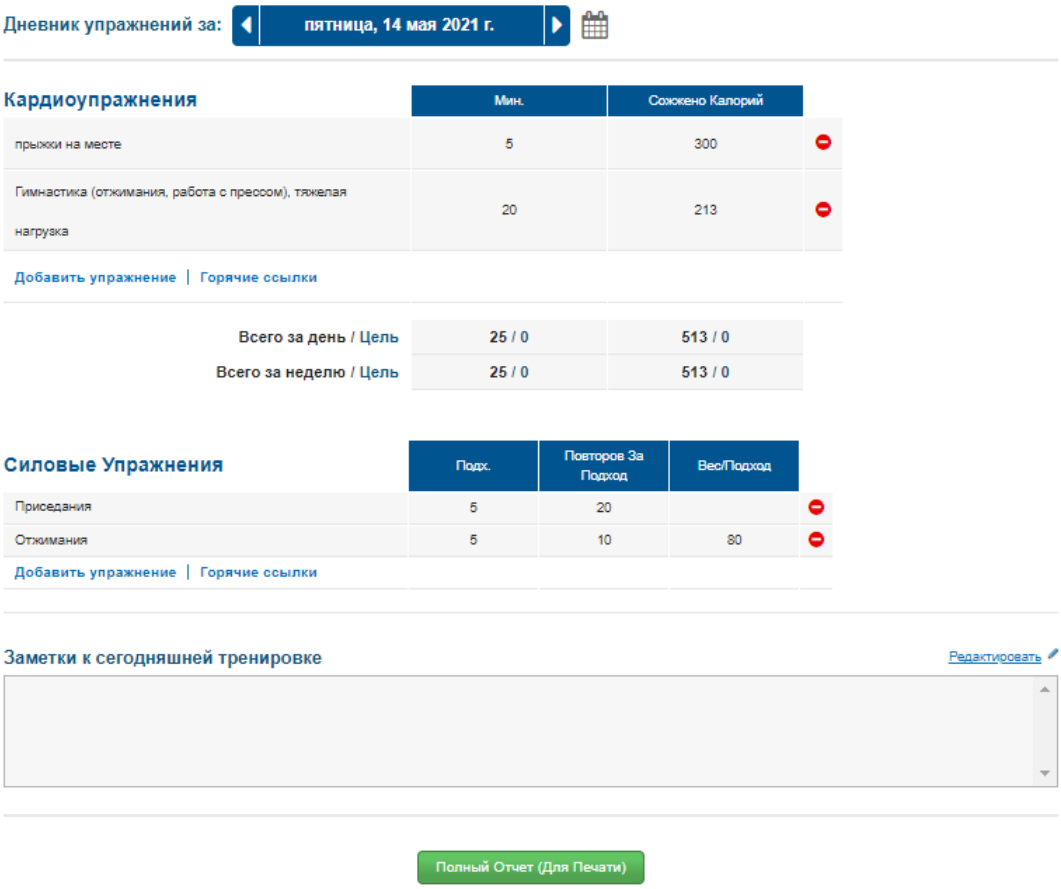
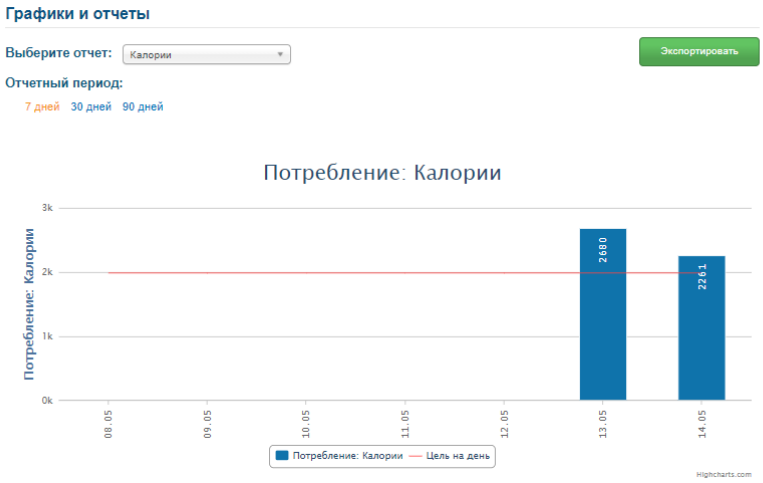


Рис. 1.9. Додавання вправ

Сторінка, на якій відображаються різноманітні графіки (рис. 1.10): прогрес схуднення, обіг калорій і так далі



Самые богатые калориями блюда

Последние 7 дней питания	Съедено раз	Всего
Быстрое добавление	x 1 =	2490
Быстрое добавление	x 1 =	600
Быстрое добавление	x 1 =	500
Быстрое добавление	x 1 =	420
Banana	x 1 =	210
жаренная картошка - жаренная картошка	x 1 =	190
Салат Цезарь - Салат Цезарь	x 1 =	163
Generic - Дієтичний суп	x 1 =	150
Borshch - Borshch Meatless	x 1 =	146
Egg	x 1 =	72

Рис. 1.10. Графіки калорій

1.2.2 Lose it!

Рис. 1.11. Lose it! Logo

Lose it! (рис. 1.11) – інша популярна програма для схуднення. Процес реєстрації схожий з попереднім аналогом MyFitnessPal : вказуються деякі особисті дані (рис. 1.12 та 1.13) , а також вказується, чи присутній у

користувача досвід використання подібних систем у минулому. При введенні даних чи обранні якогось варіанту відповіді система автоматично виводить наступне вікно для подальшого збору інформації від користувача

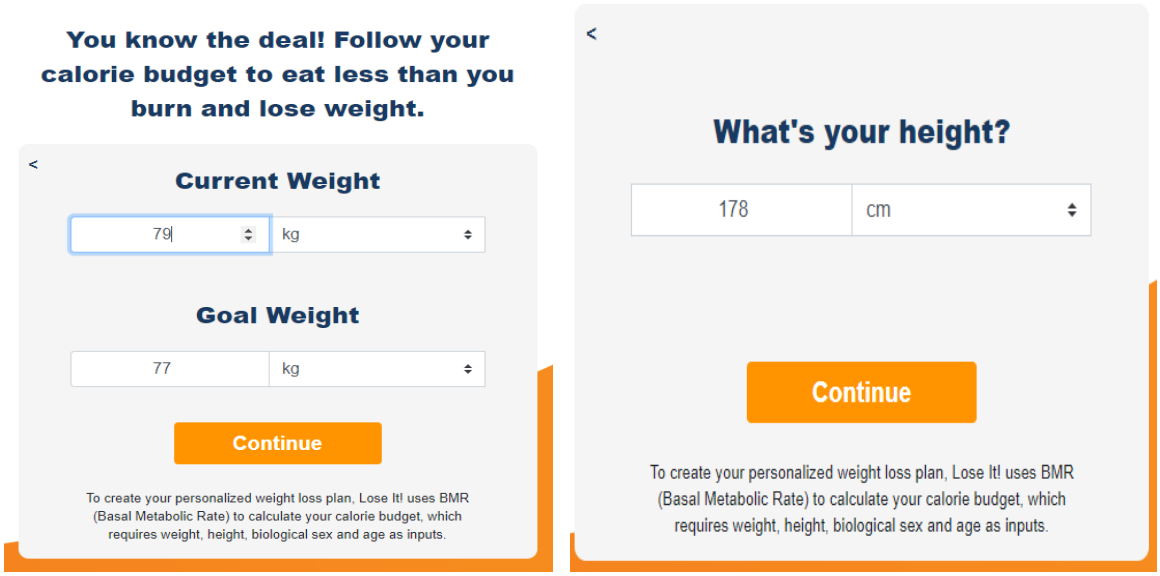


Рис. 1.12. Поля реєстрації

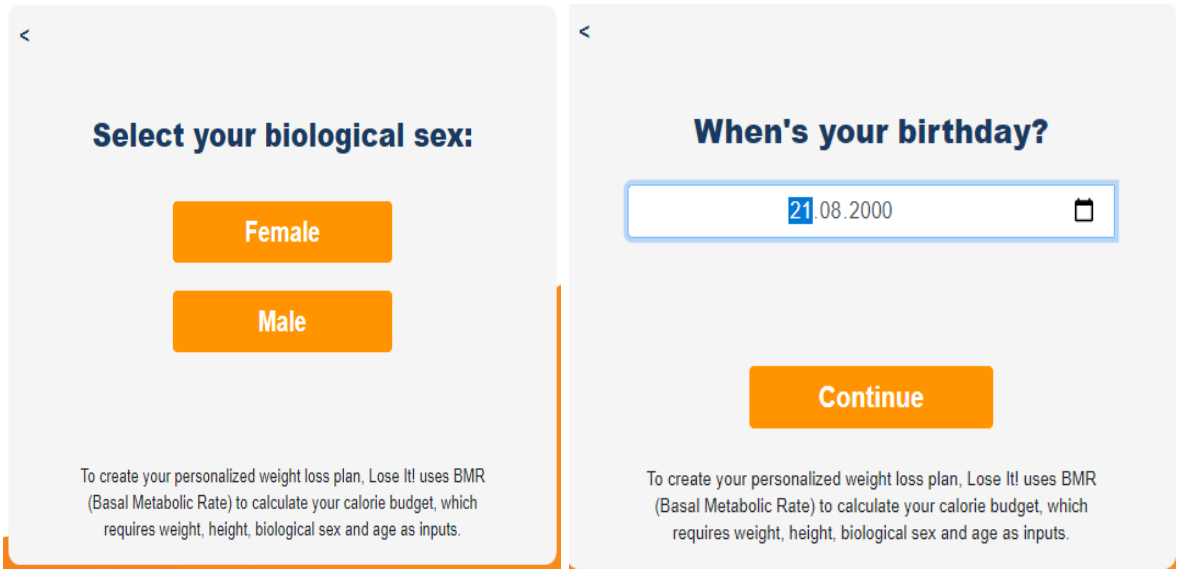


Рис. 1.13. Додаткові поля реєстрації

Після заповнення ряду необхідних полів веб-додаток підводить підсумок (рис. 1.14) попередньо введених даних і виводить план схуднення

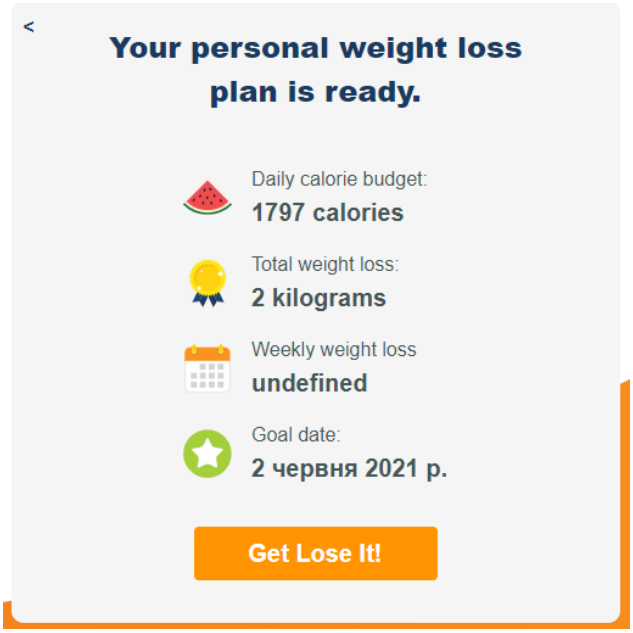


Рис. 1.14. Персональний план

Далі слідує сторінка реєстрації (рис. 1.15), яка вимагає вводу імені та даних для входу в особистий обліковий запис

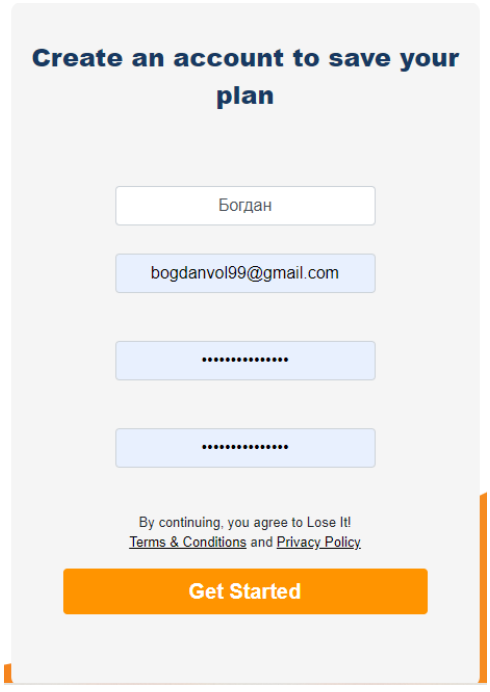


Рис. 1.15. Сторінка створення профілю

Завершальна сторінка реєстрації (рис. 1.16), на якій можна внести корективи для попередньо вказаних показників, а також вказати свою

цільову вагу після схуднення, а також бажаний темп схуднення, після чого обраховується орієнтовна дата кінця запланованого процесу схуднення.

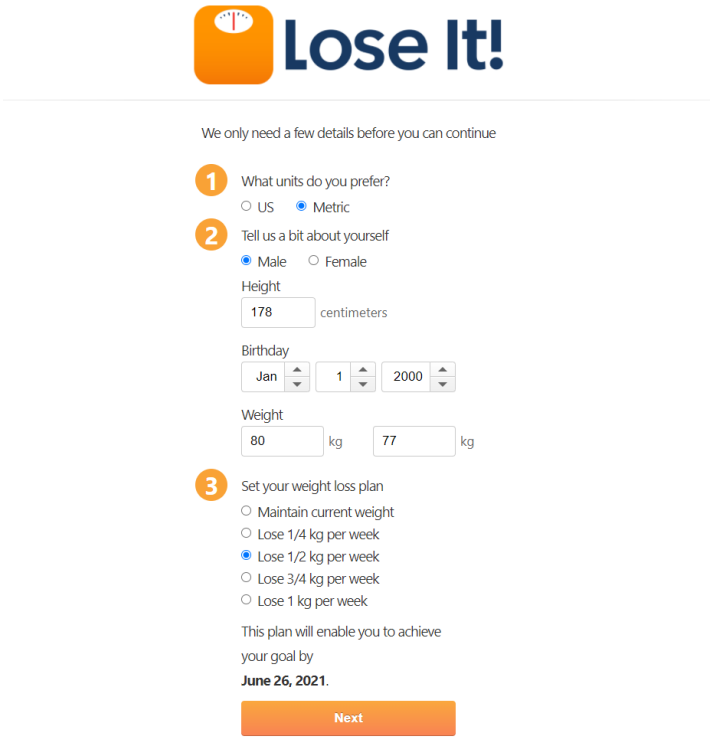


Рис. 1.16. Завершальна сторінка реєстрації

Після цього система перекидає нас на головну сторінку (рис. 1.17), яка дуже подібна за функціоналом з попереднім аналогом, за винятком того, що додавати страви і фізичні вправи можна відразу на цій же сторінці, користуючись зручним пошуком по локальній базі даних страв і вправ. З правого боку відображається інтуїтивно зрозуміла шкала отриманих за день калорій, на якій також зрозуміло, яку кількість калорій ще можна отримати, не порушуючи план. Нижче можна оновлювати показник поточної ваги задля можливості відслідковування прогресу схуднення. Ще нижче – різні активності та корисні посилання для зв’язку зі спільнотою.

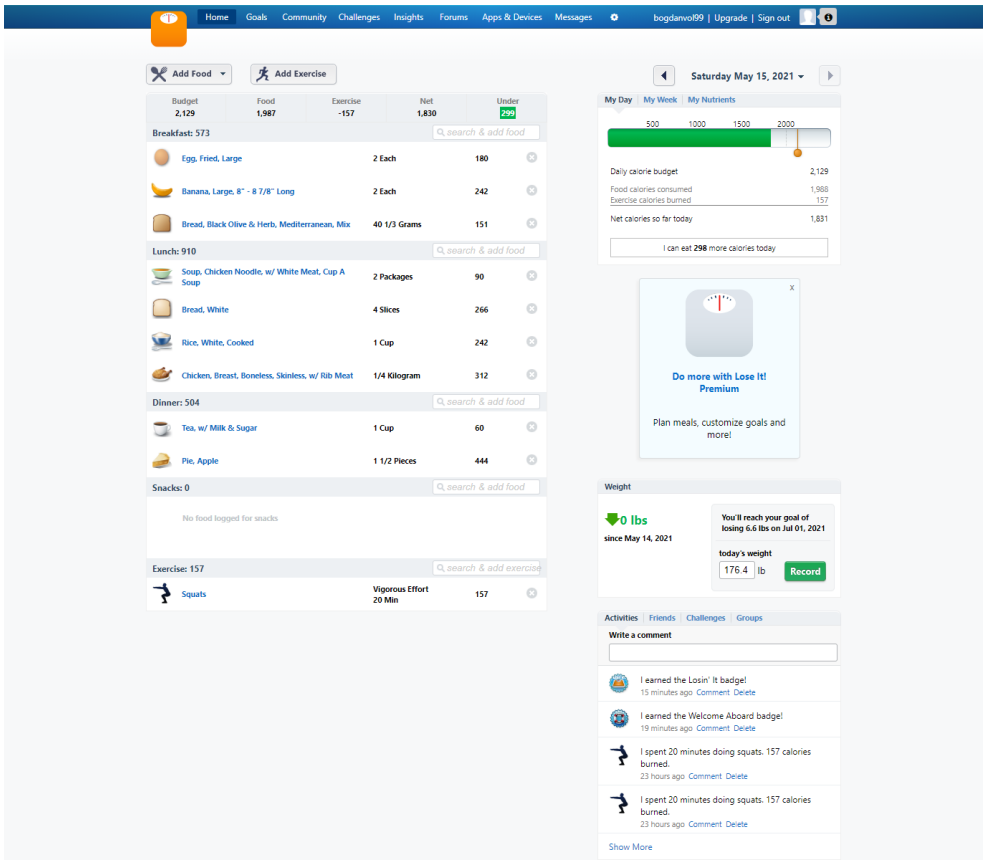


Рис. 1.17. Головна сторінка

Є можливість корегувати кількість одиниць/порцій страви і відразу бачити характеристики цієї страви (рис. 1.18)

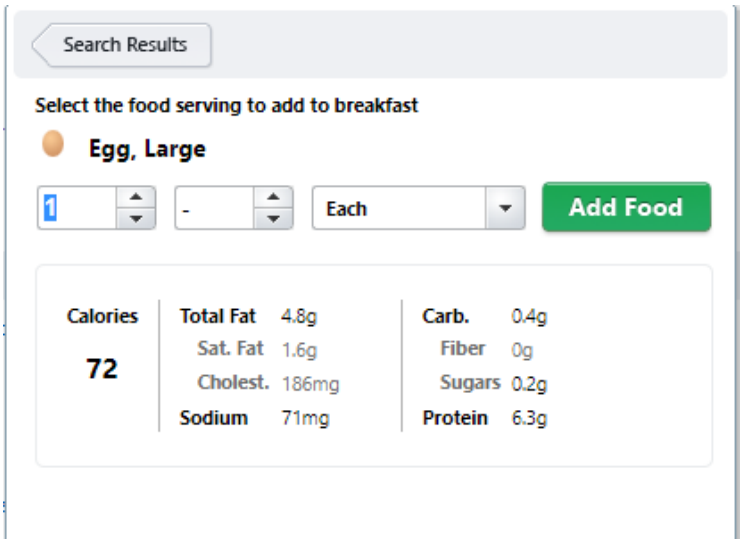


Рис. 1.18. Характеристики страви

Також надається можливість вибору конкретної дати для відображення історії обігу калорій у конкретний день (рис. 1.19)

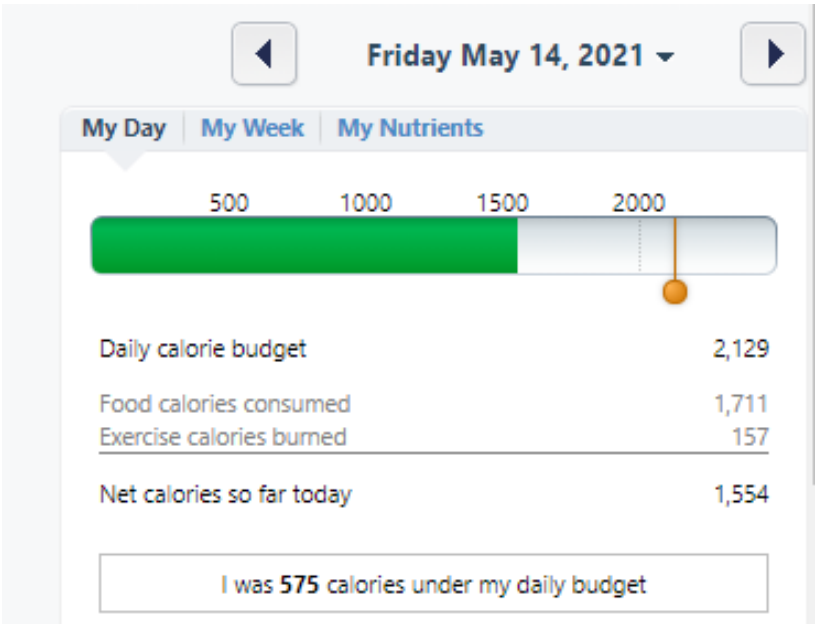


Рис. 1.19. Шкала калорій

Спливаюче вікно для додавання нових тренувань (рис. 1.20)

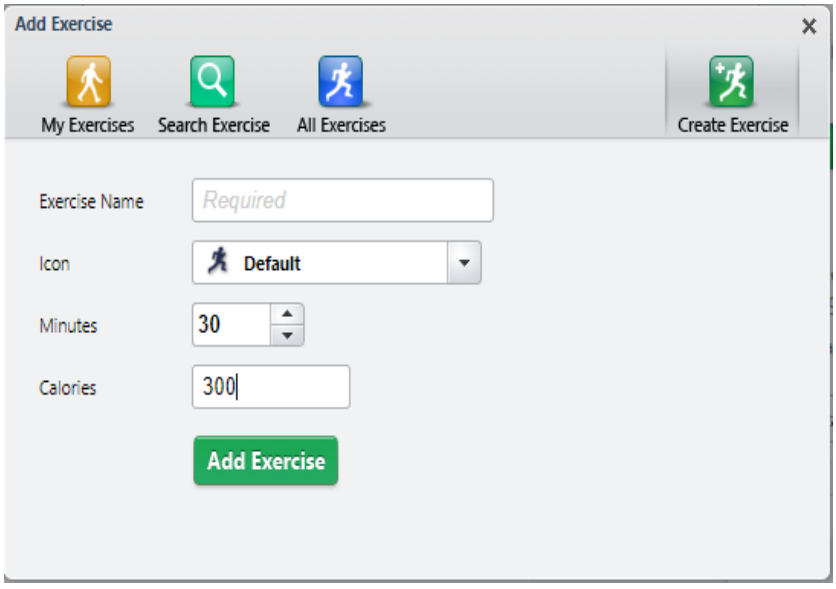


Рис. 1.20. Додавання нових тренувань

А також страв (рис. 1.21)

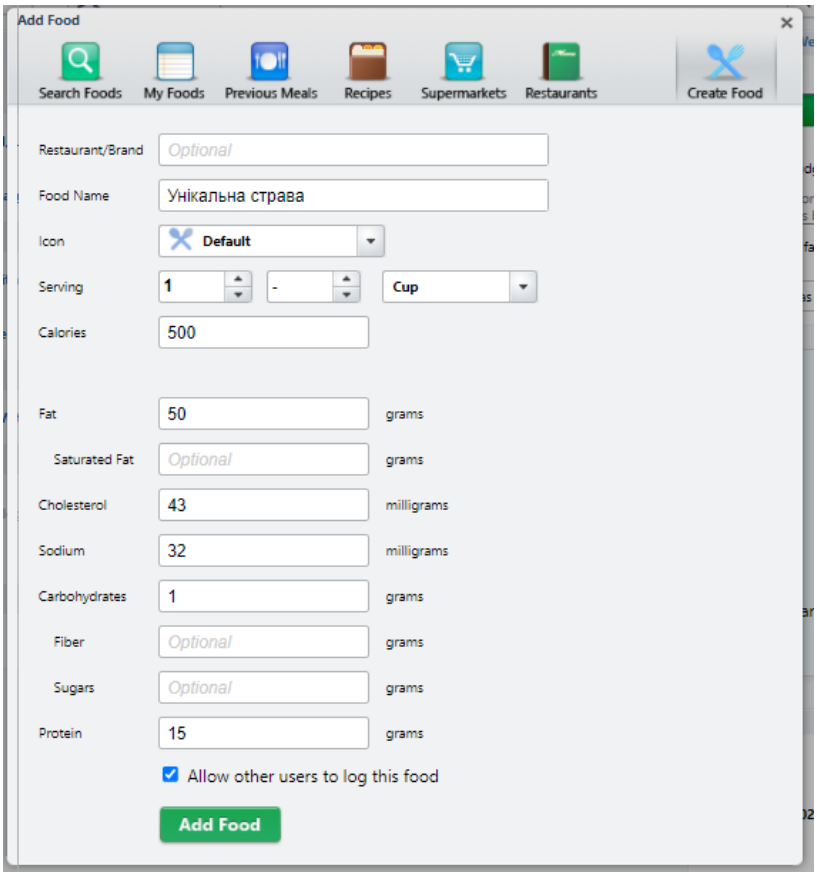


Рис. 1.21. Додавання нових страв

1.2.3. Lose Weight in 30 Days

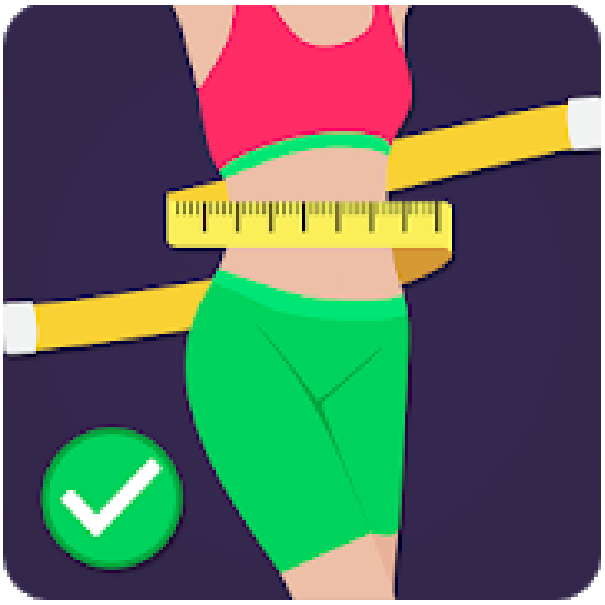


Рис. 1.22. Lose Weight in 30 Days Logo

Програма Lose Weight in 30 Days (рис. 1.22) пропонує план тренувань для фітнесу та дієту, щоб допомогти схуднути за 30 днів. План тренувань містить вправи на прес, сідниці, верхню частину тіла та ніг. В даній програмі можна відстежувати свій прогрес у схудненні та кількості спалених калорій. Програма пропонує 30-денну програму занять для :

- Втрати ваги і підтримання фізичної форми
- Тренування пресу
- Тренування ніг
- Тренування сідниць

А також 60 – денну програму, яка включає в себе тренування кардіо та тіла загалом. Тренування проходять з паузами кожні 3 дні

Сторінка редагування профілю (рис.1.23) дозволяє вказати одиниці виміру ваги та зросту, а також змінити власні дані і цільову вагу

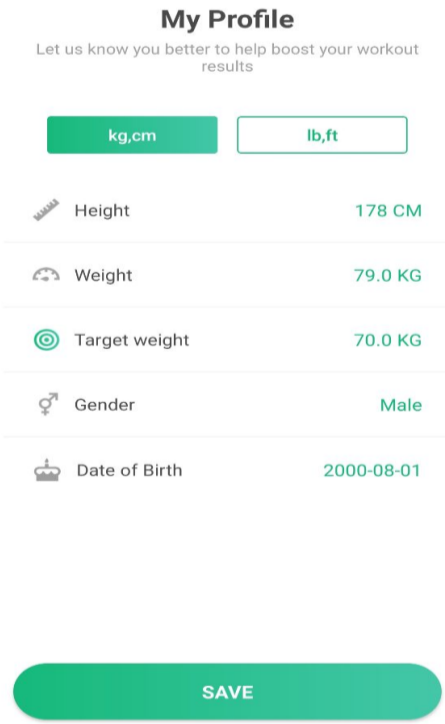


Рис. 1.23. Редагування профілю

Програма пропонує 30-денний курс фізичних навантажень (рис.1.24)

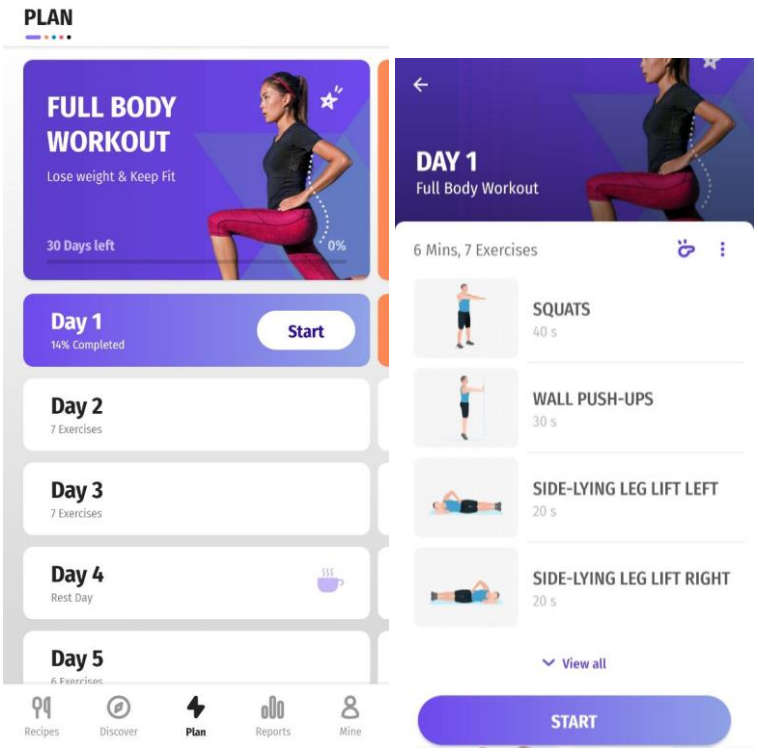


Рис. 1.24. 30-денний курс тренувань

Раціон харчування на кожен день впродовж 30-денної програми (рис. 1.25)

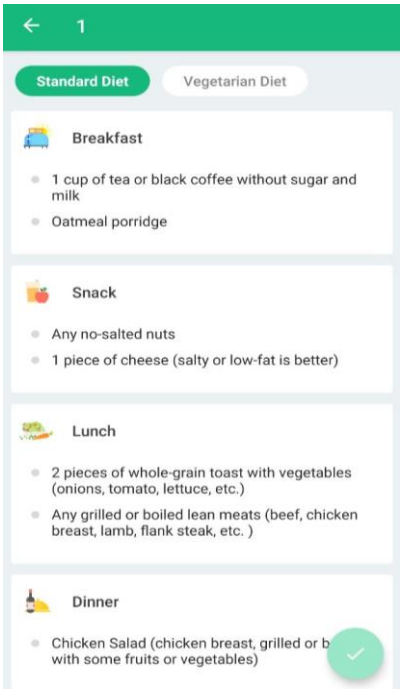


Рис. 1.25. 30-денний дієта

А також графіки калорій прогресу схуднення (рис. 1.26)

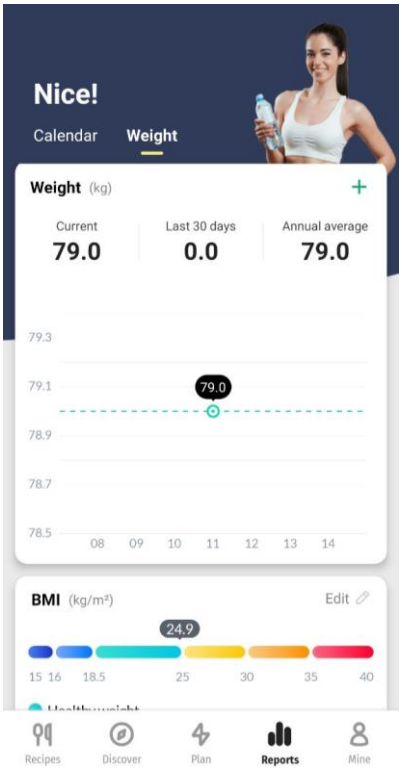


Рис. 1.26. Графіки

1.3 Після аналізу

Проаналізувавши аналоги, можна виділити головну особливість, яка не згадувалась раніше : вибір страв та вправ з існуючої бази даних. Задля виконання цих умов в роботі передбачається використання API публічної бази даних їжі, наприклад Департаменту Агрокультури Сполучених Штатів [16] чи бази даних Open Food Facts [17], створеної і підтримуваної волонтерами. Отже, тепер можна підсумувати зібрану інформацію у вигляді порівняльної таблиці 1.1

Таблиця 1.1 – Порівняльна характеристика з аналогами

	MyFitnessPal	Lost it!	Lose Weight in 30 Days	Мій веб-застосунок
Реєстрація, авторизація	Підтримує	Підтримує	Підтримує	Підтримує

Продовження таблиці 1.1

Вказання особистих показників	Підтримує	Підтримує	Підтримує	Підтримує
Калькулятор калорій	Підтримує	Підтримує	Не підтримує	Підтримує
Наявність історії	Підтримує	Підтримує	Підтримує	Підтримує
Графіки	Підтримує	Підтримує	Підтримує	Підтримує
Рекомендований план схуднення	Підтримує	Підтримує	Не підтримує	Підтримує
Додавання власних страв та вправ	Підтримує	Підтримує	Не Підтримує	Підтримує
Вибір страви чи вправи з бази даних	Підтримує	Підтримує	Не Підтримує	Підтримує
Практичні рекомендації по схудненню	Не Підтримує	Не Підтримує	Підтримує	Підтримує
Рекомендована дієта	Не Підтримує	Не Підтримує	Підтримує	Підтримує
Рекомендований фізичних навантажень	Не Підтримує	Не Підтримує	Підтримує	Підтримує

ВИСНОВКИ ДО РОЗДІЛУ 1

У даному розділі було проаналізовано та обґрунтовано вибір платформи для реалізації поставленого завдання, розглянуті основні критерії, на яких базується процес схуднення, проаналізовані характеристики можливої системи. Також для більш повного розуміння, що може бути корисним для користувачів при використанні такої системи схуднення, було проведено огляд та аналіз існуючих на ринку аналогів. Інформація, добута з такого аналізу, дала можливість зрозуміти, якого можливого функціоналу бракувало для повноцінного втілення поставленої мети. Також було здійснено порівняння аналогів та власної запланованої системи схуднення, що дає можливість оцінити користь від розробки такого продукту.

РОЗДІЛ 2

ОГЛЯД ТЕХНОЛОГІЙ, ПІДХОДІВ ТА МОВ ПРОГРАМУВАННЯ ДЛЯ РЕАЛІЗАЦІЇ ПОСТАВЛЕНОГО ЗАВДАННЯ

2.1 Огляд мов програмування

Технології у сучасному інформаційному просторі змінюються з небаченою досі швидкістю і те, що було популярним, ефективним, безпечним для використання ще декілька років тому, зараз може виявитися застарілим або ж взагалі забороненим для користування. Веб-розробка – дуже популярний нині напрямок програмування, а це означає, що на ринку знаходиться безліч інструментів, реалізованих за допомогою різних мов програмування, які допоможуть досягнути поставлених у даній роботі цілей. Дуже важливо обрати правильний засіб для досягнення високої ефективності розробки, підтримки і роботи веб-додатку.

На даний момент найпопулярнішими мовами для веб-розробки є Java, JavaScript, PHP. Кожна з них має свої недоліки та переваги, про які йтиметься далі.

JavaScript хоч і являється популярною мовою програмування для веб-розробки, проте все ж більше підходить для розробки клієнтської частини веб-застосунків. Його так звана слабка типізація, хоч і спрощує процес програмування, проте неявно ускладнює виявлення та діагностику помилок, а також збільшує ризики існування багів та інших помилок в роботі програми. Сильно типізовані мови програмування, хоч і вимагають від програміста більше зусиль при розробці, проте нівелюють це тим, що багато помилок будуть виявлені ще на стадії компіляції. І для фахівця, що дійсно гарно розбирається в технології, сильна типізація аж ніяк не ускладнює життя порівняно зі слабкою типізацією, а навпаки слугує ще одним

інструментом для розробки якісного продукту. Також ще одним недоліком JavaScript є відсутність справжньої підтримки паралельного програмування. Справжньої, тому що в цій мові є тільки імітація паралельного програмування через однопоточну природу JavaScript, в якій застосовується цикл подій. Хоча даний підхід і являється асинхронним і неблокуючим, проте це не еквівалентно паралельному програмуванню. Виклики функцій після завершення виконання іншої функції здійснюються не відразу, це займає якийсь час, потрібно дочекатись завершення функції в фоновому режимі. Асинхронний підхід JavaScript може призводити до так званого «пекла колбеків», що напряму впливає на ефективність програми, читабельність та можливість підтримки коду. Великим недоліком найпопулярнішого фреймворку JavaScript для написання серверного коду Node.js можна рахувати зміну версії Node.js і концепцію модульності. Це особливо помітно при використанні модулів з npm, які досить часто можуть бути не сумісними з новими версіями Node.js і це може викликати неочікувані помилки.

Другою надзвичайно популярною мовою в недавньому минулому є PHP. Ця мова програмування відома як найбільш часто використовувана для веб-розробки. Близько 88% вебсайтів розроблені за допомогою PHP. Ця мова використовувалась для розробки таких гігантів ринку, як Facebook, Wikipedia, Yahoo. Хоча в минулому PHP і була дуже популярною, з часом її доля на ринку помітно зменшувалась в користь новіших і більш оптимізованих аналогів і наразі PHP здебільшого використовується для підтримки legacy коду та legacy проектів, переписування яких на інших мовах є економічно необгрунтованим. PHP, як і JavaScript, є слабо типізованою мовою програмування, що призводить до таких же самих незручностей та помилок. PHP не може похвалитись якісним механізмом обробки помилок та виключних ситуацій, тут практично відсутні інструменти відладки, необхідні для нормального пошуку помилок та

попереджень. Тож перспективи PHP навіть у недалекому майбутньому не дуже втішні і викликають деякі занепокоєння стосовно життєздатності у веб-розробці. Python, мова програмування, яка на поточний момент надзвичайно стрімко розвивається і стає популярною, хоча все ще краще підходить для машинного навчання чи науки про дані, ніж для веб-розробки, вже приваблює більше нових розробників, ніж PHP. Отож, PHP наразі не користується високою популярністю для розробки чогось нового, скоріше тільки для підтримки величезної бази продуктів, розроблених в минулому і які все ще потребують підтримки.

Є ще декілька мов, про які варто згадати. Одна з них – TypeScript. Це мова програмування з відритим сирцевим кодом, розроблена компанією Microsoft, яка компілюється в код JavaScript. TypeScript в значній мірі базується на JavaScript. TypeScript не змінює JavaScript, а лише розширює її новим функціоналом. На TypeScript можна написати практично все, що написане на JavaScript. Мабуть, основною ключовою особливістю TypeScript над JavaScript є підтримувана система типів. Так як JavaScript є мовою зі слабкою динамічною типізацією, що має свої великі недоліки, TypeScript строго визначає, що може містити конкретна змінна. Та оскільки TypeScript базується на JavaScript, ця мова має ряд таких же недоліків, як JavaScript, які були описані вище. Також TypeScript більше популярний для розробки клієнтського коду, аніж серверного.

Також недоліком, який об’єднує всі вище згадані мови програмування, є те, що всі вони є високорівневими мовами програмування, а це означає, що у загальному випадку вони виявляються повільнішими, ніж низькорівневі мови чи мови середнього рівня абстракції.

На відміну від попередників, Java [6] – сильно типізована мова програмування, в основі якої лежить об’єктно-орієнтований підхід до написання коду. Це означає, що код розбитий на модулі і класи, що дає можливість повторного його використання. . Це допомагає зв’язати дані і

функції в одному місці з неможливістю, при необхідності, отримати до них доступ із зовнішнього світу. Об’єкти, створені на основі описаного класу, можуть бути перевикористані у будь-якій частині коду зі збереженням стану об’єкта. Все це підвищує безпеку написаних на Java програм. Іншою величезною перевагою Java є її кросплатформеність, яка досягається за допомогою використання JVM [2]. JVM – це середовище, в якому виконується програма, написана на Java. JVM слідує концепції «Write once, run anywhere». Це технологія, яка відповідає за незалежність від апаратного забезпечення і операційної системи, виконання скомпільованого байт-коду невеликого розміру і оптимізацію використання пам’яті, що використовується застосунком на Java. Іншою особливістю даної мови програмування є її безпека в плані несанкціонованого доступу до пам’яті. Java знижує ризики безпеки, тому що не використовує явних вказівників для доступу до пам’яті. Також Java містить вбудовані можливості для написання багатопоточних програм. Взагалі Java є однією з перших мов програмування, де стало можливим використання багатопоточності для розпаралелювання програм з метою оптимізації використання ресурсів сучасних пристроїв. Це дозволяє використовувати можливості процесора по максимуму. Можливість паралельного програмування була включено ще у першу версію Java. Потоки використовують спільну область пам’яті та тим самим підвищують ефективність застосунків. Такі потоки працюють незалежно один від другого і не впливають на роботу один одного. Стандартна бібліотека містить безліч різних класів для роботи з різноманітними структурами даних, з потоками вводу/виводу, забезпечує можливість мультипоточного програмування, роботи з датою та часом та інші корисні можливості. Стандартна бібліотека містить реалізацію великої кількості структур даних, які стануть в нагоді в різних ситуаціях в залежності від цілей, які переслідуються при написанні програм. Java протягом дуже довгого часу є лідером серед мов програмування на ринку і

ця тенденція має властивість зберігатись ще впродовж багатьох років. Незважаючи на те, що ця мова є далеко не новою, проте вона активно підтримується і покращується розробниками, щоб йти в ногу з часом. Java являється дуже гарним інструментом при серверній розробці. Для цього існує велика, гарно протестована екосистема фреймворків та бібліотек, які дозволяють розробляти серверний код швидко та легко. Незважаючи на деякі недоліки цієї мови, наприклад більше споживання пам'яті через те, що програми на Java запускаються на Java Virtual Machine (JVM), все ж зазначені переваги разом з наявністю надзвичайно потужного фреймворку Spring для розробки бекенд застосунків, являються ключовими при виборі цієї мови для реалізації поставленого в роботі завдання.

2.2 Spring Framework. Загальний огляд

Spring Framework [4] (або коротко Spring) – це універсальний фреймворк з відкритим сирцевим кодом для Java-платформи.

Перша версія даного фреймворку була опублікована ще в далекому 2003 році Родом Джонсоном.

Вперше цей фреймворк був випущений під ліцензією Apache 2.0 license у червні 2003 року. Перша версія 1.0 , яка могла вважатися стабільною була випущена у березні 2004. Інша версія Spring 2.0 була випущена в жовтні 2006, наступна значуща версія, що принесла серйозні нововведення, Spring 2.5, - в листопаді наступного року, подальші версії Spring 3.0 та 3.1 були випущені в грудні 2009 та 2011 року відповідно. Поточна версія - 5.3.x.

Попри те, що Spring Framework не забезпечував якоїсь конкретної моделі програмування, він все одно став широко поширеним і набув популярності в спільноті Java головним чином ставши хорошою альтернативою та заміною моделі Enterprise JavaBeans. Spring надає великі можливості, а також свободу Java-розробникам в проектуванні; крім цього,

Spring Framework надає добре документовані і легкі у використанні інструменти та засоби для вирішення проблем, що часто виникають при створенні та розробці додатків корпоративного масштабу.

Spring Framework забезпечує можливість вирішення великої кількості завдань, з якими часто стикаються Java-розробники і організації, які працюють над створенням інформаційної системи, що базується на платформі Java. Цей фреймворк містить в собі широкий спектр інструментів розробки, тому важко визначити найбільш значущі та найважливіші структурні елементи, з яких складається цей фреймворк. Незважаючи на масштабну інтеграцію з платформою Java Enterprise, Spring Framework не повністю пов'язаний з нею, хоча це і являється важливою причиною популярності цього фреймворку. Цей фреймворк пропонує розробникам послідовну модель та робить її придатною для переважної більшості типів додатків, які вже були створені на основі платформи Java. Також вважається, що Spring Framework реалізує таку модель розробки програмного забезпечення, що базується на кращих стандартах та концепціях індустрії, і робить його доступним у багатьох областях Java.

Spring Framework також може розглядатись як набір менших фреймворків. Практично кожен із цих фреймворків може працювати незалежно один від одного і забезпечувати розв'язання конкретних завдань, з якими розробник стикається під час роботи, проте рекомендується за можливості використовувати їх одночасно в одному спільному середовищі, таким чином буде забезпечуватись більша функціональність . По суті Spring Framework являє собою просто контейнер ін'єкції залежностей, з декількома зручними рівнями (наприклад: доступ до бази даних, проксі, аспектно-орієнтоване програмування, RPC, веб-інфраструктура MVC). Це все дозволяє швидше та зручніше створювати застосунки на Java. Розглянемо основні компоненти Spring фреймворку.

2.2.1 Ін’єкція залежностей

Ін’єкція залежностей, або ж Dependency Injection - це стиль налаштування об'єкта, при якому поля об'єкта задаються зовнішньою сутністю. Іншими словами, об’єкти налаштовуються зовнішніми об'єктами. DI - це альтернатива самостійному налаштуванню об'єктів.

Шаблон проектування ін’єкція залежностей включає 3 типи класів:

- Клієнтський клас – це залежний клас, який залежить від класу сервісу
- Сервісний клас – клас, який надає сервіс клієнтському класу
- Клас-інжектор – клас-інжектор сервісного класу в клієнтський клас

Ілюстрація взаємозв'язку (рис. 2.1) між цими класами:

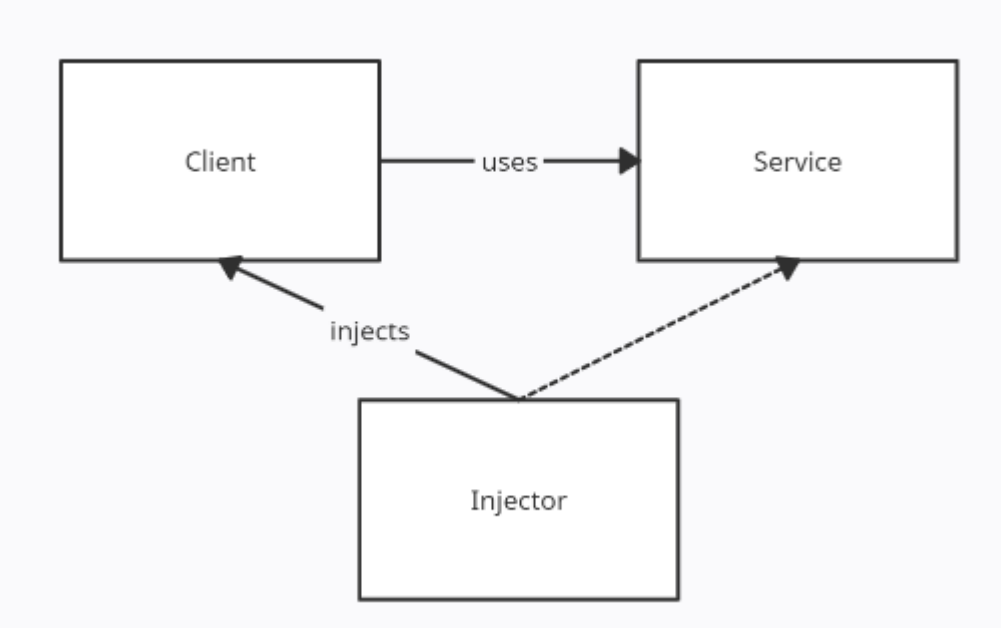


Рис 2.1. Взаємодія класів при ін’єкції залежностей

Як видно з рисунку 2.1, клас інжектора створює об'єкт класу сервісу і вводить цей об'єкт в об'єкт клієнта. Таким чином, шаблон DI відокремлює відповідальність за створення об’єкта класу сервісу з класу клієнта. Існує 3 основні типи ін’єкції залежностей. Зі схеми, показаної вище, клас інжектора вводить клас сервісу (залежність) клієнту (залежний клас). Клас інжектора

вводить залежності в цілому трьома способами: за допомогою конструктора, через поле класу або за допомогою методу.

Ін’єкція за допомогою конструктора: при ін’єкції за допомогою конструктора клас інжектора поставляє клас сервісу (залежність) через конструктор класу клієнта.

Ін’єкція за допомогою полів класу: при ін’єкції залежностей за допомогою полів класу інжектор забезпечує залежність через публічне поле класу клієнта.

Ін’єкція за допомогою методу: у цьому типі ін’єкції клас клієнта реалізує інтерфейс, який оголошує метод чи методи для забезпечення залежності, а інжектор використовує цей інтерфейс для забезпечення залежності для класу клієнта.

2.2.2 Інверсія управління

Інверсія управління (англ. Inversion of Control, IoC) - важливий принцип об'єктно-орієнтованого програмування, що використовується для зменшення зчеплення (зв'язаності) в комп'ютерних програмах. Також це архітектурне рішення інтеграції, що спрощує розширення можливостей системи, при якому потік управління програми контролюється фреймворком.

У звичайній програмі програміст сам вирішує, в якій послідовності робити виклики процедур. Але, якщо використовується фреймворк, програміст може розмістити свій код в певних точках виконання (використовуючи callback або інші механізми), потім запустити «головну функцію» фреймворка, яка забезпечить всі виконання і дозволить викликати код програми тоді, коли це буде необхідно. Як наслідок, відбувається втрата контролю над виконанням коду - це і називається інверсією управління (фреймворк управляє кодом програміста, а не програміст управляє фреймворком). Однією з реалізацій інверсії управління в застосуванні до

управління залежностями є ін’єкція залежностей (Dependency injection). Ін’єкція залежностей використовується в багатьох фреймворках, які називаються IoC-контейнерами. Якщо порівняти з більш низькорівневими технологіями, IoC-контейнер - це компоновщик, який збирає не об'єктні файли, а об'єкти ООП (екземпляри класу) під час виконання програми. Очевидно, для реалізації подібної ідеї було необхідно створити не тільки сам компоновщик, але і фабрику, що виробляє об'єкти. Аналогом такого компоновщика (природно, більш функціональним) є компілятор, однією з функцій якого є створення об'єктних файлів. В ідеї компонування програми під час виконання немає нічого нового. Надання програмісту інструментів впровадження залежностей дало значно більшу гнучкість у розробці та зручність у тестуванні коду Inversion of Control-контейнер: делегована конфігурація компонентів додатків і управління життєвим циклом Java-об'єктів.

2.2.3 Аспектно-орієнтоване програмування

Фреймворк АОП надає можливість групування функцій та поведінки, яка зазвичай не може повністю бути реорганізована в окремі модулі з подальшою можливістю використання цих функцій. Можливості ООП в Java не надають можливості для реалізації такої функціональності, тож постає необхідність у використанні даного фреймворку.

Отже, АОП - аспектно-орієнтоване програмування - це парадигма, спрямована на підвищення модульності різних частин програми за рахунок поділу наскрізних завдань. Для цього до вже існуючого коду додається додаткова поведінка, без змін у початковому коді.

Іншими словами, на методи і клас зверху накладається додаткова функціональність, при цьому в самі методи і не вносяться жодні поправки. Це важливо, тому що звичайний об'єктно-орієнтований підхід не завжди може ефективно вирішити ті чи інші завдання. В такий момент на допомогу

приходить АОП і дає додаткові інструменти для побудови програми. А додаткові інструменти - це збільшення гнучкості при розробці, завдяки якій з'являється більше варіантів вирішення того чи іншого завдання.

Аспектно-орієнтоване програмування призначене для вирішення наскрізних завдань, які можуть являти собою будь-який код, постійно повторюваний різними методами, який не можна повністю структурувати в окремий модуль.

Відповідно, за допомогою АОП ми можемо залишити якусь функціональність за межами основного коду і визначити де в програмі «навішується» ця функціональність (рис. 2.2). Як приклад можна привести застосування політики безпеки в будь-якому додатку.

Як правило, безпека проходить крізь багато елементів програми. Тим більше, політика безпеки програми повинна застосовуватися однаково до всіх існуючих і нових частин програми. При цьому використовується політика безпеки може і сама розвиватися. Ось тут нам відмінно може стане в нагоді використання АОП.

Також в якості ще одного прикладу можна навести процес під назвою logging. У використання АОП підходу до логування є кілька переваг у порівнянні з ручною вставкою логування:

- Код для логування легко впроваджувати і видаляти: всього-то потрібно додати або видалити пару конфігурацій деякого аспекту.
- Весь вихідний код для логування зберігається в одному місці і не потрібно знаходити вручну все місця використання.
- Код, призначений для логування, можна додати в будь-яке місце, будь то вже написані методи і класи або ж новий функціонал. Це зменшує кількість помилок розробника. Також при видаленні аспекту з конфігурації конструкції можна бути абсолютно впевненим, що весь код трасування вилучений і нічого не пропущено.

- Аспекти - це винесений окремо код, який можна багаторазово перевикористати і покращувати.

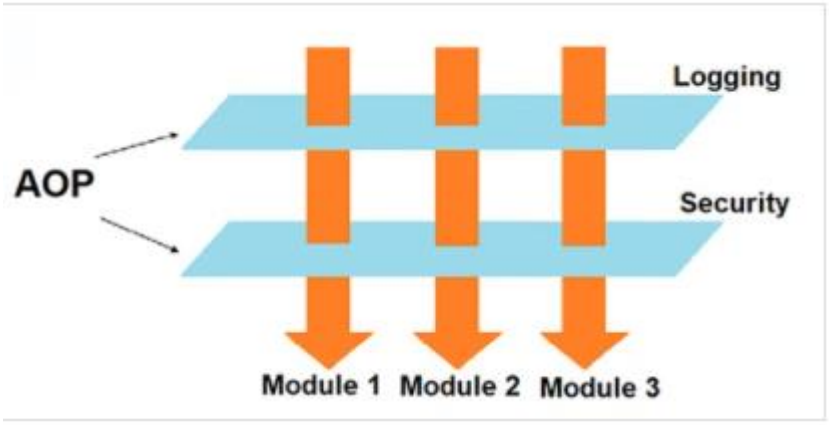


Рис. 2.2 Приклад використання аспектно-орієнтованого підходу

2.2.4 Spring MVC

Фреймворк Spring MVC забезпечує архітектуру паттерна Model - View - Controller (Модель - Відображення - Контролер) за допомогою слабо пов'язаних готових компонентів. Патерн MVC розділяє аспекти додатку (логіку введення, бізнес-логіку і логіку UI), забезпечуючи при цьому вільний зв'язок між ними.

- Model - інкапсулює (об'єднує) дані програми
- View - відповідає за відображення даних Model, як правило, генеруючи HTML, який ми бачимо в своєму браузері.
- Controller - обробляє запит користувача, створює відповідну Model і передає її для відображення в View.

Вся логіка роботи Spring MVC побудована навколо DispatcherServlet, який приймає і обробляє всі HTTP-запити (з UI) і відповіді на них. Робочий процес обробки запиту DispatcherServlet проілюстрований на рисунку 2.3

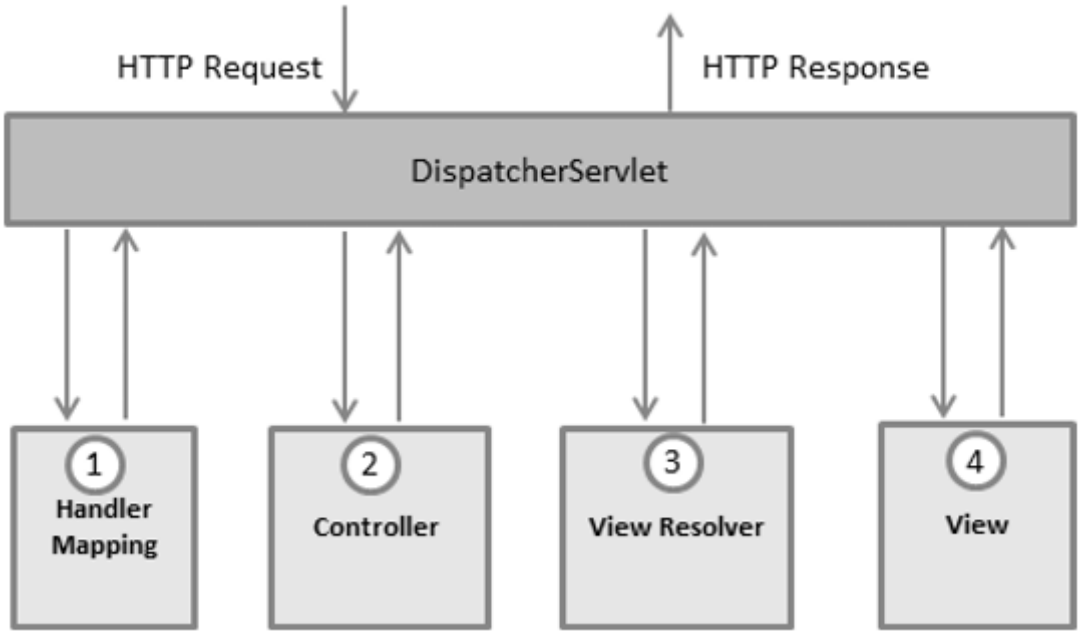


Рисунок 2.3. Процес обробки запиту DispatcherServlet

Насправді DispatcherServlet - повністю інтегрований сервлет в Spring IoC контейнер, а отже він отримує доступ до всіх можливостей Spring. DispatcherServlet - це звичайний сервлет (успадковується від базового класу HttpServlet), і його також необхідно описувати в web.xml веб-додатку. Розробнику необхідно вказати меппінг запитів, які будуть оброблятися в DispatcherServlet, шляхом вказання URL в web.xml.

Спочатку вхідний запит потрапляє в Front controller. Далі він перенаправляє його в Controller. Саме за цю частину і відповідає розробник. Після обробки запиту надсилається відповідь до Front controller, і після цього результат використовується для відображення View. Все це працює в контейнері сервлетів (Tomcat)

Спочатку DispatcherServlet (диспетчер сервлетів) отримує запит, далі він переглядає власні налаштування, щоб зрозуміти який контролер використовувати (на малюнку Handler Mapping).

Після отримання імені контролера запит передається в нього (на малюнку Controller). У контролері відбувається обробка запиту та назад надсилається ModelAndView (model - самі дані; view - як ці дані

відображати). DispatcherServlet на основі отриманого ModelAndView проводить представлення, яке йому слід використовувати (View Resolver) і після цього отримує у відповідь конкретний View. У представлення передаються дані (model) і назад, якщо необхідно, надсилається відповідь від представлення.

2.3 Бази даних

База даних – це організована колекція даних , зібрана в одному місці та яка характеризується правилами, що поширюються на ці дані і зв'язки між ними. Зазвичай зберігається комп'ютерній системі та може бути отримана з комп'ютерної системи. У базах даних дані представляються у вигляді таблиць, також вони містять схеми, процедури та інші об'єкти. Система керування базами даних (СКБД) – це система, що надає можливість створення, керування, використання і маніпулювання базами даних .

Існують різні архітектури СКБД: архітектура «файл-сервер» передбачає те, що одна з машин мережі буде використовуватися в якості серверу. Архітетура «клієнт-сервер» - в мережі існує сервер баз даних, ним виступає комп'ютер, на якому встановлено програмне забезпечення для баз даних і самі бази даних. Для використання баз даних в такій архітектурі клієнти виконують запити на сервер. Після отримання запиту сервер опрацьовує його та надсилає клієнту відповіді. Види баз даних: мережева – дані структоровані як в ієрархічної, але у об'єкта-нащадка може буде багато батьків. В ієрархічній базі даних дані представляються у вигляді дерева об'єктів і зв'язків між ними. В об'єкто-орієнтованих базах дані зберігаються у вигляді моделей об'єктів. У реляційних базах дані зберігаються у виді пов'язаних між собою таблиць.

Також існує ще один поділ баз даних на реляційні та нереляційні. SQL – мова для реляційних баз даних. Відмінність SQL та NoSQL полягає в тому, що в реляційних базах дані однозначно повинні бути представлені в певній

структурі, а в NoSQL таких обмежень немає. Мова SQL використовується для роботи з реляційними базами даних, оскільки ці бази даних використовують SQL-стандарти.. NoSQL може використовувати свій спосіб для кожної окремої бази даних. Горизонтальне масштабування краще у NoSQL, хоча обидва типи баз даних добре маштабуються вертикально. Реляційних бази даних володіють кращою надійністю. SQL бази даних дуже популярні, тому вирішення прикладних задач зазвичай знаходиться доволі легко, що не можна сказати про нові бази даних як наприклад MongoDB.

PostgreSQL [3] - це дуже популярна і потужна об'єктно-реляційна база даних з відкритим кодом. PostgreSQL випустилась в 1989 році під ліцензією BSD і досі активно працює вже понад 30 років. Це забезпечило їй міцну репутацію, високу надійність та продуктивність. Ця база даних підтримує SQL і JSON для реляційних та нереляційних запитів. PostgreSQL підтримує функції оптимізації продуктивності та розширені типи даних. Функції – це блоки коду, написані на SQL, хочу і можливе використання певних розширень та інших мов. Функції оптимізації і розширені типи даних доступні лише у дорогих комерційних базах даних, що робить її ще більш цінною при використанні. PostgreSQL підтримують досвідчені та професіональні розробники, які зробили суттєвий внесок для створення СКБД, що являється не просто зручною, а й дуже надійною.

Ось деякі найголовніші переваги PostgreSQL [1]:

- Сумісна з різними платформами
- Об’єктно-реляційна база даних
- Пропонує складний механізм запитів
- Підтримка баз даних необмеженого розміру
- Зріла функціональність програмування на стороні сервера
- Підтримка управління кількома версіями одночасно
- Легка розширюваність

- Повна підтримка архітектури мережі клієнт-сервер
- Підтримка JSON надає можливість зв'язку з іншими сховищами даних, наприклад з NoSQL

Нижче наведені суттєві плюси PostgreSQL:

- Сирцевий код знаходиться у відкритому доступі. Така особливість дозволяє використовувати, впроваджувати та модифікувати PostgreSQL відповідно до різних бізнес-потреб.
- PostgreSQL підтримує географічні об'єкти, тому можна використовувати її для служб, що базуються на розташуванні, та геоінформаційних систем
- Низький рівень адміністрування та обслуговування для вбудованого і корпоративного використання.
- Postgres простий у використанні

2.4 Система автоматизації побудови проекту

Усі програми повинні проходити певний ряд процесів, незалежно від того, наскільки вони великі за розміром, наприклад, генерацію та компіляцію вихідного коду. Розробники можуть налаштовувати ці процеси вручну, але це займає багато часу.

Для вирішення цієї проблеми існують системи автоматизації побудови проекту і яскравий представник таких систем - Apache Maven. Він автоматизує весь процес і полегшує роботу розробників

Цей інструмент автоматизації було створено ще у 2002 році і і головною ціллю було бажання замінити Apache Ant – дуже незручний на той час засіб автоматизації побудови проекту.

Maven - це популярний інструмент збірки з відкритим кодом, розроблений Apache Group для створення, публікації та розгортання декількох проектів одночасно для кращого управління проектами. Цей

інструмент дозволяє розробникам створювати та документувати структуру життєвого циклу.

Основним файлом, з якого Maven бере інформації щодо конфігурації програми – це pom.xml. Тут містяться всі залежності, плагіни, необхідні для складання проекту і роботи програми.

Java – мова програмування, на якій був написаний Maven. Він використовується для побудови проектів, що написані на таких мовах програмування, як Ruby, C #, Scala.

Всі залежності – це звичайні бібліотеки, написані на Java. Вони зберігаються в одному місці, що називається Maven Repository. При імпорті якоїсь залежності Maven шукає файли з цього сховища.

Maven зосереджується на спрощенні та стандартизації процесу побудови програми. Maven завантажений багатьма цінними та корисними функціями, що дуже легко пояснює його популярність. Нижче наведені деякі значущі особливості Maven:

- Величезне сховище бібліотек, яке з часом тільки зростає
- Можливість легко створювати проекти, маючи доступ лише до одного сховища залежностей
- Зворотна сумісність із попередніми версіями
- Управління залежностями, що включає автоматичне оновлення
- Сильна звітність про помилки та цілісність
- Забезпечує послідовне використання у всіх проектах
- Можна легко писати плагіни, використовуючи Java.
- Автоматичне встановлення батьківських версій

2.5 Система контролю версій Git

Однією з найбільших переваг Git є його можливості розгалуження. На відміну від централізованих систем контролю версій, гілки Git легко об’єднуються.

Гілки забезпечують ізольоване середовище для кожної зміни кодової бази. Коли розробник хоче почати щось робити - незалежно від того, наскільки великим чи малим буде його проект – він може створити нову гілку, яка є повністю незалежною від головної гілки проекту. Це гарантує, що головна гілка завжди містить основний бізнес-код.

Використання гілок не тільки надійніше, ніж безпосереднє редагування коду, але й забезпечує організаційні переваги. Наприклад, ви можете реалізувати політику, згідно з якою кожен квиток на Jira адресується у своїй гілці функцій.

У SVN кожен розробник отримує робочу копію, яка вказує на єдине центральне сховище. Однак Git - це розподілена система контролю версій. Замість робочої копії кожен розробник отримує власне локальне сховище, в комплекті з повною історією комітів. Повна локальна історія робить Git швидким і це означає, що не потрібно мережеве підключення для створення комітів, перевірки попередніх версій файлу або перегляд відмінностей між комітами.

2.6 Шаблонізатор Thymeleaf

Thymeleaf [7] – це повномасштабний та сучасний механізм шаблонів Java на стороні сервера. Thymeleaf можна використовувати для веб, автономних і локальних середовищ. Механізм шаблонів Thymeleaf пропонує спрощені шаблони для процесів розробки, по факту це набір шаблонів HTML, що зберігається на сервері і покликаний спростити співпрацю між командами розробників.

Thymeleaf розроблявся під Java-платформу, тому його можна використовувати з Spring, Servlets та іншими технологіями платформи. Thymeleaf дуже добре інтегрований з Spring Framework, що являється великою перевагою при виборі даного шаблонізатора, адже серверна частина додатку буде розроблятись за допомогою Spring Framework.

Thymeleaf - це повністю відкритий проект із детальною документацією та залученою спільнотою розробників. Це забезпечує частий вихід оновлень з усуненням виявлених помилок. Thymeleaf – це потужна мова вираження. Він використовує Standard Dialect як мову виразів, що набагато краще, ніж інші механізми шаблонів, такі як JSP.

ВИСНОВКИ ДО РОЗДІЛУ 2

У даному розділі було розглянуто найпопулярніші мови програмування, які використовуються для веб-розробки та написання клієнт-серверних застосунків. Обгрунтовано вибір однієї з мов, а також зроблено огляд технологій, необхідних для створення веб-застосунків. Було розглянуто деякі архітектурні підходи, що використовуються при розробці та реалізації веб-застосунків.

Детальний аналіз мов програмування показав, що більшість популярних мов мають слабку типізацію, відсутність можливостей для паралельного програмування, а також є високорівневими мовами програмування, що в загальному випадку негативно впливає на швидкість роботи програм, написаних на таких мовах. Тому для реалізації поставлених цілей була обрана мова програмування Java, яка позбавлена всіх цих недоліків, а також є надзвичайно популярною мовою програмування для створення веб-додатків та має свої чудові технології для створення таких додатків.

Наступним кроком став детальний огляд екосистеми Spring Framework – надзвичайно потужного інструменту для веб-розробки. Були перераховані та описані основні компоненти цього фреймворку а також концепції, на яких базується його робота.

Далі був зроблений огляд інших інструментів, необхідних або ж корисних для веб-розробки, та важливих концепцій, як от система керування базами даних, протоколи роботи мереж, система автоматизації побудови проекту, система контролю версій, шаблонізатори.

Всі ці технології допоможуть глибше зрозуміти суть веб-розробки та стануть зручним та необхідним інструментом для реалізації поставленої мети.

РОЗДІЛ 3

ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Проектування компонентів системи

В системі передбачено існування двох типів користувачів: звичайний користувач та адміністратор. Адміністратор існує тільки один. При реєстрації нового користувача йому автоматично надається роль звичайного користувача. При користуванні веб-додатком користувач має в своєму арсеналі такий функціонал:

- реєстрація користувача у веб-додатку
- заповнення та корегування особистої інформації
- перегляд страв та вправ з бази даних
- додавання нових страв та вправ до бази даних
- вибір страви з вказанням порції та періоду прийому їжі
- перегляд та видалення спожитих страв
- перегляд статистики спожитих калорій, білків, жирів, вуглеводів
- історія спожитих страв зі статистикою на кожен день
- аналогічний функціонал з фізичними навантаженнями
- перегляд порад по здоровому харчуванню та схудненню
- перегляд плану схуднення
- перегляд денного бюджету та обігу калорій за поточний день
- рекомендована дієта при схудненні
- персональний план тренувань
- можливість перемикання мови інтерфейсу на англійську, російську та українську

Адміністратор також має змогу переглядати всіх зареєстрованих користувачів, а також додавати нові страви чи вправи до бази даних чи видаляти з бази даних.

Надзвичайно важливим етапом є вказання особистих показників, адже на основі цих даних формується план схуднення, бюджет калорій та інші важливі показники. Такі дані вносяться в систему при реєстрації користувача, тому на рисунку 3.1 демонструється процес реєстрації користувача:

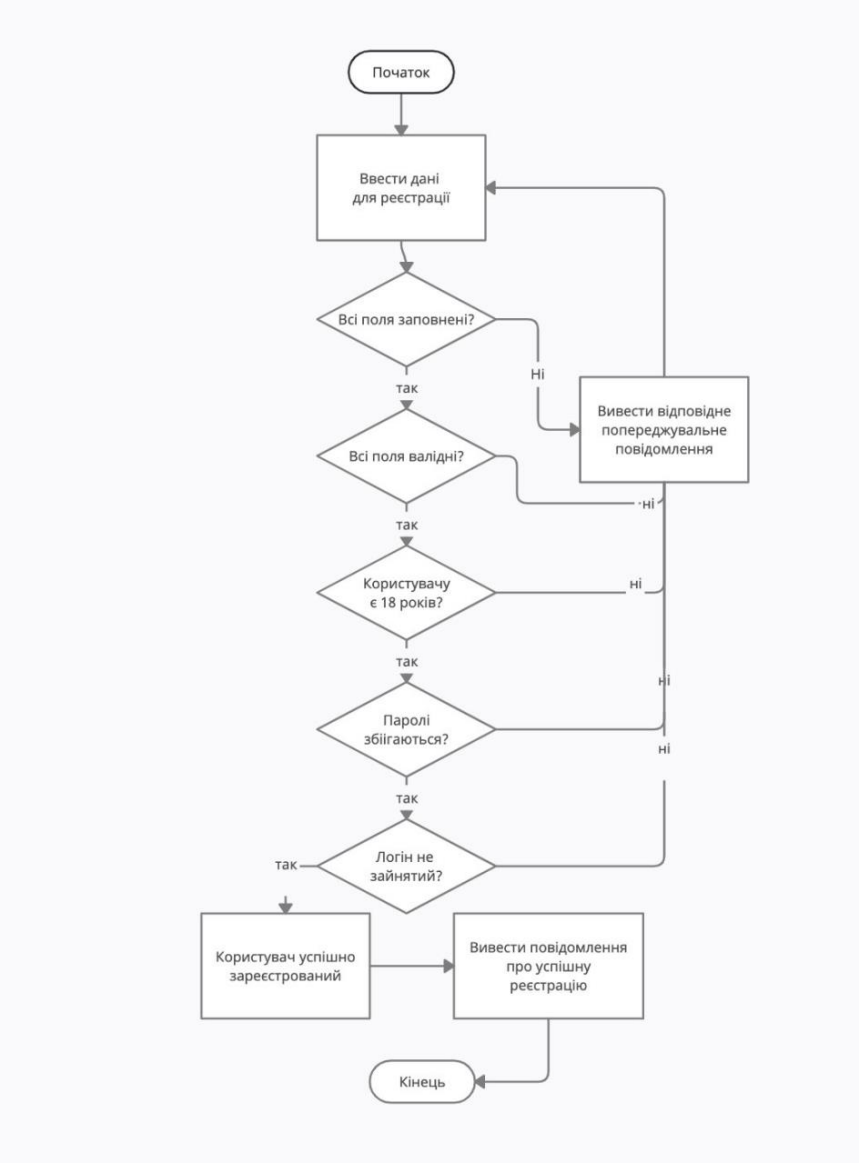


Рис. 3.1 Демонстрація роботи сторінки реєстрації

На програмному рівні це виглядає так, як показано на наступному рисунку (рис 3.2). Якщо виникла якась виключна ситуація (див. рис. 3.1), в коді це реалізується за допомогою механізму роботи виключних ситуацій. В разі, якщо ніяких виключних ситуацій не було виявлено, відбувається реєстрація нового користувача та пов’язаної з ним інформації.

```
public void saveNewUser (UserDTO userDto, UserInfoDto userInfoDto) throws RegFailedException {
    String emptyField;
    if((emptyField = isCompleteUserInfoOrElseGetEmptyField(userInfoDto)) != null ||
       (emptyField = isCompleteUserOrElseGetEmptyField(userDto)) != null)
        throw new EmptyFieldException(emptyField);

    String msg;
    if((msg = getMessageIfWrongRange(userInfoDto)) != null)
        throw new ParameterRangeException(msg);

    Optional<MyUser> optionalMyUser = userRepository.findByLogin(userDto.getLogin());

    if(optionalMyUser.isPresent())
        throw new UserAlreadyExistsException(userDto.getLogin());
    if(!userDto.getPassword().equals(userDto.getConfirmedPassword()))
        throw new DifferentPasswordsException();

    int age = getAge(userInfoDto.getDob());
    if(age < 18)
        throw new NotAdultException(age);

    MyUser user = MyUser.builder()
                .login(userDto.getLogin())
                .password(passwordEncoder.encode(userDto.getPassword()))
                .role(UserRole.USER)
                .isEnabled(true)
                .isAccountNonExpired(true)
                .isCredentialsNonExpired(true)
                .isAccountNonLocked(true)
            .build();
    userRepository.save(user);

    UserInfo userInfo = UserInfo.builder()
                .nickname(userInfoDto.getNick())
                .weight(userInfoDto.getWeight())
                .goalWeight(userInfoDto.getGoalWeight())
                .height(userInfoDto.getHeight())
                .age(age)
                .male(Male.valueOf(userInfoDto.getMale()))
                .lifestyle(Lifestyle.valueOf(userInfoDto.getLifestyle()))
                .exercisesPerWeek(userInfoDto.getExercisesPerWeek())
                .weightLossPerWeek(userInfoDto.getWeightLossPerWeek())
                .user(user)
            .build();
    userInfoRepository.save(userInfo);
}
```

Рис 3.2. Приклад обробки запиту на реєстрацію

3.2 Опис алгоритму обрахунку калорій

Після реєстрації, веб-застосунок зберігає ці дані в базі даних і потім на їх основі формує денний бюджет калорій, який розраховується за формулою Міфліна-Сан Жеора [10], яка була представлена в 90-х роках минулого століття. Проте вона не тільки не втратила актуальності в наші дні, а й являється найпопулярнішою формулою для розрахунку необхідного рівня калорій по причині того, що вона найкраще підходить під сучасний стиль життя і харчування. Хоча в цій формулі є недоліки, наприклад вона не враховує відсотковий показник жиру в організмі і тому вважається, що ця формула завищує потребу в калоріях, але з огляду на те, що цей веб-додаток орієнтований на звичайних людей, які наврядчи коли-небудь задавались подібним питанням, а отже далеко не всі користувачі знають відсоток підшкірного жиру, і це унеможливить користування даним додатком.

Формула Сан-Жеора відрізняється для чоловіків (формула 3.1) та для жінок (формула 3.2):

$$BMR = [10*\text{вага}] + [6.25*\text{зріст}] - [5 * \text{вік}] + 5 \quad (3.1)$$

$$BMR = [10*\text{вага}] + [6.25*\text{зріст}] - [5 * \text{вік}] - 161 \quad (3.2),$$

де BMR – базальний рівень метаболізму, тобто мінімальна кількість калорій (енергії), необхідна людині для підтримання внутрішніх процесів організму;

вага – це вага людини, вказана в кілограмах;

зріст – це зріст людини, вказаний в сантиметрах;

вік – кількість повних років.

Зважаючи на те, що дана формула розраховує мінімальну кількість калорій для підтримання життя, а людина протягом дня ще додатково витрачає енергію, отриманий результат ще слід помножити на коефіцієнт, який визначається стилем життя (денні фізична активність) людини.

Основні категорії стилю життя:

- сидячий стиль життя – мала кількість або ж взагалі відсутність фізичних занять, людина більшу частина дня проводить сидячи;
- малорухливий – небагато денної активності + легкі фізичні навантаження 1-3 рази на тиждень;
- активний – активний стиль життя і помірні фізичні навантаження 3-5 раз на тиждень;
- дуже активний – спортивний стиль життя, тяжка фізична праця, щоденні тренування і так далі.

Кожен зі стилів життя має такий коефіцієнт, на який множиться мінімальна кількість калорій для підтримання функціонування організму:

- сидячий стиль – 1.2;
- малорухливий стиль – 1.375;
- активний – 1.55;
- дуже активний – 1.725.

BMR множиться на коефіцієнт обраного стилю життя і після цього формується фінальне значення бюджету калорій на день. Для схуднення денний рівень калорій повинен зменшитись. Це залежить від обраного користувачем темпу схуднення, який визначається кількістю скинутих кілограм в тиждень. Для схуднення на півкілограма в тиждень бюджет калорій слід зменшити на приблизно на 20%, що в середньому для середньостатистичної людини ставить близько 450-500 калорій.

3.3 Проектування бази даних

Для підтримання функціоналу веб-додатку база даних має бути спроектована так, щоб зберігати всю необхідну інформацію про користувача, страви, вправи, обрані страви та вправи та інші корисні дані. Для цього слід правильно розробити зв’язок таблиць, щоб мати змогу з легкістю будувати запити на отримання даних з таблиць бази даних.

Для досягнення зазначених цілей було створено 6 різних таблиць (рис. 3.3) :

- users – таблиця, що містить інформацію про обліковий запис користувача
- user-info – таблиця для зберігання особистих показників користувача а також інформації, пов'язаної зі схудненням. Має відношення "one to one" з таблицею users.
- dishes – таблиця, яка містить інформацію про всі страви в системі, включаючи особисті страви
- exercises – таблиця з інформацією про всі наявні в системі фізичні активності
- user-dish – таблиця, що зберігає інформацію про обрані користувачем страви. Містить поле "date" для фіксації дати обрання страви і можливості доступу до історії спожитих страв. Має відношення "many to one" з таблицею users і dishes
- user-dish – таблиця, що зберігає інформацію про обрані користувачем фізичні навантаження. Містить поле "date" для фіксації дати обрання вправи і можливості доступу до історії виконаних вправ. Має відношення "many to one" з таблицею users і exercises

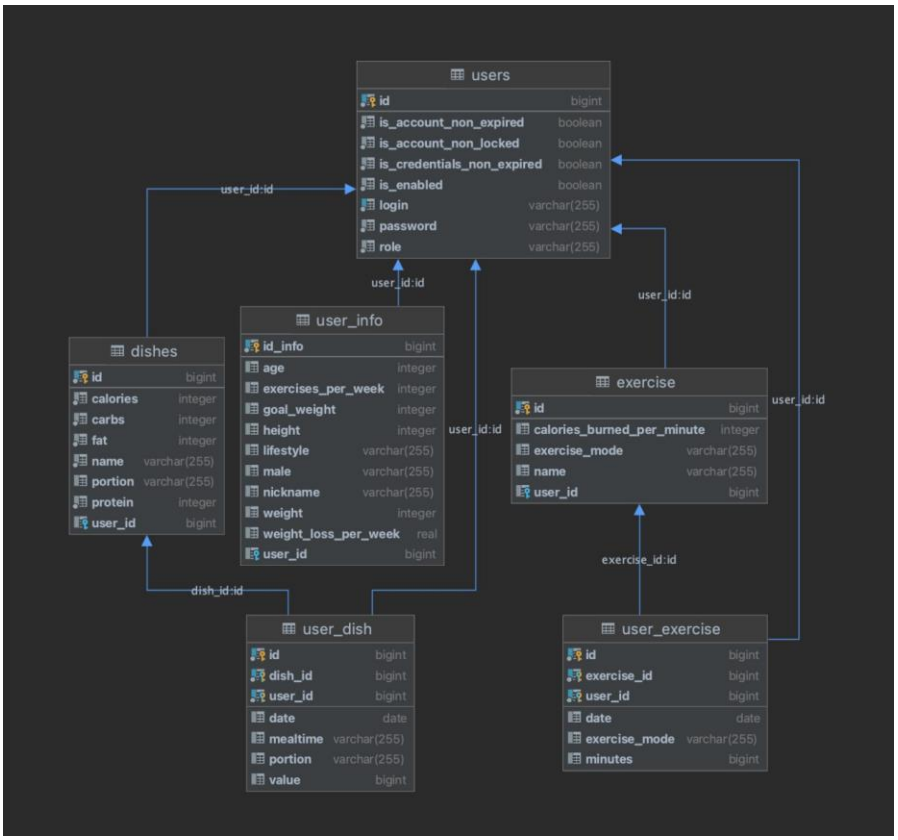


Рис. 3.3 Взаємодія сутностей в базі даних

Для зручності використання бази даних в коді використовується ORM, вбудована в фреймворк Spring під назвою Spring Data Jpa [18]. Дана технологія значно полегшує доступ до сутностей бази даних, а також дає можливість позбутись написання поширених та тривіальних запитів до бази даних. Це досягається за допомогою позначення класів сутностей за допомогою анотації @Entity. Для визначення первинного ключа застосовується анотація @Id. Також для позначення відношень між сутностями доступні деякі анотацій, наприклад @OneToOne, @ManyToOne, @ManyToMany. Приклад написання сутності (рис. 3.4) з даними анотаціями продемонстрований на наступному рисунку:

```java
@Entity
public class UserInfo {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(name = "id_info", nullable = false, unique = true)
    private Long id;
    private String nickname;

    @OneToOne
    private MyUser user;
    private int weight;
    private int goalWeight;
    private int height;
    private int age;

    @Enumerated(EnumType.STRING)
    private Male male;

    @Enumerated(EnumType.STRING)
    private Lifestyle lifestyle;

    private int exercisesPerWeek;
    private float weightLossPerWeek;
}
```

Рис. 3.4 JPA сутність таблиці userInfo

ВИСНОВКИ ДО РОЗДІЛУ 3

Були спроектовані компоненти веб-додатку з врахуванням логіки роботи програми. На основі проведених у першому розділі досліджень та порівняльної характеристики аналогів, було розроблено веб-застосунок з визначеним у цьому розділі функціоналом, що об'єднує основні можливості популярних веб-систем для схуднення.

В даному розділі було наведено приклад роботи частини програми, а також показано як це реалізовано у програмному коді. Пояснена необхідність комплексної обробки запиту на реєстрацію, адже інформація, що постачається користувачем під час цього етапу – є ключовою у розрахунку формул та для якісного постачання рекомендацій та інших даних користувачу.

Також у даному розділі були розглянуті ключові формули, на основі яких розраховуються важливі для користувача показники, наприклад BMR – кількість енергії в калоріях, необхідна користувачу для підтримання функціонування організму, денний бюджет калорій, та бюджет калорій, якого слід дотримуватись при схудненні.

Підсумком цього розділу став розгляд архітектури розробленої бази даних, а також сутностей, що її наповнюють.

РОЗДІЛ 4

ІНСТРУКЦІЯ КОРИСТУВАЧА

При відкритті сайту, користувача зустрічає сторінка авторизації (рис. 4.1) в систему, яка пропонує ввести особисті дані для входу при умові, що акаунт вже існує, або ж зареєструватись в іншому випадку.

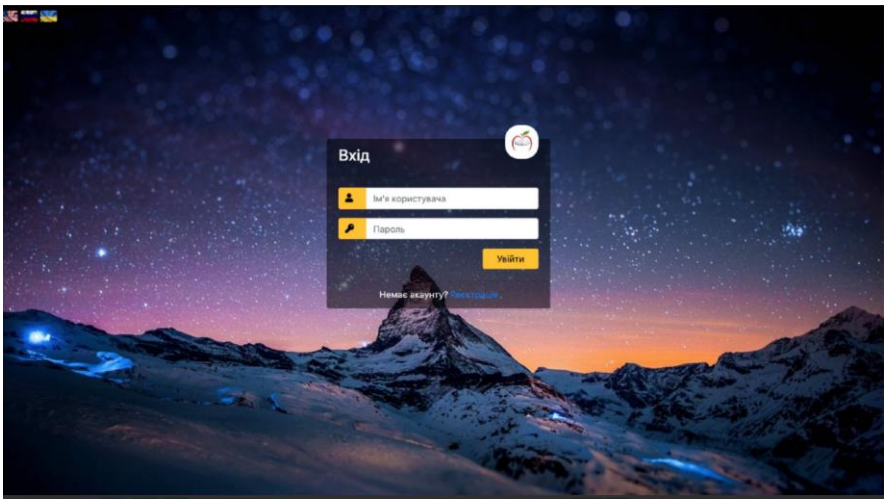


Рис. 4.1. Сторінка авторизації українською

Також зверху зліва є кнопки вибору мови інтерфейсу системи: англійська, російська та українська. При зміні мови на англійську сторінка авторизації виглядає так (рис 4.2) :

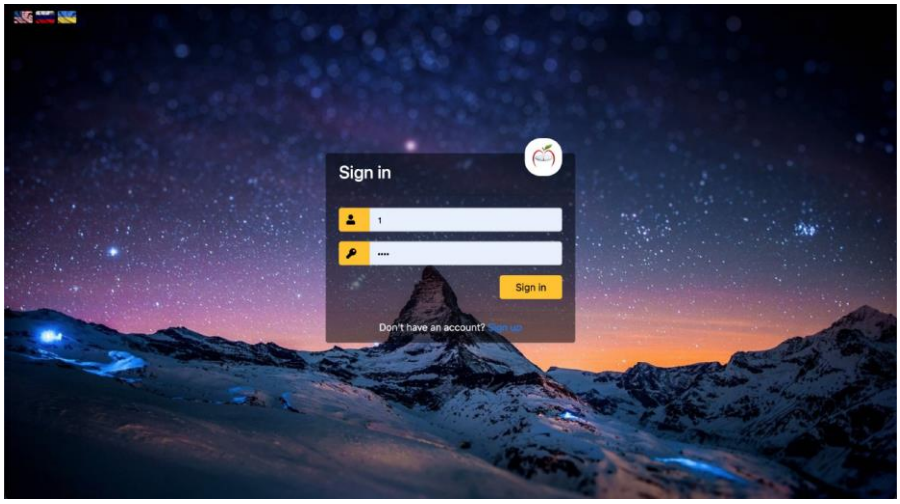


Рис. 4.2. Сторінка авторизації англійською

На сторінці реєстрації нового облікового запису користувачу необхідно заповнити деякі відомості про себе (рис. 4.3-4.5) , а також ввести дані для входу в систему (рис. 4.6)

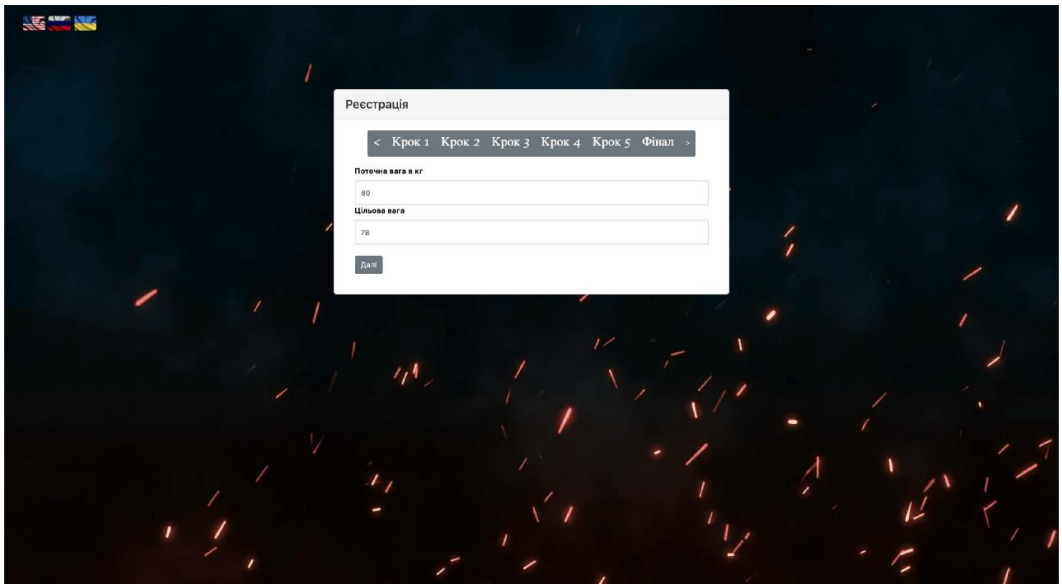


Рис. 4.3. Реєстрація нового облікового запису

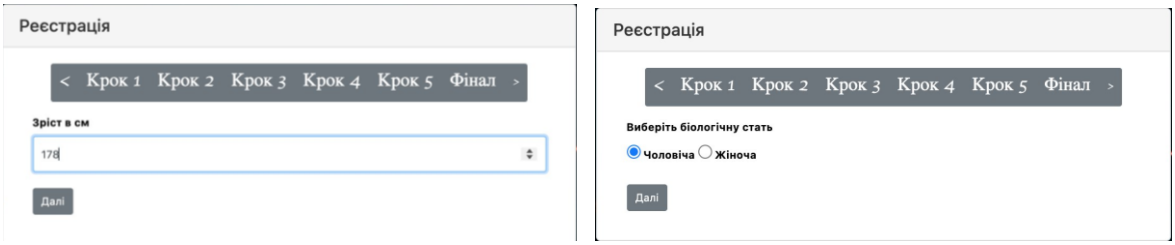


Рис. 4.4. Заповнення полів

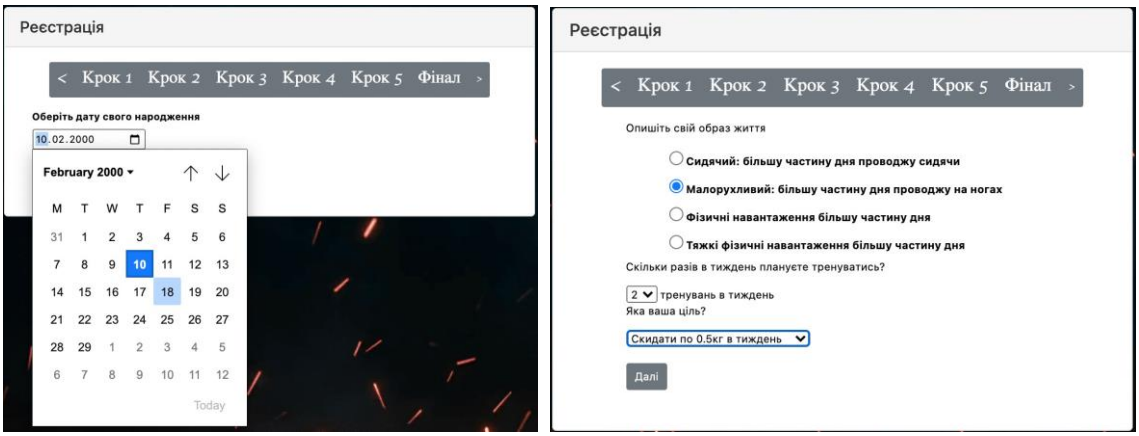


Рис. 4.5. Заповнення інших показників

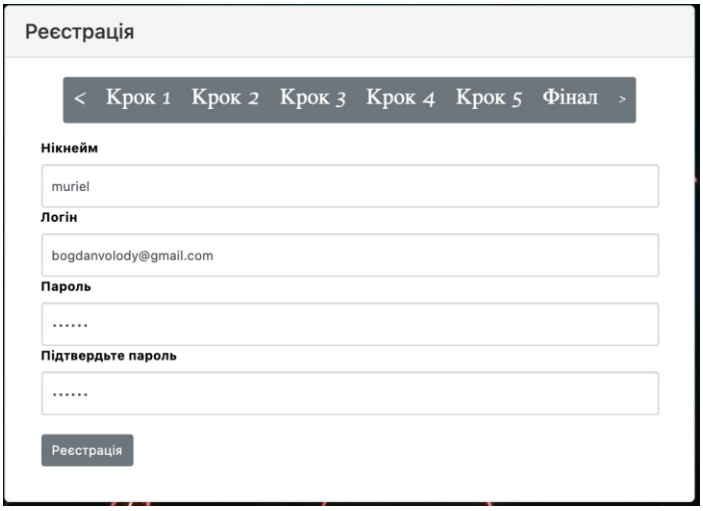


Рис. 4.6. Вказання даних для входу в акаунт

Якщо якесь з полів буде незаповненим користувачем, реєстрація не відбудеться і система виведе повідомлення про те, в яке поле слід внести дані (рис. 4.7)

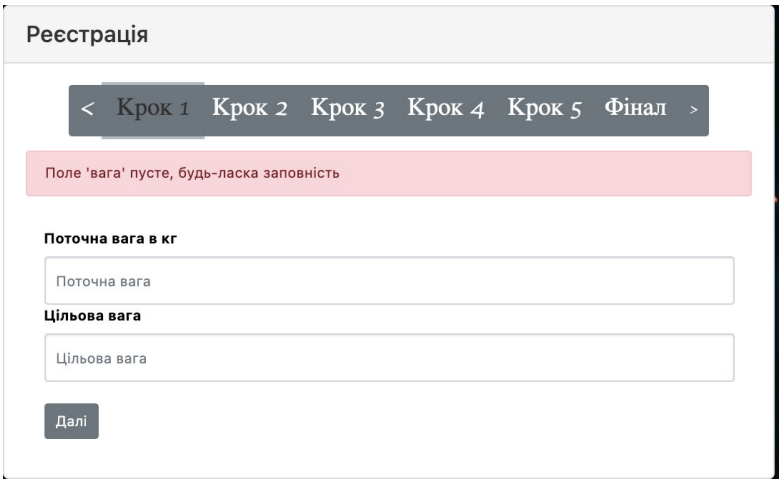


Рис. 4.7. Повідомлення при не заповненому полі

Якщо всі дані внесені, система проводить валідацію даних і виводить конкретизуюче попереджувальне повідомлення, що не так (рис.4.8-4.9)

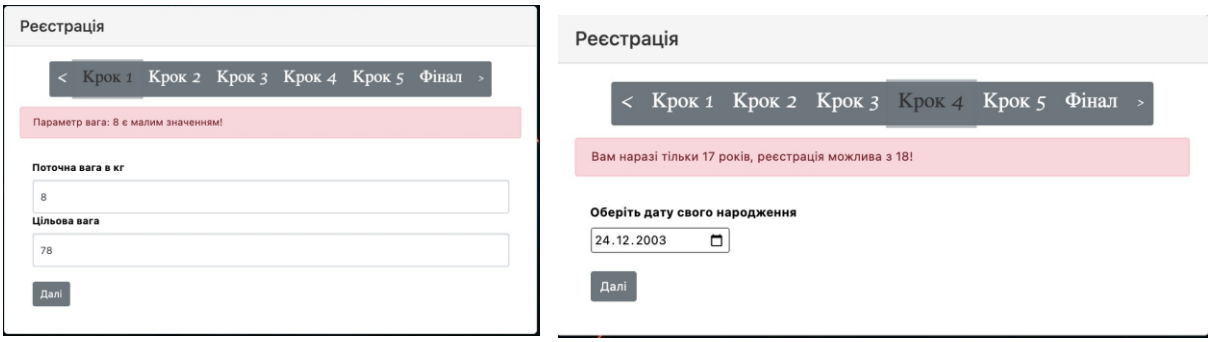


Рис. 4.8. Повідомлення при не валідному полі

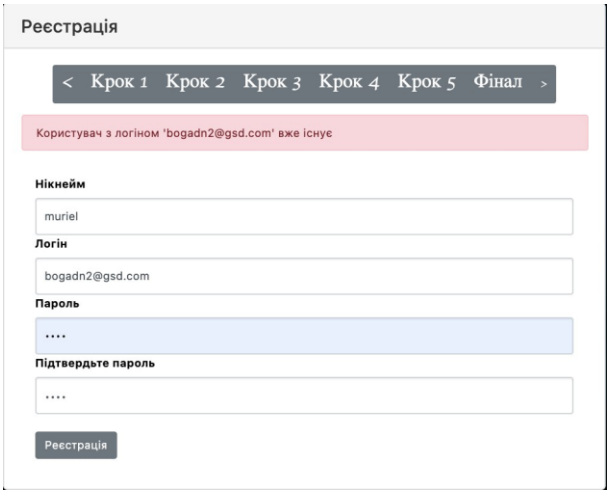


Рис. 4.9. Повідомлення якщо користувач вже існує

При успішній реєстрації виводиться відповідне повідомлення та з’являється кнопка переходу на сторінку входу (рис. 4.10)

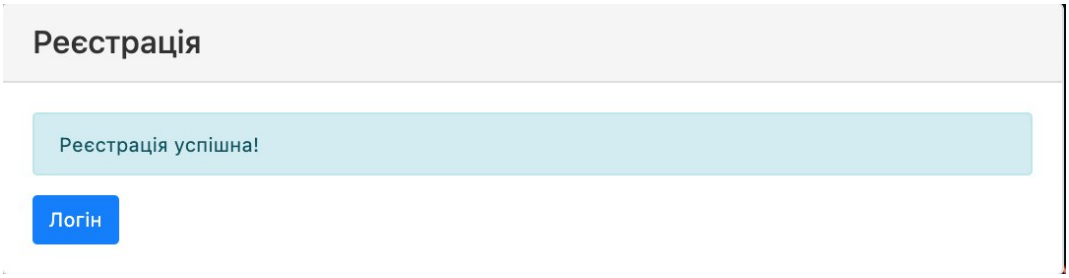


Рис. 4.10. Реєстрації успішна

Після успішної авторизації користувач направляється на сторінку персонального плану схуднення (рис 4.11) , що включає в себе такі показники, як денний бюджет калорій, необхідний для підтримання поточної ваги (кількість калорій, необхідних для схуднення,

відображаються на іншій сторінці), запланована вага схуднення, темп схуднення та орієнтовна дата досягнення цілі при умові, що будуть виконуватись всі рекомендації

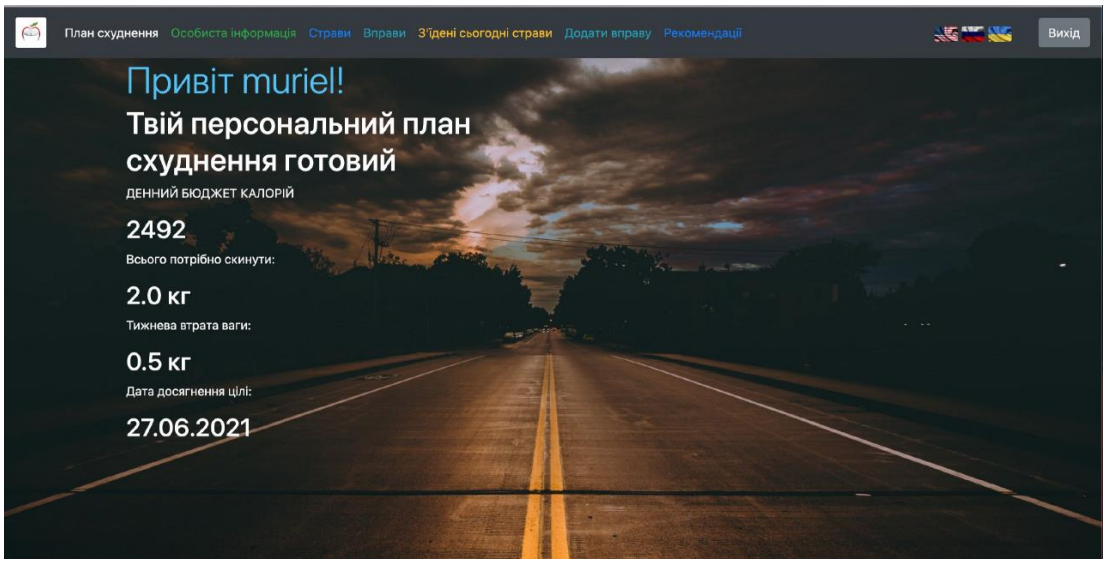


Рис. 4.11. Сторінка персонального плану схуднення

За потреби користувач в будь-який момент часу може підкорегувати деякі показники на сторінці редагування показників (рис. 4.12). Тут відображаються всі характеристики користувача, що він вводив під час реєстрації.

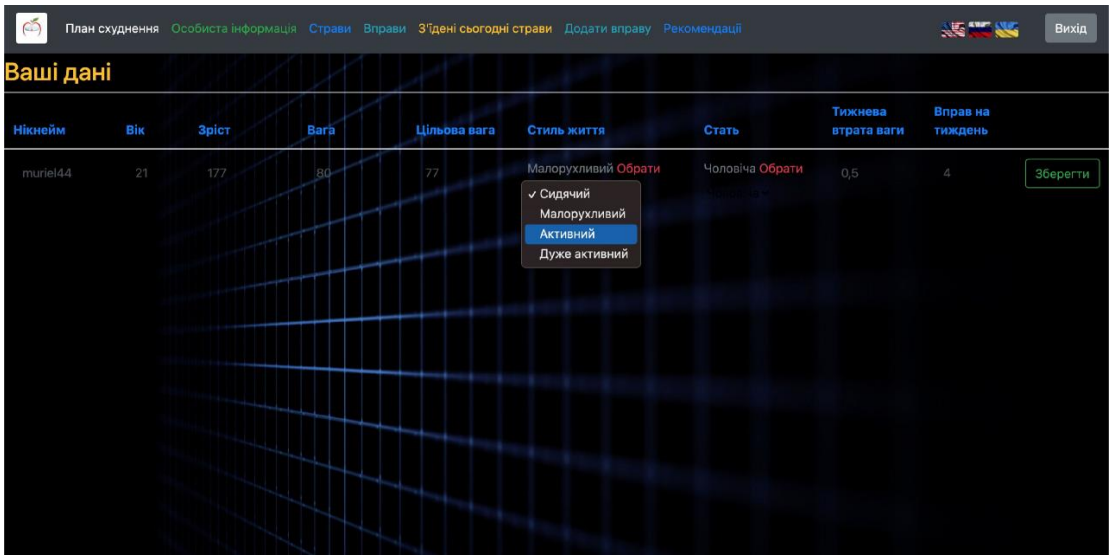
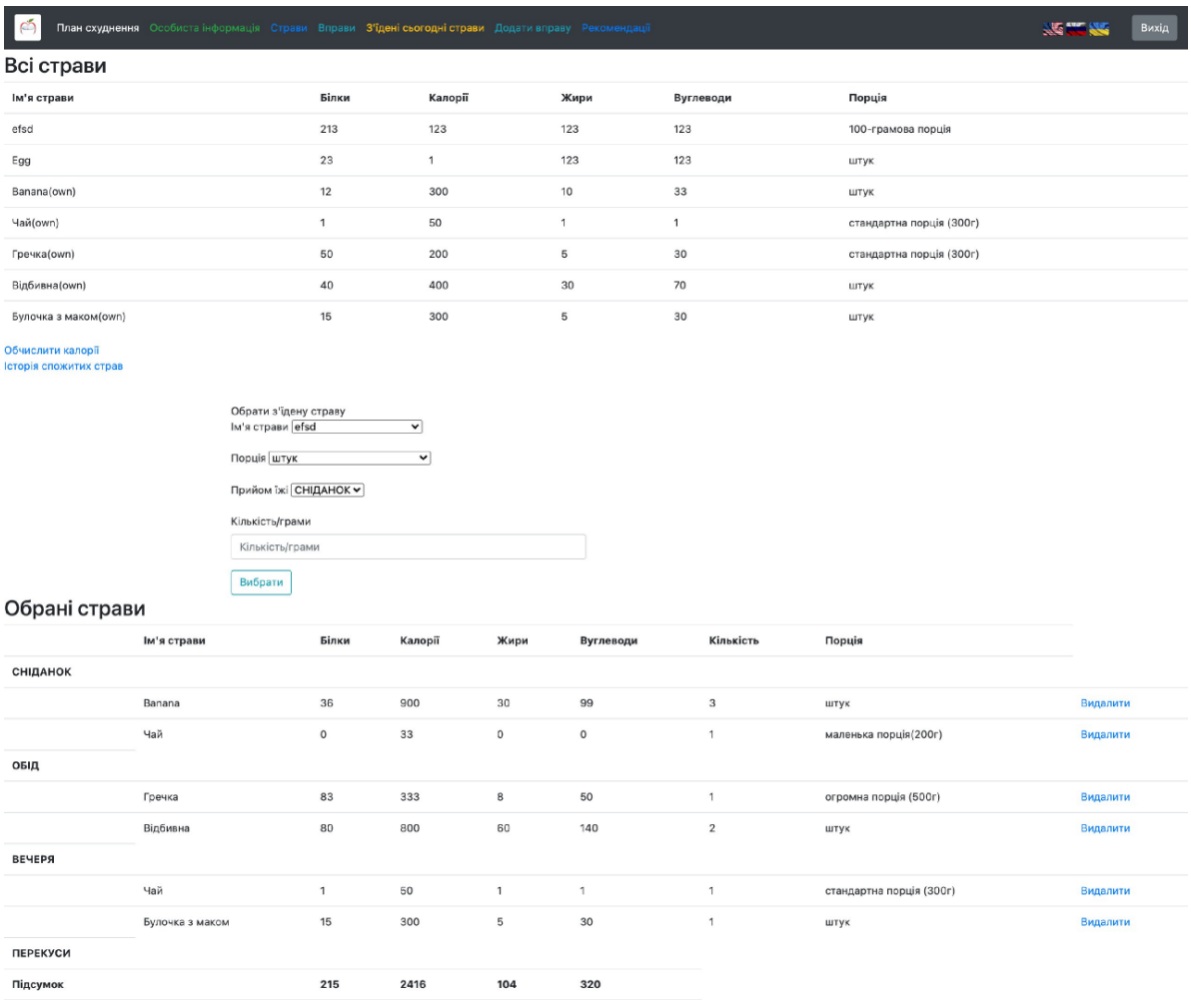


Рис. 4.12. Сторінка редагування показників

Також користувач має можливість вибору страв, які він з’їв протягом дня на сторінці вибору страв (рис. 4.13). На цій сторінці відображається інформація про популярні страви, їхні характеристики (білки, жири, вуглеводи та калорії). Користувач може обирати будь-яку страву з бази даних, обирати порцію страви а також час прийому їжі (сніданок, обід, вечеря чи перекус). Обрані страви зберігаються до бази даних, а також виводяться на цій же сторінці у вигляді списку, відсортованого по прийомам їжі. Також є можливість видалення вибраних вправ кнопкою ‘Видалити’



Ім'я страви	Білки	Калорії	Жири	Вуглеводи	Порція
efsd	213	123	123	123	100-грамова порція
Egg	23	1	123	123	штук
Banana(own)	12	300	10	33	штук
Чай(own)	1	50	1	1	стандартна порція (300г)
Гречка(own)	50	200	5	30	стандартна порція (300г)
Відбивна(own)	40	400	30	70	штук
Булочка з маком(own)	15	300	5	30	штук

	Ім'я страви	Білки	Калорії	Жири	Вуглеводи	Кількість	Порція	
СНІДАНОК								
	Banana	36	900	30	99	3	штук	Видалити
	Чай	0	33	0	0	1	маленька порція(200г)	Видалити
ОБІД								
	Гречка	83	333	8	50	1	огромна порція (500г)	Видалити
	Відбивна	80	800	60	140	2	штук	Видалити
ВЕЧЕРЯ								
	Чай	1	50	1	1	1	стандартна порція (300г)	Видалити
	Булочка з маком	15	300	5	30	1	штук	Видалити
ПЕРЕКУСИ								
Підсумок		215	2416	104	320			

Рис. 4.13. Сторінка вибору страв

За потреби, якщо в базі даних не було знайдено необхідної страви, користувач має можливість додавання власної страви з вказанням показників цієї страви (рис. 4.14):

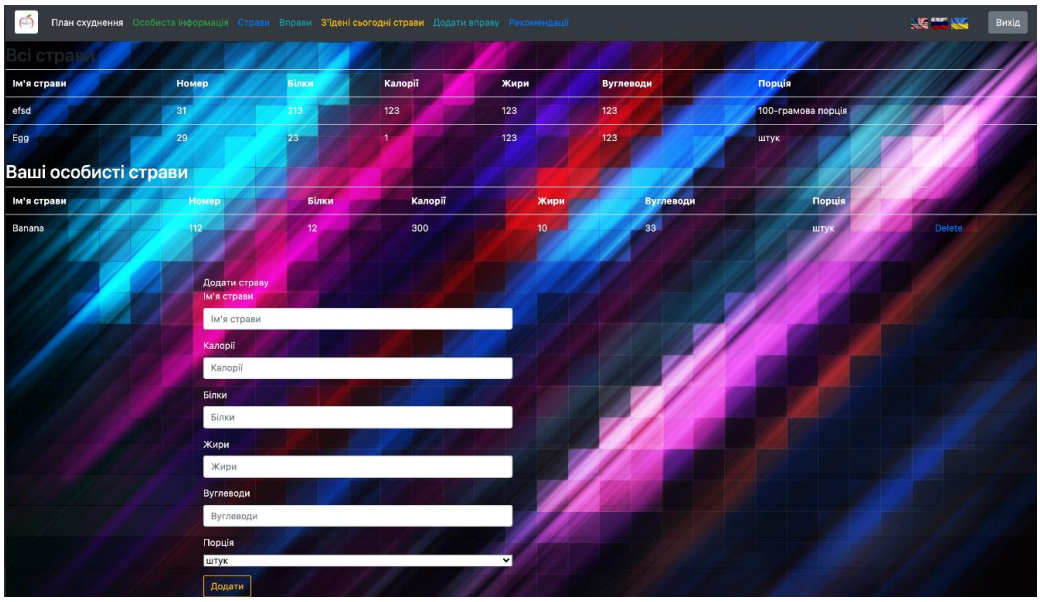


Рис. 4.14. Сторінка додавання нової страви

Такий же функціонал існує для вибору виконаних впродовж дня фізичних навантажень (рис 4.16), а також додавання нової вправи, якщо потрібної не знайдено в базі даних (рис. 4.15)

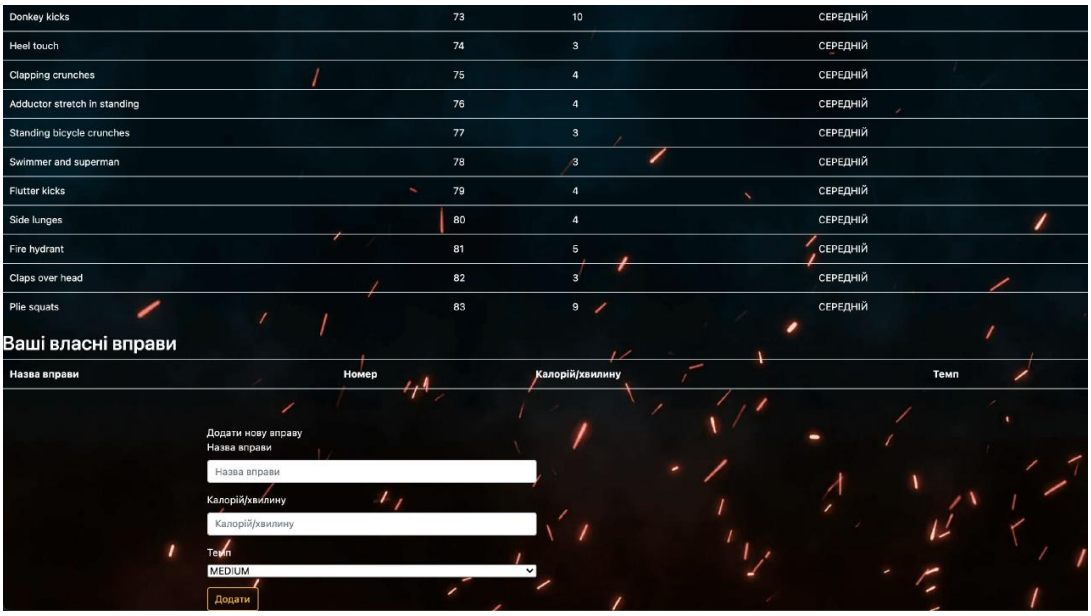


Рис. 4.15. Сторінка вибору вправ

Рис. 4.16. Сторінка додавання нової вправи

В будь-який момент часу користувач за потреби має можливість переглянути історію спожитих страв за весь час або ж в якийсь конкретний день при виборі дати з календаря (рис.4.17). При перегляді всієї історії дані

сортуються за датою (спочатку новіші), а також наявний підсумок спожитих калорій, білків, жирів, вуглеводів за цей день

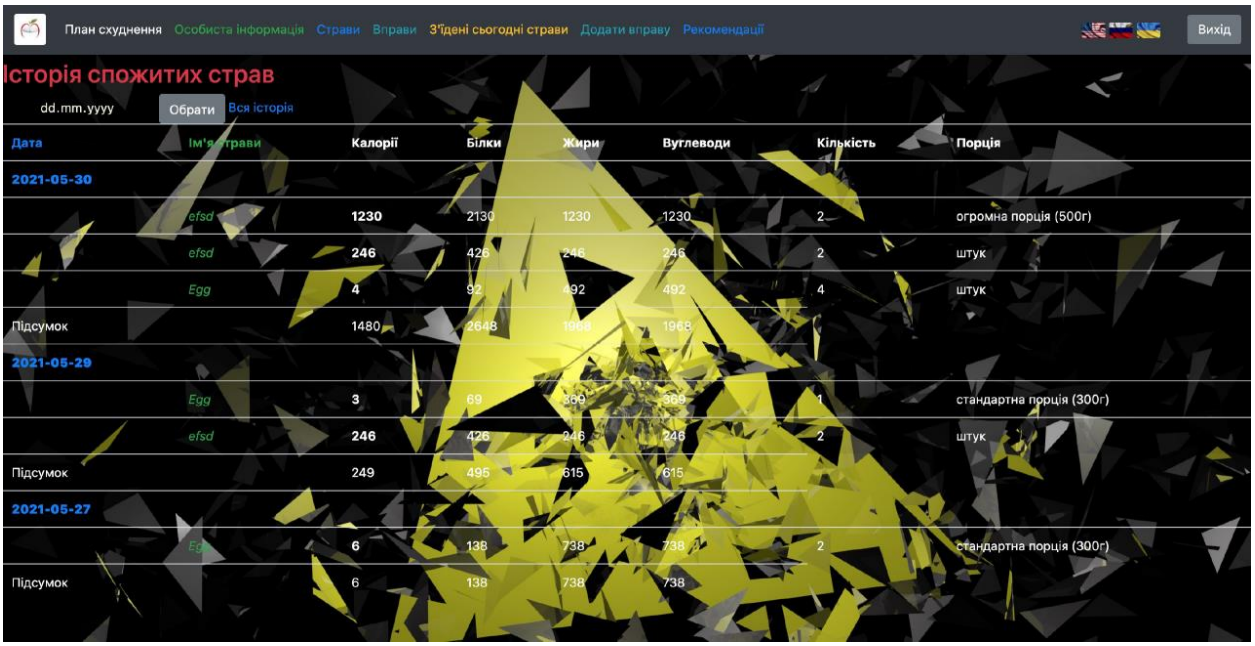


Рис. 4.17. Історія спожитих страв

А також історія виконаних вправ (рис.4.18)

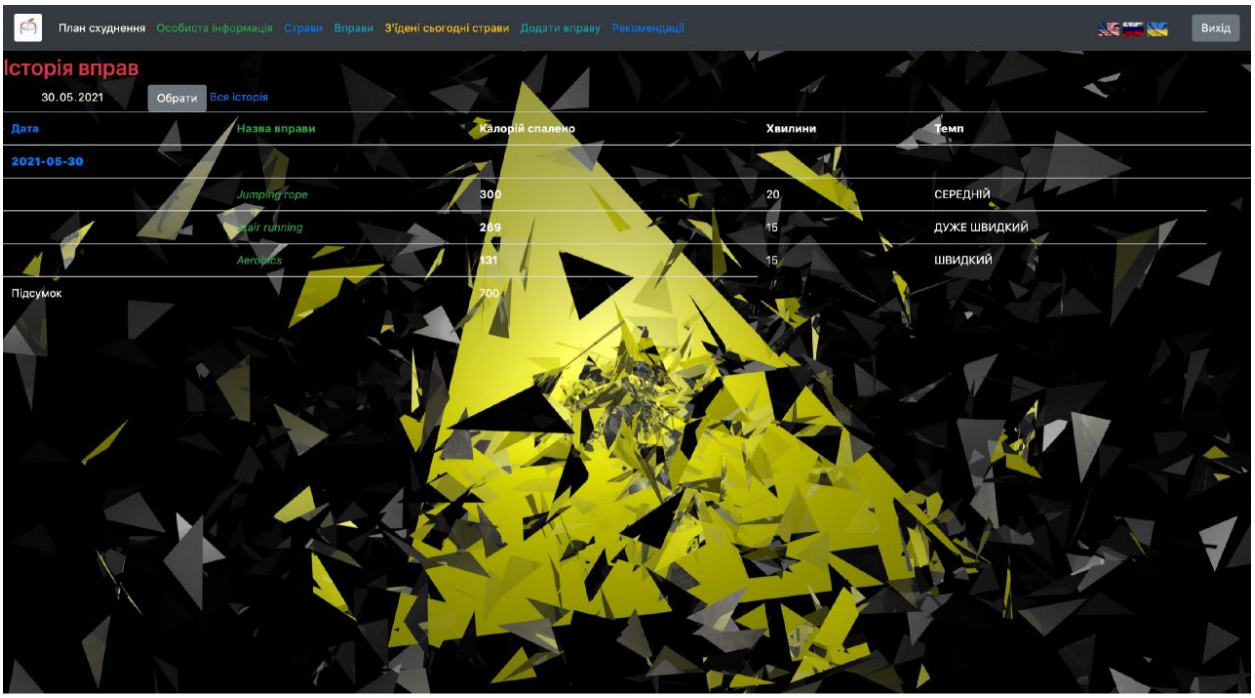


Рис. 4.18. Історія виконаних вправ

На сторінках додавання страв і вправ є посилання на сторінку, на якій демонструється компактна та лаконічна інформація по прогресу надходження калорій за поточний день, спалення калорій, скільки калорій ще можна спожити (рис 4.19). Зверху відображається денний баланс калорій, які необхідно споживати для того, щоб скидати вагу в тому темпі, який було вказано при реєстрації (в кілограмах в тиждень). Є також картки, що демонструють кількість спожитих калорій в кожен прийом їжі, при наведенні відображаються страви, що були спожиті в цей прийом їжі. Також є картка, що демонструє сумарну кількість спалених калорій під час виконання вправ.

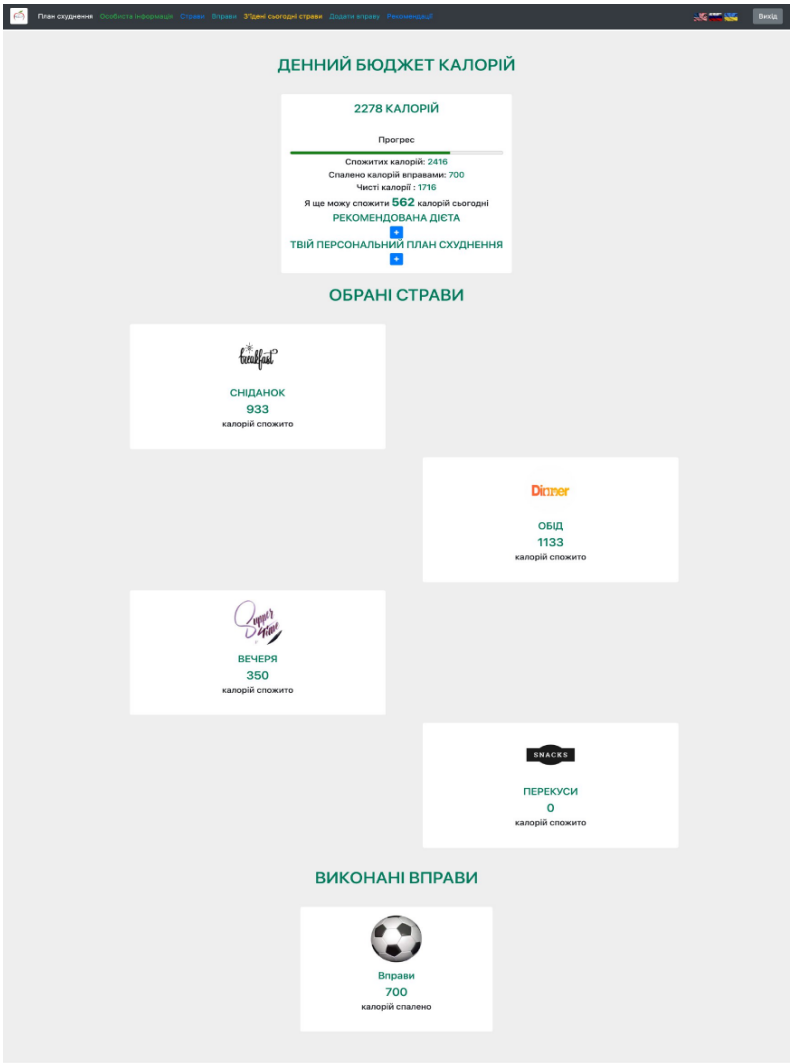


Рис. 4.19. Сторінка прогресу та бюджету калорій

При наведенні на картку страв відображається повна статистика спожитих справ у цей прийом їжі (Рис.4.20)

Рис. 4.20. Перевернута картка страв

При наведенні на картку вправ відображається повна статистика виконаних за цей день вправ (Рис.4.21)

Рис. 4.21. Перевернута картка вправ

Також на даній сторінці є посилання на сторінки рекомендованої дієти, а також персонального плану виконання вправ для схуднення. Сторінка персонального плану виконання вправ містить список днів, кожен з яких містить інформацію по тому, які вправи слід виконувати, а також в якій кількості. Це залежить від того, яку ціль поставив користувач, а саме з яким темпом він хоче худнути: якщо виставлений вищий темп – потрібно довше виконувати вправи і навпаки. Сторінка рекомендованих вправ (рис.4.22) містить список днів, для кожного з яких розроблено план виконання вправ в залежності від темпу схуднення. Якщо сьогодні заплановано відпочинок – система сповістить про це.

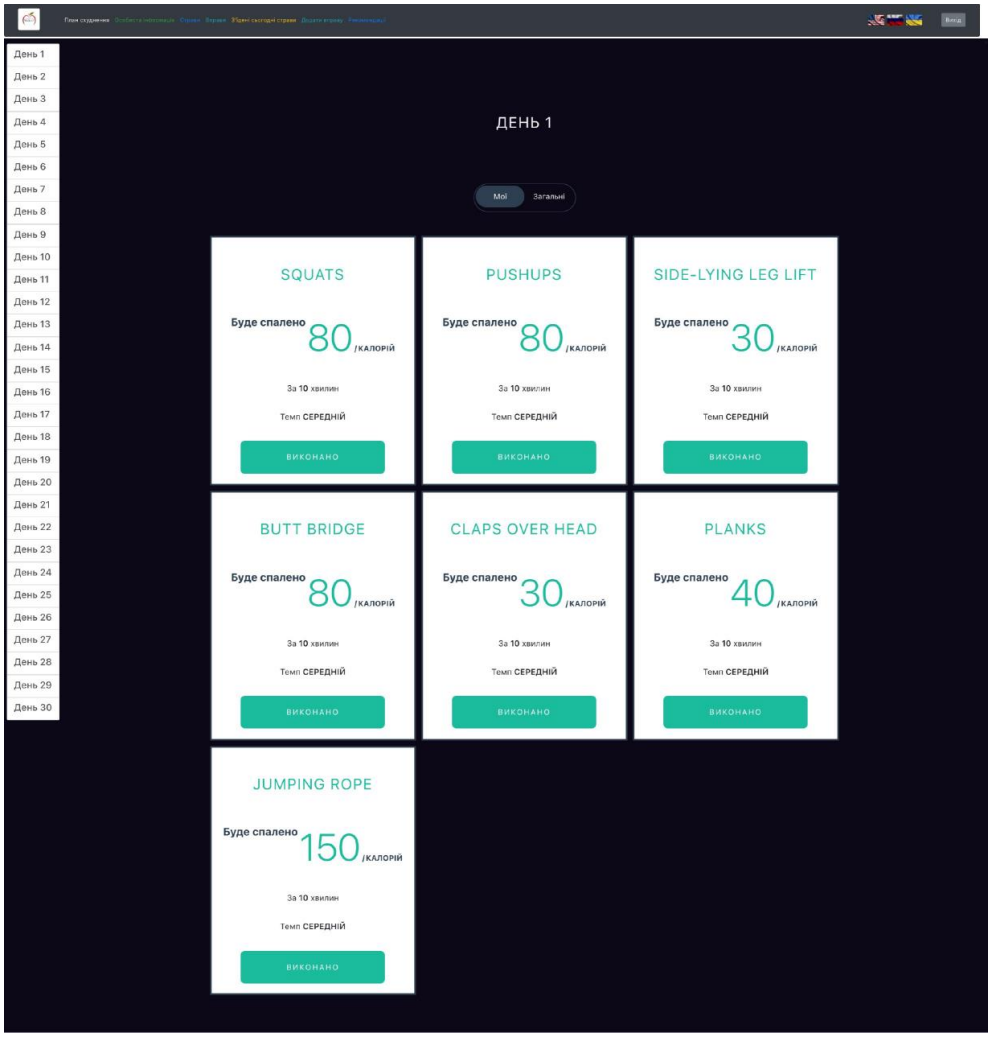


Рис. 4.22 Рекомендований план виконання вправ на перший день

Натиснувши на кнопку перемикання вправ на загальні відображається загальна інформація по кожній з запропонованих вправ (рис.4.23)

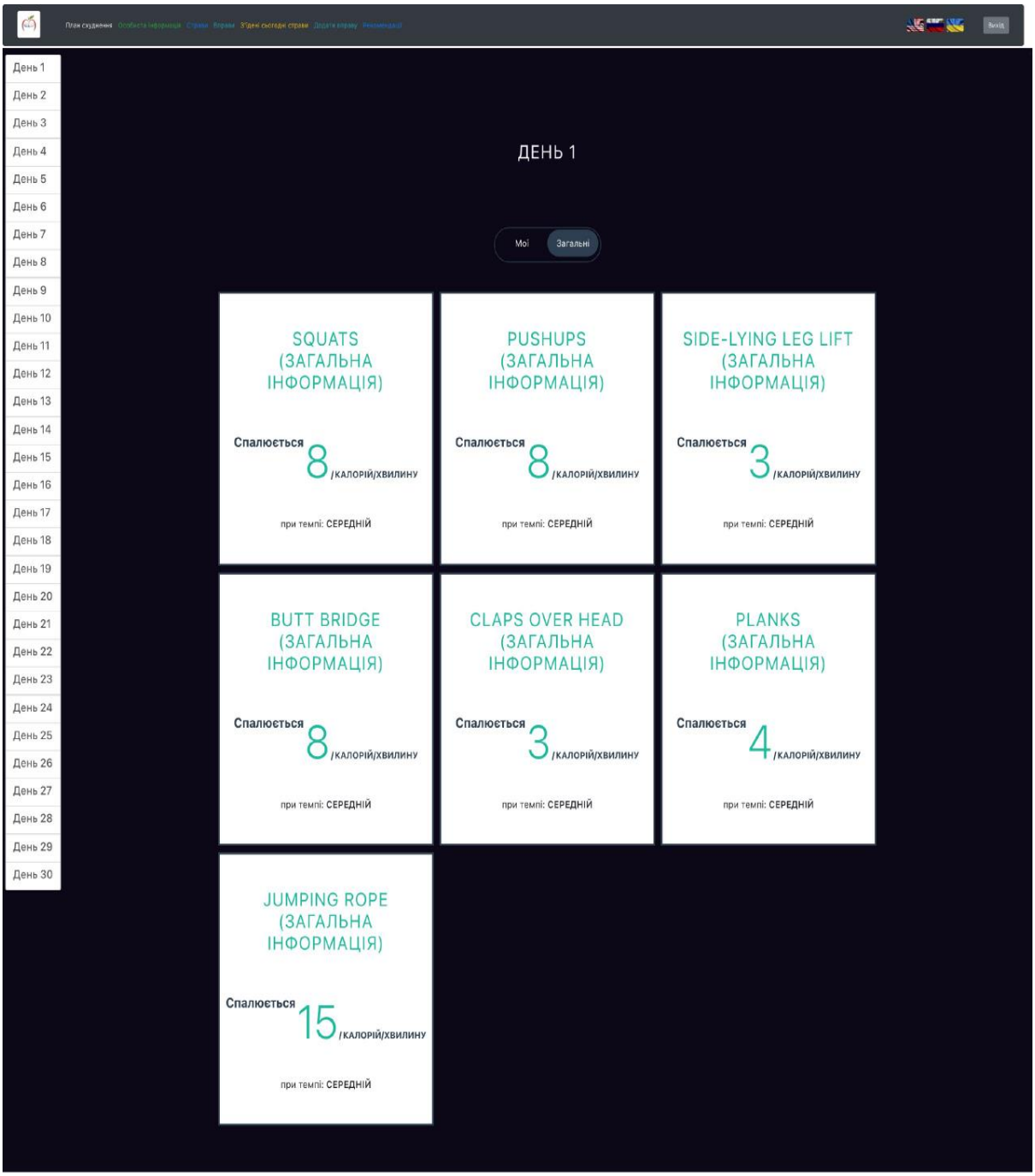


Рис. 4.23. Загальна інформація по рекомендованим вправам

Рекомендований план вправ на інший день (рис. 4.24)

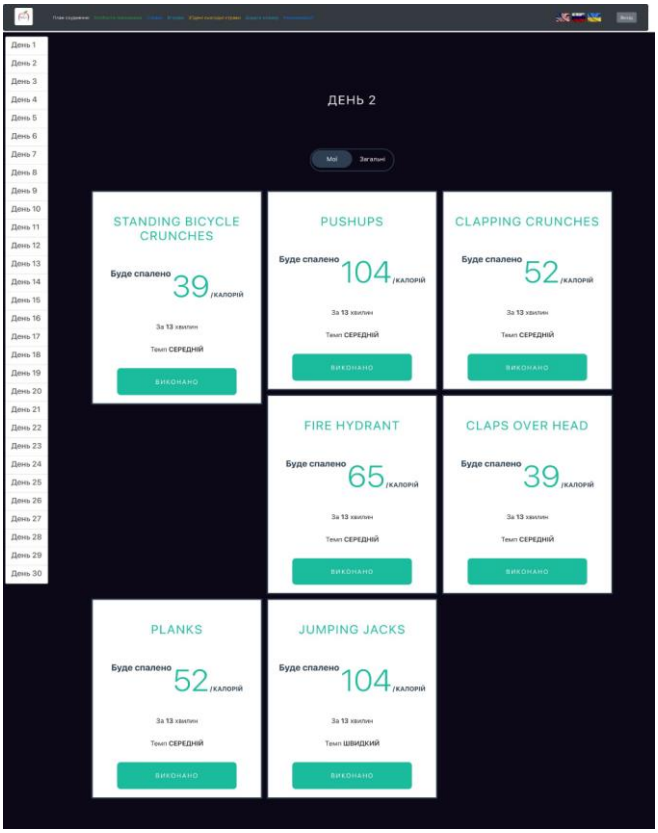


Рис. 4.24. Рекомендований план виконання вправ на другий день

Сторінка рекомендованої дієти містить інформацію по загальному рекомендованому графіку харчування і являється рекомендацією, як можна побудувати свою дієту (рис. 4.25)

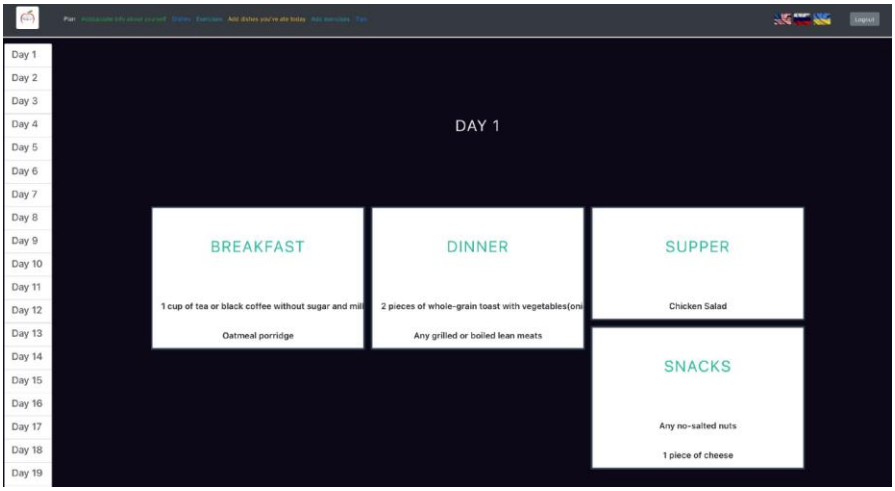


Рис. 4.25. Сторінка загальної дієти

Також за допомогою навігаційної панелі зверху користувач може перейти на сторінку практичних рекомендації по схудненню, яка являє собою набір статтей та порад, яких бажано було б дотримуватись для ефективного схуднення (рис.4.26)

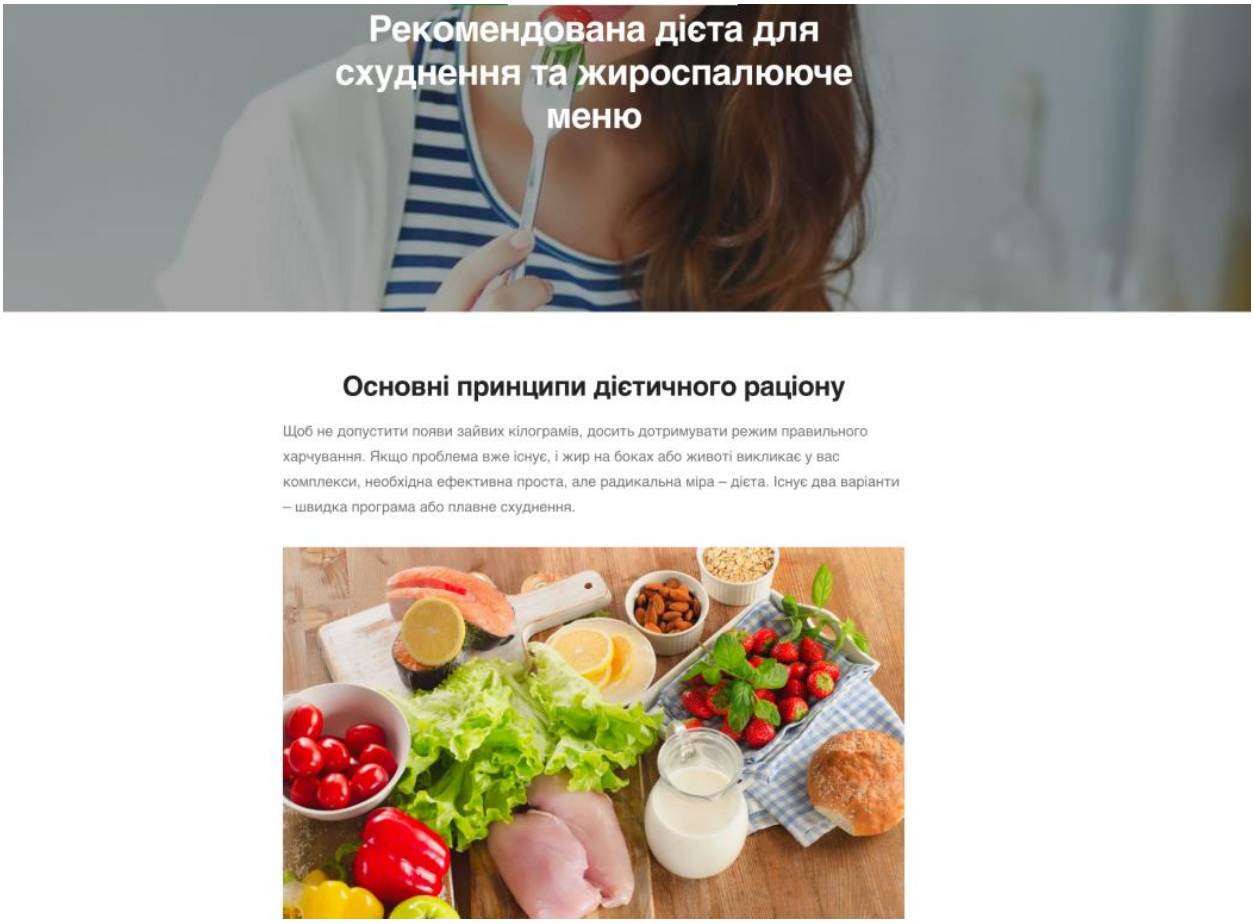


Рис.4.26. Сторінка порад та рекомендацій

ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі була представлена інструкція користувача для реалізованого веб-застосунку. Дана інструкція демонструє функціональність системи, а також те, як користувач може використовувати цю функціональність і своїх потребах.

ЗАГАЛЬНІ ВИСНОВКИ

В ході виконання даної дипломної роботи було реалізовано програмне забезпечення для веб-додатку з метою допомогти користувачам позбутись лишньої ваги та контролювати обіг спожитих та спалених калорій.

У процесі розробки веб-додатку були досліджені і вирішені такі задачі:

- Проведений аналіз предметної області та сформульовані основні вимоги до системи.
- Проаналізовано існуючі популярні веб-додатки та виконано порівняльний аналіз аналогів.
- На основі порівняльного аналізу були виявлені додаткові корисні можливості розроблюваної системи.
- Обґрунтовано вибір платформи для системи.
- Здійснено огляд та вибір технологій для розробки системи, які стали зручним та необхідним інструментом для реалізації поставленої мети.
- Описано проектування програмної системи, що відповідає поставленим цілям. Наведено приклад роботи частини програми, а також показана реалізація в програмному коді.
- Описано архітектуру розробленої бази даних, а також сутностей, що її наповнюють.
- Реалізована клієнт-серверна система.
- Представлена інструкція користувача, яка дозволить з легкістю використовувати даний веб-додаток.

Підсумовуючи проведену роботу та одержані результати, можна відзначити, що всі поставлені задачі вирішені, а мета дипломної роботи є досягнута.

ЛІТЕРАТУРА

1. PostgreSQL: альтернатива системам управління базами даних з відкритим кодом [Електронний ресурс] - Режим доступу: https://pdfs.semanticscholar.org/c95a/fa04bc1b8d6cd7273349948fad509955 94b9.pdf
2. The Java Virtual Machine Specification — Спецификация JVM [Електронний ресурс] - Режим доступу: https://docs.oracle.com/javase/specs/.
3. Офіційний сайт PostgreSQL [Електронний ресурс] - Режим доступу: https://www.postgresql.org/.
4. Spring Framework [Електронний ресурс] - Режим доступу: https://docs.spring.io/spring/docs/current/spring-frameworkreference
5. Ben A. Spring Security Reference [Електронний ресурс] / A. Ben, T. Luke, W. Rob. – 2020. – [Електронний ресурс] - Режим доступу: https://docs.spring.io/springsecurity/site/docs/current/reference/html5/.
6. Java SE 9 Documentation [Електронний ресурс]. – Режим доступу:http://docs.oracle.com/javase/9/docs
7. Introduction to Using Thymeleaf in Spring [Електронний ресурс] - Режим доступу: https://www.baeldung.com/thymeleaf-in-spring-mvc
8. Front end та back end [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Front_end_%D1%82%D0%B0_back_end.
9. Comparison of predictive equations for resting metabolic rate in healthy nonobese and obese adults: a systematic review [Електронний ресурс] - Режим доступу: https://pubmed.ncbi.nlm.nih.gov/15883556/
10. Mifflin-St Jeor prediction equation for resting energy expenditure [Електронний ресурс]. – Режим доступу: https://academic.oup.com/ajcn/article-abstract/51/2/241/4695104

11. Hot to Cut Calories Without Starving Yourself [Електронний ресурс] - Режим доступу: https://blog.myfitnesspal.com/how-to-cut-calories-without-starving-yourself
12. Weight loss: 6 strategies for success [Електронний ресурс] - Режим доступу: https://www.mayoclinic.org/healthy-lifestyle/weight-loss/in-depth/weight-loss/art-20047752
13. Obesity and overweight statistic [Електронний ресурс] - Режим доступу: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight#:~:text=In%202016%2C%20more%20than%201.9%20billion%20adults%20aged%2018%20years,women)%20were%20obese%20in%202016
14. The 10 Best Weight Loss Apps Central [Електронний ресурс] - Режим доступу: https://www.healthline.com/nutrition/10-best-weight-loss-apps
15. Essential Guide to Metabolism [Електронний ресурс] - Режим доступу: https://blog.myfitnesspal.com/essential-guide-metabolism/
16. U.S. DEPARTMENT OF AGRICULTURE FoodData Central [Електронний ресурс] - Режим доступу: https://fdc.nal.usda.gov/api-guide.html
17. Open Food Facts – the free food products database [Електронний ресурс] - Режим доступу: https://world.openfoodfacts.org/
18. Jpa Repositories Documentation database [Електронний ресурс] – Режим доступу: https://docs.spring.io/springdata/jpa/docs/current/reference/html

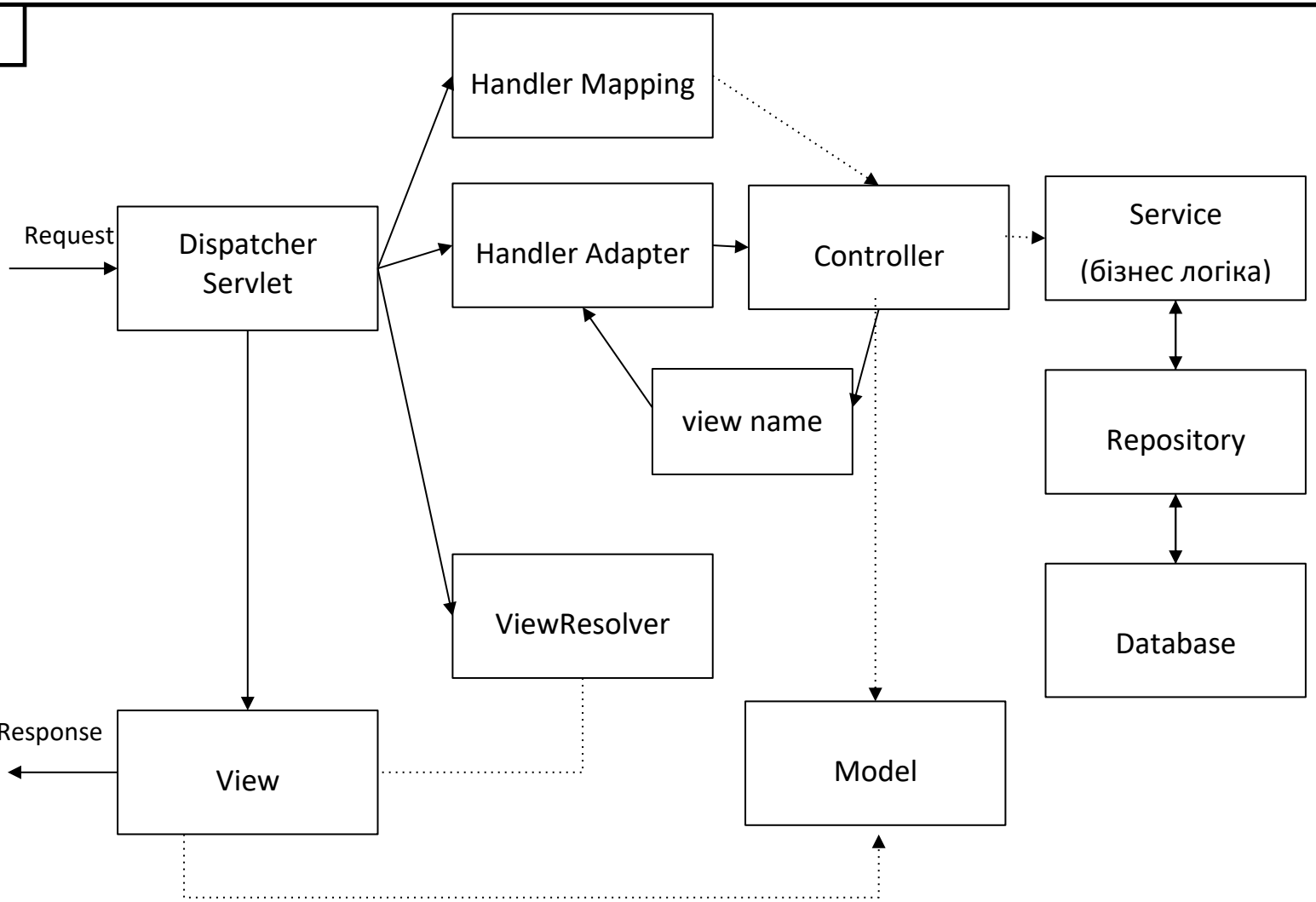
ДОДАТОК А

Веб-додаток для комплексної системи схуднення

СХЕМА ВЗАЄМОДІЇ

Аркушів 1

Київ 2021 р.



Зм.	Арк.	Прізвище	Підп.	Дата	ІАЛЦ 467200.004 Д1		
Розроб.		Симончук Б.Я.			*Веб-додаток для комплексної системи схуднення* *Додаток*	*Літ.*	*Арк.* / *Аркушів* 1
Перевірив.		Іванішев Б.В.					
Н. кон.		Сімоненко В.П.				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, кафедра ОТ гр. ІП-74*	
Затв.		Стіренко С.Г.					

ДОДАТОК Б
Веб-додаток для комплексної системи схуднення

ДІАГРАМА КЛАСІВ

Аркушів 1

Київ 2021 р.

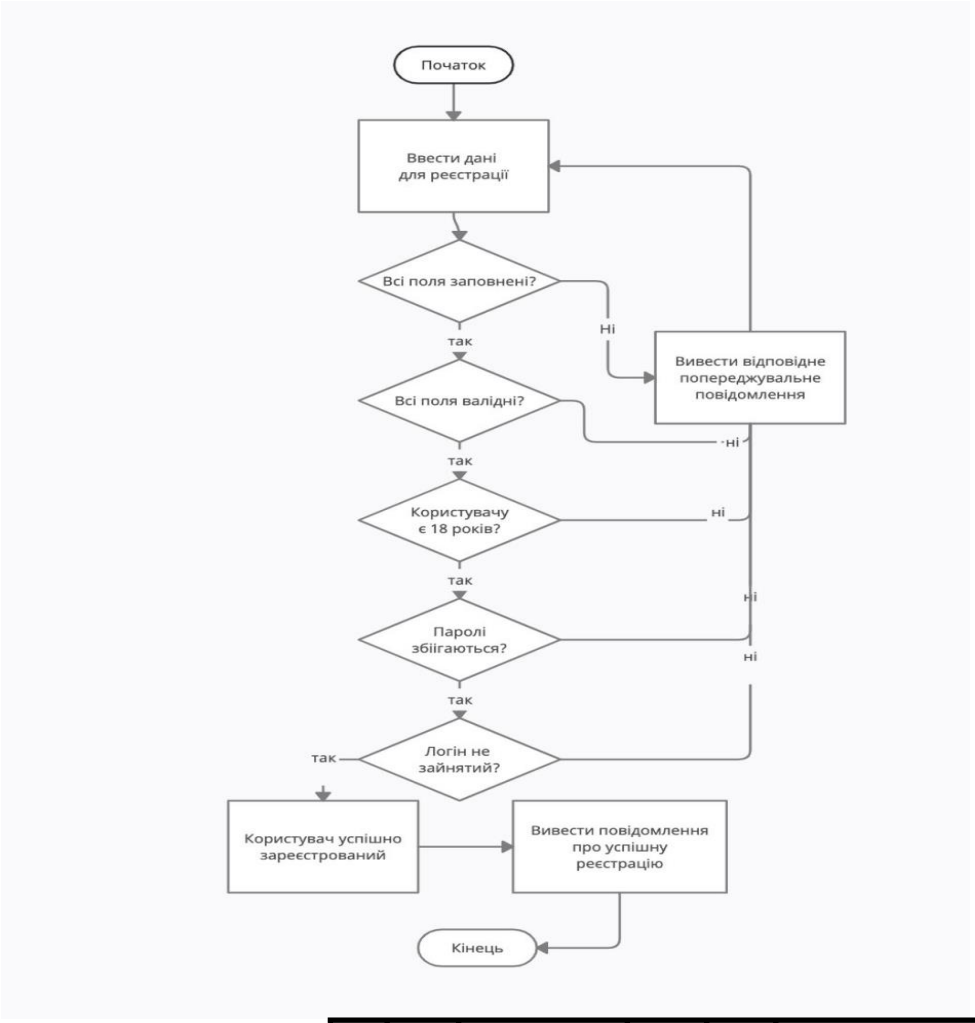
Зм.	Арк.	Прізвище	Підп.	Дата	ІАЛЦ 467200.005 Д2		
Розроб.		Симончук Б.Я.			*Веб-додаток для комплексної системи схуднення Додаток*	*Літ.*	*Аркушів*
Перевірив.		Іванішев Б.В.					1
Н. кон.		Сімоненко В.П.				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, кафедра ОТ гр. ІП-74*	
Затв.		Стіренко С.Г.					

ДОДАТОК В
Веб-додаток для комплексної системи схуднення

АЛГОРИТМ АВТОРИЗАЦІЇ В СИСТЕМУ

Аркушів 1

Київ 2021 р.



Зм.	Арк.	Прізвище	Підп.	Дата	ІАЛЦ 467200.006 ДЗ		
Розроб.		Симончук Б.Я.			*Веб-додаток для комплексної системи схуднення* *Додаток*	*Літ.*	*Арк.* / *Аркушів*
Перевірив.		Іваніщев Б.В.					1
Н. кон.		Сімоненко В.П.				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, кафедра ОТ гр. ІП-74*	
Затв.		Стіренко С.Г.					

ДОДАТОК Г
Веб-додаток для комплексної системи схуднення

ЛІСТИНГ ПРОГРАМИ

Аркушів 20

Київ 2021 р.

```
@Slf4j
@Service
public class UserService implements UserDetailsService {

    private final UserRepository userRepository;
    private final DishesRepository dishesRepository;
    private final UserInfoRepository userInfoRepository;
    private final UserDishRepository userDishRepository;
    private final PasswordEncoder passwordEncoder;

    @Autowired
    public UserService(UserRepository userRepository, DishesRepository dishesRepository, UserInfoRepository userInfoRepository, UserDishRepository userDishRepository, PasswordEncoder passwordEncoder) {
        this.userRepository = userRepository;
        this.dishesRepository = dishesRepository;
        this.userInfoRepository = userInfoRepository;
        this.userDishRepository = userDishRepository;
        this.passwordEncoder = passwordEncoder;
    }

    @Override
    public UserDetails loadUserByUsername(String login) throws UsernameNotFoundException {
        return userRepository.findByLogin(login)
                .orElseThrow(()->new UsernameNotFoundException(login));
    }
    public MyUser getUserByUsername(String login) throws UsernameNotFoundException{
      return (MyUser) loadUserByUsername(login);//TODO delete this method and use loadUserByUsername
```

```
    }

    public UsersDTO getAllUsers() {
        //TODO checking for an empty user list
        return new UsersDTO(userRepository.findAll());
    }

    private int getAge(LocalDate dob){
        return (int)ChronoUnit.YEARS.between(dob,LocalDate.now());
    }

    private String isCompleteUserOrElseGetEmptyField(UserDTO userDto){
        return !userDto.getPassword().isEmpty() ?
                (!userDto.getLogin().isEmpty() ?
                        (!userDto.getConfirmedPassword().isEmpty() ?
                                null : "підтвердження паролю") : "логін") : "пароль";
    }

    private boolean isPresent(Number field){return field != null &&
field.floatValue() != 0;}
    private String isCompleteUserInfoOrElseGetEmptyField(UserInfoDto
userInfoDto){
        return isPresent(userInfoDto.getWeight()) ?
                (isPresent(userInfoDto.getGoalWeight()) ?
                        (isPresent(userInfoDto.getHeight()) ?
                                (userInfoDto.getMale() != null ?
                                        (userInfoDto.getDob() != null ?
                                                (userInfoDto.getLifestyle() != null ?
```

```
                                        (isPresent(userInfoDto.getExercisesPerWeek())?

(isPresent(userInfoDto.getWeightLossPerWeek()) ?

                                                                    null : "втрата ваги в тиждень")
: "кількість вправ в тиждень") : "стиль життя") : "день народження"): "стать")
: "зріст") : "цільова вага") : "вага";
        }


        private String getMessageIfWrongRange(UserInfoDto userInfoDto){
            String message = "Параметр  %s: %d є %s!";


            boolean isBig = false;


            if((isBig = userInfoDto.getWeight() > 300) || userInfoDto.getWeight() <
30)
                return String.format(message, "вага", userInfoDto.getWeight(), (isBig
? "великим" : "малим") + " значенням");
            if((isBig = userInfoDto.getGoalWeight() > 300) ||
userInfoDto.getGoalWeight() < 30)
                return String.format(message, "цільова вага",
userInfoDto.getGoalWeight(), (isBig ? "великим" : "малим") + " значенням");
            if((isBig = userInfoDto.getHeight() > 230) || userInfoDto.getHeight() <
70)
                return String.format(message, "зріст", userInfoDto.getHeight(), (isBig
? "великим" : "малим") + " значенням");


            return null;
        }


        public void saveNewUser (UserDTO userDto, UserInfoDto userInfoDto)
throws RegFailedException {
            String emptyField;
```

```
        if((emptyField =
isCompleteUserInfoOrElseGetEmptyField(userInfoDto)) != null ||
           (emptyField = isCompleteUserOrElseGetEmptyField(userDto)) !=
null)
            throw new EmptyFieldException(emptyField);

        String msg;
        if((msg = getMessageIfWrongRange(userInfoDto)) !=null)
            throw new ParameterRangeException(msg);

        Optional<MyUser> optionalMyUser =
userRepository.findByLogin(userDto.getLogin());

        if(optionalMyUser.isPresent())
            throw new UserAlreadyExistsException(userDto.getLogin());
        if(!userDto.getPassword().equals(userDto.getConfirmedPassword()))
            throw new DifferentPasswordsException();

        int age = getAge(userInfoDto.getDob());
        if(age < 18)
            throw new NotAdultException(age);

        MyUser user = MyUser.builder()
                .login(userDto.getLogin())
                .password(passwordEncoder.encode(userDto.getPassword()))
                .role(UserRole.USER)
                .isEnabled(true)
                .isAccountNonExpired(true)
                .isCredentialsNonExpired(true)
```

```
                .isAccountNonLocked(true)
            .build();
        userRepository.save(user);

        UserInfo userInfo = UserInfo.builder()
                .nickname(userInfoDto.getNick())
                .weight(userInfoDto.getWeight())
                .goalWeight(userInfoDto.getGoalWeight())
                .height(userInfoDto.getHeight())
                .age(age)
                .male(Male.valueOf(userInfoDto.getMale()))
                .lifestyle(Lifestyle.valueOf(userInfoDto.getLifestyle()))
                .exercisesPerWeek(userInfoDto.getExercisesPerWeek())
                .weightLossPerWeek(userInfoDto.getWeightLossPerWeek())
                .user(user)
            .build();
        userInfoRepository.save(userInfo);
    }


    public void saveUser(MyUser user){
        userRepository.save(user);
    }
    public MyUser findById(Long id){
        return userRepository.findById(id).get();
    }
}
```

```
@Service
public class UserInfoService {
    private final UserInfoRepository userInfoRepository;
    private final UserRepository userRepository;

    @Autowired
    UserInfoService(UserInfoRepository userInfoRepository,
                    UserRepository userRepository){
        this.userInfoRepository = userInfoRepository;
        this.userRepository = userRepository;
    }
    public UserInfo findByUser(MyUser user){
        return userInfoRepository.findByUser(user);
    }
    public void saveNewUserInfo (UserInfo userInfo) {
        userInfoRepository.save(userInfo);
    }

    public List<UserInfo> getAllUserInfos(){
        return userInfoRepository.findAll();
    }

    public UserInfo findActiveInfoByUser(MyUser user){
        return userInfoRepository.findByUser(user);
    }

    public List<UserInfo> findAllActiveInfoByUser(MyUser user){
```

```
        return userInfoRepository.findAllByUser(user);
    }

    //@Transactional -> userInfo = userInfoRepo.findByUser or byId ->
userInfo.age(newAge).height(newHeight) -> that's all need
    @Transactional
    public void updateUserInfo(UserInfo userInfo, MyUser user){
        UserInfo info = userInfoRepository.findByUser(user);
        info.setAge(userInfo.getAge());
        info.setWeight(userInfo.getWeight());
        info.setGoalWeight(userInfo.getGoalWeight());
        info.setHeight(userInfo.getHeight());
        info.setNickname(userInfo.getNickname());
        info.setMale(userInfo.getMale());
        info.setLifestyle(userInfo.getLifestyle());
        info.setExercisesPerWeek(userInfo.getExercisesPerWeek());
        info.setWeightLossPerWeek(userInfo.getWeightLossPerWeek());
//        userInfoRepository.updateUserInfo(age,height,weight,lifestyle,user);
    }

    private double getLifeStyleCoef(Lifestyle ls){
        return ls == Lifestyle.PASSIVE ? 1.2 :
                ls == Lifestyle.SEDENTARY ? 1.375 :
                ls == Lifestyle.ACTIVE ? 1.55 :
                        1.725;//             ls == Lifestyle.VERY_ACTIVE ? 1.725 : 0;
    }

    private int calcDailyCalBudget(UserInfo userInfo){
        return (int) ((10*userInfo.getWeight()
```

```
                + 6.25 * userInfo.getHeight()
                - 5*userInfo.getAge()
                + (userInfo.getMale() == Male.MALE ? 5: -161))
                * getLifeStyleCoef(userInfo.getLifestyle()));
        }

        public int calcCaloriesReduce(UserInfo userInfo){
            return (int)(calcDailyCalBudget(userInfo) *coeffReduce(userInfo));

    //        return (int)(calcDailyCalBudget(userInfo) * coeff2);
        }

        private double coeffReduce(UserInfo userInfo){
            double coeff = Math.pow(1.375 /
getLifeStyleCoef(userInfo.getLifestyle()), .6);

            double coeff2 = userInfo.getWeightLossPerWeek() / 0.5;

            coeff2*=0.2 * coeff;
            return coeff2;
    //        return (int)(calcDailyCalBudget(userInfo) * coeff2);
        }

        public int calcDailyBudgetWhenLooseWeight(UserInfo userInfo){
            int budgetWithoutLosingWeight = calcDailyCalBudget(userInfo);
            System.out.println(budgetWithoutLosingWeight);

            double v = budgetWithoutLosingWeight * (1 - coeffReduce(userInfo));
            System.out.println(v);
```

```
            return (int) v;
        }

        public PlanDto getPlan(MyUser user){
            UserInfo info = userInfoRepository.findByUser(user);

            float totalWeightLoss = info.getWeight() - info.getGoalWeight();
            int daysForGoal =
(int)Math.ceil(totalWeightLoss/info.getWeightLossPerWeek()*7);

            return PlanDto.builder()
                    .dailyCalorieBudget(calcDailyCalBudget(info))
                    .totalWeightLoss(totalWeightLoss)
                    .weeklyWeightLoss(info.getWeightLossPerWeek())

.goalDate(LocalDate.now().plusDays(daysForGoal).format(DateTimeFormatter.of
Pattern("dd.MM.yyyy")))
                .build();
        }
    }
    @Service
    public class UserExerciseService {
        private final UserExerciseRepository userExerciseRepository;
        private final ExerciseRepository exerciseRepository;
        private final UserInfoService userInfoService;
        @Autowired
        public UserExerciseService(UserExerciseRepository
userExerciseRepository, ExerciseRepository exerciseRepository, UserInfoService
userInfoService){
            this.userExerciseRepository = userExerciseRepository;
```

```
        this.exerciseRepository = exerciseRepository;
        this.userInfoService = userInfoService;
    }
    public void save(UserExercise userExercise){
        userExerciseRepository.save(userExercise);
    }
    public List<UserExercise> findAllByUser(MyUser user){
        return userExerciseRepository.findAllByUser(user);
    }

    public List<UserExercise> findAllByUserAndDate(MyUser user,
LocalDate localDate){
        return userExerciseRepository.findAllByUserAndDate(user, localDate);
    }

    public List<UserExercise> findAllByExercise(Exercise exercise){
        return userExerciseRepository.findAllByExercise(exercise);
    }
    public UserExercise findByExerciseAndUser(Exercise exercise, MyUser
user){
        return userExerciseRepository.findByExerciseAndUser(exercise,user);
    }

    @Transactional
    public void deleteByExerciseAndUser(Exercise exercise, MyUser user){

userExerciseRepository.deleteUserExerciseByUserAndExercise(user,exercise);
    }
```

```
    @Transactional
    public void deleteById(Long userDishId, MyUser user, Exercise exercise){

userExerciseRepository.deleteByIdAndUserAndExercise(userDishId,user,exercise)
;
    }


    @Transactional
    public void deleteAllByExercise(Exercise exercise){
       userExerciseRepository.deleteAllByExercise(exercise);
    }


    public List<UserExercise> fitUserExercisesToMinutes(List<UserExercise>
userExercises){
        userExercises.forEach(userExercise -> {
            Exercise userExercise_ = userExercise.getExercise();


            double times_bigger = userExercise.getMinutes() * 1. *
userExercise.getExerciseMode().getCaloriesMultCoeff() /
userExercise_.getExerciseMode().getCaloriesMultCoeff();


            userExercise.setExercise(
                Exercise.builder()
                    .id(userExercise_.getId())
                    .name(userExercise_.getName())
                    .caloriesBurnedPerMinute((int)
(userExercise_.getCaloriesBurnedPerMinute() * times_bigger))
                    .build()
            );
```

```
        });
        return userExercises;
    }

    public Exercise getSummary(List<UserExercise> userExercises){
        int calories = 0;
        for (UserExercise userExercise : userExercises) {
            calories+=userExercise.getExercise().getCaloriesBurnedPerMinute();
        }
        return Exercise.builder()
                .caloriesBurnedPerMinute(calories)
                .build();
    };

    public Long biggest_id(){
        List<UserExercise> all = userExerciseRepository.findAll();

        if(all == null) return 0L;

        return all.stream().map(UserExercise::getId).mapToLong(x ->
x).max().orElse(0L);
    }

    public int getExercisePerformMinutes(UserInfo userInfo, List<Exercise>
exercises){
        int caloriesReduce = userInfoService.calcCaloriesReduce(userInfo);

        int exercisesBurnedCalPerMinute =
exercises.stream().mapToInt(Exercise::getCaloriesBurnedPerMinute).sum();
```

```
        int minutes = caloriesReduce/exercisesBurnedCalPerMinute;
        return minutes;
        /*List<UserExercisePlanDto> dtos = new ArrayList<>();
        exercises.forEach(exercise -> dtos.add(
            UserExercisePlanDto.builder()
                .exerciseMode(exercise.getExerciseMode())
                .caloriesPerMin(exercise.getCaloriesBurnedPerMinute())
                .minutes(minutes)

                .build()
        ));*/
    }

    public int getExercisePerformMinutesInMode(UserInfo userInfo,
List<Exercise> exercises, ExerciseMode mode){
        int caloriesReduce = userInfoService.calcCaloriesReduce(userInfo);

        int exercisesBurnedCalPerMinuteInMode = (int)
exercises.stream().mapToDouble(exercise->
            exercise.getCaloriesBurnedPerMinute() *
(mode.getCaloriesMultCoeff()/exercise.getExerciseMode().getCaloriesMultCoeff()
)
        ).sum();

        return caloriesReduce/exercisesBurnedCalPerMinuteInMode;
    }
}

@Service
```

```
public class UserDishService {
    private final UserDishRepository userDishRepository;

    @Autowired
    public UserDishService(UserDishRepository userDishRepository){
        this.userDishRepository = userDishRepository;
    }
    public void save(UserDish userDish){
        userDishRepository.save(userDish);
    }
    public List<UserDish> findAllByUser(MyUser user){
        return userDishRepository.findAllByUser(user);
    }

    public List<UserDish> findAllByUserAndDate(MyUser user, LocalDate
localDate){
        return userDishRepository.findAllByUserAndDate(user, localDate);
    }

    public List<UserDish> findAllByDish(Dish dish){
        return userDishRepository.findAllByDish(dish);
    }
    public UserDish findByDishAndUser(Dish dish, MyUser user){
        return userDishRepository.findByDishAndUser(dish,user);
    }

    @Transactional
    public void deleteByDishAndUser(Dish dish, MyUser user){
        userDishRepository.deleteUserDishByDishAndUser(dish,user);
```

```
    }

    @Transactional
    public void deleteById(Long id, MyUser user, Dish dish){
       userDishRepository.deleteByIdAndUserAndDish(id,user,dish);
    }

    @Transactional
    public void deleteAllByDish(Dish dish){
       userDishRepository.deleteAllByDish(dish);
    }

    public List<UserDish> fitUserDishesToPortions(List<UserDish>
userDishes){
       userDishes.forEach(userDish -> {
          Dish userDish_ = userDish.getDish();

          double times_bigger = userDish.getValue() * 1. *
userDish.getPortion().getGramsInPortion() /
userDish_.getPortion().getGramsInPortion();

          userDish.setDish(
             Dish.builder()
                .id(userDish_.getId())
                .name(userDish_.getName())
                .calories((int) (userDish_.getCalories() * times_bigger))
                .carbs((int)(userDish_.getCarbs() * times_bigger))
                .fat((int)(userDish_.getFat() * times_bigger))
                .protein((int)(userDish_.getProtein()*times_bigger))
```

```
                        .build()
                );
            });
            return userDishes;
        }

        public Dish getSummary(List<UserDish> userDishes){
            int calories = 0;
            int carbs = 0;
            int fats = 0;
            int protein = 0;
            for (UserDish userDish : userDishes) {
                calories+=userDish.getDish().getCalories();
                carbs+=userDish.getDish().getCarbs();
                fats+=userDish.getDish().getFat();
                protein+=userDish.getDish().getProtein();
            }

            return Dish.builder()
                    .calories(calories)
                    .carbs(carbs)
                    .fat(fats)
                    .protein(protein)
                .build();
        };

        public Map<String, Integer>
calcCaloriesForEachMealtime(List<UserDish> userDishes) {
            Map<String, Integer> map = new LinkedHashMap<>();
```

```
        for (Mealtime mealtime : Mealtime.values()) {
            map.put(mealtime.name(),0);
        }

        userDishes.forEach(userDish -> {
            map.computeIfPresent(userDish.getMealtime().name(),
                    (k,v)->v+userDish.getDish().getCalories());
        });

        return map;
    }
}

@Service
public class ExerciseService {
    private final ExerciseRepository exerciseRepository;

    @Autowired
    public ExerciseService(ExerciseRepository exerciseRepository) {
        this.exerciseRepository = exerciseRepository;
    }

    public void saveNewExercise(Exercise exercise) throws Exception{
        exerciseRepository.save(exercise);
    }

    public void deleteExerciseById(Long id){
        exerciseRepository.deleteById(id);
```

```
    }

    public Exercise findById(Long id){
        return exerciseRepository.findById(id).get();
    }

    public List<Exercise> findAllByUser(MyUser user){
        return exerciseRepository.findAllByUser(user);
    }

    public List<Exercise> findAllByUserIsNull(){
        return exerciseRepository.findAllByUserIsNull();
    }

    public boolean existsExerciseByNameAndUserIsNull(String name){
        return exerciseRepository.existsExerciseByNameAndUserIsNull(name);
    }

}

@Slf4j
@Service
public class DishesService {
    private final DishesRepository dishesRepository;

    @Autowired
    public DishesService(DishesRepository dishesRepository) {
        this.dishesRepository = dishesRepository;
```

```
}

public DishesDTO getAllDishes() {
    //TODO checking for an empty user list

    return new DishesDTO(dishesRepository.findAll());
}

public void saveNewDish (Dish dish) throws Exception{

    dishesRepository.save(dish);
}

public void deleteDishById(Long id) {
    dishesRepository.deleteById(id);
}
public Dish findById(Long id){
    return dishesRepository.findById(id).get();
}

public List<Dish> findAllByUser(MyUser user){
    return dishesRepository.findAllByUser(user);
}

public List<Dish> findAllWhereUserIsNull(){
    return dishesRepository.findAllByUserIsNull();
```

```
    }

    public boolean existsDishByNameAndUserIsNull(String name){
        return dishesRepository.existsDishByNameAndUserIsNull(name);
    }
}
```